AWS FSx para NetApp ONTAP (FSxN) para MLOps (en inglés)

NetApp Solutions

NetApp
April 26, 2024

Tabla de contenidos

AWS FSx para NetApp ONTAP (FSxN) para MLOps (en inglés)

Autor(es):
Jian Jian (KEN), científico sénior de datos y aplicado, NetApp

En esta sección se profundiza en la aplicación práctica del desarrollo de la infraestructura de IA, ofreciendo un tutorial integral de la construcción de una canalización de MLOps con FSxN. Compuesto por tres ejemplos completos, te guía para satisfacer tus necesidades de MLOps a través de esta potente plataforma de gestión de datos.

Estos artículos se centran en:

1. "1 parte: Integración de AWS FSx para NetApp ONTAP (FSxN) como bloque de S3 privado en AWS SageMaker"
2. "Parte 2: Aprovechamiento de AWS FSx para NetApp ONTAP (FSxN) como fuente de datos para el entrenamiento de modelos en SageMaker"
3. "Parte 3: Creación de Una canalización simplificada de MLOps (CI/CT/CD)"

Al final de esta sección, habrá obtenido una sólida comprensión de cómo usar FSxN para agilizar los procesos de MLOps.

1 parte: Integración de AWS FSx para NetApp ONTAP (FSxN) como bloque de S3 privado en AWS SageMaker

Autor(es):
Jian Jian (KEN), científico sénior de datos y aplicado, NetApp

Introducción

Utilizando SageMaker como ejemplo, esta página proporciona una guía para configurar FSxN como un bucket privado de S3.

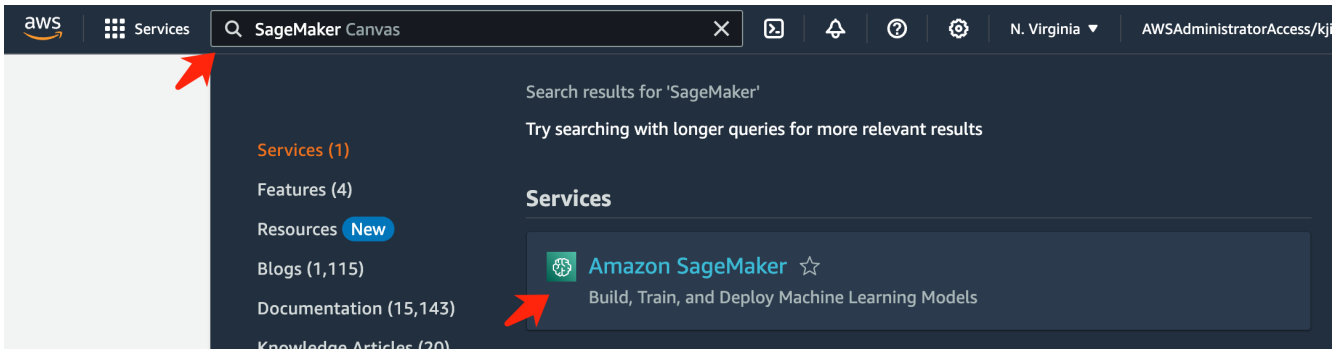
Para obtener más información sobre FSxN, por favor eche un vistazo a esta presentación ("Enlace de vídeo")

Guía del usuario

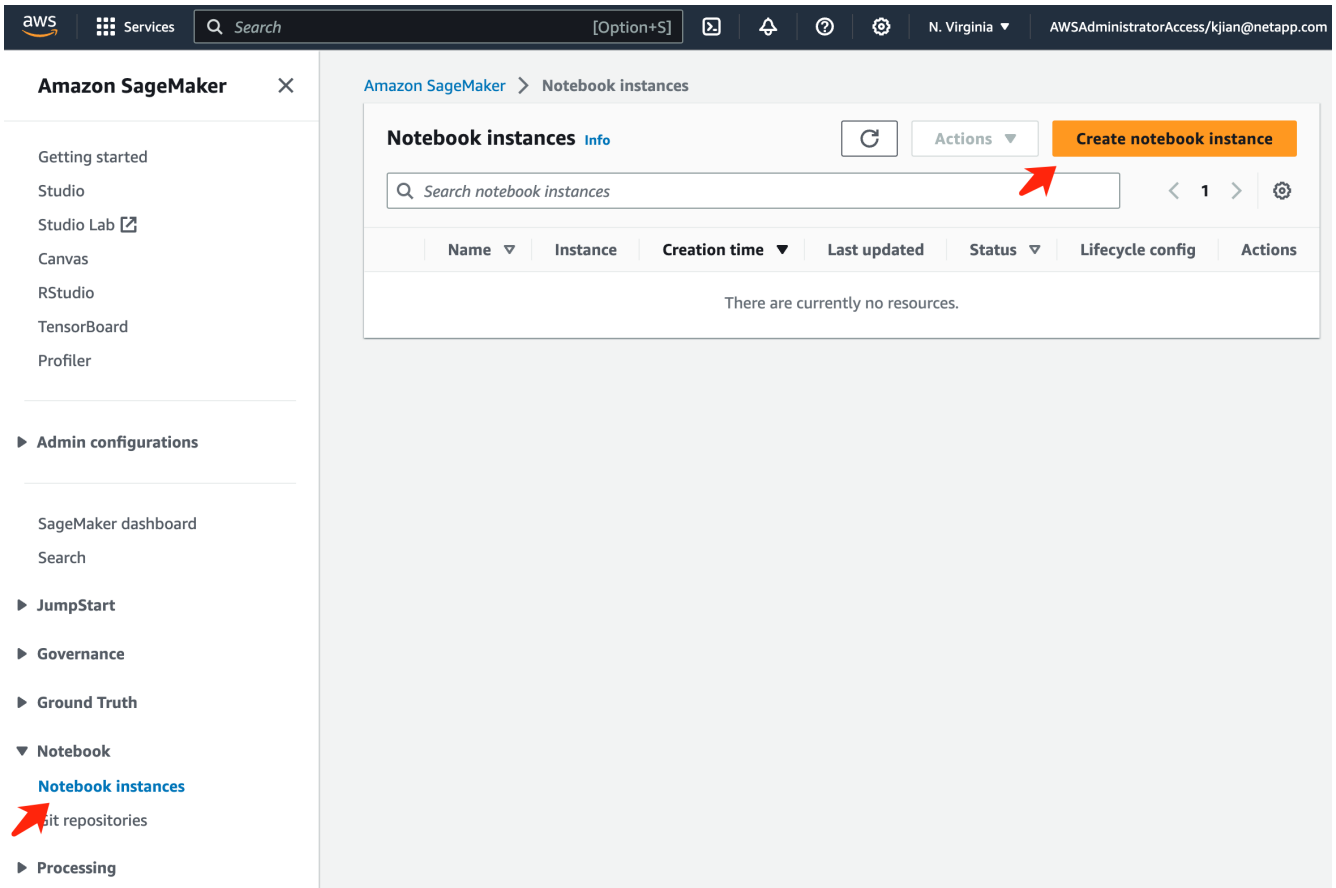
Creación de servidores

Cree una instancia de SageMaker Notebook

1. Abra la consola de AWS. En el panel de búsqueda, busque en SageMaker y haga clic en el servicio **Amazon SageMaker**.



2. Abra las instancias del bloc de notas * en la pestaña del bloc de notas, haga clic en el botón naranja * Crear instancia del bloc de notas *.



3. En la página de creación,
   Introduzca el nombre de la instancia de **Notebook**
   Expanda el panel **Network**
   Deje otras entradas predeterminadas y seleccione un **VPC**, **Subnet** y **Grupo(s) de seguridad**. (Este **VPC** y **Subnet** se utilizarán para crear el sistema de archivos FSxN más adelante)
   Haga clic en el botón naranja **Crear instancia de bloc de notas** en la parte inferior derecha.

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. Learn more

Notebook instance settings

Notebook instance name

fsxn-demo

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.t3.medium

Elastic Inference Learn more

none

Platform identifier Learn more

Amazon Linux 2, Jupyter Lab 3

▶ Additional configuration

Permissions and encryption

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the **AmazonSageMakerFullAccess** IAM policy attached.

AmazonSageMakerServiceCatalogProductsUseRole

Create role using the role creation wizard

Root access - *optional*

- (●) Enable - Give users root access to the notebook
- ( ) Disable - Don't give users root access to the notebook
  Lifecycle configurations always have root access

Encryption key - *optional*

Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption

▼ Network - *optional*

VPC - *optional*

Default vpc-0df3956ab1fca2ec9 (172.31.0.0/16)

Subnet

Choose a subnet in an availability zone supported by Amazon SageMaker.

subnet-00060df0d0f562672 (172.31.16.0/20) | us-east-1a

Security group(s)

sg-0a39b3985770e9256 (default) ✕

Direct internet access

- (●) Enable — Access the internet directly through Amazon SageMaker
- ( ) Disable — Access the internet through a VPC
  To train or host models from a notebook, you need internet access. To enable internet access, make sure that your VPC has a NAT gateway and your security group allows outbound connections. Learn more

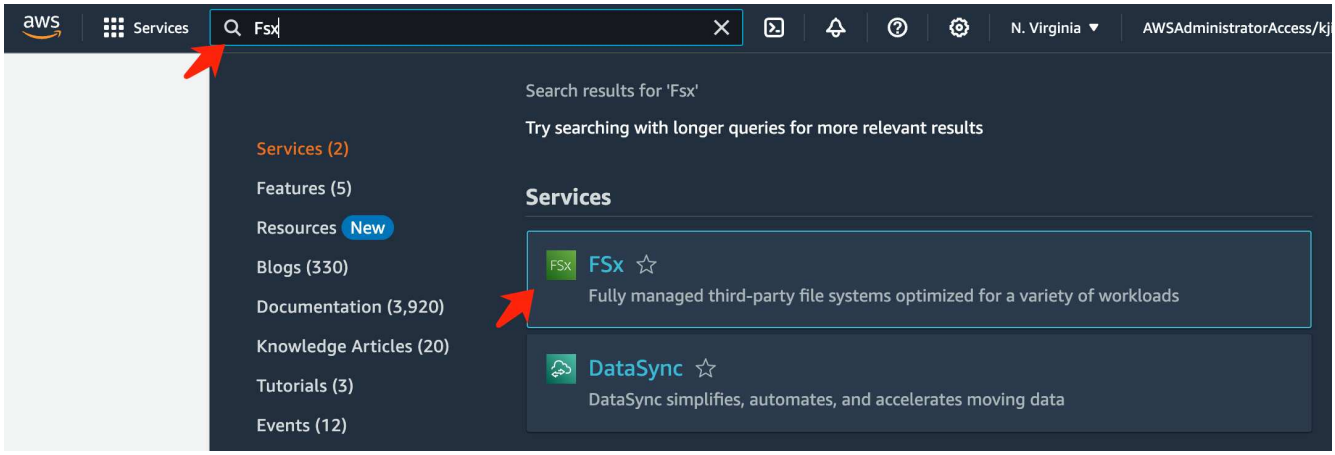
▶ Git repositories- *optional*

▶ Tags - *optional*

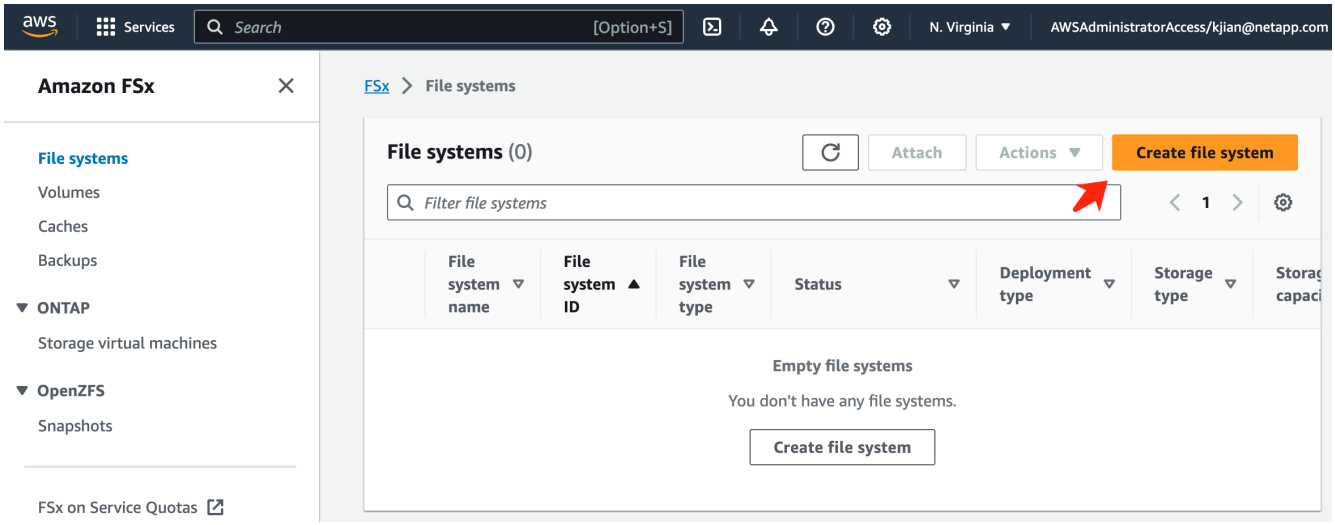
Cancel | Create notebook instance

Crear un sistema de archivos FSxN

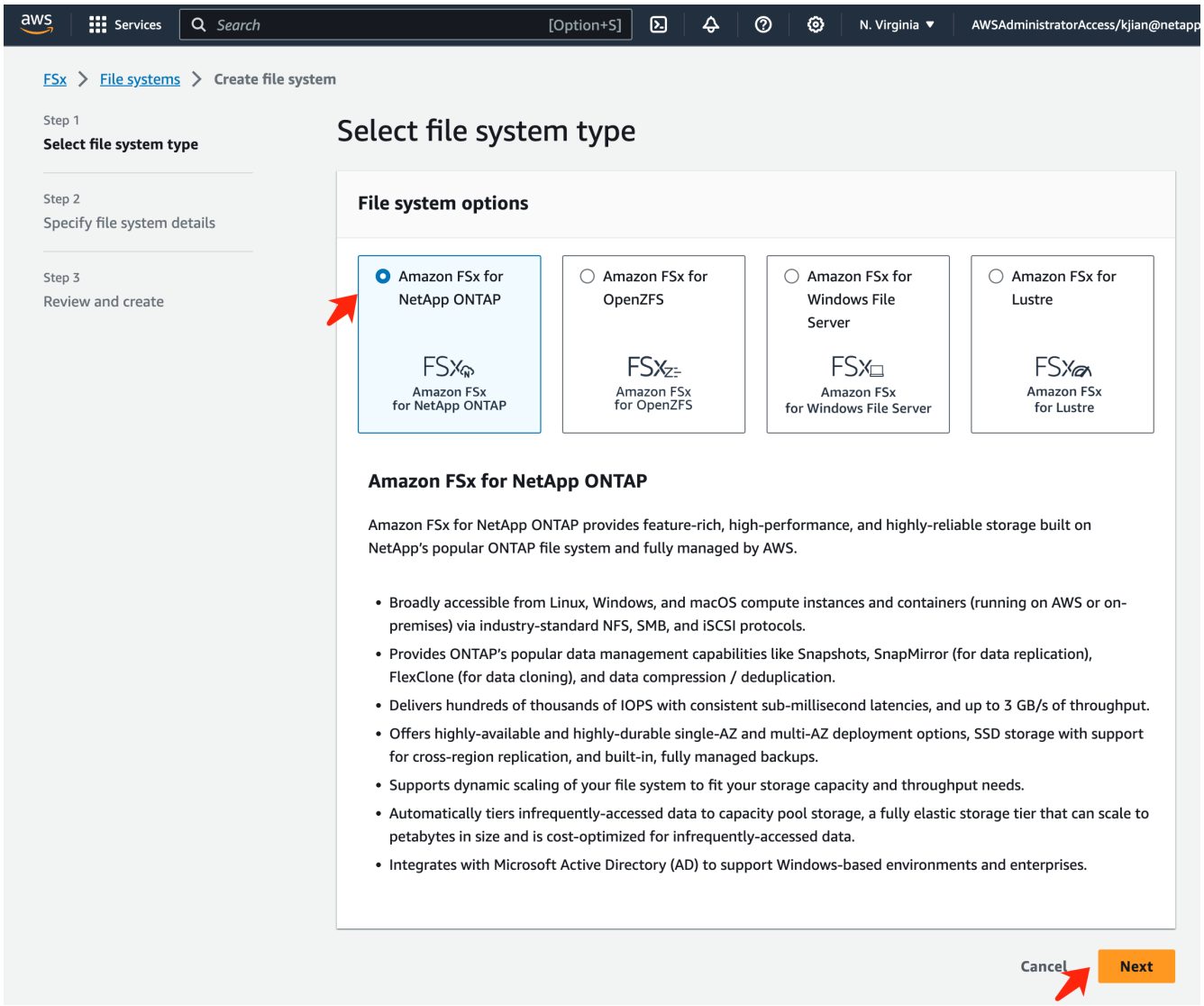
1. Abra la consola de AWS. En el panel de búsqueda, busca FSX y haz clic en el servicio **FSX**.



