Hands on

Exploring Machine learning with Spark &
Real-world Scalable Architecture

Prakshi Yadav
Data Engineer @Episource

Prerequisites

- Python Programming
- Higher level understanding of Distributed computing
- Map-Reduce Paradigm

What was covered earlier?

- Installation
- Resilient Distributed Datasets (RDD)
  - Create RDDs
  - RDD transformations
- Spark SQL
- Introduction Spark Streaming

What we will cover today?

- Exploring Real-world Architectures
  - Scalable Data Processing Pipeline

Business Problem at EPISOURCE

- Clinical Text - Medical Records of patients
- Distributions of Medical Charts
  - Client → Operations Team → Team Leads → Medical Coders
- ICD-10 Coding by Medical Coders
  - International Classification of Diseases -
    codes for diseases, signs and symptoms, abnormal findings, complaints, social circumstances, and external causes of injury or diseases
  - Category - Hierarchical Condition Category (HCC) vs No HCC
    Example: Viral Fever - *A92.8 - HCC - billable*
    Whooping cough - *A37 - No HCC - Non Billable*
- Revenue based on no. of HCC Charts being coded
  - Each Medical Coder codes 10-12 charts/day depending on expertise

Challenges:

- Processing of thousands of files
- Dynamic distribution of files for processing
- Fault tolerant architecture
- Data security is the priority
- Cost minimization
- Fully Automated data pipeline

General way to APPROACH

Maintain a separate data store

Keep dedicated servers running all day to process files.

Server maintenance is an overhead

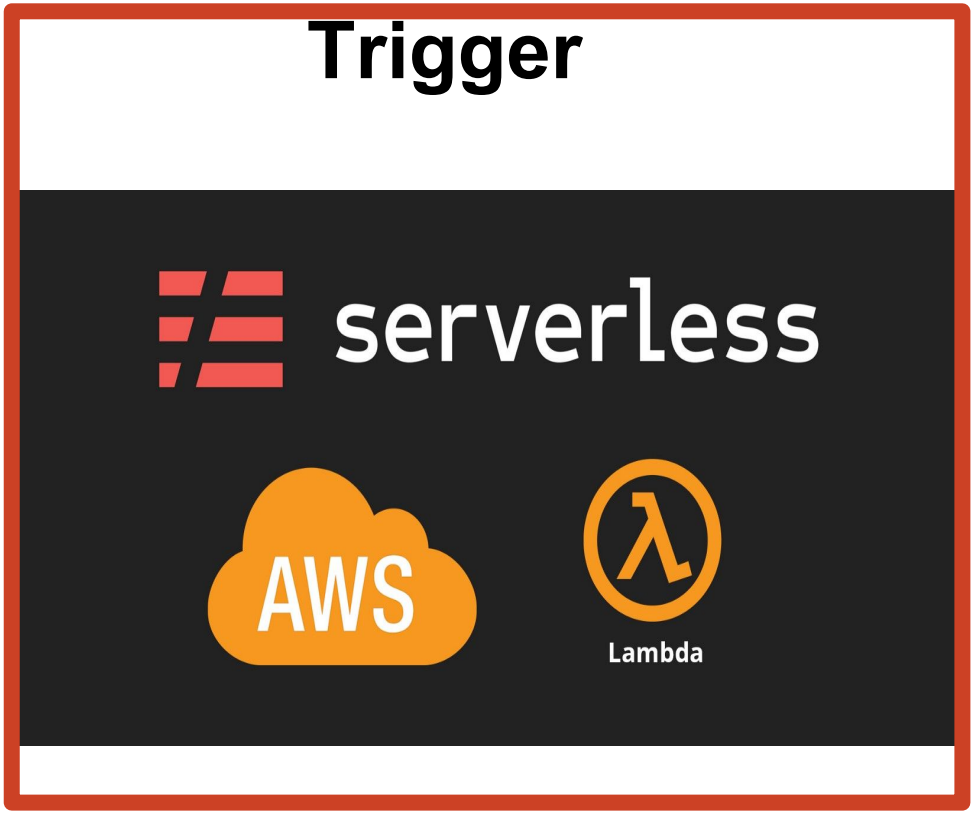
Less flexibility to scale the resources.

