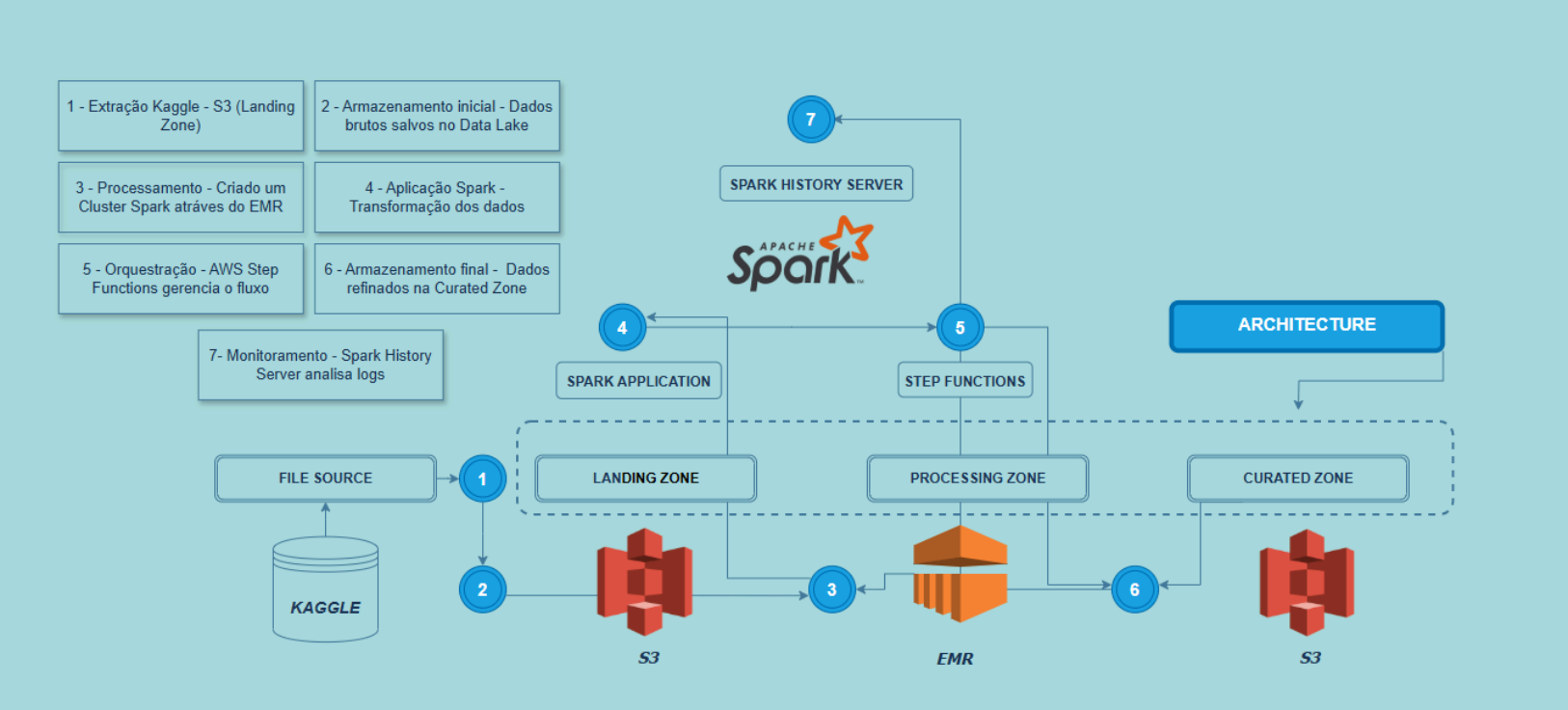
Pipeline de Dados com AWS

Este projeto consiste em um pipeline de processamento de dados na AWS, utilizando uma arquitetura de 3 camadas (dados, lógica e aplicação) e em diferentes zonas de armazenamento (Landing, Processing, Curated) para organizar e processar dados de forma estruturada. O processamento é realizado usando um cluster no Amazon EMR com um job escrito em PySpark. O cluster Spark lê os dados brutos da Landing Zone, realiza transformações e agregações, e salva os resultados nas zonas subsequentes.



Arquitetura

Usaremos a arquitetura 3 camadas (dados, lógica, aplicação) que nos dá a praticidade para utilizar as tecnologias que preferimos em cada camada, sem a necessidade de concentrar de forma monolítica.

Landing Zone

Aqui é a camada de **dados**: a área onde faço o armazenamento dos dados raw (brutos) extraindo da minha fonte de dados. aqui geralmente não faço grandes transformações, apenas o armazenamento mesmo.

- **Características:**

Contém dados no formato original, como foram extraídos. Pode ter dados estruturados, semiestruturados (JSON, XML) ou não estruturados (imagens, vídeos, logs). Utilizada para garantir uma cópia fiel do dado original, preservando a integridade.

Processing Zone

Essa é a camada de **Lógica**: Aqui, os dados são processados e transformados. Isso inclui limpeza, normalização, agregação ou qualquer tipo de tratamento necessário.

- **Características**: Dados começam a ser estruturados e organizados. Pode envolver processos de deduplicação, correção de erros, conversão de formatos e enriquecimento.

Curated Zone

Camada de **Aplicação/Apresentação**: Essa é a camada onde os dados processados e organizados são armazenados de forma otimizada para serem consumidos por relatórios ou análises.

- **Características:** Dados preparados para relatórios, análises e machine learning. Estruturados em formatos prontos para consulta, como tabelas SQL ou datasets otimizados (Parquet, ORC).

Tecnologias Utilizadas

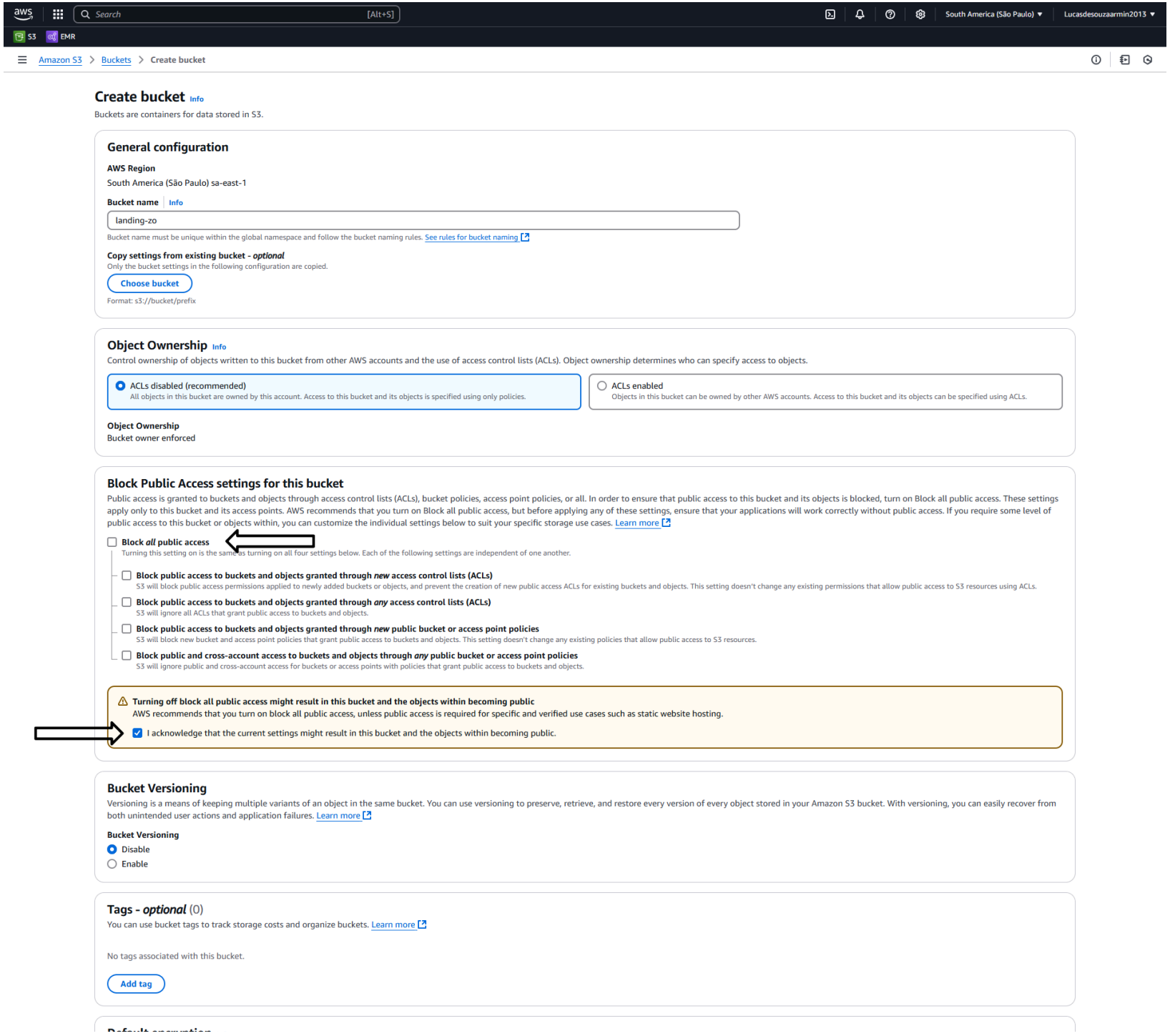
- **AWS S3**: Armazenamento de dados nas zonas Landing, Processing e Curated.
- **Amazon EMR**: Execução do job Spark.
- **Apache Spark**: Processamento distribuído de dados.
- **PySpark**: API de Python utilizada para o job Spark.
- **Spark History Server**: Visualização do histórico de execução dos jobs Spark.
- **AWS Step Functions**: Orquestração da Pipeline

