

The background features a dark blue gradient with abstract geometric shapes. On the left, a large triangle is formed by a vertical orange line and a diagonal orange line. On the right, a large curved shape in shades of blue and orange sweeps across the frame. The text is positioned in the upper right area.

AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

ANT401

Deep dive: Accelerating Apache Spark with Amazon EMR

Radhika Ravirala

Principal Product Manager, Amazon EMR
Amazon Web Services

Neil Mukerje

Principal Product Manager, Amazon EMR
Amazon Web Services



Amazon EMR

EASILY RUN SPARK WORKLOADS AT SCALE

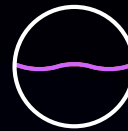
Latest versions
and 100% API
compatible



Differentiated
performance at
lower cost



Optimized for
Amazon S3 as
a data lake



Easy and
scalable



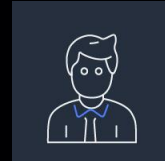
Key personas in a typical organization



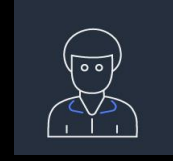
Maria – the data scientist



Ana – the data analyst



Richard – the data engineer



Carlos – the administrator

**How does Spark on Amazon EMR
empower these personas?**

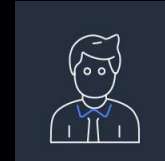
The data analyst and data scientist



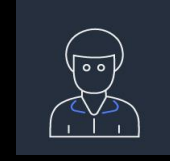
Maria – **the data scientist**



Ana – **the data analyst**



Richard – the data engineer



Carlos – the administrator

Builds and trains
ML models

Performs
interactive data
analysis in
notebooks

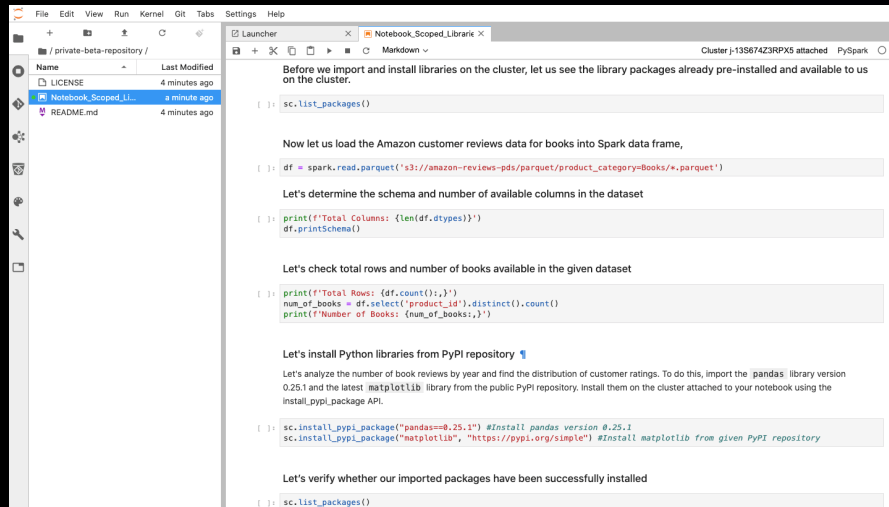
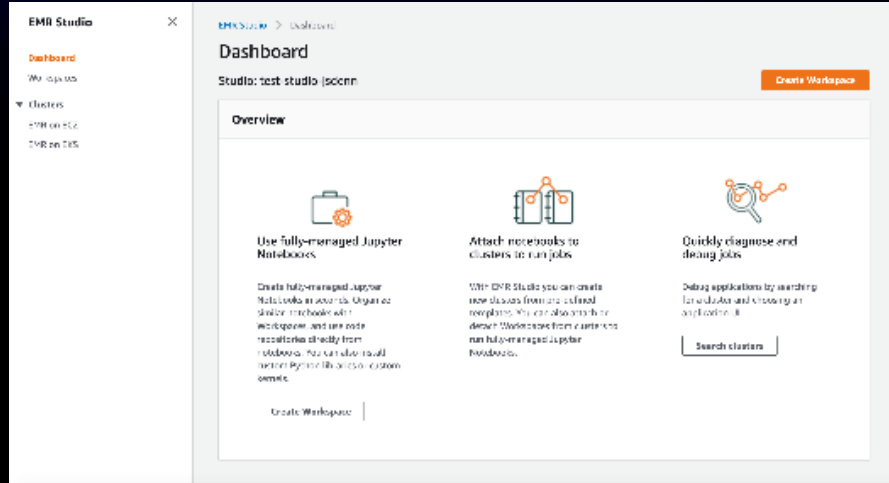
Prepares reports

Performs ad-hoc
analysis (often
using SQL)

Simplify ML and interactive analysis using Amazon EMR

Amazon EMR Studio

FULLY MANAGED IDE FOR INTERACTIVE DATA ANALYTICS: DEVELOP, VISUALIZE, AND DEBUG APPLICATIONS



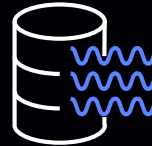
Single sign-on
integration with IdP



Fully managed Jupyter
notebooks



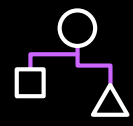
Integrated with Git
repositories



Simplified debugging
with Spark UI and
YARN UI



Browse, create, or delete
Amazon EMR clusters



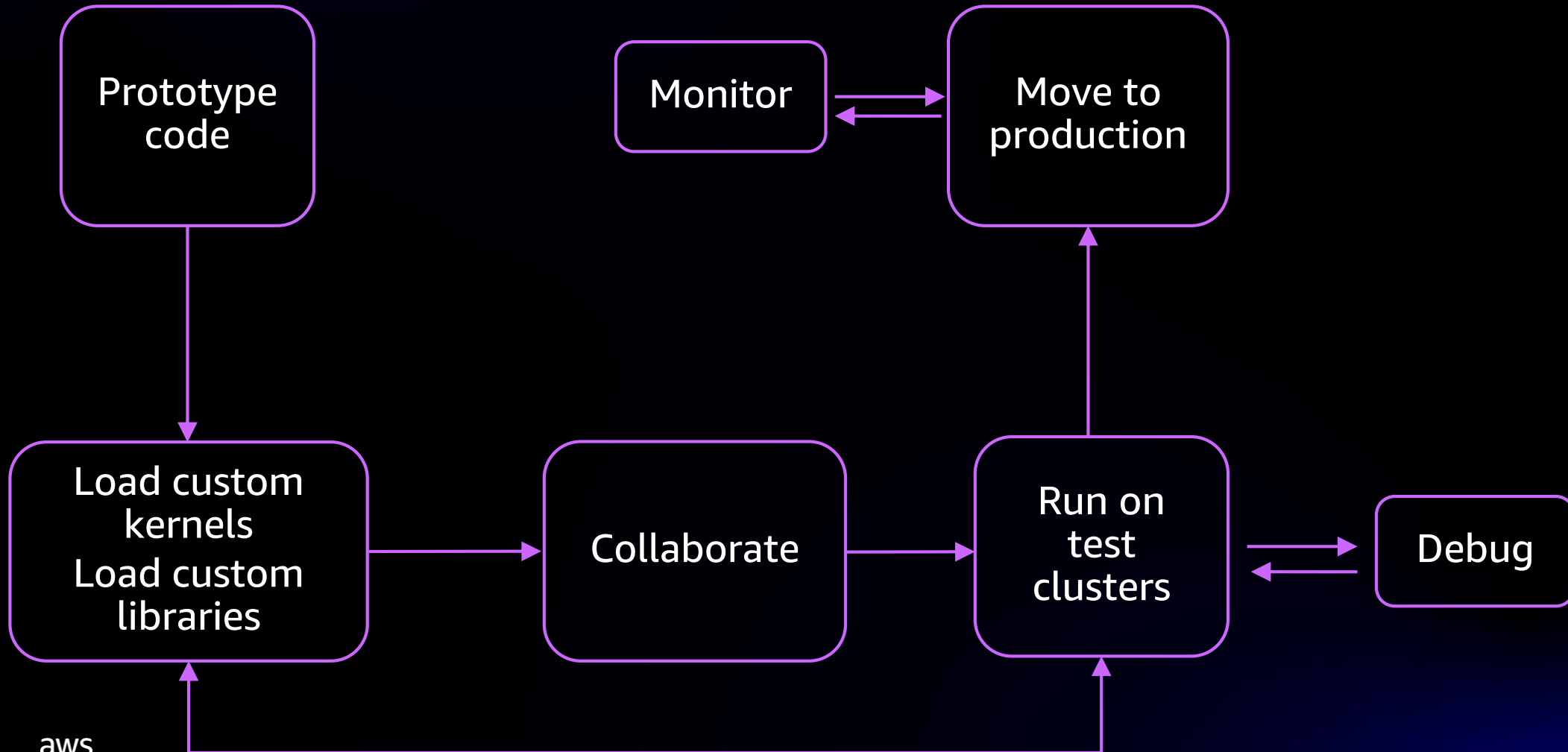
Run notebooks in
workflows using APIs



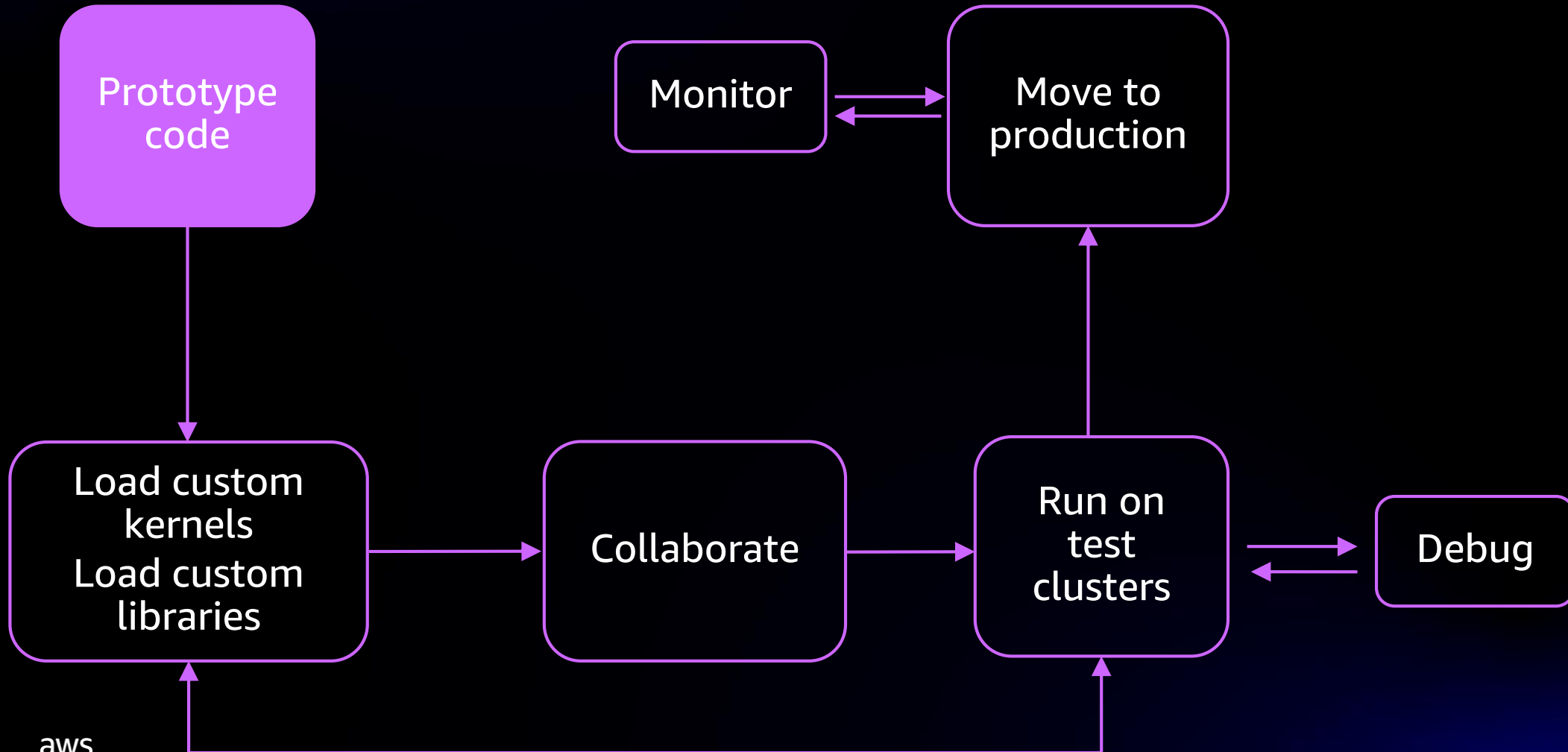
Run interactive data analysis
using Amazon EMR or
Amazon EKS clusters



