

AWS Data Analytics Products

AWS Data Analytics Products

- AWS offers services to handle analytics process chain including data warehousing, business intelligence, batch processing, stream processing, machine learning, and data workflow orchestration.

Service	Product Type
Amazon Athena	Serverless Query Service
Amazon EMR	Hadoop
Amazon Elasticsearch Service	Elasticsearch
Amazon Kinesis	Streaming Data
Amazon QuickSight	Business Analytics
Amazon Redshift	Data Warehouse
AWS Glue	ETL
AWS Data Pipeline	Data Workflow Orchestration

AWS Athena

Athena

- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL (HIVE SQL)
- Athena is serverless, so there is no infrastructure to setup or manage, and customers pay only for the queries they run.
- You can use Athena to process logs, perform ad-hoc analysis, and run interactive queries.
- Athena scales automatically – executing queries in parallel – so results are fast, even with large datasets and complex queries.
- Athena can generate tables directly from data available in S3 in different formats (CSV, JSON, TSV, amazon logs, and others)
- Athena uses prestodb, a distributed SQL query engine developed by Facebook that offers a very rich library of SQL functions, which are well suited to do data transformations and feature engineering.
- See the video
 - https://www.youtube.com/watch?v=DxAuj_Ky5aw

Key Features



Serverless. Zero Infrastructure. Zero Administration

Amazon Athena is serverless, so there is no infrastructure to manage. You don't need to worry about configuration, software updates, failures or scaling your infrastructure as your datasets and number of users grow. Athena automatically takes care of all of this for you, so you can focus on the data, not the infrastructure.



Easy To Get Started

To get started, log into the Athena console, define your schema using the console wizard or by entering DDL statements, and immediately start querying using the built-in query editor. Results are displayed in the console within seconds, and automatically written to a location of your choice in S3. You can also download them to your desktop. With Athena, there's no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.



Easy To Query, Just Use Standard SQL

Amazon Athena uses Presto, an open source, distributed SQL query engine optimized for low latency, ad hoc analysis of data. This means you can run queries against large datasets in Amazon S3 using ANSI SQL, with full support for large joins, window functions, and arrays. Athena supports a wide variety of data formats such as CSV, JSON, ORC, Avro, or Parquet. You can also connect to Athena from a wide variety of BI tools using Athena's JDBC driver.



Pay Per Query

With Amazon Athena, you pay only for the queries that you run. You are charged based on the amount of data scanned by each query. You can get significant cost savings and performance gains by compressing, partitioning, or converting your data to a columnar format, because each of those operations reduces the amount of data that Athena needs to scan to execute a query.



Fast Performance

With Amazon Athena, you don't have to worry about managing or tuning clusters to get fast performance. Athena is optimized for fast performance with Amazon S3. Athena automatically executes queries in parallel, so that you get query results in seconds, even on large datasets.



Highly Available & Durable

Amazon Athena is highly available and executes queries using compute resources across multiple facilities, automatically routing queries appropriately if a particular facility is unreachable. Athena uses Amazon S3 as its underlying data store, making your data highly available and durable. Amazon S3 provides durable infrastructure to store important data and is designed for durability of 99.999999999% of objects. Your data is redundantly stored across multiple facilities and multiple devices in each facility.



Secure

Amazon Athena allows you to control access to your data by using AWS Identity and Access Management (IAM) policies, access control lists (ACLs), and Amazon S3 bucket policies. With IAM policies, you can grant IAM users fine-grained control to your S3 buckets. By controlling access to data in S3, you can restrict users from querying it using Athena. Athena also allows you to easily query encrypted data stored in Amazon S3 and write encrypted results back to your S3 bucket. Both, server-side encryption and client-side encryption are supported.

A Brief Tour of Athena

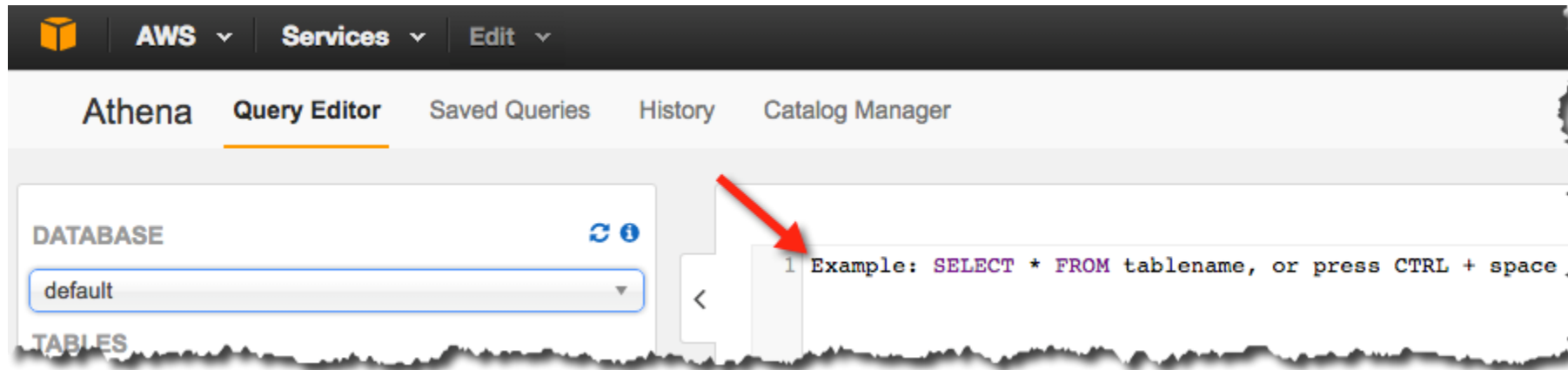
- In Athena, the data is not stored in a database; it remains in S3. When you create a table in Athena, you are creating an information layer that tells Athena where to find the data, how it is structured, and what format it is in. The schema in Athena is a logical namespace of objects. Once the data structure and location are known to Athena, you can query the data via standard SQL statements.
- Athena uses Hive Data Definition Language (DDL) to create or drop databases and tables.
- Athena can understand multiple formats (CSV, TSV, JSON, and so on) through the use of serializer-deserializer (SerDes) libraries.
- Athena is available either via the console (<https://console.aws.amazon.com/athena/>) or via JDBC connection (<http://docs.aws.amazon.com/athena/latest/ug/connect-with-jdbc.html>). We will use the console.

Getting start with Athena

- **Step 1: Create a Database**
- **Step 2: Create a Table**
- **Step 3: Query Data**
- **<http://docs.aws.amazon.com/athena/latest/ug/getting-started.html>**

Create a database

- 1) Open the [AWS Management Console for Athena](#).
- 2) If this is your first time visiting the AWS Management Console for Athena, you'll go to a Getting Started page. Choose **Get Started** to open the Query Editor. If it isn't your first time, the Athena Query Editor opens.
- 3) In the Athena Query Editor, you see a query pane with an example query. Start typing your query anywhere in the query pane.

