适用于 MLOps 的 FSx ONTAP

NetApp artificial intelligence solutions

NetApp
February 12, 2026

目录

适用于 MLOps 的 FSx ONTAP . . . 1
适用于 MLOps 的Amazon FSx for NetApp ONTAP (FSx ONTAP) . . . 1
第 1 部分 - 将Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为私有 S3 存储桶集成到 AWS SageMaker . . . 1
简介 . . . 1
用户指南 . . . 1
有用的调试清单 . . . 14
常见问题解答（截至 2023 年 9 月 27 日） . . . 15
第 2 部分 - 利用 AWS Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为 SageMaker 模型训练的数据源 . . . 15
简介 . . . 15
什么是 FSx ONTAP . . . 15
前提条件 . . . 15
集成概述 . . . 16
逐步集成 . . . 17
第 3 部分 - 构建简化的 MLOps 管道 (CI/CT/CD) . . . 24
简介 . . . 24
显现 . . . 24
前提条件 . . . 24
架构 . . . 25
分步配置 . . . 25

适用于 MLOps 的 FSx ONTAP

适用于 MLOps 的Amazon FSx for NetApp ONTAP (FSx ONTAP)

本节深入探讨 AI 基础架构开发的实际应用，提供使用 FSx ONTAP构建 MLOps 管道的端到端演练。它包含三个综合示例，指导您通过这个强大的数据管理平台满足您的 MLOps 需求。

这些文章重点关注：

1. "第 1 部分 - 将Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为私有 S3 存储桶集成到 AWS SageMaker"
2. "第 2 部分 - 利用Amazon FSx for NetApp ONTAP (FSx ONTAP) 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)"

到本节结束时，您将对如何使用 FSx ONTAP简化 MLOps 流程有深入的了解。

第 1 部分 - 将Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为私有 S3 存储桶集成到 AWS SageMaker

