

Introduction to AWS Big Data

Omid Vahdaty, Big Data Ninja

When the data outgrows your ability to process

- Volume
- Velocity
- Variety



EMR

- Basic, the simplest distribution of all hadoop distributions.
- Not as good as cloudera.




Collect


- Firehose
- Snowball
- SQS
- Ec2



Store

- S3
 - Kinesis
 - RDS
 - DynmoDB
 - Clloud Search
 - lot
- 

Process + Analyze

- Lambda
 - EMR
 - Redshift
 - Machine learning
 - Elastic search
 - Data Pipeline
 - Athena
- 

Visulize

- Quick sight
- ElasticSearch service



History of tools

HDFS - from google


Cassandra - from facebook, Columnar store NoSQL , materialized view, secondary index

Kafka - linkedin.


Hbase, no sql , hadoop eco system.



EMR Hadoop ecosystem

- Spark - recommend options. Can do all the below...
 - Hive
 - Oozie
 - Mahout - Machine library (mllib is better, runs on spark)
 - Presto, better than hive? More generic than hive.
 - Pig - scripting for big data.
 - Impala - Cloudera, and part of EMR.
- 

ETL Tools


- Attunity
 - Splunk
 - Semarchy
 - Informatica
 - Tibco
 - Clarit
- 

Architecture


- Decouple :
 - Store
 - Process
 - Store
 - Process
 - insight...
- Rule of thumb: 3 tech in dc, 7 techs max in cloud
 - Do use more b/c: maintenance



Architecture considerations

- unStructured? Structured? Semi structured?
 - Latency?
 - Throughput?
 - Concurrency?
 - Access patterns?
 - Pass? Max 7 technologies
 - IaaS? Max 4 technologies
- 

EcoSystem

- Redshift = analytics
 - Aurora = OLTP
 - DynamoDB = NoSQL like mongodb.
 - Lambda
 - SQS
 - Cloud Search
 - Elastic Search
 - Data Pipeline
 - Beanstalk - i have jar, install it for me...
 - AWS machine learning
- 

Data ingestions Architecture challenges

- Durability
- HA
- Ingestions types:
 - Batch
 - Stream
- Transaction: OLTP, noSQL.



Ingestion options


- Kinesis
- Flume
- Kafka
- S3DistCP copy from
 - S3 to HDFS
 - S3 to s3
 - Cross account
 - Supports compression.

Transfer

- VPN
- Direct Connect
- S3 multipart upload
- Snowball
- IoT




Steaming

- Streaming
 - Batch
 - Collect
 - Kinesys
 - DynamoDB streams
 - SQS (pull)
 - SNS (push)
 - KAFKA (Most recommend)
- 

Comparison

	Amazon DynamoDB Streams	Amazon Kinesis Streams	Amazon Kinesis Firehose	Amazon SQS (Standard)	Amazon SQS (FIFO)	Apache Kafka
AWS managed	✓	✓	✓	✓	✓	✗
Guaranteed ordering	✓	✓	✗	✗	✓	✓
Delivery (deduping)	Exactly-once	At-least-once	At-least-once	At-least-once	Exactly-once	At-least-once
Data retention	24 hours	7 days	N/A	14 days	14 days	Configurable
Availability	3 AZ	3 AZ	3 AZ	3 AZ	3 AZ	Configurable
Scale/ throughput	No limit/ ~ table IOPS	No limit/ ~ shards	No limit/ automatic	No limits/ automatic	300 TPS/ queue	No limit/ ~ nodes
Parallel consumption	✓	✓	✗	✗	✗	✓
Stream MapReduce	✓	✓	N/A	N/A	N/A	✓
Row/object size	400 KB	1 MB	Destination row/object size	256 KB	256 KB	Configurable
Cost	Higher (table cost)	Low	Low	Low-medium	Low-medium	Low (+admin)

Streaming

- Low latency
 - Message delivery
 - Lambda architecture implementation
 - State management
 - Time or count based windowing support
 - Fault tolerant
- 

