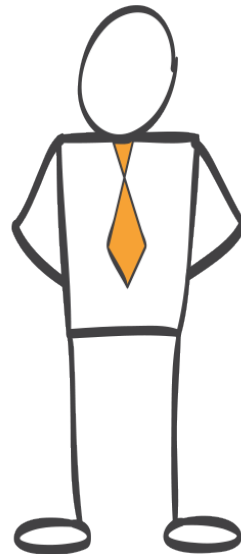


Building 1000 node *Spark* cluster on EMR

Manjeet Chayel

What is EMR?

Amazon Elastic MapReduce



What is EMR?



Hadoop-as-a-service

Integrated with tools

Integrated to AWS services

Cost effective AWS wrapper

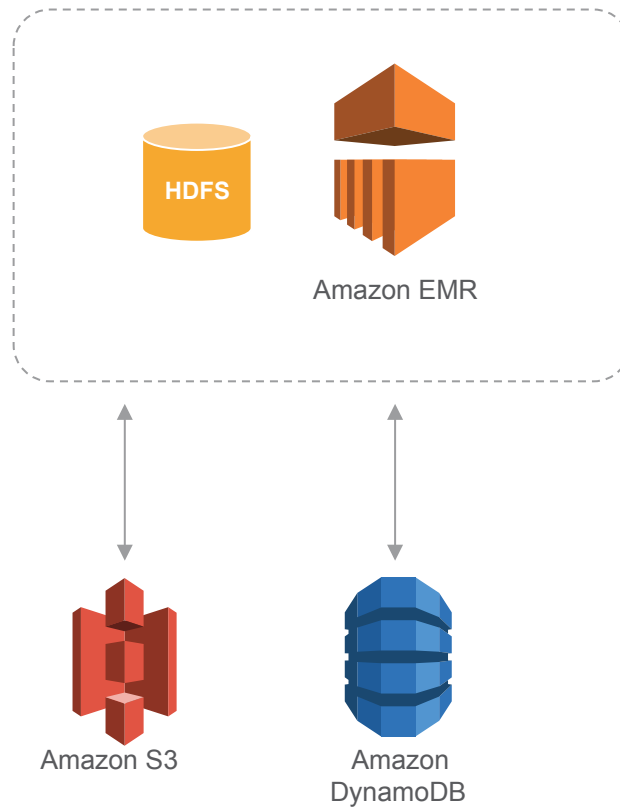
Map-Reduce engine

Massively parallel

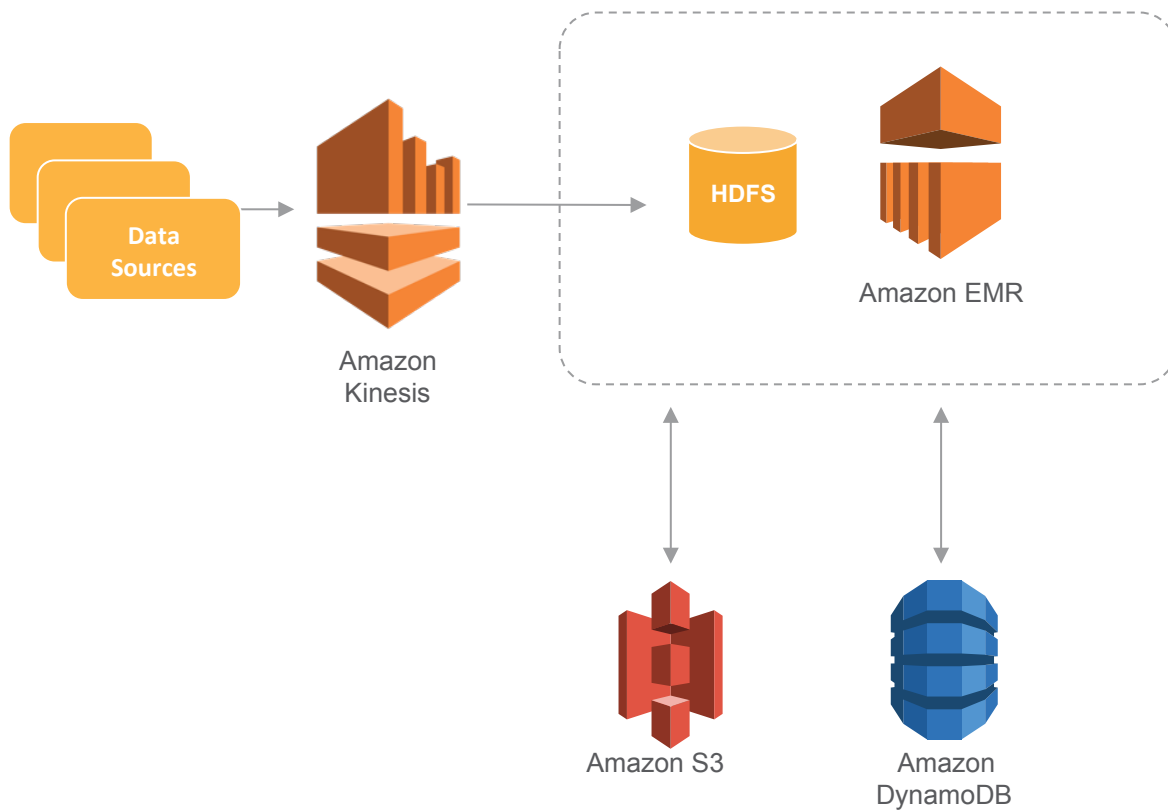
Integration



Integration



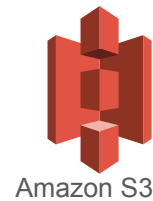
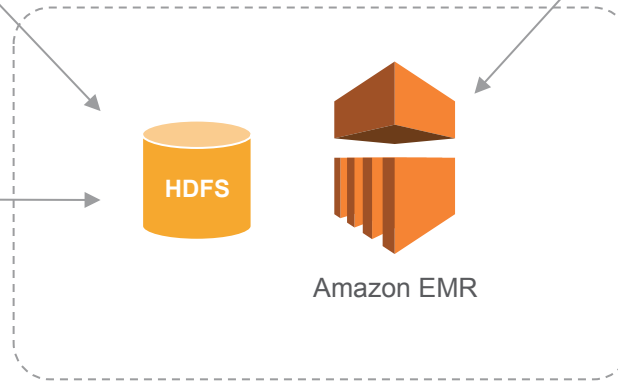
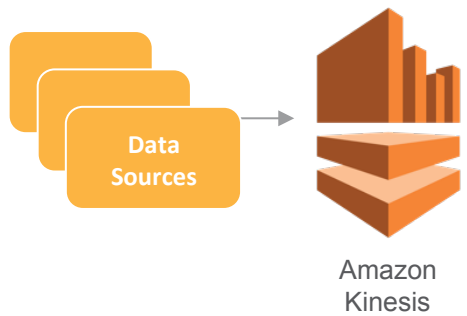
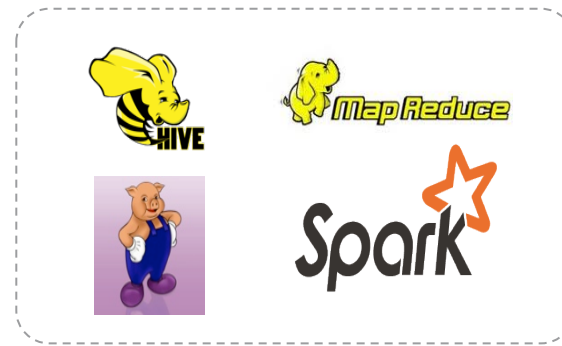
Integration