本节提供了使用 AWS SageMaker 将 FSx ONTAP配置为私有 S3 存储桶的指南。

简介

以 SageMaker 为例，本页面提供了将 FSx ONTAP配置为私有 S3 存储桶的指导。

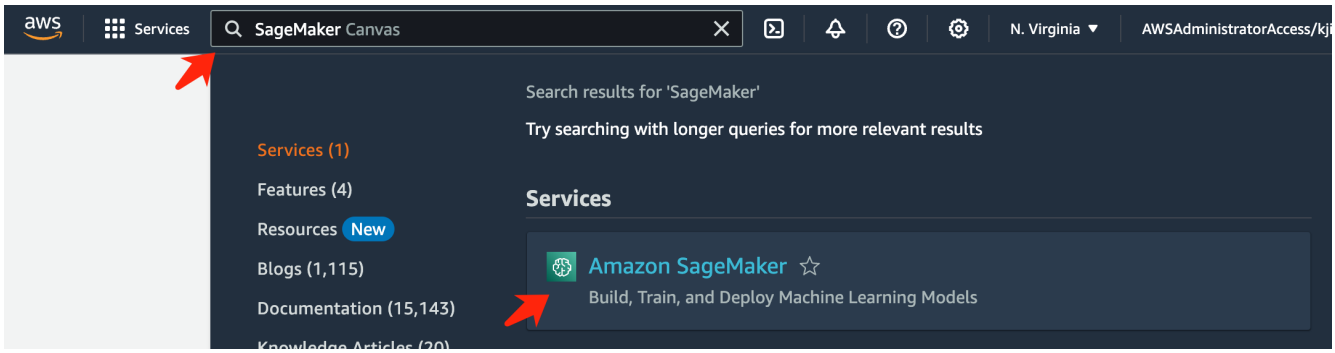
有关 FSx ONTAP的更多信息，请查看此演示文稿（"视频链接"）

用户指南

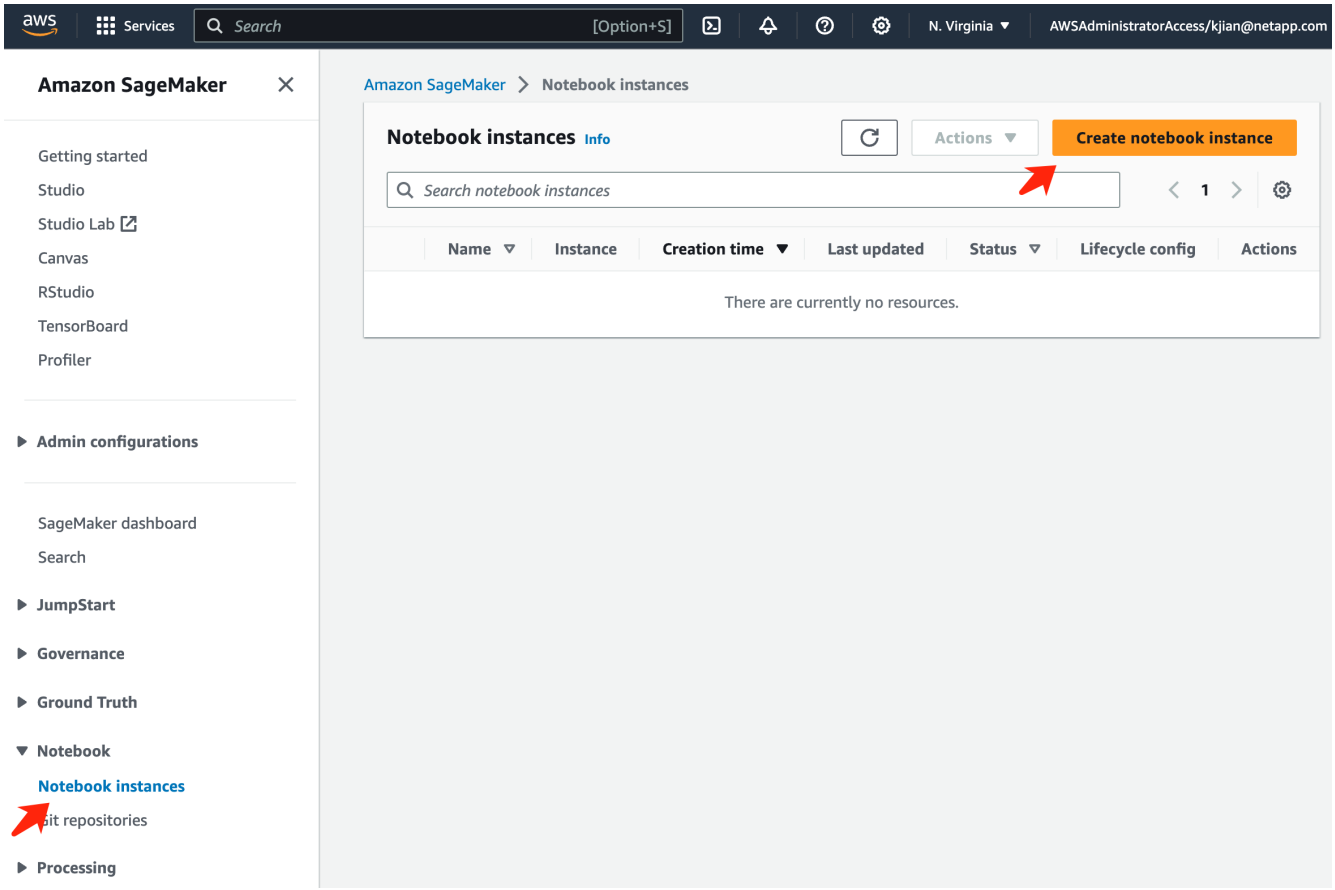
