Amazon Elastic Map Reduce: the concepts

Julien Simon, AI Evangelist, EMEA
@julsimon

What to expect

- Amazon EMR
- Amazon S3 as HDFS
- Core node and Task nodes
- Elastic clusters
- Beyond Map Reduce: Spark
- Q&A

What Hadoop is good at

Semi-structured/unstructured data

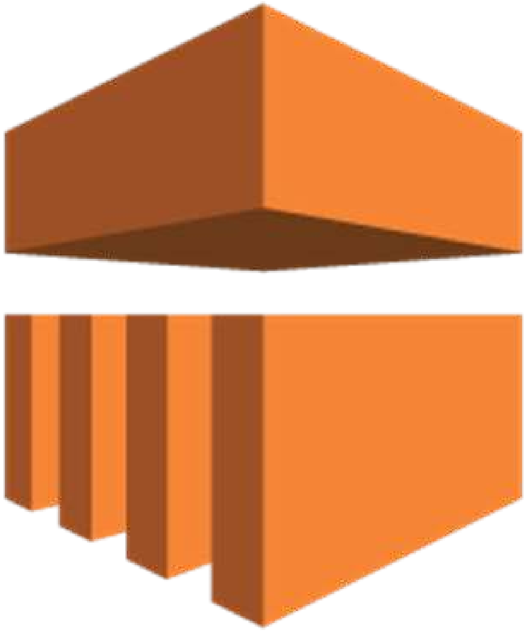
Disparate Data Sets

ETL at scale

Batch Analytics

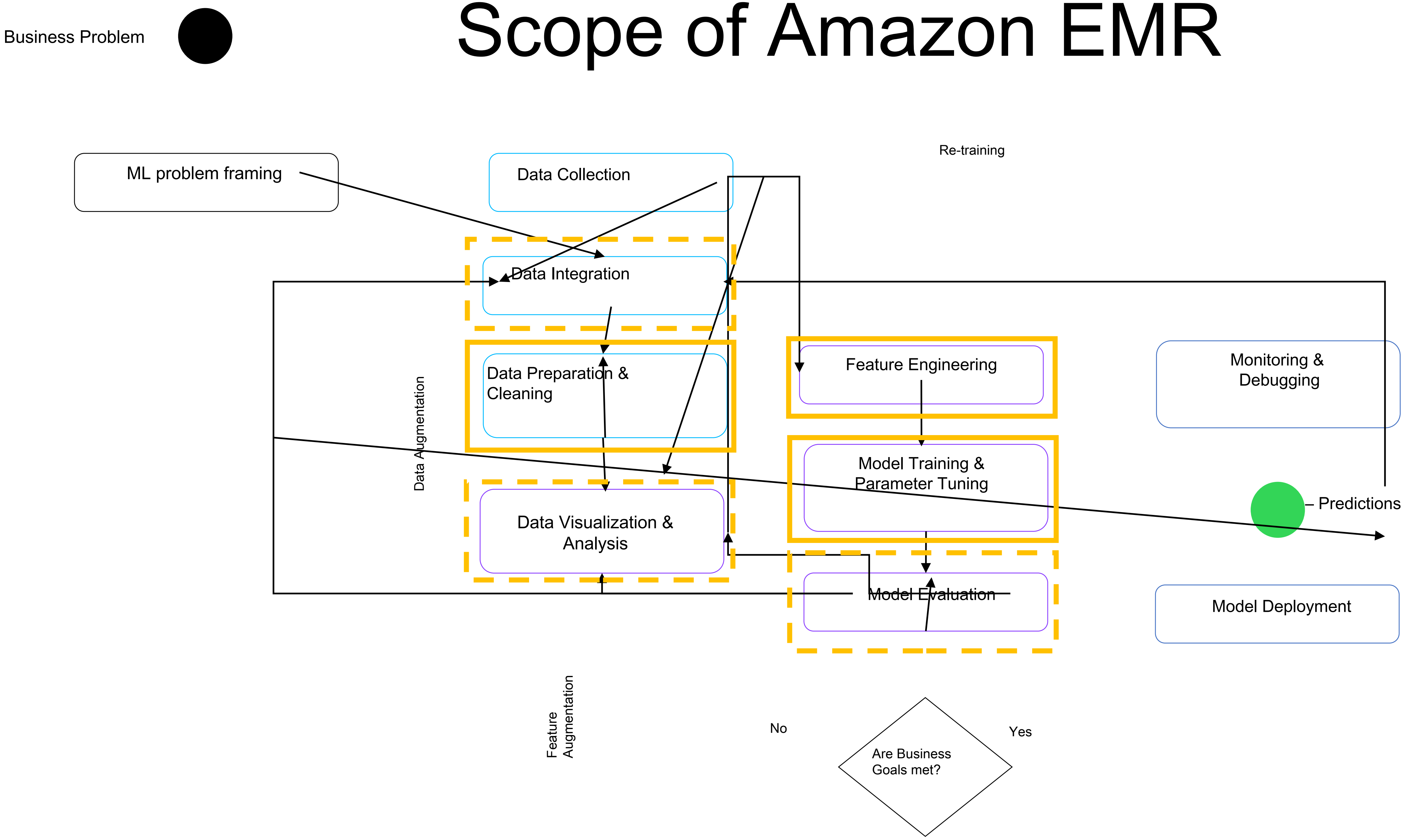
Log Processing & Aggregation

Amazon EMR – Hadoop, Spark, Presto in the Cloud



- Managed platform
- Launch a cluster in minutes
- Leverage the elasticity of the cloud
- Baked in security features
- Pay by the hour and save with Spot
- Flexibility to customize

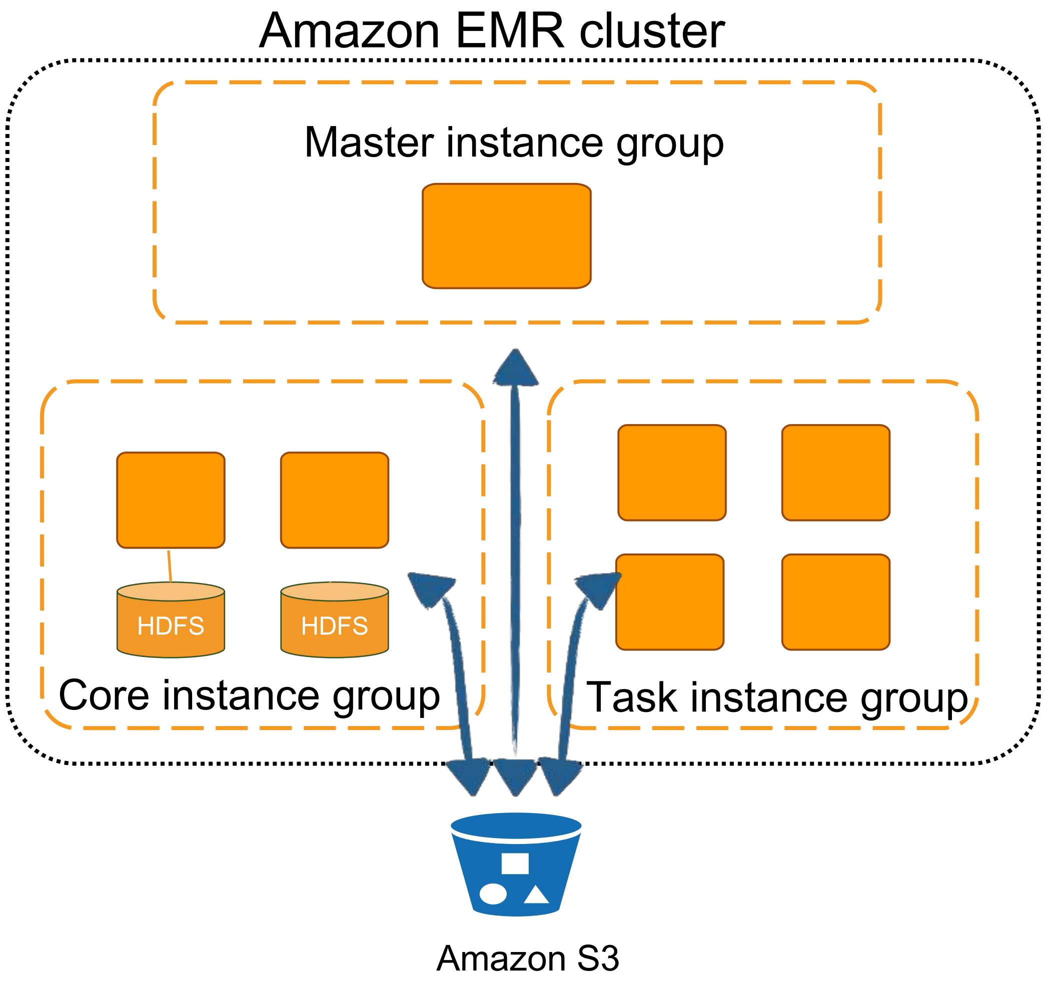
Scope of Amazon EMR



Amazon S3 as HDFS

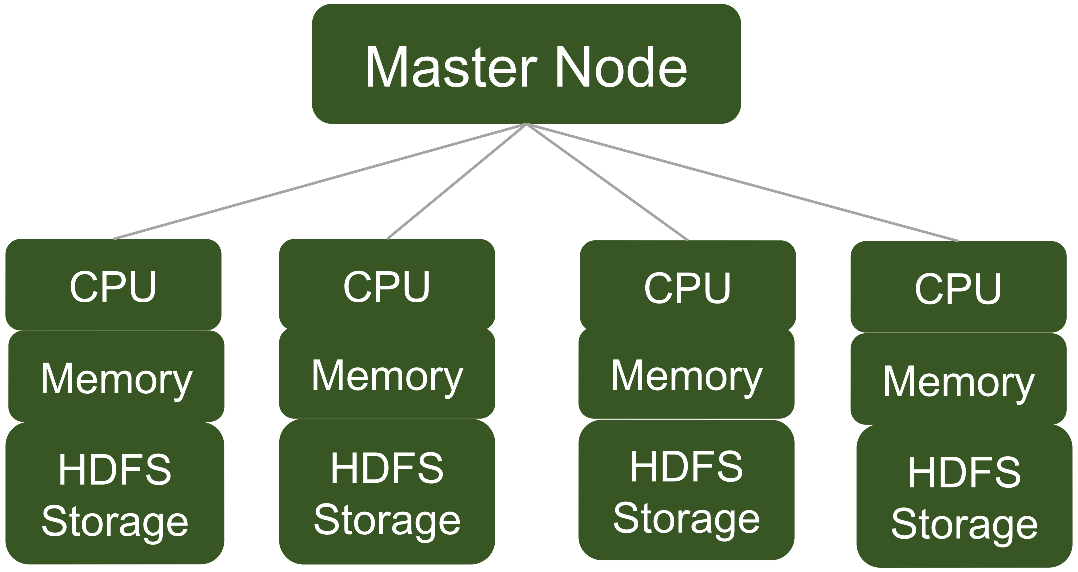
Amazon S3 as HDFS

- Use Amazon S3 as your permanent data store
- HDFS for temporary storage data between jobs
- No additional step to copy data to HDFS



