AWS re:Invent

ANT340

A Deep Dive into What's New with Amazon EMR

Damon Cortesi
Big Data Architect
Amazon Web Services

Abhishek Sinha
Principal Product Manager
Amazon Web Services

EMR 5.x Release Velocity

Existing Apps

- Hadoop
- Flink
- Ganglia
- Hbase
- HBase on S3
- Hive & Hcatalog
- Hue
- Mahout
- Oozie
- Phoenix
- Pig

Existing Apps

- Presto
- Spark
- Sqoop
- Tez
- Zeppelin
- Zookeeper

New Apps this year

- JupyterHub
- Livy
- Tensorflow
- MXNET

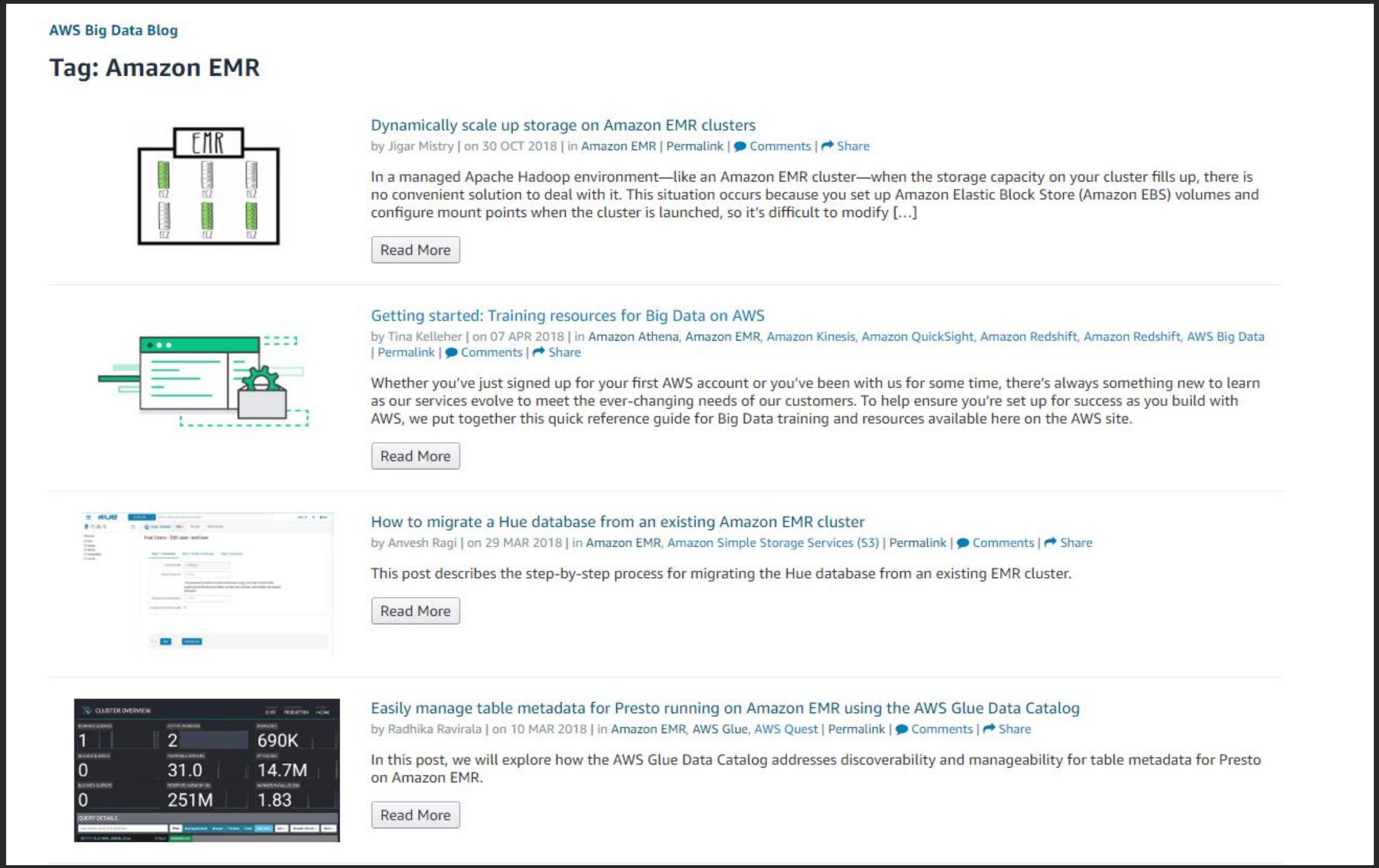
Notable New Features

- Detailed guide to migrate to HBase on S3
- EMRFS Role Auditing in CloudTrail
- Tensorflow optimizations
- SparkMagic and Livy User Impersonation
- FairScheduler support for YARN labels

Application Versions in Amazon EMR 5.x Releases

	5.0.0 July 2016	5.0.3 Oct. 2016	5.1.0 Nov. 2016	5.2.0 Nov. 2016	5.2.1 Dec. 2016	5.2.2 May 2017	5.3.0 Jan. 2017	5.4.0 March 2017	5.5.0 April 2017	5.5.1 & 2 & 3 Jan. & March & Aug. 2018	5.6.0 June 2017	5.7.0 July 2017	5.8.0 Sept. 2017	5.8.1 & 2 Jan. and Aug. 2018	5.9.0 Oct. 2017	5.10.0 Nov. 2017	5.11.0 Dec. 2017	5.11.1 & 2 Jan. & Aug. 2018	5.12.0 Feb. 2018	5.12.1 & 2 March & Aug. 2018	5.13.0 March 2018	5.14.0 June 2018	5.15.0 June 2018	5.16.0 July 2018	5.17.0 August 2018	5.18.0 Oct. 2018	5.19.0 Nov. 2018	
Hadoop	2.7.2	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.7.3	2.8.3	2.8.3	2.8.3	2.8.3	2.8.3	2.8.4	2.8.4	2.8.4	2.8.5	Hadoop
Flink			1.1.3	1.1.3	1.1.3	1.1.3	1.1.4	1.2.0	1.2.0	1.2.0	1.2.1	1.3.0	1.3.1	1.3.1	1.3.2	1.3.2	1.3.2	1.3.2	1.4.0	1.4.0	1.4.0	1.4.2	1.4.2	1.5.0	1.5.2	1.6.0	1.6.1	Flink
Ganglia	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	3.7.2	Ganglia
HBase	1.2.2	1.2.2	1.2.3	1.2.3	1.2.3	1.2.3	1.2.3	1.3.0	1.3.0	1.3.0	1.3.0	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1	1.4.0	1.4.0	1.4.2	1.4.2	1.4.4	1.4.4	1.4.6	1.4.7	1.4.7	HBase
Hive & HCatalog	2.1.0	2.1.0	2.1.0	2.1.0	2.1.0	2.1.0	2.1.1	2.1.1	2.1.1	2.1.1	2.1.1	2.1.1	2.3.0	2.3.0	2.3.0	2.3.1	2.3.2	2.3.2	2.3.2	2.3.2	2.3.2	2.3.2	2.3.3	2.3.3	2.3.3	2.3.3	2.3.3	Hive & HCatalog
Hue	3.10.0	3.10.0	3.10.0	3.10.0	3.10.0	3.10.0	3.11.0	3.11.0	3.12.0	3.12.0	3.12.0	3.12.0	3.12.0	3.12.0	4.0.1	4.0.1	4.0.1	4.0.1	4.1.0	4.1.0	4.1.0	4.1.0	4.2.0	4.2.0	4.2.0	4.2.0	4.2.0	Hue
JupyterHub																						0.8.1	0.8.1	0.8.1	0.8.1	0.8.1	0.9.4	JupyterHub
Livy															0.4.0	0.4.0	0.4.0	0.4.0	0.4.0	0.4.0	0.4.0	0.4.0	0.4.0	0.5.0	0.5.0	0.5.0	0.5.0	Livy
Mahout	0.12.2	0.12.2	0.12.2	0.12.2	0.12.2	0.12.2	0.12.2	0.12.2	0.12.2	0.12.2	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	Mahout
MXNet																0.12.0	0.12.0	0.12.0	1.0.0	1.0.0	1.0.0	1.1.0	1.1.0	1.2.0	1.2.0	1.2.0	1.3.0	MXNet
Oozie	4.2.0	4.2.0	4.2.0	4.2.0	4.2.0	4.2.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	4.3.0	5.0.0	5.0.0	5.0.0	5.0.0	5.0.0	Oozie
Phoenix	4.7.0	4.7.0	4.7.0	4.7.0	4.7.0	4.7.0	4.7.0	4.9.0	4.9.0	4.9.0	4.9.0	4.11.0	4.11.0	4.11.0	4.11.0	4.11.0	4.11.0	4.11.0	4.13.0	4.13.0	4.13.0	4.13.0	4.13.0	4.14.0	4.14.0	4.14.0	4.14.0	Phoenix
Pig	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.16.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	Pig
Presto	0.150	0.152.3	0.152.3	0.152.3	0.157.1	0.157.1	0.157.1	0.166	0.170	0.170	0.170	0.170	0.170	0.170	0.184	0.187	0.187	0.187	0.188	0.188	0.194	0.194	0.194	0.203	0.206	0.210	0.212	Presto
Spark	2.0.0	2.0.1	2.0.1	2.0.2	2.0.2	2.0.2	2.1.0	2.1.0	2.1.0	2.1.0	2.1.1	2.1.1	2.2.0	2.2.0	2.2.0	2.2.0	2.2.1	2.2.1	2.2.1	2.2.1	2.3.0	2.3.0	2.3.0	2.3.1	2.3.1	2.3.2	2.3.2	Spark
Sqoop	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.7	1.4.7	1.4.7	1.4.7	1.4.7	1.4.7	Sqoop
TensorFlow																									1.9.0	1.9.0	1.11.0	Tensorflow
Tez	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	0.8.4	Tez
Zeppelin	0.6.1	0.6.1	0.6.2	0.6.2	0.6.2	0.6.2	0.6.2	0.7.0	0.7.1	0.7.1	0.7.1	0.7.2	0.7.2	0.7.2	0.7.2	0.7.3	0.7.3	0.7.3	0.7.3	0.7.3	0.7.3	0.7.3	0.7.3	0.7.3	0.7.3	0.8.0	0.8.0	Zeppelin
Zookeeper	3.4.8	3.4.8	3.4.8	3.4.8	3.4.9	3.4.9	3.4.9	3.4.9	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.10	3.4.12	3.4.12	3.4.12	3.4.12	3.4.13	Zookeeper
AWS SDK for Java	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.10.75	1.11.160	1.11.160	1.11.183	1.11.221	1.11.238	1.11.238	1.11.267	1.11.267	1.11.297	1.11.297	1.11.333	1.11.336	1.11.336	1.11.393	1.11.433	AWS SDK for Java
	5.0.0 July 2016	5.0.3 Oct. 2016	5.1.0 Nov. 2016	5.2.0 Nov. 2016	5.2.1 Dec. 2016	5.2.2 May 2017	5.3.0 Jan. 2017	5.4.0 March 2017	5.5.0 April 2017	5.5.1 & 2 & 3 Jan. & March & Aug. 2018	5.6.0 June 2017	5.7.0 July 2017	5.8.0 Sept. 2017	5.8.1 & 2 Jan. & Aug. 2018	5.9.0 Oct. 2017	5.10.0 Nov. 2017	5.11.0 Dec. 2017	5.11.1 & 2 Jan. & Aug. 2018	5.12.0 Feb. 2018	5.12.1 & 2 March & Aug. 2018	5.13.0 March 2018	5.14.0 June 2018	5.15.0 June 2018	5.16.0 July 2018	5.17.0 August 2018	5.18.0 Oct. 2018	5.19.0 Nov. 2018	

AWS Big Data Blog



AWS Big Data Blog

Tag: Amazon EMR

Dynamically scale up storage on Amazon EMR clusters

by Jigar Mistry | on 30 OCT 2018 | in Amazon EMR | Permalink | Comments | Share

In a managed Apache Hadoop environment—like an Amazon EMR cluster—when the storage capacity on your cluster fills up, there is no convenient solution to deal with it. This situation occurs because you set up Amazon Elastic Block Store (Amazon EBS) volumes and configure mount points when the cluster is launched, so it's difficult to modify […]

Read More

Getting started: Training resources for Big Data on AWS

by Tina Kelleher | on 07 APR 2018 | in Amazon Athena, Amazon EMR, Amazon Kinesis, Amazon QuickSight, Amazon Redshift, Amazon Redshift, AWS Big Data | Permalink | Comments | Share

Whether you've just signed up for your first AWS account or you've been with us for some time, there's always something new to learn as our services evolve to meet the ever-changing needs of our customers. To help ensure you're set up for success as you build with AWS, we put together this quick reference guide for Big Data training and resources available here on the AWS site.

Read More

How to migrate a Hue database from an existing Amazon EMR cluster

by Anvesh Ragi | on 29 MAR 2018 | in Amazon EMR, Amazon Simple Storage Services (S3) | Permalink | Comments | Share

This post describes the step-by-step process for migrating the Hue database from an existing EMR cluster.

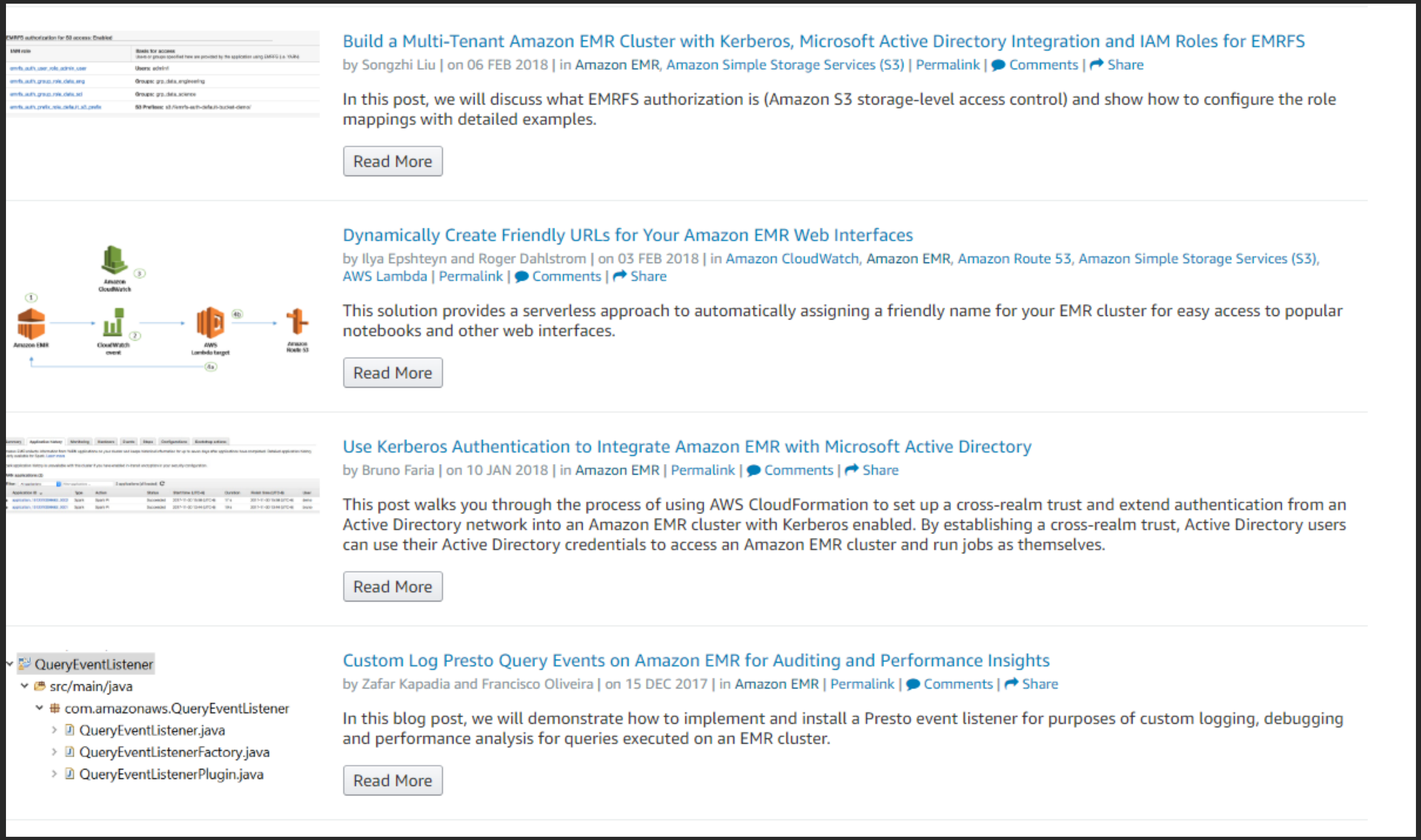
Read More

Easily manage table metadata for Presto running on Amazon EMR using the AWS Glue Data Catalog

by Radhika Ravirala | on 10 MAR 2018 | in Amazon EMR, AWS Glue, AWS Quest | Permalink | Comments | Share

In this post, we will explore how the AWS Glue Data Catalog addresses discoverability and manageability for table metadata for Presto on Amazon EMR.

Read More



Build a Multi-Tenant Amazon EMR Cluster with Kerberos, Microsoft Active Directory Integration and IAM Roles for EMRFS

by Songzhi Liu | on 06 FEB 2018 | in Amazon EMR, Amazon Simple Storage Services (S3) | Permalink | Comments | Share

In this post, we will discuss what EMRFS authorization is (Amazon S3 storage-level access control) and show how to configure the role mappings with detailed examples.

Read More

Dynamically Create Friendly URLs for Your Amazon EMR Web Interfaces

by Ilya Epshteyn and Roger Dahlstrom | on 03 FEB 2018 | in Amazon CloudWatch, Amazon EMR, Amazon Route 53, Amazon Simple Storage Services (S3), AWS Lambda | Permalink | Comments | Share

This solution provides a serverless approach to automatically assigning a friendly name for your EMR cluster for easy access to popular notebooks and other web interfaces.

Read More

Use Kerberos Authentication to Integrate Amazon EMR with Microsoft Active Directory

by Bruno Faria | on 10 JAN 2018 | in Amazon EMR | Permalink | Comments | Share

This post walks you through the process of using AWS CloudFormation to set up a cross-realm trust and extend authentication from an Active Directory network into an Amazon EMR cluster with Kerberos enabled. By establishing a cross-realm trust, Active Directory users can use their Active Directory credentials to access an Amazon EMR cluster and run jobs as themselves.

