# FSx ONTAP para MLOps

## NetApp artificial intelligence solutions

NetApp
August 18, 2025

This PDF was generated from https://docs.netapp.com/es-es/netapp-solutions-ai/cloud/ai-mlops-fsxn-intro.html on August 18, 2025. Always check docs.netapp.com for the latest.

# Tabla de contenidos


FSx ONTAP para MLOps . . . 1
- Amazon FSx for NetApp ONTAP (FSx ONTAP) para MLOps . . . 1
- Parte 1: Integración de Amazon FSx for NetApp ONTAP (FSx ONTAP) como un bucket S3 privado en AWS SageMaker . . . 1
  - Introducción . . . 1
  - Guía del usuario . . . 1
  - Lista de verificación de depuración útil . . . 14
  - Preguntas frecuentes (al 27 de septiembre de 2023) . . . 15
- Parte 2: Aprovechamiento de AWS Amazon FSx for NetApp ONTAP (FSx ONTAP) como fuente de datos para el entrenamiento de modelos en SageMaker . . . 15
  - Introducción . . . 15
  - ¿Qué es FSx ONTAP? . . . 15
  - Requisito previo . . . 16
  - Descripción general de la integración . . . 16
  - Integración paso a paso . . . 17
- Parte 3: Creación de una canalización de MLOps simplificada (CI/CT/CD) . . . 24
  - Introducción . . . 24
  - Manifiesto . . . 24
  - Requisito previo . . . 25
  - Arquitectura . . . 25
  - Configuración paso a paso . . . 25

# FSx ONTAP para MLOps

## Amazon FSx for NetApp ONTAP (FSx ONTAP) para MLOps

Esta sección profundiza en la aplicación práctica del desarrollo de infraestructura de IA y proporciona un tutorial de principio a fin sobre la construcción de una canalización MLOps utilizando FSx ONTAP. Compuesto por tres ejemplos completos, lo guía para satisfacer sus necesidades de MLOps a través de esta poderosa plataforma de gestión de datos.

Estos artículos se centran en:

1. "Parte 1: Integración de Amazon FSx for NetApp ONTAP (FSx ONTAP) como un bucket S3 privado en AWS SageMaker"
2. "Parte 2: Aprovechamiento de Amazon FSx for NetApp ONTAP (FSx ONTAP) como fuente de datos para el entrenamiento de modelos en SageMaker"
3. "Parte 3: Creación de una canalización de MLOps simplificada (CI/CT/CD)"

Al final de esta sección, habrá adquirido una comprensión sólida de cómo utilizar FSx ONTAP para optimizar los procesos de MLOps.

## Parte 1: Integración de Amazon FSx for NetApp ONTAP (FSx ONTAP) como un bucket S3 privado en AWS SageMaker

Esta sección proporciona una guía sobre cómo configurar FSx ONTAP como un depósito S3 privado mediante AWS SageMaker.

### Introducción

Usando SageMaker como ejemplo, esta página proporciona orientación sobre cómo configurar FSx ONTAP como un depósito S3 privado.

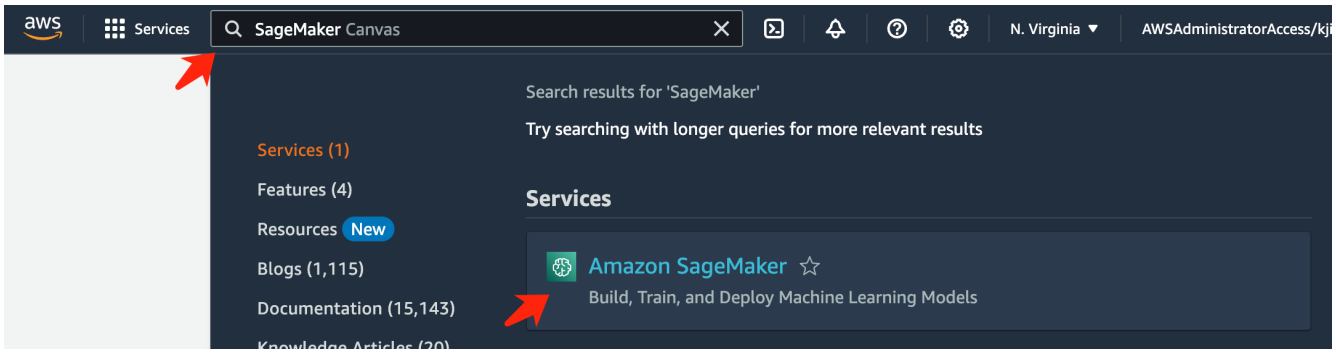
Para obtener más información sobre FSx ONTAP, consulte esta presentación ("Enlace de vídeo" )

### Guía del usuario

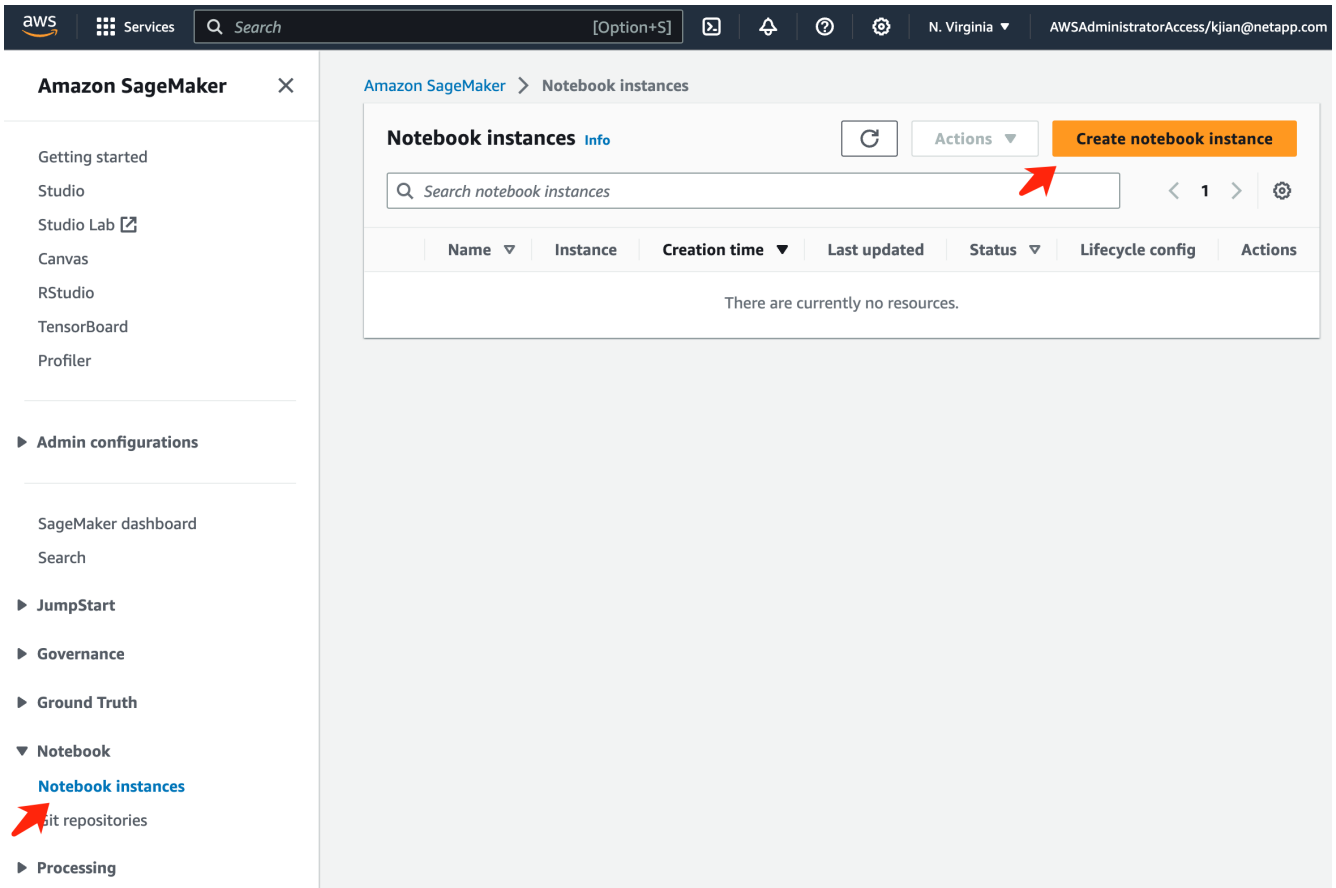
#### Creación de servidores

##### Crear una instancia de SageMaker Notebook

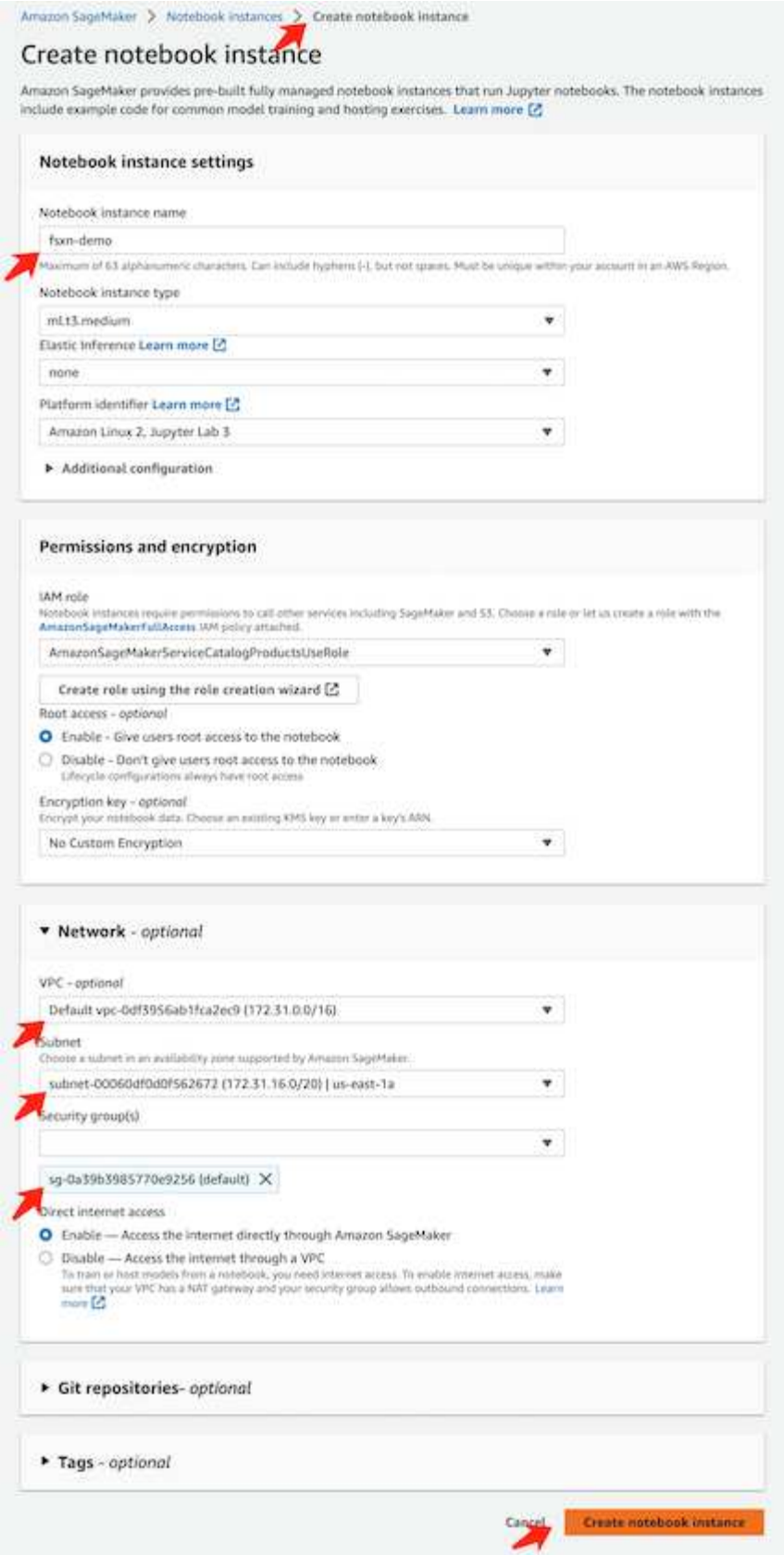
1. Abra la consola de AWS. En el panel de búsqueda, busque SageMaker y haga clic en el servicio **Amazon SageMaker**.




2. Abra las **Instancias de Notebook** en la pestaña Notebook, haga clic en el botón naranja **Crear instancia de notebook**.

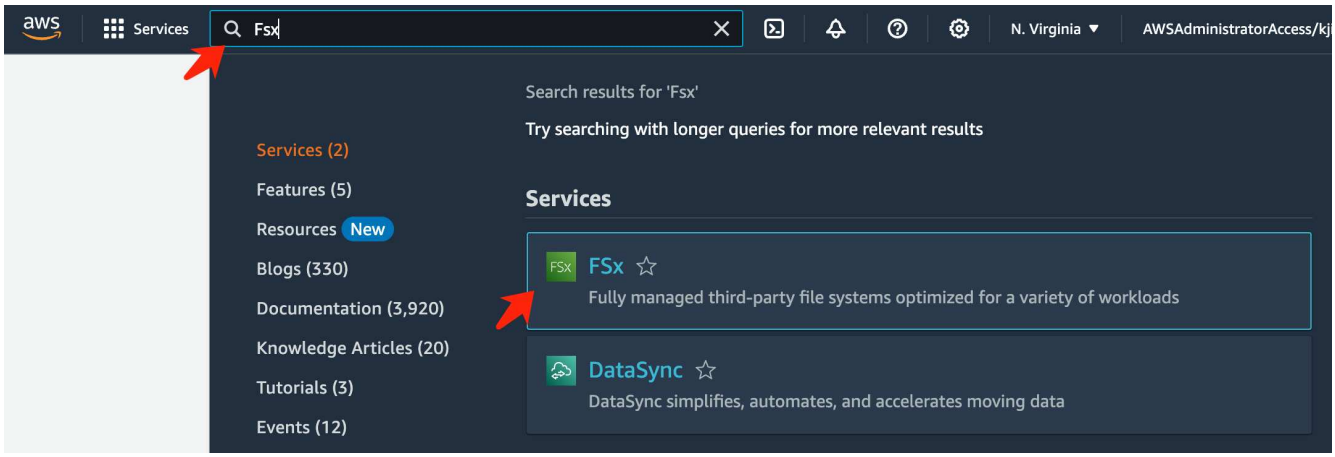



3. En la página de creación, ingrese el **nombre de la instancia de Notebook**, expanda el panel **Red**, deje las demás entradas predeterminadas y seleccione una **VPC**, una **Subred** y **Grupos de seguridad**. (Esta **VPC** y **Subred** se usarán para crear el sistema de archivos FSx ONTAP más adelante) Haga clic en el botón naranja **Crear instancia de notebook** en la parte inferior derecha.