Data science and engineering workflows



Data science and engineering workflows



Log into Amazon EMR Studio without logging into the AWS Management Console



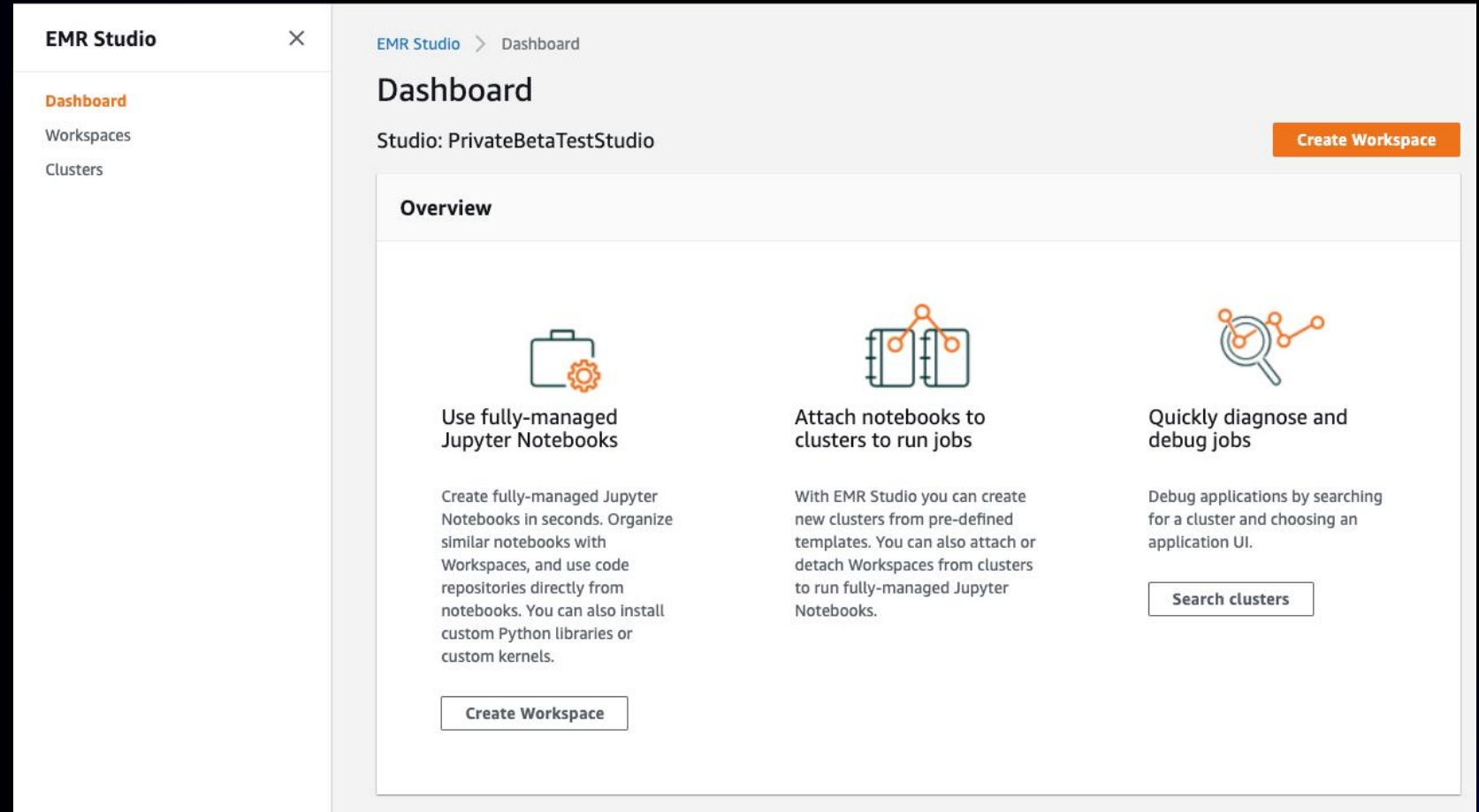
Data scientist



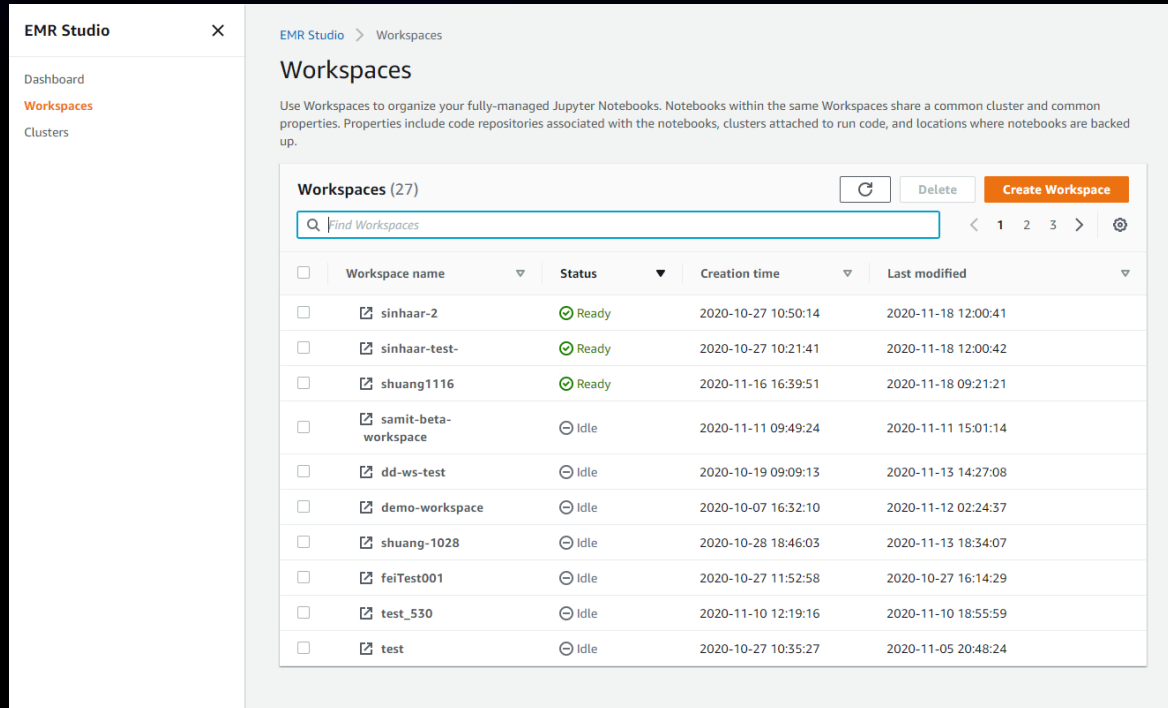
Data engineer



Log in with corporate identity using AWS Single Sign-On/ AWS IAM

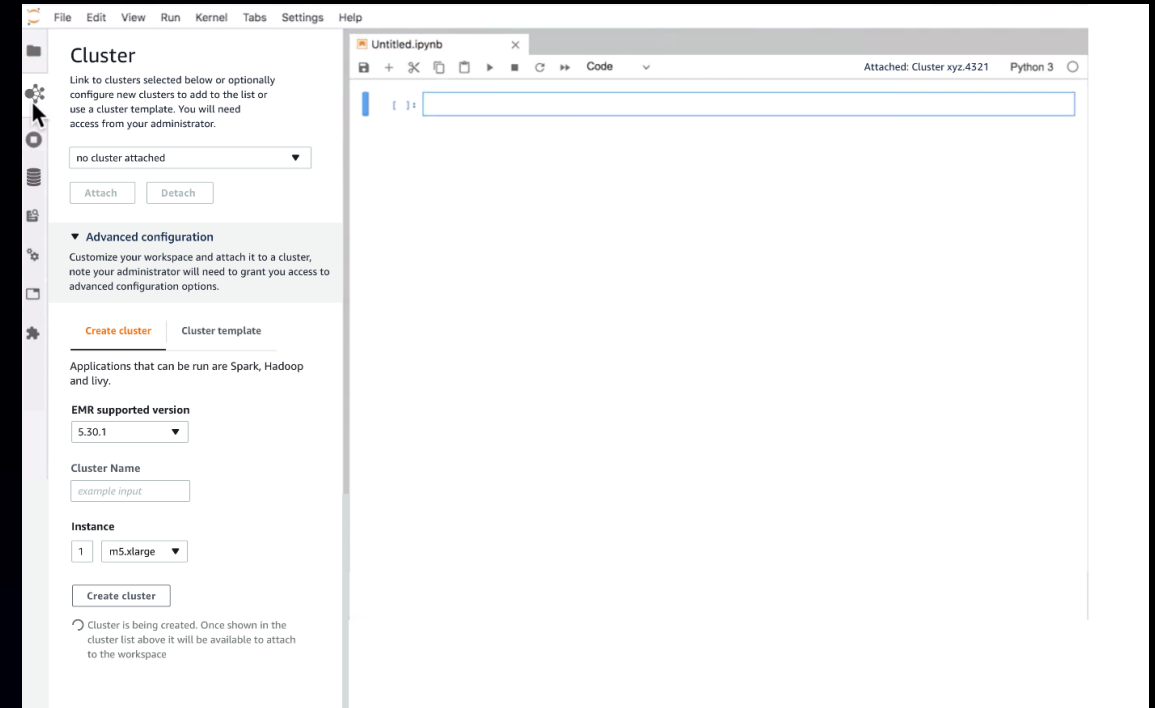


Amazon EMR Studio gives you a fully managed notebook



Workspaces help organize notebooks

Workspaces share similar properties



Fully managed Jupyter notebooks

Write Python, R, PySpark, Scala

Workspace: Single IDE for interactive data analysis

WITH CURATED LIST OF EXTENSIONS TO ENHANCE THE JUPYTER EXPERIENCE

File browser

**Attach/detach
Amazon EMR clusters**

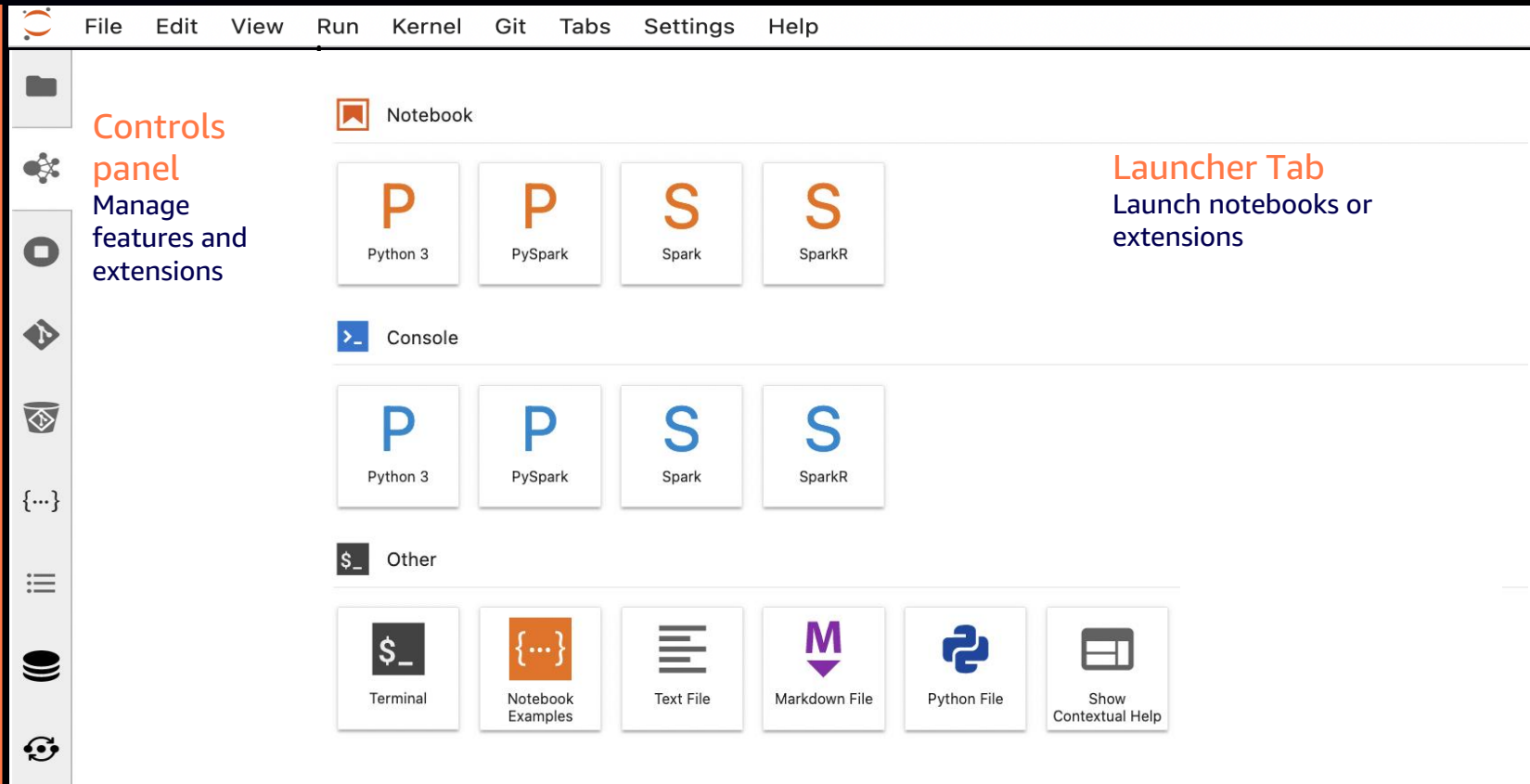
Kernel management

Git operations

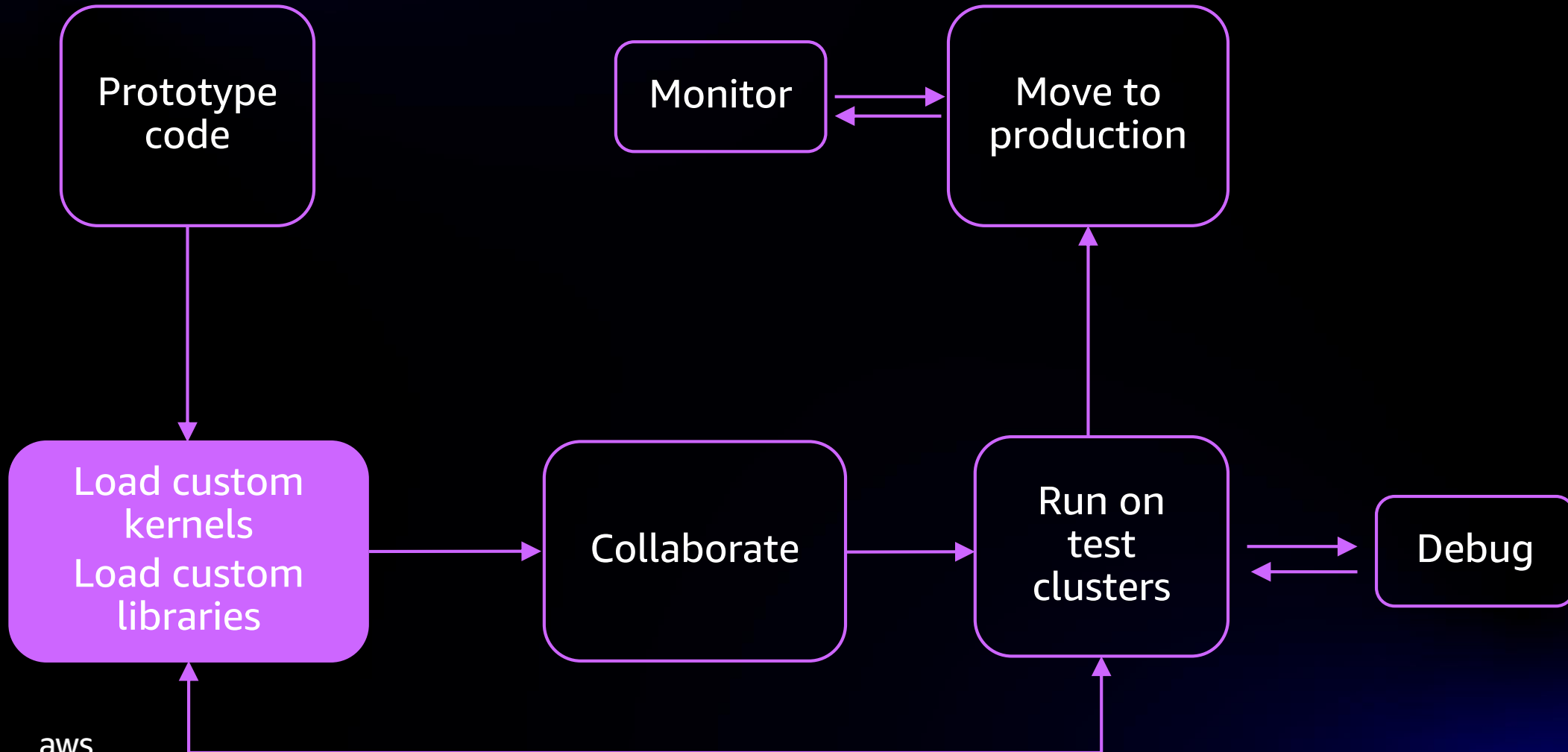
Workspace Git link

Sample notebooks

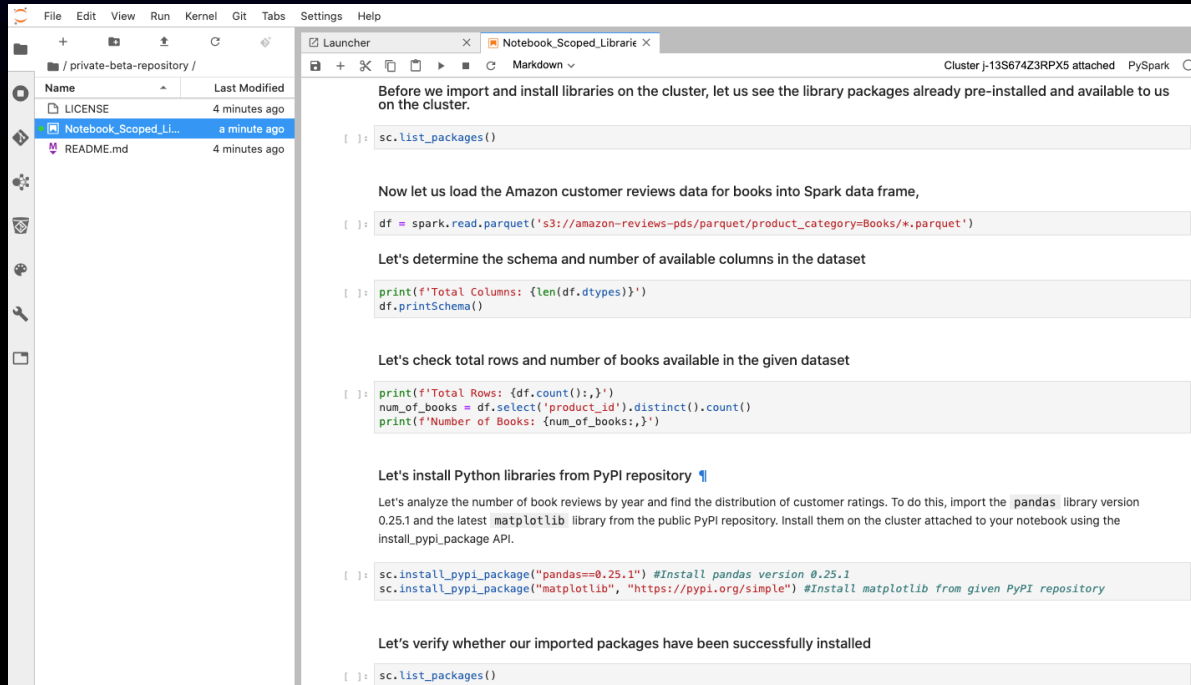
Table of contents



Data science and engineering workflows



Simple to load custom libraries and kernels



The screenshot shows a Jupyter Notebook with a file explorer on the left and a code editor on the right. The file explorer shows a directory with files like LICENSE, Notebook_Scoped_Li..., and README.md. The code editor contains the following text:

```
Before we import and install libraries on the cluster, let us see the library packages already pre-installed and available to us on the cluster.
```

```
[ ]: sc.list_packages()
```

Now let us load the Amazon customer reviews data for books into Spark data frame,

```
[ ]: df = spark.read.parquet('s3://amazon-reviews-pds/parquet/product_category=Books/*.parquet')
```

Let's determine the schema and number of available columns in the dataset

```
[ ]: print(f'Total Columns: {len(df.dtypes)}')
df.printSchema()
```

Let's check total rows and number of books available in the given dataset

```
[ ]: print(f'Total Rows: {df.count().count()}')
num_of_books = df.select('product_id').distinct().count()
print(f'Number of Books: {num_of_books}')

```

Let's install Python libraries from PyPI repository

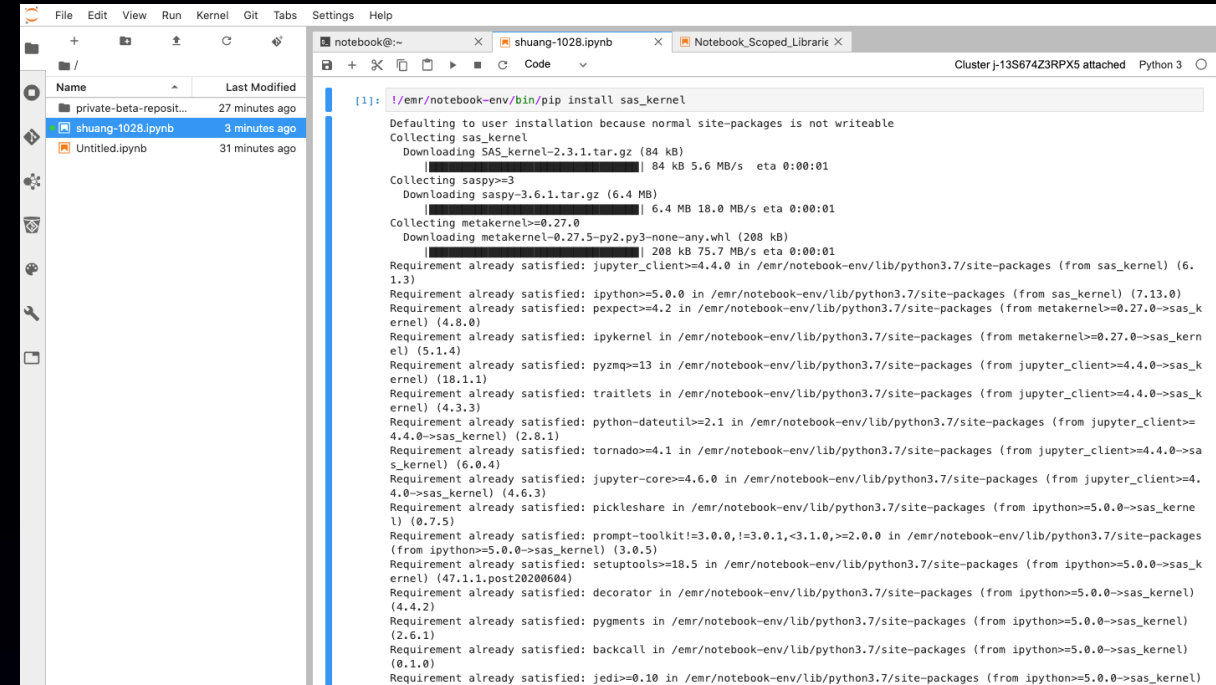
Let's analyze the number of book reviews by year and find the distribution of customer ratings. To do this, import the pandas library version 0.25.1 and the latest matplotlib library from the public PyPI repository. Install them on the cluster attached to your notebook using the install_pypi_package API.

```
[ ]: sc.install_pypi_package("pandas==0.25.1") #Install pandas version 0.25.1
sc.install_pypi_package("matplotlib", "https://pypi.org/simple") #Install matplotlib from given PyPI repository
```

Let's verify whether our imported packages have been successfully installed

```
[ ]: sc.list_packages()
```

Install notebook-scoped libraries with PySpark kernel



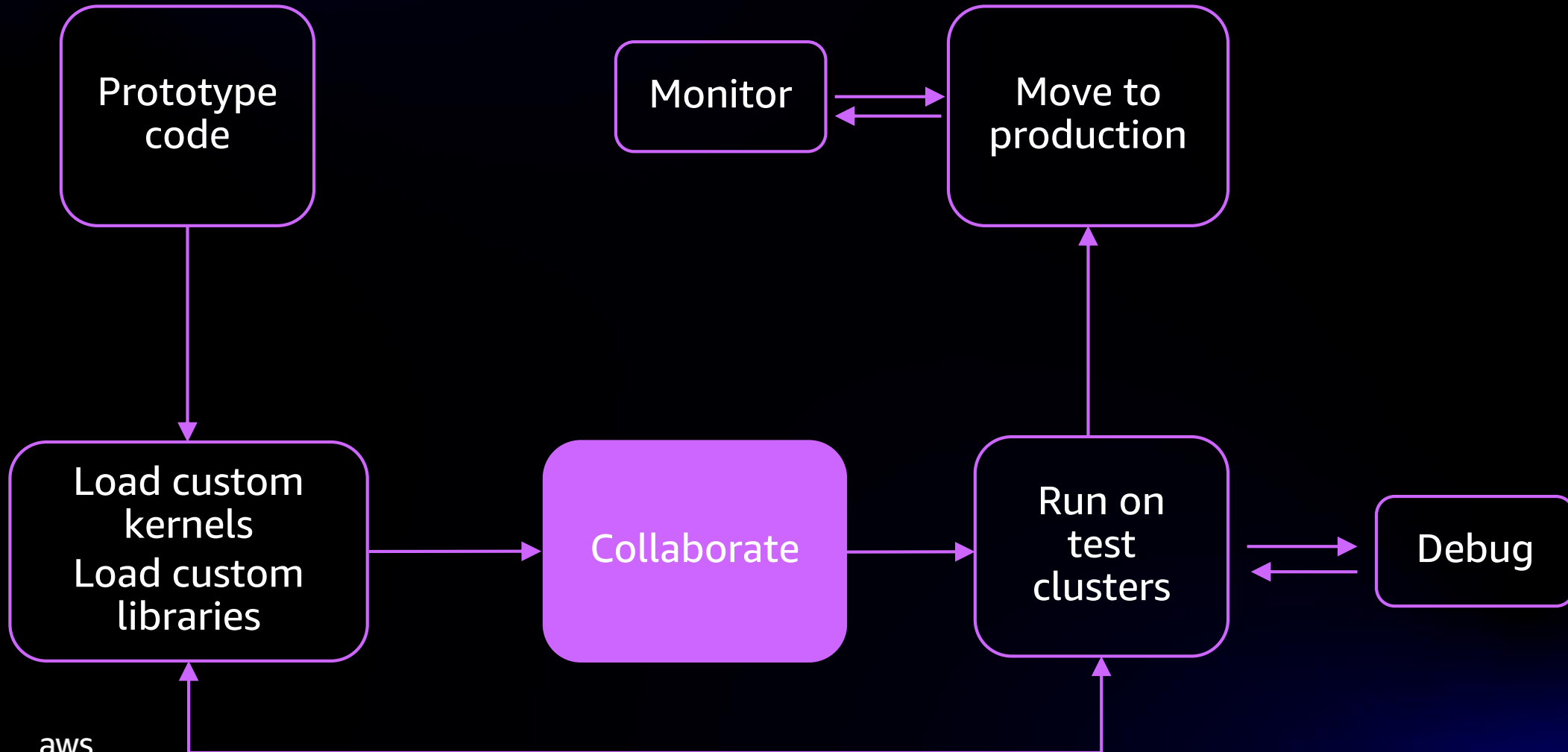
The screenshot shows a Jupyter Notebook with a file explorer on the left and a code editor on the right. The file explorer shows a directory with files like private-beta-reposit..., shuang-1028.ipynb, and Untitled.ipynb. The code editor contains the following text:

```
[1]: !/emr/notebook-env/bin/pip install sas_kernel
```

