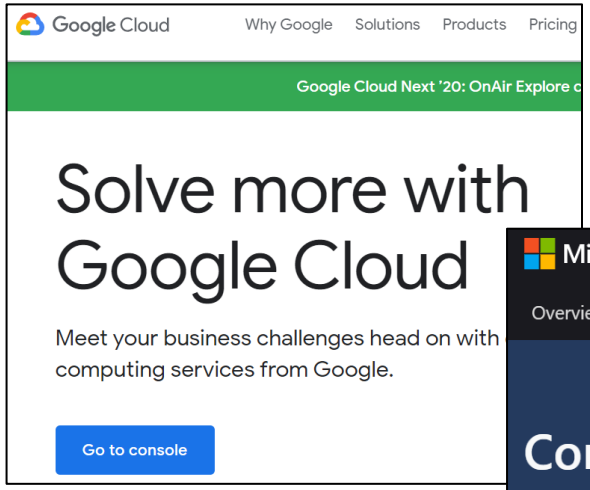
Big Data
Products and Practices

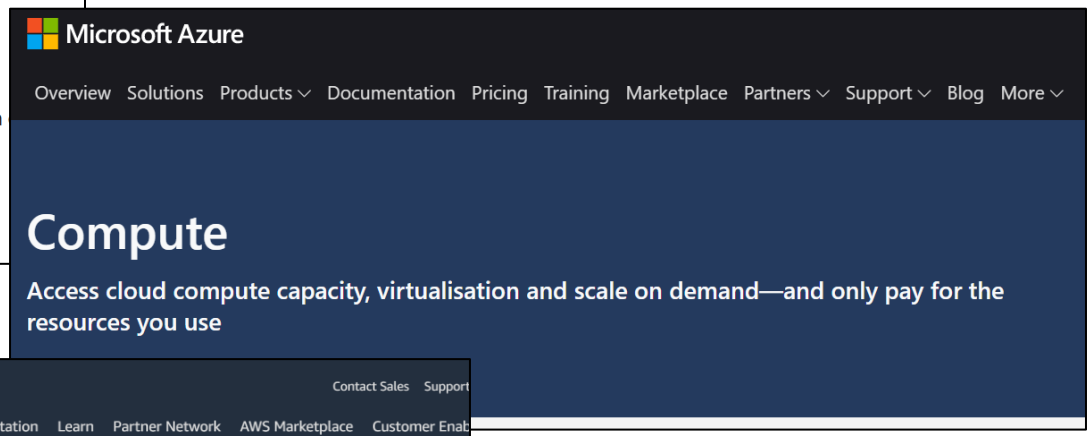
Venkatesh Vinayakarao
venkateshv@cmi.ac.in
http://vvtesh.co.in

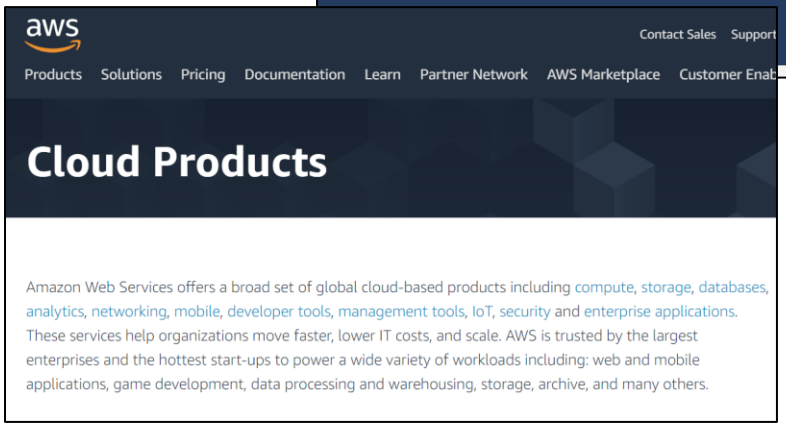
Cloud Platforms

Cloud Services:

- **SaaS**
- **PaaS**
- **IaaS**



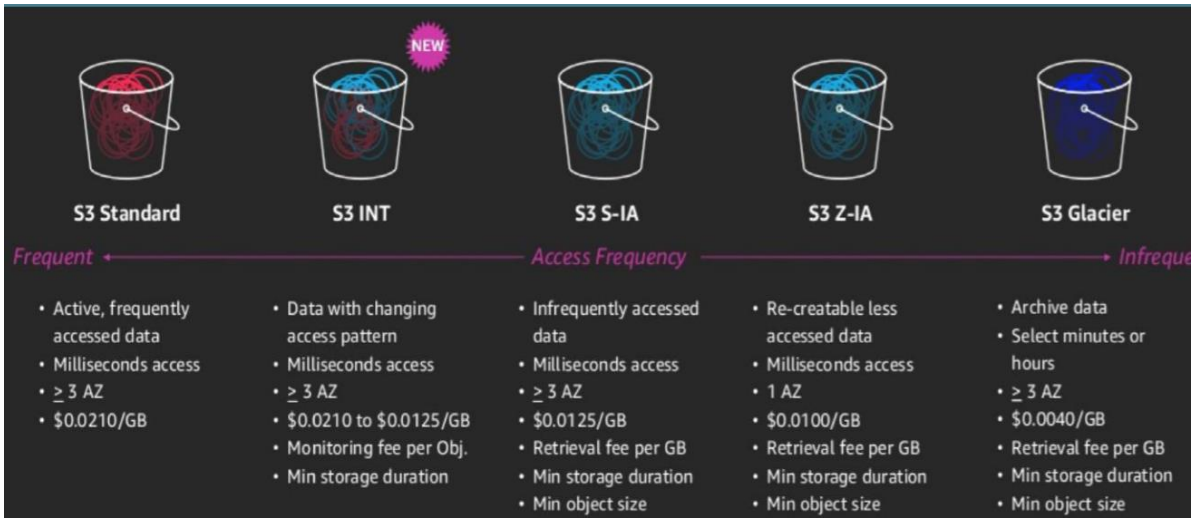




Cloud Platforms

Ref: https://maelfabien.github.io/bigdata/gcps_1/#what-is-gcp

Storage Service (Amazon S3 Example)



Compute Services (Google Cloud Example)

