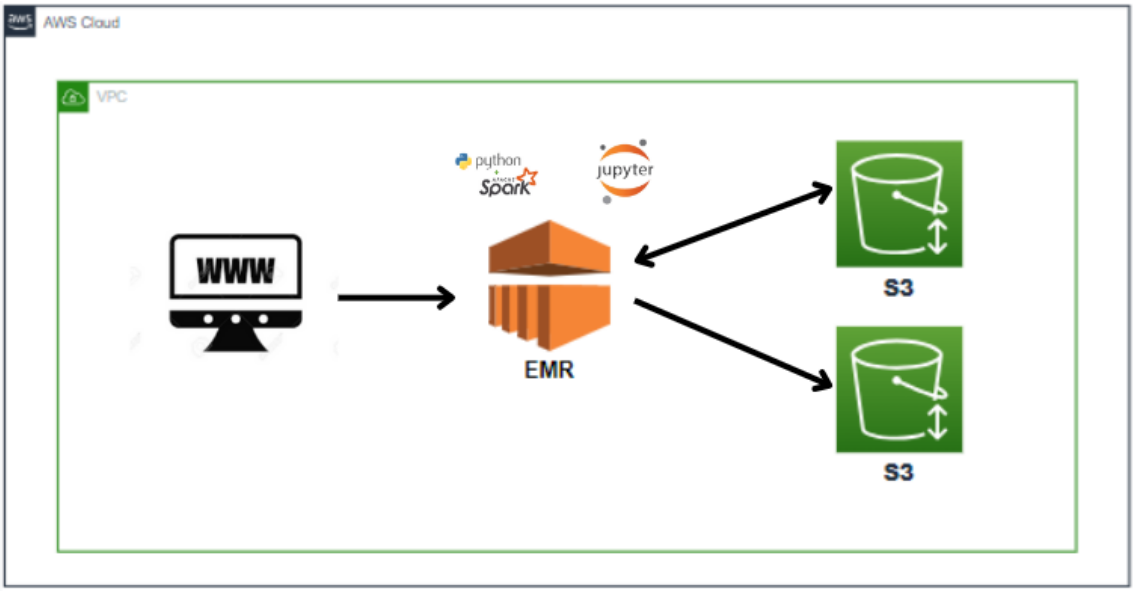
Criação de Pipeline ETL em clusters EMR utilizando PySpark, Jupyter Notebook e S3 Buckets

1.Objetivo

Neste projeto construí um pipeline de dados na nuvem AWS, desde o zero, incluindo configurações de ambiente, criação de usuários, redes, subredes, EC2, grupos de segurança, clusters EMR, criação dos buckets S3, transformação e limpeza dos dados brutos, utilizando PySpark, até a disponibilização da base contendo os dados prontos para uso destinada ao usuário final. A fonte de dados é o site da Redfin, uma das maiores imobiliárias dos EUA e Canadá, que disponibiliza livremente dados do mercado imobiliário para áreas metropolitanas, cidades, bairros e códigos postais. Abaixo seguem descritos o passo a passo de cada etapa.

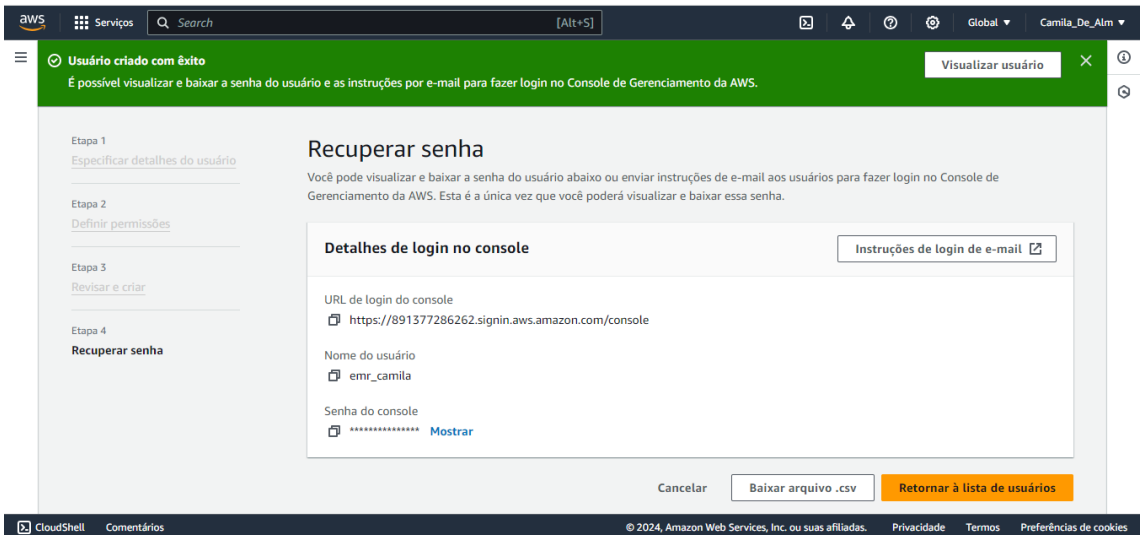
Demonstração do fluxograma final da arquitetura na AWS:



2.Criação do usuário Administrador

Será necessário ter uma conta ativa na AWS – Amazon World Services para execução dos itens a seguir. Devido a questões de segurança o usuário root, que é o usuário responsável pela criação da conta AWS, deverá criar um outro usuário administrador para que em caso de invasão da conta, ela se mantenha segura. Para este projeto o usuário administrador foi criado no AWS IAM (Identity and Access Management) que é um serviço da AWS que ajuda a controlar o acesso aos recursos da AWS de forma segura. Logada como usuário root, criei as credenciais e

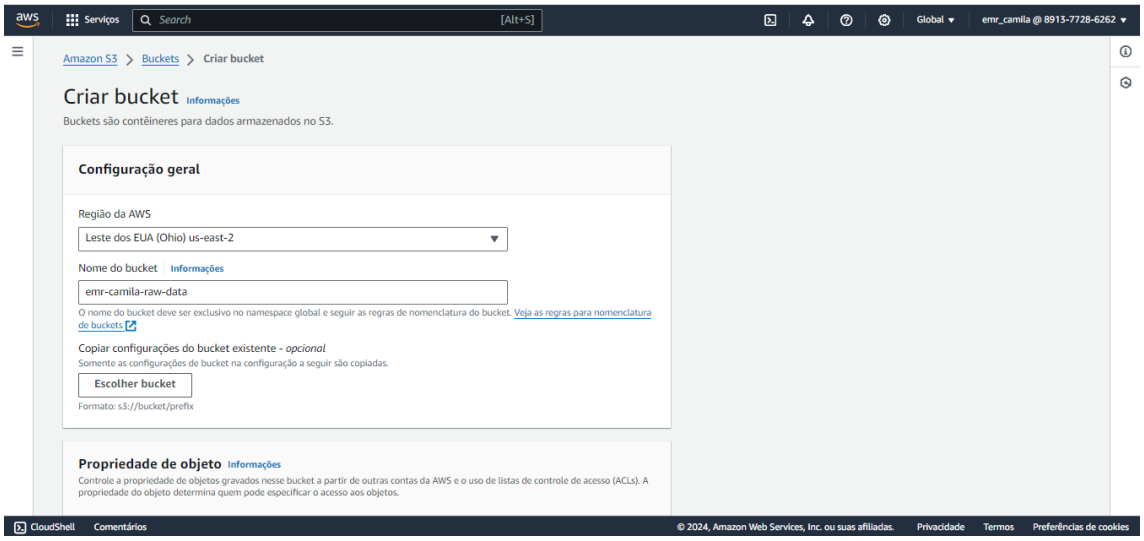
keypairs do usuário administrador, além disso habilitei as permissões necessárias para utilizar o EMR e os Buckets S3.



Por questões de segurança, também fiz o download das credenciais em csv, para poder recuperar no futuro, em caso de perda ou esquecimento de senha. Após seguir as práticas de segurança recomendadas pela AWS, fiz o log off do usuário Root e fiz log in no novo usuário Administrador.