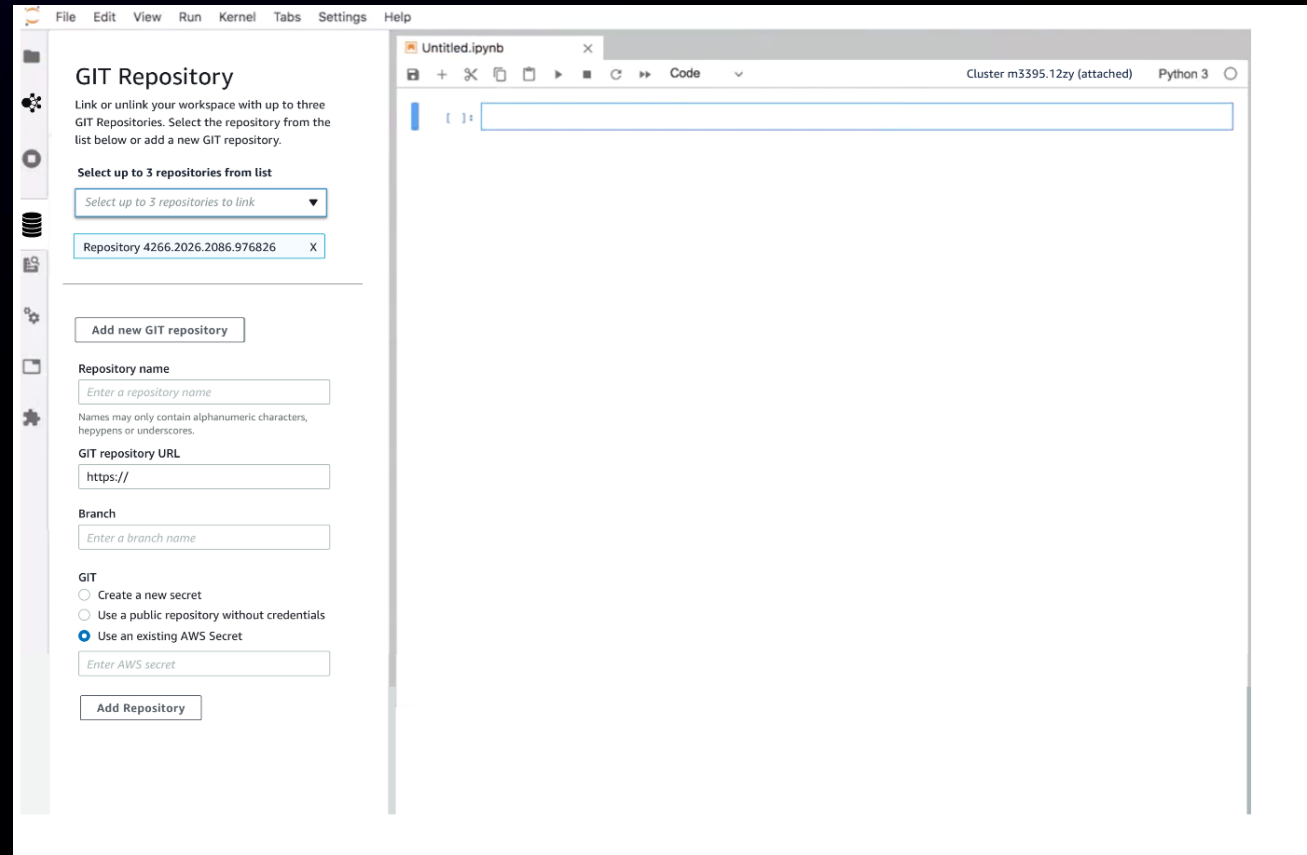
Defaulting to user installation because normal site-packages is not writeable
Collecting sas_kernel
 Downloading SAS_kernel-2.3.1.tar.gz (84 kB)
 [REDACTED] 84 kB 5.6 MB/s eta 0:00:01
Collecting saspy>=3
 Downloading saspy-3.6.1.tar.gz (6.4 MB)
 [REDACTED] 6.4 MB 18.0 MB/s eta 0:00:01
Collecting metakernel>=0.27.0
 Downloading metakernel-0.27.5-py2.py3-none-any.whl (208 kB)
 [REDACTED] 208 kB 75.7 MB/s eta 0:00:01
Requirement already satisfied: jupyter_client>=4.4.0 in /emr/notebook-env/lib/python3.7/site-packages (from sas_kernel) (6.1.3)
Requirement already satisfied: ipython>=5.0.0 in /emr/notebook-env/lib/python3.7/site-packages (from sas_kernel) (7.13.0)
Requirement already satisfied: pexpect>=4.2 in /emr/notebook-env/lib/python3.7/site-packages (from metakernel>=0.27.0->sas_kernel) (4.8.0)
Requirement already satisfied: ipykernel in /emr/notebook-env/lib/python3.7/site-packages (from metakernel>=0.27.0->sas_kernel) (5.1.4)
Requirement already satisfied: pyzmq>=13 in /emr/notebook-env/lib/python3.7/site-packages (from jupyter_client>=4.4.0->sas_kernel) (18.1.1)
Requirement already satisfied: traitlets in /emr/notebook-env/lib/python3.7/site-packages (from jupyter_client>=4.4.0->sas_kernel) (4.3.3)
Requirement already satisfied: python-dateutil>=2.1 in /emr/notebook-env/lib/python3.7/site-packages (from jupyter_client>=4.4.0->sas_kernel) (2.8.1)
Requirement already satisfied: tornado>=4.1 in /emr/notebook-env/lib/python3.7/site-packages (from jupyter_client>=4.4.0->sas_kernel) (6.0.4)
Requirement already satisfied: jupyter-core>=4.6.0 in /emr/notebook-env/lib/python3.7/site-packages (from jupyter_client>=4.4.0->sas_kernel) (4.6.3)
Requirement already satisfied: pickleshare in /emr/notebook-env/lib/python3.7/site-packages (from ipython>=5.0.0->sas_kernel) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /emr/notebook-env/lib/python3.7/site-packages (from ipython>=5.0.0->sas_kernel) (3.0.5)
Requirement already satisfied: setuptools>=18.5 in /emr/notebook-env/lib/python3.7/site-packages (from ipython>=5.0.0->sas_kernel) (47.1.1.post20200604)
Requirement already satisfied: decorator in /emr/notebook-env/lib/python3.7/site-packages (from ipython>=5.0.0->sas_kernel) (4.4.2)
Requirement already satisfied: pygments in /emr/notebook-env/lib/python3.7/site-packages (from ipython>=5.0.0->sas_kernel) (2.6.1)
Requirement already satisfied: backcall in /emr/notebook-env/lib/python3.7/site-packages (from ipython>=5.0.0->sas_kernel) (0.1.0)
Requirement already satisfied: jedi>=0.10 in /emr/notebook-env/lib/python3.7/site-packages (from ipython>=5.0.0->sas_kernel)

Install additional Python libraries and kernels on the leader node of the cluster

Data science and engineering workflows



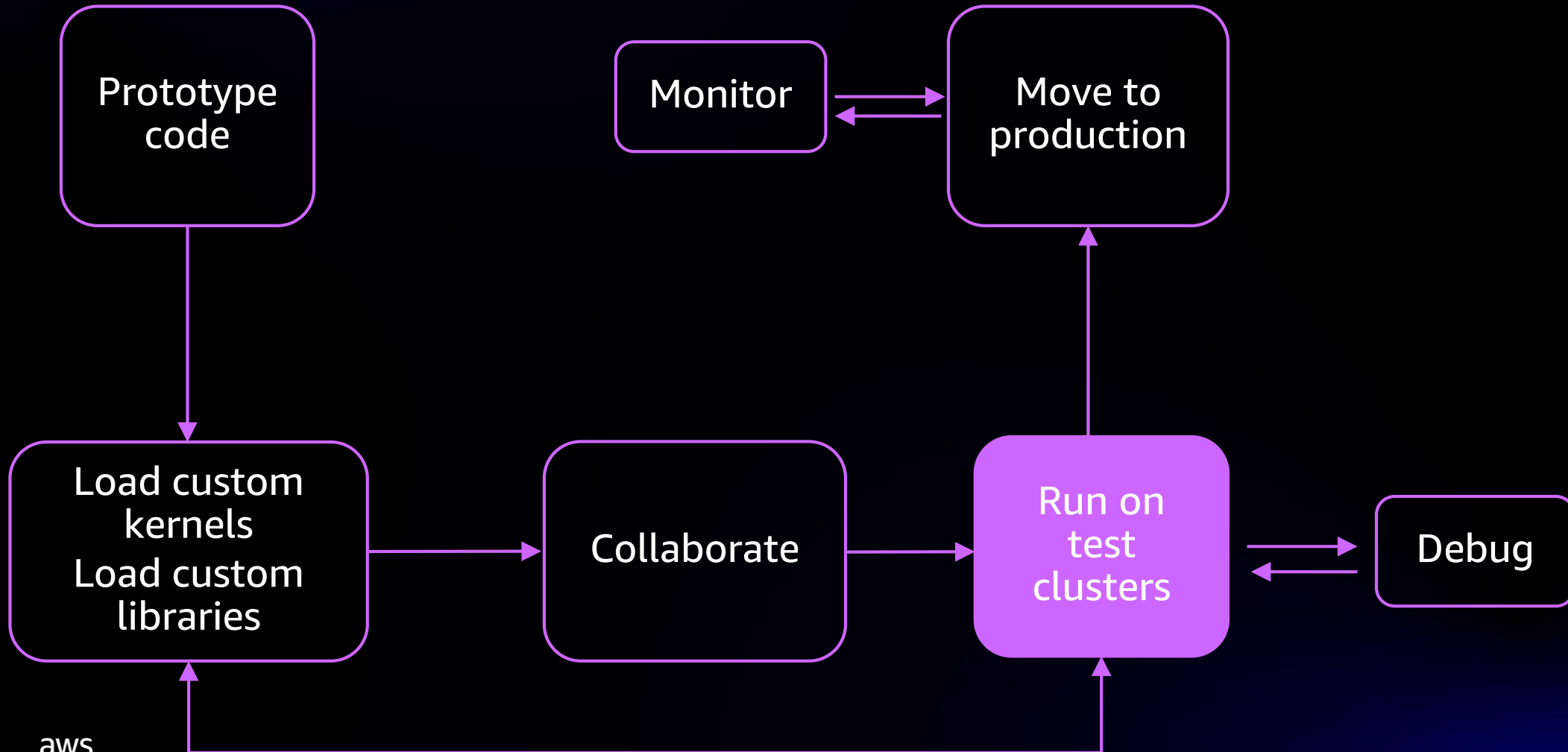
Simple to connect to code repositories



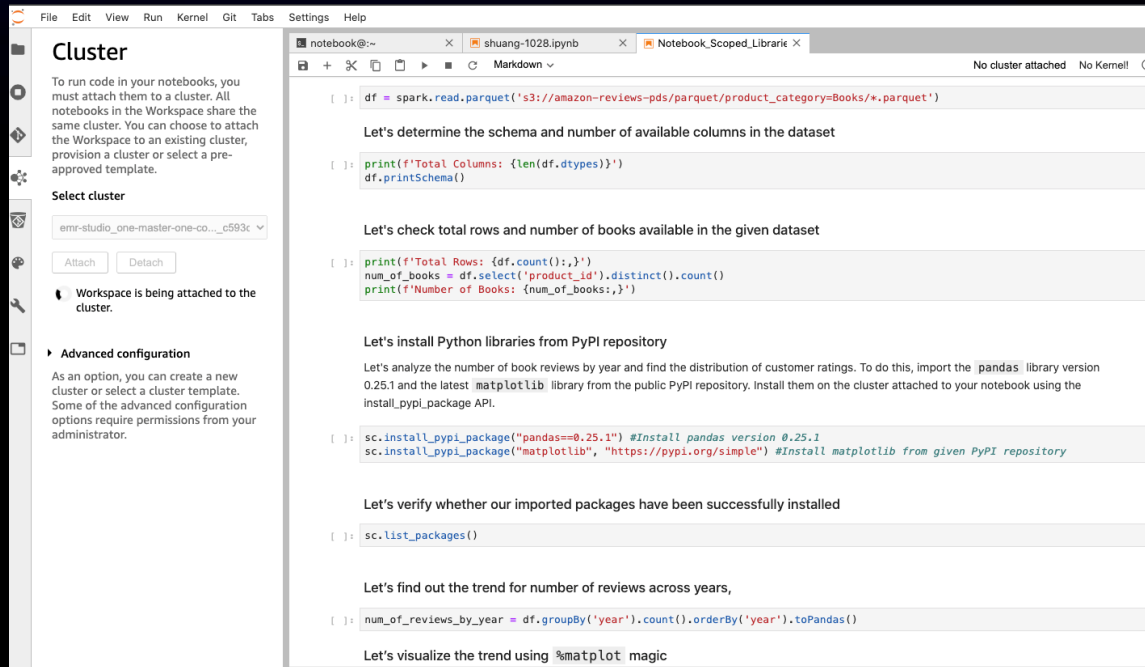
Connect to AWS CodeCommit, GitHub, and Bitbucket

Select existing or add new Git repositories

Data science and engineering workflows



Single-click attach to clusters to run jobs

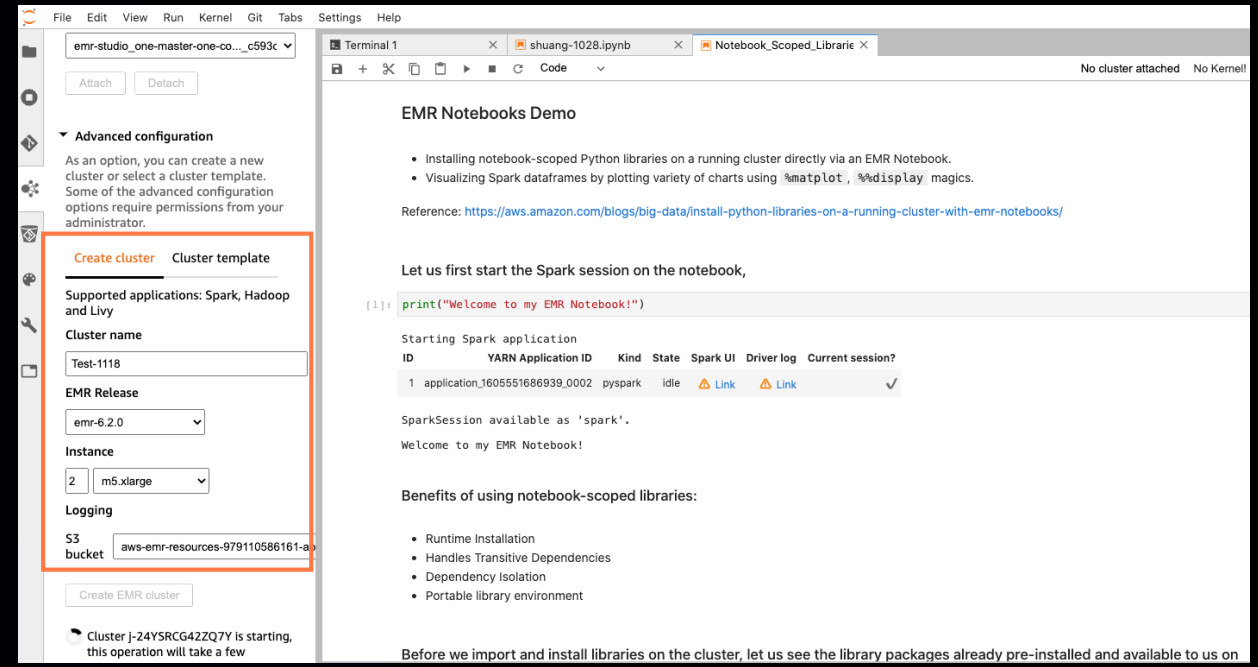


The screenshot shows the AWS EMR Studio interface. On the left, the 'Cluster' sidebar is visible, showing a 'Select cluster' dropdown with 'emr-studio_one-master-one-co..._c593c' selected. Below it are 'Attach' and 'Detach' buttons. The main notebook area displays a series of code cells and text instructions. The code includes:

```
df = spark.read.parquet('s3://amazon-reviews-pds/parquet/product_category=Books/*.parquet')
```

 followed by instructions to determine schema and columns, check total rows and number of books, install Python libraries (pandas, matplotlib), and visualize the trend using %matplotlib magic.

Attach Workspace to an existing Amazon EMR cluster



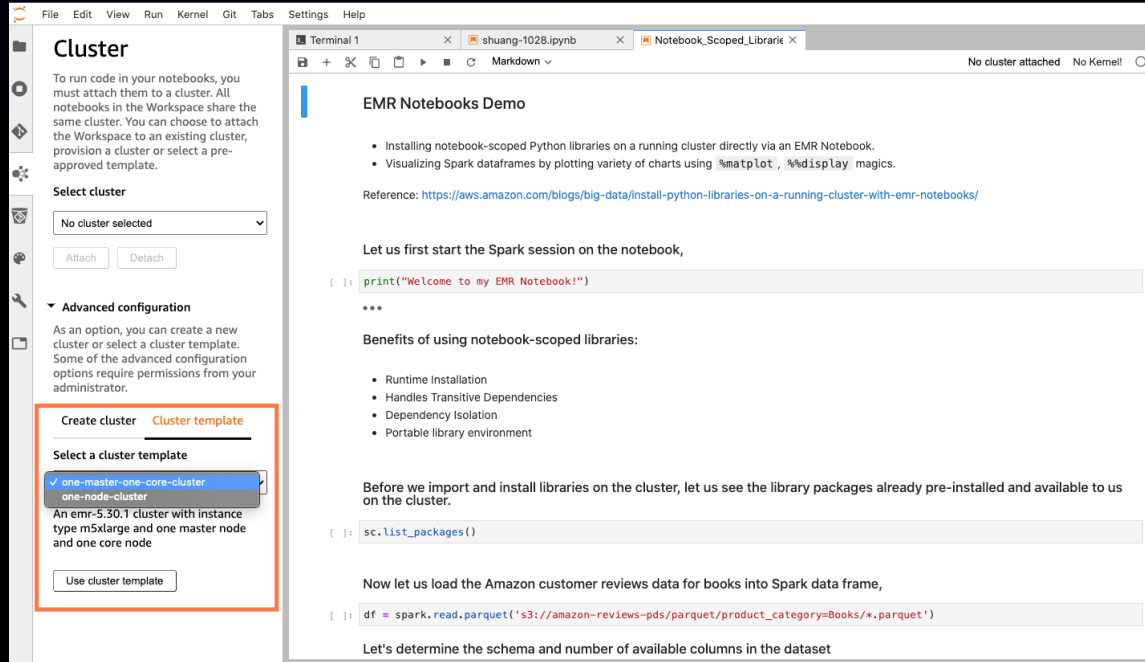
The screenshot shows the AWS EMR Studio interface with the 'Advanced configuration' sidebar. It includes a 'Create cluster' button and a 'Cluster template' dropdown. Below these are fields for 'Supported applications: Spark, Hadoop and Livy', 'Cluster name' (Test-1118), 'EMR Release' (emr-6.2.0), 'Instance' (2 m5.xlarge), and 'Logging' (S3 bucket: aws-emr-resources-979110586161-a). A 'Create EMR cluster' button is at the bottom. The main notebook area shows a code cell with

```
print("Welcome to my EMR Notebook!")
```

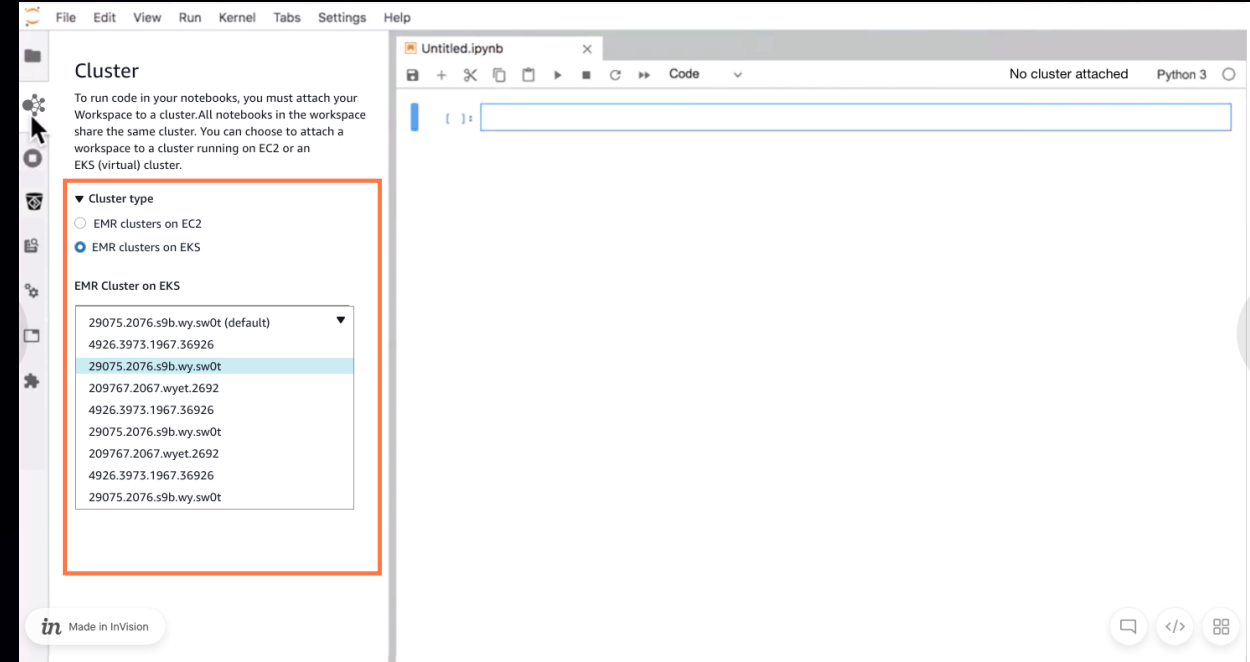
 and a message indicating the Spark session is available as 'spark'.