Stream processor comparison

	Amazon EMR (Spark Streaming)	KCL Application	Amazon Kinesis Analytics	AWS Lambda	Apache Storm
AWS managed	✓ (Amazon EMR)	✗ (EC2 + Auto Scaling)	✓	✓	✗
Serverless	✗	✗	✓	✓	✗
Scale / throughput	No limits / ~ nodes	No limits / ~ nodes	Up to 8 KPU / automatic	No limits / automatic	No limits / ~ nodes
Availability	Single AZ	Multi-AZ	Multi-AZ	Multi-AZ	Configurable
Programming languages	Java, Python, Scala	Java, others via MultiLangDaemon	ANSI SQL with extensions	Node.js, Java, Python	Almost any language via Thrift
Uses	Multistage processing	Single stage processing	Multistage processing	Simple event-based triggers	Multistage processing
Reliability	KCL and Spark checkpoints	Managed by KCL	Managed by Amazon Kinesis Analytics	Managed by AWS Lambda	Framework managed

Stream collection options

Kinesis client library

AWS lambda

EMR

Third party

Spark streaming (latency min =1 sec) , near real time, with lot of libraries.

Storm - Most real time (sub millisec), java code based.

Flink (similar to spark)

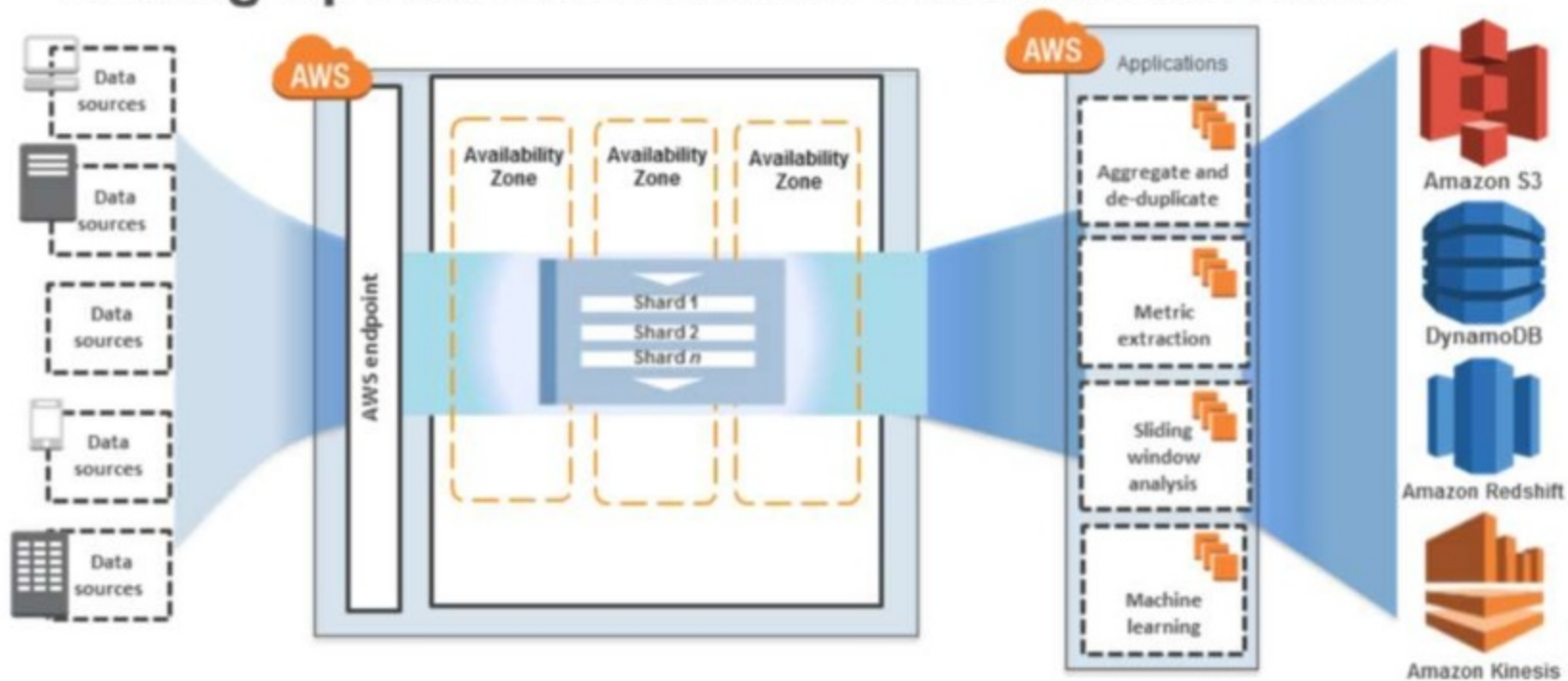


Kinesys

- Stream - **collect@source** and near real time processing
 - Near real time
 - High throughput
 - Low cost
 - Easy administration - set desired level of capacity
 - Delivery to : s3,redshift, Dynamo
 - Ingress 1mb, egress 2mbs. Upto 1000 Transaction per second.
- Analytics - in flight analytics.
- Firehose - Park you **data @ destination**.