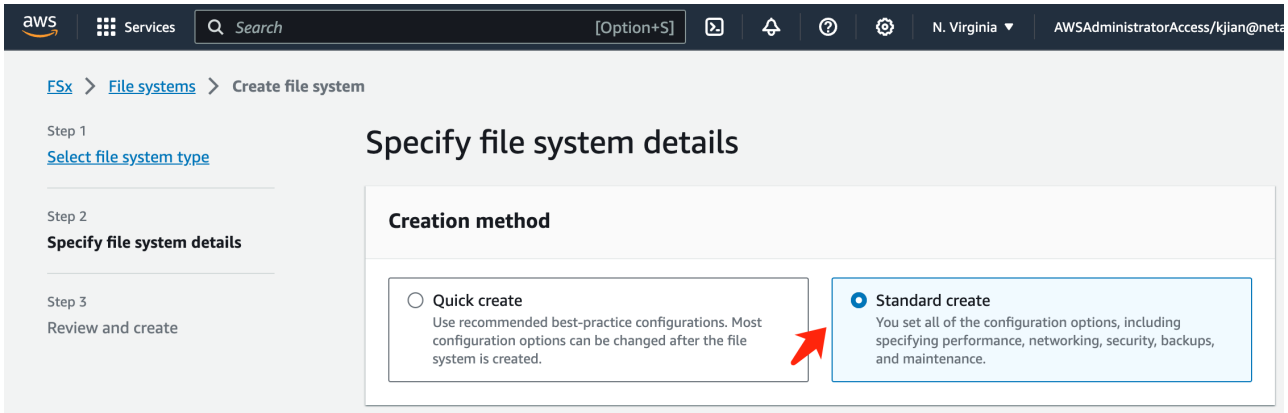
2. Haga clic en **Crear sistema de archivos**.



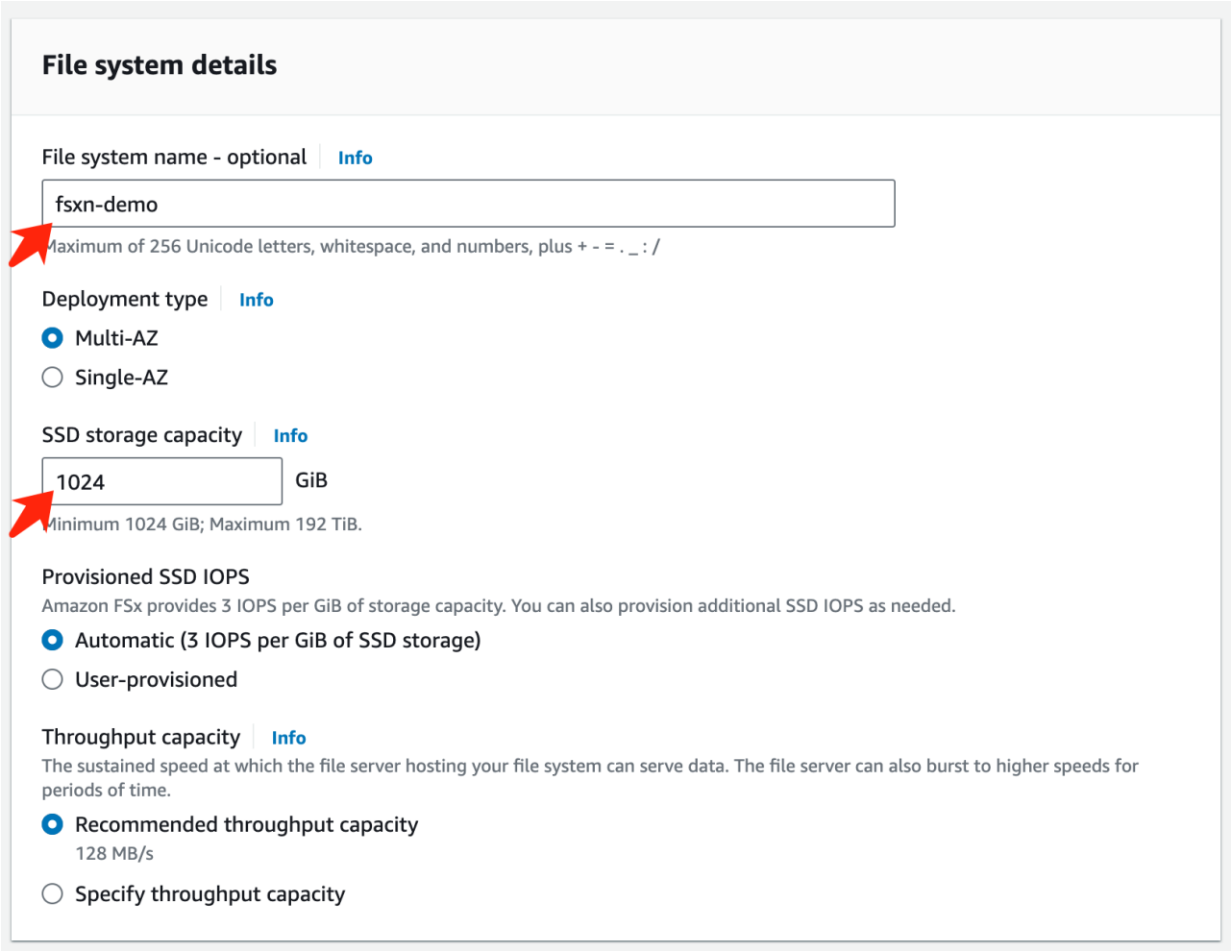
3. Seleccione la primera tarjeta **FSX for NetApp ONTAP** y haga clic en **Siguiente**.



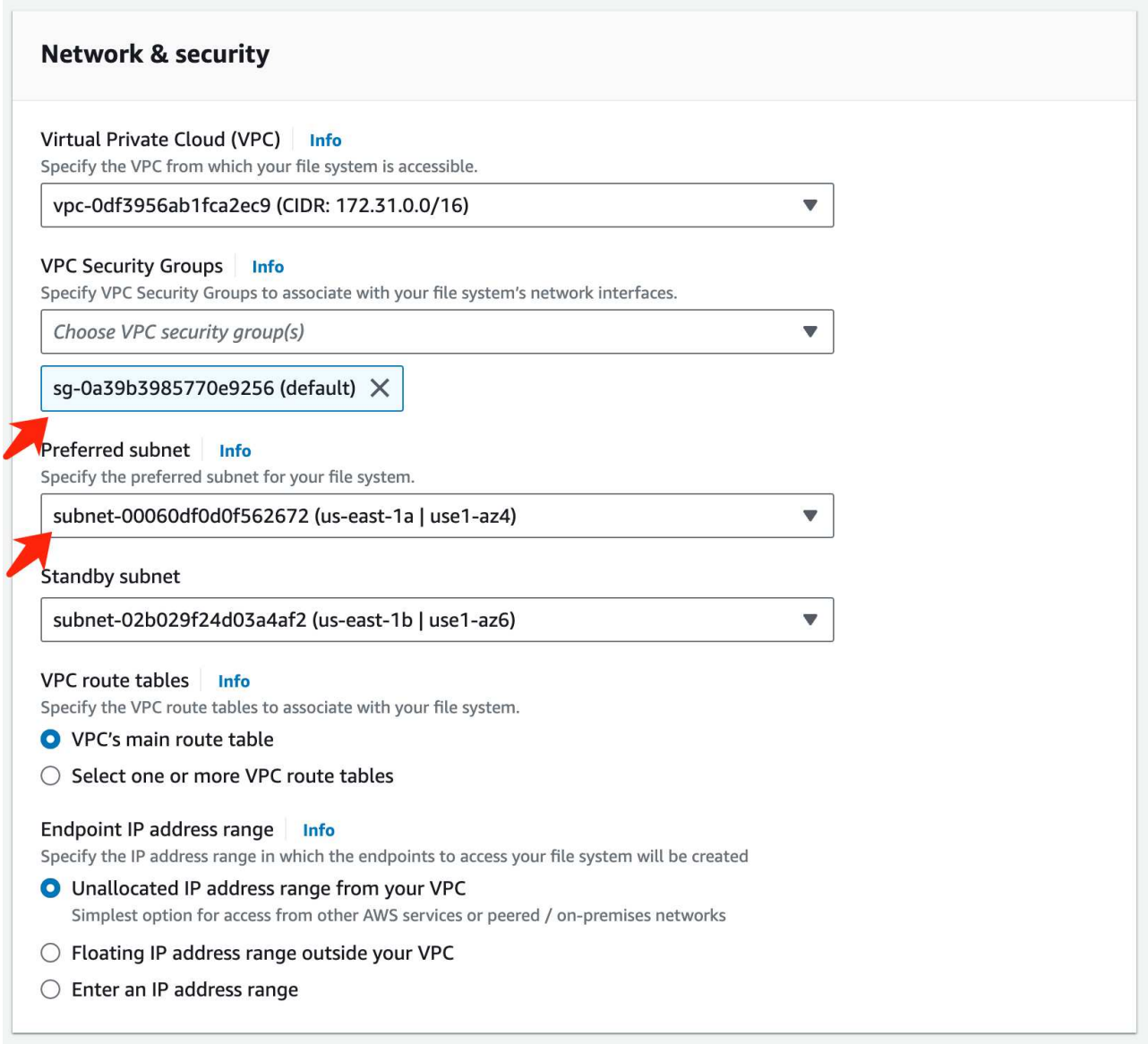
4. En la página de configuración de detalles.
   a. Seleccione la opción **Standard create**.



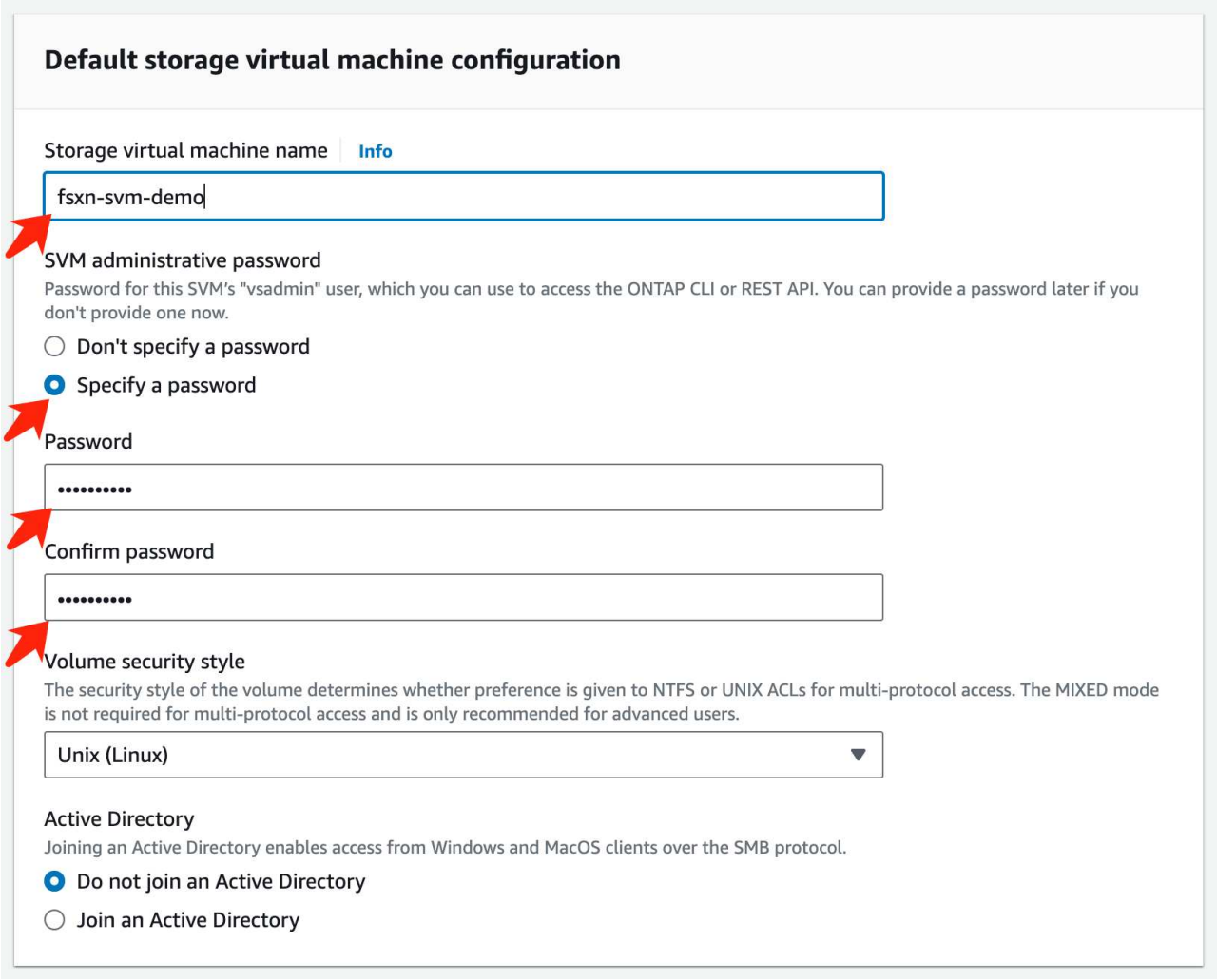
   b. Introduzca el **Nombre del sistema de archivos** y la **Capacidad de almacenamiento SSD**.



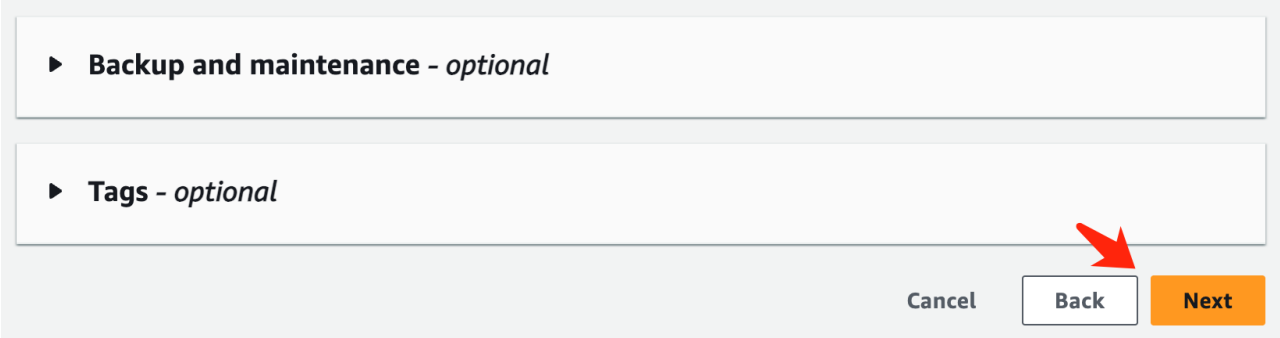
c. Asegúrese de usar **VPC** y **subnet** igual a la instancia **SageMaker Notebook**.