Understanding Decoupled Storage & Compute

Old Clustering / Localized Model



Master Node

CPU
Memory
HDFS Storage

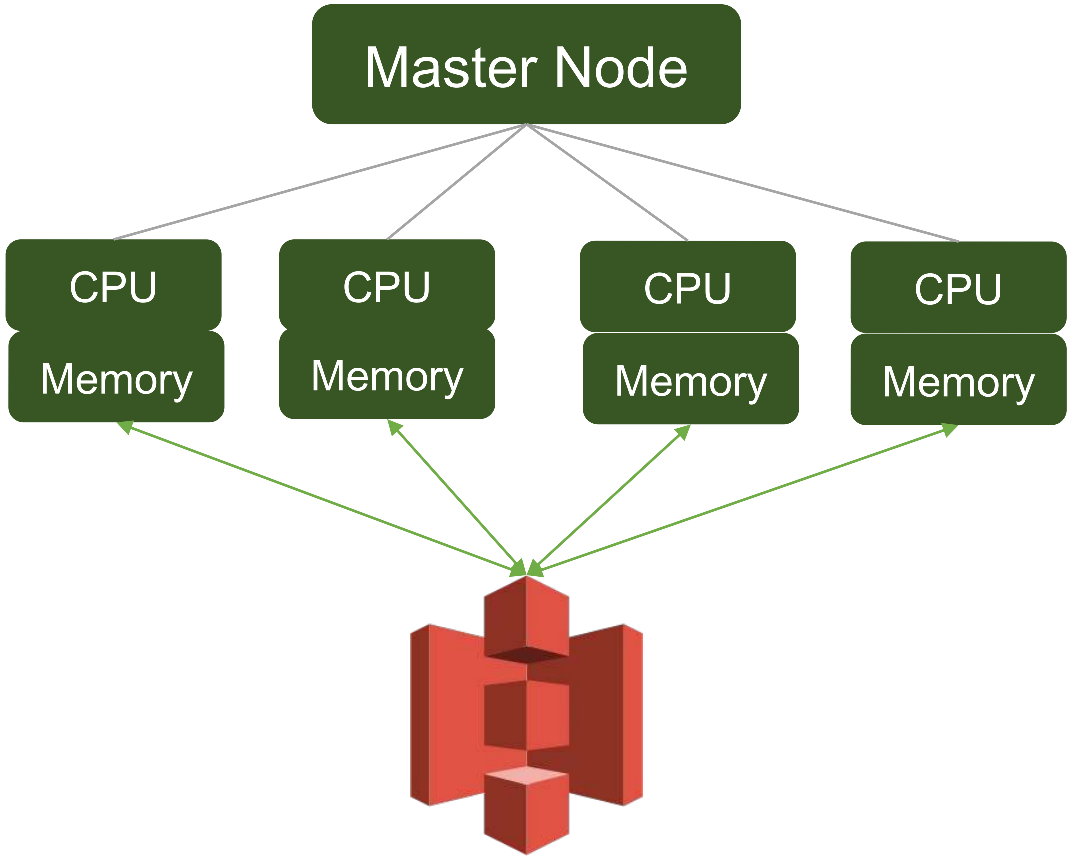
CPU
Memory
HDFS Storage

CPU
Memory
HDFS Storage

CPU
Memory
HDFS Storage

HDFS = 3X Replication
500 TB Dataset = 1.5 PB cluster with replication

Amazon EMR Decoupled Model



Master Node

CPU
Memory

CPU
Memory

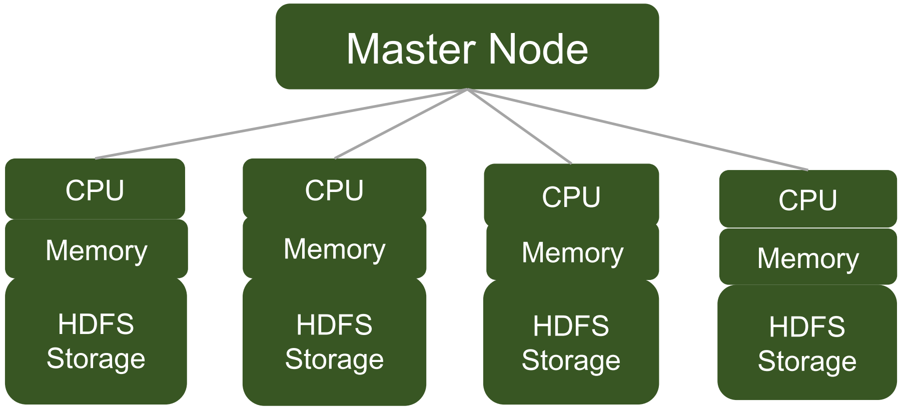
CPU
Memory

CPU
Memory

S3 as Streaming HDFS via EMRFS

Understanding Decoupled Storage & Compute

Hadoop on EC2



Master Node

CPU
Memory
HDFS Storage

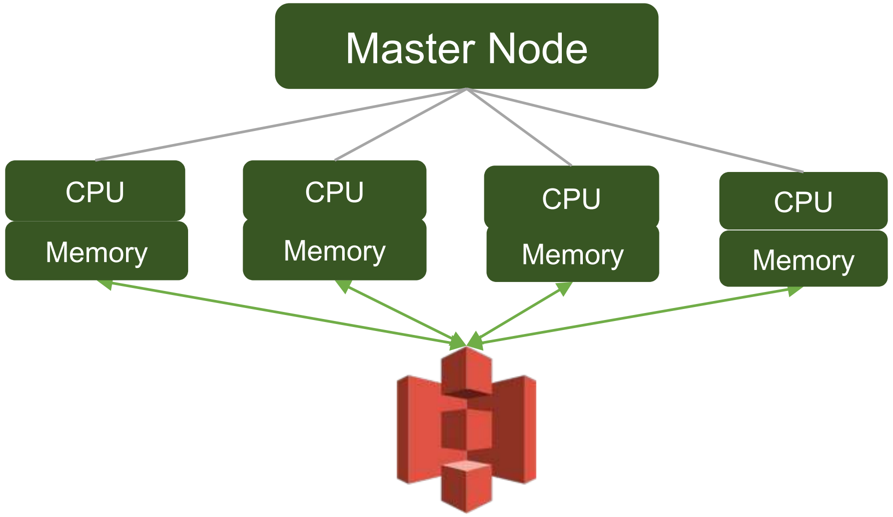
CPU
Memory
HDFS Storage

CPU
Memory
HDFS Storage

CPU
Memory
HDFS Storage

- 3X HDFS Replication of Data across Nodes
- Protected against data loss from node failure
- Single AZ replication
- Paying for storage units via EC2 ephemeral drives or EBS volumes
- Data stored locally so cluster = 24x7 by default

EMR



Master Node

CPU
Memory

CPU
Memory

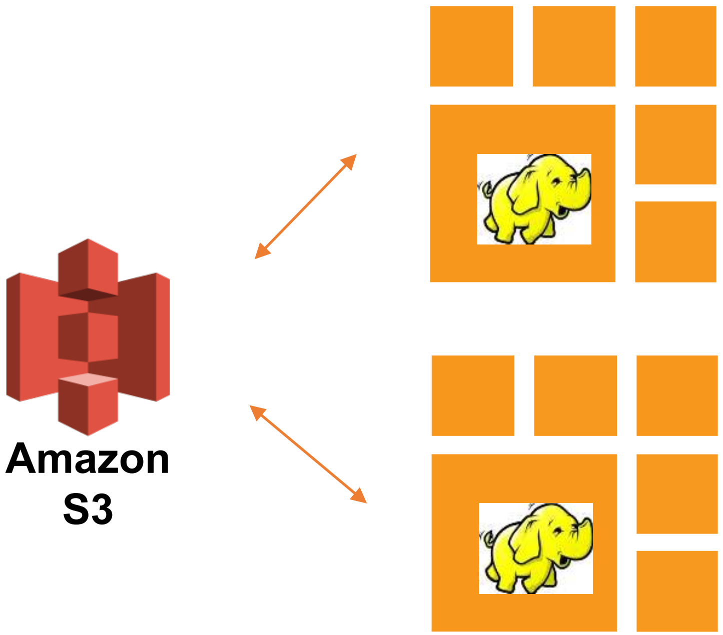
CPU
Memory

CPU
Memory

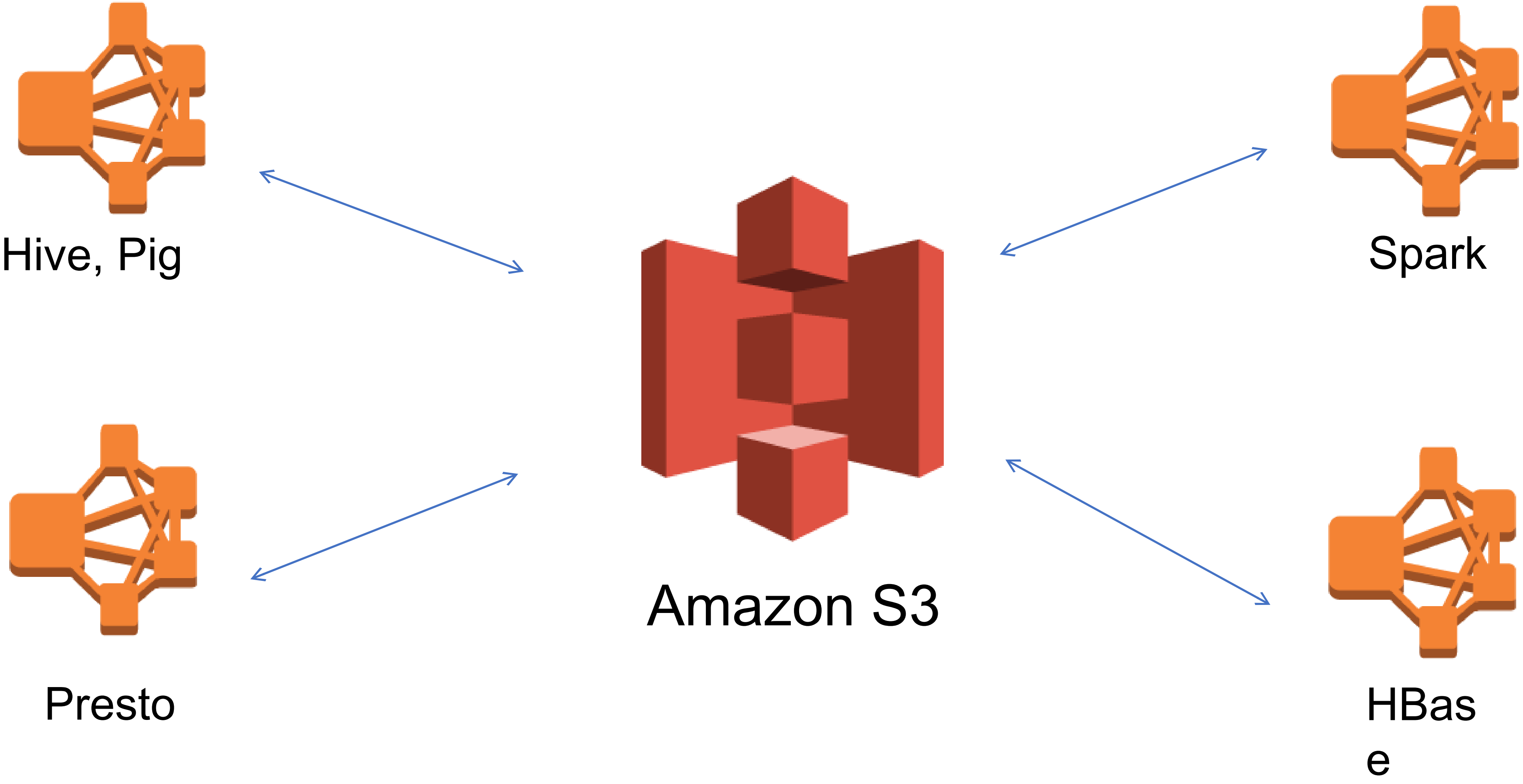
- Native S3 multi-AZ replication
- Protected against data loss from node failure, cluster failure or AZ failure
- Multiple physical facility replication
- Paying for storage units via S3 (cheap!!!)
- Spin clusters up and down or turn off!
- Stream multiple EMR clusters from same dataset in S3