Setting up Amazon Kinesis Streams: In Detail



Kinesis analytics example

```
CREATE STREAM "INTERMEDIATE_STREAM" (  
  hostname VARCHAR(1024),  
  logname VARCHAR(1024),  
  username VARCHAR(1024),  
  requesttime VARCHAR(1024),  
  request VARCHAR(1024),  
  status VARCHAR(32),  
  responsesize VARCHAR(32)  
);
```

-- Data Pump: Take incoming data from SOURCE_SQL_STREAM_001 and insert into INTERMEDIATE_STREAM

KCL

- Read from stream using get api
- Build application with the KCL
- Leverage kinesis spout for storm
- Leverage EMR connector



Firehose - for parking

- Not for fast lane - no in flight analytics
- Capture , transform and load.
 - Kinesis
 - S3
 - Redshift
 - elastic search
- Managed Service
- Producer - you input to delivery stream
- Buffer size MB
- Buffer in seconds.



Comparison of Kinesis product

- Stream
 - **Sub 1 sec processing latency**
 - Choice of stream processor (generic)
 - For smaller events
- Firehose
 - Zero admin
 - 4 targets built in (redshift, s3, search, etc)
 - **Latency 60 sec minimum.**
 - For larger "events"



DynamoDB

- Fully managed NoSql document store key value
 - Tables have not fixed schema
- High performance
 - Single digit latency
 - Runs on solid drives
- Durable
 - Multi az
 - Fault tolerant, replicated 3 AZ.




Durability

- Read

- Eventually consistent
- Strongly consistent

- Write

- Quorum ack
 - 3 replication - always - we can't change it.
 - Persistence to disk.
- 

Indexing and partitioning

- Indexing

- LSI - local secondary index, kind of alternate range key
- GSI - global secondary index - “pivot charts” for your table, kind of projection (vertical)

- Partitioning

- Automatic
- Hash key spread data across partitions




DynamoDB Items and Attributes


- Partitions key
- Sort key (optional)
- LSI
- attributes



AWS Titan(on DynamoDB) - Graph DB

- **Vertex** - nodes
 - **Edge**- relationship b/w nodes.
 - Good when you need to investigate more than 2 layer relationship (join of 4 tables)
 - Based on TinkerPop (open source)
 - Full text search with lucene SOLR or elasticsearch
 - HA using multi master replication (cassandra backed)
 - Scale using DynamoDB backed.
 - Use cases: Cyber , Social networks, Risk management
- 

Elasticache: MemCache / Redis

- Cache sub millisecond.
 - In memory
 - No disk I/O when querying
 - High throughput
 - High availability
 - Fully managed
- 

Redshift

- Peta scale database for analytics
 - Columnar
 - MPP
- Complex SQL



RDS

- Flavours
 - Mysql
 - Auraora
 - postgresSQL
 - Oracle
 - MSSQL
 - Multi AZ, HA
 - Managed services
- 

Data Processing

- Batch
 - Ad hoc queries
 - Message
 - Stream
 - Machine learning
 - (by the use parquet, not ORC, more commonly used in the ecosystem)
- 

Athena

- Presto
- In memory
- Hive meta store for DDL functionality
 - Complex data types
 - Multiple formats
 - Partitions



EMR

- Emr , pig and hive can work on top of spark , but not yet in EMR
- TEZ is not working. Horton gave up on it.
- Can run presto on hive



Best practice

- Bzip2 - splittable compression
 - You don't have to open all the file, you can `bzip2` only one block in it and work in parallel
- Snappy - encoding /decoding time - VERY fast, not compress well.
- Partitions
- Ephemeral EMR



EMR ecosystem

- Hive
- Pig
- Hue
- Spark
- Oozie
- Presto
- Ganglia
- Zookeeper
- zeppelin
- For research - Public data sets exist of AWS...



