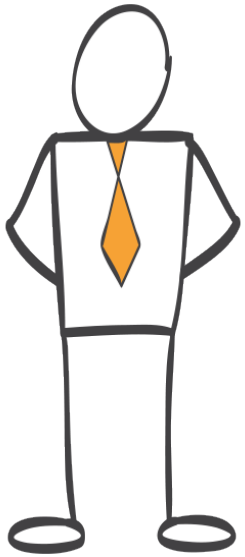
Building 1000 node Spark cluster on EMR

Manjeet Chayel

amazon web services

What is EMR?

Amazon Elastic MapReduce



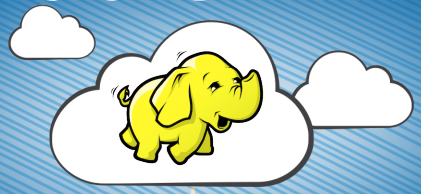
amazon web services

What is EMR?

Hadoop-as-a-service

Map-Reduce engine

Integrated with tools



Massively parallel

Integrated to AWS services

Cost effective AWS wrapper

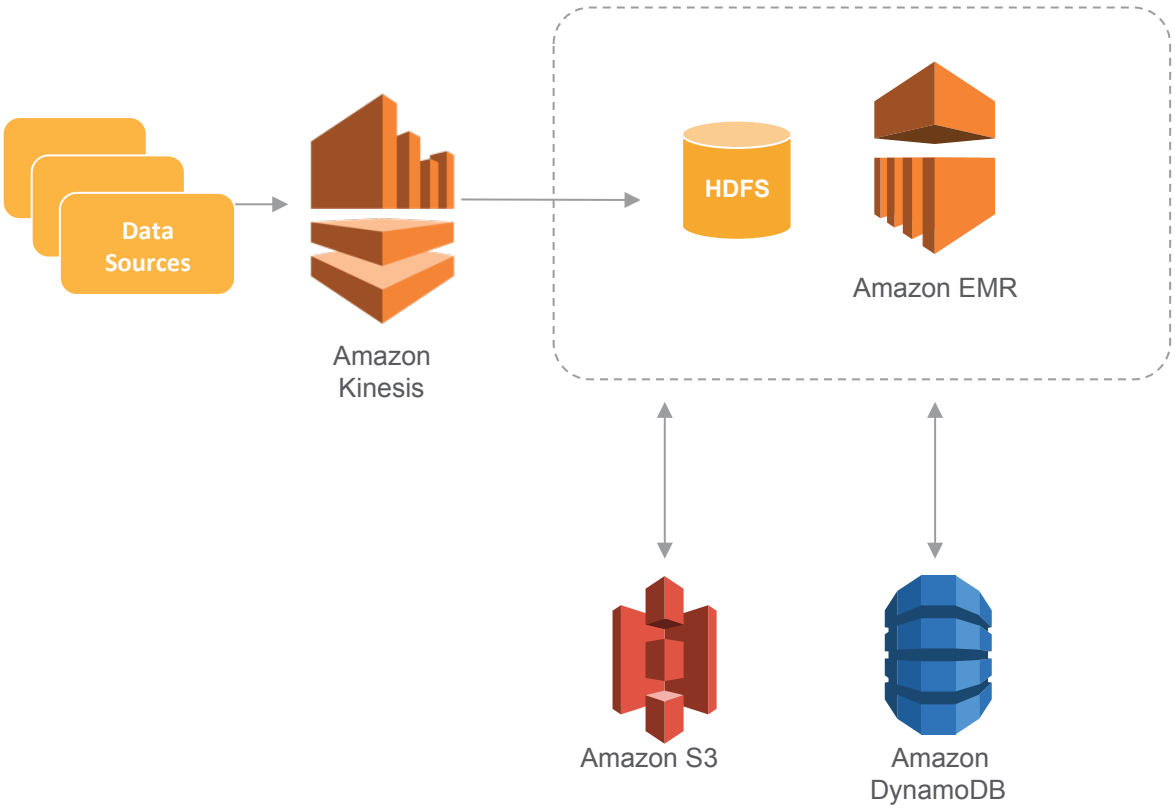
amazon web services

Integration

Integration

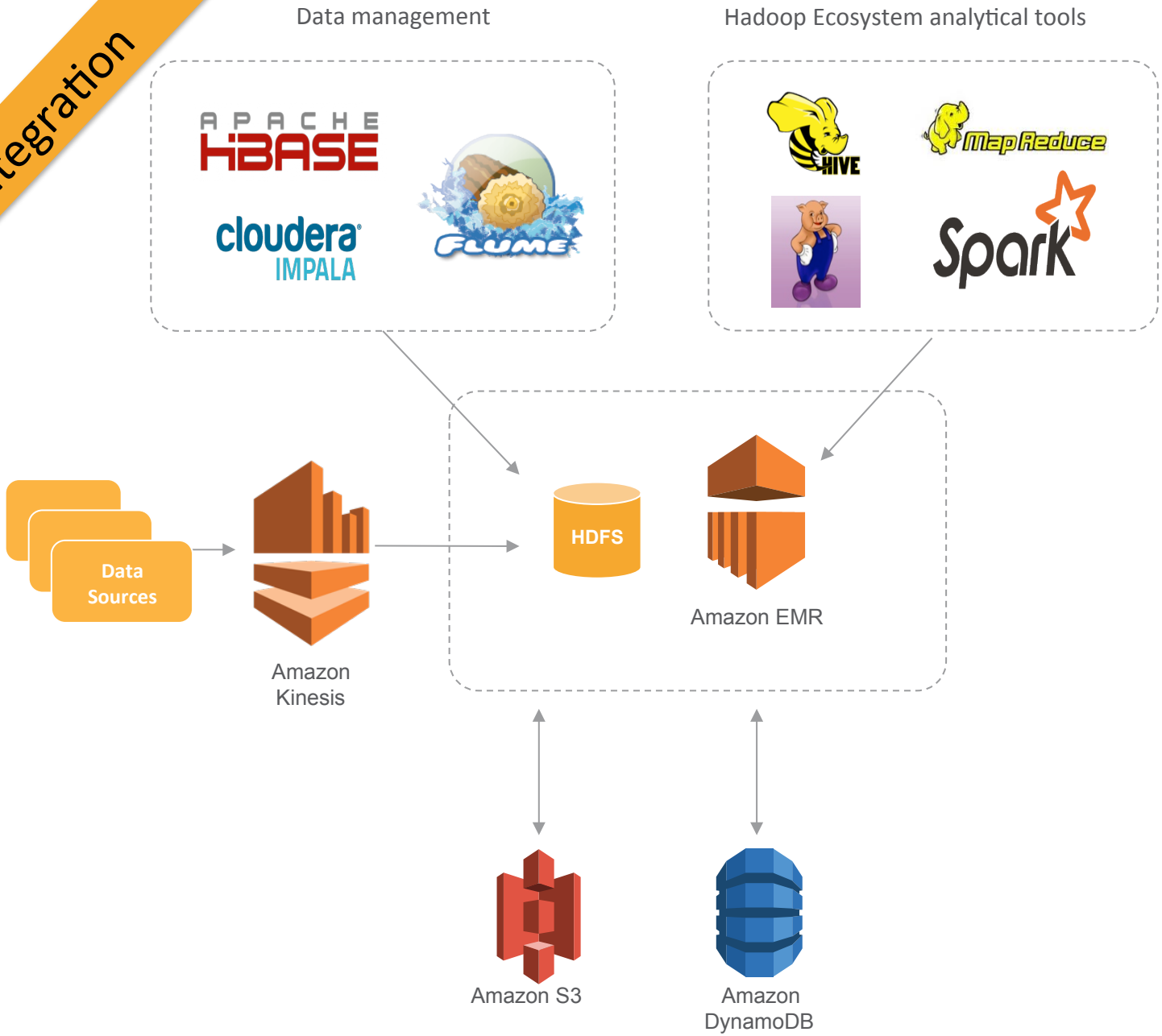
amazon web services

Integration

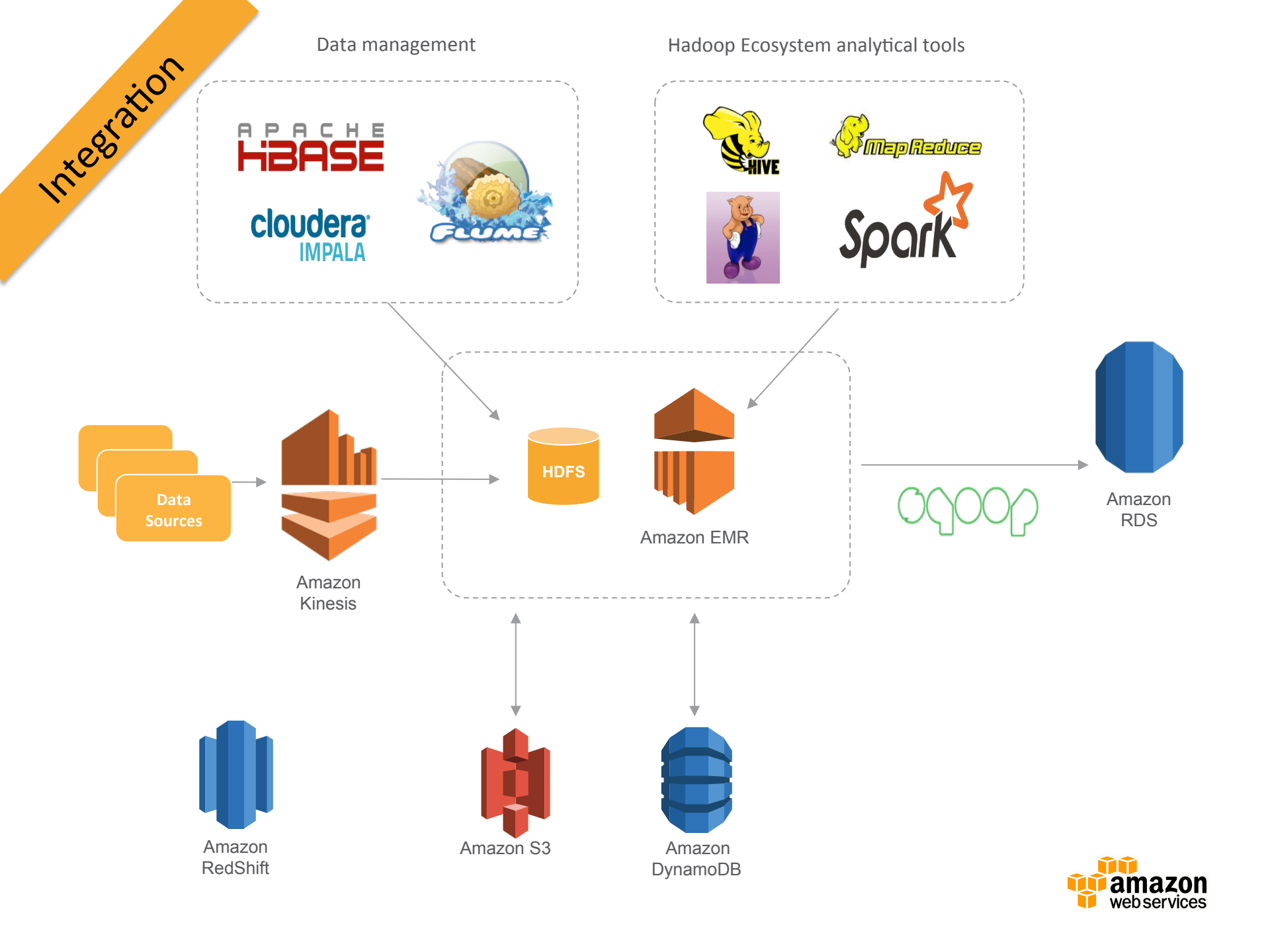


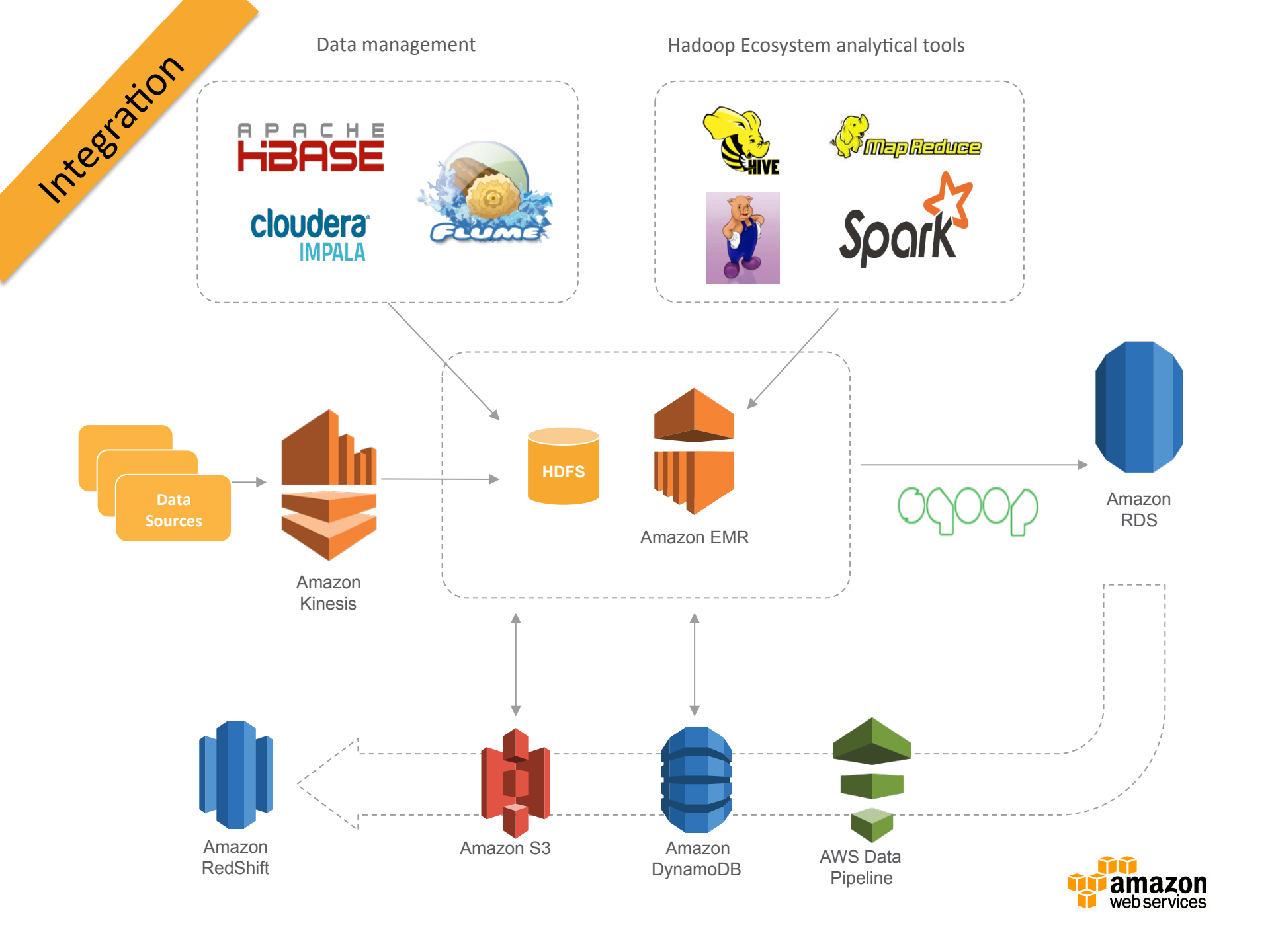
amazon web services

Integration



amazon web services


Integration