Amazon S3 as your persistent data store

- Designed for 99.999999999% durability
- Separate compute and storage
- Resize and shut down clusters with no data loss
- Point multiple clusters at the same data in S3
- Easily evolve your analytics infrastructure as technology evolves



EMR with Amazon S3



Hive, Pig

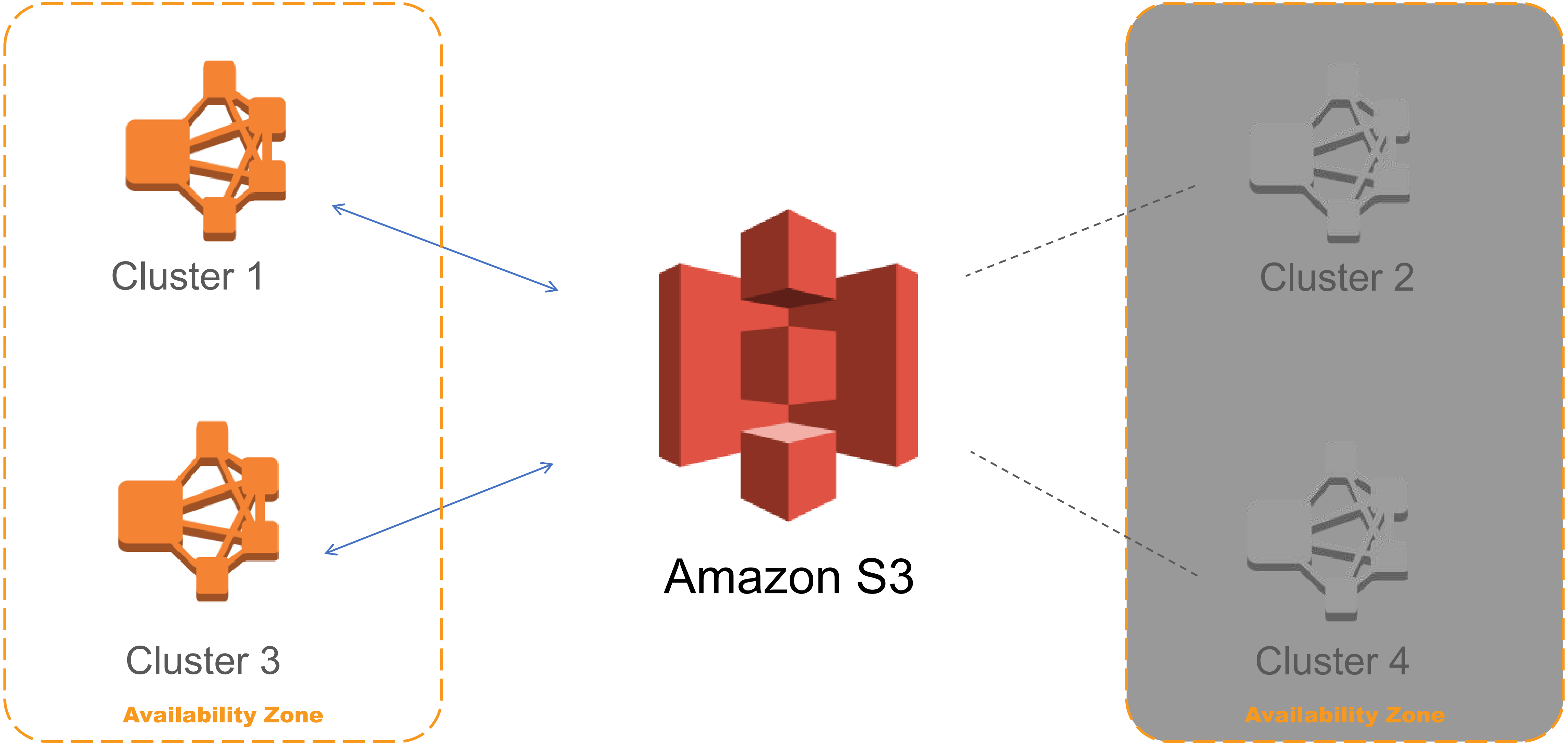
Presto

Amazon S3

Spark

HBas
e

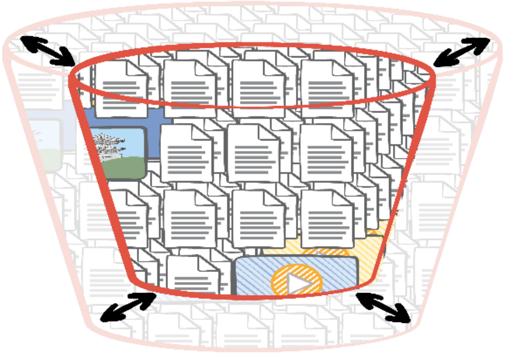
Disaster Recovery built in



Benefits: Amazon S3 as HDFS

- No need to scale HDFS
  - Capacity?
  - Replication?

- Amazon S3 scales with your data
  - Both in IOPS and storage
  - Your data can grow independent of CPU and RAM



Benefits: Amazon S3 as HDFS

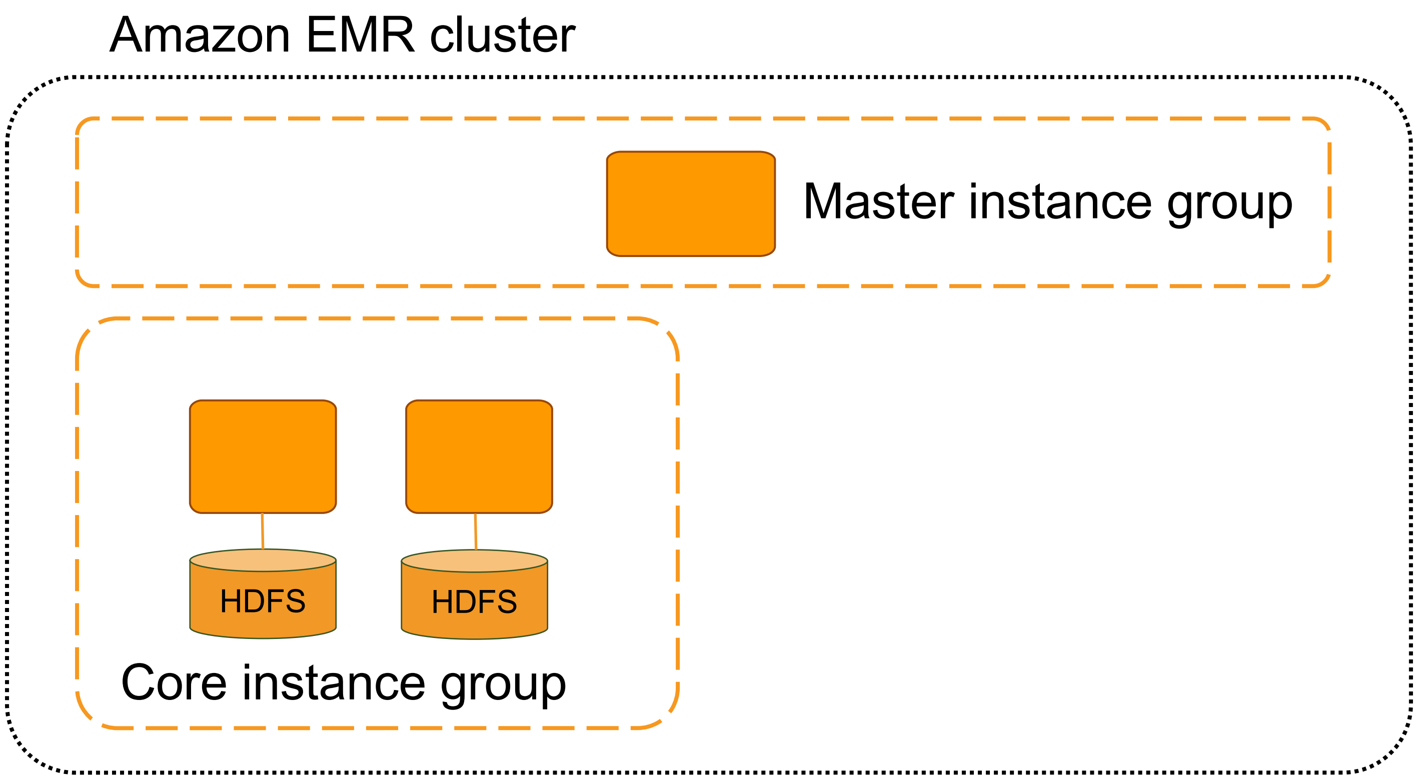
- Take advantage of S3 features
  - Server-side encryption
  - Lifecycle policies
  - Versioning to protect against corruption
- Build elastic clusters
  - Add nodes to read from Amazon S3
  - Remove nodes with data safe on Amazon S3

Core nodes and task nodes

Amazon EMR Core nodes

Run TaskTrackers (Compute)

Run DataNode (HDFS)