Integration

Data management

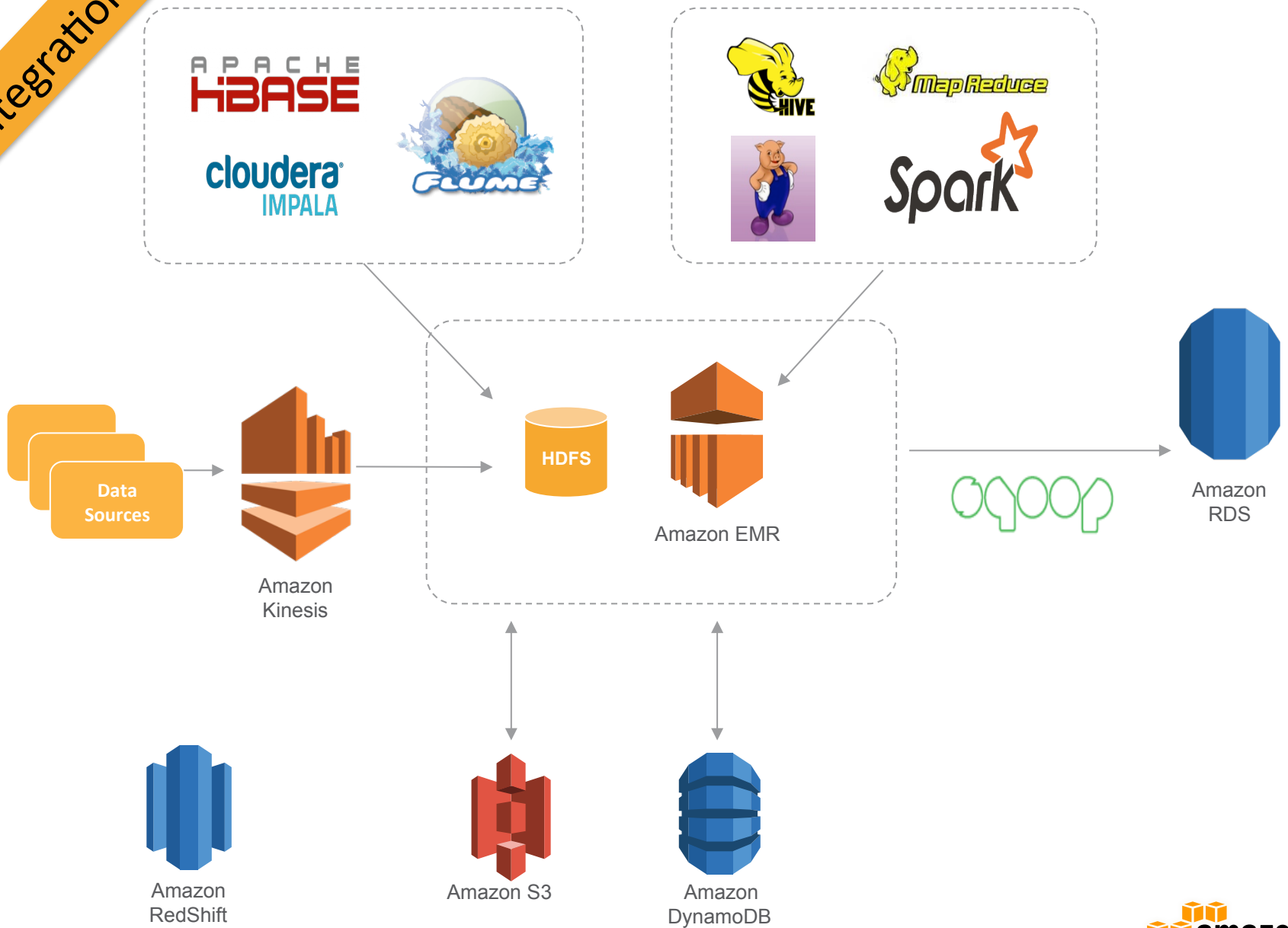
Hadoop Ecosystem analytical tools



Integration

Data management

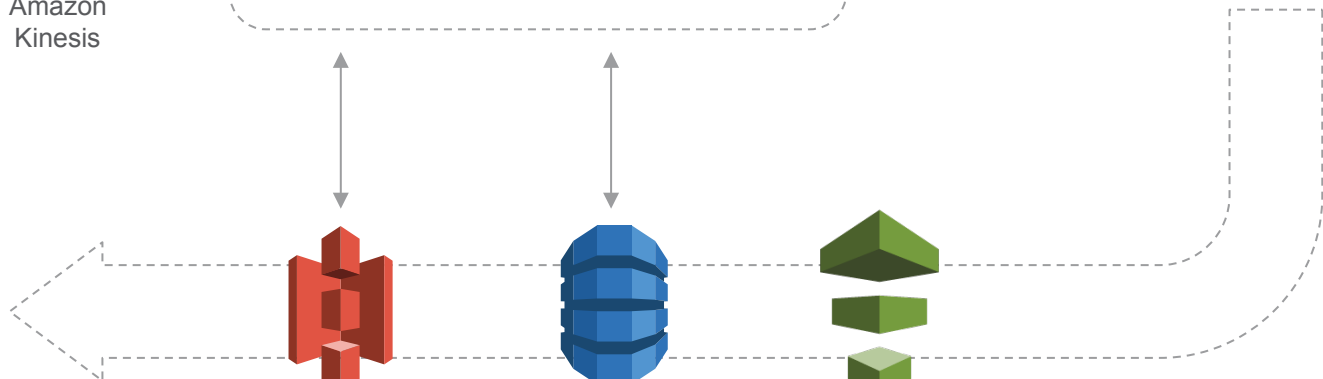
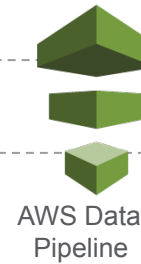
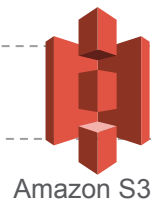
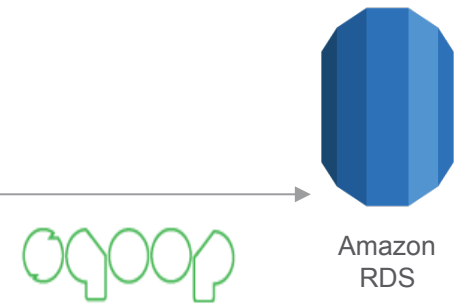
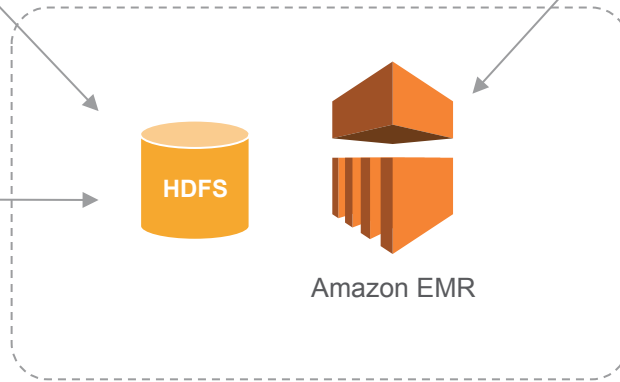
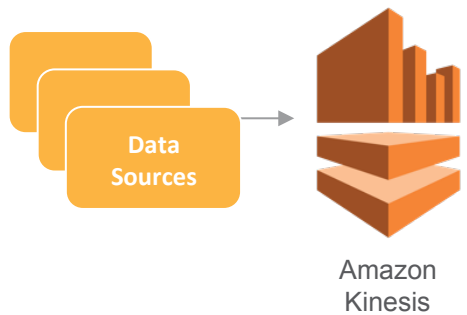
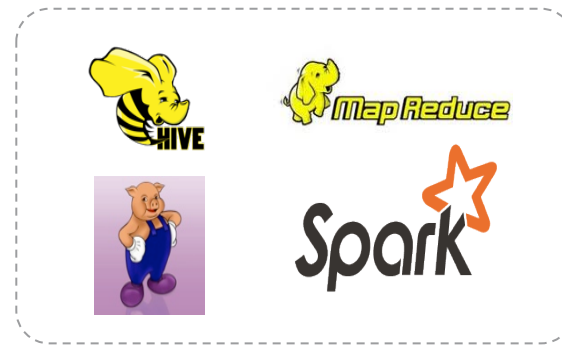
Hadoop Ecosystem analytical tools



