Laboratório de AWS Glue

Introdução

Processos de ETL (Extract, Transform and Load) estão presentes em todos os projetos de dados. O cenário costuma ser o mesmo: fontes de dados diversas com datasets de interesse que precisam ser ingeridos, transformados e armazenados em um ou mais destinos, com formatos diferentes da origem.

Neste laboratório você será guiado na construção de um processo de ETL simplificado utilizando o serviço AWS Glue.

1 - Preparando os dados de origem

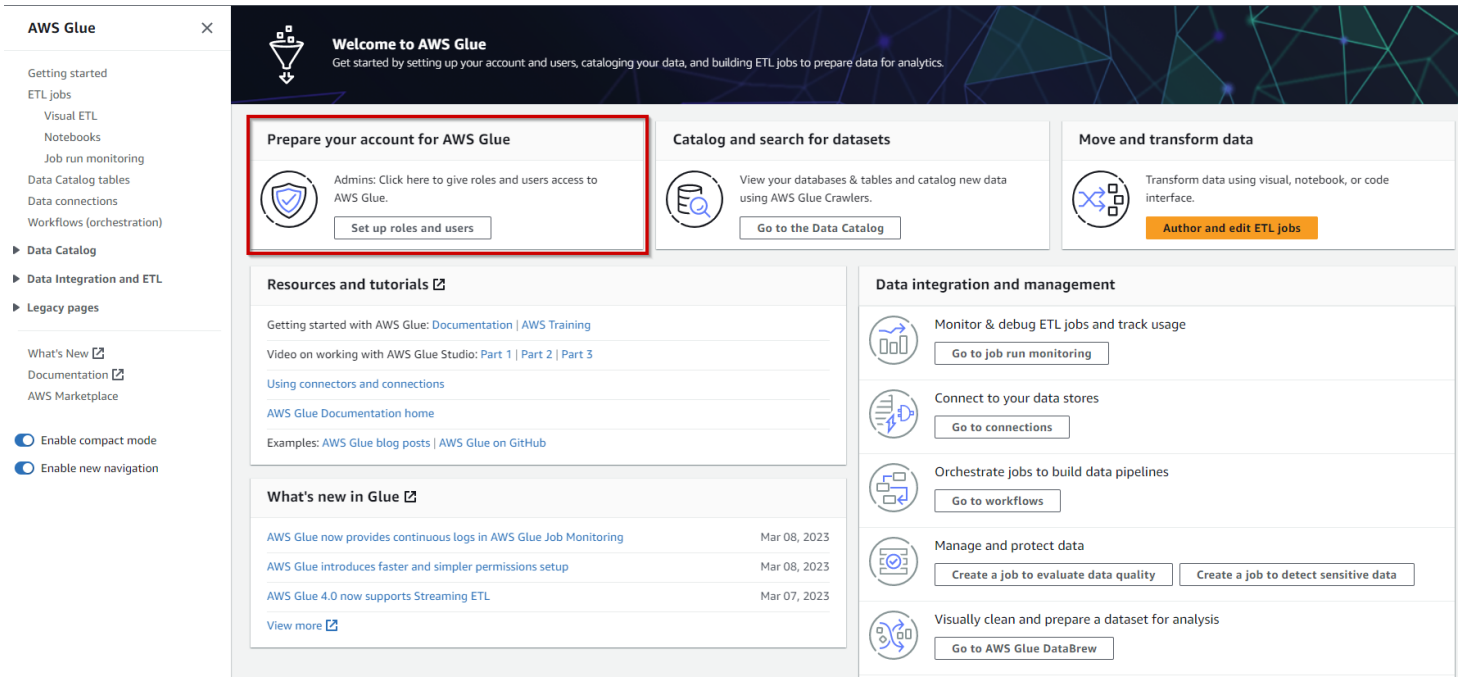
Faremos uso do arquivo *nomes.csv*, um dataset que contém os nomes mais comuns de registro de nascimento dos cartórios americanos entre os anos de 1880 e 2014. Trata-se de um arquivos CSV, com a estrutura descrita na amostra a seguir.

```
nome,sexo,total,ano
Jennifer,F,54336,1983
```

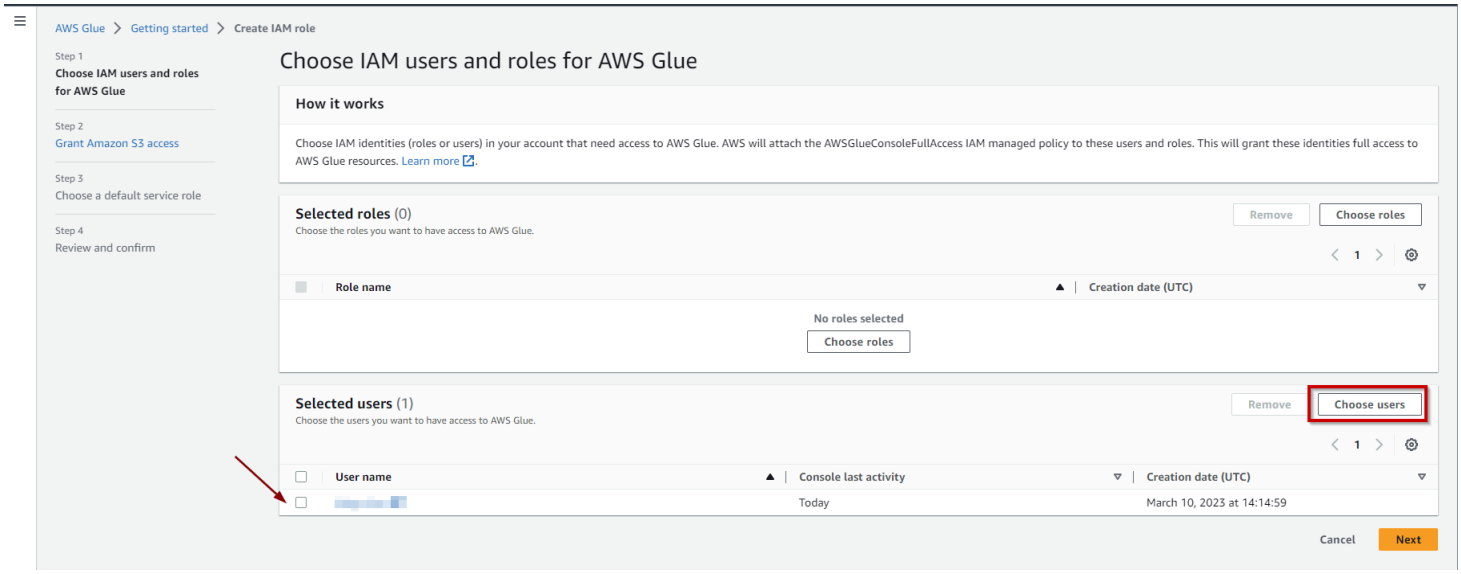
Para nosso laboratório, o arquivo deverá estar em um bucket do S3. Vamos considerar que o path do arquivo seja `s3://{BUCKET}/lab-glue/input/nomes.csv` . Lembre-se que o valor `{BUCKET}` deve ser substituído por um dos disponíveis em sua conta.

2 - Configurando sua conta para utilizar o AWS Glue

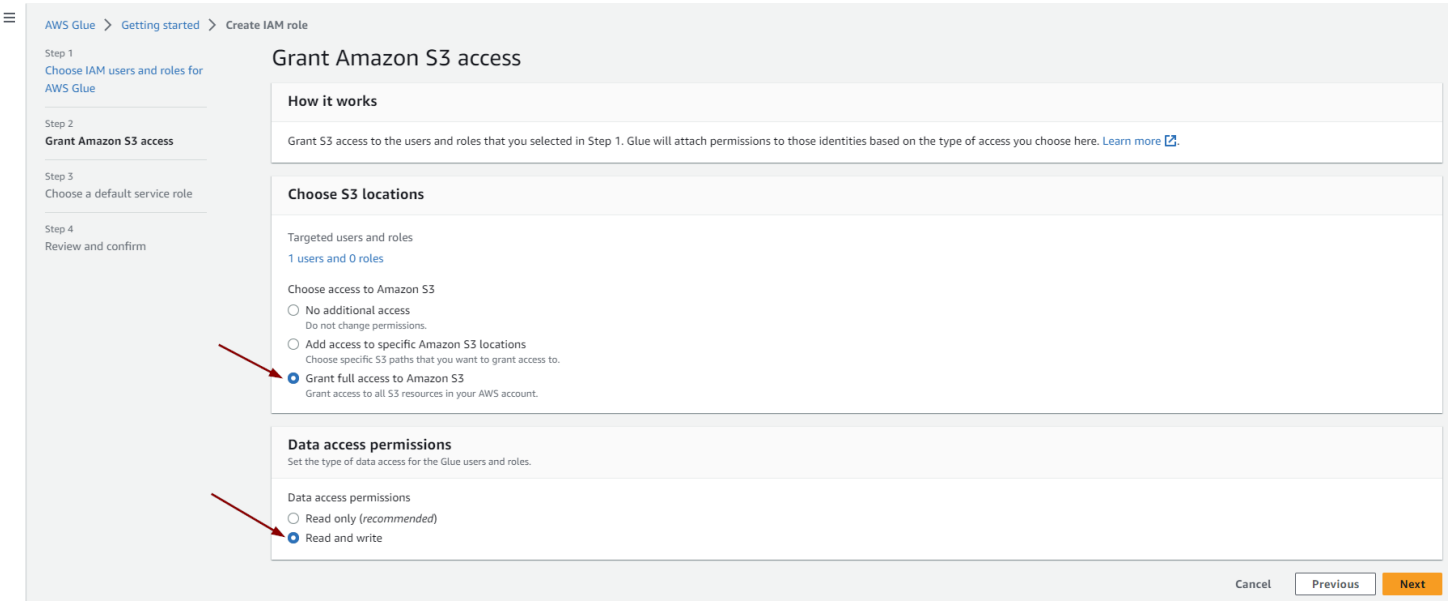
Acesse a página inicial do serviço AWS Glue. Para que possamos utilizar o serviço com as permissões necessárias, devemos seguir o passo-a-passo disponível a partir da opção **Set up roles and users** no *card* **Prepare your account for AWS Glue**.



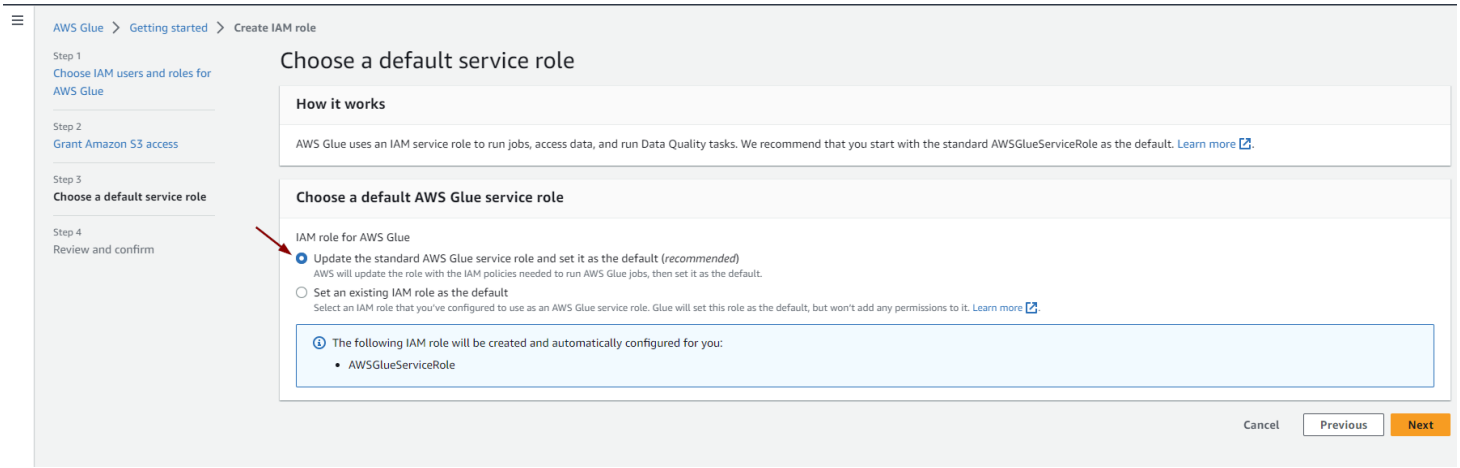
No primeiro passo devemos indicar quais *roles* e *usuários* terão acesso ao serviço AWS Glue. Procure pelo seu usuário em **Choose users** e o adicione à lista.



