Apache Spark on Amazon EMR

Pooja Chikkala, Solutions Architect

04/20/2020

Agenda

Introduction to Apache Spark

Spark on Amazon EMR

Tuning best practices

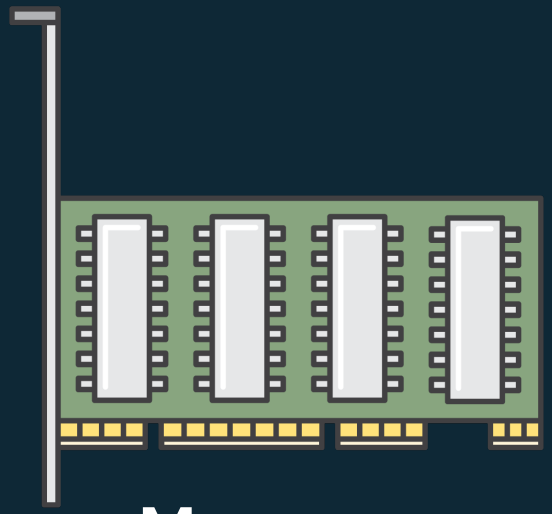
aws

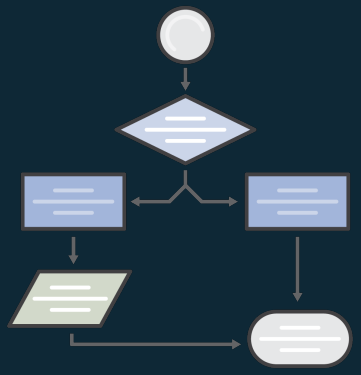
FAST

RICH

aws

Fast

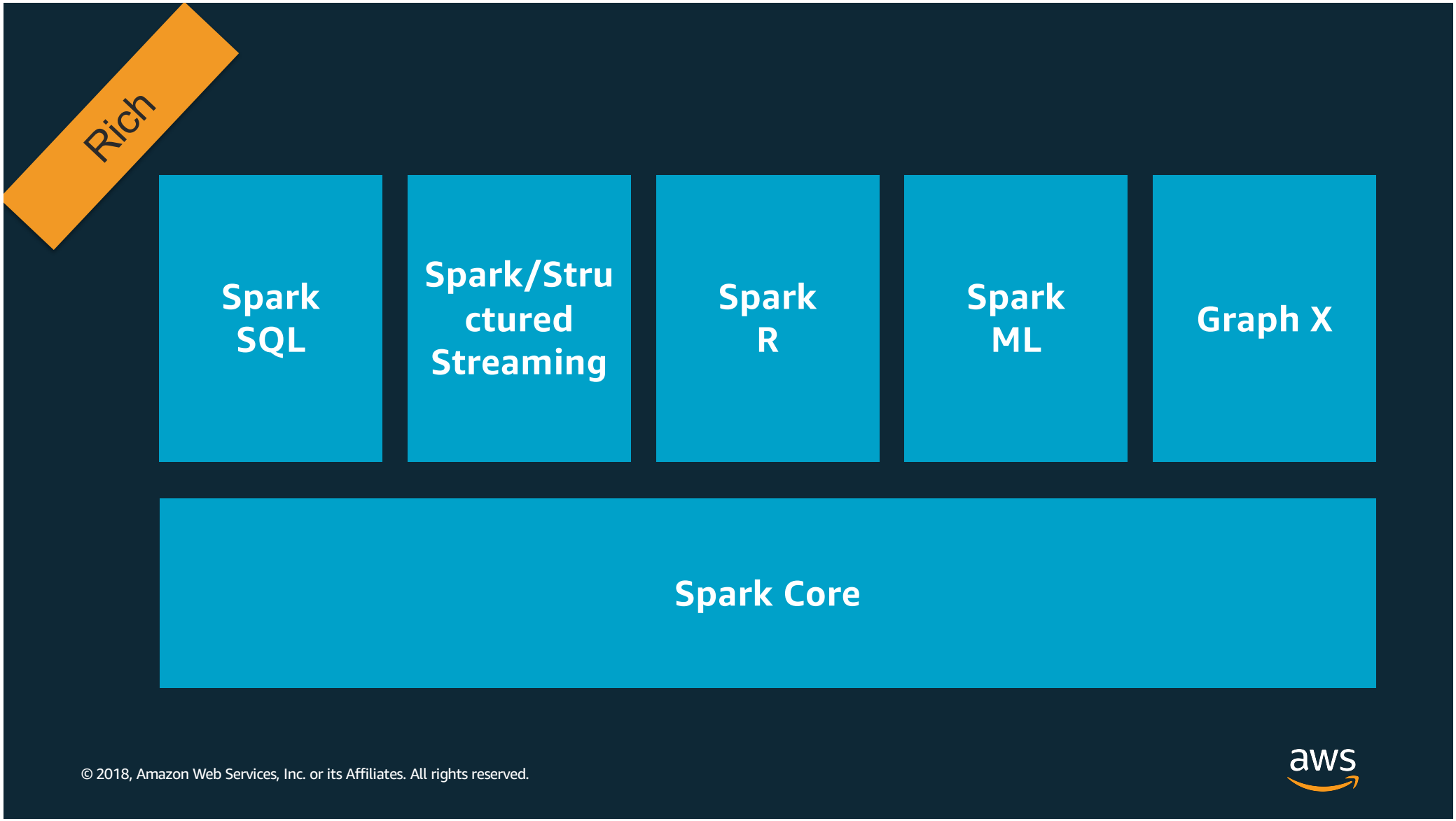




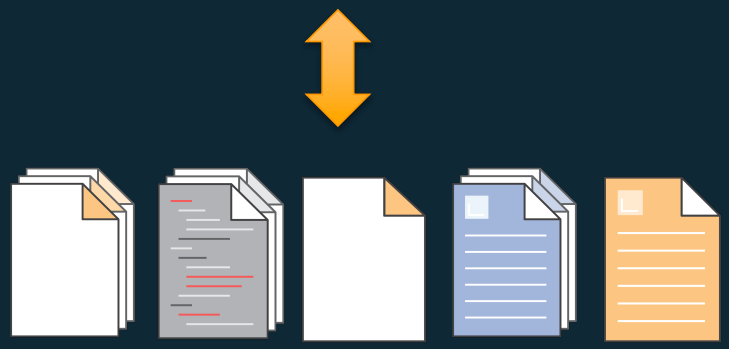
Memory

Processing

aws



Spark Core

