Amazon EMR on Amazon EKS

Simplify Infrastructure Management with Amazon EMR on Amazon EKS

Damon Cortesi

Amazon EMR

Easily Run Spark, Hive, Presto, HBase, Flink, and more big data apps on AWS

Latest versions

Updated with latest open source frameworks within 30 days

Support for popular OSS like Flink, Hudi

Best Performance at Lowest cost

Spark workloads run 2.4x faster compared to Open Source

50–80% reduction in costs with EC2 Spot and Reserved Instances

Per-second billing for flexibility

Use S3 storage

Process data in S3 securely with high performance using the EMRFS connector

Scale Compute and Storage independent of each other

Easy & Scalable

Fully managed, no cluster setup, node provisioning or cluster tuning

Vertical and Horizontal Auto-Scaling to suit workload demands

aws

Customers are running Big Data / Analytics workloads on Kubernetes

Highly Scalable

Better resource utilization

Increasingly diverse

Accessed by many teams

Connected by many applications

Open source Community

aws

However, managing and maintaining open source workloads is challenging

- Container build and deployment
- Job submission
- Spark UI accessibility
- Performance optimization
- Autoscaling considerations
- Security and patch management

aws

Amazon EMR on Amazon EKS

NEW!

Run Apache Spark jobs on demand using **Amazon EMR** on **Amazon Elastic Kubernetes Service (EKS)** - without needing to provision EMR clusters - to improve resource utilization and simplify infrastructure management.

aws

How Amazon EMR on Amazon EKS solves the problem

DevOps Team

- Autoscaling considerations
- Patching and upgrading
- Integration with the rest of platform