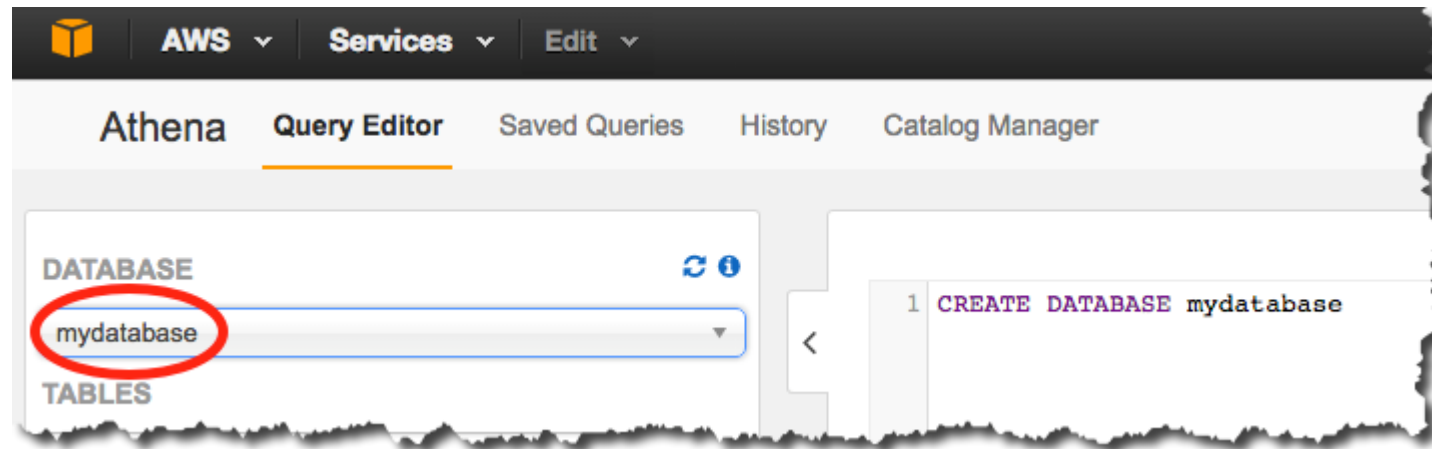


Create a database

4) To create a database named mydatabase, enter the following CREATE DATABASE statement, and then choose Run Query:

```
CREATE DATABASE mydatabase
```

5) Confirm that the catalog display refreshes and mydatabase appears in the DATABASE list in the Catalog dashboard on the left side.



Create a table

Now that you have a database, you're ready to create a table that's based on the sample data file. You define columns that map to the data, specify how the data is delimited, and provide the location in Amazon S3 for the file.

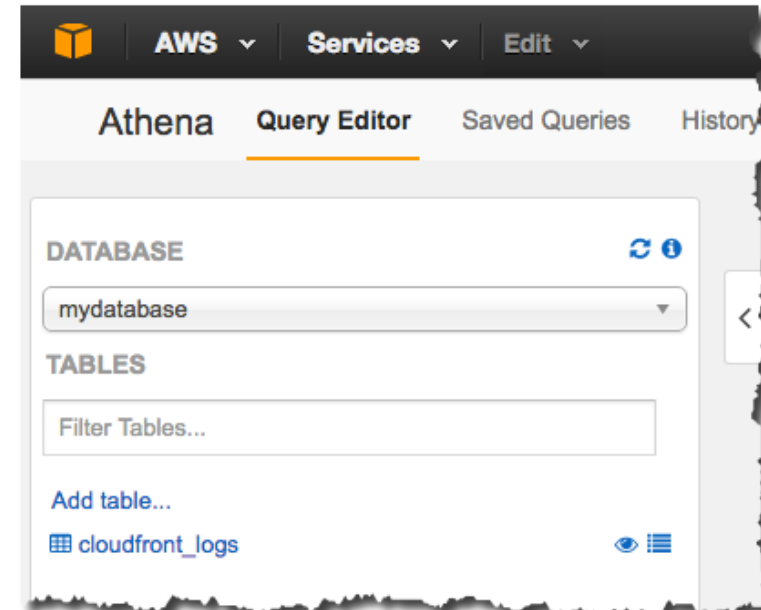
1. Make sure that `mydatabase` is selected for **DATABASE** and then choose **New Query**.
2. In the query pane, enter the following **CREATE TABLE** statement, and then choose **Run Query**:

Note

You can query data in regions other than the region where you run Athena. Standard inter-region data transfer rates for Amazon S3 apply in addition to standard Athena charges. To reduce data transfer charges, replace *myregion* in `s3://athena-examples-myregion/path/to/data/` with the region identifier where you run Athena, for example, `s3://athena-examples-us-east-1/path/to/data/`.

[illegible]

The table is created and appears in the **Catalog** dashboard for your database.



Query data

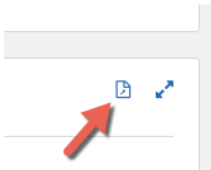
1. Choose **New Query**, enter the following statement anywhere in the query pane, and then choose **Run Query**:

```
SELECT os, COUNT(*) count FROM cloudfront_logs WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
```

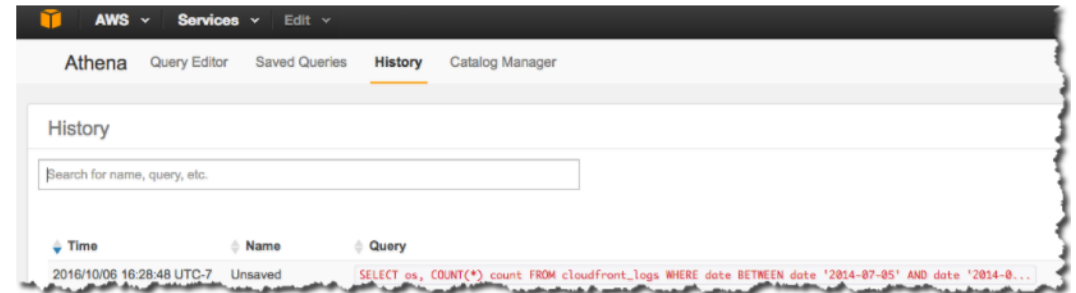
Results are returned that look like the following:

Results		
	os	count
1	iOS	794
2	MacOS	852
3	OSX	799
4	Windows	883
5	Linux	813
6	Android	855

2. Optionally, you can save the results of a query to CSV by choosing the file icon on the **Results** pane.



3. You can also view the results of previous queries or queries that may take some time to complete. Choose **History** then either search for your query or choose **View** or **Download** to view or download the results of previous completed queries. This also displays the status of queries that are currently running.



Query results are also stored in Amazon S3 in a bucket called `aws-athena-query-results-ACCOUNTID-REGION`. You can change the default location in the console and encryption options by choosing **Settings** in the upper right pane. For more information, see [Queries](#) and [Query Result Files](#).