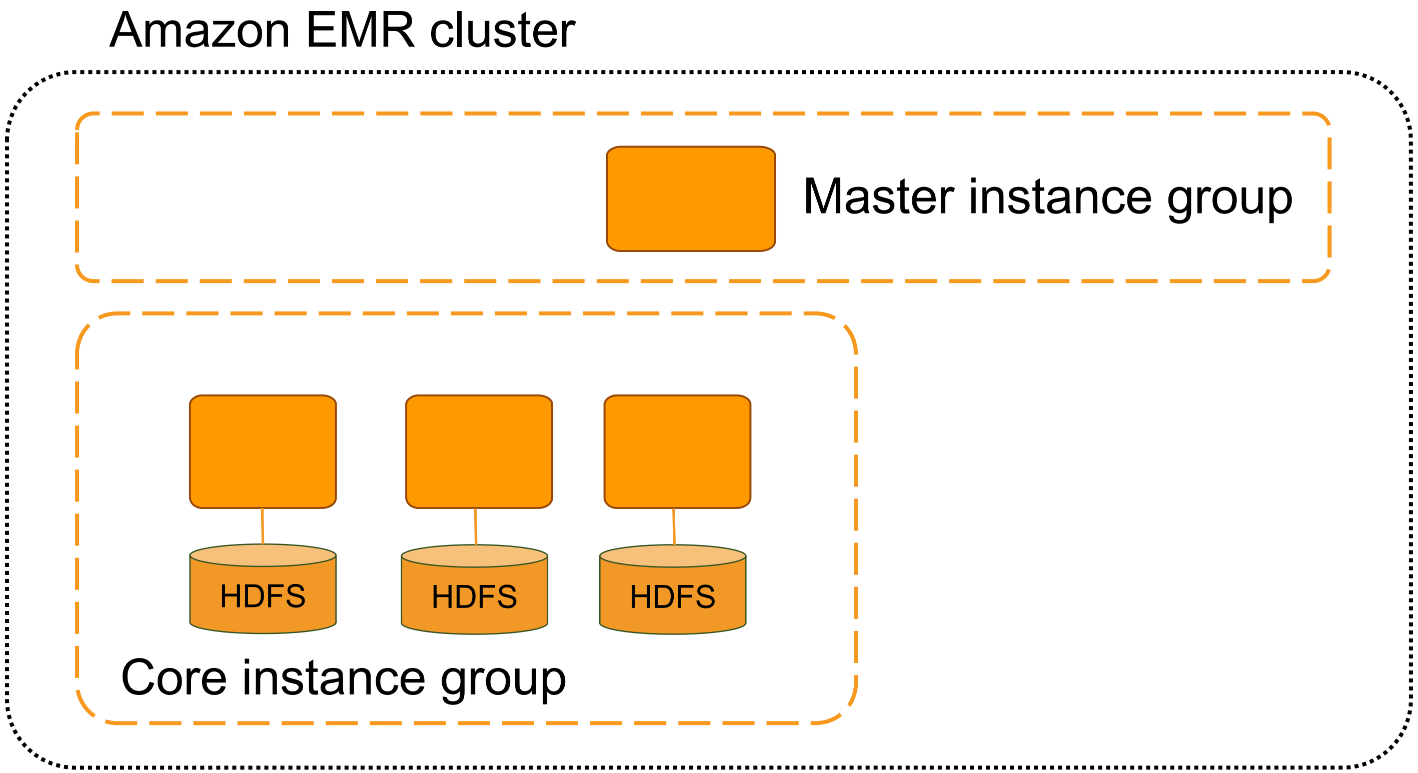


Amazon EMR Core nodes

Can add core nodes

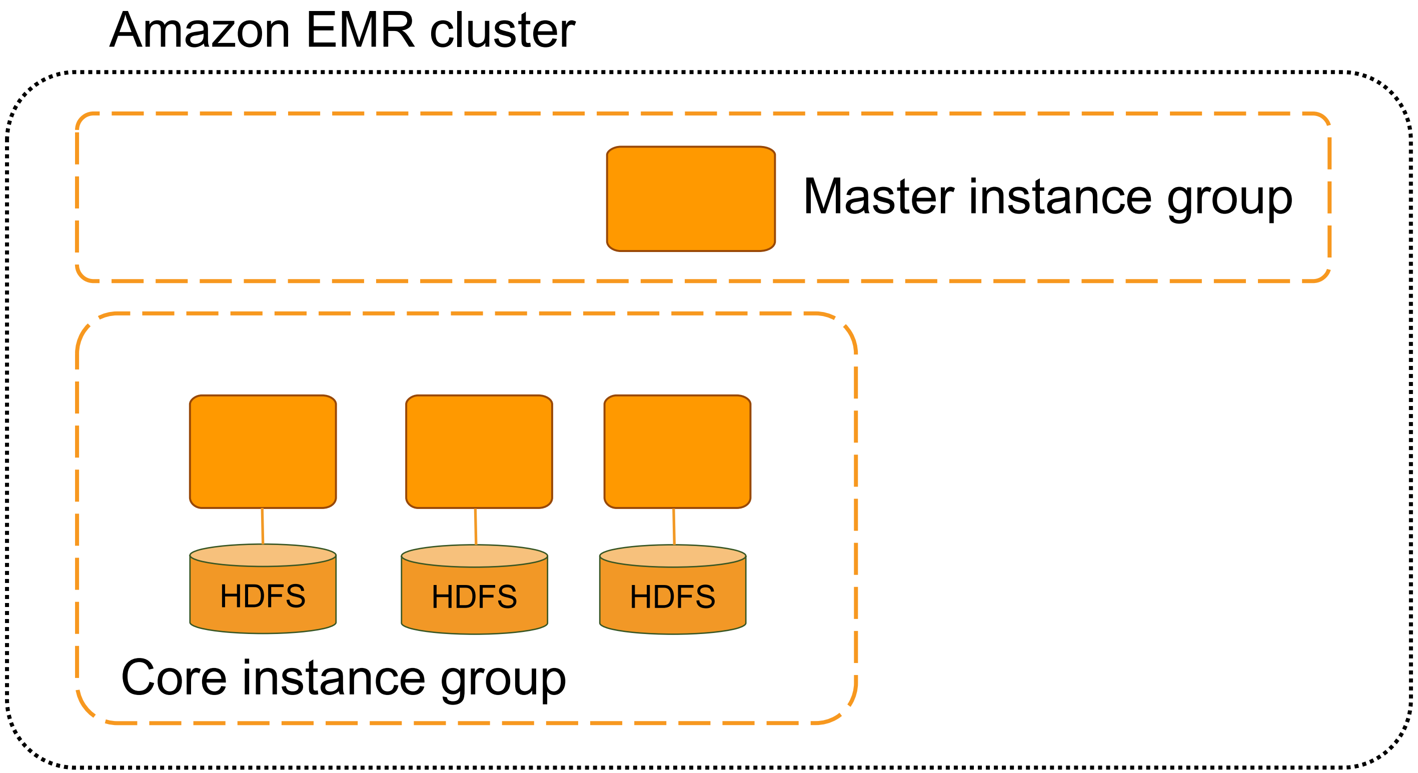
More HDFS space

More CPU/memory



Amazon EMR Core nodes

Can’t remove core nodes because of HDFS

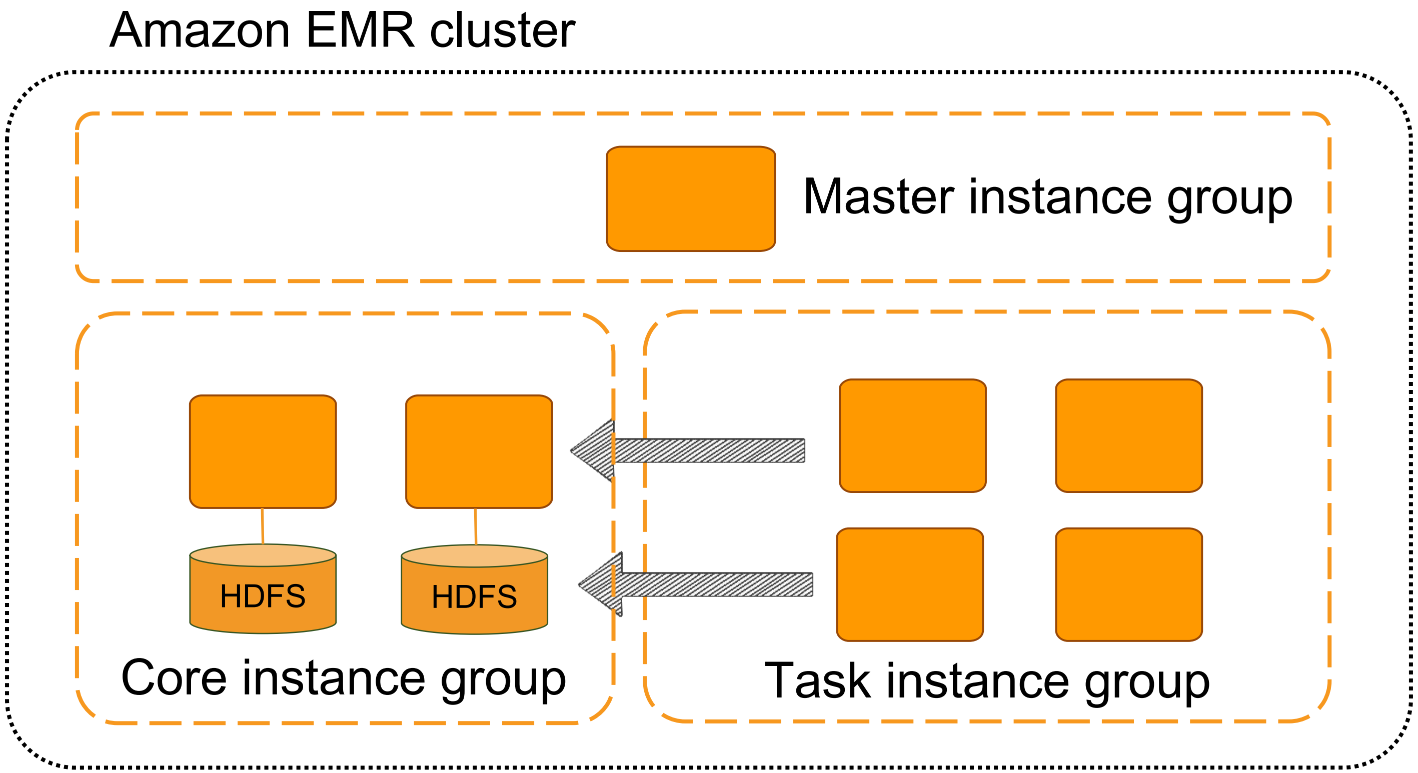


Amazon EMR Task nodes

Run TaskTrackers

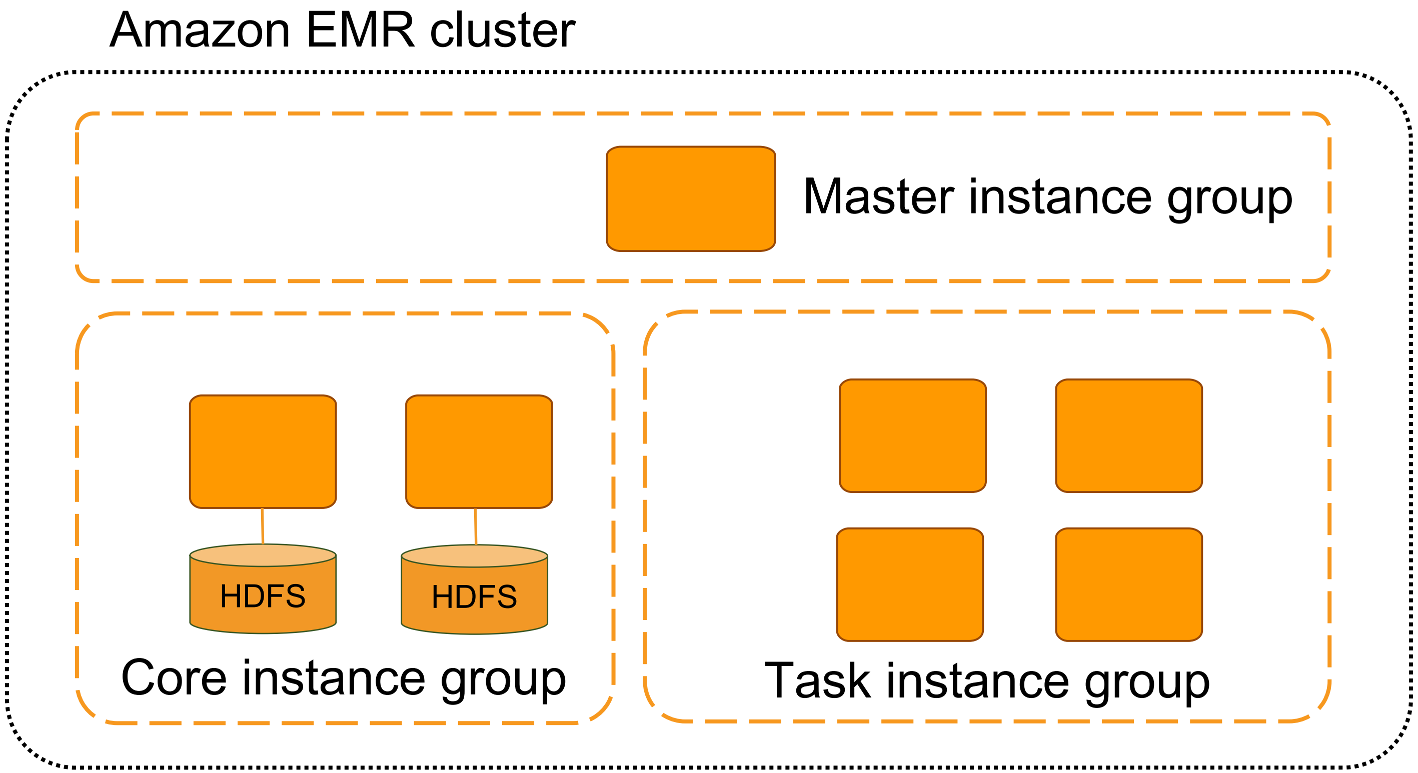