EMR on EKS as a managed service, is integrated with AWS natively

Data Engineers / Data Scientists

- Performance optimization
- Cold start-up time
- Data isolation

EMR on EKS provides 2.4x better performance; K8s multi-tenancy design

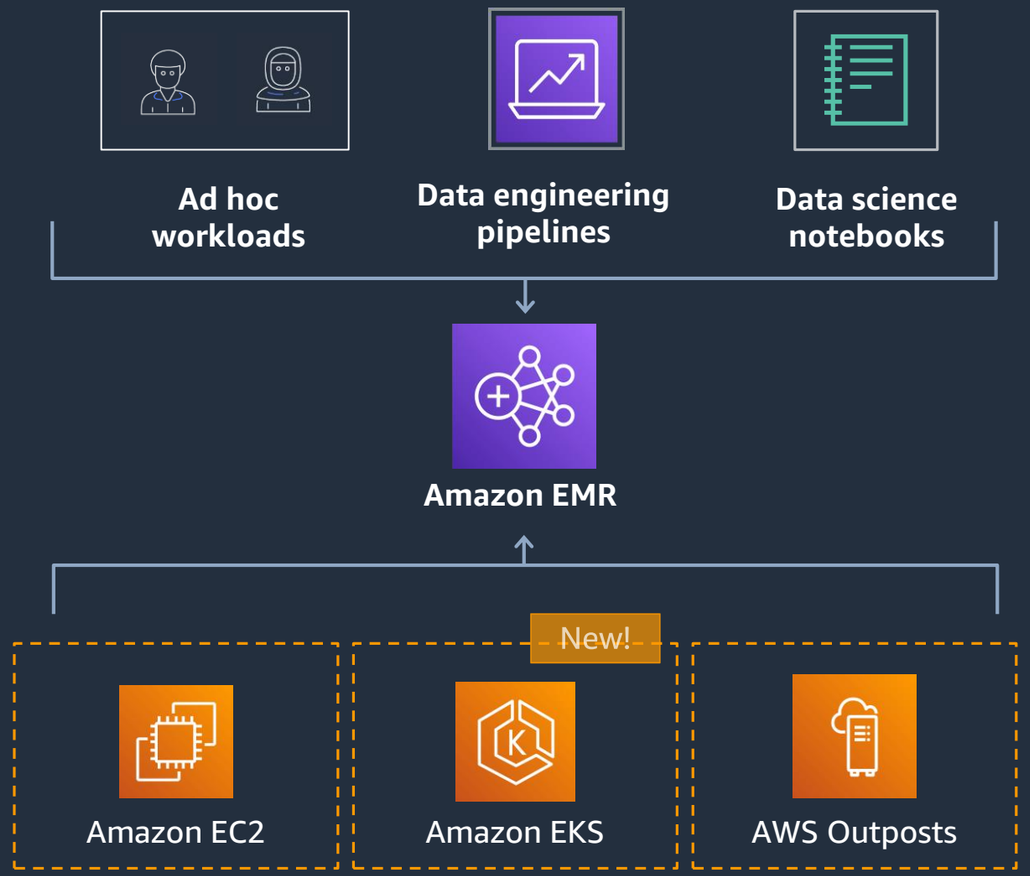
C-level

- Cost optimization
- Security
- Access control

EMR on EKS charges per job and achieves higher resource utilization with other micro-services

aws

A new deployment model for Amazon EMR



Run Amazon EMR on Amazon EKS

In addition to existing deployment modes

Simplifies running Spark on Kubernetes

aws

Amazon EMR on Amazon EC2

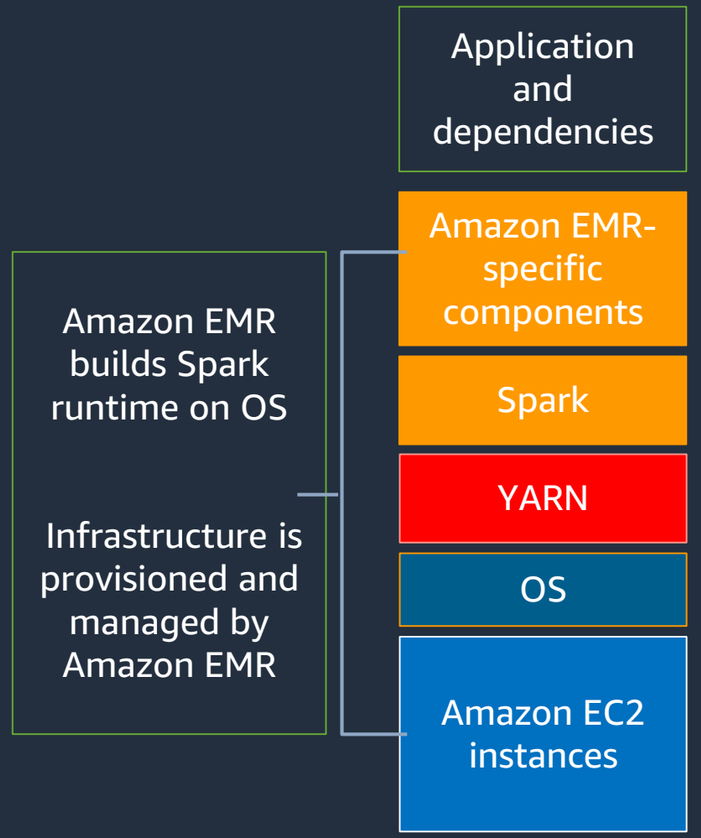
Control over instances drives cluster-centric model

