适用于MLOps的AWS FSx for NetApp ONTAP (FSxN)

NetApp Solutions

NetApp
April 12, 2024

目录

适用于MLOps的AWS FSx for NetApp ONTAP (FSxN)

作者：
Jian Jian (Ken)、NetApp高级数据和应用科学人员

本节将深入介绍AI基础架构开发的实际应用、提供使用FSxN构建MLOps管道的端到端逐步介绍。它包含三个全面的示例、可指导您通过这一强大的数据管理平台满足MLOps需求。

这些文章侧重于：

1. "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"
2. "第2部分—利用AWS FSx for NetApp ONTAP (FSxN)作为SageMaker模型训练的数据源"
3. "第3部分-构建简化的MLOps管道(CI/CT/CD)"

本节结束时、您将深入了解如何使用FSxN简化MLOps流程。

第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中

作者：
Jian Jian (Ken)、NetApp高级数据和应用科学人员

简介

本页以SageMaker为例、提供将FSxN配置为私有S3存储分段的指导。

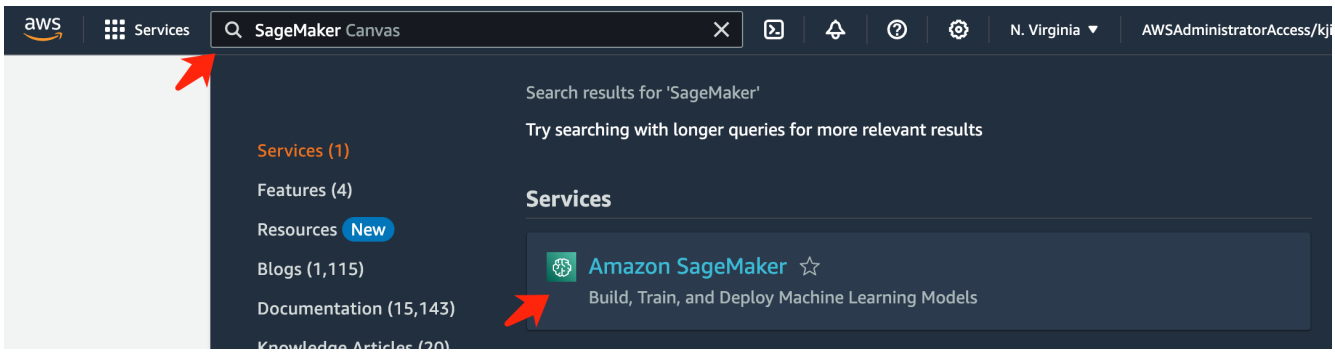
有关FSxN的更多信息、请观看此演示文稿("视频链接")

用户指南

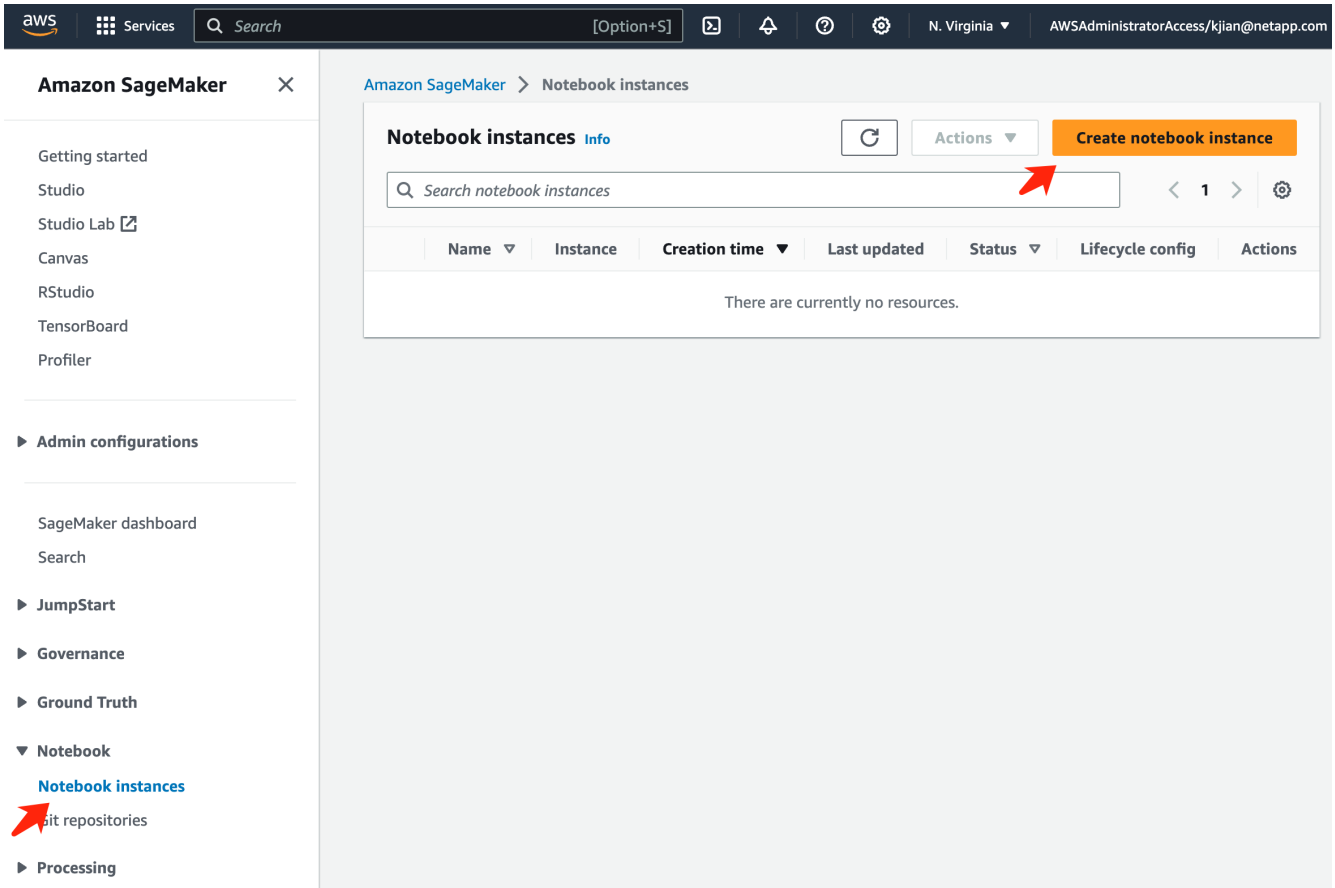
创建服务器

创建**SageMaker**笔记本实例

1. 打开AWS控制台。在搜索面板中、搜索SageMaker并单击服务*亚马逊SageMaker *。



2. 打开“笔记本”选项卡下的“笔记本实例”，单击橙色按钮*创建笔记本实例*。



3. 在创建页面中、
   输入*笔记本实例名称*
   展开*Network*面板
   保留其它条目默认值，然后选择*VPC*、**Subnet***和***Security group**。(稍后将使用此*VPC*和*Subnet*创建FSxN文件系统)
   单击右下角的橙色按钮*创建笔记本实例*。