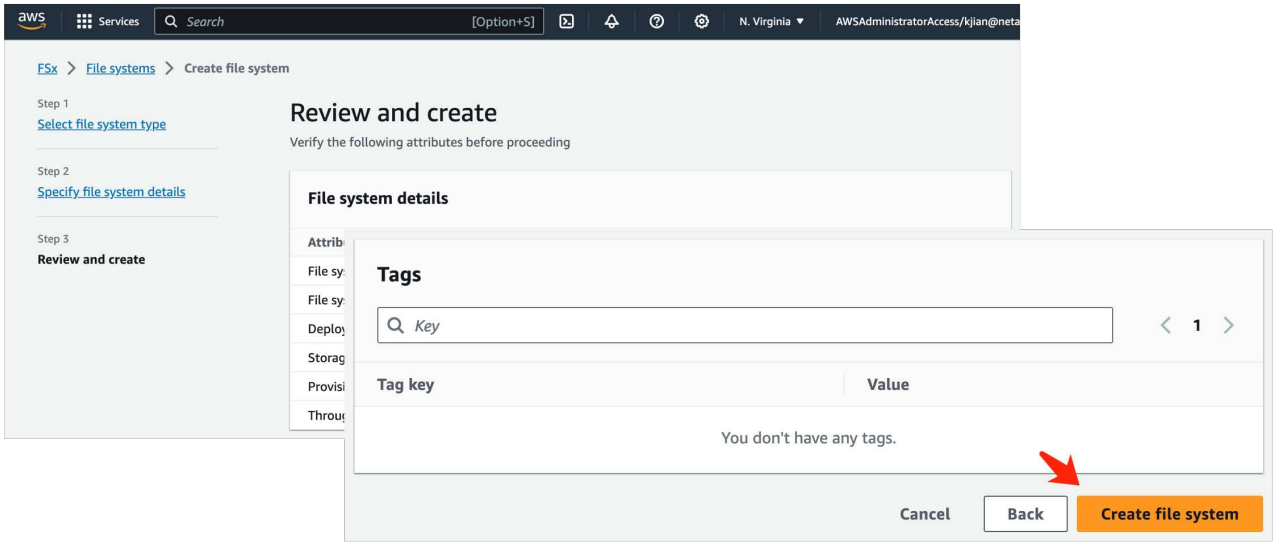
d. Introduzca el nombre de la máquina virtual **Storage** y **especifique una contraseña** para su SVM (máquina virtual de almacenamiento).



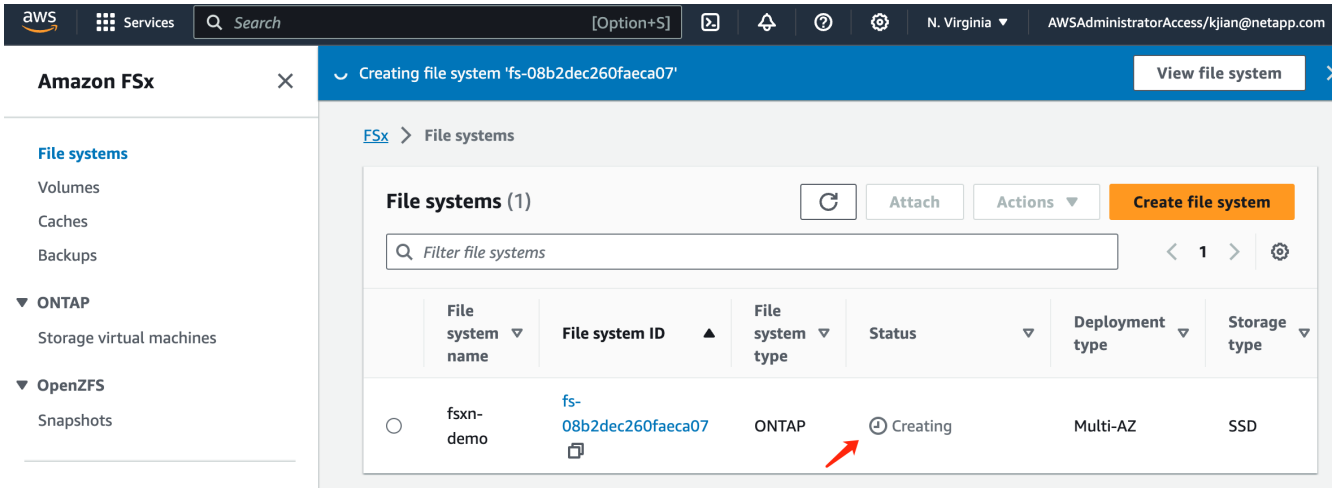
e. Deja otras entradas predeterminadas y haz clic en el botón naranja **Siguiente** en la parte inferior derecha.



f. Haga clic en el botón naranja **Crear sistema de archivos** en la parte inferior derecha de la página de revisión.



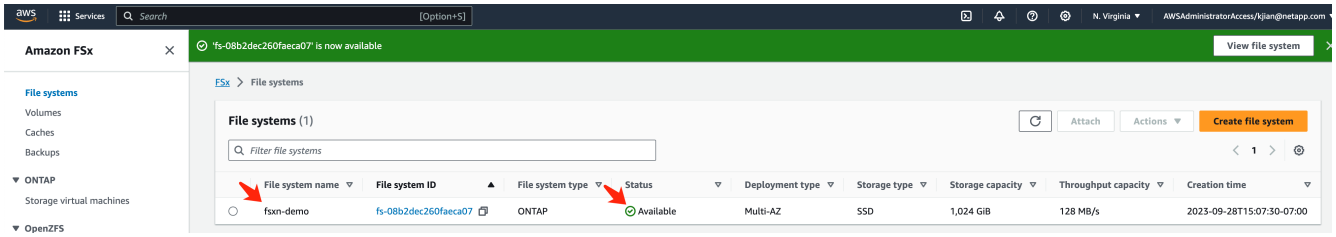
5. Puede tardar unos **20-40 minutos** en activar el sistema de archivos FSX.



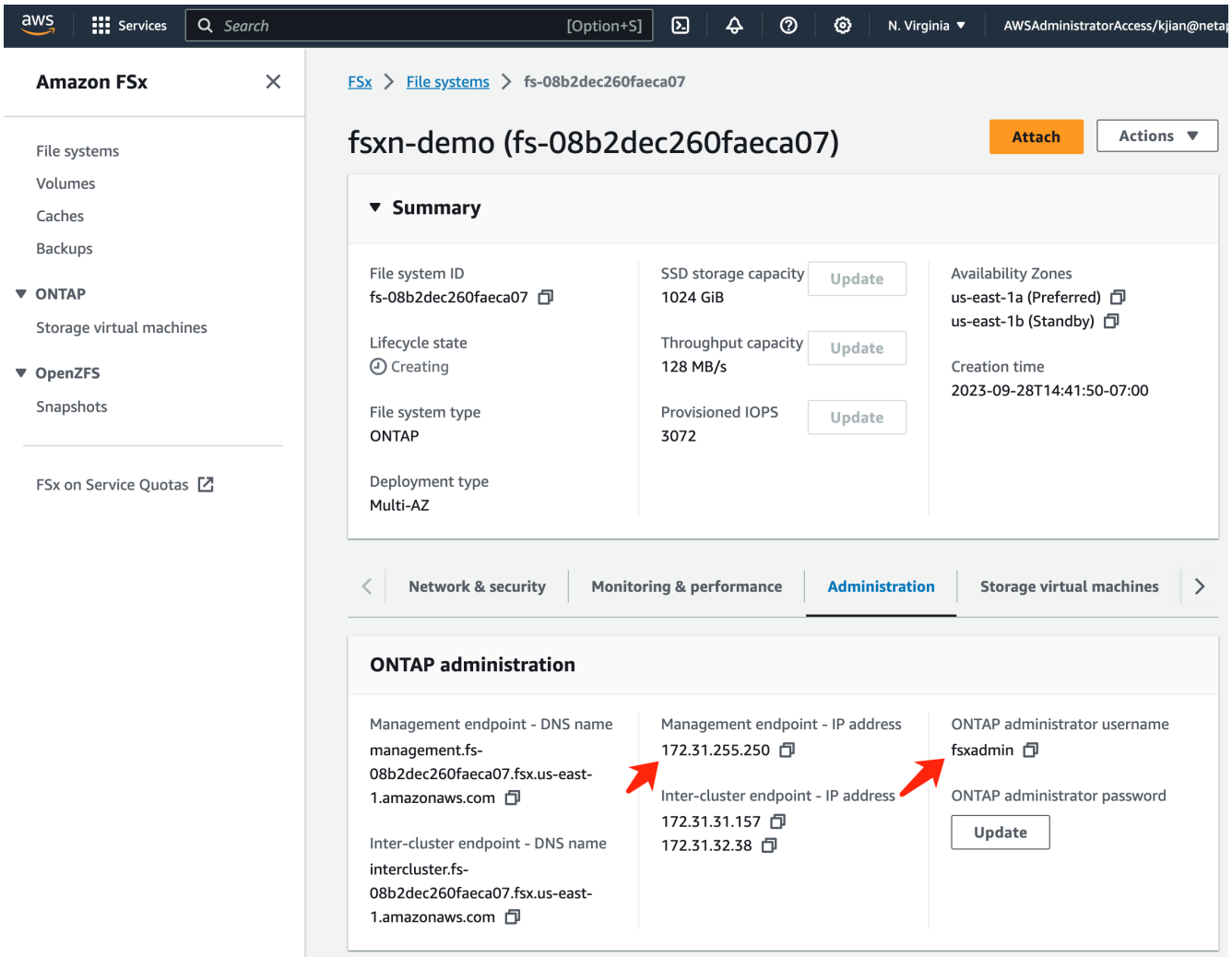
Configuración del servidor

Configuración de ONTAP

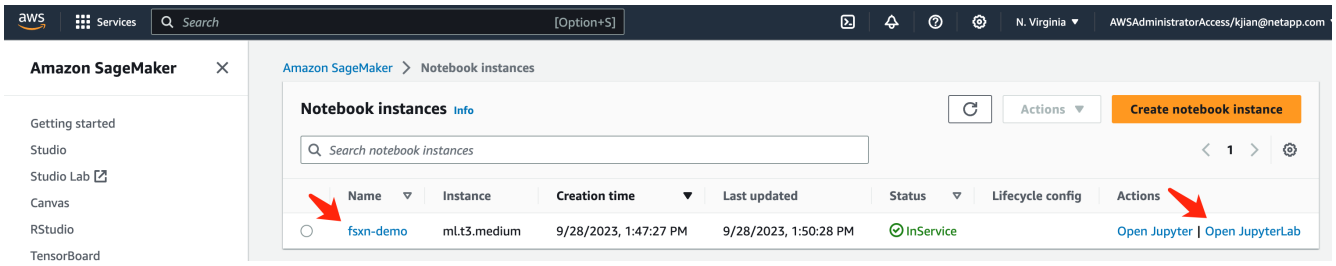
1. Abra el sistema de archivos FSX creado. Por favor, asegúrese de que el estado es **disponible**.



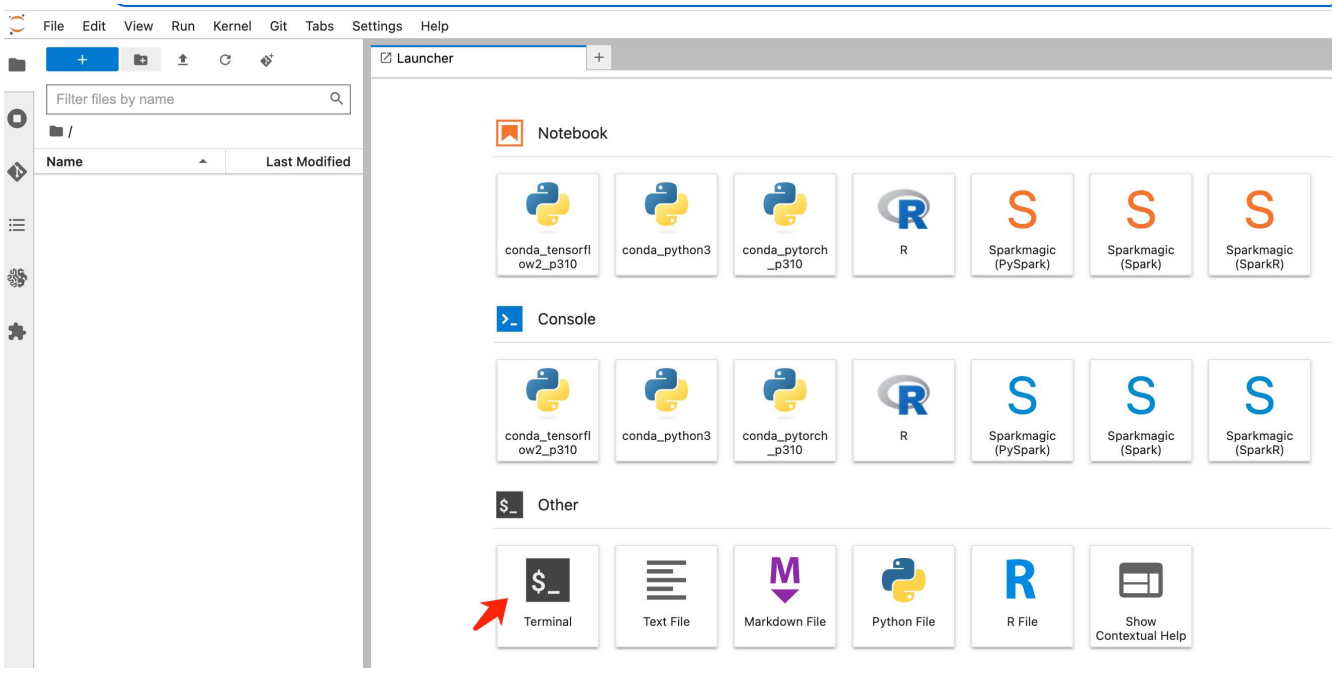
2. Seleccione la pestaña **Administración** y mantenga el **Punto final de administración - dirección IP** y el **Nombre de usuario del administrador de ONTAP**.



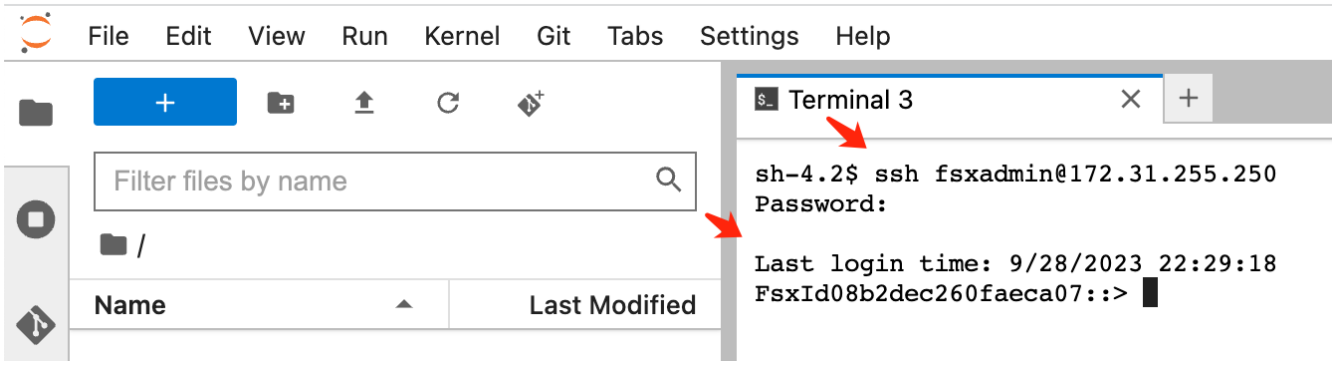
3. Abra la instancia creada de **SageMaker Notebook** y haga clic en **Abrir JupyterLab**.