Read More

Custom Log Presto Query Events on Amazon EMR for Auditing and Performance Insights

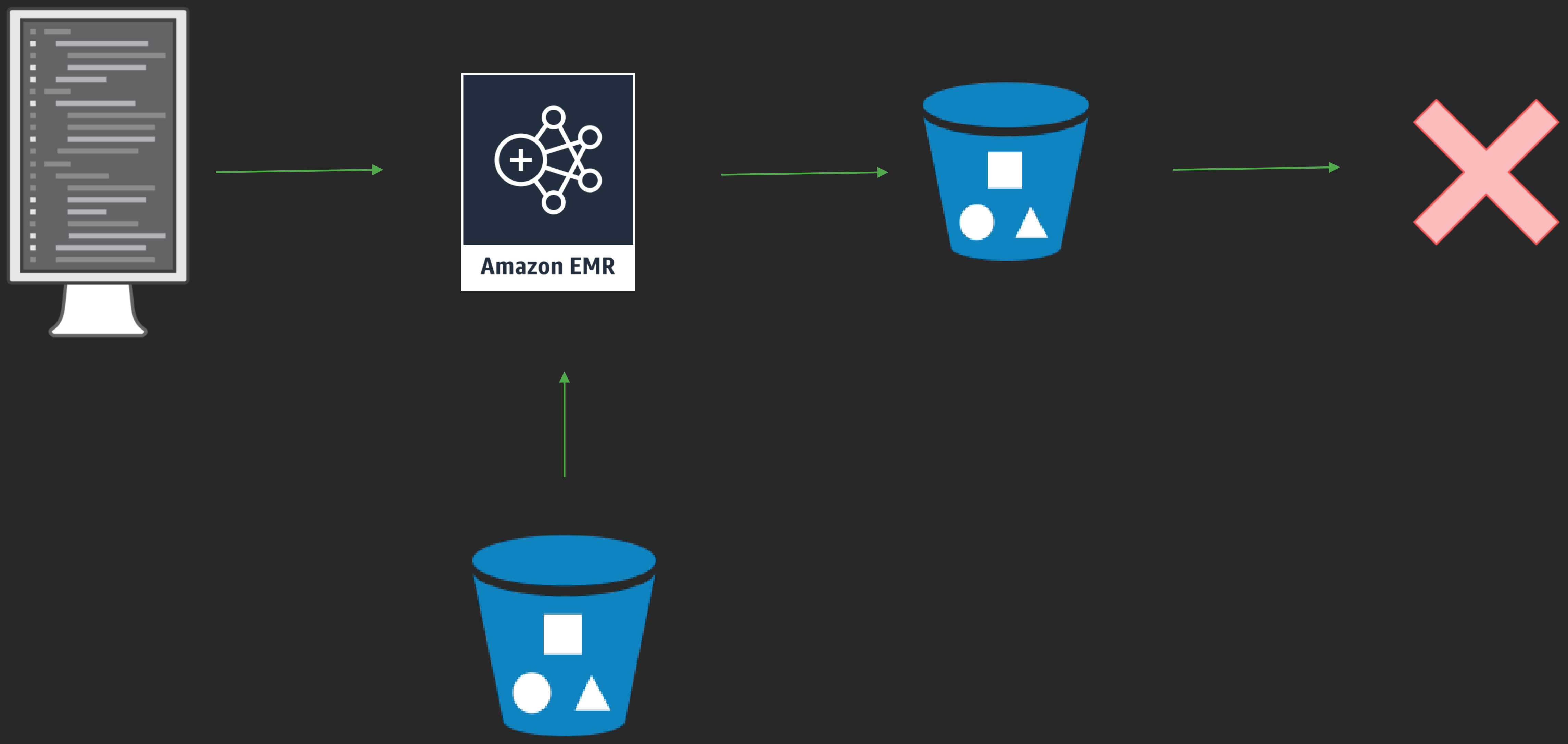
by Zafar Kapadia and Francisco Oliveira | on 15 DEC 2017 | in Amazon EMR | Permalink | Comments | Share

In this blog post, we will demonstrate how to implement and install a Presto event listener for purposes of custom logging, debugging and performance analysis for queries executed on an EMR cluster.

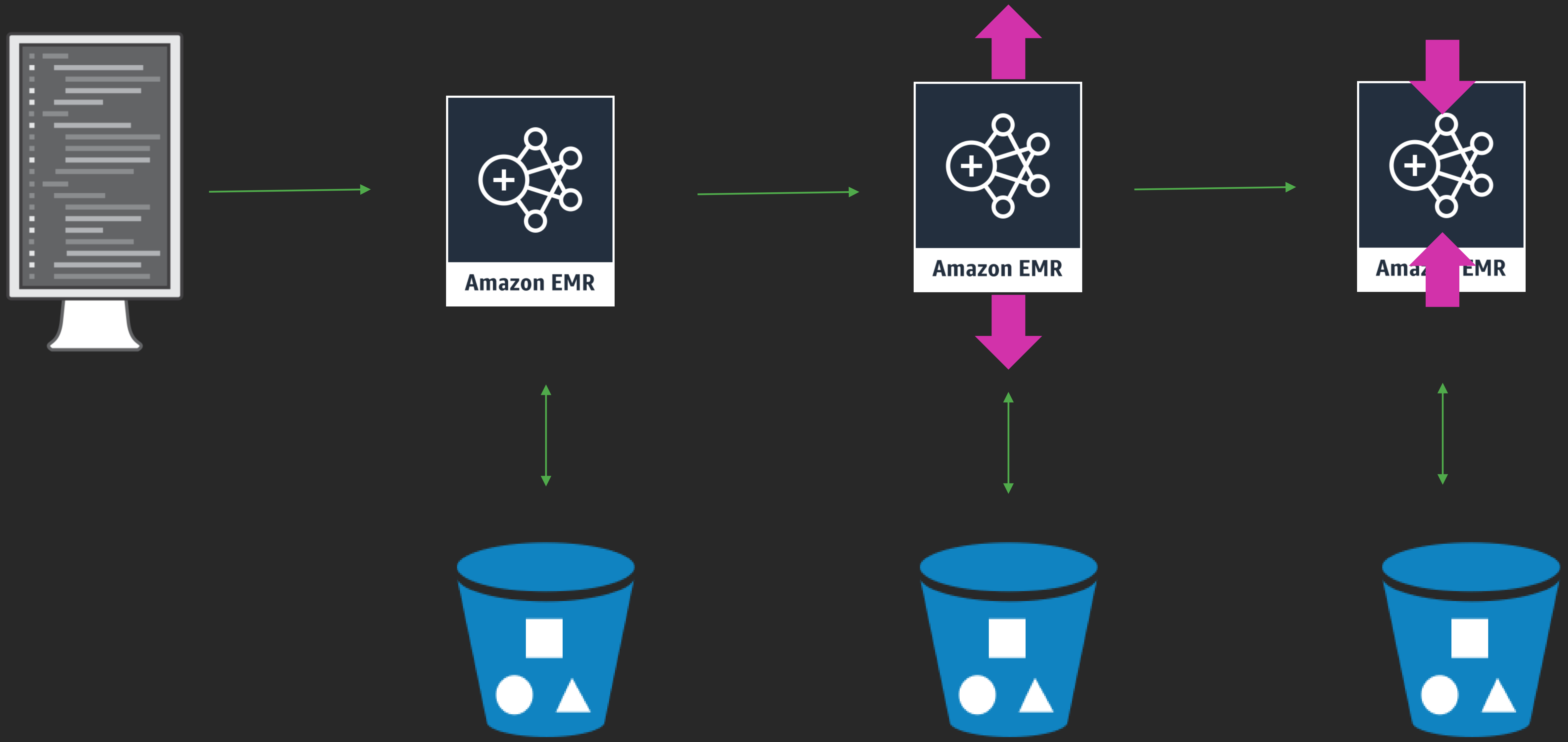
Read More

Architectural Patterns

Transient Clusters



Persistent Clusters



Two architectural patterns

Transient Clusters

- Large-scale transformation
- ETL to other DWH or Data Lake
- Building ML jobs

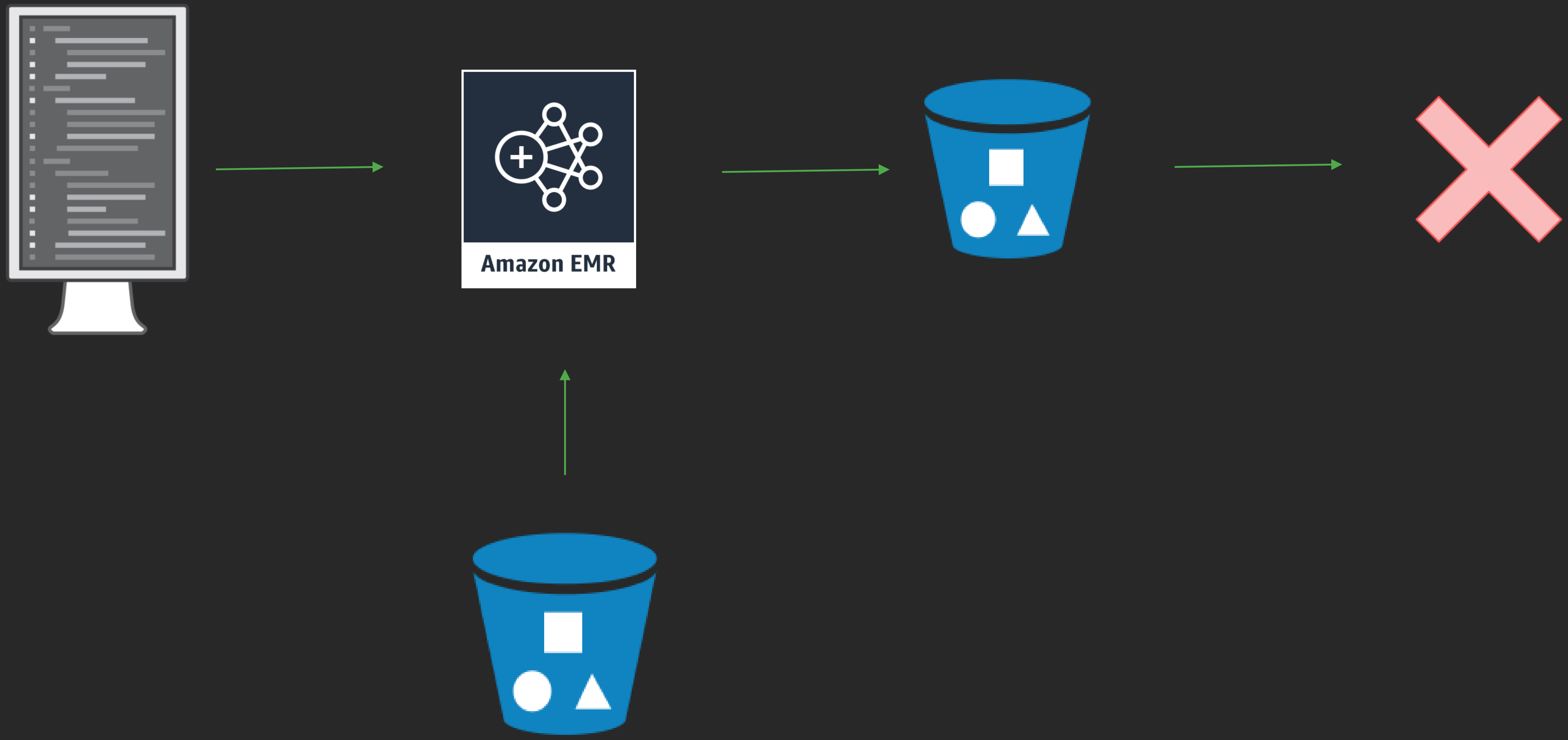
Persistent Clusters

- Notebooks
- Experimentation
- Ad-hoc jobs
- Streaming
- Continuous transformation

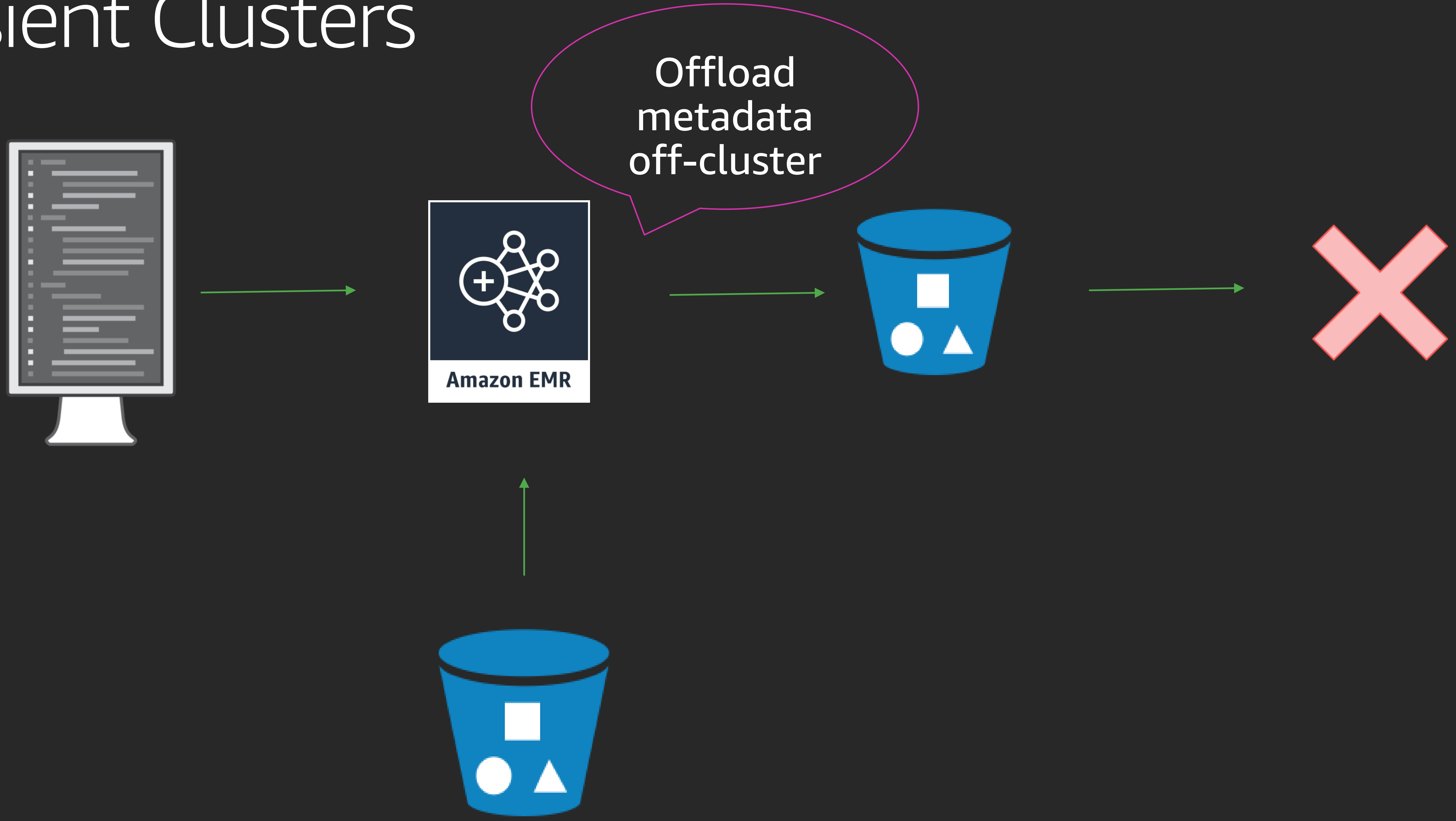
Architecture Patterns – Transient Clusters

“Run stateless, Automate everything, Enable self-service”