No passo seguinte, informe acesso total ao S3 para leitura e escrita.



Por fim, marque a opção **Update the standard AWS Glue service role and set it as the default (recommended)** e finalize o processo.



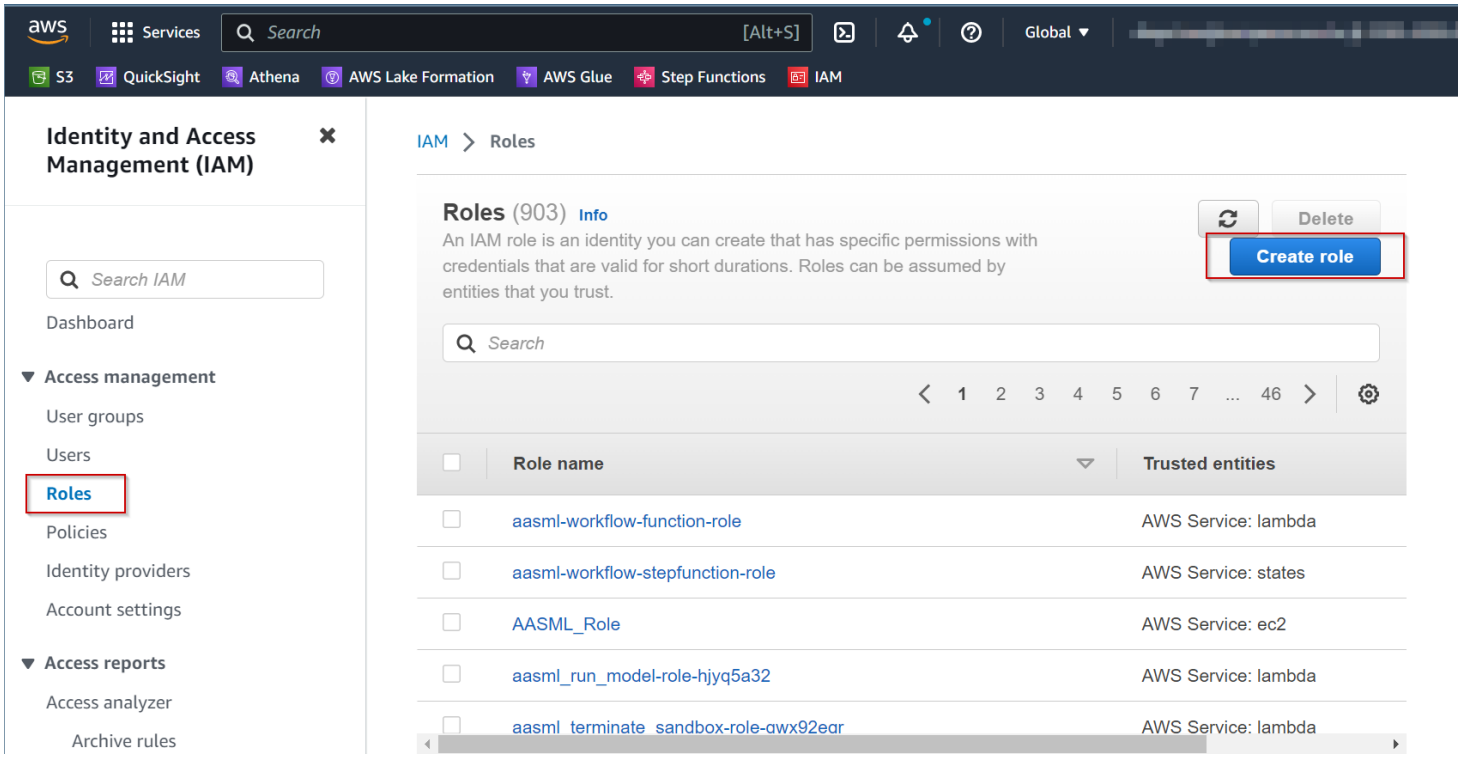
3 - Criando a IAM Role para os jobs do AWS Glue

Você deve estar lembrado que *Roles* são credenciais temporárias assumidas por serviços e aplicações para realizar operações em favor do usuário.

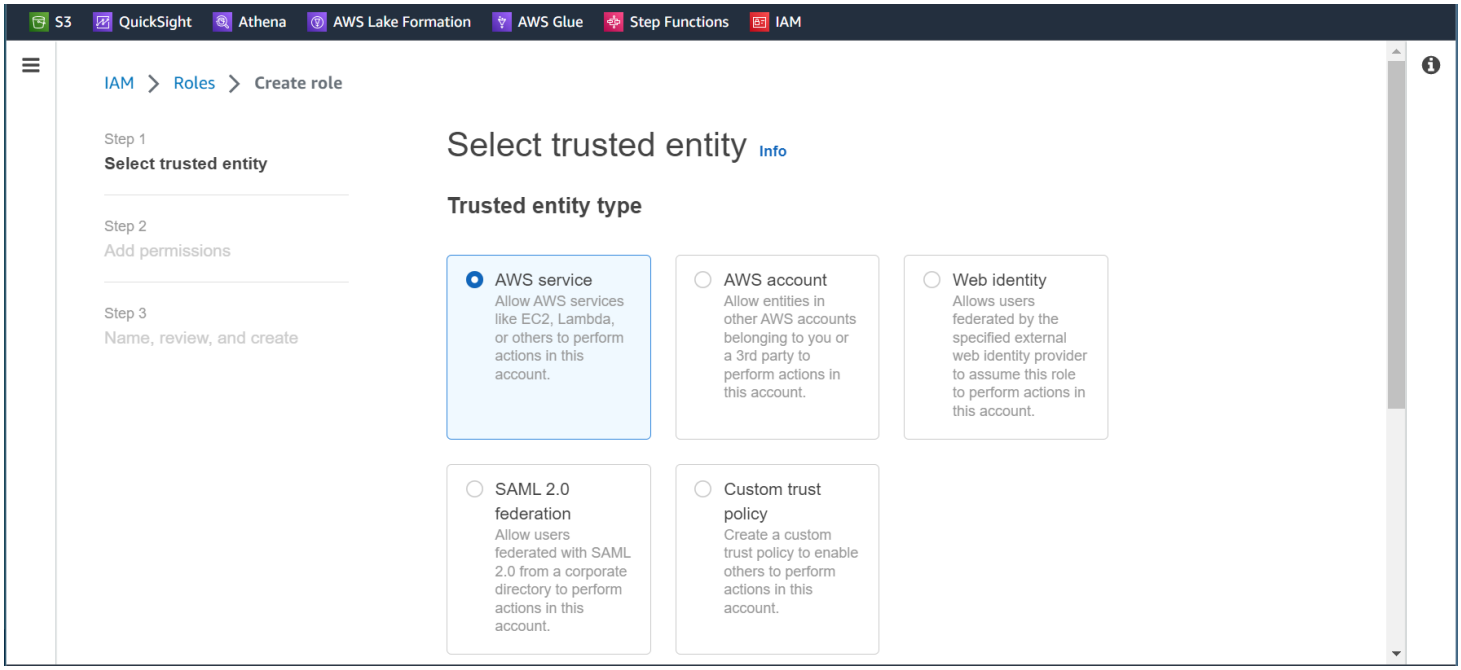
Logo, criaremos uma nova *role* chamada *AWSGlueServiceRole-Lab4*, associada a *policies* geridas pela AWS (*AmazonS3FullAccess*, *AWSLakeFormationDataAdmin*, *AWSGlueConsoleFullAccess* e *CloudWatchFullAccess*). Tais policies irão permitir acesso ao serviço do Glue ao **S3**, bem como outras ações, como executar códigos via *Notebooks*. Observe que estamos utilizando *policies* permissavas, o que vai de encontro ao princípio de privilégio mínimo que deve-se seguir em projetos reais. O objetivo aqui é simplificar o processo, apenas.

Vamos aos passos:

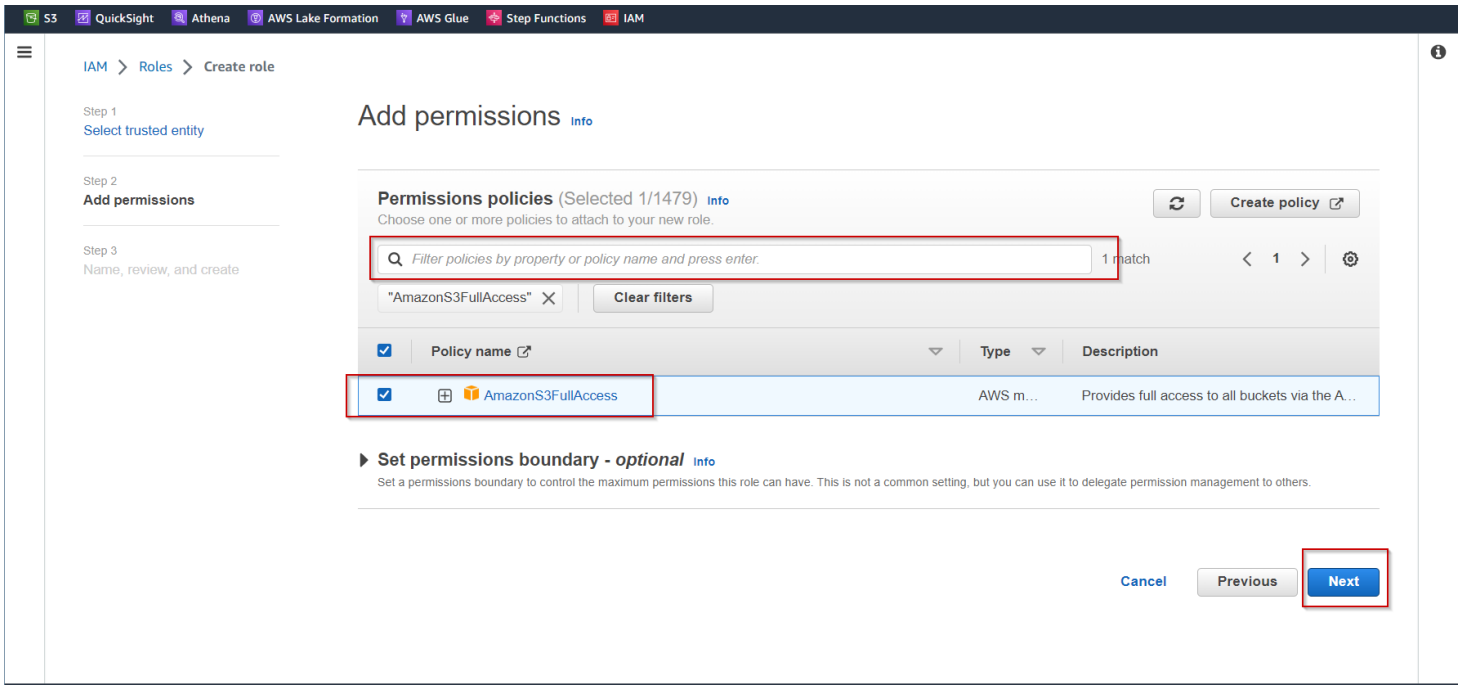
- No console, acesse a página do serviço **Identity and Access Management (IAM)** e clique no menu **Roles** à esquerda. Na sequência, clique no botão **Create Role**.