4. En la página Jupyter Lab, abre un nuevo **Terminal**.

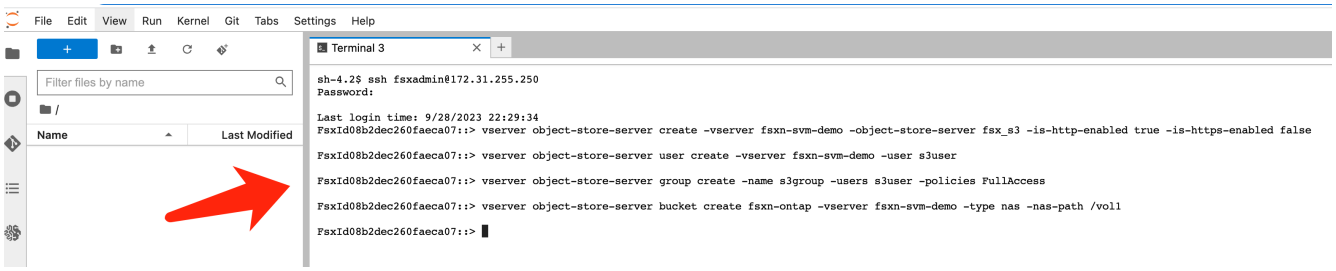


5. Introduzca el comando ssh ssh <nombre de usuario admin>@<IP del servidor de ONTAP> para iniciar sesión en el sistema de archivos ONTAP FSxN. (El nombre de usuario y la dirección IP se recuperan del paso 2)
   Utilice la contraseña utilizada al crear la **Storage virtual machine**.



6. Ejecute los comandos en el siguiente orden.
   Utilizamos **fsxn-ontap** como nombre para el **FSxN private S3 bucket name**.
   Utilice el **nombre de máquina virtual de almacenamiento** para el argumento **-Vserver**.

```
vserver object-store-server create -vserver fsxn-svm-demo -object-store
-server fsx_s3 -is-http-enabled true -is-https-enabled false

vserver object-store-server user create -vserver fsxn-svm-demo -user
s3user

vserver object-store-server group create -name s3group -users s3user
-policies FullAccess

vserver object-store-server bucket create fsxn-ontap -vserver fsxn-svm-
demo -type nas -nas-path /vol1
```



7. Ejecute los siguientes comandos para recuperar la IP de punto final y las credenciales para FSxN private S3.

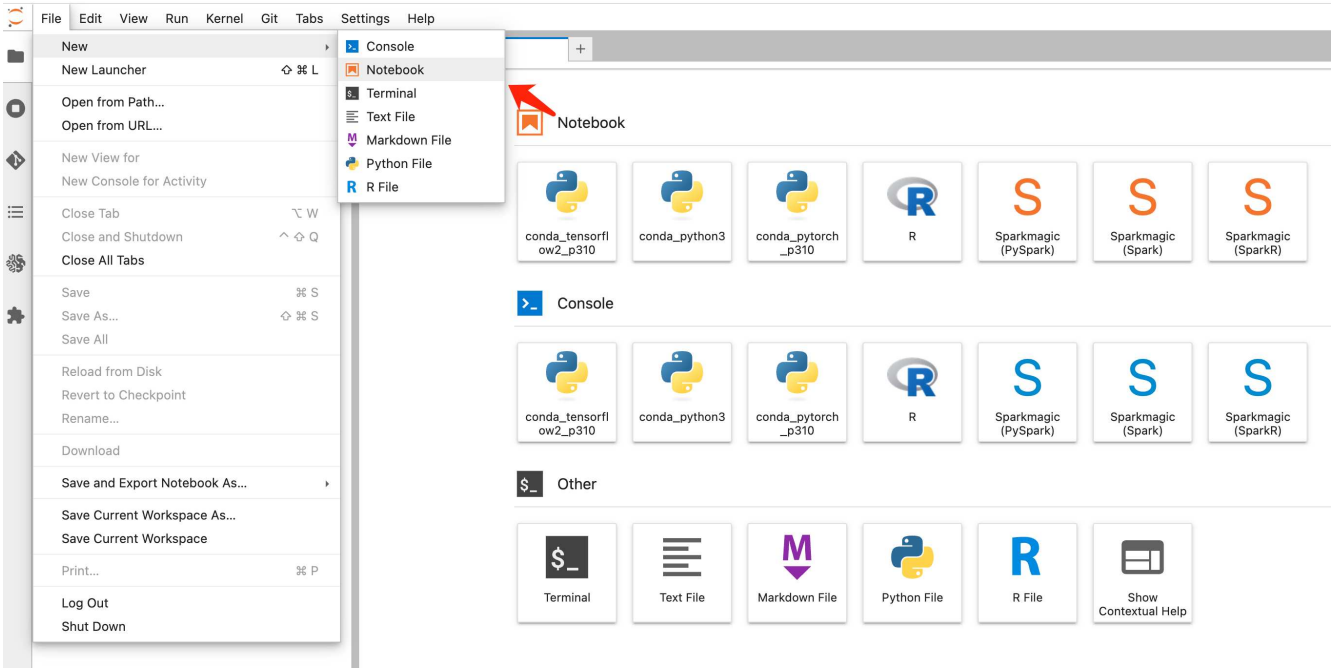
```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show
```

8. Conserve la IP del extremo y las credenciales para usarlo en el futuro.

```
sh-4.2$ ssh fsxadmin@172.31.255.250
Password:

Last login time: 9/28/2023 22:32:42
FsxId08b2dec260faeca07::> network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

                    Vserver Name: fsxn-svm-demo
            Logical Interface Name: nfs_smb_management_1
                    Service Policy: default-data-files
                      Service List: data-core, data-nfs, data-cifs,
                                    management-ssh, management-https,
                                    data-s3-server, data-dns-server
                (DEPRECATED)-Role: data
                  Data Protocol: nfs, cifs, s3
                  Network Address: [redacted]
                          Netmask: 255.255.255.192
              Bits in the Netmask: 26
                       Is VIP LIF: false
                      Subnet Name: -
                        Home Node: FsxId08b2dec260faeca07-01
                        Home Port: e0e
                     Current Node: FsxId08b2dec260faeca07-01
                     Current Port: e0e
                 Operational Status: up
                    Extended Status: -
                            Is Home: true
             Administrative Status: up
                   Failover Policy: system-defined
      (DEPRECATED)-Firewall Policy: data
                       Auto Revert: true
     Fully Qualified DNS Zone Name: none
           DNS Query Listen Enable: false
               Failover Group Name: Fsx
                          FCP WWPN: -
                     Address family: ipv4
                            Comment: -
                     IPspace of LIF: Default
    Is Dynamic DNS Update Enabled?: true
Probe-port for Cloud Load Balancer: -
                  Broadcast Domain: Fsx
                      Vserver Type: data
   Required RDMA offload protocols: -

FsxId08b2dec260faeca07::> set adv

Warning: These advanced commands are potentially dangerous; use them only when directed to do so by NetApp personnel.
Do you want to continue? {y|n}: y

FsxId08b2dec260faeca07::*> vserver object-store-server user show
Vserver     User            ID        Access Key           Secret Key
----------- --------------- --------- -------------------- --------------------
fsxn-svm-demo
            root            0         -                    -
   Comment: Root User
fsxn-svm-demo
            s3user          1         [redacted]           [redacted]

2 entries were displayed.

FsxId08b2dec260faeca07::*>
```

Configuración del cliente

1. En la instancia de SageMaker Notebook, cree un nuevo cuaderno Jupyter.



2. Utilice el siguiente código como solución alternativa para cargar archivos en el cubo privado de FSxN S3. Para obtener un ejemplo de código completo, consulte este cuaderno. "fsxn_demo.ipynb"

```
# Setup configurations
# -------- Manual configurations --------
seed: int = 77                                                  # Random
seed
bucket_name: str = 'fsxn-ontap'                                 # The bucket
name in ONTAP
aws_access_key_id = '<Your ONTAP bucket key id>'                # Please get
this credential from ONTAP
aws_secret_access_key = '<Your ONTAP bucket access key>'        # Please get
this credential from ONTAP
fsx_endpoint_ip: str = '<Your FSxN IP address>'                 # Please get
this IP address from FSXN
# -------- Manual configurations --------

# Workaround
## Permission patch
!mkdir -p vol1
!sudo mount -t nfs $fsx_endpoint_ip:/vol1 /home/ec2-user/SageMaker/vol1
!sudo chmod 777 /home/ec2-user/SageMaker/vol1

## Authentication for FSxN as a Private S3 Bucket
!aws configure set aws_access_key_id $aws_access_key_id
!aws configure set aws_secret_access_key $aws_secret_access_key

## Upload file to the FSxN Private S3 Bucket
%%capture
local_file_path: str = <Your local file path>

!aws s3 cp --endpoint-url http://$fsx_endpoint_ip /home/ec2-user
/SageMaker/$local_file_path  s3://$bucket_name/$local_file_path

# Read data from FSxN Private S3 bucket
## Initialize a s3 resource client
import boto3

# Get session info
region_name = boto3.session.Session().region_name

# Initialize Fsxn S3 bucket object
# --- Start integrating SageMaker with FSXN ---
# This is the only code change we need to incorporate SageMaker with
FSXN
```

```
s3_client: boto3.client = boto3.resource(
    's3',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key,
    use_ssl=False,
    endpoint_url=f'http://{fsx_endpoint_ip}',
    config=boto3.session.Config(
        signature_version='s3v4',
        s3={'addressing_style': 'path'}
    )
)
# --- End integrating SageMaker with FSXN ---

## Read file byte content
bucket = s3_client.Bucket(bucket_name)

binary_data = bucket.Object(data.filename).get()['Body']
```

Esto concluye la integración entre FSxN y la instancia de SageMaker.

Lista de comprobación de depuración útil

- Asegúrese de que la instancia de SageMaker Notebook y el sistema de archivos FSxN estén en la misma VPC.
- Recuerde ejecutar el comando **set dev** en ONTAP para establecer el nivel de privilegio en **dev**.

Preguntas frecuentes (a partir del 27 de septiembre de 2023)

P: ¿Por qué recibo el error "**Se ha producido un error (NotImplemented) al llamar a la operación CreateMultipartUpload: El comando S3 que solicitó no está implementado**" al cargar archivos a FSxN?

R: Como depósito privado de S3, FSxN admite la carga de archivos de hasta 100MB GB. Cuando se utiliza el protocolo S3, los archivos de más de 100MB MB se dividen en 100MB fragmentos y se llama a la función 'CreateMultipartUpload'. Sin embargo, la implementación actual de FSxN PRIVATE S3 no soporta esta función.

P: ¿Por qué recibo el error "**Se ha producido un error (ACCESSDENIED) al llamar a las operaciones PutObject: Acceso denegado**" al cargar archivos a FSxN?

R: Para acceder al bucket S3 privado FSxN desde una instancia de Notebook de SageMaker, cambie las credenciales de AWS a las credenciales FSxN. Sin embargo, otorgar permiso de escritura a la instancia requiere una solución provisional que implique montar el bucket y ejecutar el comando shell 'chmod' para cambiar los permisos.

P: ¿Cómo puedo integrar el cubo FSxN private S3 con otros servicios de SageMaker ML?

R: Desafortunadamente, el SDK de servicios de SageMaker no proporciona una forma de especificar el punto final para el cubo privado de S3. Como resultado, FSxN S3 no es compatible con los servicios de SageMaker tales como Sagemaker Data Wrangler, Sagemaker Clarify, Sagemaker Glue, Sagemaker Athena, Sagemaker

AutoML, Sagemaker AutoML, y otros.

Parte 2: Aprovechamiento de AWS FSx para NetApp ONTAP (FSxN) como fuente de datos para el entrenamiento de modelos en SageMaker

Autor(es):

Jian Jian (KEN), científico sénior de datos y aplicado, NetApp

Introducción

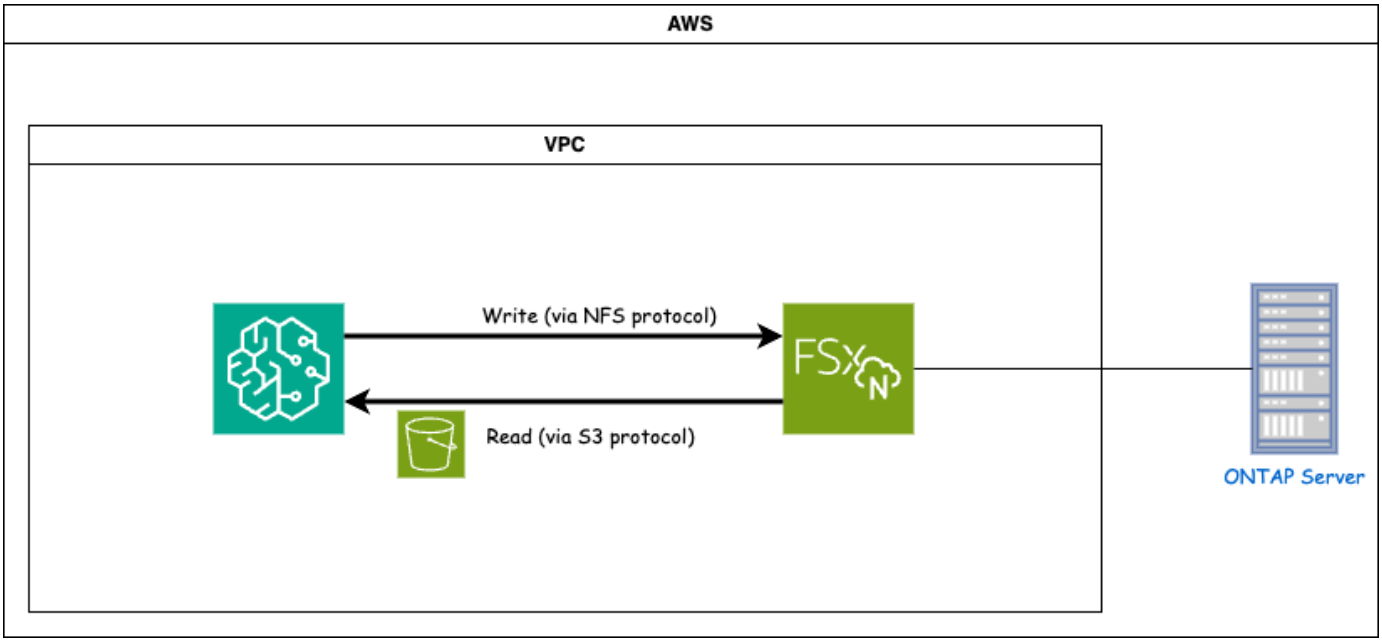
Este tutorial ofrece un ejemplo práctico de un proyecto de clasificación de visión por computadora, que proporciona experiencia práctica en la construcción de modelos ML que utilizan FSxN como fuente de datos dentro del entorno de SageMaker. El proyecto se centra en el uso de PyTorch, un marco de aprendizaje profundo, para clasificar la calidad de los neumáticos en función de las imágenes de los neumáticos. Hace hincapié en el desarrollo de modelos de aprendizaje automático utilizando FSxN como fuente de datos en Amazon SageMaker.