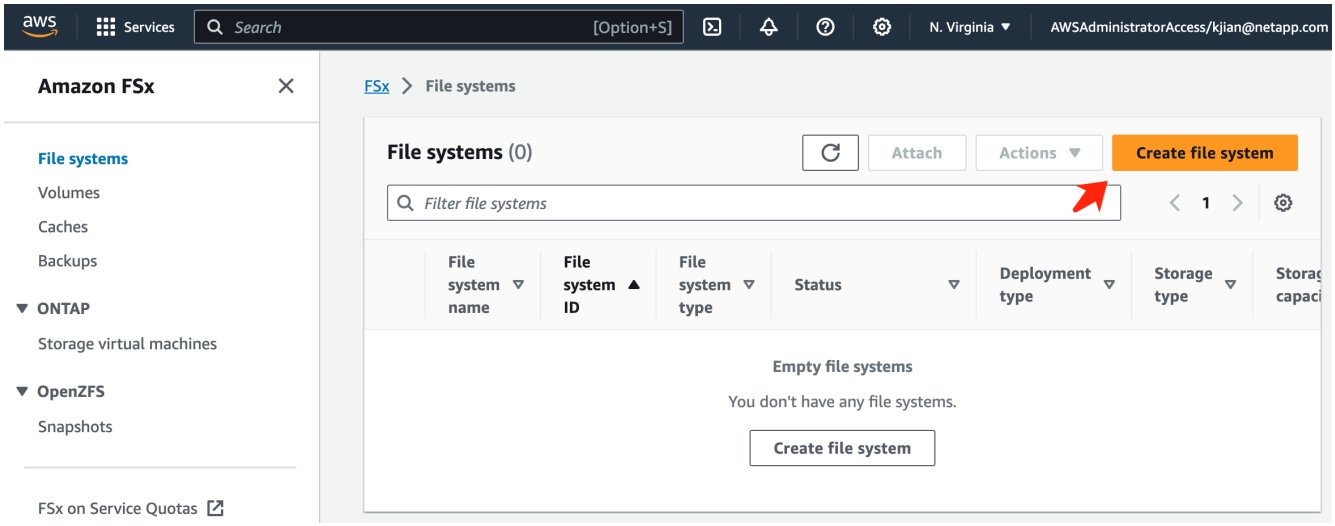



**Crear un sistema de archivos FSx ONTAP**

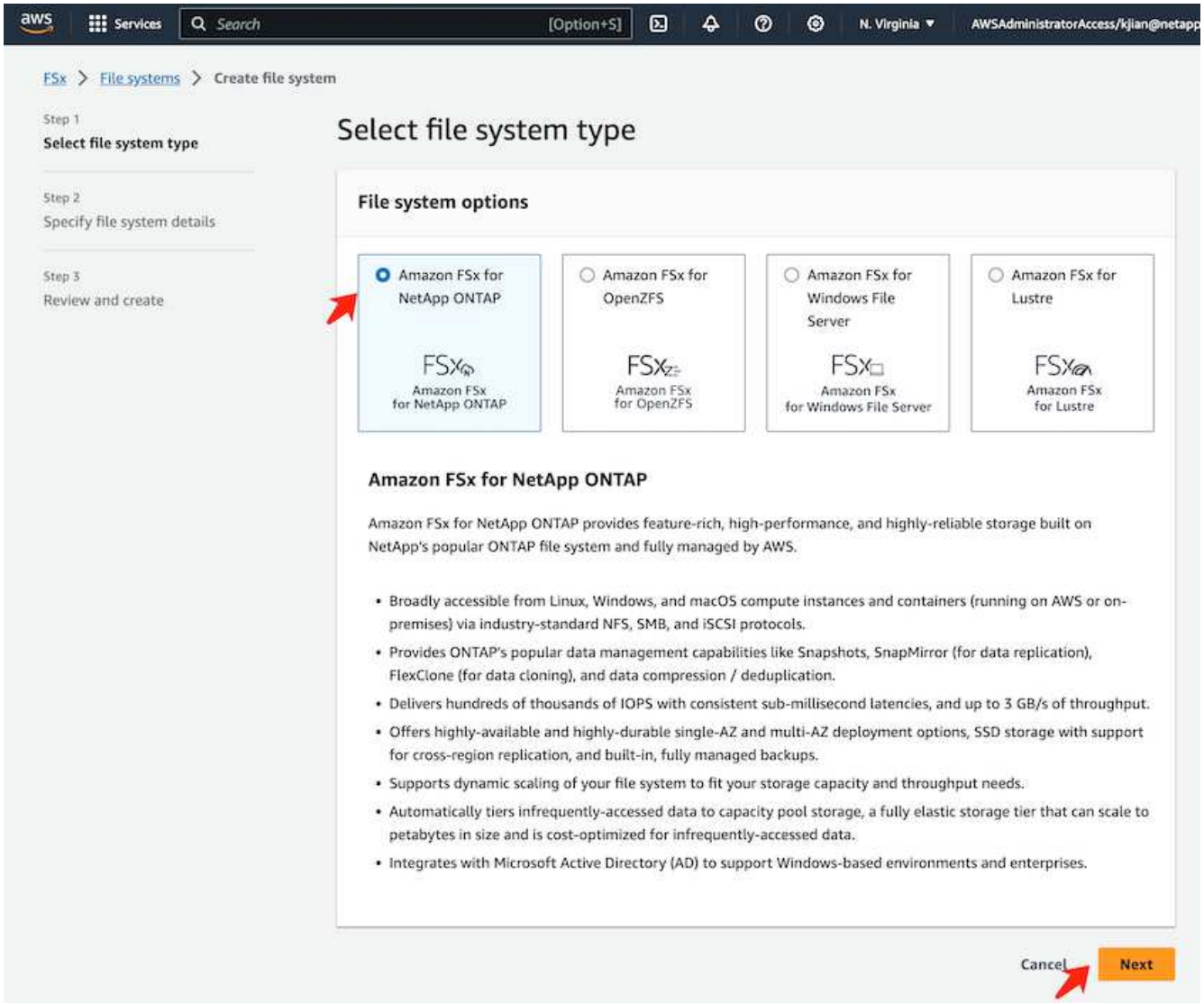
1. Abra la consola de AWS. En el panel de búsqueda, busque Fsx y haga clic en el servicio **FSx**.