- Na primeira etapa, **Select trusted entity**, escolha *AWS Service* e para **Use Case**, infome *Glue*. Clique em **Next**.



- Na etapa **Add permissions**, pesquise por *AmazonS3FullAccess*, selecione a mesma da lista. Repita o processo para adicionar as demais *policies* necessárias: AWSLakeFormationDataAdmin_, *AWSGlueConsoleFullAccess* e *CloudWatchFullAccess*. Em seguida, clique em **Next**.

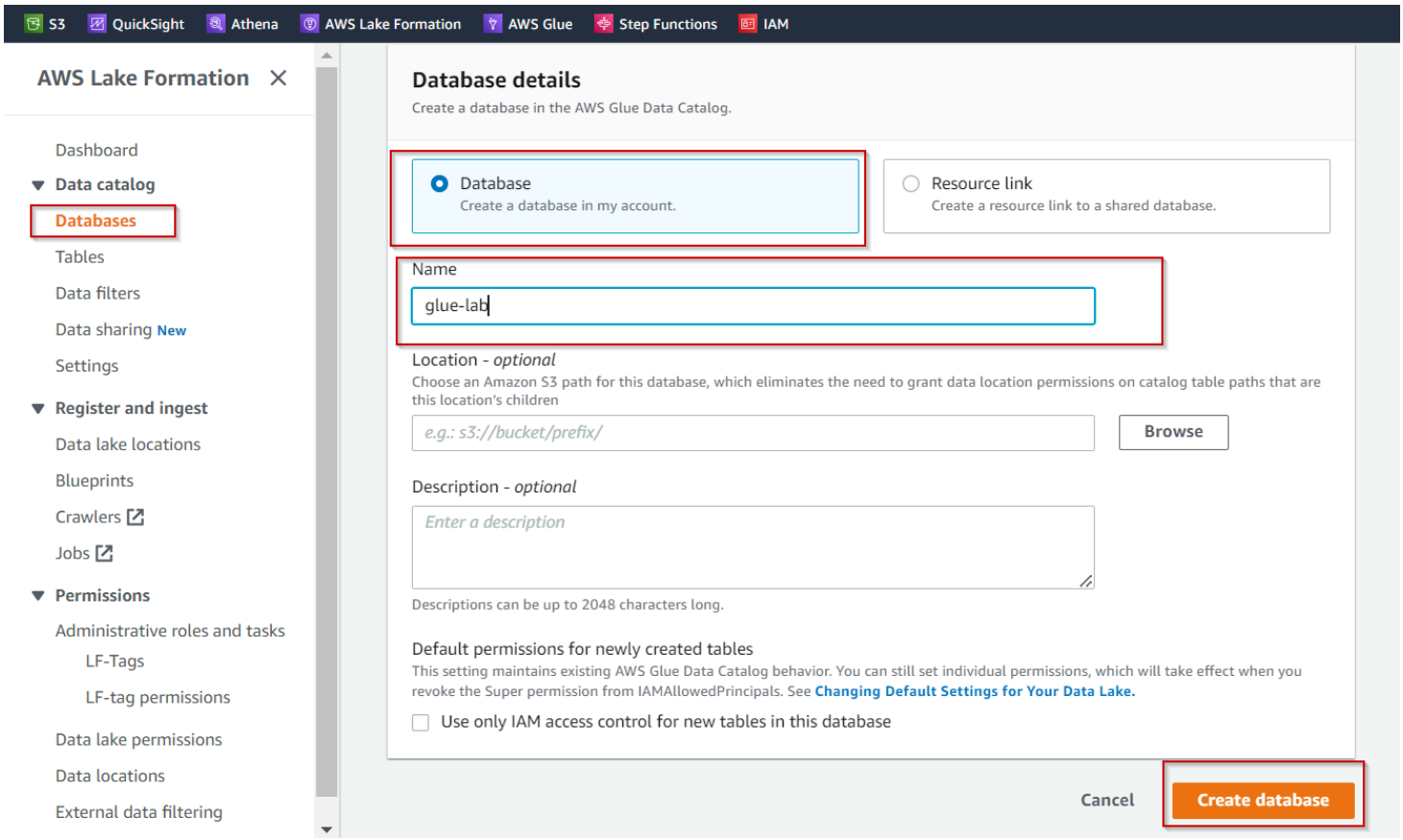


- Na última etapa, informe em **Role name** o valor *AWSGlueServiceRole-Lab4* e, para finalizar, clique em **Create Role**.

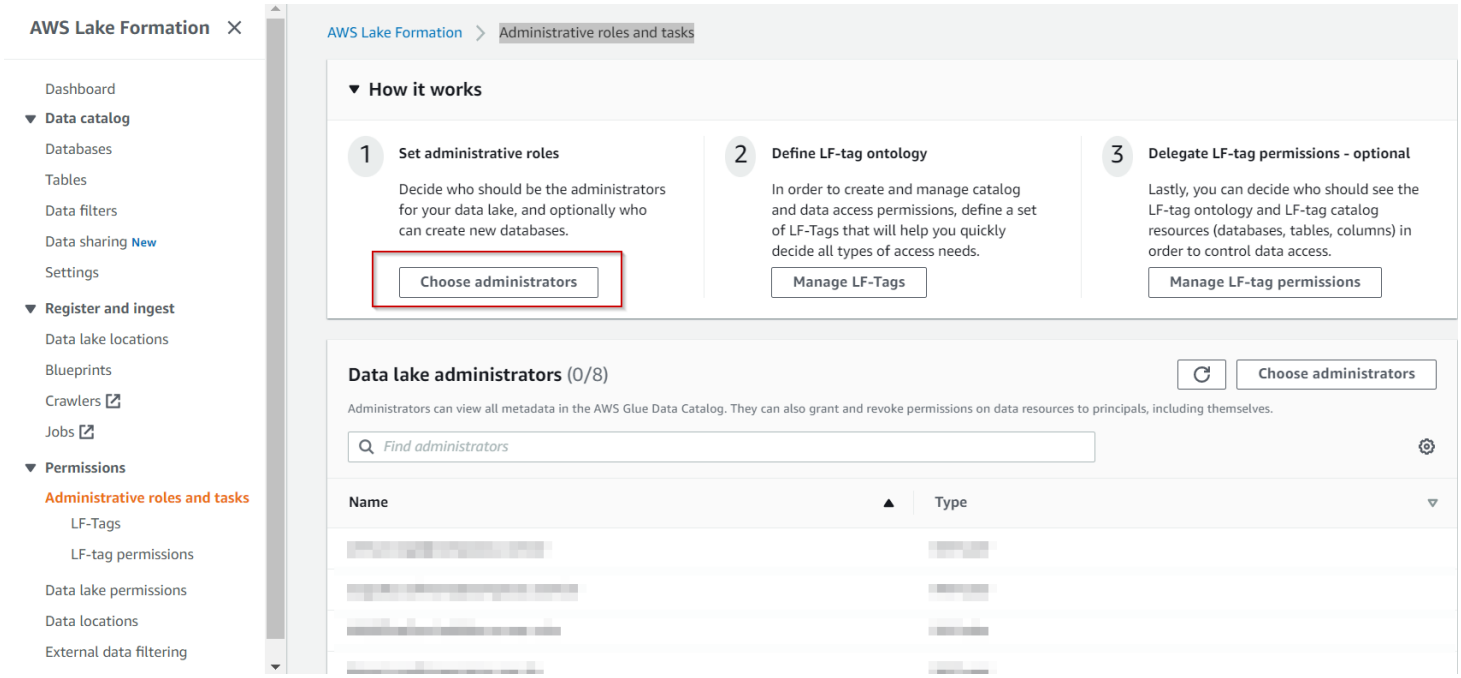
4 - Configurando as permissões no AWS Lake Formation

AWS Lake Formation é um serviço que facilita a criação e gerenciamento de *data lakes*. Nos iremos utilizá-lo para criar o banco de dados no qual nosso *crawler* irá adicionar automaticamente uma tabela a partir dos dados armazenados no S3.

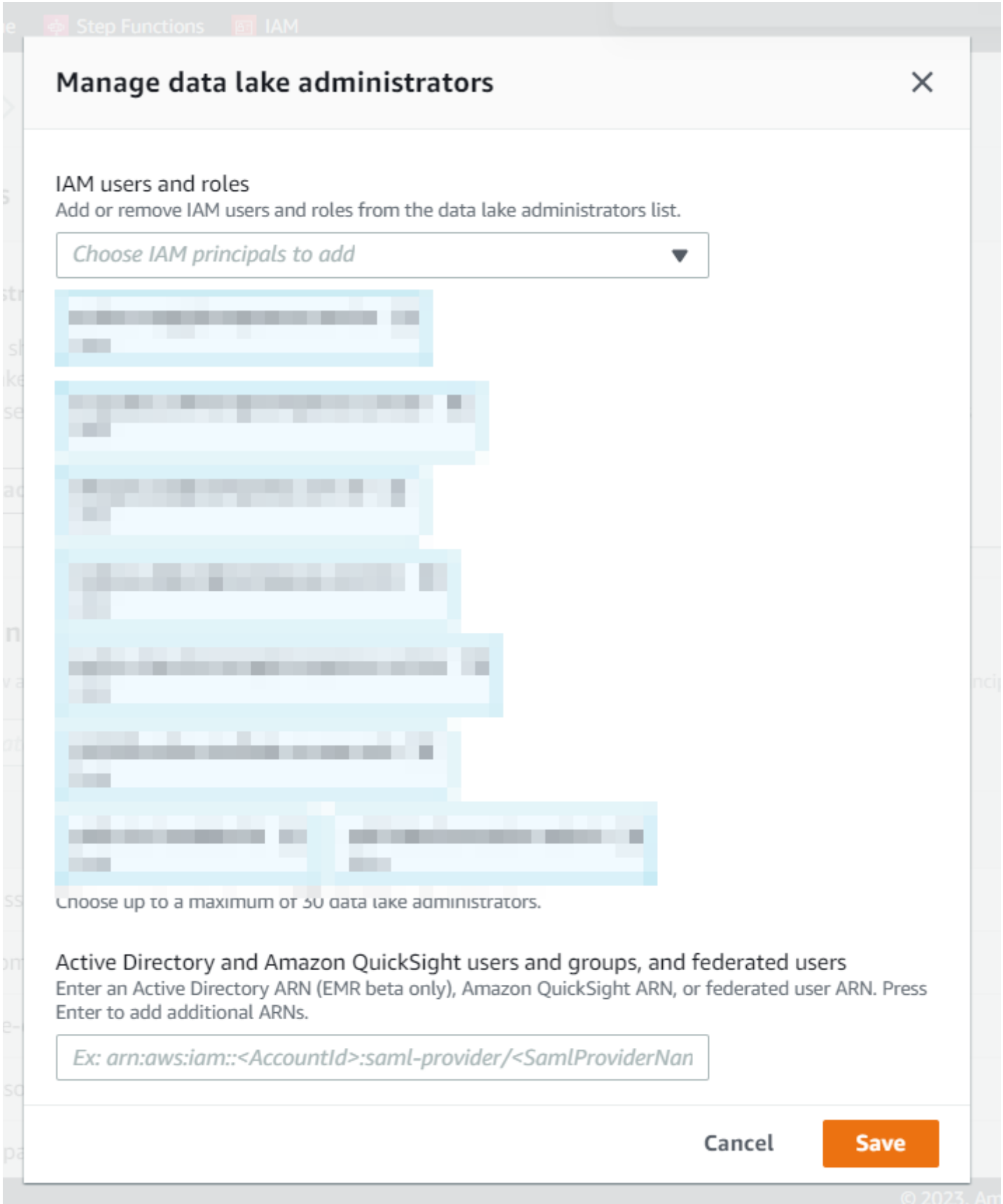
Após acessar o serviço **AWS Lake Formation** no console, clique na opção **Databases**, no menu à esquerda. Na sequência, clique no botão **Create Database**. O nome do novo banco deverá ser *glue-lab*. Observe que estamos criando um banco de dados no catálogo do Glue e não um banco de dados das características dos SGBD Relacionais.



