



AWS Big Data Demystified #1.1

Big Data Architecture

Lessons Learned

Omid Vahdaty, Big Data Ninja

Disclaimer

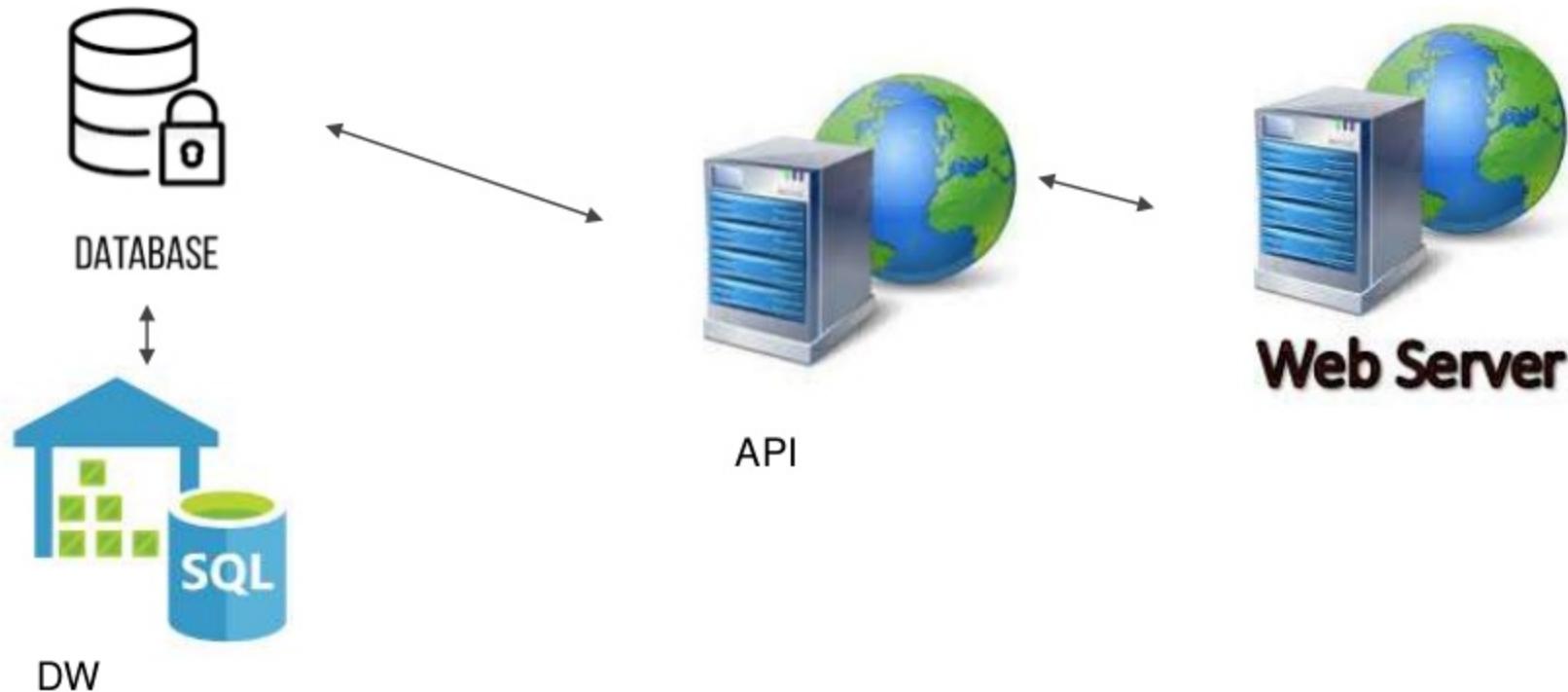
- I am not trying to sell anything.
- This is purely **knowledge transfer** session.
- You are more than welcome to **challenge** each slide, during the lecture and afterwards :)
- This lecture was **released at 2018**, take into account things change over time.

When the data outgrows your ability to process

- Volume (100TB processing per day)
- Velocity (4GB/s)
- Variety (JSON, CSV, events etc)
- Veracity (how much of the data is accurate?)



In the past (web,api, ops db, data warehouse)



TODAY'S BIG DATA APPLICATION STACK

PaaS and DC...



MY BIG DATA APPLICATION STACK “demystified”...



Amazon Kinesis



Kinesis Streams



Kinesis Firehose



Kinesis Analytics



redis



amazon
REDSHIFT



DynamoDB



Amazon S3



Amazon Athena



Spark SQL

L \leftrightarrow VY

ATACHE
SHIRO



TEZ

presto



MySQL
workbench

R



Apache Zeppelin

OOZIE
HUE

MY AWS BIG DATA
APPLICATION STACK
With some “Myst”.



redis



amazon
REDSHIFT

presto

MySQL
workbench



amazon
R D S



DynamoDB



Amazon S3



Amazon Athena



Amazon Kinesis



Kinesis Streams



Kinesis Firehose



Kinesis Analytics



amazon
EMR



Challenges creating big data architecture?