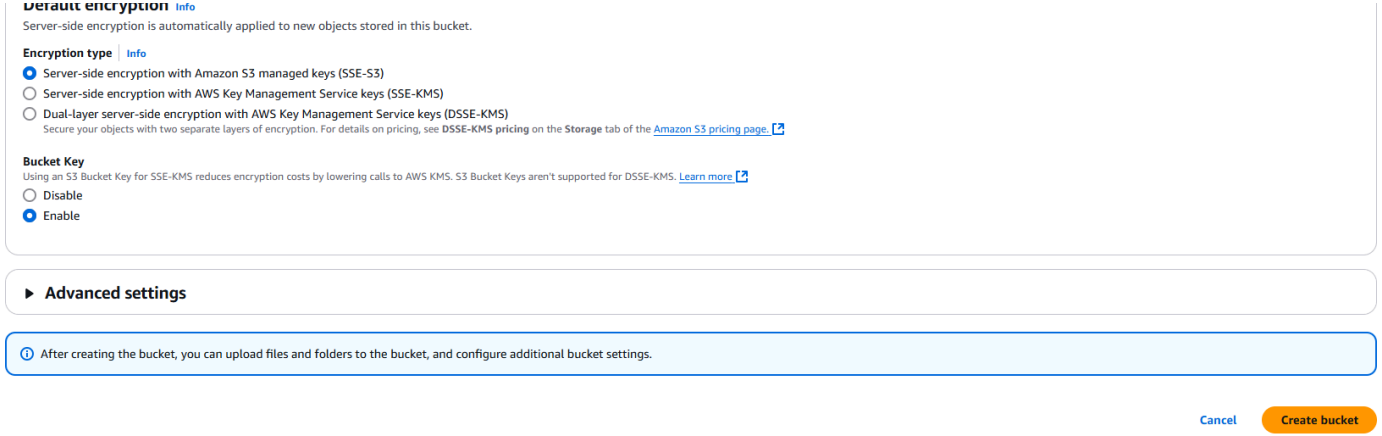
Guia Prático do projeto

1° Inicio criando os buckets no Amazon S3 para o processamento dos dados

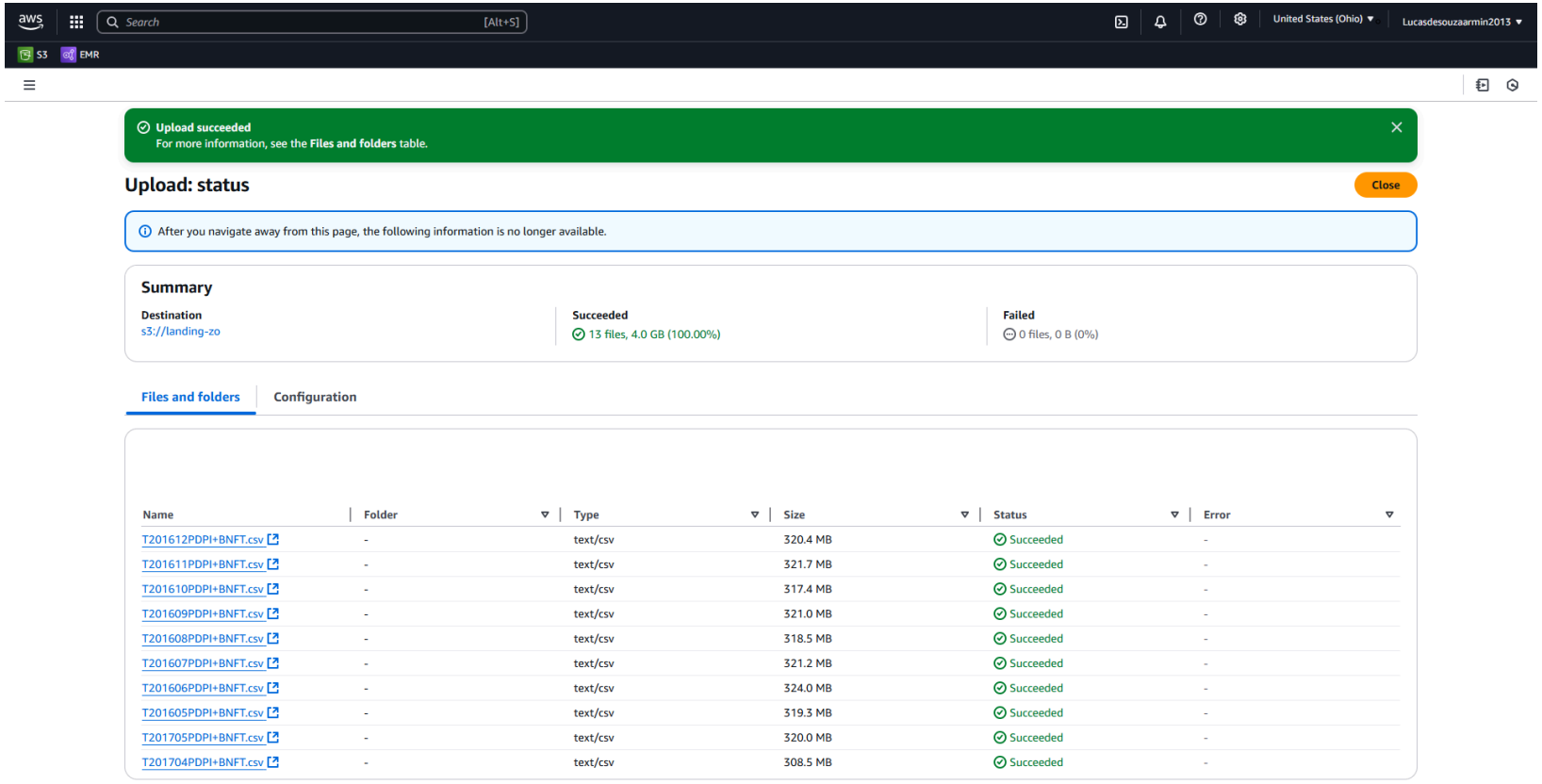
- Landing.zo
- Processing.zo
- Curated.zo

Desmarque o bloqueio ao acesso público. Por padrão, a AWS vai perguntar se você realmente quer fazer isso por questões de segurança. Mas, como esse bucket será usado somente para esse projeto e não para um ambiente de produção, não tem problema confiar e marcar a opção.


Search [Alt+S]
South America (São Paulo)
Lucasdesouzaarmin2013
S3
EMR
Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
South America (São Paulo) sa-east-1
Bucket name Info
landing-zo
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix
Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.
Object Ownership
Bucket owner enforced
Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
Block public access to buckets and objects granted through new access control lists (ACLs)
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.
Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.
I acknowledge that the current settings might result in this bucket and the objects within becoming public.
Bucket Versioning
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more
Bucket Versioning
Disable
Enable
Tags - optional (0)
You can use bucket tags to track storage costs and organize buckets. Learn more
No tags associated with this bucket.
Add tag



2° Faço o upload dos arquivos que eu já extrair da nossa fonte de dados diretamente para o bucket **Landing.zo**