Data management
Hadoop Ecosystem analytical tools
APACHE HBASE
cloudera IMPALA
FLUME
HIVE
MapReduce
Spark
Data Sources
Amazon Kinesis
HDFS
Amazon EMR
Amazon RDS
Amazon RedShift
Amazon S3
Amazon DynamoDB
amazon web services

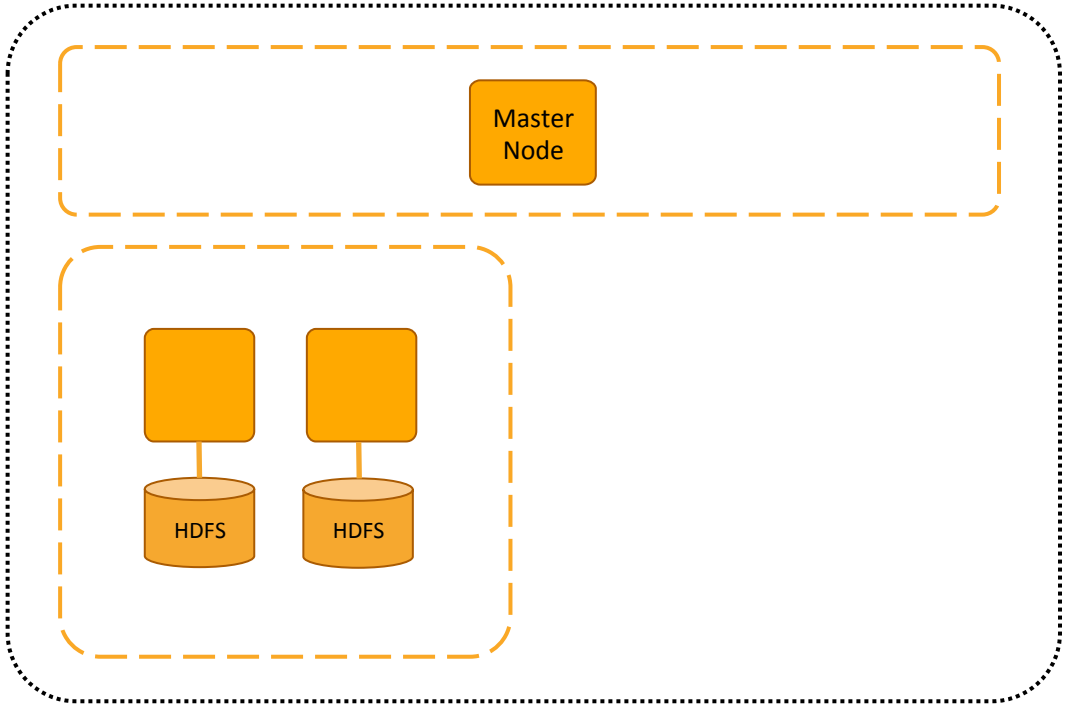

Integration
Data management
Hadoop Ecosystem analytical tools
APACHE HBASE
cloudera IMPALA
FLUME
HIVE
MapReduce
Spark
Data Sources
Amazon Kinesis
HDFS
Amazon EMR
Amazon RDS
Amazon RedShift
Amazon S3
Amazon DynamoDB
AWS Data Pipeline
amazon web services

Amazon EMR Concepts

- Master Node
- Core Nodes
- Task Nodes

Core Nodes

DataNode (HDFS)



amazon web services

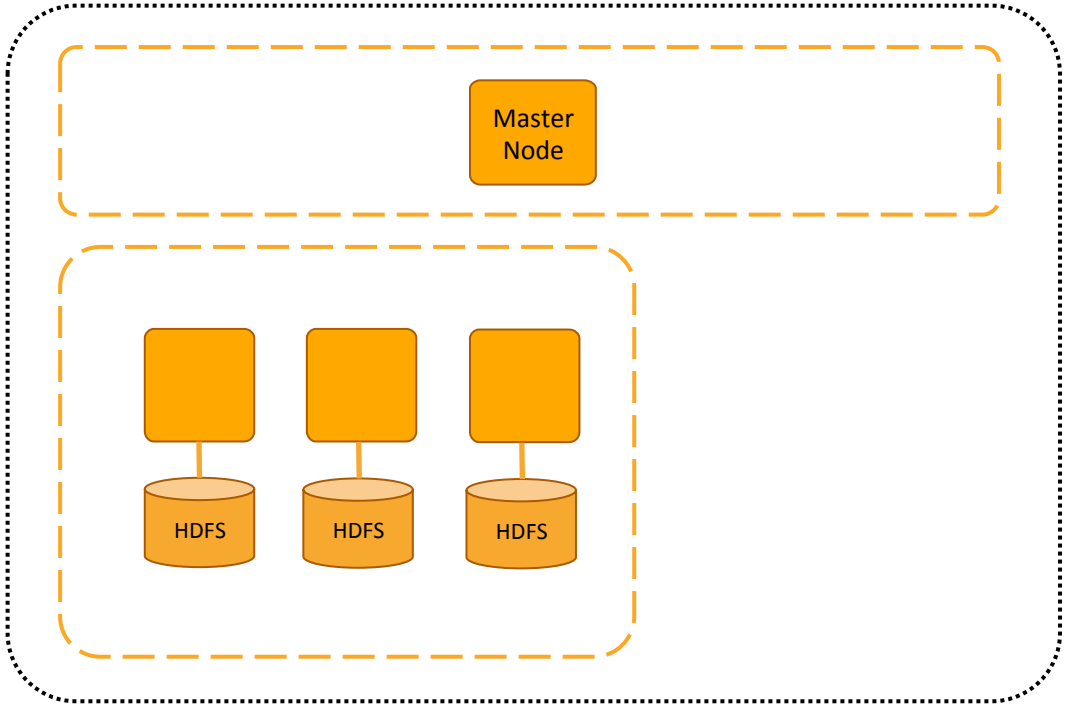
Core Nodes

Can Add Core Nodes:

More CPU

More Memory

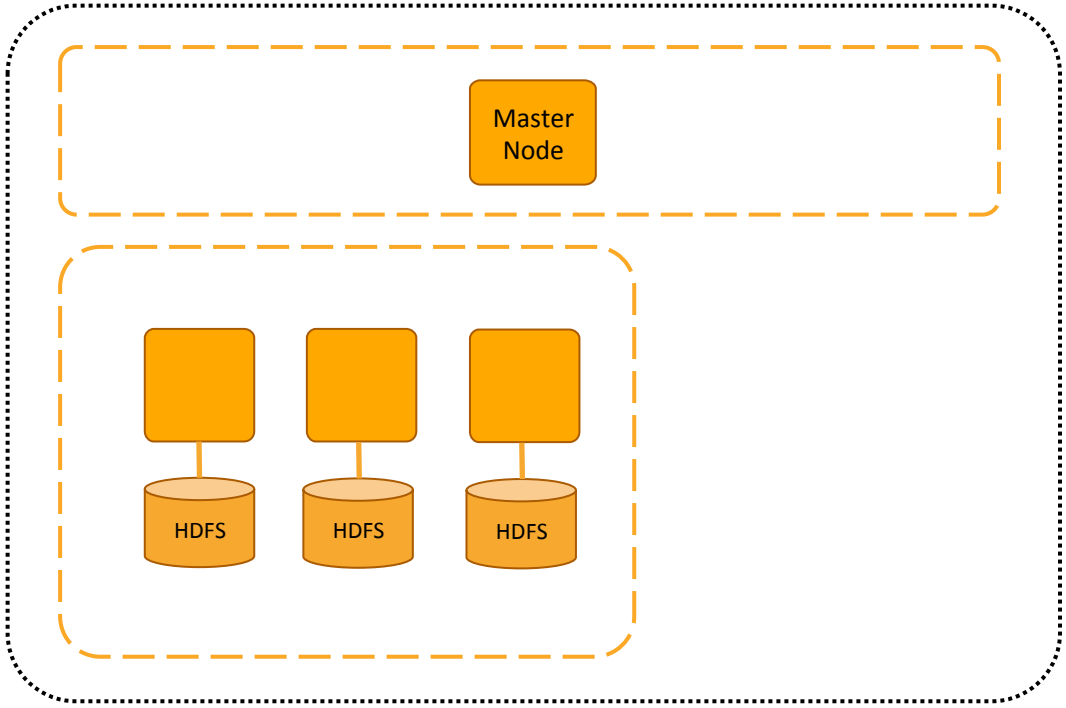
More HDFS Space



Core Nodes

Can’t remove core nodes:

HDFS corruption



amazon web services

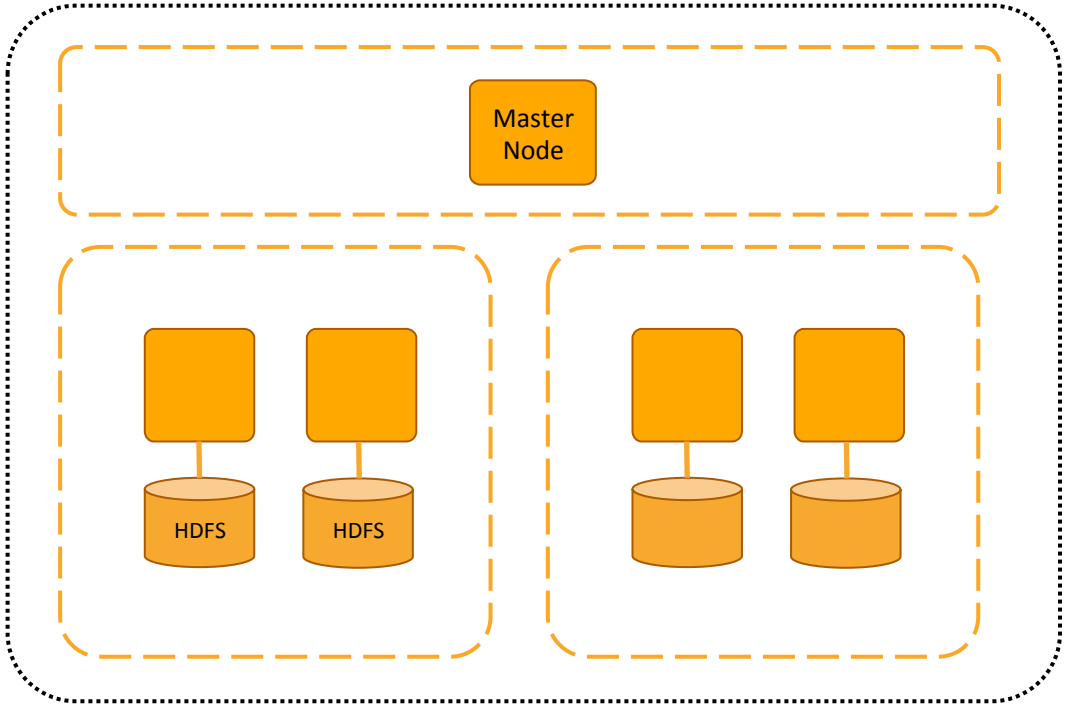
Task Nodes

No HDFS

Provides compute resources:

CPU

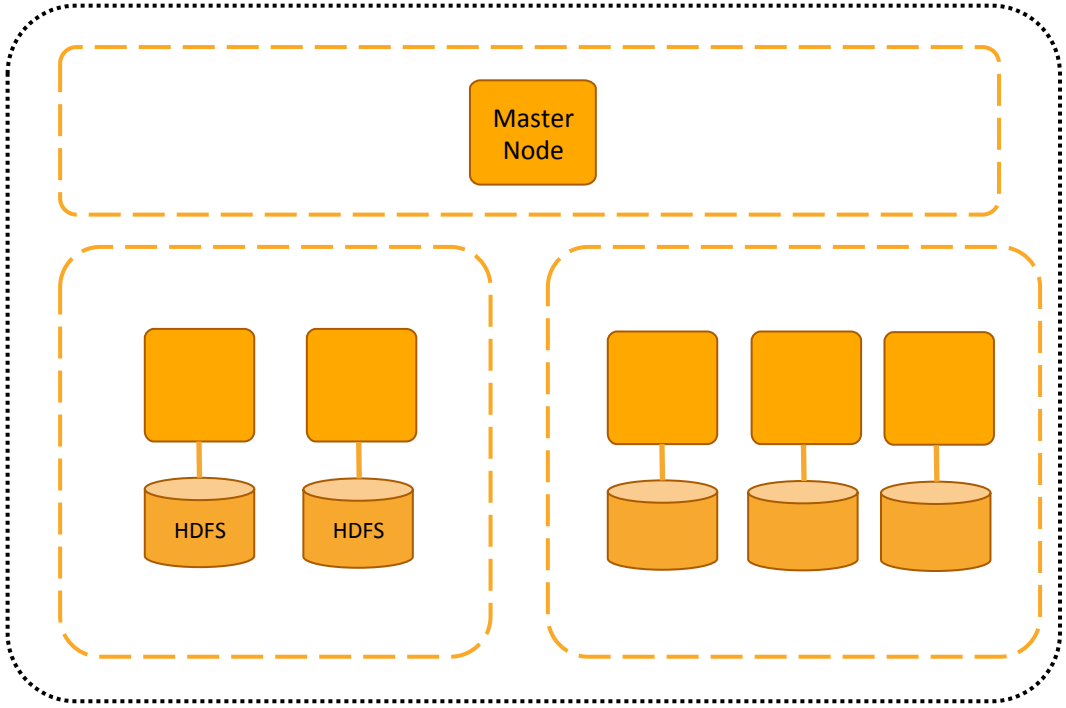
Memory



amazon web services

Task Nodes

Can add and remove task nodes



amazon web services

Spark On Amazon EMR

amazon web services

Bootstrap Actions

- Ability to run or install additional packages/ software on EMR nodes
- Simple bash script stored on S3
- Script gets executed during node/instance boot time
- Script gets executed on every node that gets added to the cluster

Spark on Amazon EMR

- Bootstrap action installs Spark on EMR nodes
- Currently on Spark 0.8.1 & upgrading to 1.0 very soon

amazon web services

Why Spark on Amazon EMR?