- What is the business use case ? How fast do u need the insights?
 - 15 min - 24 hours delay and above → use batch
 - Less than 15 min?
 - Might be batch - depends data source is files or events?.
 - Streaming?
 - Sub seconds delay?
 - Sub minute delay?
 - Streaming with in flight analytics ?
 - How complex is the compute jobs? Aggregations? joins?



Challenges creating big data architecture?



- What is the velocity?
 - Under 100K events per second? Not a problem
 - Over 1M events per second? Costly. But doable.
 - Over 1B events per seconds? Not trivial at all.
 - Volume ?
 - ~1TB a day ? Not a problem
 - Over ? it depends.
 - Over a petabyte? Well.... It depends.
 - Veracity (how are you going to handle different data sources?)
 - Structured (CSV)
 - Semi structured (JSON,XML)
 - Unstructured (pictures, movies etc)



Challenges creating big data architecture?

- Performance targets?
- Costs targets?
- Security restrictions?
- Regulation restriction? privacy?
- Which technology to choose?
- Datacenter or cloud?
- Latency?
- Throughput?
- Concurrency?
- Security Access patterns?
- Paas? Max 7 technologies
- Iaas? Max 4 technologies





Cloud Architecture rules of thumb...

- **Decouple :**

- Store
- Process
- Store
- Process
- insight...

- **Rule of thumb: max 3 technologies in dc, 7 tech max in cloud**

- Do use more b/c: maintenance
- Training time
- complexity/simplicity



CASE

Use Case 1: Analyzing browsing history

- Data Collection: browsing history from an ISP
- Product - derives user intent and interest for marketing purposes.
- Challenges
 - Velocity: 1 TB per day
 - History of: 3M
 - Remote DC
 - Enterprise grade security
 - Privacy



Use Case 2: Insights from location based data

- Data collection: from a Mobile operator
- Products:
 - derives user intent and interest for marketing purposes.
 - derive location based intent for marketing purposes.
- Challenges
 - **Velocity: 4GB/s ...**
 - **Scalability: Rate was expected double every year...**
 - Remote DC
 - Enterprise grade security
 - Privacy
 - **Edge analytics**

CASE



Use Case 3: Analyzing location based events.

- Data collection: streaming
- Product: building location based audiences
- Challenges: minimizing DevOps work on maintenance of a DIY streaming system



So what is the product?

- Big data platform that
 - collects data from multiple sources
 - Analyzes the data
 - Generates insights :
 - Smart Segments (online marketing)
 - Smart reports (for marketer)
 - Audience analysis (for agencies)
- Customers?
 - Marketers
 - Publishers
 - Agencies



My Big Data platform is about:

- **Data Collection**
 - **Online**
 - messaging
 - Streaming
 - **Offline**
 - Batch
 - Performance aspects
- **Data Transformation (Hive)**
 - JSON, CSV, TXT, PARQUET, Binary
- **Data Modeling - (R, ML, AI, DEEP, SPARK)**
- **Data Visualization (choose your poison)**
- **PII regulation + GDPR regulation**
- And: **Performance... Cost... Security... Simple... Cloud best practices...**





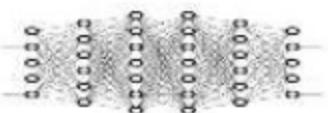
Big Data Generic Architecture



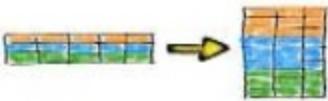
Data **Visualization**



Data **Modeling** (joins/agg/ML/R)



Data **Transformation** (row to **columnar** + cleansing)



Data **Collection**
(file based ETL from remote DC)



Big Data Generic Architecture | Data Collection



Batch Data collection considerations

- Every hour , about 30GB compressed CSV file
- Why s3
 - Multi part upload
 - S3 CLI
 - S3 SDK
 - (tip : gzip!)
- Why Client - needs to run at remote DC
- Why NOT your **own client**
 - Involves code →
 - Bugs?
 - maintenance
 - Don't analyze data at Edge , since you cant go back in time
- Why **Not Streaming?**
 - less accurate
 - Expensive





S3 Considerations

- Security
 - at rest: server side S3-Managed Keys (**SSE-S3**)
 - at transit: **SSL** / VPN
 - Hardening: **user, IP ACL, write permission only.**
- Upload
 - AWS **s3 cli**
 - **Multi part upload**
 - **Aborting Incomplete Multipart Uploads Using a Bucket Lifecycle Policy**
 - Consider S3 CLI **Sync** command instead of **CP**