Qué es FSxN

Amazon FSx para NetApp ONTAP es, de hecho, una solución de almacenamiento totalmente gestionada que ofrece AWS. Aprovecha el sistema de archivos ONTAP de NetApp para ofrecer un almacenamiento fiable y de alto rendimiento. Con su compatibilidad con protocolos como NFS, SMB e iSCSI, permite un acceso fluido desde diferentes instancias de computación y contenedores. El servicio está diseñado para ofrecer un rendimiento excepcional, lo que garantiza operaciones de datos rápidas y eficaces. También ofrece alta disponibilidad y durabilidad, lo que garantiza que sus datos permanezcan accesibles y protegidos. Además, la capacidad de almacenamiento de Amazon FSx para NetApp ONTAP es escalable, lo que te permite ajustarla fácilmente según tus necesidades.

Requisito previo

Entorno de red



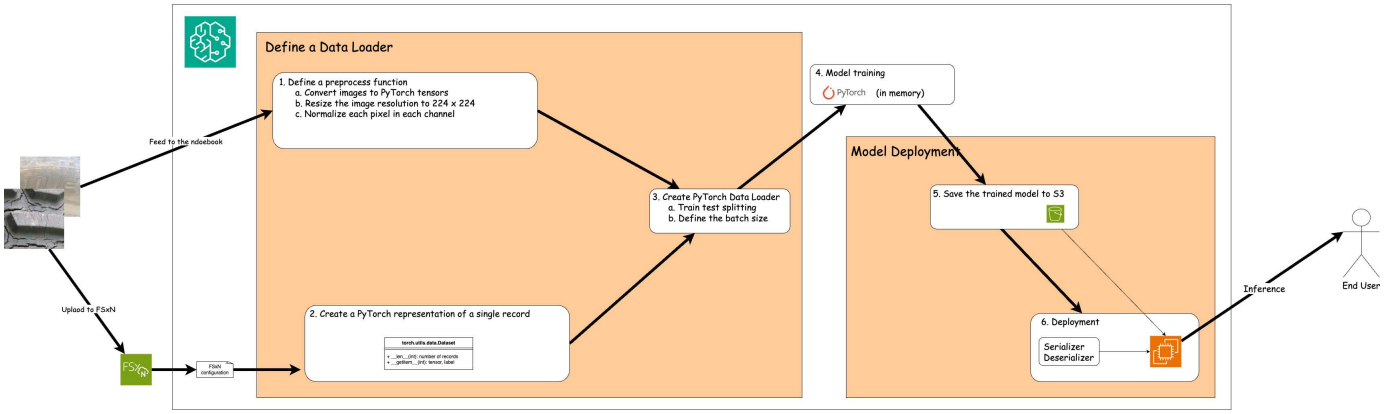
FSxN (Amazon FSx para NetApp ONTAP) es un servicio de almacenamiento de AWS. Incluye un sistema de archivos que se ejecuta en el sistema NetApp ONTAP y una máquina virtual de sistema gestionado por AWS (SVM) que se conecta a él. En el diagrama proporcionado, el servidor NetApp ONTAP gestionado por AWS se encuentra fuera del VPC. El SVM sirve como intermediario entre SageMaker y el sistema NetApp ONTAP, al recibir solicitudes de operaciones de SageMaker y reenviarlas al almacenamiento subyacente. Para acceder a FSxN, SageMaker debe colocarse dentro de la misma VPC que la implementación FSxN. Esta configuración garantiza la comunicación y el acceso a los datos entre SageMaker y FSxN.

Acceso a los datos

En escenarios del mundo real, los científicos de datos suelen utilizar los datos existentes almacenados en FSxN para crear sus modelos de aprendizaje automático. Sin embargo, a efectos de demostración, dado que el sistema de archivos FSxN está inicialmente vacío después de la creación, es necesario cargar manualmente los datos de entrenamiento. Esto se puede lograr mediante el montaje de FSxN como un volumen a SageMaker. Una vez que el sistema de archivos se ha montado correctamente, puede cargar su conjunto de datos en la ubicación montada, lo que lo hace accesible para el entrenamiento de sus modelos dentro del entorno de SageMaker. Este enfoque le permite aprovechar la capacidad de almacenamiento y las capacidades de FSxN mientras trabaja con SageMaker para el desarrollo y entrenamiento de modelos.

El proceso de lectura de datos implica la configuración de FSxN como un bucket S3 privado. Para obtener más información sobre las instrucciones de configuración detalladas, consulte "1 parte: Integración de AWS FSx para NetApp ONTAP (FSxN) como bloque de S3 privado en AWS SageMaker"

Visión General de la Integración



El flujo de trabajo del uso de datos de entrenamiento en FSxN para construir un modelo de aprendizaje profundo en SageMaker se puede resumir en tres pasos principales: Definición de cargador de datos, entrenamiento de modelos e implementación. En líneas generales, estos pasos forman la base de una canalización de MLOps. Sin embargo, cada paso implica varios subpasos detallados para una implementación integral. Estos subpasos abarcan diversas tareas, como el preprocesamiento de datos, la división de conjuntos de datos, la configuración del modelo, el ajuste de hiperparámetros, la evaluación de modelos, y la puesta en marcha de modelos. Estos pasos garantizan un proceso completo y eficaz para construir e implementar modelos de aprendizaje profundo utilizando datos de entrenamiento de FSxN dentro del entorno de SageMaker.

Integración paso a paso

Cargador de datos

Para entrenar una red de aprendizaje profundo de PyTorch con datos, se crea un cargador de datos para facilitar la alimentación de datos. El cargador de datos no sólo define el tamaño del lote, sino que también

determina el procedimiento para leer y preprocesar cada registro del lote. Al configurar el cargador de datos, podemos manejar el procesamiento de datos en lotes, lo que permite el entrenamiento de la red de aprendizaje profundo.

El cargador de datos consta de 3 partes.

Función de preprocesamiento

```
from torchvision import transforms

preprocess = transforms.Compose([
    transforms.ToTensor(),
    transforms.Resize((224,224)),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])
```

El fragmento de código anterior demuestra la definición de las transformaciones de preprocesamiento de imágenes utilizando el módulo **torchvision.transform**. En este tutorial, se crea el objeto de preproceso para aplicar una serie de transformaciones. En primer lugar, la transformación **ToTensor()** convierte la imagen en una representación tensora. Posteriormente, la transformación **Resize 224.224** cambia el tamaño de la imagen a un tamaño fijo de 224x224 píxeles. Finalmente, la transformación **Normalize()** normaliza los valores del tensor restando la media y dividiendo por la desviación estándar a lo largo de cada canal. Los valores de desviación media y estándar utilizados para la normalización se emplean comúnmente en modelos de redes neuronales pre-entrenados. En general, este código prepara los datos de la imagen para su posterior procesamiento o entrada en un modelo preentrenado convirtiéndolo en un tensor, ajustándolo y normalizando los valores de píxeles.

La clase de conjunto de datos de PyTorch

```python
import torch
from io import BytesIO
from PIL import Image


class FSxNImageDataset(torch.utils.data.Dataset):
    def __init__(self, bucket, prefix='', preprocess=None):
        self.image_keys = [
            s3_obj.key
            for s3_obj in list(bucket.objects.filter(Prefix=prefix).all())
        ]
        self.preprocess = preprocess

    def __len__(self):
        return len(self.image_keys)

    def __getitem__(self, index):
        key = self.image_keys[index]
        response = bucket.Object(key)

        label = 1 if key[13:].startswith('defective') else 0

        image_bytes = response.get()['Body'].read()
        image = Image.open(BytesIO(image_bytes))
        if image.mode == 'L':
            image = image.convert('RGB')

        if self.preprocess is not None:
            image = self.preprocess(image)
        return image, label
```

Esta clase proporciona funcionalidad para obtener el número total de registros en el conjunto de datos y define el método para leer datos para cada registro. Dentro de la función ***getitem***, el código utiliza el objeto bucket boto3 S3 para recuperar los datos binarios de FSxN. El estilo de código para acceder a los datos de FSxN es similar a la lectura de datos de Amazon S3. La explicación subsiguiente profundiza en el proceso de creación del objeto privado S3 **bucket**.

FSxN como repositorio S3 privado

```
seed = 77                                                    # Random seed
bucket_name = '<Your ONTAP bucket name>'                     # The bucket
name in ONTAP
aws_access_key_id = '<Your ONTAP bucket key id>'             # Please get
this credential from ONTAP
aws_secret_access_key = '<Your ONTAP bucket access key>'     # Please get
this credential from ONTAP
fsx_endpoint_ip = '<Your FSxN IP address>'                   # Please get
this IP address from FSXN
```

```
import boto3

# Get session info
region_name = boto3.session.Session().region_name

# Initialize Fsxn S3 bucket object
# --- Start integrating SageMaker with FSXN ---
# This is the only code change we need to incorporate SageMaker with FSXN
s3_client: boto3.client = boto3.resource(
    's3',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key,
    use_ssl=False,
    endpoint_url=f'http://{fsx_endpoint_ip}',
    config=boto3.session.Config(
        signature_version='s3v4',
        s3={'addressing_style': 'path'}
    )
)
# s3_client = boto3.resource('s3')
bucket = s3_client.Bucket(bucket_name)
# --- End integrating SageMaker with FSXN ---
```

Para leer datos de FSxN en SageMaker, se crea un manejador que apunta al almacenamiento FSxN mediante el protocolo S3. Esto permite que FSxN se trate como un cubo privado de S3. La configuración del manejador incluye especificar la dirección IP de la SVM FSxN, el nombre del depósito y las credenciales necesarias. Para obtener una explicación completa sobre la obtención de estos elementos de configuración, consulte el documento en "1 parte: Integración de AWS FSx para NetApp ONTAP (FSxN) como bloque de S3 privado en AWS SageMaker".

En el ejemplo mencionado anteriormente, el objeto bucket se utiliza para instanciar el objeto de conjunto de datos PyTorch. El objeto del conjunto de datos se explicará con más detalle en la sección siguiente.

El cargador de datos de PyTorch

```
from torch.utils.data import DataLoader
torch.manual_seed(seed)

# 1. Hyperparameters
batch_size = 64

# 2. Preparing for the dataset
dataset = FSxNImageDataset(bucket, 'dataset/tyre', preprocess=preprocess)

train, test = torch.utils.data.random_split(dataset, [1500, 356])

data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
```

En el ejemplo proporcionado, se especifica un tamaño de lote de 64, lo que indica que cada lote contendrá 64 registros. Al combinar la clase PyTorch **Dataset**, la función de preprocesamiento y el tamaño de lote de entrenamiento, obtenemos el cargador de datos para el entrenamiento. Este cargador de datos facilita el proceso de iteración por el conjunto de datos en lotes durante la fase de entrenamiento.

Entrenamiento de modelos

```
from torch import nn


class TyreQualityClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3,32,(3,3)),
            nn.ReLU(),
            nn.Conv2d(32,32,(3,3)),
            nn.ReLU(),
            nn.Conv2d(32,64,(3,3)),
            nn.ReLU(),
            nn.Flatten(),
            nn.Linear(64*(224-6)*(224-6),2)
        )
    def forward(self, x):
        return self.model(x)
```

```
import datetime

num_epochs = 2
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = TyreQualityClassifier()
fn_loss = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)


model.to(device)
for epoch in range(num_epochs):
    for idx, (X, y) in enumerate(data_loader):
        X = X.to(device)
        y = y.to(device)

        y_hat = model(X)

        loss = fn_loss(y_hat, y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        current_time = datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
        print(f"Current Time: {current_time} - Epoch [{epoch+1}/
{num_epochs}]- Batch [{idx + 1}] - Loss: {loss}", end='\r')
```

Este código implementa un proceso de entrenamiento estándar de PyTorch. Define un modelo de red neuronal llamado **TyreQualityClassifier** usando capas convolucionales y una capa lineal para clasificar la calidad de los neumáticos. El bucle de entrenamiento itera sobre los lotes de datos, calcula la pérdida y actualiza los parámetros del modelo mediante retropropagación y optimización. Además, imprime la hora actual, la época, el lote y la pérdida con fines de monitorización.

Puesta en marcha de modelos

Puesta en marcha

```python
import io
import os
import tarfile
import sagemaker

# 1. Save the PyTorch model to memory
buffer_model = io.BytesIO()
traced_model = torch.jit.script(model)
torch.jit.save(traced_model, buffer_model)

# 2. Upload to AWS S3
sagemaker_session = sagemaker.Session()
bucket_name_default = sagemaker_session.default_bucket()
model_name = f'tyre_quality_classifier.pth'

# 2.1. Zip PyTorch model into tar.gz file
buffer_zip = io.BytesIO()
with tarfile.open(fileobj=buffer_zip, mode="w:gz") as tar:
    # Add PyTorch pt file
    file_name = os.path.basename(model_name)
    file_name_with_extension = os.path.split(file_name)[-1]
    tarinfo = tarfile.TarInfo(file_name_with_extension)
    tarinfo.size = len(buffer_model.getbuffer())
    buffer_model.seek(0)
    tar.addfile(tarinfo, buffer_model)

# 2.2. Upload the tar.gz file to S3 bucket
buffer_zip.seek(0)
boto3.resource('s3') \
    .Bucket(bucket_name_default) \
    .Object(f'pytorch/{model_name}.tar.gz') \
    .put(Body=buffer_zip.getvalue())
```

El código guarda el modelo de PyTorch en **Amazon S3** porque SageMaker requiere que el modelo se almacene en S3 para su implementación. Al subir el modelo a **Amazon S3**, se vuelve accesible para SageMaker, lo que permite la implementación e inferencia en el modelo desplegado.

```python
import time
from sagemaker.pytorch import PyTorchModel
from sagemaker.predictor import Predictor
from sagemaker.serializers import IdentitySerializer
from sagemaker.deserializers import JSONDeserializer


class TyreQualitySerializer(IdentitySerializer):
```

```
    CONTENT_TYPE = 'application/x-torch'

    def serialize(self, data):
        transformed_image = preprocess(data)
        tensor_image = torch.Tensor(transformed_image)

        serialized_data = io.BytesIO()
        torch.save(tensor_image, serialized_data)
        serialized_data.seek(0)
        serialized_data = serialized_data.read()

        return serialized_data


class TyreQualityPredictor(Predictor):
    def __init__(self, endpoint_name, sagemaker_session):
        super().__init__(
            endpoint_name,
            sagemaker_session=sagemaker_session,
            serializer=TyreQualitySerializer(),
            deserializer=JSONDeserializer(),
        )

sagemaker_model = PyTorchModel(
    model_data=f's3://{bucket_name_default}/pytorch/{model_name}.tar.gz',
    role=sagemaker.get_execution_role(),
    framework_version='2.0.1',
    py_version='py310',
    predictor_cls=TyreQualityPredictor,
    entry_point='inference.py',
    source_dir='code',
)

timestamp = int(time.time())
pytorch_endpoint_name = '{}-{}-{}'.format('tyre-quality-classifier', 'pt',
timestamp)
sagemaker_predictor = sagemaker_model.deploy(
    initial_instance_count=1,
    instance_type='ml.p3.2xlarge',
    endpoint_name=pytorch_endpoint_name
)
```

Este código facilita el despliegue de un modelo PyTorch en SageMaker. Define un serializador personalizado, **TyreQualitySerializer**, que preprocesa y serializa los datos de entrada como un tensor PyTorch. La clase **TyreQualityPredictor** es un predictor personalizado que utiliza el serializador definido y un **JSONDeserializer**. El código también crea un objeto **PyTorchModel** para especificar la ubicación S3 del modelo, el rol IAM, la versión del marco y el punto de entrada para la inferencia. El código genera una marca

de tiempo y construye un nombre de punto final basado en el modelo y la marca de tiempo. Por último, el modelo se despliega mediante el método de despliegue, especificando el recuento de instancias, el tipo de instancia y el nombre de punto final generado. Esto permite que el modelo de PyTorch se despliegue y sea accesible para la inferencia en SageMaker.