Integration

Data management

Hadoop Ecosystem analytical tools

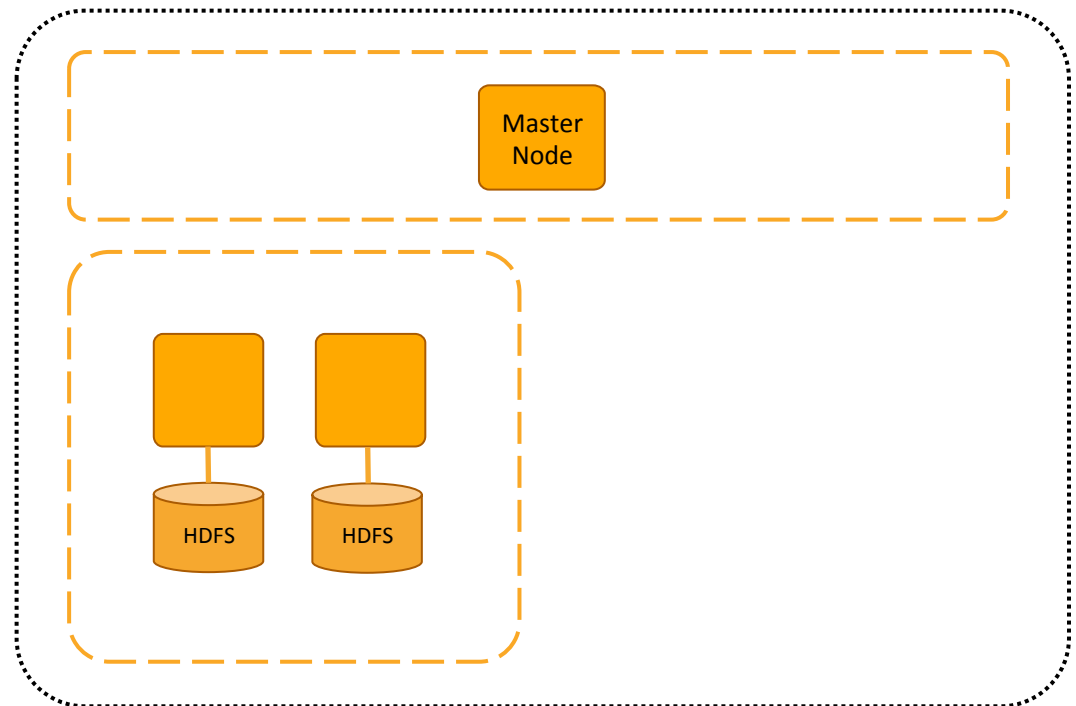


Amazon EMR Concepts

- Master Node
- Core Nodes
- Task Nodes

Core Nodes

DataNode (HDFS)



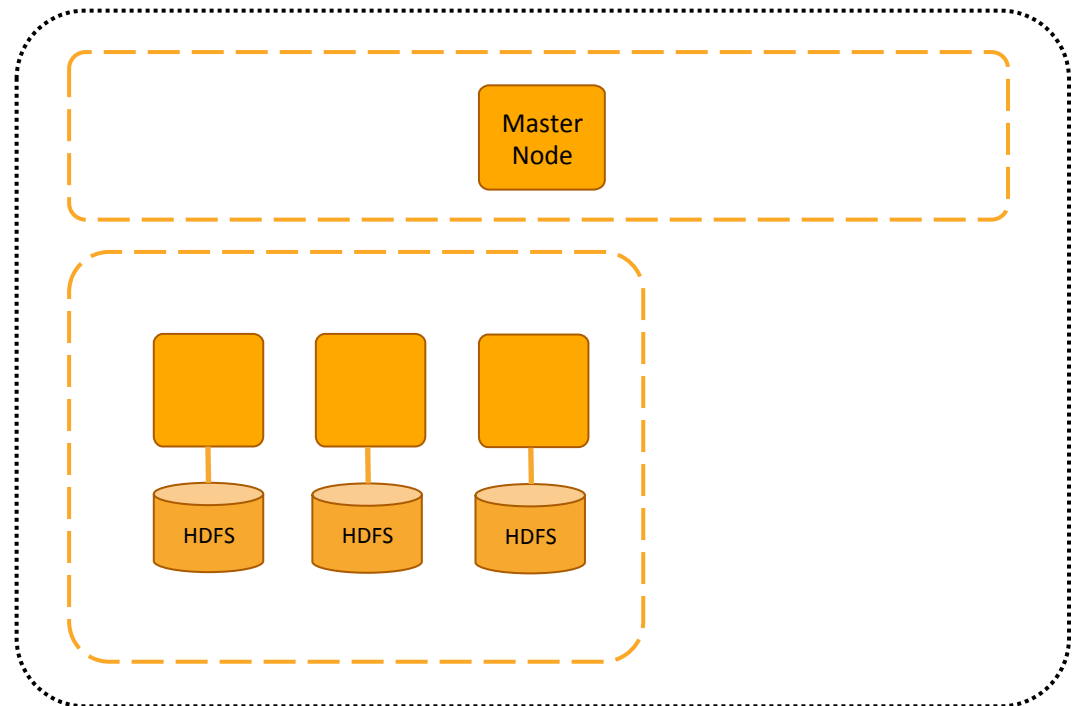
Core Nodes

Can Add Core Nodes:

More CPU

More Memory

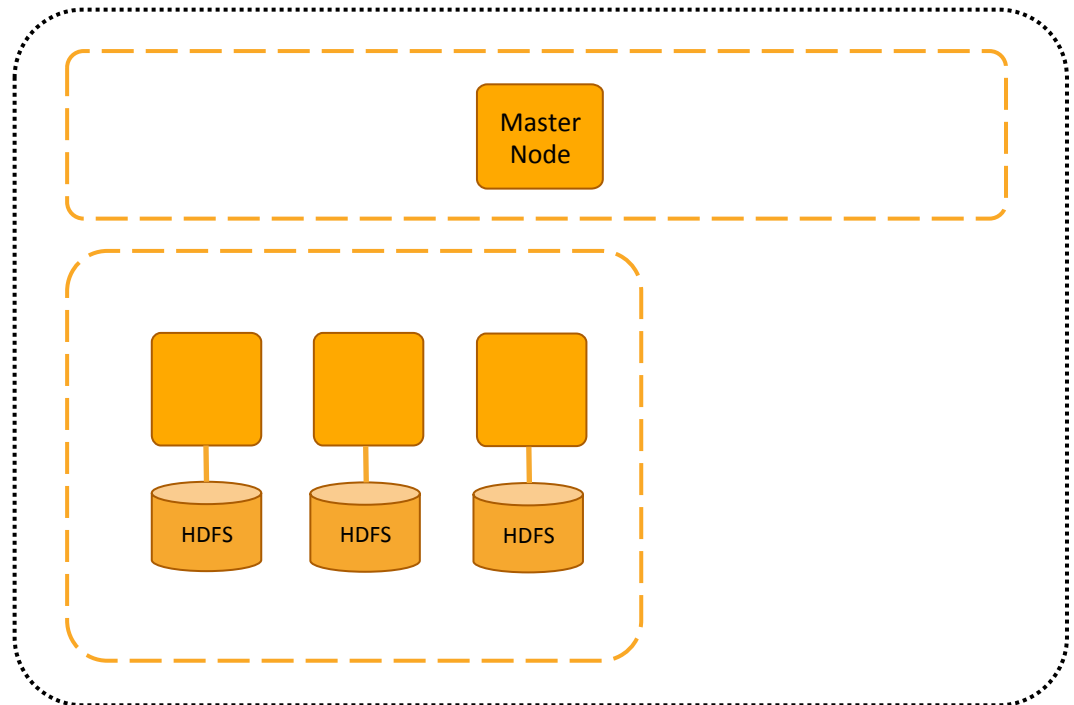
More HDFS Space



Core Nodes

Can't remove core
nodes:

HDFS corruption



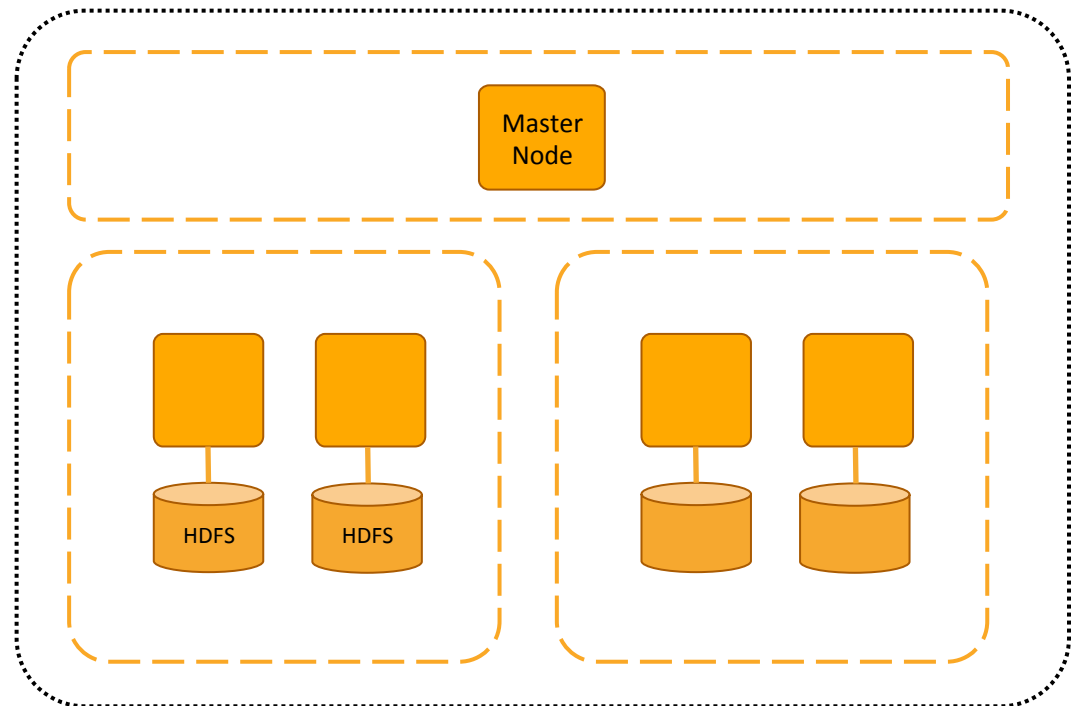
Task Nodes

No HDFS

Provides compute
resources:

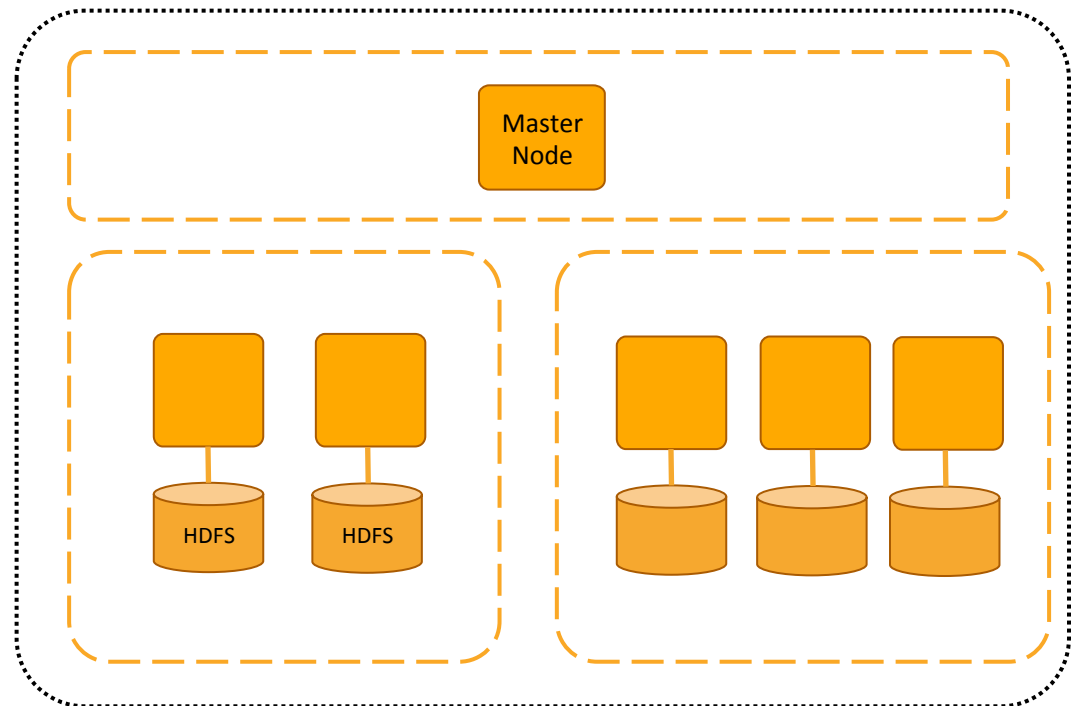
CPU

Memory



Task Nodes

Can add and remove
task nodes



Spark On Amazon EMR



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

Why Spark on Amazon EMR?

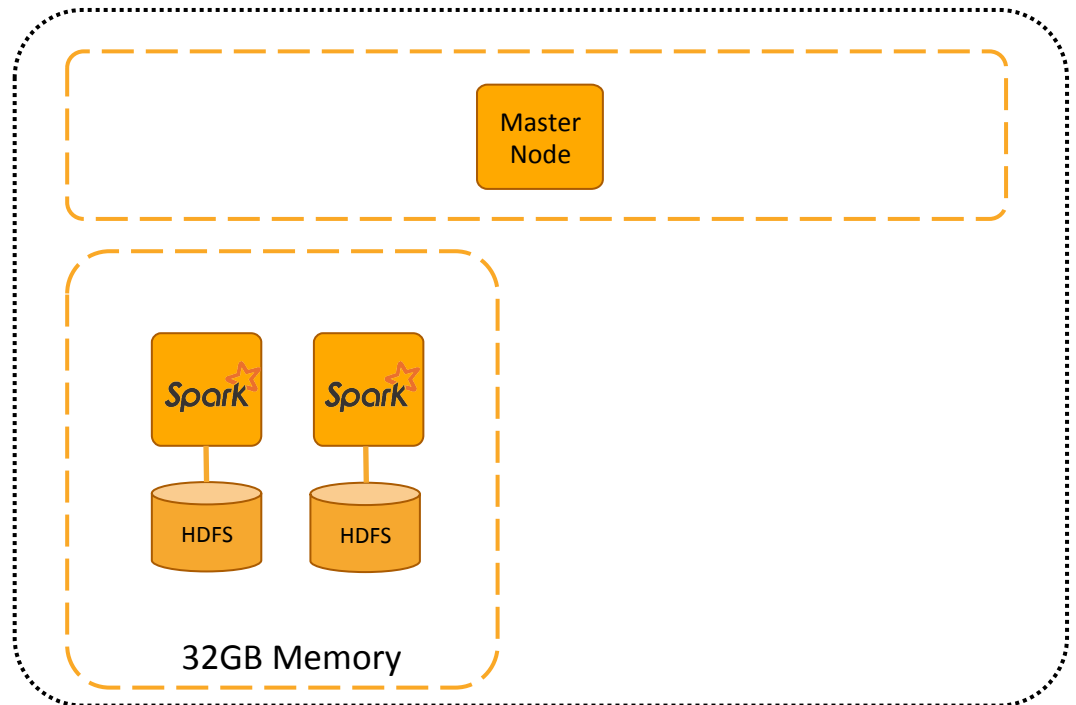
- Deploy small and large Spark clusters in minutes
- EMR manages your cluster, and handles node recover in case of failures
- Integration with EC2 Spot Market, Amazon Redshift, Amazon Data pipeline, Amazon Cloudwatch and etc
- Tight integration with Amazon S3

Why Spark on Amazon EMR?