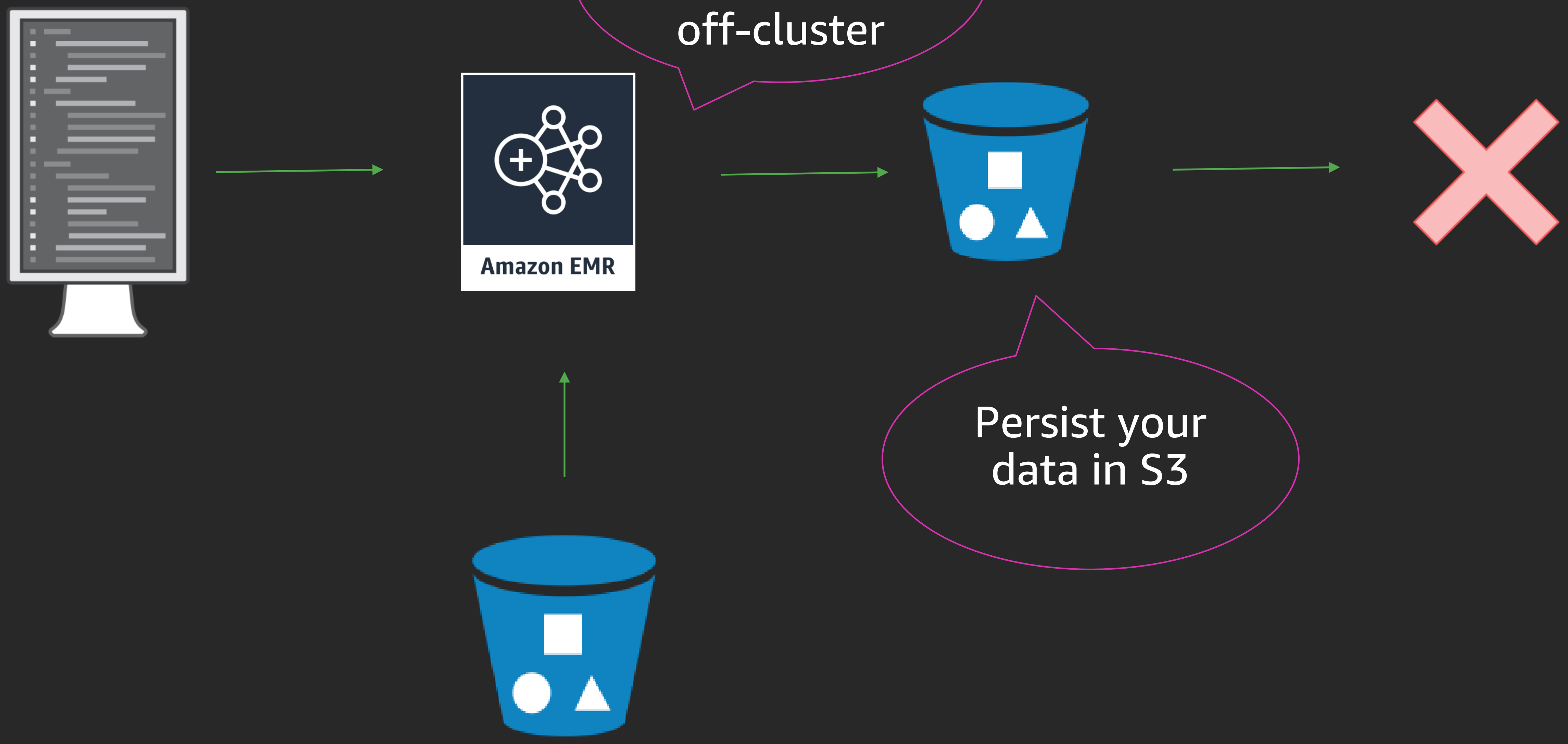
Transient Clusters



Transient Clusters



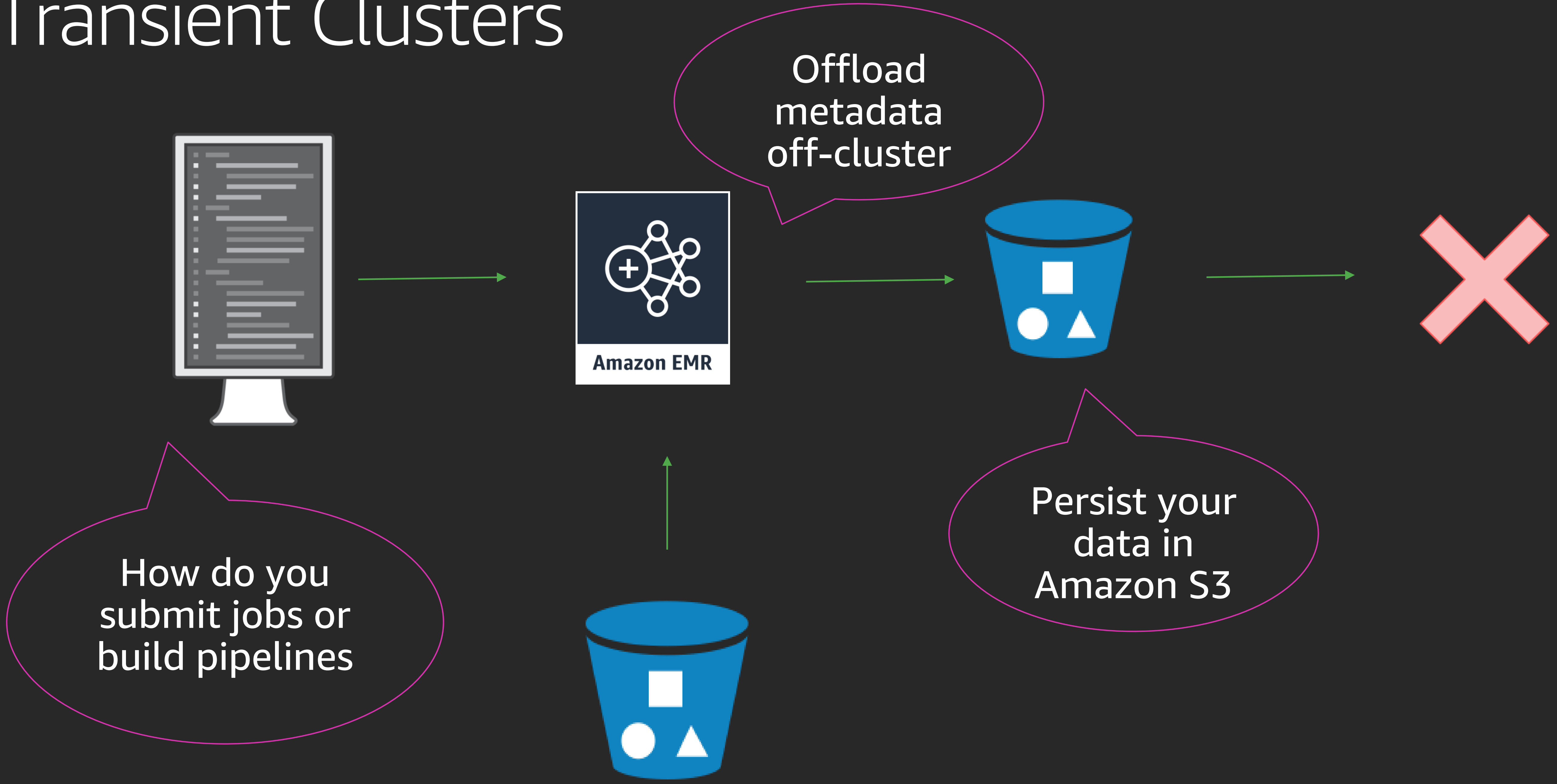
Transient Clusters



Offload metadata off-cluster

Persist your data in S3

Transient Clusters

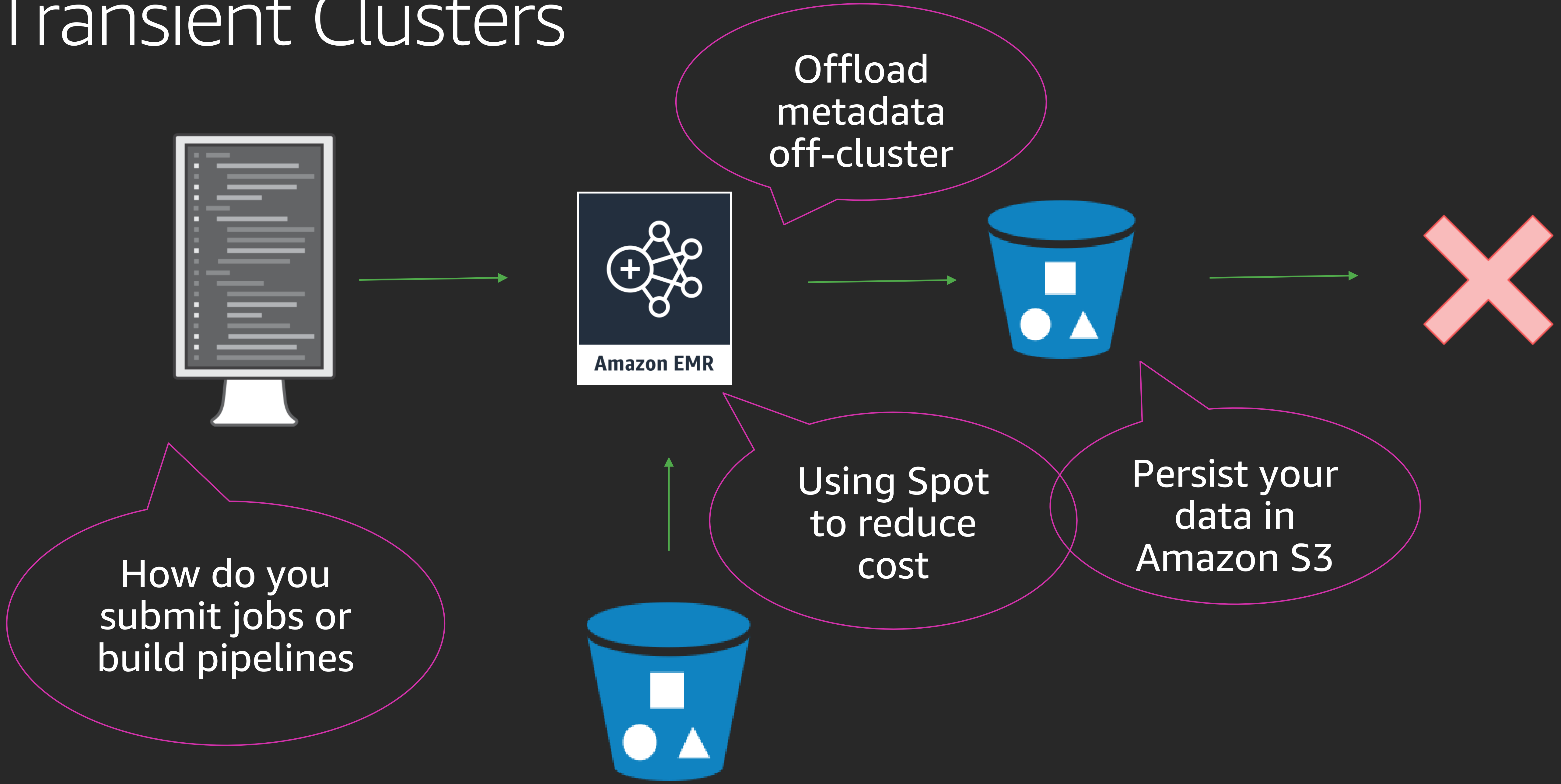


Offload metadata off-cluster

How do you submit jobs or build pipelines

Persist your data in Amazon S3

Transient Clusters



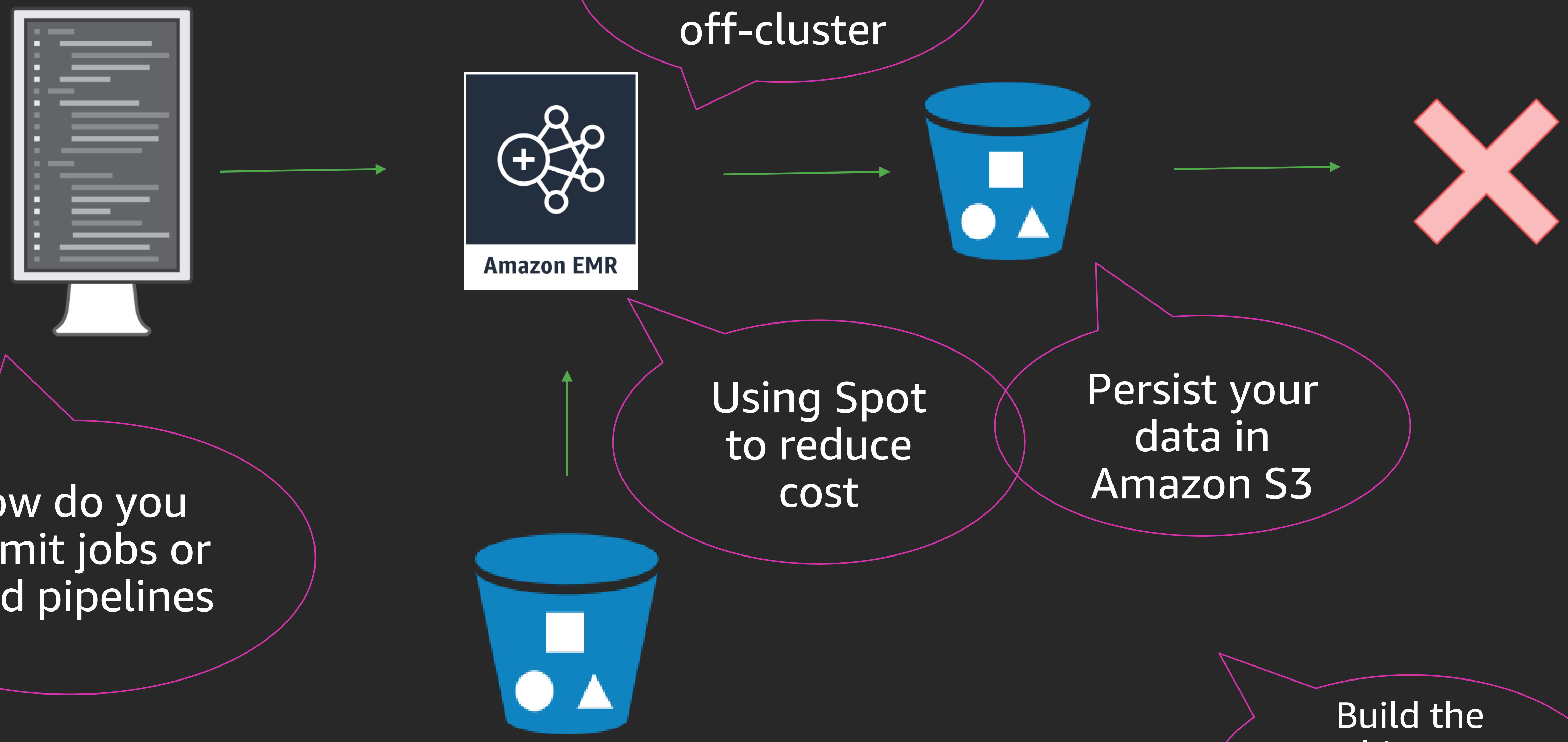
Offload metadata off-cluster

How do you submit jobs or build pipelines

Using Spot to reduce cost

Persist your data in Amazon S3

Transient Clusters



Offload metadata off-cluster

Using Spot to reduce cost

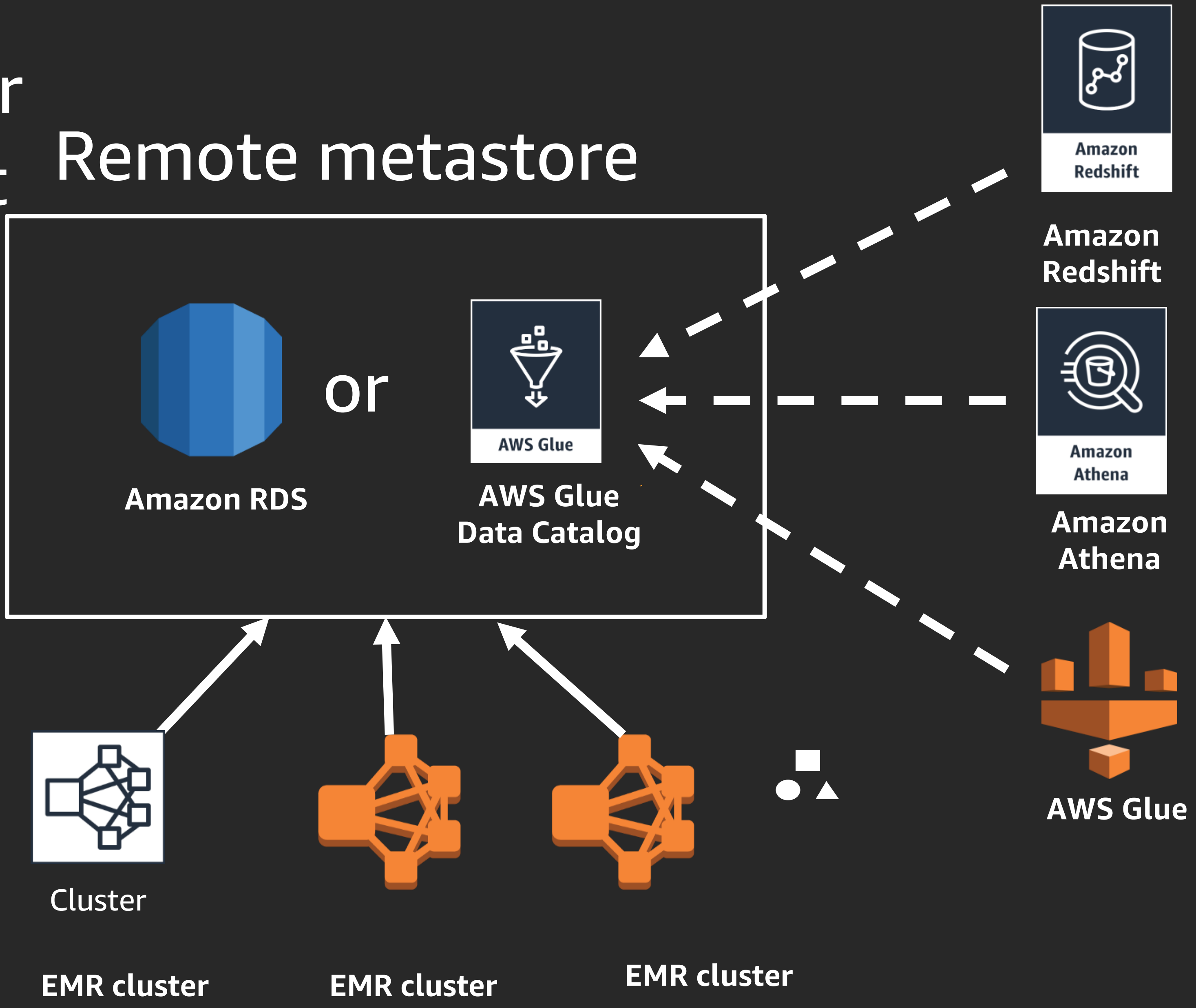
Persist your data in Amazon S3

How do you submit jobs or build pipelines

Build the architecture as a template for your entire org

Run Stateless

- Maintain metastores off cluster
- Faster startup time lowers cost



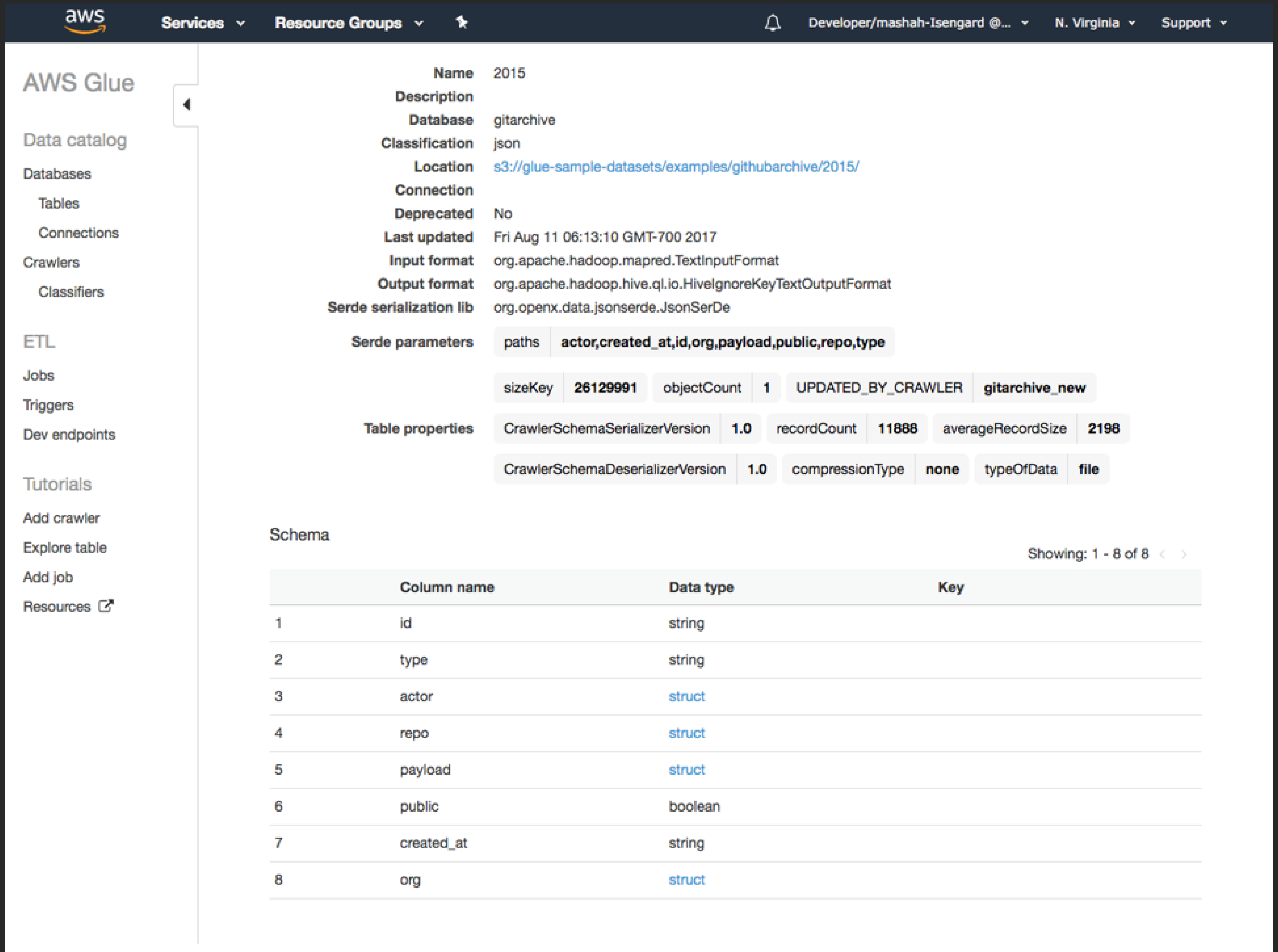
Connecting to Hive Metastore

```
aws emr create-cluster

--release-label emr-5.19.0

--instance-type m4.large --instance-count 2  --applications Name=Hive

--configurations ./hiveConfiguration.json

--use-default-roles
```

Note: Limit access to elasticmapreduce:DescribeCluster

Use AWS Glue Data Catalog as Common Metadata Store

New in 2018



Name 2015
Description
Database gitarchive
Classification json
Location s3://glue-sample-datasets/examples/githubarchive/2015/
Connection
Deprecated No
Last updated Fri Aug 11 06:13:10 GMT-700 2017
Input format org.apache.hadoop.mapred.TextInputFormat
Output format org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib org.openx.data.jsonserde.JsonSerDe
Serde parameters paths **actor,created_at,id,org,payload,public,repo,type**
Table properties sizeKey **26129991** objectCount **1** UPDATED_BY_CRAWLER **gitarchive_new** CrawlerSchemaSerializerVersion **1.0** recordCount **11888** averageRecordSize **2198** CrawlerSchemaDeserializerVersion **1.0** compressionType **none** typeOfData **file**

Schema

Showing: 1 - 8 of 8

	Column name	Data type	Key
1	id	string	
2	type	string	
3	actor	struct	
4	repo	struct	
5	payload	struct	
6	public	boolean	
7	created_at	string	
8	org	struct	

- Support for Spark, Hive, and Presto
- Auto-generate schema and partitions
- Managed table updates
- Fine-grained access control to databases and tables
- Cross-account data catalog access

Connecting to the AWS Glue Data Catalog

```
[
  {
    "Classification": "hive-site",
    "Properties": {
      "hive.metastore.client.factory.class": “
       com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
      "hive.metastore.glue.catalogid": "acct-id”
    }
  }
]
```

Fine Grained Access Control on Glue Data Catalog

New in 2018

```
"arn:aws:glue:us-east-1:123456789012:catalog",
"arn:aws:glue:us-east-1:123456789012:database/finegrainaccess",
"arn:aws:glue:us-east-1:123456789012:tables/finegrainaccess/dev_*"
```