AWS Kinesis

AWS KINESIS

- A real-time data processing service that continuously captures (and stores) large amount of data that power real-time streaming dash boards
- Using AWS provided SDKs, you can create real-time dashboards, integrate dynamic pricing strategies, and export data from Kinesis to other AWS services including
 - S3
 - EMR
 - Redshift
 - Lambda
- Kinesis components
 - Stream
 - Producers (data creators)
 - Consumers (data consumers)
 - Shards (processing power)

Benefits

- Real-time processing
 - Continuously collect and build applications that analyze the data as it is generated
- Parallel Processing
 - Multiple Kinesis applications can be processing the same incoming data streaming concurrently
- Durable
 - Kinesis synchronously replicates the streaming data across three data centers within a single AWS region and preserves the data for up to 24 hours
- Scales
 - Can stream from as little a few megabytes to several terabytes per hour

When to Use Kinesis

- Gaming
 - Collect gaming data such as player actions and feed the data into the gaming platform, for example a reactive environment based off of real-time actions of the player
- Real-time analytics
 - Collect IOT (sensors) from many sources and high amounts of frequency and process it using Kinesis to gain insights as data arrives in your environment
- Application alerts
 - Build a Kinesis application that monitors incoming application logs in real-time and trigger events based off the data
- Log / Event Data collection
 - Log data from any number of devices and use Kinesis application to continuously process the incoming data, power real-time dashboards and store the data in S3 when completed
- Mobile data capture
 - Mobile applications can push data to Kinesis from countless numbers of devices which makes the data available as soon as it is produced

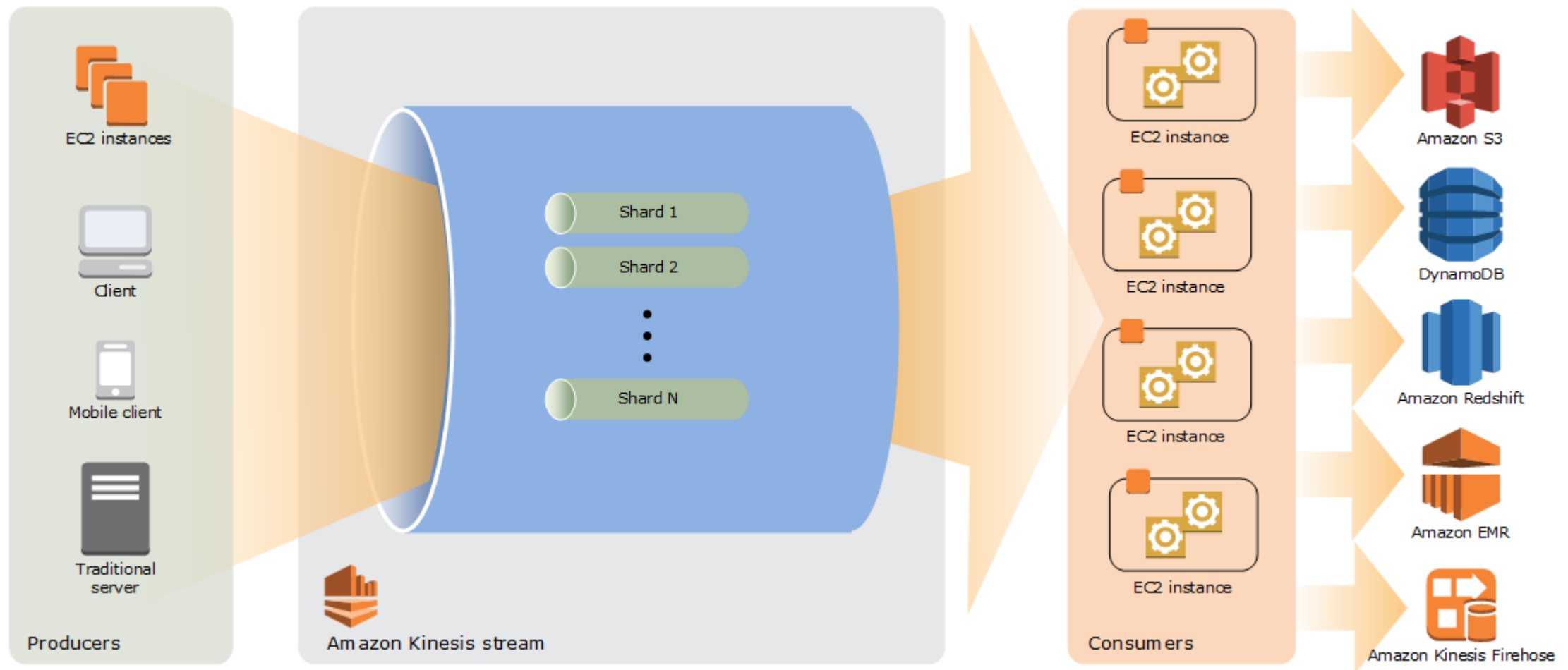
Kinesis Producers

- Devices that collect data for Kinesis processing
- You build producers to continuously input data into a Kinesis stream
- Producers can include (but not limited to)
 - IoT Sensors
 - Mobile devices (cell phones)
- You can have literally thousands of different producers and scale based on need
 - The more data you want to process more “shards” you add to your Kinesis stream
 - Each “shard” can process 2MB of read data per second, and 1MB of write data per second

Kinesis Consumers

- Consume the stream's data
- This is done concurrently (multiple consumers can consume the same data at the same time).
- Consumers include (but are not limited to)
 - Real-time dashboards
 - S3
 - Redshift
 - EMR
- Any application (one you create) can consume the streams data
- Kinesis keeps 24 hours of streaming data stored by default, but can be configured to store up to 7 days

Kinesis Streams High-Level Architecture



AWS EMR

AWS EMR

- Amazon EMR provides a hosted Hadoop, Pig, Hive, and HBase services for developers and businesses to help them build Big Data applications without worrying about the deployment complexity or managing Hadoop clusters with underlying infrastructure