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

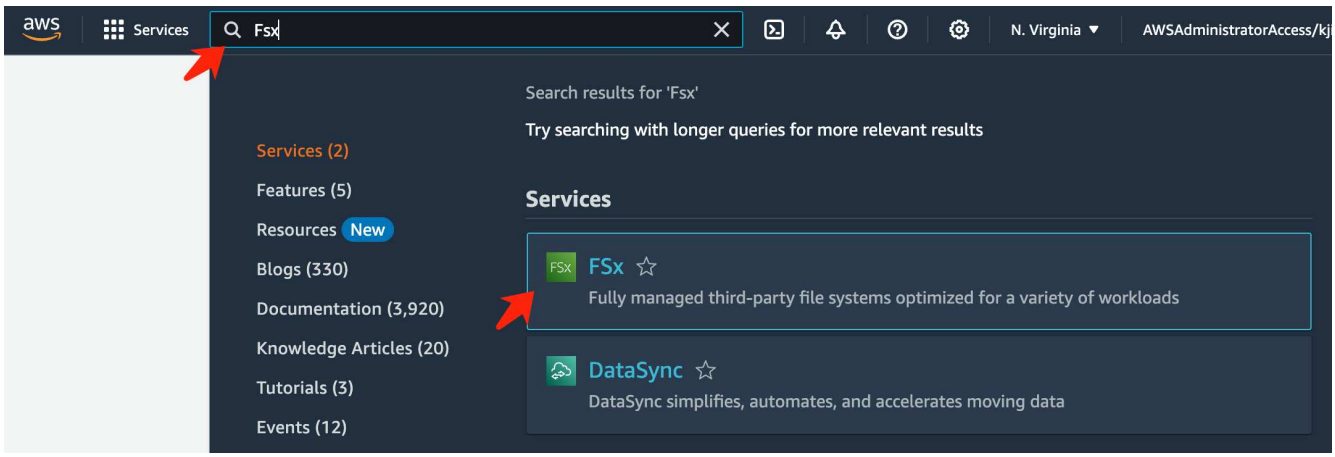
▶ Git repositories- *optional*

▶ Tags - *optional*

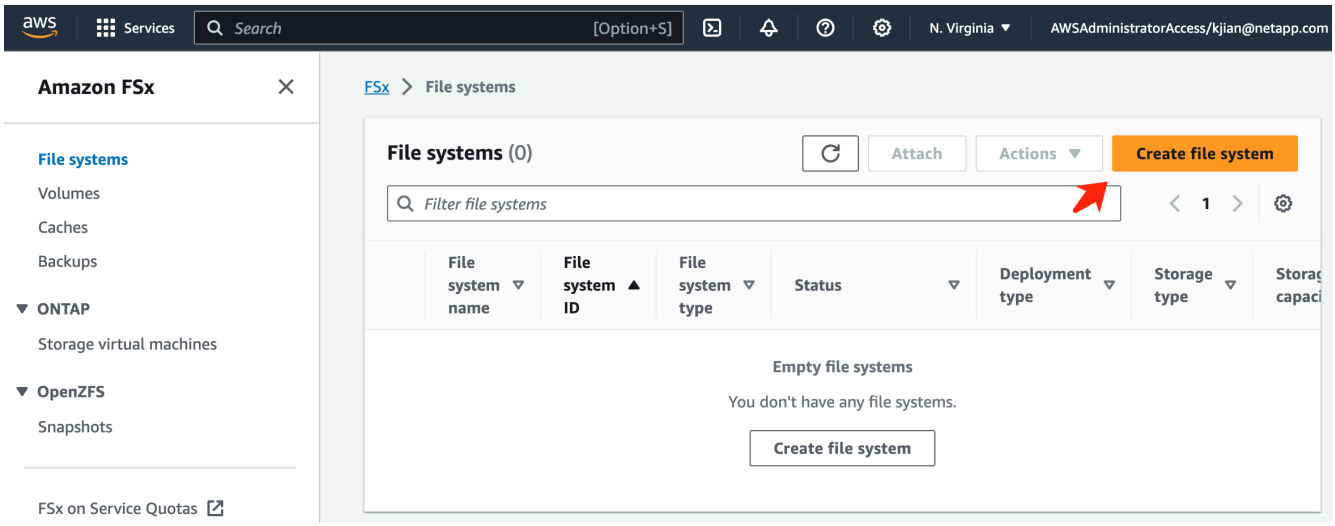
Cancel | Create notebook instance

创建**FSxN**文件系统

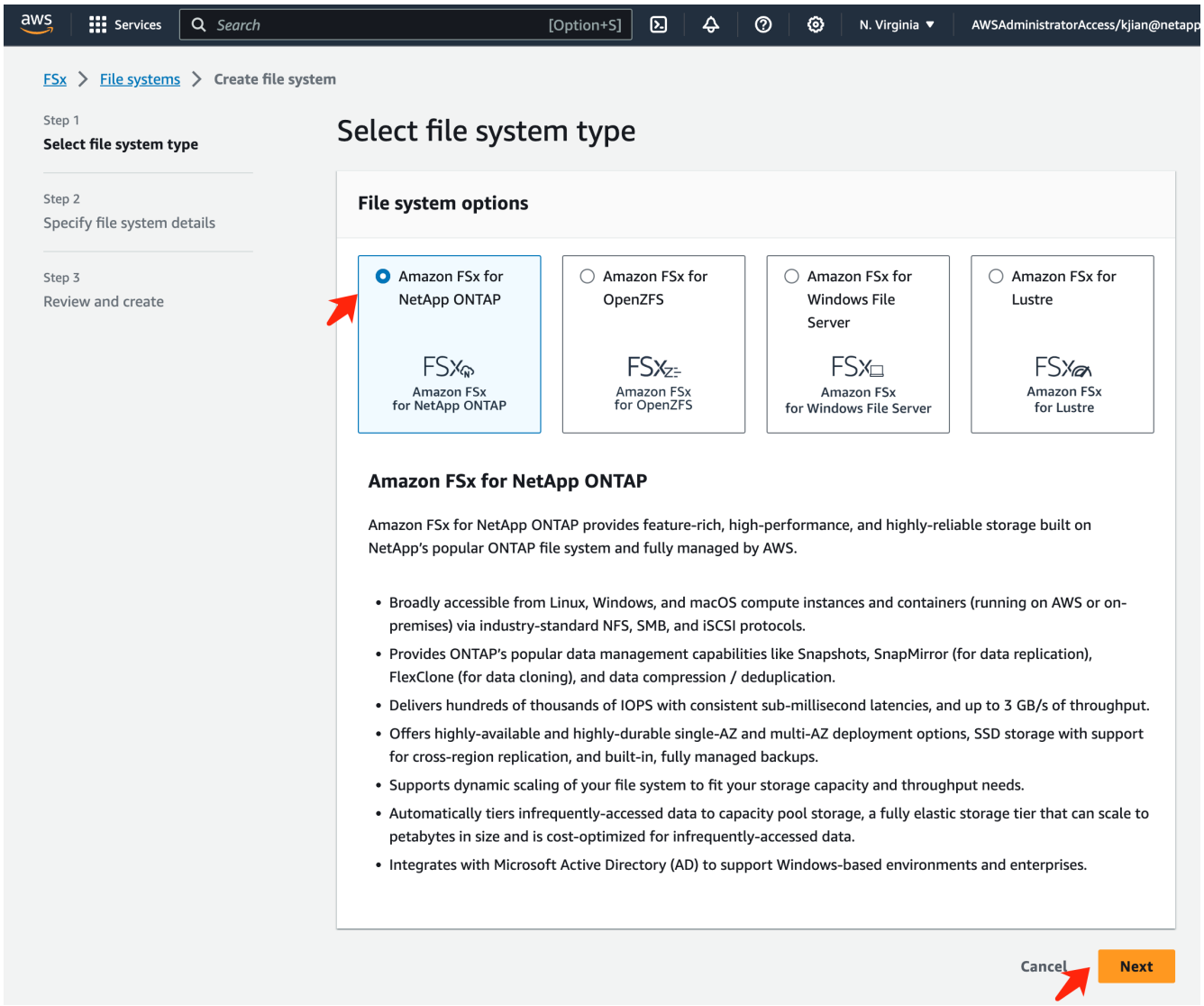
1. 打开AWS控制台。在搜索面板中，搜索FSx并单击服务*FSX*。



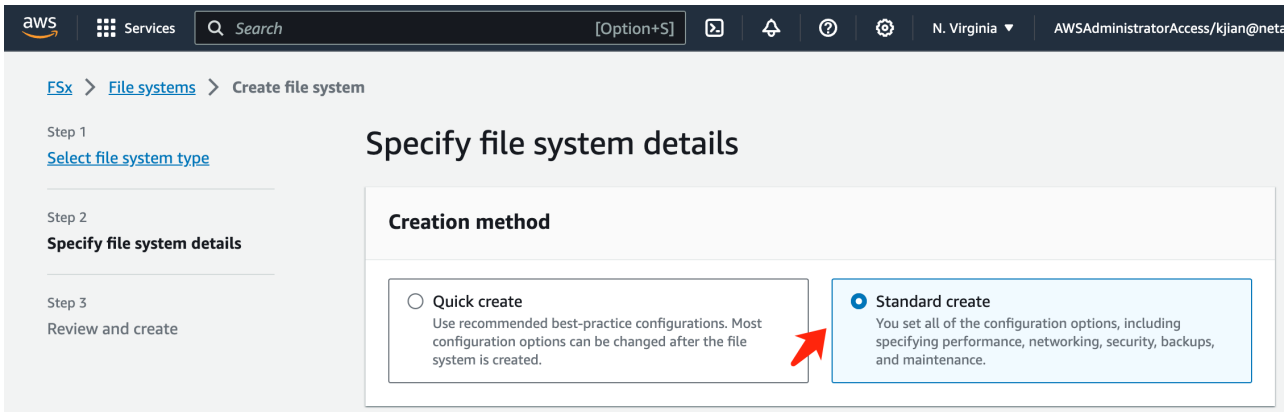
2. 单击*创建文件系统*。



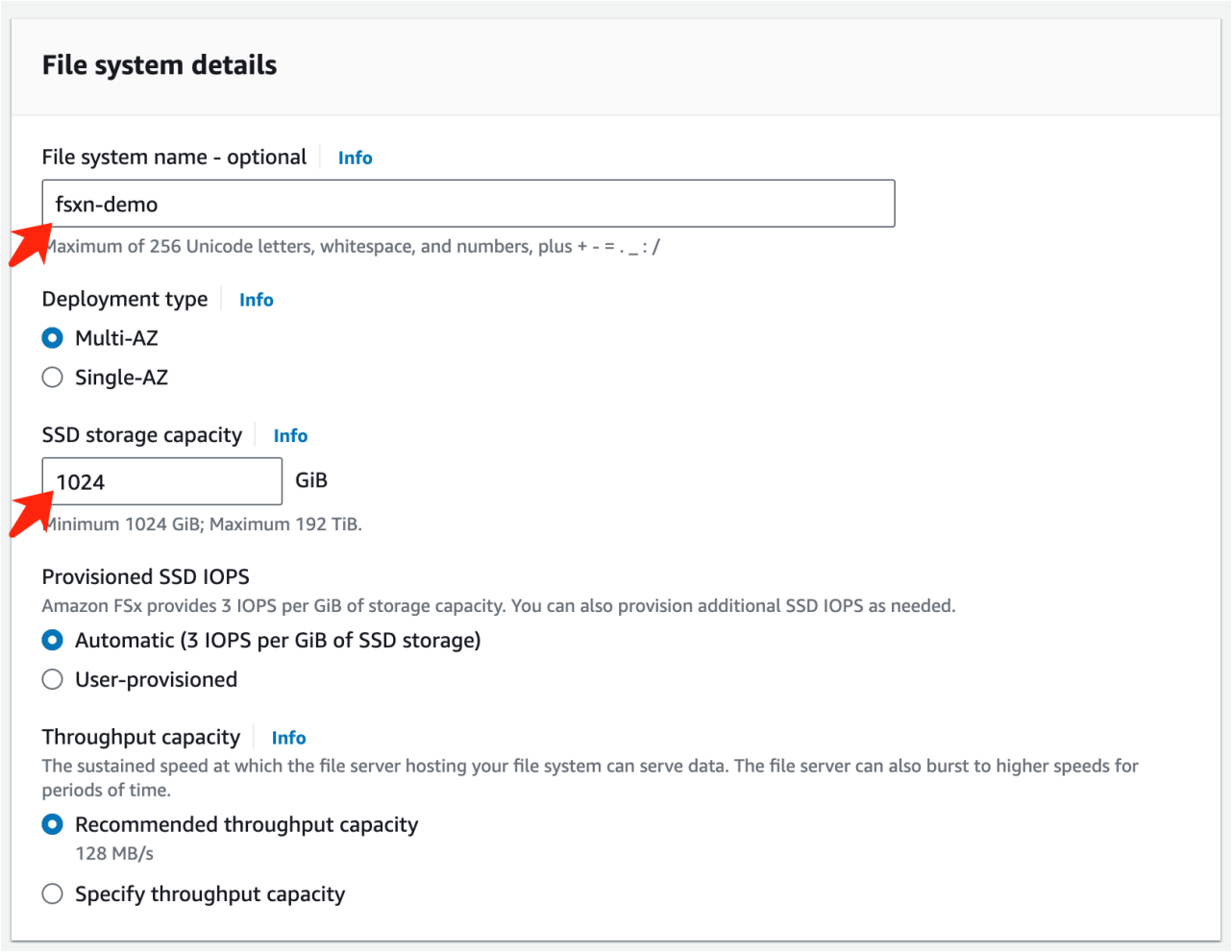
3. 选择第一张卡*FSx for FS* NetApp ONTAP，然后单击*Next*。



