

# Amazon EMR Immersion Day



# Agenda

7:00 AM – 7:15 AM. Welcome and Introductions

7:15 AM - 8:00 AM. Introduction to Amazon EMR

8:00 AM – 8:15 AM. *Break / Q &A*

8:15 AM – 8:45 AM. EMR Cluster Creation Overview

**8:45 AM – 9:15 AM. Lab – EMR Cluster Creation**

9:15 AM – 9:30 AM. *Break / Q &A*

**9:30 AM – 10:00 AM. Lab – Hive, Pig, & EMR**

10:00 PM – 11:00 AM. *Lunch Break*

11:00 AM – 11:45 AM. Amazon EMR Well-Architected and Best Practices

11:45 AM – 12:00 PM. *Break / Q &A*

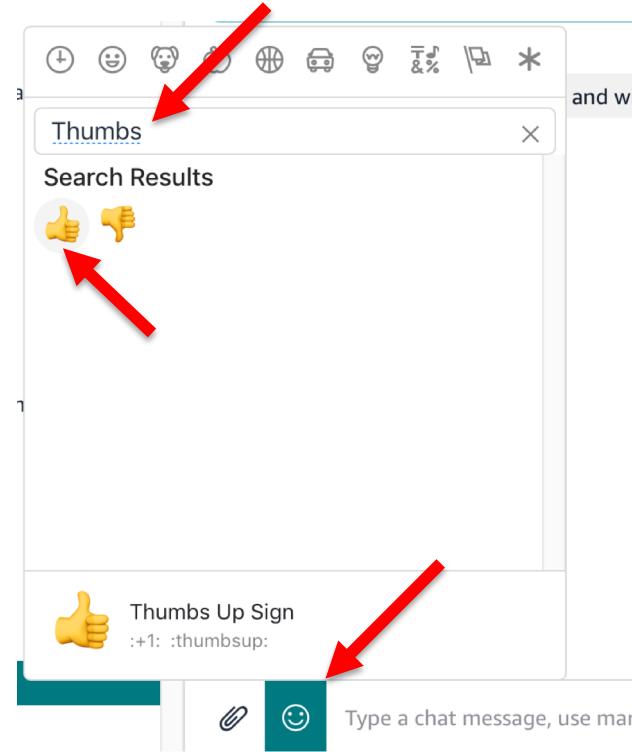
**12:00 PM – 12:45 PM. Lab – Spark-based ETL and Notebooks on EMR**

12:45 PM – 1:00 PM. Close Out and Next Steps

To participate in the labs,  
please send an email to  
**dbayard@amazon.com**

# Level-set: Raise your hand if know these acronyms/terms?

- HDFS?
- Hive?
- Spark?
- Jupyter?
- EC2?
- S3?
- EMR?



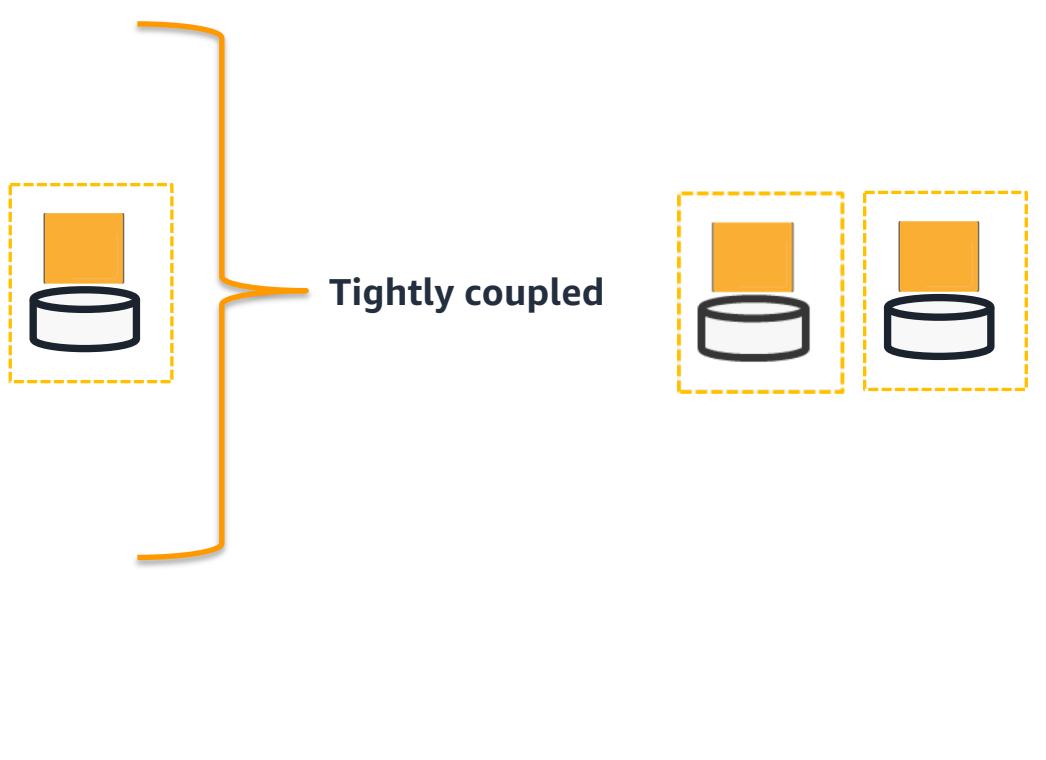
# The Need for Change. Challenges of on-premises clusters.

```
userDetailsCardInHover = showOnHover(userDetailsCard);
userLink = {
  ...
  secondaryLink,
  href,
  userAvatar,
  name,
  ...
  className(styles.container),
  includeAvatar: true,
  userDetailsCardInHover,
  user: user,
  delay: CARD_HOVER_DELAY
}

userDetailsCardInHover user={user} delay={CARD_HOVER_DELAY}-
  <Link to={pathUser.buildUserUrl(user)}>
    <ClassNames>{classNames(styles.name, {
      [styles.alt]: type === 'ALT',
      [styles.centerName]: !secondaryLink,
      [styles.inlineLink]: inline,
    })}
    ...
    {children || user.name}
  </Link>
  ...
  {secondaryLink
    ? null
    : ...
      href={secondaryLink.href}
      className={classNames(styles.name, {
        [styles.alt]: type === 'ALT',
        [styles.secondaryLink]: secondaryLink,
      })}
      ...
      {secondaryLink.label}
    </a>
  }
</div>
</div>
</div>
```

```
145      </div>
146    </div>
147  </div>
148  </div>
149  );
150}
151
152  renderBatchedLinks() {
153    return [
154      <div className={style.container}>
155        <h4 className={style.title}>
156          <a className={style.link}>
157            {this.renderBatchedLinksHeader}
158          </a>
159        </h4>
160        {this.renderBatchedLinksList}
161      </div>
162    ];
163  }
164
165  renderBatchedLinksHeader() {
166    <div>
167      <h4>
168        {this.renderBatchedLinksTitle}
169      </h4>
170    </div>
171  }
172
173  renderBatchedLinksList() {
174    return [
175      <ul style={{listStyleType: 'none'}}>
176        {this.renderBatchedLinksListItems}
177      </ul>
178    ];
179  }
180
181  renderBatchedLinksListItems() {
182    return [
183      ...this.state.links.map(link => {
184        const title = link.title || link.name;
185        const href = link.href || link.url;
186        const icon = link.icon || trackIcon;
187        const logo = link.logo || userIcon;
188        const slogan = link.slogan || '';
189
190        return [
191          <li key={link.id}>
192            <div style={{display: 'flex', gap: 10}}>
193              {icon}
194              <div>
195                {title}
196                <br/>
197                {link.description}
198              </div>
199            </div>
200          </li>
201        ];
202      })
203    ];
204  }
205
206  renderFooterSub() {
207    return [
208      <div className={styles.footerSub}>
209        <Link to="#" title="Home - Unsplash">
210          <Icon type="logo" className={styles.footerSubLogo}>
211            ...
212          </Icon>
213          <span className={styles.footerSubSlogan}>
214            {slogan}
215          </span>
216        </Link>
217      </div>
218    ];
219  }
220
221  render() {
222    return [
223      <div>
224        <div>
225          <div>
226            <div>
227              {this.renderFooterMain()}
228            </div>
229            {this.renderFooterSub()}
230          </div>
231        </div>
232      </div>
233    ];
234  }
235}
```