Restrict access to a catalog

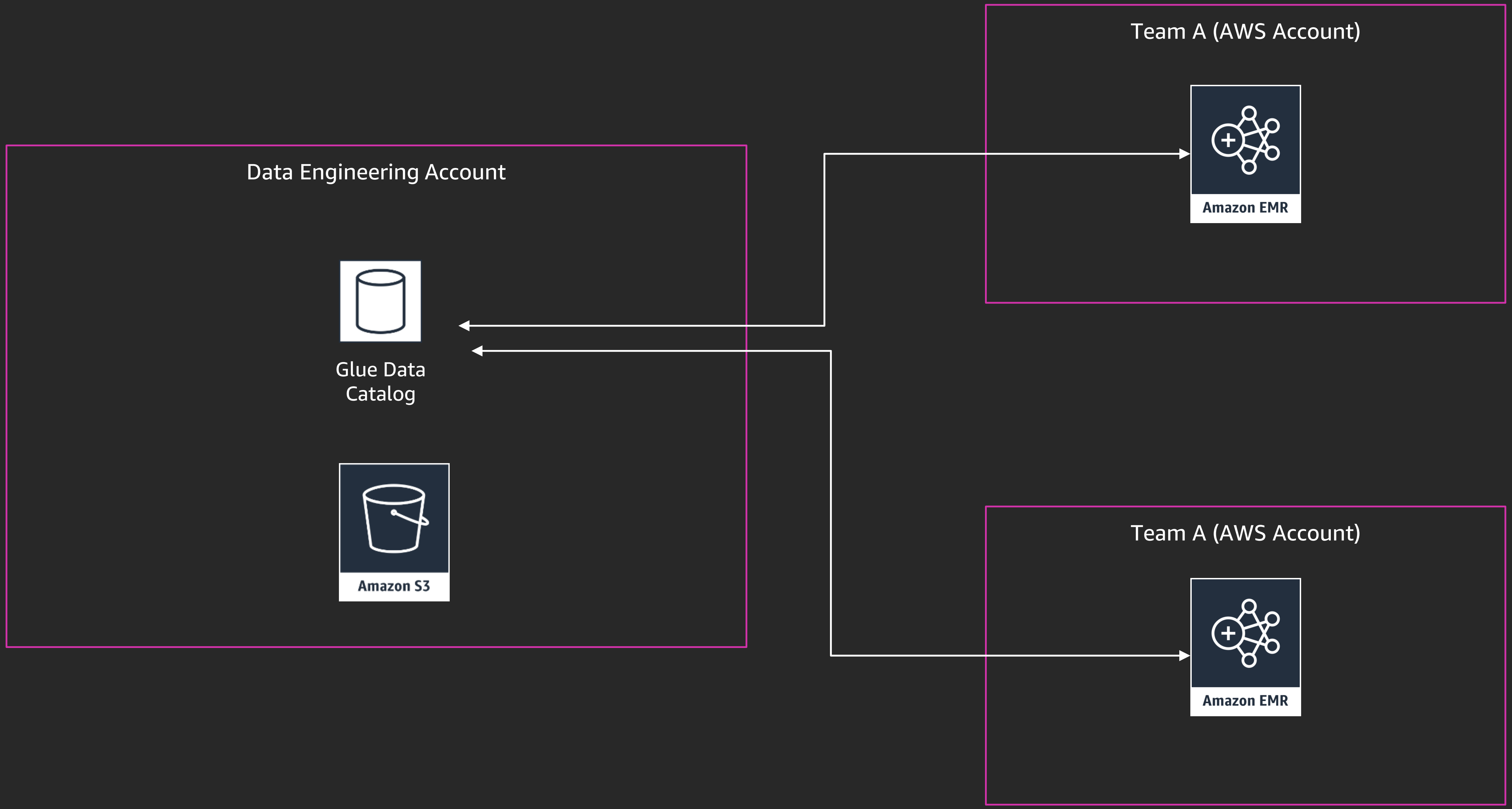
Restrict access to a table

Restrict access to only a certain account

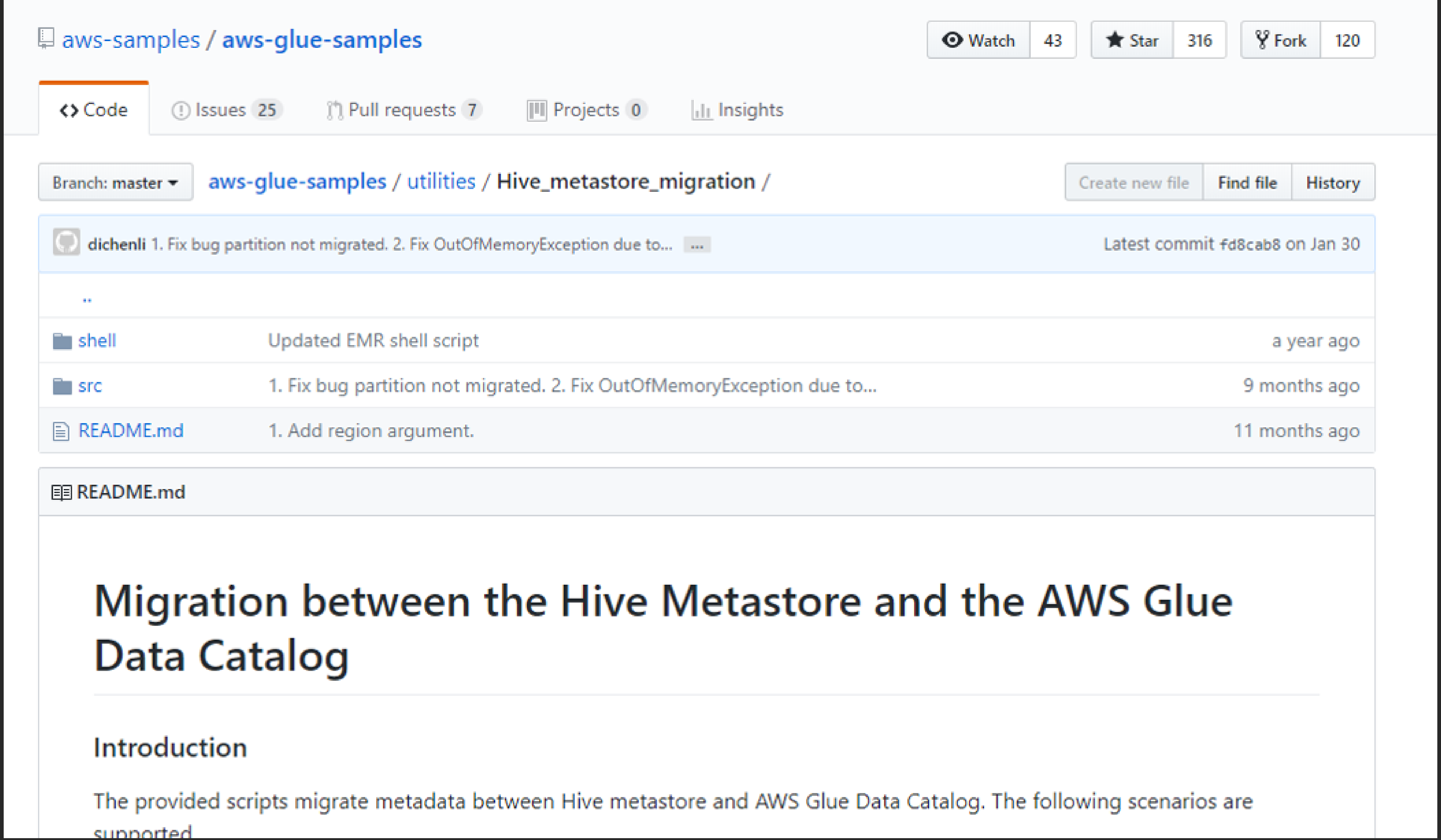
Restrict access to a tables starting with dev

Cross-Account AWS Glue Data Catalog Access

New in 2018



Migrate from Hive Metastore to AWS Glue Data Catalog



aws-glue-samples / utilities / Hive_metastore_migration /

dichenli 1. Fix bug partition not migrated. 2. Fix OutOfMemoryException due to... … Latest commit fd8cab8 on Jan 30

..

shell	Updated EMR shell script	a year ago
src	1. Fix bug partition not migrated. 2. Fix OutOfMemoryException due to...	9 months ago
README.md	1. Add region argument.	11 months ago

README.md

Migration between the Hive Metastore and the AWS Glue Data Catalog

Introduction

The provided scripts migrate metadata between Hive metastore and AWS Glue Data Catalog. The following scenarios are supported

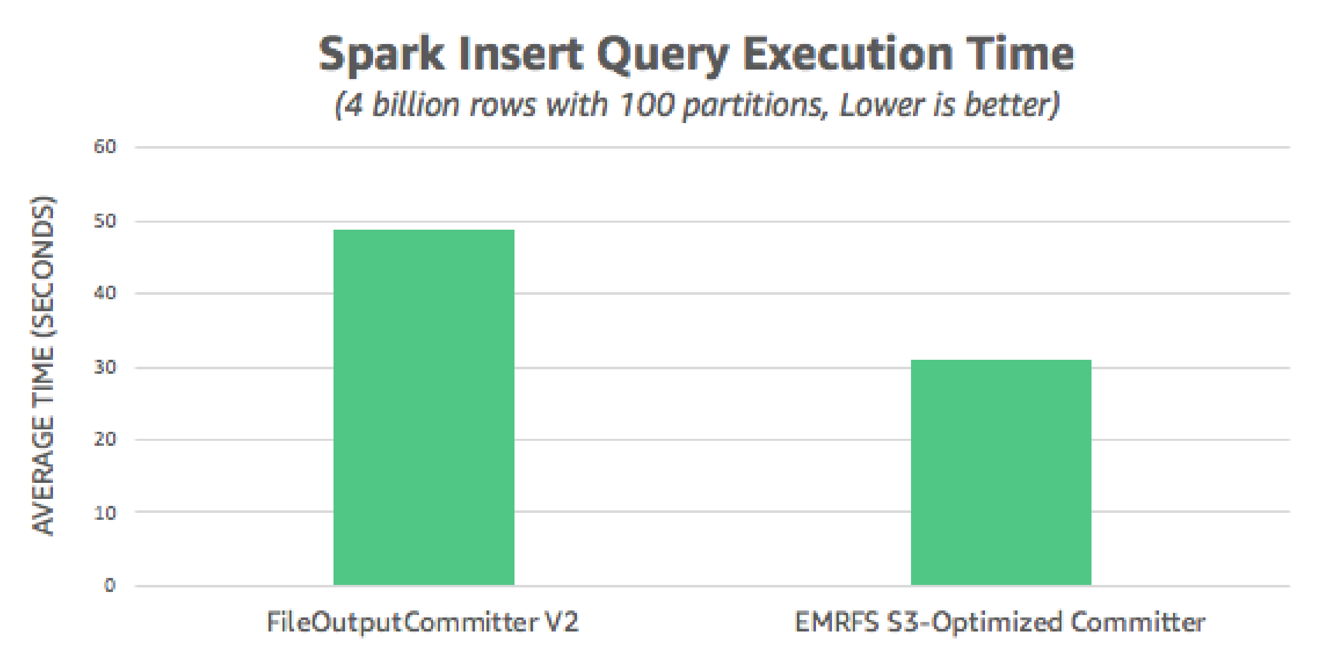
Using Amazon S3 to persist your data

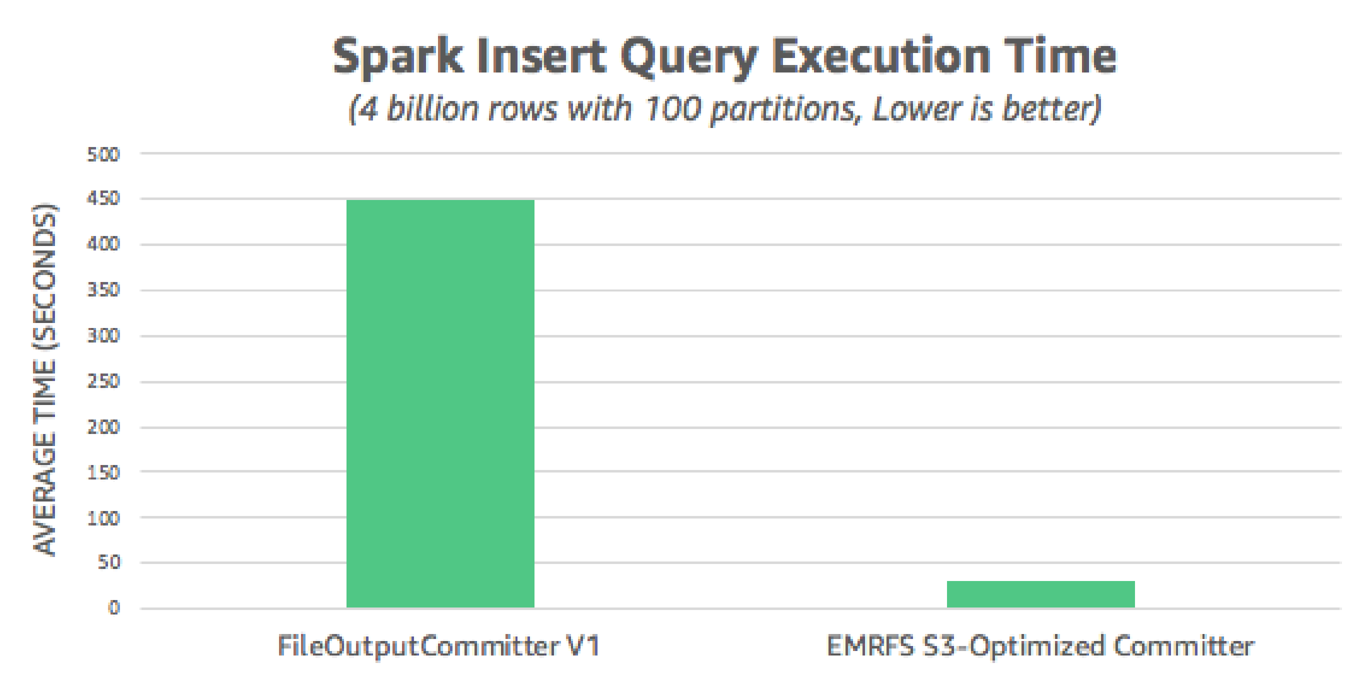
- Decoupling Storage and Compute
- Read depends on aggregate throughput
- Write committers
  - File Output Committer
  - File Output Committer 2
  - Direct Write Committer
  - EMRFS S3-optimized Committer (Spark and Parquet only)

New in 2018

EMRFS performance

- Committer Performance
- Allows you to enable speculative execution, which improves straggler performance
- Does not need EMRFS Consistent View
- Faster performance with EMRFS Consistent View enabled





Optimized committer performance improvements

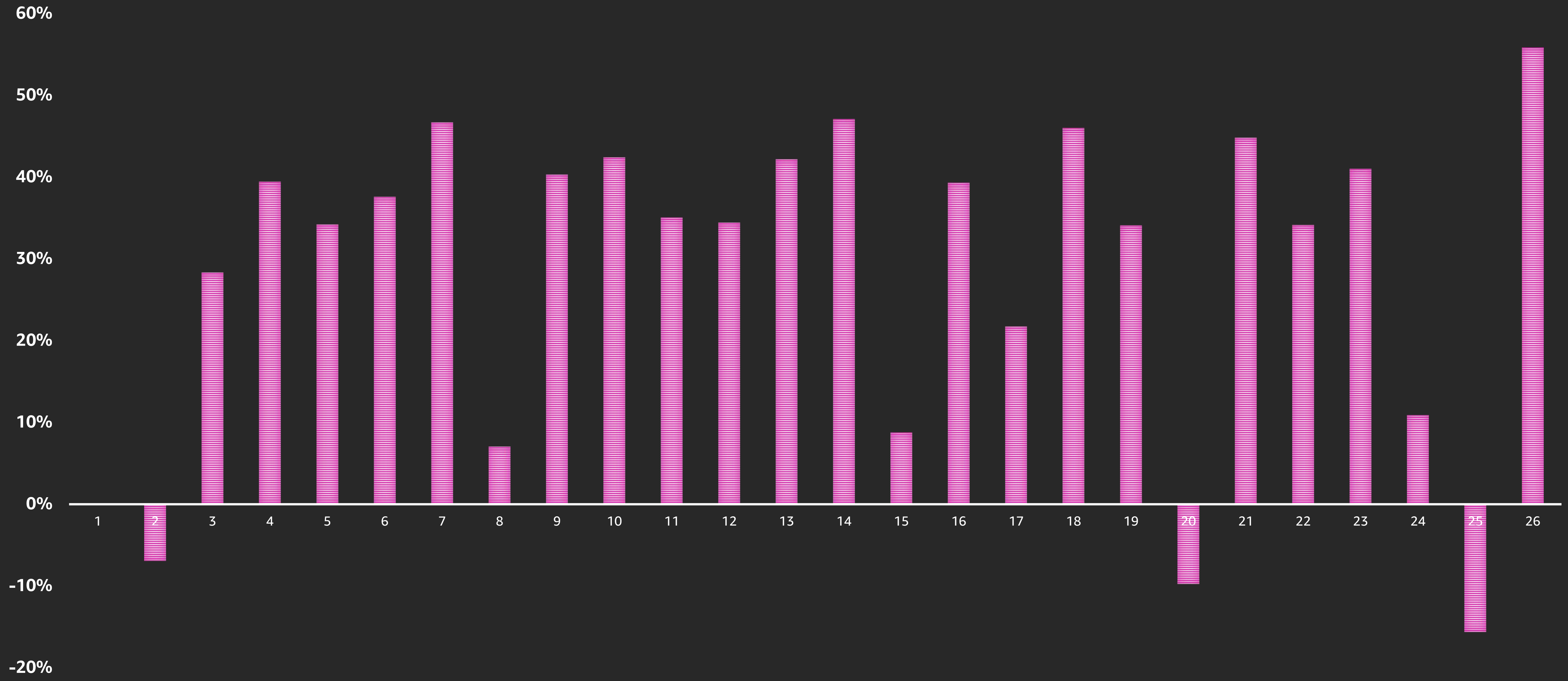
Integration with S3-Select

New in 2018

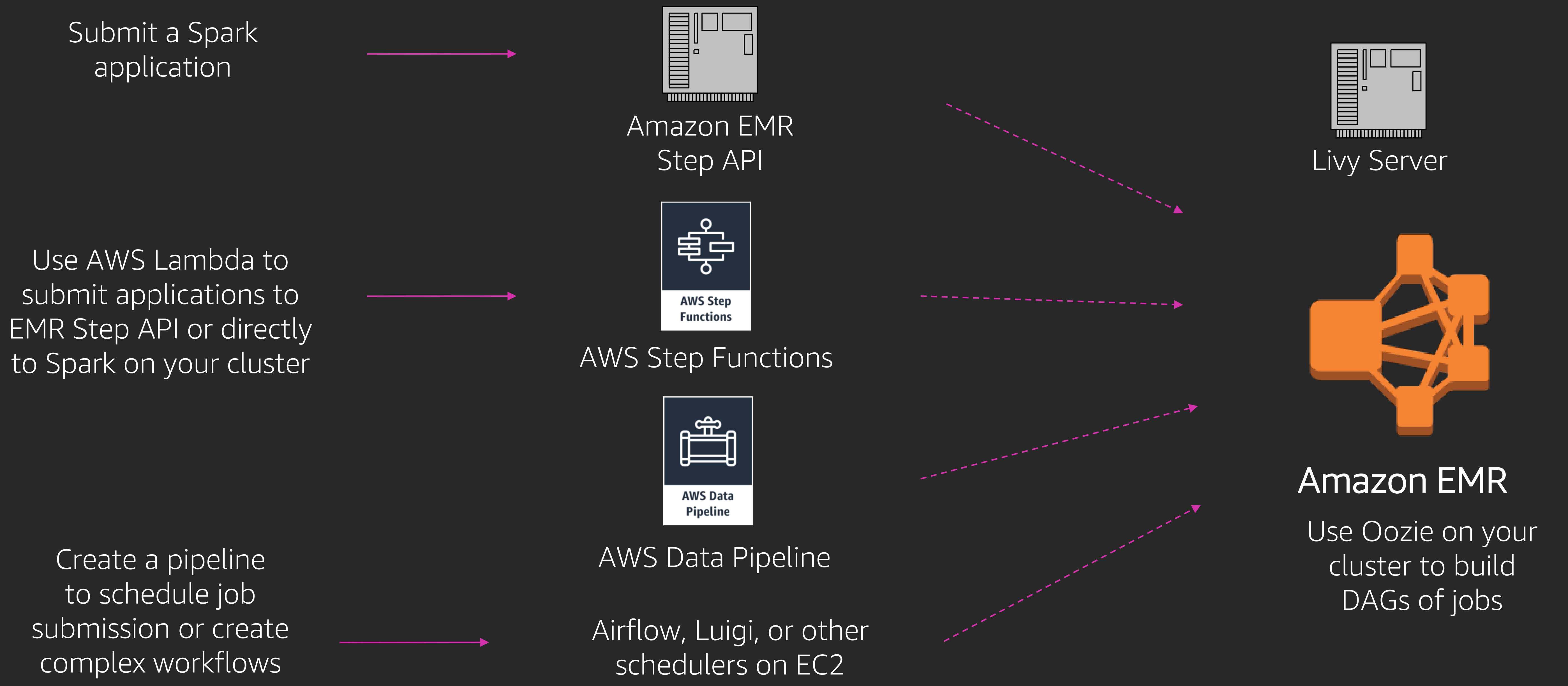
Available in Spark, Presto and Hive

```
spark
   .read
   .format("s3selectCSV") // "s3selectJson" for Json
   .schema(...) // optional, but recommended
   .options(...) // optional
   .load("s3://path/to/my/datafiles")
```