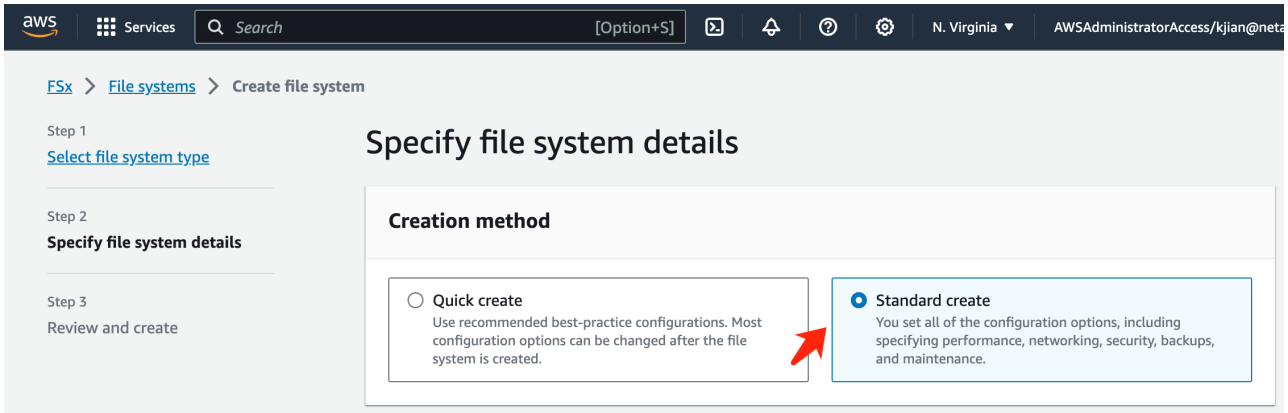

2. Haga clic en **Crear sistema de archivos**.



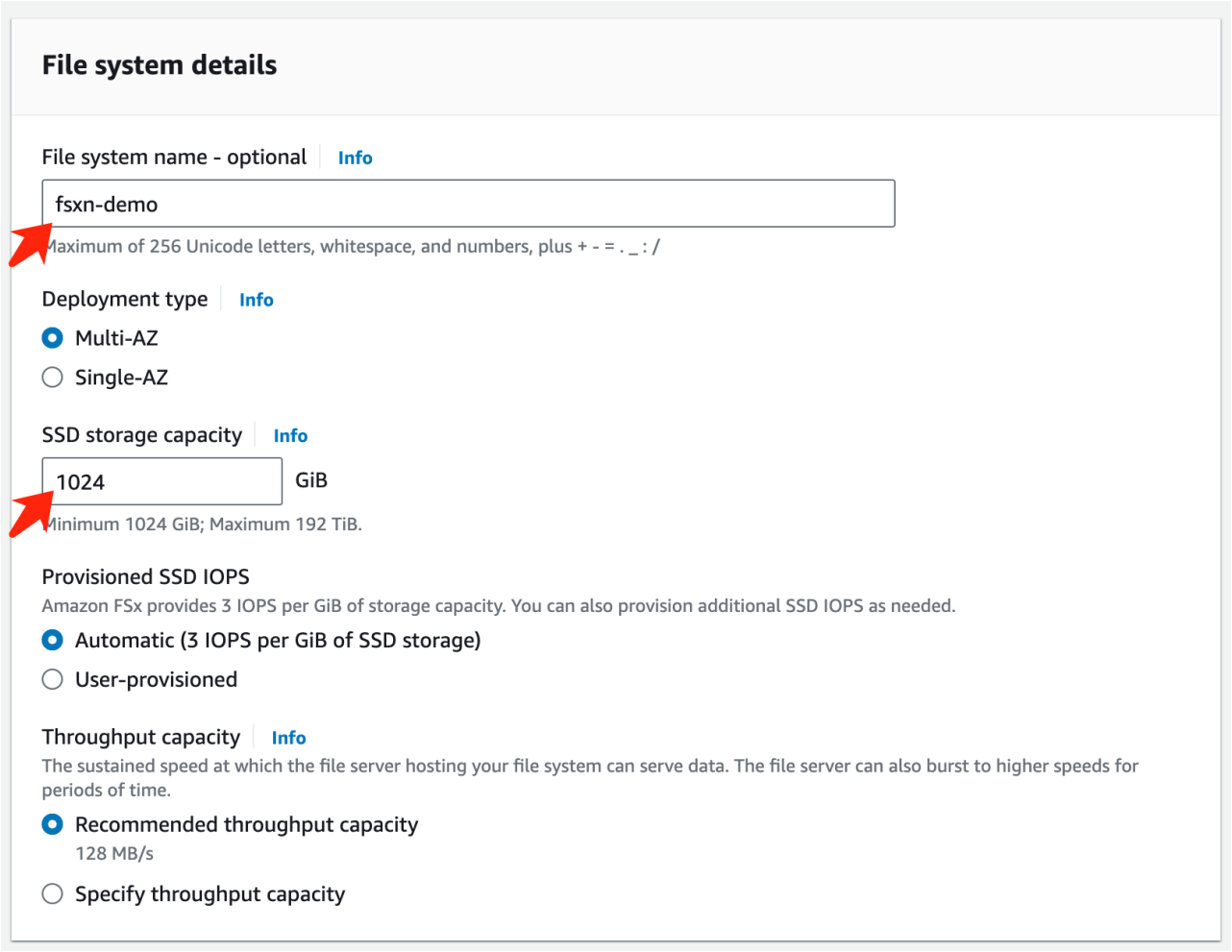

3. Seleccione la primera tarjeta **FSx ONTAP** y haga clic en **Siguiente**.



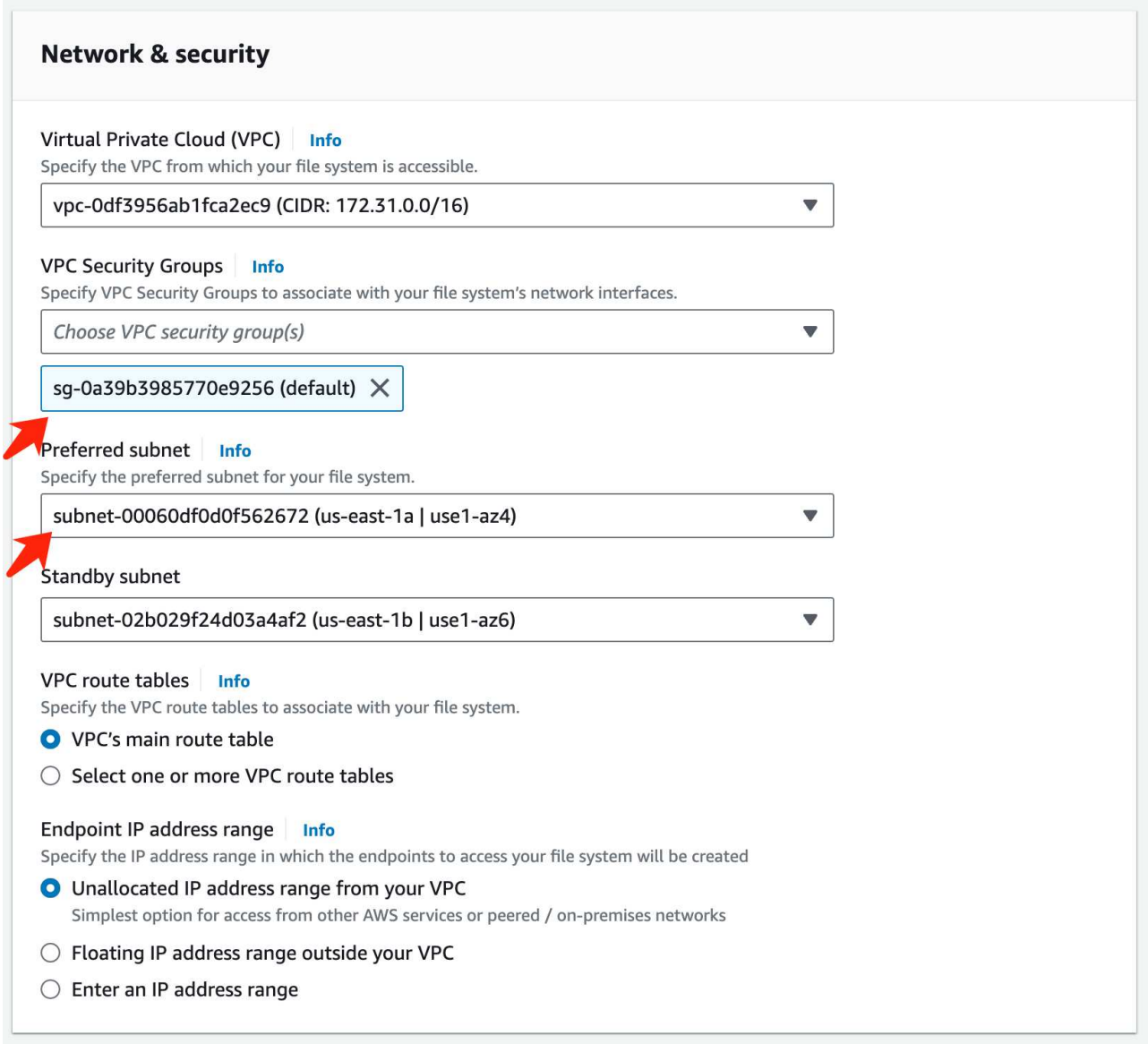

4. En la página de configuración de detalles.
   a. Seleccione la opción **Creación estándar**.



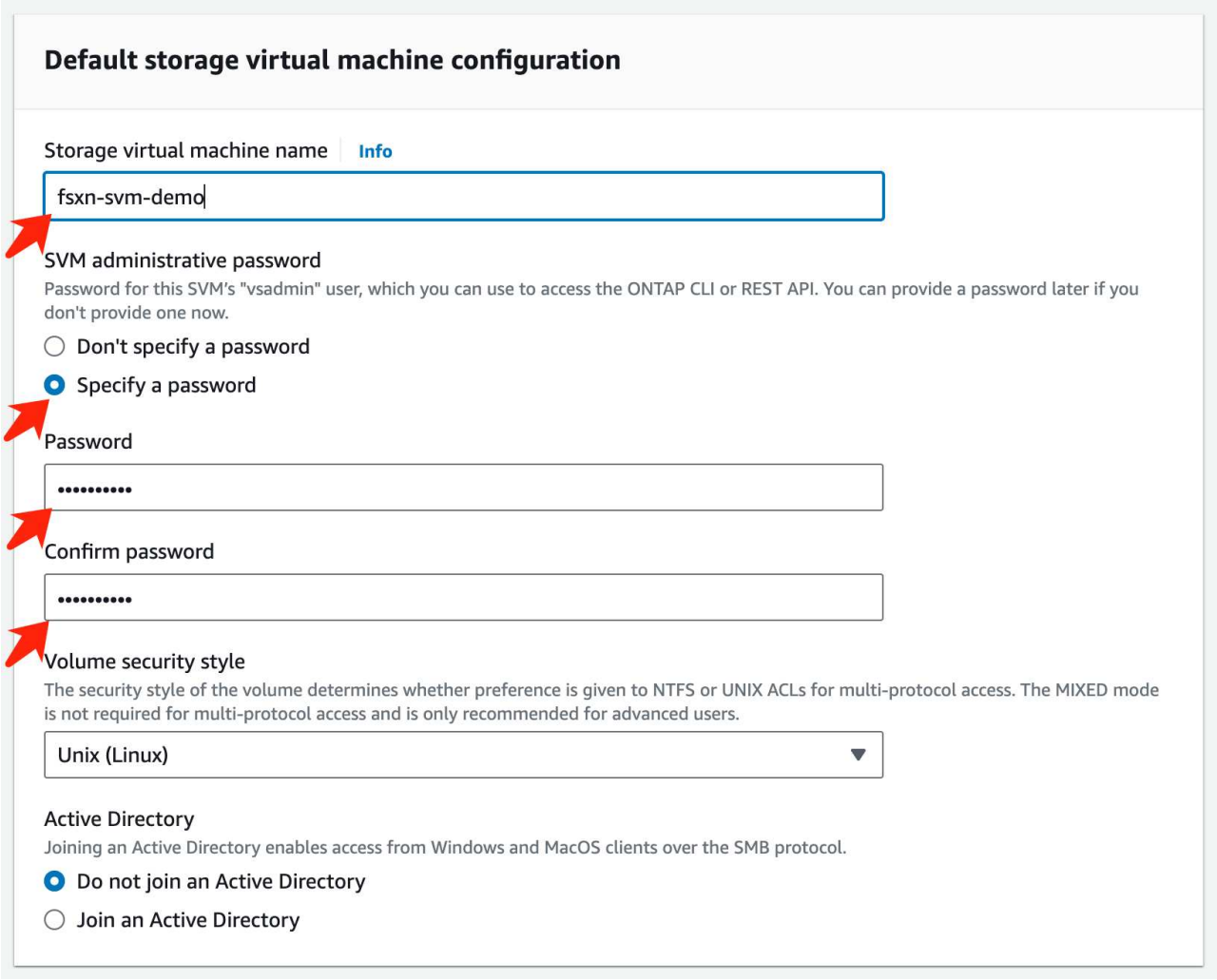

   b. Introduzca el **Nombre del sistema de archivos** y la **Capacidad de almacenamiento SSD**.



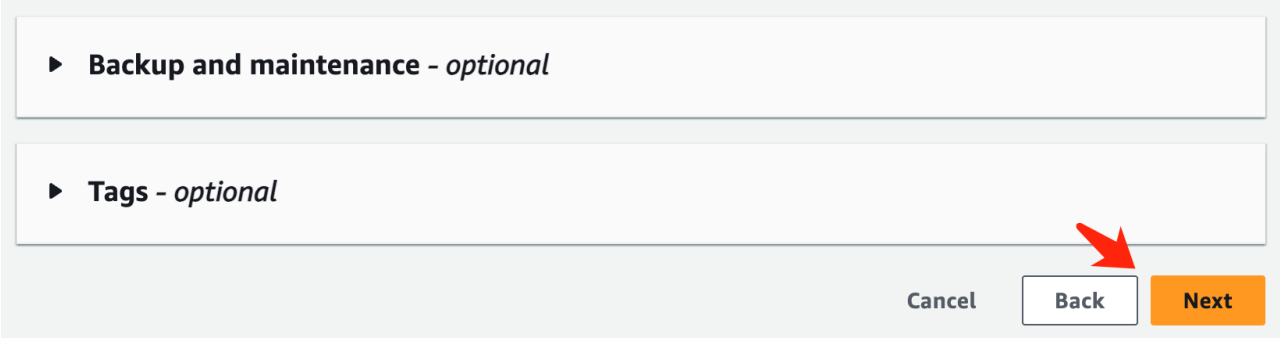

c. Asegúrese de utilizar la **VPC** y la **subred** mismas que para la instancia de **SageMaker Notebook**.



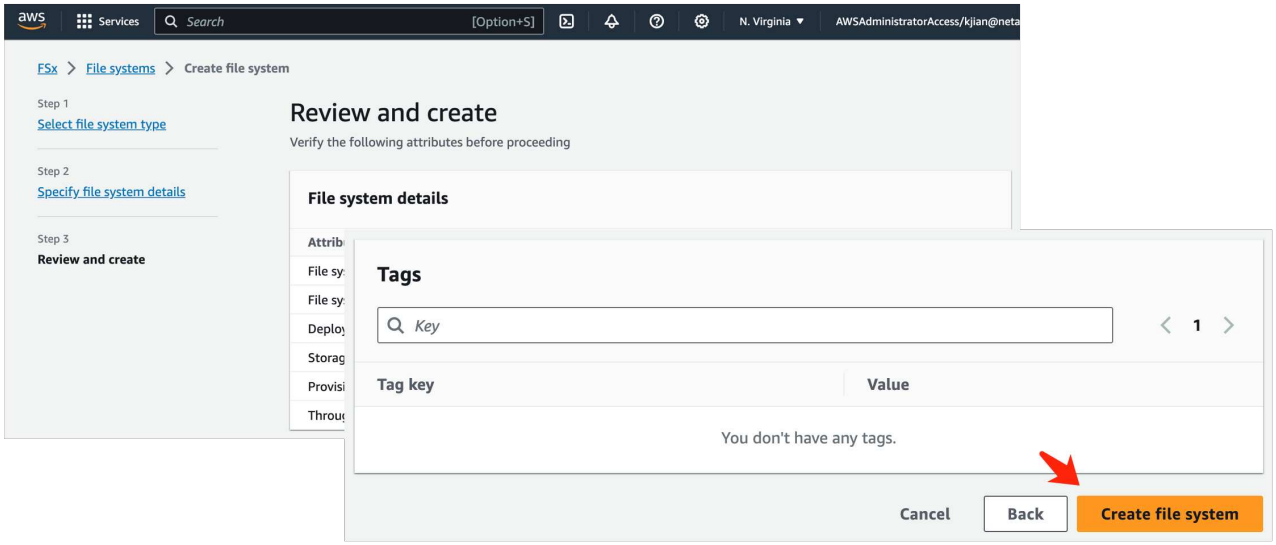

d. Ingrese el nombre de la **máquina virtual de almacenamiento** y **especifique una contraseña** para su SVM (máquina virtual de almacenamiento).



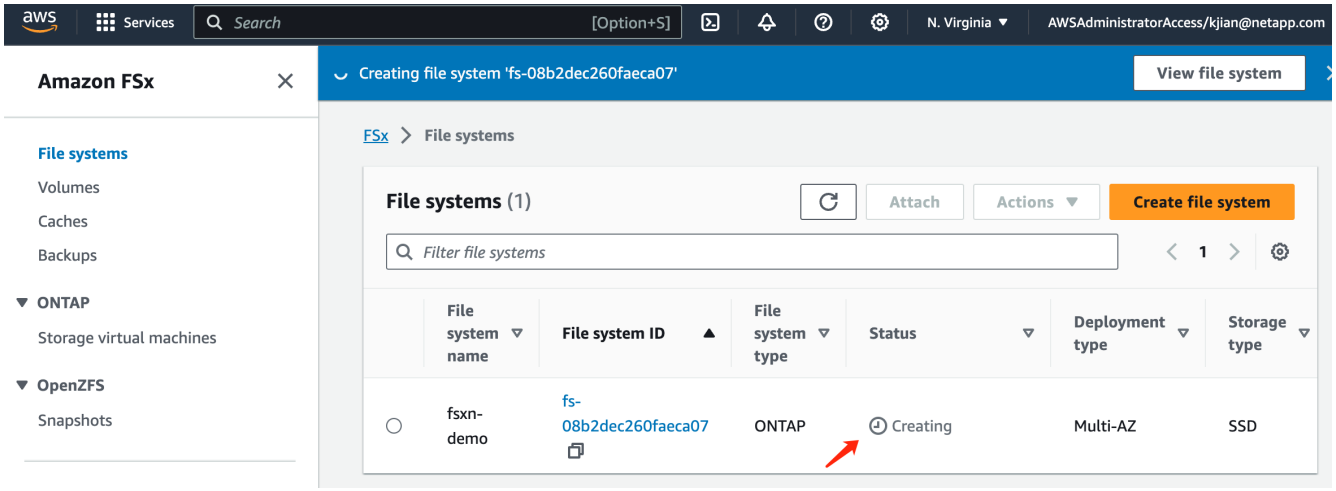

e. Deje las demás entradas predeterminadas y haga clic en el botón naranja **Siguiente** en la parte inferior derecha.