# Compute and storage grow together



- Storage grows along with compute
- Compute requirements vary

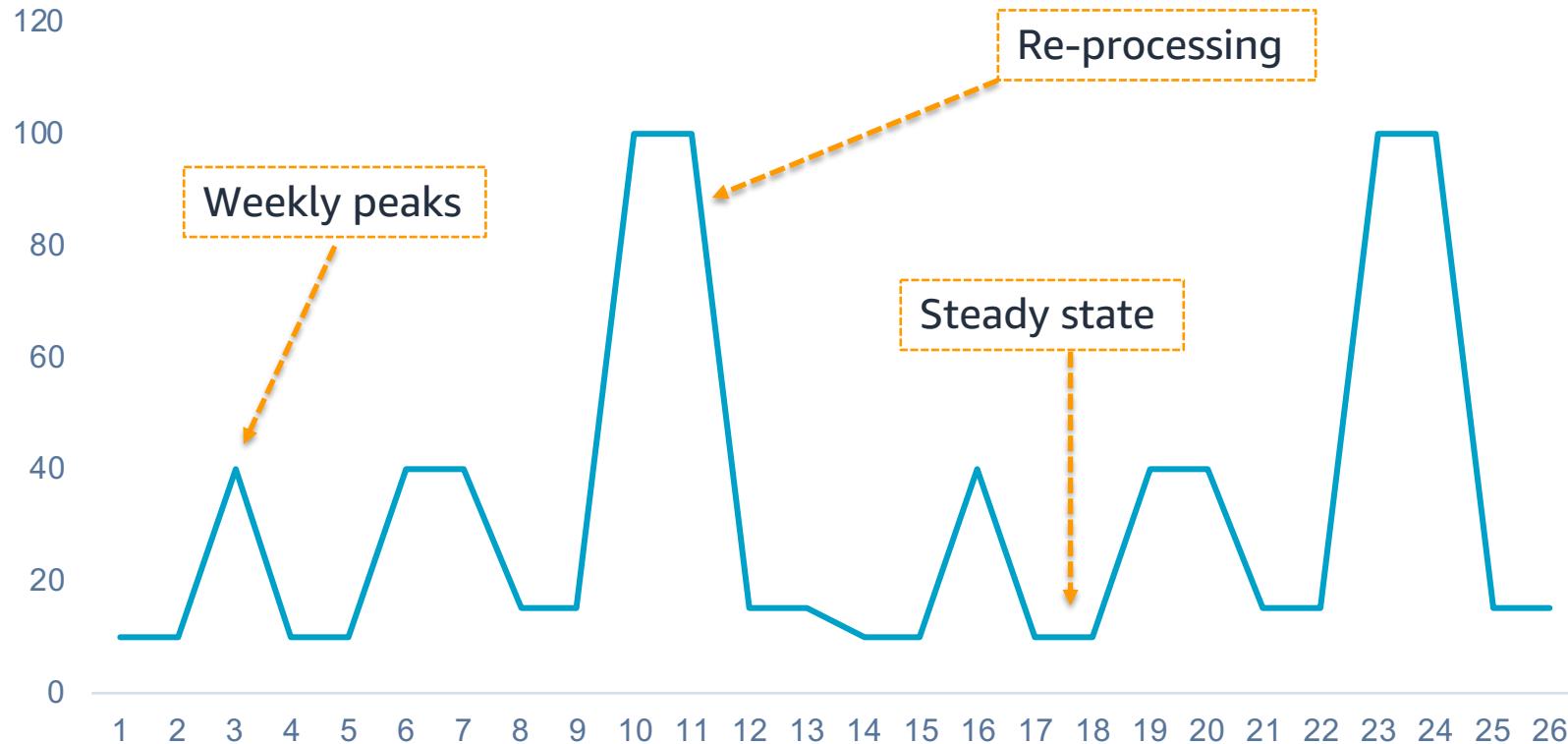
# Replication adds to cost



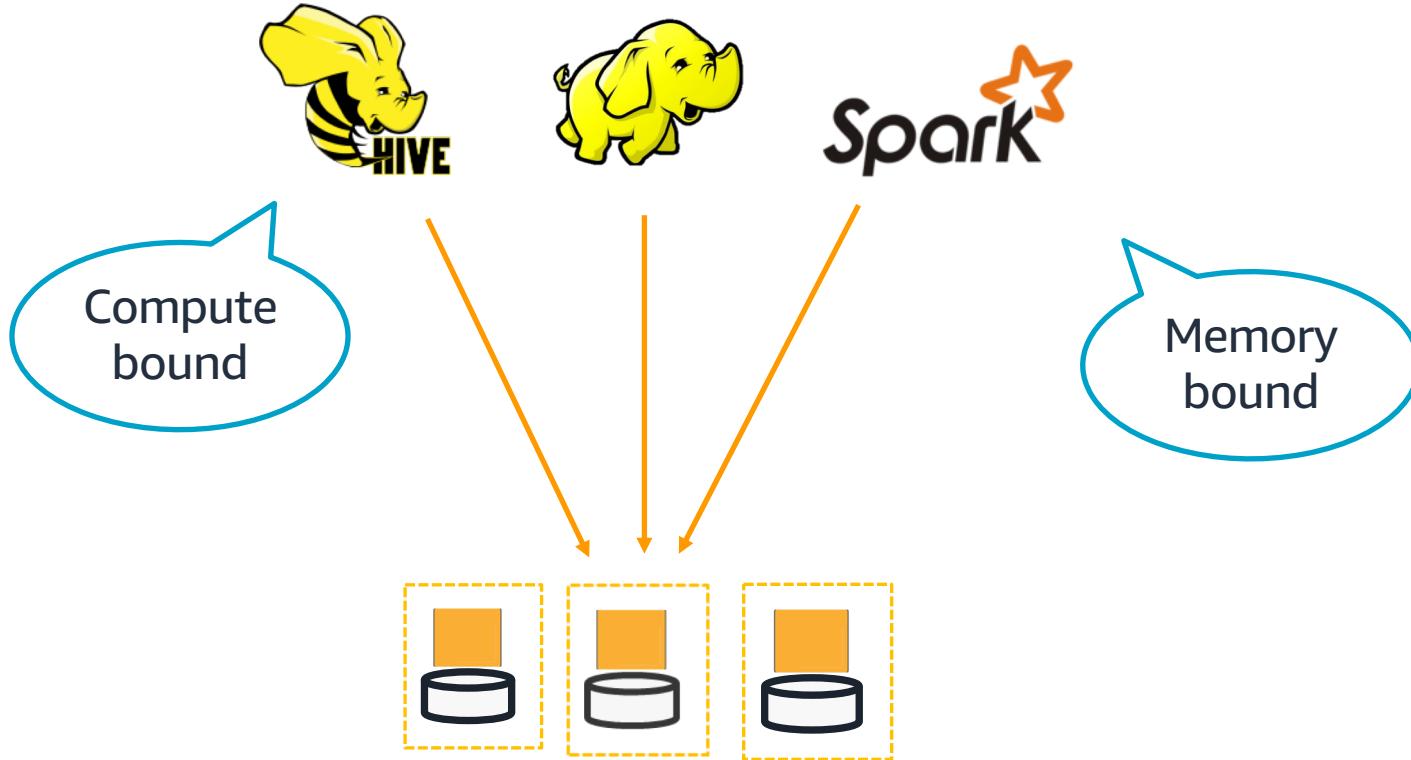
3x

- Data is replicated several times
- Typically only in one data center

# Underutilized or Scarce resources



# Contention for the same resources



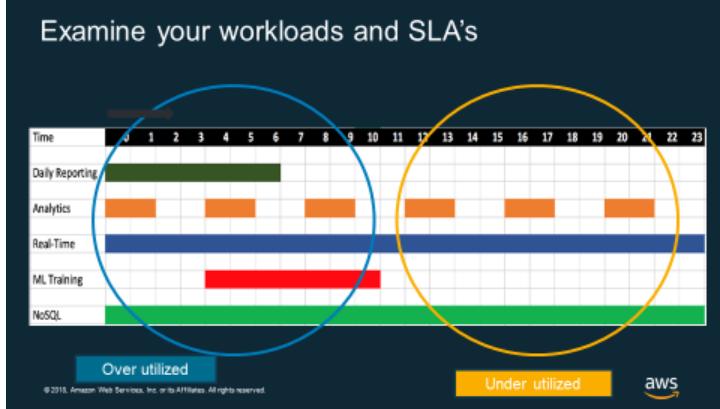
# Limited on Fast Following Application Versions

With a monolithic cluster, there may be dependencies of downstream applications that impact the inability to upgrade versions. By not upgrading, organizations could be limiting innovation.

- **Large Scale Transformation:** Map/Reduce, Hive, Pig, Spark
- **Interactive Queries:** Impala, Spark SQL, Presto
- **Machine Learning:** Spark ML, MxNet, Tensorflow
- **Interactive Notebooks:** Jupyter, Zeppelin
- **NoSQL:** HBase

# Summary of Challenges

1. **Fixed Cost:** Forced into entering into a multi-year commitment for HW, SW and associated Support Fees driving a sizable capital outlay.
2. **Storage / Compute:** These are tightly coupled, so even if you just want to increase storage for a data retention use case, you incur cost for unnecessary compute.
3. **Always On:** Cluster was always on, which is not resource and cost efficient. Consider the cost of power, FTE's and other data center demands.
4. **Self-Service:** Users want options to deploy only certain services such as Hive, Spark, Presto only; oftentimes just want to test a new service.
5. **Static, Not-scalable & Unused Capacity:** Always fixed number of nodes used; unable to scale based on input size of data (small data sets vs. larger data sets) and lacked consideration for cluster contention.
6. **Outages Impact:** During Production outages, no alternative means to access the cluster. Must consider the implication of downtimes.
7. **Production Upgrade:** Upgrading individual services (i.e. Hive) is laborious to run through DEV, TEST, and PROD cluster environments.
8. **Slow Deployment Cycle:** The cumulative effect of above impacts speed of readying the right infrastructure to solve business and technical use cases.



# Introducing Amazon EMR

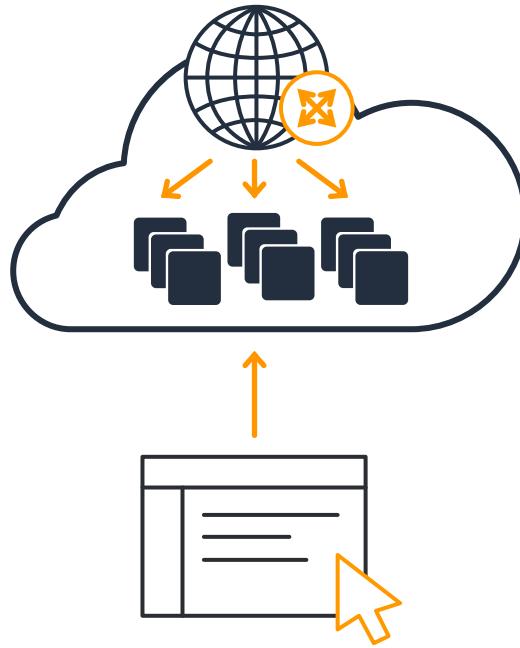
## Managed Hadoop and Spark in the cloud



Enterprise-grade

Easy

Lowest cost



- Launch Hadoop & Spark clusters in minutes
- No need to install or maintain Hadoop
- Cluster tuning, and configuration done for you
- Automatically scale clusters based on policies
- Optimized for both transient and long-running clusters

# Amazon EMR



## Application Index

- Amazon EMR is one of the largest Spark and Hadoop service providers in the world, enabling customers to run ETL, machine learning, real-time processing, data science, and low-latency SQL at petabyte scale.
- See the [EMR Release](#) page for up to date schedules. Users can bootstrap applications not listed below.



### Administrative

- Ganglia Cluster monitoring
- Livy REST API interacting with Spark
- Oozie Workflow Scheduler
- ZooKeeper Configure & Sync node

### Machine Learning

- Mahout Machine Learning
- MXNet Deep Learning
- SparkML Machine Learning
- TensorFlow Deep Learning

### Data Movement

- Sqoop Relational DB connector

### NoSQL

- HBase Hadoop's non-relational DB

### Data Processing

- Flink Stream Processing
- Hive Hadoop Data warehouse
- MapReduce Batch Data Processing
- Presto Distributed SQL for Big Data
- Spark In-memory Data Processing & Machine Learning
- Tez Interactive Data Processing
- Hudi Update Data Lake Storage

### Query Tools

- EMR Notebooks Serverless notebook
- Hue Visualization and Querying for Hadoop
- JupyterHub Multi-user Jupyter Notebook (installed on cluster)
- Phoenix Querying for HBase
- Pig Scripting language for MapReduce jobs
- Zeppelin Data Science notebook

# Enterprise-grade Hadoop & Spark

Deploy latest releases in Hadoop and Spark ecosystems



5.23.0 April 2019	5.24.0 & 1 June 2019	5.25.0 July 2019	5.26.0 Aug 2019	5.27.0 Sep. 2019	5.28.0 Nov. 2019	5.28.1 Jan. 2020	5.29.0 Jan. 2020
2.8.5	2.8.5	2.8.5	2.8.5	2.8.5	2.8.5	2.8.5	2.8.5
1.7.1	1.8.0	1.8.0	1.8.0	1.8.1	1.9.0	1.9.1	1.9.1
37.2	37.2	37.2	37.2	37.2	37.2	37.2	37.2
14.9	14.9	14.9	14.10	14.10	14.10	14.10	14.10
2.3.4	2.3.4	2.3.5	2.3.5	2.3.6	2.3.6	2.3.6	2.3.6
4.3.0	4.4.0	4.4.0	4.4.0	4.4.0	4.4.0	4.4.0	4.4.0
0.9.4	0.9.6	0.9.6	0.9.6	1.0.0	1.0.0	1.0.0	1.0.0
0.5.0	0.6.0	0.6.0	0.6.0	0.6.0	0.6.0	0.6.0	0.6.0
0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0	0.13.0
1.3.1	1.4.0	1.4.0	1.4.0	1.4.0	1.5.1	1.5.1	1.5.1
51.0	51.0	51.0	51.0	51.0	51.0	51.0	51.0
4.14.1	4.14.1	4.14.2	4.14.2	4.14.3	4.14.3	4.14.3	4.14.3
0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0	0.17.0
0.215	0.219	0.220	0.220	0.224	0.227	0.227	0.227
24.0	24.2	24.3	24.3	24.4	24.4	24.4	24.4
1.4.7	1.4.7	1.4.7	1.4.7	1.4.7	1.4.7	1.4.7	1.4.7
1.12.0	1.12.0	1.13.1	1.13.1	1.14.0	1.14.0	1.14.0	1.14.0
0.9.1	0.9.1	0.9.2	0.9.2	0.9.2	0.9.2	0.9.2	0.9.2
0.8.1	0.8.1	0.8.1	0.8.1	0.8.1	0.8.2	0.8.2	0.8.2
3.4.13	3.4.13	3.4.14	3.4.14	3.4.14	3.4.14	3.4.14	3.4.14
1.11.519	1.11.546	1.11.566	1.11.595	1.11.615	1.11.659	1.11.659	1.11.682
5.23.0 April 2019	5.24.0 June 2019	5.25.0 July 2019	5.26.0 Aug 2019	5.27.0 Sep. 2019	5.28.0 Nov. 2019	5.28.1 Jan. 2020	5.29.0 Jan. 2020

- 19 open-source projects: Apache Hadoop, Spark, HBase, Presto, and more
- Updated with the latest open source frameworks within 30 days of release





- 19 open-source projects: Apache Hadoop, Spark, HBase, Presto, and more
- Updated with the latest open source frameworks within 30 days of release

# High impact results with Amazon EMR



near real-time analytics for 140M players



scales 3,000 transient clusters on a daily basis



GE POWER

powers the Predix solution processing 1,000,000 data executions/day



achieves costs savings of 55% when compared to on-demand pricing and 40% savings when compared to Reserved Instances



computes Zestimates on 100M +homes in hours instead of 1 day



# On-premises migrations to Amazon EMR



Processes 135B events/day and have cost savings of 60% (~\$20M)



decreased costs by \$600k in less than 5 months



reduced cost of operation and improved Spark performance 3x

Aol.

saves 75% and is 60% more efficient



re-architects 1 monolithic pipeline into 3 purpose built clusters



# The value of cloud-based Hadoop/Spark with Amazon EMR.

```
userDetailsCardOnHover = showOnHover(userDetailsCard);

userLink = {
  ...
  secondaryLink,
  dran,
  userAvatar,
  ...
}

in className(styles.container)

includeAvatar @6 {
  <userDetailsCardOnHover
    user={user}
    delay={CARD_HOVER_DELAY}
    wrapperClassName={styles.avatarContainer}
  >
}

classNames(classNames)
style={style}
initialStyle={initialStyle}
)
}

userDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}
  <a href={pathway.buildUserSet(user)}>
    <div className={classNames(styles.name, {
      [styles.alt]: type === 'ALT',
      [styles.secondaryLink]: secondaryLink,
      [styles.inlineLink]: inline,
    })}>
      {children || user.name}
    </div>
    {secondaryLINK ? null : 
      <a href={secondaryLink.href}>
        <div className={classNames(styles.name, {
          [styles.alt]: type === 'ALT',
          [styles.secondaryLink]: secondaryLink,
        })}>
          {secondaryLink.label}
        </div>
      </a>
    }
  </userDetailsCardOnHover>
</div>
</div>
```