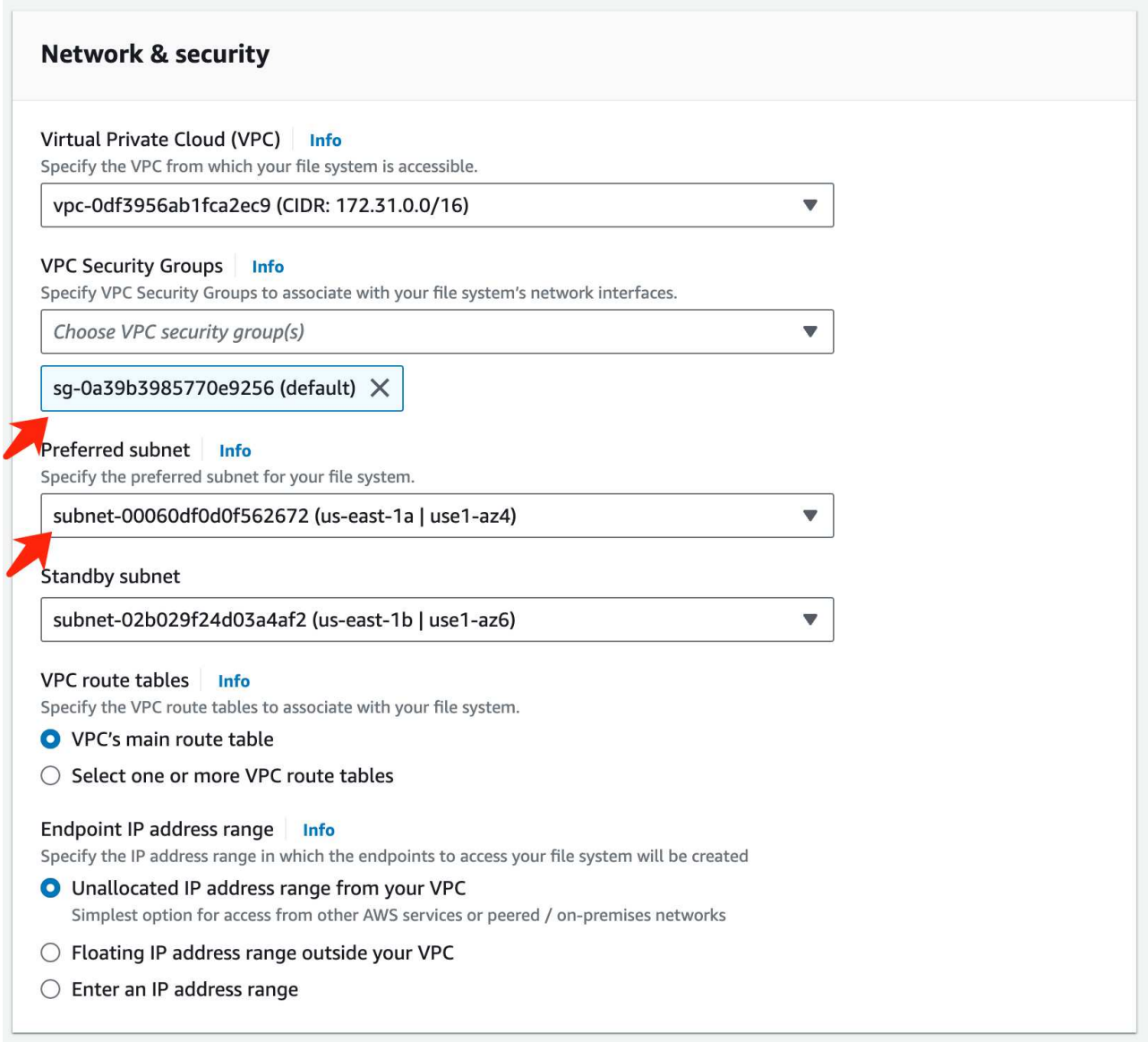
4. 在详细信息配置页面中。
   a. 选择*标准创建*选项。



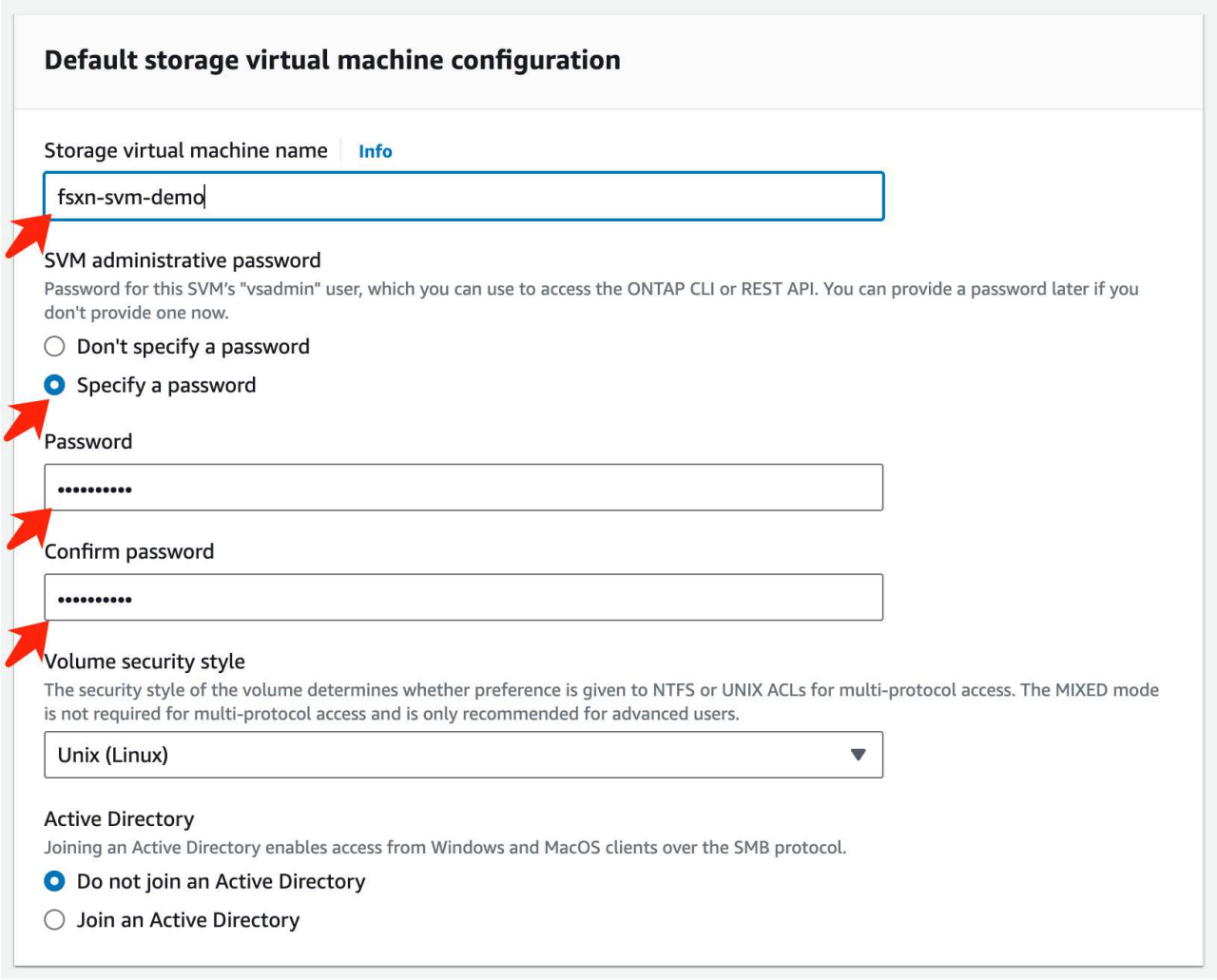
   b. 输入*文件系统名称*和* SSD存储容量*。



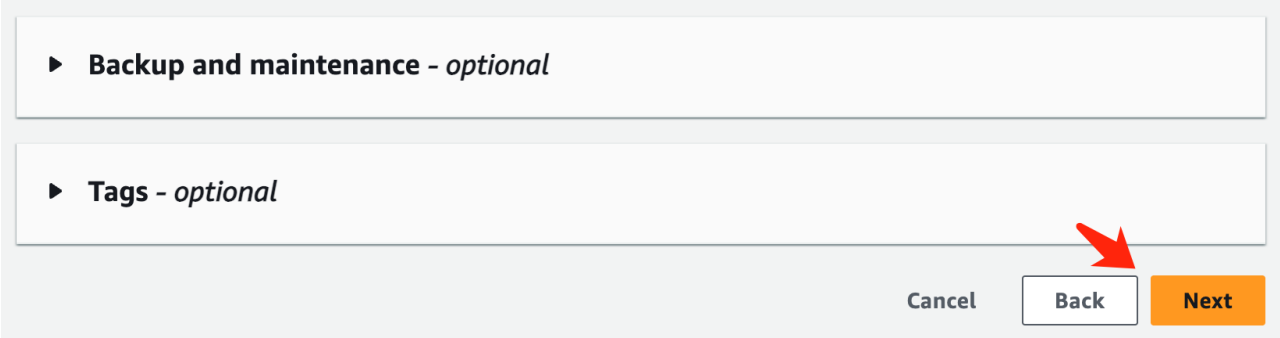
c. 确保使用与*SageMaker记事本*实例相同的*vpc*和*subnet*。



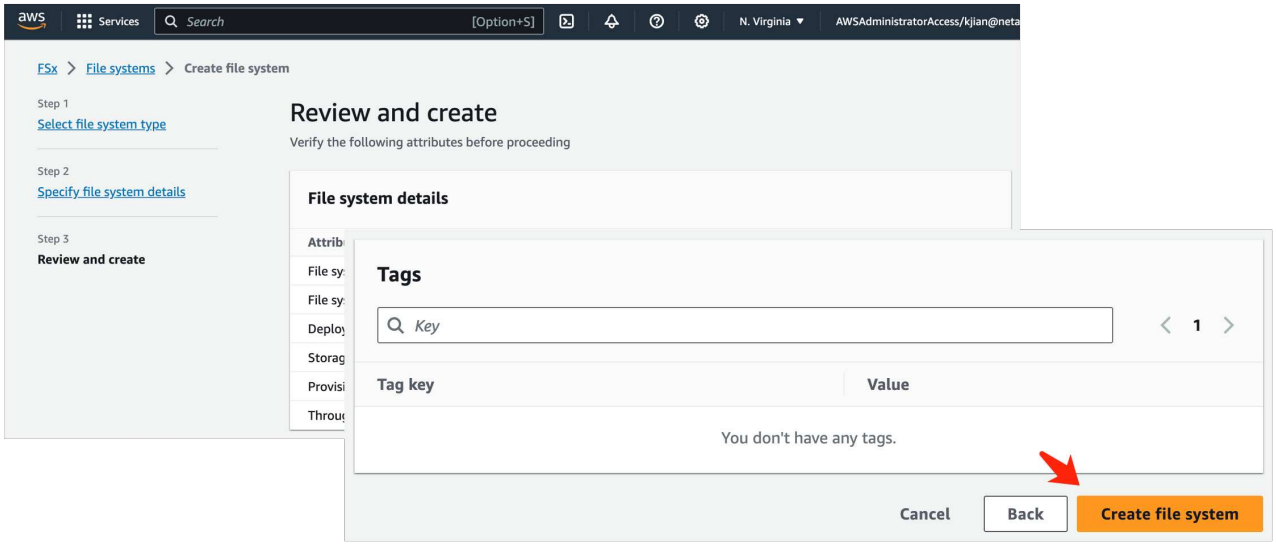
d. 输入SVM (Storage Virtual Machine)的* Storage Virtual Machine*名称和*指定密码*。