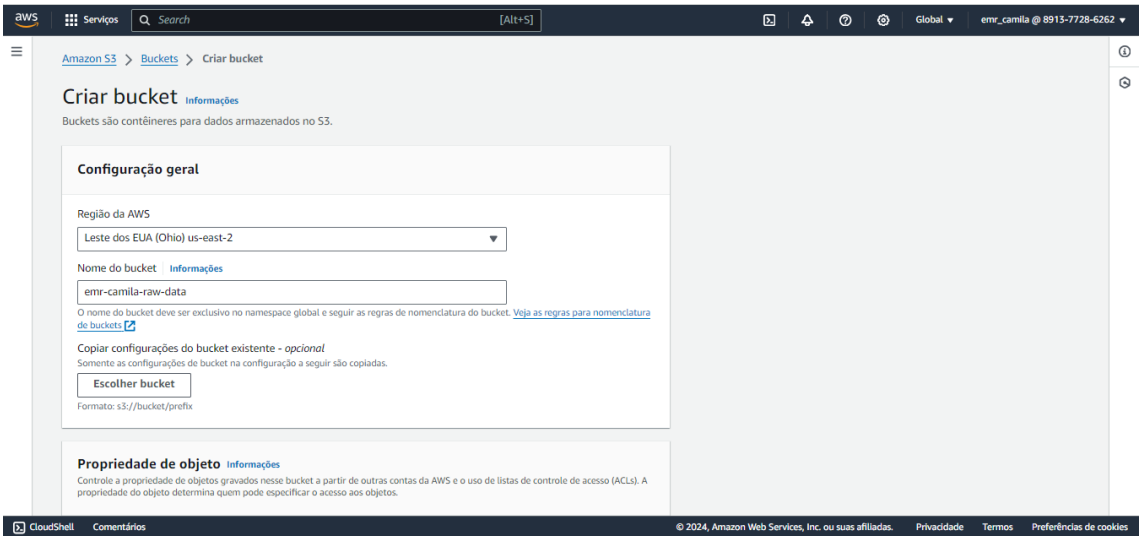
3. Criação dos Buckets no AWS S3

Através do usuário Administrador fiz a criação de dois buckets S3, o primeiro para receber a extração dos dados da fonte localizada no site Redfin e o segundo bucket para receber os dados transformados e limpos. No console AWS, pesquise por S3, após criei o bucket. O nome do bucket deve ser exclusivo globalmente e seguir as regras de nomenclatura determinadas pela AWS, conforme imagem abaixo.



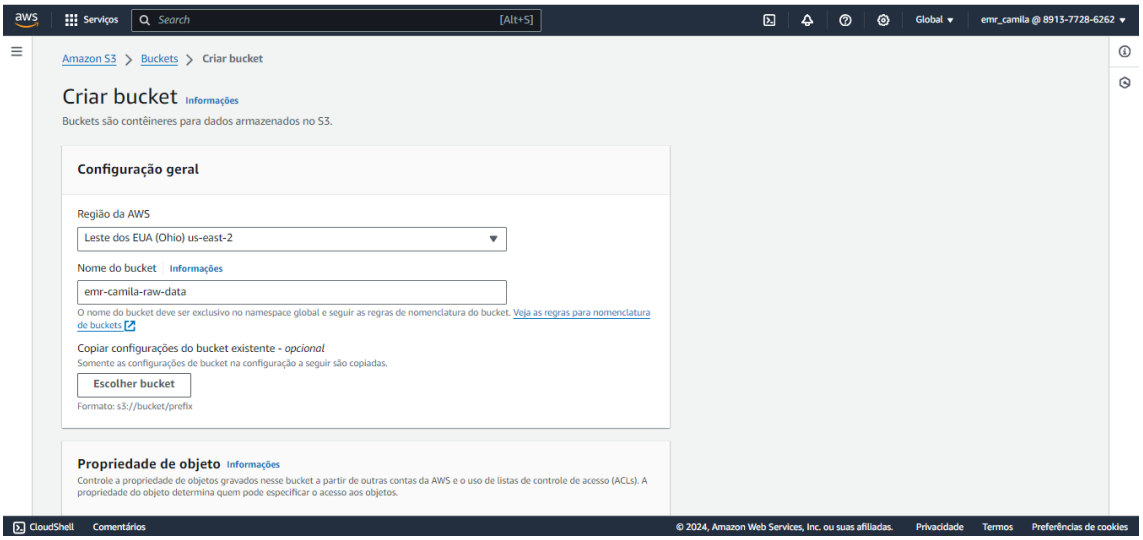
A região escolhida foi a us-east dos EUA, devido ao custo ser o mais recomendado, mas devido a latência, é recomendado que a escolha da Região seja a mesma

aonde o seu usuário está localizado. Além disso, é possível determinar propriedades, tags e versionamento, mas mantive as opções padrão sugeridas pela AWS. Para o segundo bucket, segui o mesmo procedimento acima.

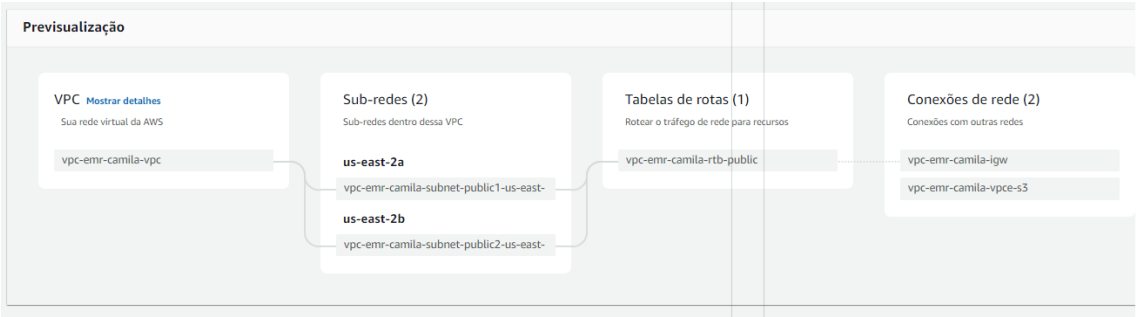


4.Criação da VPC e Subredes

A VPC é uma Virtual Private Cloud, ou seja, uma nuvem privada virtual, ela cria uma rede privada virtual isolada na AWS, protegendo meu cluster EMR e meus dados contra acesso não autorizado e criptografa os dados em repouso e em trânsito entre o Amazon S3 e o Amazon EMR para garantir a confidencialidade das informações. No console, pesquise por VPC, depois clique em “Criar VPC”. Escolhi um nome para VPC, devemos criar a VPC, subredes, tabelas de rotas e determinar as zonas de disponibilidade, gateway e o endpoint. As zonas de disponibilidade, configurei como duas, pois caso haja algum desastre e uma região se torne indisponível, o trânsito de dados será redirecionado para a segunda, setei o mesmo valor para as redes públicas e nenhuma para as redes privadas, porque serão dados internos. O endpoint será o Gateway S3.



A seguir, visualização de todo o fluxo da VPC.



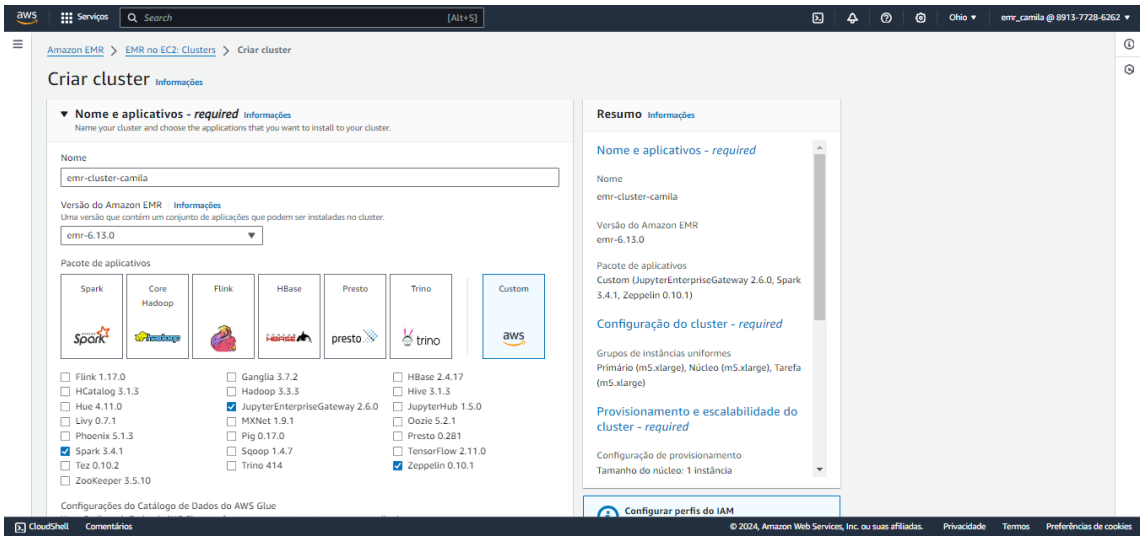
5. Clusters EMR

Após configuração de usuário, ambiente e segurança, criei o cluster AWS EMR (Elastic Map Reduce) que é um serviço da AWS que facilita a execução de frameworks de big data, como Apache Hadoop e Apache Spark, para processar e analisar grandes volumes de dados, neste projeto utilizei o Apache Spark.