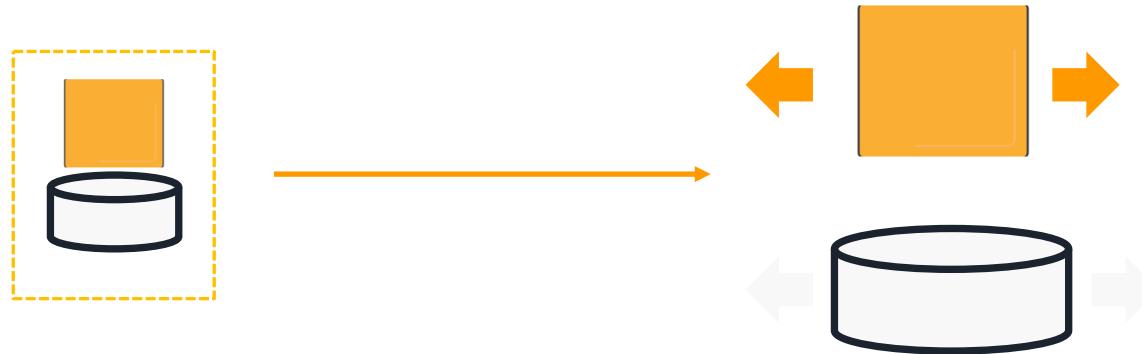
```
145   </div>
146   </div>
147   </div>
148   </div>
149   );
150 }
151
152 renderBatchedLinks() {
153   return [
154     <div className={style}>
155       <h4 className={style}>
156         <a className={style}>
157           {this.renderBatchedLinks}
158         </a>
159       </h4>
160       {this.renderBatchedLinks}
161     </div>
162   ];
163 }
164
165 </div>
166 </div>
167 </div>
168 </div>
169 </div>
170 </div>
171 <li className={styles.footer}>
172   {this.renderBatchedLinks}
173 </li>
174 </ul>
175 <div title={title}>
176   <img alt={image}>
177 </div>
178 <div>
179   <div>
180     <img alt={image}>
181   </div>
182 </div>
183 renderFooterSub() {
184   return [
185     <div className={styles.footerSub}>
186       <a href="#" title="Home - Unagi">
187         <Icon type="logo" className={styles.footerSubLogo}>
188           />
189         </Icon>
190         <span className={styles.footerSubSlogan}>
191           Unagi
192         </span>
193       </a>
194     </div>
195   ];
196 }
197
198 render() {
199   return [
200     <footer className={styles.footerGlobal}>
201       <div className="container">
202         {this.renderFooterMain()}
203         {this.renderFooterSub()}
204       </div>
205     </footer>
206   ];
207 }
```

# Benefits of EMR



1. Decoupled compute & storage
2. Built-in disaster recovery
3. Turn off your clusters through transient clusters
4. Agility of Auto-scaling persistent clusters
5. Leverage EC2 Spot pricing
6. Self-service with AWS Service Catalog
7. Spark performance improvements
8. Fully-managed EMR Notebooks
9. Lowest TCO in the industry, analysts confirm
10. EMR is surrounded by the industry's broadest Analytics ecosystem

# Benefit 1: Decouple storage and compute



# Benefit 1: Amazon S3 is your persistent data store



Amazon S3

- 99.999999% durability
- Low cost and many varieties
- Life Cycle Policies
- Versioning
- Distributed by default
- EMR FS

# Benefits Summary

## 10 Benefits of migrating to Amazon EMR

1. Decoupled Compute & Storage
2. Built-in Disaster Recovery
3. Turn off your clusters through Transient clusters
4. Agility of Auto-scaling Persistent clusters
5. Leverage Spot pricing for unused EC2 capacity
6. Self-service with AWS Service Catalog
7. Spark Performance Improvements
8. Fully Managed EMR Notebooks
9. Lowest TCO in the Industry, analysts confirm
10. EMR is surrounded by the industry's broadest Analytics ecosystem

Nov. 2018, IDC report confirms:

"EMR provides 57% reduced costs vs. on-premise resulting in 342% ROI over 5 years."



Dec. 2018, Gartner suggests:

"AWS remains the largest Hadoop provider in terms of both revenue and user base."

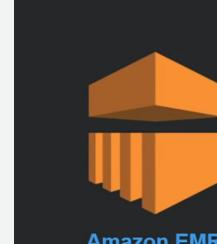


Feb. 2019, Forrester recognizes:

AWS EMR as the Cloud Hadoop/Spark (HARK) Leader



## EMR Design Options

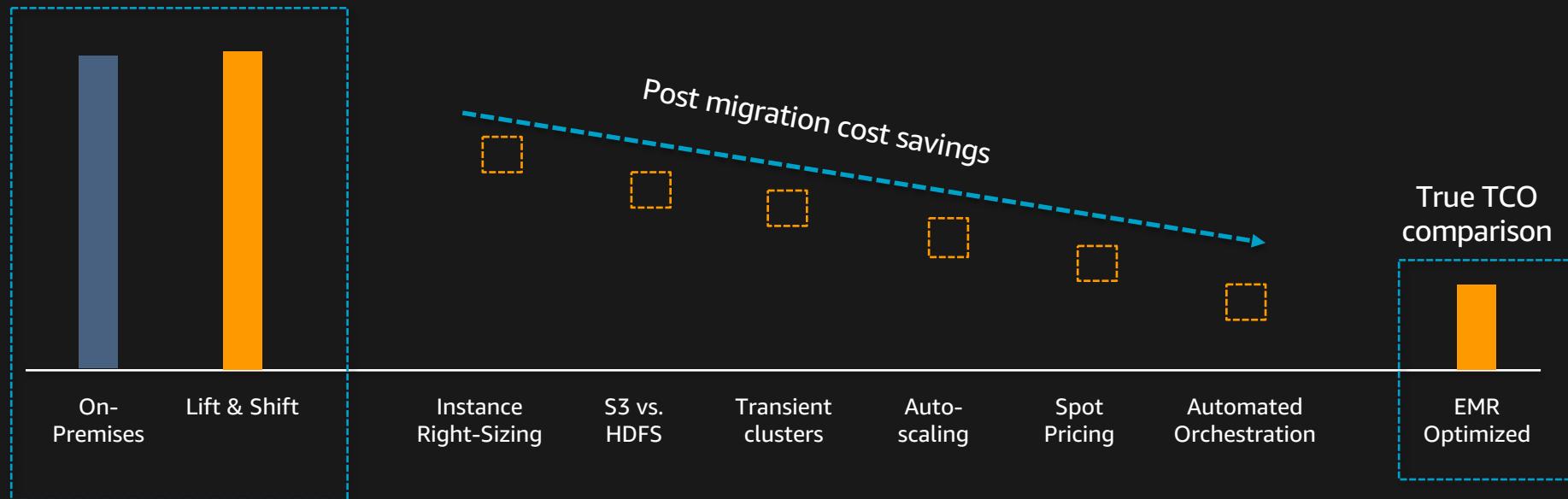


Transient vs. Persistent Cluster  
Amazon S3 vs. local HDFS  
Elastic Cluster vs. Static Cluster  
On-Demand vs. Reserved vs. Spot  
Core Nodes vs. Task Nodes

# A multi-phased approach to best practice deployment

Go beyond a lift & shift to optimize for scale and cost.

Typical TCO comparison



# Resources

Service page  
<https://aws.amazon.com/emr>

Getting started  
<https://aws.amazon.com/emr/getting-started/>

EMR Migration Program  
<https://aws.amazon.com/emr/emr-migration>

AWS Big Data Blog  
<https://aws.amazon.com/blogs/big-data/>



# EMR Cluster Creation



# Creation Options



AWS  
Management  
Console



AWS CLI



AWS  
SDKs

**Step 1: Software and Steps**

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

**Software Configuration**Release   

- |  |   |  |
|--|---|--|
| <input checked="" type="checkbox"/> Hadoop 2.8.5 | <input type="checkbox"/> Zeppelin 0.8.1 | <input type="checkbox"/> Livy 0.6.0            |
| <input type="checkbox"/> JupyterHub 0.9.6        | <input type="checkbox"/> Tez 0.9.1      | <input type="checkbox"/> Flink 1.8.0           |
| <input type="checkbox"/> Ganglia 3.7.2           | <input type="checkbox"/> HBase 1.4.9    | <input checked="" type="checkbox"/> Pig 0.17.0 |
| <input checked="" type="checkbox"/> Hive 2.3.4   | <input type="checkbox"/> Presto 0.219   | <input type="checkbox"/> ZooKeeper 3.4.13      |
| <input type="checkbox"/> MXNet 1.4.0             | <input type="checkbox"/> Sqoop 1.4.7    | <input type="checkbox"/> Mahout 0.13.0         |
| <input checked="" type="checkbox"/> Hue 4.4.0    | <input type="checkbox"/> Phoenix 4.14.1 | <input type="checkbox"/> Oozie 5.1.0           |
| <input type="checkbox"/> Spark 2.4.2             | <input type="checkbox"/> HCatalog 2.3.4 | <input type="checkbox"/> TensorFlow 1.12.0     |

**Multi-master support** Enable multi-master support **AWS Glue Data Catalog settings (optional)** Use for Hive table metadata **Edit software settings**  Enter configuration  Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

**Add steps (optional)** Step type    Auto-terminate cluster after the last step is completed[Cancel](#)[Next](#)

# EMR Multi-Master

- When selected EMR multi-master will create a cluster with three master nodes
- Multi-master grants support for high availability for Hbase, YARN Resource Manager, HDFS Name Node, Spark, Hive, and Ganglia
- With multi-master Amazon EMR automatically fails over to a standby master node if the primary master node fails

**Step 1: Software and Steps**

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

**Software Configuration****Release** emr-5.24.1

- |  |  |  |
|--|--|--|
| <input checked="" type="checkbox"/> Hadoop 2.8.5 | <input type="checkbox"/> Zeppelin 0.8.1          | <input type="checkbox"/> Livy 0.6.0            |
| <input type="checkbox"/> JupyterHub 0.9.6        | <input type="checkbox"/> Tez 0.9.1               | <input type="checkbox"/> Flink 1.8.0           |
| <input type="checkbox"/> Ganglia 3.7.2           | <input type="checkbox"/> HBase 1.4.9             | <input checked="" type="checkbox"/> Pig 0.17.0 |
| <input checked="" type="checkbox"/> Hive 2.3.4   | <input checked="" type="checkbox"/> Presto 0.219 | <input type="checkbox"/> ZooKeeper 3.4.13      |
| <input type="checkbox"/> MXNet 1.4.0             | <input type="checkbox"/> Sqoop 1.4.7             | <input type="checkbox"/> Mahout 0.13.0         |
| <input checked="" type="checkbox"/> Hue 4.4.0    | <input type="checkbox"/> Phoenix 4.14.1          | <input type="checkbox"/> Oozie 5.1.0           |
| <input checked="" type="checkbox"/> Spark 2.4.2  | <input type="checkbox"/> HCatalog 2.3.4          | <input type="checkbox"/> TensorFlow 1.12.0     |

**Multi-master support** Enable multi-master support **AWS Glue Data Catalog settings (optional)**

- Use for Hive table metadata 
- Use for Presto table metadata 
- Use for Spark table metadata 

**Edit software settings**  Enter configuration  Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

**Add steps (optional)** **Step type**  Auto-terminate cluster after the last step is completed[Cancel](#)[Next](#)

# Glue Data Catalog Settings

- Specify the Glue data catalog as your external metastore supported in Hive, Spark, and Presto
- This configuration option is useful when you require a persistent metastore or a metastore shared by different clusters, services, applications, or AWS accounts

**Step 1: Software and Steps**

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

**Software Configuration**Release   

- |  |   |  |
|--|---|--|
| <input checked="" type="checkbox"/> Hadoop 2.8.5 | <input type="checkbox"/> Zeppelin 0.8.1 | <input type="checkbox"/> Livy 0.6.0            |
| <input type="checkbox"/> JupyterHub 0.9.6        | <input type="checkbox"/> Tez 0.9.1      | <input type="checkbox"/> Flink 1.8.0           |
| <input type="checkbox"/> Ganglia 3.7.2           | <input type="checkbox"/> HBase 1.4.9    | <input checked="" type="checkbox"/> Pig 0.17.0 |
| <input checked="" type="checkbox"/> Hive 2.3.4   | <input type="checkbox"/> Presto 0.219   | <input type="checkbox"/> ZooKeeper 3.4.13      |
| <input type="checkbox"/> MXNet 1.4.0             | <input type="checkbox"/> Sqoop 1.4.7    | <input type="checkbox"/> Mahout 0.13.0         |
| <input checked="" type="checkbox"/> Hue 4.4.0    | <input type="checkbox"/> Phoenix 4.14.1 | <input type="checkbox"/> Oozie 5.1.0           |
| <input type="checkbox"/> Spark 2.4.2             | <input type="checkbox"/> HCatalog 2.3.4 | <input type="checkbox"/> TensorFlow 1.12.0     |

**Multi-master support** Enable multi-master support **AWS Glue Data Catalog settings (optional)** Use for Hive table metadata **Edit software settings**  Enter configuration  Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

**Add steps (optional)** Step type    Auto-terminate cluster after the last step is completed[Cancel](#)[Next](#)

# Edit Software Settings

- Override default configurations for Hadoop applications
- Inline or JSON file from S3
- capacity-scheduler, core-site, hadoop-env, hadoop-log4j, hdfs-site, httpfs-env, https-site, mapred-env, mapred-site, yarn-env, yarn-site, hive-env, hive-exec-log4j, hive-log4j, hive-site, pig-properties, pig-log4j etc.