Sqoop - ETL

- Open source , **part** of EMR
- **HDFS** to **RDMS** and back. Via **JDBC**.
- E.g BiDirectional **ETL** from **RDS** to **HDFS**
- **Unlikely use case: ETL from customer source DB.**

sqoop



Flume & Kafka

- Opens source project for streaming & messaging
- Popular
- Generic
- Good practice for many use cases. (a meetup by it self)
- Highly durable, scalable, extension etc.
- **Downside : DIY, Non trivial to get started**



X



X





Data Transfer Options

- VPN
- Direct Connect (4GB/s?)
- For all other use case
 - S3 multipart upload
 - Compression
 - Security
 - Data at motion
 - Data at rest
- bandwidth



AWS Direct Connect





Quick intro to Stream collection

- Kinesis Client Library ([code](#))
- AWS lambda ([code](#))
- EMR ([managed hadoop](#))
- Third party ([DIY](#))
 - **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)





Kinesis

- 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.
 - Not managed!
- Analytics - in flight analytics.
- Firehose - Park your data @ destination.





Firehose - for Data parking

- Not for fast lane - **no in flight analytics**
- Capture , transform and load.
 - Kinesis
 - S3
 - Redshift
 - elastic search
- Managed Service



Comparison of Kinesis product

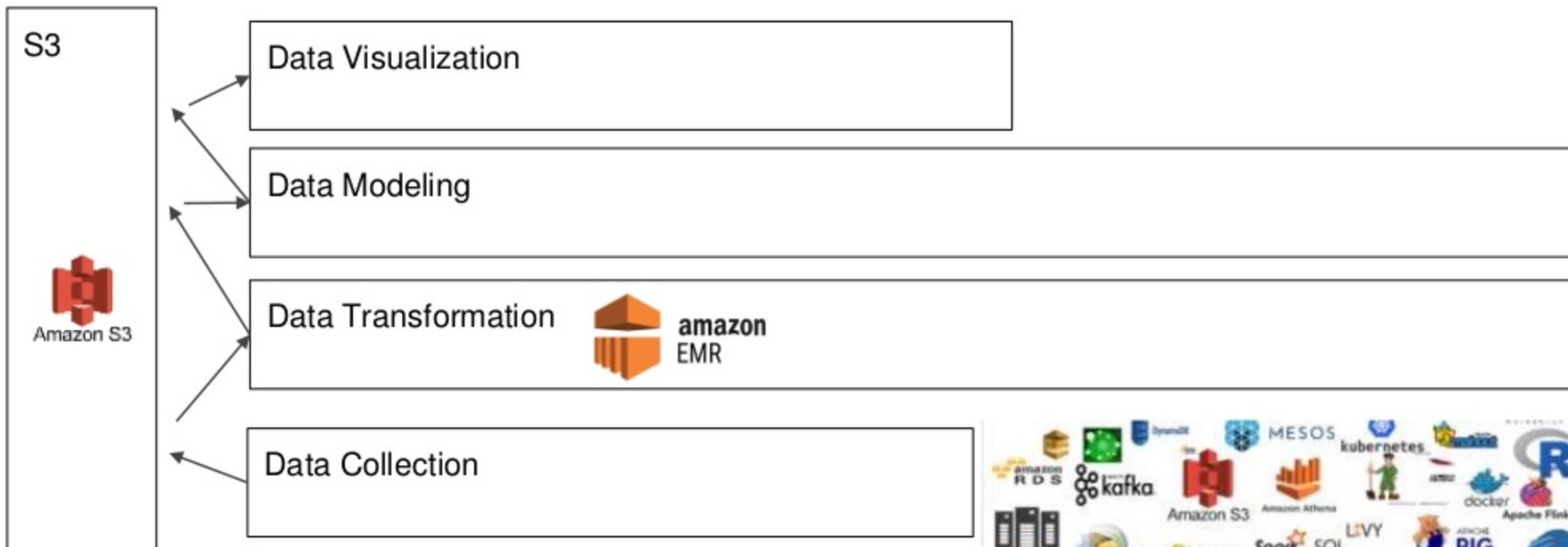
- Streams
 - Sub 1 sec processing latency
 - Choice of stream processor (generic)
 - For smaller events
- Firehose
 - Zero admin
 - 4 targets built in (redshift, s3, search, etc)
 - Buffering 60 sec minimum.
 - For larger “events”



Big Data Generic Architecture | Data Collection



Big Data Generic Architecture | Transformation





EMR ecosystem

- Hive
- Pig
- Hue
- Spark
- Oozie
- Presto
- Ganglia
- Zookeeper (hbase)
- zeppelin



amazon EMR



EMR Architecture



- Master node
- Core nodes - like data nodes (with storage: HDFS)
- Task nodes - (extends compute)
- **Does Not have Standby Master node**
- Best for transient cluster (goes up and down every night)





EMR lesson learned...