Provision Amazon EMR clusters using simple configurations
(You can limit users to either cluster templates or creating their own Amazon EMR cluster)

Single-click attach to clusters to run jobs

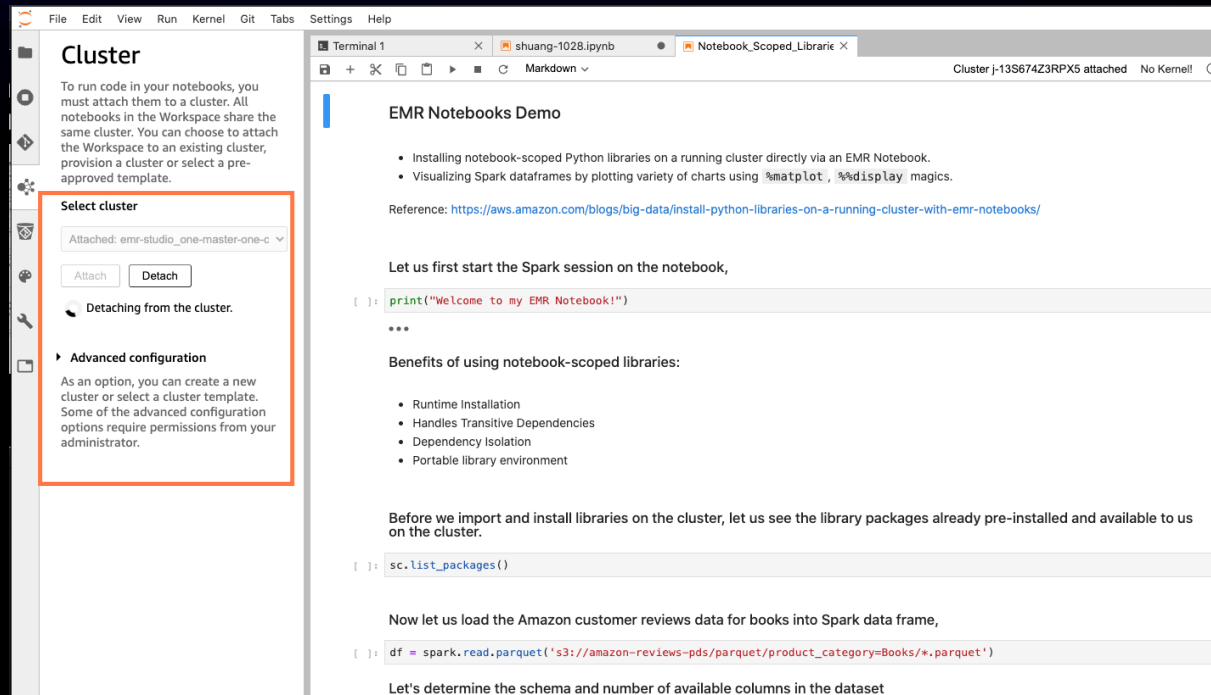


Provision Amazon EMR clusters using preconfigured cluster templates via AWS Service Catalog



Connecting to clusters from Amazon EKS

Detach from and re-attach to clusters



The screenshot shows the AWS EMR Studio interface. On the left, the 'Cluster' sidebar is visible. It contains a 'Select cluster' dropdown menu showing 'Attached: emr-studio_one-master-one-c'. Below this are 'Attach' and 'Detach' buttons. The 'Detach' button is highlighted with an orange box. Below the buttons, there is a section titled 'Advanced configuration' with text about creating a new cluster or selecting a template. The main workspace area shows a terminal window with the command 'print("Welcome to my EMR Notebook!")' and a code editor with a Spark session setup.

Cluster

To run code in your notebooks, you must attach them to a cluster. All notebooks in the Workspace share the same cluster. You can choose to attach the Workspace to an existing cluster, provision a cluster or select a pre-approved template.

Select cluster

Attached: emr-studio_one-master-one-c

Attach Detach

Detaching from the cluster.

Advanced configuration

As an option, you can create a new cluster or select a cluster template. Some of the advanced configuration options require permissions from your administrator.

EMR Notebooks Demo

- Installing notebook-scoped Python libraries on a running cluster directly via an EMR Notebook.
- Visualizing Spark dataframes by plotting variety of charts using `%matplotlib`, `%display` magics.

Reference: <https://aws.amazon.com/blogs/big-data/install-python-libraries-on-a-running-cluster-with-emr-notebooks/>

Let us first start the Spark session on the notebook,

```
[ ]: print("Welcome to my EMR Notebook!")
***
```

Benefits of using notebook-scoped libraries:

- Runtime Installation
- Handles Transitive Dependencies
- Dependency Isolation
- Portable library environment

Before we import and install libraries on the cluster, let us see the library packages already pre-installed and available to us on the cluster.

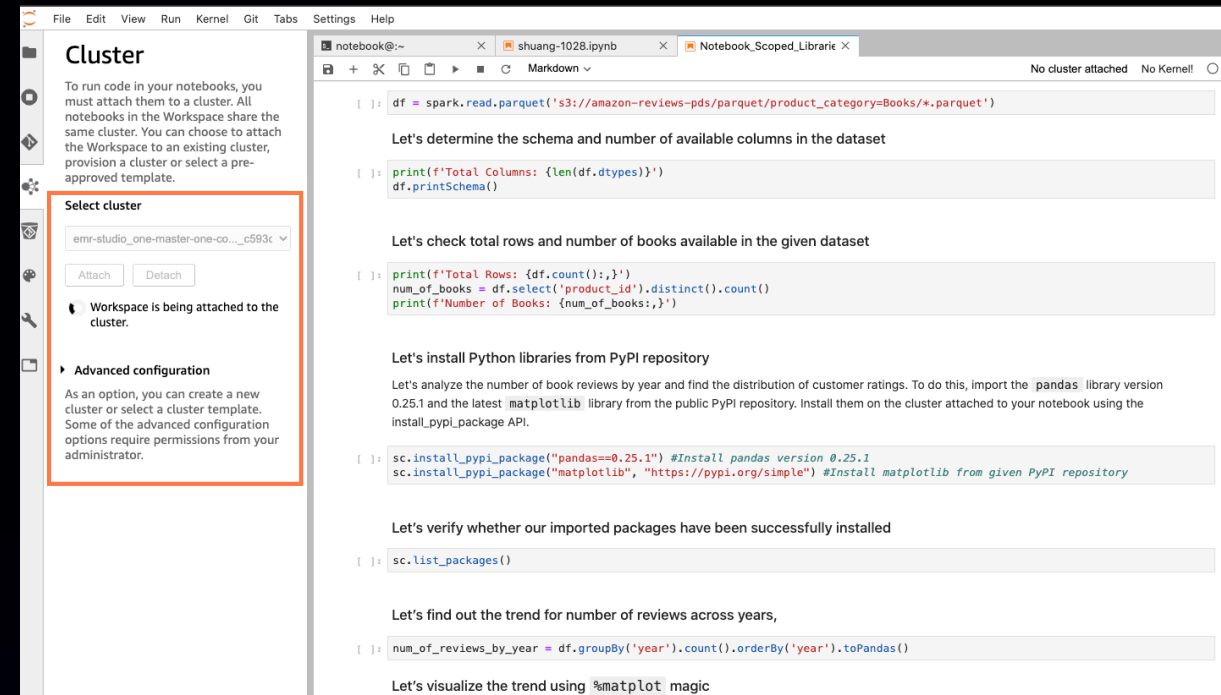
```
[ ]: sc.list_packages()
```

Now let us load the Amazon customer reviews data for books into Spark data frame,

```
[ ]: df = spark.read.parquet('s3://amazon-reviews-pds/parquet/product_category=Books/*.parquet')
```

Let's determine the schema and number of available columns in the dataset

Detach Workspace from a cluster



The screenshot shows the AWS EMR Studio interface. On the left, the 'Cluster' sidebar is visible. It contains a 'Select cluster' dropdown menu showing 'emr-studio_one-master-one-co...c593c'. Below this are 'Attach' and 'Detach' buttons. The 'Attach' button is highlighted with an orange box. Below the buttons, there is a section titled 'Advanced configuration' with text about creating a new cluster or selecting a template. The main workspace area shows a terminal window with the command 'print("Total Columns: {len(df.dtypes)}')" and a code editor with a Spark session setup.

Cluster

To run code in your notebooks, you must attach them to a cluster. All notebooks in the Workspace share the same cluster. You can choose to attach the Workspace to an existing cluster, provision a cluster or select a pre-approved template.

Select cluster

emr-studio_one-master-one-co...c593c

Attach Detach

Workspace is being attached to the cluster.

Advanced configuration

As an option, you can create a new cluster or select a cluster template. Some of the advanced configuration options require permissions from your administrator.

```
[ ]: df = spark.read.parquet('s3://amazon-reviews-pds/parquet/product_category=Books/*.parquet')
```

Let's determine the schema and number of available columns in the dataset

```
[ ]: print(f'Total Columns: {len(df.dtypes)}')
df.printSchema()
```

Let's check total rows and number of books available in the given dataset

```
[ ]: print(f'Total Rows: {df.count():,}')
num_of_books = df.select('product_id').distinct().count()
print(f'Number of Books: {num_of_books:,}')
```

Let's install Python libraries from PyPI repository

Let's analyze the number of book reviews by year and find the distribution of customer ratings. To do this, import the `pandas` library version 0.25.1 and the latest `matplotlib` library from the public PyPI repository. Install them on the cluster attached to your notebook using the `install_pypi_package` API.

```
[ ]: sc.install_pypi_package("pandas==0.25.1") #Install pandas version 0.25.1
sc.install_pypi_package("matplotlib", "https://pypi.org/simple") #Install matplotlib from given PyPI repository
```

Let's verify whether our imported packages have been successfully installed

```
[ ]: sc.list_packages()
```

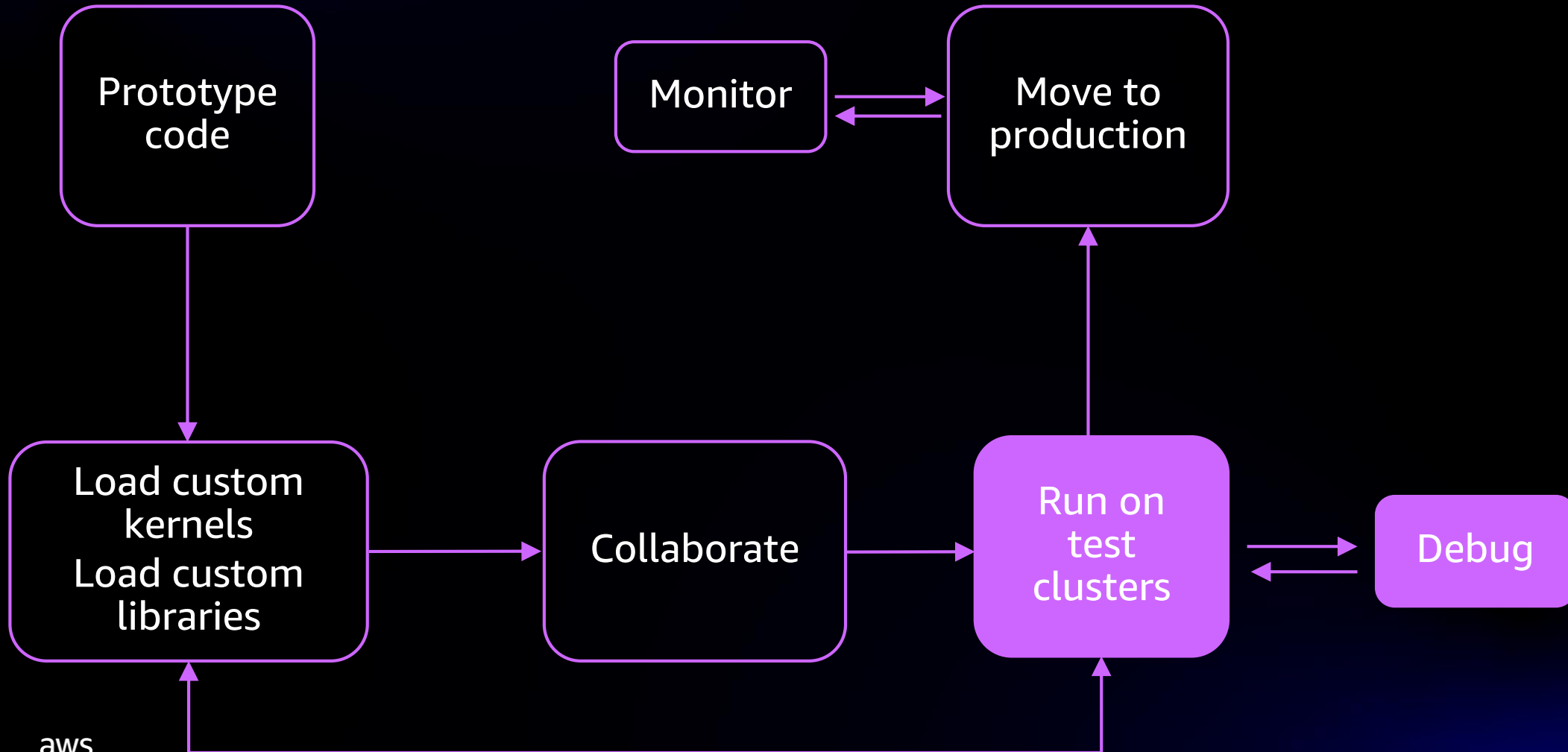
Let's find out the trend for number of reviews across years,

```
[ ]: num_of_reviews_by_year = df.groupBy('year').count().orderBy('year').toPandas()
```

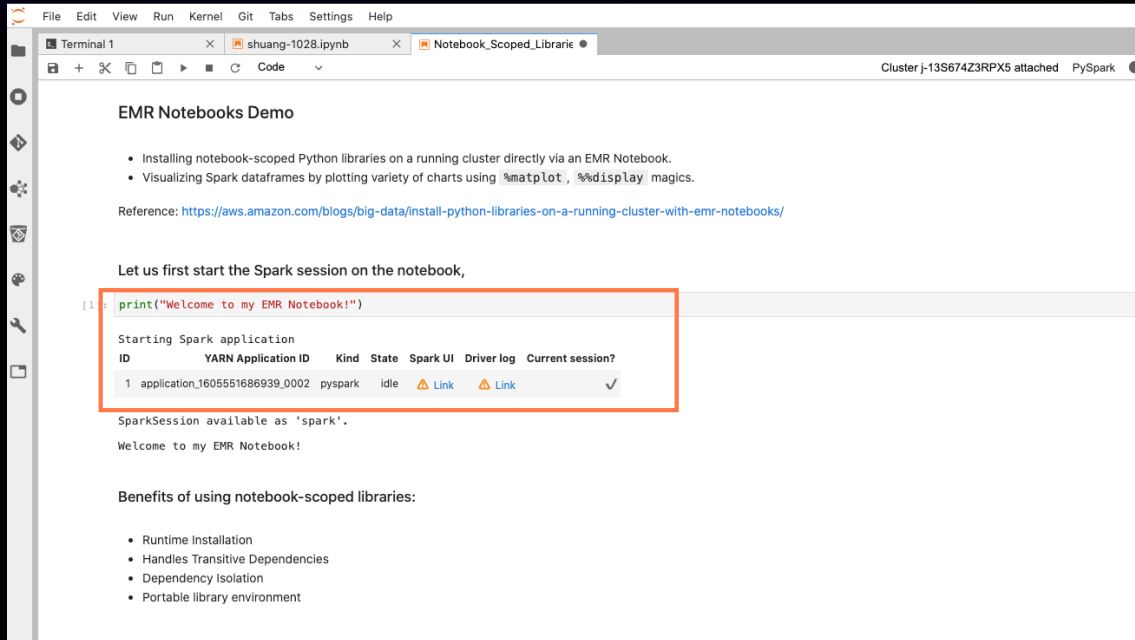
Let's visualize the trend using `%matplotlib` magic

Re-attach Workspace to another cluster

Data science and engineering workflows



Live debugging is simple



The screenshot shows the EMR Notebooks Demo interface. It includes a menu bar (File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help) and a toolbar with icons for file operations and code execution. The main content area is titled "EMR Notebooks Demo" and contains a list of bullet points about installing notebook-scoped Python libraries and visualizing Spark dataframes. A reference link is provided: <https://aws.amazon.com/blogs/big-data/install-python-libraries-on-a-running-cluster-with-emr-notebooks/>. Below this, a text prompt says "Let us first start the Spark session on the notebook," followed by a code cell with the command `print("Welcome to my EMR Notebook!")`. The output of the code cell shows the Spark application starting, with a table of application details. The "Spark UI" link in the table is highlighted with a red box. Below the table, it says "SparkSession available as 'spark'." and "Welcome to my EMR Notebook!". At the bottom, it lists the "Benefits of using notebook-scoped libraries:" including Runtime Installation, Handles Transitive Dependencies, Dependency Isolation, and Portable library environment.

File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 x shuang-1028.jpynb x Notebook_Scoped_Librarie

Cluster j-13S674Z3RPX5 attached PySpark

EMR Notebooks Demo

- Installing notebook-scoped Python libraries on a running cluster directly via an EMR Notebook.
- Visualizing Spark dataframes by plotting variety of charts using `%matplotlib`, `%display` magics.

Reference: <https://aws.amazon.com/blogs/big-data/install-python-libraries-on-a-running-cluster-with-emr-notebooks/>