EMR ARchitecture

- Master node
- Core nodes - like data nodes (with storage)
- Task nodes - (not like regular hadoop, extends compute)
- Default replication factor
 - Nodes 1-3 \Rightarrow 1 factor
 - Nodes 4-9 \Rightarrow 2 replication factor
 - Nodes 10+ \Rightarrow 3 replication factor
 - **Not relevant in external tables**
- Does Not have Standby Master node
- Best for transient cluster (goes up and down every night)



EMR

- Use SCALA
- If you use Spark SQL , even in python it is ok.
- For code - python is full of bugs, connects well to R
- Scala is better - but not connects easily to data science \ R
- Use cloud formation when ready to deploy fast.
- Check instance I ,
- Dense instance is good architecture
- User spot instances - for the tasks.
- Use TAGs
- Constant upgrade of the AMI version
- Don't use TEZ
- Make sure your choose instance with network optimized
- Resize cluster is not recommended
- Bootstrap to automate cluster upon provisioning
- Steps to automate steps on running cluster
- Use RDS to share Hive MetaStore (the metastore is mysql based)
- **Use r, kafka and impala via bootstrap actions and many others**



CPU features for instances

AWS Instance Type	High Memory X1	Compute Optimized C4	Storage Optimized D2	General Purpose M4	Memory Optimized R3	IO Optimized I2	Graphics Optimized G2	Burstable Performance T2
Intel Processor	Intel Xeon E7-8880 v3	Custom Intel Xeon E5-2666 v3	Custom Intel Xeon E5-2676 v3	Custom Intel Xeon E5-2676 v3	Intel Xeon E5-2670 v2	Intel Xeon E5-2670 v2	Intel Xeon E5-2670	Intel Xeon Family
Intel AVX	AVX 2.0	AVX 2.0	AVX 2.0	AVX 2.0	✓	✓	✓	✓
Intel AES-NI	✓	✓	✓	✓	✓	✓	✗	✗
Intel Turbo Boost	✓	✓	✓	✓	✓	✓	✓	✓
Intel TSX	✓	✗	✗	✗	✗	✗	✗	✗
Per core P- and C-state control	✗	✓ (8xlarge only)	✗	✗	✗	✗	✗	✗
SSD Storage	EBS Optimized by default	EBS Optimized by default	✗	EBS Optimized by default	✓	✓	✓	EBS only

Sending work to emr


- Steps -
 - can be added while cluster is running
 - Can be added from the UI / CLI
 - FIFO scheduler by default
- Emr api
- Ganglia: `jvm.JvmMetrics.MemHeapUsedM`




Landscape

Batch Processing	Interactive Analytics	Machine Learning	Stream Processing	
			Micro-Batch	Near Real-Time
Spark		Amazon ML	Amazon Kinesis	
MapReduce	Presto	SparkML	Spark Streaming	AWS Lambda
Hive	Impala	Mahout	Hive	Apache Flink
Pig			Pig	Apache Storm


Hive

- SQL over hadoop.
 - Engine: spark, tez, ML
 - JDBC / ODBC
 - Not good when need to shuffle.
 - Connect well with DynamoDB
 - SerDe json, parquet,regex etc.
- 

Hbase

- Nosql like cassandra
 - Below the Yarn
 - Agent of HBASE on each data node. Like impala.
 - Write on HDFS
 - Multi level Index.
 - (AWS avoid talking about it b/c of Dynamo, used only when you want to save money)
 - Driver from hive (work from hive on top of HBase)
- 

Presto

- Like Hive, from facebook.
 - Not good always for join on 2 large tables.
 - Limited by memory
 - Not fault tolerant like hive.
 - Optimized for ad hoc queries
- 

Pig

- Distributed Shell scripting
- Generating SQL like operations.
- Engine: MR, Tez
- S3, DynamoDB access
- Use Case: for data science who don't know SQL, for system people, for those who want to avoid java/scala
- Fair fight compared to hive in term of performance only
- **Good for unstructured files ETL : file to file , and use sqoop.**