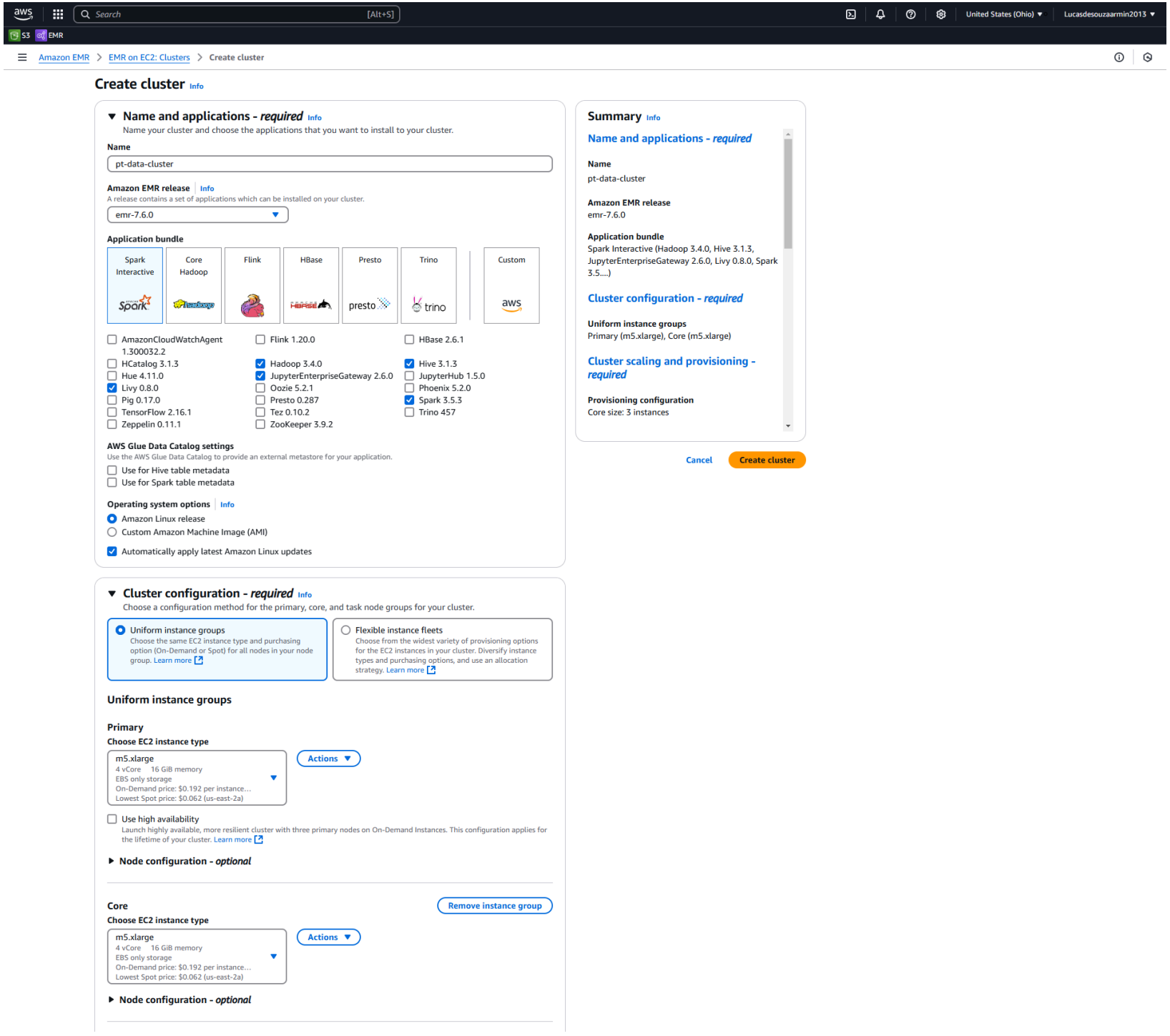
3° Crio um cluster no Amazon EMR

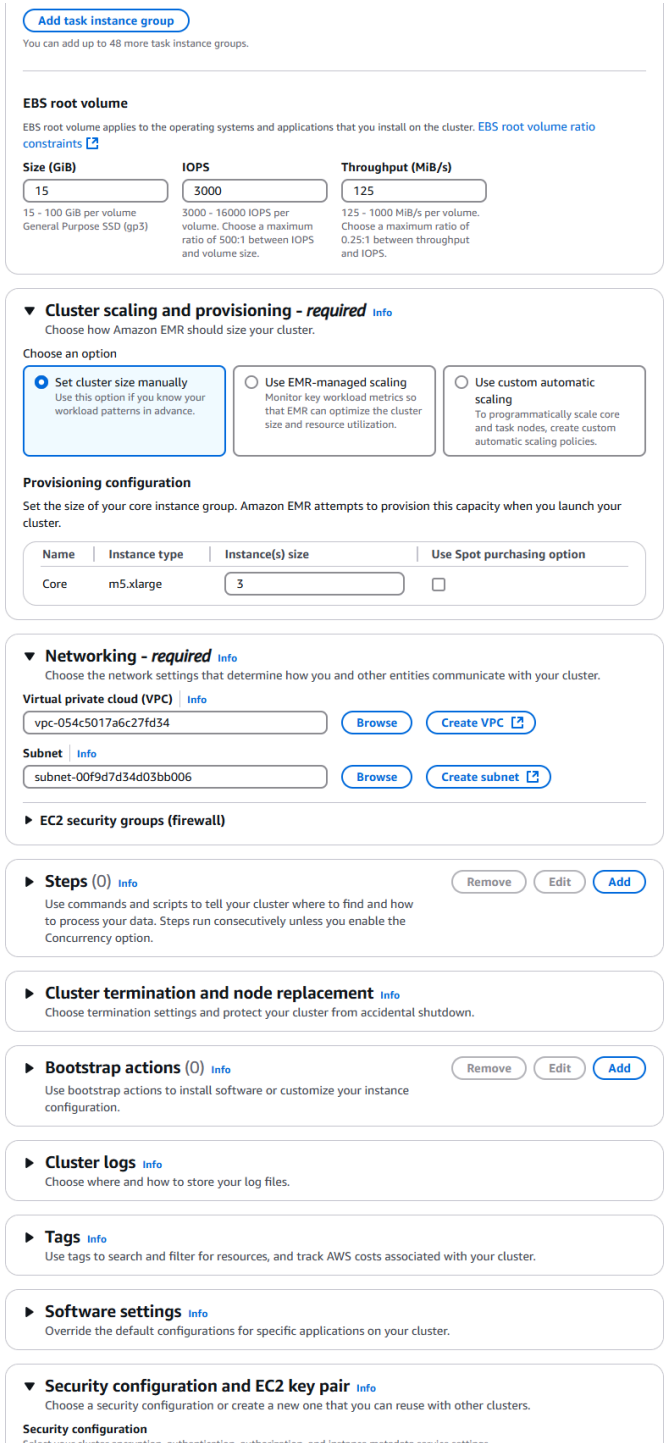
- **Name**: Coloco o nome do meu cluster 'pt-data-cluster'.
- **Amazon EMR release**: Escolho a versão EMR 7.6 e deixo os pacotes de aplicação como default.
- **Cluster configuration**: Deixo selecionado o campo **Uniform instance** opção que oferece facilidade, estabilidade enquanto a **Flexible Instance Fleets** Permite otimizar custos e desempenho.
- **Primary**: Escolho a instância EC2 que irá atuar como nó mestre, responsável por gerenciar o cluster.
- **Core Instance**: Define os nós principais que processarão os dados. Esses nós armazenam e executam os workloads do cluster.
- **Cluster Scaling and Provisioning**: Seleciono **Set cluster size manually** que me permite definir o tamanho do cluster de forma manual, assim evitando escalonamento automático e custos inesperados.

- **Provisioning Configuration**: Defino 3 instâncias para o meu uso. Assim, aumentando minha escalabilidade, tolerância a falhas e desempenho.
- **Virtual Private Cloud**: Define a rede onde o cluster roda, controlando segurança, acesso e conectividade com outros serviços da AWS. Normalmente, já vem uma VPC padrão quando se cria uma conta na AWS, mas você pode criar uma que mais se encaixar no seu uso.
- **Security Configuration and EC2 Key Pair**: Serve para acessar as instâncias do cluster via SSH.
- **Identity and Access Management (IAM)**: Seleciono **Create a service role** e depois, em **Security group**, coloco a vpc que eu já selecionei acima em **Networking**. E permito o acesso a todos os meus buckets.
- **EC2 instance profile for Amazon EMR**: Escolho a opção **Create an instance profile** O EMR criará automaticamente uma IAM Role nova com permissões padrão, facilitando para quem não quer configurar a Role manualmente.

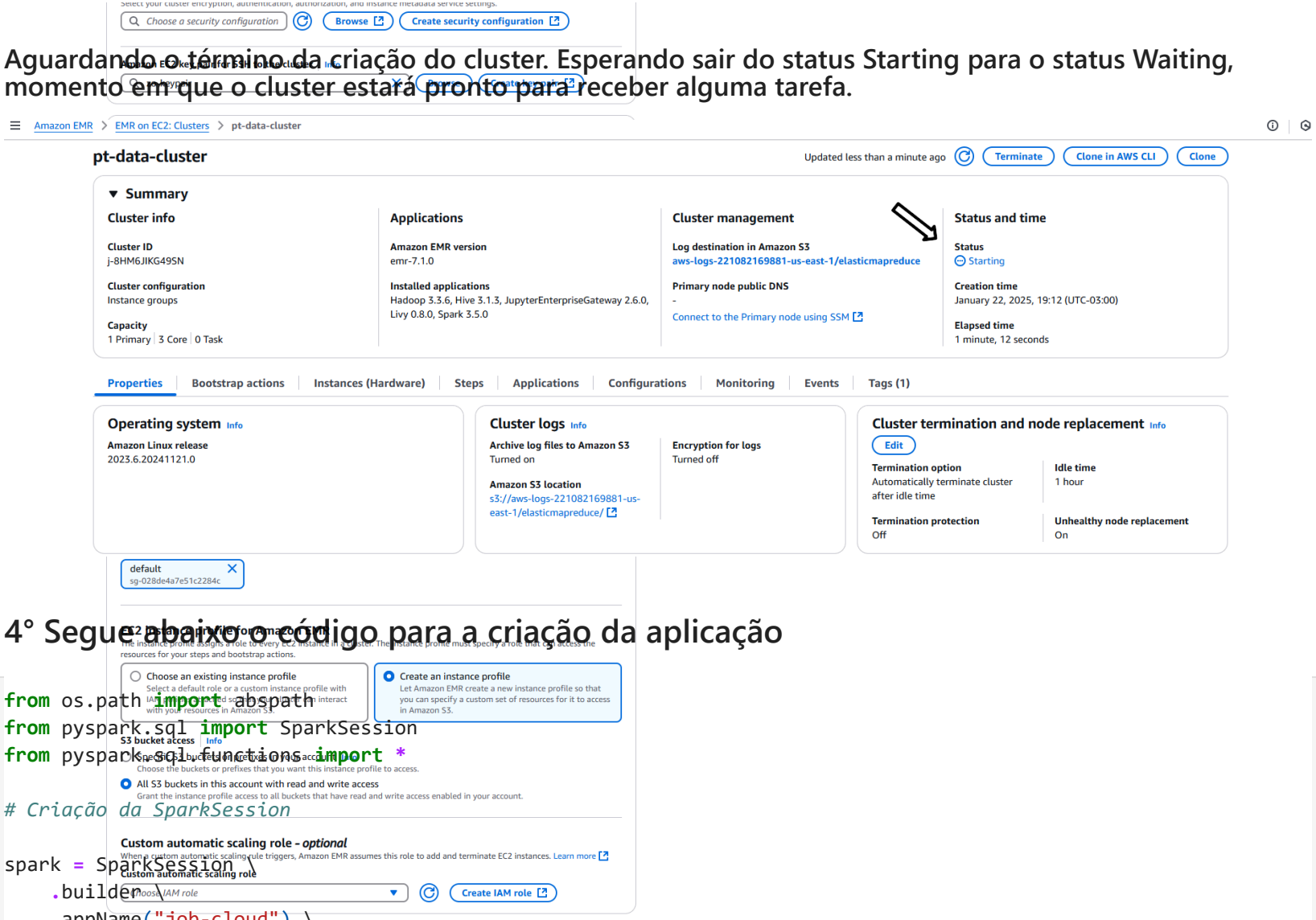
Lembrando que há diversas outras possibilidades de configurações no cluster, mas aqui coloquei somente as que eu fiz algum tipo de alteração.

- ## Após esses processo, nosso cluster está pronto para ser criado.