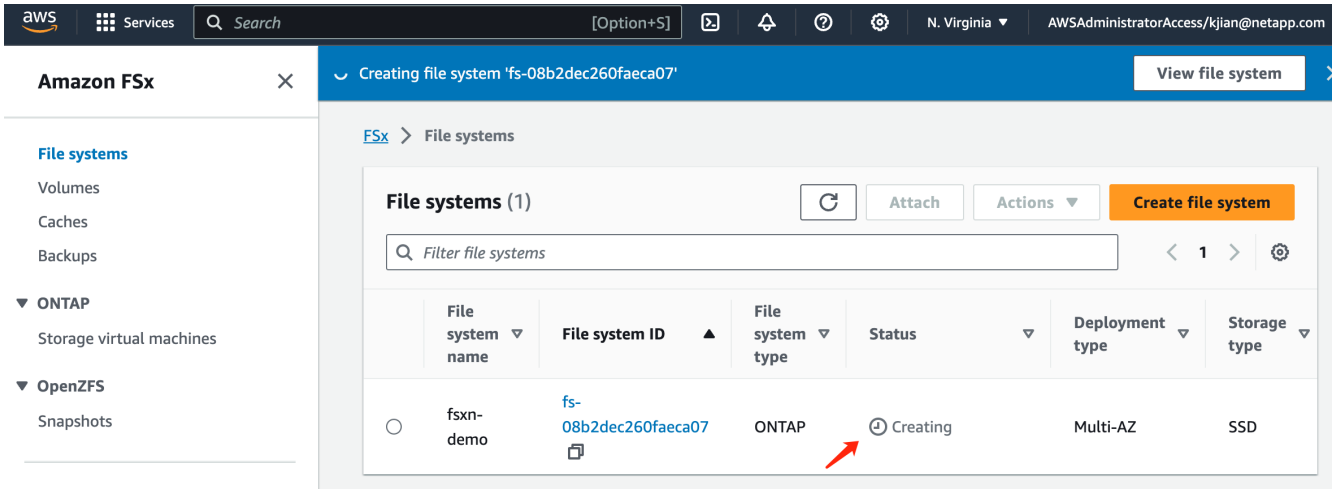
e. 保留其它条目的默认值，然后单击右下角的橙色按钮*Next*。



f. 单击查看页面右下角的橙色按钮*创建文件系统*。



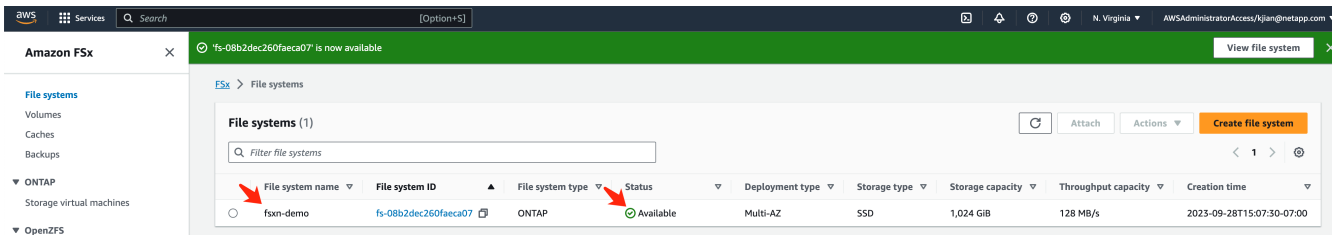
5. 启动FSx文件系统可能需要大约*20-40分钟*。



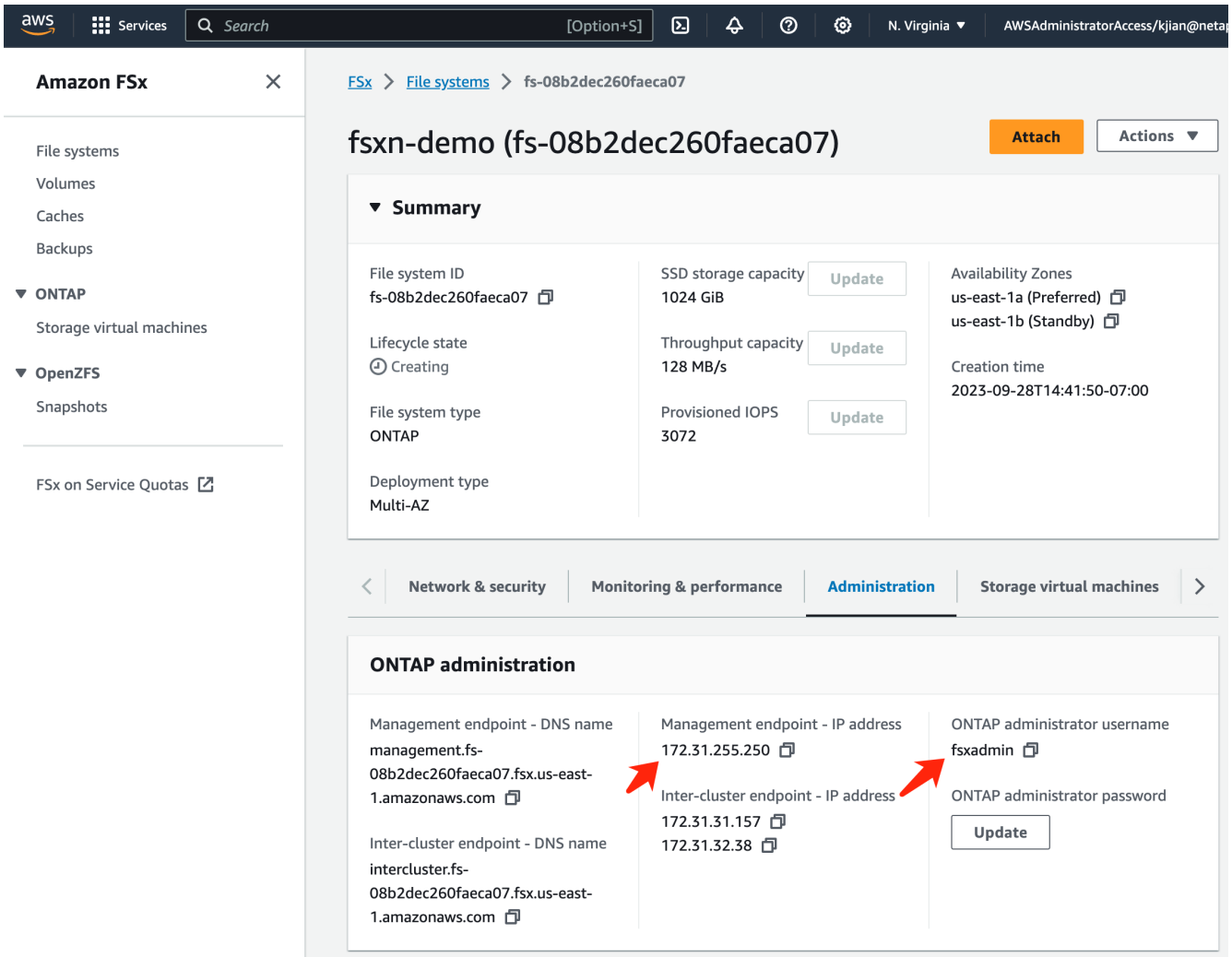
服务器配置

ONTAP配置

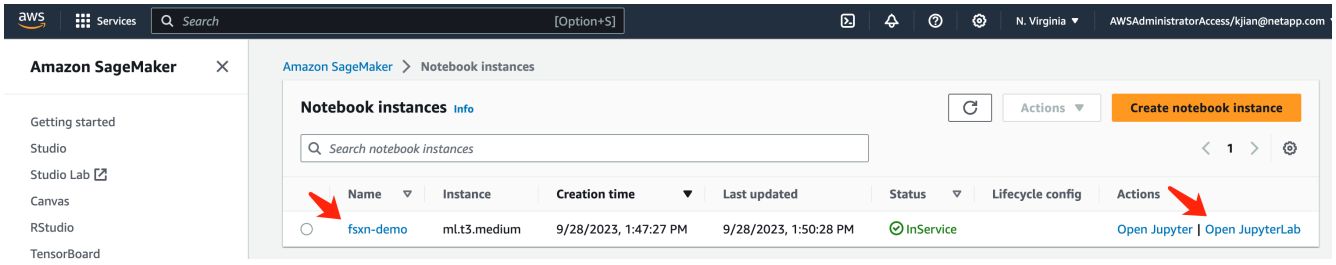
1. 打开创建的FSx文件系统。请确保状态为*可用*。



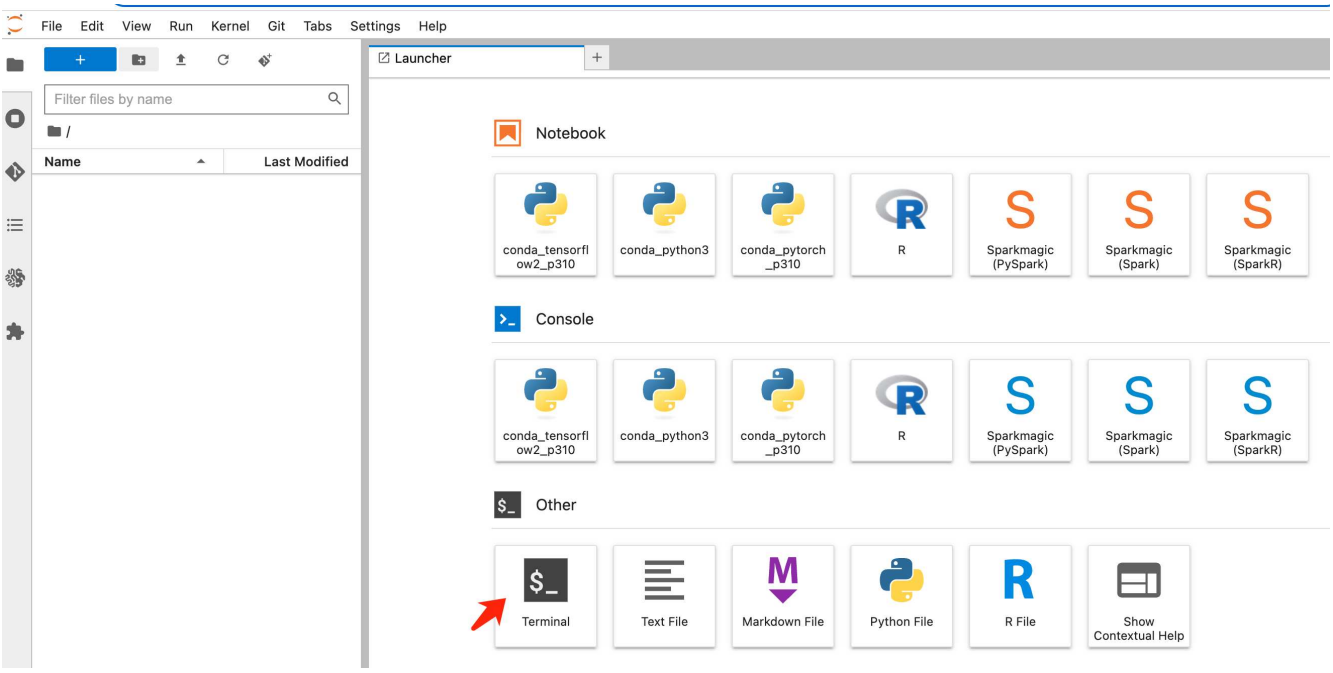
2. 选择*管理*选项卡并保留*管理端点- IP地址*和* ONTAP管理员用户名*。



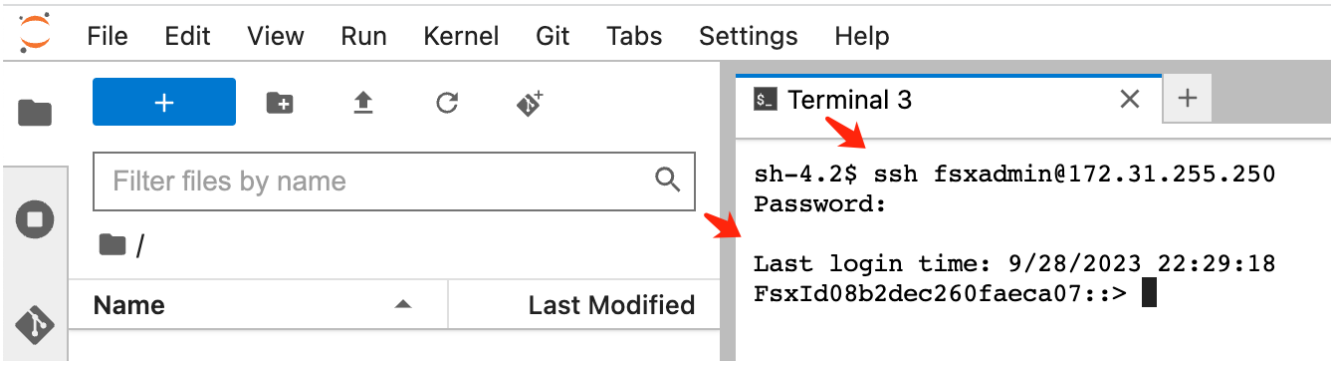
3. 打开创建的*SageMaker笔记本实例*，然后单击*Open JupyterLab*。



4. 在Jupyter Lab页面中，打开一个新的*Terminal *。

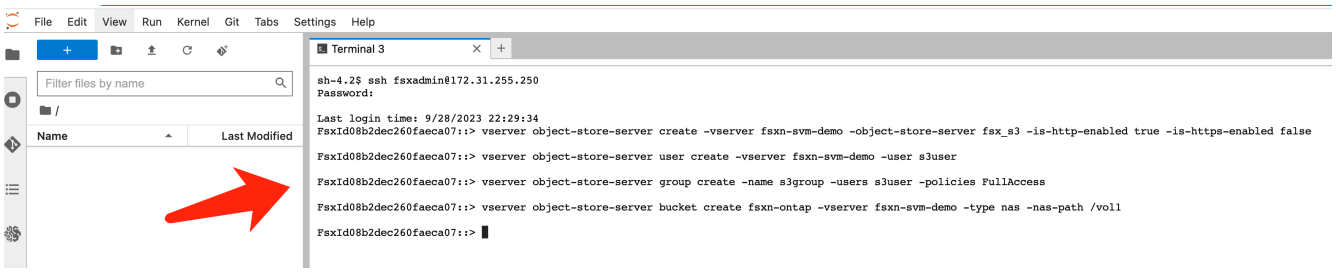


5. 输入ssh命令ssh <admin user name>@<ONTAP server IP>登录到FSxN ONTAP文件系统。(用户名和IP地址从步骤2中检索)
请使用创建*Storage Virtual Machine*时使用的密码。



6. 按以下顺序执行命令。
我们使用*fsxn-ONTAP 作为 FSxN专用S3存储分段名称*的名称。
请使用*Storage Virtual Machine name*作为*-vserver*参数。

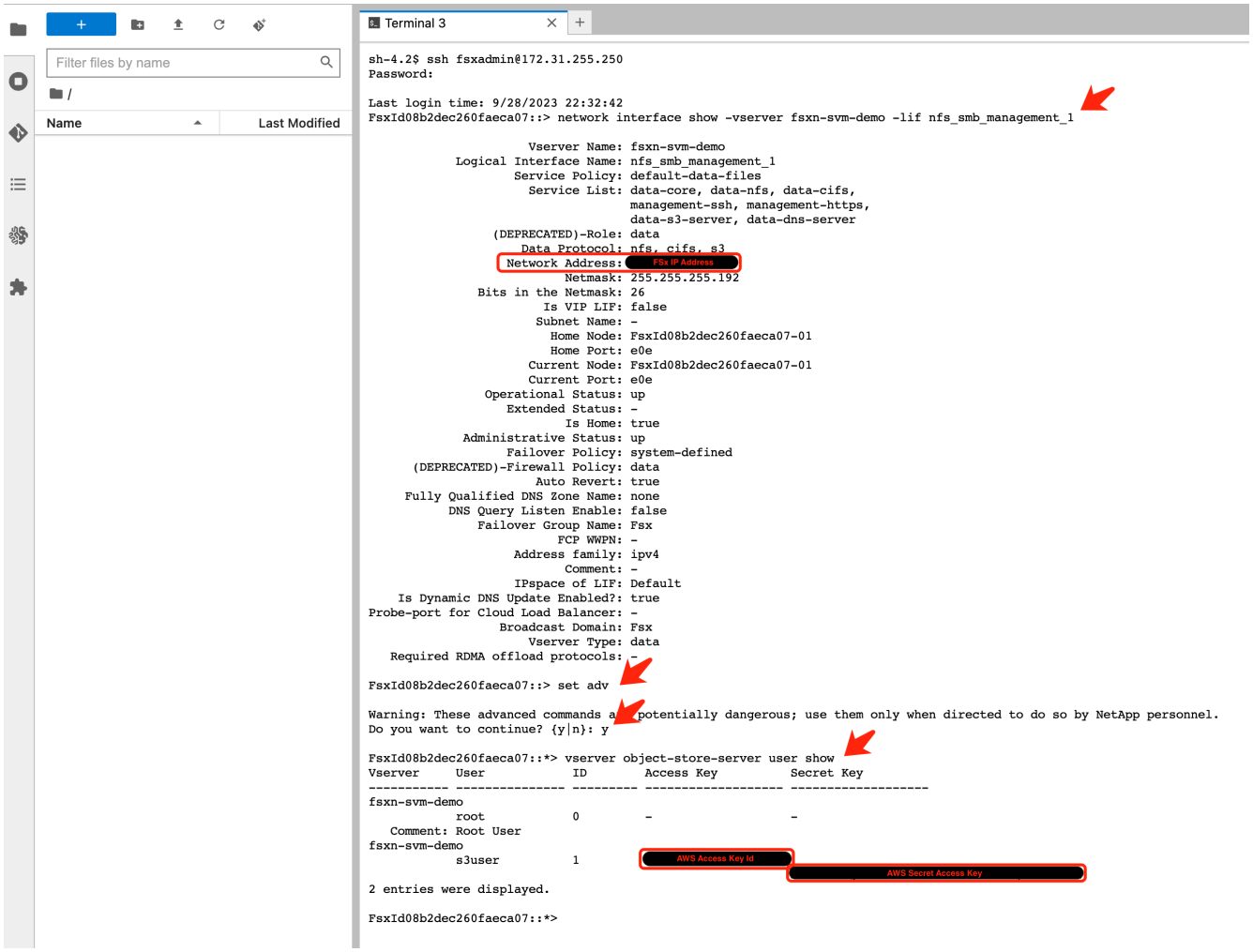
```
vserver object-store-server create -vserver fsxn-svm-demo -object-store
-server fsx_s3 -is-http-enabled true -is-https-enabled false

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. 执行以下命令以检索FSxN Private S3的端点IP和凭据。

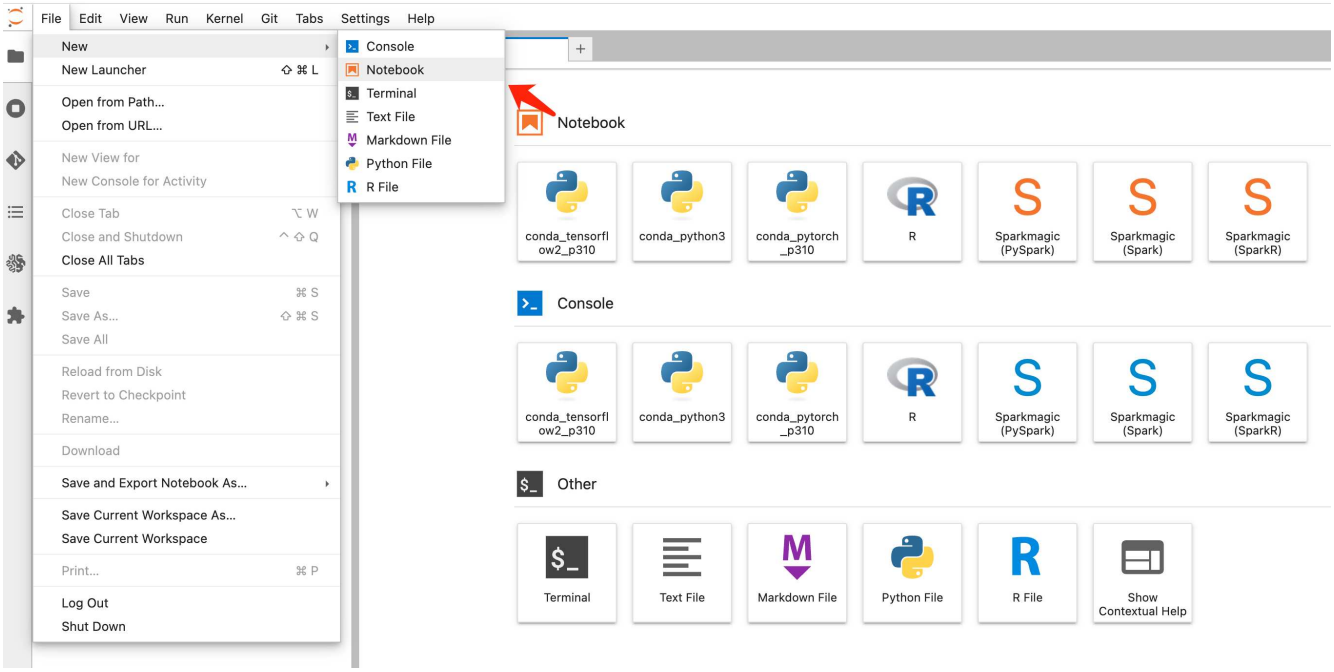
```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show
```

8. 保留端点IP和凭据以供将来使用。



客户端配置

1. 在SageMaker笔记本实例中、创建新的Jupyter笔记本。



2. 使用以下代码作为解决解决方案问题的方法、将文件上传到FSxN私有S3存储分段。
有关完整的代码示例、请参阅本笔记本。
"fsxn_dema.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']
```

FSxN与SageMaker实例之间的集成到此结束。

有用的调试检查清单

- 确保SageMaker笔记本实例和FSxN文件系统位于同一个VPC中。

- 请记得在ONTAP上运行*set dev*命令，将权限级别设置为*dev*。

常见问题解答(截至2023年9月27日)

问：为什么在将文件上传到FSxN时、我在调用CreateMultipartUpload操作时收到错误"发生错误(未实施)：您请求的S3命令未实施"？

答：作为私有S3存储分段、FSxN支持上传高达100 MB的文件。使用S3协议时、大于100 MB的文件会划分为100 MB的区块、并调用"CreateMultipartUpload"函数。但是、当前实施的FSxN Private S3不支持此功能。