Let us first start the Spark session on the notebook,

```
[1]: print("Welcome to my EMR Notebook!")
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
1	application_1605551686939_0002	pyspark	idle	Link	Link	✓

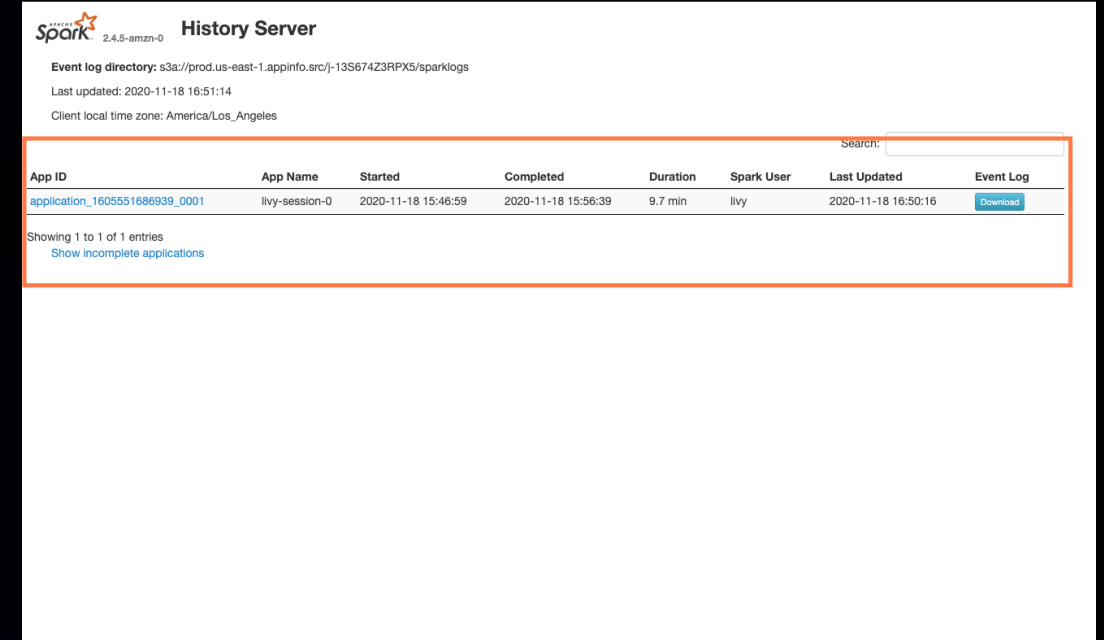
SparkSession available as 'spark'.

Welcome to my EMR Notebook!

Benefits of using notebook-scoped libraries:

- Runtime Installation
- Handles Transitive Dependencies
- Dependency Isolation
- Portable library environment

Debug by selecting the **Spark UI** link in notebook to navigate to the live on-cluster Spark UI



The screenshot shows the Spark History Server interface. It includes the Spark logo and version (2.4.5-amzn-0) and the title "History Server". Below this, it shows the "Event log directory" as `s3a://prod.us-east-1.appinfo.src/j-13S674Z3RPX5/sparklogs`, the "Last updated" time as "2020-11-18 16:51:14", and the "Client local time zone" as "America/Los_Angeles". A search bar is located at the top right. Below the search bar is a table of application details. The table has columns for App ID, App Name, Started, Completed, Duration, Spark User, Last Updated, and Event Log. The first row shows an application with ID `application_1605551686939_0001`, Name `livy-session-0`, Started at `2020-11-18 15:46:59`, Completed at `2020-11-18 15:56:39`, Duration of `9.7 min`, Spark User `livy`, and Last Updated at `2020-11-18 16:50:16`. A "Download" button is next to the Event Log column. Below the table, it says "Showing 1 to 1 of 1 entries" and provides a link to "Show incomplete applications".

Spark 2.4.5-amzn-0 History Server

Event log directory: `s3a://prod.us-east-1.appinfo.src/j-13S674Z3RPX5/sparklogs`

Last updated: 2020-11-18 16:51:14

Client local time zone: America/Los_Angeles

Search:

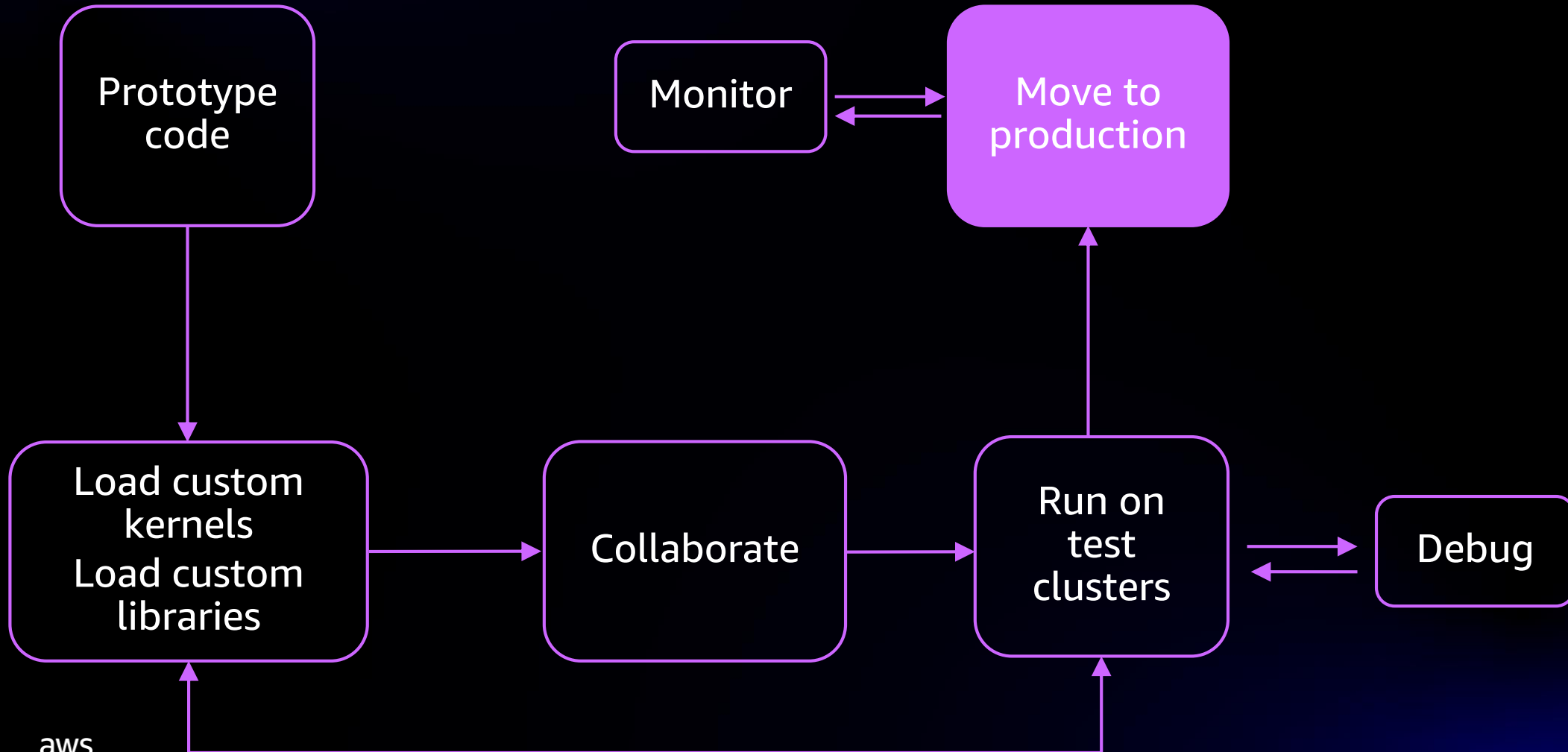
App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1605551686939_0001	livy-session-0	2020-11-18 15:46:59	2020-11-18 15:56:39	9.7 min	livy	2020-11-18 16:50:16	Download

Showing 1 to 1 of 1 entries

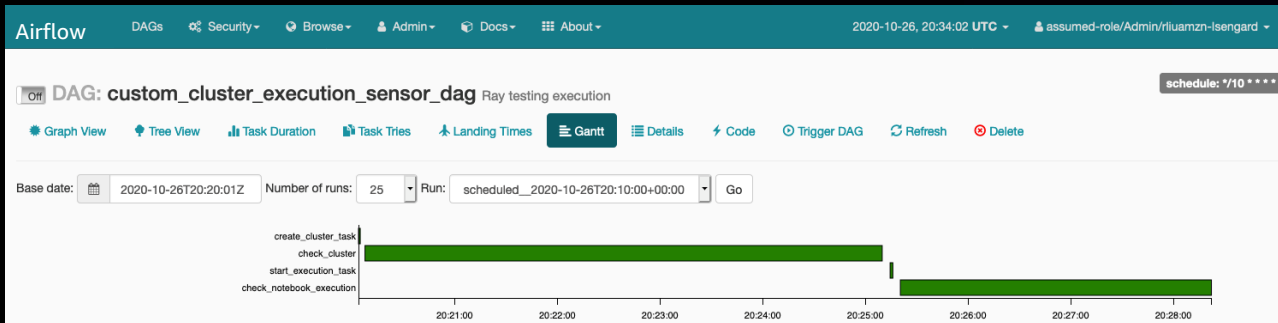
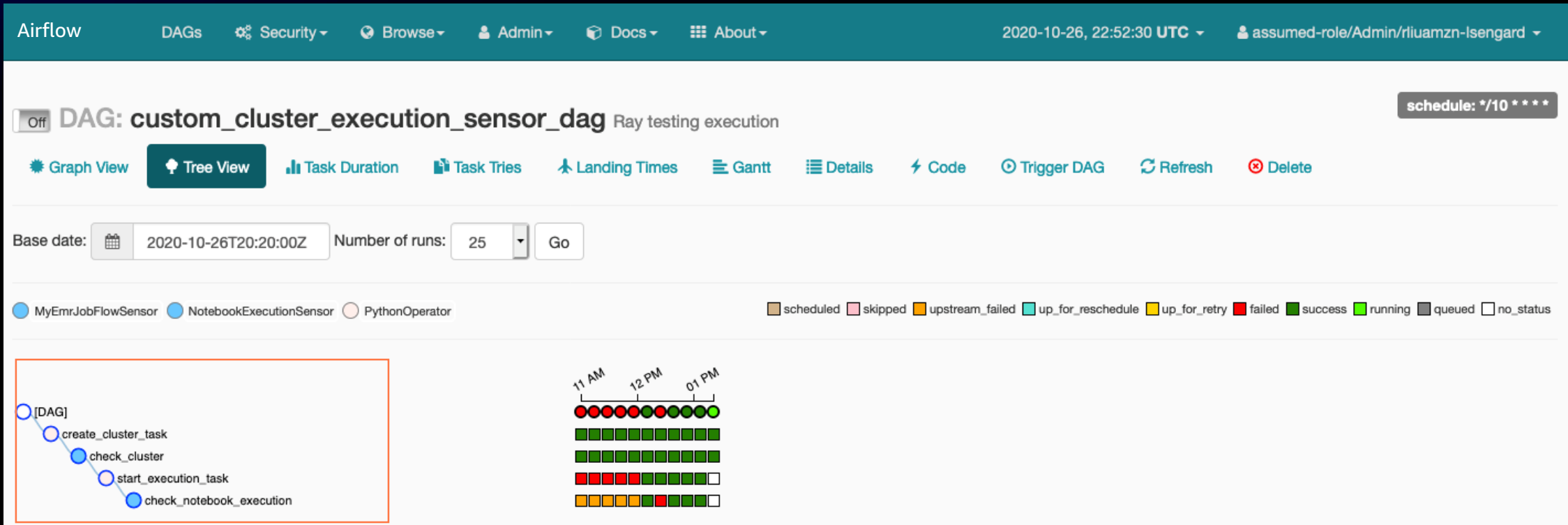
[Show incomplete applications](#)

View debugging information in Spark **History Server** for the application in a separate browser tab

Data science and engineering workflows



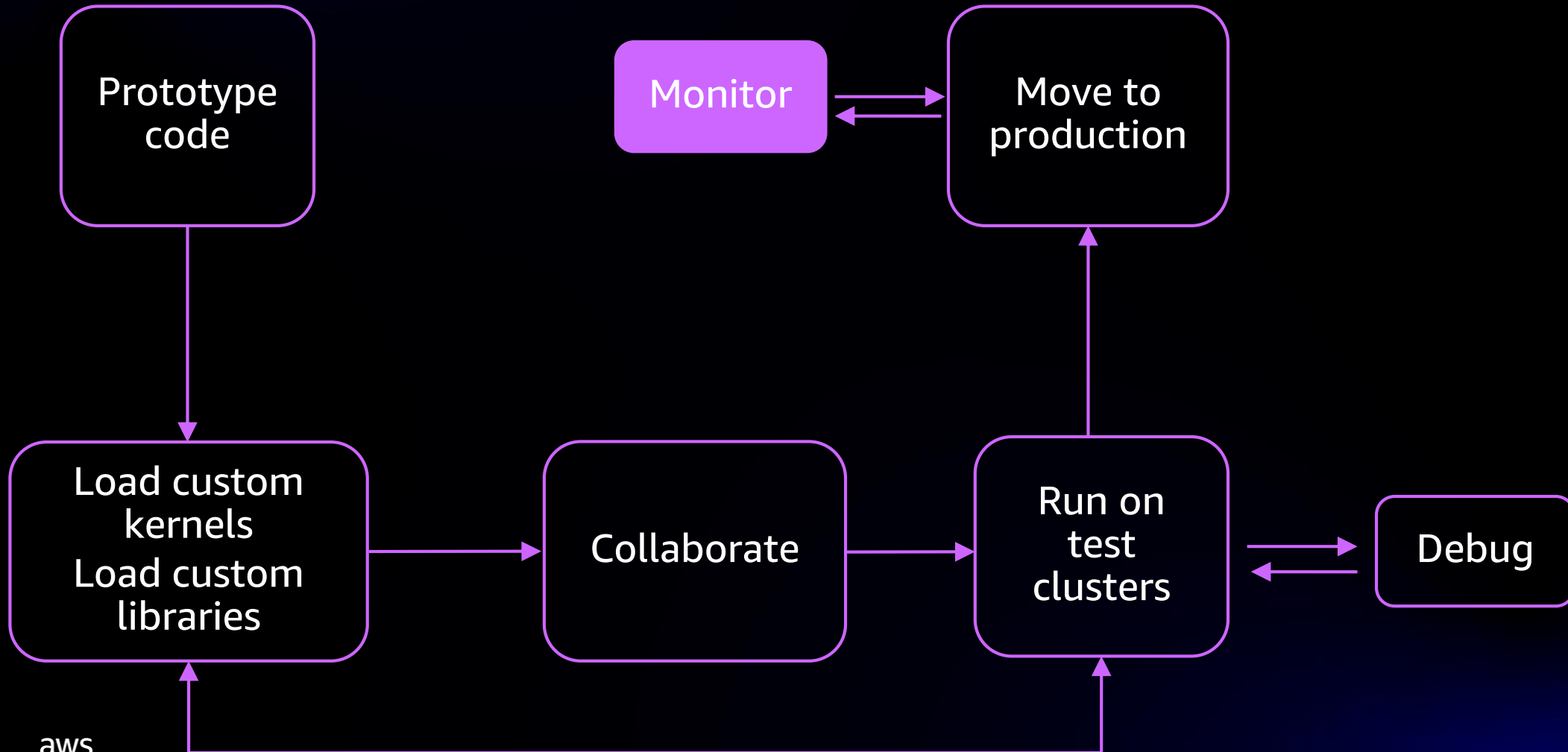
Simplify building pipelines from notebooks



Run notebooks as pipelines via Amazon Managed Workflows for Apache Airflow (MWAA)

Parameterize and chain notebooks that can be run as pipelines

Data science and engineering workflows



Monitoring production pipelines is easy

EMR Studio > Clusters

Clusters

Debug applications by searching for a cluster and choosing an application UI. Filter clusters by state, ID, and time range. Once located, select the cluster and then choose "Launch application UIs" to start debugging your application.

Clusters (47)

Filter by state or search cluster ID

Earliest time: YYYY/MM/DD 00:00:00 Latest time: YYYY/MM/DD 23:59:59

	Cluster ID	Cluster name	Start time	Elapsed time	State	Status
<input type="radio"/>	j-24YSRCG42ZQ7Y	Test-1118	2020-11-18 17:02:...	1 hour	Active	Waiting Clu...
<input type="radio"/>	j-1XQGN1TKK44Q	emr-studio_one-master...	2020-11-18 14:51:...	4 hours	Active	Waiting Clu...
<input type="radio"/>	j-13S674Z3RPX5	emr-studio_one-master...	2020-11-16 10:29:...	2 days	Active	Waiting Clu...
<input type="radio"/>	j-4I9Z8N9GIUS	test	2020-11-16 10:13:...	2 days	Active	Waiting Clu...
<input type="radio"/>	j-UE40BTPHCXCA	emr-studio_one-master...	2020-11-15 20:21:...	13 hours	Termin...	Terminated ...
<input type="radio"/>	j-3U2U52Y0U1PXX	test-5.30.0	2020-11-10 12:12:...	1 week, 1 day	Active	Waiting Clu...
<input type="radio"/>	j-C5ARB4K0UGB3	DO NOT TERMINATE Pr...	2020-11-05 16:11:...	1 week, 3 days	Termin...	Terminated ...
<input type="radio"/>	j-3LXFPT2IS9HUB	DO NOT TERMINATE Pr...	2020-11-05 16:05:...	1 week, 3 days	Termin...	Terminated ...
<input type="radio"/>	j-2D1TUBZCGBIGG	DO NOT TERMINATE Pr...	2020-11-04 15:48:...	1 week, 4 days	Termin...	Terminated ...
<input type="radio"/>	j-2PLQ7PSZ1KD1A	DO NOT TERMINATE Pr...	2020-11-04 15:34:...	1 week, 4 days	Termin...	Terminated ...
<input type="radio"/>	j-2Y5ODBYI0HDGN	DO NOT TERMINATE Pr...	2020-11-04 15:33:...	1 week, 4 days	Termin...	Terminated ...
<input type="radio"/>	j-30KPHCFAB09N	DO NOT TERMINATE Pr...	2020-11-04 15:33:...	1 week, 4 days	Termin...	Terminated ...

Browse all clusters in one place

EMR Studio > Clusters

Clusters

Debug applications by searching for a cluster and choosing an application UI. Filter clusters by state, ID, and time range. Once located, select the cluster and then choose "Launch application UIs" to start debugging your application.

Clusters (47)

State: |

State values

- State: Active
- State: Terminated
- State: Failed

Filter by state or search cluster ID

Earliest time: YYYY/MM/DD 00:00:00 Latest time: YYYY/MM/DD 23:59:59

	Cluster ID	Cluster name	Start time	Elapsed time	State	Status
<input type="radio"/>	j-1XQGN1TKK44Q	emr-studio_one-master...	2020-11-18 14:51:...	4 hours	Active	Waiting Clu...
<input type="radio"/>	j-13S674Z3RPX5	emr-studio_one-master...	2020-11-16 10:29:...	2 days	Active	Waiting Clu...
<input type="radio"/>	j-4I9Z8N9GIUS	test	2020-11-16 10:13:...	2 days	Active	Waiting Clu...
<input type="radio"/>	j-UE40BTPHCXCA	emr-studio_one-master...	2020-11-15 20:21:...	13 hours	Termin...	Terminated ...
<input type="radio"/>	j-3U2U52Y0U1PXX	test-5.30.0	2020-11-10 12:12:...	1 week, 1 day	Active	Waiting Clu...
<input type="radio"/>	j-C5ARB4K0UGB3	DO NOT TERMINATE Pr...	2020-11-05 16:11:...	1 week, 3 days	Termin...	Terminated ...
<input type="radio"/>	j-3LXFPT2IS9HUB	DO NOT TERMINATE Pr...	2020-11-05 16:05:...	1 week, 3 days	Termin...	Terminated ...
<input type="radio"/>	j-2D1TUBZCGBIGG	DO NOT TERMINATE Pr...	2020-11-04 15:48:...	1 week, 4 days	Termin...	Terminated ...
<input type="radio"/>	j-2PLQ7PSZ1KD1A	DO NOT TERMINATE Pr...	2020-11-04 15:34:...	1 week, 4 days	Termin...	Terminated ...
<input type="radio"/>	j-2Y5ODBYI0HDGN	DO NOT TERMINATE Pr...	2020-11-04 15:33:...	1 week, 4 days	Termin...	Terminated ...
<input type="radio"/>	j-30KPHCFAB09N	DO NOT TERMINATE Pr...	2020-11-04 15:33:...	1 week, 4 days	Termin...	Terminated ...

Narrow down clusters for investigation using filters such as cluster state

Monitoring production pipelines is easy

EMR Studio > Clusters

Clusters

Debug applications by searching for a cluster and choosing an application UI. Filter clusters by state, ID, and time range. Once located, select the cluster and then choose "Launch application UIs" to start debugging your application.

Clusters (47)

Filter by state or search cluster ID

Earliest time: YYYY/MM/DD 00:00:00 Latest time: YYYY/MM/DD

Launch application UIs

- Spark History Server
- YARN Timeline Server
- Tez UI

Cluster ID	Cluster name	Start time	Elapsed time	State	Status
J-24Y5RCG42ZQ7Y	Test-1118	2020-11-18 17:02:...	1 hour	Active	Waiting Clu...
J-1XQGNN1TKK44Q	emr-studio_one-master...	2020-11-18 14:51:...	4 hours	Active	Waiting Clu...
J-13S674Z3RPX5	emr-studio_one-master...	2020-11-16 10:29:...	2 days	Active	Waiting Clu...