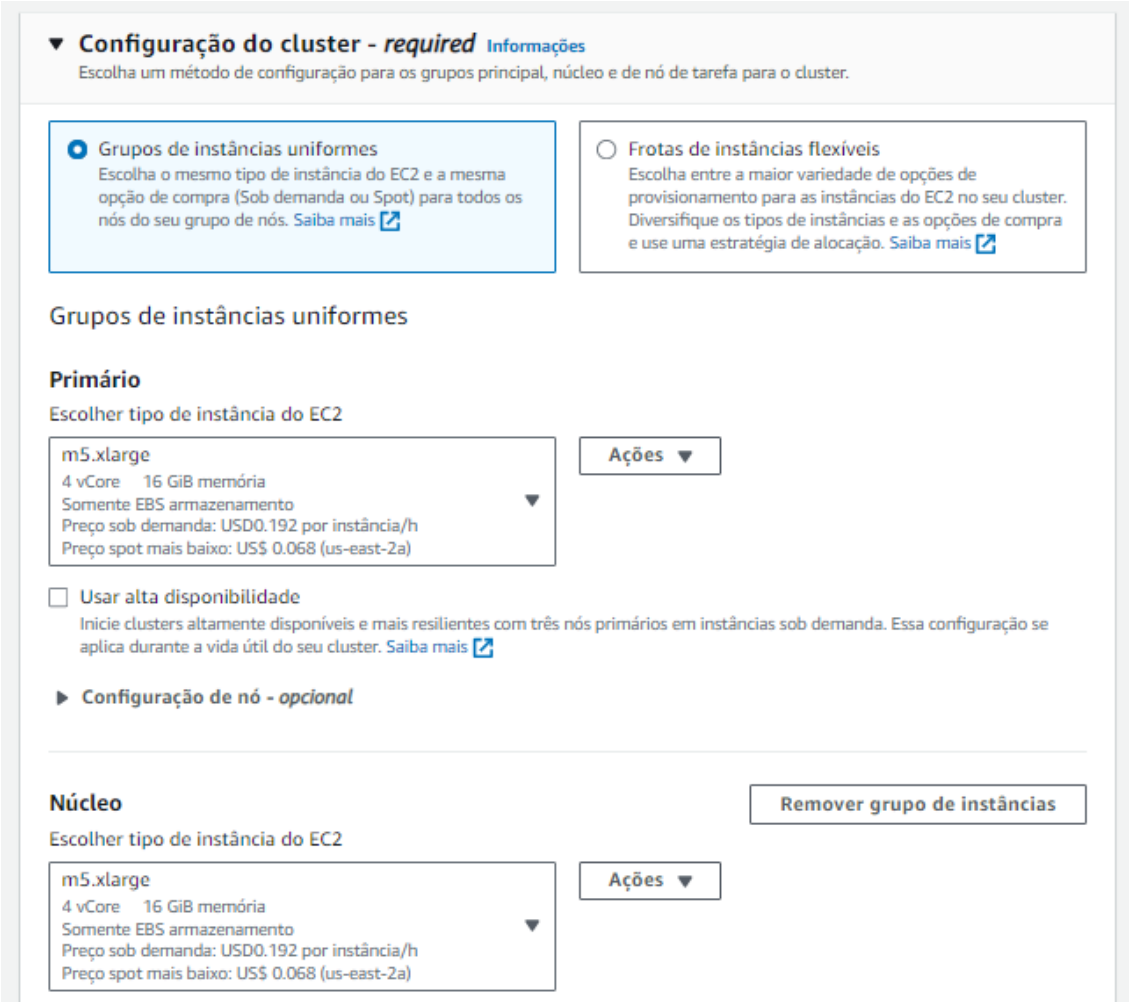
Nos clusters EMR da AWS, os nós são instâncias do EC2 que se unem para fornecer a capacidade de processamento e armazenamento necessária para executar os jobs de big data. Existem três tipos principais de nós em um cluster EMR:

- **Node ou Nó primário (Primary Node):** É responsável por coordenar todo o trabalho no cluster, gerenciando tarefas, monitorando a saúde do cluster e rastreando o status dos jobs.
- **Nós centrais (Core Nodes):** Eles executam tarefas de processamento de dados e pode haver vários nós centrais para distribuir e executar jobs em paralelo.
- **Nós de tarefa (Task Nodes) (Opcional):** Fornecem capacidade de processamento adicional para executar tarefas intensivas em CPU.

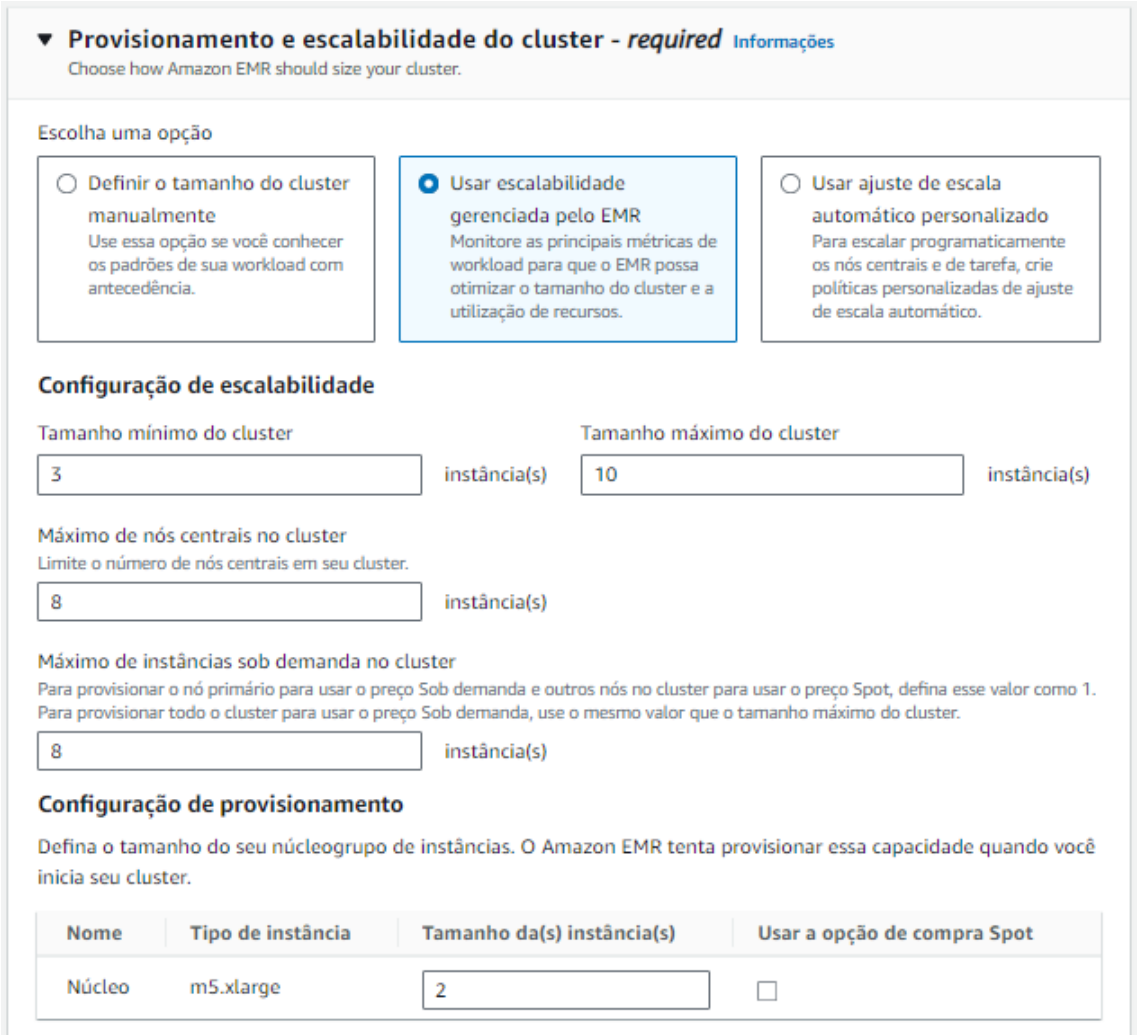
Neste projeto, criei o Nó primário e o Nó secundário, devido a frequência e tamanho da carga. O EMR já vem com principais frameworks, por isso pode ser configurado, diretamente no console, o que diminui o custo. O setup escolhido foi a versão 6.13.0, não escolhi a mais recente, selecionei os aplicativos Spark, Jupyter e Zeppelin, conforme abaixo.