Run single version of Spark per cluster

Shared execution role

Great for jobs with cluster-scoped dependencies

Great for clusters running at high utilization



aws

Amazon EMR on Amazon EKS

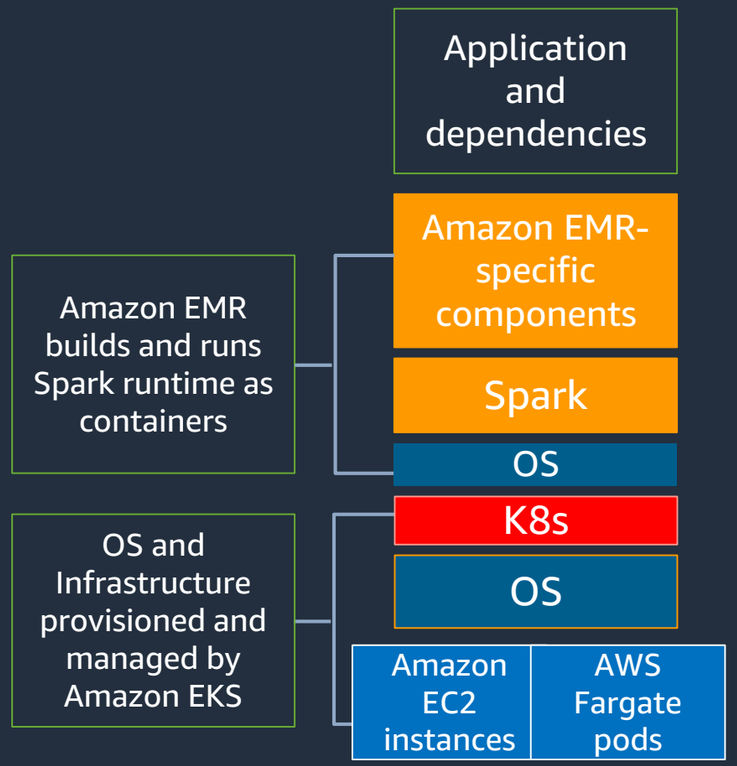
Containerization drives job-centric model