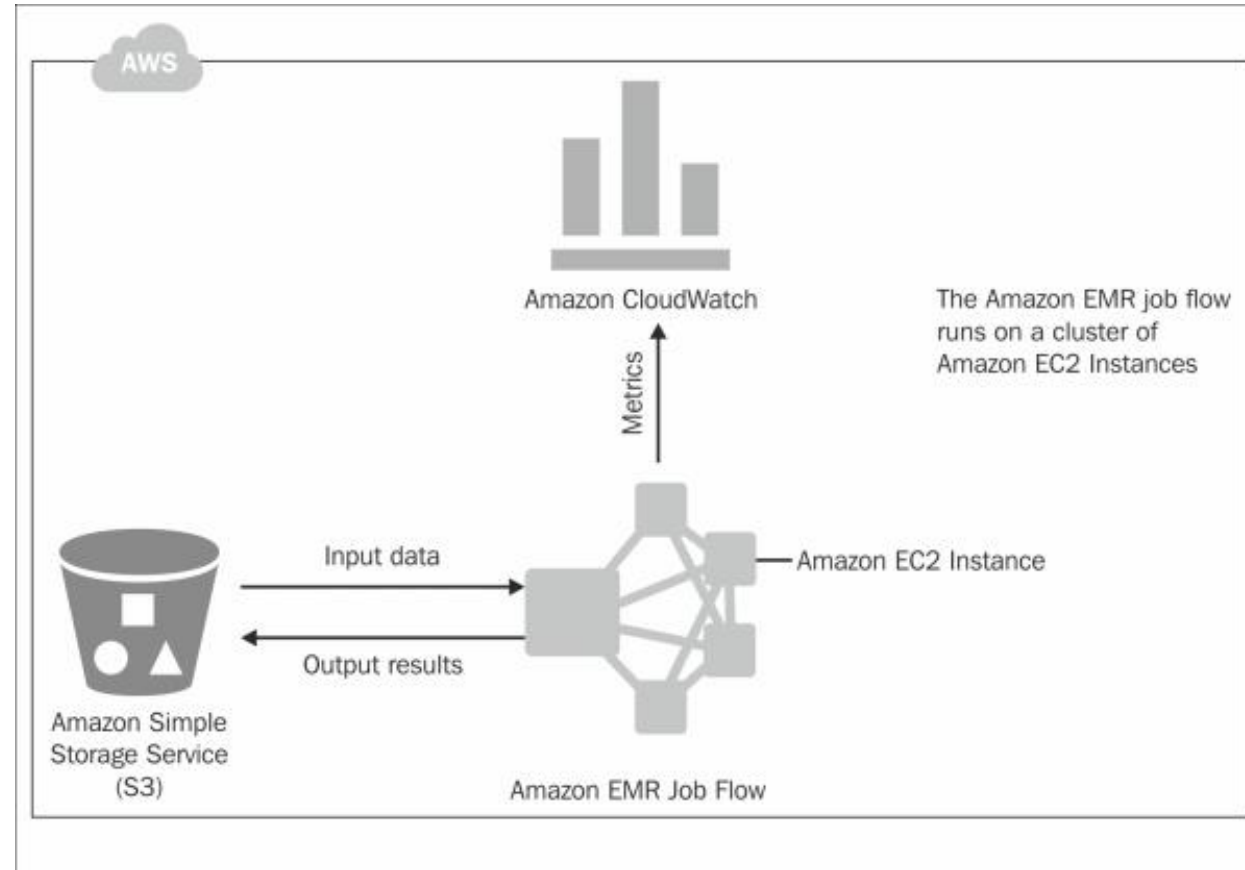
EMR Key Features

- **Ease of use:** EMR provides a hosted Hadoop service without worrying about deployment complexity or configuration challenges. You can use multiple Hadoop distributions and third-party libraries with EMR. We can easily integrate EMR with other AWS services such as S3, DynamoDB, Redshift, CloudWatch, and many more.
- **Elasticity:** EMR allows you to scale up and scale down the nodes in a Hadoop cluster without worrying about underlying infrastructure. You can easily launch a cluster with a few nodes or thousands of nodes in a matter of minutes.
- **Scalable storage:** You can use AWS S3 as a filesystem with EMR, which is scalable to petabytes of storage. We don't need to worry about managing the filesystem for our applications or underlying failures of disks. You can be rest assured about not losing your data if you use S3 as the final output location of your Hadoop tasks.
- **Cost effective:** EMR allows us to leverage their spot pricing for EMR to build large-scale clusters at a lower cost. With EMR, we can quickly launch a five-node cluster to analyze data for a few hours at the cost of less than \$1. It also gives you the ability to start with a small number of nodes and increase them on the fly when the need arises. For example, you might want to speed up the process or when after completion of a small task, you want to process another task using the same cluster that needs more nodes to finish within a stipulated time. This effectively reduces your cost.
- **Configurable:** EMR provides a custom bootstrapping capability to override default configurations or packages. You have complete control over the cluster and can get root access to underlying instances in the EMR cluster to make any required changes. It's also easy to use various third-party tools with EMR as AWS provides configuration and customization options. You can also use monitoring tools such as Ganglia along with your EMR cluster.
- **Programmable:** This is one feature of AWS that makes it easy to build custom application layers on top of its services. EMR provides API and client SDK to programmatically create and manage clusters. You can programmatically increase or decrease the existing cluster to leverage AWS spot pricing to reduce your costs. You can monitor your running jobs and raise alerts such as e-mails by writing an application using APIs and SDKs provided by AWS.