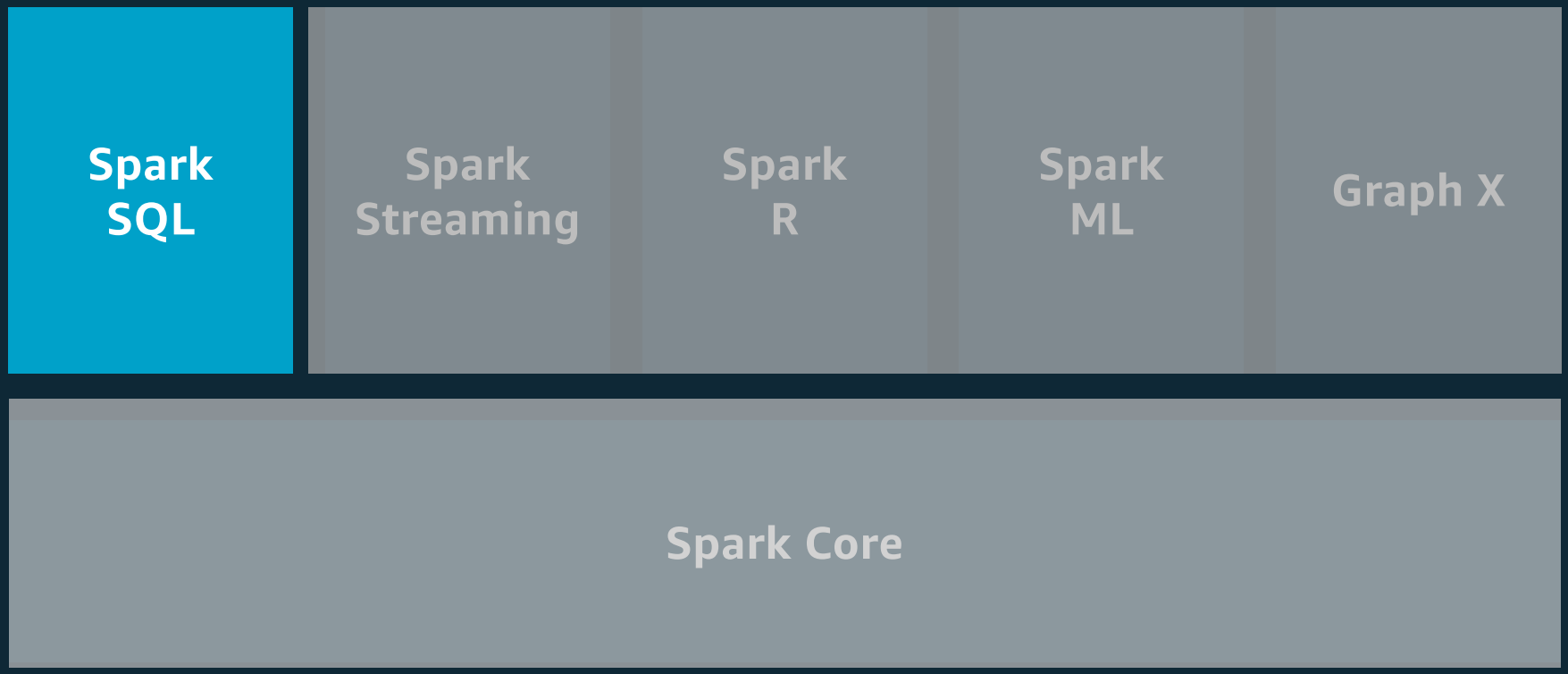


aws

Spark Core

Connectors

aws



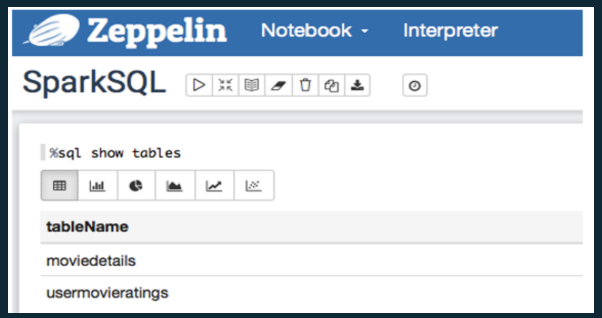
Spark SQL

aws

Spark SQL

BI Tool

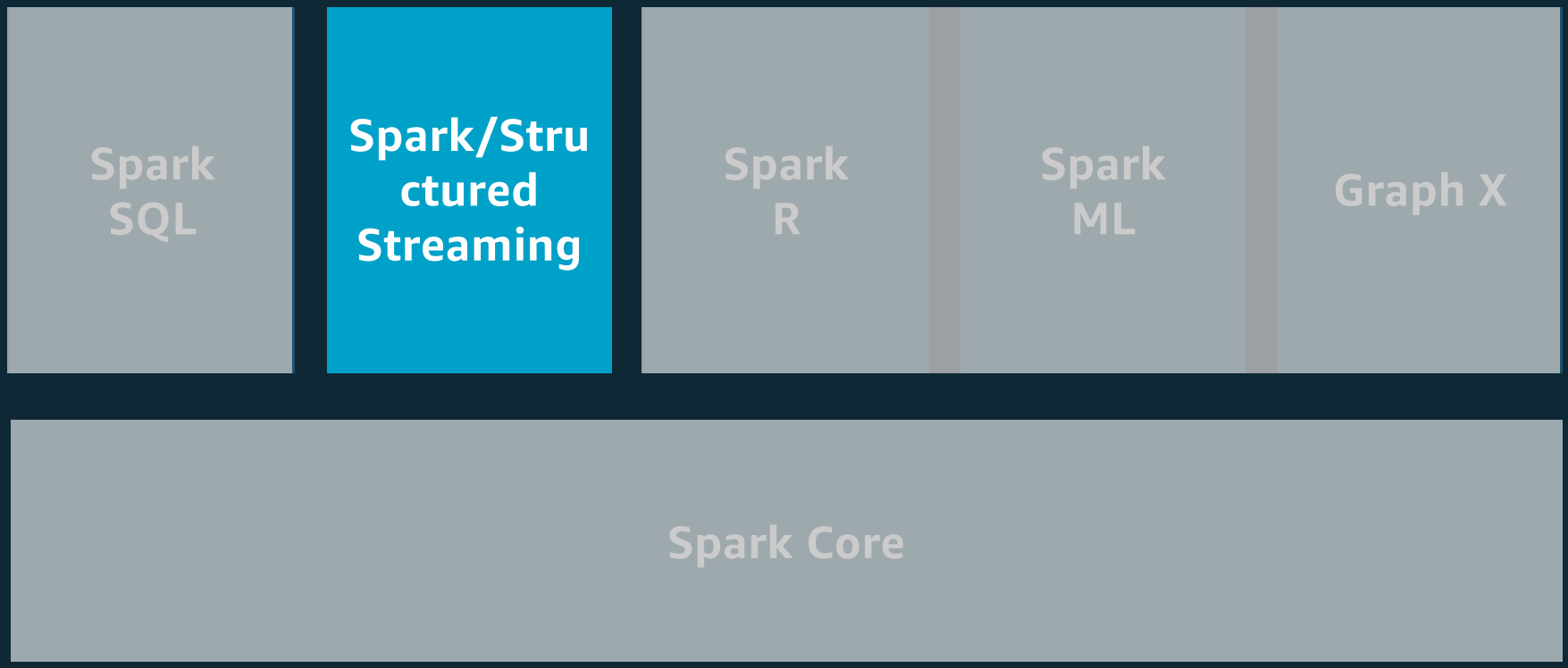
SQL Editor