Run multiple versions of Spark per cluster

Per job execution role

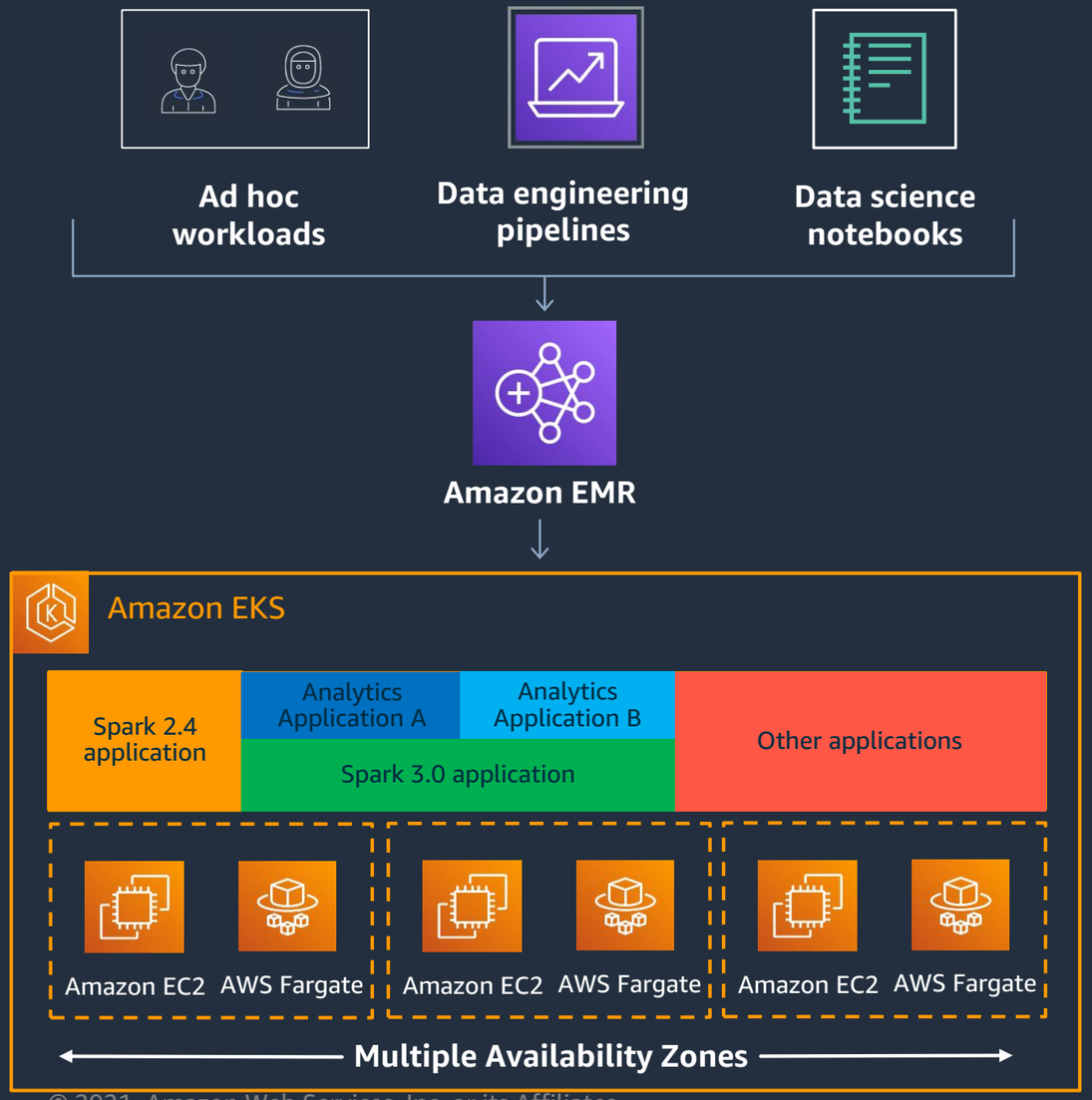
Great for quick test and upgrade cycles

Great for consolidating resource utilization



aws

Run Spark on Amazon EKS clusters



Register Amazon EKS clusters with Amazon EMR

Consolidate infrastructure across organization

Manage resource limits by teams and workload

Start jobs quickly, no cluster provisioning delays

Run application on single AZ or across multiple AZs

Choose serverless with AWS Fargate on Amazon EKS

Amazon EMR helps accelerate move to EKS

Provide managed distribution Spark on Kubernetes (2.4 and 3.1)