**Step 1: Software and Steps**

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

**Software Configuration****Release** emr-5.24.1

- |  |  |  |
|--|--|--|
| <input checked="" type="checkbox"/> Hadoop 2.8.5 | <input type="checkbox"/> Zeppelin 0.8.1          | <input type="checkbox"/> Livy 0.6.0            |
| <input type="checkbox"/> JupyterHub 0.9.6        | <input type="checkbox"/> Tez 0.9.1               | <input type="checkbox"/> Flink 1.8.0           |
| <input type="checkbox"/> Ganglia 3.7.2           | <input type="checkbox"/> HBase 1.4.9             | <input checked="" type="checkbox"/> Pig 0.17.0 |
| <input checked="" type="checkbox"/> Hive 2.3.4   | <input checked="" type="checkbox"/> Presto 0.219 | <input type="checkbox"/> ZooKeeper 3.4.13      |
| <input type="checkbox"/> MXNet 1.4.0             | <input type="checkbox"/> Sqoop 1.4.7             | <input type="checkbox"/> Mahout 0.13.0         |
| <input checked="" type="checkbox"/> Hue 4.4.0    | <input type="checkbox"/> Phoenix 4.14.1          | <input type="checkbox"/> Oozie 5.1.0           |
| <input checked="" type="checkbox"/> Spark 2.4.2  | <input type="checkbox"/> HCatalog 2.3.4          | <input type="checkbox"/> TensorFlow 1.12.0     |

**Multi-master support**

- Enable multi-master support [i](#)

**AWS Glue Data Catalog settings (optional)**

- Use for Hive table metadata [i](#)  
 Use for Presto table metadata [i](#)  
 Use for Spark table metadata [i](#)

**Edit software settings** [i](#)

- Enter configuration  Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

**Add steps (optional)** [i](#)**Step type** [Select a step](#)[Configure](#)

- Auto-terminate cluster after the last step is completed

[Cancel](#)[Next](#)

# Add Steps

- A step is a unit of work you submit to the cluster
- Can choose to auto-terminate cluster upon completion
- Preconfigured options for streaming, Hive, Pig, Spark, and customer JARs

## Add steps (optional)

Step type  Auto-terminate

 Select a step

- Streaming program
- Hive program
- Pig program
- Spark application
- Custom JAR

 Configure

# Hardware Configuration

## Hardware Configuration ?

If you need more than 20 EC2 instances, [see this topic](#)

### Instance group configuration

#### Uniform instance groups

Specify a single instance type and purchasing option for each node type.

#### Instance fleets

Specify target capacity and how Amazon EMR fulfills it for each node type. Mix instance types and purchasing options. [Learn more](#)

### Network

vpc-227bc846 (172.31.0.0/16) (default) | private

[Create a VPC](#) ?

### EC2 Subnet

subnet-30d2ee1b | Default in us-east-1a

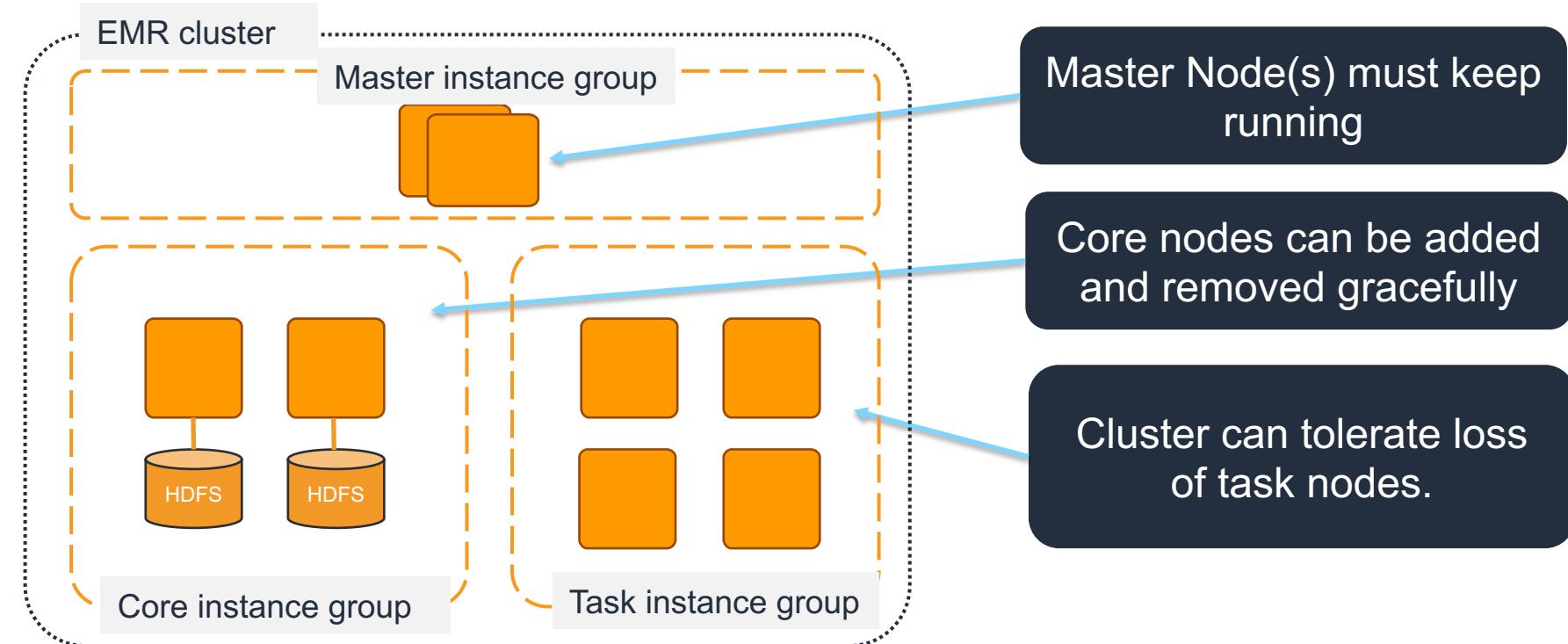
Root device EBS volume size  GiB ?

Choose the instance type, number of instances, and a purchasing option. You can choose to use On-Demand Instances, Spot Instances, or both. The instance type and purchasing option apply to all EC2 instances in each instance group, and you can only specify these options for an instance group when you create it. [Learn more about instance purchasing options](#)

Node type	Instance type	Instance count	Purchasing option	Auto Scaling
Master Master - 1	m3.xlarge 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none Add configuration settings	1 Instances	<input checked="" type="radio"/> On-demand <span style="color: #0072bc;">?</span> <input type="radio"/> Spot <span style="color: #0072bc;">?</span> Use on-demand as max price	Not available for Master
Core Core - 2	m3.xlarge 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none Add configuration settings	2 Instances	<input checked="" type="radio"/> On-demand <span style="color: #0072bc;">?</span> <input type="radio"/> Spot <span style="color: #0072bc;">?</span> Use on-demand as max price	Not enabled
Task Task - 3	m3.xlarge 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none Add configuration settings	0 Instances	<input checked="" type="radio"/> On-demand <span style="color: #0072bc;">?</span> <input type="radio"/> Spot <span style="color: #0072bc;">?</span> Use on-demand as max price	Not enabled



# EMR Clusters



## Auto Scaling rules



Maximum instances:  ⓘ

Minimum instances:  ⓘ

### Scale out

**Default-scale-out-1:** Add  1 instance if YARNMemoryAvailablePercentage is less than  15 for  1 five-minute period with a cooldown of  300 seconds



**Default-scale-out-2:** Add  1 instance if ContainerPendingRatio is greater than  0.75 for  1 five-minute period with a cooldown of  300 seconds



[+ Add rule](#)

### Scale in

**Default-scale-in:** Terminate  1 instance if YARNMemoryAvailablePercentage is greater than  75 for  1 five-minute period with a cooldown of  300 seconds



[+ Add rule](#)



# General Options and Tags

## General Options

Cluster name

Logging i

S3 folder  

Debugging i

Termination protection i

## Tags i

Key	Value (optional)	
Environment	Dev	
Team	Software	
Add a key to create a tag		

# Security – Security Configuration & Security Groups

## ▼ Encryption Options

### Security configuration

Dev Config



## ▼ EC2 Security Groups

An EC2 security group acts as a virtual firewall for your cluster nodes to control inbound and outbound traffic. There are two types of security groups you can configure, [EMR managed security groups](#) and [additional security groups](#). EMR will [automatically update](#) the rules in the EMR managed security groups in order to launch a cluster. [Learn more.](#)

Type	EMR managed security groups EMR will <a href="#">automatically update</a> the selected group	Additional security groups EMR will not modify the selected groups
Master	Default: sg-f712b29a (ElasticMapReduce-master)	No security groups selected
Core & Task	Default: sg-f112b29c (ElasticMapReduce-slave)	No security groups selected

[Create a security group](#)

## Create security configuration

Name Dev Config

### At-rest encryption

Enable and choose options for at-rest data encryption features in Amazon EMR, including Amazon S3 with EMRFS, local volumes attached to cluster instances, and block-transfer encryption for HDFS. [Learn more](#)

Needs to be created  
before you start cluster  
creation

### S3 encryption i

Encryption mode SSE-KMS i

AWS KMS Key SSE-RedshiftCOPY i

### Local disk encryption i

Key provider type AWS KMS i

AWS KMS Key Enter a Key ARN i

### In-transit encryption

Enable and choose options for open-source encryption features that apply to in-transit data for specific applications. Available encryption options may vary by Amazon EMR release. [Learn more](#)

### TLS certificate provider

Certificate provider type PEM i

S3 object s3:///i

## Authentication

### Kerberos

Enable Kerberos authentication for interactions between certain application components on your cluster using Kerberos principals. You can choose between having EMR install a KDC server on the master node of the cluster or you can share your own KDC details that EMR cluster can use. [Learn more](#)

Provider  Cluster dedicated KDC [i](#)  External KDC [i](#)

Ticket lifetime  hours [i](#)

Admin server  [i](#)

KDC server  [i](#)

Active Directory integration  [i](#)

Specify local or external KDC

## IAM roles for EMRFS

### Use IAM roles for EMRFS requests to Amazon S3

When an Amazon S3 request is made through EMRFS, each **Basis for access** is evaluated in order. EMRFS assumes the corresponding **IAM role** for the first match. Specify the cluster **Users** or **Groups**, or **S3 prefixes** as the **Basis for access**. If no **Basis for access** matches the request, EMRFS uses the cluster's EMR role for EC2. [Learn more](#)

IAM role	Basis for access
Users or groups are matched to the cluster application identity making the request.	
<input type="text" value="Admin"/> <a href="#">▼</a>	<input type="text" value="Users"/> <a href="#">▼</a> <input type="text" value="Enter user names, separated by comma"/>
<a href="#">+ Add role mapping</a>	<a href="#">X</a>