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



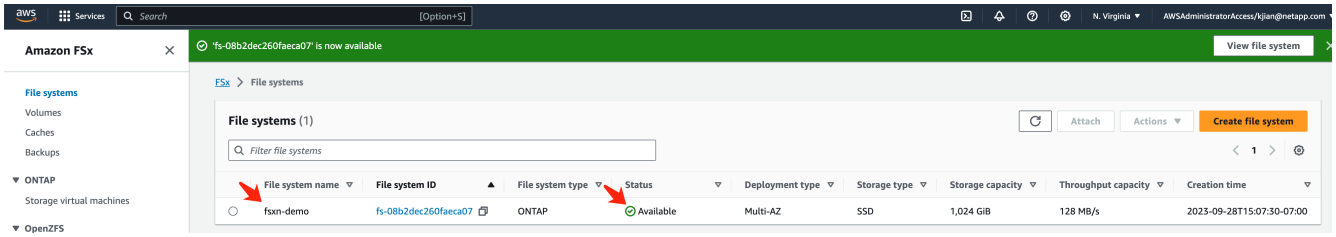

5. Puede tomar alrededor de **20 a 40 minutos** activar el sistema de archivos FSx.



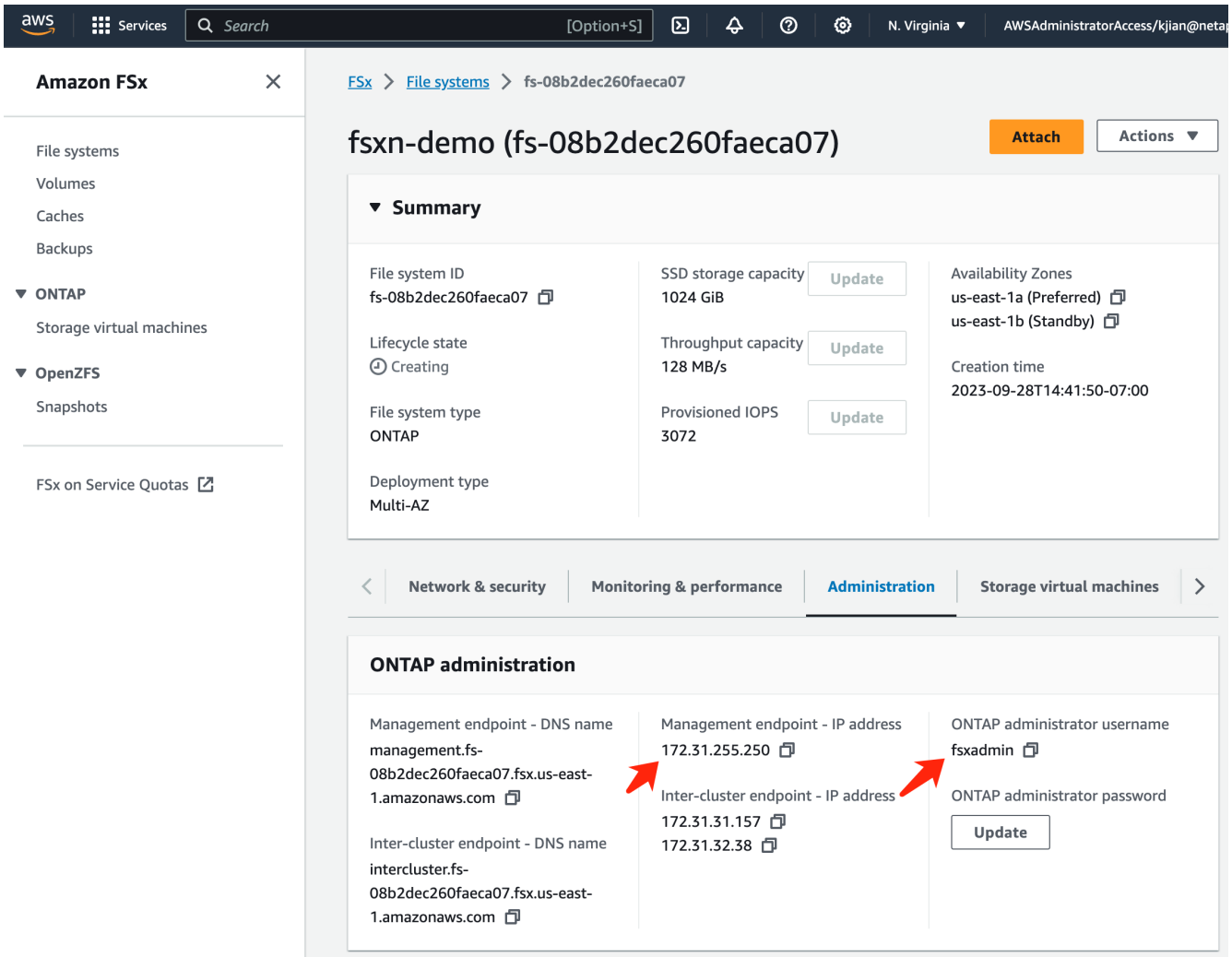

## Configuración del servidor

### Configuración de ONTAP

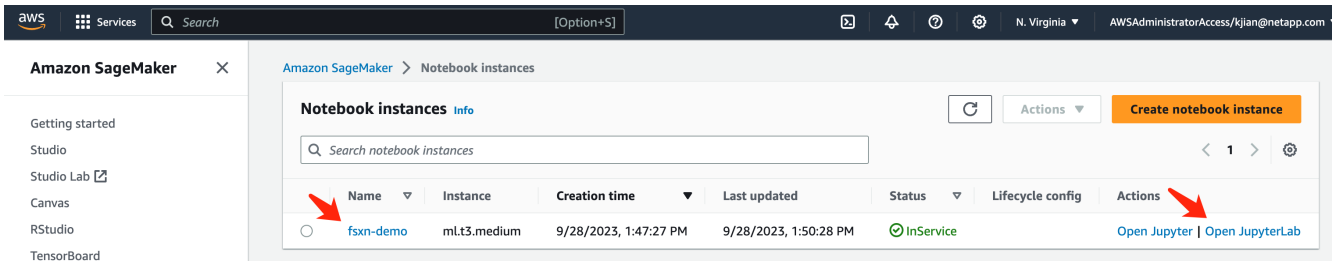
1. Abra el sistema de archivos FSx creado. Asegúrese de que el estado sea **Disponible**.



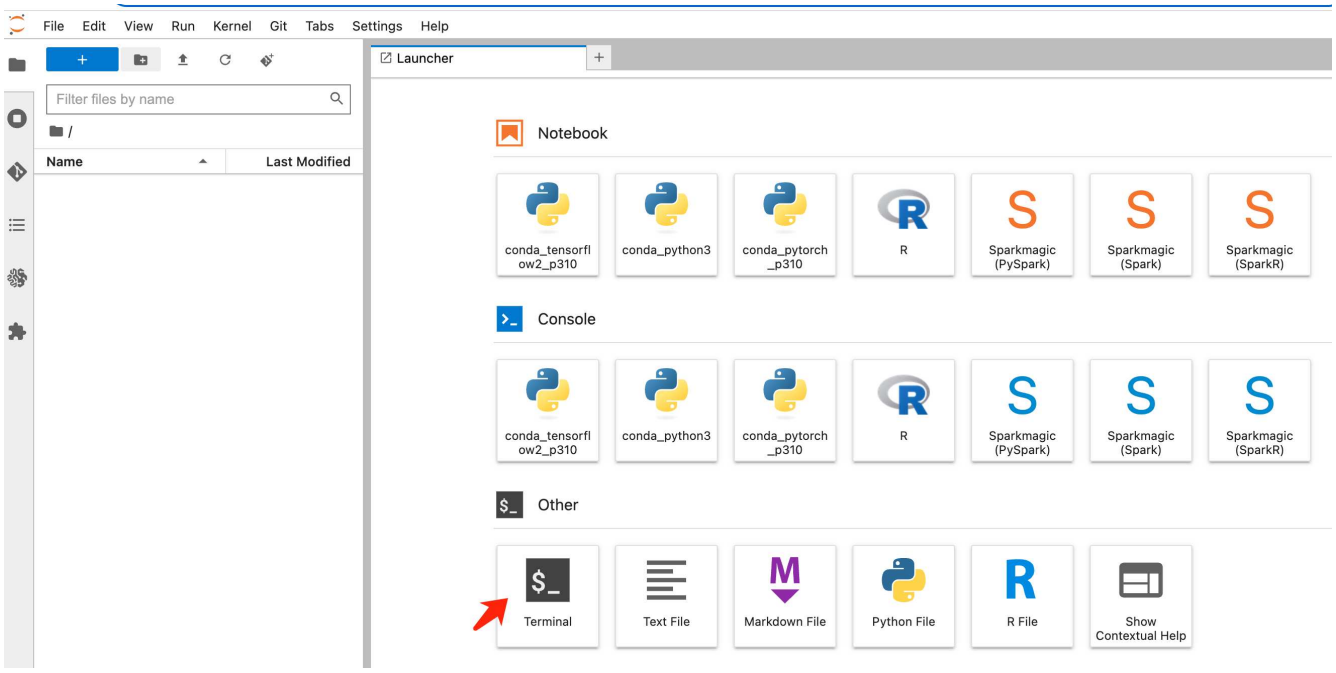

2. Seleccione la pestaña **Administración** y conserve la **Dirección IP del punto final de administración** y el *Nombre de usuario del administrador de ONTAP *.



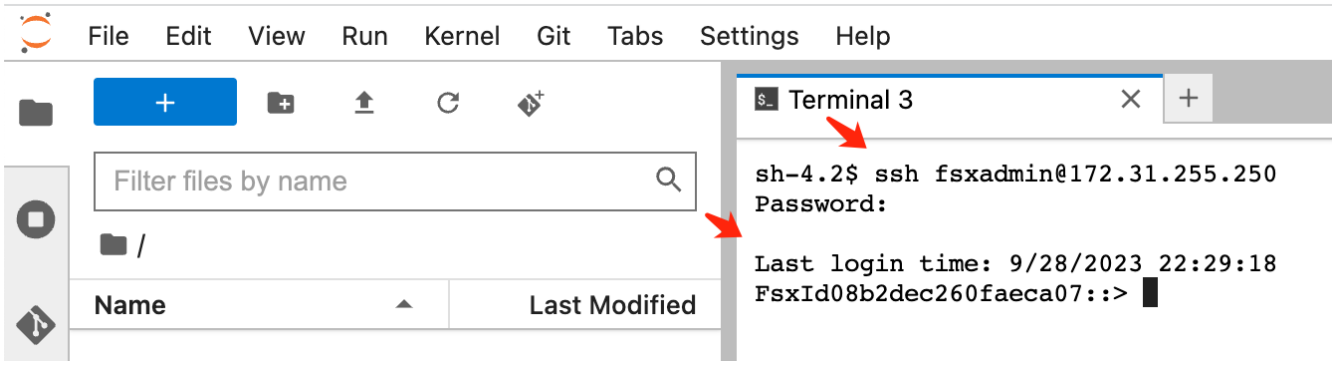

3. Abra la instancia **SageMaker Notebook** creada y haga clic en **Abrir JupyterLab**.




4. En la página de Jupyter Lab, abra una nueva **Terminal**.

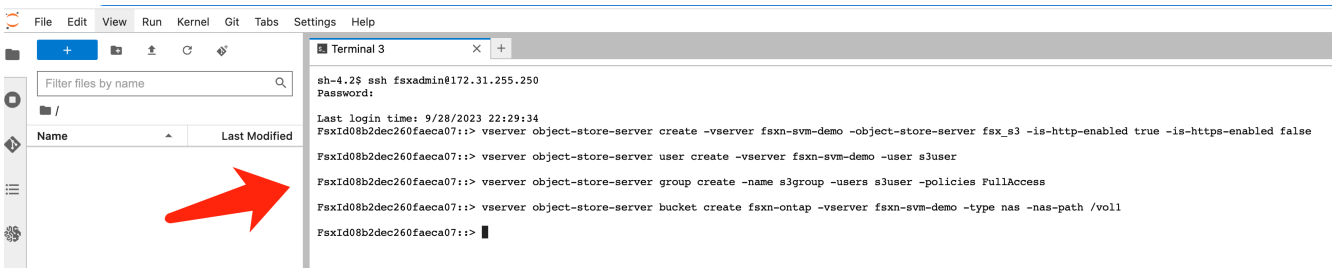



5. Ingrese el comando ssh ssh <nombre de usuario administrador>@<IP del servidor ONTAP > para iniciar sesión en el sistema de archivos FSx ONTAP . (El nombre de usuario y la dirección IP se recuperan del paso 2) Utilice la contraseña utilizada al crear la **máquina virtual de almacenamiento**.




6. Ejecute los comandos en el siguiente orden. Usamos **fsxn-ontap** como nombre para el **nombre del depósito S3 privado de FSx ONTAP *. Utilice el *nombre de la máquina virtual de almacenamiento** para el argumento **-vserver**.

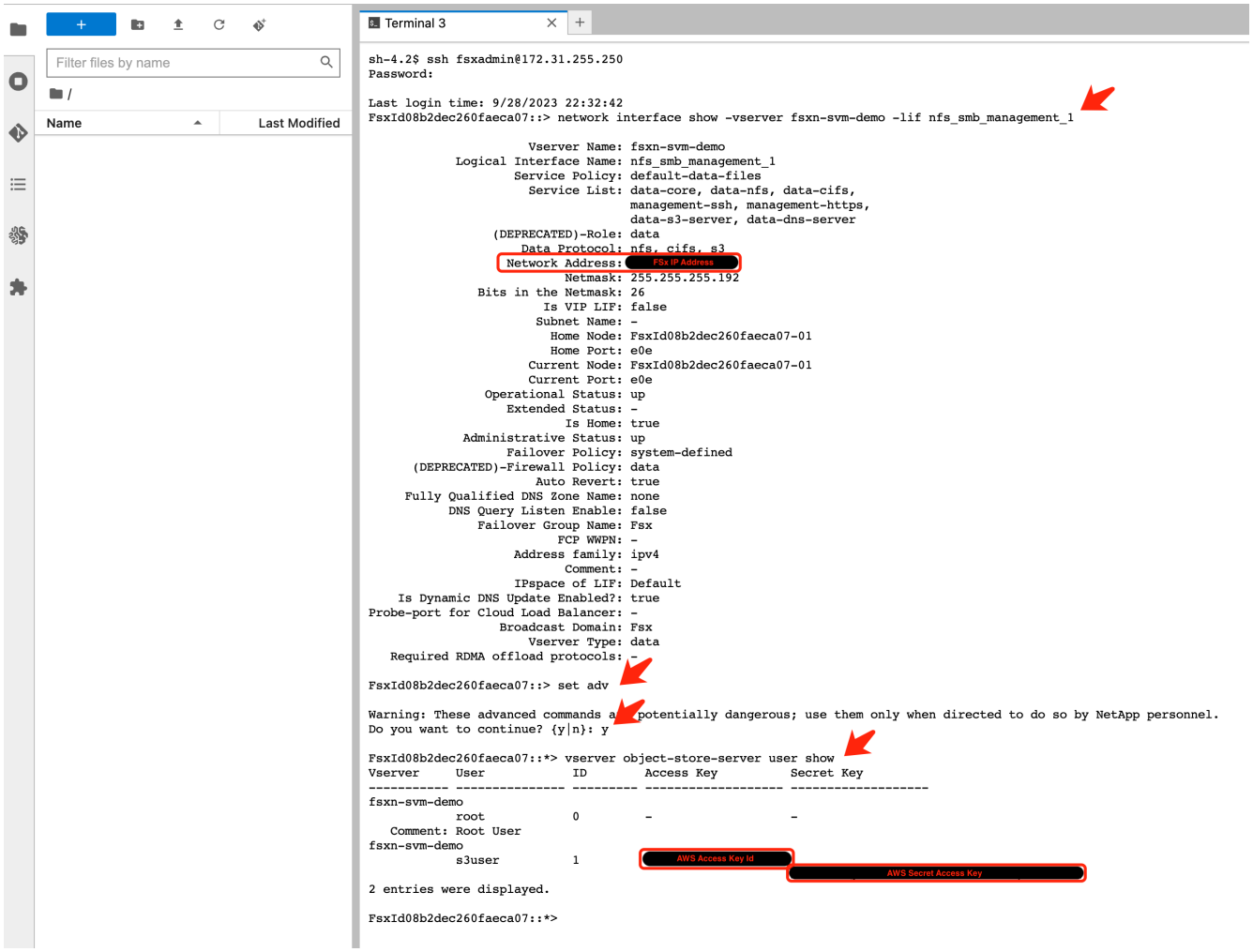
```
vserver object-store-server create -vserver fsxn-svm-demo -object-store
-server fsx_s3 -is-http-enabled true -is-https-enabled false

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 del punto final y las credenciales para FSx ONTAP Private S3.

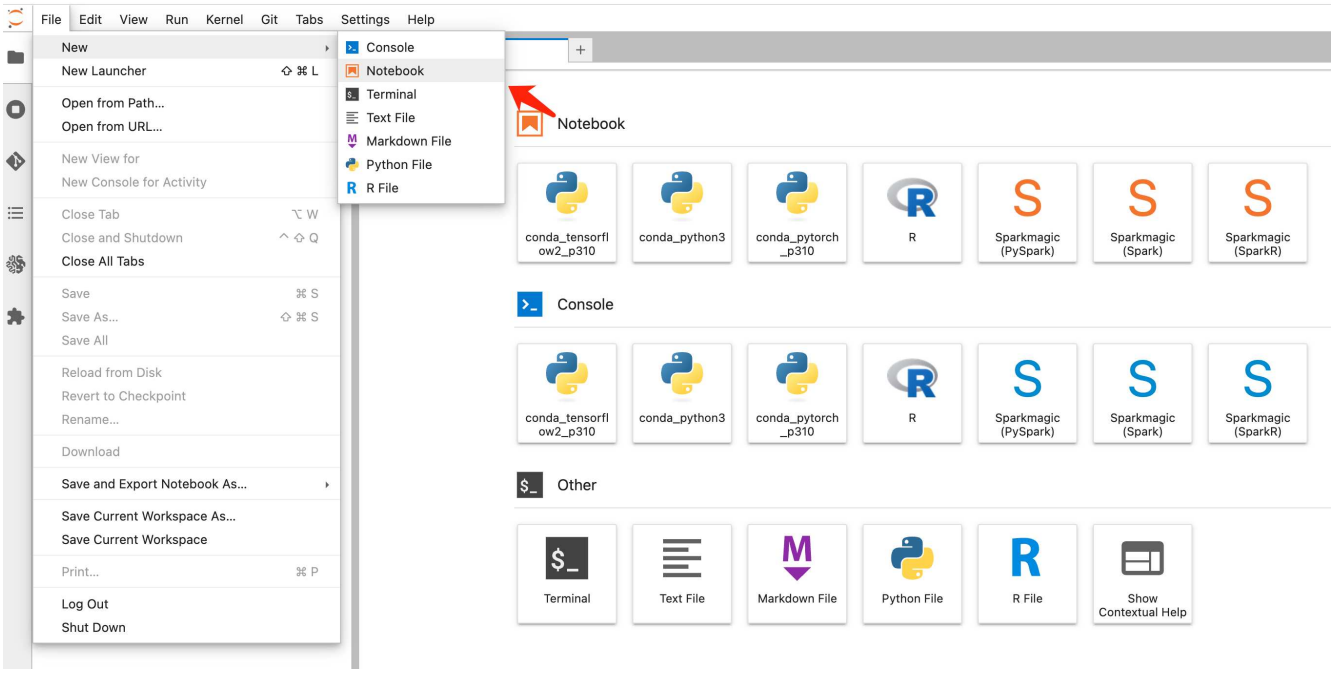
```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show
```

8. Conserve la IP y las credenciales del punto final para uso futuro.



**Configuración del cliente**

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




2. Utilice el código siguiente como solución alternativa para cargar archivos al depósito S3 privado de FSx ONTAP . 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 FSx ONTAP IP address>'            # Please get
this IP address from FSx ONTAP
# -------- 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 FSx ONTAP 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 FSx ONTAP 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 FSx ONTAP 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']
```

Con esto concluye la integración entre FSx ONTAP y la instancia de SageMaker.