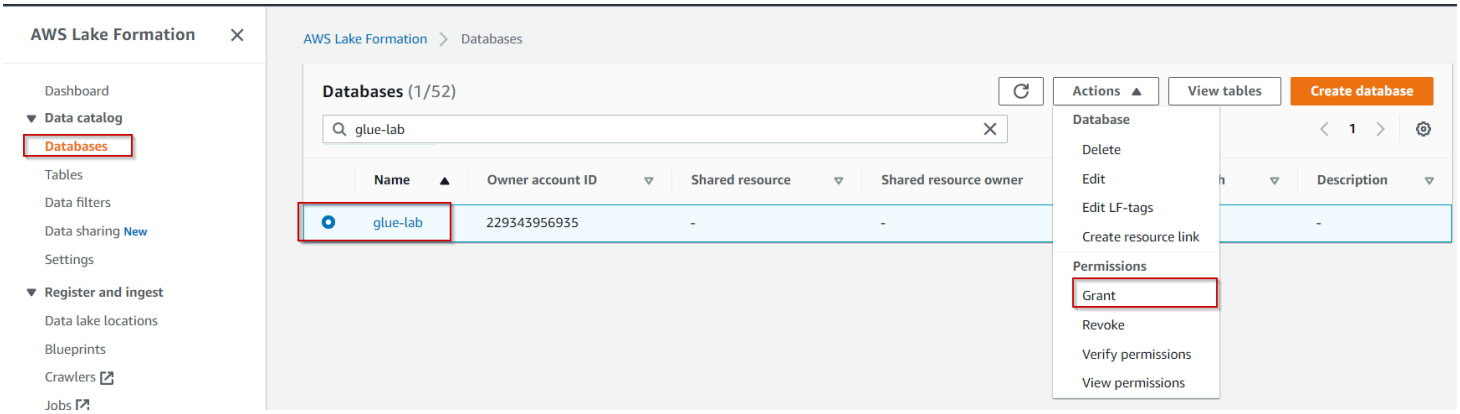
Agora precisamos que você adicione seu usuário IAM como administrador do *data lake*. Para tal, acesse a opção **Administrative roles and tasks** no menu à esquerda. Na tela que se apresenta, clique em **Choose administrators**.



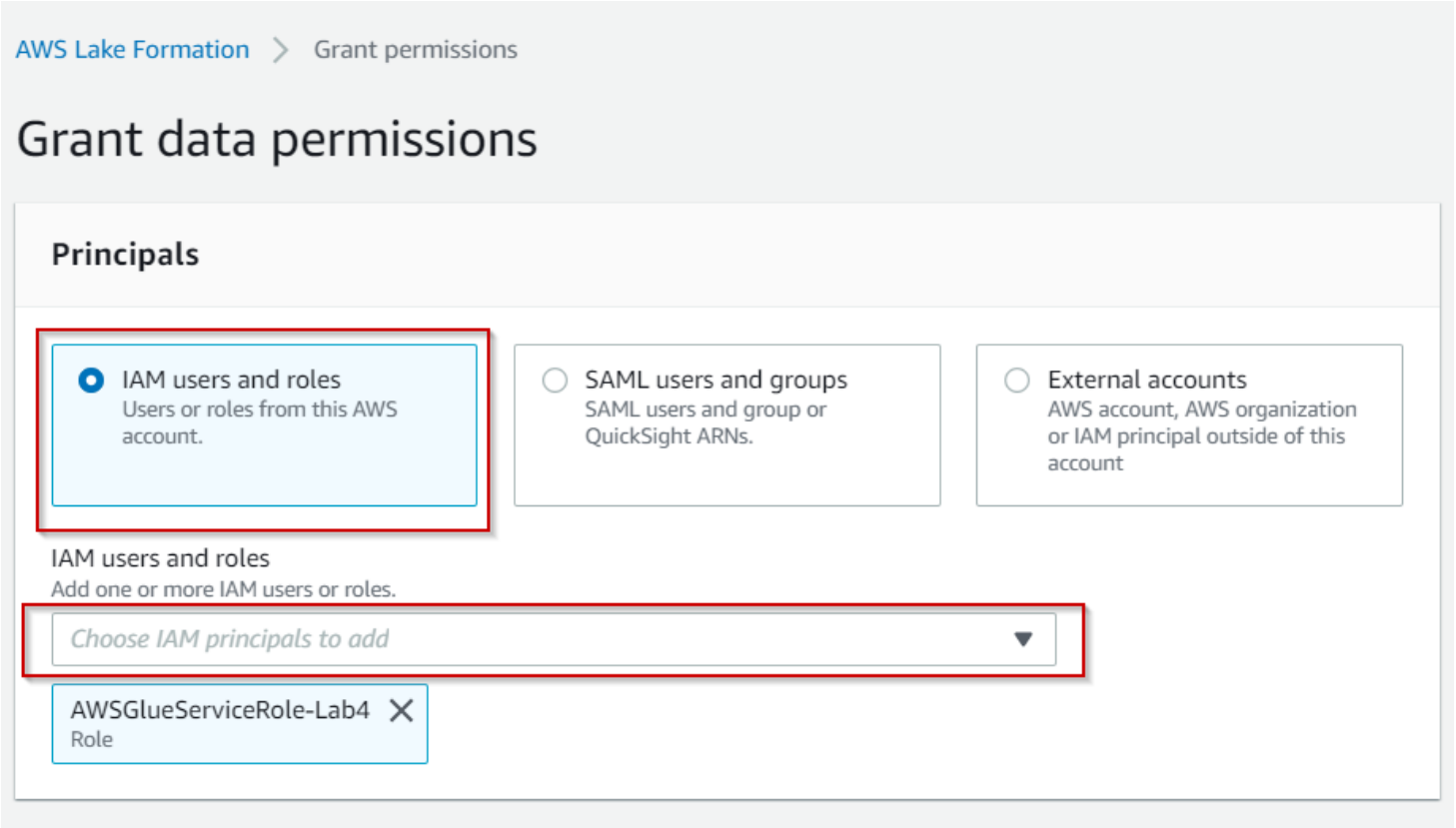
Procure pelo seu usuário em **IAM User and roles**. Adicione-o à lista e clique em **Save**.



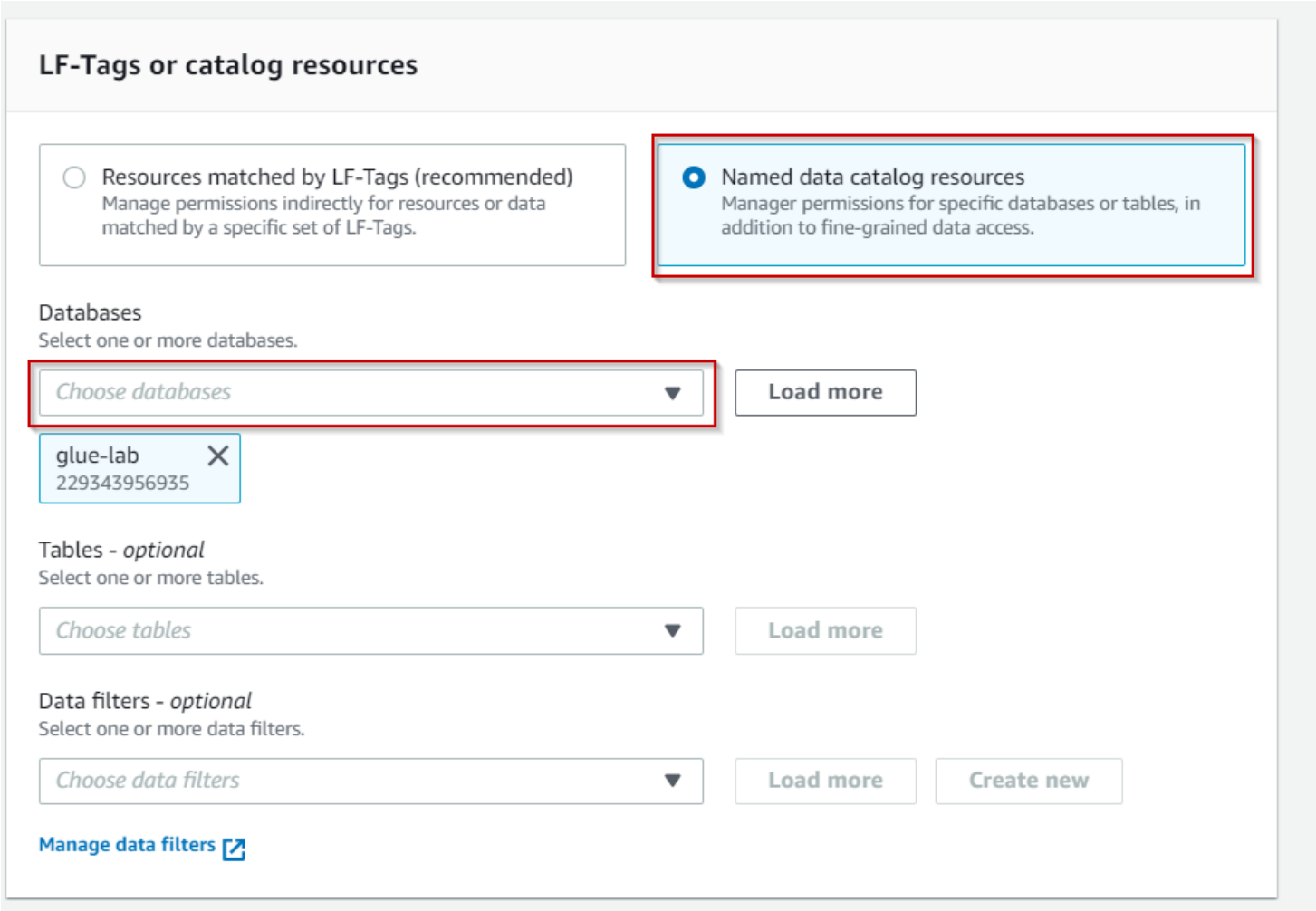
Retorne ao menu **Databases**, busque pelo banco *glue-lab* criado anteriormente. Selecione-o e vá em **Actions**, escolhendo a opção **Grant**