AWS EMR Use Cases

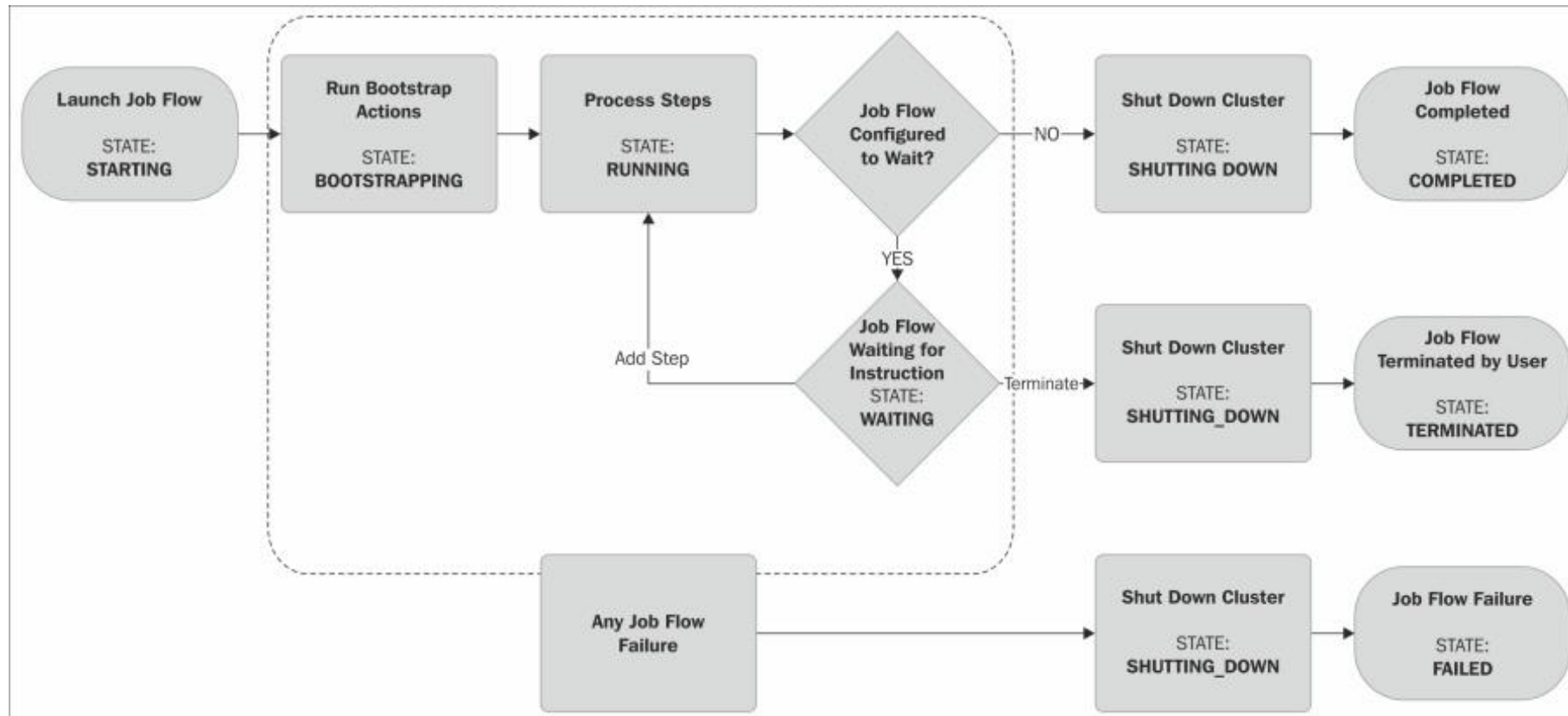
- **Web log processing**
 - You can use EMR to process logs to understand the usage of content such as video, file downloads, top web URLs accessed by end users, user consumption from different parts of the world, and many more. We can process any web or mobile application logs using EMR to understand specific business insights relevant for your business. We can move all our web access application or mobile logs to Amazon S3 for analysis using EMR even if we are not using AWS for running our production applications.
- **Clickstream analysis**
 - By using clickstream analysis, we can segment users into different groups and understand their behaviors with respect to advertisements or application usage. Ad networks or advertisers can perform clickstream analysis on ad-impression logs to deliver more effective campaigns or advertisements to end users. Reports generated from this analysis can include various metrics such as source traffic distribution, purchase funnel, lead source ROI, and abandoned carts among others. You can also use EMR to move data between different systems in AWS such as DynamoDB, Redshift, S3, and many more.
- **Product recommendation engine**
 - Recommendation engines can be built using EMR for e-commerce, retail, or web businesses. With recommendation engines, You can help end users to quickly find relevant products or suggest products based on what they are viewing and so on. You can also want to notify users via an e-mail based on their past purchase behavior.
- **Scientific simulations**
 - When you need distributed processing with large-scale infrastructure for scientific or research simulations, EMR can be of great help. We can quickly launch large clusters in a matter of minutes and install specific MapReduce programs for analysis using EMR. AWS also offers genomics datasets for free on S3.
- **Data transformations**
 - We can perform complex **extract, transform, and load(ETL)** processes using EMR for either data analysis or data warehousing needs. It can be as simple as transforming XML file data into JSON data for further usage or moving all financial transaction records of a bank into a common date-time format for archiving purposes

AWS EMR Job Flow



Source: Amarkant Singh & Vijay Rayapati, 2014

EMR Cluster Life Cycle



Source: Amarkant Singh & Vijay Rayapati, 2014

EMR Nodes Types

- Master node — Manages the cluster: coordinating the distribution of the MapReduce executable and subsets of the raw data, to the core and task instance groups. It also tracks the status of each task performed, and monitors the health of the instance groups. There is only one master node in a cluster. This maps to the Hadoop master node.
- Core nodes — Runs tasks and stores data using the Hadoop Distributed File System (HDFS). This maps to a Hadoop slave node.
- Task nodes (optional) — Run tasks. This maps to a Hadoop slave node.

EMR Process Steps

- A cluster contains one or more steps.
- Steps are processed in the order in which they are listed in the cluster.
- Steps are run following this sequence:
 - all steps have their state set to PENDING.
 - The first step is run and the step's state is set to RUNNING.
 - When the step is completed, the step's state changes to COMPLETED.
 - The next step in the queue is run, and the step's state is set to RUNNING.
 - After each step completes, the step's state is set to COMPLETED and the next step in the queue is run.
 - Steps are run until there are no more steps. Processing flow returns to the cluster.
 - If a step fails, the step state is FAILED and all remaining steps with a PENDING state are marked as CANCELLED. No further steps are run and processing returns to the cluster.



How to Use AWS EMR

- Develop your data processing application. You can use Java, Hive (a SQL-like language), Pig (a data processing language), Cascading, Ruby, Perl, Python, R, PHP, C++, or Node.js. Amazon EMR provides [code samples and tutorials](#) to get you up and running quickly.
- Upload your application and data to Amazon S3.
- Configure and launch your cluster. Using the [AWS Management Console](#), the [AWS CLI](#), [SDKs](#), or [APIs](#), specify the number of Amazon EC2 instances to provision in your cluster, the types of instances to use (standard, high memory, high CPU, high I/O, etc.), the applications to install (Hive, Pig, HBase, etc.), and the location of your application and data. You can use [Bootstrap Actions](#) to install additional software or change default settings.
- Monitor the cluster (*Optional*). You can monitor the cluster's health and progress using the Management Console, Command Line Interface, SDKs, or APIs. EMR integrates with [Amazon CloudWatch](#) for monitoring/alarming and supports popular monitoring tools like [Ganglia](#). You can [add/remove capacity](#) to the cluster at any time to handle more or less data. For troubleshooting, you can use the console's simple [debugging GUI](#).
- Retrieve the output. Retrieve the output from Amazon S3 or HDFS on the cluster. Visualize the data with [tools](#) like Tableau and MicroStrategy. Amazon EMR will automatically terminate the cluster when processing is complete. Alternatively you can leave the cluster running and give it more work to do.



amazon
web services

Training and
Certification

Introduction to Amazon Elastic MapReduce (EMR)



0:07 / 10:04



Walkthrough: EMR Group

- Make sure you have a group with the right permissions

[IAM](#) > [Groups](#) > [emr](#)

▼ Summary

Group ARN: `arn:aws:iam::780843837262:group/emr`
Users (in this group): 1
Path: /
Creation Time: 2017-07-17 13:48 EDT

Users





Permissions

Access Advisor

Managed Policies

The following managed policies are attached to this group. You can attach up to 10 managed policies.