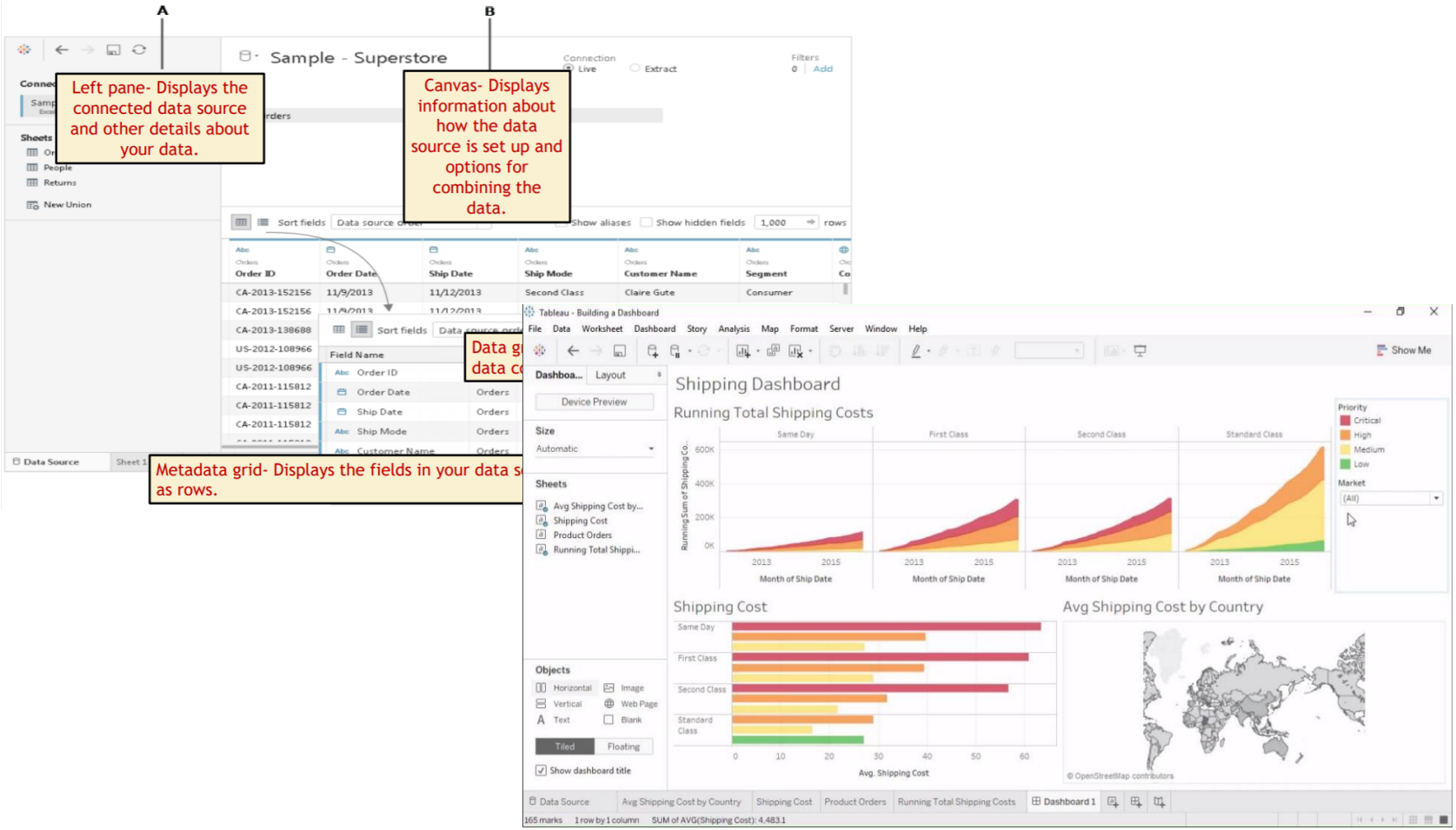
Network Services (Azure Example)

- Azure Traffic Manager is a DNS-based traffic load balancer that distributes traffic optimally to services across global Azure regions, while providing high availability.
- Traffic Manager directs client requests to the most appropriate service endpoint.

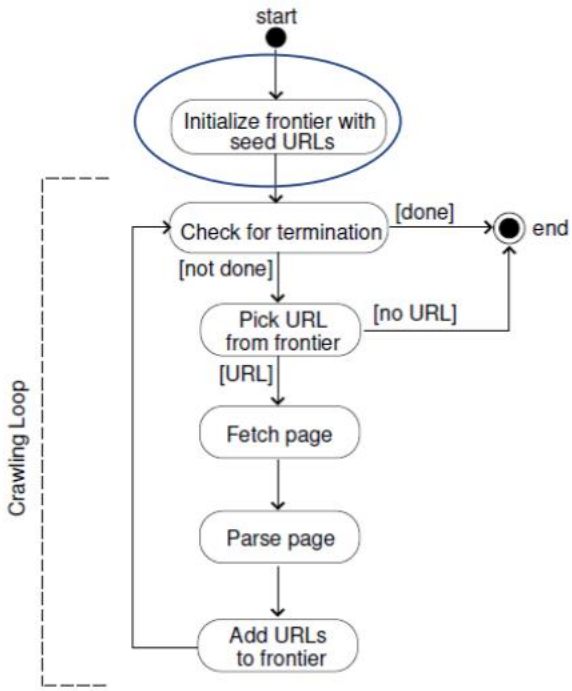
Building Great Apps/Services

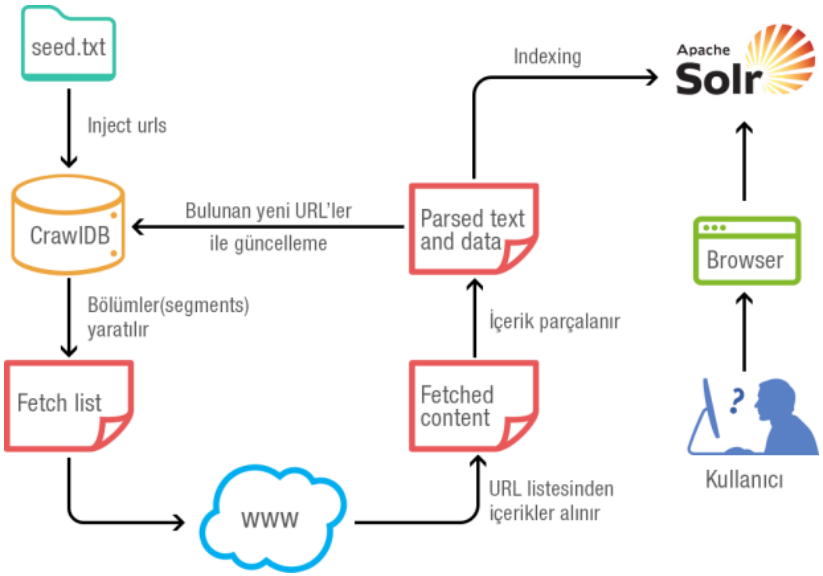
- We need
  - Products that make certain features easy to implement
    - Visualization
    - Crawling/Search
    - Log Aggregation
    - Graph DB
    - Synchronization