# Labs

# Agenda

7:00 AM – 7:15 AM. Welcome and Introductions

7:15 AM - 8:00 AM. Introduction to Amazon EMR

8:00 AM – 8:15 AM. Break / Q &A

8:15 AM – 8:45 AM. EMR Cluster Creation Overview

8:45 AM – 9:15 AM. Lab – EMR Cluster Creation

9:15 AM – 9:30 AM. Break / Q &A

9:30 AM – 10:00 AM. Lab – Hive, Pig, & EMR

10:00 PM – 11:00 AM. Lunch Break

11:00 AM – 11:45 AM. Amazon EMR Well-Architected and Best Practices

11:45 AM – 12:00 PM. Break / Q &A

12:00 PM – 12:45 PM. Lab – Spark-based ETL and Notebooks on EMR

12:45 PM – 1:00 PM. Close Out and Next Steps

To participate in the  
labs, please send an  
email to  
**dbayard@amazon.com**



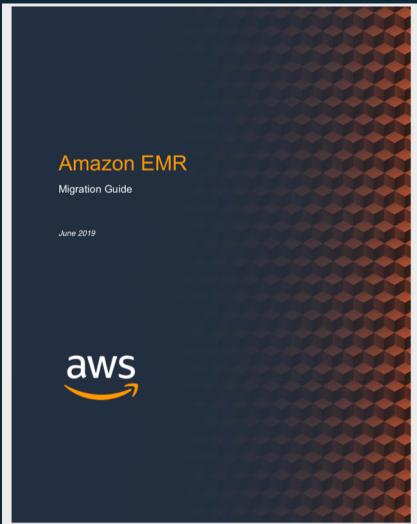
# EMR Well Architected

Feb 24, 2020

# Tip #1: EMR Migration Program

# EMR Migration Program & EMR Migration Guide

<https://aws.amazon.com/emr/emr-migration/>



## Contents

Introduction .....	1	Software Patching .....	40	Data Quality Overview .....	110
Starting Your Journey.....	3	Software Upgrades .....	41	Check your Ingestion Pipeline .....	111
Migration Approaches .....	3	Common Customer Use Cases .....	42	Overall Data Quality Policy .....	112
Prototyping .....	6	Data Migration .....	46	Estimating Impact of Data Quality .....	113
Choosing a Team .....	8	Using Amazon S3 as the Central Data Repository .....	46	Tools to Help with Data Quality .....	115
General Best Practices for Migration .....	9	Large Quantities of Data on an Ongoing Basis .....	49	Support for Your Migration .....	116
Gathering Requirements .....	11	Event and Streaming Data on a Continuous Basis .....	52	AWS Professional Services .....	116
Obtaining On-Premises Metrics .....	11	Optimizing an Amazon S3-Based Central Data Repository .....	54	AWS Partners .....	118
Cost Estimation and Optimization .....	12	Optimizing Cost and Performance .....	56	AWS Support .....	119
Optimizing Costs .....	12	Data Catalog Migration .....	60	Contributors .....	120
Storage Optimization .....	13	Hive Metastore Deployment Patterns .....	60	Additional Resources .....	121
Compute Optimization .....	17	Hive Metastore Migration Options .....	65	Document Revisions .....	122
Cost Estimation Summary .....	18	Multitenancy on EMR .....	68	Appendix A: Questionnaire for Requirements Gathering .....	123
Optimizing Apache Hadoop YARN-based Applications .....	19	Silo Mode .....	68	Security Requirements .....	125
Amazon EMR Cluster Segmentation Schemes .....	22	Shared Mode .....	70	TCO Considerations .....	125
Cluster Characteristics .....	22	Considerations for Implementing Multitenancy on Amazon EMR .....	71	Appendix B: EMR Kerberos Workflow .....	126
Common Cluster Segmentation Schemes .....	24	Extract, Transform, Load (ETL) on Amazon EMR .....	76	EMR Kerberos Cluster Startup Flow for KDC with One-Way Trust .....	126
Additional Considerations for Segmentation .....	25	Orchestration on Amazon EMR .....	76	EMR Kerberos Flow Through Hue Access .....	127
Securing your Resources on Amazon EMR .....	26	Migrating Apache Spark .....	87	EMR Kerberos Flow for Directly Interacting with HiveServer2 .....	128
EMR Security Best Practices .....	26	Migrating Apache Hive .....	90	EMR Kerberos Cluster Startup Flow .....	129
Authentication .....	27	Providing Ad Hoc Query Capabilities .....	97	Appendix C: Sample LDAP Configurations .....	131
Authorization .....	30	Considerations for Presto .....	97	Example LDAP Configuration for Hadoop Group Mapping .....	131
Encryption .....	35	HBase Workloads on Amazon EMR .....	99	Example LDAP Configuration for Hue .....	132
Perimeter Security .....	36	Migrating Apache Impala .....	104	Appendix D: Data Catalog Migration FAQs .....	134
Network Security .....	38	Operational Excellence .....	105		
Auditing .....	38	Upgrading Amazon EMR Versions .....	105		
		General Best Practices for Operational Excellence .....	109		
		Testing and Validation .....	110		

# Tip #2: Use S3



's3://' URI with EMR is the EMRFS Hadoop filesystem implementation.

EMRFS provides features such as Consistent View, S3 Client-Side Encryption, S3 Server-Side Encryption, highly-tuned S3 performance for EMR use-cases, etc:

<https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-fs.html>

<https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-file-systems.html>

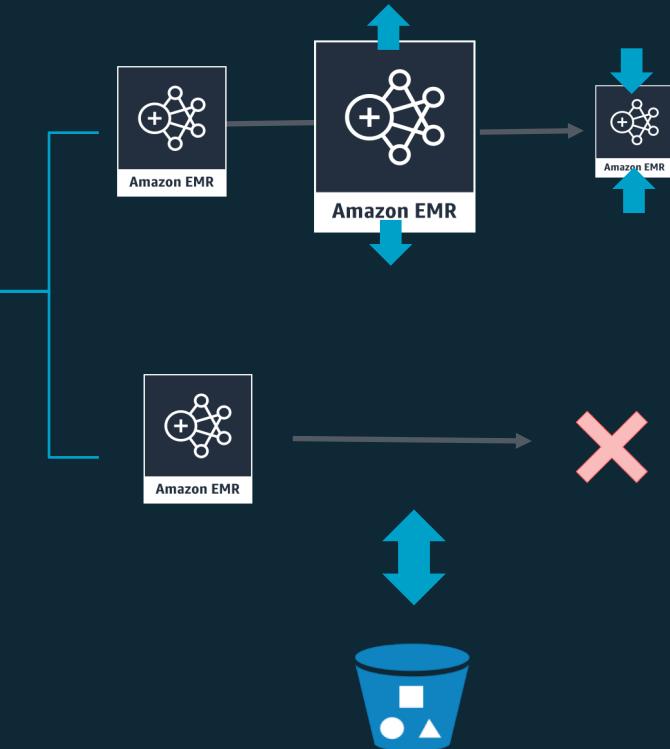
s3a:// is open source and is not maintained by AWS, as such we do not provide engineering support for s3a. s3a does work with EMR, however it's the responsibility of the user to troubleshoot s3a issues.



- Partition your data for improved read performance
  - Read only files the query needs
  - Reduce amount of data scanned
- Optimize file sizes
  - Avoid files that are too small (generally, anything less than 128MB)
  - Fewer calls to S3 (faster listing)
  - Fewer network/HDFS requests
- Compress data set to minimize bandwidth from S3 to EC2
  - Make sure you use splittable compression or have each file be the optimal size for parallelization on your cluster
- Columnar file formats like Parquet can give increased performance on reads

# Cloud Native Patterns: Architect for Elasticity

# Use cloud-native patterns for efficient resource use



**Long running & auto scaling cluster**

**Purpose-built, job-scoped clusters**

# How do you decide?

## Long-running and auto scaling

1. Great for lines of business clusters
2. Great for short-running jobs
3. Ideal to save costs for multi-tenanted data science and data engineering jobs

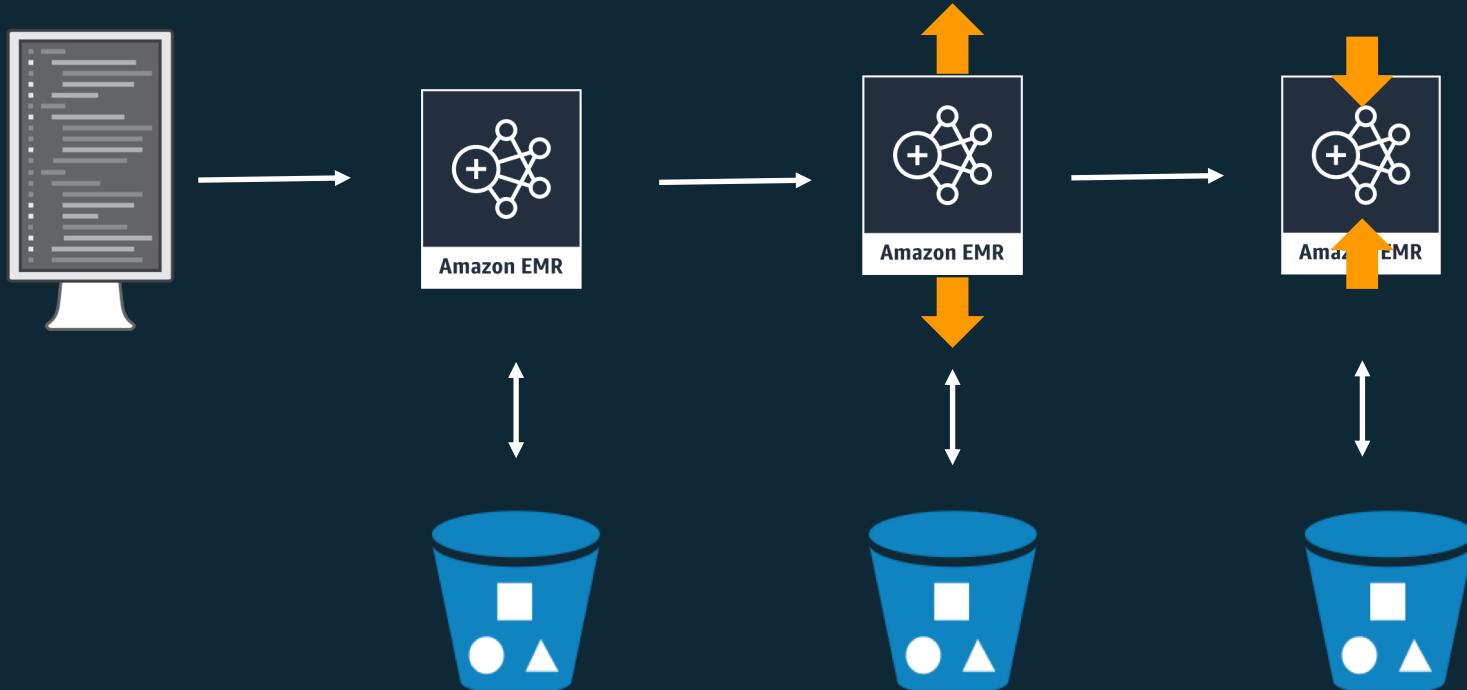
## Transient and job scoped

1. Work well for long-running, job-scoped pipelines
2. Separating production pipelines into job-scoped clusters reduces blast radius

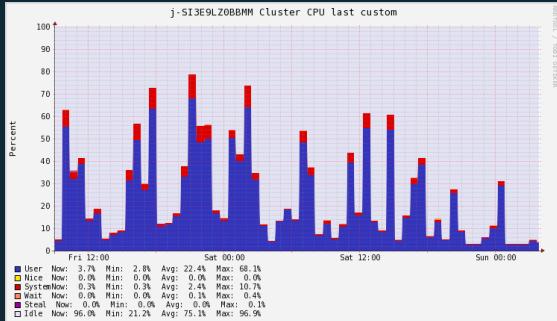
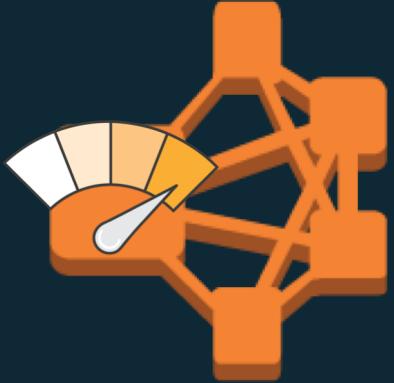
# Persistent Clusters

“Scale up and down”

# Persistent Clusters

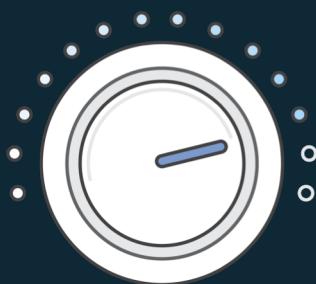
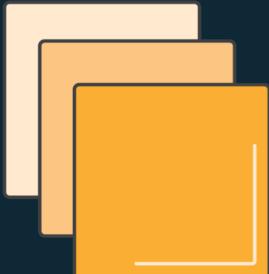


# Auto Scaling Clusters



Threshold

CloudWatch or custom metric



Scaling options



# Hardware Configuration

## Hardware Configuration i

If you need more than 20 EC2 instances, [see this topic](#) .

### Instance group configuration

**Uniform instance groups**

Specify a single instance type and purchasing option for each node type.

**Instance fleets**

Specify target capacity and how Amazon EMR fulfills it for each node type. Mix instance types and purchasing options. [Learn more](#) .

**Network**

vpc-227bc846 (172.31.0.0/16) (default) | private

[Create a VPC](#) 

**EC2 Subnet**

subnet-30d2ee1b | Default in us-east-1a

**Root device EBS volume size**

10 GiB 

Choose the instance type, number of instances, and a purchasing option. You can choose to use On-Demand Instances, Spot Instances, or both. The instance type and purchasing option apply to all EC2 instances in each instance group, and you can only specify these options for an instance group when you create it. [Learn more about instance purchasing options](#) .