Serverless path

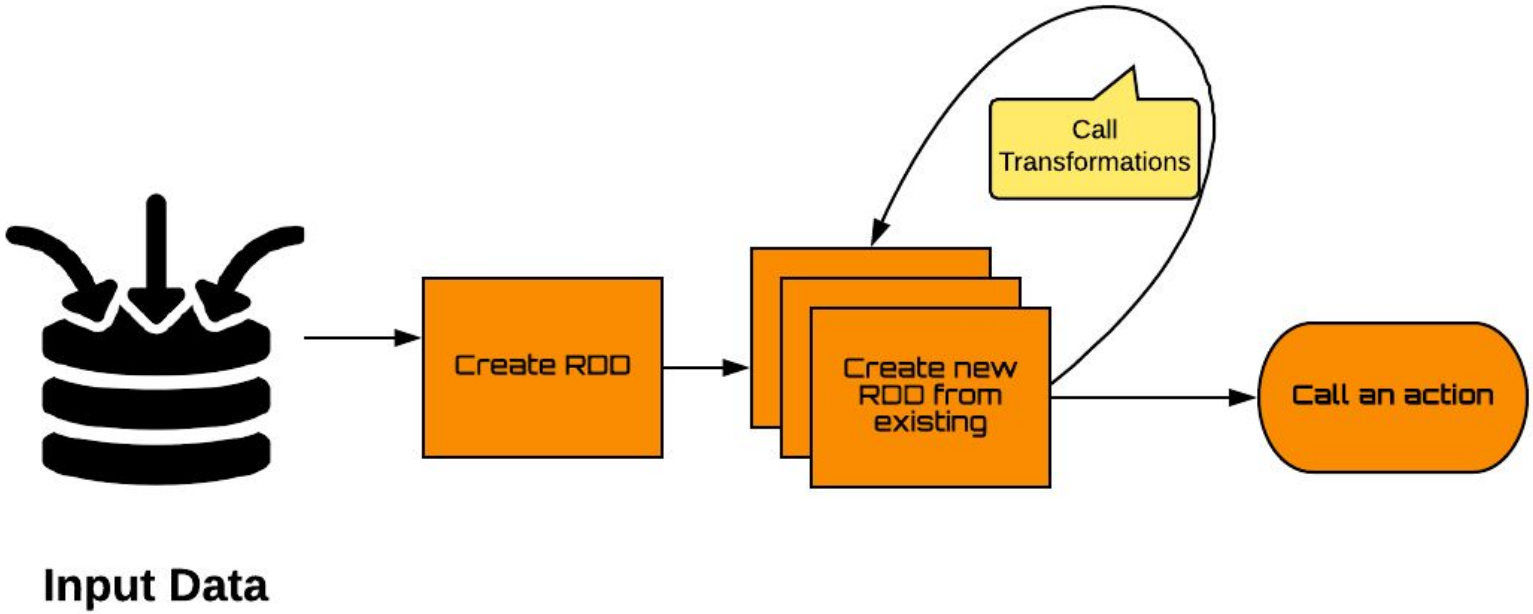
- Event-based asynchronous approach to application design
- Launch clusters only when required.
- Three simple steps:
  - Start
  - Process
  - Terminate
- Reduced operational costs
- Deploy in Minutes
- Enhanced scalability, better performance, and lower infrastructure and maintenance costs

What makes it serverless?