- **Bigger** instance type is good architecture
- Use **spot** instances - for the tasks.
- Don't always use **TEZ (MR? Spark?)**
- Make sure you choose instance with **network optimized**
- **Resize** cluster is not recommended
- **Bootstrap** to automate cluster upon provisioning
- Use **Steps** to automate steps on running cluster
- Use **Glue** to share Hive MetaStore



amazon
EMR



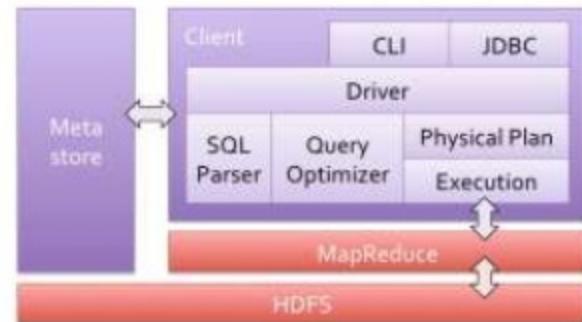
So use EMR for ...



- Most dominant
 - Hive
 - Spark
 - Presto
- And many more....
- Good for:
 - Data transformation
 - Data modeling
 - Batch
 - Machine learning



Hive Architecture





Hive

- SQL over hadoop.
- Engine: spark, tez, MR
- JDBC / ODBC
- Not good when need to **shuffle**.
- **Not peta scale.**
- SerDe json, parquet,regex,text etc.
- Dynamic partitions
- Insert overwrite
- Data Transformation
- Convert to Columnar



Presto

- SQL over hadoop
- Not good always for join on 2 large tables.
- Limited by memory
- Not fault tolerant like hive.
- Optimized for ad hoc queries
- No insert overwrite
- No dynamic partitions.
- Has some connectors : redshift and more
- <https://amazon-aws-big-data-demystified.ninja/2018/07/02/aws-emr-presto-demystified-everything-you-wanted-to-know-about-presto/>



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.**



Hue

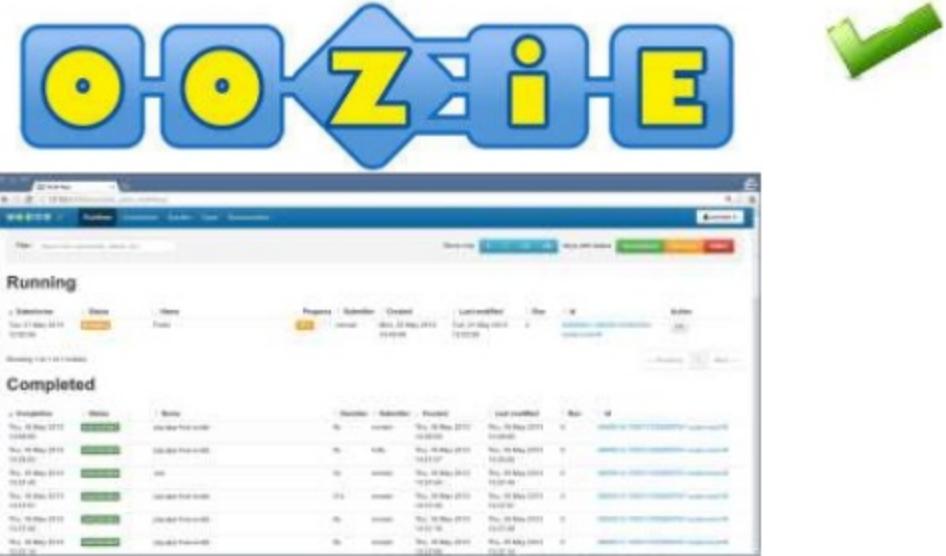


- Hadoop user experience
- Logs in real time and failures.
- Multiple users
- Native access to S3.
- File browser to HDFS.
- Manipulate metascore
- Job Browser
- **Query editor**
- Hbase browser
- Sqoop editor, oozier editor, Pig Editor