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

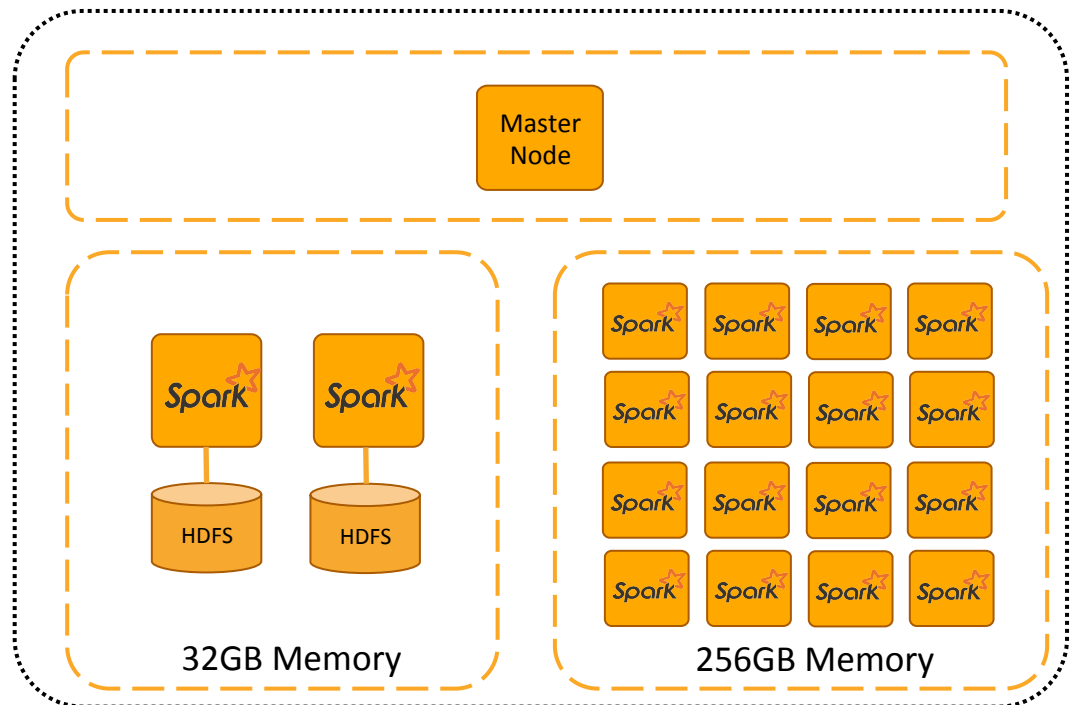
Scaling Spark on EMR

- 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

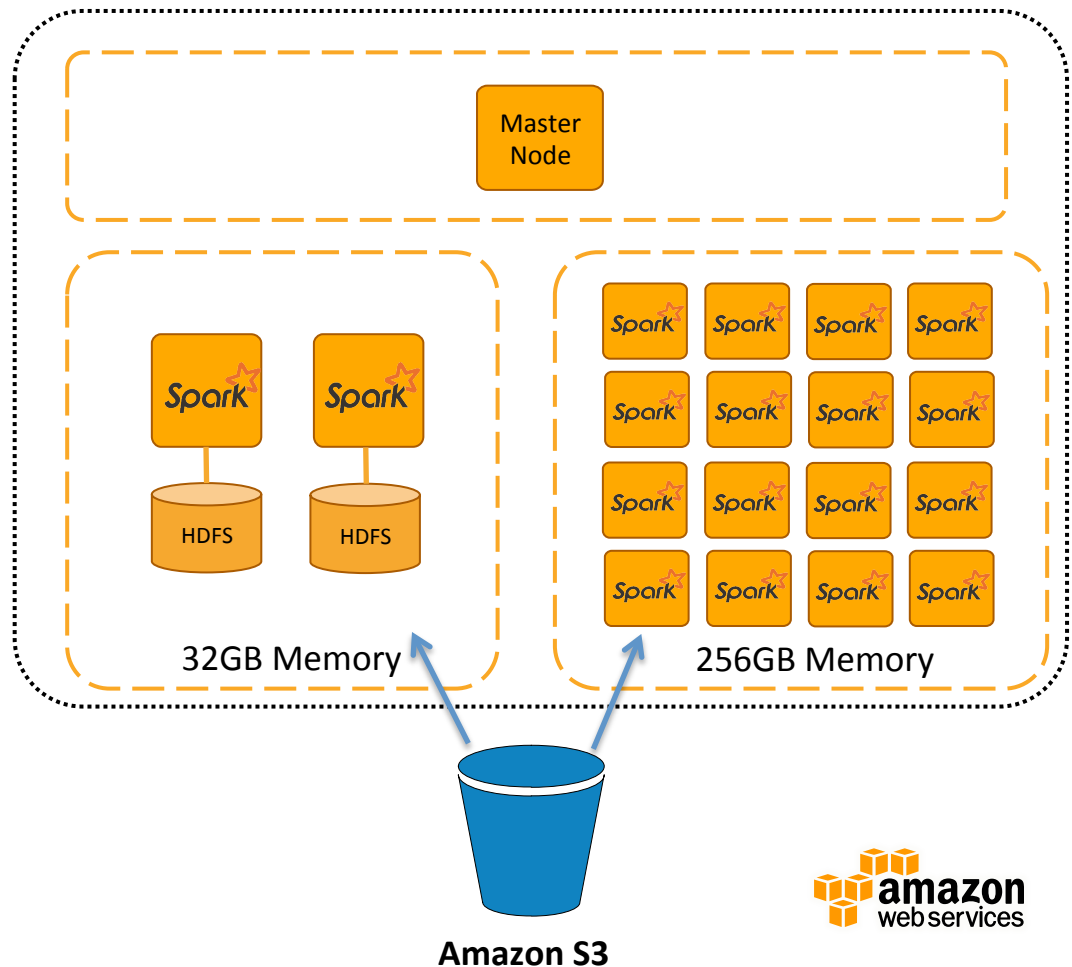
- Create RDDs from HDFS or Amazon S3 with:

`sc.textFile`

OR

`sc.sequenceFile`

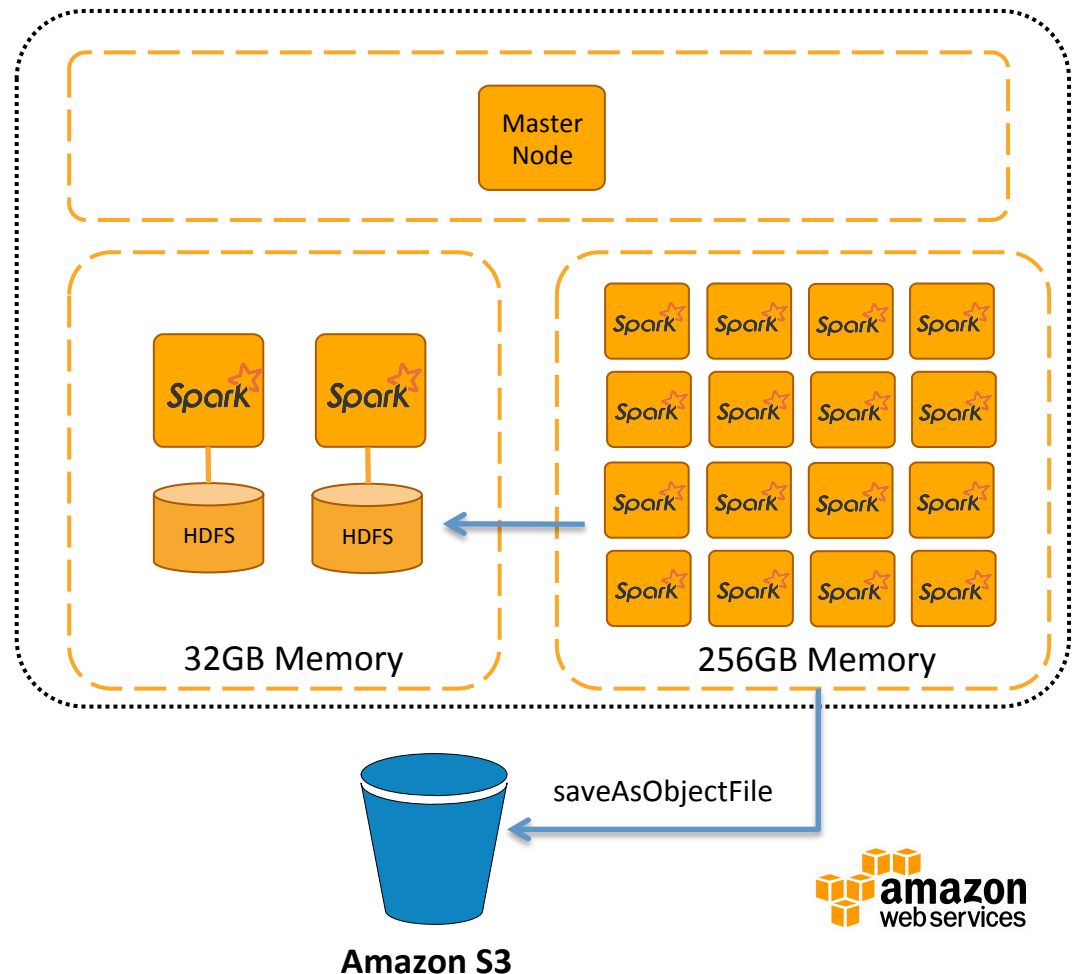
- Run Computation on RDDs



Scaling Spark on EMR

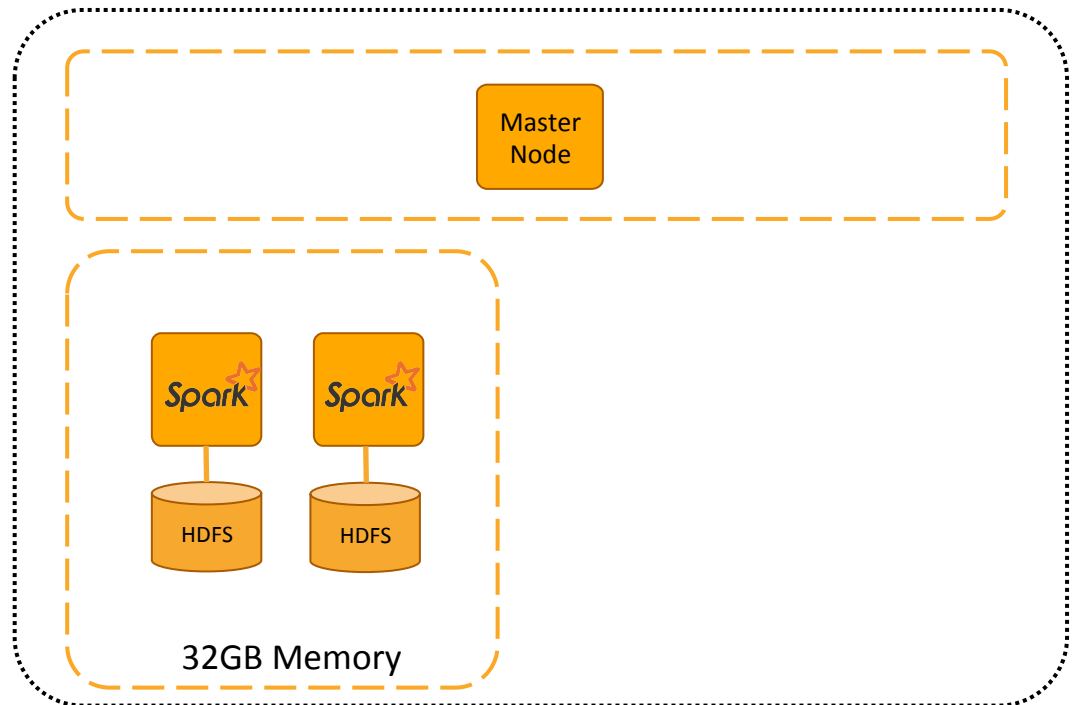
- Save the resulting RDDs to HDFS or S3 with:

`rdd.saveAsSequenceFile`
OR
`rdd.saveAsObjectFile`

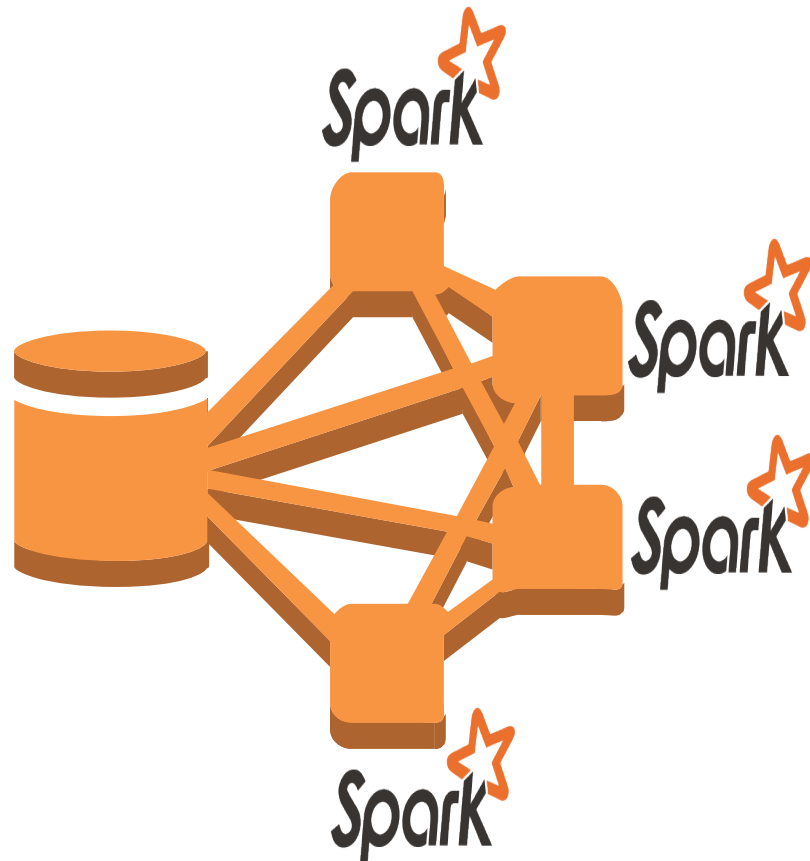


Scaling Spark on EMR

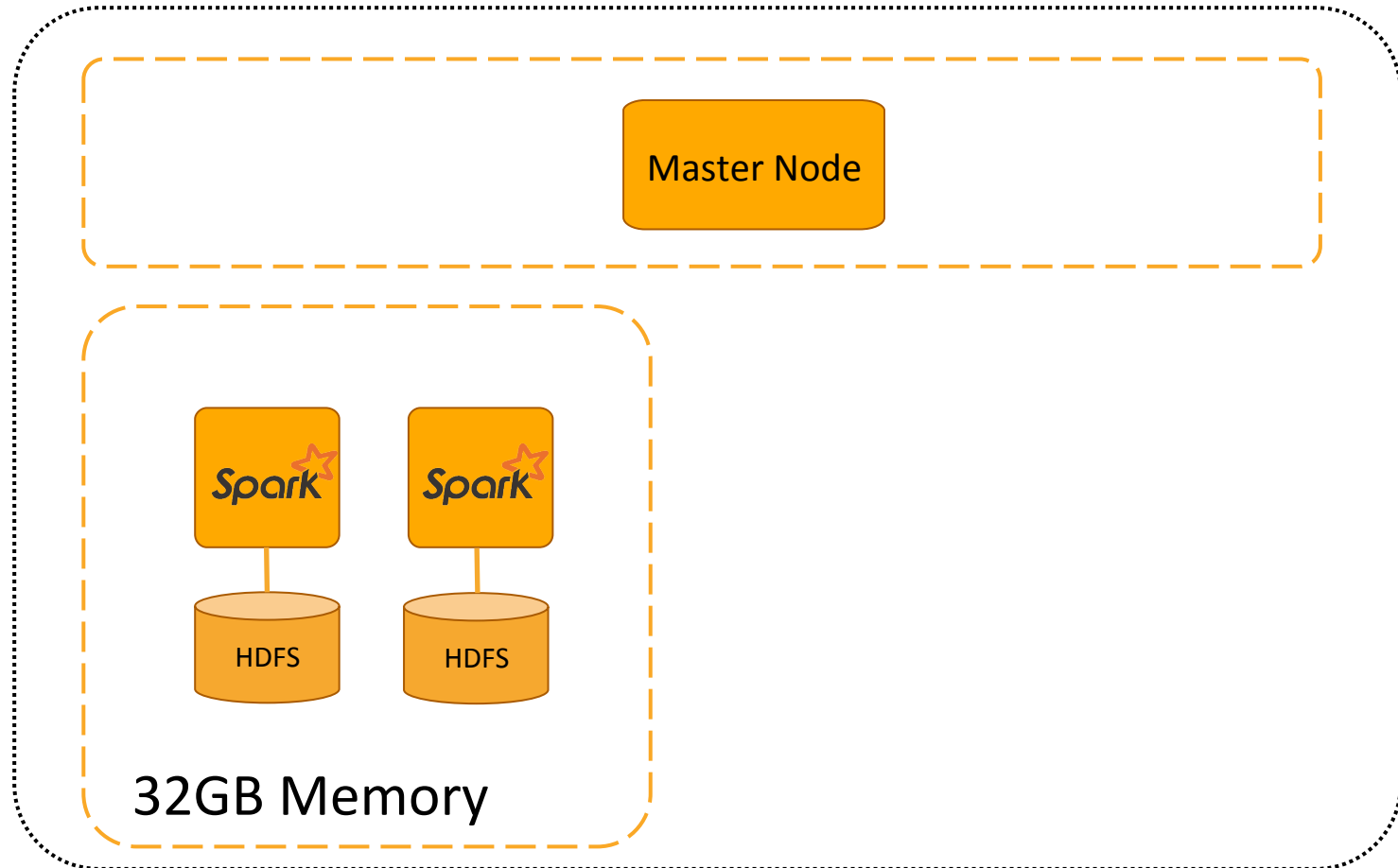
- Shutdown TaskNodes when your job is done