Search [Alt+S]
United States (Ohio)
Lucasdesouzaarmin2013
S3 EMR
Amazon EMR > EMR on EC2: Clusters > Create cluster
Create cluster Info
Name and applications - required Info
Name your cluster and choose the applications that you want to install to your cluster.
Name
pt-data-cluster
Amazon EMR release Info
A release contains a set of applications which can be installed on your cluster.
emr-7.6.0
Application bundle
Spark Interactive
Core Hadoop
Flink
HBase
Presto
Trino
Custom
AmazonCloudWatchAgent 1.300032.2
Flink 1.20.0
HBase 2.6.1
HCatalog 3.1.3
Hadoop 3.4.0
Hive 3.1.3
Hue 4.11.0
JupyterEnterpriseGateway 2.6.0
JupyterHub 1.5.0
Livy 0.8.0
Oozie 5.2.1
Phoenix 5.2.0
Pig 0.17.0
Presto 0.287
Spark 3.5.3
TensorFlow 2.16.1
Tez 0.10.2
Trino 457
Zeppelin 0.11.1
ZooKeeper 3.9.2
AWS Glue Data Catalog settings
Use the AWS Glue Data Catalog to provide an external metastore for your application.
Use for Hive table metadata
Use for Spark table metadata
Operating system options Info
Amazon Linux release
Custom Amazon Machine Image (AMI)
Automatically apply latest Amazon Linux updates
Summary Info
Name and applications - required
Name
pt-data-cluster
Amazon EMR release
emr-7.6.0
Application bundle
Spark Interactive (Hadoop 3.4.0, Hive 3.1.3, JupyterEnterpriseGateway 2.6.0, Livy 0.8.0, Spark 3.5....)
Cluster configuration - required
Uniform instance groups
Primary (m5.xlarge), Core (m5.xlarge)
Cluster scaling and provisioning - required
Provisioning configuration
Core size: 3 instances
Cancel
Create cluster
Cluster configuration - required Info
Choose a configuration method for the primary, core, and task node groups for your cluster.
Uniform instance groups
Choose the same EC2 instance type and purchasing option (On-Demand or Spot) for all nodes in your node group. Learn more
Flexible instance fleets
Choose from the widest variety of provisioning options for the EC2 instances in your cluster. Diversify instance types and purchasing options, and use an allocation strategy. Learn more
Uniform instance groups
Primary
Choose EC2 instance type
m5.xlarge
4 vCore 16 GiB memory
EBS only storage
On-Demand price: $0.192 per instance...
Lowest Spot price: $0.062 (us-east-2a)
Actions
Use high availability
Launch highly available, more resilient cluster with three primary nodes on On-Demand Instances. This configuration applies for the lifetime of your cluster. Learn more
Node configuration - optional
Core
Remove instance group
Choose EC2 instance type
m5.xlarge
4 vCore 16 GiB memory
EBS only storage
On-Demand price: $0.192 per instance...
Lowest Spot price: $0.062 (us-east-2a)
Actions
Node configuration - optional


Add task instance group
You can add up to 48 more task instance groups.
EBS root volume
EBS root volume applies to the operating systems and applications that you install on the cluster. EBS root volume ratio constraints
Size (GiB)
15
15 - 100 GiB per volume General Purpose SSD (gp3)
IOPS
3000
3000 - 16000 IOPS per volume. Choose a maximum ratio of 500:1 between IOPS and volume size.
Throughput (MiB/s)
125
125 - 1000 MiB/s per volume. Choose a maximum ratio of 0.25:1 between throughput and IOPS.
Cluster scaling and provisioning - required Info
Choose how Amazon EMR should size your cluster.
Choose an option
Set cluster size manually
Use this option if you know your workload patterns in advance.
Use EMR-managed scaling
Monitor key workload metrics so that EMR can optimize the cluster size and resource utilization.
Use custom automatic scaling
To programmatically scale core and task nodes, create custom automatic scaling policies.
Provisioning configuration
Set the size of your core instance group. Amazon EMR attempts to provision this capacity when you launch your cluster.
Name
Instance type
Instance(s) size
Use Spot purchasing option
Core
m5.xlarge
3
Networking - required Info
Choose the network settings that determine how you and other entities communicate with your cluster.
Virtual private cloud (VPC) Info
vpc-054c5017a6c27fd34
Browse
Create VPC
Subnet Info
subnet-00f9d7d34d03bb006
Browse
Create subnet
EC2 security groups (firewall)
Steps (0) Info
Use commands and scripts to tell your cluster where to find and how to process your data. Steps run consecutively unless you enable the Concurrency option.
Remove
Edit
Add
Cluster termination and node replacement Info
Choose termination settings and protect your cluster from accidental shutdown.
Bootstrap actions (0) Info
Use bootstrap actions to install software or customize your instance configuration.
Remove
Edit
Add
Cluster logs Info
Choose where and how to store your log files.
Tags Info
Use tags to search and filter for resources, and track AWS costs associated with your cluster.
Software settings Info
Override the default configurations for specific applications on your cluster.
Security configuration and EC2 key pair Info
Choose a security configuration or create a new one that you can reuse with other clusters.
Security configuration

Aguardando o término da criação do cluster. Esperando sair do status Starting para o status Waiting, momento em que o cluster estará pronto para receber alguma tarefa.



4° Segue abaixo o código para a criação da aplicação

In [ ]:
```
from os.path import abspath
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

# Criação da SparkSession

spark = SparkSession \
    .builder \
    .appName("job-cloud") \
    .getOrCreate()


# Configurando o nível de logging
spark.sparkContext.setLogLevel("ERROR")

# Lendo os dados do Data Lake
df = spark.read.format("csv") \
```

```
    .option("header", "True") \
    .option("inferSchema", "True") \
    .csv("s3a://landing-zo/*.csv")

# Imprimindo os dados lidos da raw
print("\nImprime os dados lidos da landing:")
df.show()

# Imprimindo o schema do dataframe
print("\nImprime o schema do dataframe lido da raw:")
df.printSchema()

# Convertendo para formato parquet
print("\nEscrevendo os dados lidos da raw para parquet na processing zone...")
df.write.format("parquet") \
    .mode("overwrite") \
    .save("s3a://processing-zo/df-parquet-file.parquet")

# Lendo arquivos parquet
df_parquet = spark.read.format("parquet") \
    .load("s3a://processing-zo/df-parquet-file.parquet")

# Imprimindo os dados lidos em parquet
print("\nImprime os dados lidos em parquet da processing zone")
df_parquet.show()

# Criando uma view para trabalhar com SQL
df_parquet.createOrReplaceTempView("view_df_parquet")

# Processando os dados conforme regra de negócio
df_result = spark.sql("""
    SELECT
        BNF_CODE AS Bnf_code,
        SUM(ACT_COST) AS Soma_Act_cost,
        SUM(QUANTITY) AS Soma_Quantity,
        SUM(ITEMS) AS Soma_items,
        AVG(ACT_COST) AS Media_Act_cost
    FROM view_df_parquet
    GROUP BY BNF_CODE
""")

# Imprimindo o resultado do dataframe criado
```

```
print("\n========= Imprime o resultado do dataframe processado =========\n")
df_result.show()

# Convertendo os dados processados para parquet e escrevendo na curated zone
print("\nEscrevendo os dados processados na Curated Zone...")
df_result.write.format("parquet") \
    .mode("overwrite") \
    .save("s3a://curated-zo/df-result-file.parquet")

# Finalizando a aplicação
spark.stop()
```

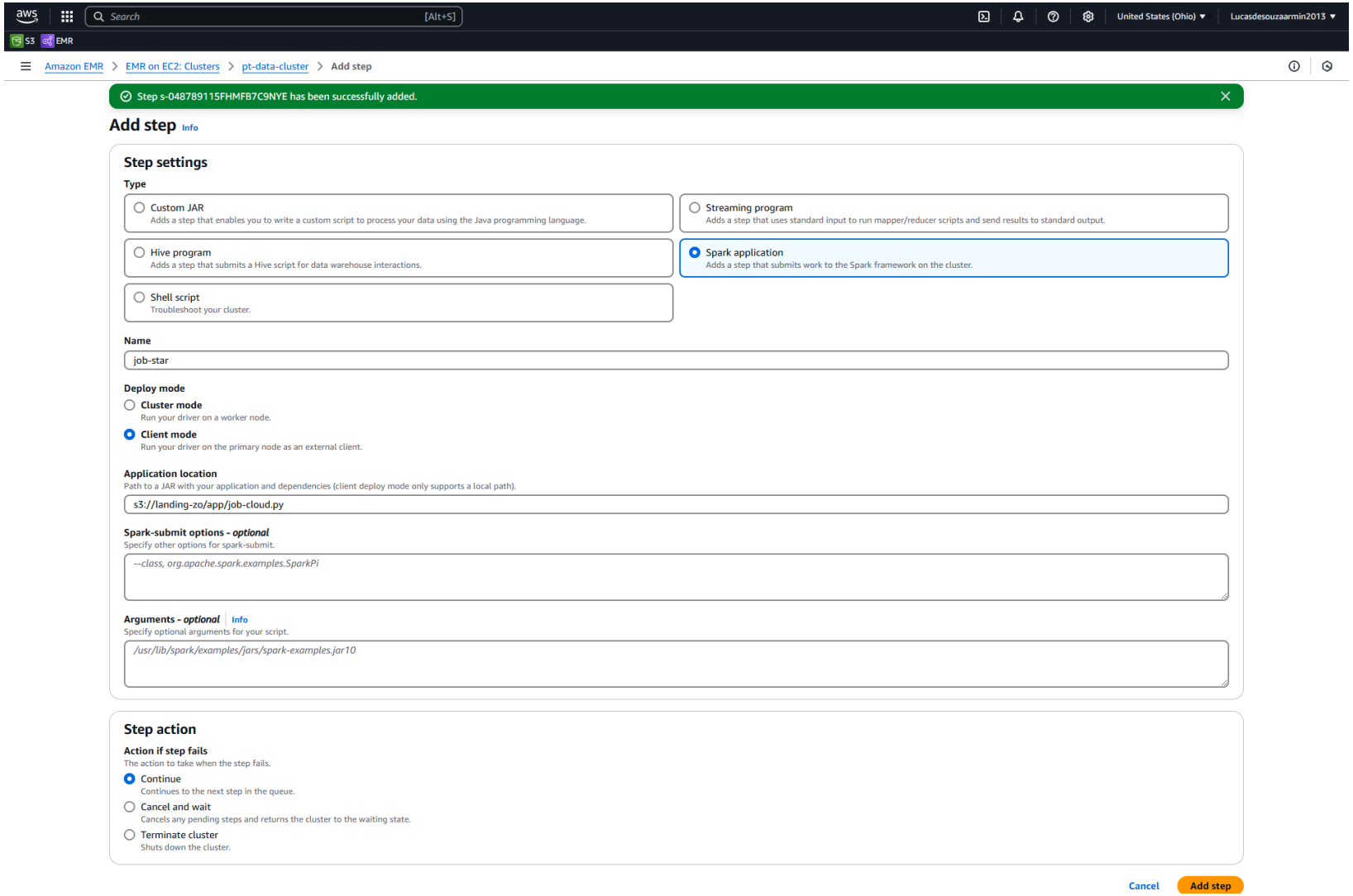
5° Agora iniciaremos o processo de criação de uma **Step Functions**

Step Functions: Servirá para a automação do nosso ETL

- Crio uma pasta chamada App dentro do bucket Landing-zo para ser alocado a nossa aplicação Spark.