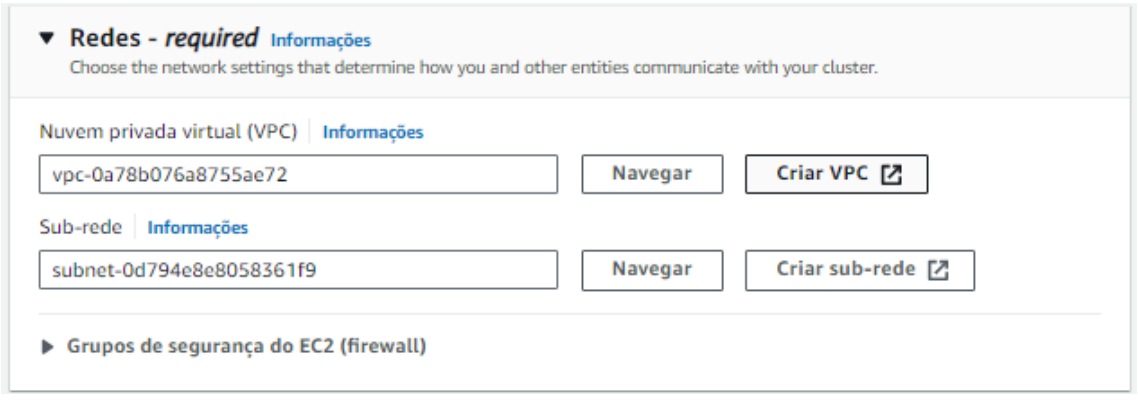
O grupo de instâncias escolhido foi o uniforme, para usar AWS EC2 sob demanda ou spot, devido a economia de escala, o tipo escolhido foi o M5.xlarge, que contém 4v core e 16Gib de memória, para o núcleo escolhi as mesmas configurações e removi a opção de ter um cluster de task, porque ele é opcional. As instâncias M5 do Amazon EC2 são alimentadas pelos processadores Intel Xeon Scalable mais rápidos da nuvem, com uma frequência turbo "all-core" de até 4,5 GHz. Além disso, as instâncias M5zn apresentam capacidade de rede de 100 Gbps, sendo ideais para aplicativos de computação intensiva e de alto consumo de recursos de rede, mas a instância pode ser alterada a depender do tamanho da carga, por medidas de desempenho escolhi a M5.Xlarge.



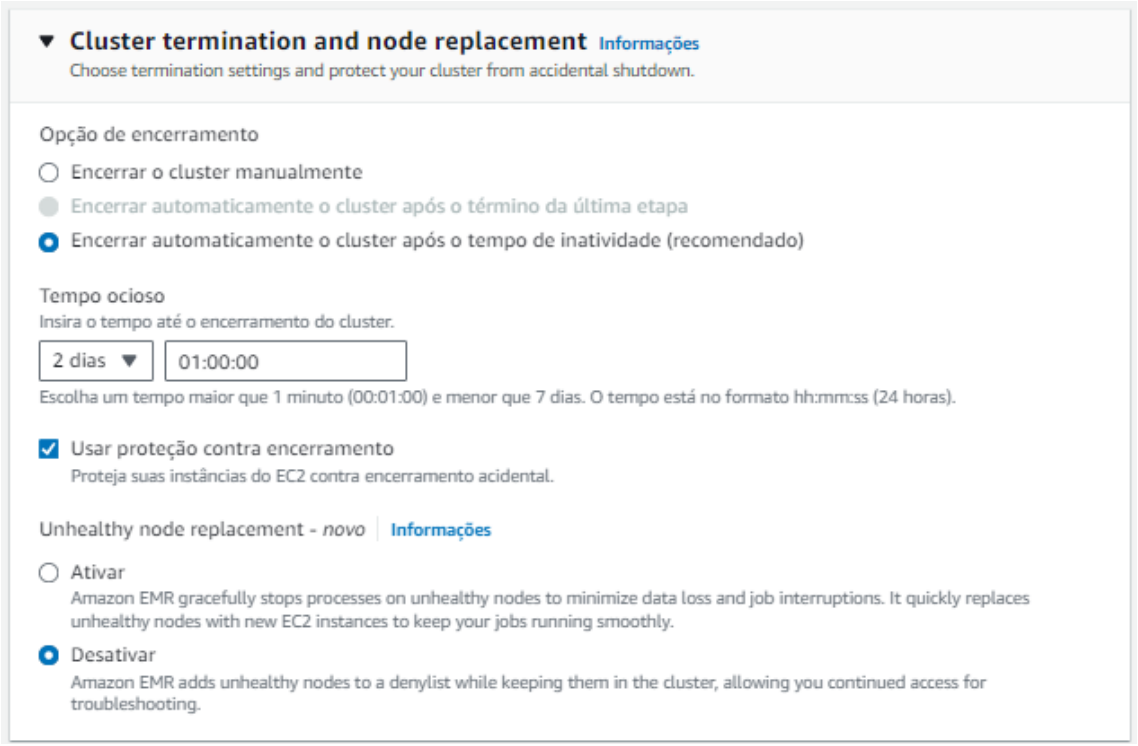
Para a seção “Provisionamento e escalabilidade do cluster”, sobre a escalabilidade escolhi a opção gerenciado pelo EMR, porque assim o EMR se adapta conforme a variação do fluxo de dados. O Tamanho mínimo de cluster(número de instâncias) foram três, porque uma fica para o Nó Principal e a segunda e terceira ficam para o Nó Central. Quando a demanda estiver alta o número máximo do cluster será de 10 instâncias EC2, já para o Nó Central serão 8. Configuração de provisionamento ficou para duas, que seria a capacidade inicial da instância do núcleo.



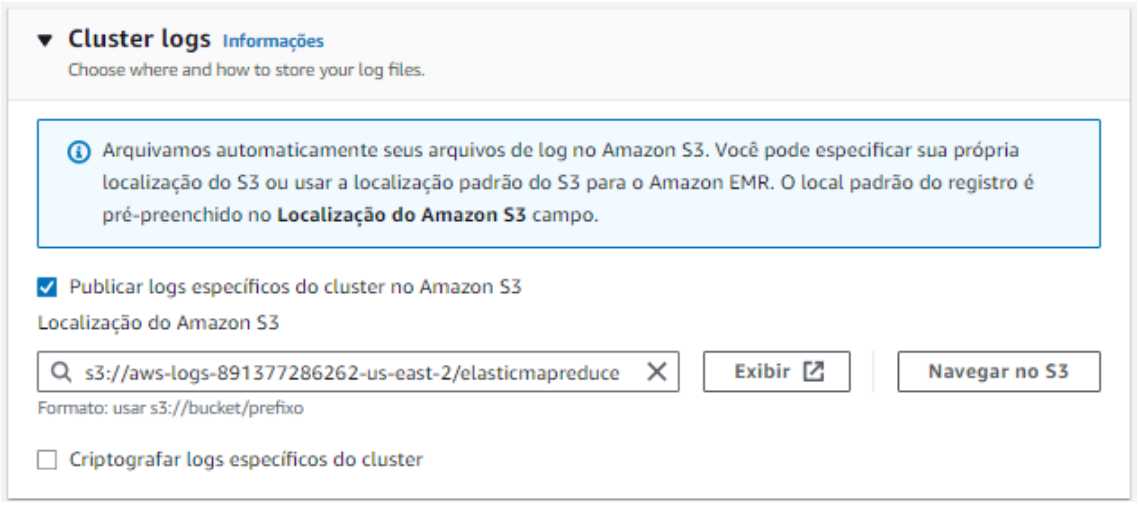
Na seção, “Redes”, selecionei a VPC que havia criado anteriormente, pesquisei em navegar, selecionei a VPC e automaticamente o AWS preenche com as informações, da VPC e das Subredes.



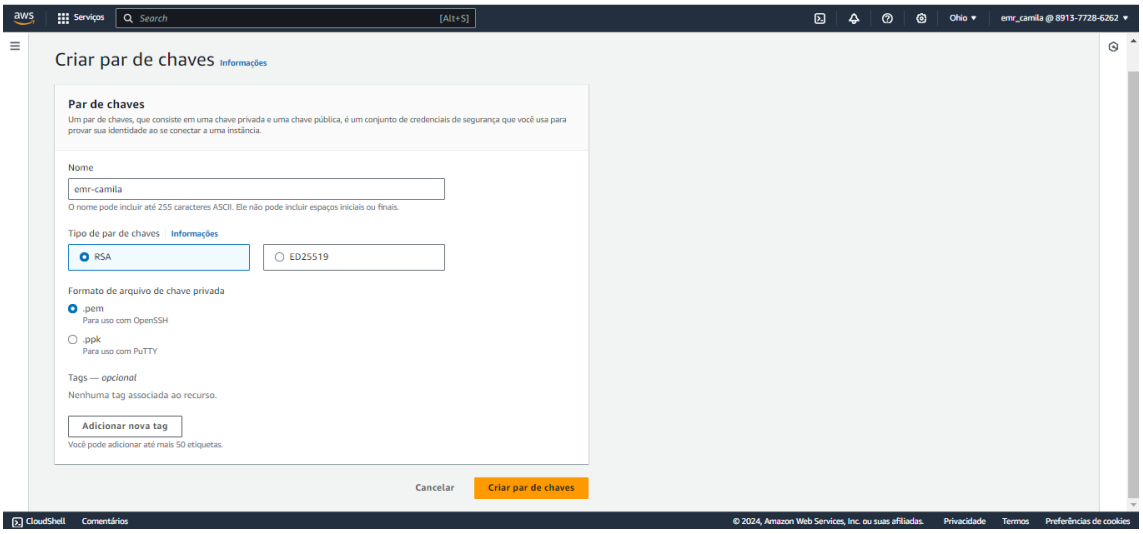
No tópico “Encerramento de cluster e substituição de nó”, selecionei “Encerrar automaticamente o cluster após o tempo de inatividade”, porque é uma forma de reduzir custo, assim ele será encerrado em momentos de inatividade. Selecionei a opção “Usar proteção contra encerramento” se refere as EC2, assim deleta os clusters, mas mantém os dados armazenados.