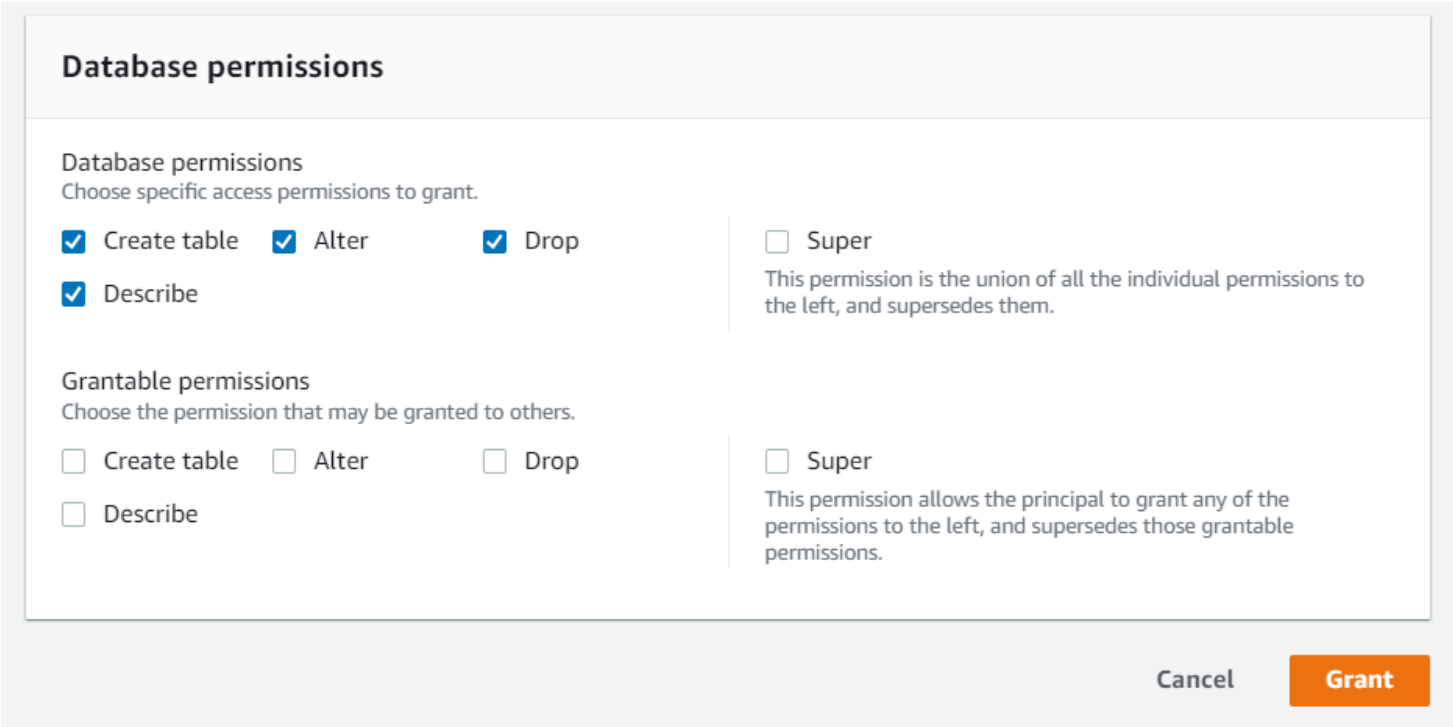
Vamos conceder privilégios para a role do IAM criada anteriormente (*AWSGlueServiceRole-Lab4*). Para tal, escolha a opção **IAM users and roles** na seção **Principals**. Selecione a role a partir da lista apresentada.



Na seção **LF-Tags or catalog resources**, escolha **Named data catalog resources**, procurando pela base *glue-lab* em **Databases**.



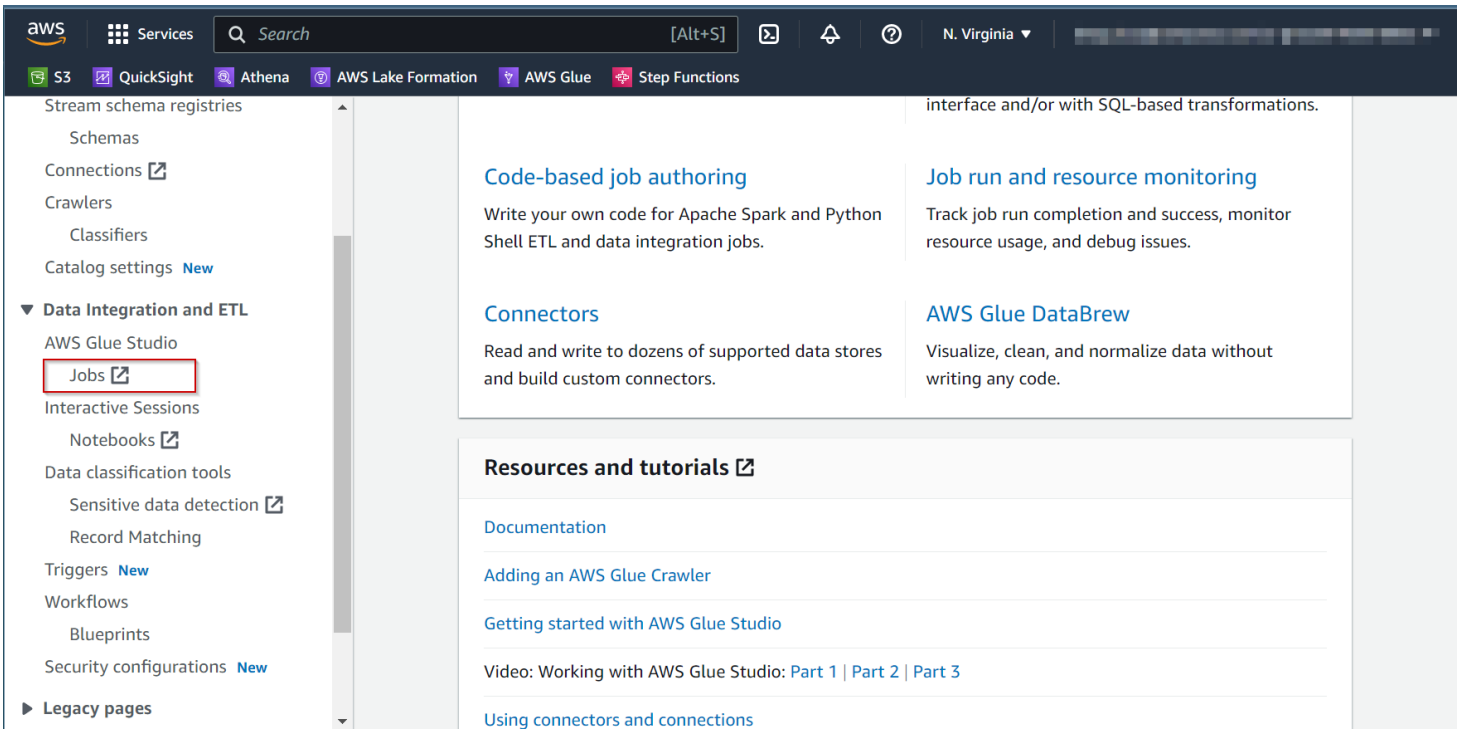
E, finalmente, na seção **Database permissions**, em **Database permissions**, escolha as opções **Create table**, **Alter**, **Drop** e **Describe**. Para finalizar, clique em **Grant**.



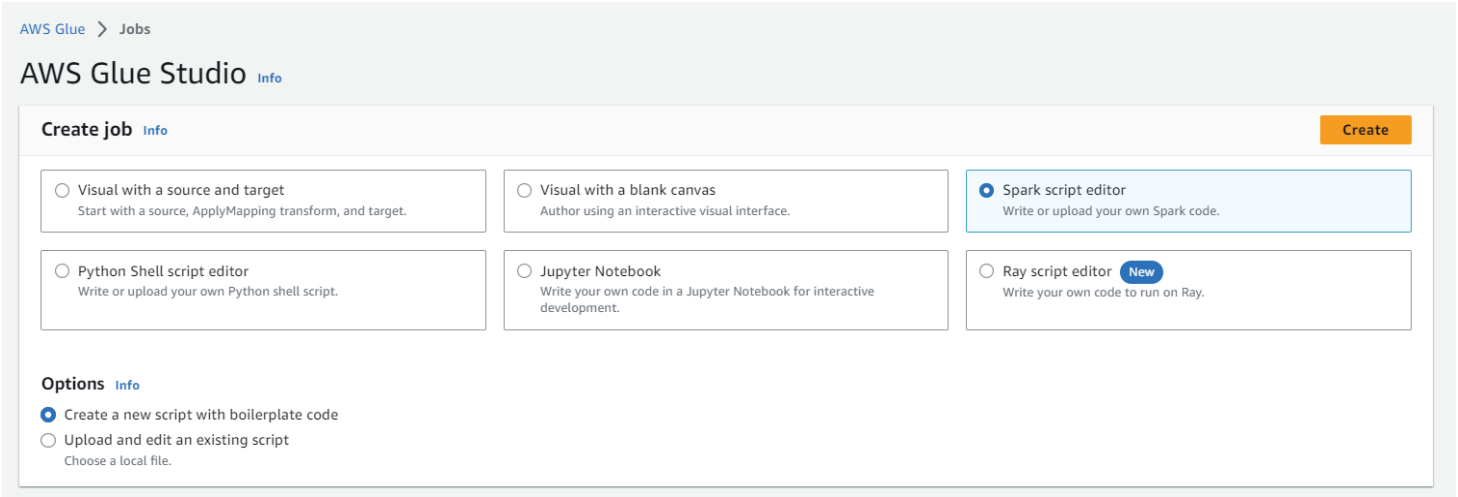
5 - Criando novo job no AWS Glue

Para realizar o processamento do arquivo *nomes.csv* iremos criar um *job* através do serviço **AWS Glue**.

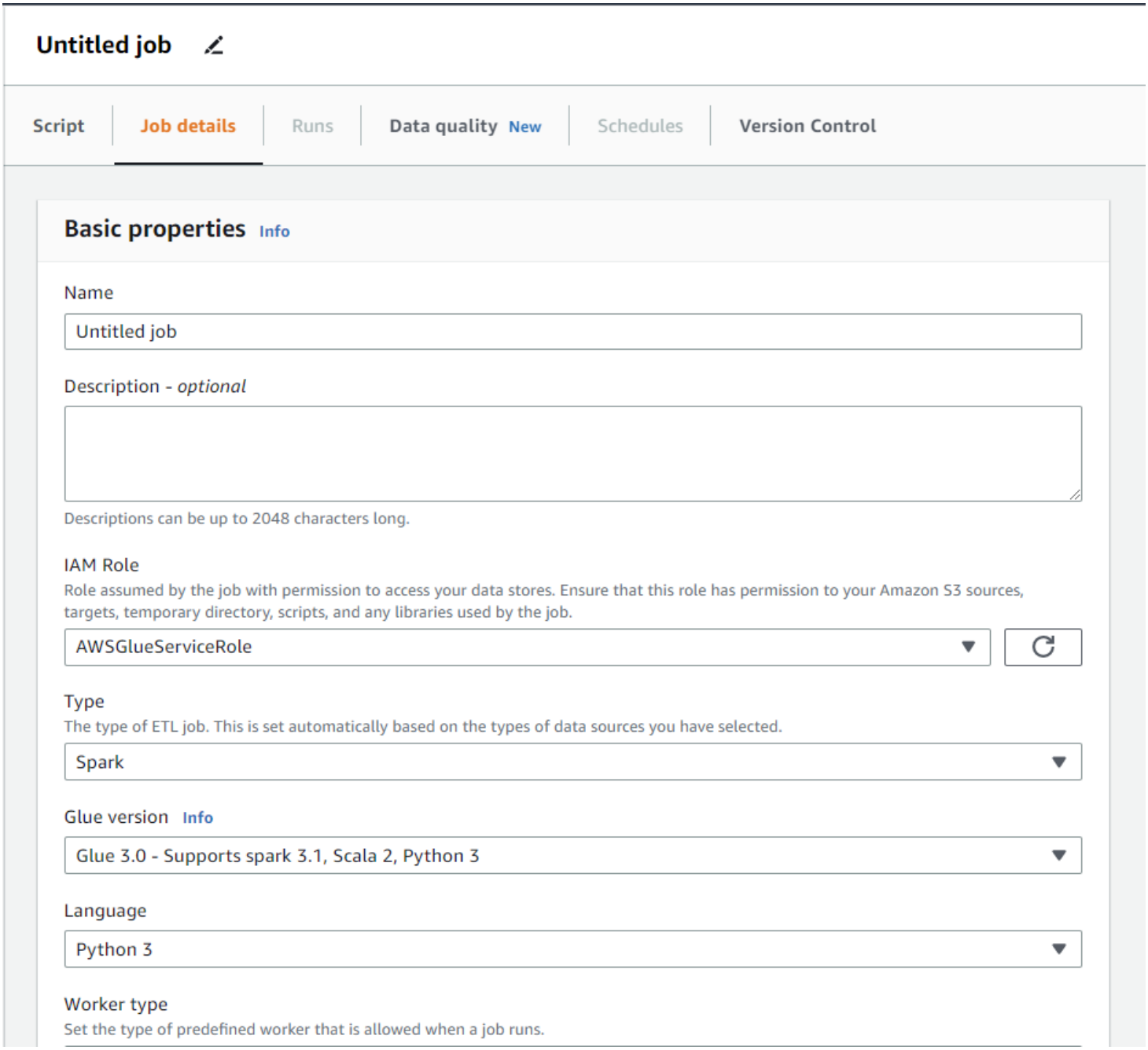
Após acessar a página inicial na console, busque pela opção *Job* no menu à esquerda.