No HDFS

Reads from core nodes



Amazon EMR Task nodes

Can add task nodes

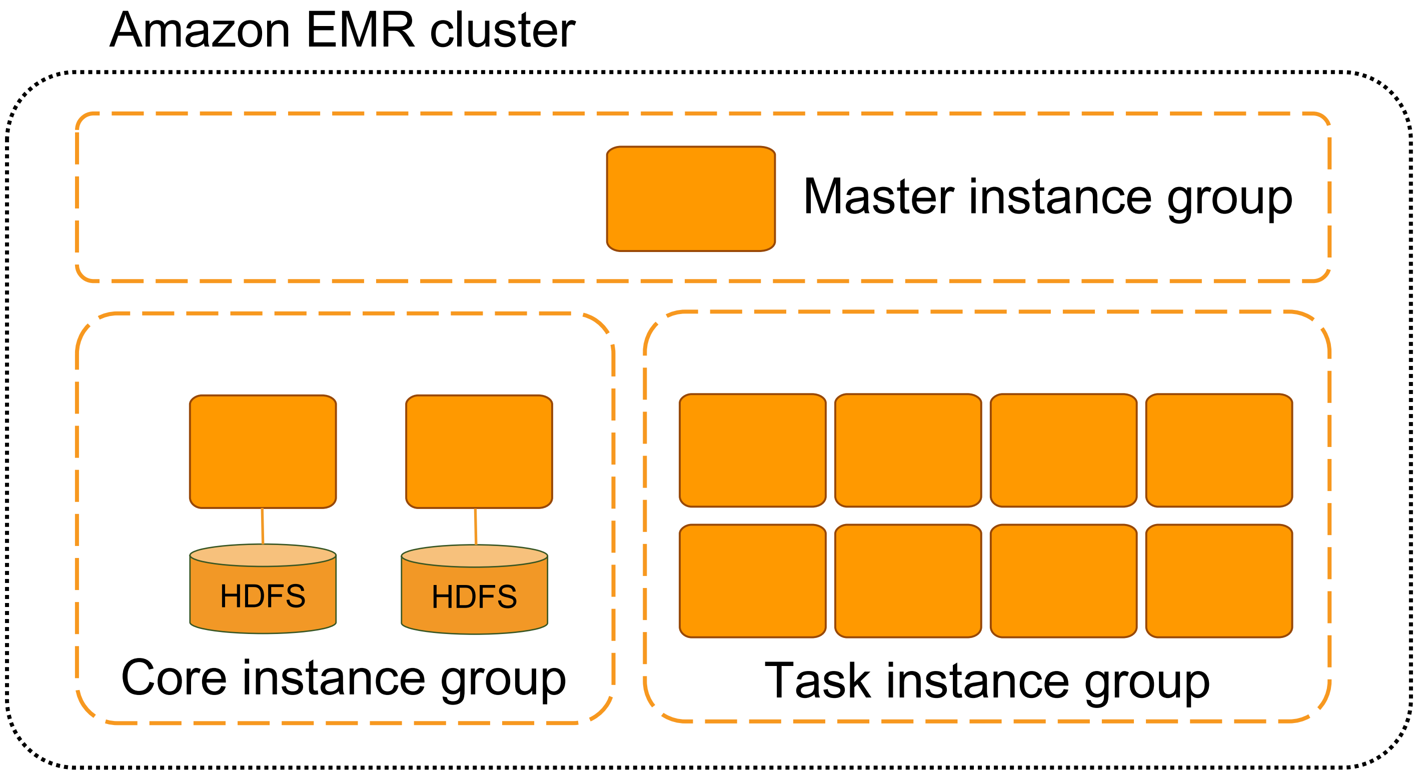


Amazon EMR Task nodes

More CPU power

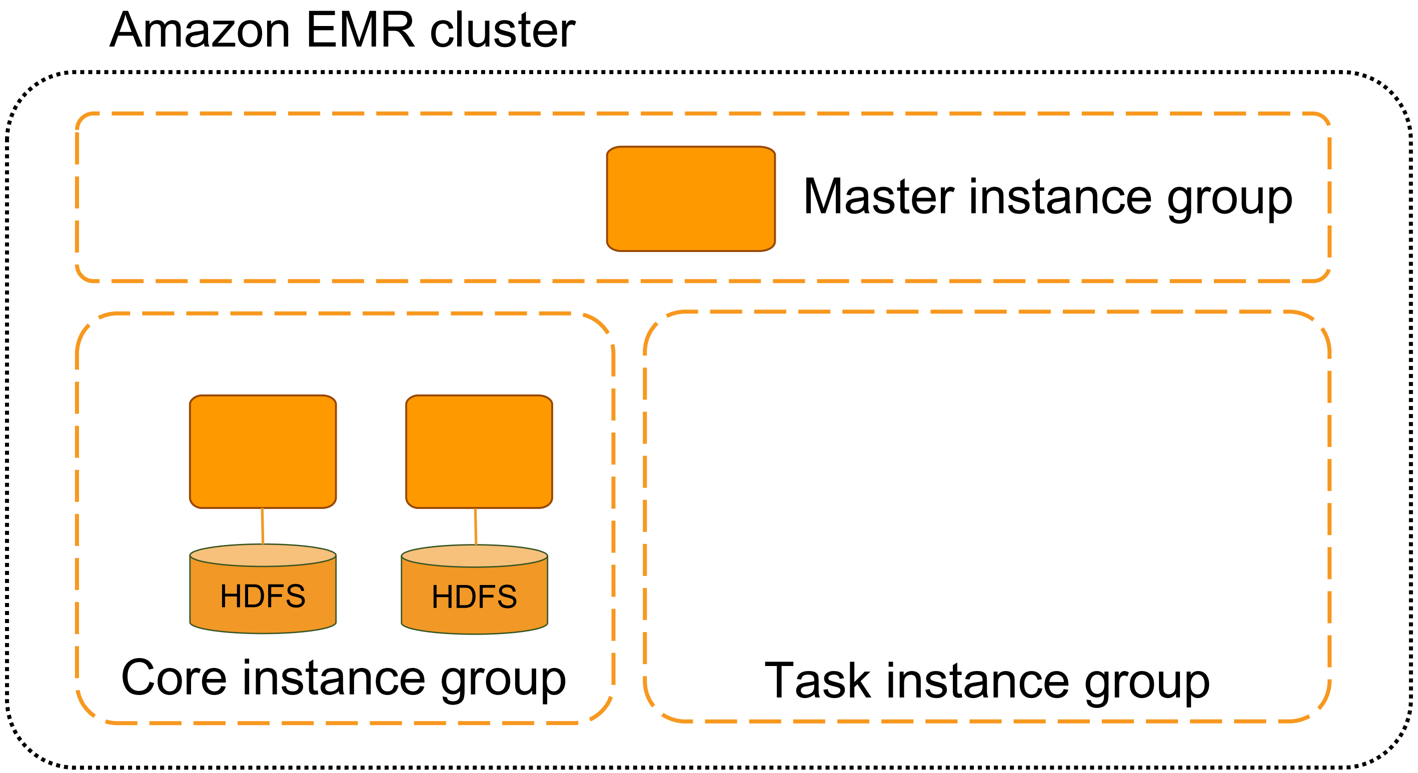
More memory

Jobs done faster



Amazon EMR Task nodes

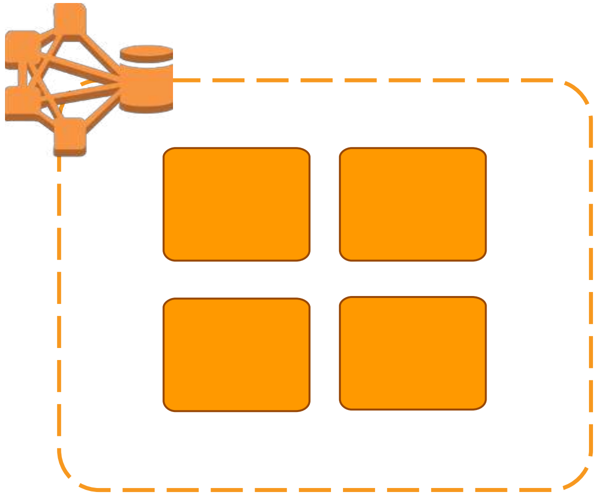
You can remove task nodes when processing is completed