问：为什么在将文件上传到FSxN时、调用PutObject操作时收到错误"发生错误(AccessDenied)：访问被拒绝"？

答：要从SageMaker笔记本实例访问FSxN私有S3存储分段、请将AWS凭据切换到FSxN凭据。但是、要为实例授予写入权限、需要使用 临时解决策 解决方案 挂载存储分段并运行"chmod" shell命令来更改权限。

问：如何将FSxN Private S3存储分段与其他SageMaker ML服务集成？

答：遗憾的是、SageMaker服务SDK无法为专用S3存储分段指定端点。因此、FSxN S3与SageMaker服务不兼容、例如、SagMaker Data Rangler、SagMaker Clarify、SagMaker Glue、SagMaker Athena、SagMaker AutoML、 等。

第2部分—利用AWS FSx for NetApp ONTAP (FSxN)作为SageMaker模型训练的数据源

作者：
Jian Jian (Ken)、NetApp高级数据和应用科学人员

简介

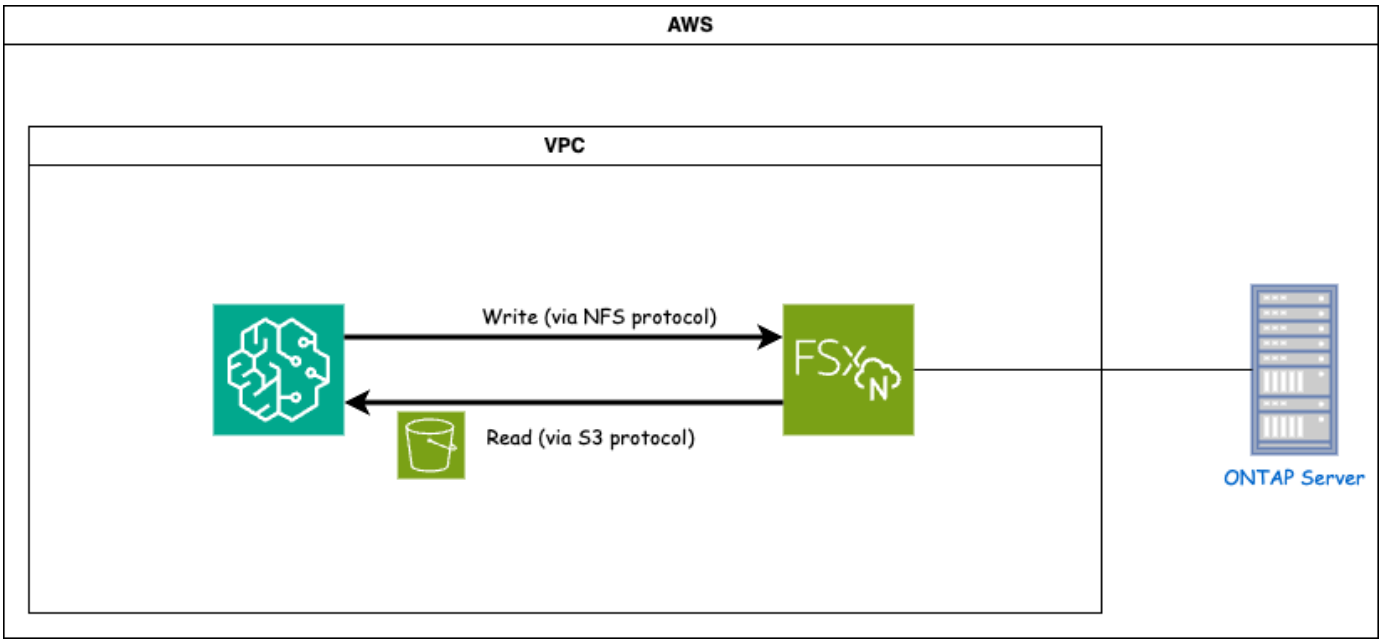
本教程提供了计算机视觉分类项目的一个实际示例、提供了在SageMaker环境中构建ML模型以利用FSxN作为数据源的实践经验。该项目侧重于使用深度学习框架PyTorch、根据轮胎图像对轮胎质量进行分类。它侧重于使用FSxN作为Amazon SageMaker中的数据源来开发机器学习模型。

什么是FSxN

Amazon FSx for NetApp ONTAP确实是AWS提供的一款完全托管的存储解决方案。它利用NetApp的ONTAP文件系统提供可靠的高性能存储。由于支持NFS、SMB和iSCSI等协议、因此可以从不同的计算实例和容器无缝访问。该服务旨在提供卓越的性能、确保快速高效的数据运营。它还提供高可用性和持久性、确保您的数据始终可访问并受到保护。此外、Amazon FSx for NetApp ONTAP的存储容量可扩展、使您可以根据需要轻松调整。

前提条件

网络环境



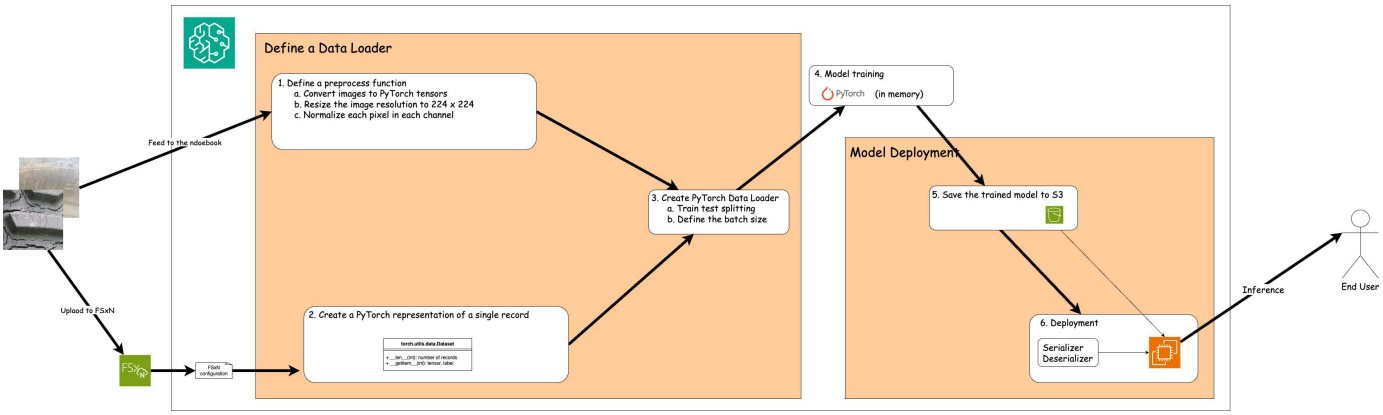
FSxN (Amazon FSx for NetApp ONTAP)是一项AWS存储服务。它包括在NetApp ONTAP系统上运行的文件系统以及与其连接的AWS托管系统虚拟机(SVM)。在提供的图中、由AWS管理的NetApp ONTAP服务器位于VPC之外。SVM充当SageMaker和NetApp ONTAP系统之间的中介、接收来自SageMaker的操作请求并将其转发到底层存储。要访问FSxN、SageMaker必须与FSxN部署位于同一个VPC中。此配置可确保SageMaker和FSxN之间的通信和数据访问。

数据访问

在实际场景中、数据科学家通常会利用存储在FSxN中的现有数据来构建机器学习模型。但是、出于演示目的、由于FSxN文件系统在创建后最初为空、因此需要手动上传训练数据。这可以通过将FSxN作为卷挂载到SageMaker来实现。成功挂载文件系统后、您可以将数据集上传到挂载位置、以便在SageMaker环境中训练模型。通过这种方法、您可以在与SageMaker合作进行模型开发和训练时利用FSxN的存储容量和功能。

数据读取过程会将FSxN配置为专用S3存储分段。要了解详细的配置说明、请参见 "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"

集成概述



使用FSxN中的训练数据在SageMaker中构建深度学习模型的工作流可概括为三个主要步骤：数据加载程序定义、模型训练和部署。总体而言、这些步骤构成了MLOps管道的基础。但是、每个步骤都涉及多个详细的子步骤、以实现全面实施。这些子步骤包括各种任务、例如数据预处理、数据集拆分、模型配置、超参数调整、模型

评估、 和型号部署。这些步骤可确保在SageMaker环境中使用来自FSxN的训练数据构建和部署深度学习模型的流程全面有效。

分步集成

数据加载程序

为了训练使用数据的PyTorch深度学习网络、我们创建了一个数据加载程序来促进数据馈送。数据加载程序不仅可以定义批大小、还可以确定用于读取和预处理批处理中每个记录的操作步骤。通过配置数据加载程序、我们可以处理批量数据处理、从而实现深度学习网络的训练。

数据加载程序由3个部分组成。

预处理功能

```
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]
    )
])
```

上述代码段演示了使用*torchVISION．transforms*模块的图像预处理转换的定义。在此图示中、创建预处理对象以应用一系列转换。首先，*ToTendor()*转换将图像转换为张图表示。随后，*Resize　224224*转换将图像的大小调整为固定的224x224像素大小。最后，*NORMDE()*转换通过减去平均值并除以沿每个通道的标准偏差来使张量值标准化。用于标准化的平均值和标准偏差值通常用于经过预先训练的神经网络模型。总之、该代码通过将图像数据转换为张量、调整图像大小和使像素值标准化来准备图像数据、以便进一步处理或输入到预先训练的模型中。

PyTorch数据集类

```
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
```

此类提供了获取数据集中记录总数的功能，并定义了读取每个记录的数据的方法。在*_gottim*函数中，代码利用boto3 S3存储分段对象从FSxN中检索二进制数据。从FSxN访问数据的代码模式类似于从Amazon S3读取数据。后面的说明将深入介绍私有S3对象*bket*的创建过程。

FSxN作为私有S3存储库

```
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 ---
```

要从SageMaker中的FSxN读取数据、需要创建一个处理程序、该处理程序使用S3协议指向FSxN存储。这样就可以将FSxN视为专用S3存储分段。处理程序配置包括指定FSxN SVM的IP地址、分段名称和所需凭据。有关获取这些配置项的完整说明、请参阅上的文档 "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"。

在上述示例中、b分 段对象用于实例化PyTorch DataSet对象。数据集对象将在后续章节中进一步说明。

PyTorch数据加载程序

```
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)
```

在提供的示例中、指定的批大小为64、表示每个批将包含64条记录。通过将PyTorch *DataT*类、预处理功能和训练批大小相结合，我们可以获得训练所需的数据加载程序。此数据加载程序有助于在训练阶段批量迭代数据集。

模型训练

```
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')
```

本规范实施标准的PyTorch培训流程。它定义了一个名为*TireQualityClassifyer*的神经网络模型，该模型使用卷积层和线性层对轮胎质量进行分类。训练循环会迭代数据批处理、并使用反向传播和优化功能来确定损失、然后更新模型的参数。此外、它还会打印当前时间、时期、批处理和损失、以供监控。

模型部署

部署

```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())
```

此代码会将PyTorch模型保存到*Amazon S3*中，因为SageMaker要求将模型存储在S3中进行部署。通过将模型上传到*Amazon S3*，SageMaker便可访问模型，从而可以在已部署的模型上进行部署和引用。

```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
)
```

此代码有助于在SageMaker上部署PyTorch模型。它定义了一个自定义的串口器*TireQuality串 口器*，该串口器可将输入数据作为PyTorch张量进行预处理和串口处理。TireQuality谓 词*类是一个自定义的预测程序，它利用定义的串列器和*JSONDeseriiter*。该代码还会创建一个*PyTorchModel*对象，用于指定模型的S3位置、IAM角色、框架版本和引用入口点。代码会生成时间戳并根据模型和时间戳构建端点名称。最后、使用Deploy方法部署模型、并指定实例计数、实例类型和生成的端点名称。这样、可以在SageMaker上部署PyTorch模型并可用于进行推入。

```
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)
```

这是使用已部署端点执行此假定的示例。

第3部分-构建简化的MLOps管道(CI/CT/CD)

作者：
Jian Jian (Ken)、NetApp高级数据和应用科学人员

简介

在本教程中、您将了解如何利用各种AWS服务构建一个简单的MLOps管道、其中包括持续集成(CI)、持续培训(CT)和持续部署(CD)。与传统DevOps管道不同、MLOps需要额外的注意事项才能完成运营周期。通过学习本教程、您将深入了解如何将CT整合到MLOps循环中、从而可以持续训练您的模型并无缝部署数据进行推导。本教程将指导您完成利用AWS服务建立此端到端MLOps管道的过程。

清单文件

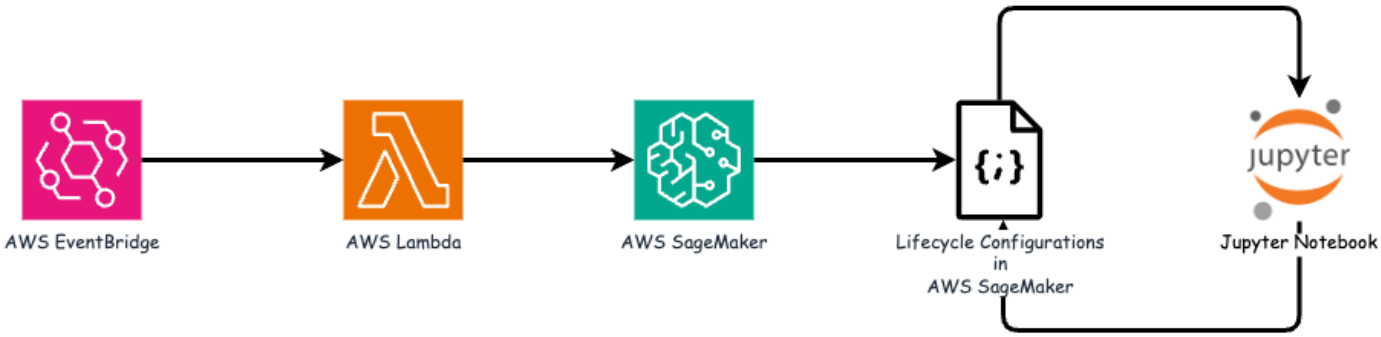
功能	Name	comment
数据存储	AWS FSxN	请参见 "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"。
数据科学IDE	AWS SageMaker	本教程基于中提供的Jupyter笔记本"第2部分—利用AWS FSx for NetApp ONTAP (FSxN)作为SageMaker模型训练的数据源"。
用于触发MLOps管道的功能	AWS Lamb开发 函数	-
cron作业触发器	AWS EventBridge	-
深度学习框架	PyTorch	-
AWS Python SDK	僵尸3	-
编程语言	Python	v3.10