Presto/S3Select TPCDS-100 Query speed-up



Options to submit jobs



Submit a Spark application

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

Create a pipeline to schedule job submission or create complex workflows

Amazon EMR Step API

AWS Step Functions

AWS Data Pipeline

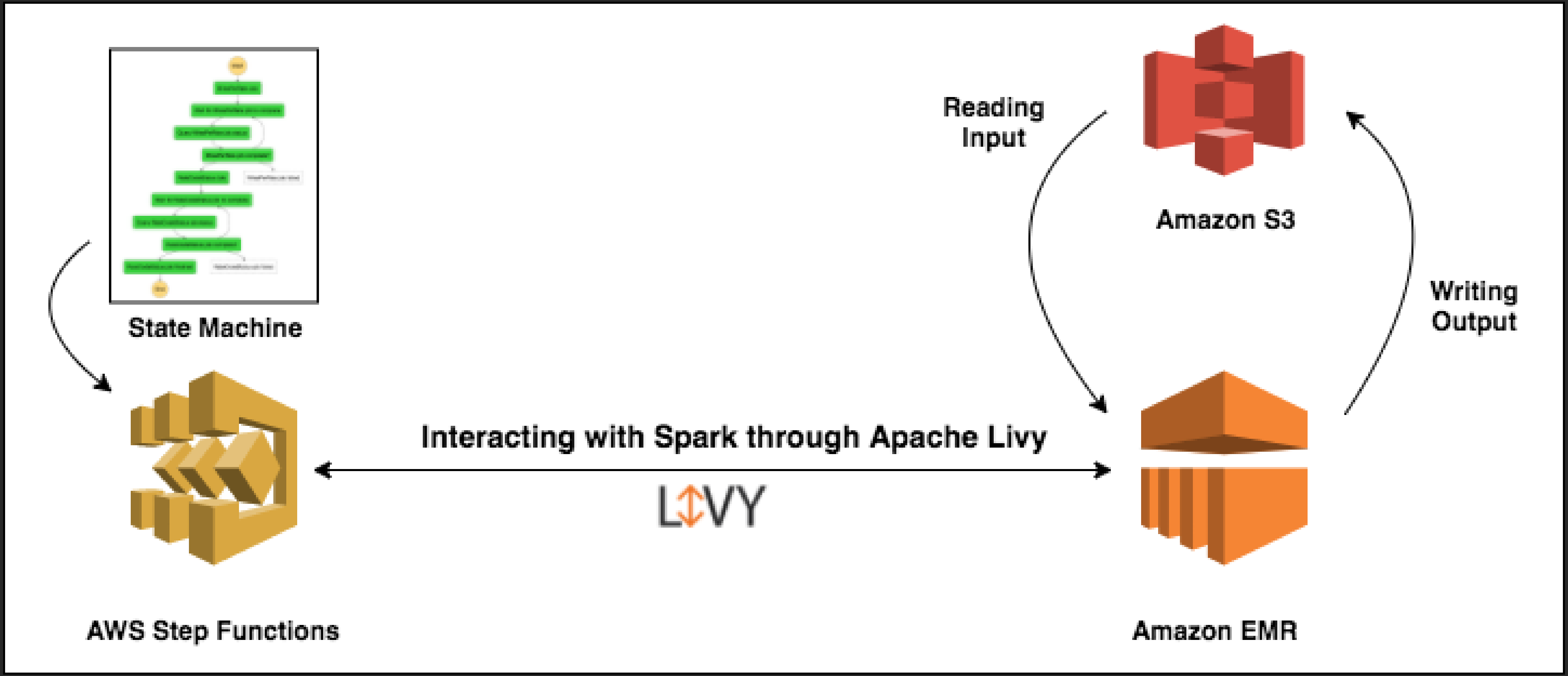
Airflow, Luigi, or other schedulers on EC2

Livy Server

Amazon EMR

Use Oozie on your cluster to build DAGs of jobs

Serverless Job submission: Step Functions & Livy



Serverless Job submission

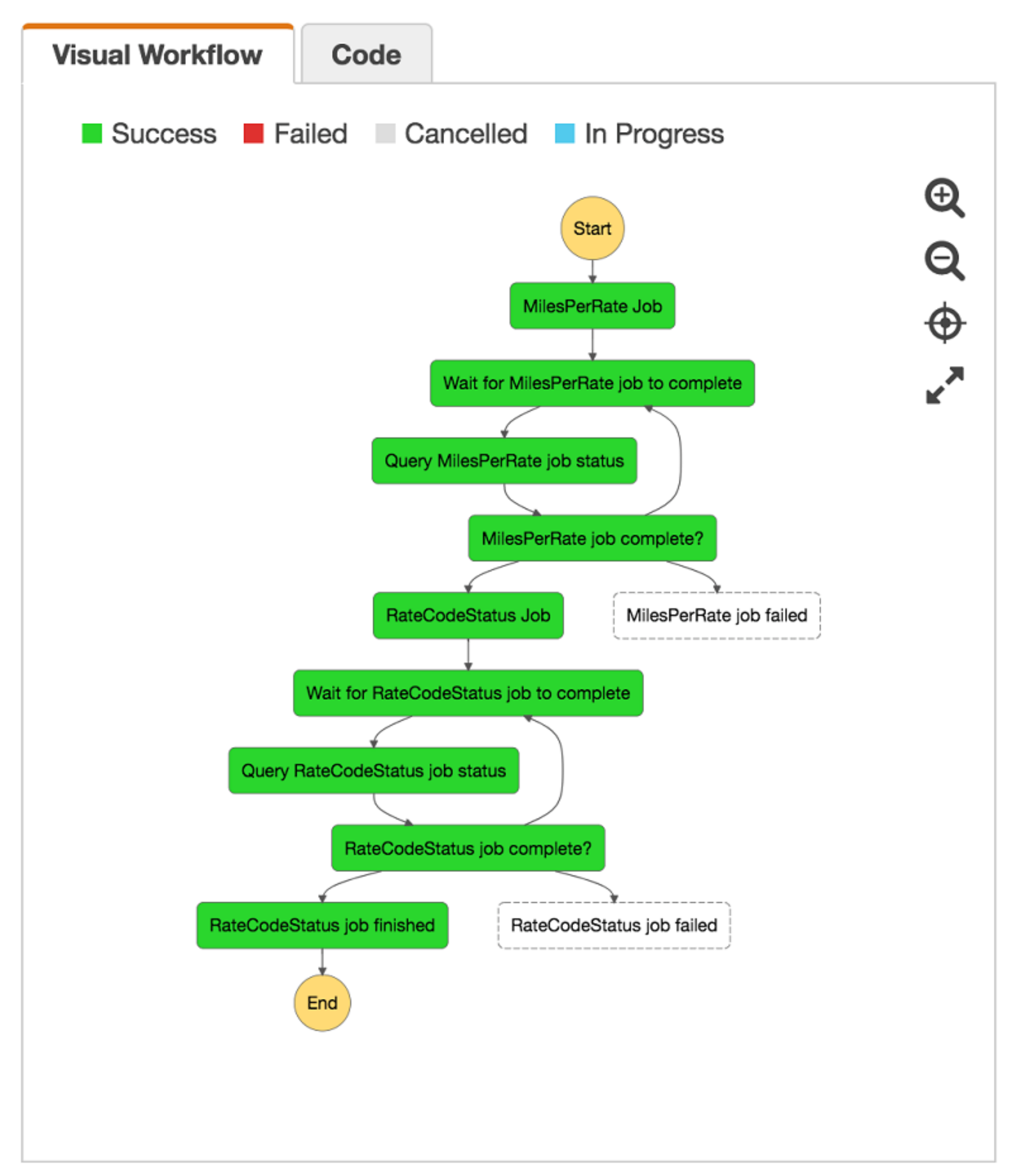
Task state – Invokes a Lambda function.

The first Task state submits the Spark job on Amazon EMR,
Next Task state is used to retrieve the previous Spark job status.

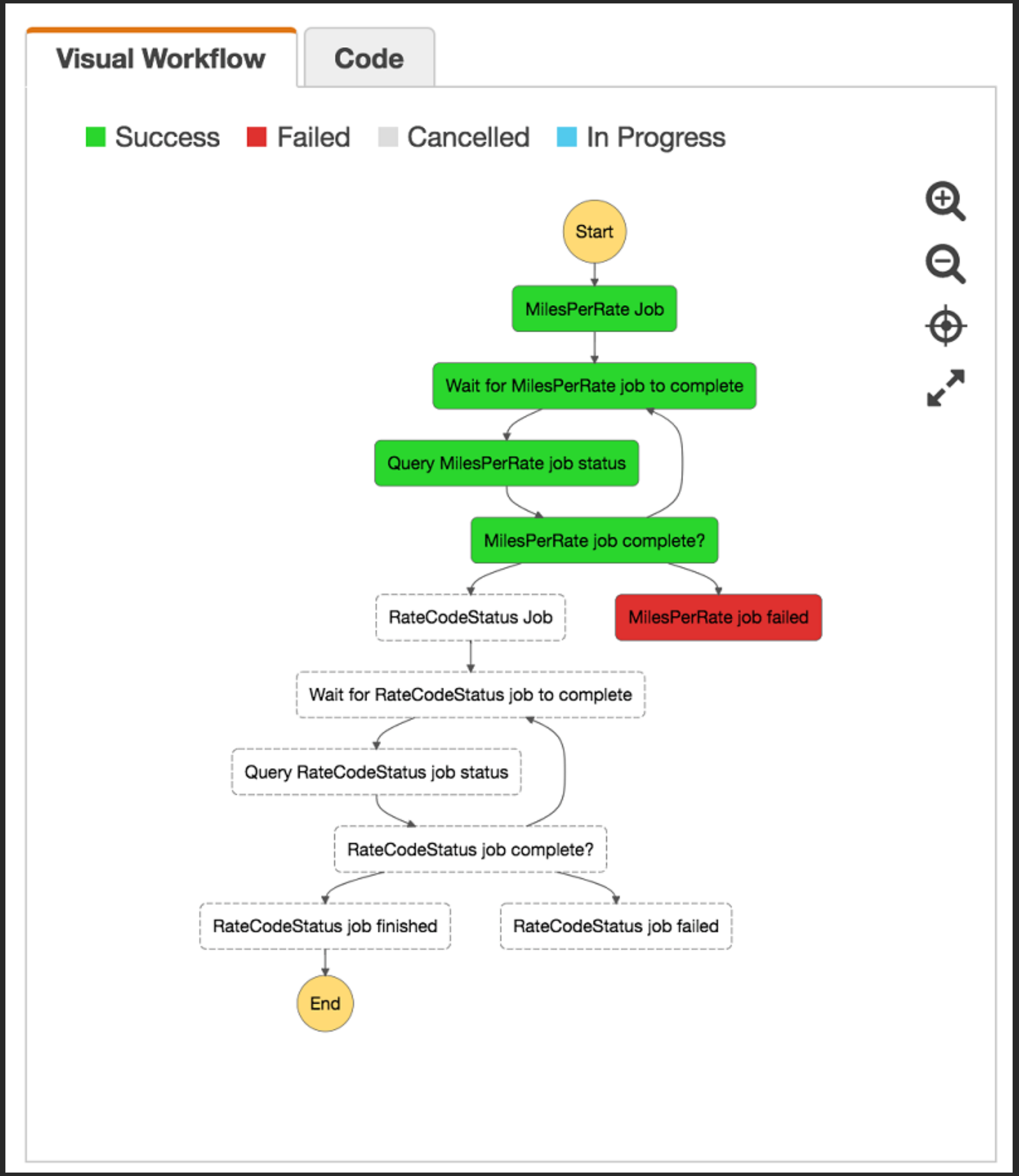
Wait state – Pauses the state machine until a job completes execution.

Choice state – Each Spark job execution can return a failure, an error, or a success state. Choice state to create a rule that specifies the next action or Next Step

Visual Graph Step Functions

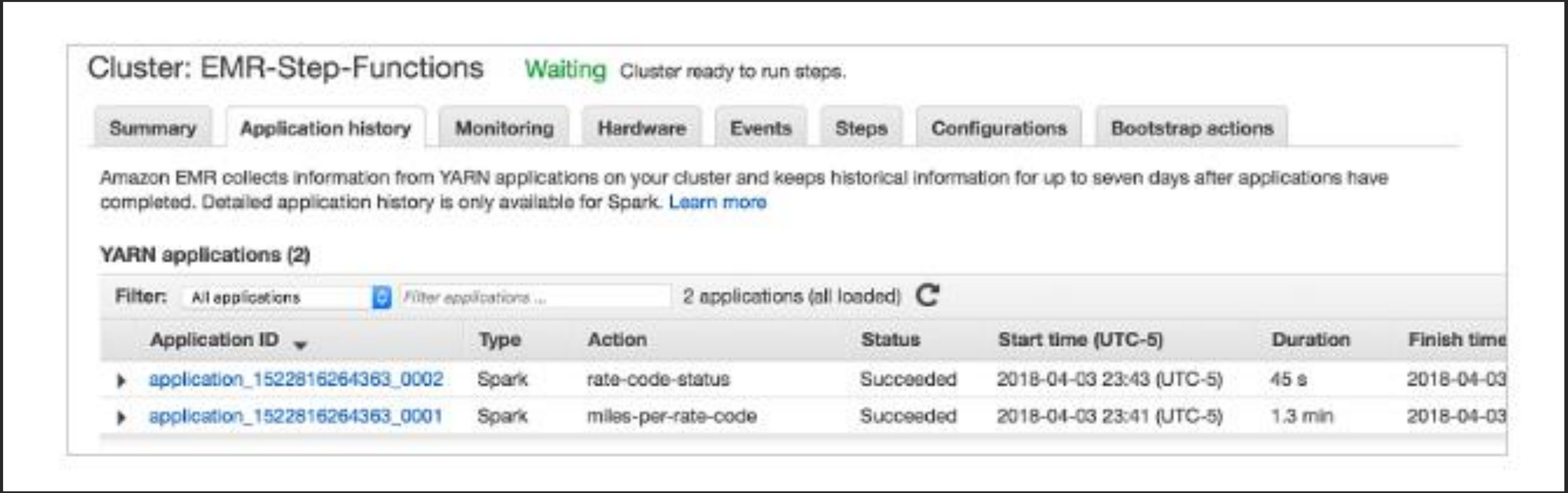


Visual Workflow
Code
🟩 Success
🟥 Failed
⬜ Cancelled
🟦 In Progress
Start
MilesPerRate Job
Wait for MilesPerRate job to complete
Query MilesPerRate job status
MilesPerRate job complete?
RateCodeStatus Job
MilesPerRate job failed
Wait for RateCodeStatus job to complete
Query RateCodeStatus job status
RateCodeStatus job complete?
RateCodeStatus job finished
RateCodeStatus job failed
End



Visual Workflow
Code
🟩 Success
🟥 Failed
⬜ Cancelled
🟦 In Progress
Start
MilesPerRate Job
Wait for MilesPerRate job to complete
Query MilesPerRate job status
MilesPerRate job complete?
RateCodeStatus Job
MilesPerRate job failed
Wait for RateCodeStatus job to complete
Query RateCodeStatus job status
RateCodeStatus job complete?
RateCodeStatus job finished
RateCodeStatus job failed
End

Spark Application History on the EMR Console



Cluster: EMR-Step-Functions Waiting Cluster ready to run steps.