Spark

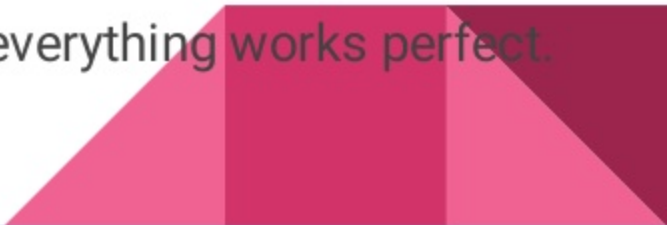
	Query Latency	SQL Compatibility
Hive	High (MapReduce)	✓
	Medium (Tez)	
Pig	High (MapReduce)	✗
	Medium (Tez)	
Spark	Low	✓ (SparkSQL)
Presto	Low	✓

Mahut

- Machine library for hadoop - Not used
- Spark ML lib is used instead.



R

- Open source package for statistical computing.
 - Works with EMR
 - “Matlab” equivalent
 - Works with spark
 - Not for developer :) for statistician
 - R is single threaded - use spark R to distribute. Not everything works perfect.
- 

Apache Zeppelin

- Notebook - visualizer
- Built in spark integration
- Interactive data analytics
- Easy collaboration.
- Uses SQL
- works on top of Hive
- Inside EMR.
- Give more feedback to let you know where u are




Hue

- Hadoop user experience
- Logs in real time and failures.
- Multiple users
- Native access to S3.
- File browser to HDFS.
- Manipulate metastore
- Job Browser
- Query editor
- Hbase browser
- Sqoop editor, oozier editor, Pig Editor



Spark


- In memory
 - X10 to X100 times faster
 - Good optimizer for distribution
 - Rich API
 - Spark SQL
 - Spark Streaming
 - Spark ML (ML lib)
 - Spark GraphX (DB graphs)
 - SparkR
- 

Spark

- RDD
 - An array (data set)
 - Read only distributed objects cached in mem across cluster
 - Allow apps to keep working set in mem for reuse
 - Fault tolerant
 - Ops:
 - Transformation: map/ filter / groupby / join
 - Actions: count / reduce/ collect / save / persist
 - Object oriented ops
 - High level expression like lambda, functions, map



Data Frames

- Dataset containing named columns
 - Access ordinal positions
 - Tungsten execution backed - **framework for distributed memory (open source?)**
 - Abstraction for selecting , filtering, agg, and plot structure data.
 - Run sql on it.
- 

Streaming

- Near real time (1 sec latency) , like batch of 1sec windows
- Same spark as before
- Streaming jobs with API



Spark ML

- Classification
 - Regression
 - Collaborative filtering
 - Clustering
 - Decomposition
 - Code: java, scala, python, sparkR
- 

Spark GraphX

- Works with graphs for parallel computations
- Access to access of library of algorithm
 - Page rank
 - Connected components
 - Label propagation
 - SVD++
 - Strongly connected components



Hive on Spark

- Will replace tez.



Spark flavours

- Own cluster
- With yarn
- With mesos



Downside

- Compute intensive
- Performance gain over mapreduce is not guaranteed.
- Streaming processing is actually batch with very small window.




Redshift


- OLAP, not OLTP→ analytics , not transaction
- Fully SQL
- Fully ACID
- No indexing
- Fully managed
- Petabyte Scale
- MPP
- Can create slow queue for queries which are long lasting.
- DO NOT USE FOR transformation.



EMR vs Redshift

- How much data loaded and unloaded?
 - Which operations need to be performed?
 - Recycling data? → EMR
 - History to be analyzed again and again ? → emr
 - What the data needs to end up? BI?
 - Use spectrum in some use cases.
 - Raw data? s3.
- 

Hive VS. Redshift

- Amount of concurrency ? low → hive, high → redshift
 - Access to customers? Redshift?
 - Transformation, Unstructured , batch, ETL → hive.
 - Peta scal ? redshift
 - Complex joins → Redshift
- 

Presto VS redshift

- Not a true DW , but can be used.
- Require S3, or HDFS
- Netflix uses presto for analytics.