## Lista de verificación de depuración útil

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

### Preguntas frecuentes (al 27 de septiembre de 2023)

P: ¿Por qué recibo el error "**Se produjo un error (no implementado) al llamar a la operación CreateMultipartUpload: el comando s3 que solicitó no está implementado**" al cargar archivos en FSx ONTAP?

R: Como depósito S3 privado, FSx ONTAP admite la carga de archivos de hasta 100 MB. Al utilizar el protocolo S3, los archivos de más de 100 MB se dividen en fragmentos de 100 MB y se llama a la función "CreateMultipartUpload". Sin embargo, la implementación actual de FSx ONTAP Private S3 no admite esta función.

P: ¿Por qué recibo el error "**Se produjo un error (Acceso denegado) al llamar a las operaciones PutObject: Acceso denegado**" al cargar archivos en FSx ONTAP?

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

P: ¿Cómo puedo integrar el bucket S3 privado de FSx ONTAP con otros servicios de SageMaker ML?

R: Lamentablemente, el SDK de servicios de SageMaker no proporciona una manera de especificar el punto final para el bucket S3 privado. Como resultado, FSx ONTAP S3 no es compatible con servicios de SageMaker como Sagemaker Data Wrangler, Sagemaker Clarify, Sagemaker Glue, Sagemaker Athena, Sagemaker AutoML y otros.

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

Este artículo es un tutorial sobre el uso de Amazon FSx for NetApp ONTAP (FSx ONTAP) para entrenar modelos de PyTorch en SageMaker, específicamente para un proyecto de clasificación de calidad de neumáticos.

## Introducción

Este tutorial ofrece un ejemplo práctico de un proyecto de clasificación de visión artificial y proporciona experiencia práctica en la creación de modelos de aprendizaje automático que utilizan FSx ONTAP 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 basándose en imágenes de neumáticos. Se enfatiza en el desarrollo de modelos de aprendizaje automático utilizando FSx ONTAP como fuente de datos en Amazon SageMaker.

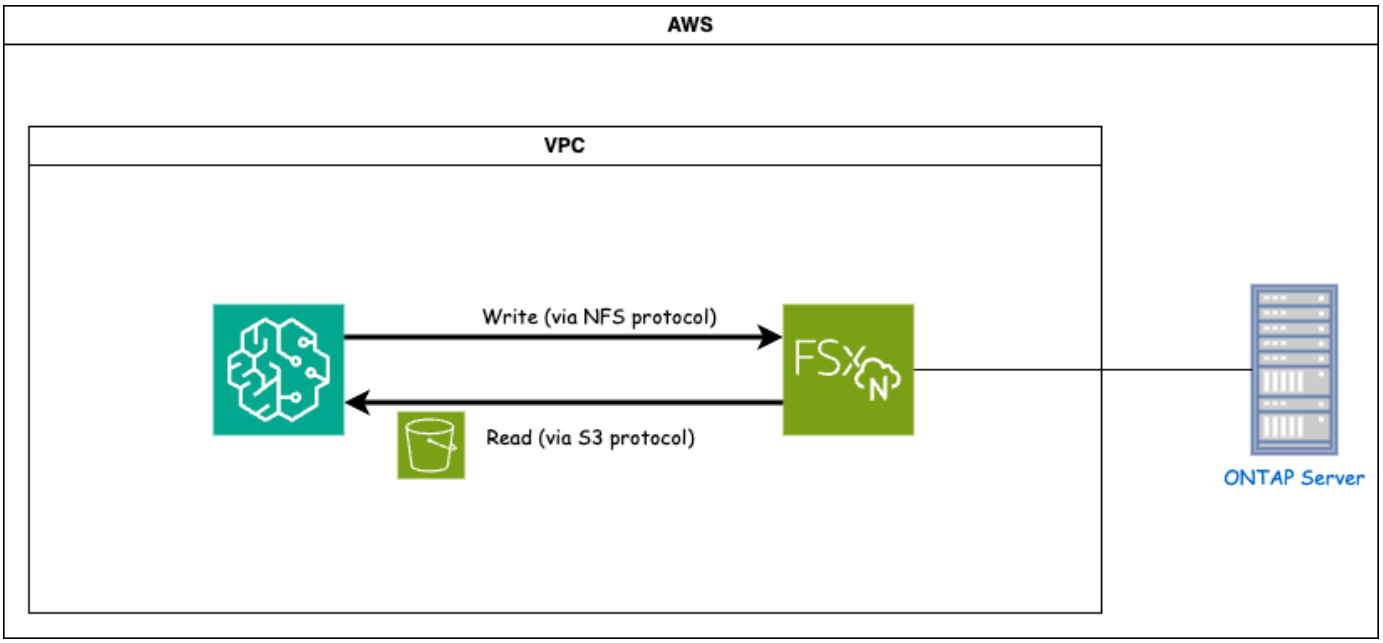
## ¿Qué es FSx ONTAP?

Amazon FSx ONTAP es de hecho una solución de almacenamiento totalmente administrada ofrecida por AWS. Aprovecha el sistema de archivos ONTAP de NetApp para proporcionar almacenamiento confiable y de alto rendimiento. Con soporte para protocolos como NFS, SMB e iSCSI, permite un acceso sin inconvenientes desde diferentes instancias de cómputo y contenedores. El servicio está diseñado para ofrecer un rendimiento excepcional, garantizando operaciones de datos rápidas y eficientes. También ofrece alta disponibilidad y durabilidad, garantizando que sus datos permanezcan accesibles y protegidos. Además, la capacidad de almacenamiento de Amazon FSx ONTAP es escalable, lo que le permite ajustarla fácilmente según sus

necesidades.

## Requisito previo

### Entorno de red



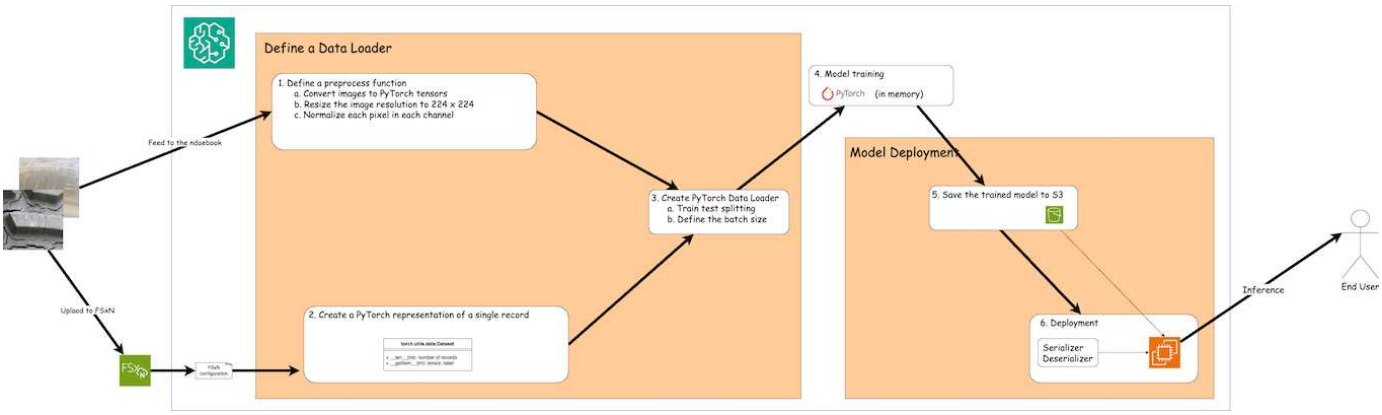

FSx ONTAP (Amazon FSx ONTAP) es un servicio de almacenamiento de AWS. Incluye un sistema de archivos que se ejecuta en el sistema NetApp ONTAP y una máquina virtual del sistema (SVM) administrada por AWS que se conecta a él. En el diagrama proporcionado, el servidor NetApp ONTAP administrado por AWS está ubicado fuera de la VPC. La SVM actúa como intermediario entre SageMaker y el sistema NetApp ONTAP , recibiendo solicitudes de operación de SageMaker y reenviándolas al almacenamiento subyacente. Para acceder a FSx ONTAP, SageMaker debe ubicarse dentro de la misma VPC que la implementación de FSx ONTAP . Esta configuración garantiza la comunicación y el acceso a los datos entre SageMaker y FSx ONTAP.

### Acceso a datos

En escenarios del mundo real, los científicos de datos normalmente utilizan los datos existentes almacenados en FSx ONTAP para construir sus modelos de aprendizaje automático. Sin embargo, para fines de demostración, dado que el sistema de archivos FSx ONTAP está inicialmente vacío después de su creación, es necesario cargar manualmente los datos de entrenamiento. Esto se puede lograr montando FSx ONTAP como un volumen en SageMaker. Una vez que el sistema de archivos esté montado correctamente, puede cargar su conjunto de datos en la ubicación montada, lo que lo hará accesible para entrenar sus modelos dentro del entorno de SageMaker. Este enfoque le permite aprovechar la capacidad de almacenamiento y las capacidades de FSx ONTAP mientras trabaja con SageMaker para el desarrollo y entrenamiento de modelos.

El proceso de lectura de datos implica configurar FSx ONTAP como un depósito S3 privado. Para conocer las instrucciones de configuración detalladas, consulte"Parte 1: Integración de Amazon FSx for NetApp ONTAP (FSx ONTAP) como un bucket S3 privado en AWS SageMaker"

## Descripción general de la integración




El flujo de trabajo de uso de datos de entrenamiento en FSx ONTAP para crear un modelo de aprendizaje profundo en SageMaker se puede resumir en tres pasos principales: definición del cargador de datos, entrenamiento del modelo e implementación. En un nivel alto, estos pasos forman la base de un flujo de trabajo MLOps. Sin embargo, cada paso implica varios subpasos detallados para una implementación integral. Estos subpasos abarcan varias tareas, como el preprocesamiento de datos, la división del conjunto de datos, la configuración del modelo, el ajuste de hiperparámetros, la evaluación del modelo y la implementación del modelo. Estos pasos garantizan un proceso exhaustivo y eficaz para crear e implementar modelos de aprendizaje profundo utilizando datos de entrenamiento de FSx ONTAP dentro del entorno de SageMaker.

## Integración paso a paso

### Loader 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 solo define el tamaño del lote, sino que también determina el procedimiento para leer y preprocesar cada registro dentro del lote. Al configurar el cargador de datos, podemos gestionar 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 transformaciones de preprocesamiento de imágenes utilizando el módulo **torchvision.transforms**. 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 tensorial. Posteriormente, la transformación **Resize  224,224** redimensiona la imagen a un

tamaño fijo de 224x224 píxeles. Finalmente, la transformación **Normalize()** normaliza los valores del tensor restando la media y dividiéndolos por la desviación estándar a lo largo de cada canal. Los valores de media y desviación estándar utilizados para la normalización se emplean comúnmente en modelos de redes neuronales preentrenados. En general, este código prepara los datos de la imagen para su posterior procesamiento o ingreso a un modelo entrenado previamente convirtiéndolos en un tensor, redimensionándolos y normalizando los valores de los píxeles.

#### La clase de conjunto de datos de PyTorch

```
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 S3 boto3 para recuperar los datos binarios de FSx ONTAP. El estilo de código para acceder a datos de FSx ONTAP es similar al de leer datos de Amazon S3. La explicación siguiente profundiza en el proceso de creación del objeto privado S3 **bucket**.

**FSx ONTAP como repositorio privado de S3**

```
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 FSx ONTAP 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 FSx ONTAP en SageMaker, se crea un controlador que apunta al almacenamiento de FSx ONTAP mediante el protocolo S3. Esto permite que FSx ONTAP sea tratado como un depósito S3 privado. La configuración del controlador incluye la especificación de la dirección IP del SVM de FSx ONTAP , el nombre del depósito y las credenciales necesarias. Para obtener una explicación completa sobre cómo obtener estos elementos de configuración, consulte el documento en"Parte 1: Integración de Amazon FSx for NetApp ONTAP (FSx ONTAP) como un bucket S3 privado en AWS SageMaker" .

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

#### El Loader 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 **Dataset** de PyTorch, la función de preprocesamiento y el tamaño del lote de entrenamiento, obtenemos el cargador de datos para el entrenamiento. Este cargador de datos facilita el proceso de iteración a través del 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** que utiliza capas convolucionales y una capa lineal para clasificar la calidad de los neumáticos. El bucle de entrenamiento itera sobre 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 para fines de monitoreo.

### Implementación del modelo

#### Despliegue

```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 cargar el modelo en **Amazon S3**, se vuelve accesible para SageMaker, lo que permite la implementación y la inferencia en el modelo implementado.

```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 la implementación de un modelo de PyTorch en SageMaker. Define un serializador personalizado, **TyreQualitySerializer**, que preprocesa y serializa los datos de entrada como un tensor de 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, la función 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. Finalmente, el modelo se implementa utilizando el método de implementación, especificando la cantidad de instancias, el tipo de instancia y el nombre del punto final generado. Esto permite que el modelo PyTorch se implemente y sea accesible para 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 del uso del punto final implementado para realizar la inferencia.