Tableau


A
B
Sample - Superstore
Left pane- Displays the connected data source and other details about your data.
Canvas- Displays information about how the data source is set up and options for combining the data.
Metadata grid- Displays the fields in your data s
as rows.
Tableau - Building a Dashboard
Shipping Dashboard
Running Total Shipping Costs
Shipping Cost
Avg Shipping Cost by Country

Crawling with Nutch





Solr Integration Image Src: https://suyashaoc.wordpress.com/2016/12/04/nutch-2-3-1-hbase-0-98-8-hadoop-2-5-2-solr-4-1-web-crawling-and-indexing/

Log Files are an Important Source of Big Data

Log4j

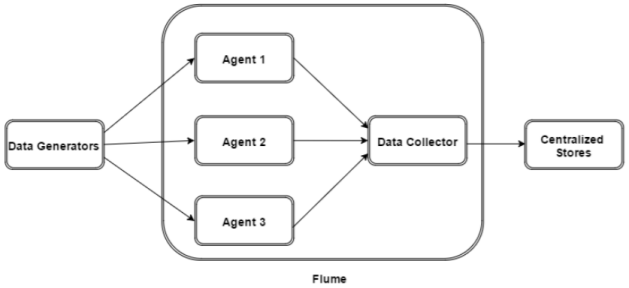
log4j.properties Syntax

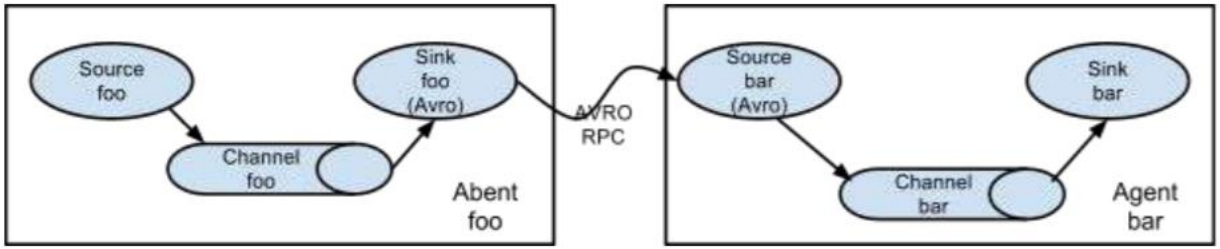
Following is the syntax of *log4j.properties* file for an appender X:

```
# Define the root logger with appender X
log4j.rootLogger = DEBUG, X

# Set the appender named X to be a File appender
log4j.appender.X=org.apache.log4j.FileAppender

# Define the layout for X appender
log4j.appender.X.layout=org.apache.log4j.PatternLayout
log4j.appender.X.layout.conversionPattern=%m%n
```

```java
package test;

import org.apache.log4j.Logger;

public class Log4JTest {
    static Logger log = Logger.getLogger(Log4JTest.class);

    public static void main(String[] args) {

        log.debug("This is a debug message");
    }
}
```

Flume



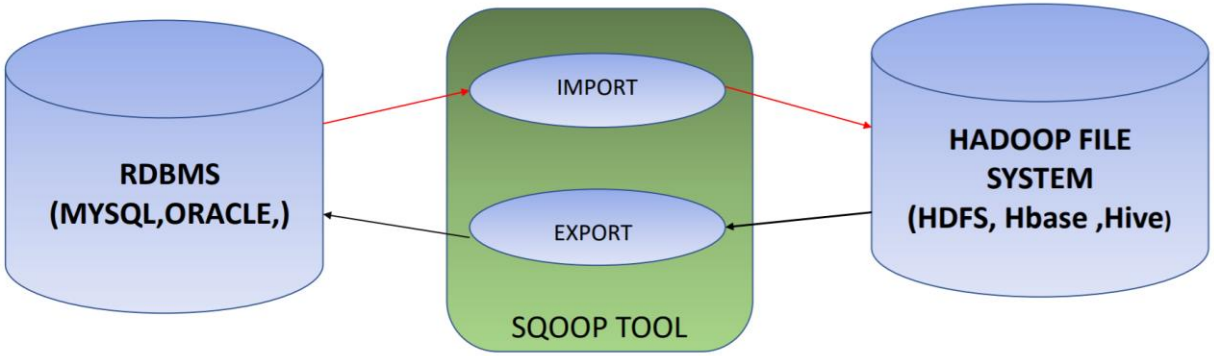


Flume event

```
usingFlume.sources = usingFlumeSource
usingFlume.channels = memory

usingFlume.sources.usingFlumeSource.type = avro
usingFlume.sources.usingFlumeSource.channels = memory
usingFlume.sources.usingFlumeSource.port = 7877
usingFlume.sources.usingFlumeSource.bind = 0.0.0.0
```

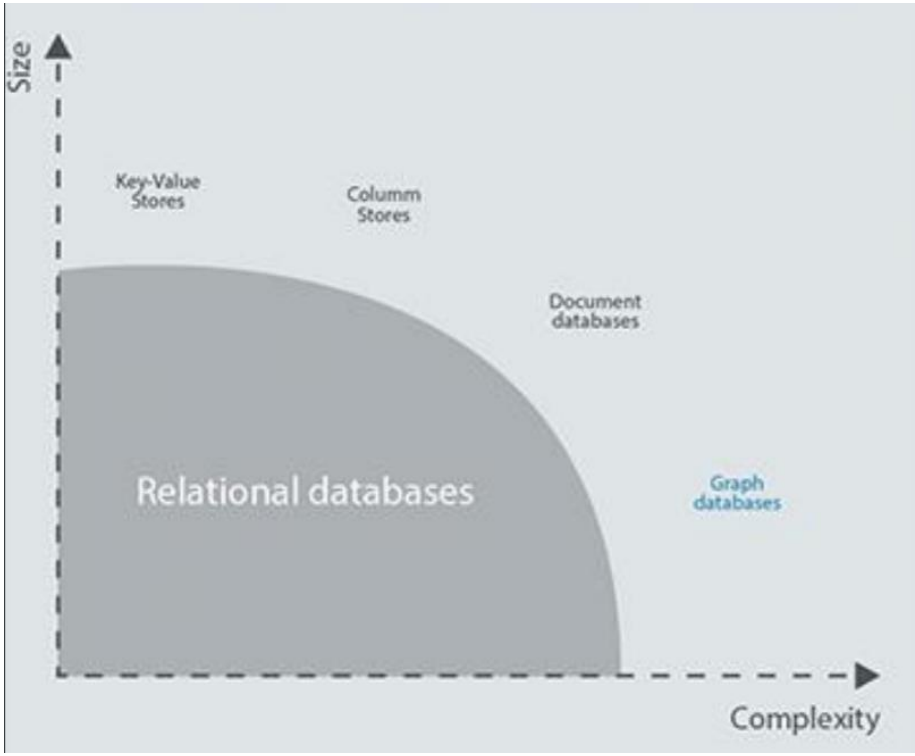
Flume Config Files

Sqoop



Designed for efficiently transferring bulk data between Hadoop and RDBMS

GraphDB – Neo4j



ACID compliant graph database management system

neo4j	Cypher	Most famous graph database, Cypher O(1) access using fixed-size array
DSE Graph DATASTAX AURELIUS	Gremlin	Distributed graph system based on Cassandra
ArangoDB	AQL	Multi-model database (Document + Graph)
OrientDB	OQL	Multi-model database (Document + Graph)