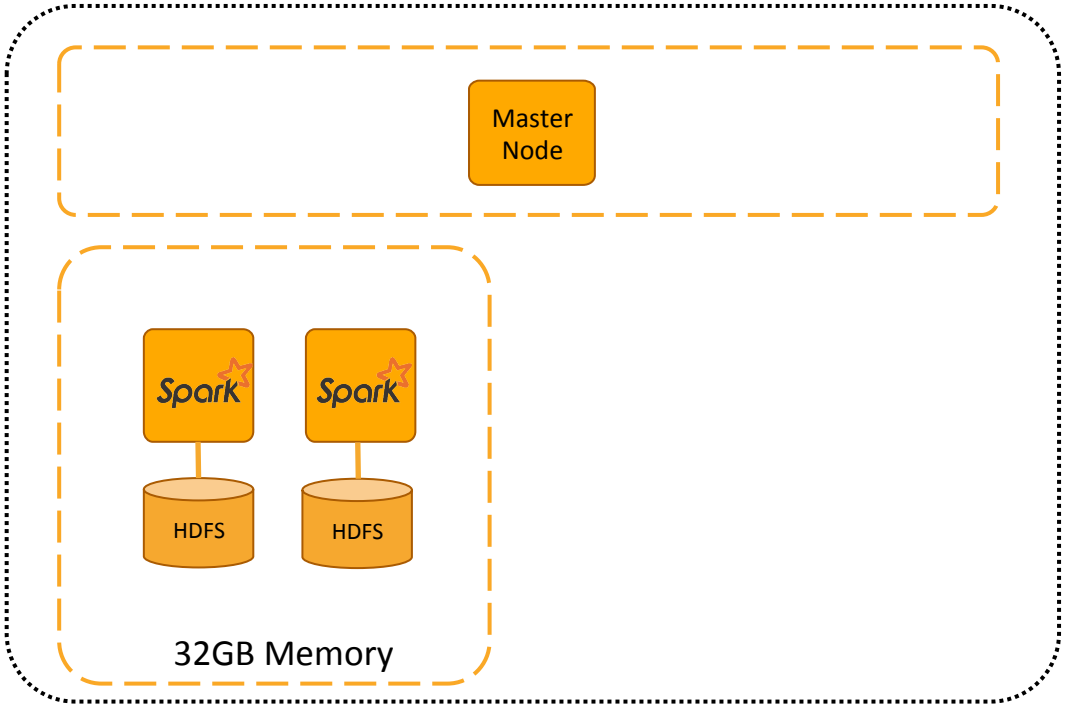
- Deploy small and large Spark clusters in minutes
- EMR manages your cluster, and handles node recover in case of failures
- Integration with EC2 Spot Market, Amazon Redshift, Amazon Data pipeline, Amazon Cloudwatch and etc
- Tight integration with Amazon S3

Why Spark on Amazon EMR?

- Shipping Spark logs to S3 for debugging
  - Define S3 bucket at cluster deploy time

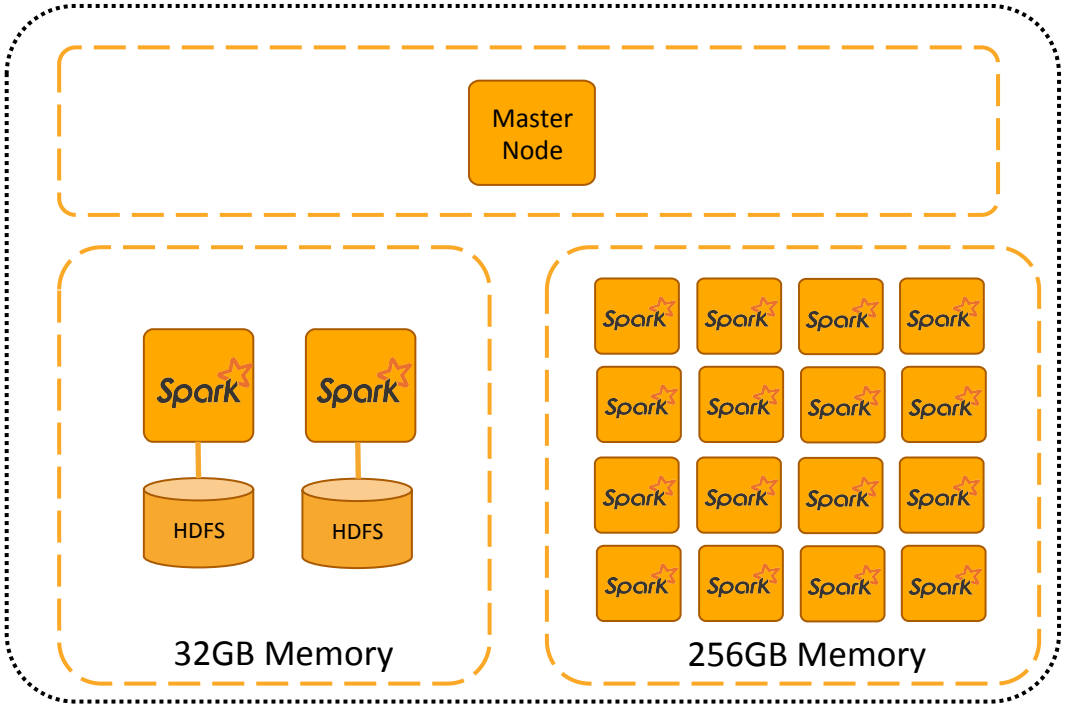
Scaling Spark on EMR

- Launch initial Spark cluster with core nodes
- HDFS to store and checkpoint RDDs



Scaling Spark on EMR

- Add Task nodes in spot market to increase memory capacity



Scaling Spark on EMR

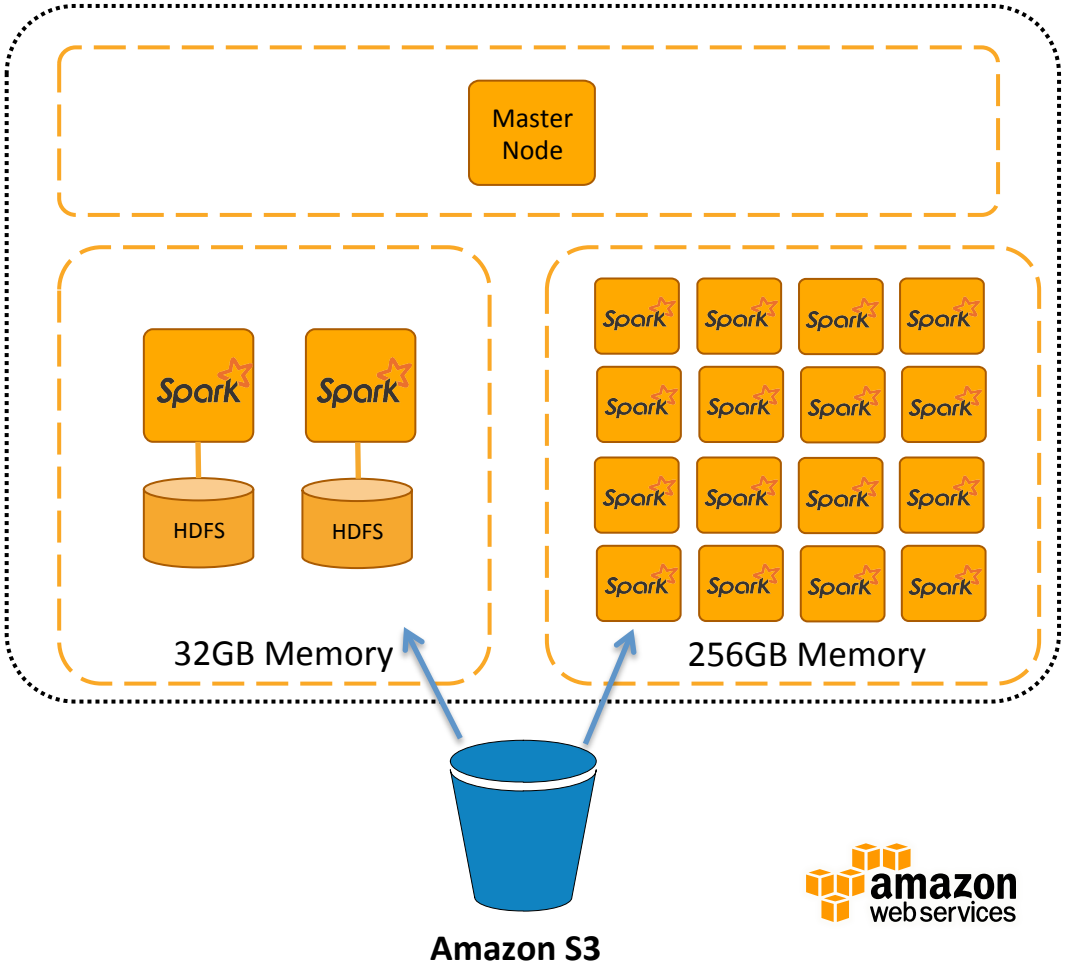
- Create RDDs from HDFS or Amazon S3 with:

  sc.textFile

  OR

  sc.sequenceFile

- Run Computation on RDDs



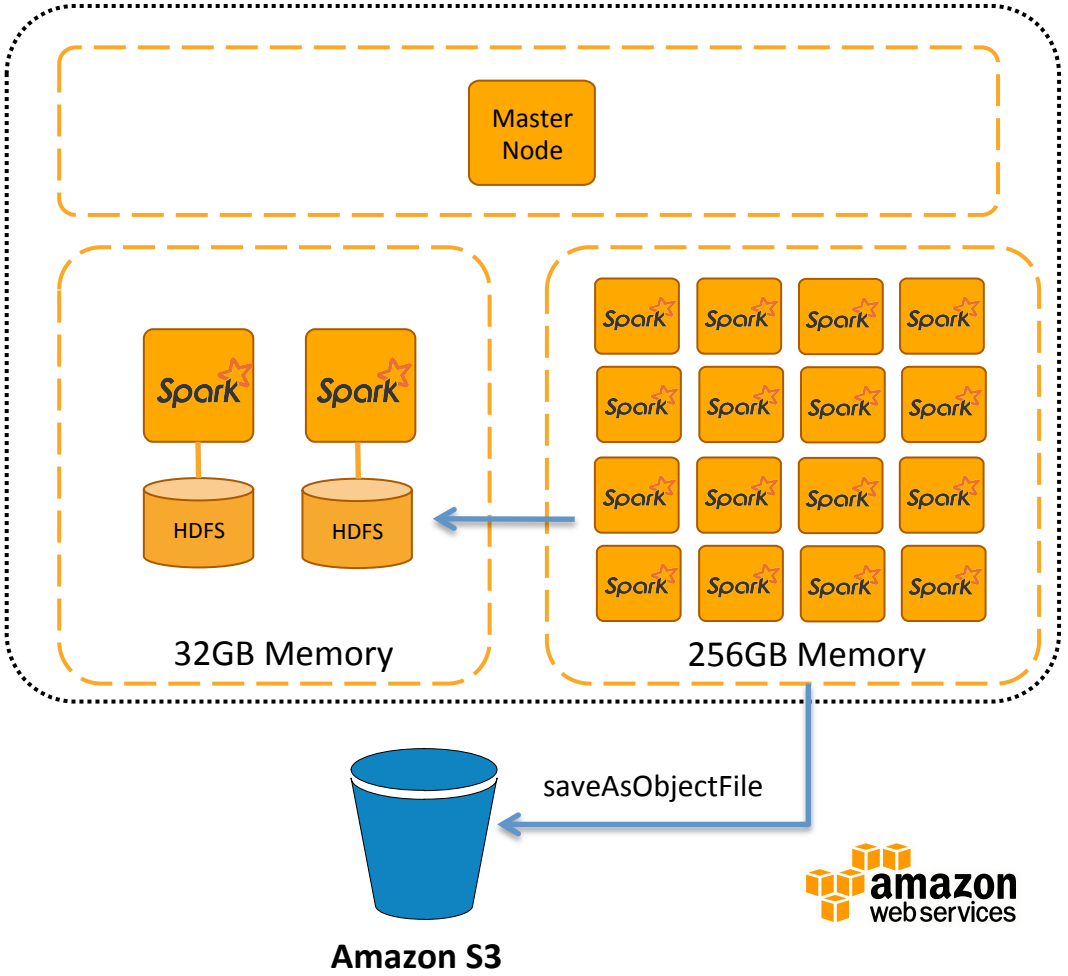
Scaling Spark on EMR

- Save the resulting RDDs to HDFS or S3 with:

  rdd.saveAsSequenceFile

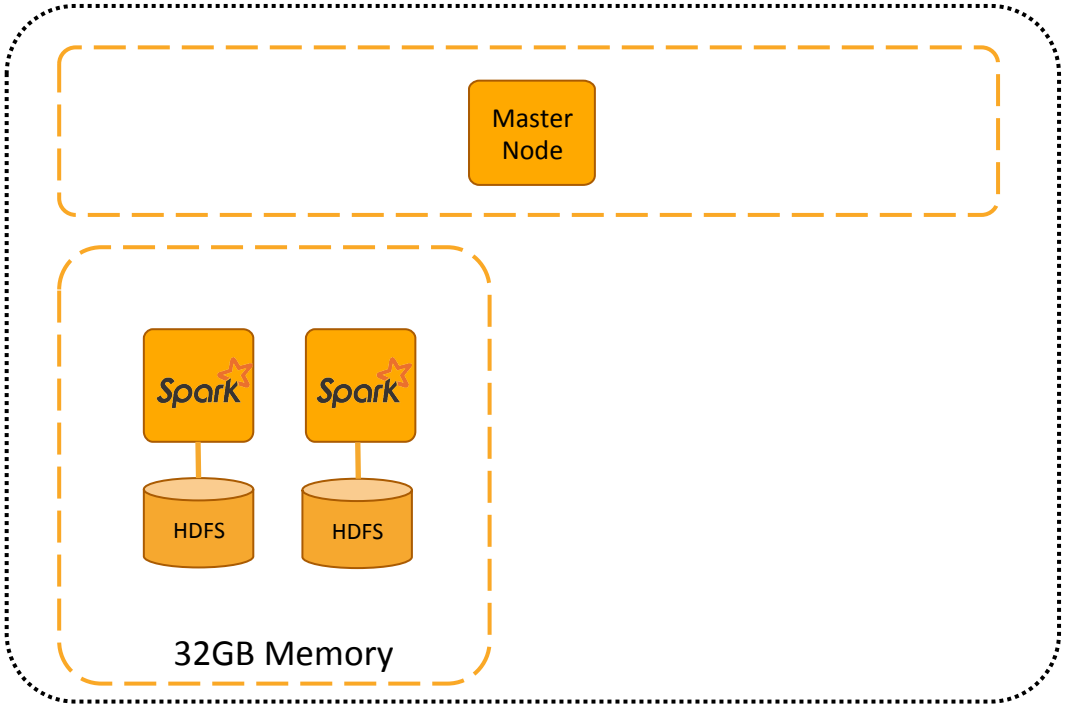
  OR

  rdd.saveAsObjectFile

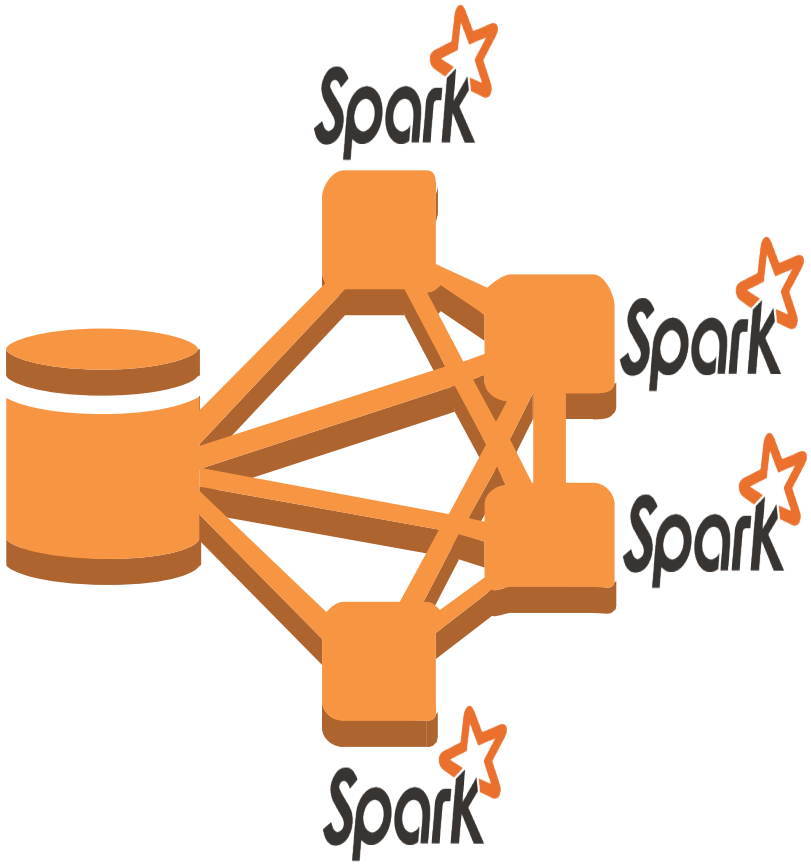


Scaling Spark on EMR

- Shutdown TaskNodes when your job is done

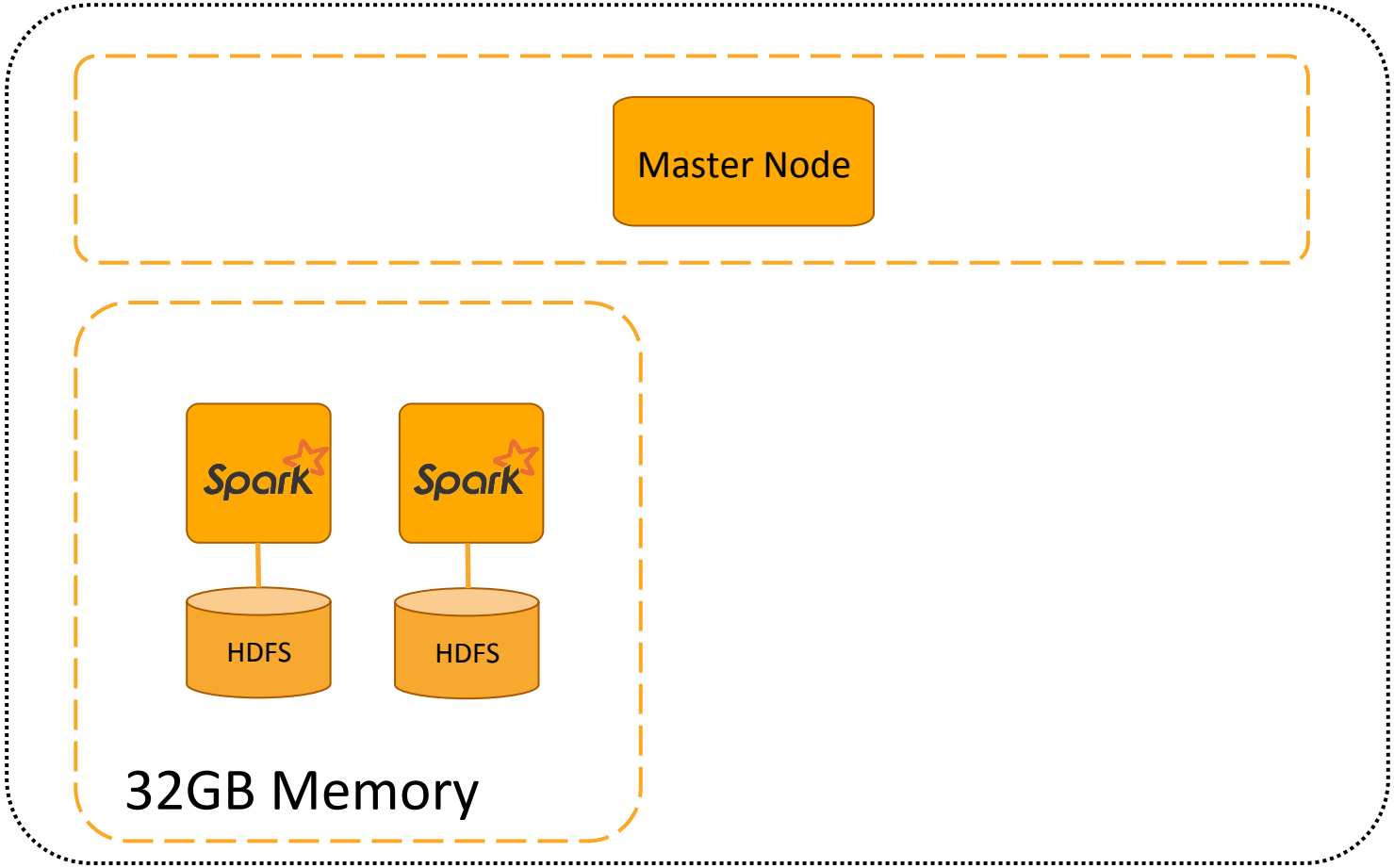


Elastic Spark With Amazon EMR

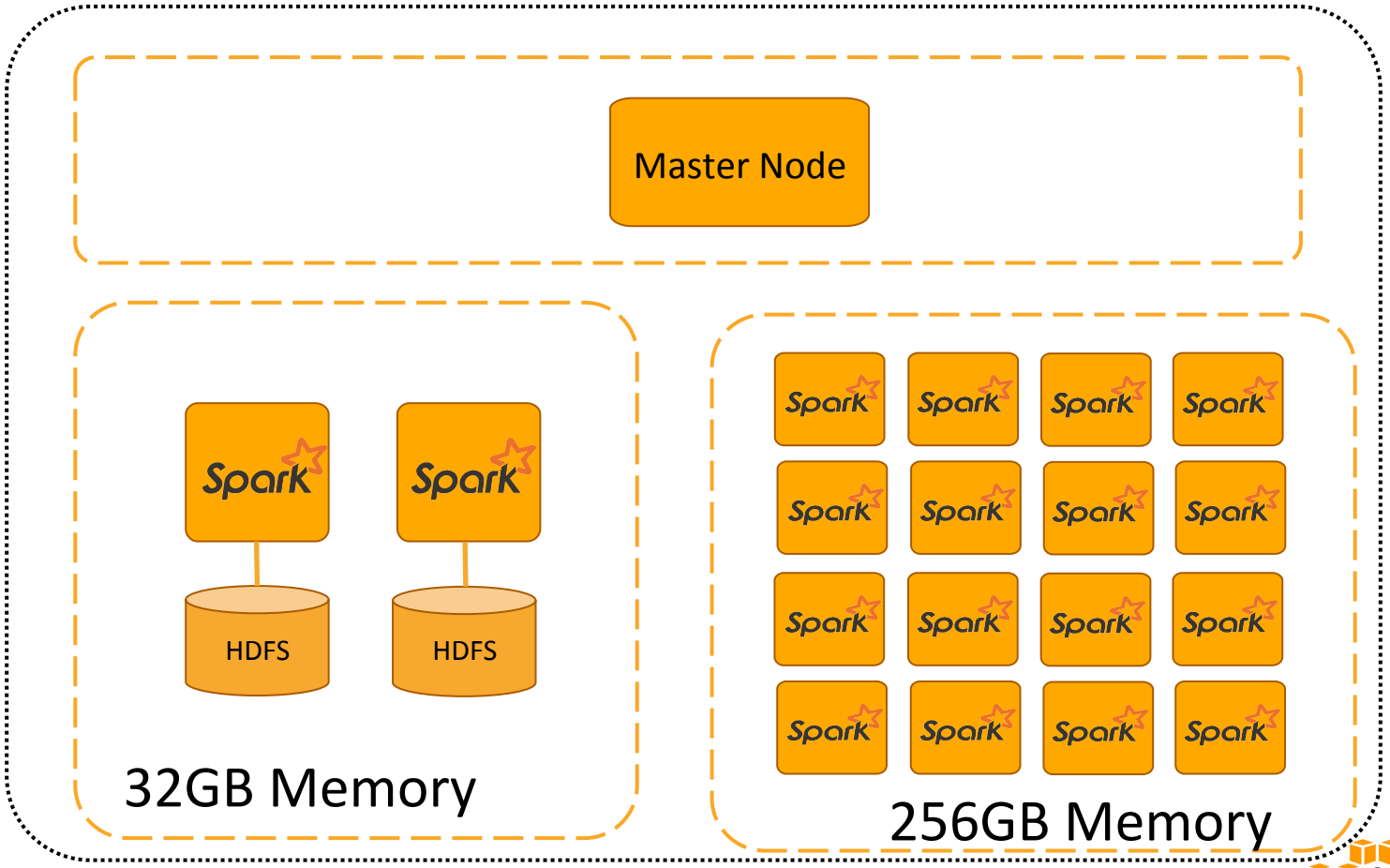


amazon web services

Autoscaling Spark



Autoscaling Spark



amazon web services

Elastic Spark

- When to Scale?
  - Depends on your job
- CPU bounded or Memory intensive?
  - Probably both for Spark jobs
- Use CPU/Memory util. metrics to decide when to scale

Spark Autoscaling Based Memory

- Spark needs memory
  - Lots of it!!
- How to scale based on the memory usage?

Launching a 1000 node Cluster