Você perceberá que existem diferentes opções para criarmos um *job*. Em nosso laboratório, faremos uso da opção **Spark script editor**, escolhendo a alternativa **Create a new script with boilerplate code**. Após, clique em **Create**.



Na sequência você estará na tela de edição do código e das configurações do seu *job*. Iniciaremos pela configuração, através da aba **Job details**.



As propriedades que iremos configurar serão:

- *Name*: Corresponde ao nome do job. Informe *job_aws_glue_lab_4*.
- *IAM Role*: Informe *AWSGlueServiceRole-Lab4*.
- *Type*: Mantenha *Spark*.
- *Glue version*: Manhanha *Glue 3*
- *Language*: Python 3
- *Worker Type*: Escolha *G 1x*, ou seja, a opção com menos vCPUs e RAM.
- A opção *Automatically scale the number of workers* deve estar desmarcada.
- *Requested number of workers*: Informe *2*.
- *Number of retries*: Informe *0*.
- *Job timeout (minutes)*: Informe *5*.

Em **Advanced properties**, informe:

- *Script filename*: Defina o nome do seu script.
- *Spark UI*: Desmarque a opção.

Agora você pode clicar em **Save**.

Você deve ter percebido que na aba **Script** há um código base para você iniciar o desenvolvimento. O código é semelhante a este:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
job.commit()
```

Perceba que o código já oferece um objeto de sessão do Spark (`spark = glueContext.spark_session`) que você pode utilizar para realizar as atividades propostas na sequência.

Todo código que você construir, deverá estar entre os comandos `job.init(args['JOB_NAME'], args)` e `job.commit()` .

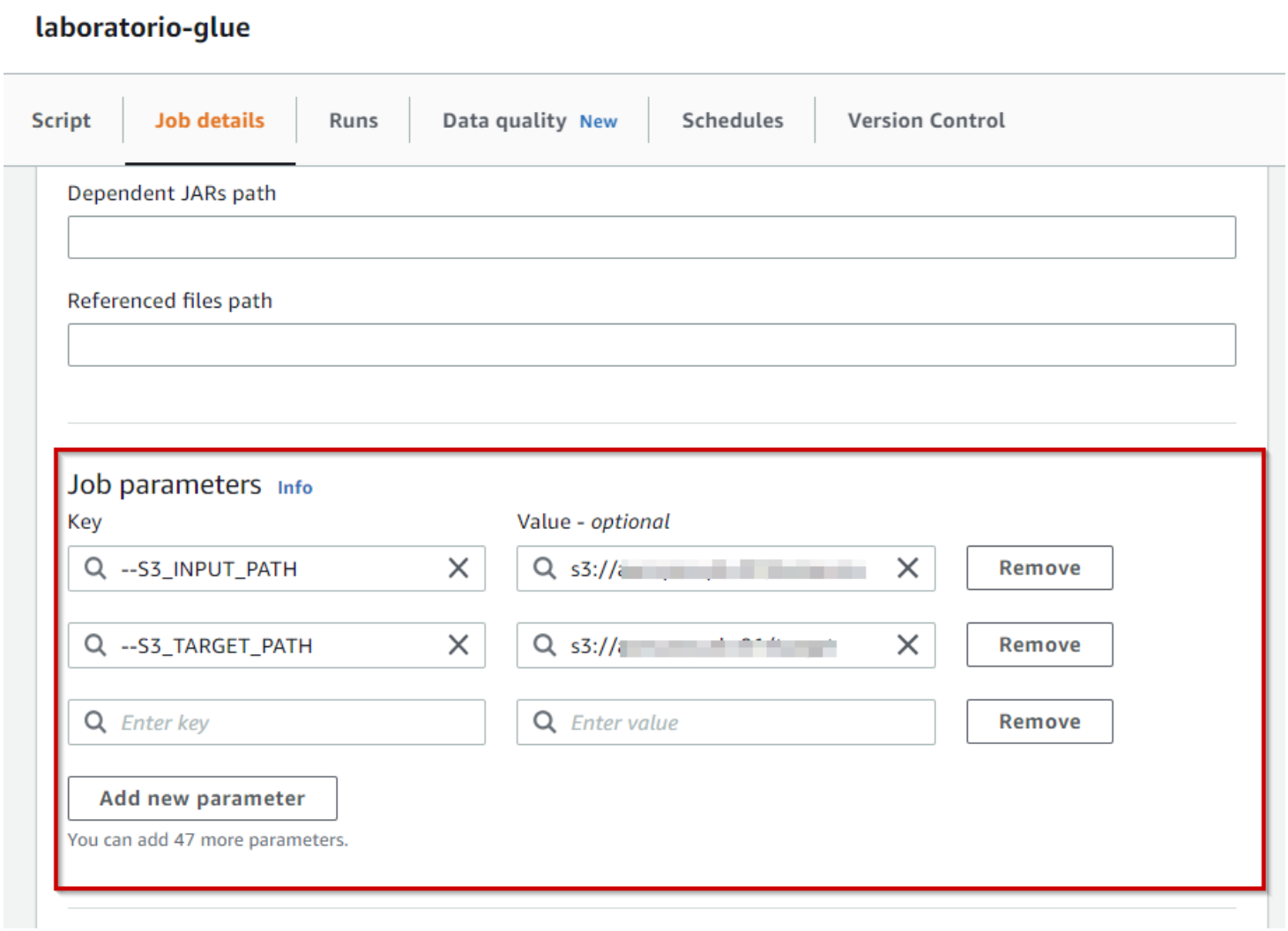
Vamos imaginar que o objetivo seja ler um arquivo CSV do S3, filtrar os dados pelo ano de 1934 e armazenar o resultado para PARQUET, em outro local do S3. O código a seguir realiza justamente tal tarefa.

Para abordar parâmetros, vamos considerar a existência de 2 neste *job*:

- **S3_INPUT_PATH**: Indica nosso caminho do origem no S3.
- **S3_TARGET_PATH**: Indica nosso caminho de destino no S3.

Os parâmetros são utilizados para tornar o código flexível, genérico. Este é sempre um ponto importante a considerar. Você pode criá-los na aba *Job Details*, opção *Advanced Options*, *Job*

parameters. Eles devem iniciar com --.



Veja o código de exemplo:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME','S3_INPUT_PATH','S3_TARGET_PATH'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

source_file = args['S3_INPUT_PATH']
target_path = args['S3_TARGET_PATH']

df = glueContext.create_dynamic_frame.from_options(
    "s3",
    {
        "paths": [
            source_file
        ]
    },
    "csv",
    {"withHeader": True, "separator":"|"},
    )


only_1934 = df.filter(lambda row: row['anoLancamento']=='1934')


glueContext.write_dynamic_frame.from_options(
       frame = only_1934 ,
       connection_type = "s3",
       connection_options = {"path": target_path},
       format = "parquet")



job.commit()
```

No exemplo estamos utilizando *dynamic_frames*, um abstração sobre um dataframe Spark oferecida pelo Glue. Naturalmente nós podemos alternar entre *dynamic frames* e *dataframes* conforme demonstramos no exemplo que segue.

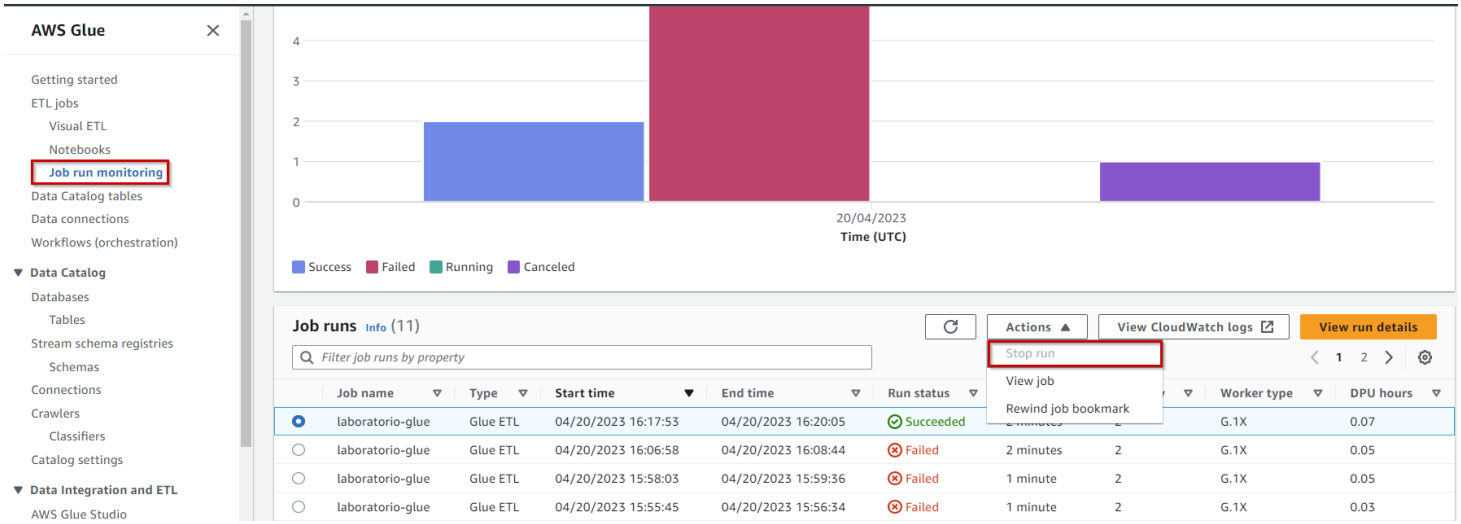
```
#Obtendo um dataframe Spark a partir de um dataframe dinâmico do Glue
spark_df = my_dynamic_df.toDF()

#Obtendo um dataframe dinâmico do Glue a partir de um dataframe Spark
dynamic_df = DynamicFrame(spark_df, glueContext)
```

Para mais informações sobre o desenvolvimento de *jobs* ETL com Glue, você pode acessar o endereço Program AWS Glue ETL scripts in PySpark.

5.1 - Eliminando execuções de jobs

Após executar *jobs*, devemos nos certificar que não hajam sessões em execução desnecessárias. Para tal, acesse a opção **Job run monitoring** no menu à esquerda do Console.

Caso hajam execuções em andamento e você não precisa mais delas, solicite a finalização da mesmas. O processo é simples e compreende escolher a execução e ir até o botão **Actions**, opção **Stop run**.