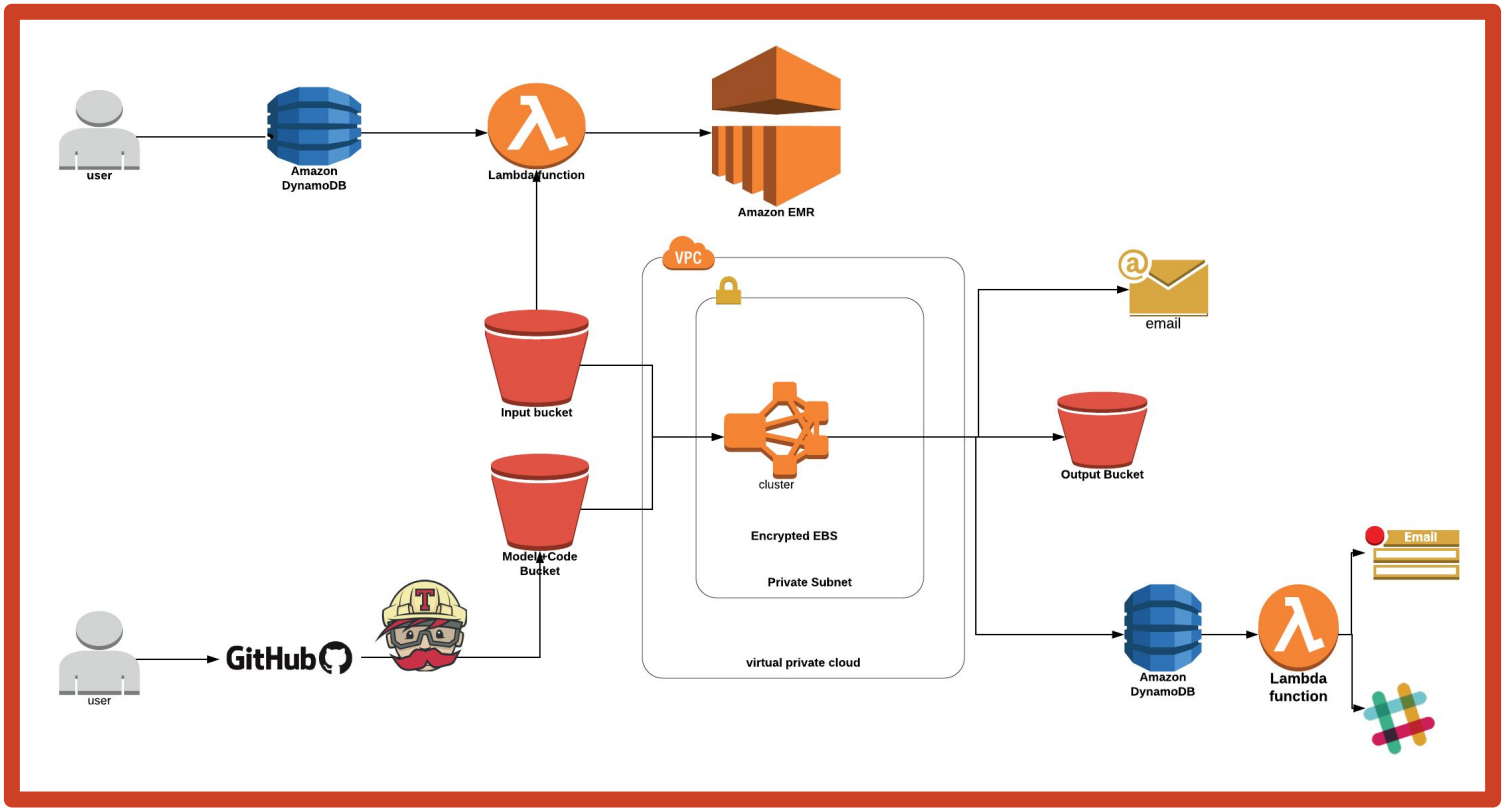
- EMR Cluster is launched only when there are files to process.
- Lambda function is triggered by a cloudwatch event
- Lambda function contains the code to launch an EMR Cluster.
- Cluster is terminated automatically once the processing is over.