Amazon EMR

amazon web services

Launching a 1000 node Cluster

- What do I need to launch a cluster?
  - AWS Account
  - Amazon EMR CLI

Launching a 1000 node Cluster

- Easy to launch – 1 command

```
./elastic-mapreduce --create –alive
    --name "Spark/Shark Cluster"   \
    --bootstrap-action s3://elasticmapreduce/samples/spark/0.8.1/install-spark-shark.sh
    --bootstrap-name "Spark/Shark"
    --instance-type m1.xlarge
    --instance-count 1000
```

- Comes up in 15-20mins

amazon web services

Launching a 1000 node Cluster

- Adding Task nodes

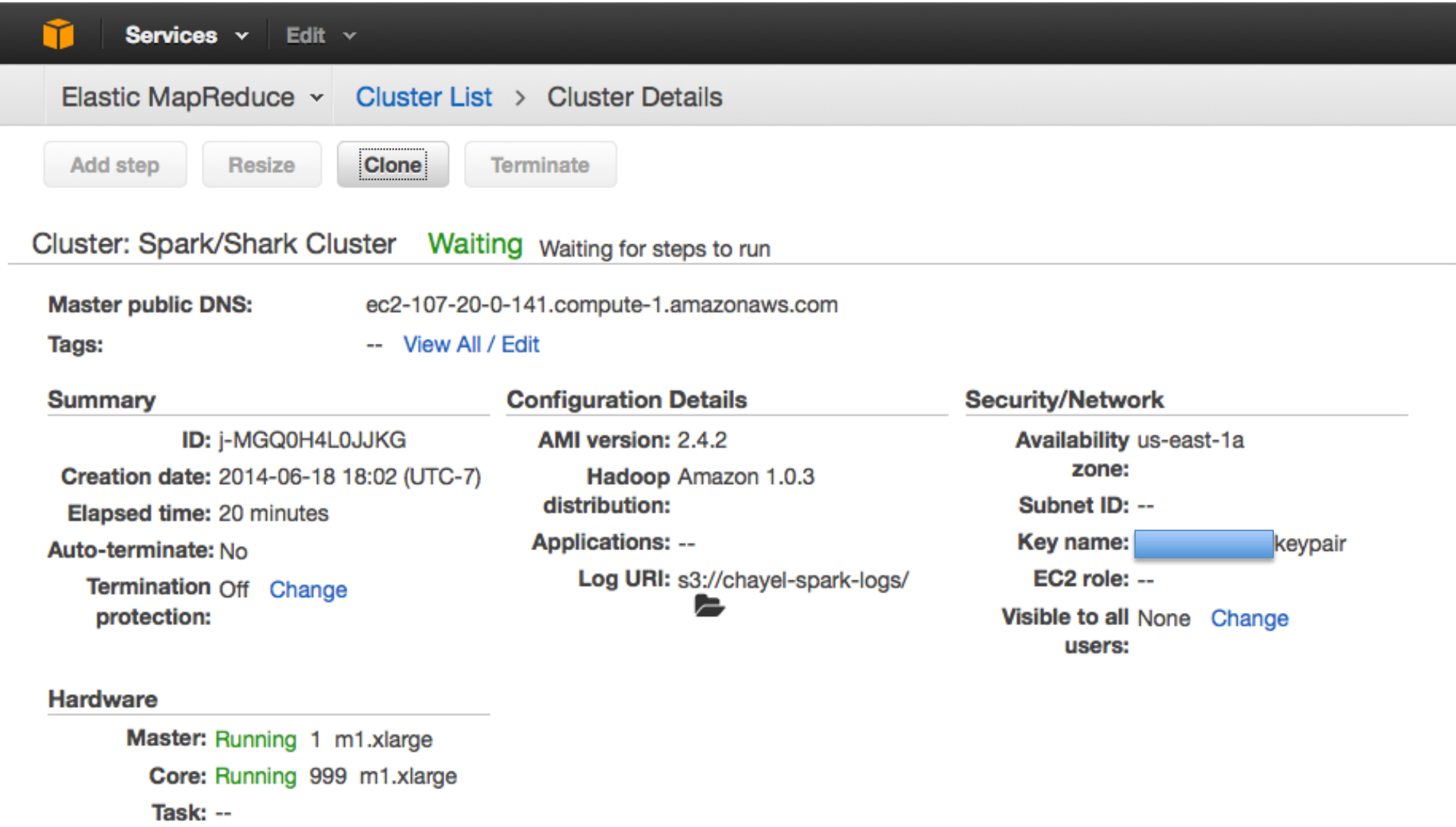
--add-instance-group *TASK*

--instance-count *INSTANCE_COUNT*

--instance-type *INSTANCE_TYPE*

Is cluster ready?