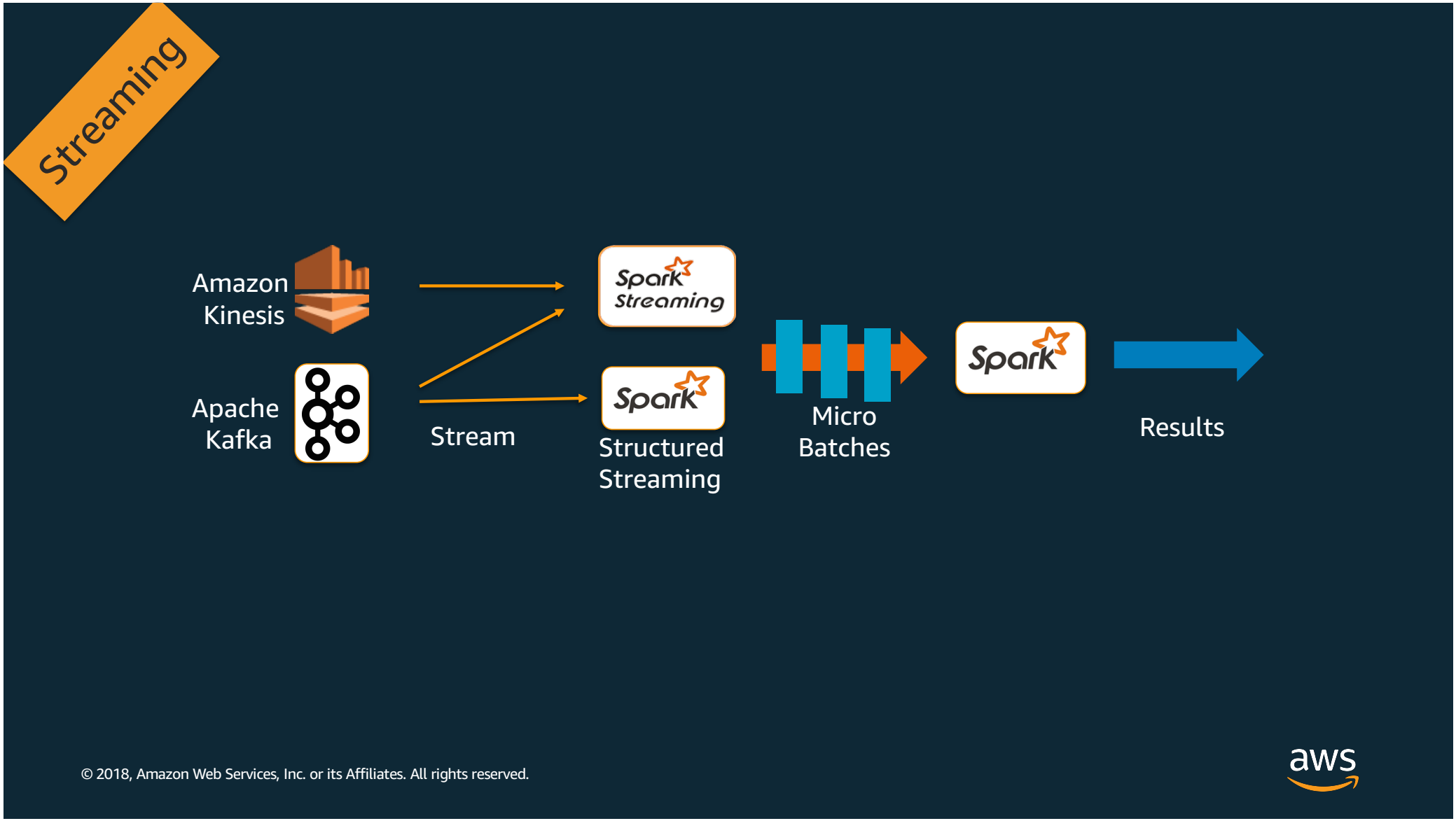


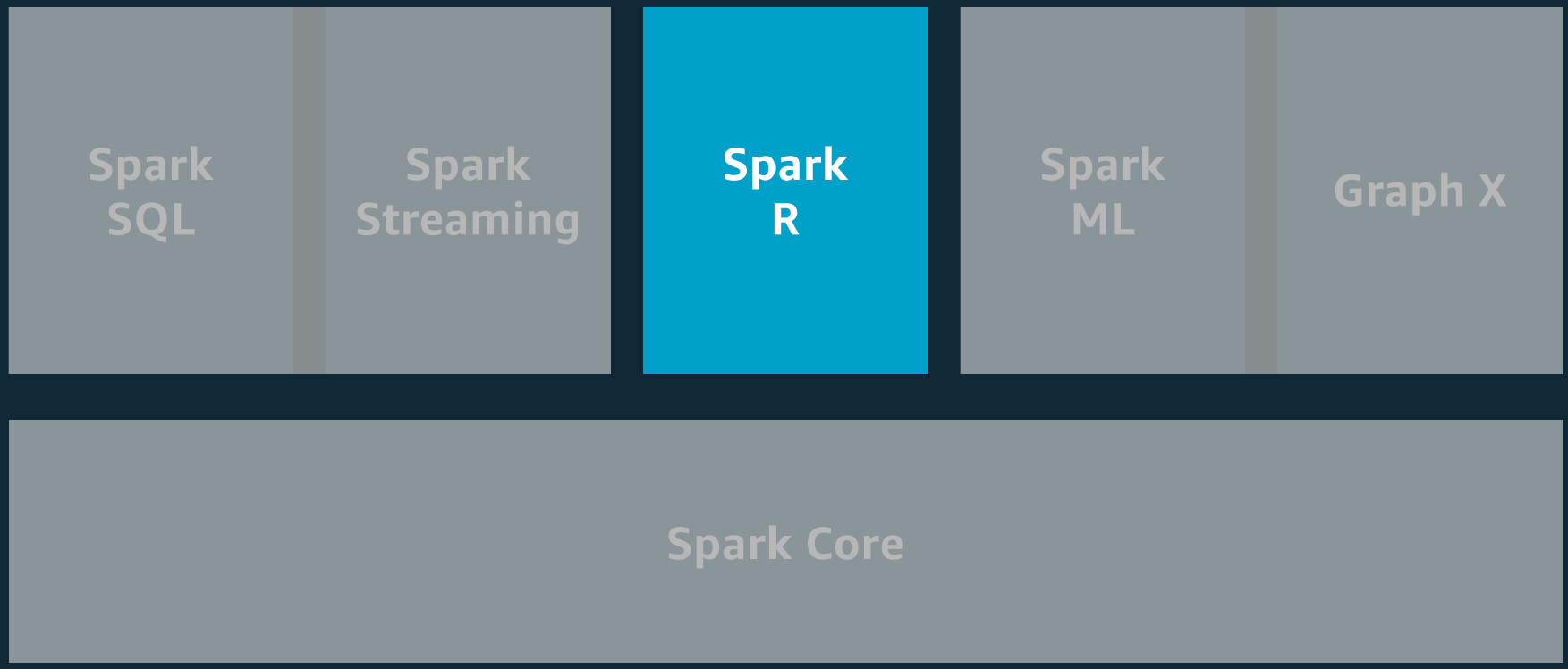
CLI

```
select movie-name, avg(rating) r
from moviedetails
group by movie-name
order by r desc
limit 10;
```