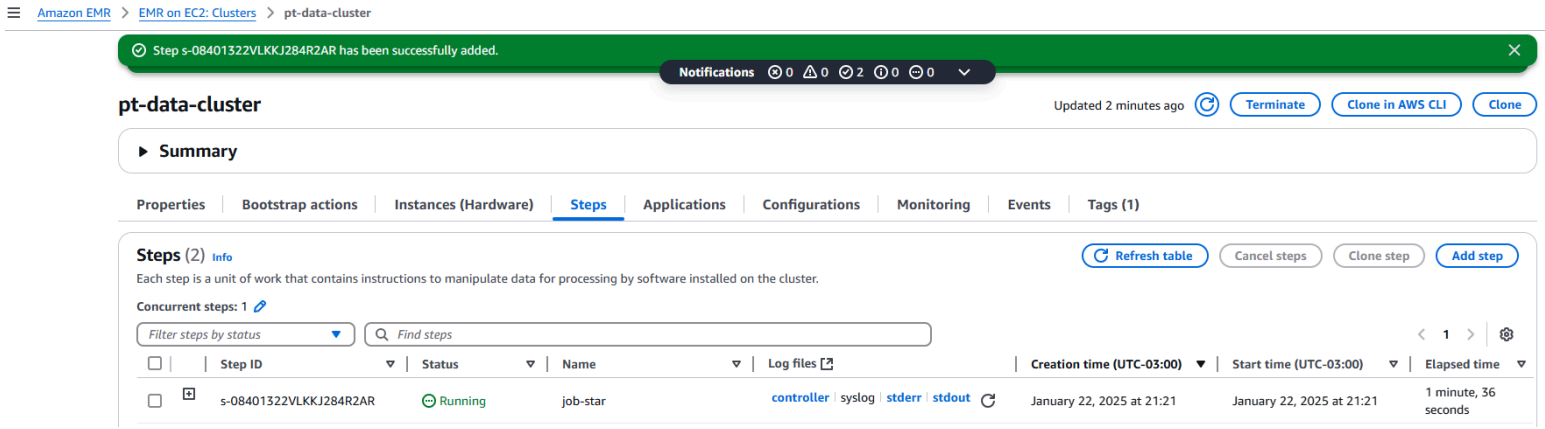
Criando uma step

- **Step setting:** Escolho uma Spark applicattion.
- **Name:** job-star.
- **Deploy mode:** Coloco a opção **Client mode** que é voltada para testes e desenvolvimento.
- **Aplicattion location:** Passo o caminho onde se encontra a aplicação Spark.



Step em execução

- **Log files:** Em stdout já conseguimos verificar a primeira saída da nossa aplicação


Amazon EMR > EMR on EC2: Clusters > pt-data-cluster
Step s-08401322VLKKJ284R2AR has been successfully added.
Notifications 0 0 2 0 0
pt-data-cluster
Updated 2 minutes ago
Terminate
Clone in AWS CLI
Clone
Summary
Properties
Bootstrap actions
Instances (Hardware)
Steps
Applications
Configurations
Monitoring
Events
Tags (1)
Steps (2) Info
Each step is a unit of work that contains instructions to manipulate data for processing by software installed on the cluster.
Refresh table
Cancel steps
Clone step
Add step
Concurrent steps: 1
Filter steps by status
Find steps
Step ID
Status
Name
Log files
Creation time (UTC-03:00)
Start time (UTC-03:00)
Elapsed time
s-08401322VLKKJ284R2AR
Running
job-star
controller
syslog
stderr
stdout
January 22, 2025 at 21:21
January 22, 2025 at 21:21
1 minute, 36 seconds

Saída

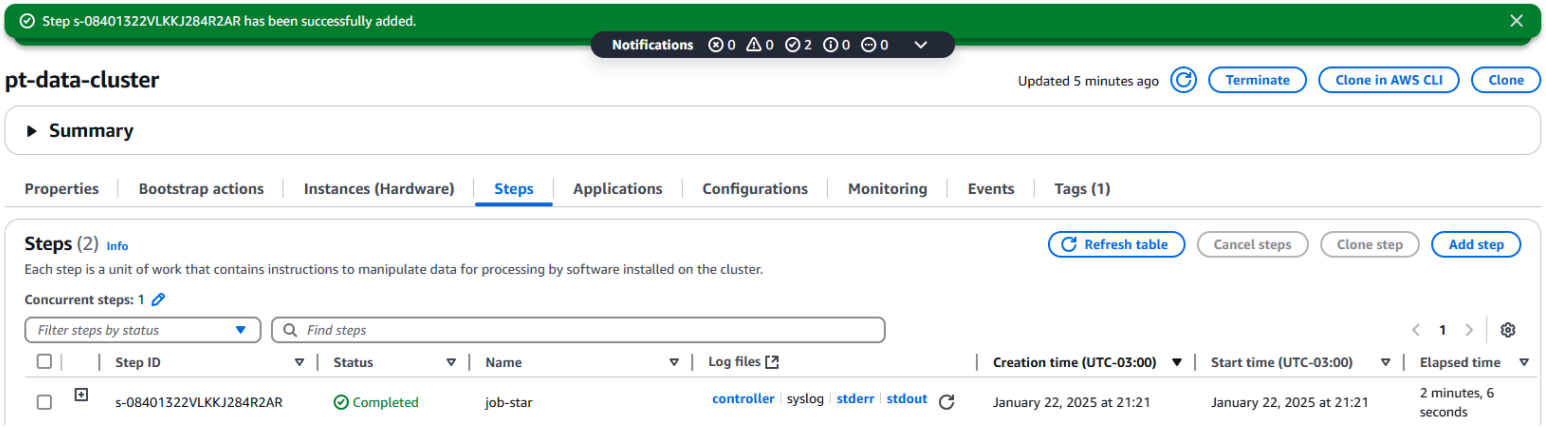
```
Imprime os dados lidos da landing:
+--------+--------+--------+-----+------+--------+--------+
|practice|bnf_code|bnf_name|items|   nic|act_cost|quantity|
+--------+--------+--------+-----+------+--------+--------+
|    5668|    8092|     592|    2|  44.1|   40.84|     189|
|    1596|   17512|   16983|    2|  1.64|    1.64|      35|
|    1596|   25587|   16124|    1|  1.26|    1.28|      42|
|    1596|   12551|    1282|    2|  0.86|    1.02|      42|
|    1596|   18938|   10575|    1|  1.85|    1.82|      56|
|    1596|    8777|   21507|    1|  3.31|    3.18|      56|
|    1596|    9369|   12008|    1| 63.15|   58.56|      56|
|    1596|   27926|   17643|    2|158.66|  147.07|      56|
|    1596|   26148|   10230|    1|  0.35|    0.44|      14|
|    1596|    9148|    3381|    1|  0.26|    0.35|       7|
|    1596|   19500|   18008|    1|  0.85|     0.9|      14|
|    1596|    5312|    2870|    1|  0.56|    0.63|      14|
|    1596|    5008|   15013|    1|  0.19|    0.29|       7|
|    1596|    2885|    1851|    2|  1.99|    2.06|      59|
|    1596|    5737|   10645|    4|  4.59|     4.6|      91|
|    1596|   14030|   17315|    1|  1.16|    1.19|      21|
|    1596|   22190|   12644|    2|  2.26|    2.32|      49|
|    1596|   25059|   20447|    1|  1.23|    1.25|      28|
|    1596|   24523|    7372|    2|  1.68|    1.68|      35|
|    1596|   22227|    5860|    4|  2.69|    2.84|      57|
+--------+--------+--------+-----+------+--------+--------+
only showing top 20 rows


Imprime o schema do dataframe lido da raw:
root
 |-- practice: integer (nullable = true)
 |-- bnf_code: integer (nullable = true)
 |-- bnf_name: integer (nullable = true)
 |-- items: integer (nullable = true)
 |-- nic: double (nullable = true)
 |-- act_cost: double (nullable = true)
 |-- quantity: integer (nullable = true)


Escrevendo os dados lidos da raw para parquet na processing zone...
```