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

Este artículo proporciona una guía para crear una canalización de MLOps con servicios de AWS, centrándose en el reentrenamiento automatizado de modelos, la implementación y la optimización de costos.

## Introducción

En este tutorial, aprenderá cómo aprovechar varios servicios de AWS para construir una canalización MLOps simple que abarque integración continua (CI), capacitación continua (CT) e implementación continua (CD). A diferencia de los pipelines DevOps tradicionales, MLOps requiere consideraciones adicionales para completar el ciclo operativo. Al seguir este tutorial, obtendrá conocimientos sobre cómo incorporar CT en el ciclo MLOps, lo que permitirá el entrenamiento continuo de sus modelos y una implementación perfecta para la inferencia. El tutorial lo guiará a través del proceso de utilización de los servicios de AWS para establecer esta canalización MLOps de extremo a extremo.

## Manifiesto

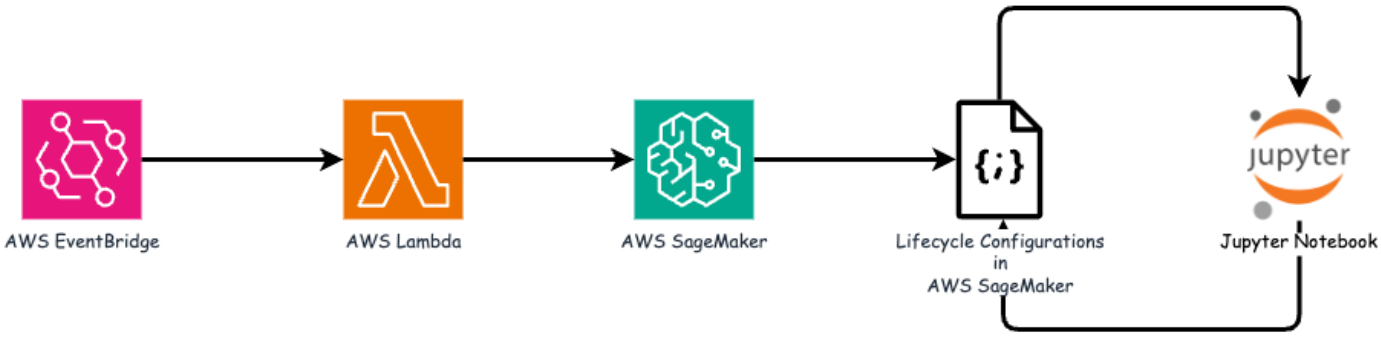
| Funcionalidad | Nombre | Comentario |
|---|---|---|
| Almacenamiento de datos | AWS FSx ONTAP | Consulte "Parte 1: Integración de Amazon FSx for NetApp ONTAP (FSx ONTAP) como un bucket S3 privado en AWS SageMaker" . |
| IDE de ciencia de datos | AWS SageMaker | Este tutorial se basa en el cuaderno Jupyter presentado en"Parte 2: Aprovechamiento de Amazon FSx for NetApp ONTAP (FSx ONTAP) como fuente de datos para el entrenamiento de modelos en SageMaker" . |

| Funcionalidad | Nombre | Comentario |
|---|---|---|
| Función para activar la canalización MLOps | Función AWS Lambda | - |
| Desencadenador de trabajo cron | Puente de eventos de AWS | - |
| Marco de aprendizaje profundo | PyTorch | - |
| SDK de Python de AWS | boto3 | - |
| Lenguaje de programación | Pitón | v3.10 |

## Requisito previo

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

## Arquitectura




Esta canalización 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. **AWS EventBridge** actúa como trabajo cron. Periódicamente invoca una **función AWS Lambda** responsable de volver a entrenar y reimplementar el modelo. Este proceso implica poner en marcha la instancia **AWS SageMaker Notebook** para realizar las tareas necesarias.

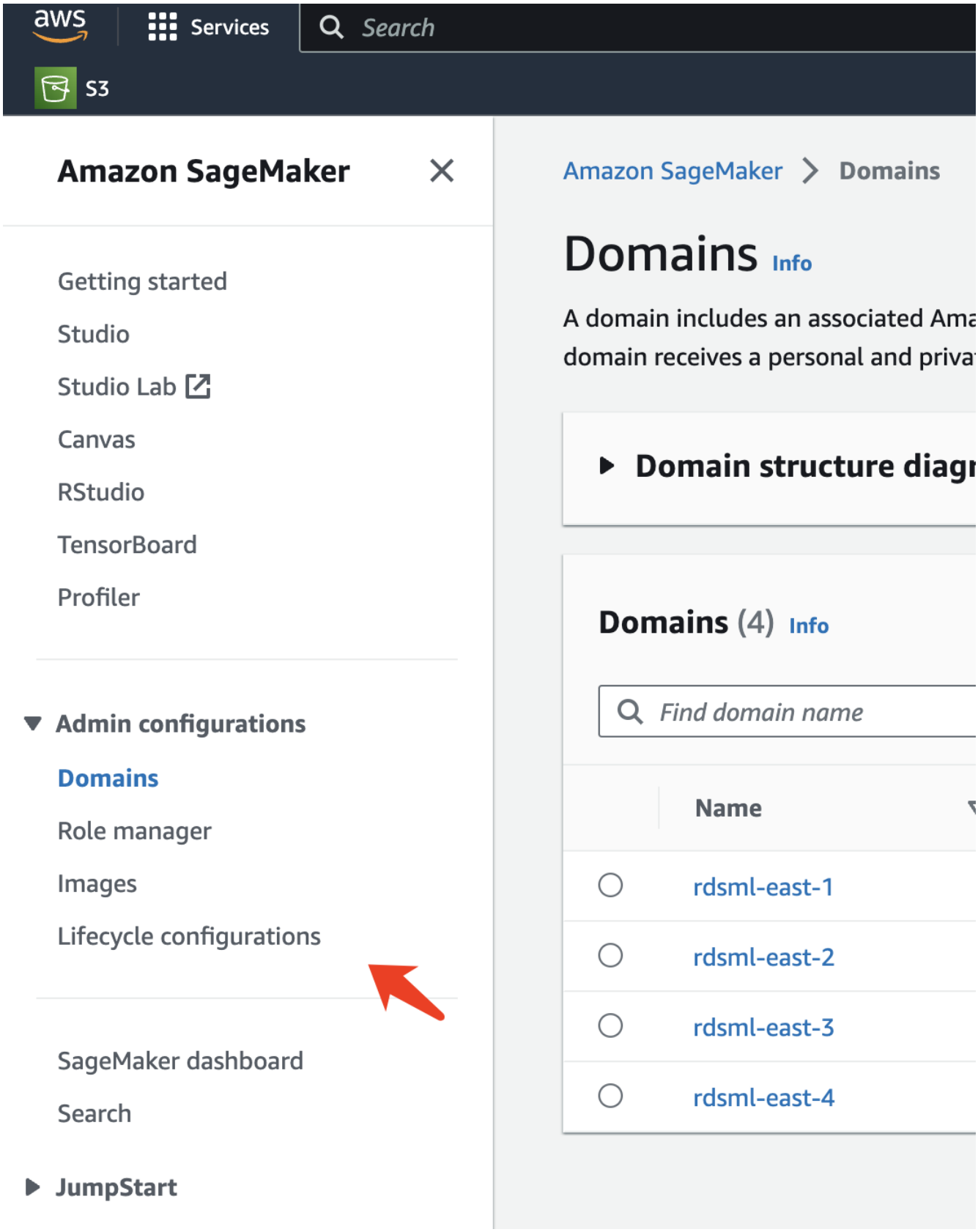
## Configuración paso a paso

### Configuraciones del ciclo de vida

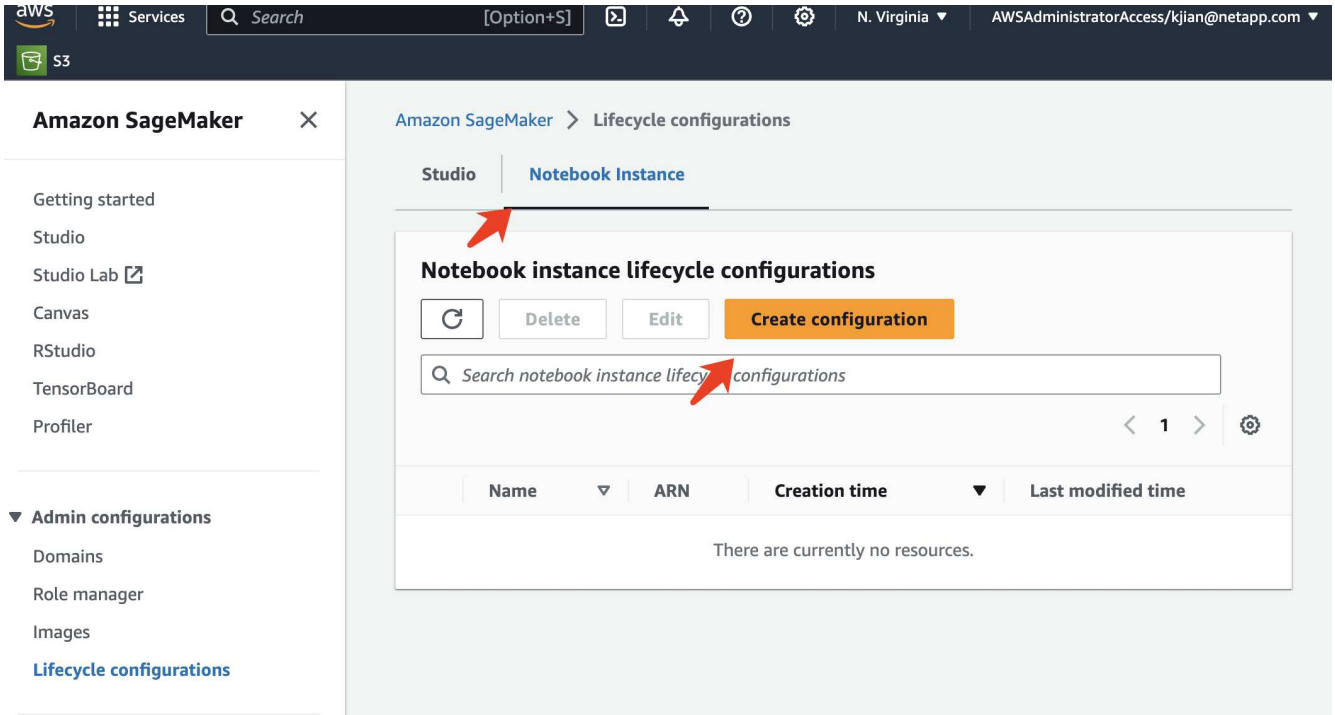
Para configurar la función de devolución de llamada del ciclo de vida para la instancia de AWS SageMaker Notebook, deberá utilizar **Configuraciones de ciclo de vida**. Este servicio le permite definir las acciones necesarias que se deben realizar durante el inicio de la instancia del notebook. Específicamente, se puede implementar un script de shell dentro de las **Configuraciones del ciclo de vida** para apagar automáticamente la instancia del notebook una vez que se hayan completado los procesos de entrenamiento e implementación. Esta es una configuración obligatoria ya que el costo es una de las principales consideraciones en MLOps.

Es importante tener en cuenta que la configuración para las **Configuraciones del ciclo de vida** debe realizarse con anticipación. Por lo tanto, se recomienda priorizar la configuración de este aspecto antes de continuar con la configuración del resto del pipeline de MLOps.