Inferencia

```
image_object = list(bucket.objects.filter('dataset/tyre'))[0].get()
image_bytes = image_object['Body'].read()

with Image.open(with Image.open(BytesIO(image_bytes)) as image:
    predicted_classes = sagemaker_predictor.predict(image)

    print(predicted_classes)
```

Este es el ejemplo de utilizar el punto final desplegado para llevar a cabo la inferencia.

Parte 3: Creación de Una canalización simplificada de MLOps (CI/CT/CD)

Autor(es):

Jian Jian (KEN), científico sénior de datos y aplicado, NetApp

Introducción

En este tutorial, aprenderá cómo aprovechar varios servicios de AWS para construir una canalización simple de MLOps que abarque la integración continua (CI), el entrenamiento continuo (CT) y la implementación continua (CD). A diferencia de las canalizaciones tradicionales de DevOps, MLOps requiere consideraciones adicionales para completar el ciclo operativo. Al seguir este tutorial, obtendrá información sobre la incorporación de CT en el bucle de MLOps, lo que permite el entrenamiento continuo de sus modelos y la implementación sin problemas para la inferencia. El tutorial le guiará a través del proceso de uso de los servicios de AWS para establecer este pipeline de MLOps de extremo a extremo.

Manifiesto

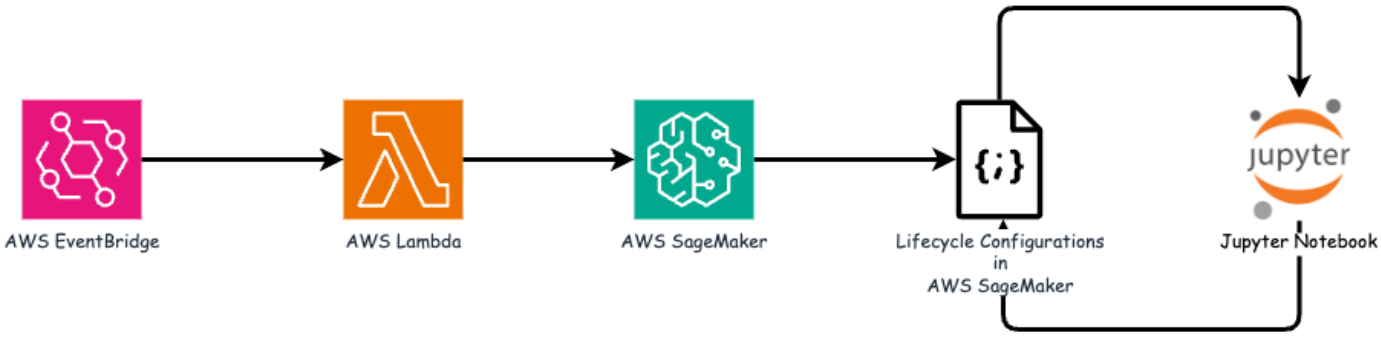
Funcionalidad	Nombre	Comentar
Almacenamiento de datos	FSxN de AWS	Consulte "1 parte: Integración de AWS FSx para NetApp ONTAP (FSxN) como bloque de S3 privado en AWS SageMaker".
IDE de ciencia de datos	SageMaker de AWS	Este tutorial se basa en el cuaderno Jupyter que se presenta en la "Parte 2: Aprovechamiento de AWS FSx para NetApp ONTAP (FSxN) como fuente de datos para el entrenamiento de modelos en SageMaker".

Funcionalidad	Nombre	Comentar
Función para activar el pipeline de MLOps	Función AWS Lambda	-
Disparador de trabajo CRON	EventBridge de AWS	-
Marco de aprendizaje profundo	PyTorch	-
SDK de AWS Python	boto3	-
Lenguaje de programación	Python	v3,10

Requisito previo

- Un sistema de archivos FSxN preconfigurado. Este tutorial utiliza los datos almacenados en FSxN para el proceso de entrenamiento.
- Una instancia **SageMaker Notebook** que está configurada para compartir la misma VPC que el sistema de archivos FSxN mencionado anteriormente.
- Antes de activar la función **AWS Lambda**, asegúrese de que la instancia **SageMaker Notebook** esté en estado **Detenido**.
- El tipo de instancia **ML.g4dn.xlarge** es necesario para aprovechar la aceleración de GPU necesaria para los cálculos de redes neuronales profundas.

Arquitectura



Esta canalización de MLOps es una implementación práctica que utiliza un trabajo cron para activar una función sin servidor, que a su vez ejecuta un servicio de AWS registrado con una función de devolución de llamada de ciclo de vida. El **AWS EventBridge** actúa como el trabajo cron. Invoca periódicamente una función **AWS Lambda** responsable de reciclar y reimplementar el modelo. Este proceso implica poner en marcha la instancia de **AWS SageMaker Notebook** para realizar las tareas necesarias.

Configuración paso a paso

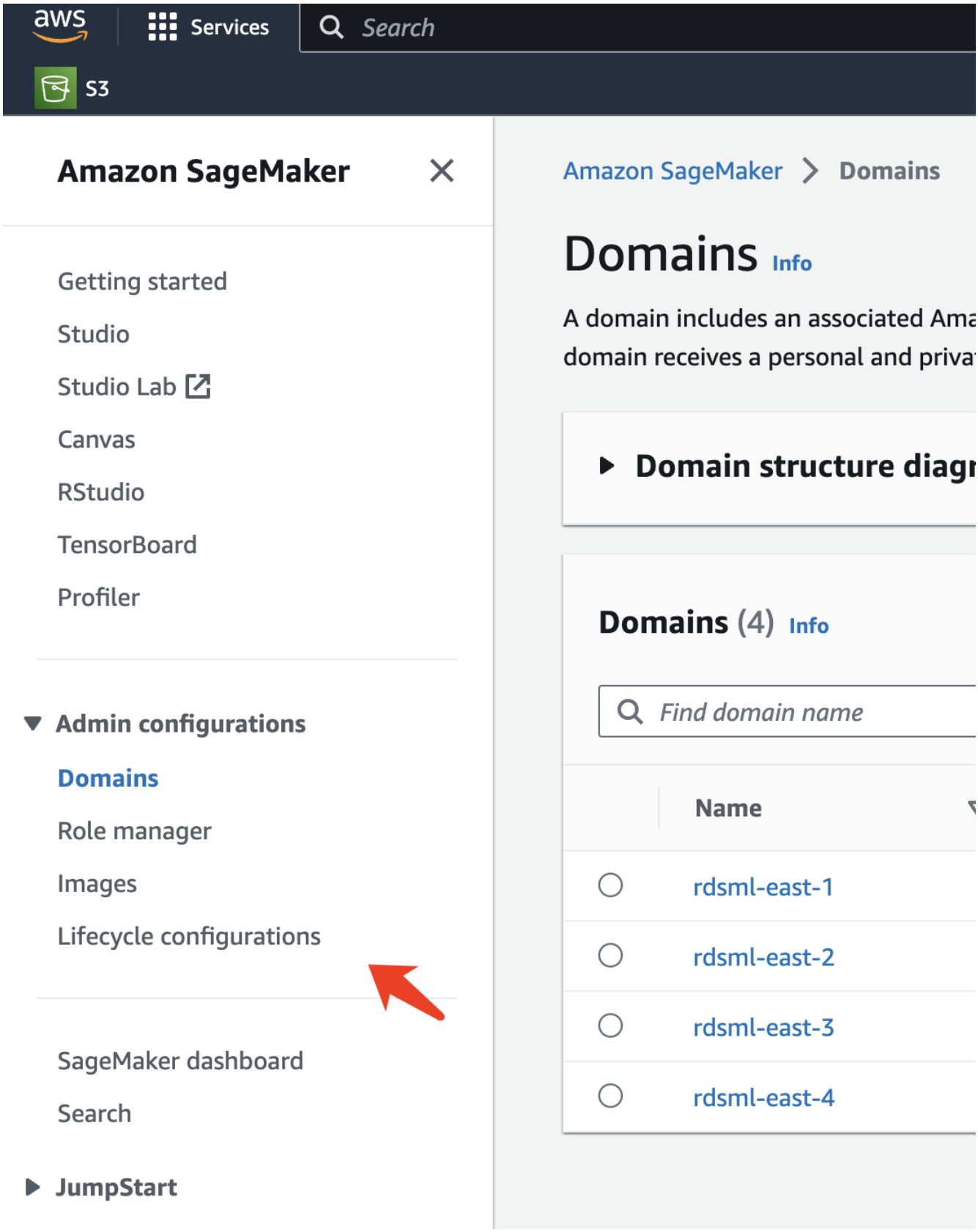
Configuraciones de ciclo de vida

Para configurar la función de devolución de llamada de ciclo de vida para la instancia de AWS SageMaker Notebook, utilizaría **Configuraciones de ciclo de vida**. Este servicio le permite definir las acciones necesarias que se deben realizar durante el giro de la instancia del bloc de notas. Específicamente, se puede implementar un script de shell dentro de las configuraciones de ciclo de vida * para cerrar automáticamente la instancia de notebook una vez que se completen los procesos de entrenamiento e implementación. Esta es una configuración necesaria, ya que el coste es uno de los principales factores que hay que tener en cuenta

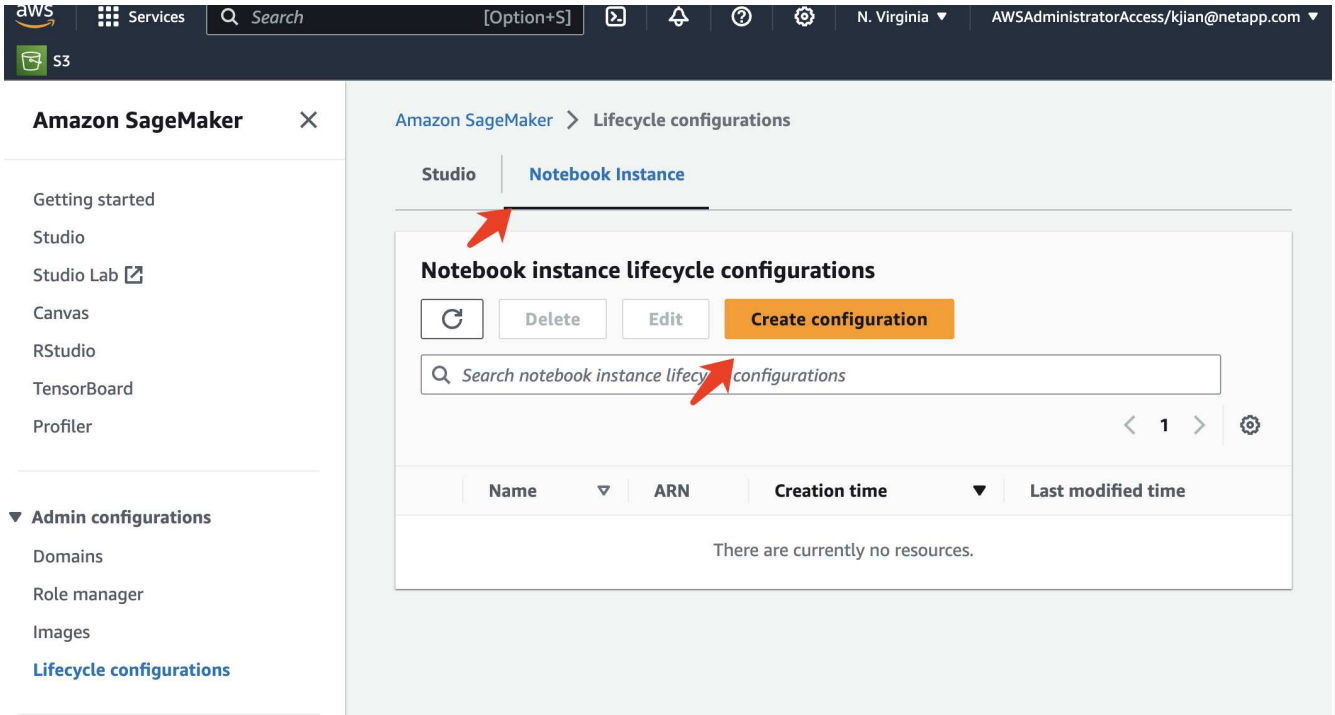
en MLOps.

Es importante tener en cuenta que la configuración de **configuraciones de ciclo de vida** debe configurarse con antelación. Por lo tanto, se recomienda priorizar la configuración de este aspecto antes de continuar con la otra configuración de pipeline de MLOps.

1. Para configurar una configuración de ciclo de vida, abra el panel **Sagemaker** y vaya a **Configuraciones de ciclo de vida** en la sección **Configuraciones de administración**.



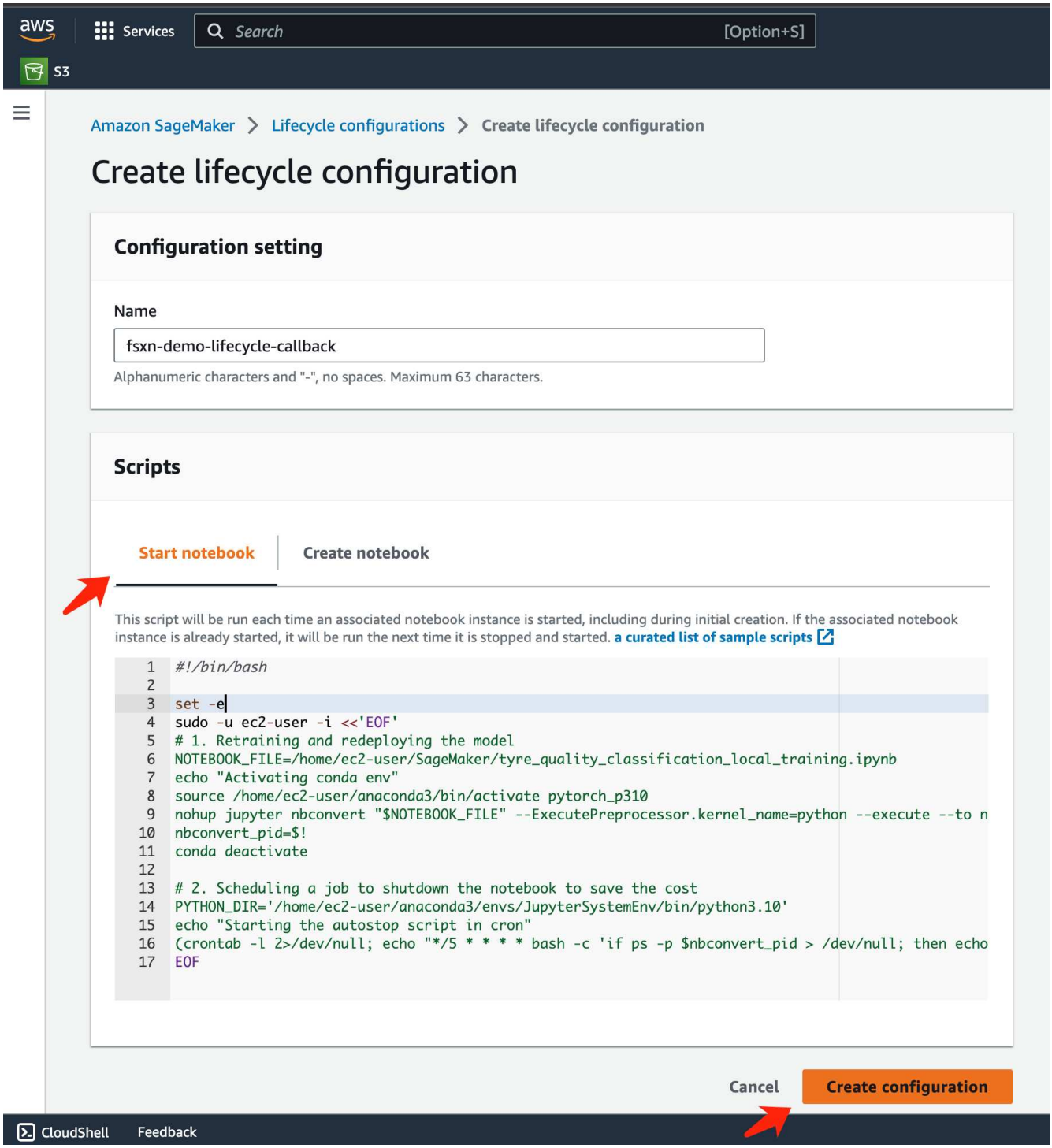
2. Seleccione la pestaña **Instancia de bloc de notas** y haga clic en el botón **Crear configuración**