Elastic clusters

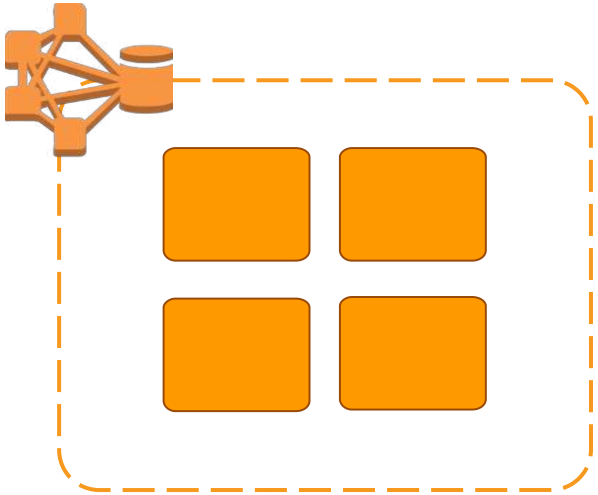
Elastic Clusters

1. Start cluster with certain number of nodes



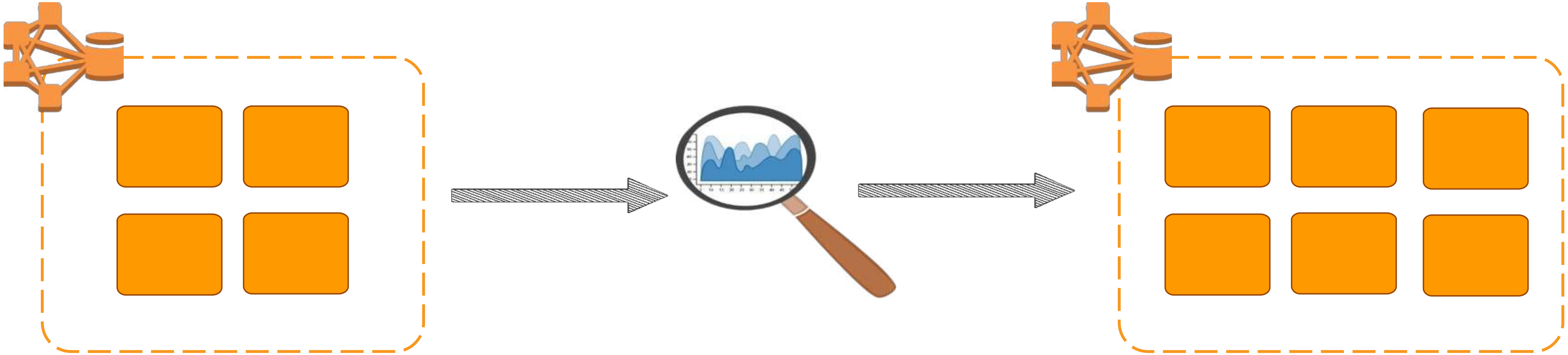
Elastic Clusters

2. Set Auto-Scaling policies in CloudWatch

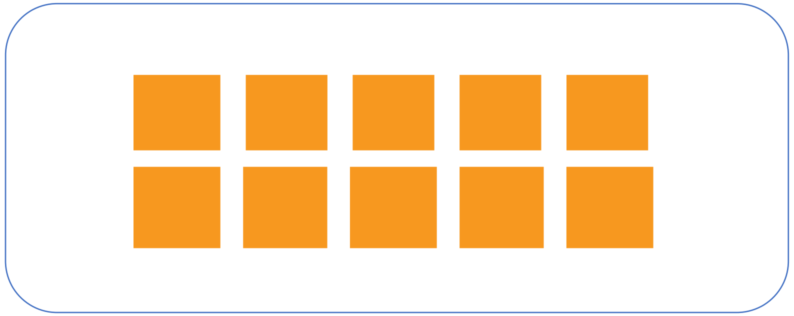


Elastic Clusters

3. Leverage EMR Auto-Scaling to add nodes or gracefully remove nodes based on CloudWatch policies



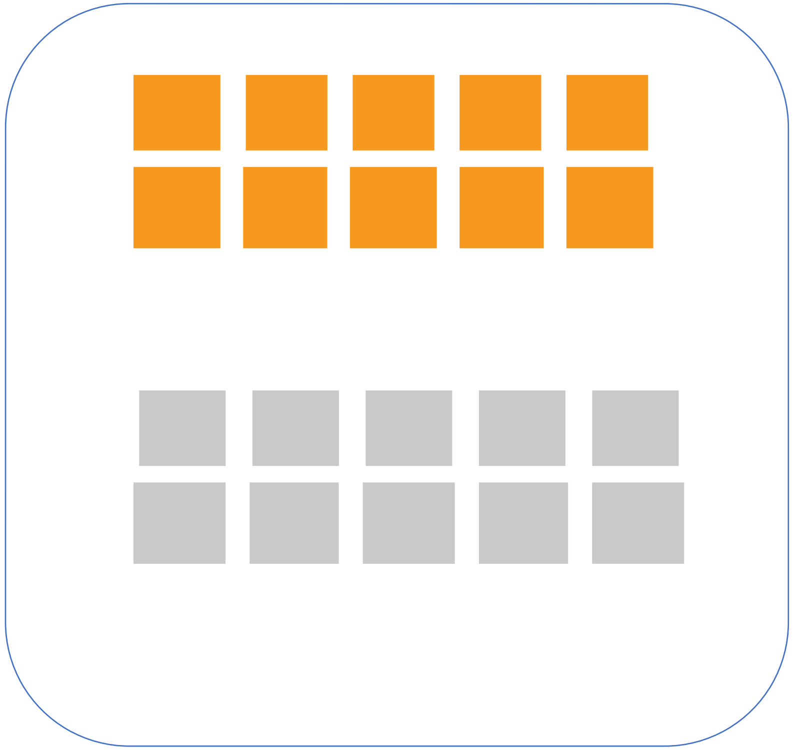
Scale up with Spot instances



10 node cluster running for 14 hours

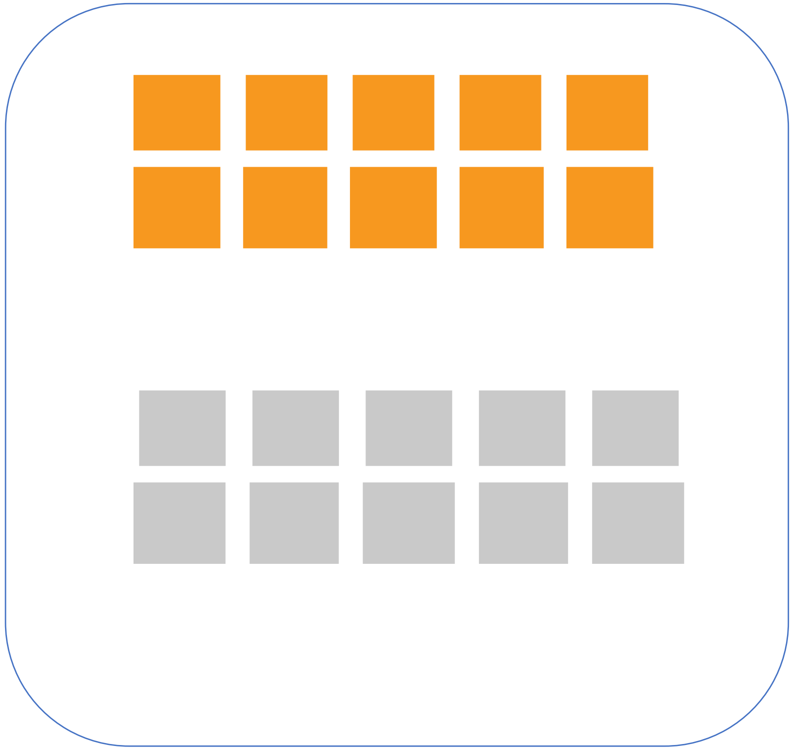
Cost = $1 * 10 * 14 = $140

Resize nodes with Spot instances



Add 10 more nodes on Spot

Resize nodes with Spot instances



20 node cluster running for 7 hours

Cost = $1 * 10 * 7 $70

+ $0.5 * 10 * 7 $35

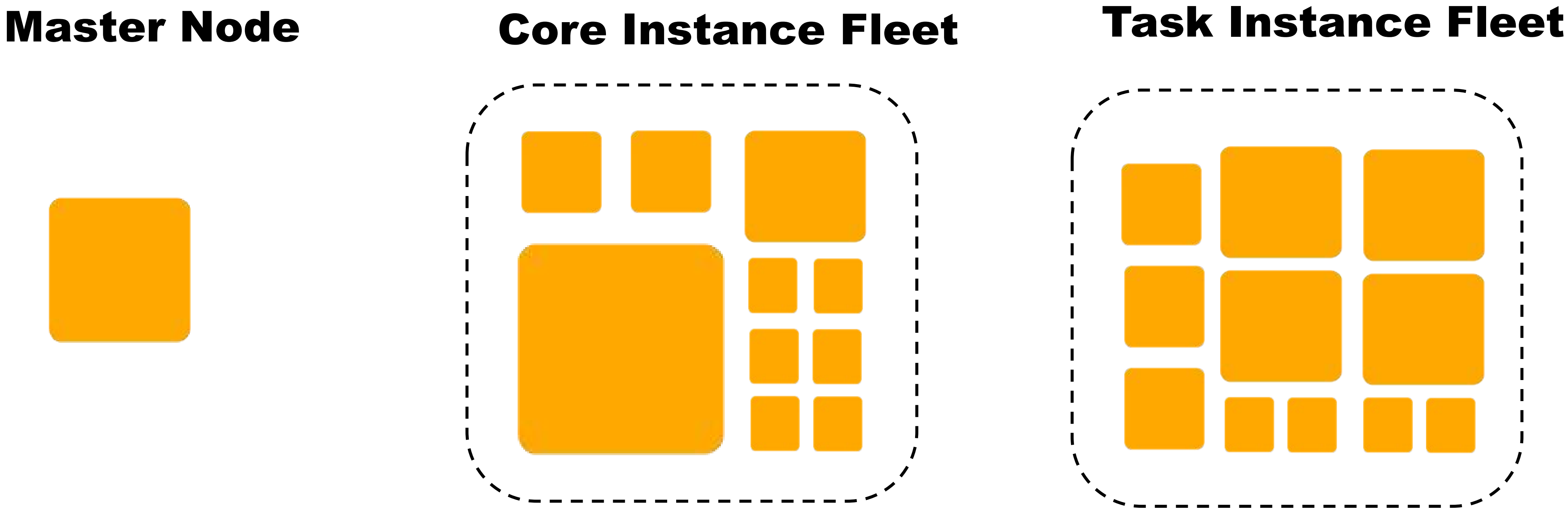
Total $105

Resize nodes with Spot instances

50% less run-time: 14 → 7 hours

25% less cost: $140 → $105

Instance fleets for advanced Spot provisioning

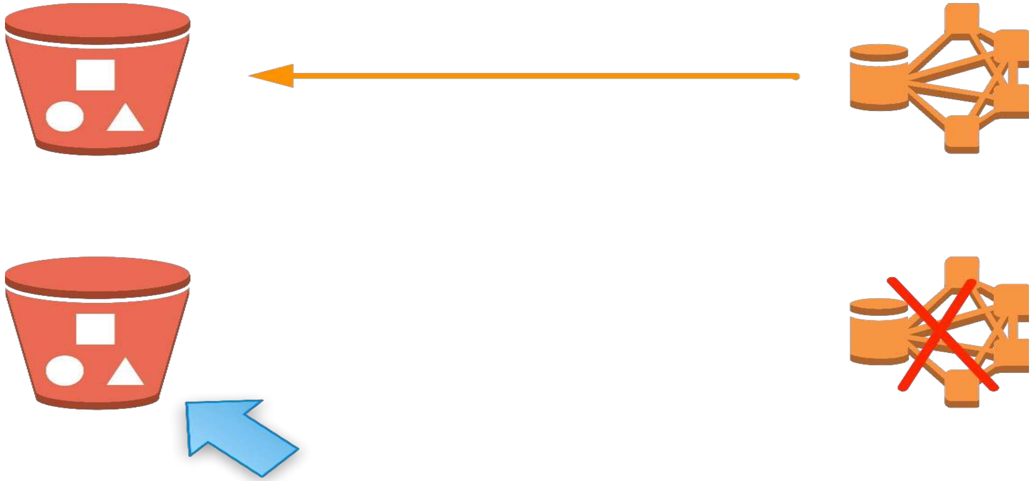


- Provision from a list of instance types with Spot and On-Demand
- Launch in the most optimal Availability Zone based on capacity/price
- Spot Block support (Defined Duration)

https://aws.amazon.com/blogs/aws/new-amazon-emr-instance-fleets/

EMR Enables Transient Clusters

- Cluster lives for the duration of the job
- Shut down the cluster when the job is done
- Input and output data persists on Amazon S3