Neo4j

- A leading graph database, with native graph storage and processing.
- Open Source
- NoSQL
- ACID compliant

Neo4j Sandbox

https://sandbox.neo4j.com/

Neo4j Desktop

https://neo4j.com/download

Data Model

- create (p:Person {name:'Venkatesh'})-[:Teaches]->(c:Course {name:'BigData'})

Query Language

- Cypher Query Language
  - Similar to SQL
  - Optimized for graphs
  - Used by Neo4j, SAP HANA Graph, Redis Graph, etc.

CQL

- create (p:Person {name:'Venkatesh'})-[:Teaches]->(c:Course {name:'BigData'})
- Don’t forget the single quotes.

```
neo4j$ create (p:Person {name:'Venkatesh'})-[:Teaches]→(c:Course {name:'BigData'})
```

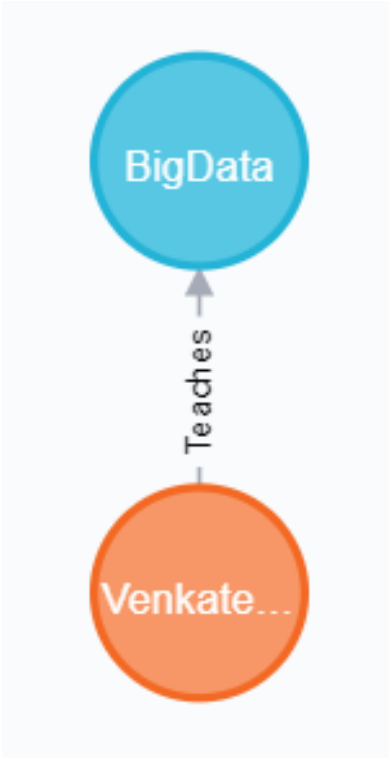
Table

Code

Added 2 labels, created 2 nodes, set 2 properties, created 1 relationship, completed after 30 ms.

CQL

- Match (n) return n



CQL

- match(p:Person {name:'Venkatesh'}) set p.surname='Vinayakarao' return p

```
neo4j$ match(p:Person {name:'Venkatesh'}) set p.surname='Vinayakarao' return p
```

Graph

Table

Text

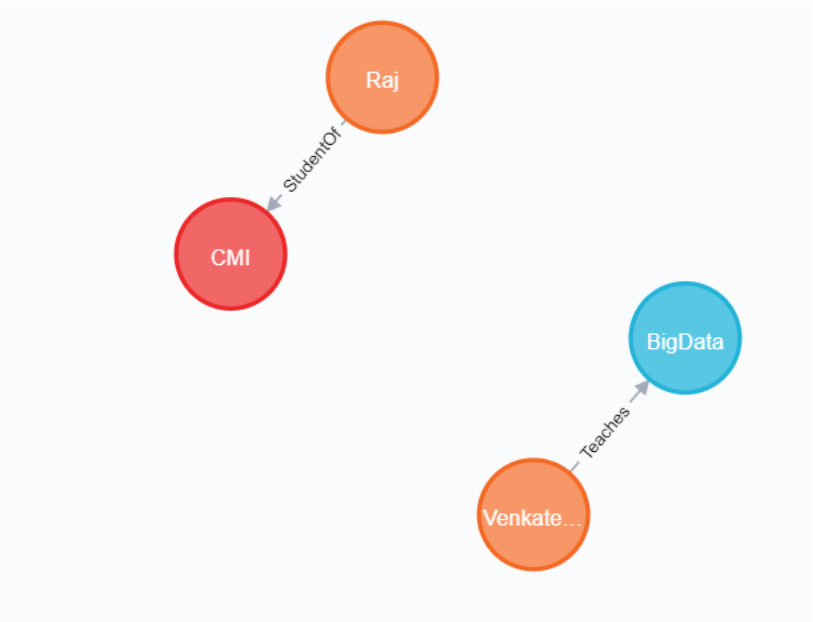
Code

p

```
{
  "name": "Venkatesh",
  "surname": "Vinayakarao"
}
```

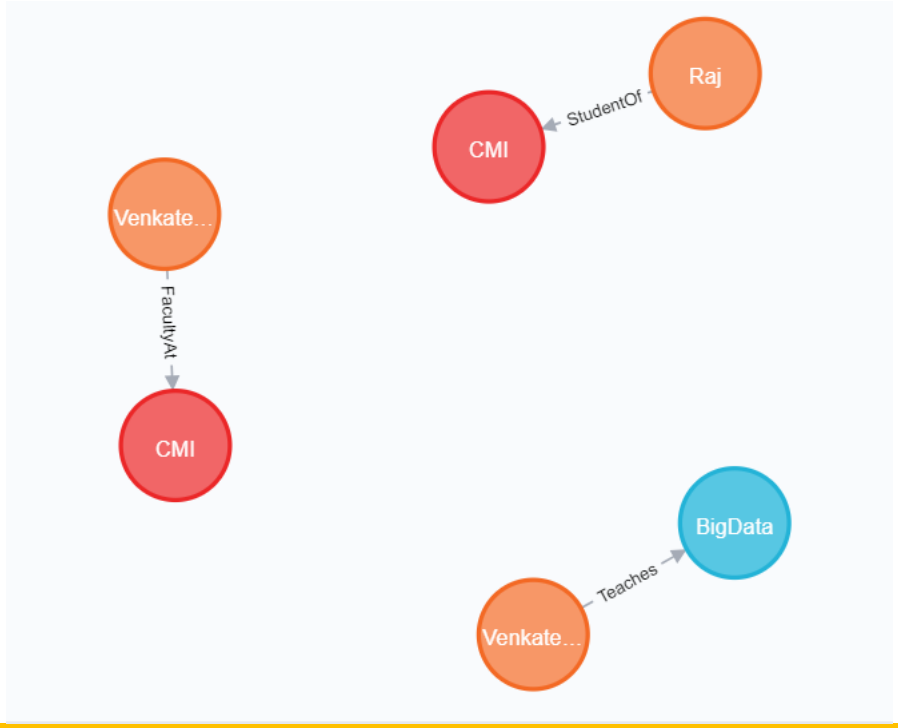
CQL

- Create (p:Person {name:’Raj’})-[:StudentOf]->(o:Org {name:’CMI’})
- Match (n) return n