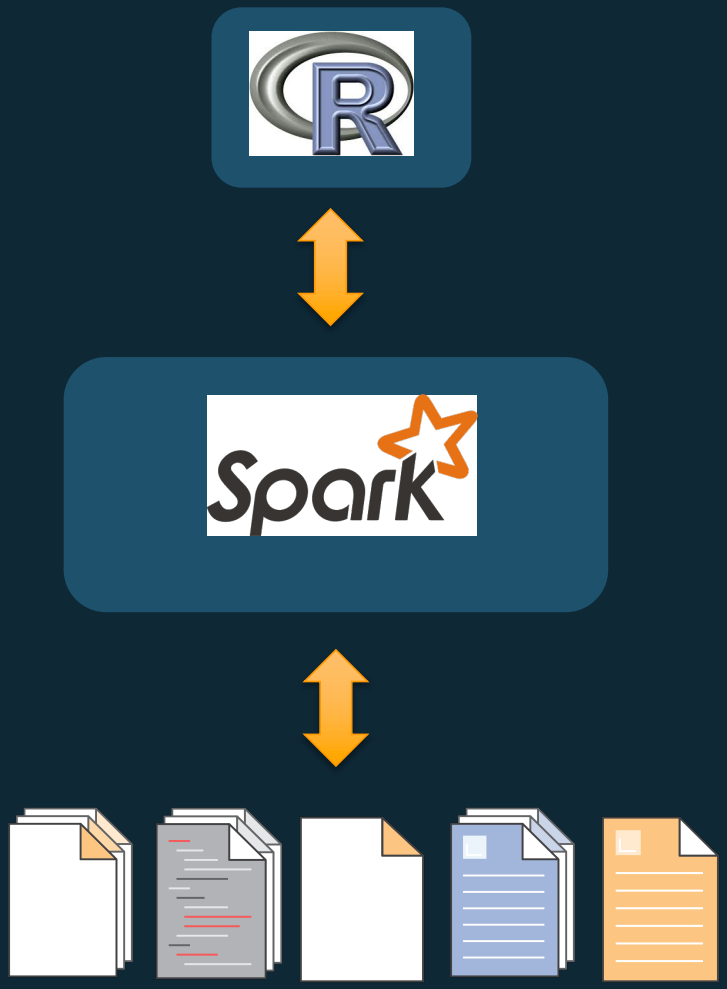

aws



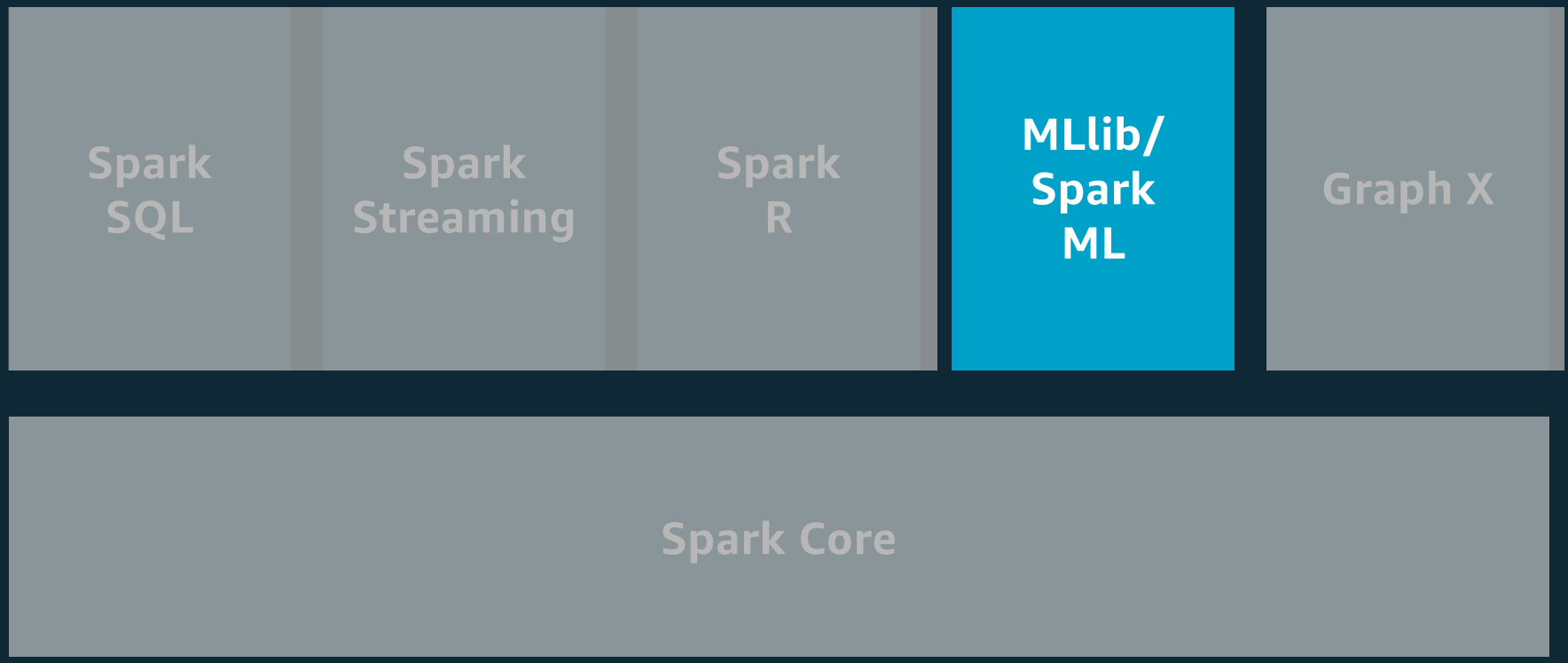


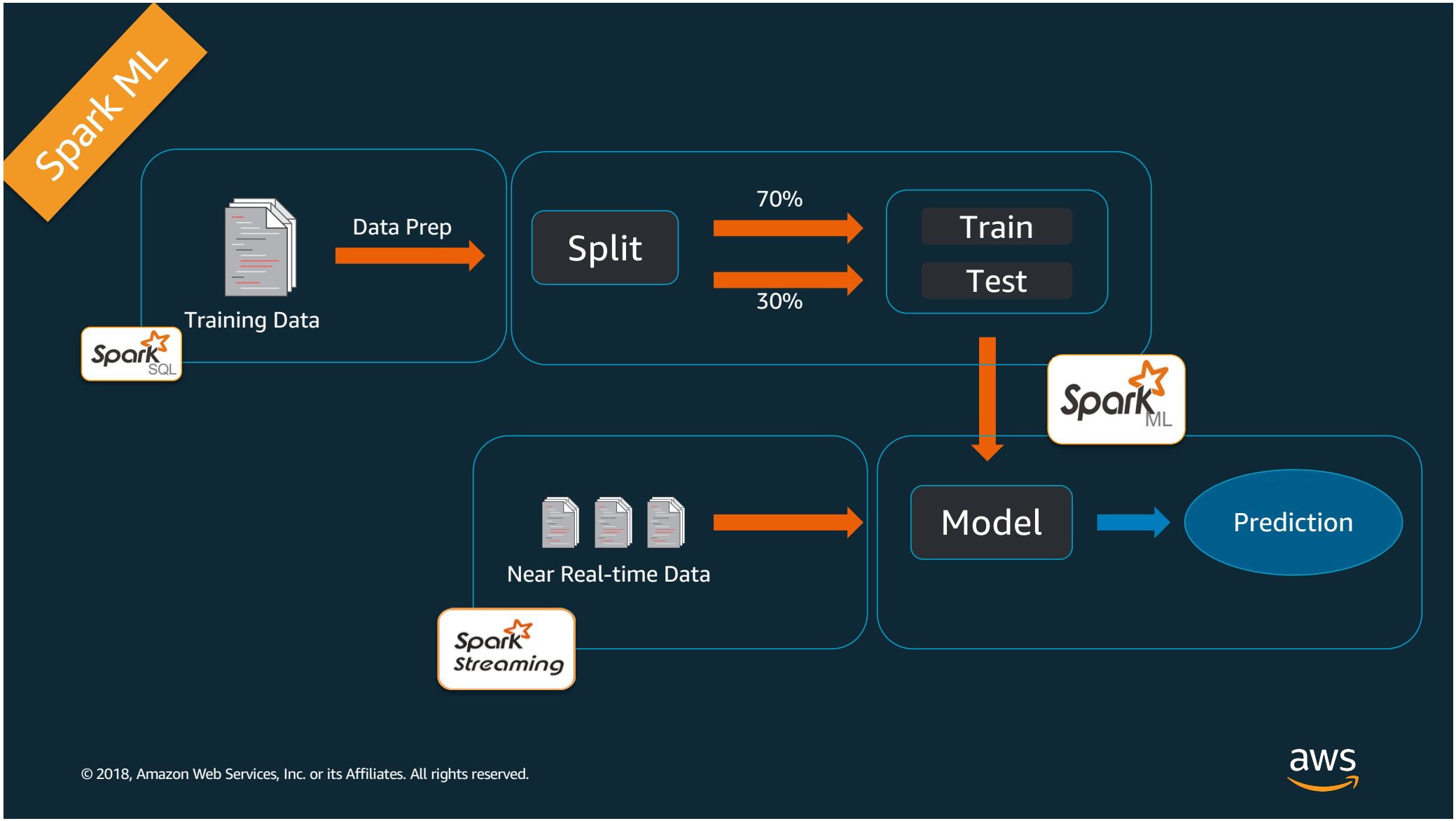


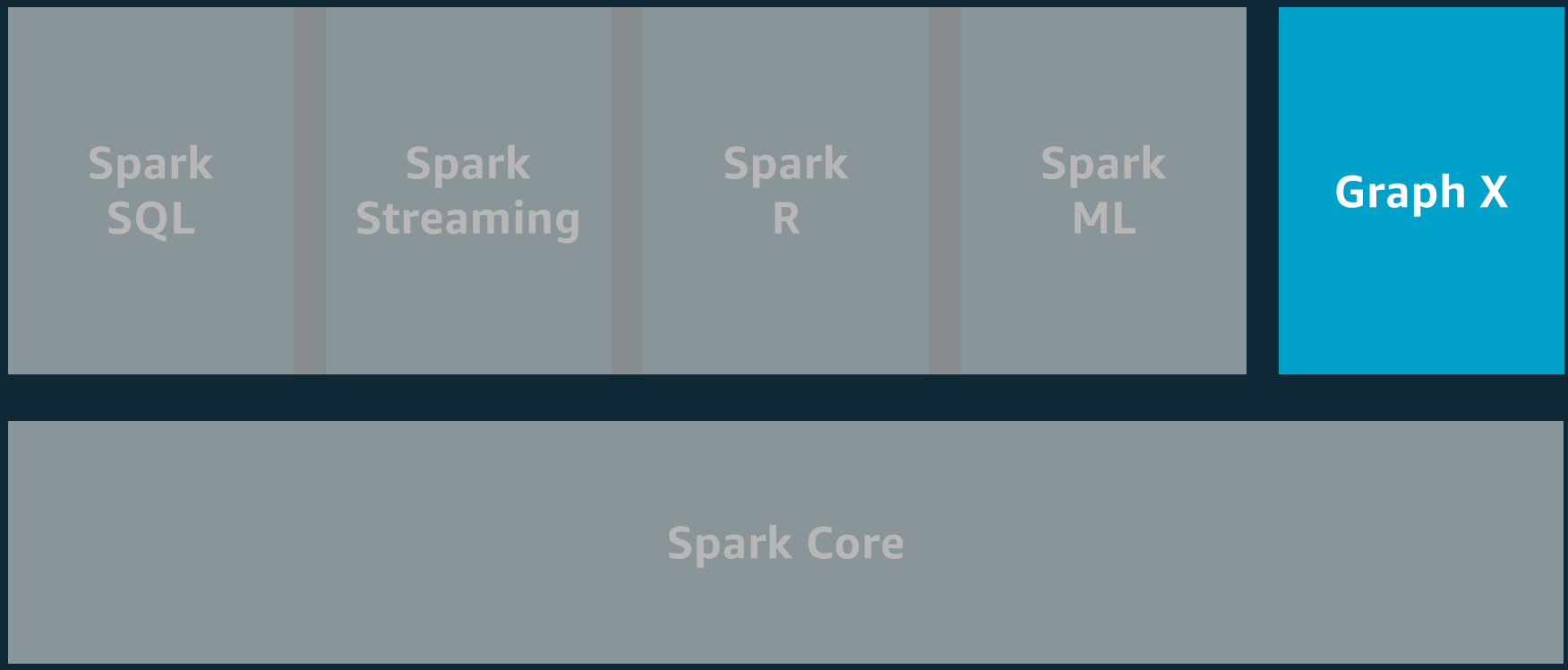
Spark R



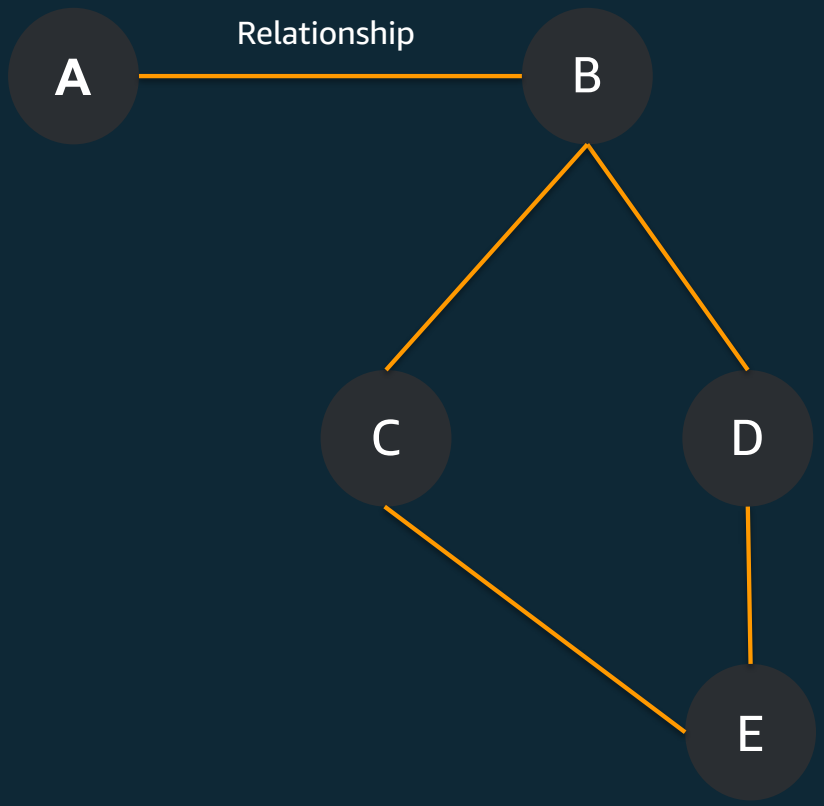
aws







Graph X



aws

Apache Spark on Amazon EMR

aws

Launching a cluster with Spark

```
// CLI
aws emr create-cluster --name "Spark cluster" --release-label emr-5.16.0 --applications
Name=Spark \ --ec2-attributes KeyName=myKey --instance-type m4.large --instance-count 3 --use-
default-roles

// SDK
AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(credentials);
Application sparkApp = new Application().withName("Spark");