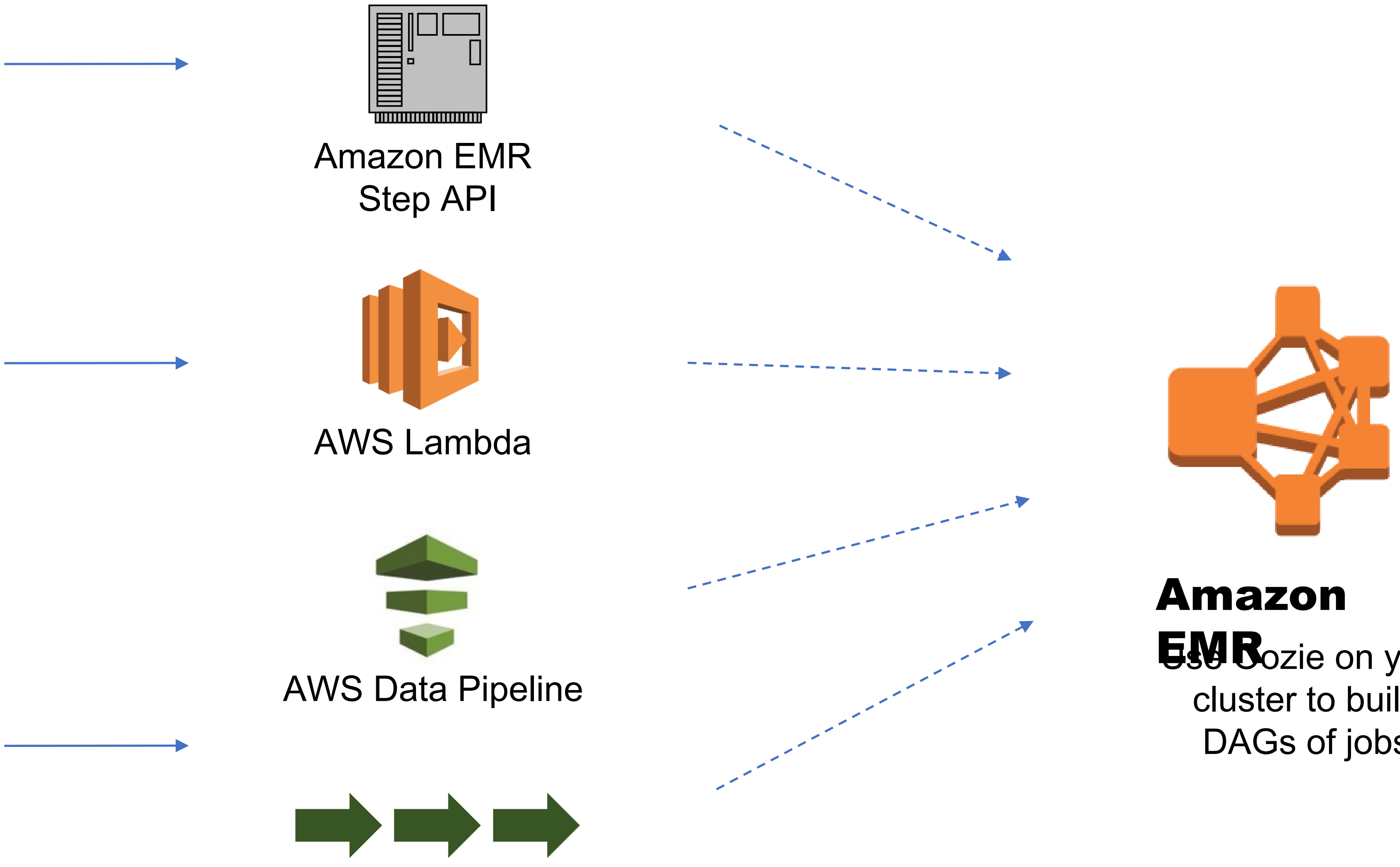
Data on Amazon S3

Options to submit jobs

Submit a application

Use AWS Lambda to submit applications to EMR Step API or directly on your cluster

Create a pipeline to schedule job submission or create complex workflows



Amazon EMR Step API

AWS Lambda

AWS Data Pipeline

Airflow, Luigi, or other schedulers on EC2

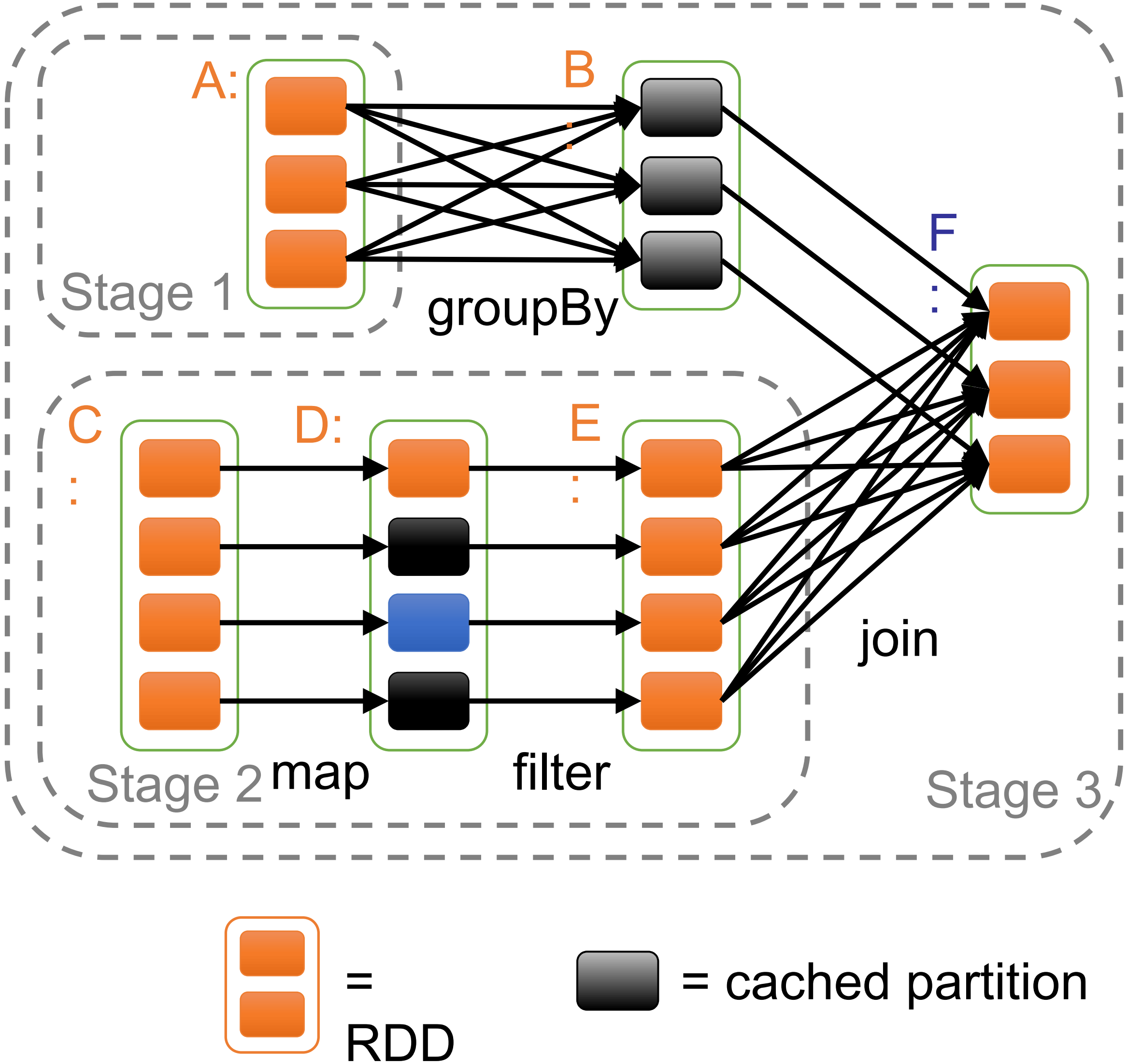
Amazon EMR

Use Oozie on your cluster to build DAGs of jobs

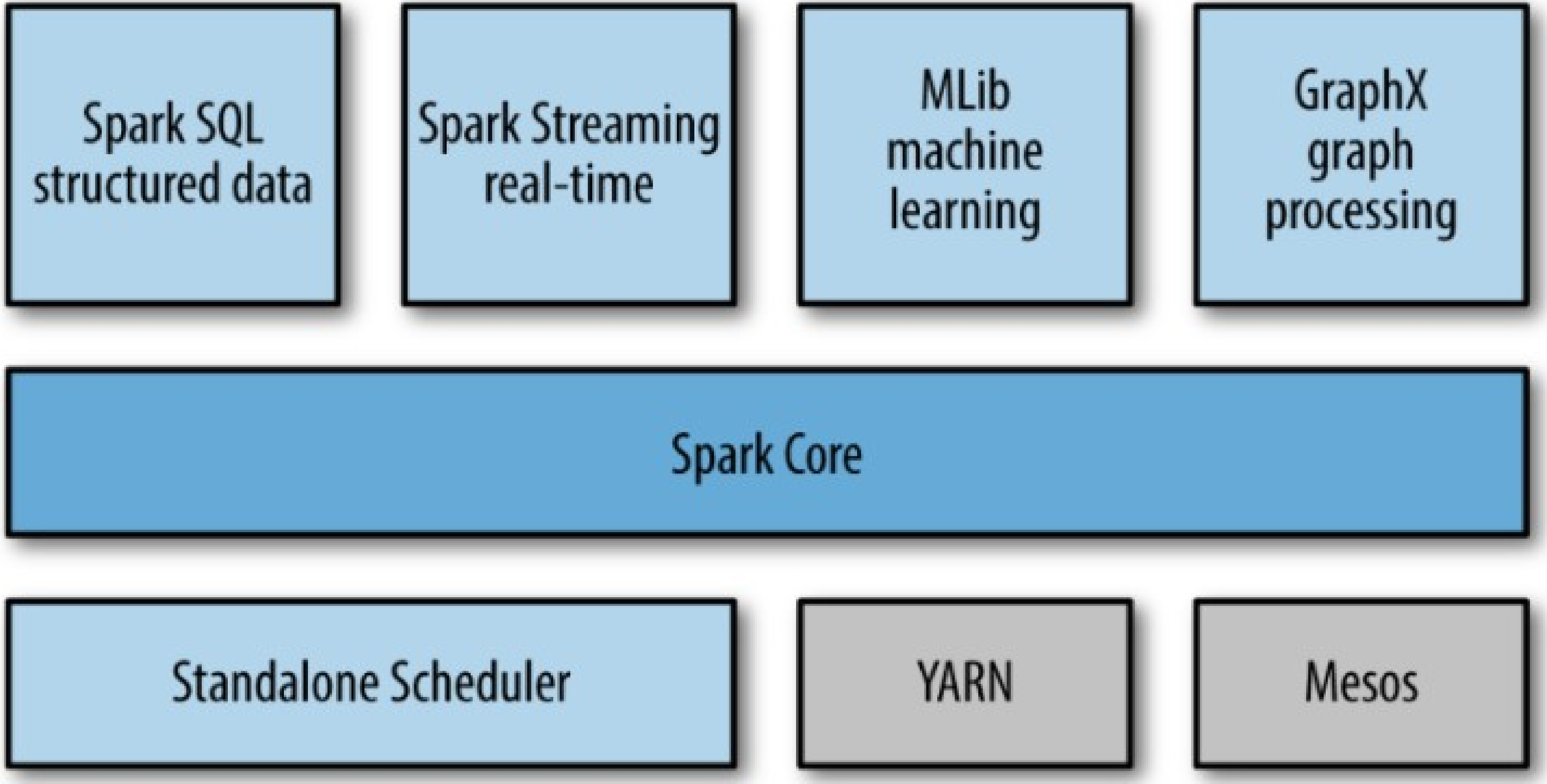
Beyond Map Reduce: Spark

Spark moves at interactive speed

- Massively parallel
- Uses DAGs instead of map-reduce for execution
- Minimizes I/O by storing data in memory
- Partitioning-aware to avoid network-intensive shuffle