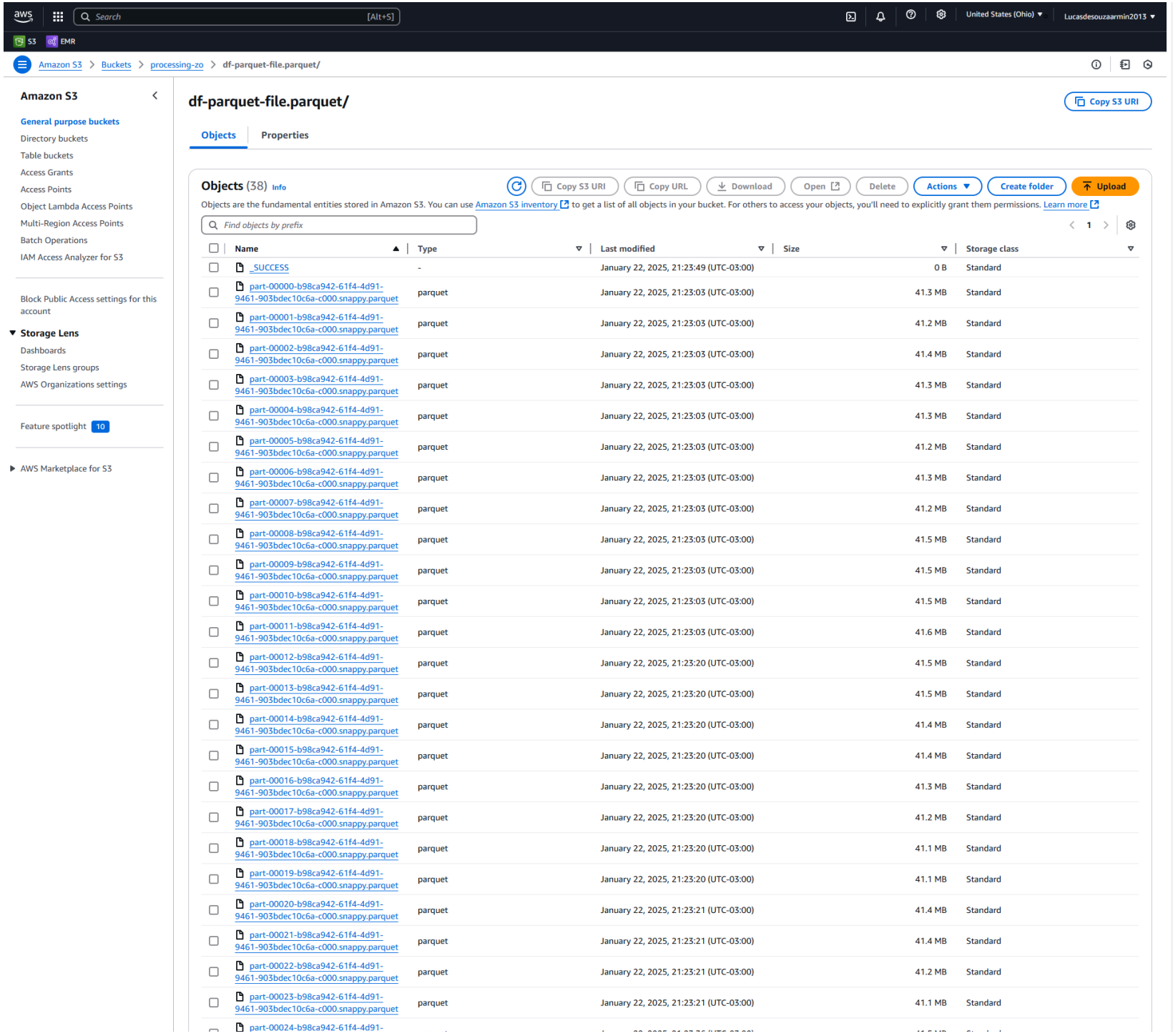
Step concluída com sucesso

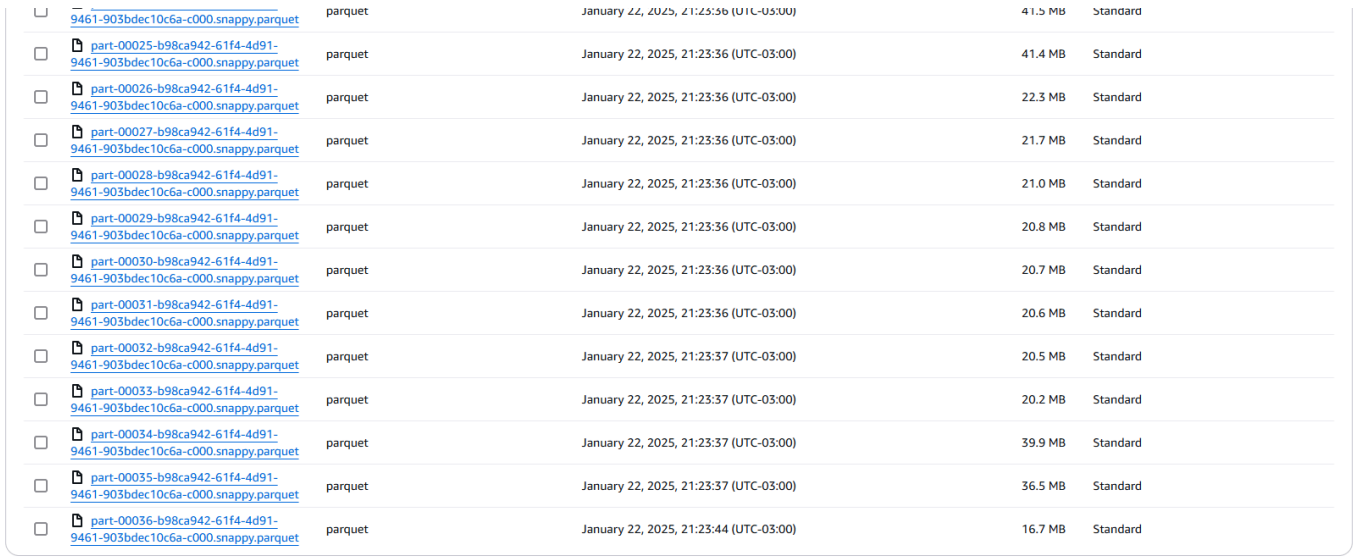
- Caso o job tivesse alguma falha, iria mostrar em Status.



6° Verificado os resultados dentro do datalake

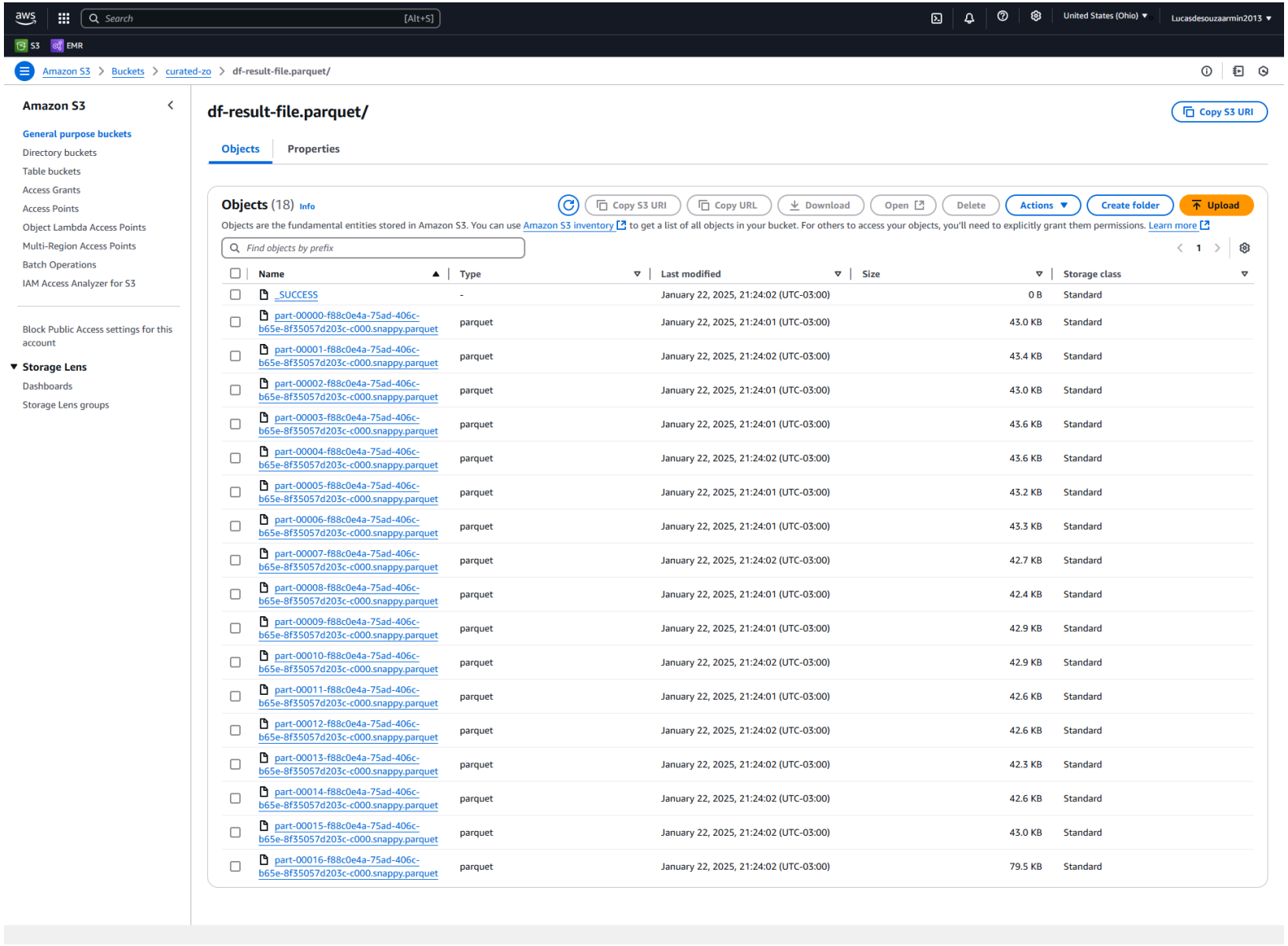
Processing zone


aws
Search
[Alt+S]
United States (Ohio)
Lucasdesouzaarmin2013
S3
EMR
Amazon S3 > Buckets > processing-zo > df-parquet-file.parquet/
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 10
AWS Marketplace for S3
df-parquet-file.parquet/
Copy S3 URI
Objects
Properties
Objects (38) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
_SUCCESS - January 22, 2025, 21:23:49 (UTC-03:00) 0 B Standard
part-00000-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.3 MB Standard
part-00001-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.2 MB Standard
part-00002-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.4 MB Standard
part-00003-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.3 MB Standard
part-00004-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.3 MB Standard
part-00005-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.2 MB Standard
part-00006-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.3 MB Standard
part-00007-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.2 MB Standard
part-00008-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.5 MB Standard
part-00009-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.5 MB Standard
part-00010-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.5 MB Standard
part-00011-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:03 (UTC-03:00) 41.6 MB Standard
part-00012-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.5 MB Standard
part-00013-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.5 MB Standard
part-00014-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.4 MB Standard
part-00015-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.4 MB Standard
part-00016-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.3 MB Standard
part-00017-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.2 MB Standard
part-00018-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.1 MB Standard
part-00019-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:20 (UTC-03:00) 41.1 MB Standard
part-00020-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:21 (UTC-03:00) 41.4 MB Standard
part-00021-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:21 (UTC-03:00) 41.4 MB Standard
part-00022-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:21 (UTC-03:00) 41.2 MB Standard
part-00023-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet parquet January 22, 2025, 21:23:21 (UTC-03:00) 41.1 MB Standard
part-00024-b98ca942-61f4-4d91-



part-00025-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:36 (UTC-03:00)	41.4 MB	Standard
part-00026-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:36 (UTC-03:00)	22.3 MB	Standard
part-00027-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:36 (UTC-03:00)	21.7 MB	Standard
part-00028-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:36 (UTC-03:00)	21.0 MB	Standard
part-00029-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:36 (UTC-03:00)	20.8 MB	Standard
part-00030-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:36 (UTC-03:00)	20.7 MB	Standard
part-00031-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:36 (UTC-03:00)	20.6 MB	Standard
part-00032-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:37 (UTC-03:00)	20.5 MB	Standard
part-00033-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:37 (UTC-03:00)	20.2 MB	Standard
part-00034-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:37 (UTC-03:00)	39.9 MB	Standard
part-00035-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:37 (UTC-03:00)	36.5 MB	Standard
part-00036-b98ca942-61f4-4d91-9461-903bdec10c6a-c000.snappy.parquet	parquet	January 22, 2025, 21:23:44 (UTC-03:00)	16.7 MB	Standard