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



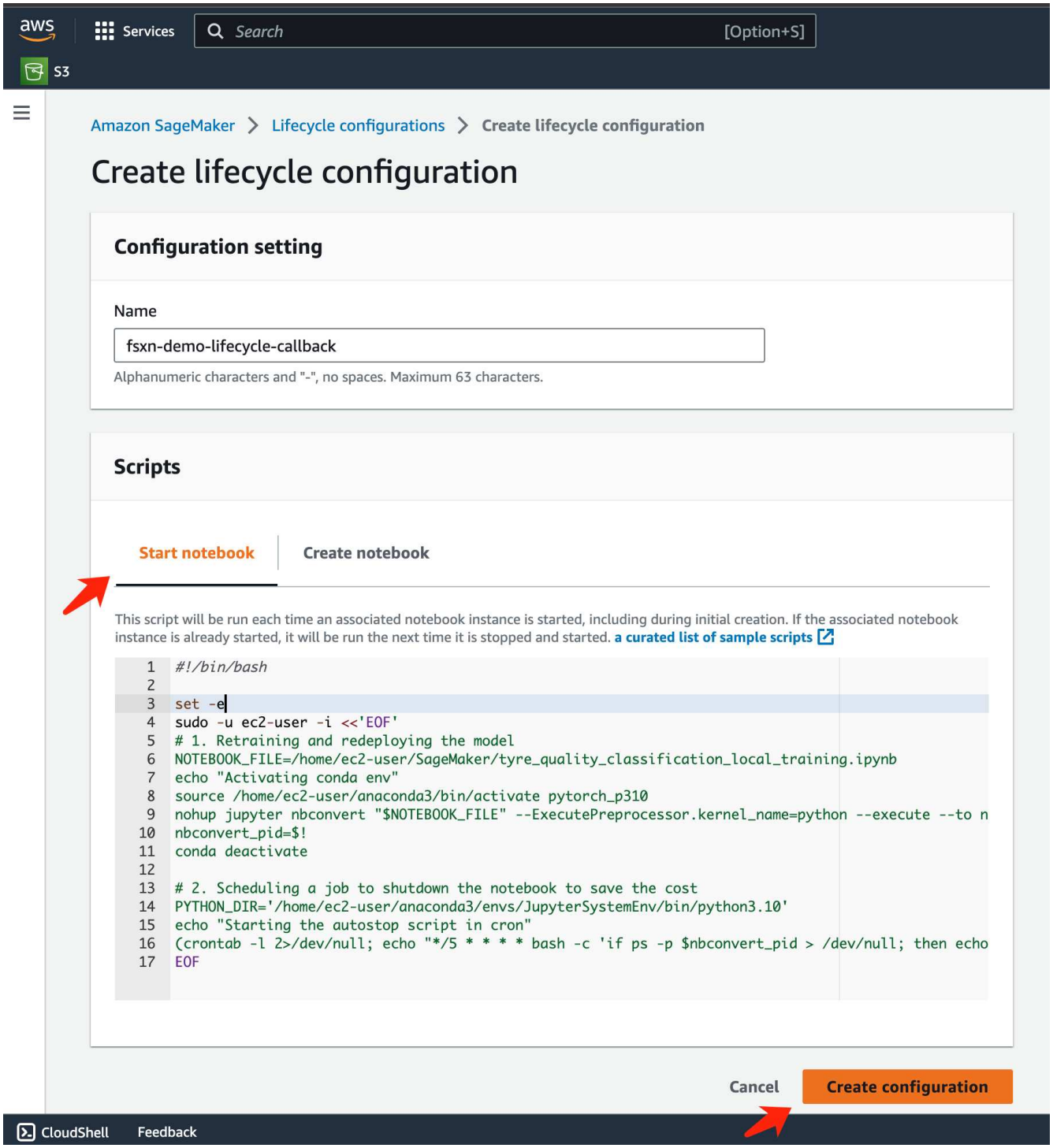

2. Seleccione la pestaña **Instancia de Notebook** y haga clic en el botón **Crear configuración**



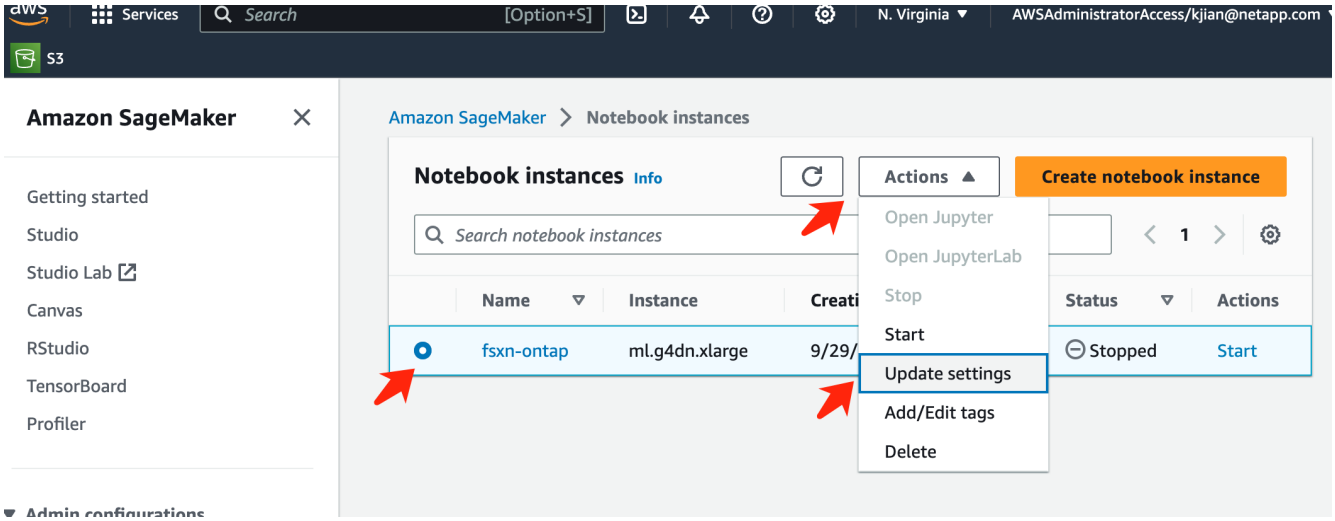

3. Pegue el siguiente código en el área de entrada.

```
#!/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
```

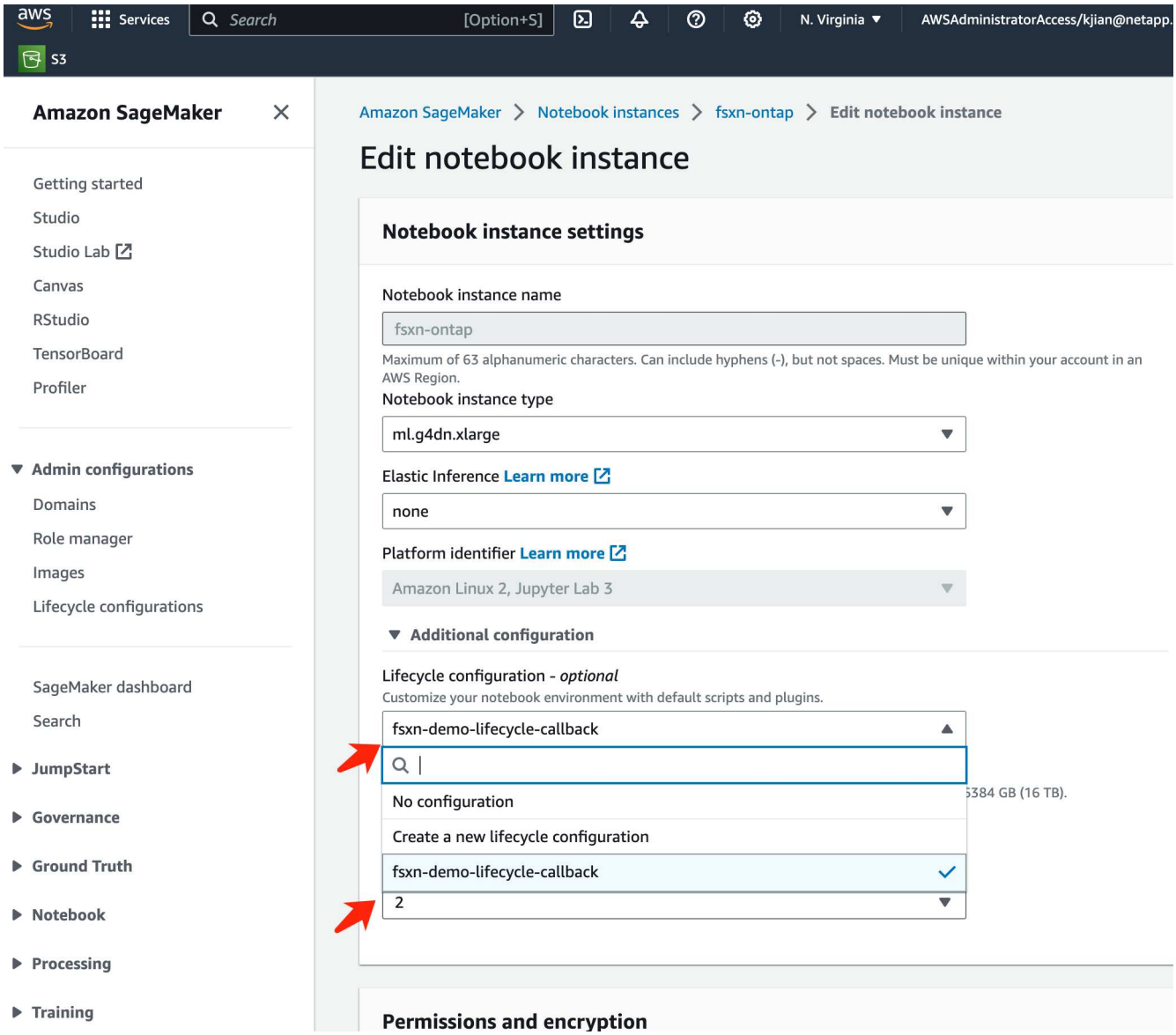
4. Este script ejecuta Jupyter Notebook, que maneja el reentrenamiento y la redistribución del modelo para la inferencia. Una vez completada la ejecución, la computadora portátil se apagará automáticamente en 5 minutos. Para obtener más información sobre el enunciado del problema y la implementación del código, consulte"Parte 2: Aprovechamiento de Amazon FSx for NetApp ONTAP (FSx ONTAP) como fuente de datos para el entrenamiento de modelos en SageMaker" .