Amazon EMR collects information from YARN applications on your cluster and keeps historical information for up to seven days after applications have completed. Detailed application history is only available for Spark. Learn more

YARN applications (2)

Filter: All applications | Filter applications ... | 2 applications (all loaded)

Application ID	Type	Action	Status	Start time (UTC-5)	Duration	Finish time
application_1522816264363_0002	Spark	rate-code-status	Succeeded	2018-04-03 23:43 (UTC-5)	45 s	2018-04-03
application_1522816264363_0001	Spark	miles-per-rate-code	Succeeded	2018-04-03 23:41 (UTC-5)	1.3 min	2018-04-03

AWS re:Invent

aws

Advanced orchestration

Trigger jobs using CloudWatch Events

- Based on arrival of an event
- Based on an schedule

Alerting in case of Failures

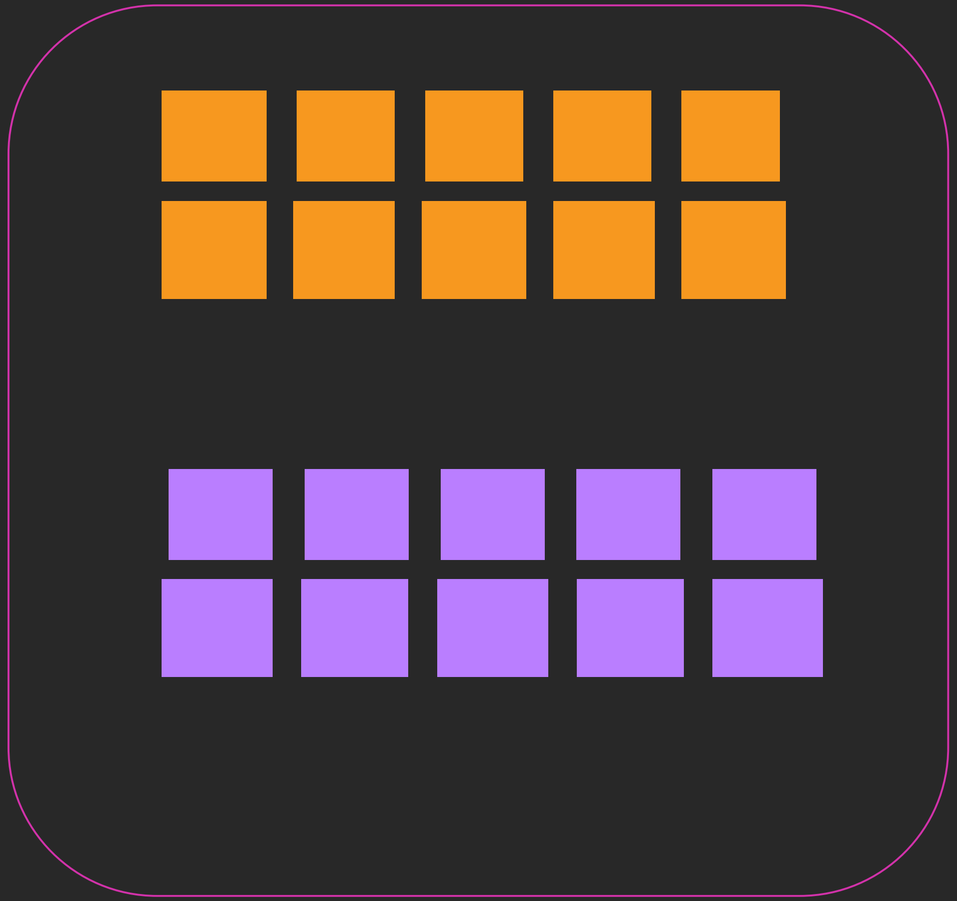
Parallel Execution Steps

Scale up with Spot Instances

10 node cluster running for 14 hours

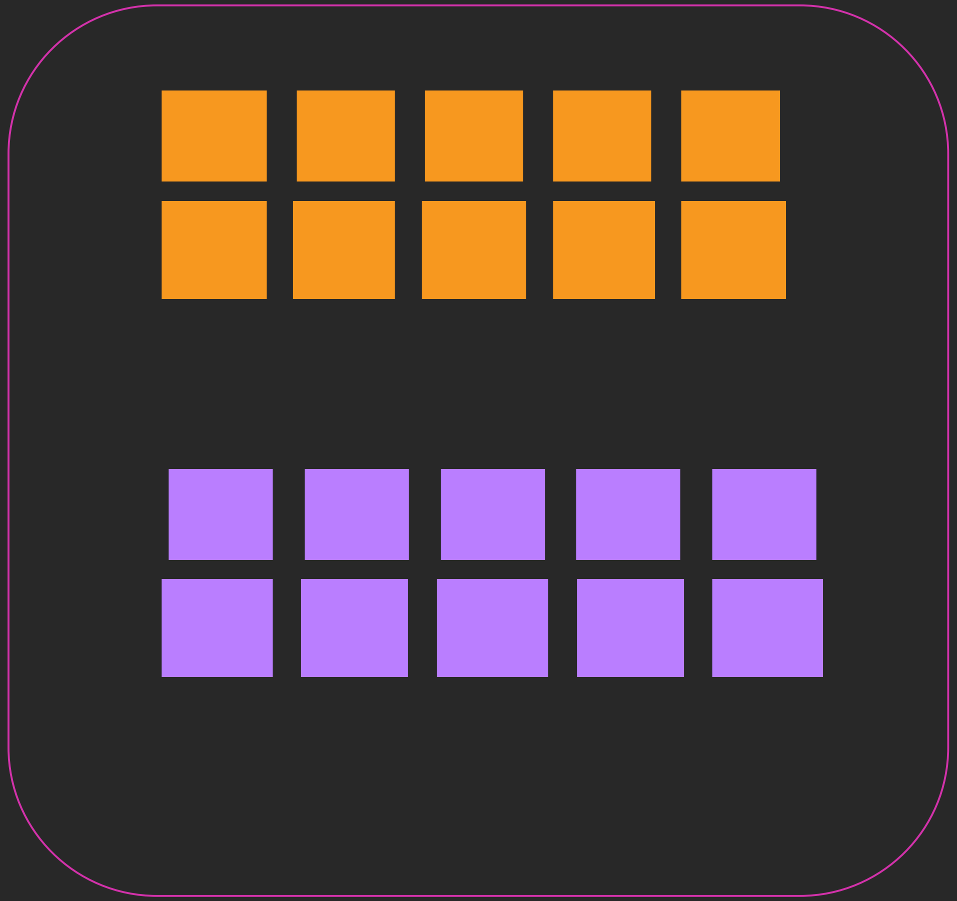
Cost = 1.0 * 10 * 14 = $140

Scale Up Cluster with Spot Instances



Add 10 more nodes on Spot

Scale Up Cluster with Spot Instances



20 node cluster running for 7 hours

Cost = 1.0 * 10 * 7 = $70

= 0.5 * 10 * 7 = $35

Total $105

AWS re:Invent

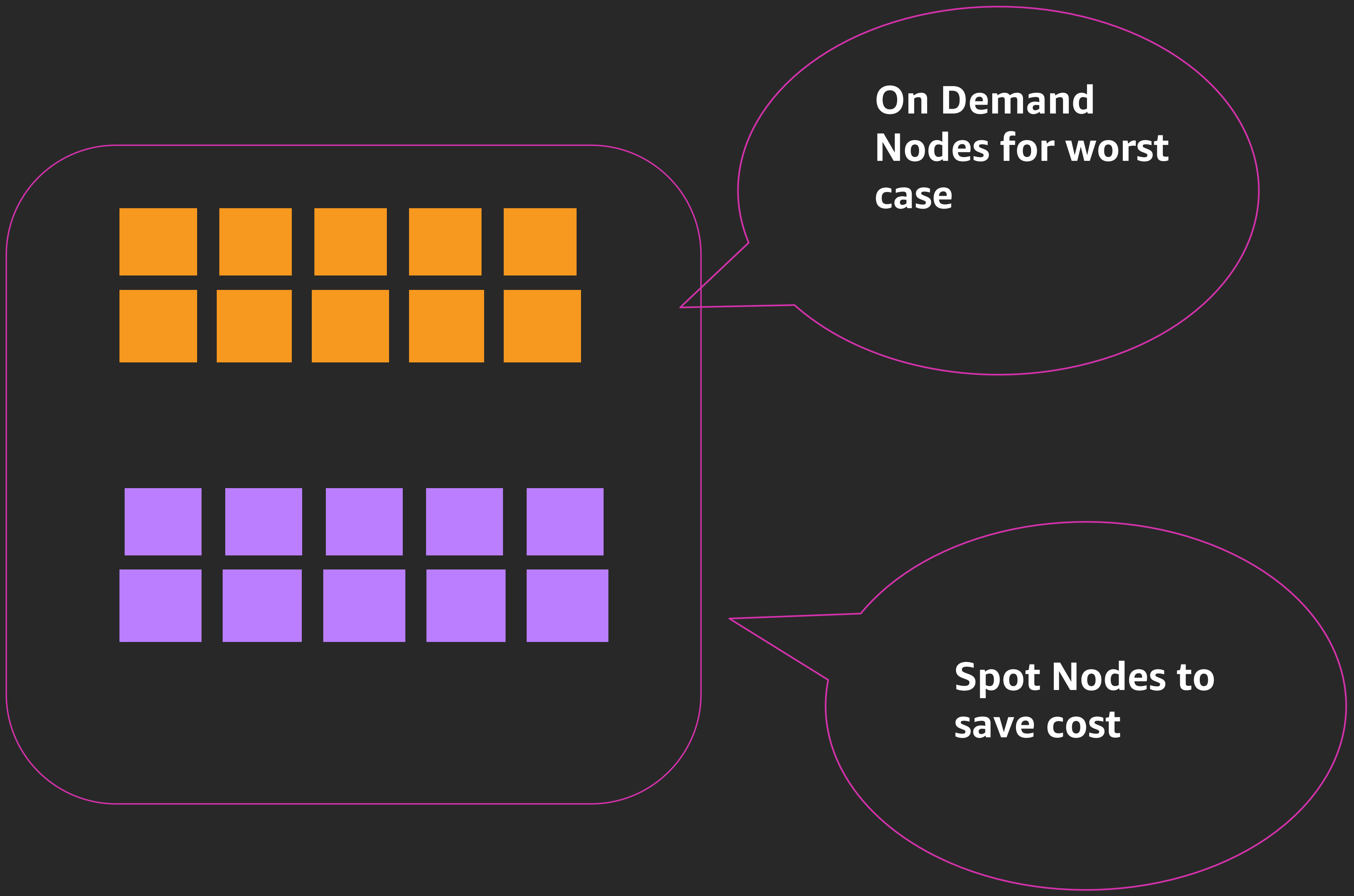
aws

Scale Up Cluster with Spot Instances

50 % less run-time (14 → 7)

25% less cost (140 → 105)

Scale Up Cluster with Spot Instances



On Demand Nodes for worst case

Spot Nodes to save cost

AWS re:Invent

aws

What do customers tell us about Spot

Capacity Related

“I need a instance type and the capacity is not available”

“I am AZ agnostic, which AZ should I use to get the cheapest capacity”

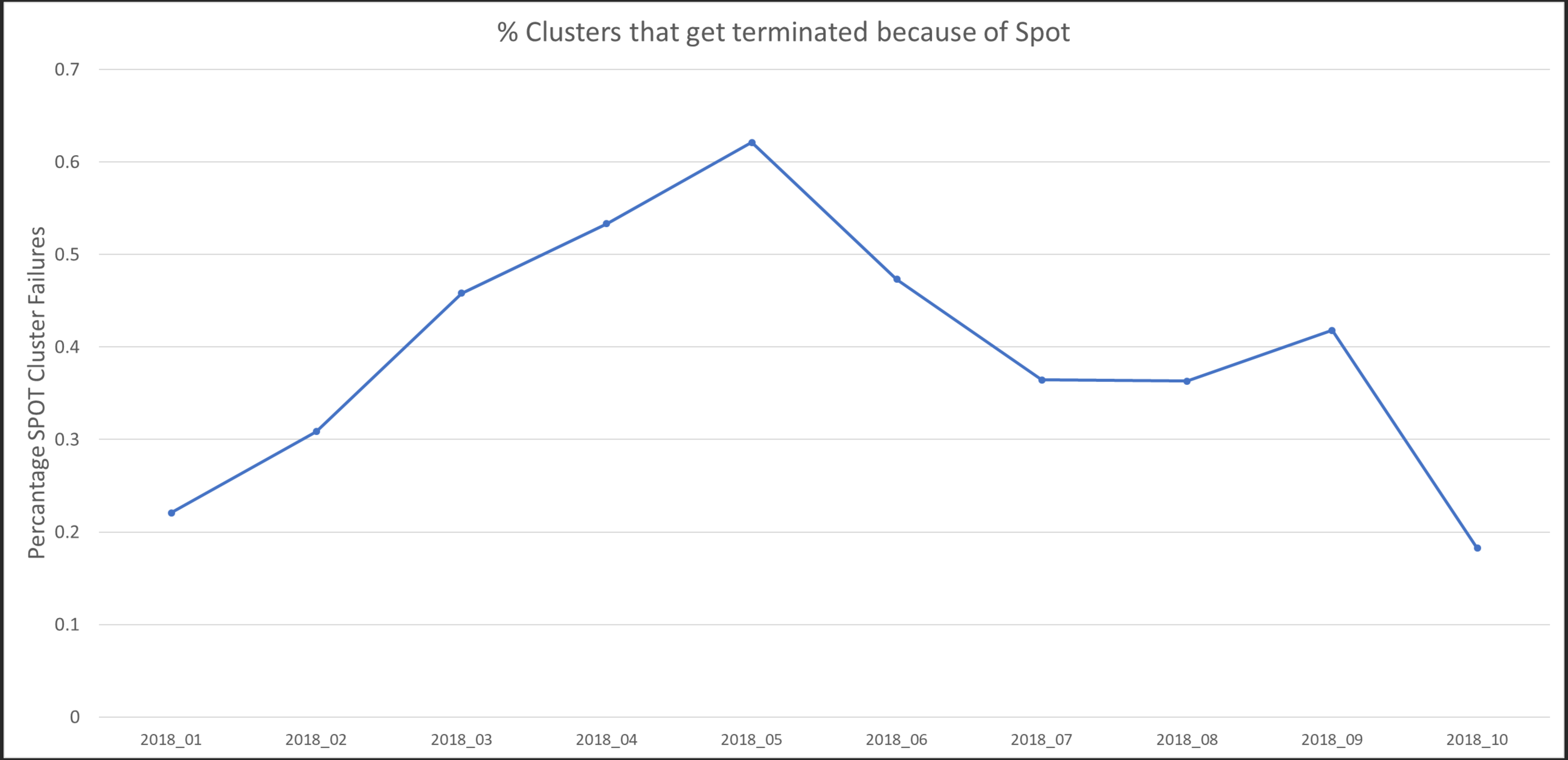
Interruption Related

“I was able to receive the capacity, but I was interrupted”

“I lost the cluster because only one instance was interrupted”

“How does Spark behave with interruption”

Spot Interruption



3 Reasons for Spot Clusters to fail

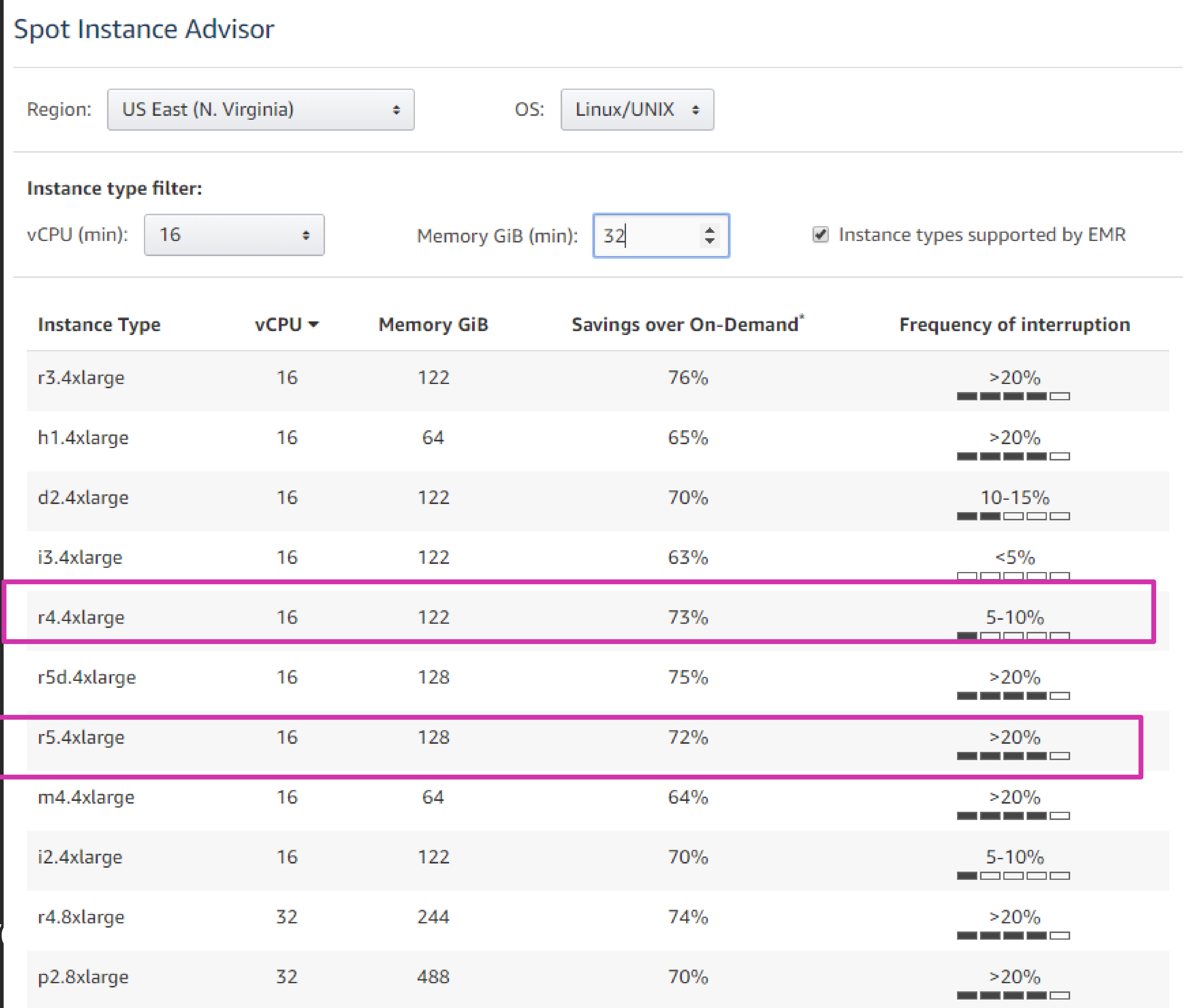
1. Requested Capacity in a certain instance type/AZ was not available
2. Requested Capacity in a certain AZ was not available
3. Termination of a single instance caused the entire cluster to fail

3 Reasons for Spot Clusters to fail

1. Requested Capacity in a certain instance type/AZ was not available
2. Requested Capacity in a certain AZ was not available

Requested capacity available in different AZ or similar instance type

3. Termination of a single instance caused the cluster to fail



Spot Instance Advisor

Region: US East (N. Virginia)   OS: Linux/UNIX

Instance type filter:

vCPU (min): 16   Memory GiB (min): 32   ☑ Instance types supported by EMR

Instance Type	vCPU ▾	Memory GiB	Savings over On-Demand*	Frequency of interruption
r3.4xlarge	16	122	76%	>20% ■■■■□
h1.4xlarge	16	64	65%	>20% ■■■■□
d2.4xlarge	16	122	70%	10-15% ■■□□□
i3.4xlarge	16	122	63%	<5% □□□□□
r4.4xlarge	16	122	73%	5-10% ■□□□□
r5d.4xlarge	16	128	75%	>20% ■■■■□
r5.4xlarge	16	128	72%	>20% ■■■■□
m4.4xlarge	16	64	64%	>20% ■■■■□
i2.4xlarge	16	122	70%	5-10% ■□□□□
r4.8xlarge	32	244	74%	>20% ■■■■□
p2.8xlarge	32	488	70%	>20% ■■■■□

3 Reasons for Spot Clusters to fail

1. Requested Capacity in a certain instance type/AZ was not available
2. Requested Capacity in a certain AZ was not available

Requested capacity available in different AZ or similar instance type

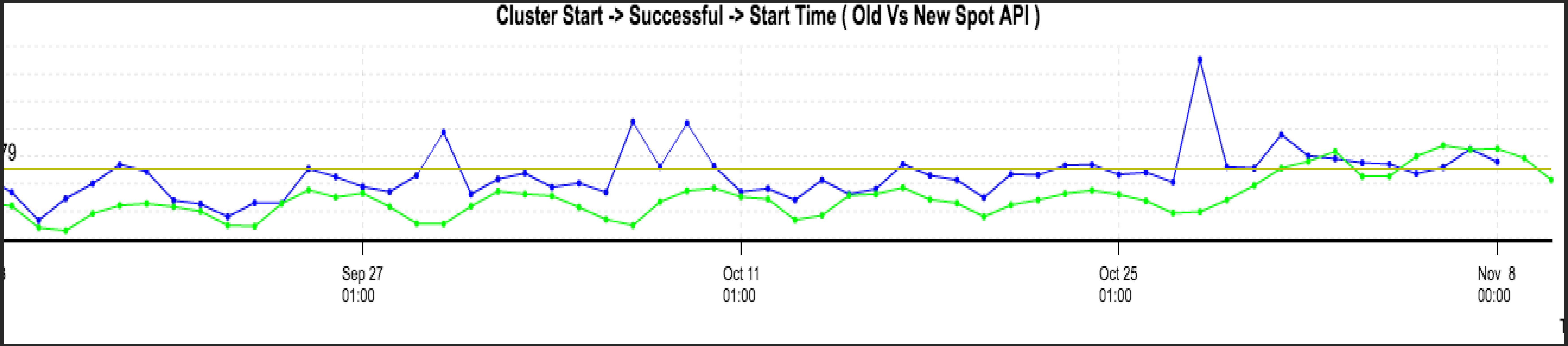
3. Termination of a single instance caused the cluster to fail

New Implementation that changes the way we provision Spot instances

New Implementation

New in 2018

- Faster startup time
- Single node termination does not terminate the cluster
- Optional Spot bids