Node type	Instance type	Instance count	Purchasing option	Auto Scaling
Master Master - 1 	m3.xlarge  8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none  <a href="#">Add configuration settings</a> 	1 Instances	<input checked="" type="radio"/> <b>On-demand</b>  <input type="radio"/> <b>Spot</b>  <a href="#">Use on-demand as max price</a> 	Not available for Master 
Core Core - 2 	m3.xlarge  8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none  <a href="#">Add configuration settings</a> 	2 Instances	<input checked="" type="radio"/> <b>On-demand</b>  <input type="radio"/> <b>Spot</b>  <a href="#">Use on-demand as max price</a> 	Not enabled 
Task Task - 3 	m3.xlarge  8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none  <a href="#">Add configuration settings</a> 	0 Instances	<input checked="" type="radio"/> <b>On-demand</b>  <input type="radio"/> <b>Spot</b>  <a href="#">Use on-demand as max price</a> 	Not enabled 

# Hardware Configuration – Auto Scaling

Node type	Instance type	Instance count	Purchasing option	Auto Scaling
Master Master - 1	m3.xlarge	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Set max price per instance/hr \$ 0.266	Not available for Master
Core Core - 2	m3.xlarge	2 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price	Not enabled
Task Task - 3	m3.xlarge	0 Instances	<input type="radio"/> On-demand <input checked="" type="radio"/> Spot Set max price per instance/hr \$ 0.266	Not enabled

Auto scaling option

## Auto Scaling rules



Maximum instances:  ⓘ

Minimum instances:  ⓘ

### Scale out

**Default-scale-out-1:** Add  1 instance if YARNMemoryAvailablePercentage is less than  15 for  1 five-minute period with a cooldown of  300 seconds



**Default-scale-out-2:** Add  1 instance if ContainerPendingRatio is greater than  0.75 for  1 five-minute period with a cooldown of  300 seconds



[+ Add rule](#)

### Scale in

**Default-scale-in:** Terminate  1 instance if YARNMemoryAvailablePercentage is greater than  75 for  1 five-minute period with a cooldown of  300 seconds



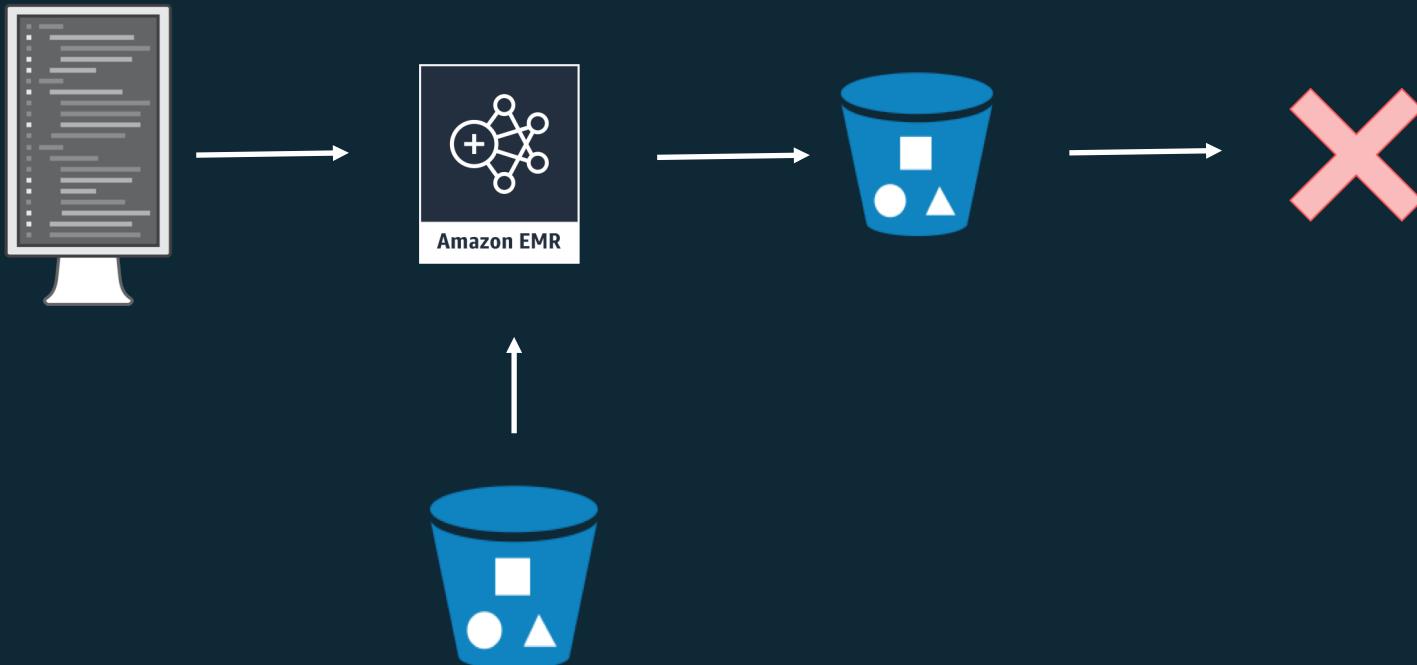
[+ Add rule](#)



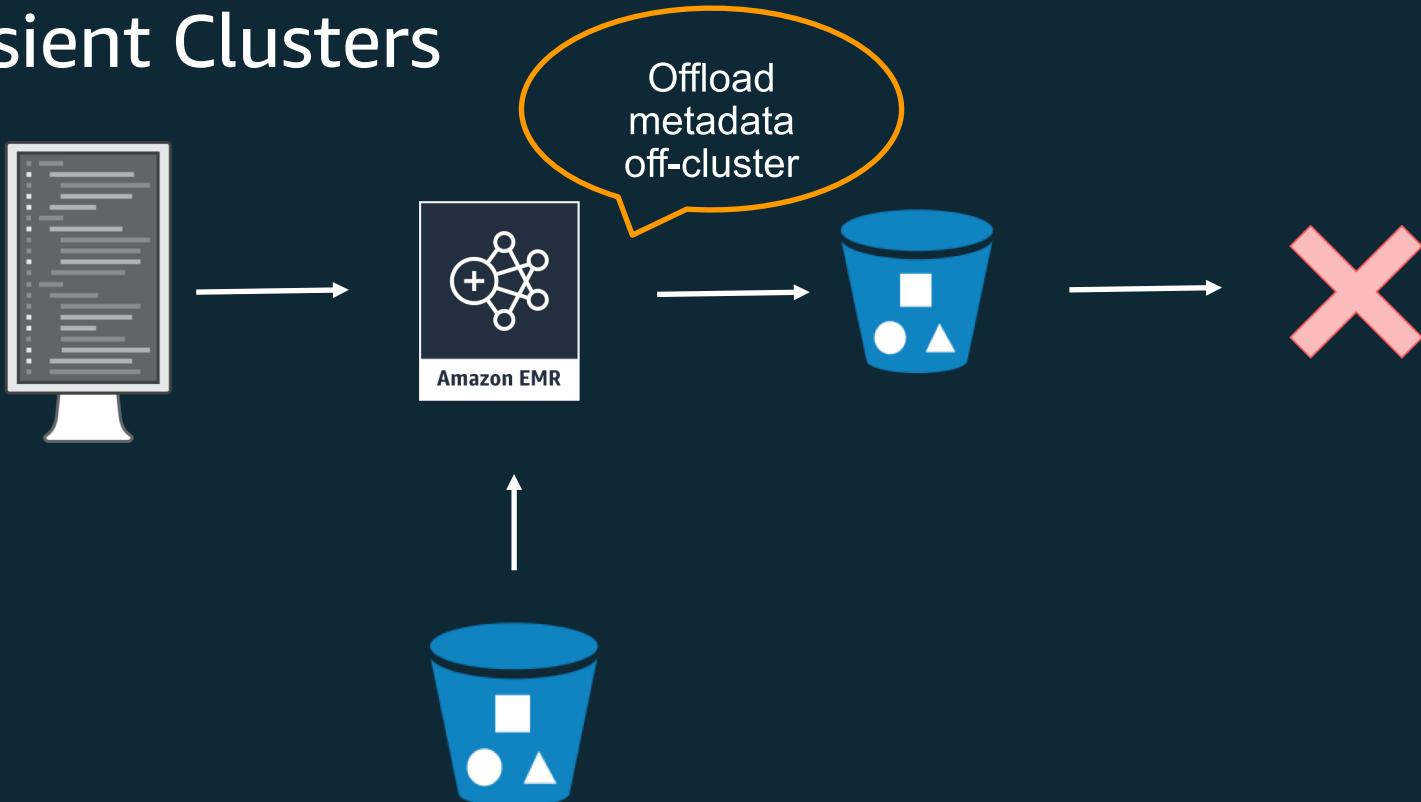
# Transient Clusters

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

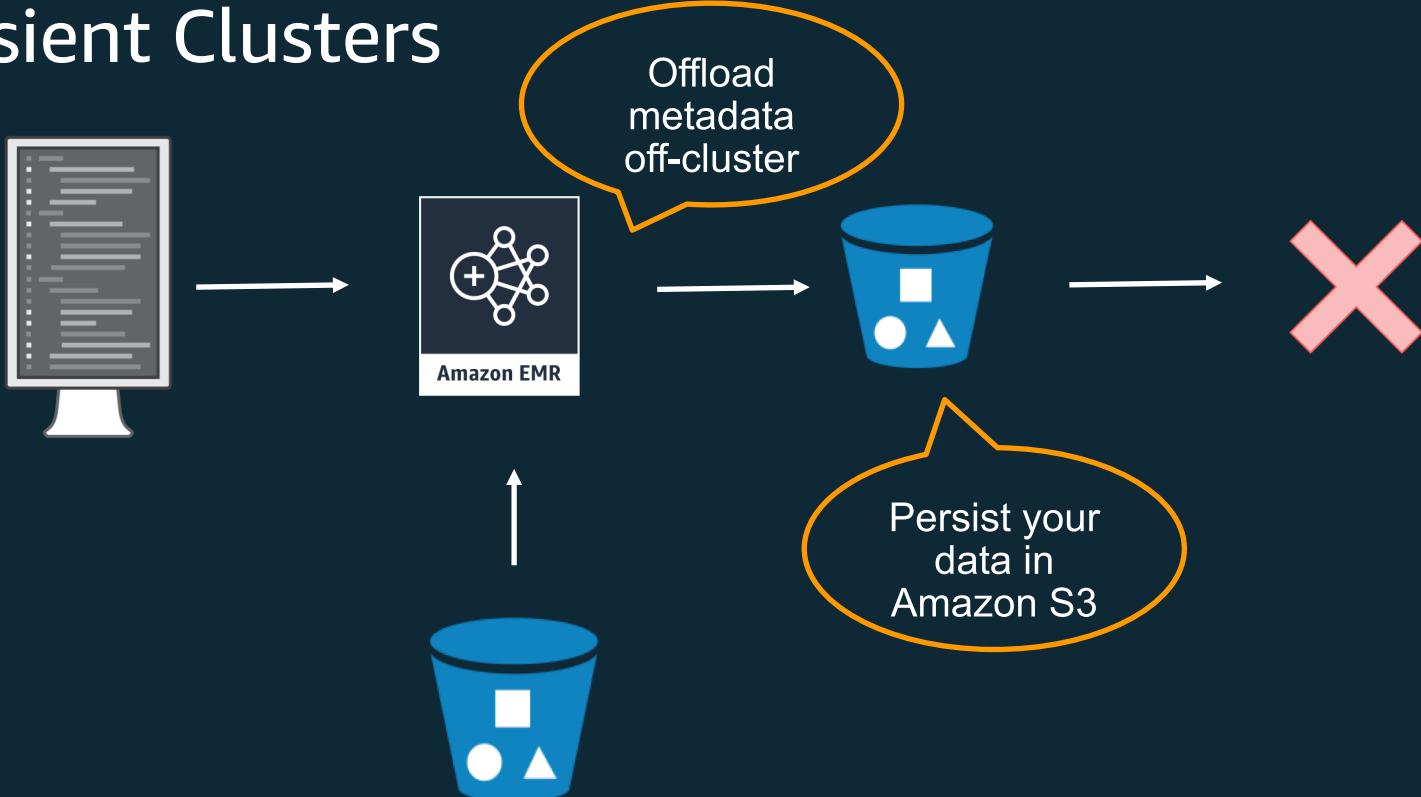
# Transient Clusters



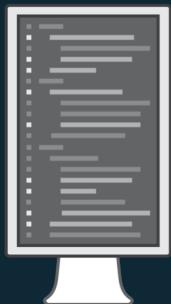
# Transient Clusters



# Transient Clusters



# Transient Clusters



Offload  
metadata  
off-cluster

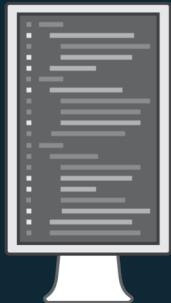


How do you  
submit jobs or  
build pipelines



Persist your  
data in  
Amazon S3

# Transient Clusters



Amazon EMR

Offload  
metadata  
off-cluster



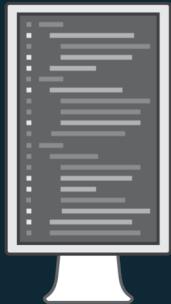
How do you  
submit jobs or  
build pipelines



Using Spot  
to reduce  
cost

Persist your  
data in  
Amazon S3

# Transient Clusters



Amazon EMR

Offload  
metadata  
off-cluster



How do you  
submit jobs or  
build pipelines



Using Spot  
to reduce  
cost

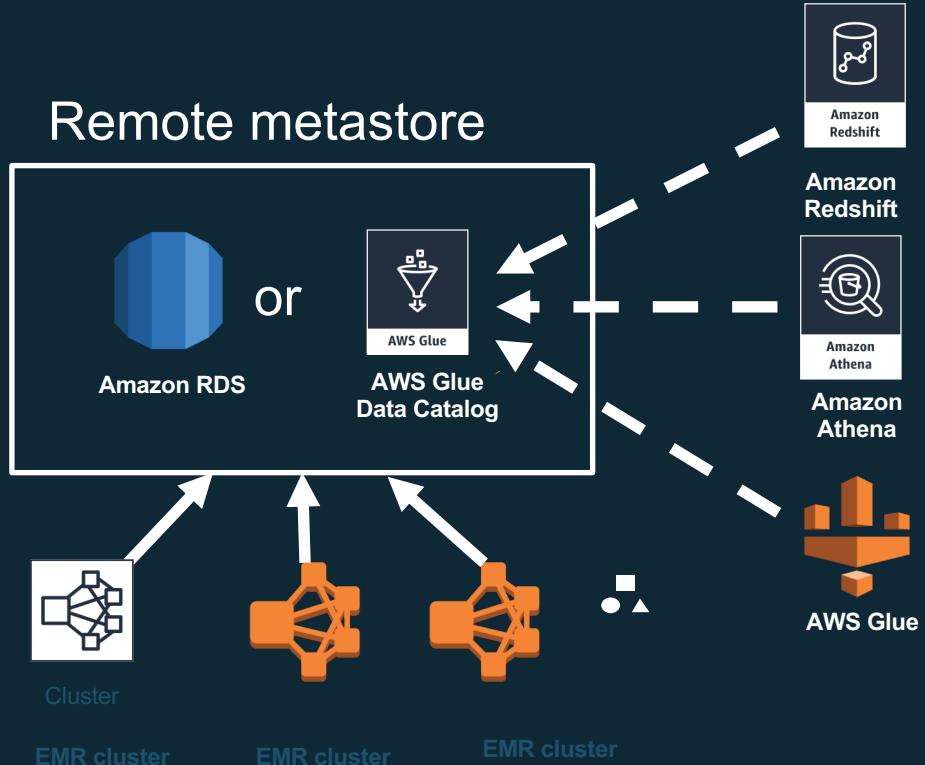
Persist your  
data in  
Amazon S3

Build the  
architecture  
as a template  
for your entire  
org



# Run Stateless

- Maintain metastores off cluster
- Faster startup time lowers cost



# Use AWS Glue Data Catalog as Common Metadata Store

The screenshot shows the AWS Glue Data Catalog interface. On the left, there's a sidebar with navigation links: AWS Glue, Data catalog, Databases, Tables, Connections, Crawlers, Classifiers, ETL, Jobs, Triggers, Dev endpoints, Tutorials, Add crawler, Explore table, Add job, and Resources. The main area displays a table named '2015'. The table details are as follows:

Name	2015
Description	gitarchive
Database	gitarchive
Classification	json
Location	s3://glue-sample-datasets/examples/gitarchive/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 org.openx.data.jsonserde.JsonSerDe
Serde serialization lib	
Serde parameters	path: actor, created_at, id, org, payload, public, repo, type
sizeKey	26129991
objectCount	1
UPDATED_BY_CRAWLER	gitarchive_new
Table properties	CrawlerSchemaSerializerVersion: 1.0, recordCount: 11888, averageRecordSize: 2198
	CrawlerSchemaDeserializerVersion: 1.0, compressionType: none, typeOfData: file

Below the table properties, there's a section titled 'Schema' with a table showing column details:

	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

# Cloud Native Patterns: Optimize Costs

# When to use Spot?

## Cluster Configuration Guidelines:

<https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-instances-guidelines.html>

### When Should You Use Spot Instances?

When you launch a cluster in Amazon EMR, you can choose to launch master, core, or task instances on Spot Instances. Because each type of instance group plays a different role in the cluster, there are implications of launching each node type on Spot Instances. You can't

Application Scenario	Master Node Purchasing Option	Core Nodes Purchasing Option	Task Nodes Purchasing Option
Long-Running Clusters and Data Warehouses	On-Demand	On-Demand or instance-fleet mix	Spot or instance-fleet mix
Cost-Driven Workloads	Spot	Spot	Spot
Data-Critical Workloads	On-Demand	On-Demand	Spot or instance-fleet mix
Application Testing	Spot	Spot	Spot

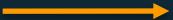
# Hardware Configuration – Spot Integration

Node type	Instance type	Instance count	Purchasing option
Master Master - 1	m3.xlarge  8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none   Add configuration settings 	1 Instances	<input checked="" type="radio"/> On-demand  <input checked="" type="radio"/> Spot  Set max price <input type="text" value="0.266"/> <input checked="" type="radio"/> On-demand  <input type="radio"/> Spot  Use on-demand as max price 
Core Core - 2	m3.xlarge  8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none   Add configuration settings 	2 Instances	<input checked="" type="radio"/> On-demand  <input checked="" type="radio"/> Spot  Set max price <input type="text" value="0.266"/> <input checked="" type="radio"/> On-demand  <input type="radio"/> Spot  Use on-demand as max price 
Task  Task - 3	m3.xlarge  8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none   Add configuration settings 	0 Instances	<input checked="" type="radio"/> On-demand  <input type="radio"/> Spot  Demand as max price 

# Cloud Native Patterns: Automate Job Submission and Scripting

# Options to submit jobs

Submit a Spark application



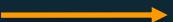
Amazon EMR  
Step API

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



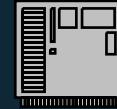
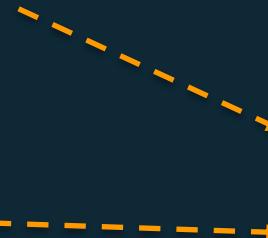
AWS Step Functions

Create a pipeline to schedule job submission or create complex workflows



AWS Data Pipeline

Airflow, Luigi, or other schedulers on EC2



Livy Server



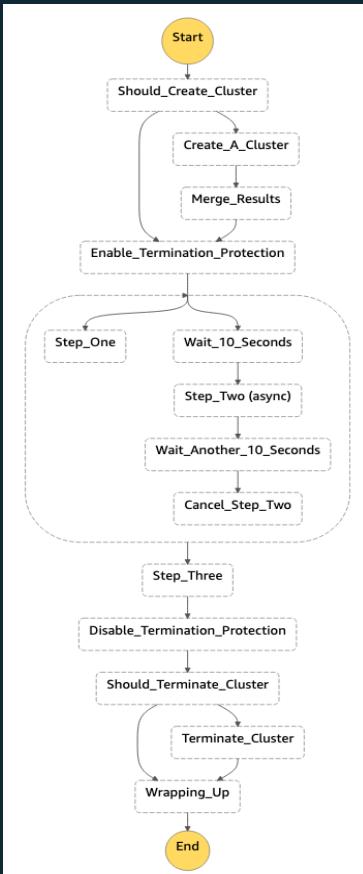
**Amazon EMR**

Use Oozie on your cluster to build DAGs of jobs



# Automate workflows using AWS Step Functions

1. Create, scale, and modify clusters
2. Add, cancel, or run steps in parallel
3. Synchronous and asynchronous steps
4. Handle exceptions/failures
5. Scale clusters up or down
6. Reuse clusters
7. Terminate them



The screenshot shows the AWS Step Functions console interface. On the left, the 'Visual workflow' panel displays the state machine's structure. In the center, the 'Step details' panel shows a specific step named 'Create\_A\_Cluster' with its status as 'In Progress'. An arrow points to the resource link 'EMR Cluster j-24OPRHBDW059'. Below these, the 'Execution history' table lists the execution details:

Name	ID	Status	Creation time (UTC+0)	Elapsed time
WorkflowCluster	j-2GV3YEDSFDF52	Waiting	2019-11-13 12:34 (UTC+0)	10 minutes
Summary				Add step
Master public: ec2-35-244-148-183.eu-west-1.compute.amazonaws.com				
DNS: west-1.compute.amazonaws.com				
Termination protection: Off				
Tags: --				
Hardware				
Master: Running 1 units				
Core: Running 1 units				
Task: --				

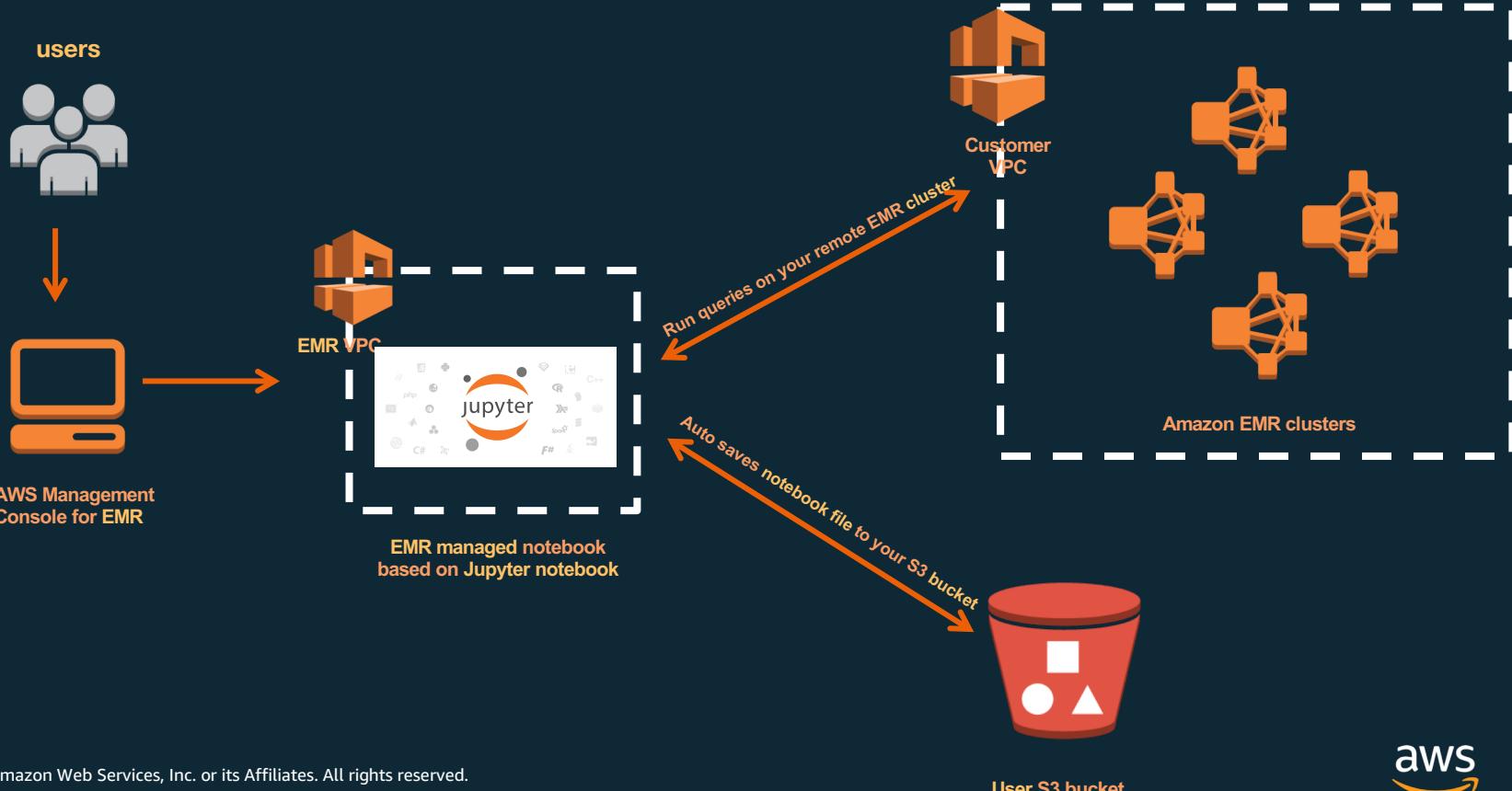
# Cloud Native Patterns: Decouple Notebooks

# EMR Notebooks

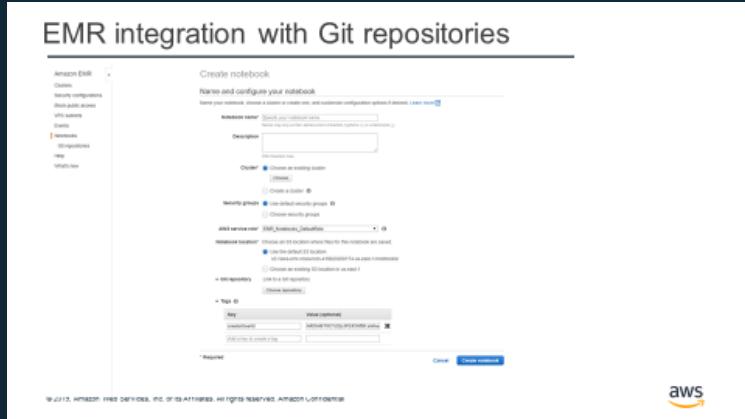
- De-couple Notebooks from Clusters
- Based on Open Source Jupyter Notebooks
- Attach to a cluster to run jobs
- Multiple users can attach to the same cluster (set Autoscaling on a cluster)
- Detach and Attach to other clusters
- Save notebooks to Amazon S3
- Tag-based permissions

# Notebooks

*Off-cluster notebook based on Jupyter notebook application*



# Associate notebooks with repositories

A screenshot of the 'Add repository' dialog in the Amazon EMR console. It asks to 'Connect a GitHub or other Git-based repository with Amazon EMR notebooks.' Fields include 'Repository name' (placeholder 'Enter a repository name'), 'Git repository URL' (placeholder 'https://'), 'Branch' (placeholder 'Enter a branch name'), and 'Git credentials' (radio button selected for 'Use an existing AWS secret', dropdown set to 'git-secret', with other options 'Create a new secret' and 'Use a public repository without credentials'). At the bottom are 'Cancel' and 'Add repository' buttons.