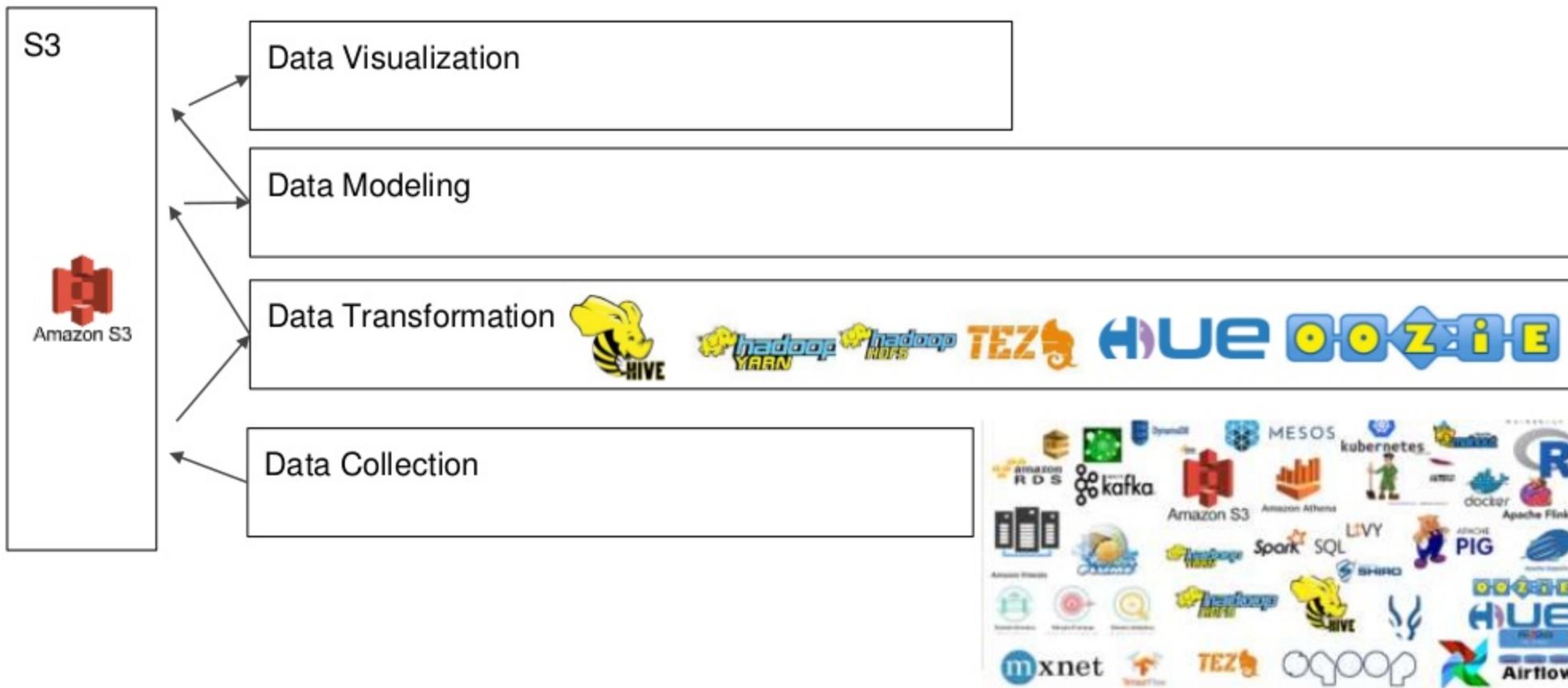


Orchestration

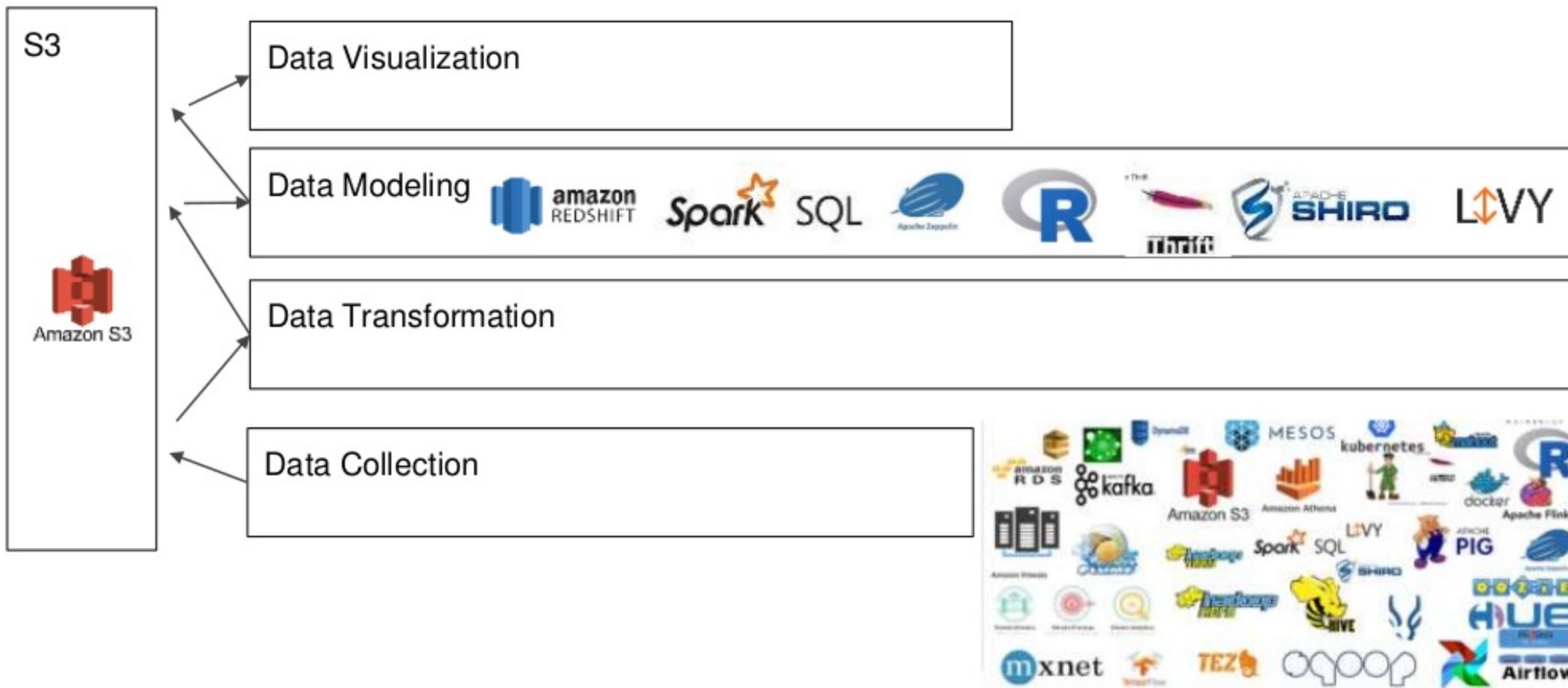
- EMR Oozie
 - Opens source workflow
 - Workflow: graph of action
 - Coordinator: scheduler jobs
 - Support: hive, sqoop , spark etc.
- Other: AirFlow, Knime, Luigi, Azkaban,AWS Data Pipeline