Ideal Spark applications



Serverless Data processing pipeline

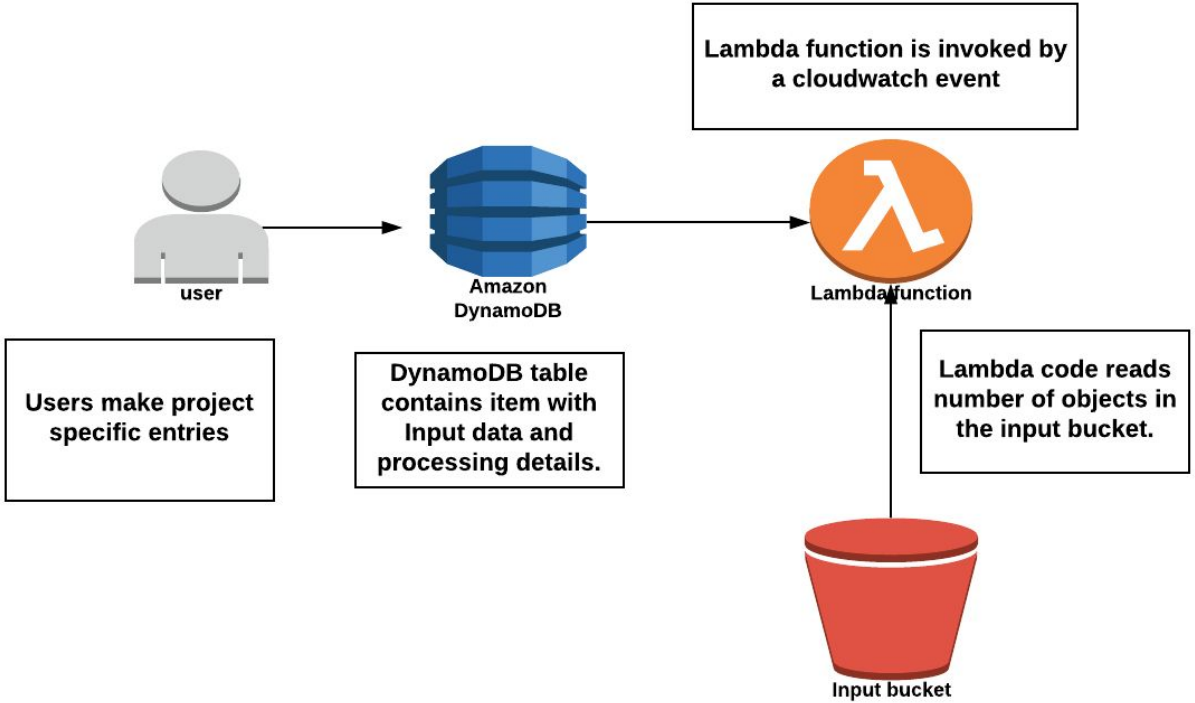


Where is data stored?

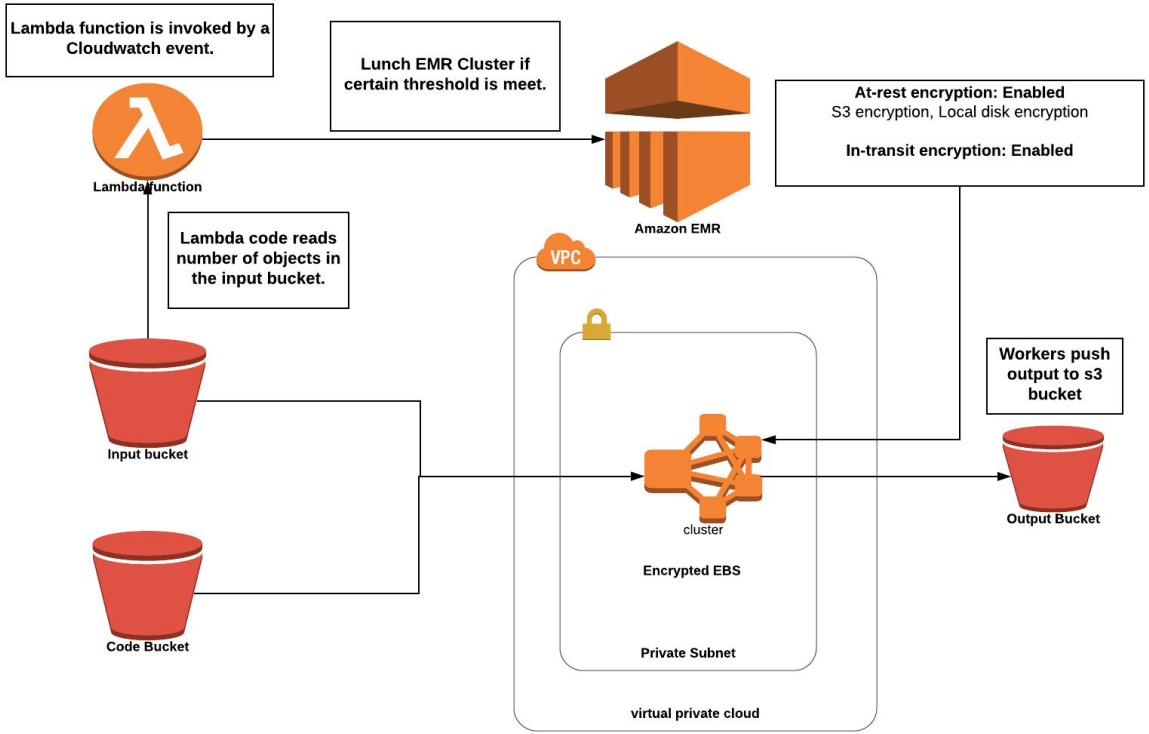
- **Scales -** just keep putting files, and it will never fill up.
- Upload and download your data with SSL encrypted end points
- Provides multiple options for encrypting data at rest.
- **Low Cost -** $0.023 per GB
- Reading data in a Spark application is as simple as calling - sc.textFile("s3n://<bucketname>")