AWS re:Invent

aws

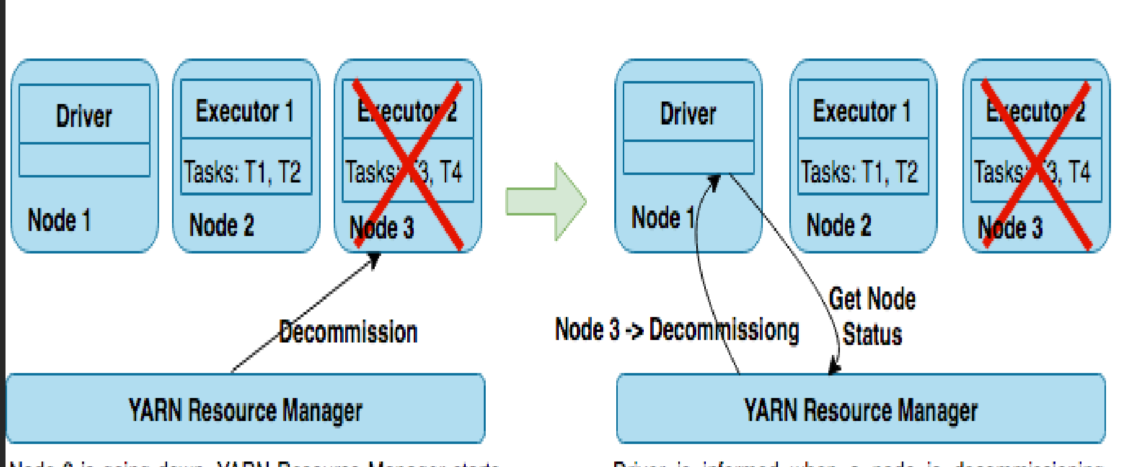
Instance Fleets

- Can mix different instance types in one group
- Can mix different markets (OD or Spot) in one group
- Each instance can have different EBS volume options
- Choose the total capacity (VCPUs) that you want (say 64)
- Diversify your instance types (c3.xlarge, c4.xlarge, c5.xlarge – all with 4 vcpu and 8 GB RAM)
- Don’t specify and AZ and we will find the cheapest one
- Template this configuration

Graceful decommissioning in Spark

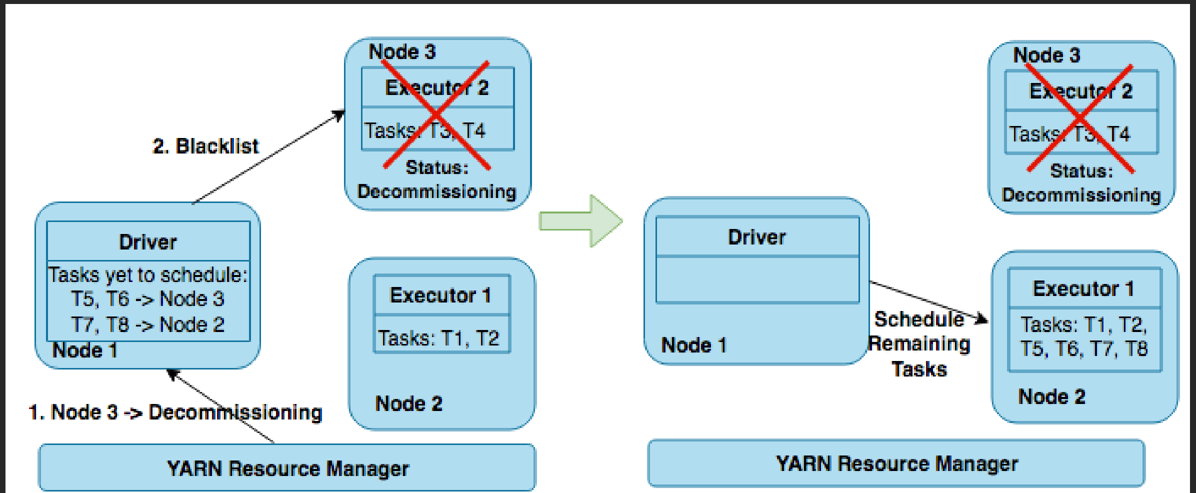
New in 2018

- Decommissioning nodes
- Blacklisting Node
- Contributed to open source



Node 3 is going down. YARN Resource Manager starts decommissioning the node.

Driver is informed when a node is decommissioning while regular checking for node status from YARN



Node 3 is going down and is put to *Decommissioning*. Driver upon getting notified blacklists the node for further task scheduling.

Remaining tasks T5, T6 which would have been scheduled on Node 3 are redirected to Node 2

How do we template best practices across the organization?

Self-service with EMR & AWS Service Catalog

New in 2018

Use Case: Make it **easy** for your customers to launch EMR clusters on AWS while:

- Remove/Reduce EMR/AWS learning curve
- Reduce on-boarding time
- Ensuring Security & Standards
- Adhere to Budgets
- Integrating with Internal Processes & Approval workflows
- Integrate with best practices

EMR with AWS Service Catalog

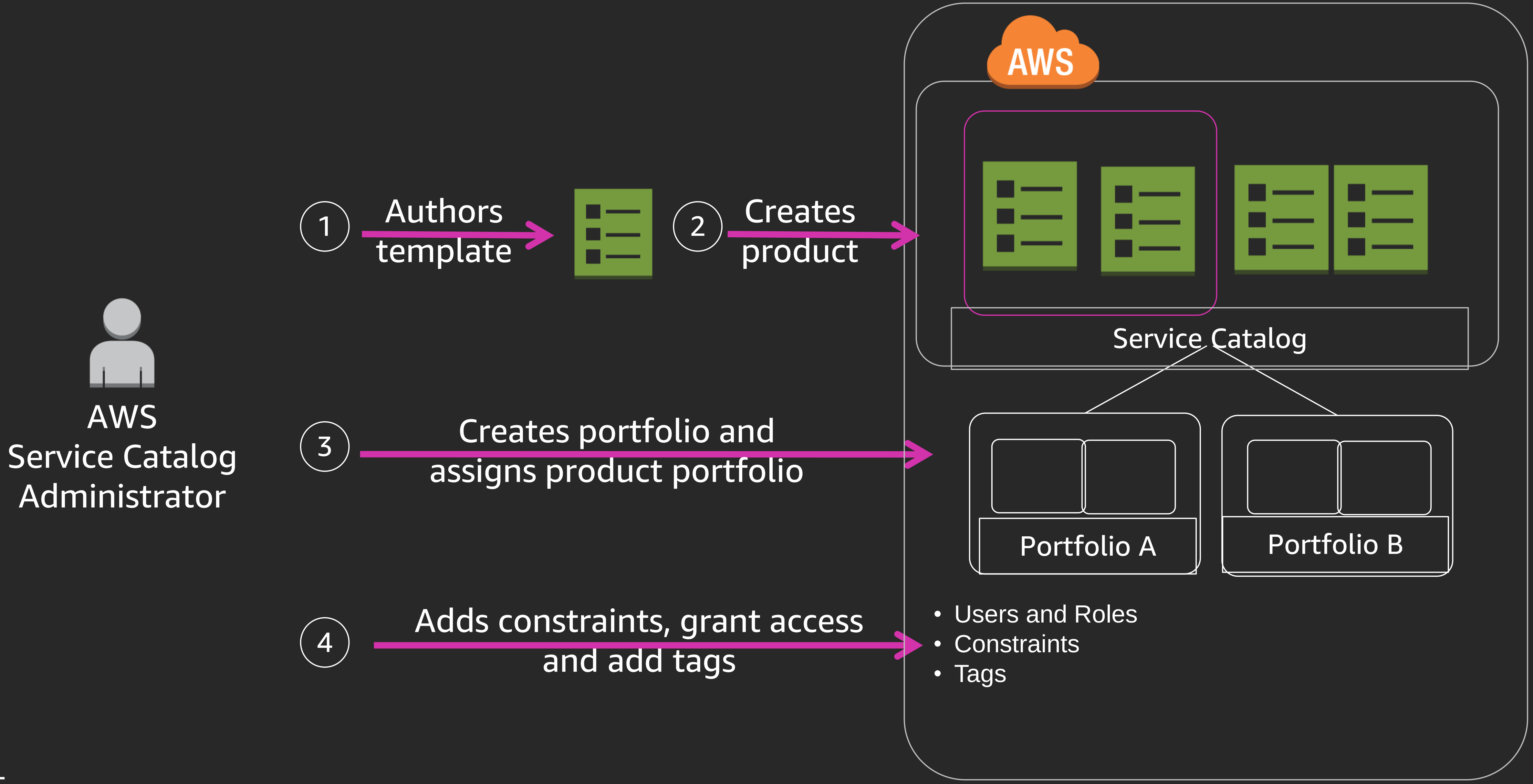
Configure

- Standardize
- Enforce Consistency and Compliance
- Limit Access
- Enforce Tagging, Security Groups

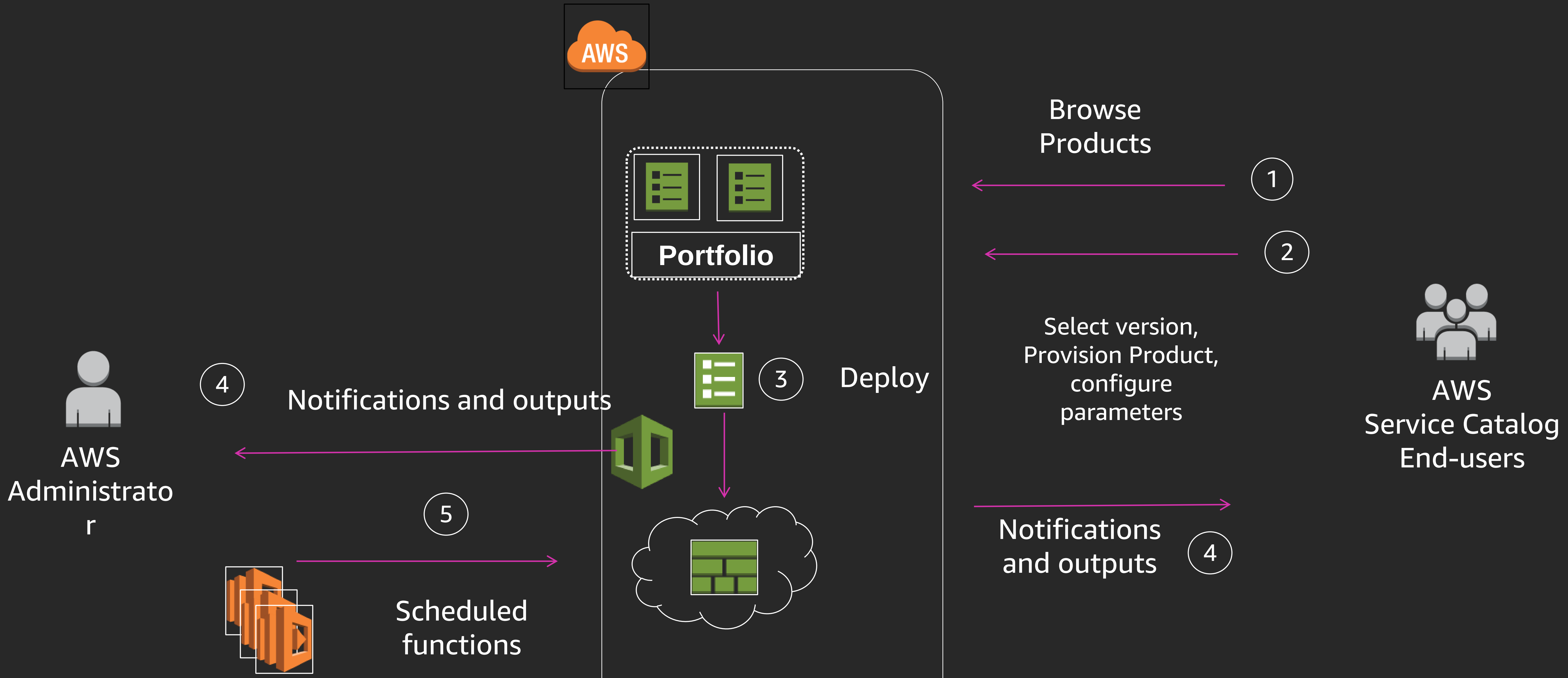
Consume

- Developer Autonomy
- One-Stop Shop
- Automate Deployments
- Agile Governance

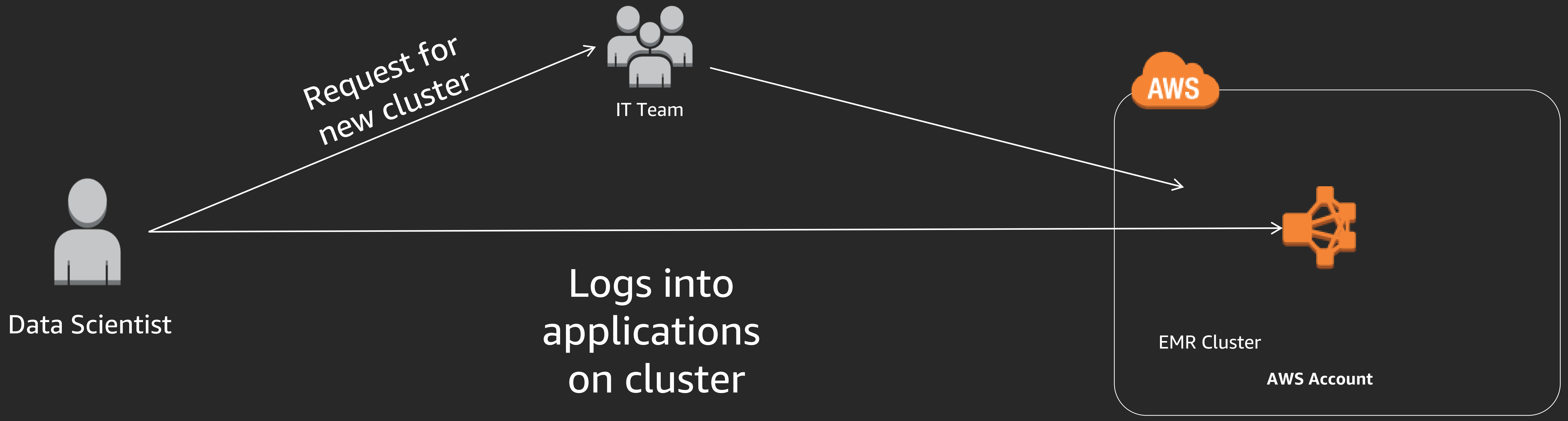
Administrator Interaction



End User Interaction

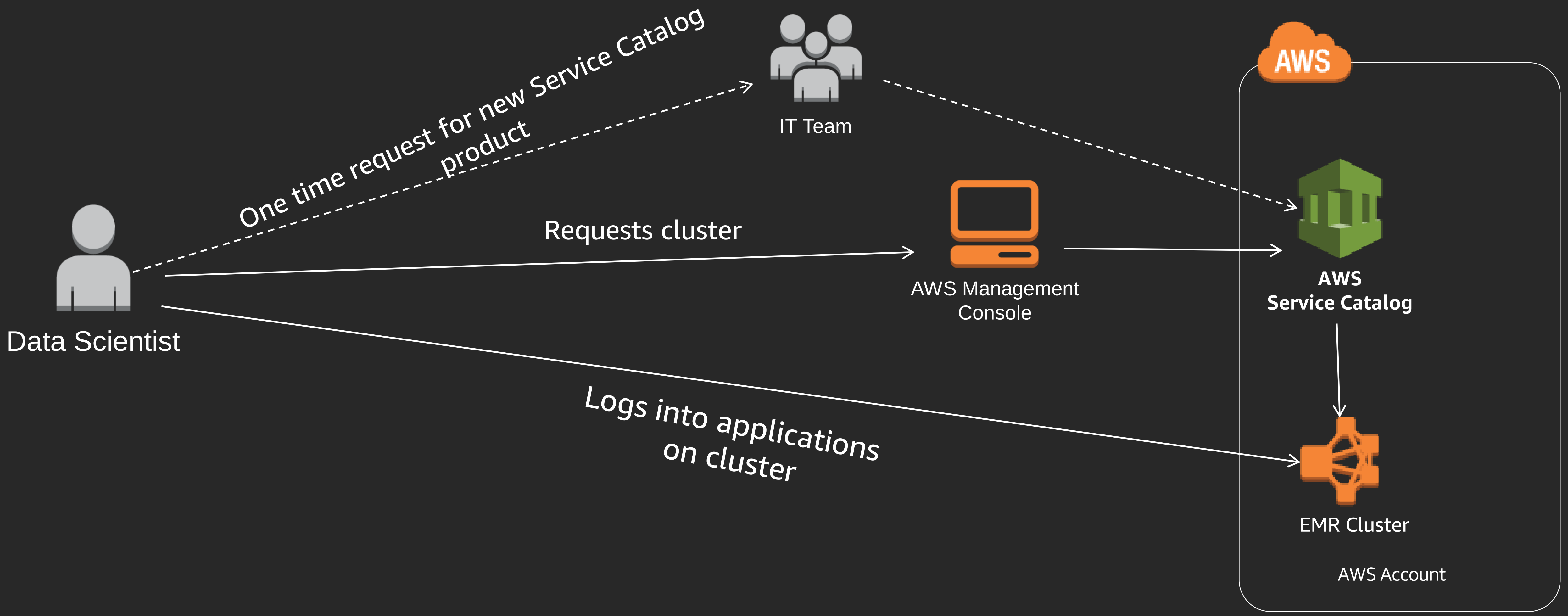

AWS
Browse Products
1
2
Portfolio
Select version, Provision Product, configure parameters
3
Deploy
AWS Service Catalog End-users
4
Notifications and outputs
AWS Administrato r
5
Scheduled functions
Notifications and outputs
4

Before: EMR Cluster Request Process



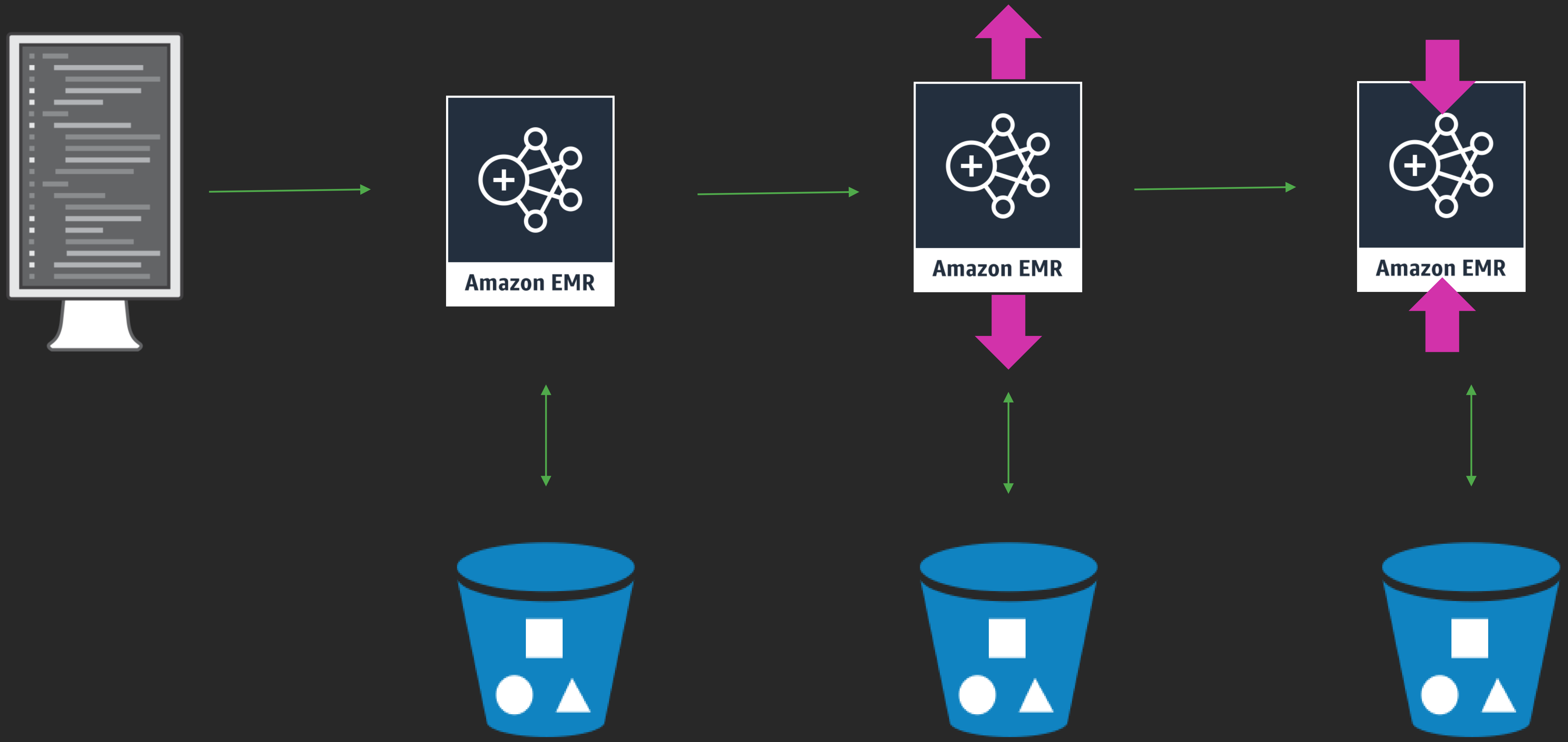
Logs into applications on cluster

EMR Self-Service with Service Catalog (SC)