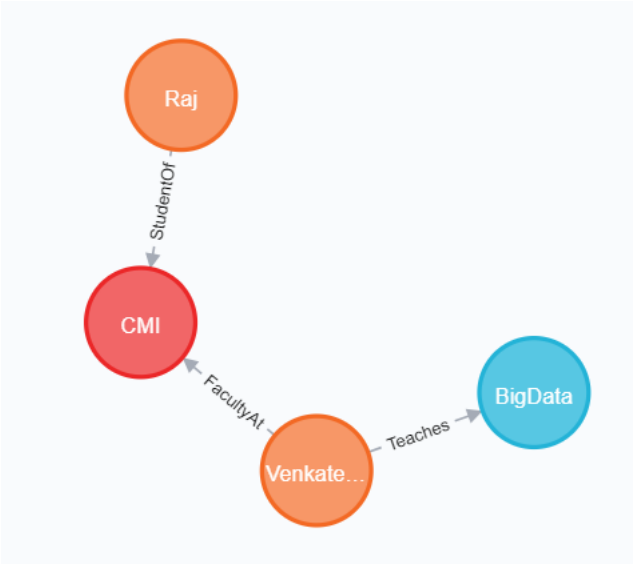
CQL

- create (p:Person {name:'Venkatesh'})-[:FacultyAt]->(o:Org {name:'CMI’})
- Match (n) return n



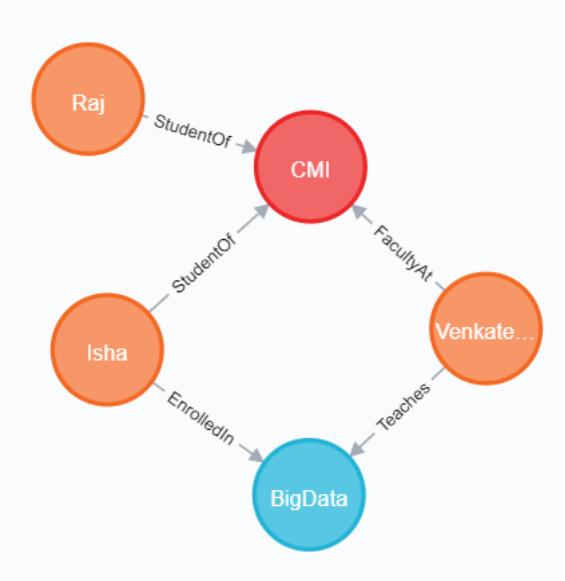
CQL

- MATCH (p:Person {name:'Venkatesh'})-[r:FacultyAt]->()
- DELETE r
- MATCH (p:Person) where ID(p)=4
- DELETE p
- MATCH (o:Org) where ID(o)=5
- DELETE o



- MATCH (a:Person),(b:Org)
- WHERE a.name = 'Venkatesh' AND b.name = 'CMI'
- CREATE (a)-[:FacultyAt]->(b)

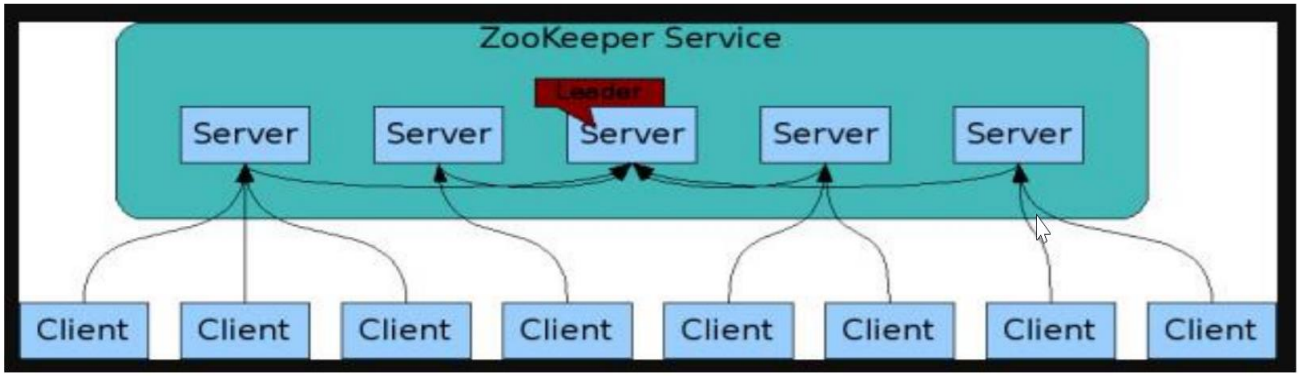
CQL




```
create (p:Person {name:'Isha'})

MATCH (a:Person),(b:Course)
WHERE a.name = 'Isha' and b.name = 'BigData'
CREATE (a)-[:StudentOf]->(b)

MATCH (a:Person)-[o:StudentOf]->(b:Course) where a.name = 'Isha' DELETE o

MATCH (a:Person),(b:Org)
WHERE a.name = 'Isha' and b.name = 'CMI'
CREATE (a)-[:StudentOf]->(b)

MATCH (a:Person),(b:Course)
WHERE a.name = 'Isha' and b.name = 'BigData'
CREATE (a)-[:EnrolledIn]->(b)
```

Apache ZooKeeper

A centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services

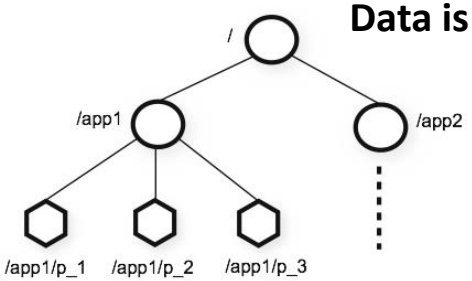


A Zookeeper Ensemble Serving Clients

It is simple to store data using zookeeper

```
$ create /zk_test my_data
$ set /zk_test junk
$ get /zk_test
junk
$ delete /zk_test
```