Object Storage

What Triggers the pipeline?

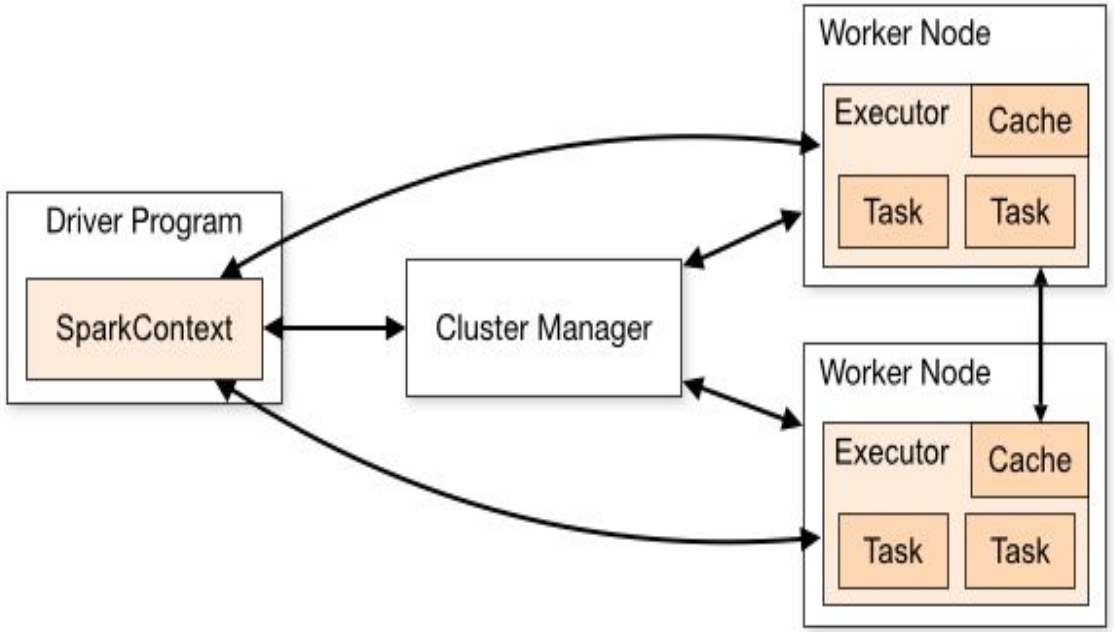


Batch processing with EMR



Apache spark architecture

- Single Master & Multiple Workers.
- Driver Program resides on Master node.
- Multiple executors may reside on a worker node.



What role does a driver play in a spark architecture?

- Runs the main () function of the application and is the place where the Spark Context is created
- Translates the RDD's into the execution graph and splits the graph into multiple stages
- Converts User Application into tasks
- Exposes the information about the running spark application through a Web UI at port 4040.

So, Smart Driver is more important!