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



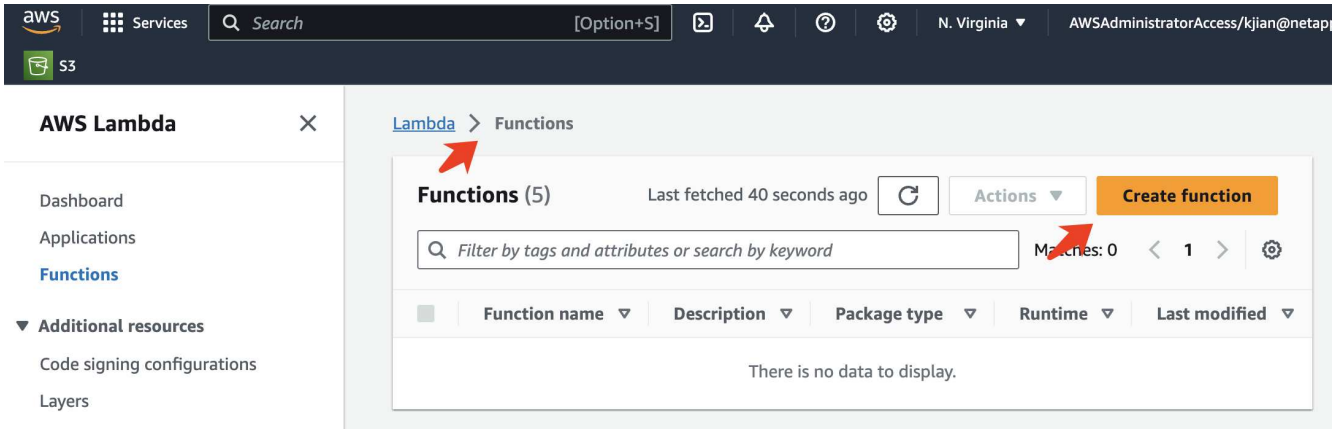

6. Seleccione la **Configuración de ciclo de vida** creada y haga clic en **Actualizar instancia de notebook**.



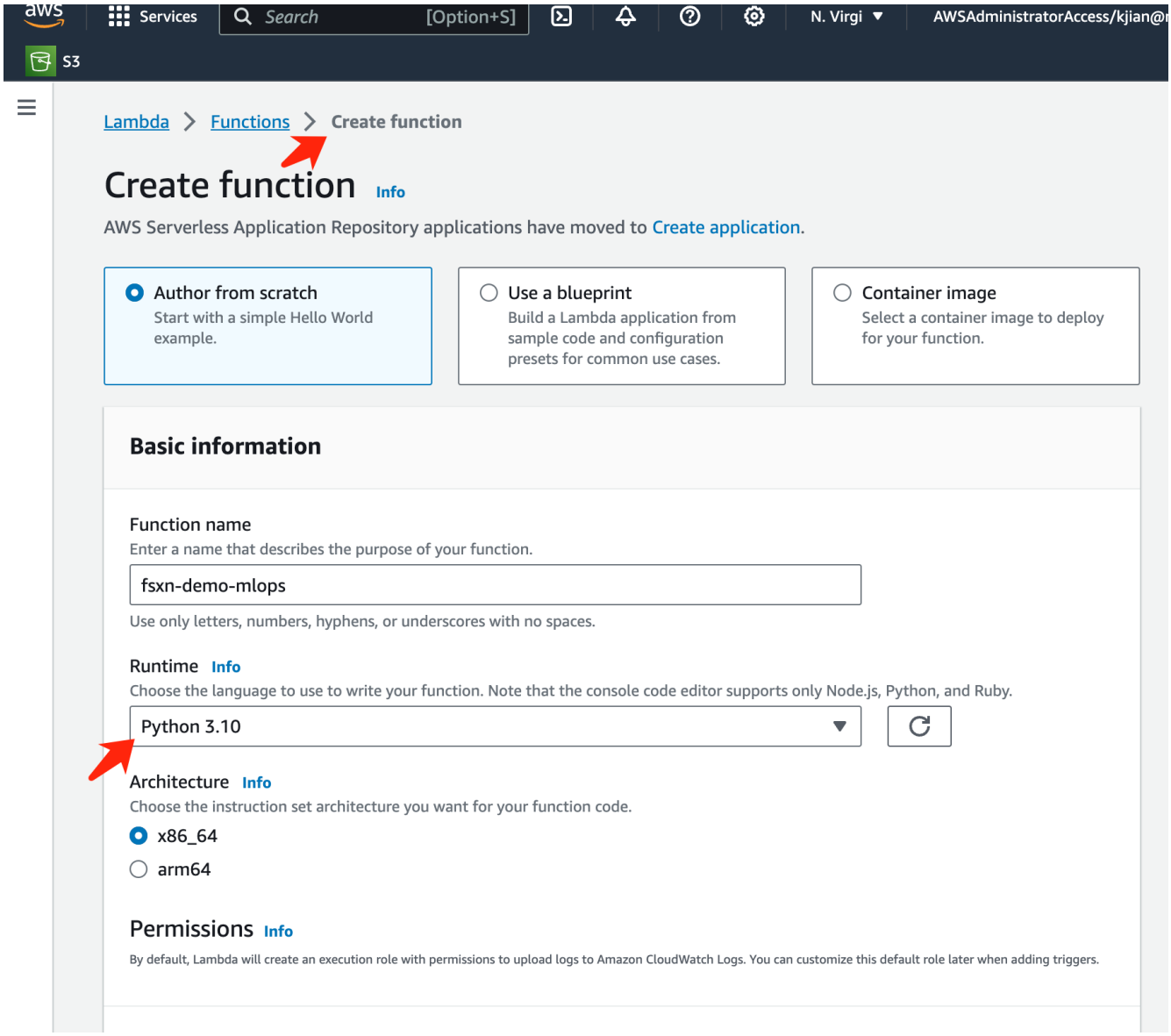
### Función sin servidor de AWS Lambda

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

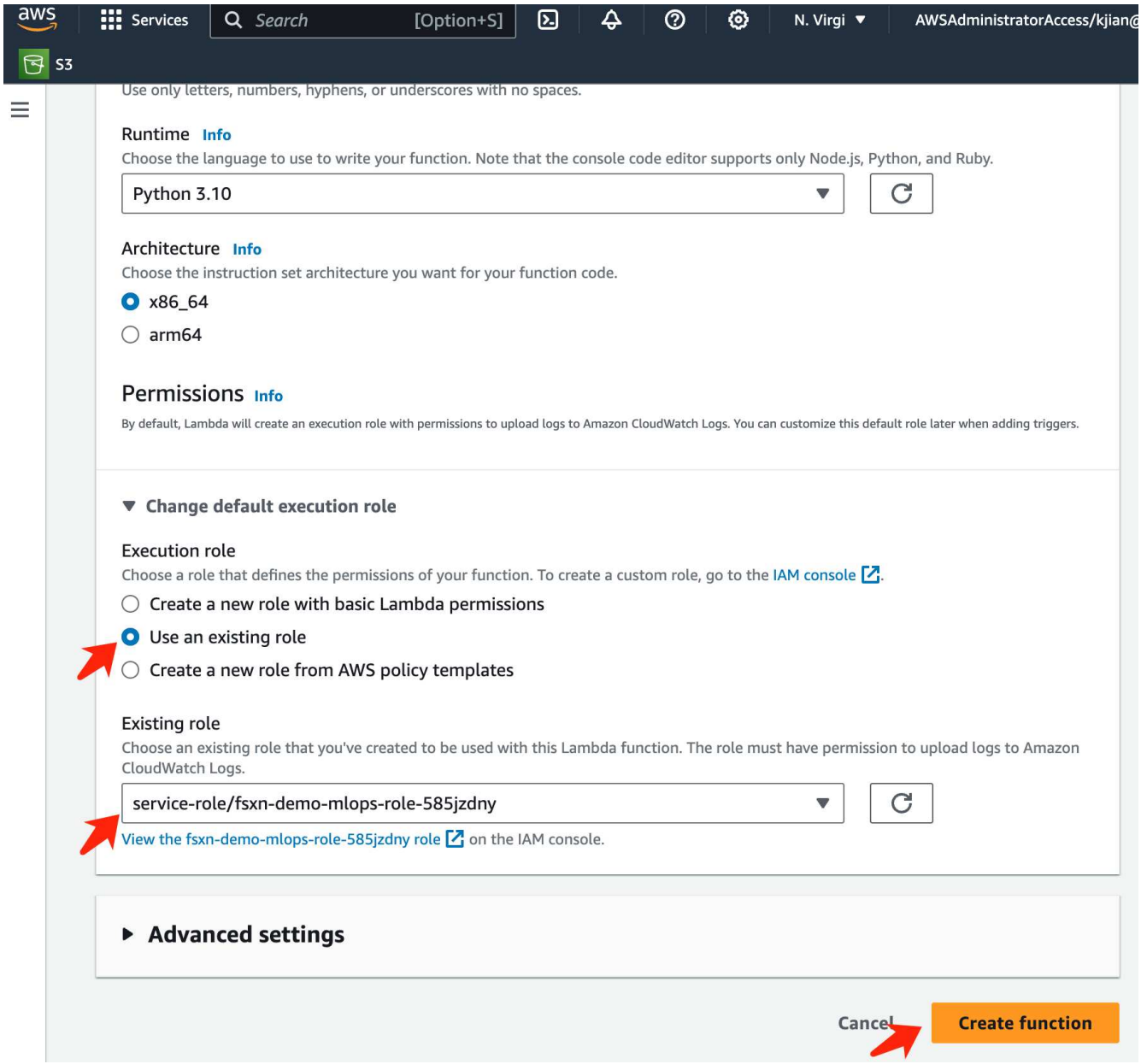
1. Para crear una **función AWS Lambda**, navegue al panel correspondiente, cambie a la pestaña **Funciones** y haga clic en **Crear función**.




2. Complete todas las entradas requeridas en la página y recuerde cambiar el tiempo de ejecución a **Python 3.10**.



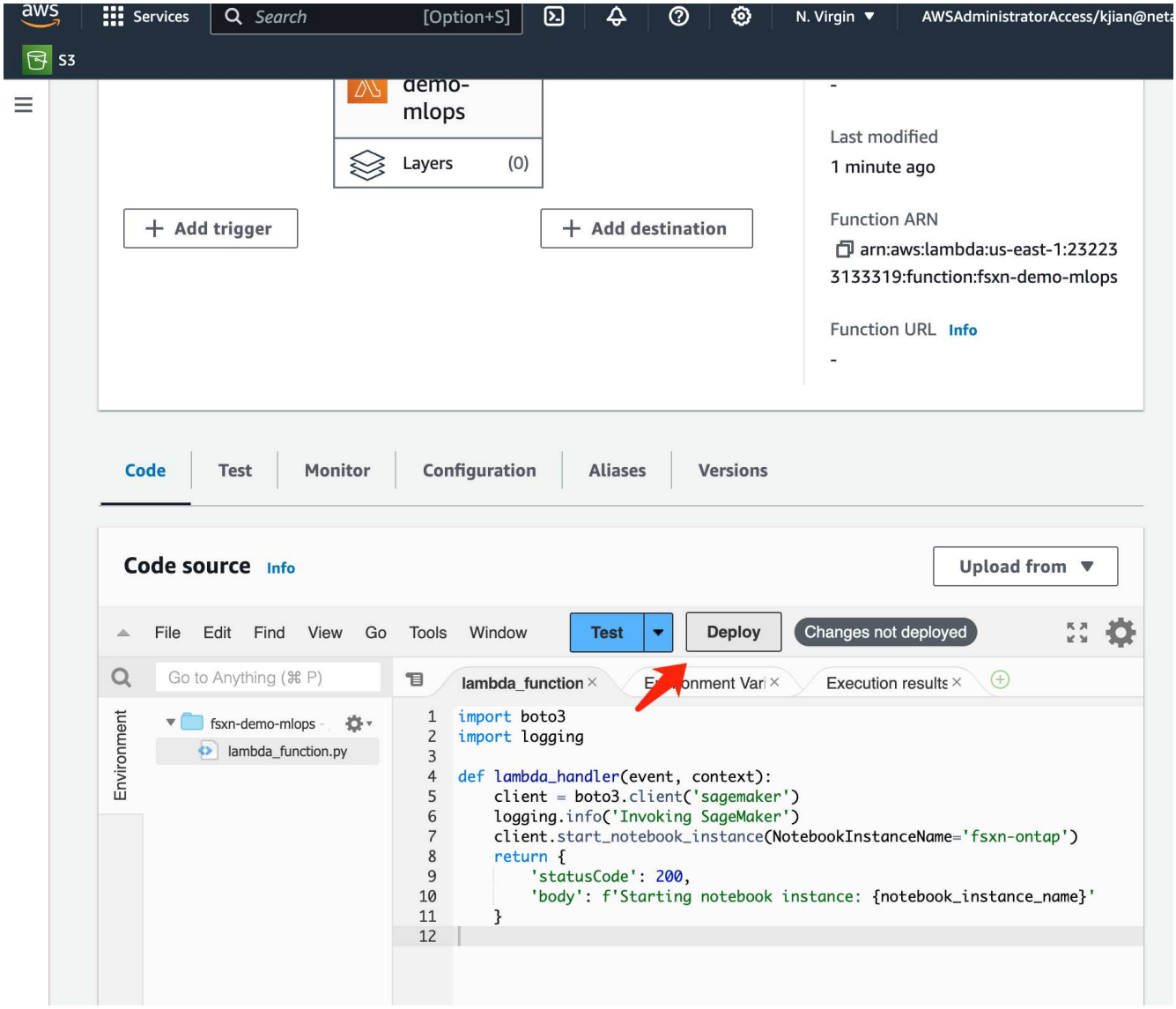

3. Verifique que el rol designado tenga el permiso requerido **AmazonSageMakerFullAccess** y haga clic en el botón **Crear función**.



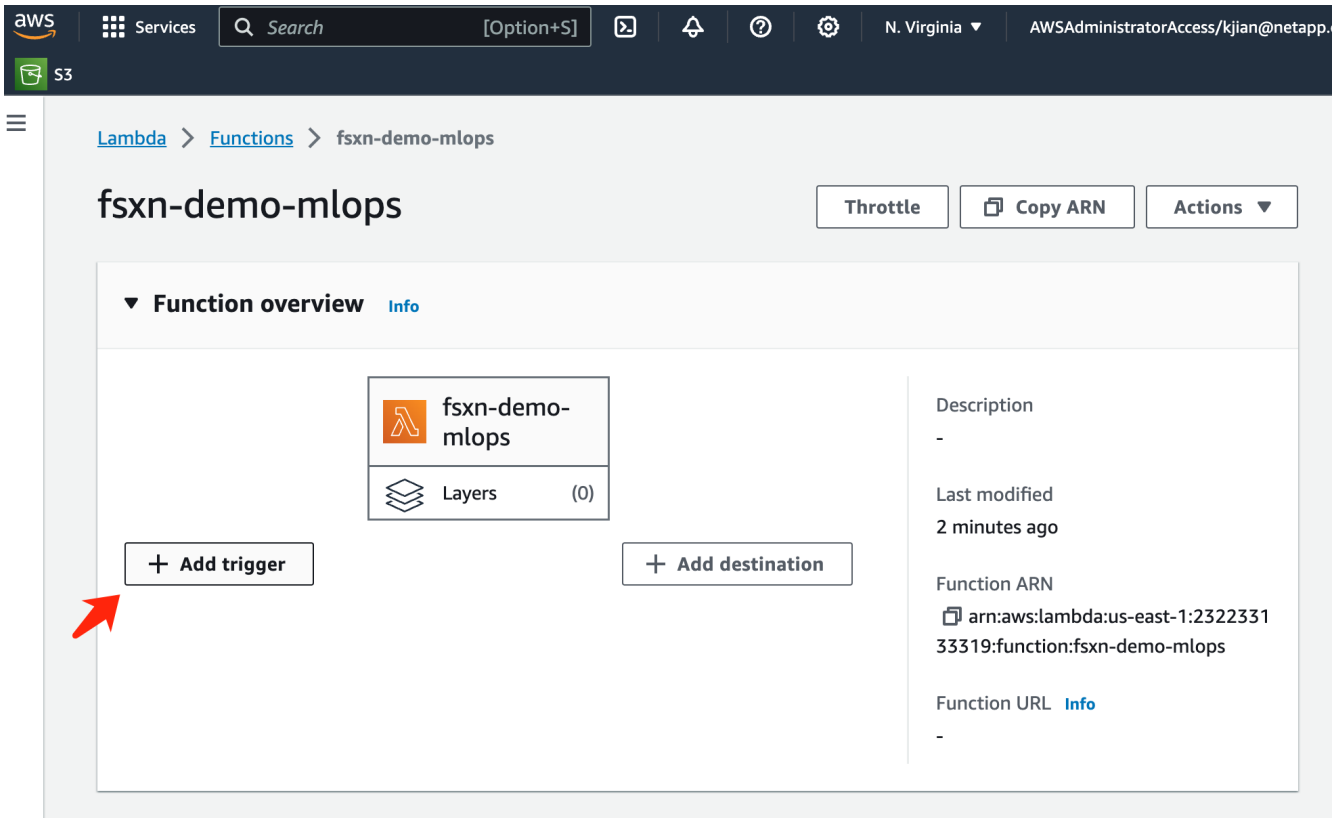

4. Seleccione la función Lambda creada. En la pestaña de código, copie y pegue el siguiente código en el área de texto. Este código inicia la instancia del cuaderno denominada **fsxn-ontap**.

```
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}'
    }
```

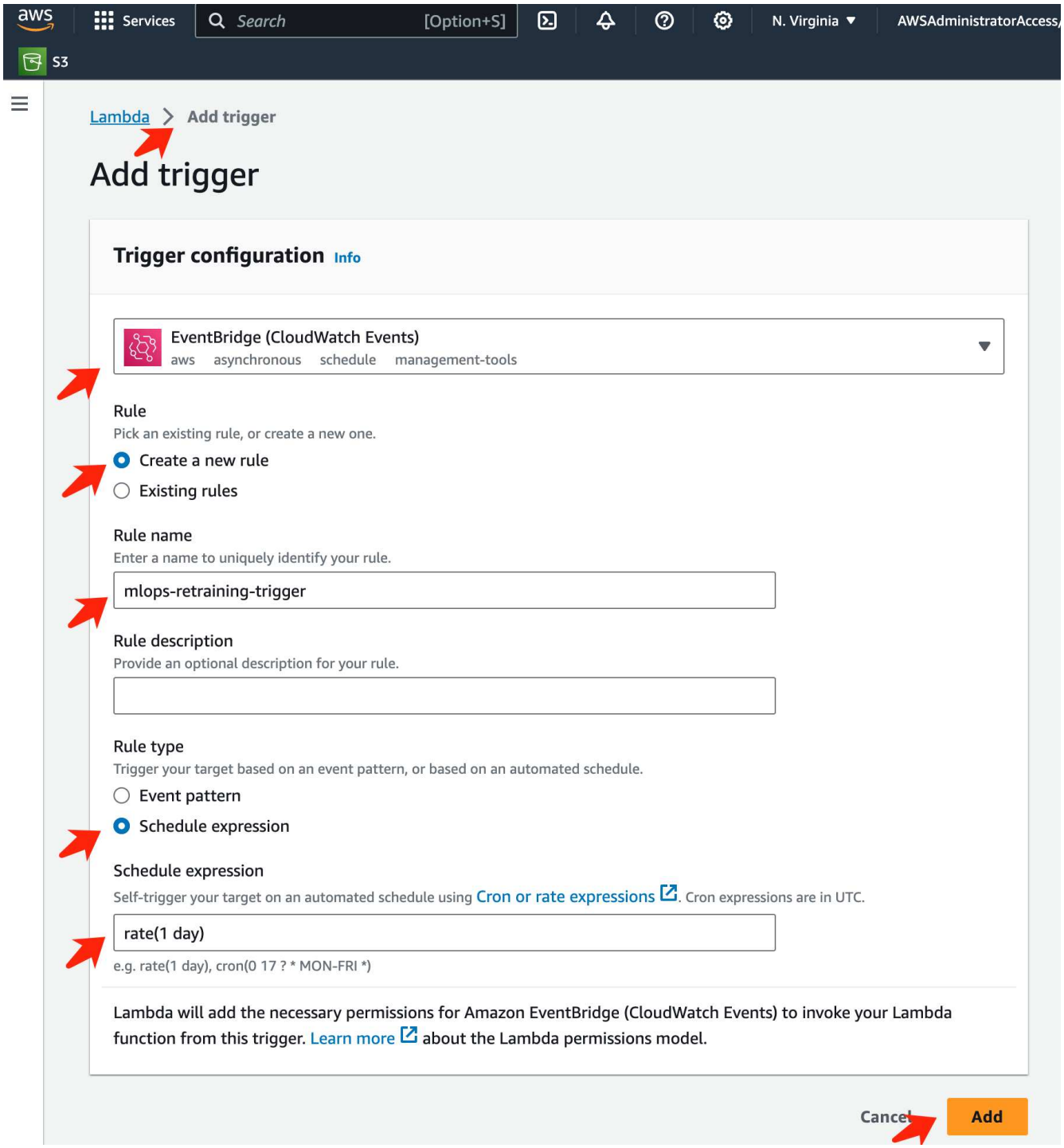
5. Haga clic en el botón **Implementar** para aplicar este cambio de código.




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




7. Seleccione EventBridge en el menú desplegable, luego haga clic en el botón denominado Crear una nueva regla. En el campo de expresión de programación, ingrese `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 de dos pasos, diariamente, la **función AWS Lambda** iniciará **SageMaker Notebook**, realizará un reentrenamiento del modelo utilizando los datos del repositorio **FSx ONTAP**, volverá a implementar el modelo actualizado en el entorno de producción y apagará automáticamente la **instancia de SageMaker Notebook** para optimizar los costos. Esto garantiza que el modelo se mantenga actualizado.

Con esto concluye el tutorial para desarrollar un pipeline MLOps.

**Información de copyright**

Copyright © 2025 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

**Información de la marca comercial**

NETAPP, el logotipo de NETAPP y las marcas que constan en http://www.netapp.com/TM son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.