Redshift

- Leader node
 - Meta data
 - Execution plan
 - Sql end point
- Data node
- Distribution key
- Sort key
- Normalize...
- Dont be afraid to duplicate data with different sort key



Redshift Spectrum

- External table to S3
- Additional compute resources to redshift cluster.
- Not good for all use cases



Cost consideration

- Region
- Data out
- Spot instances
 - 50% - 80% cost reduction.
 - Limit your bid
 - Work well with EMR. use spot instances for task core mostly. For dev - use spot.
 - May be killed in the middle :)
- reserved instances

Kinesis pricing

- Streams

- Shard hour
- Put payload unit , 25KB

- Firehose

- Volume
- 5KB

- Analytics

- Pay per container , 1 cpu, 4 GB = 11 Cents



Dynamo

- Most expensive: Pay for throughput
 - 1k write
 - 4k read
 - Eventually consistent is cheaper
- Be carefull... from 400\$ to 10000\$ simply b/s you change block size.
- You pay for storage, until 25GB free, the rest is 25 cent per GB



Optimize cost

- Alerts for cost mistakes-
 - unused machines etc.
 - on 95% from expected cost. → something is wrong...
- No need to buy RI for 3 year, 1year is better.



Visualizing - Quick Sight

- Cheap
- Connect to everything
 - S3
 - Dynamo
 - Redhsift
- SPICE
- Use storyboard to create slides of graphs like PPT



Tableau

- Connect to Redshift
- Good look and feel
- ~1000\$ per user
- Not the best in term of performance. Quick sight is faster.




Other visualizer

- Tibco - Spot file
 - Has many connector
 - Had recommendations feature
- Jaspersoft
 - Limited
- ZoomData
- Hunk -
 - Users EMR and S3
 - Scehmoe on the fly
 - Available in market place
 - Expensive.
 - Non SQL , has it own language.



Visualize - which DB to choose?


- Hive is not recommended
 - Presto a bit faster.
 - Athena is ok
 - Impala is better (has agent per machine, caching)
 - Redshift is best.
- 

Orchestration

- Oozie
 - Opens source workflow
 - Workflow: graph of action
 - Coordinator: scheduler jobs
 - Support: hive, sqoop , spark etc.
- Data pipeline
 - Move data from on prem to cloud
 - Distributed
 - Integrate well with s3, dynamodb, RDS, EMR, EC2, redshift
 - Like ETL: Input, data manipulation, output
 - Not trivial, but nicer than Oozie
- Other: AirFlow, Knime, Luigi, Azkaban




Security

- Shared security model
 - Customer:
 - OS, platform, identity, access management, role, permissions
 - Network
 - Firewall
 - AWS:
 - compute, storage, databases, networking, LB
 - Regions, AZ, edge location
 - compliance
- 


EMR secured

- IAM
- VPC
- Private subnet
- VPC endpoint to S3
- MFA to login
- STS + SAML - token to login. Complex solution
- Kerberos authentication of Hadoop nodes
- Encryption
 - SSH to master
 - TLS between nodes


IAM

- User
 - Best practice :
 - MFA
 - Don't use ROOT account
 - Group
 - Role
 - Policy best practice
 - Allow X
 - Disallow all the rest
 - Identity federation via LDAP
- 


Kinesis Security best practice

- Create IAM admin for admin
 - Create IAM entity for re sharing stream
 - Create IAM entity for produces to write
 - Create iam entity for consumer to read.
 - Allow specific source IP
 - Enforce AWS:SecureTransport condition key for every API call.
 - Use temp credentials : IAM role
- 


Dynamo Security

- Access
 - Policies
 - Roels
 - Database level access level - row level (item)/ Col level (attribute) access control
 - **STS for web identity federation.**
 - Limit of amount row u can see
 - Use SSL
 - All request can be signed via SHA256
 - PCI, SOC3, HIPPA, 270001 etc.
 - Cloud trail - for Audit log
- 

Redshift Security

- SSL in transit
 - Encryption
 - KMSAES256
 - HSM encryption (hardware)
 - VPC
 - Cloud Trail
 - All the usual regulation certification.
 - Security groups
- 

Big Data patterns

- Interactive query → mostly EMR, Athena
 - Batch Processing → reporting → redshift + EMR
 - Stream processing → kinesis, kinesis client library, lambda, EMR
 - Real time prediction → mobile, dynamoDB. → lambda+ ML.
 - Batch prediction → ML and redshift
 - Long running cluster → s3 → EMR
 - Log aggregation → s3 → EMR → redshift
- 

Stay in touch...

- [Omid Vahdaty](#) 
- +972-54-2384178



NEWS  **וואלה**




jajah
Telefonica