前提条件

- 一种预配置的FSxN文件系统。本教程将利用存储在FSxN中的数据进行培训。

- 一个* SageMaker笔记本电脑实例*，该实例配置为与上述FSxN文件系统共享同一个VPC。
- 在触发*AWS Lambd*函数之前，请确保*SageMaker笔记本实例*处于*STOPPED*状态。
- 要利用深度神经网络的必要GPU加速、需要使用*毫升g4dn.x大*实例类型。

架构



此MLOps管道是一种实际实施、它利用cron作业触发无服务器功能、进而执行使用生命周期回调函数注册的AWS服务。AWS EventBridge*用作cron作业。它会定期调用一个*AWS Lambad*函数，负责对模型进行重新培训和重新部署。此过程涉及到启动*AWS SageMaker笔记本*实例以执行必要的任务。

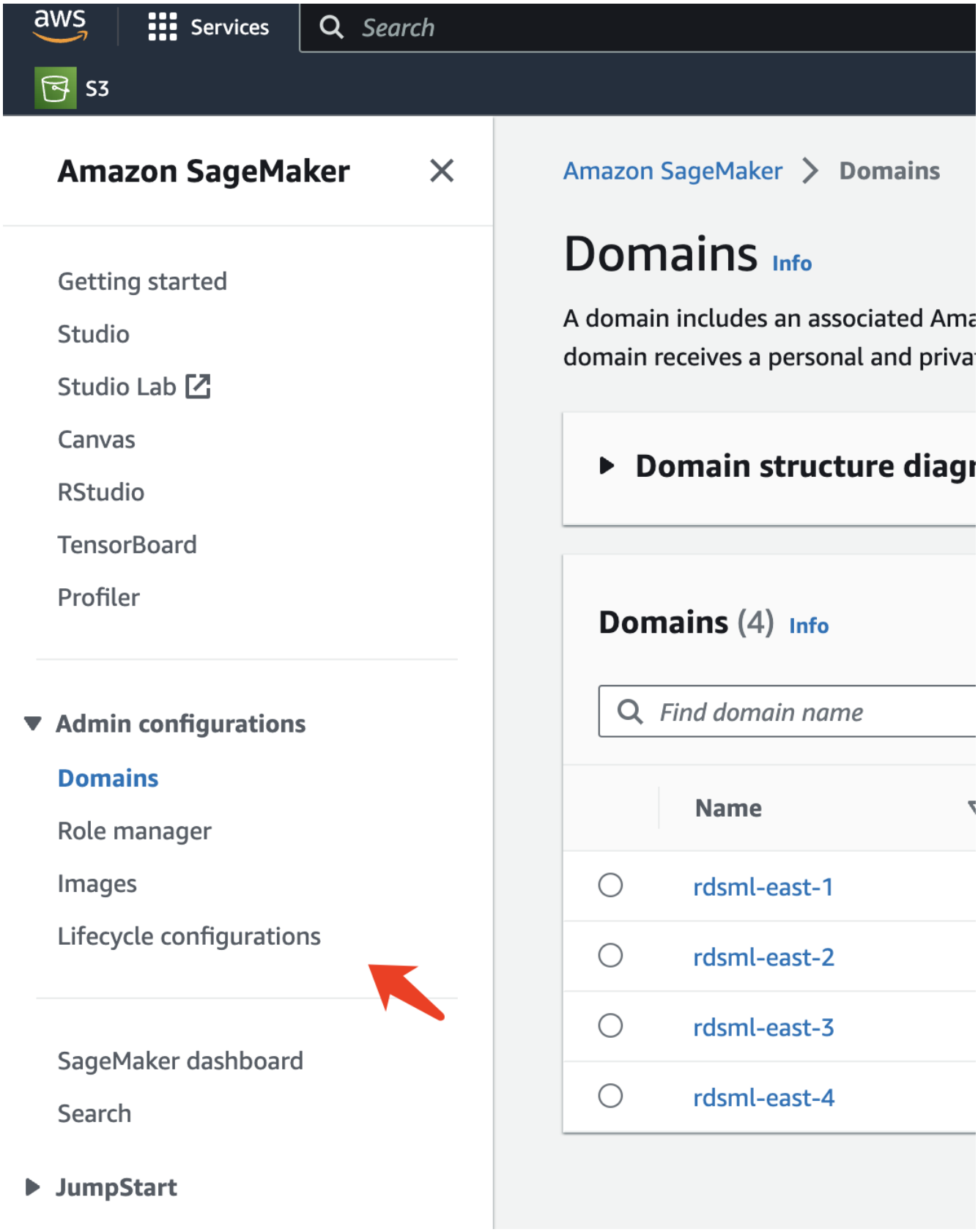
逐步配置

生命周期配置

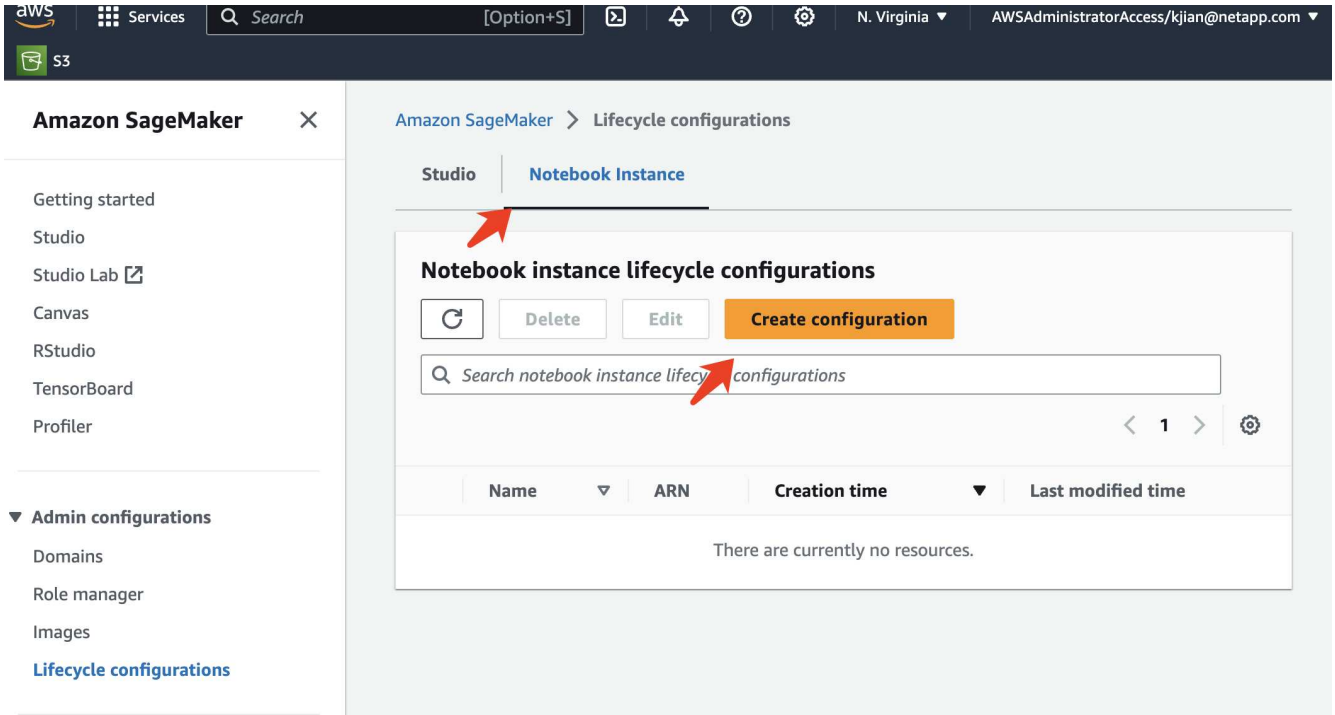
要为AWS SageMaker笔记本实例配置生命周期回调函数，应使用*Lifecycle configurations*。通过此服务，您可以定义在启动笔记本实例期间要执行的必要操作。具体而言，可以在*Lifecycle configuration*中实施shell脚本，以便在完成培训和部署过程后自动关闭笔记本实例。这是必需的配置、因为成本是MLOps中的主要考虑因素之一。

需要注意的是，需要提前设置*生命周期配置*的配置。因此、建议在继续其他MLOps管道设置之前、优先配置此方面。

1. 要设置生命周期配置，请打开*Sager*面板，然后导航到*Admin configurations*部分下的*Lifecycle configurations*。



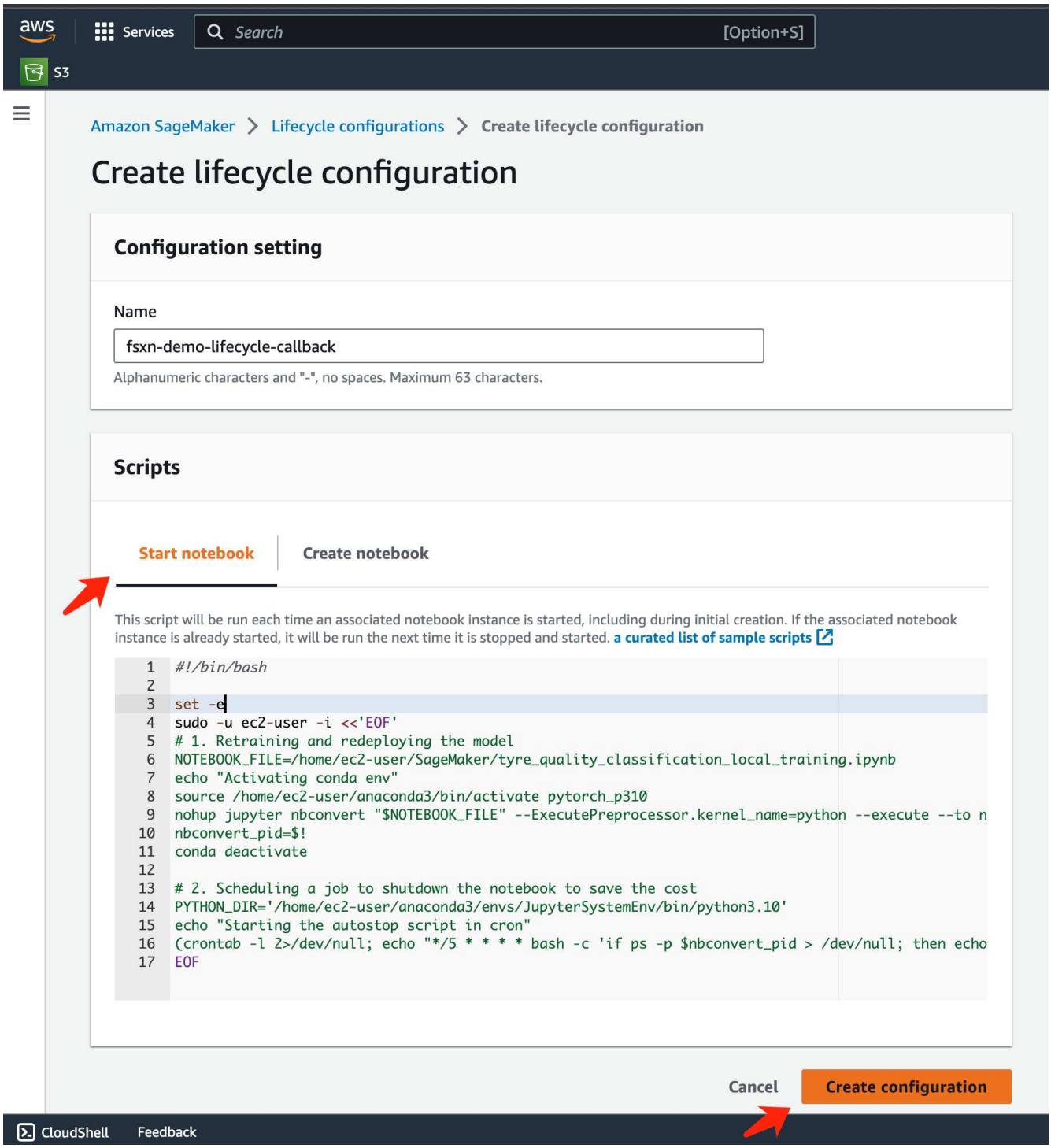
2. 选择*笔记本实例*选项卡，然后单击*创建配置*按钮



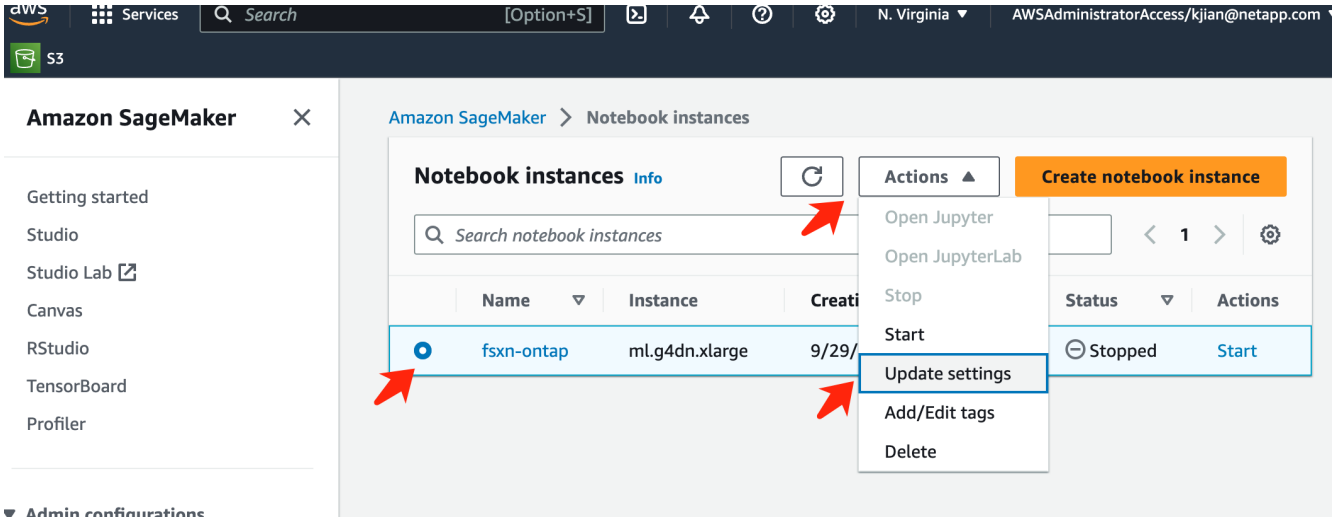
3. 将以下代码粘贴到输入区域。

```
#!/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
```