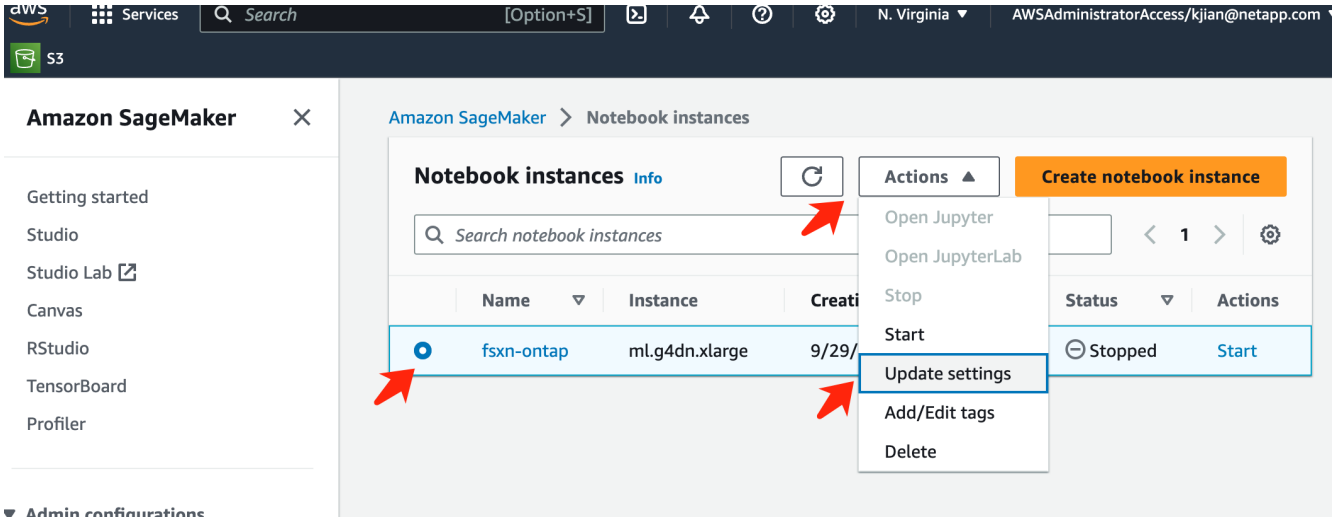
3. Pegue el siguiente código en el área de entrada.

```
#!/bin/bash

set -e
sudo -u ec2-user -i <<'EOF'
# 1. Retraining and redeploying the model
NOTEBOOK_FILE=/home/ec2-
user/SageMaker/tyre_quality_classification_local_training.ipynb
echo "Activating conda env"
source /home/ec2-user/anaconda3/bin/activate pytorch_p310
nohup jupyter nbconvert "$NOTEBOOK_FILE"
--ExecutePreprocessor.kernel_name=python --execute --to notebook &
nbconvert_pid=$!
conda deactivate

# 2. Scheduling a job to shutdown the notebook to save the cost
PYTHON_DIR='/home/ec2-
user/anaconda3/envs/JupyterSystemEnv/bin/python3.10'
echo "Starting the autostop script in cron"
(crontab -l 2>/dev/null; echo "*/5 * * * * bash -c 'if ps -p
$nbconvert_pid > /dev/null; then echo \"Notebook is still running.\" >>
/var/log/jupyter.log; else echo \"Notebook execution completed.\" >>
/var/log/jupyter.log; $PYTHON_DIR -c \"import boto3;boto3.client(
\'sagemaker\').stop_notebook_instance(NotebookInstanceName=get_notebook_
name())\" >> /var/log/jupyter.log; fi'") | crontab -
EOF
```

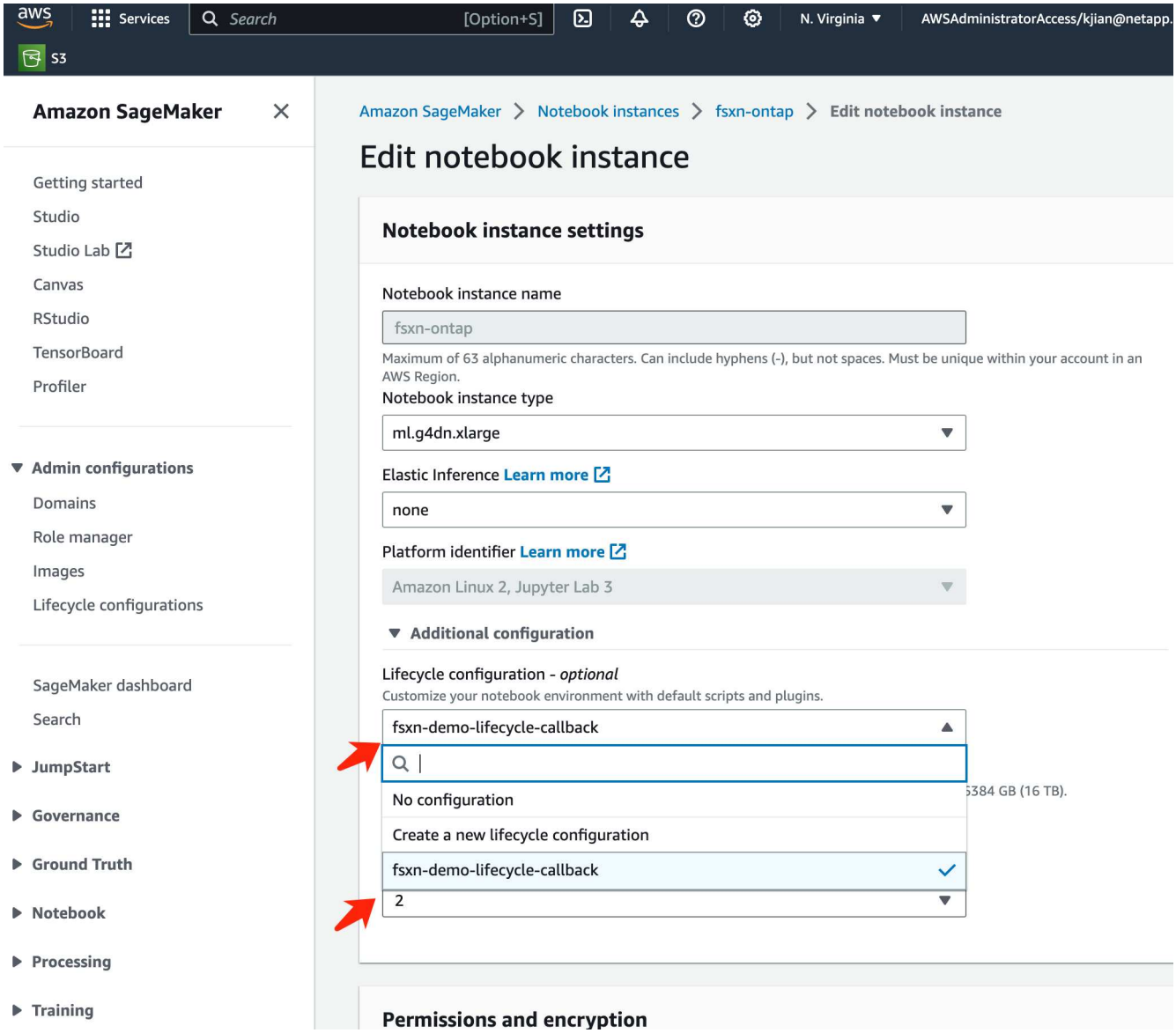
4. Este script ejecuta el Jupyter Notebook, que se encarga del reciclaje y el redespliegue del modelo para la inferencia. Una vez finalizada la ejecución, el bloc de notas se apagará automáticamente en 5 minutos. Para obtener más información sobre la declaración del problema y la implementación del código, consulte "Parte 2: Aprovechamiento de AWS FSx para NetApp ONTAP (FSxN) como fuente de datos para el entrenamiento de modelos en SageMaker".



5. Después de la creación, navegue a Instancias de bloc de notas, seleccione la instancia de destino y haga clic en **Actualizar configuración** en el menú desplegable Acciones.



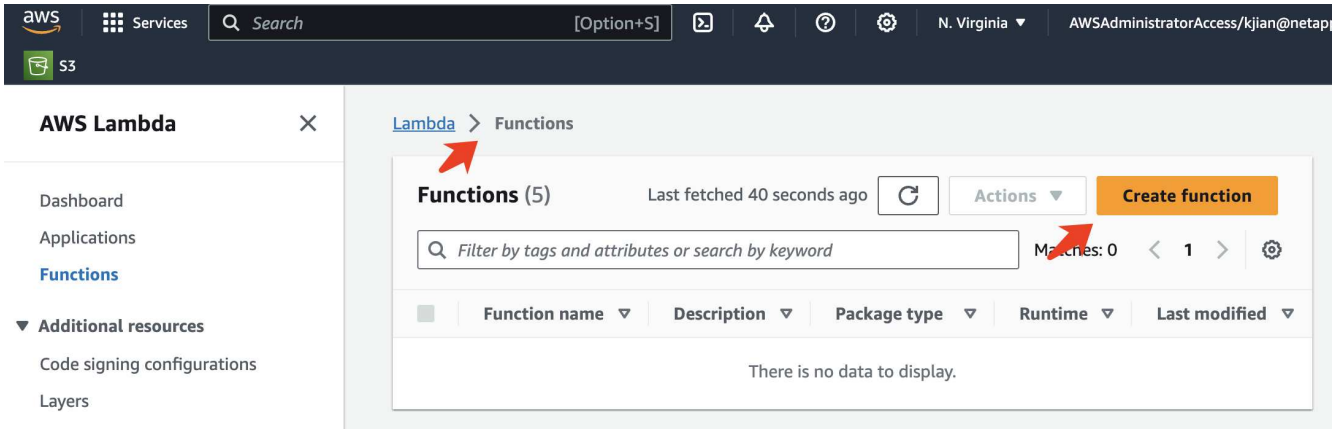
6. Seleccione la **Configuración de ciclo de vida** creada y haga clic en **Actualizar instancia de bloc de notas**.



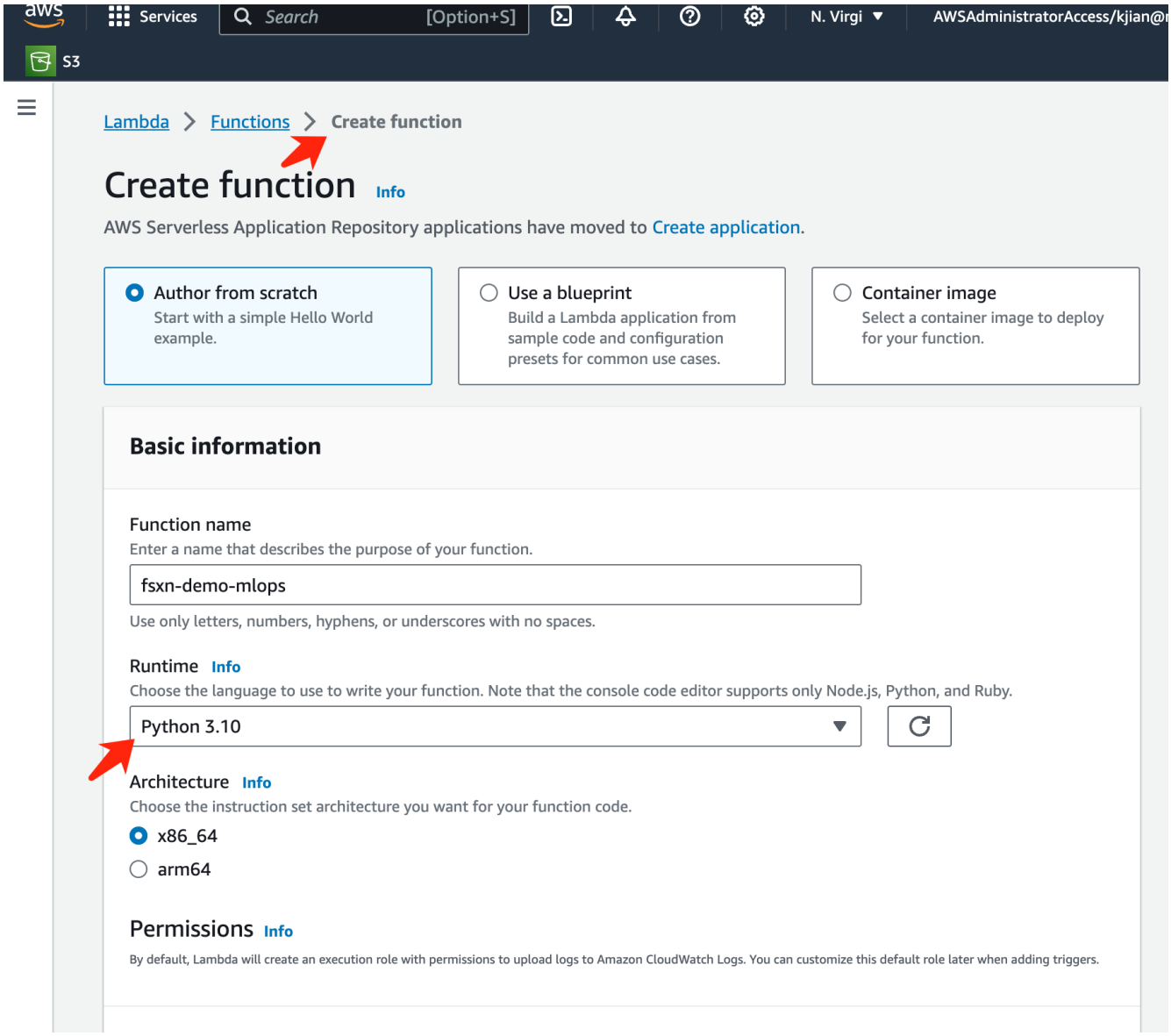
Función sin servidor de AWS Lambda

Como se mencionó anteriormente, la función **AWS Lambda** es responsable de poner en funcionamiento la instancia **AWS SageMaker Notebook**.