Data is stored hierarchically



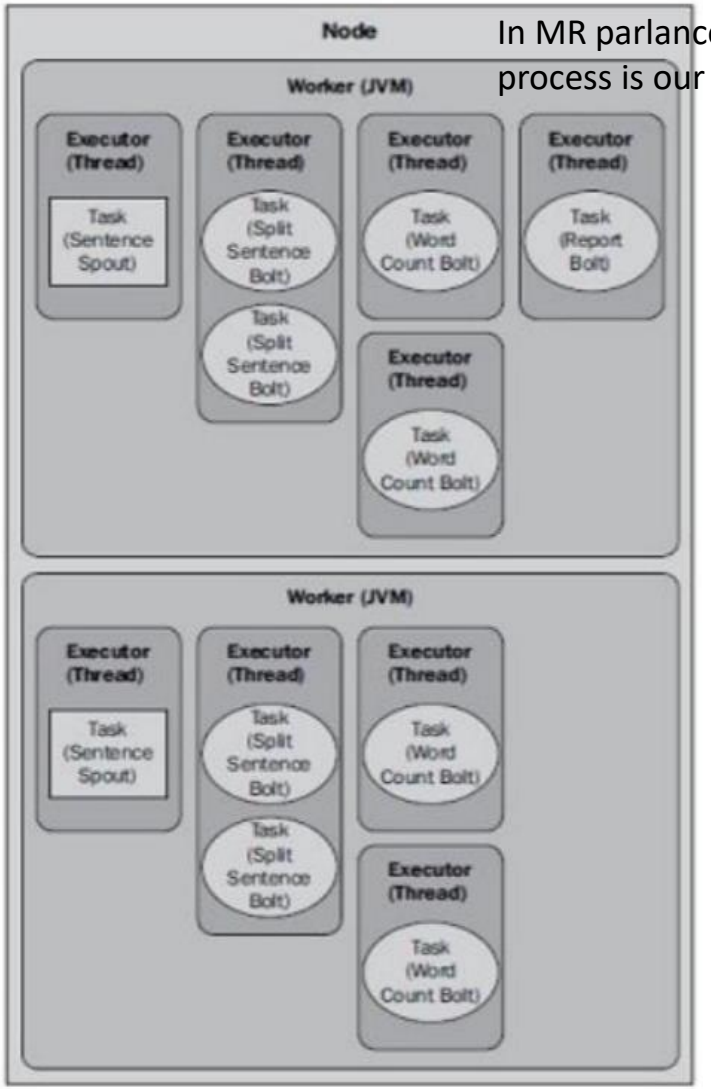
Stream Processing

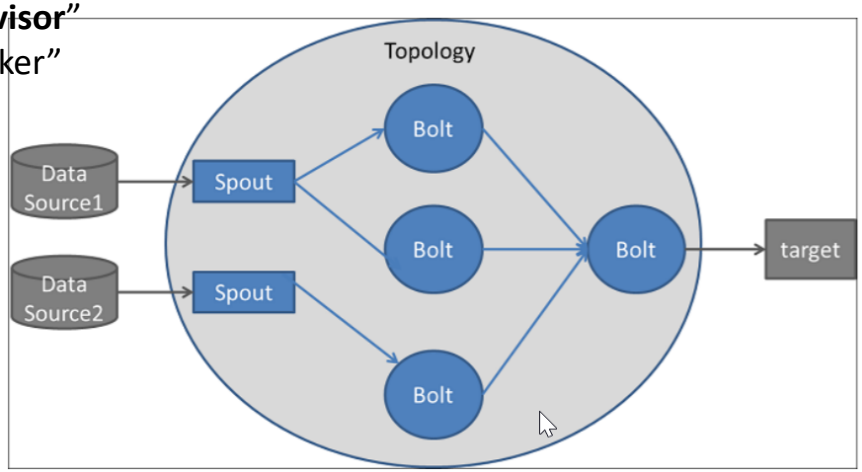
- Process data as they arrive.

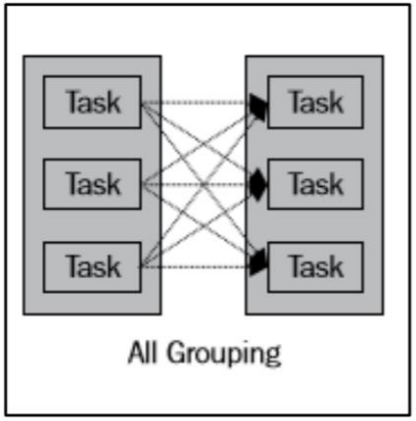
Stream Processing with Storm

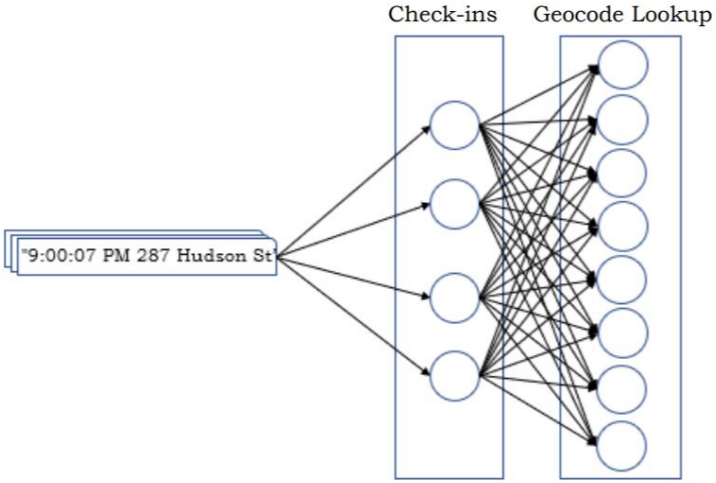
One of these is a master node. “**Nimbus**” is the “job tracker”!

In MR parlance, “**Supervisor**” process is our “task tracker”