Role of cluster manager

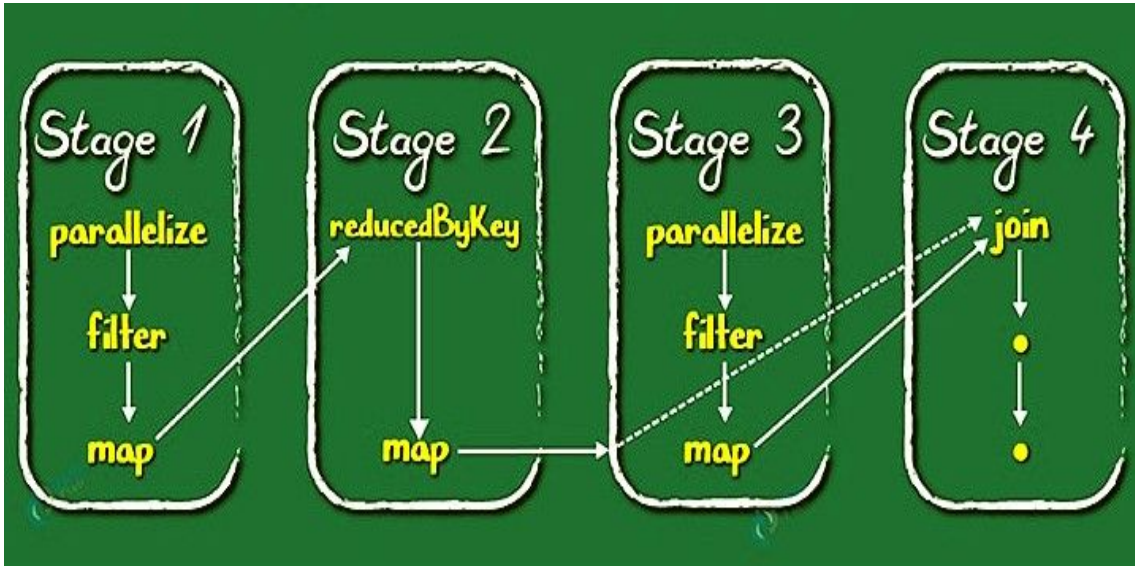
- Responsible for acquiring resources on the spark cluster and allocating them to a spark job.
- 3 different types of cluster managers:
  - Hadoop YARN
  - Apache Mesos
  - Simple standalone spark cluster manager
- Allocation and deallocation of various physical resources such as memory for client spark jobs, CPU memory, etc.

DAG SCHEDULER

- Stages depends on transformations on RDDs

- **Narrow transformation**: Doesn't require the data to be shuffled across the partitions. for example, Map, filter etc.

- **Wide transformation -** requires the data to be shuffled for example, reduceByKey etc.