J-4I9Z8N9JGIUS	test	2020-11-16 10:13:...	2 days	Active	Waiting Clu...
J-UE40BTPHCXA	emr-studio_one-master...	2020-11-15 20:21:...	13 hours	Termin...	Terminated J...
J-3U2U52Y0U1PXX	test-5.30.0	2020-11-10 12:12:...	1 week, 1 day	Active	Waiting Clu...
J-C5ARB4K0UGB3	DO NOT TERMINATE Pr...	2020-11-05 16:11:...	1 week, 3 days	Termin...	Terminated J...
J-3LXFPT2I59HUB	DO NOT TERMINATE Pr...	2020-11-05 16:05:...	1 week, 3 days	Termin...	Terminated J...
J-2D1TUBZCGBIGG	DO NOT TERMINATE Pr...	2020-11-04 15:48:...	1 week, 4 days	Termin...	Terminated J...
J-2PLQ7PS21KD1A	DO NOT TERMINATE Pr...	2020-11-04 15:34:...	1 week, 4 days	Termin...	Terminated J...
J-2Y5ODBYI0HDGN	DO NOT TERMINATE Pr...	2020-11-04 15:33:...	1 week, 4 days	Termin...	Terminated J...
J-30KPHCPFAB09N	DO NOT TERMINATE Pr...	2020-11-04 15:33:...	1 week, 4 days	Termin...	Terminated J...

Scheduling Mode: FIFO
Completed Jobs: 16

Event Timeline

Completed Jobs (16)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
15 (13)	Job group for statement 13 runJob at PythonRDD.scala:153	2020/10/02 04:49:59	1 s	1/1 (2 skipped)	4/4 (282 skipped)
14 (13)	Job group for statement 13 runJob at PythonRDD.scala:153	2020/10/02 04:49:57	2 s	2/2 (1 skipped)	201/201 (82 skipped)
13 (13)	Job group for statement 13 toJavaRDD at NativeMethodAccessorImpl.java:0	2020/10/02 04:49:41	16 s	2/2	282/282
12 (11)	Job group for statement 11 uninstall_package at <stdin>:1	2020/10/02 04:49:38	2 s	1/1	2/2
11 (8)	Job group for statement 8 toPandas at <stdin>:1	2020/10/02 04:49:35	0.2 s	1/1 (2 skipped)	21/21 (108 skipped)
10 (8)	Job group for statement 8 toPandas at <stdin>:1	2020/10/02 04:49:34	0.3 s	1/1 (1 skipped)	26/26 (82 skipped)
9 (8)	Job group for statement 8 toPandas at <stdin>:1	2020/10/02 04:49:34	0.3 s	1/1 (1 skipped)	26/26 (82 skipped)
8 (8)	Job group for statement 8 toPandas at <stdin>:1	2020/10/02 04:49:19	15 s	1/1	82/82
7 (6)	Job group for statement 6 install_pypi_package at <stdin>:2	2020/10/02 04:49:12	4 s	1/1	2/2
6 (6)	Job group for statement 6 install_pypi_package at <stdin>:1	2020/10/02 04:49:01	8 s	1/1	2/2
5 (6)	Job group for statement 5 count at NativeMethodAccessorImpl.java:0	2020/10/02 04:48:55	59 ms	1/1 (2 skipped)	1/1 (108 skipped)
4 (6)	Job group for statement 5 count at NativeMethodAccessorImpl.java:0	2020/10/02 04:48:51	3 s	1/1 (1 skipped)	26/26 (82 skipped)
3 (6)	Job group for statement 5 count at NativeMethodAccessorImpl.java:0	2020/10/02 04:48:23	29 s	1/1	82/82
2 (6)	Job group for statement 5 count at NativeMethodAccessorImpl.java:0	2020/10/02 04:48:22	0.2 s	1/1 (1 skipped)	1/1 (82 skipped)
1 (6)	Job group for statement 5 count at NativeMethodAccessorImpl.java:0	2020/10/02 04:48:13	9 s	1/1	82/82
0 (0)	Job group for statement 3	2020/10/02 04:48:08	2 s	1/1	1/1

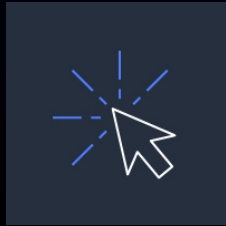
Diagnose jobs on both active and terminated clusters using Spark UI, Tez UI, and Yarn timeline service

Overlay execution context on jobs, even for terminated clusters and jobs

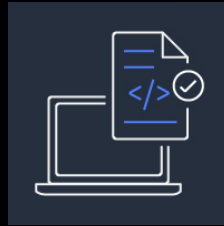
Amazon EMR Studio features: 2021

NEW!

FULLY MANAGED IDE FOR INTERACTIVE DATA ANALYTICS: DEVELOP, VISUALIZE, AND DEBUG APPLICATIONS



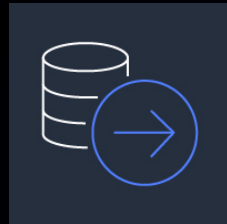
IAM authentication and
federation support



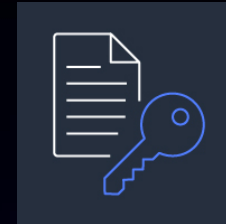
Multi-language support
(R, PySpark, Scala, SQL)



Auto-terminate idle
clusters



Mount Workspace
directories to
Amazon EMR clusters



Fine-grained access
control using AWS Lake
Formation (preview)



“ Choosing EMR Studio as our official workflow for Jupyter notebooks on EMR has enabled us to reduce costs and time spent supporting data users. The built-in Git-based workflow has streamlined our previously cluttered landscape of notebooks. Connecting to an EMR cluster is as simple as selecting it in a dropdown box, avoiding the need to have personal clusters running 24/7.”

Phil Austin

Director of DevOps

Verana Health



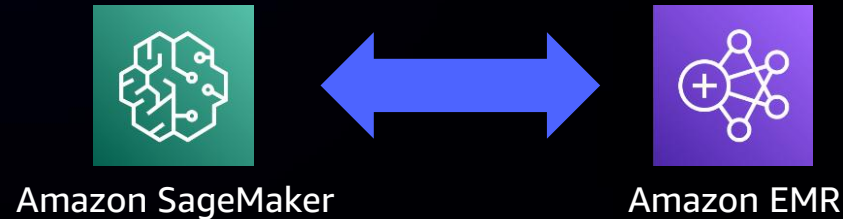


“EMR Studio allows us to prototype Spark applications and data science models that power large-scale data processing and transformations. The integrated development environment makes it easy for data scientists and engineers to perform ad hoc analysis and debug data processing workloads.”

Saba El-Hilo
Head of Data Platform
Mapbox

Deep integration with Amazon SageMaker

RUN AMAZON EMR JOBS FROM AMAZON SAGEMAKER STUDIO FOR EASY, INTERACTIVE DATA ANALYSIS OR PRE-PROCESSING



- ✓ Process petabyte-scale datasets easily using Amazon EMR from Amazon SageMaker Studio
- ✓ Use Amazon EMR native integration with Amazon EC2 Spot Instances and Graviton instances to run large-scale data processing at lower costs

Deep integration with SageMaker

RUN AMAZON EMR JOBS FROM AMAZON SAGEMAKER STUDIO FOR EASY, INTERACTIVE DATA ANALYSIS OR PRE-PROCESSING



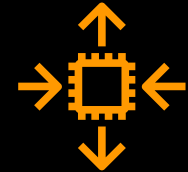
Discover and connect
to an Amazon EMR
cluster from
SageMaker Studio



Run Apache Spark, Hive,
and Presto jobs on
Amazon EMR from
SageMaker Studio



Use familiar
debugging tools
such as Spark UI



Create, scale and auto-
terminate Amazon EMR
clusters using AWS Service
Catalog templates

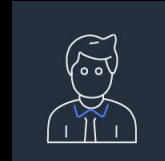
The data engineer



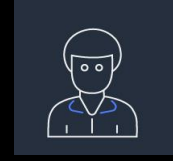
Maria – the data scientist



Ana – the data analyst



Richard – **the data engineer**

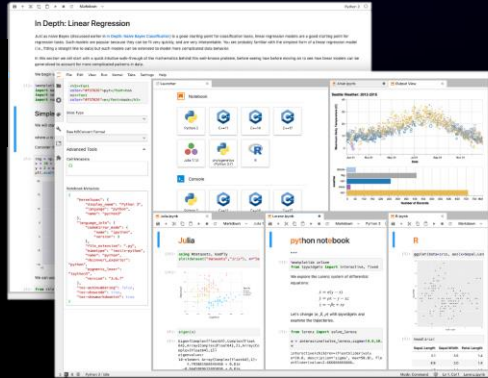


Carlos – the administrator

Builds and
deploys data
pipelines

Pipeline orchestration options

Data engineering **platforms** are evolving



Notebooks

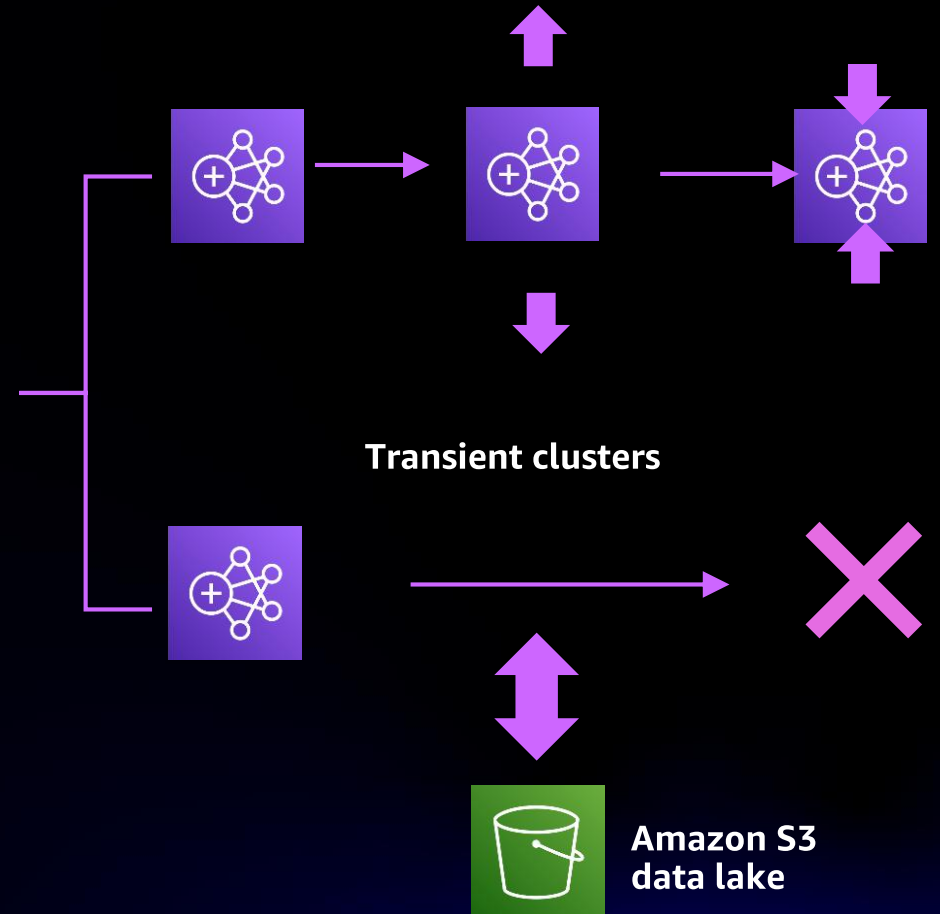


Apache
Airflow



AWS Step Functions

Elastic persistent clusters



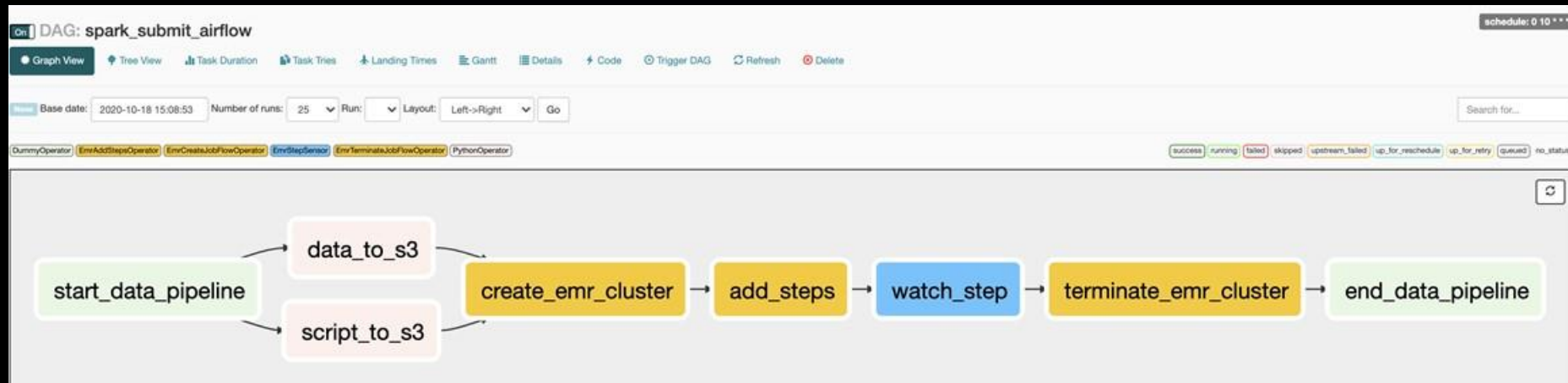
Amazon Managed Workflows for Apache Airflow (MWAA)



Amazon MWAA



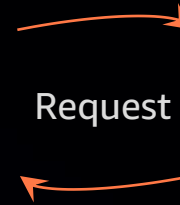
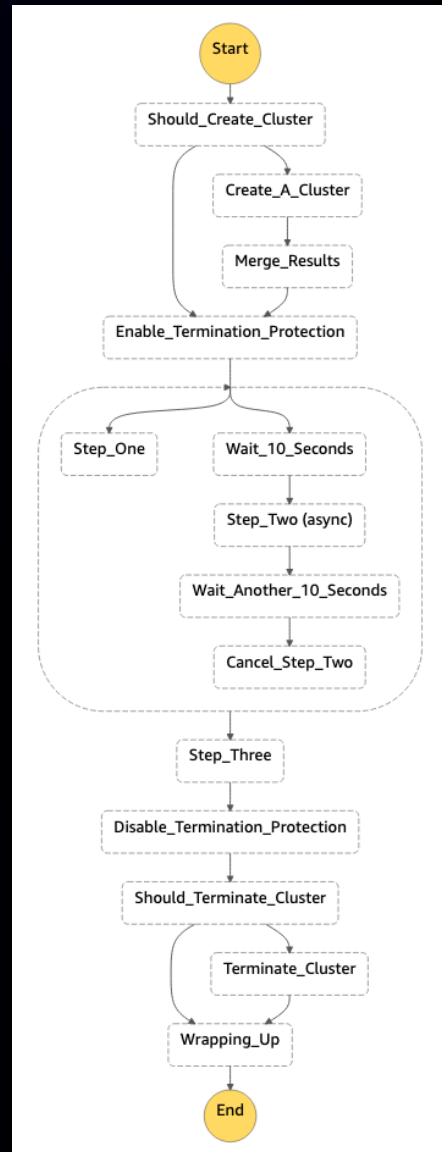
Amazon EMR



AWS Step Functions



AWS Step Functions



Amazon EMR

Step Functions
submits the jobs
and pauses until
the task completes

Performance-optimized

Differentiated Spark runtime performance

Over **3x** faster than standard Apache Spark 3.0 in derived TPC-DS 3 TB benchmark

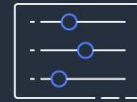
Additional **11%** performance improvement with AWS Graviton2 at **20%+** reduced cost

100% compliance with open-source APIs makes moving applications to Amazon EMR easy

Performance improvements are enabled by default

NEW!

Dynamically sized executors



NEW!

Adaptive join selection



NEW!

Dynamic pruning of data columns



NEW!

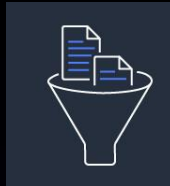
Operator optimization



Early worker allocation



Intelligent filtering



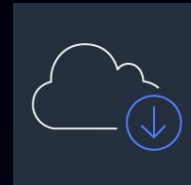
Parallel/async initialization



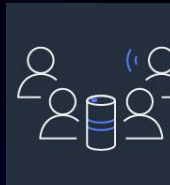
Redundant scan elimination



Data prefetch



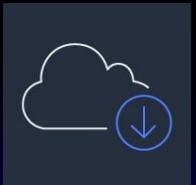
Broadcast join w/o statistics



Stats inference

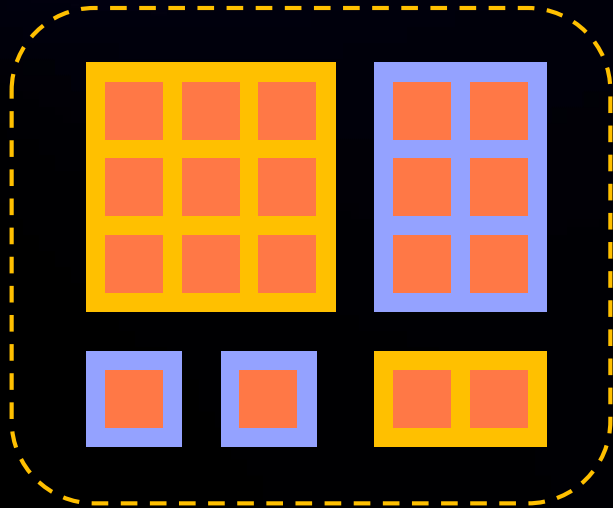


Optimized metadata fetch



Right-sizing executors is hard

A STATIC, ONE-SIZE-FITS-ALL EXECUTOR SIZE IS SUBOPTIMAL

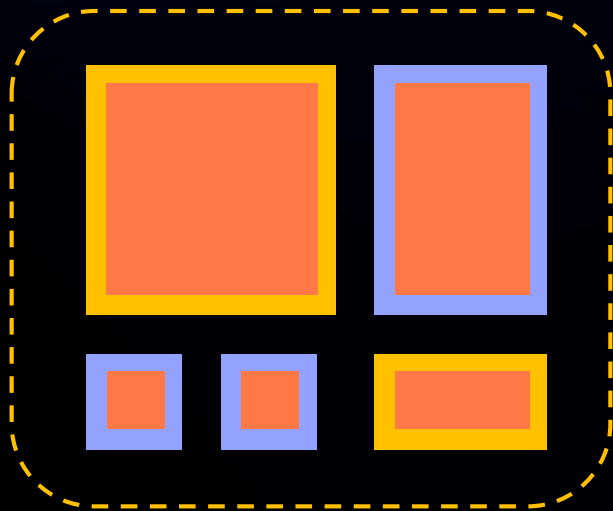