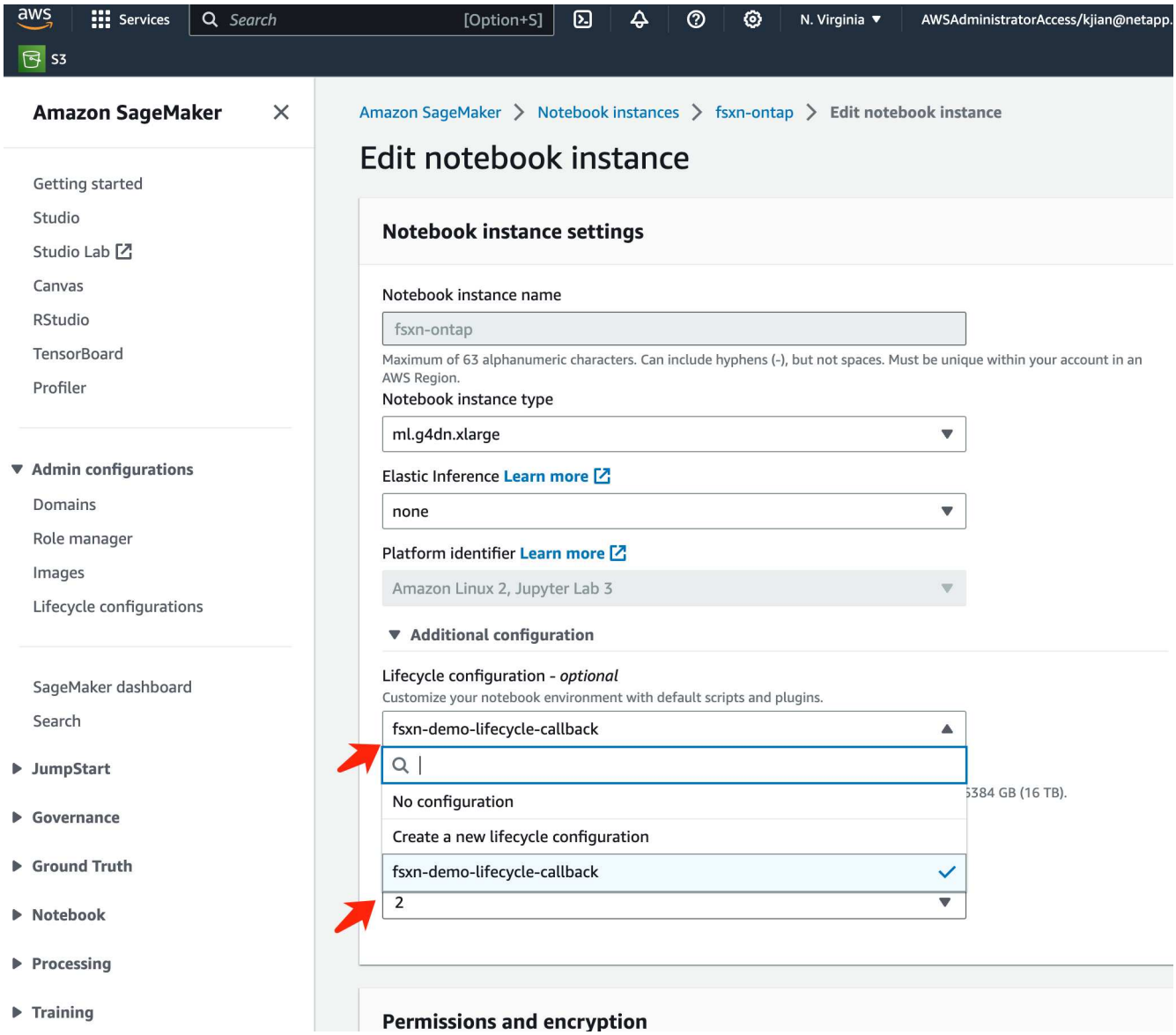
4. 此脚本执行Jupyter笔记本、该笔记本负责重新训练和重新部署模型以进行引用。执行完成后、笔记本电脑将在5分钟内自动关闭。要了解有关问题陈述和代码实施的更多信息、请参见 "第2部分—利用AWS FSx for NetApp ONTAP (FSxN)作为SageMaker模型训练的数据源"。



5. 创建后，导航到“笔记本实例”，选择目标实例，然后单击“操作”下拉列表中的*更新设置*。



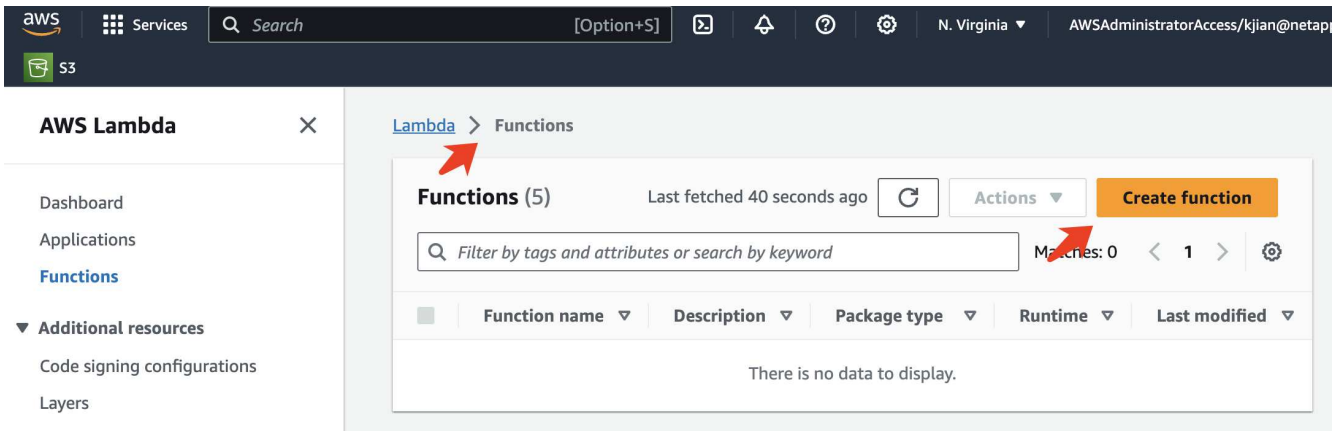
6. 选择已创建的*生命周期配置*，然后单击*更新笔记本实例*。



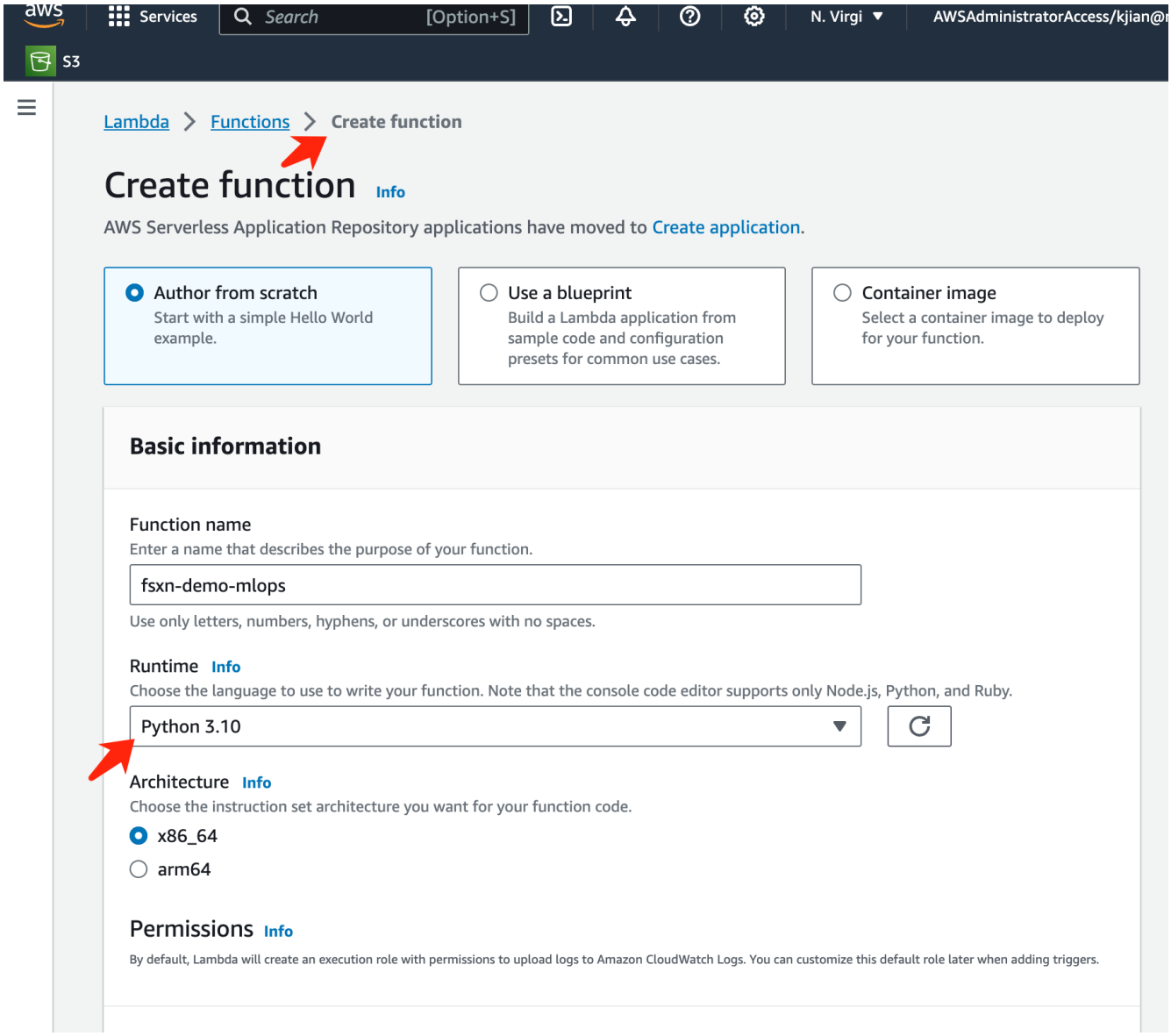
AWS Lamb达 无服务器函数

如前所述，**AWS Lambd*功能负责启动*AWS SageMaker**笔记本实例。

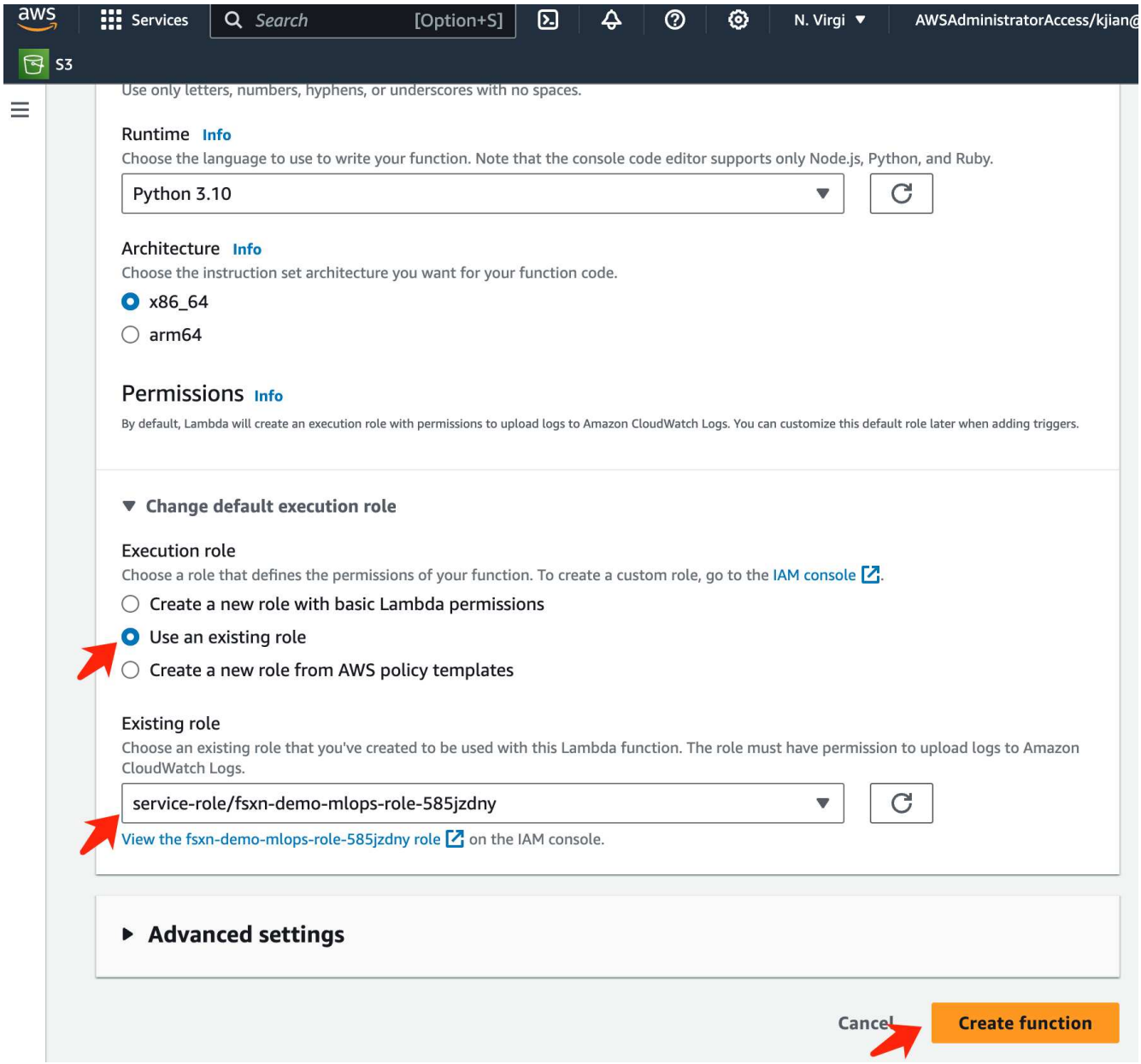
1. 要创建*AWS Lamba函数*，请导航到相应的面板，切换到*FUNCHUDENTS*选项卡，然后单击*Create FUNCHUD*。