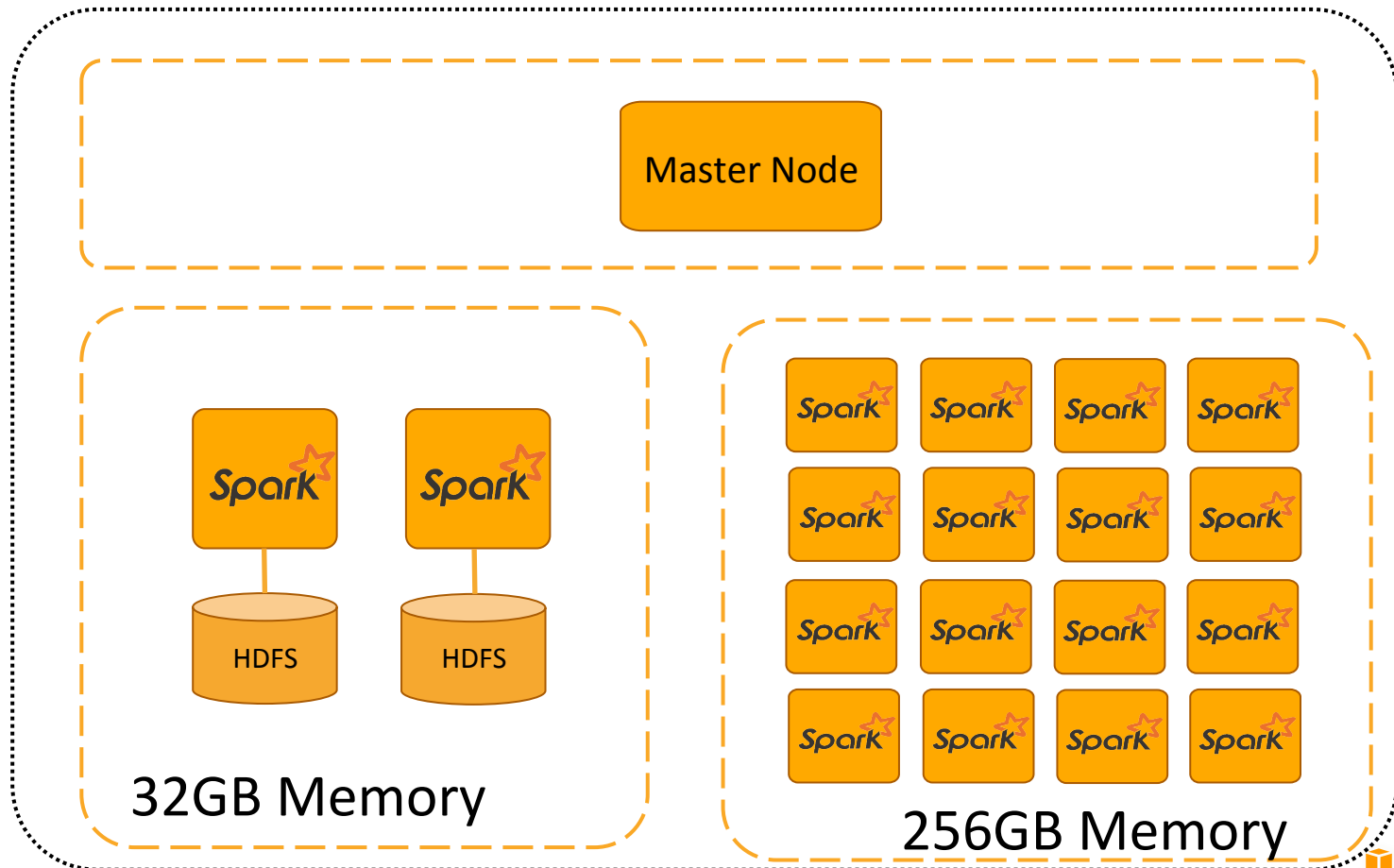
Elastic Spark With Amazon EMR



Autoscaling Spark



Autoscaling Spark



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

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

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



Services ▾

Edit ▾

Elastic MapReduce ▾

[Cluster List](#) > Cluster Details

Add step

Resize

Clone

Terminate

Cluster: Spark/Shark Cluster **Waiting** Waiting for steps to run

Master public DNS: ec2-107-20-0-141.compute-1.amazonaws.com

Tags: -- [View All / Edit](#)

Summary

ID: j-MGQ0H4L0JJKG

Creation date: 2014-06-18 18:02 (UTC-7)

Elapsed time: 20 minutes

Auto-terminate: No

Termination protection: Off [Change](#)

Configuration Details

AMI version: 2.4.2

Hadoop Amazon 1.0.3 distribution:

Applications: --

Log URI: s3://chayel-spark-logs/



Security/Network

Availability zone: us-east-1a

Subnet ID: --

Key name: keypair

EC2 role: --

Visible to all users: None [Change](#)

Hardware

Master: **Running** 1 m1.xlarge

Core: **Running** 999 m1.xlarge

Task: --

Lynx Interface

lynx http://localhost:9101

NameNode Logs

Cluster Summary

9 files and directories, 1 blocks = 10 total. Heap Size is 314 MB / 2 GB (15%)

Configured Capacity	: 1.6 PB
DFS Used	: 31.43 MB
Non DFS Used	: 0 KB
DFS Remaining	: 1.6 PB
DFS Used%	: 0 %
DFS Remaining%	: 100 %
Live Nodes	: 999
Dead Nodes	: 0
Decommissioning Nodes	: 0
Number of Under-Replicated Blocks	: 0

NameNode Storage:

Web Interface

NameNode 'ip-10-29-181-223.ec2.internal:9000'

Started: Mon Jun 30 14:28:55 UTC 2014
Version: 1.0.3, r
Compiled: Wed Oct 2 12:17:08 PDT 2013 by Elastic MapReduce
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)
[Namenode Logs](#)

Cluster Summary

9 files and directories, 1 blocks = 10 total. Heap Size is 1.23 GB / 10.67 GB (11%)

Configured Capacity	:	1.6 PB
DFS Used	:	31.43 MB
Non DFS Used	:	0 KB
DFS Remaining	:	1.6 PB
DFS Used%	:	0 %
DFS Remaining%	:	100 %
Live Nodes	:	999
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

Spark UI

Spark Spark Master at spark://10.29.181.223:7077

URL: spark://10.29.181.223:7077
Workers: 999
Cores: 3996 Total, 3996 Used
Memory: 13.3 TB Total, 9.8 TB Used
Applications: 1 Running, 0 Completed

Workers

Id	Address
worker-20140630142856-ip-10-16-139-142.ec2.internal-35690	ip-10-16-139-142.ec2.internal:7077
worker-20140630142856-ip-10-183-44-115.ec2.internal-49815	ip-10-183-44-115.ec2.internal:7077
worker-20140630142858-ip-10-146-177-207.ec2.internal-37928	ip-10-146-177-207.ec2.internal:7077
worker-20140630142858-ip-10-231-18-32.ec2.internal-53245	ip-10-231-18-32.ec2.internal:7077
worker-20140630142900-ip-10-146-248-171.ec2.internal-58652	ip-10-146-248-171.ec2.internal:7077
worker-20140630142900-ip-10-65-132-221.ec2.internal-43700	ip-10-65-132-221.ec2.internal:7077
worker-20140630142901-ip-10-146-249-189.ec2.internal-41691	ip-10-146-249-189.ec2.internal:7077

Dataset

- Wikipedia article traffic statistics
 - 4.5 TB
 - 104 Billion records
- Stored in Amazon S3
 - `s3://bigdata-spark-demo/wikistats/`

Dataset

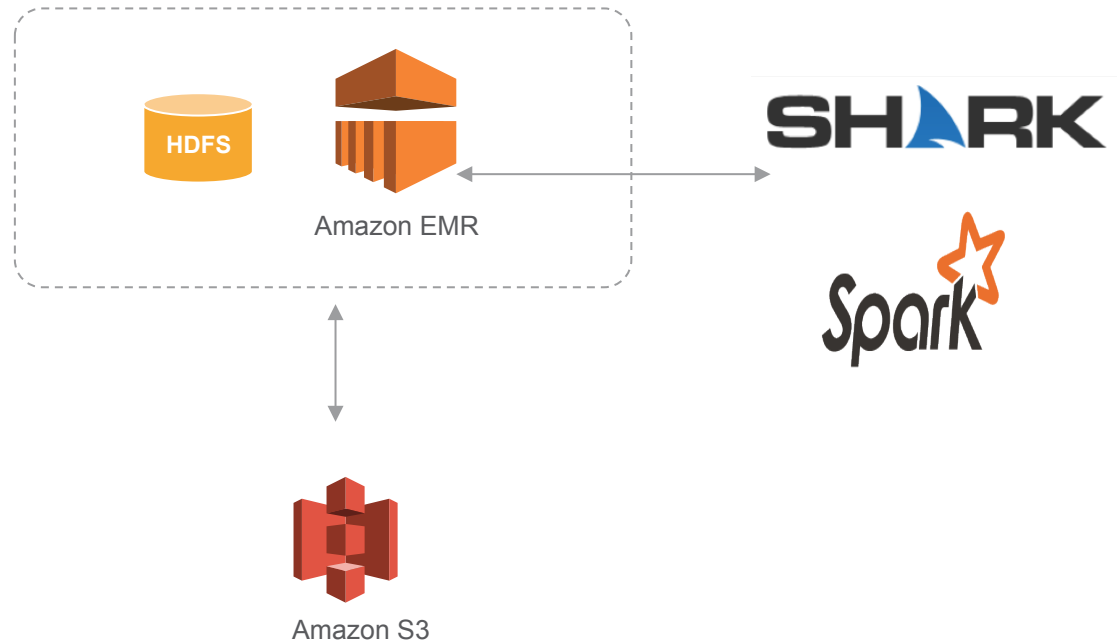
- File structure (pagecount-DATE-HOUR.gz)
- Period: Dec-2007 to Feb-2014
- Format of File (tsv)
- Fields