Curated zone

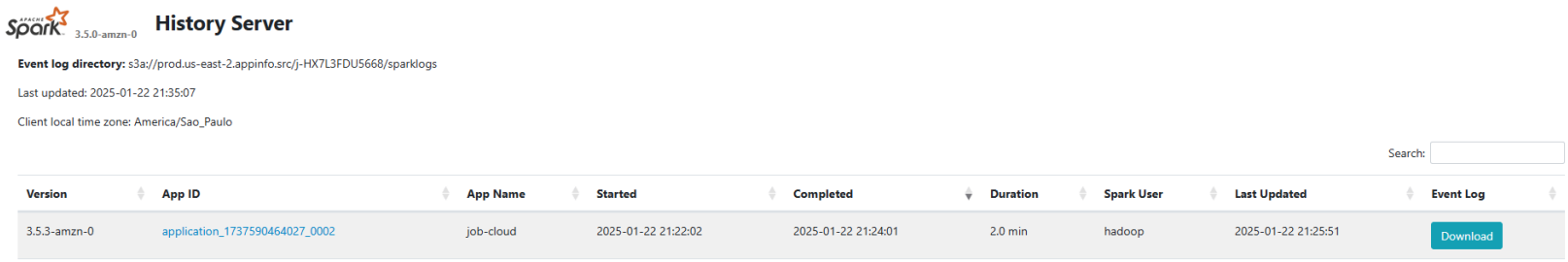


7° Monitorando a aplicação com Spark history server

O Spark History Server permite visualizar informações detalhadas sobre jobs, stages e tasks que foram executados em um cluster Spark.

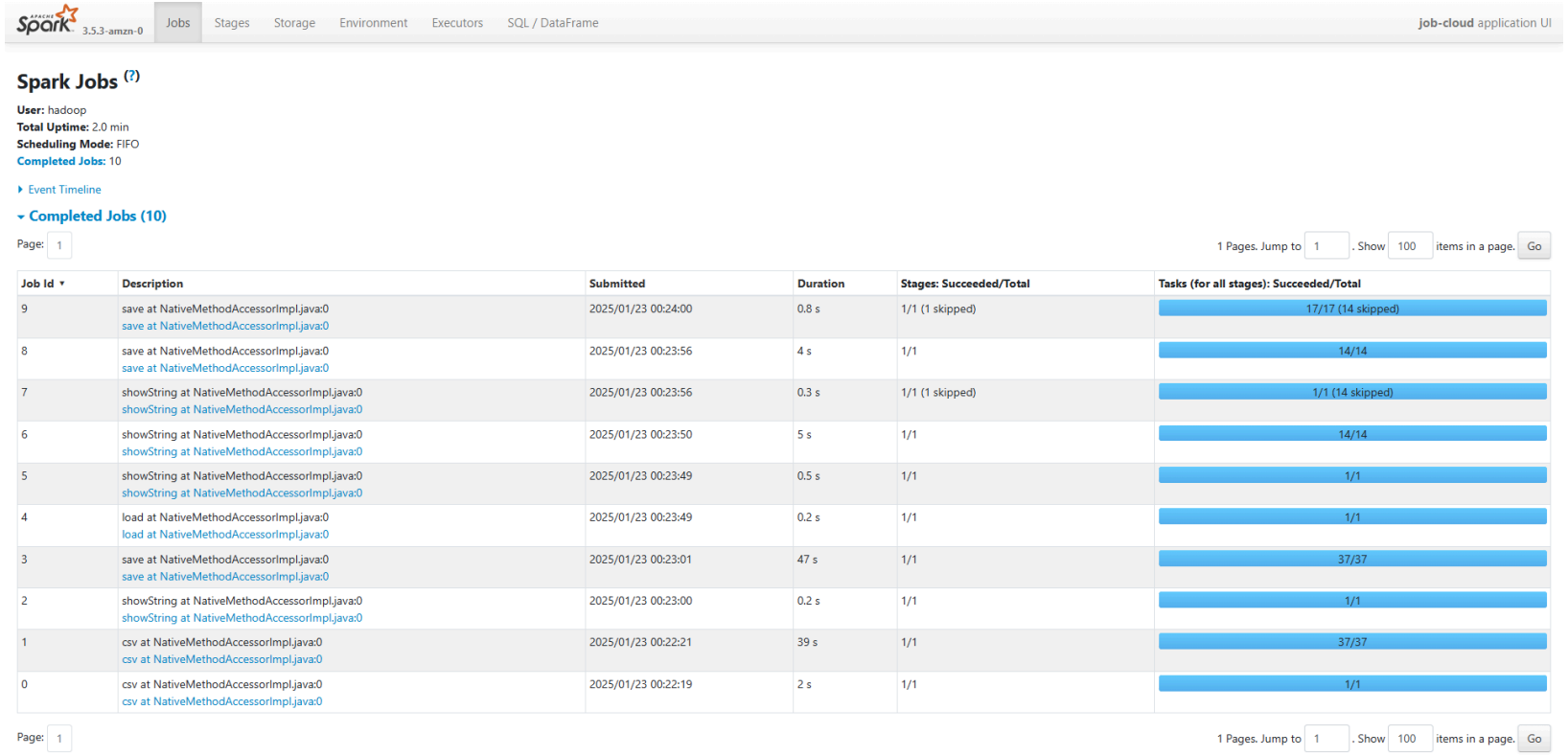
Interface web

- Já temos algumas infomações: **Nome, Inicio, Termino**, **Duração** da aplicação.
- Seleciono App ID.



Jobs

- **Mostra todos os jobs submetidos**.
- **Exibe os status**.
- **Usado para acompanhar o progresso geral da papiline**.



Spark 3.5.3-amzn-0 | Jobs | Stages | Storage | Environment | Executors | SQL / DataFrame | job-cloud application UI

Spark Jobs (?)

User: hadoop
Total Uptime: 2.0 min
Scheduling Mode: FIFO
Completed Jobs: 10

▸ Event Timeline

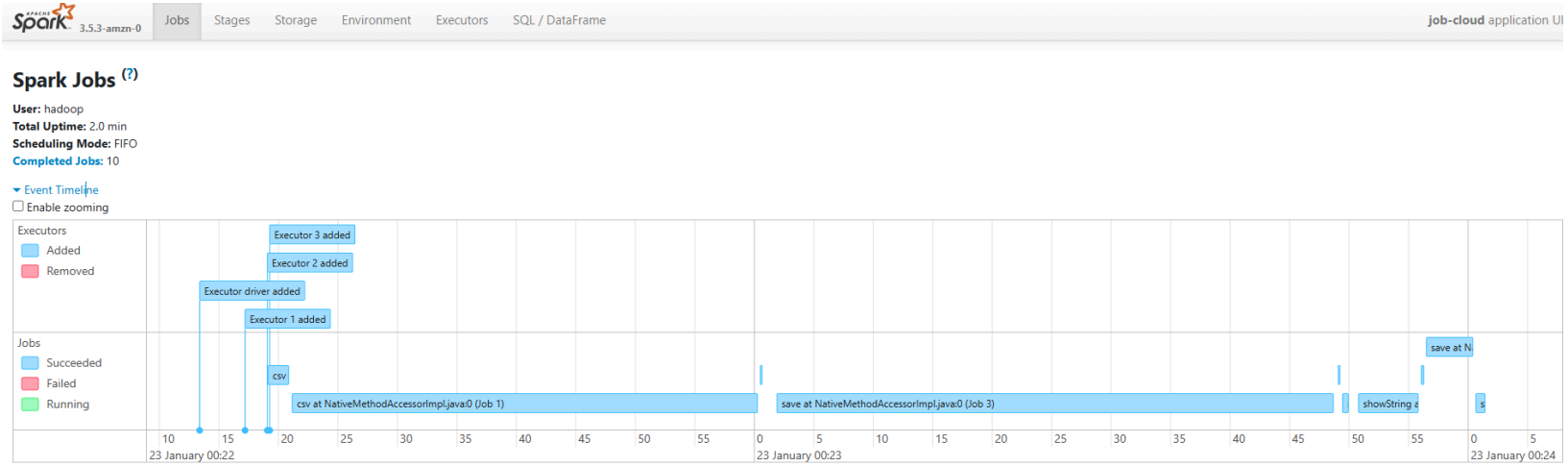
▾ **Completed Jobs (10)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
9	save at NativeMethodAccessorImpl.java:0 save at NativeMethodAccessorImpl.java:0	2025/01/23 00:24:00	0.8 s	1/1 (1 skipped)	17/17 (14 skipped)
8	save at NativeMethodAccessorImpl.java:0 save at NativeMethodAccessorImpl.java:0	2025/01/23 00:23:56	4 s	1/1	14/14
7	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2025/01/23 00:23:56	0.3 s	1/1 (1 skipped)	1/1 (14 skipped)
6	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2025/01/23 00:23:50	5 s	1/1	14/14
5	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2025/01/23 00:23:49	0.5 s	1/1	1/1
4	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2025/01/23 00:23:49	0.2 s	1/1	1/1
3	save at NativeMethodAccessorImpl.java:0 save at NativeMethodAccessorImpl.java:0	2025/01/23 00:23:01	47 s	1/1	37/37
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2025/01/23 00:23:00	0.2 s	1/1	1/1
1	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2025/01/23 00:22:21	39 s	1/1	37/37
0	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2025/01/23 00:22:19	2 s	1/1	1/1