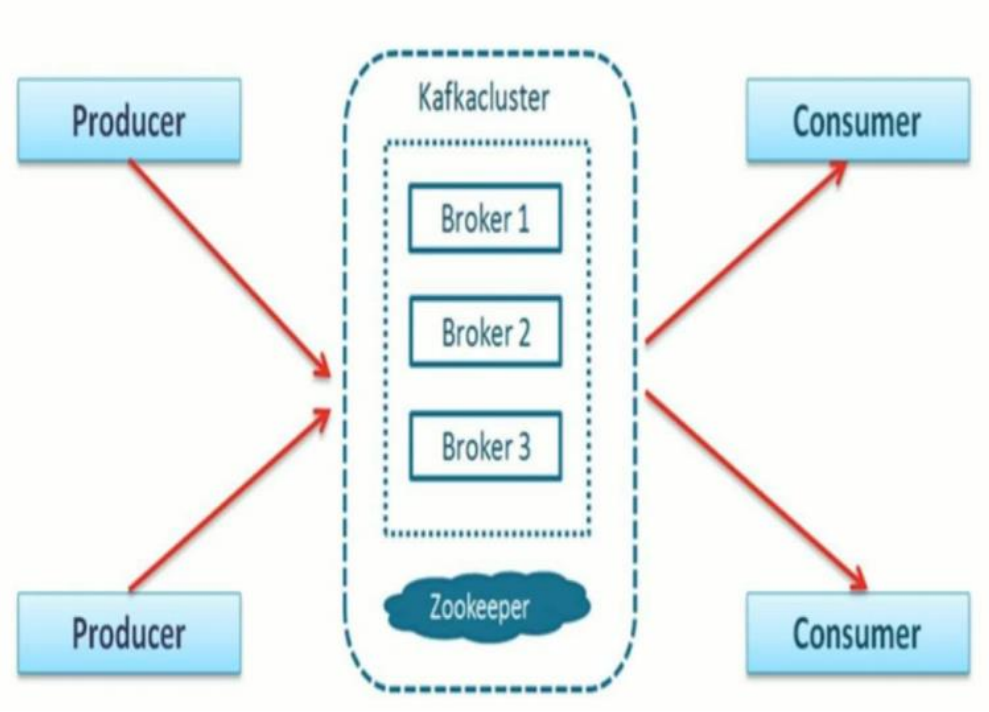






Apache Kafka

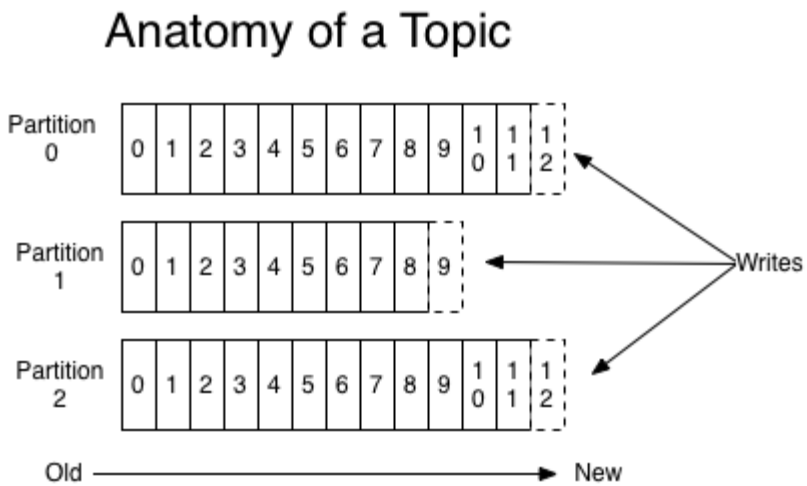
- Uses Publish-Subscribe Mechanism



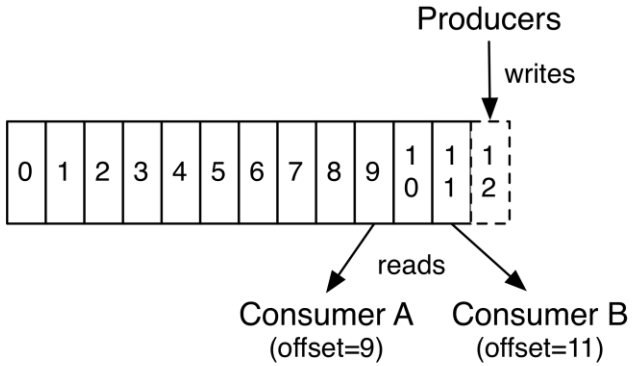
Kafka – Tutorial (Single Node)

- Create a topic
  - > bin/kafka-topics.sh ---topic test
- List all topics
  - > bin/kafka-topics.sh –list
  - > test
- Send messages
  - > bin/kafka-console-producer.sh --topic test
  - This is a message
  - This is another message
- Receive messages (subscribed to a topic)
  - > bin/kafka-console-consumer.sh --topic test --from-beginning
  - This is a message
  - This is another message

Kafka – Multi-node



- topic is a stream of records.
- for each topic, the Kafka cluster maintains a partitioned log
- records in the partitions are each assigned a sequential id number called the *offset*



Kafka Brokers

- For Kafka, a single broker is just a cluster of size one.
- We can setup multiple brokers
  - The broker.id property is the unique and permanent name of each node in the cluster.
    - > bin/kafka-server-start.sh config/server-1.properties &
    - > bin/kafka-server-start.sh config/server-2.properties &
  - Now we can create topics with replication factor
    - > bin/kafka-topics.sh --create --replication-factor 3 --partitions 1 --topic my-replicated-topic
    - > bin/kafka-topics.sh --describe --bootstrap-server localhost:9092 --topic my-replicated-topic
      - Topic: my-replicated-topic PartitionCount:1 ReplicationFactor:3
        - Partition: 0 Leader: 2 Replicas: 1,2,0

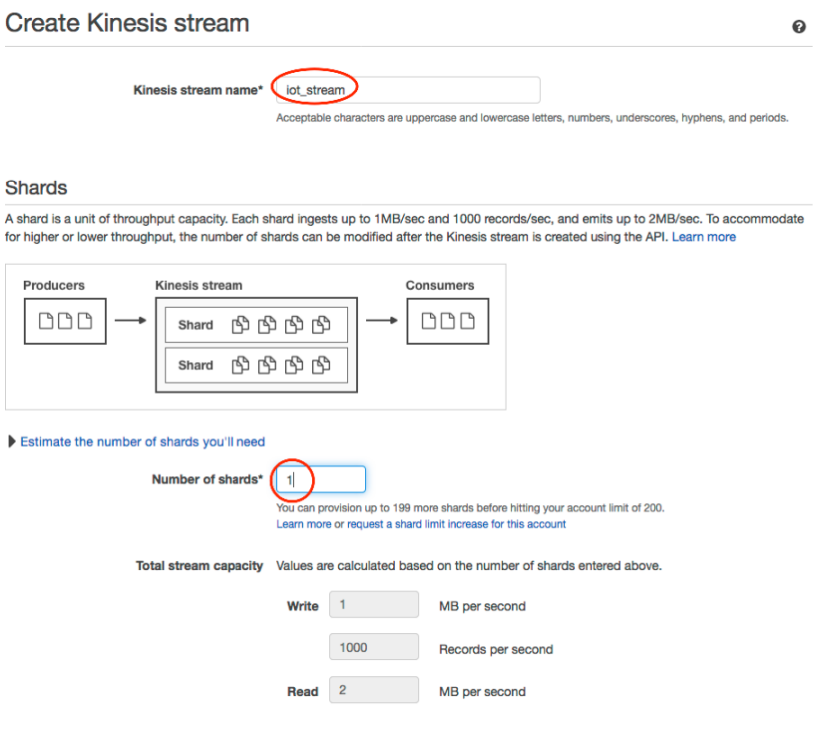
Streams API

```
StreamsBuilder builder = new StreamsBuilder();
KStream<String, String> textLines = builder.stream("TextLinesTopic");
KTable<String, Long> wordCounts = textLines
    .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
    .groupBy((key, word) -> word)
    .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("counts-store"));
wordCounts.toStream().to("WordsWithCountsTopic", Produced.with(Serdes.String(), Serdes.Long()));
```

Apache Kinesis

- Amazon Kinesis Data Streams is a **managed service** that scales elastically for real-time processing of streaming big data.