Manage job execution your behalf

Simplify secure execution using granular access control

Native integration with Amazon S3, AWS Glue Data Catalog, and more…

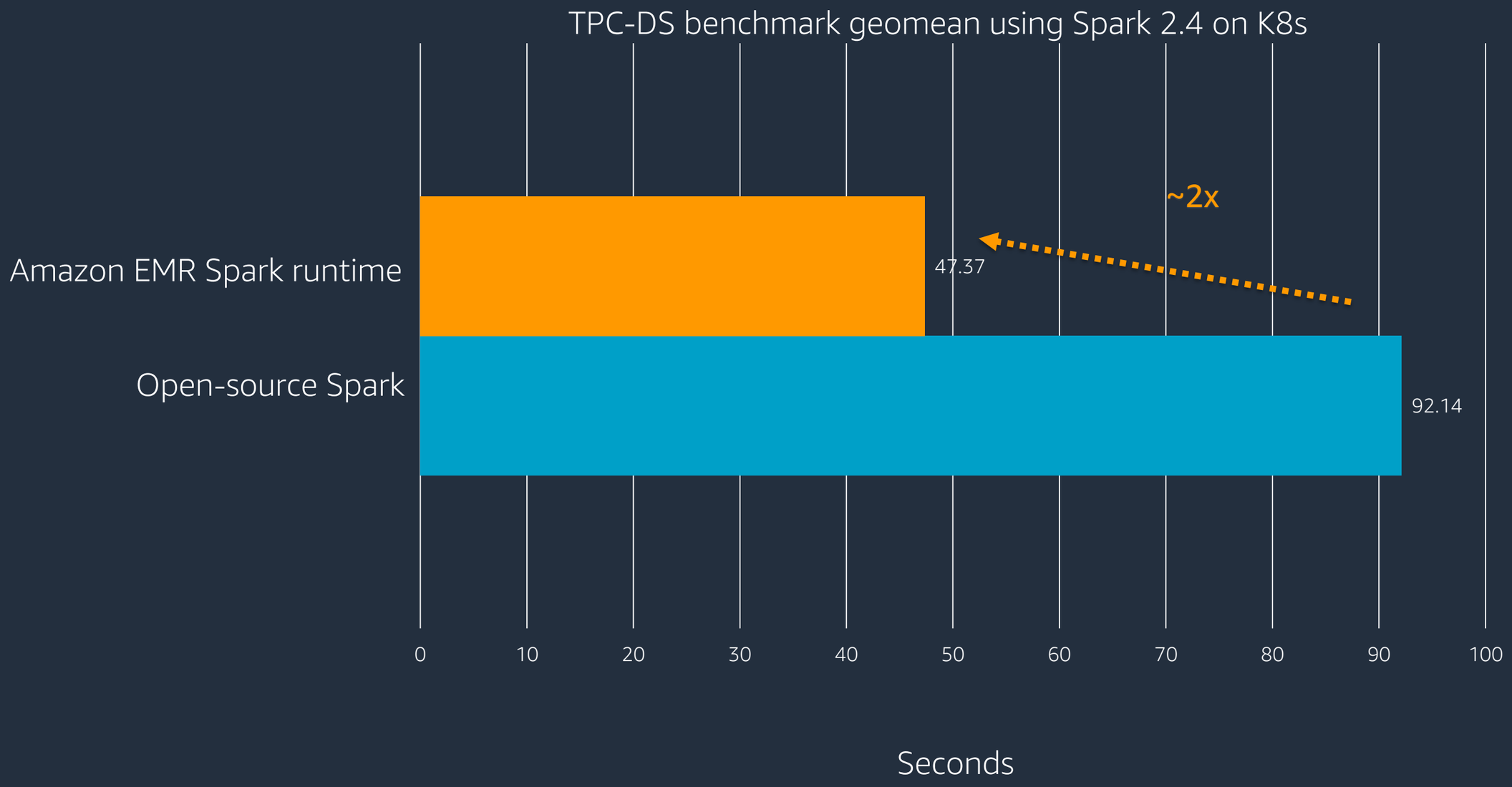
Simplify debugging with Spark History Server