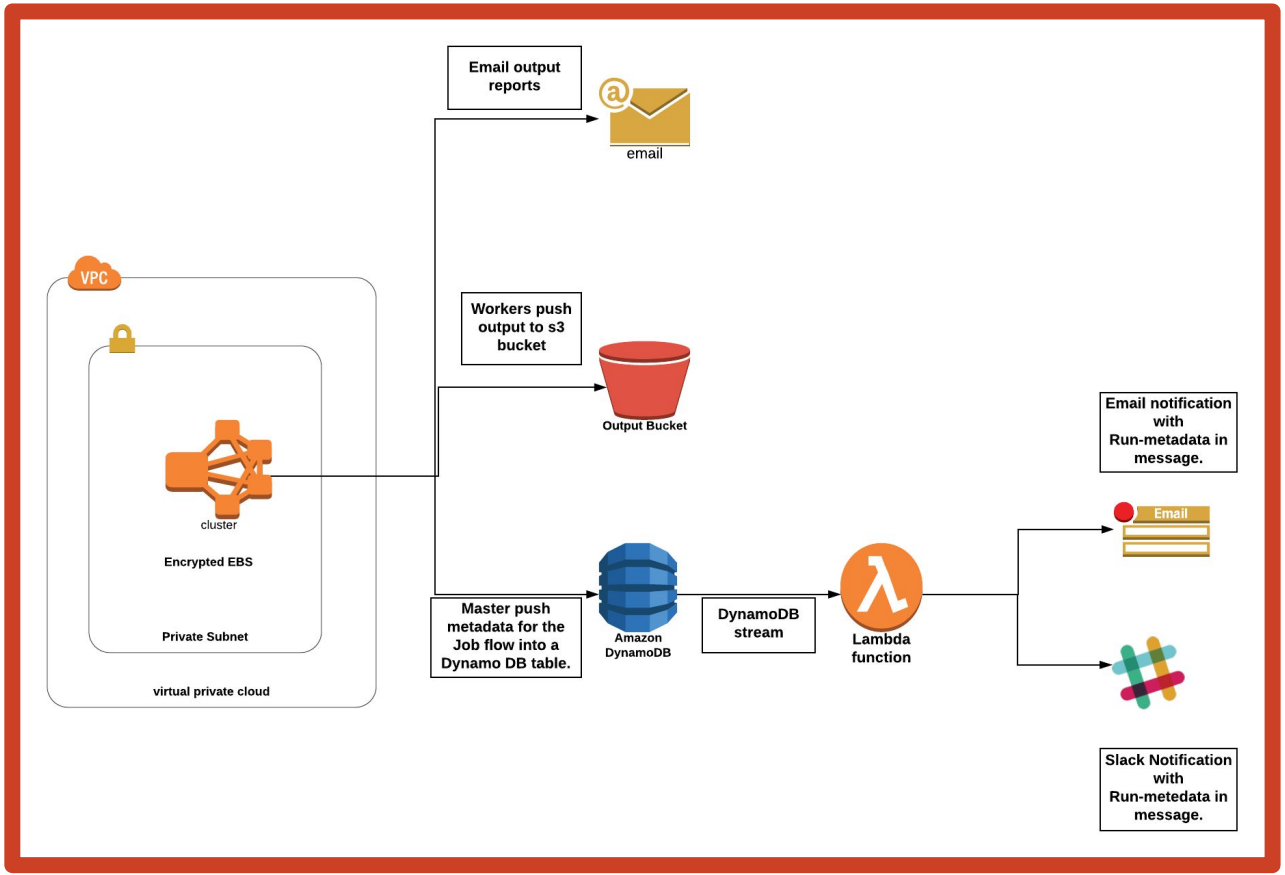
Wide transformations results in stages

RUN TIME ARCHITECTURE OF A SPARK APPLICATION

1. Creates a logical DAG from the RDD transformation and actions given in the application
2. Physical execution plan with set of stages
3. Creates tasks and send them to the Cluster
4. Cluster manager then launches executors on the worker nodes depending on resource allocation
5. Executors registers themselves with the driver program.
6. Executors start executing the various tasks assigned by the driver program
7. If main() ⇒ exits the call SparkContext.stop()