Big Data Generic Architecture | Transformation



Big Data Generic Architecture | Modeling



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



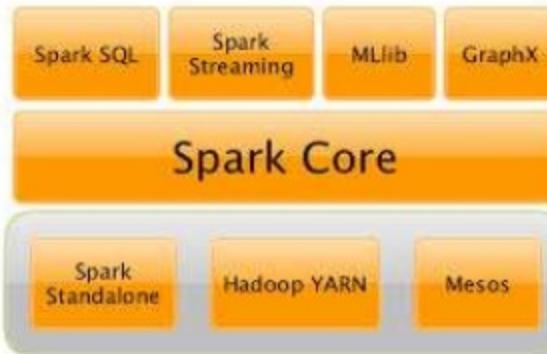
Spark Streaming

- Near real time (1 sec latency)
- like batch of 1sec windows
- Streaming jobs with API
- **Not relevant to us...**



Spark ML

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

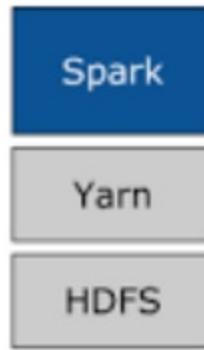


Spark flavours

- Standalone
- With yarn
- With mesos



(a) Standalone



(b) Over Yarn



(c) Spark in
MR (SIMR)





Spark Downside

- Compute intensive
- Performance gain over mapreduce is not guaranteed.
- Streaming processing is actually batch with very small window.
- **Different behaviour between hive and spark SQL**



Spark SQL

- Same syntax as hive
- Optional JDBC via thrift
- Non trivial learning curve
- Up to X10 faster than hive.
- Works well with Zeppelin (out of the box)
- Does not replace Hive
- Spark not always faster than hive
- Insert overwrite -



Apache Zeppelin

- Notebook - visualizer
- Built in spark integration
- Interactive data analytics
- Easy collaboration.
- Uses SQL
- works on top of Hive/ SparkSQL
- Inside EMR.
- Uses in the background:
 - Shiro
 - Livy

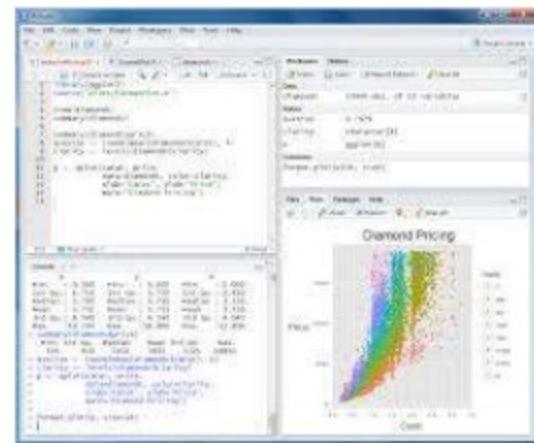


Apache Zeppelin



R + spark 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.



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.
- Good for : DW, Complex Joins.



Redshift spectrum

- Extension of Redshift, use external table on S3.
- Require redshift cluster.
- Not possible for CTAS to s3, complex data structure, joins.
- Good for
 - **Read only** Queries
 - Aggregations on Exabyte.