- Associate notebooks with repositories
- Collaborate and share notebooks
- Integrate with pipelines
- Integration with AWS Secrets Manager

# **Cloud Native Patterns: Template Best Practices across Organization**

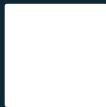
# Self-service with EMR & AWS Service Catalog

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

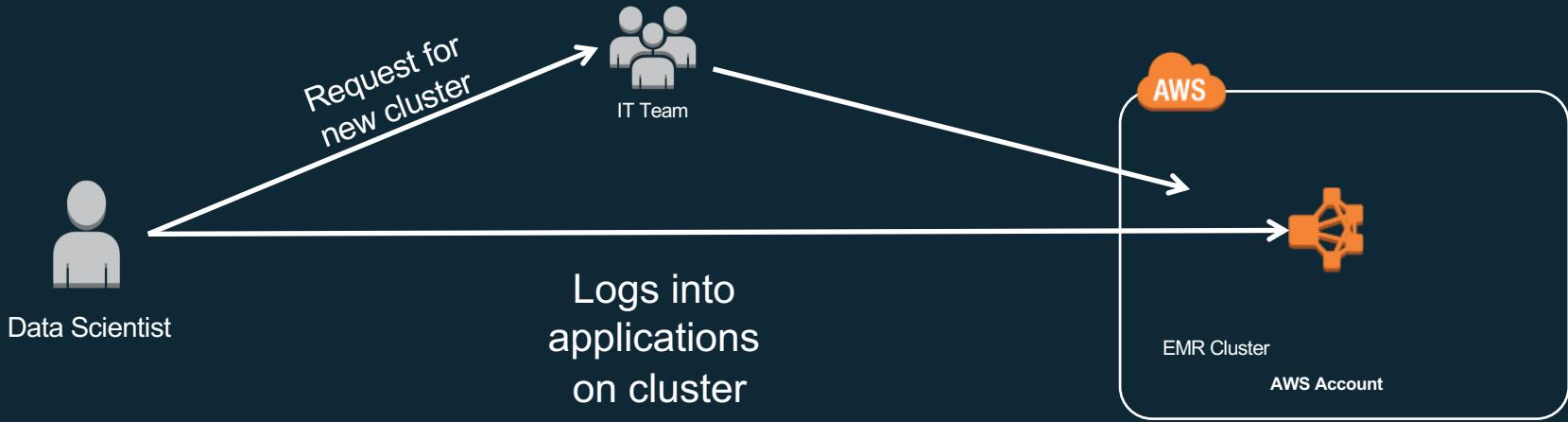


Automate Deployments

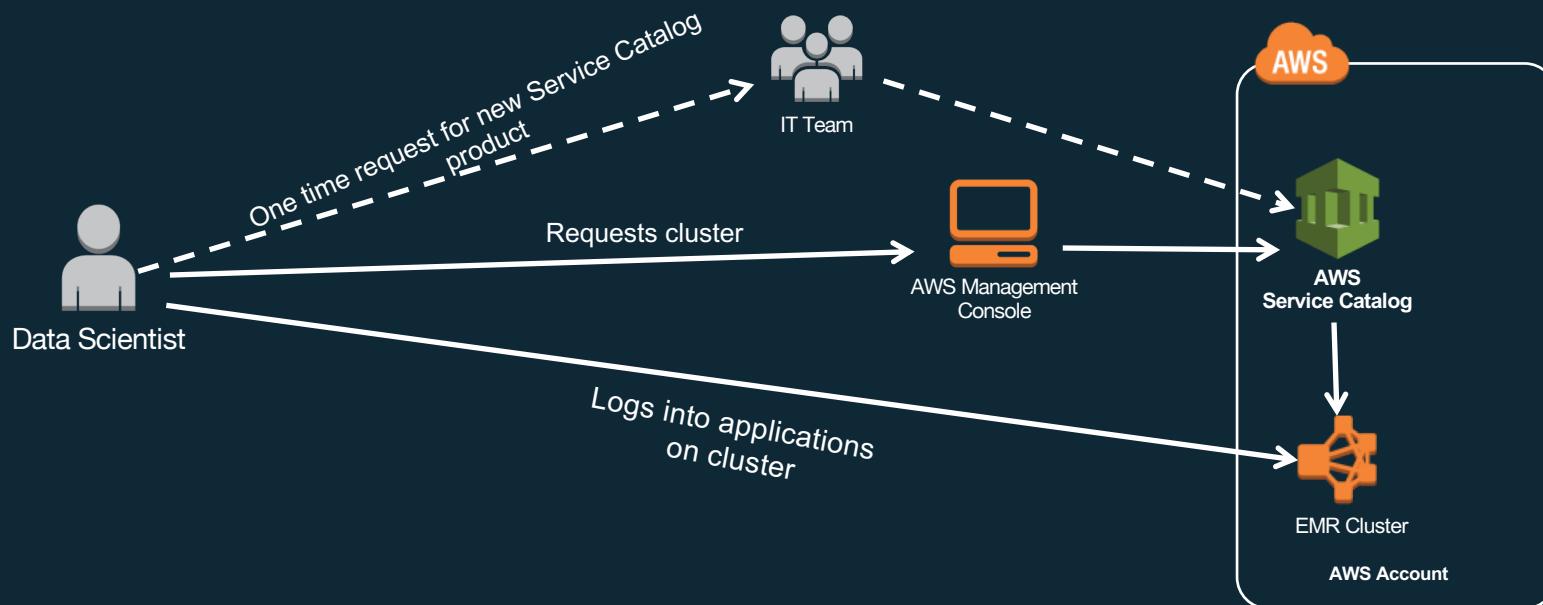


Agile Governance

# Before: EMR Cluster Request Process



# EMR Self-Service with Service Catalog (SC)



# Labs

# Agenda

7:00 AM – 7:15 AM. Welcome and Introductions

7:15 AM - 8:00 AM. Introduction to Amazon EMR

8:00 AM – 8:15 AM. Break / Q &A

8:15 AM – 8:45 AM. EMR Cluster Creation Overview

8:45 AM – 9:15 AM. Lab – EMR Cluster Creation

9:15 AM – 9:30 AM. Break / Q &A

9:30 AM – 10:00 AM. Lab – Hive, Pig, & EMR

10:00 PM – 11:00 AM. Lunch Break

11:00 AM – 11:45 AM. Amazon EMR Well-Architected and Best Practices

11:45 AM – 12:00 PM. Break / Q &A

12:00 PM – 12:45 PM. Lab – Spark-based ETL and Notebooks on EMR

12:45 PM – 1:00 PM. Close Out and Next Steps

To participate in the  
labs, please send an  
email to  
**dbayard@amazon.com**

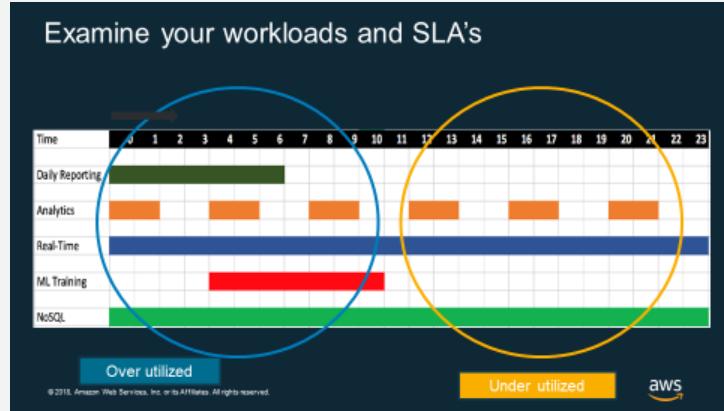
# Wrapup

# The Need for Change. Challenges of on-premises cloud

# The Need for Change. Challenges of on-premises clusters.

# Summary of Challenges

1. **Fixed Cost:** Forced into entering into a multi-year commitment for HW, SW and associated Support Fees driving a sizable capital outlay.
2. **Storage / Compute:** These are tightly coupled, so even if you just want to increase storage for a data retention use case, you incur cost for unnecessary compute.
3. **Always On:** Cluster was always on, which is not resource and cost efficient. Consider the cost of power, FTE's and other data center demands.
4. **Self-Service:** Users want options to deploy only certain services such as Hive, Spark, Presto only; oftentimes just want to test a new service.
5. **Static, Not-scalable & Unused Capacity:** Always fixed number of nodes used; unable to scale based on input size of data (small data sets vs. larger data sets) and lacked consideration for cluster contention.
6. **Outages Impact:** During Production outages, no alternative means to access the cluster. Must consider the implication of downtimes.
7. **Production Upgrade:** Upgrading individual services (i.e. Hive) is laborious to run through DEV, TEST, and PROD cluster environments.
8. **Slow Deployment Cycle:** The cumulative effect of above impacts speed of readying the right infrastructure to solve business and technical use cases.



# The value of cloud-based Hadoop/Spark with Amazon

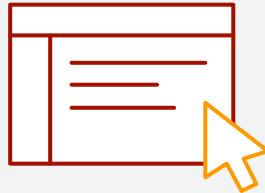
# The value of cloud-based Hadoop/Spark with Amazon EMR.

# Introducing Amazon EMR

## Managed Hadoop and Spark in the cloud



Enterprise-grade



Easy



Lowest cost

# Benefits Summary

## 10 Benefits of migrating to Amazon EMR

1. Decoupled Compute & Storage
2. Built-in Disaster Recovery
3. Turn off your clusters through Transient clusters
4. Agility of Auto-scaling Persistent clusters
5. Leverage Spot pricing for unused EC2 capacity
6. Self-service with AWS Service Catalog
7. Spark Performance Improvements
8. Fully Managed EMR Notebooks
9. Lowest TCO in the Industry, analysts confirm
10. EMR is surrounded by the industry's broadest Analytics ecosystem

Nov. 2018, IDC report confirms:

"EMR provides 57% reduced costs vs. on-premise resulting in 342% ROI over 5 years."



Dec. 2018, Gartner suggests:

"AWS remains the largest Hadoop provider in terms of both revenue and user base."



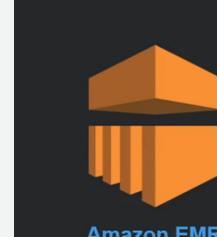
Feb. 2019, Forrester recognizes:

AWS EMR as the Cloud Hadoop/Spark (HARK) Leader



The Forrester Wave™ is a comparative analysis framework from Forrester Research, Inc. Forrester and Forrester Wave™ are trademarks of Forrester Research, Inc. The Forrester Wave™ is a graphical representation of Forrester's call on a market and is plotted using a detailed spreadsheet with exposed scores, weightings, and comments. Forrester does not endorse any vendor, product, or service depicted in the Forrester Wave™. Information is based on best available resources. Opinions reflect judgment at the time and are subject to change.

## EMR Design Options

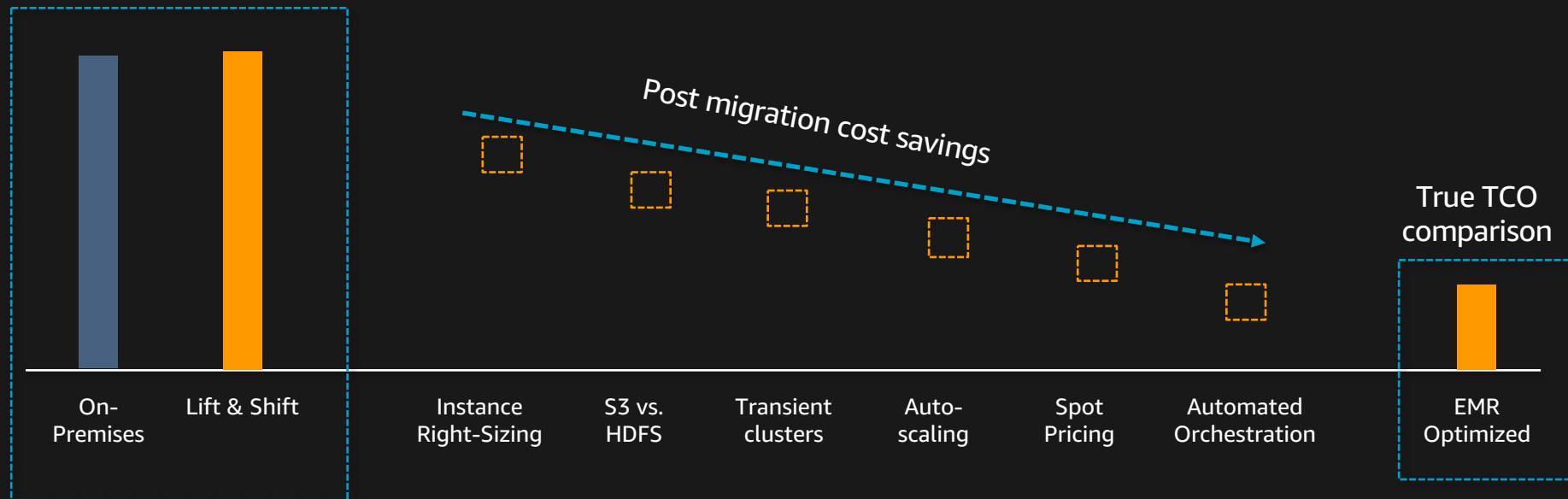


Transient vs. Persistent Cluster  
Amazon S3 vs. local HDFS  
Elastic Cluster vs. Static Cluster  
On-Demand vs. Reserved vs. Spot  
Core Nodes vs. Task Nodes

# A multi-phased approach to best practice deployment

Go beyond a lift & shift to optimize for scale and cost.

Typical TCO comparison



# Resources

Service page

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

Getting started

<https://aws.amazon.com/emr/getting-started/>

EMR Migration Program

<https://aws.amazon.com/emr/emr-migration>

AWS Big Data Blog

<https://aws.amazon.com/blogs/big-data/>