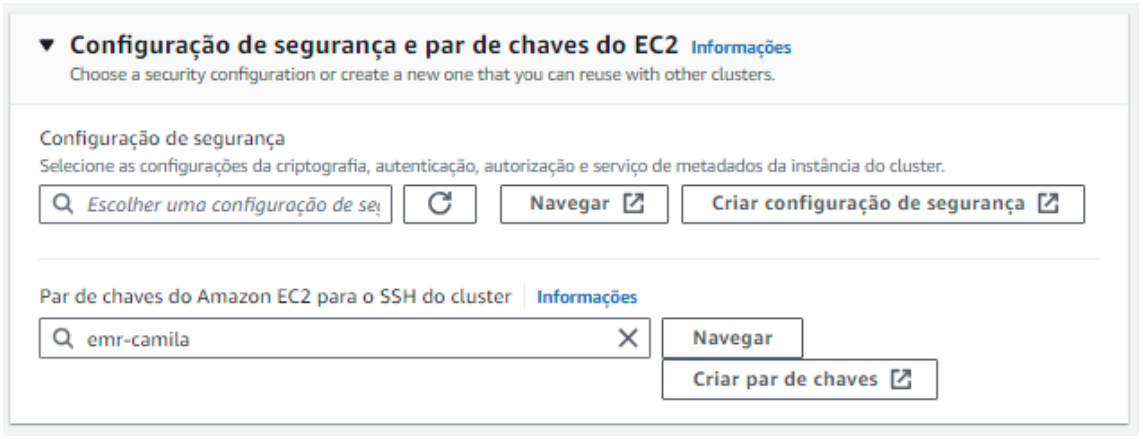
Na seção, “Cluster logs”, permiti que a AWS criasse um bucket para armazenamento dos logs dos clusters EMR.



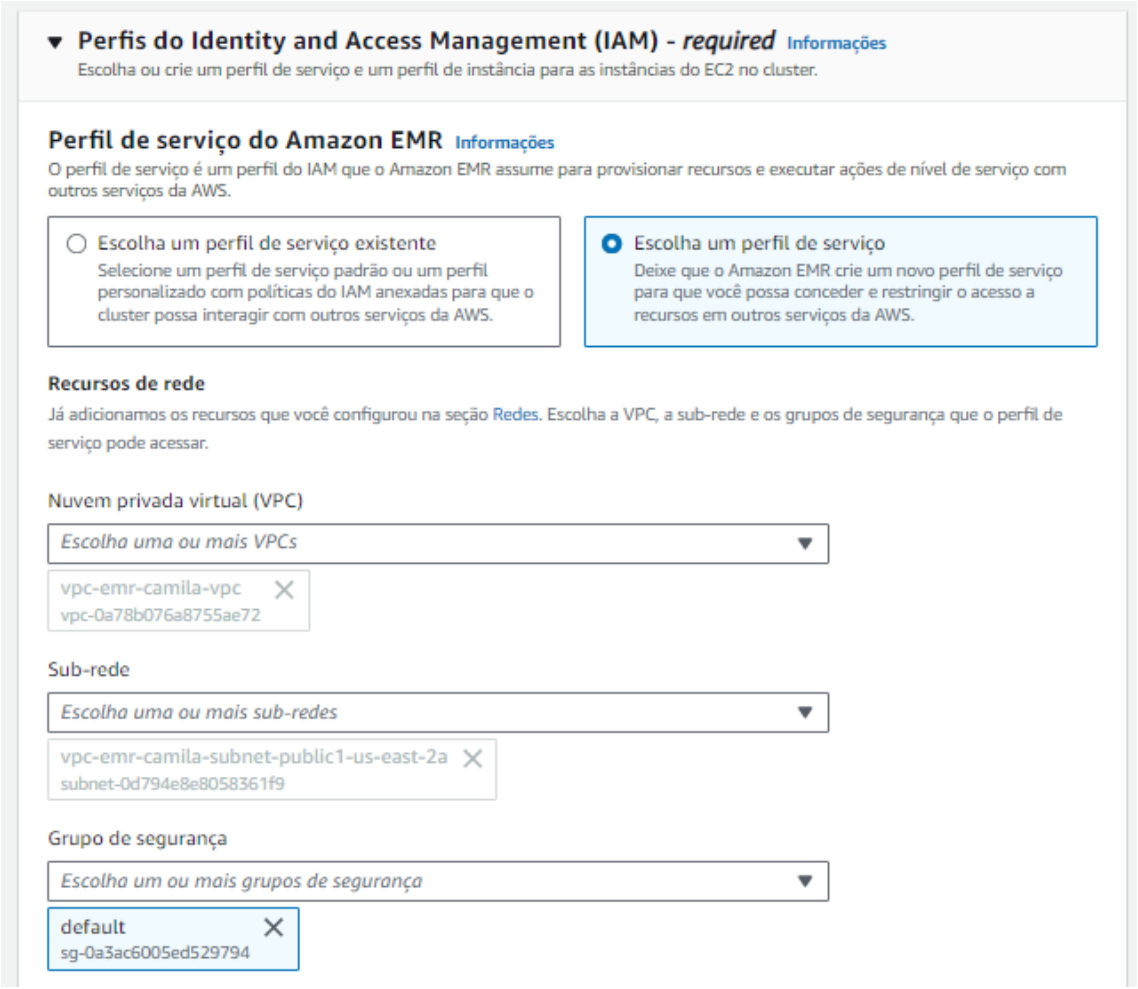
Sobre a “Configuração de segurança e par de chaves do EC2” a prática recomendada é criar uma keypair para acessar via SSH, no entanto eu criei diretamente pelo console. Clique no botão criar par de chaves, abriu outra tab, dei um nome, escolhi o par RSA e o formato .pem, salvei o arquivo gerado.



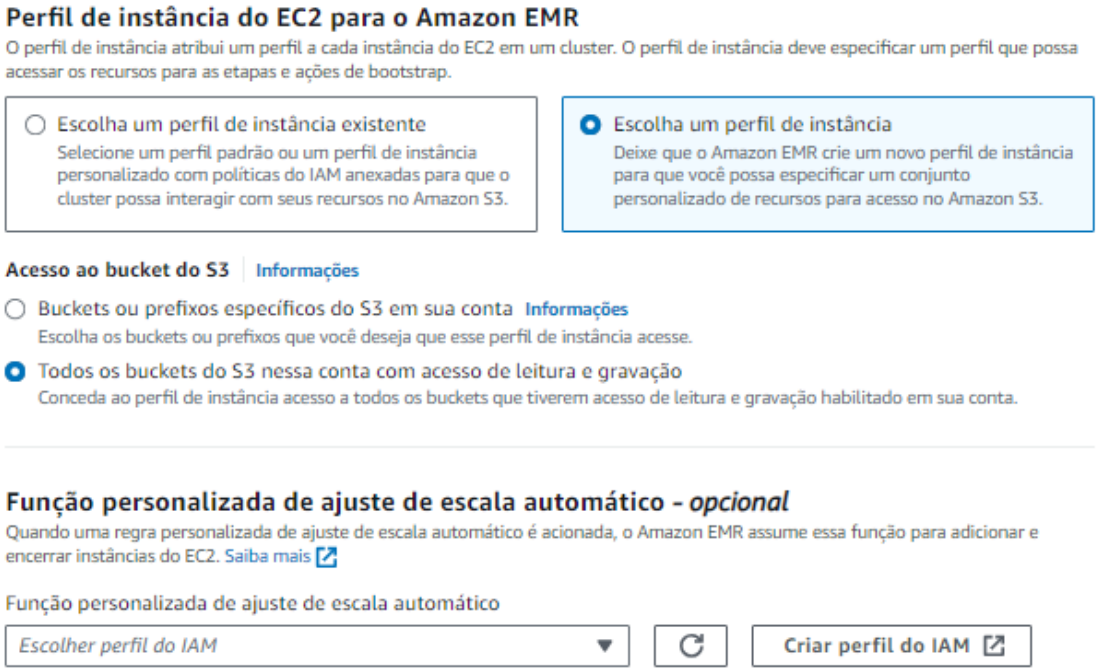
Após selecione, a keypair, na configuração de segurança conforme abaixo.