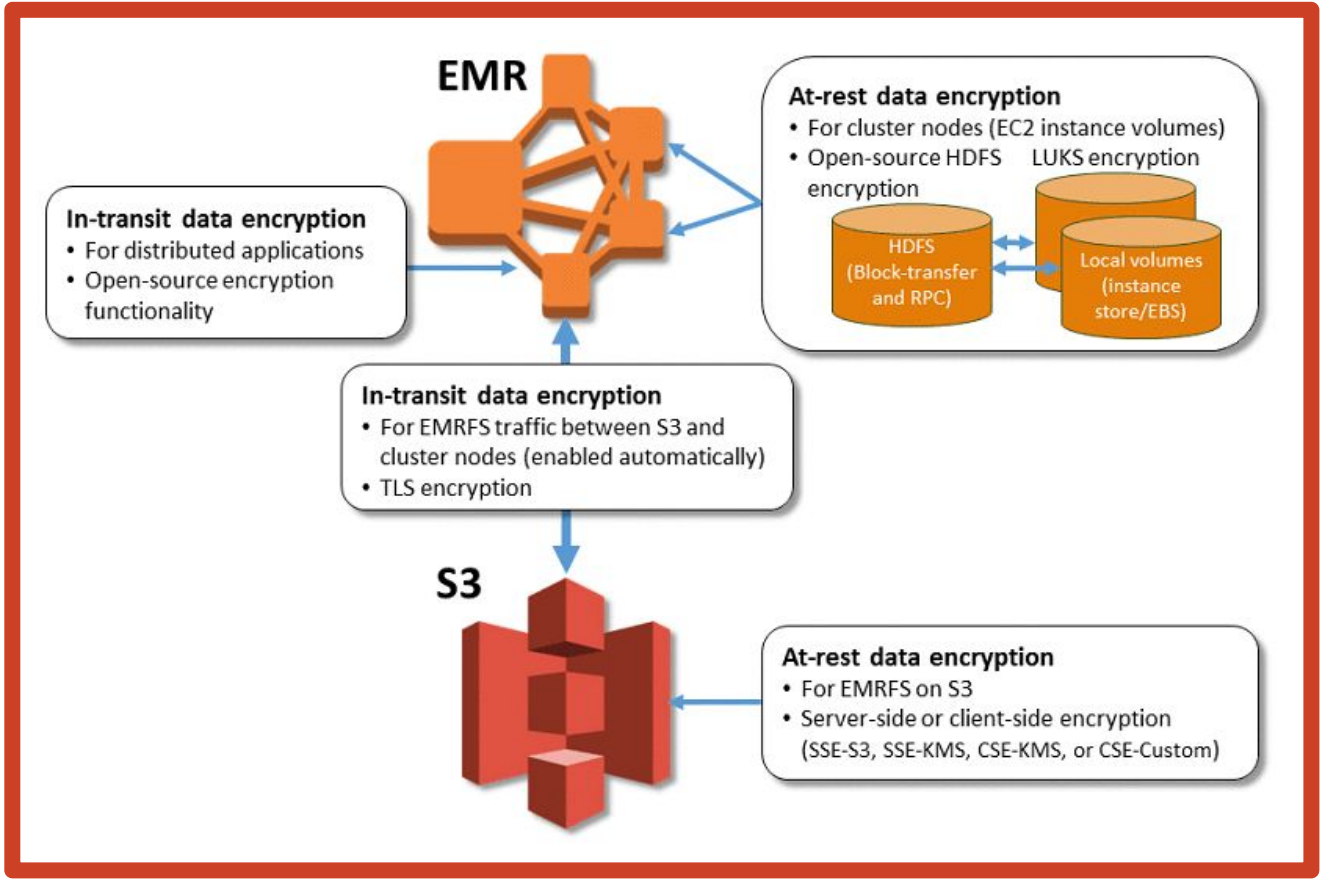
NOTIFICATION SYSTEM

After every run, a notification is sent to the user which contains run-metadata in the message



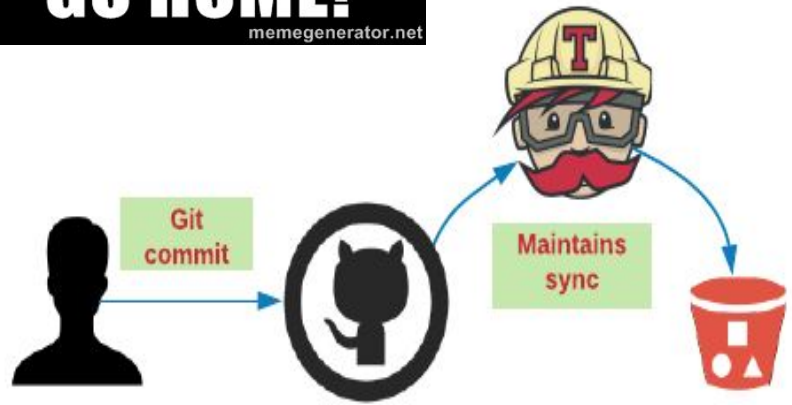
Data security in our architecture

- **Security configurations**: Templates for security configurations which can be attached to EMR Clusters.
- SC provides:
  - **At-rest data encryption**
  - **In-transit data encryption**



Code maintenance

- Our code base resides on Git.
- EMR fetches code from s3 bucket so it's necessary to create a sync between Git and s3
- Travis CI - continuous integration service
- Travis CI sync the S3 bucket with git repo on every commit to the master branch



QUESTIONS?