5.2 Sua vez!

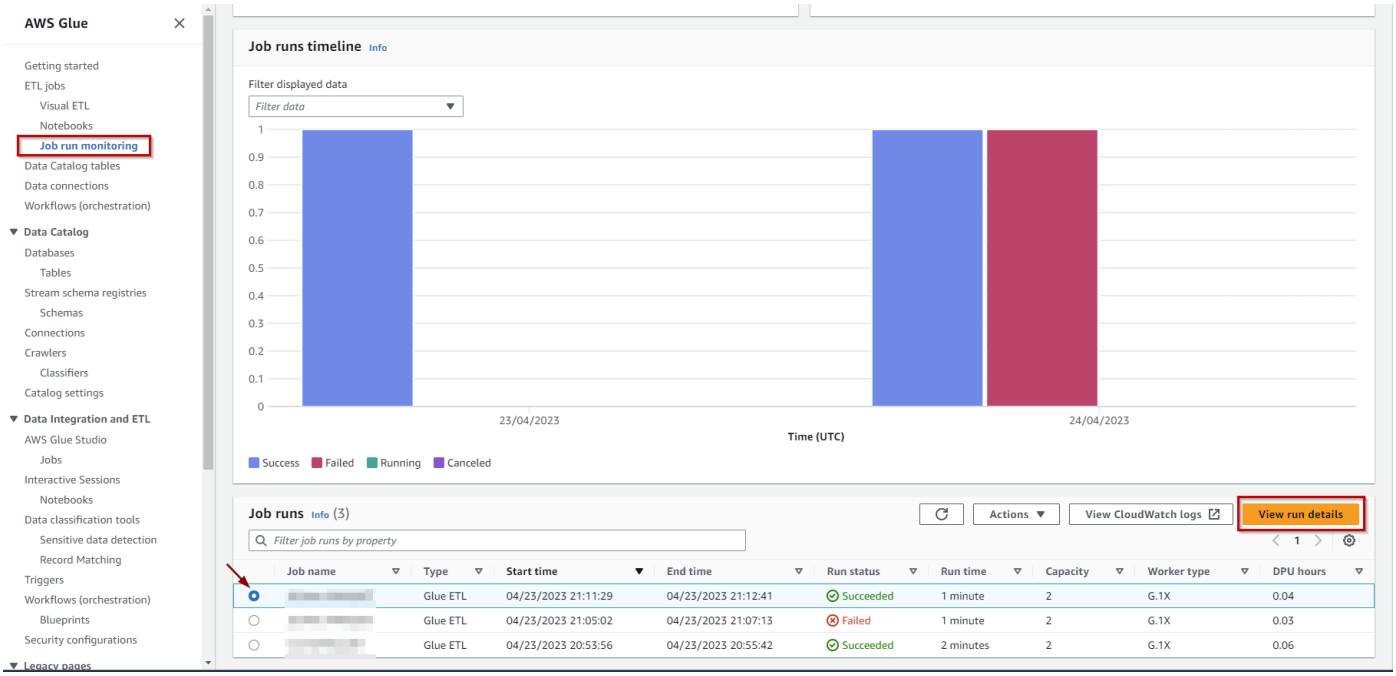
Agora vamos construir um *job* Glue nos moldes dos exemplos anteriores. Seguem os passos para você desenvolver:

- Ler o arquivo *nomes.csv* no S3 (lembre-se de realizar upload do arquivo antes).
- Imprima o schema do dataframe gerado no passo anterior.
- Escrever o código necessário para alterar a caixa dos valores da coluna *nome* para MAIÚSCULO.
- Imprimir a contagem de linhas presentes no dataframe.

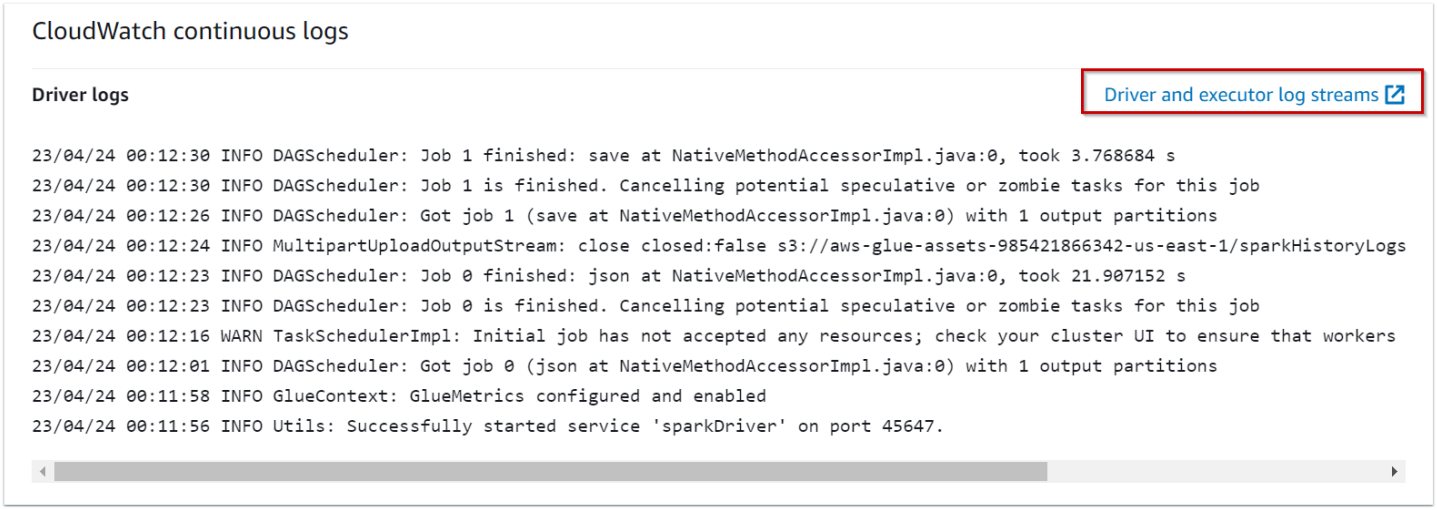
- Imprimir a contagem de nomes, agrupando os dados do dataframe pelas colunas *ano* e *sexo*. Ordene os dados de modo que o *ano* mais recente apareça como primeiro registro do dataframe.
- Apresentar qual foi o nome feminino com mais registros e em que ano ocorreu.
- Apresentar qual foi o nome masculino com mais registros e em que ano ocorreu.
- Apresentar o total de registros (masculinos e femininos) para cada ano presente no dataframe. Considere apenas as primeiras 10 linhas, ordenadas pelo ano, de forma crescente.
- Escrever o conteúdo do dataframe com os valores de nome em maiúsculo no S3.
    - Atenção aos requisitos:
        - A gravação deve ocorrer no subdiretório *frequencia_registro_nomes_eua* do path s3://<BUCKET>/lab-glue/
        - O formato deve ser JSON
        - O particionamento deverá ser realizado pelas colunas *sexo* e *ano* (nesta ordem)

Algumas dicas:

- Os logs de execução estarão disponíveis através do menu **Job run monitoring**, opção **View run details**.



- Dentre as opções apresentadas, busque pela seção **CloudWatch continuous logs**.



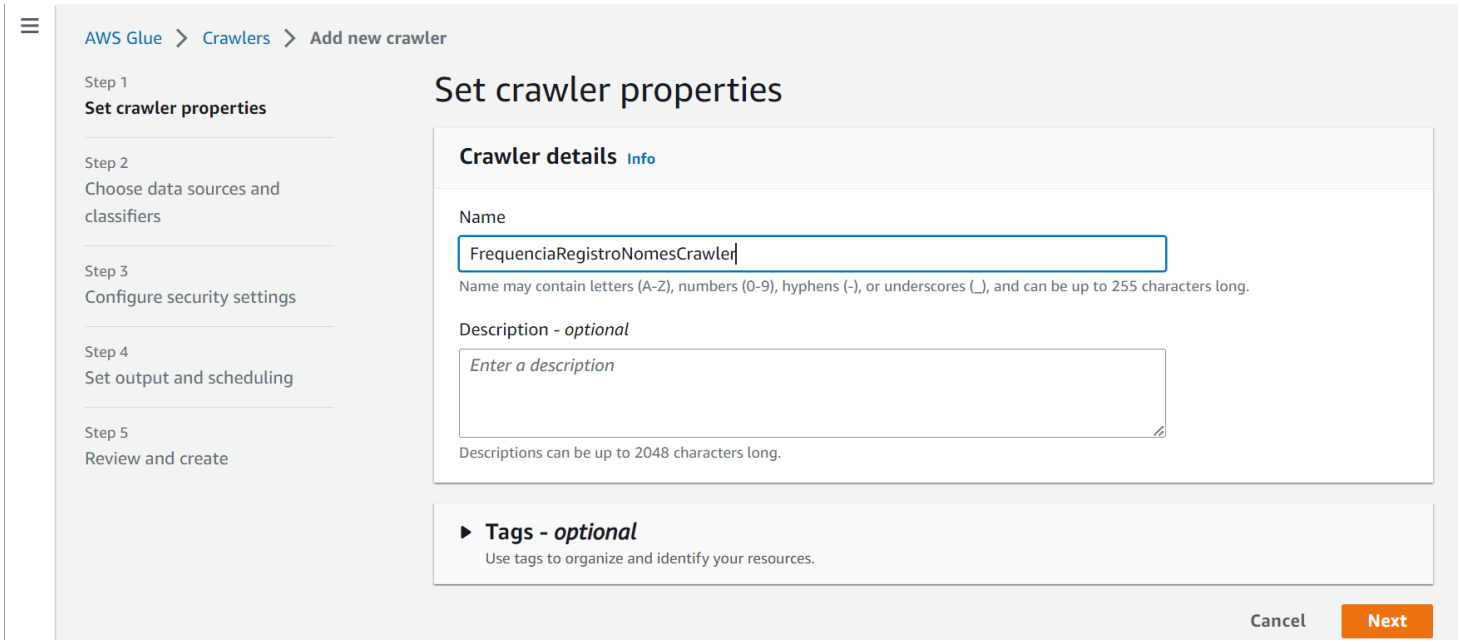
- Você encontrará informações complementares sobre desenvolvimento de Scripts ETL com Glue na documentação oficial do produto.

6 - Criando novo crawler

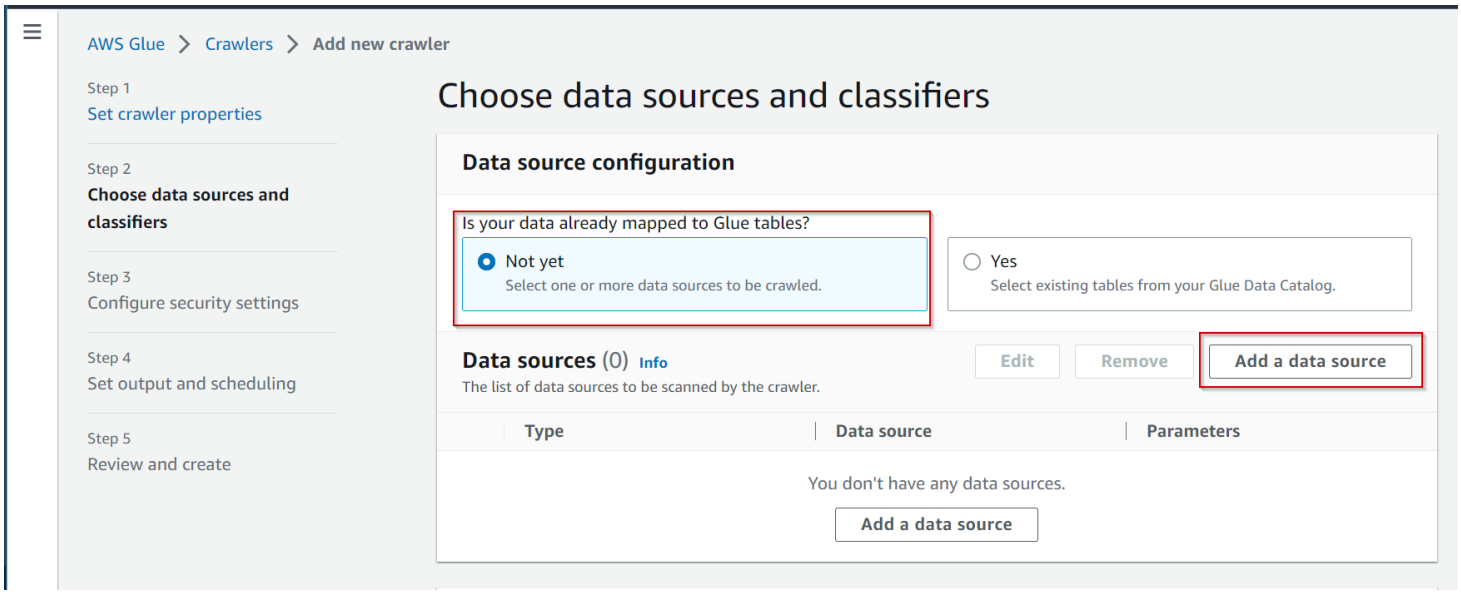
Crawlers são mecanismos que podemos utilizar para monitorar nosso armazenamento de dados de modo a criar/atualizar metadados no catálogo do Glue de forma automática. Na sequência iremos desenvolver um crawler para automaticamente criar uma tabela chamada **frequencia_registro_nomes_eua** a partir dos dados escritos no S3 (verifique a última atividade do notebook).