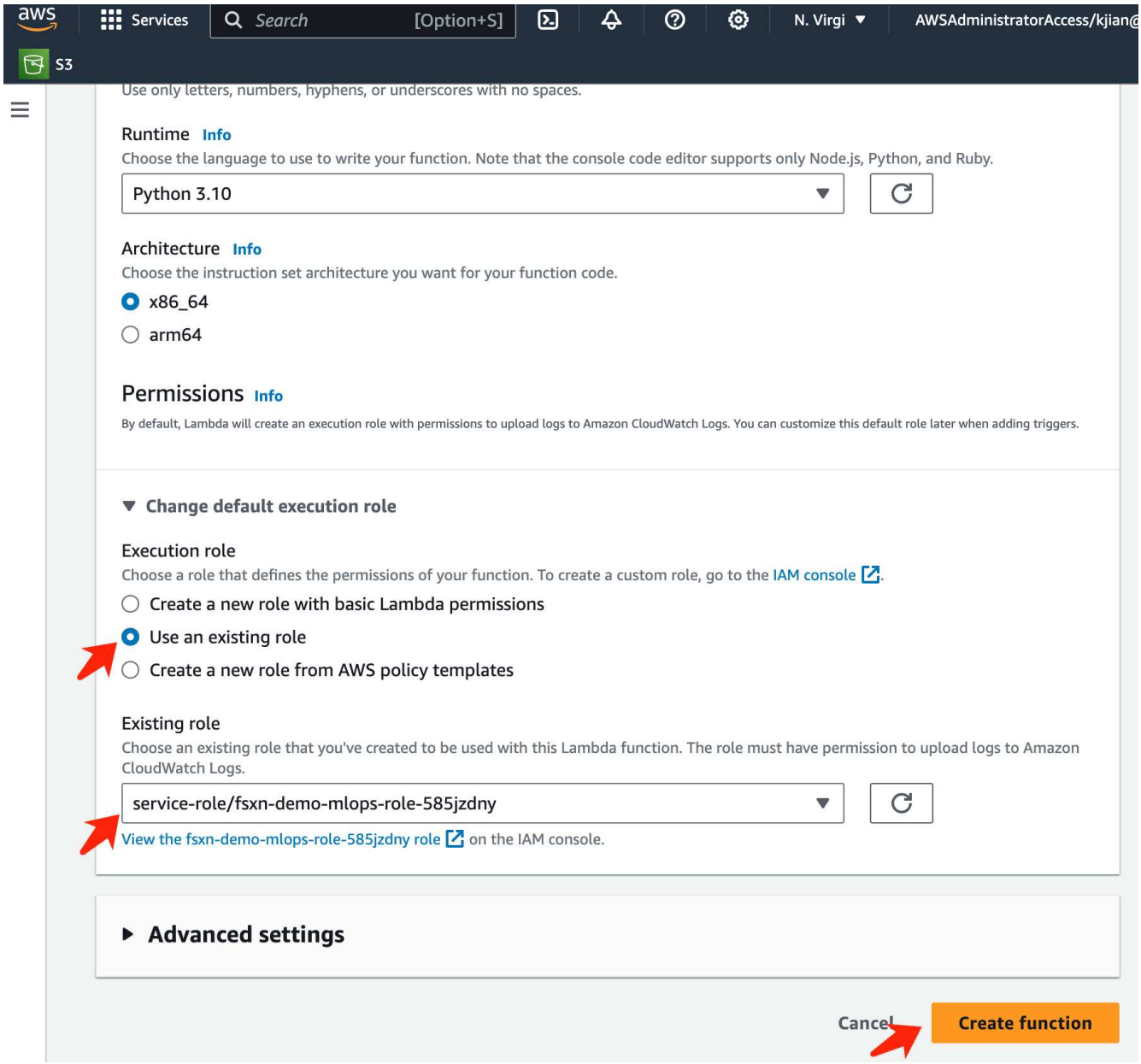
1. Para crear una función **AWS Lambda**, navegue hasta el panel correspondiente, cambie a la pestaña **Funciones** y haga clic en **Crear función**.



2. Por favor, archiva todas las entradas requeridas en la página y recuerda cambiar el tiempo de ejecución a **Python 3,10**.



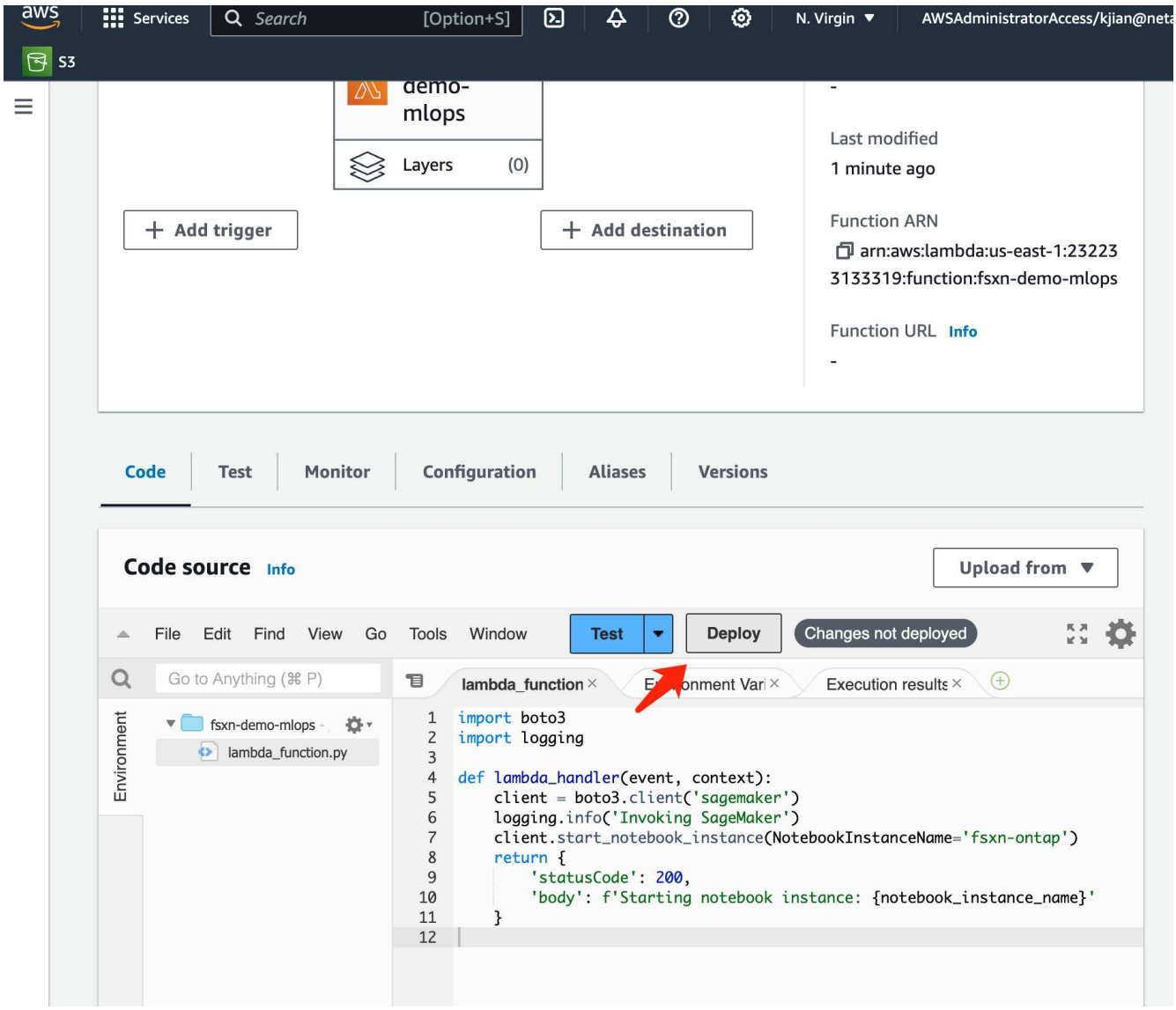
3. Verifique que el rol designado tiene el permiso requerido **AmazonSageMakerFullAccess** y haga clic en el botón **Crear función**.



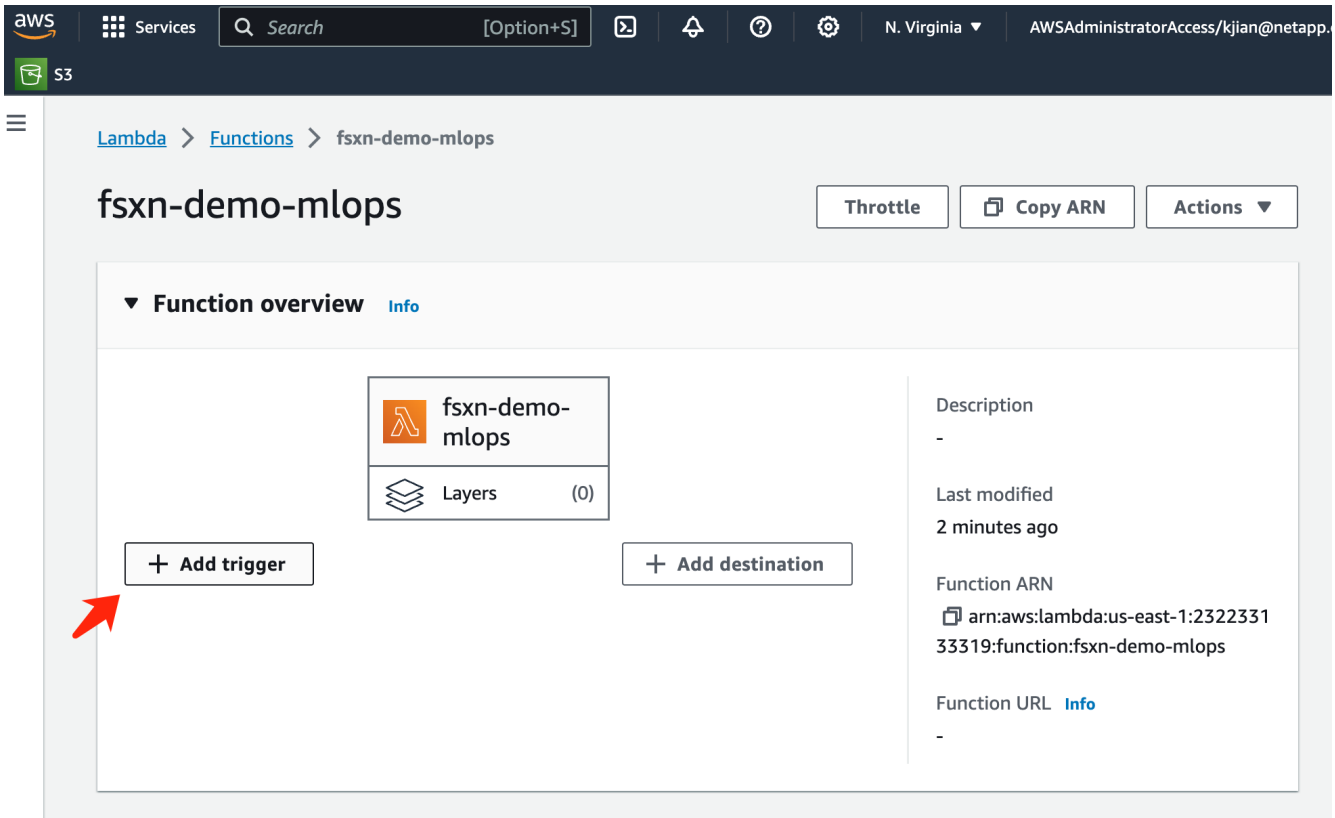
4. Seleccione la función Lambda creada. En la pestaña de código, copie y pegue el siguiente código en el área de texto. Este código inicia la instancia de notebook llamada **fsxn-ontap**.

```
import boto3
import logging

def lambda_handler(event, context):
    client = boto3.client('sagemaker')
    logging.info('Invoking SageMaker')
    client.start_notebook_instance(NotebookInstanceName='fsxn-ontap')
    return {
        'statusCode': 200,
        'body': f'Starting notebook instance: {notebook_instance_name}'
    }
```

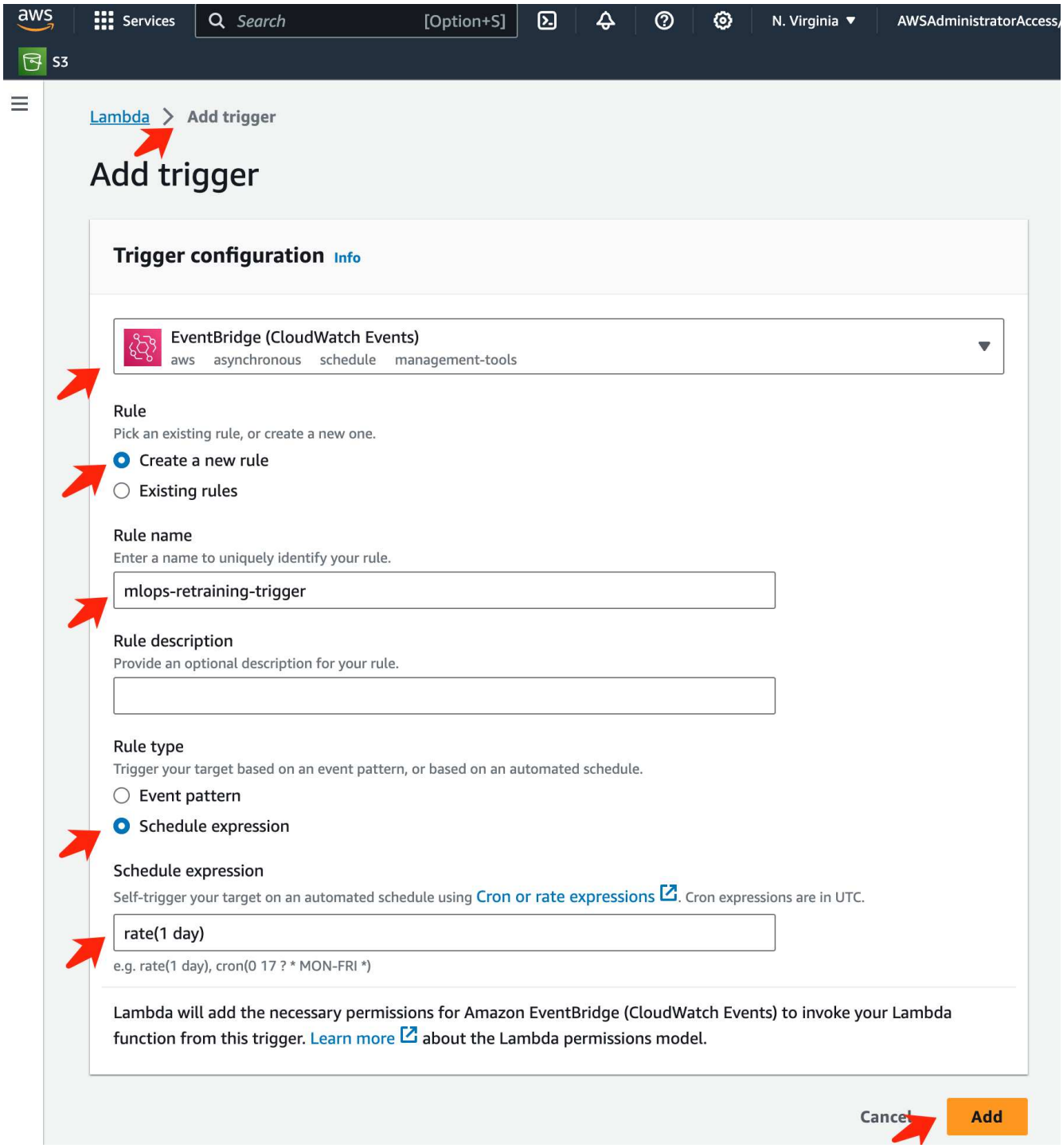
5. Haga clic en el botón **Desplegar** para aplicar este cambio de código.



6. Para especificar cómo activar esta función de AWS Lambda, haga clic en el botón Agregar Disparador.



7. Seleccione EventBridge en el menú desplegable y, a continuación, haga clic en el botón de opción con la etiqueta Crear una nueva regla. En el campo de expresión de programación, introduzca `rate(1 day)`, Y haga clic en el botón Agregar para crear y aplicar esta nueva regla de trabajo cron a la función AWS Lambda.



Después de completar la configuración en dos pasos, diariamente, la función **AWS Lambda** iniciará el **SageMaker Notebook**, realizará el reciclaje del modelo utilizando los datos del repositorio **FSxN**, volverá a desplegar el modelo actualizado en el entorno de producción y cerrará automáticamente la instancia **SageMaker Notebook** para optimizar los costos. Esto garantiza que el modelo permanezca actualizado.

Esto concluye el tutorial para desarrollar un pipeline de MLOps.