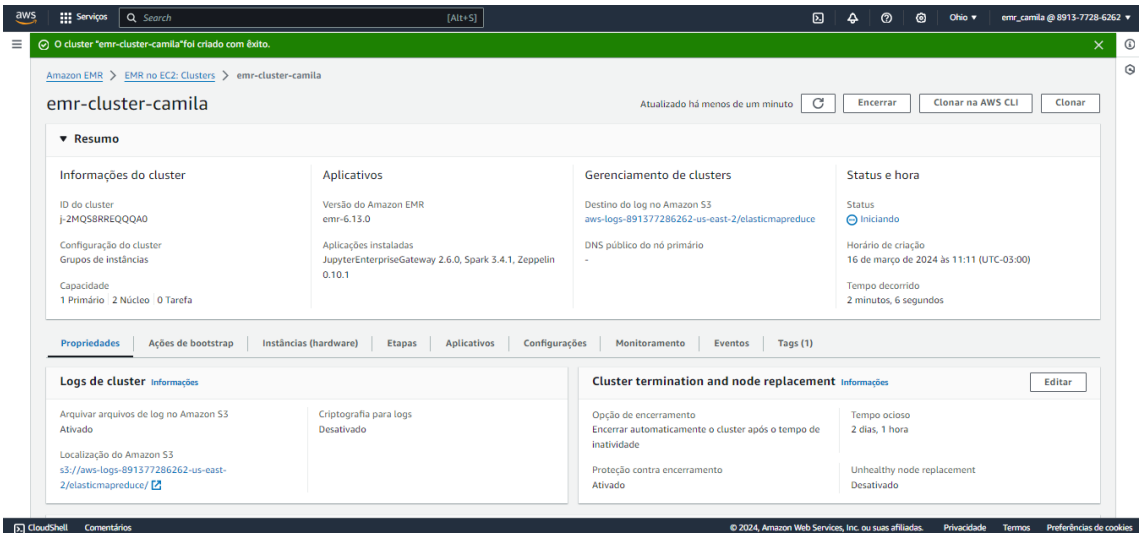
Na seção “Perfis do IAM”, selecionei a opção escolha um perfil de serviço, por isso criei a VPC anteriormente, selecionei o grupo de segurança default cada vez que criamos um VPC um grupo de segurança é criado automaticamente.



Após na seção “Perfil de instância”, escolhi um perfil de instância, dei acesso a todos os buckets para leitura e gravação, por isso os buckets foram criados antes dos clusters.

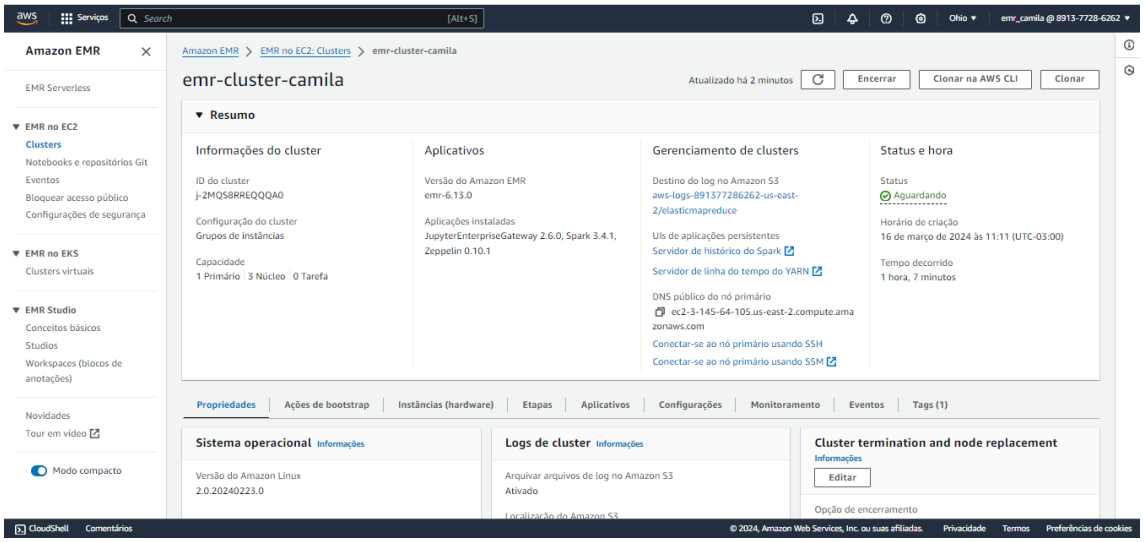


Após cliquei em “Criar cluster” o que pode levar até 7 minutos para finalizar.

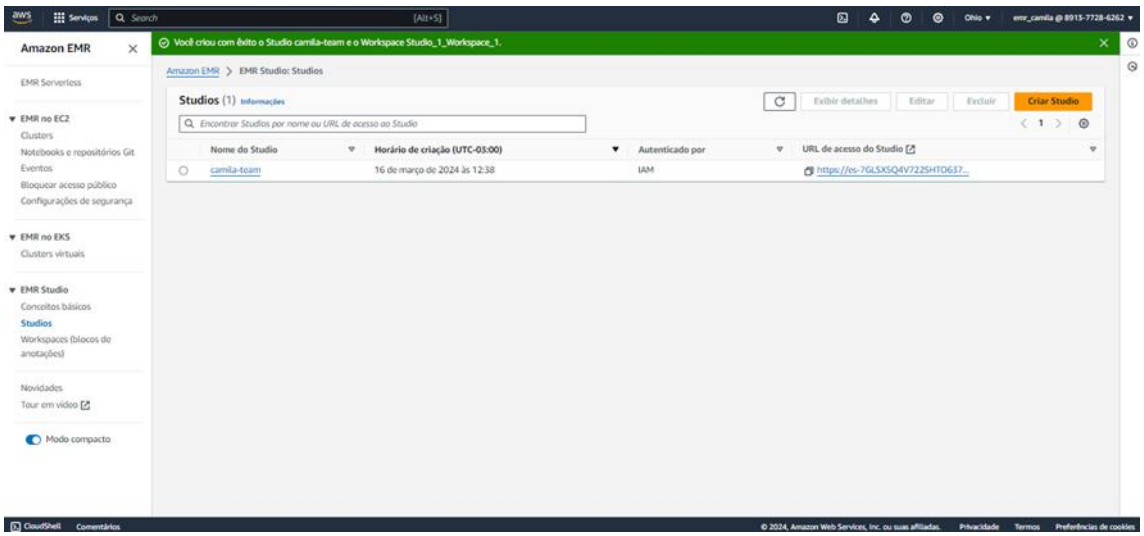


6. Configuração do Jupyter Notebook para utilizar o PySpark

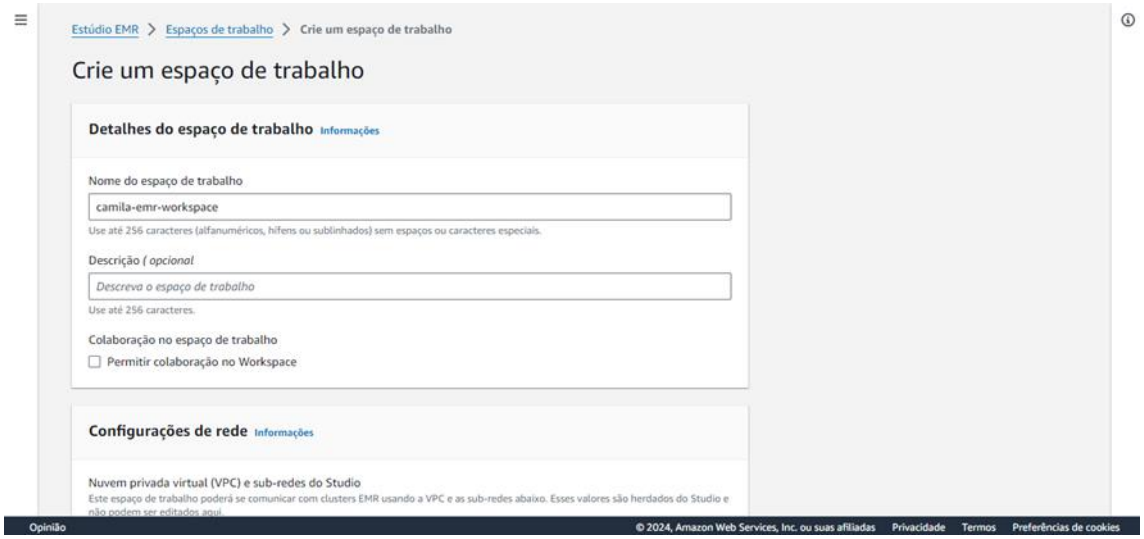
Agora é possível configurar o Jupyter Notebook para utilizar o PySpark, para isso temos que configurar um Studio. À esquerda, em seção EMR Studio, clique em conceitos básicos, imagem abaixo. Depois em criar studio, um bucket foi criado para poder anexar o Jupyter ao EMR, assim os códigos escritos no notebook, serão executados no EMR e no bucket S3 via programação. Para isso, precisei criar um bucket personalizado e atribuir para o meu usuário, primeiro dei um nome, após criei um bucket só para o meu studio e associei ao meu usuário administrador.