2. 请将页面上所有必需的条目归档，并记住将运行时切换到*Python 3.10*。



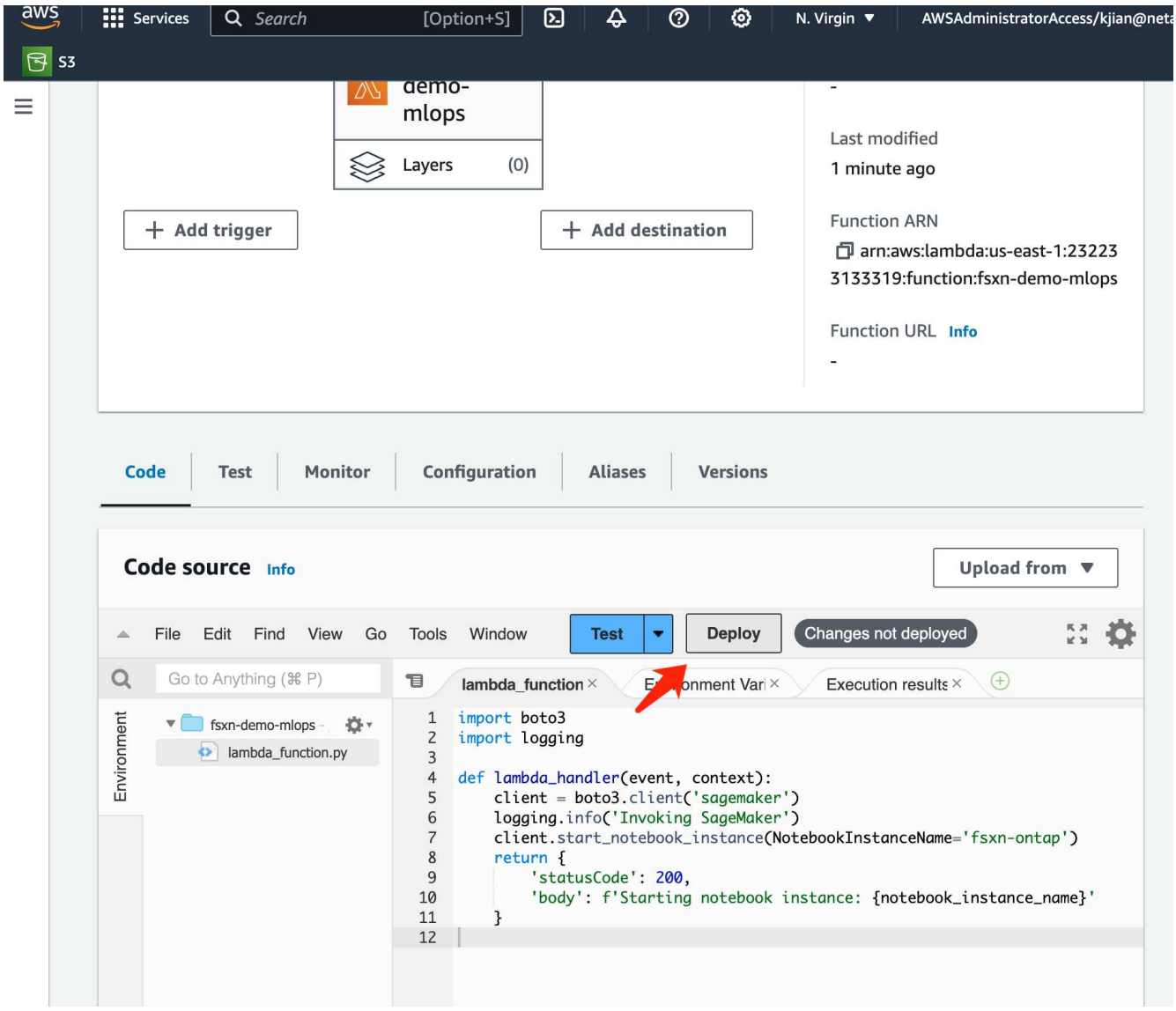
3. 请验证指定角色是否具有所需的权限*Amazon SageMakerFullAccess*，然后单击*Create Function (创建功能)*按钮。



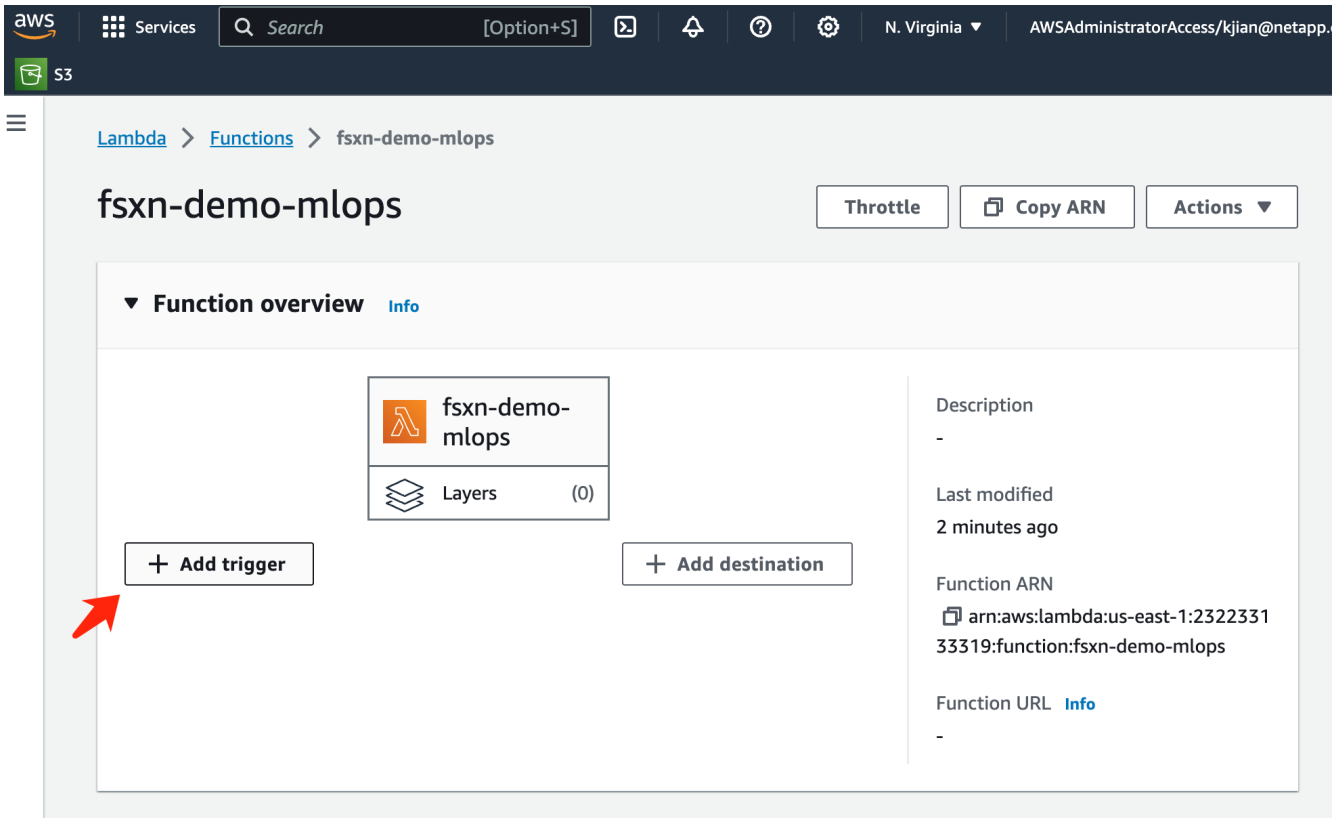
4. 选择创建的Lamb编制 函数。在代码选项卡中、将以下代码复制并粘贴到文本区域中。此代码将启动名为*fsxn-ONTAP的笔记本实例。

```
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}'
    }
```

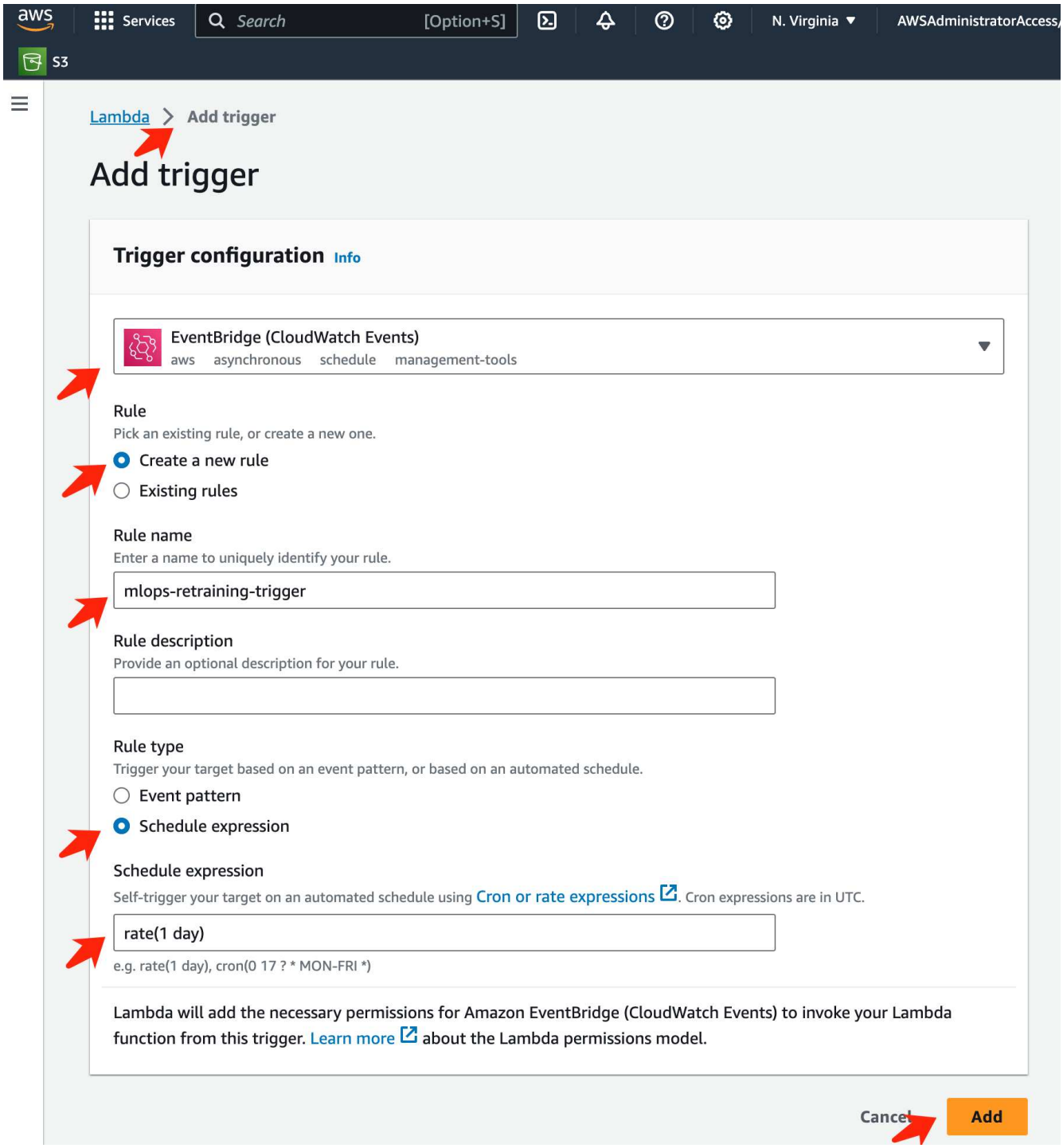
5. 单击*DEPLE*按钮以应用此代码更改。



6. 要指定如何触发此AWS Lambar函数、请单击添加触发器按钮。



7. 从下拉菜单中选择EventBridge、然后单击标有创建新规则的单选按钮。在计划表达式字段中、输入 `rate(1 day)`，然后单击添加按钮以创建此新的cron作业规则并将其应用于AWS Lamb另 一个函数。



每天完成两步配置后，**AWS Lambd***功能将启动***SageMaker**笔记本，使用*FSxN*存储库中的数据执行模型重新训练，将更新的模型重新部署到生产环境，并自动关闭*SageMaker笔记本实例*以优化成本。这可确保模型保持最新。

开发MLOps管道的教程到此结束。