"Netflix uses Amazon Kinesis to monitor the communications between all of its applications so it can detect and fix issues quickly, ensuring high service uptime and availability to its customers." – Amazon (https://aws.amazon.com/kinesis/).

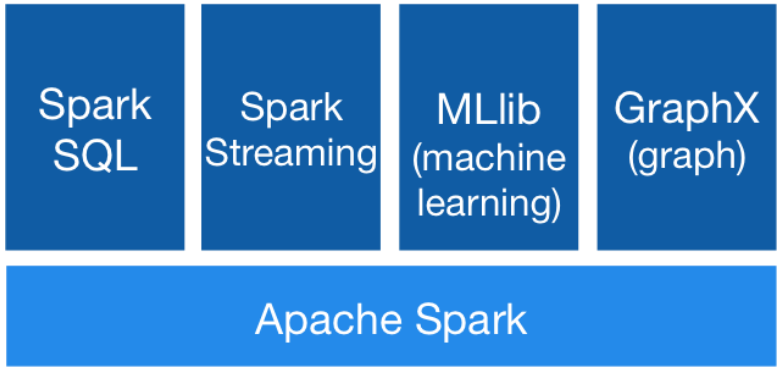


Amazon Kinesis capabilities

- Video Streams
- Data Streams
- Firehose
- Analytics

https://aws.amazon.com/kinesis/

Apache Spark (A Unified Library)



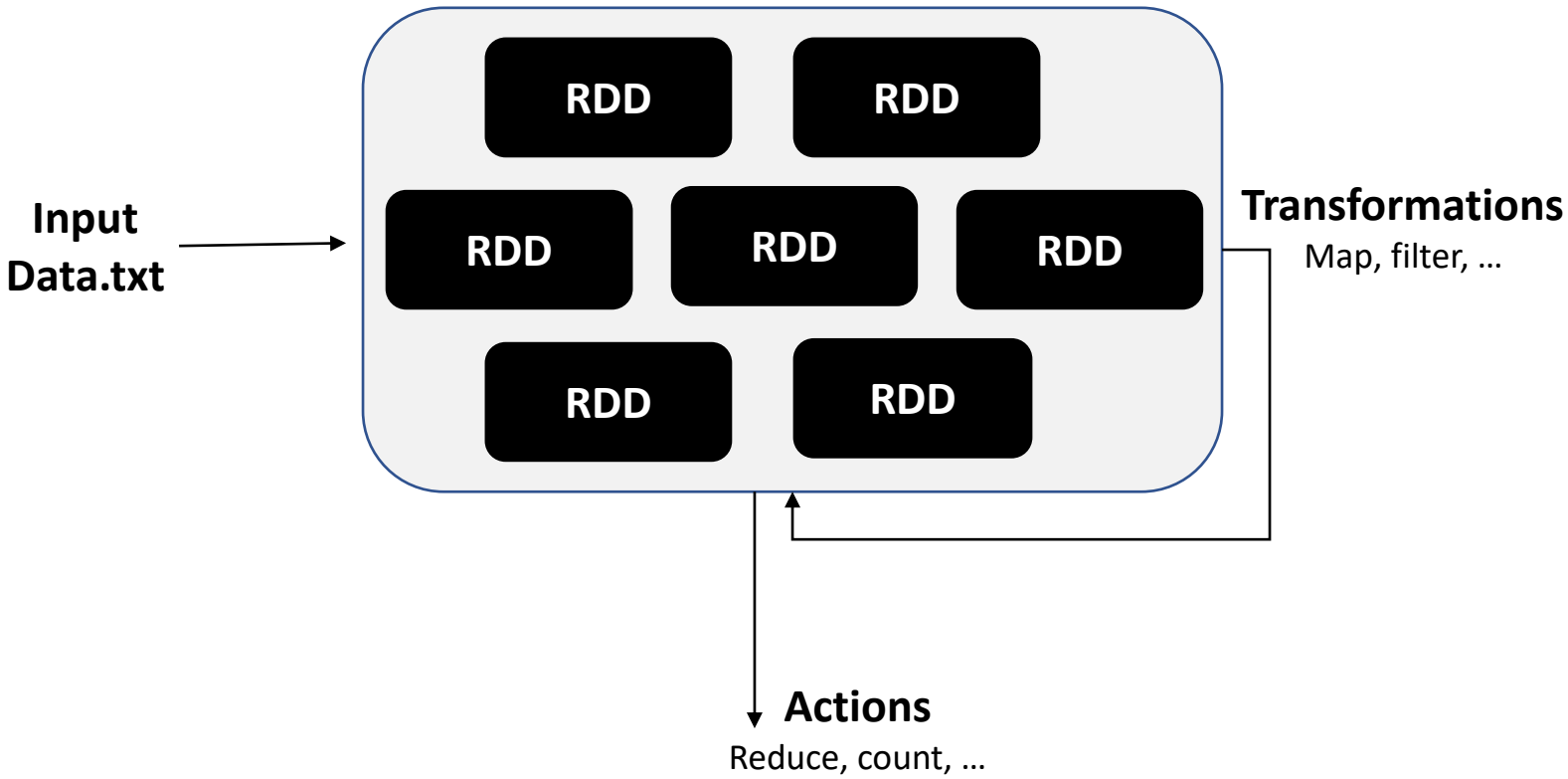

```
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

Spark's Python DataFrame API
Read JSON files with automatic schema inference

In spark, use data frames as tables

https://spark.apache.org/

Resilient Distributed Datasets (RDDs)



Spark Examples

```
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data)
```

distributed dataset can be used in parallel

```
"""MyScript.py"""
if __name__ == "__main__":
    def myFunc(s):
        words = s.split(" ")
        return len(words)

    sc = SparkContext(...)
    sc.textFile("file.txt").map(myFunc)
```

passing functions through spark

```
distFile = sc.textFile("dta.txt")
distFile.map(s => s.length).
            reduce((a, b) => a + b)
```

Map/reduce

Thank You