EMR vs Redshift

- How much data loaded and unloaded?
- Which operations need to 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. (aggregations)?
- Raw data? s3.

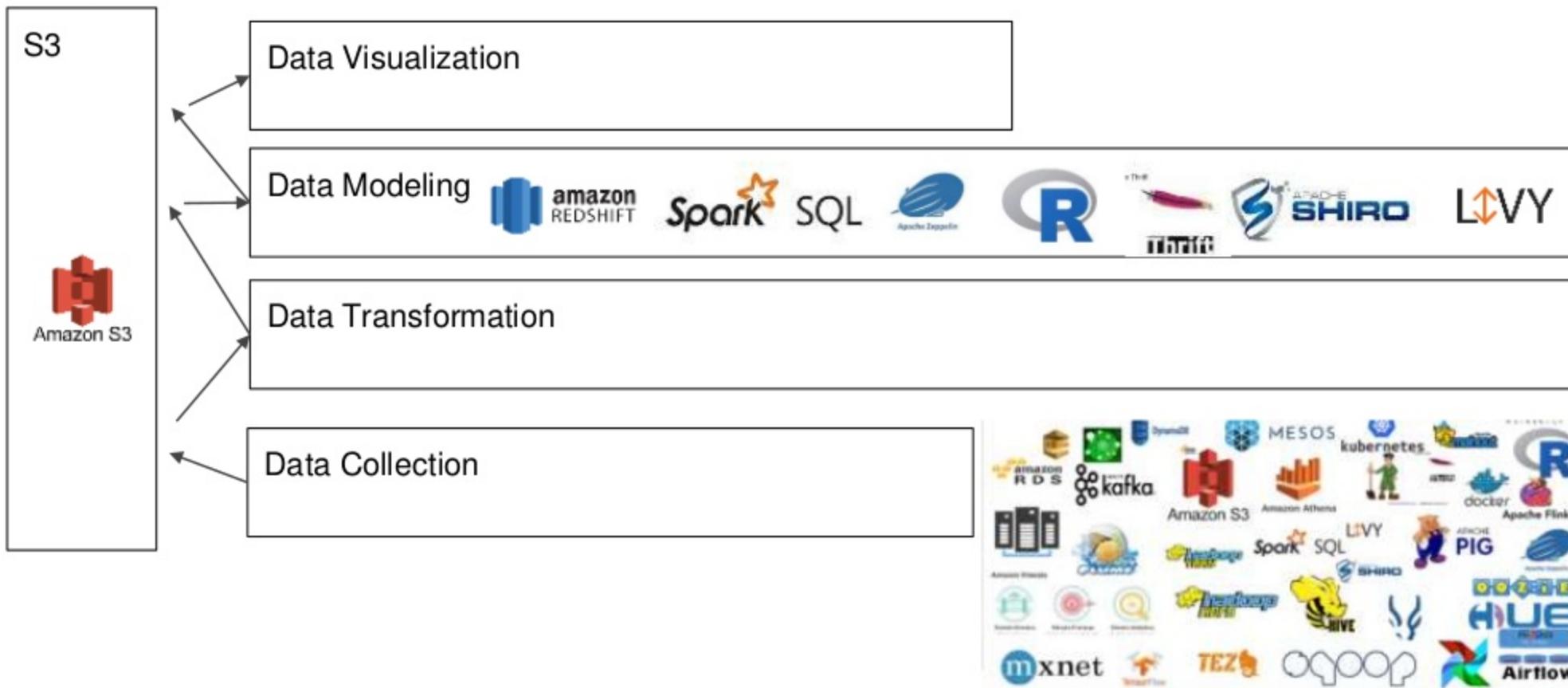


Hive VS. Redshift

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



Big Data Generic Architecture | Modeling

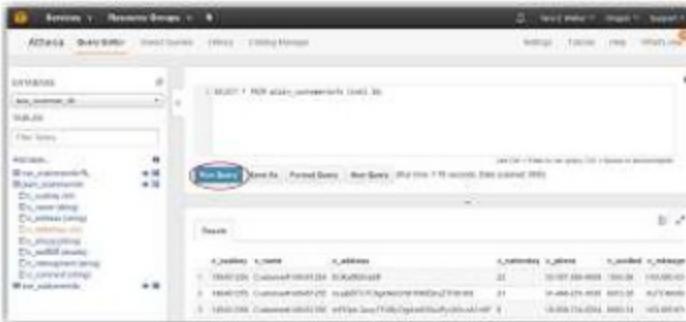


Big Data Generic Architecture | Visualize



Athena

- Presto SQL
- In memory
- Hive metastore for DDL functionality
 - Complex data types
 - Multiple formats
 - Partitions
- Good for:
 - Read only SQL,
 - Ad hoc query,
 - low cost,
 - managed