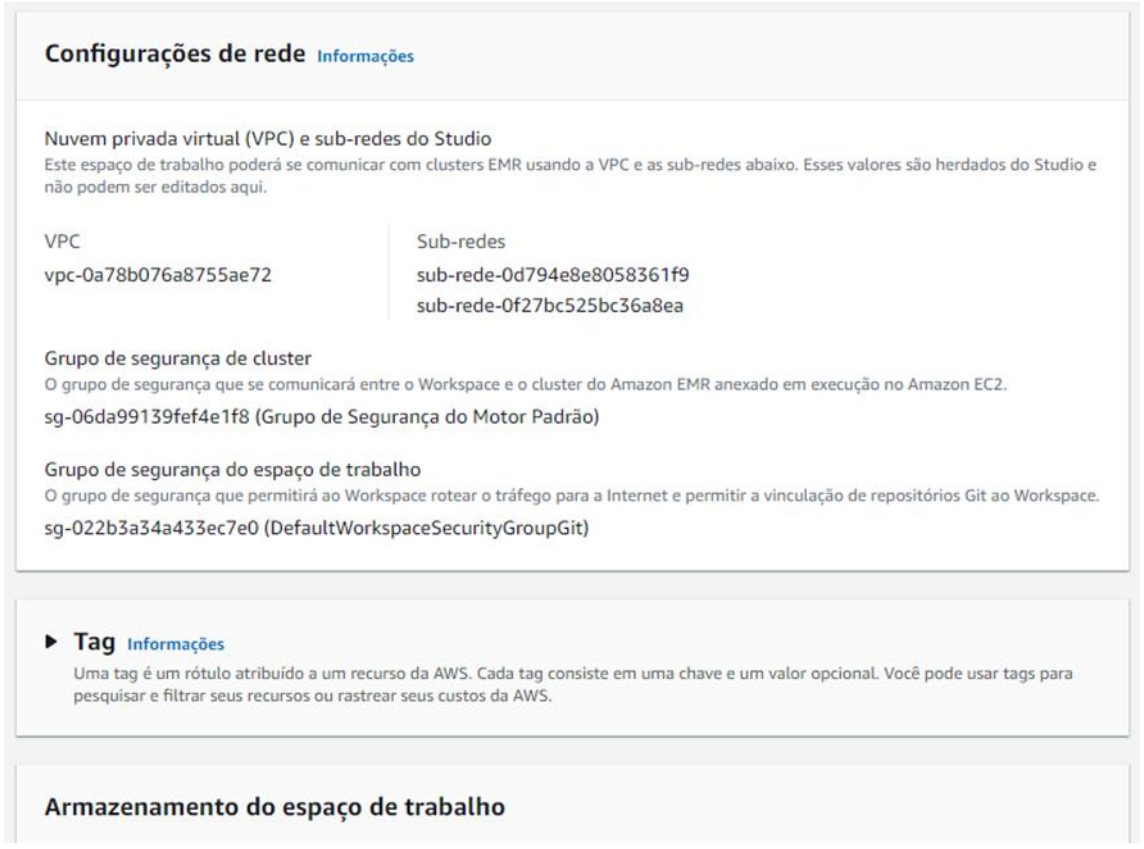
Após, na seção de “Redes e segurança” selecionei a VPC e a Subrede que já havia criado, lembrando que o usuário administrador deve ter a permissão de full acesso ao AWS S3 e ser Usuário Administrador, conforme mencionado no passo dois, após clique em “Criar Studio”. Agora podemos criar o Jupyter notebook, clique na URL do Studio criado, conforme abaixo.



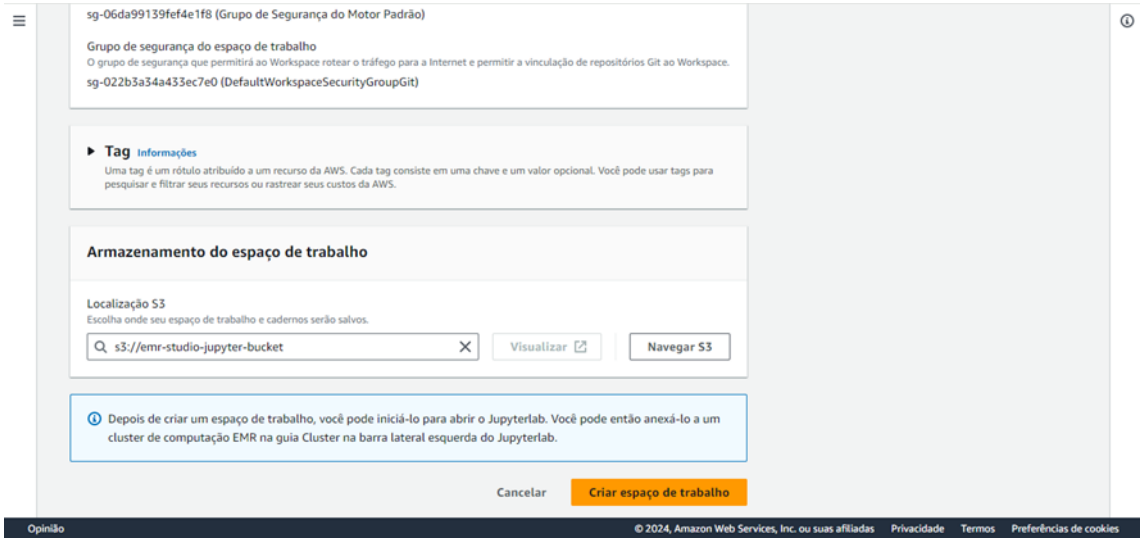
Cliquei em “Criar espaço de trabalho” escolhi um nome, as demais configurações seguem as mesmas para rede e armazenamento.



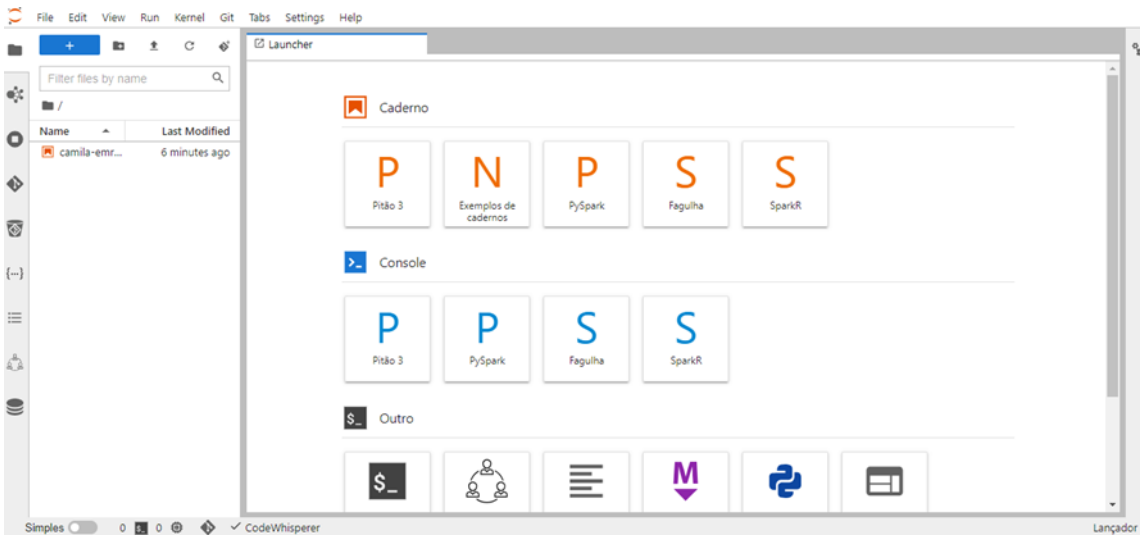
VPC que já havia criado.



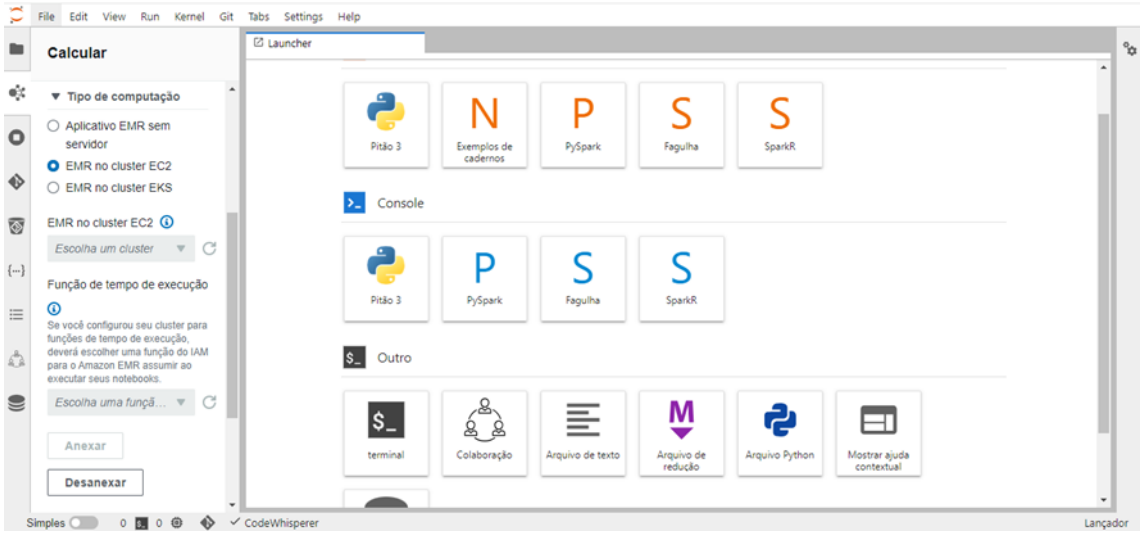
Bucket S3 criado especificamente para o Jupyter.