Applications myApps = new Applications();
myApps.add(sparkApp);
RunJobFlowRequest request = new RunJobFlowRequest().withName("Spark Cluster")
.withApplications(myApps).withReleaseLabel("emr-5.16.0").withInstances(new
JobFlowInstancesConfig().withEc2KeyName("myKeyName").withInstanceCount(1)
.withKeepJobFlowAliveWhenNoSteps(true).withMasterInstanceType("m4.large")
.withSlaveInstanceType("m4.large") );
RunJobFlowResult result = emr.runJobFlow(request);
```


aws

Using Glue Data Catalog as metastore for Spark SQL

- Configure Glue Data Catalog as metastore for Spark SQL (EMR ver > 5.8)
- Persistent metastore – shared by different clusters, services, applications
- Considerations
    - Renaming tables from within Glue – not supported
    - Column statistics, Hive authorization, Hive constraints, Cost based optimization in Hive are not supported
    - Partition values with quotes or apostrophe not supported
    - No UDFs in query predicates
    - Temporary tables not supported

aws

Submitting Spark jobs

- EMR Step API (Console, CLI, SDK)
- `Spark-submit script`

aws

Optimizations in Spark

Tricks and tips

- Use newer versions of Spark
- Avoid RDDs
- Cache and reuse results
- Use Parquet format for saving and loading data
- Partitions your inputs and outputs
- Avoid shuffling and repartitioning

aws

Tricks and tips

- Pick the right instance types
  - If memory is a constraint, use m4 instances instead of c4 instances
- Configure Spark correctly
  - default EMR configuration for Spark are very simplistic
  - For e.g., enable speculative execution
  - Enable/Disable Dynamic allocation based on workload
  - Minimize number of executor instances per machine
  - Avoid executors with too much memory
  - Number of active tasks = number of cores

aws

Tricks and tips

- Avoid ‘LIKE’ in SQL queries
- Reduce data shuffles
- Optimize joins
    - Avoid cross joins
    - Use broadcast join for smaller datasets
    - Use flatmap instead of broadcast join
- Avoid spilling to disk
- Detect and handle data skew

aws