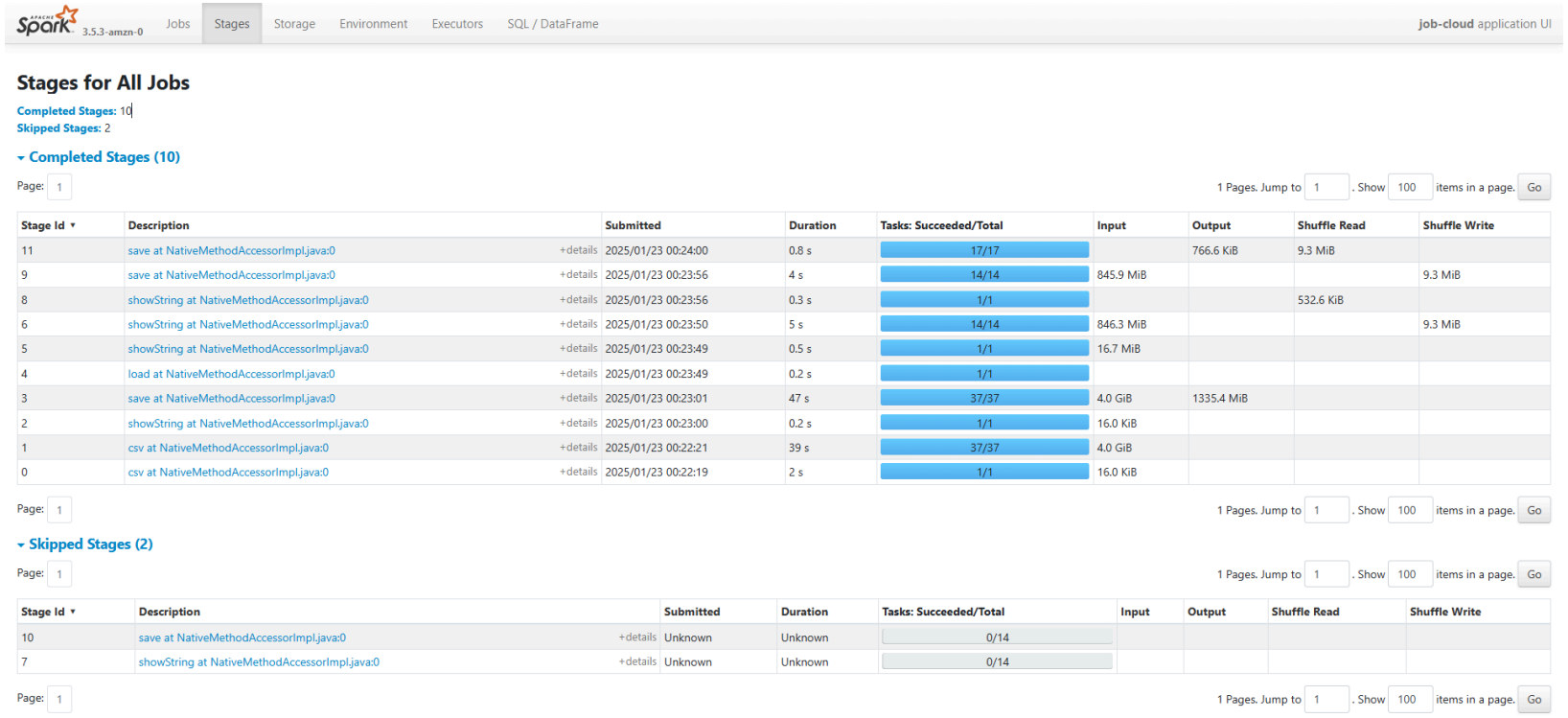
Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Time Line Jobs



Stages

- **Divide os Jobs em Stages, que são conjuntos de tarefas**.
- **Ajuda a identificar gargalos no processamento**.



Spark 3.5.3-amzn-0 Jobs Stages Storage Environment Executors SQL / DataFrame job-cloud application UI

Stages for All Jobs

Completed Stages: 10
Skipped Stages: 2

Completed Stages (10)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
11	save at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:24:00	0.8 s	17/17		766.6 KiB	9.3 MiB	
9	save at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:23:56	4 s	14/14	845.9 MiB			9.3 MiB
8	showString at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:23:56	0.3 s	1/1			532.6 KiB	
6	showString at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:23:50	5 s	14/14	846.3 MiB			9.3 MiB
5	showString at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:23:49	0.5 s	1/1	16.7 MiB			
4	load at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:23:49	0.2 s	1/1				
3	save at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:23:01	47 s	37/37	4.0 GiB	1335.4 MiB		
2	showString at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:23:00	0.2 s	1/1	16.0 KiB			
1	csv at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:22:21	39 s	37/37	4.0 GiB			
0	csv at NativeMethodAccessorImpl.java:0 +details	2025/01/23 00:22:19	2 s	1/1	16.0 KiB			

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Skipped Stages (2)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
10	save at NativeMethodAccessorImpl.java:0 +details	Unknown	Unknown	0/14				
7	showString at NativeMethodAccessorImpl.java:0 +details	Unknown	Unknown	0/14				

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Executors

- **Mostra quantos executores estão ativos, consumo de CPU, memória e tempo de execução**.

Executors

Show 20 entries — Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Add Time	Remove Time
driver	ip-172-31-9-210.us-east-2.compute.internal:38571	Active	0	0.0 B / 1 GiB	0.0 B	0	0	0	0	0	2.0 min (0.0 ms)	0.0 B	0.0 B	0.0 B		2025-01-22 21:22:13	-
1	ip-172-31-5-20.us-east-2.compute.internal:35193	Active	0	0.0 B / 4.8 GiB	0.0 B	4	0	0	39	39	5.9 min (5 s)	3.3 GiB	2.1 MiB	5.4 MiB	stderr stdout	2025-01-22 21:22:17	-
2	ip-172-31-13-122.us-east-2.compute.internal:39635	Active	0	0.0 B / 4.8 GiB	0.0 B	4	0	0	43	43	5.9 min (4 s)	3.4 GiB	3.1 MiB	7.9 MiB	stderr stdout	2025-01-22 21:22:19	-
3	ip-172-31-0-18.us-east-2.compute.internal:42957	Active	0	0.0 B / 4.8 GiB	0.0 B	4	0	0	42	42	5.8 min (4 s)	3.1 GiB	4.7 MiB	5.4 MiB	stderr stdout	2025-01-22 21:22:19	-

Showing 1 to 4 of 4 entries — Previous 1 Next

Conclusão

📂 Arquitetura de 3 Camadas (Landing, Processing e Curated)

📌 O que foi feito?

Os dados são armazenados de forma organizada em três camadas: Landing Zone: recebe os dados brutos sem transformações. Processing Zone: onde ocorrem as transformações e limpeza dos dados. Curated Zone: onde os dados tratados ficam prontos para consumo em relatórios e análises.

🎯 Por que vale a pena?

Organização e governança dos dados: evita que dados corrompidos ou errados cheguem na análise. Facilidade na auditoria e reprocessamento: se houver erro na transformação, basta refazer a partir da camada anterior. Escalabilidade: permite expandir o pipeline sem comprometer a estrutura.

☁️ Uso do Amazon S3 para Data Lake

📌 O que foi feito?

Os dados foram armazenados no Amazon S3, criando uma arquitetura flexível e econômica.

🎯 Por que vale a pena?

Baixo custo: S3 é mais barato do que armazenar em bancos de dados relacionais. Escalabilidade infinita: permite lidar com grandes volumes de dados sem precisar se preocupar com capacidade de armazenamento. Compatibilidade com diversos formatos: aceita CSV, JSON, Parquet, ORC, Avro, etc.

🚀 Processamento Distribuído com Apache Spark no EMR

📌 O que foi feito?

Criado um cluster Amazon EMR para processar os dados com PySpark. O Spark lê os dados brutos da Landing Zone, realiza transformações e grava os resultados na Processing e Curated Zone.

🎯 Por que vale a pena?

Alta performance: Spark processa terabytes de dados de forma paralela. Escalabilidade dinâmica: podemos aumentar ou reduzir o número de nós do cluster conforme a demanda. Custo sob demanda: em vez de pagar por uma infraestrutura sempre ativa, o cluster pode ser desligado após o processamento.

⚙️ Automação com AWS Step Functions

📌 O que foi feito?

Criada uma Step Function para executar o pipeline automaticamente.

🎯 Por que vale a pena?

Redução de erros manuais: evita a necessidade de rodar o pipeline manualmente. Orquestração eficiente: garante que cada etapa (ingestão, transformação, gravação) ocorra na sequência correta. Monitoramento simplificado: facilita a identificação de falhas e o reprocessamento de etapas específicas.

📉 Monitoramento com Spark History Server

📌 O que foi feito?

O Spark History Server foi ativado para visualizar logs e métricas de execução dos jobs.

🎯 Por que vale a pena?

permite ver quais tarefas falharam e por quê. Análise de performance: ajuda a otimizar o uso de memória e CPU do cluster.

Por que essa solução é eficiente?

- **Escalável:** Pode lidar com grandes volumes de dados sem impacto no desempenho.
- **Econômica:** Usa recursos sob demanda, evitando gastos desnecessários.
- **Automatizada:** Menos intervenção manual, mais confiabilidade no processamento.
- **Organizada:** Segue boas práticas de Data Lake e processamento distribuído.
- **Monitorável:** Garante que erros sejam identificados rapidamente para ajustes.