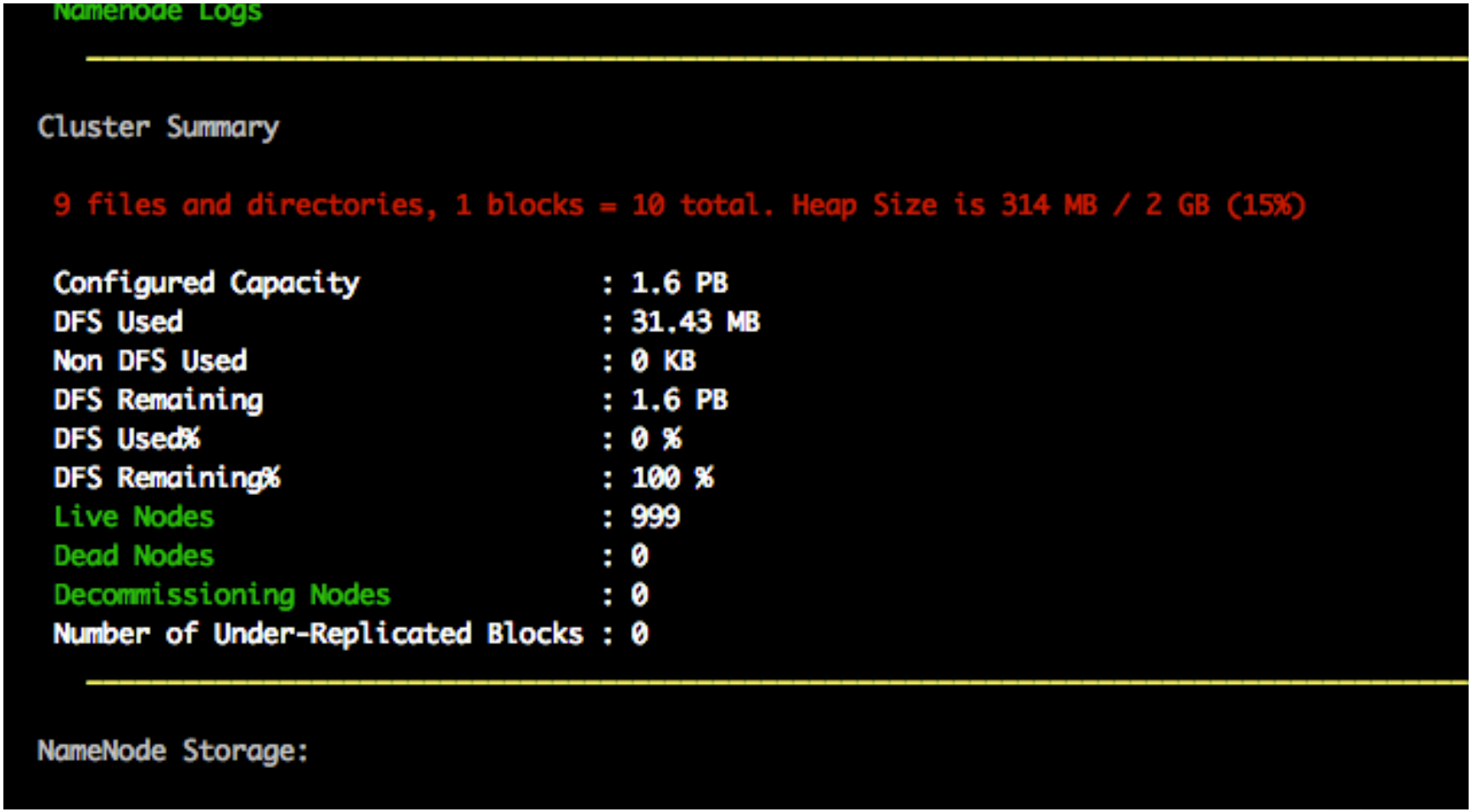
- Cluster will be in Waiting state



amazon web services

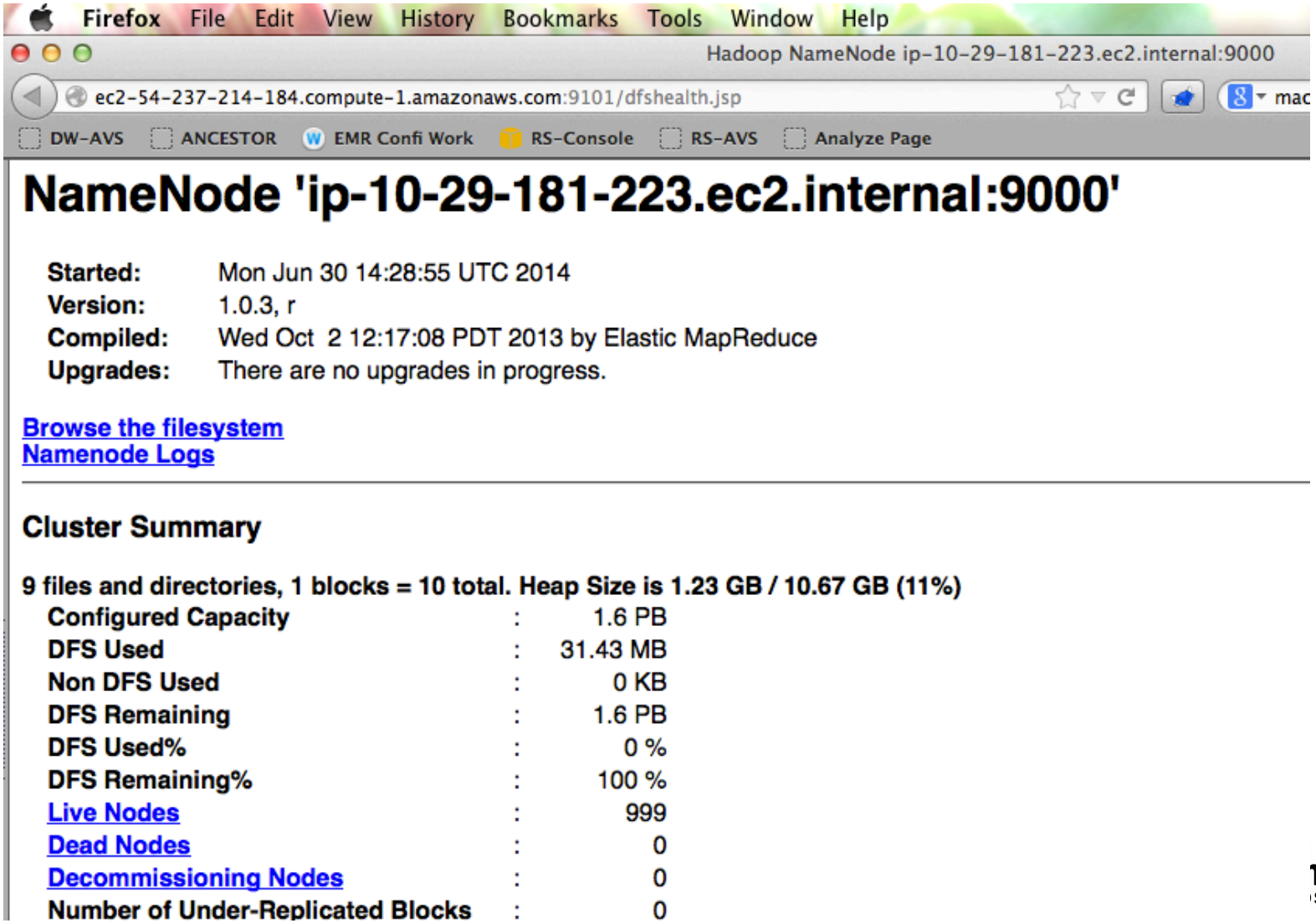
Lynx Interface

lynx http://localhost:9101

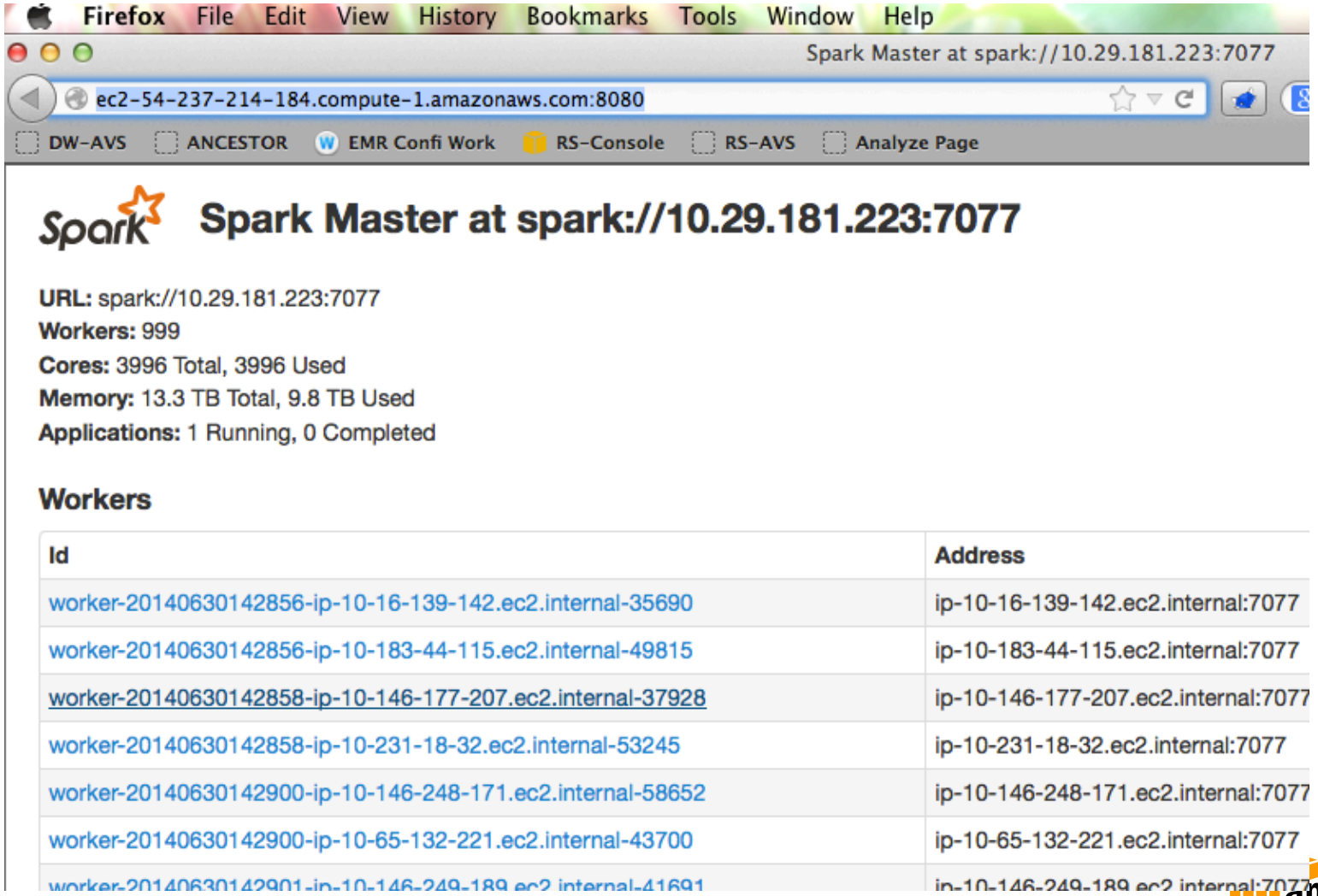


amazon web services

Web Interface



Spark UI



Dataset

- Wikipedia article traffic statistics
  - 4.5 TB
  - 104 Billion records

- Stored in Amazon S3
  - s3://bigdata-spark-demo/wikistats/

Dataset

- File structure (pagecount-DATE-HOUR.gz)
- Period: Dec-2007 to Feb-2014

- Format of File (tsv)
- Feilds

projectcode, pagename, pageviews, and bytes

Sample dataset

Projectcode pagename pageviews bytes

en Barack_Obama 997 123091092

en Barack_Obama%27s_first_100_days 8 850127

en Barack_Obama,_Jr 1 144103

en Barack_Obama,_Sr. 37 938821

en Barack_Obama_%22HOPE%22_poster 4 81005

en Barack_Obama_%22Hope%22_poster 5 102081

Loading data

Analyze options

Table structure

```
create external table wikistats
(
projectcode string,
pagename string,
pageviews int,
pagesize int
)
ROW FORMAT
DELIMITED FIELDS
TERMINATED BY ' '
LOCATION 's3n://bigdata-spark-demo/wikistats/';

ALTER TABLE wikistats add partition(dt='2007-12') location 's3n://bigdata-spark-demo//wikistats/2007/2007-12';
…….
```