Amazon Athena



Visualize

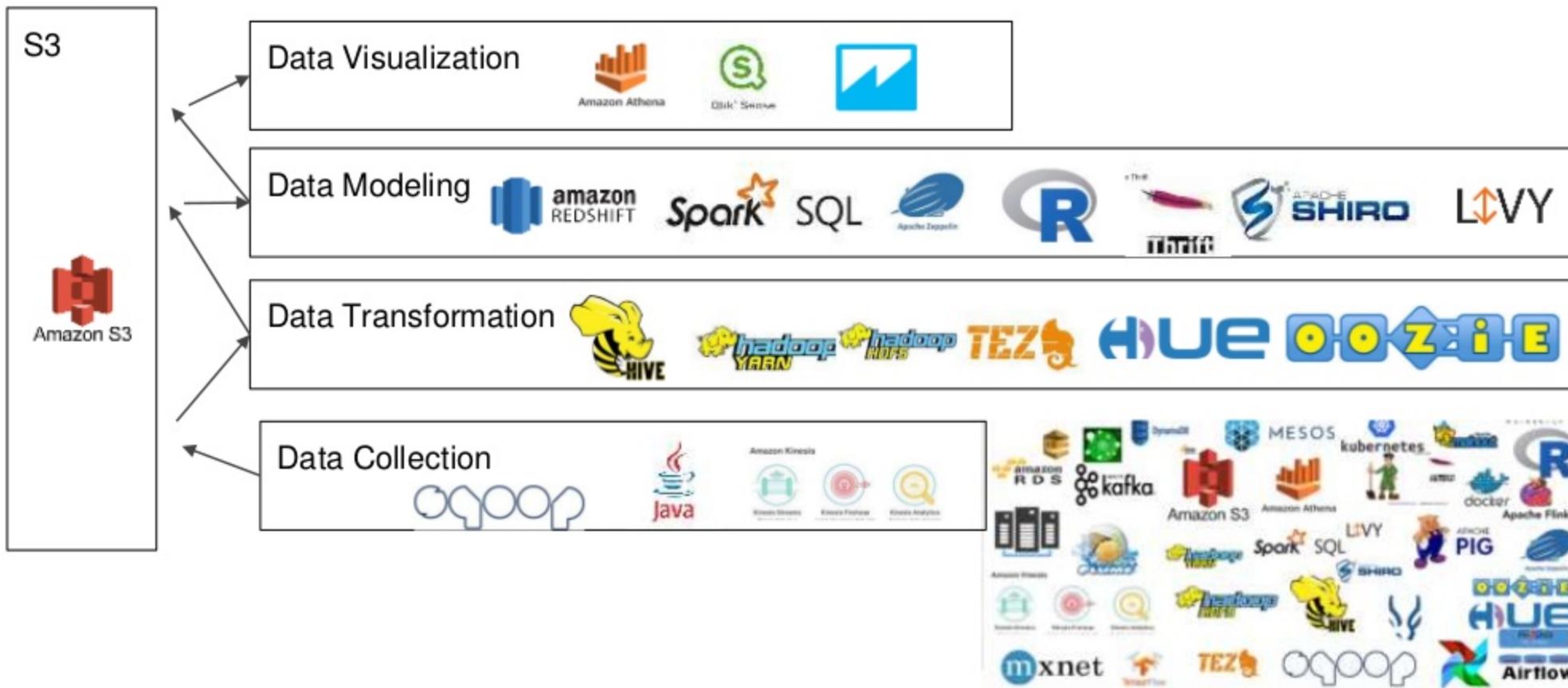
- QuickSight
- Managed Visualizer, simple, cheap



Qlik Sense®



Big Data Generic Architecture | Summary



Summary: Lesson learned

- Productivity of Data Science and Data engineering
 - Common language of both teams IS SQL!
 - Spark cluster has many bridges: SparkR, Spark ML, SparkSQL , Spark core.
- **Minimize** the amount DB's used
 - Different **syntax** (presto/hive/redshift)
 - Different **data types**
 - **Minimize ETLs via External Tables+Glue!**
- Not always **Streaming** is justified (what is the business use case? PaaS?)
- **Spark SQL**
 - Sometimes **faster** than **redshift**
 - Sometimes **slower** than **hive**
 - Learning curve is **non trivial**
- Smart Big Data Architecture is all about:
 - **Faster, Cheaper, Simpler, More Secured.**



Stay in touch...

- [Omid Vahdaty](#) 
- +972-54-2384178
- <https://amazon-aws-big-data-demystified.ninja/>
- Join our meetup, FB group and youtube channel
 - <https://www.meetup.com/AWS-Big-Data-Demystified/>
 - <https://www.facebook.com/groups/amazon.aws.big.data.demystified/>
 - https://www.youtube.com/channel/UCzeGqhZIWU-hIDczWa8GtgQ?view_as=subscriber