Demo - Self-service Transient Clusters with AWS Service Catalog

Persistent Clusters



“Scale up and down”

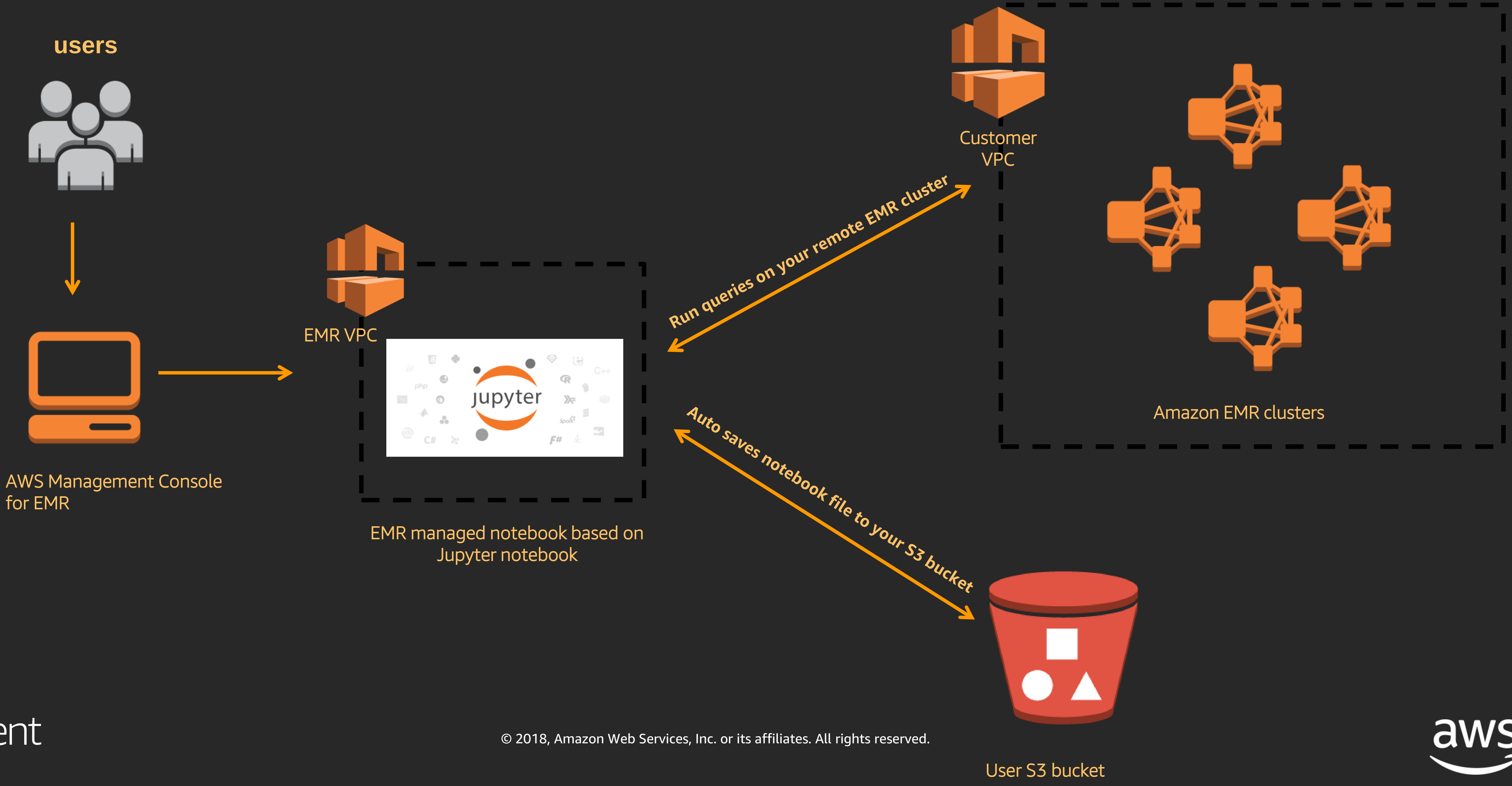
Introducing EMR Notebooks

New in 2018

- De-couple Notebooks from Clusters
- Based on Open Source Jupyter Notebooks
- Attach to a cluster to run jobs
- Multiple users can attach to the same cluster (set Autoscaling on a cluster)
- Detach and Attach to other clusters
- Save notebooks to Amazon S3
- Tag-based permissions

Notebooks

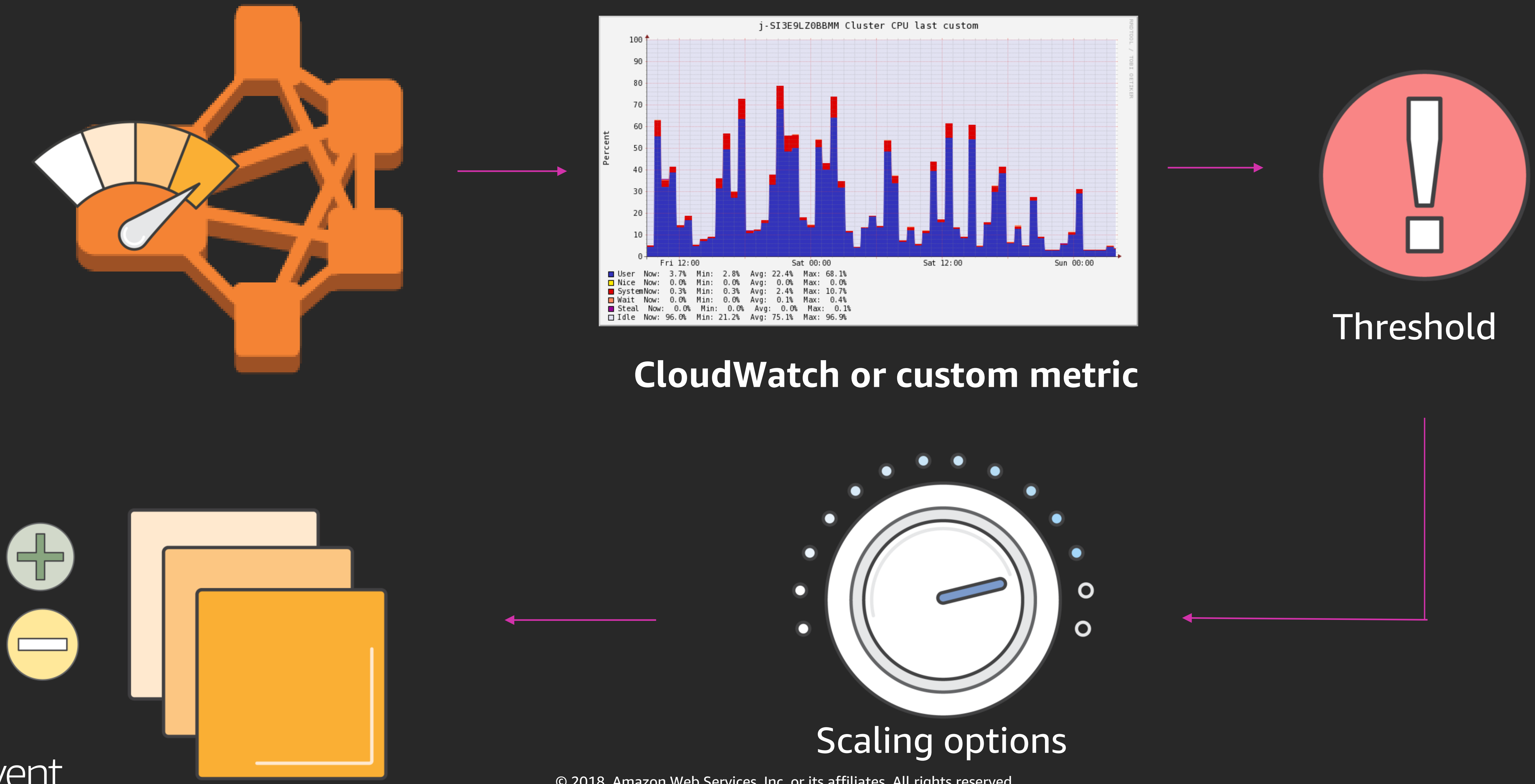
Off-cluster notebook based on Jupyter notebook application


users
Customer VPC
EMR VPC
jupyter
Run queries on your remote EMR cluster
Auto saves notebook file to your S3 bucket
Amazon EMR clusters
AWS Management Console for EMR
EMR managed notebook based on Jupyter notebook
User S3 bucket

AWS re:Invent

aws

Auto Scaling Clusters



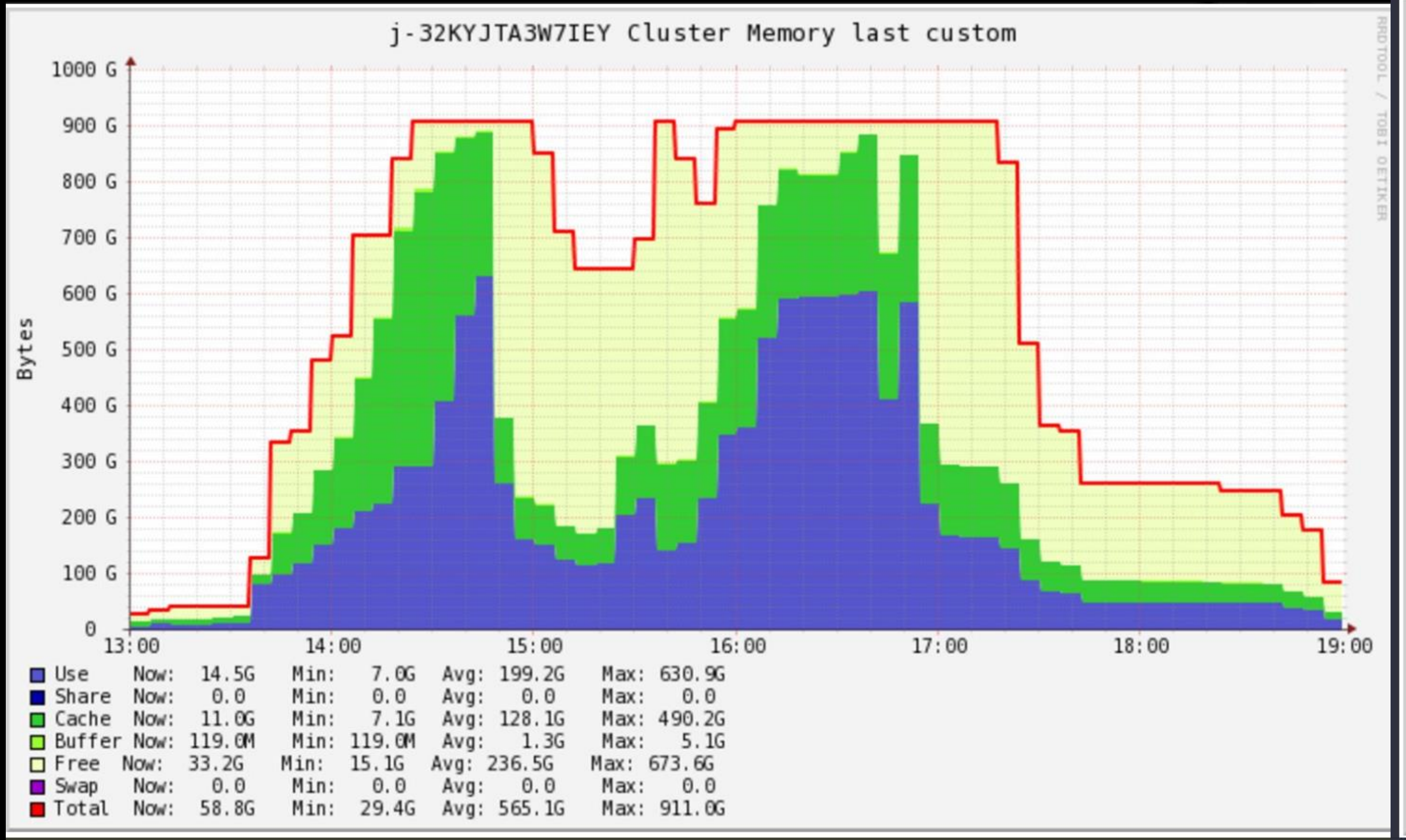
```
■ User   Now:  3.7%  Min:  2.8%  Avg: 22.4%  Max: 68.1%
■ Nice   Now:  0.0%  Min:  0.0%  Avg:  0.0%  Max:  0.0%
■ SystemNow:  0.3%  Min:  0.3%  Avg:  2.4%  Max: 10.7%
■ Wait   Now:  0.0%  Min:  0.0%  Avg:  0.1%  Max:  0.4%
■ Steal   Now:  0.0%  Min:  0.0%  Avg:  0.0%  Max:  0.1%
□ Idle   Now: 96.0%  Min: 21.2%  Avg: 75.1%  Max: 96.9%
```

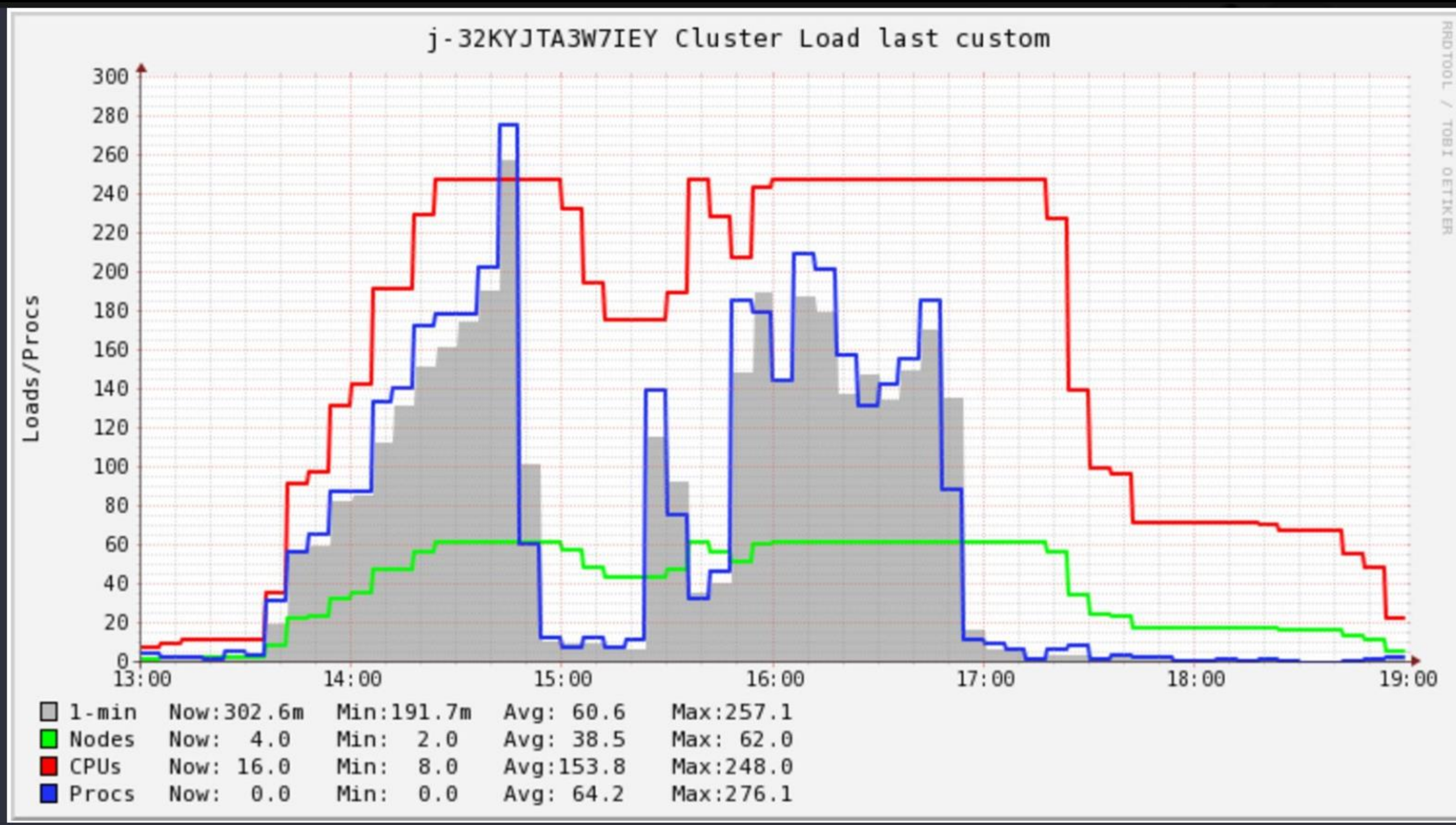
CloudWatch or custom metric

Threshold

Scaling options

Autoscaling Clusters





Source:@badcafe

Auto Scaling

- EMR scales-in at YARN task completion
- Selectively removes nodes with no running tasks
- yarn.resourcemanager.decommissioning.timeout
    - Default timeout is one hour
- Dynamically scale storage (HDFS and EBS)
- Spark scale-in contributions
    - Spark specific blacklisting of tasks
    - Unregistering cached data and shuffle blocks
    - Advanced error handling

Multi-Master support for EMR Applications

Coming Soon

Application	Multi-master behavior	HA details	Notes
YARN	HA	Active/Standby with automatic failover and recovery	Limited to YARN ResourceManager service
HDFS	HA	Active/Standby with automatic failover using quorum journaling	Limited to HDFS NameNode/metadata service
HBase	HA	Active/Standby utilizing zookeeper	
Zookeeper	HA	Ensemble with automatic quorum	
Ganglia	Available on all masters	HA agnostic	
Hive	HA (service components only)	Active/Standby	Limited to Metastore and HiveServer2; Requires external metastore database or catalog
Spark	Job specific	Supported by YARN and HDFS HA	Requires job designed for fault recovery
Flink	Job specific	YARN HA session supported by YARN, HDFS and Zookeeper HA	Requires job designed for fault recovery
Livy	Non-HA, single master only		State recovery supported
Oozie	Non-HA, single master only		
Hue	Non-HA, single master only		
Zeppelin	Non-HA, single master only		
JupyterHub	Non-HA, single master only		

Reconfiguration of EMR Applications

Coming Soon

Reconfigure applications on a running cluster

Reconfiguration applied to each instance group

Rolling restart of data nodes to prevent data loss

Automatic revert to last successfully applied version on failure

```
aws emr modify-instance-groups --instance-groups InstanceGroupId=ig-123,Configurations=file://new-configurations.json
```

Demo 2 – EMR Notebooks

Thank you!

AWS re:Invent

aws

!

Please complete the session survey in the mobile app.

AWS re:Invent

aws