[Attach Policy](#)

Policy Name	Actions
 AmazonElasticMapReduceRole	Show Policy Detach Policy Simulate Policy
 AmazonS3FullAccess	Show Policy Detach Policy Simulate Policy
 AmazonElasticMapReduceforEC2Role	Show Policy Detach Policy Simulate Policy
 AmazonElasticMapReduceFullAccess	Show Policy Detach Policy Simulate Policy

Walkthrough: EMR User

- Make sure you have a user with the right permissions

Summary



User ARN `arn:aws:iam::780843837262:user/emr`

Path `/`

Creation time 2017-07-17 13:50 EDT

Permissions

Groups (2)

Security credentials

Access Advisor

Add permissions

Attached policies: 5


Policy name ▾	Policy type ▾	
Attached from group		
▶  AmazonElasticMapReduceRole	AWS managed policy from group emr	✕
▶  AmazonS3FullAccess	AWS managed policy from group emr	✕
▶  AmazonElasticMapReduceforEC2Role	AWS managed policy from group emr	✕
▶  AmazonElasticMapReduceFullAccess	AWS managed policy from group emr	✕
▶  AmazonEC2FullAccess	AWS managed policy from group ec2developer	✕

Walkthrough: Create cluster

General Configuration

Cluster name

☒ Logging ⓘ

S3 folder 

Launch mode ☒ Cluster ⓘ ☐ Step execution ⓘ

Software configuration

Release ⓘ

Applications ☒ Core Hadoop: Hadoop 2.7.3 with Ganglia 3.7.2, Hive 2.1.1, Hue 3.12.0, Mahout 0.13.0, Pig 0.16.0, and Tez 0.8.4

☐ HBase: HBase 1.3.1 with Ganglia 3.7.2, Hadoop 2.7.3, Hive 2.1.1, Hue 3.12.0, Phoenix 4.11.0, and ZooKeeper 3.4.10

☐ Presto: Presto 0.170 with Hadoop 2.7.3 HDFS and Hive 2.1.1 Metastore

☐ Spark: Spark 2.1.1 on Hadoop 2.7.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.2

Hardware configuration

Instance type

Number of instances (1 master and 2 core nodes)

Security and access

EC2 key pair ⓘ [Learn how to create an EC2 key pair.](#)

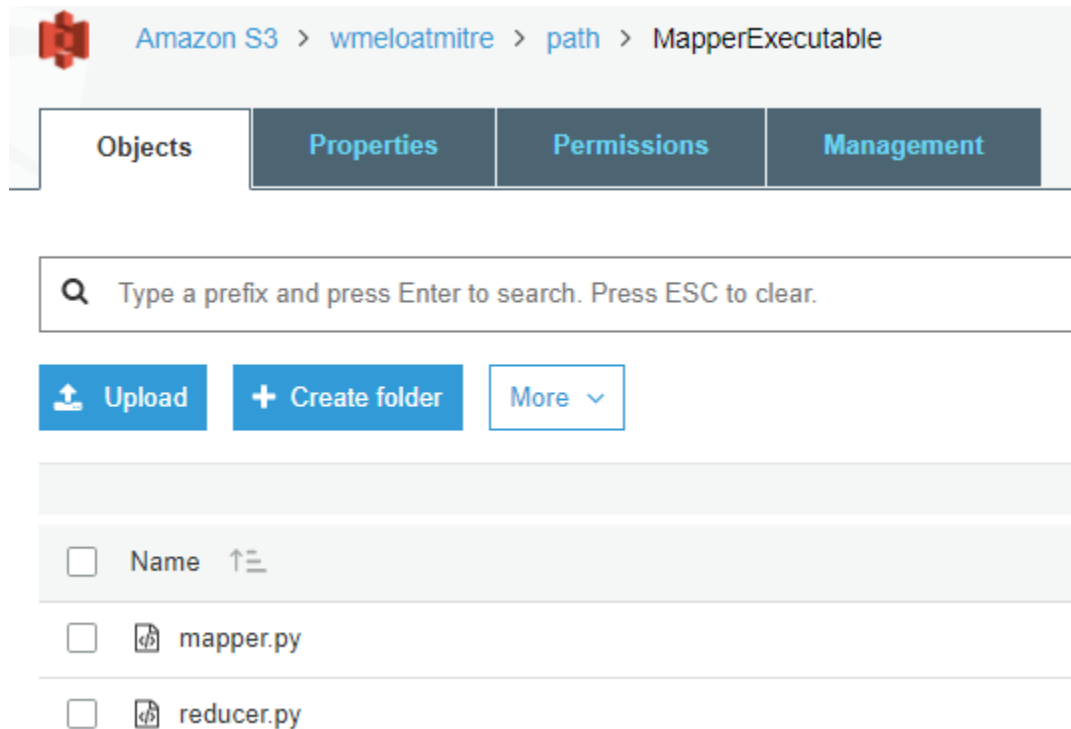
Permissions ☒ Default ☐ Custom

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](#) ⓘ

Walkthrough: Upload mapper/reducer/inputs into AWS S3 buckets





The screenshot shows the AWS S3 console interface for a bucket named 'wmeloatmitre'. The breadcrumb navigation path is 'Amazon S3 > wmeloatmitre > path > MapperExecutable'. The 'Objects' tab is selected, displaying a list of objects. The search bar contains the text 'Type a prefix and press Enter to search. Press ESC to clear.' The action buttons are 'Upload', 'Create folder', and 'More'. The object list has a header row with a checkbox and the text 'Name ↑'. Below the header, two objects are listed: 'mapper.py' and 'reducer.py', each with a checkbox and a document icon.

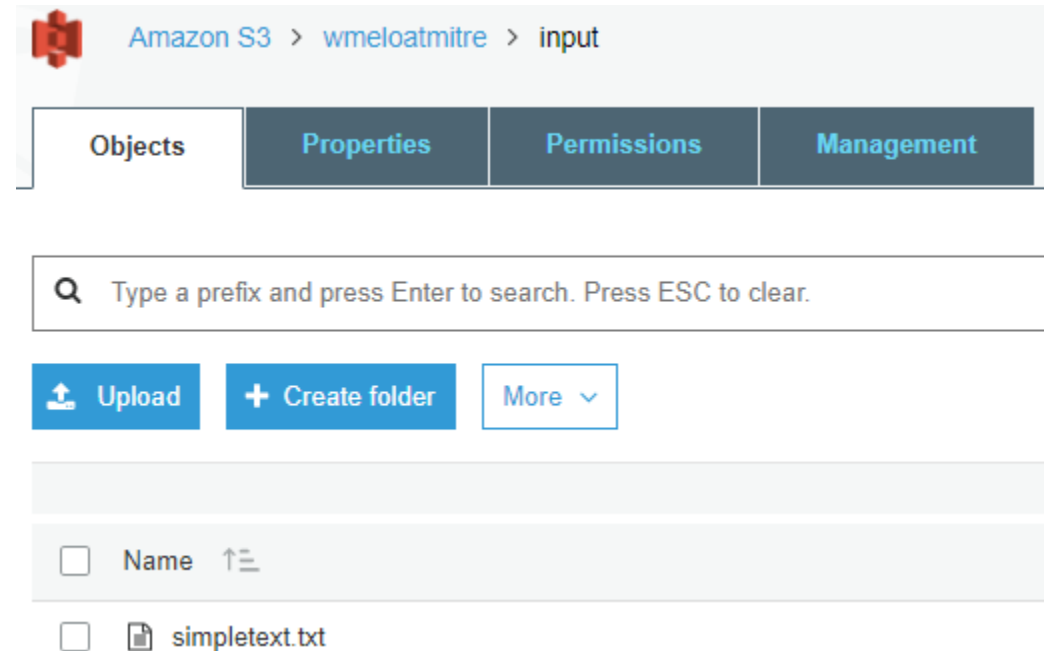
Amazon S3 > wmeloatmitre > path > MapperExecutable