Adding partitions for every month till 2014-04

Analyze using Shark

Top 10 Page Views in Jan 2014

```
shark> select pagename, sum(pageviews) c
     > from wikistats_cached
     > where dt='2014-01'
     > group by pagename
     > order by c desc
     > limit 10;
```

Exec time: 26 secs

Scanning 250GB of data

Analyze using Shark

Query 2:

Top 10 Page Views Overall

```
shark> select pagename, sum(pageviews) c
     > from wikistats_cached
     > group by pagename
     > order by c desc
     > limit 10;
```

- Exec time: 45 sec
- Scanning 4.5TB of data

Analyze using Shark

Query 3:

No of pages in each projectcodes

```
shark> select projectcode, count(pagename)
     > from wikistats_cached
     > group by projectcode
     > ;
OK
```

Exec time: 48 secs

Scanning 4.5TB of data / 104 Billion records

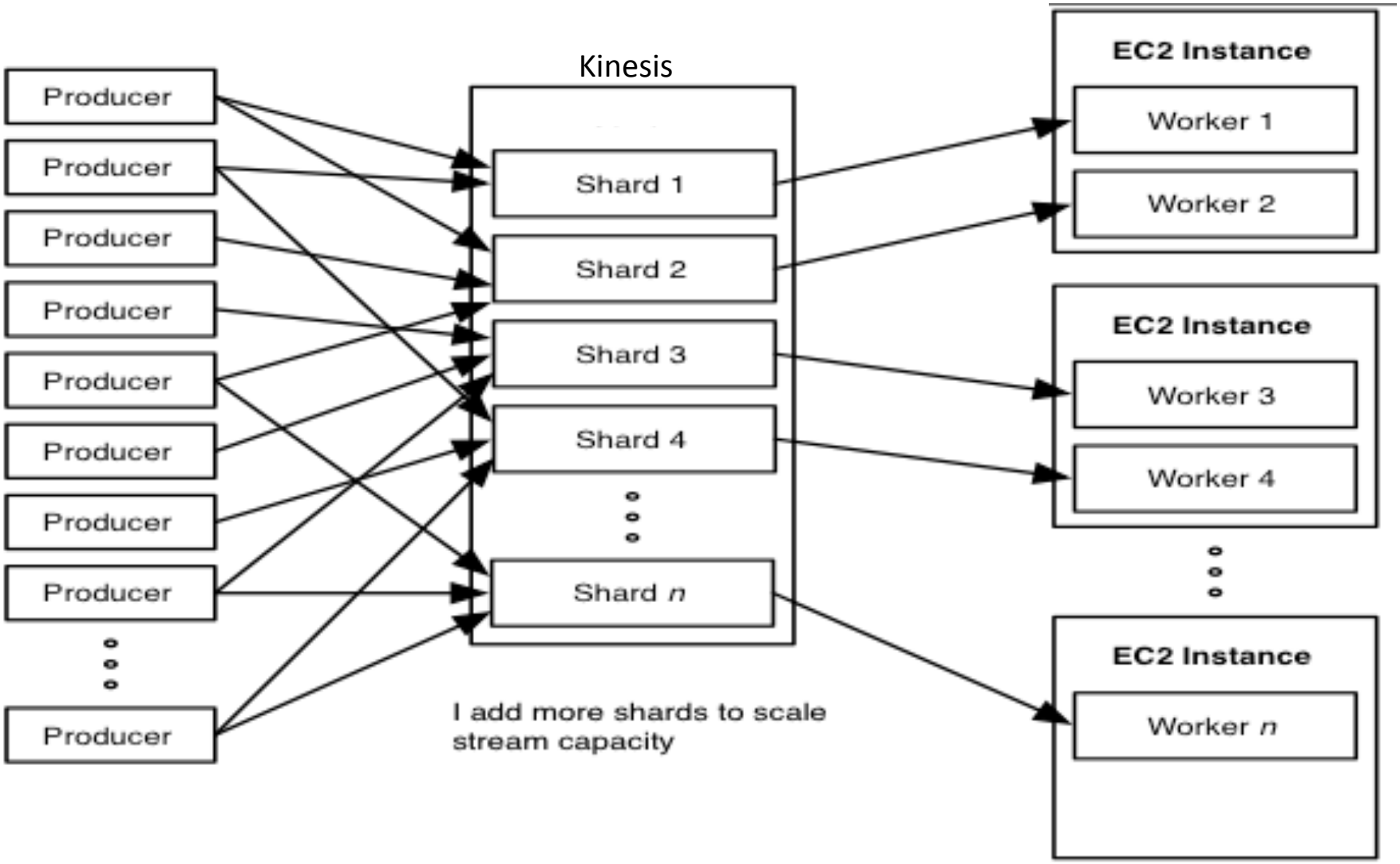
Spark Streaming and Amazon Kinesis

amazon web services

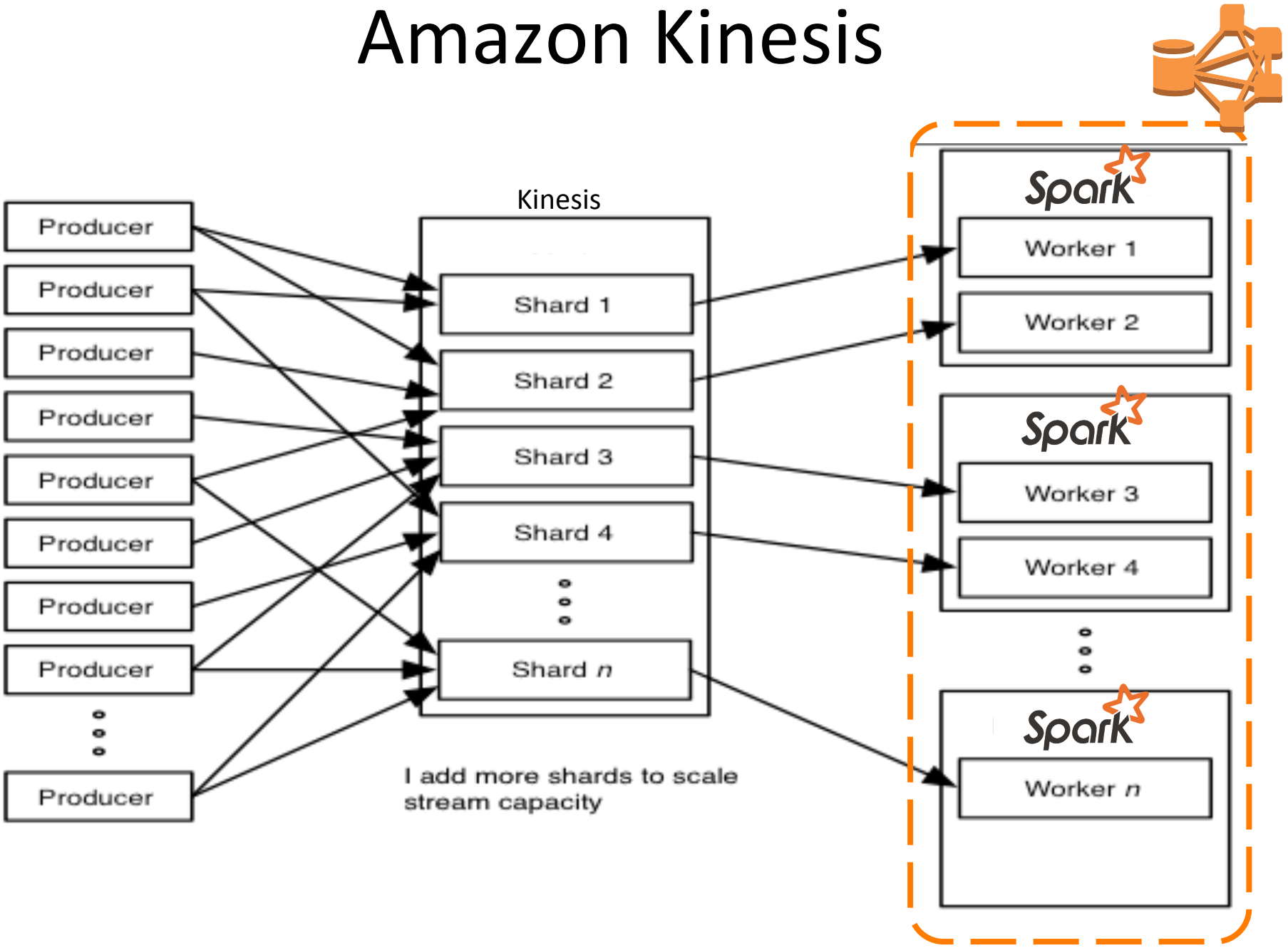
Amazon Kinesis

- **CreateStream**
  - Creates a new Data Stream within the Kinesis Service
- **PutRecord**
  - Adds new records to a Kinesis Stream
- **DescribeStream**
  - Provides metadata about the Stream, including name, status, Shards, etc.
- **GetNextRecord**
  - Fetches next record for processing by user business logic
- **MergeShard / SplitShard**
  - Scales Stream up/ down
- **DeleteStream**
  - Deletes the Stream

Amazon Kinesis



Amazon Kinesis



What Do You Like To See?

Send Feedbacks To:

Manjeet Chayel

chayel@amazon.com

http://bit.ly/sparkemr