Support integration with Apache Airflow

Differentiated performance with Amazon EMR runtime for Apache Spark

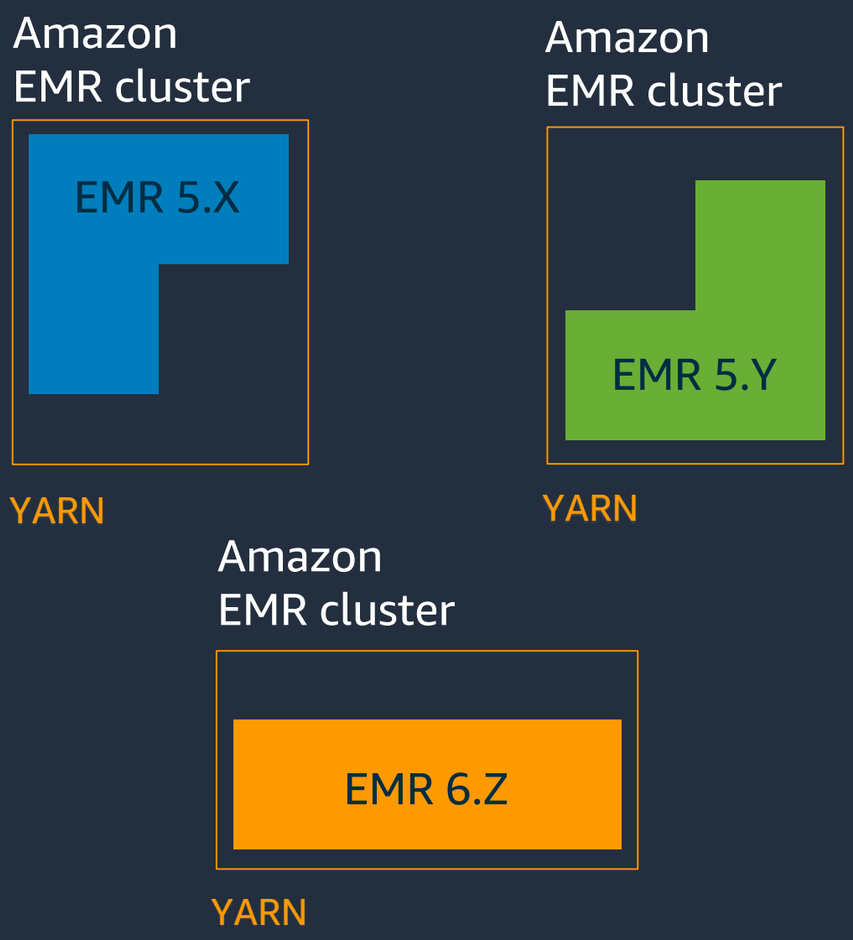
aws

Spark performance with Amazon EMR runtime

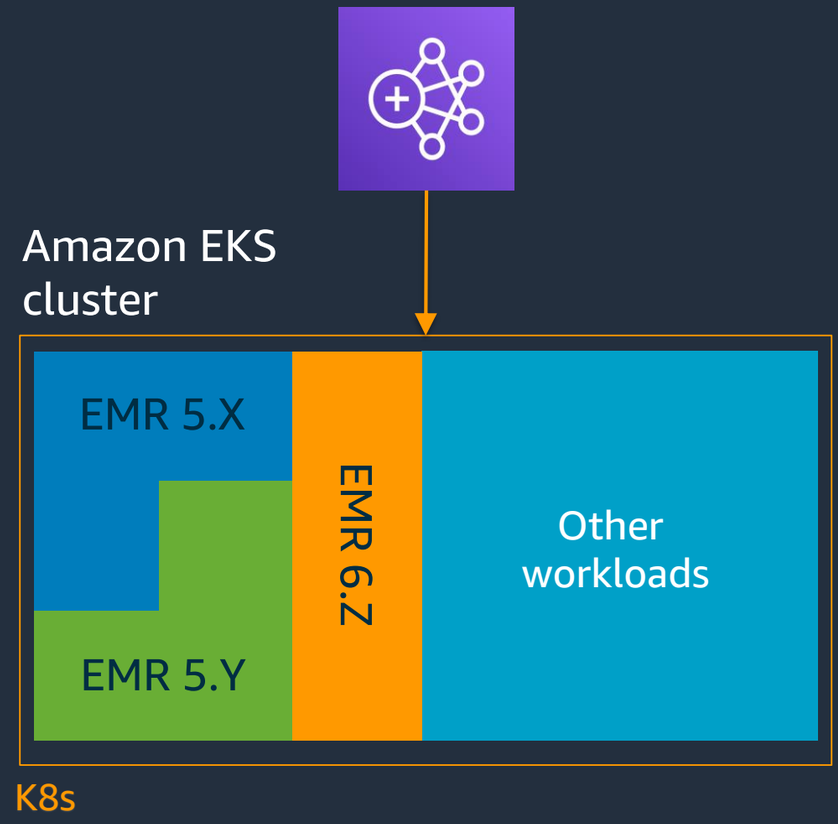


Consolidation saves costs

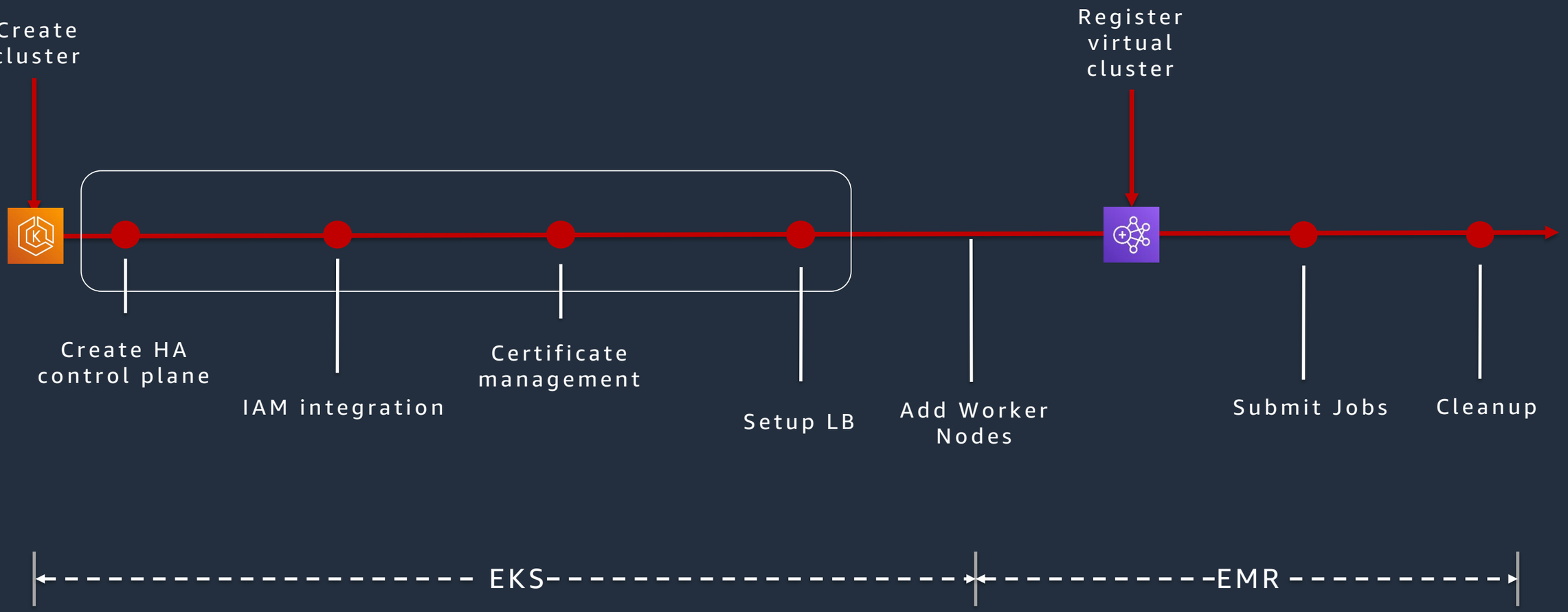
Amazon EMR on Amazon EC2



Amazon EMR on Amazon EKS



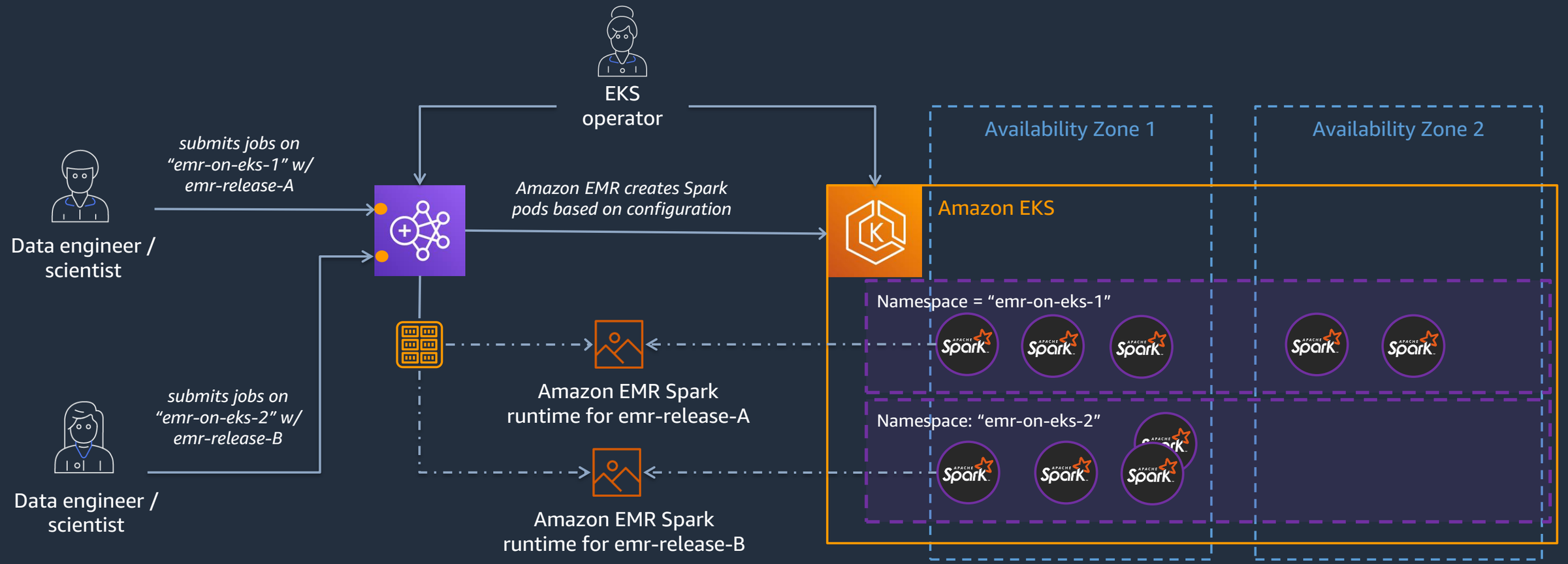
How to get started



Example Architecture

aws

Job-centric workflow

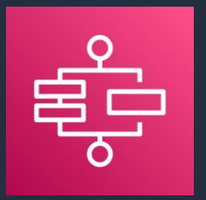


Job submission options

AWS Command Line Interface

AWS Tools and AWS SDK

Apache Airflow



AWS Step Functions

aws

Job debugging options

Amazon CloudWatch

Amazon Simple Storage Service (Amazon S3)

AWS Management Console

Amazon EMR Studio

aws