Objects Properties Permissions Management

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

Upload Create folder More

<input type="checkbox"/>	Name ↑
<input type="checkbox"/>	 mapper.py
<input type="checkbox"/>	 reducer.py




The screenshot shows the AWS S3 console interface for the same bucket 'wmeloatmitre'. The breadcrumb navigation path is 'Amazon S3 > wmeloatmitre > input'. The 'Objects' tab is selected, displaying a list of objects. The search bar contains the text 'Type a prefix and press Enter to search. Press ESC to clear.' The action buttons are 'Upload', 'Create folder', and 'More'. The object list has a header row with a checkbox and the text 'Name ↑'. Below the header, one object is listed: 'simpletext.txt', with a checkbox and a document icon.

Amazon S3 > wmeloatmitre > input

Objects Properties Permissions Management

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

Upload Create folder More

<input type="checkbox"/>	Name ↑
<input type="checkbox"/>	 simpletext.txt

Walkthrough: Cluster create / add steps


Cluster: MyEmrClust **Waiting** Cluster ready after last step completed.

Connections: [Enable Web Connection](#) – Hue, Ganglia, Resource Manager ... (View All)
Master public DNS: ec2-52-14-144-245.us-east-2.compute.amazonaws.com [SSH](#)
Tags: -- [View All](#) / [Edit](#)

Summary

ID: j-3LO5GCAZ8FV88
Creation date: 2017-07-17 16:12 (UTC-4)
Elapsed time: 50 minutes
Auto-terminate: No
Termination protection: Off [Change](#)

Configuration Details

Release label: emr-5.7.0
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://aws-logs-780843837262-us-east-2/elasticmapreduce/ 
EMRFS consistent view: Disabled

Network and Hardware

Availability zone: us-east-2c
Subnet ID: [subnet-37a2707a](#)
Master: **Running** 1 m4.large
Core: **Running** 2 m4.large
Task: --

Security and Access

Key name: aws_emr_key_pair
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All [Change](#)
Security groups for [sg-2d8a9344](#) (ElasticMapReduce-**Master:** master)
Security groups for [sg-3f978e56](#) (ElasticMapReduce-**Core & Task:** slave)

► Monitoring

► Hardware

► Steps

► Configurations

► Events

► Bootstrap Actions

Walkthrough: Add Execution Step

Add Step

Step type

Streaming program

Name*

Streaming program

Mapper*

s3://wmeloatmitre/path/MappperExecutable/mapper.py

S3 location of the map function or the name of the Hadoop streaming command to run.

Reducer*

s3://wmeloatmitre/path/MappperExecutable/reducer.py

S3 location of the reduce function or the name of the Hadoop streaming command to run.

Input S3 location*

s3://wmeloatmitre/input/simpletext.txt

s3://<bucket-name>/<folder>/

Output S3 location*

s3://wmeloatmitre/notexistingoutputfolder

s3://<bucket-name>/<folder>/

Arguments

Action on failure

Continue

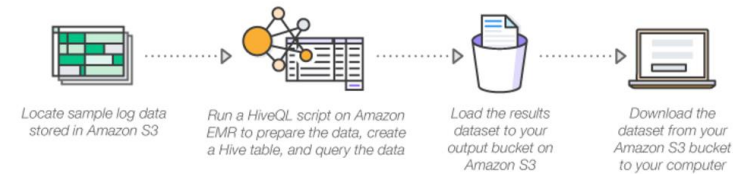
What to do if the step fails.

Cancel

Add

HIVE

- After creating your EMR Cluster, create a new process step
- For Step type, choose Hive program.
- For Name, accept the default name (Hive program) or type a new name.
- For Script S3 location, type
- `s3://us-east-1.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q`
- For Input S3 location, type `s3:// us-east-1.elasticmapreduce.samples`
- For Output S3 location, type or browse to the output bucket that you created in Create an Amazon S3 Bucket.
- For Arguments, include the following argument to allow column names that are the same as reserved words:
 - `-hiveconf hive.support.sql11.reserved.keywords=false`



**Don't forget to terminate your EMR cluster and Amazon S3 bucket after you complete the project to avoid additional charges*

Make Sure Cluster is Ready

Create cluster

View details

Clone

Terminate

Filter:

All clusters

Filter clusters ...

3 clusters (all loaded)

	Name	ID	Status
<input type="checkbox"/>	▶ ● MyEmrClust	j-5FFZ7G5NN075	Waiting Cluster ready

Input Data

- s3://us-east-1.elasticmapreduce.samples
- 2014-07-05 20:00:00 LHR3 4260 10.0.0.15 GET
eabcd12345678.cloudfront.net /test-image-1.jpeg 200 -
Mozilla/5.0%20(MacOS;%20U;%20Windows%20NT%205.1;%20en-
US;%20rv:1.9.0.9)%20Gecko/2009040821%20IE/3.0.9

Hive Script

[illegible]

Create Step

Add Step

Step type

Hive program

Name

Hive program

Script S3 location*

s3://us-east-1.elasticmapreduce.samples/cloudfront/cc
s3://<bucket-name>/<path-to-file>

S3 location of your Hive script.

Input S3 location

s3://us-east-1.elasticmapreduce.samples
s3://<bucket-name>/<folder>/

S3 location of your Hive input files.

Output S3 location

s3://wmeloatmitre/output
s3://<bucket-name>/<folder>/

S3 location of your Hive output files.

Arguments

```
-hiveconf  
hive.support.sql11.reserved.keywords  
=false
```

Specify optional arguments for your script.

Action on failure

Continue

What to do if the step fails.

Cancel

Add

Verify Results

Cluster: MyEmrClust **Waiting** Cluster ready after last step completed.