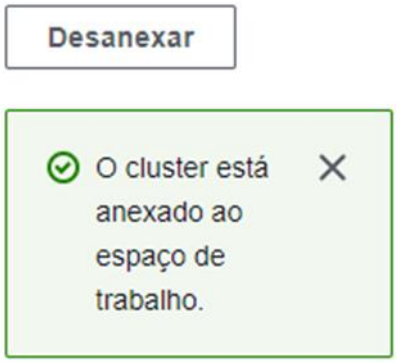
Cliquei no Studio abaixo, e ele abriu o Jupyter Notebook em uma nova aba.



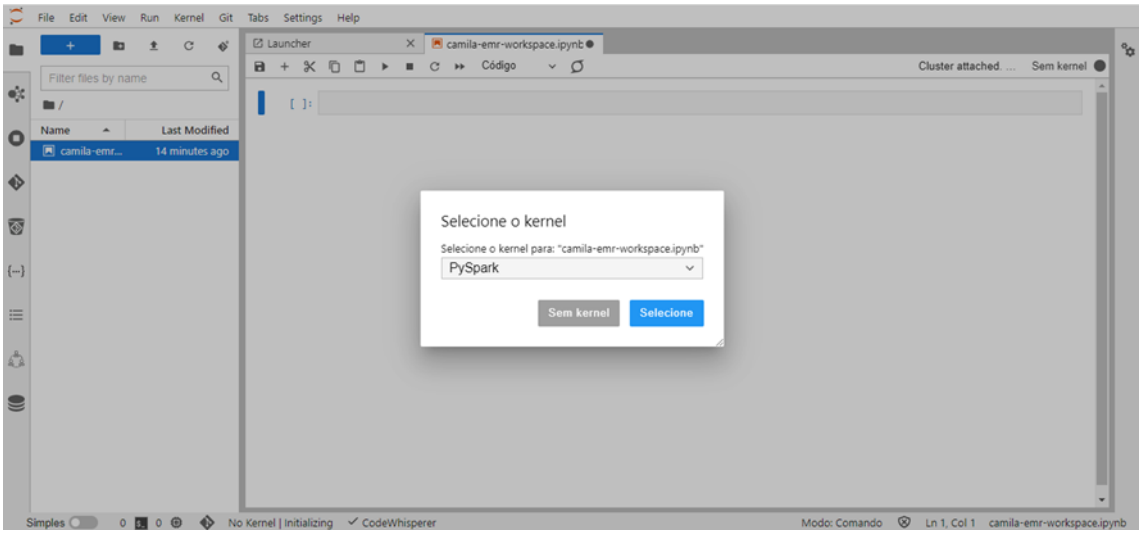
Agora é necessário anexar o Jupyter ao cluster que criamos e devemos selecionar EC2.



Atualizei a página e apareceu a mensagem abaixo em “Computação EMR”.

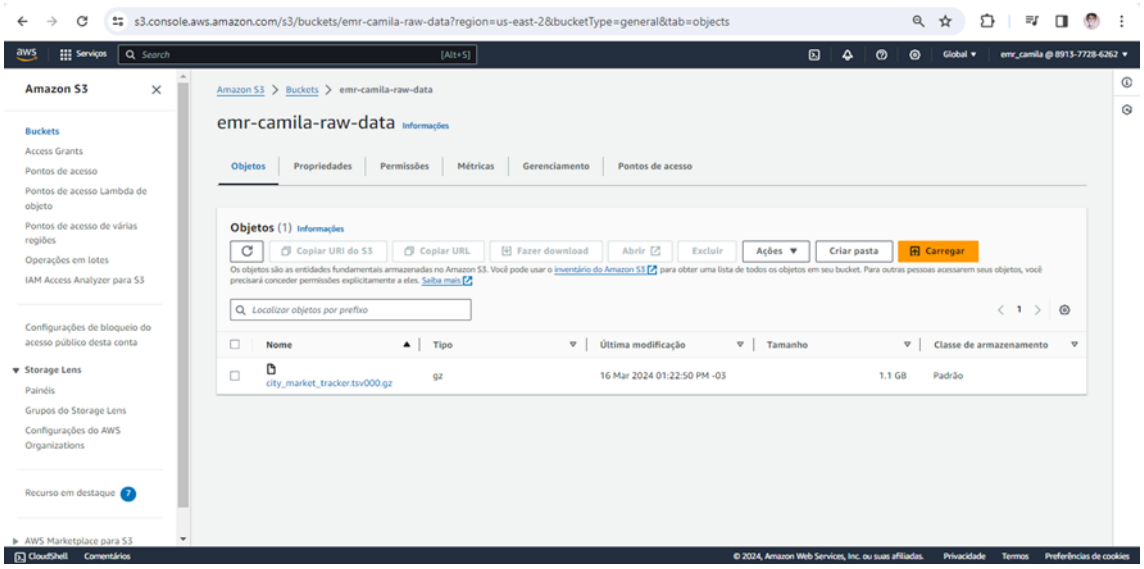


Após, voltei no meu workspace, cliquei duas vezes no caderno e escolhi o kernel como Pyspark.

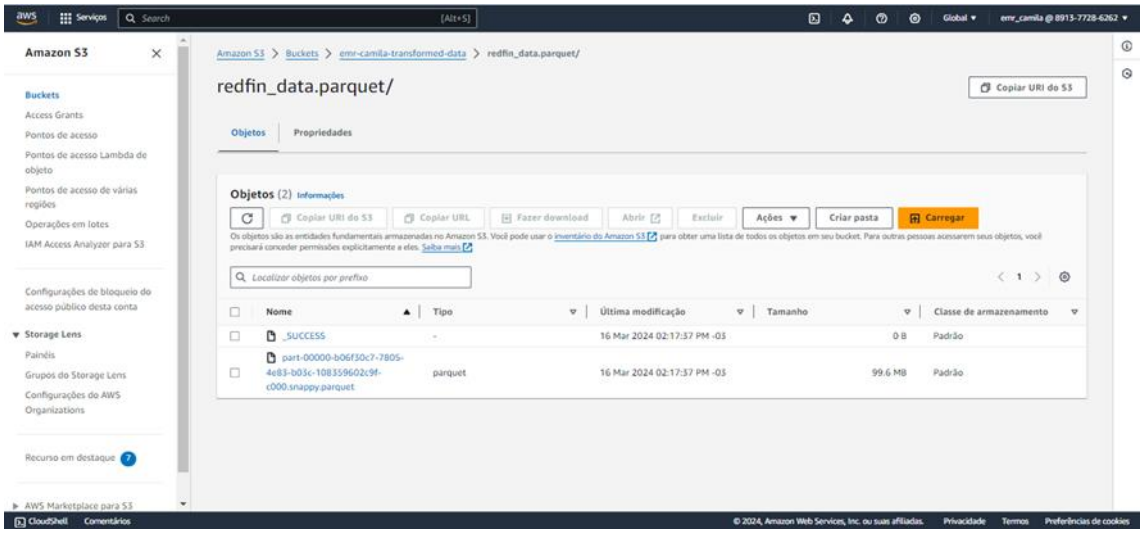


7. Explicação do código

Na primeira célula, importamos o Spark, na segunda, realizei o carregamento direto da página da Redfin e armazenei no primeiro bucket criado, o emr-camila-raw-data, o bucket que estava vazio, já recebeu a carga, conforme abaixo.



Após processar e limpar, salvei os dados transformados no segundo bucket que criei, conforme abaixo, temos o arquivo e o log de sucesso. O arquivo foi salvo em Parquet por causa do tamanho, mas poderia ter sido em csv, mas visando o melhor armazenamento e menor custo, escolhi a extensão Parquet.



8. Jupyter Notebook completo

A seguir, anexei o Jupyter Notebook com todos os códigos, transformações e alterações, as células possuem código comentado. Importante lembrar que a base disponibilizada pela RedFin, estava em sua maior parte limpa e organizada, no entanto, ajuste de colunas, verificação de valores NA, Null e mudança de nomes de variáveis foram aplicadas. Todo o embasamento foi retirado do site da AWS e a ideia para projeto foi retirada do canal Telespectra.