Vamos as passos para criação de nosso crawler:

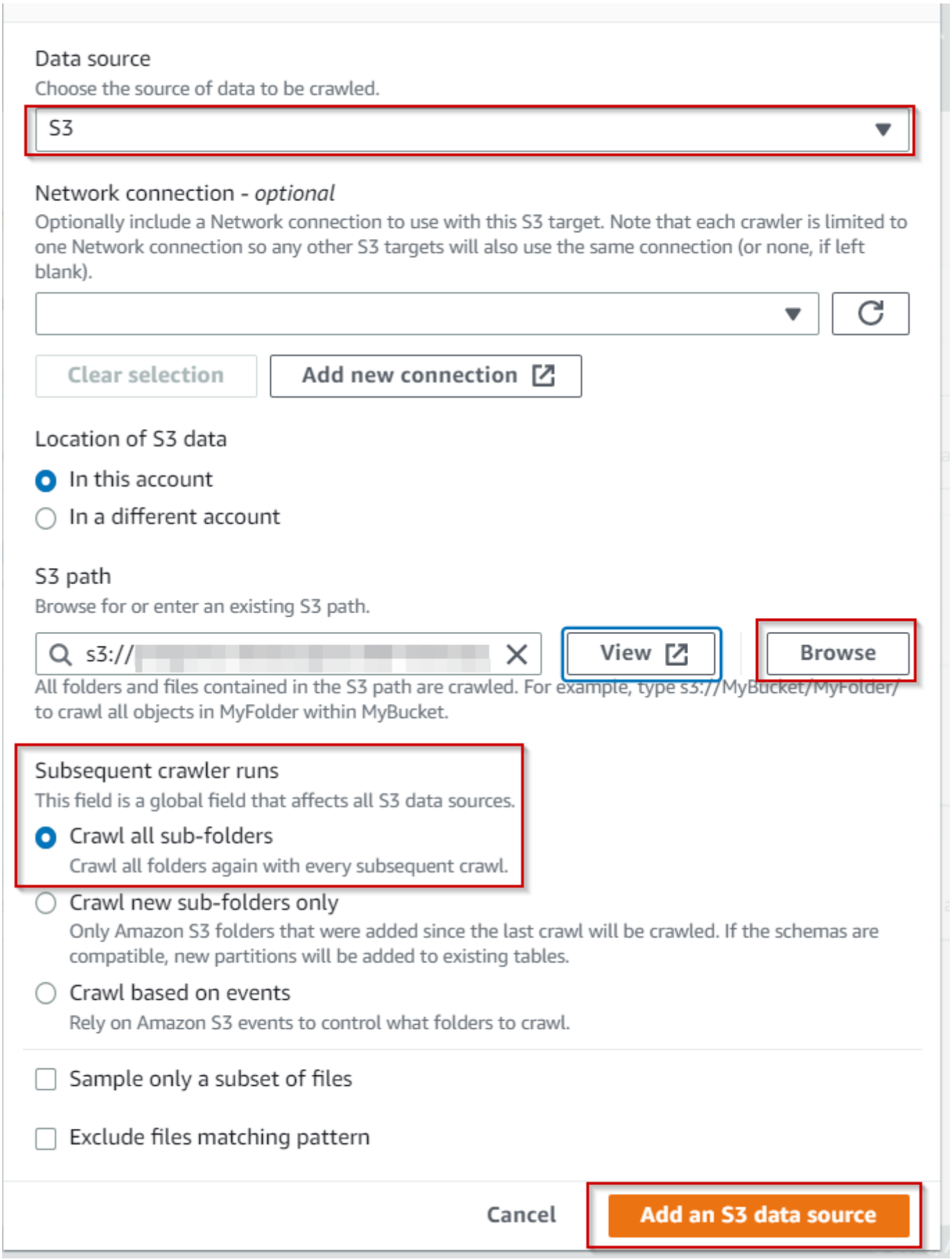
- No console, acesse o serviço **AWS Glue**. Na página do serviço, escolha a opção **Crawlers** no menu à esquerda. Na sequência, clique no botão **Create**.
- No primeiro passo de criação do **Crawler**, informe *FrequenciaRegistroNomesCrawler* no campo **Name**. Clique em **Next**.



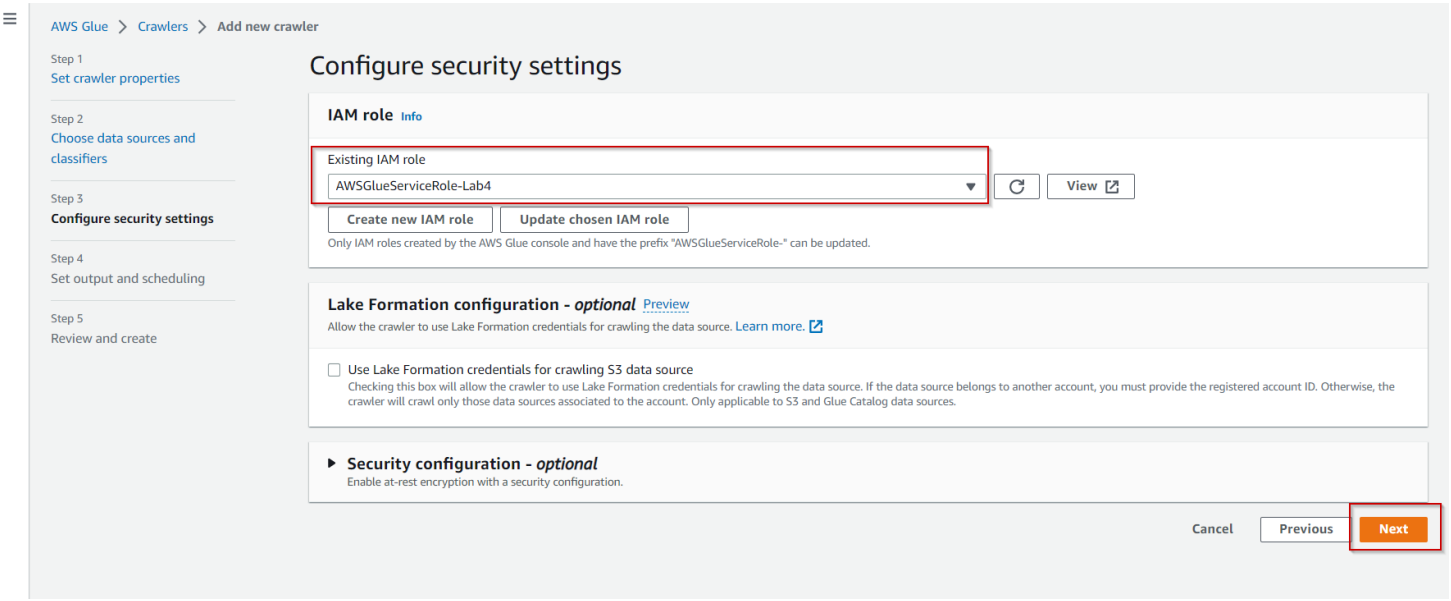
- Em **Choose data sources and classifiers**, devemos informar o caminho do S3 a ser monitorado. Para **Is your data already mapped to Glue tables?**, informe **Not yet**. E, na sequência, clique em **Add a data source**.



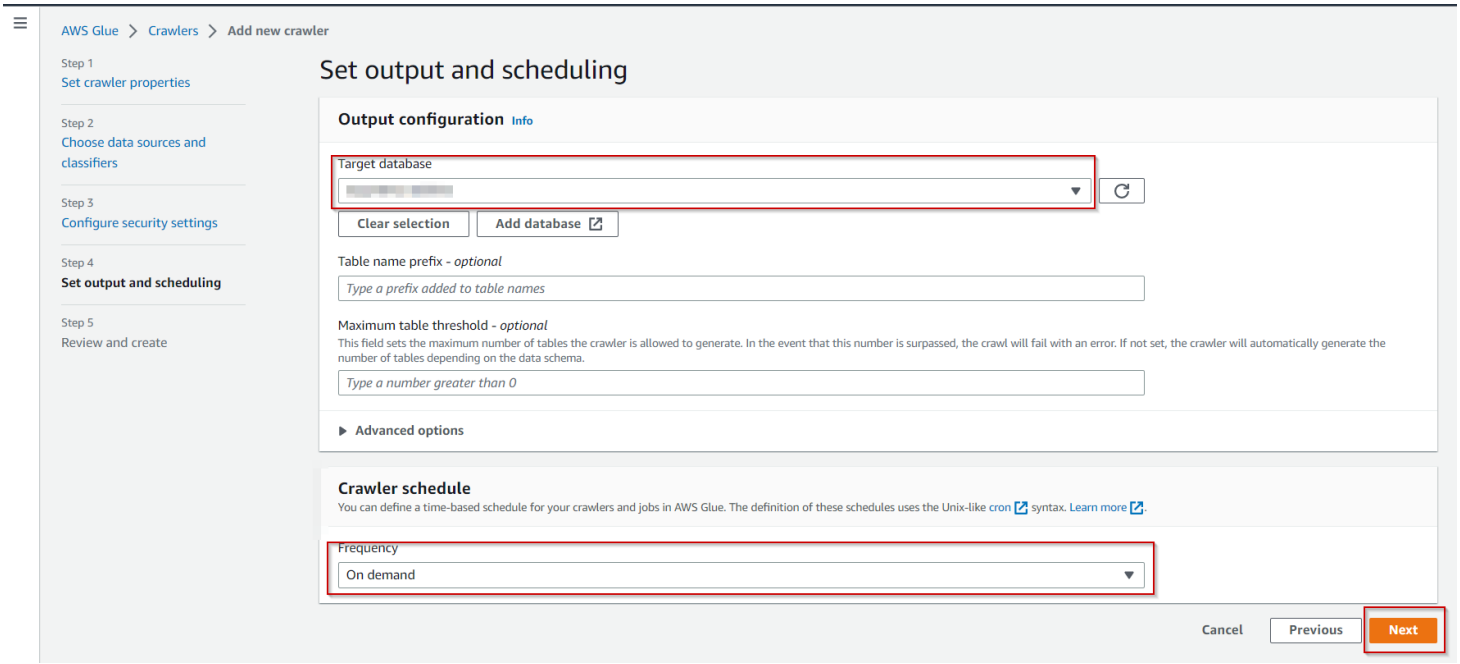
- Na tela aberta, em **Data source**, certifique que esteja S3. Em **Location of S3 data**, informe **In this account**. Finalmente, no campo **S3 path**, informe o caminho `s3://<BUCKET>/lab-glue/frequencia_registro_nomes_eua/` , lembrando de substituir `<BUCKET>` pelo utilizado anteriormente.



- Na etapa **Configure security settings** informe a role *AWSGlueServiceRole-Lab4* no campo **Existing IAM role**. Avance clicando em **Next**.



- Em **Set output and scheduling**, no campo **Target database**, informe **glue-lab**. Em **Crawler schedule**, no campo **Frequency**, defina **On Demand**. Avance e finalize o processo de criação.



Crawler criado, agora vamos executá-lo. Na tela inicial (Crawlers), selecione *FrequenciaRegistroNomesCrawler* e clique em **Run**. A execução pode leva alguns segundos e você pode acompanhar o resultado na própria tela em que está.

Se a execução for bem sucedida, nós esperamos que uma nova tabela, de nome *frequencia_registro_nomes_eua* tenha sido criada na base *glue-lab*. Você pode vê-la por meio do **Glue Catalog** e também no **Athena**.

Para consultar os dados, você deverá conceder privilégios de **DESCRIBE** e **SELECT** no **Lake Formation**. Vamos deixar essa parte de desafio para você 😁.