Connections: [Enable Web Connection](#) – Hue, Ganglia, Resource Manager ... (View All)
Master public DNS: ec2-13-59-226-45.us-east-2.compute.amazonaws.com [SSH](#)
Tags: -- [View All](#) / [Edit](#)

Summary
ID: j-5FFZ7G5NN075
Creation date: 2017-07-18 12:33 (UTC-4)
Elapsed time: 24 minutes
Auto-terminate: No
Termination protection: Off [Change](#)

Configuration Details
Release label: emr-5.7.0
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://aws-logs-780843837262-us-east-2/elasticmapreduce/
EMRFS consistent view: Disabled

Network and Hardware
Availability zone: us-east-2b
Subnet ID: [subnet-b6850acd](#)
Master: **Running** 1 m4.large
Core: **Running** 2 m4.large
Task: --

Security and Access
Key name: aws_emr_key_pair
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All [Change](#)
Security groups for [sg-2d8a9344](#) (ElasticMapReduce-master) Master:
Security groups for [sg-3f978e56](#) (ElasticMapReduce-slave) Core & Task:

► Monitoring

► Hardware

▼ Steps

[Add step](#) [Clone step](#) [Cancel step](#)

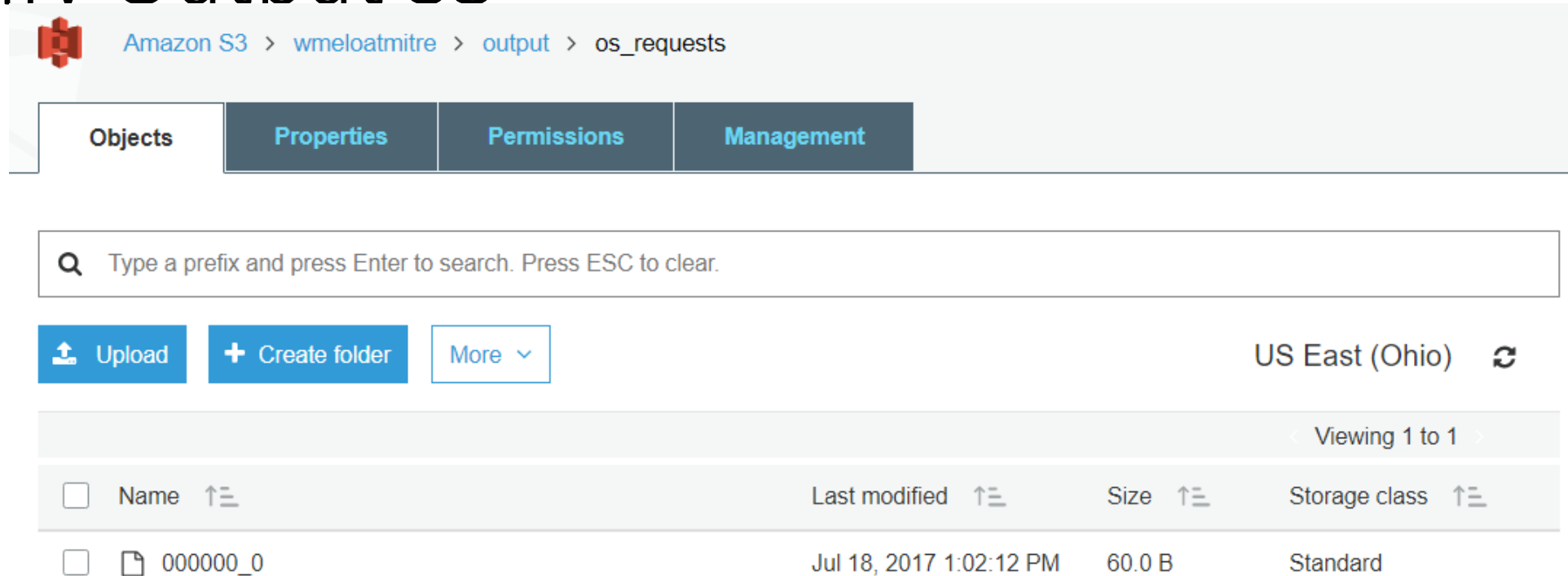
Steps [View all interactive jobs](#) | [View all jobs](#)

Filter: All steps 2 steps (all loaded)

	ID	Name	Status	Start time (UTC-4) ▼	Elapsed time	Log files	Actions
<input type="radio"/>	s-20K16CGX7T209	Hive program	Completed	2017-07-18 13:01 (UTC-4)	1 minute	View logs	View jobs

JAR location : command-runner.jar
Main class : None
hive-script --run-hive-script --args -f s3://us-east-1.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q -d INPUT=s3://us-east-1.elasticmapreduce.samples -d OUTPUT=s3://wmeloatmitre/output -hiveconf
Arguments : hive.support.sql11.reserved.keywords=false
Action on failure : Continue

Verify Output S3



The screenshot shows the Amazon S3 console interface. At the top, the breadcrumb navigation reads: Amazon S3 > wmeloatmitre > output > os_requests. Below this is a tabbed interface with four tabs: Objects, Properties, Permissions, and Management. The 'Objects' tab is selected. A search bar is present with the placeholder text: 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are three buttons: 'Upload', 'Create folder', and 'More'. To the right of these buttons, the region 'US East (Ohio)' is displayed with a refresh icon. Below the buttons is a table showing the objects in the bucket. The table has four columns: Name, Last modified, Size, and Storage class. There is one object listed: '000000_0', which was last modified on 'Jul 18, 2017 1:02:12 PM', has a size of '60.0 B', and is stored in the 'Standard' storage class.

Amazon S3 > wmeloatmitre > output > os_requests

Objects Properties Permissions Management

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

Upload Create folder More

US East (Ohio)

Viewing 1 to 1

Name	Last modified	Size	Storage class
000000_0	Jul 18, 2017 1:02:12 PM	60.0 B	Standard

Android855
Linux813
MacOS852
OSX799
Windows883
iOS794

Accessing Master Node from the Internet

- Change security group to accept ssh from the internet

[Create Security Group](#) [Actions](#)

search : sg-53ce8522 [Add filter](#)

1 to 2 of 2

<input type="checkbox"/>	Name	Group ID	Group Name	VPC ID	Description
<input checked="" type="checkbox"/>		sg-53ce8522	ElasticMapReduce-master	vpc-c5871cbc	Master group for Elastic MapReduce created on 2017-07-17T16:05:38.663Z
<input type="checkbox"/>		sg-94ce85e5	ElasticMapReduce-slave	vpc-c5871cbc	Slave group for Elastic MapReduce created on 2017-07-17T16:05:38.663Z

All TCP	TCP	0 - 65535	sg-53ce8522 (ElasticMapReduce-master)
All TCP	TCP	0 - 65535	sg-94ce85e5 (ElasticMapReduce-slave)
SSH	TCP	22	0.0.0.0/0

Accessing Master Node from the Internet

- Find the public ip address of the master
- Download PuTTY.exe to your computer from: <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
- Start PuTTY.
- In the Category list, click Session.
- In the Host Name field, type **hadoop@ec2-34-229-171-166.compute-1.amazonaws.com**
- In the Category list, expand Connection > SSH, and then click Auth.
- For Private key file for authentication, click Browse and select the private key file (**hadoop.ppk**) used to launch the cluster.
- Click Open.
- Click Yes to dismiss the security alert.

Cluster: HadoopCluster **Waiting** Cluster ready after last step completed.

Connections: [Enable Web Connection](#) – Hue, Ganglia, Resource Manager ... (View All)

Master public DNS: ec2-34-229-171-166.compute-1.amazonaws.com [SSH](#)

Tags: -- [View All](#) / [Edit](#)