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

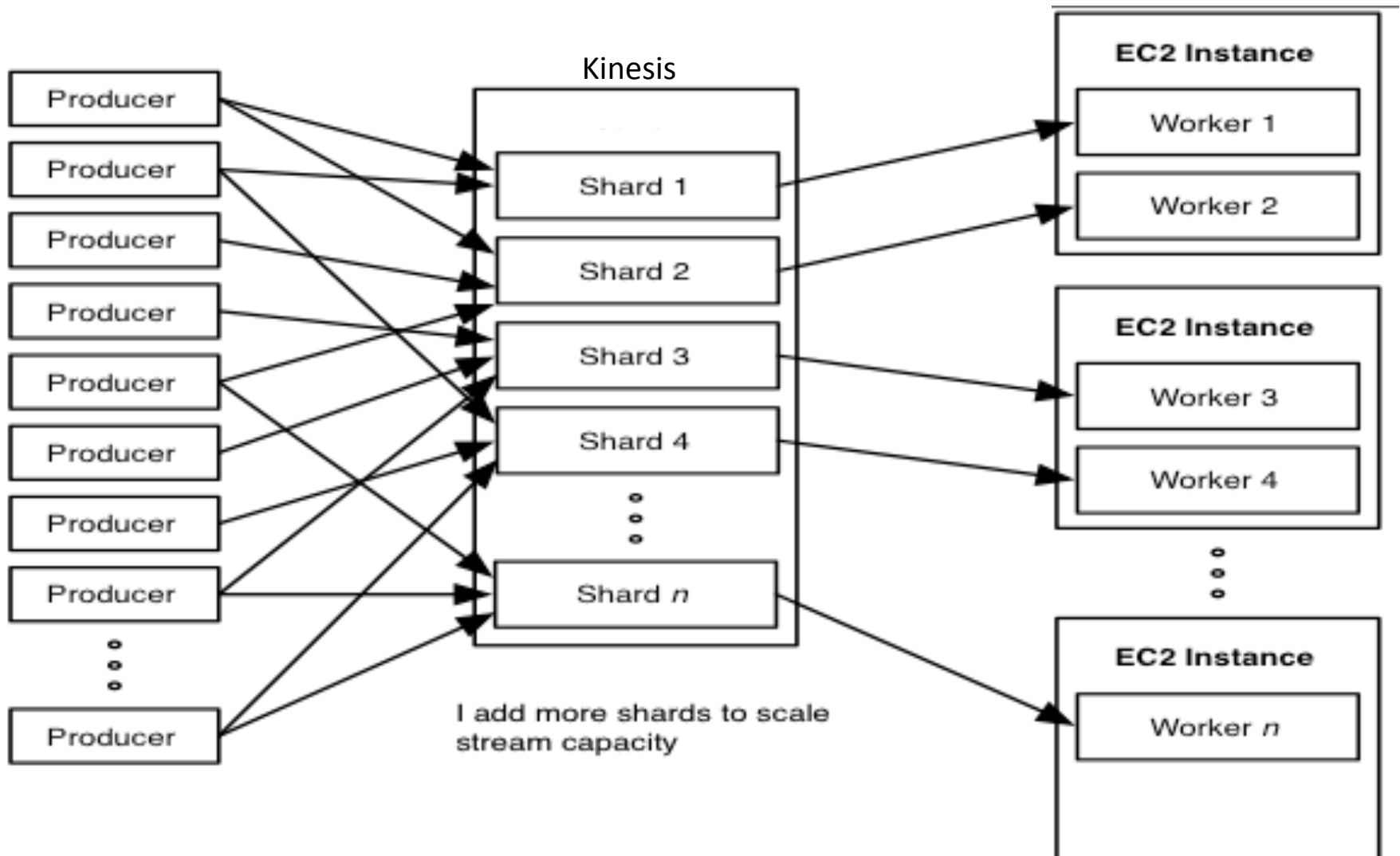
Spark Streaming and Amazon Kinesis



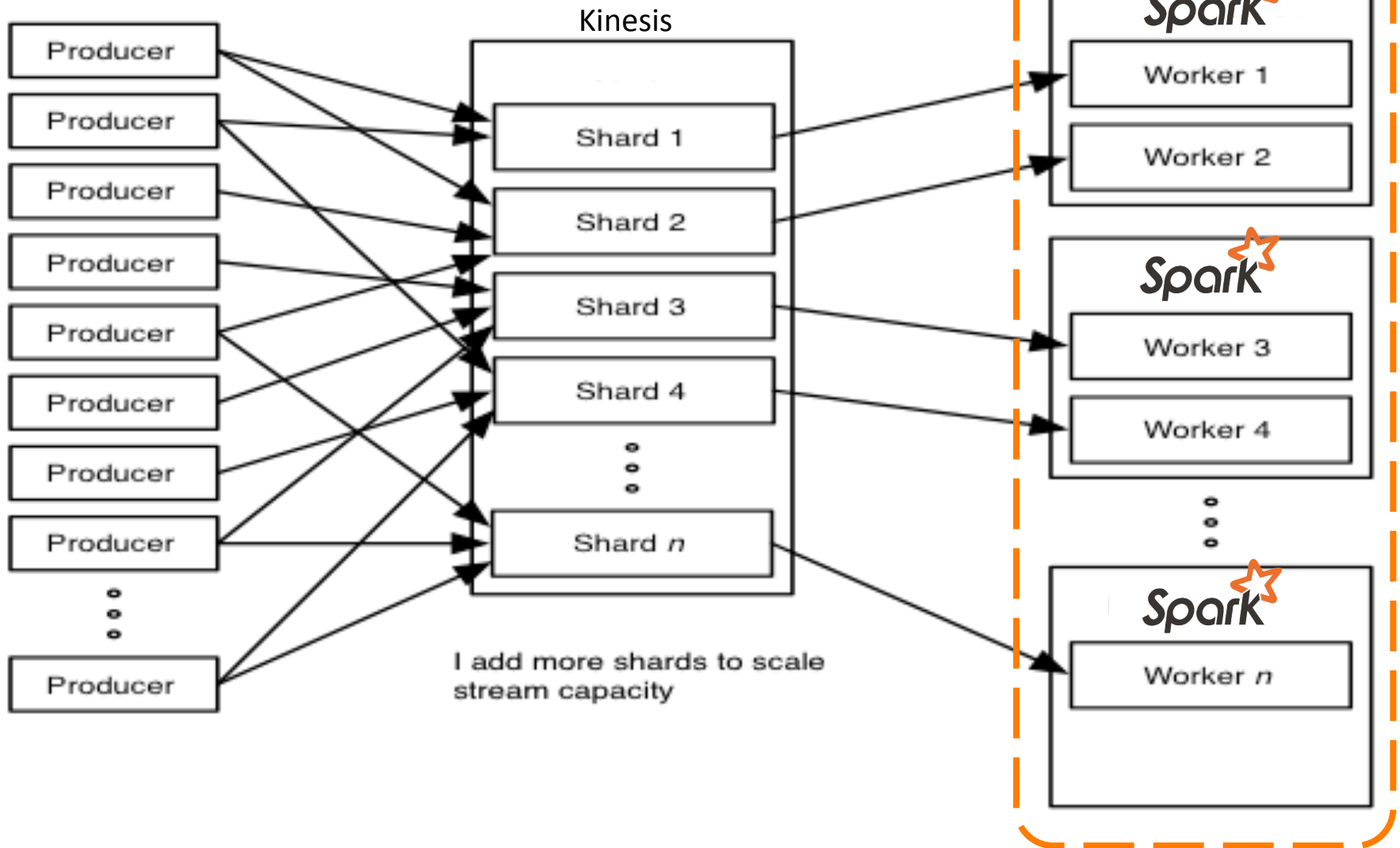
Amazon Kinesis

- **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>