```
spark-submit my_job.py
-num-executors 21
--conf spark.executor.cores=3
--conf spark.executor.memory 12g
```

Challenges

- Need to specify the executor size to fit the smallest instance
- Redundant inter-process communication
- More overhead
- Wasted capacity on heterogenous infrastructure

Dynamically sized executors make it easy



`spark-submit my_job.py`

`spark.yarn.heterogeneousExecutors.enabled = true`

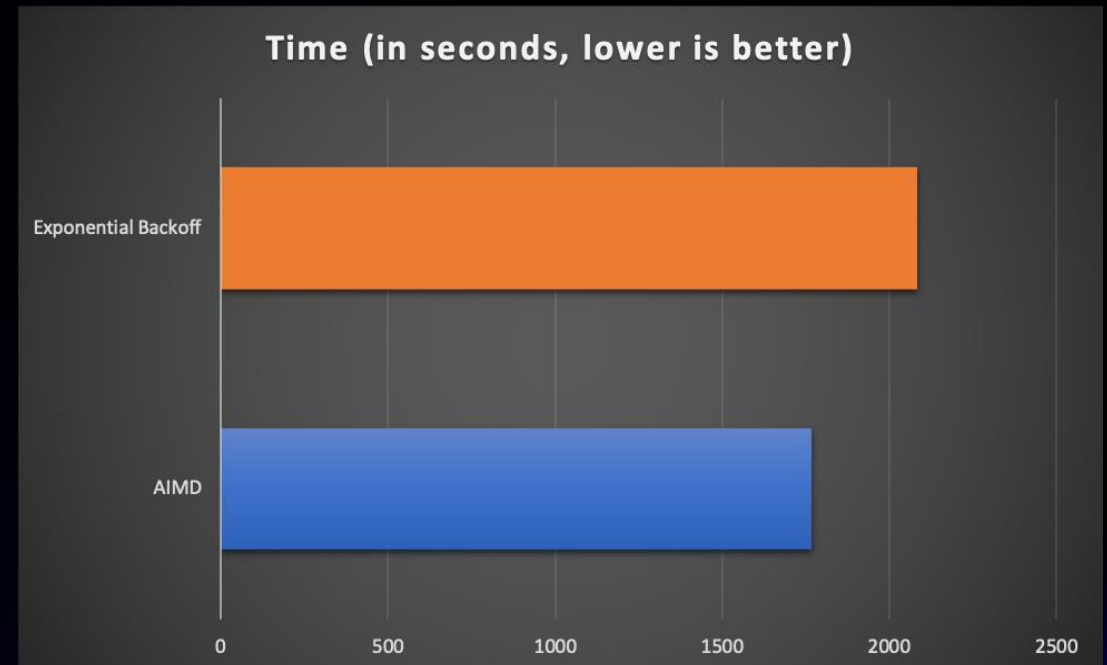
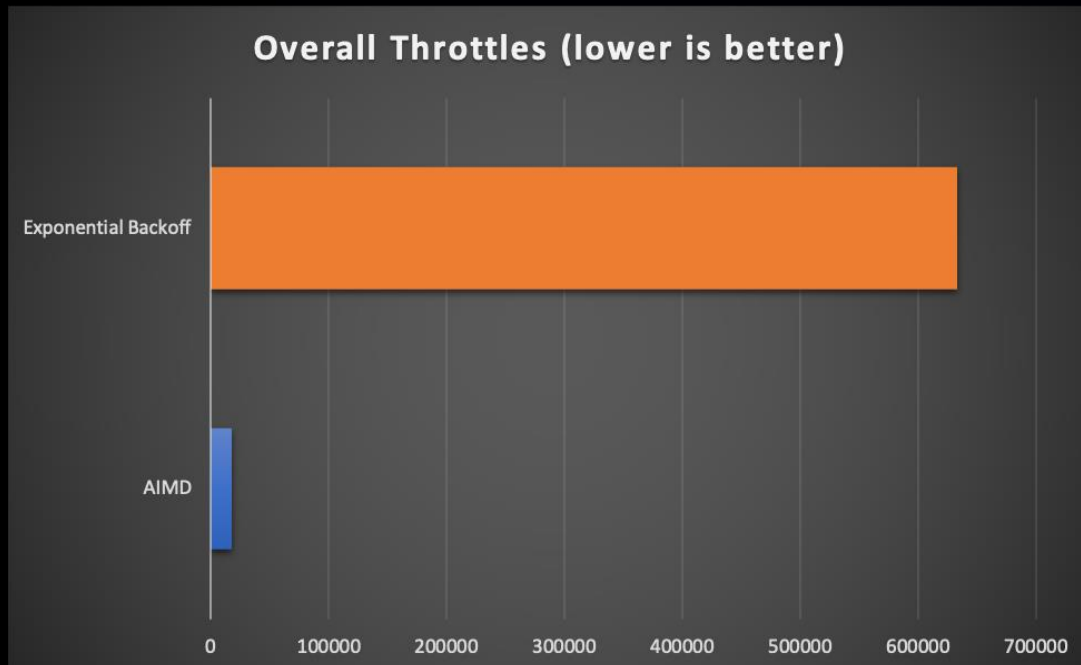
- Amazon EMR dynamically sizes the executor for the available capacity on each instance
- TPC-DS benchmark shows
 - **10% faster on total time**
 - **30% faster for geometric mean**
- **One fewer decision to make**

Optimize writes to S3 at scale using AIMD

NEW!

- Alternative retry strategy – additive-increase/multiplicative-decrease (AIMD) strategy
- Up to 90% reduction in S3 throttles

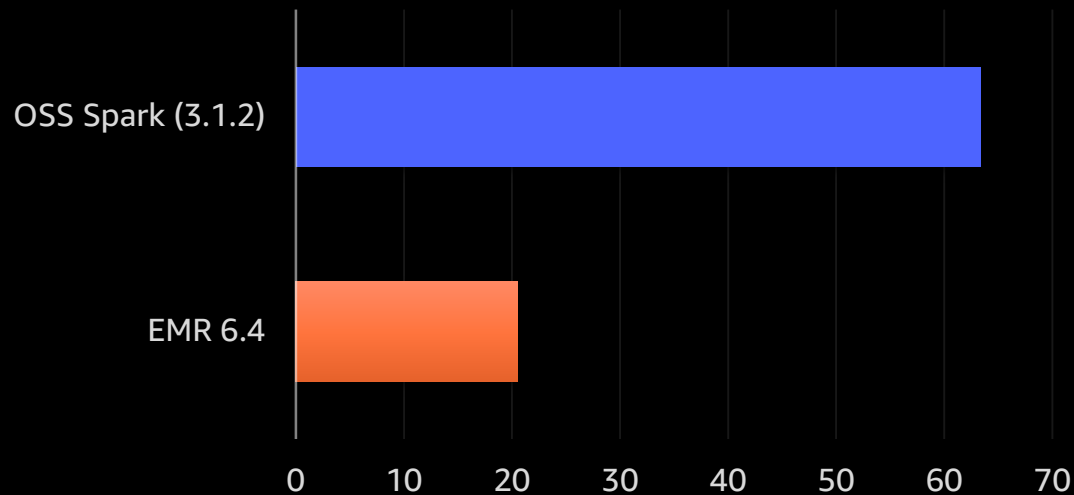
```
[{"Classification": "emrfs-site", "Properties": {"fs.s3.aimd.enabled": "true"}}]
```



Amazon EMR Spark Runtime vs. OSS Spark

3.0X FASTER PERFORMANCE FOR APACHE SPARK 3.0 AT 40% OF THE COST

Geometric mean of runtime in
seconds (lower is better)



Spark 3.1.2 on EMR 6.4.0

**Based on TPC-DS 3TB Benchmarking running 6 node
C5.9XL cluster and EMR 6.4.0 running Spark 3.0*

EMR's performance-optimized Apache Spark runtime

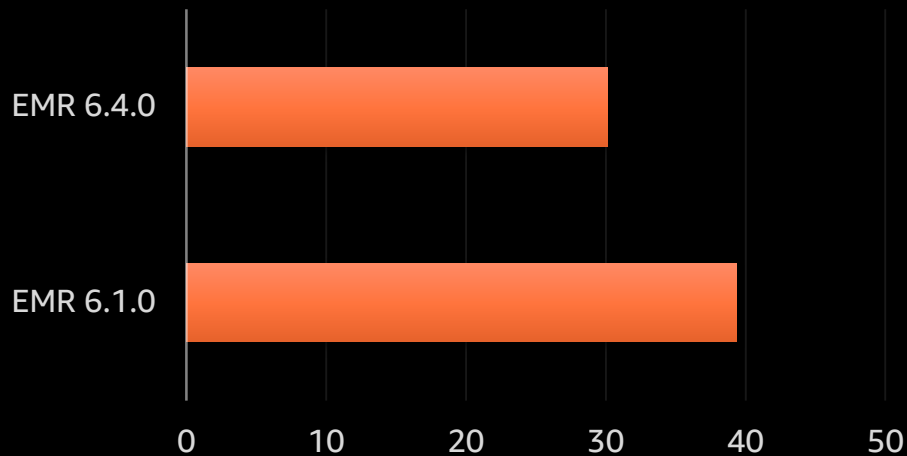
Best performance

- **3.0x faster on geometric mean**
- **2.7x faster for total time**

Amazon EMR Runtime for Apache Spark: Performance improvements - 2021

NEW!

Geometric mean of runtime in seconds (lower is better)



Spark 3.1.2 on EMR 6.4.0

**Based on TPC-DS 3TB Benchmarking running 6 node C5.9XL cluster and EMR 6.5.0 running Spark 3.0*

EMR's performance-optimized Apache Spark runtime

Best performance

- **1.3X** faster on geometric mean in 2021
- Up to **4.8x** faster for individual queries

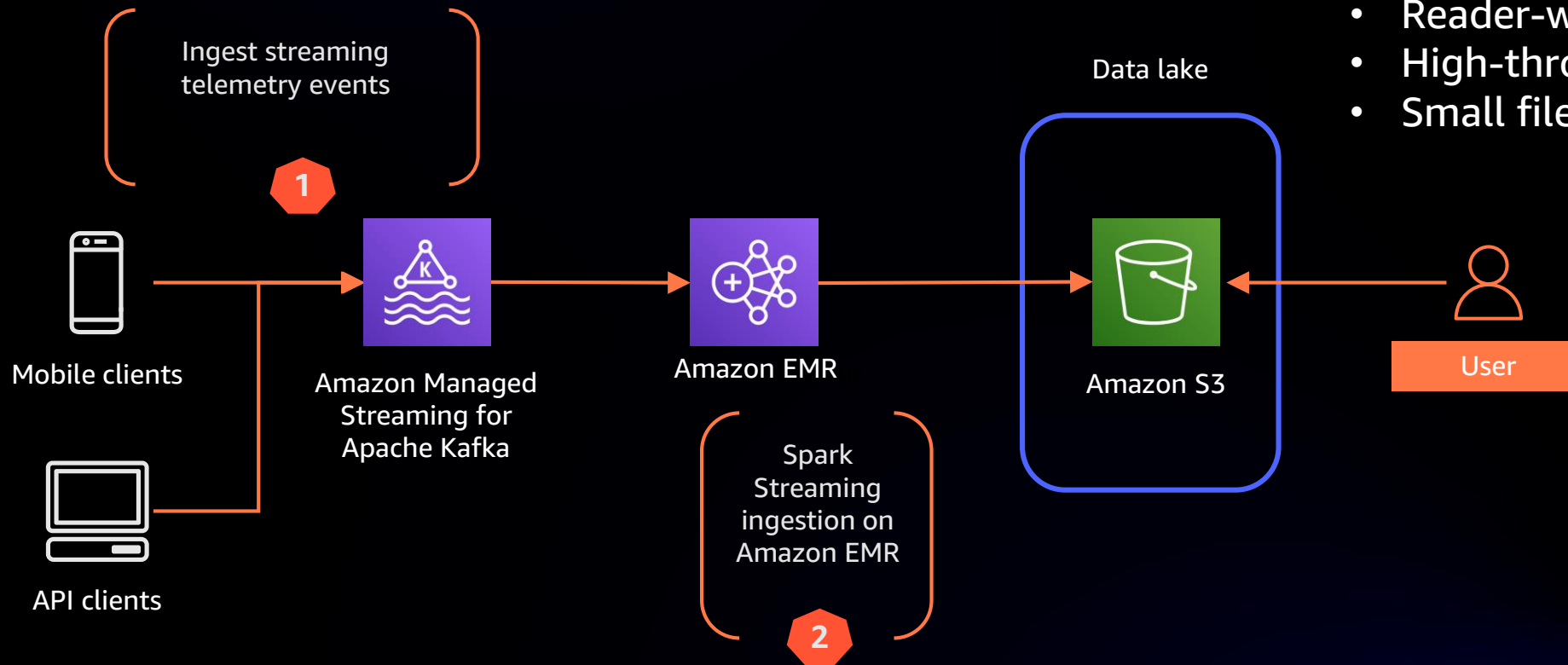
100% compliant with Apache Spark APIs

Transactional data lakes: Apache Hudi

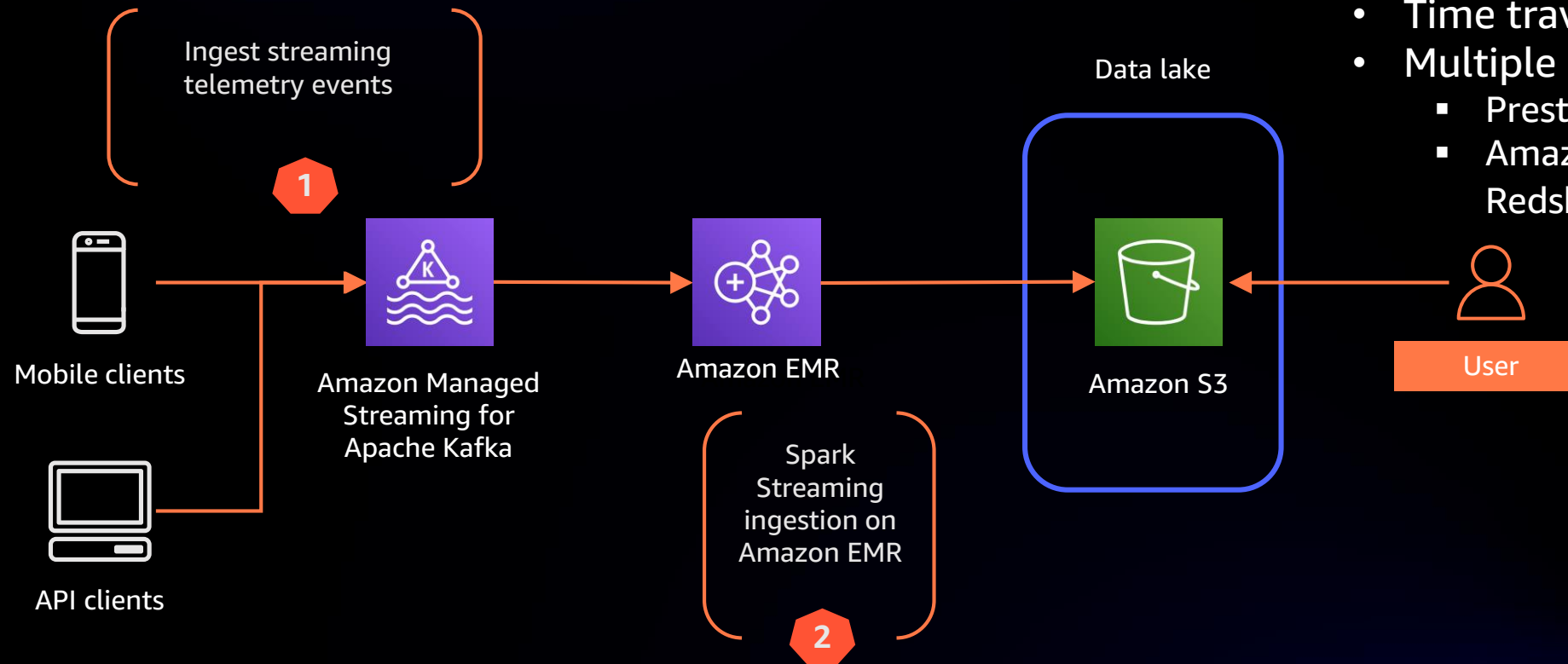
Streaming ingestion pipeline challenges

Challenges

- Make atomic changes
- Reader-writer isolation
- High-throughput ingestion
- Small file compactions



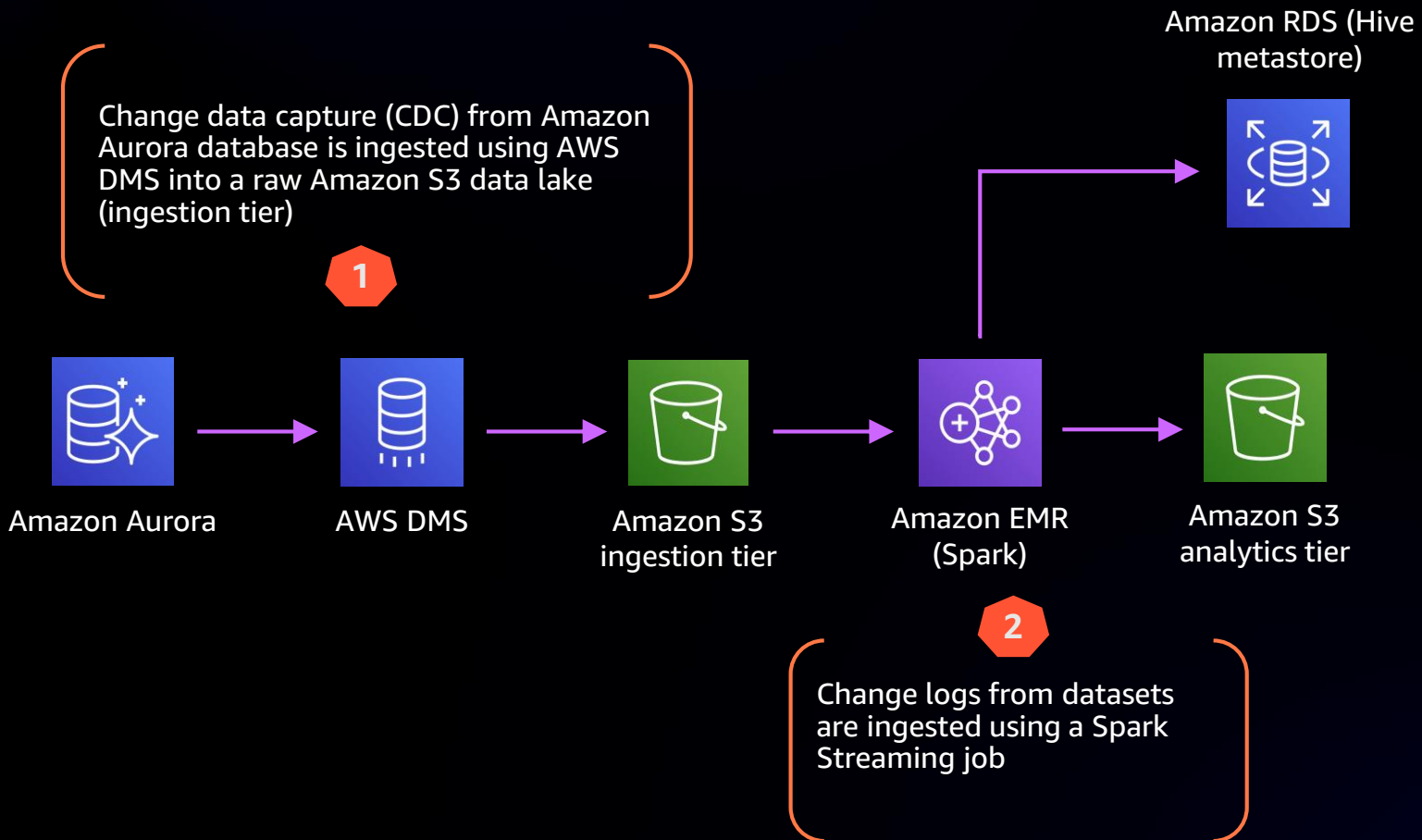
Streaming ingestion pipeline challenges



Query-side challenges

- Incremental query
- Time travel query
- Multiple engine support
 - Presto, Hive, Spark SQL
 - Amazon Athena, Amazon Redshift Spectrum

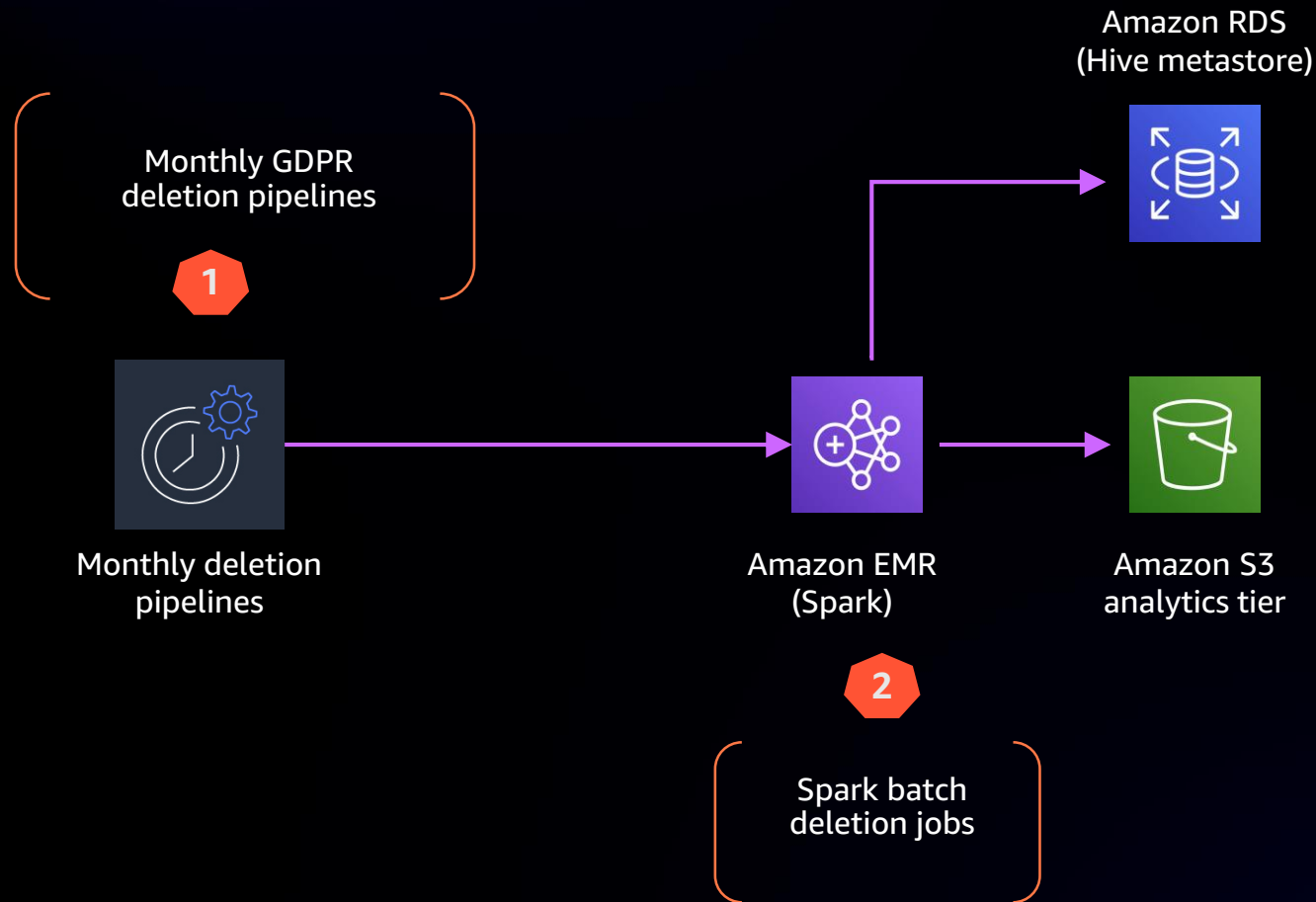
CDC ingestion pipelines challenges



Challenges

- Make atomic changes
- Reader-writer isolation
- High-throughput ingestion
- Small file compactions
- Row-level upserts and deletes
- Clustering by secondary keys

GDPR (data erasure) pipeline challenges



Challenges

- Row-level upserts and deletes
- **Concurrent writers**

Apache Hudi allows transactional data lakes

TRANSACTIONS, RECORD-LEVEL UPDATES/DELETES, AND CHANGE STREAMS TO DATA LAKES

Ingestion



- Transactions (ACID) – reader and writer isolation
- Transactions (ACID) – concurrent writer support
- Record-level upserts and deletes
- High-throughput streaming ingestion
- Spark, Flink, and Java writer support
- Automatic compaction of small files

Query



- Spark, PrestoDB/Trino, and Hive support
- Efficient queries across partitions and files
- Incremental query support
- Time-travel query support

Apache Hudi enables transactional data lakes

AUTOMATE TABLE MANAGEMENT ACTIVITIES

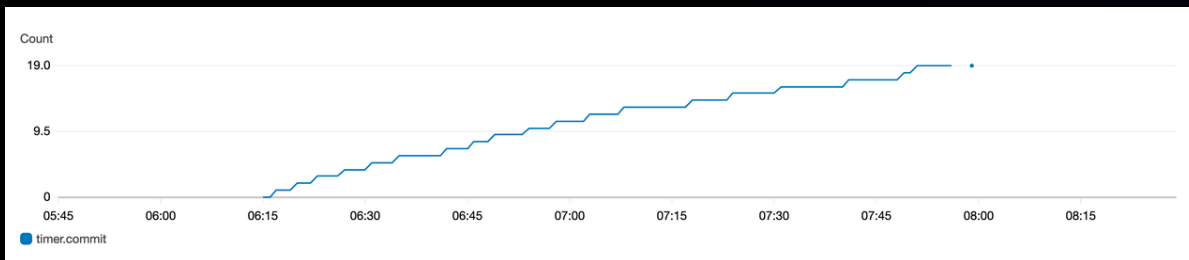
Administration



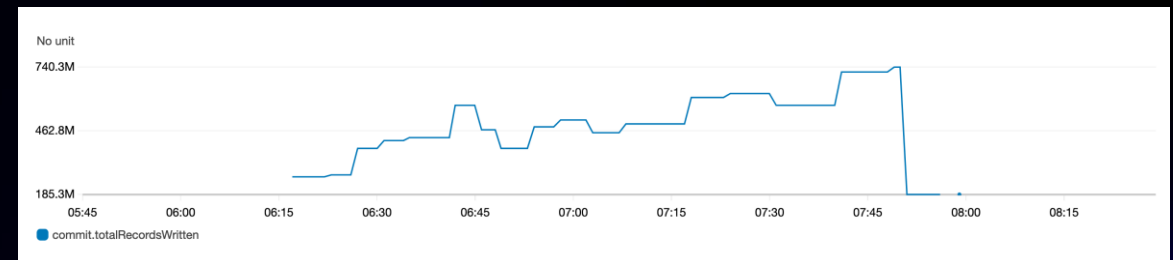
- Async background file compaction
- Async background key sorting and clustering
- Automatically cleanup files beyond retention period
- Metrics for past commits or rollbacks

Easily operationalize at scale using detailed metrics: Amazon CloudWatch integration

All metrics	Graphed metrics	Graph options	Source
N. Virginia ▾	All > Hudi > Hudi Table, Metric Type	Q Search for any metric, dimension or resource id	Graph search
<input type="checkbox"/> Hudi Table (40)	Metric Type	Metric Name	
<input type="checkbox"/> tpcds_store_sales_3TB_08	gauge	commit.totalScanTime	
<input type="checkbox"/> tpcds_store_sales_3TB_08	gauge	commit.totalUpdateRecordsWritten	
<input type="checkbox"/> tpcds_store_sales_3TB_08	gauge	commit.totalUpsertTime	
<input type="checkbox"/> tpcds_store_sales_3TB_08	gauge	finalize.duration	
<input type="checkbox"/> tpcds_store_sales_3TB_08	gauge	finalize.numFilesFinalized	
<input type="checkbox"/> tpcds_store_sales_3TB_08 ▾	count ▾	timer.clean ▾	
<input type="checkbox"/> tpcds_store_sales_3TB_08	count	timer.commit	
<input type="checkbox"/> tpcds_store_sales_3TB_08	gauge	TimelineService.TOTAL_CHECK_TIME	



Number of commits



Total records written



Data pipelines are SQL statements

NEW!

Create Hudi table