Job customization options

- Pod Templates
    - Node selectors – Run on specific node groups (Spot, Fargate)
    - Sidecar containers – Custom code alongside Spark pod

aws

Run a custom sidecar

```yaml
apiVersion: v1
kind: Pod
spec:
  containers:
    - image: busybox
      command: ["/bin/sh"]
      args:
        [
          "-c",
          "while true; do echo echo $(date -u) 'Hi I am from Sidecar container'; sleep 5;done",
        ]
      name: sidecar-container
      resources: {}
      volumeMounts:
        - name: emr-container-application-log-dir
          mountPath: /var/log/spark/user
```


aws

Run on Spot

```yaml
# spot_pod_template.yaml
apiVersion: v1
kind: Pod
spec:
  nodeSelector:
    eks.amazonaws.com/capacityType: SPOT
```

```
aws emr-containers start-job-run \
    --virtual-cluster-id ${EMR_EKS_CLUSTER_ID} \
    --name dacort-windycity \
    --execution-role-arn ${EMR_EKS_EXECUTION_ARN} \
    --release-label emr-5.33.0-latest \
    --job-driver '{
        "sparkSubmitJobDriver": {
            "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
            "sparkSubmitParameters": "--conf spark.executor.instances=5 --conf spark.executor.memory=2G
--conf spark.executor.cores=2 --conf spark.driver.cores=1 --conf
spark.kubernetes.executor.podTemplateFile=s3://'${S3_BUCKET}'/spot_pod_template.yaml"
        }
    }'
```


aws

Resources

- EMR On EKS Workshop: https://emr-on-eks.workshop.aws/
- EMR Containers Best Practices: https://aws.github.io/aws-emr-containers-best-practices/

aws

Thank you!