Spark components to match your use case



Spark speaks your language

Scala

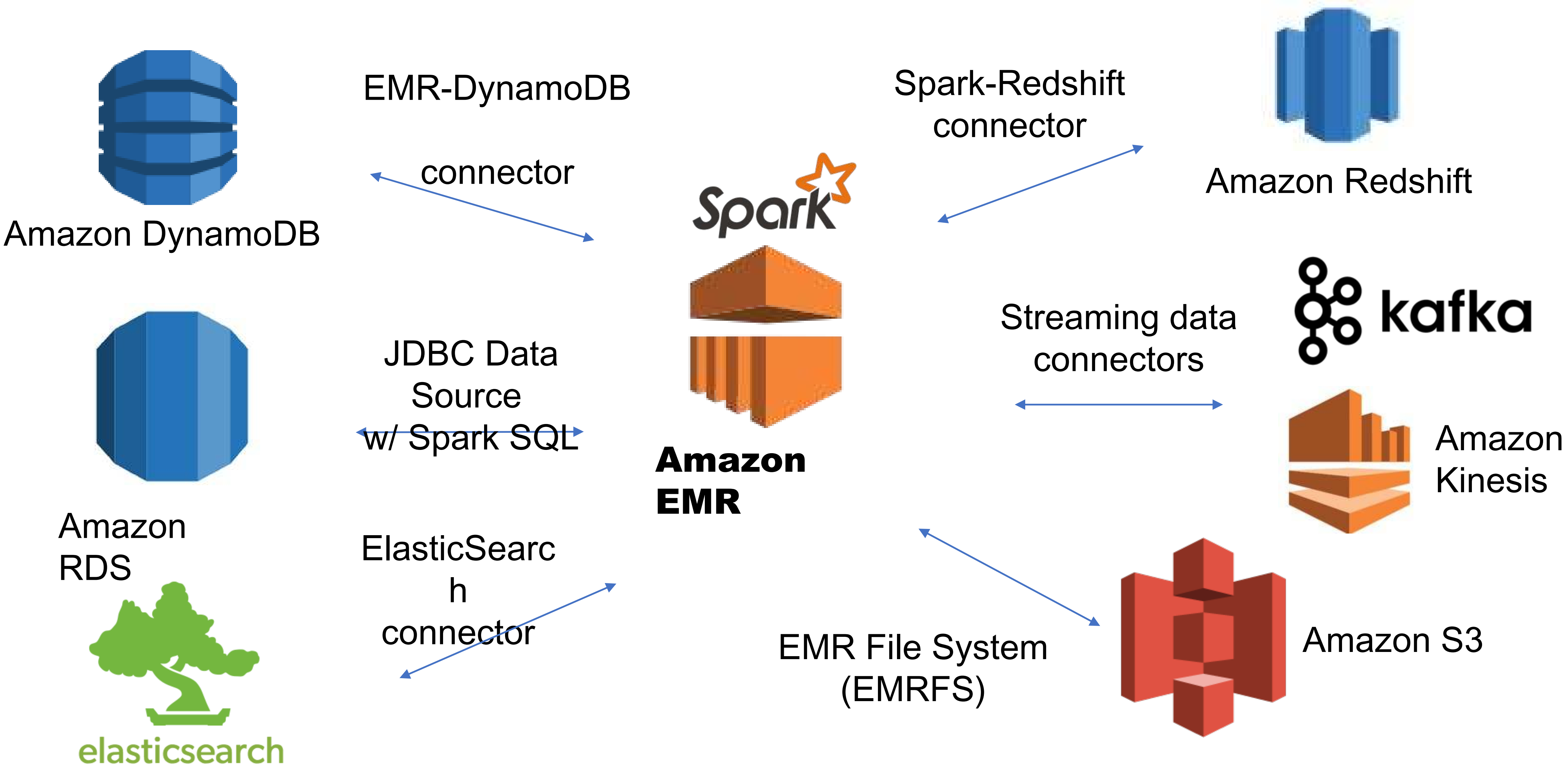
python

Java™

SQL

R

Many storage layers to choose from



Amazon DynamoDB

EMR-DynamoDB connector

Amazon RDS

JDBC Data Source w/ Spark SQL

ElasticSearch connector

Amazon EMR

Spark-Redshift connector

Amazon Redshift

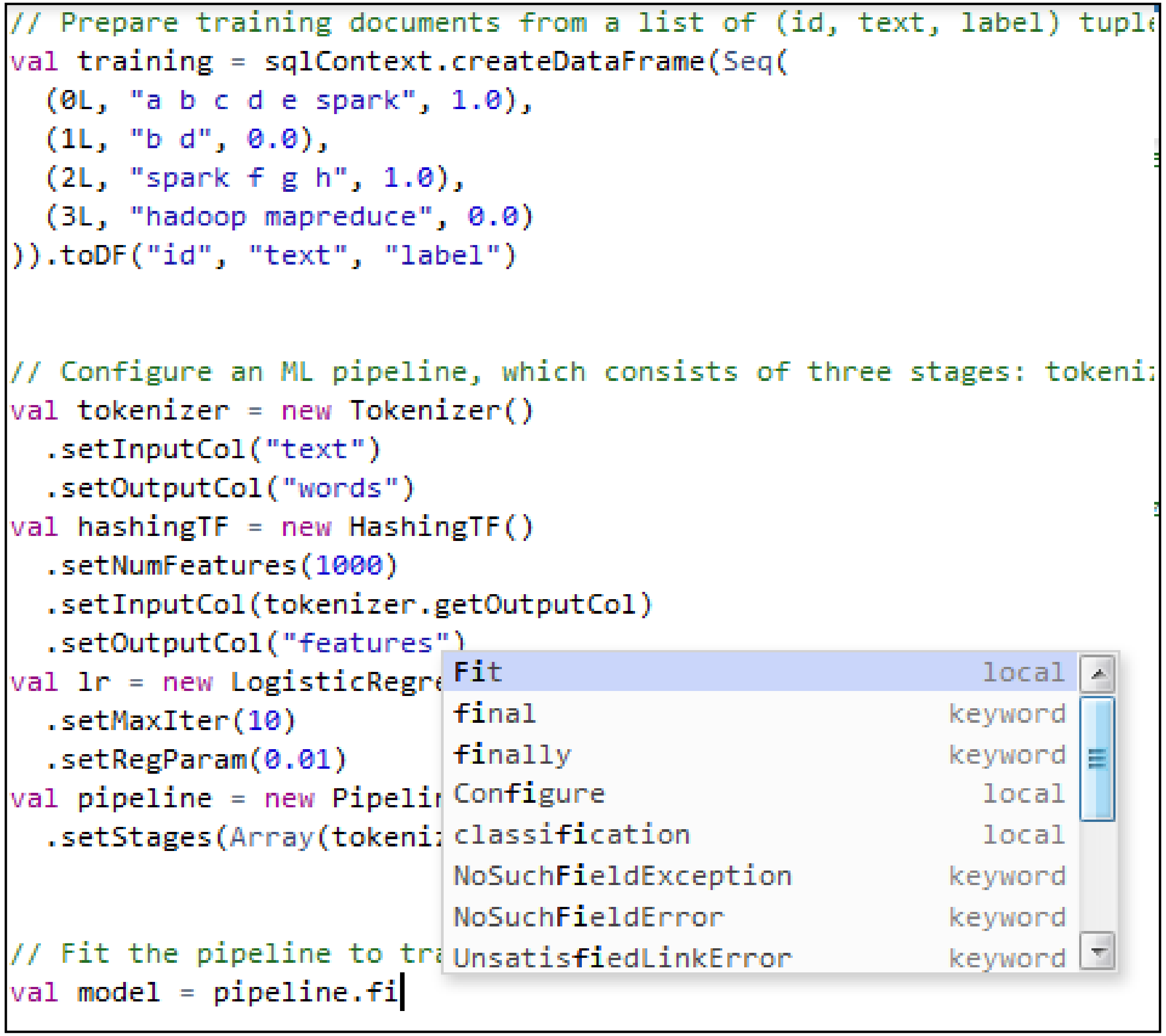
Streaming data connectors

Amazon Kinesis

EMR File System (EMRFS)

Amazon S3

Use DataFrames for machine learning



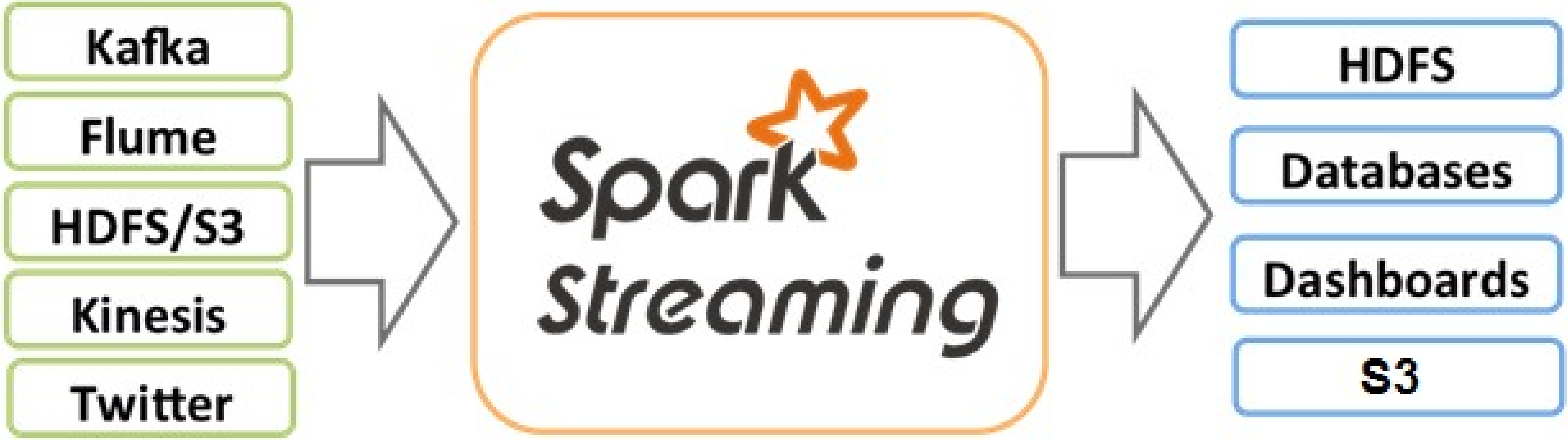
```
// Prepare training documents from a list of (id, text, label) tupl
val training = sqlContext.createDataFrame(Seq(
  (0L, "a b c d e spark", 1.0),
  (1L, "b d", 0.0),
  (2L, "spark f g h", 1.0),
  (3L, "hadoop mapreduce", 0.0)
)).toDF("id", "text", "label")


// Configure an ML pipeline, which consists of three stages: tokeni
val tokenizer = new Tokenizer()
  .setInputCol("text")
  .setOutputCol("words")
val hashingTF = new HashingTF()
  .setNumFeatures(1000)
  .setInputCol(tokenizer.getOutputCol)
  .setOutputCol("features")
val lr = new LogisticRegre
  .setMaxIter(10)
  .setRegParam(0.01)
val pipeline = new Pipelin
  .setStages(Array(tokeni


// Fit the pipeline to tra
val model = pipeline.fi
```

- Spark ML (replacing MLlib) uses DataFrames as input/output for models
- Create ML pipelines with a variety of distributed algorithms

Create DataFrames on streaming data



Amazon EMR

Easy to Use
Launch a cluster in minutes

Low Cost
Pay an hourly rate

Elastic
Easily add or remove capacity

Reliable
Spend less time monitoring

Secure
Manage firewalls

Flexible
Customize the cluster

Resources

https://aws.amazon.com/emr/

https://aws.amazon.com/blogs/bigdata

https://aws.amazon.com/whitepapers/

https://medium.com/@julsimon

Thank you!

Julien Simon, AI Evangelist, EMEA
@julsimon