```
CREATE TABLE IF NOT EXISTS amazon_product_review_hudi
(
  marketplace    STRING,
  review_id      STRING,
  customer_id    STRING,
  product_title  STRING,
  star_rating    INT,
  timestamp      LONG,
  review_date    DATE,
  year           STRING,
  month          STRING,
  date           STRING
) USING hudi LOCATION 's3://EXAMPLE-BUCKET/my-hudi-
dataset/'
OPTIONS( type = 'cow',
primarykey = 'review_id',
precombinefield = 'timestamp' )
PARTITIONED BY (year, month, date);
```

Upsert into table

```
MERGE INTO amazon_product_review_hudi a0
USING (
  SELECT * AS dt FROM amazon_product_reviews
) d0
ON d0.review_id = a0.review_id
WHEN MATCHED THEN UPDATE SET star_rating =
d0.star_rating, review_date = d0.review_date
WHEN NOT MATCHED THEN INSERT *
```

Apache Hudi is widely supported on AWS

BROAD SUPPORT FOR APACHE HUDI ON AWS



Spark, Hive, Presto, Flink
support on Amazon EMR



AWS Glue Catalog
and ETL support



AWS Lake Formation
FGAC support



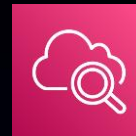
Amazon Athena
native query support



Amazon Redshift Spectrum
native query support



AWS DMS CDC
ingestion support

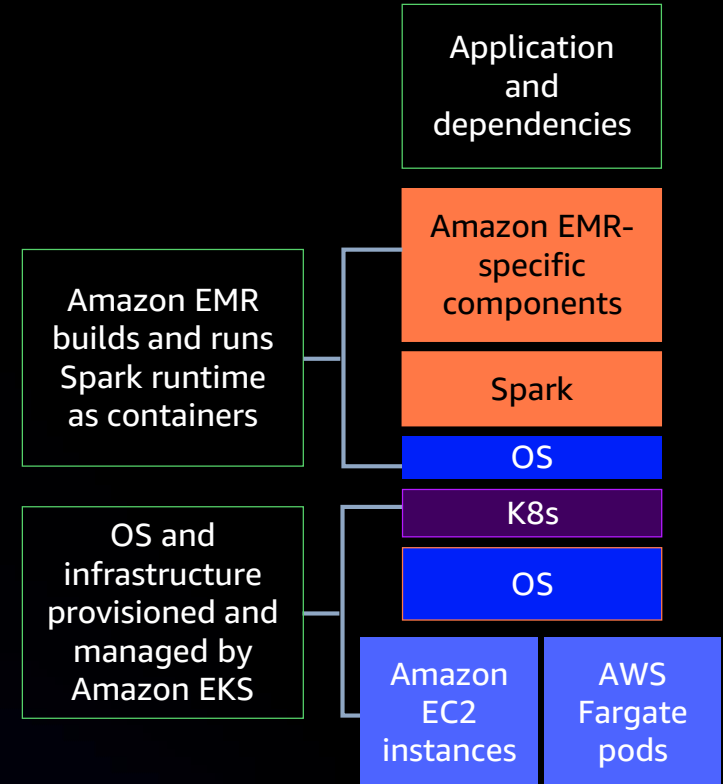


Amazon CloudWatch
integration for metrics

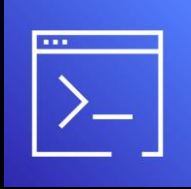
Multiple deployment options for Amazon EMR

Amazon EMR on Amazon EKS

- Simplify infrastructure management
- Containerization drives **job-centric model**
- Run multiple versions of Spark per cluster/**per job execution role**
- Great for faster upgrade cycles
- Great for consolidating resources
- Run application on single AZ or **across multiple AZs**



Job submission options



AWS CLI/SDK



Amazon EMR Studio,
self-managed
notebooks

**Apache
Airflow**



AWS Step
Functions

```
aws emr-containers start-job-run \  
  --virtual-cluster-id cluster_id \  
  --name sample-job-name \  
  --execution-role-arn execution-role-arn \  
  --release-label emr-6.3.0-latest \  
  --job-driver '{  
    "sparkSubmitJobDriver": {  
      "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",  
      "sparkSubmitParameters": "--conf spark.executor.instances=2 --conf spark.executor.memory=2G --conf  
spark.executor.cores=2 --conf spark.driver.cores=1"  
    }  
  }'
```

Custom container images

NEW!

- Install and configure packages specific to your workload
- Set environment variables
- Incorporate data pipeline into CI/CD



```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.3.0-latest
```

```
USER root
```

```
# Install Chrome
```

```
RUN curl https://intoli.com/install-google-chrome.sh \
    | bash && \
    mv /usr/bin/google-chrome-stable /usr/bin/chrome
```

```
# Install bokeh and sampledata
```

```
RUN pip3 install \
    bokeh>=2.3.2 \
    chromedriver-py>=91.0.4472.19.0 \
    selenium>=3.141.0
```

```
RUN bokeh sampledata
```

```
USER hadoop:hadoop
```

Pod templates

NEW!

- Schedule Spark executors to run on Amazon EC2 Spot Instances or Graviton instances
- Run a separate **sidecar** container next to the Spark driver or executor – logging, additional monitoring
- Run an **init** container that prepares the environment (e.g., downloading jars)

```
apiVersion: v1
kind: Pod
spec:
  volumes:
    - name: source-data-volume
      emptyDir: {}
    - name: metrics-files-volume
      emptyDir: {}
  nodeSelector:
    eks.amazonaws.com/nodegroup: emr-containers-nodegroup
  containers:
    - name: custom-side-car-container # Sidecar container
      image: <side_car_container_image>
      env:
        - name: RANDOM_SIDE CAR
          value: random
      volumeMounts:
        - name: metrics-files-volume
          mountPath: /var/metrics/data
      command:
        - /bin/sh
        - '-c'
        - <command-to-upload-metrics-files>
    - name: spark-init-container-driver # Init container
      image: <spark-pre-step-image>
      volumeMounts:
        - name: source-data-volume # Use EMR predefined volumes
          mountPath: /var/data
      command:
        - /bin/sh
        - '-c'
        - <command-to-download-dependency-jars>
```

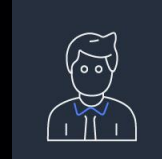
The administrator



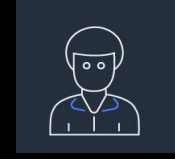
Maria – the data scientist



Ana – the data analyst



Richard – the data engineer



Carlos – **the administrator**

Manages data lakes,
security, and cost

Monitors health and
performance of data
systems like clusters

Amazon EMR deployment options



Amazon EMR on Amazon EC2

Choose instances that offer the best price performance for your workload



Amazon EMR on Amazon EKS

Automate provisioning, management, and scaling of Apache Spark jobs on Amazon EKS



Amazon EMR Serverless

Run applications using open source frameworks like Apache Spark, Hive, and Presto without having to configure, optimize, operate, or secure clusters



Amazon EMR on AWS Outposts

Set up, manage, and scale Amazon EMR in your on-premises environments, just as you would in the cloud

Cost-optimized



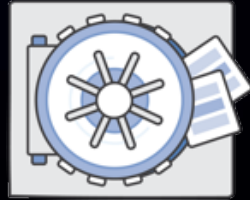
Cost-optimization options

WITH AMAZON EMR, DO MORE WITH LESS!



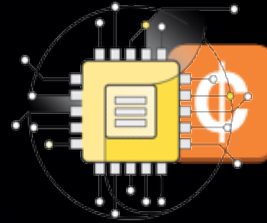
Performance optimizations

- Runtime improvements
- Transactions in data lakes



Compute optimizations

- Graviton instances
- Spot Instances
- Instance fleets



Cluster management

- Managed scaling
- Cluster auto-termination



Containerization

Consolidate analytics and other workloads on Amazon EKS using Amazon EMR on Amazon EKS

AWS **Graviton2** instances have the best price performance within their instance families

We compared M5 vs. M6g using Amazon EMR 5.30.1 using TPC-DS 3 TB benchmark queries with data in Amazon S3



12–16% performance improvement
compared to M5
instance types



20% lower cost
vs. same-sized
comparable
M5 instances



Up to 30% better price performance

Spot Instances are perfect for Amazon EMR instance fleet



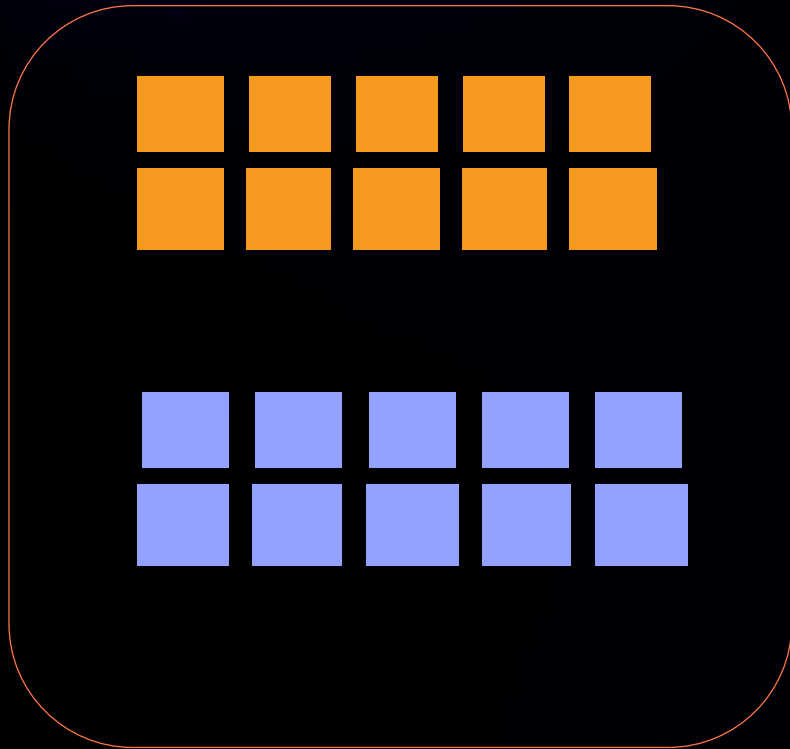
- ✓ Nodes can be configured for a mix of On-Demand and Spot Instances
- ✓ Finds the highest capacity instance at the lowest price
- ✓ If a Spot Instance in a task node is reclaimed, then another instance in your fleet will replace it
- ✓ **Spark on Amazon EMR** adds additional resiliency to handle Spot Instance interruptions gracefully

Scale up with Spot Instances



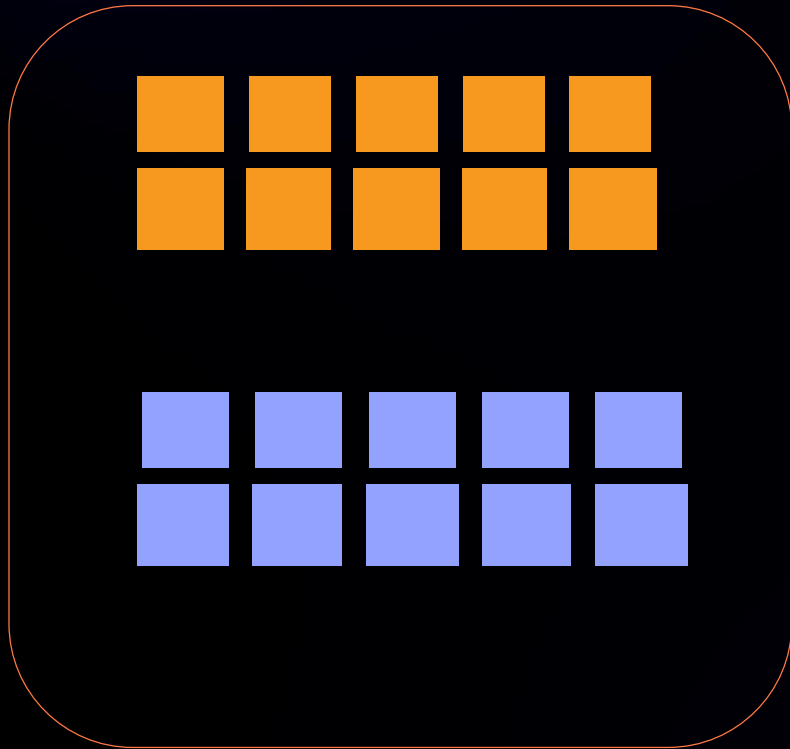
10-node cluster running for 14 hours
Cost = 1.0 * 10 * 14 = \$140

Scale up cluster with Spot Instances



Add 10 more nodes on Spot

Scale up cluster with Spot Instances



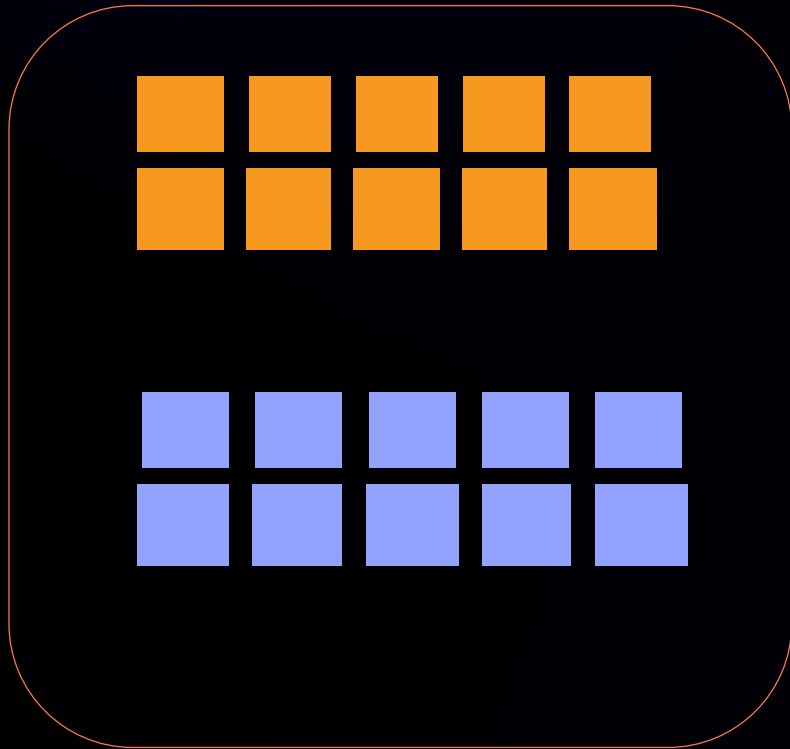
20-node cluster running for 7 hours

$\text{Cost} = 1.0 * 10 * 7 = \70

$= 0.5 * 10 * 7 = \$35$

Total \$105

Scale up cluster with Spot Instances




Use 2x the capacity

50% less runtime (14 → 7)



25% less cost (140 → 105)

Instance fleets: Allocation strategy

Allocation Strategy

The allocation strategy option is an improved method of launching clusters with lowest-priced On-Demand instances and capacity-optimized Spot instances. This option is recommended for faster cluster provisioning, more accurate Spot instance allocation, and fewer Spot instance interruptions compared to default EMR instance fleet allocation. [Learn more](#) 

☒ Apply allocation strategy (Recommended)

 Apply required EC2 service role permissions to your EMR cluster to enable allocation strategies. The Default EMR service role and managed policy already have the necessary service role permissions. [Learn more](#) 

- Capacity-optimized allocation strategy uses real-time capacity data to allocate instances from the Spot Instance pools
- Chooses pools with optimal capacity for the number of instances that are launching
- Appropriate for workloads that have a higher cost of interruption
 - Long-running jobs and multi-tenant persistent clusters running Apache Spark, Apache Hive, and Presto
- Specify up to 30 Amazon EC2 instance types on task instance fleets to diversify your Spot fleet and get better price performance



“Acxiom uses Spark on Amazon EMR on Spot Instances to run 3 trillion inferences in less than 15 hours. By using Amazon EMR, we could utilize spot compute capacity across the entire AWS Region and speed up the run time of our inference pipeline that typically took 11–15 days every month to under 15 hours.”

Varadarajan “Raj” Srinivasan

Senior Director of ML Engineering and Data Science

Acxiom



Managed Scaling feature overview

COMPLETELY MANAGED ENVIRONMENT FOR AUTOMATICALLY RESIZING AMAZON EMR ON EC2 CLUSTERS



Amazon EMR-managed algorithm that constantly improves, giving you a completely managed experience



High resolution metrics enabled with managed scaling



Only min/max cost constraints configurations required



More data points and faster reaction time than earlier auto-scaling feature



Save 20–60% depending on your workload patterns

Managed Scaling enhancements

NEW!

ADDITIONAL ENHANCEMENTS ENABLED BY DEFAULT TO FURTHER REDUCE COSTS



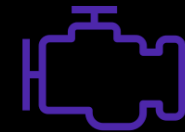
Capacity awareness in
instance groups
enabled by default

Integrated with
real-time Amazon EC2
Spot Instance capacity
metrics to scale the
right task group based
on instance pool depth



Shuffle awareness
enabled by default
from Amazon EMR 6.4

Ensure nodes with
active shuffle data are
not scaled down

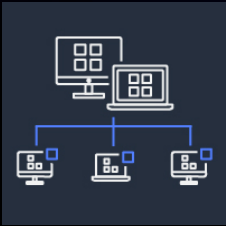


Support for PrestoDB and
Trino available from
Amazon EMR 6.4

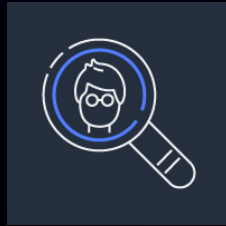
Sign up for
preview access via
aws-support@amazon.com

Security

Amazon EMR provides comprehensive, end-to-end security



Isolation



Authentication



Authorization



Encryption



Audit

VPC

Private subnets

Security groups

LDAP

Kerberos

AWS SSO (Amazon
EMR Studio)

AWS IAM (Amazon
EMR Studio)

Cluster IAM role

FGAC using Apache
Ranger

FGAC using
AWS Lake
Formation
(**preview**)

NEW!

Encryption at rest

Encryption in transit

Key management

Audit using
Ranger

Audit using
AWS Lake
Formation
(**preview**)

NEW!

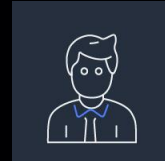
In conclusion



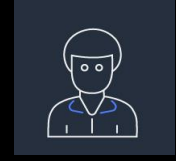
Maria – the data scientist



Ana – the data analyst



Richard – the data engineer



Carlos – the administrator

Cost-optimized

Security controls

Performance-optimized

Transactional data lakes –
Hudi

Amazon MWAA and AWS
Step Functions integrations

Amazon EMR Studio

Amazon SageMaker integrations

Thank you!

Radhika Ravirala
ravirala@amazon.com

Neil Mukerje
mukerjen@amazon.com