服务器创建

创建 **SageMaker Notebook** 实例

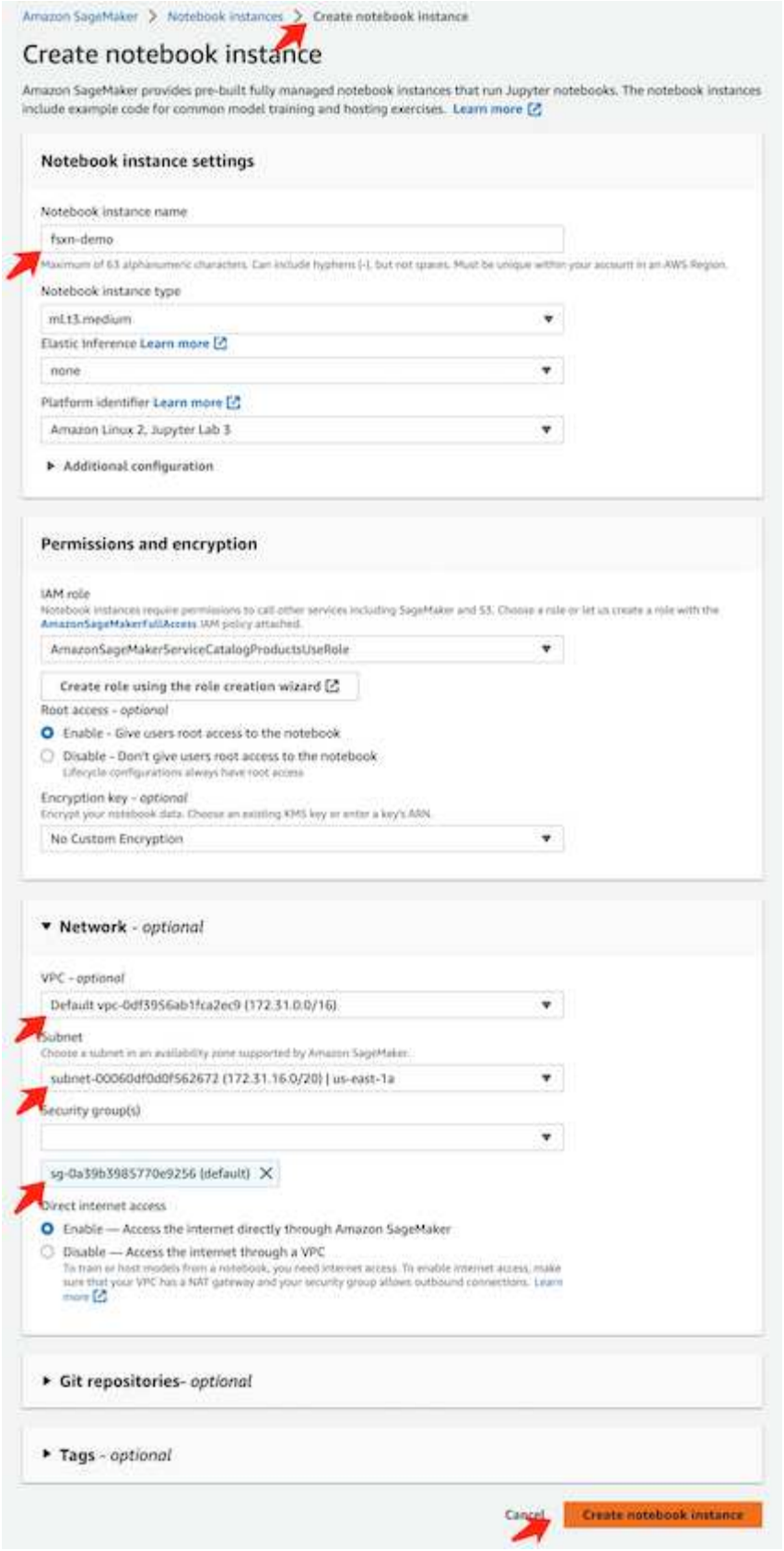
1. 打开 AWS 控制台。在搜索面板中，搜索 SageMaker 并单击服务 **Amazon SageMaker**。



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

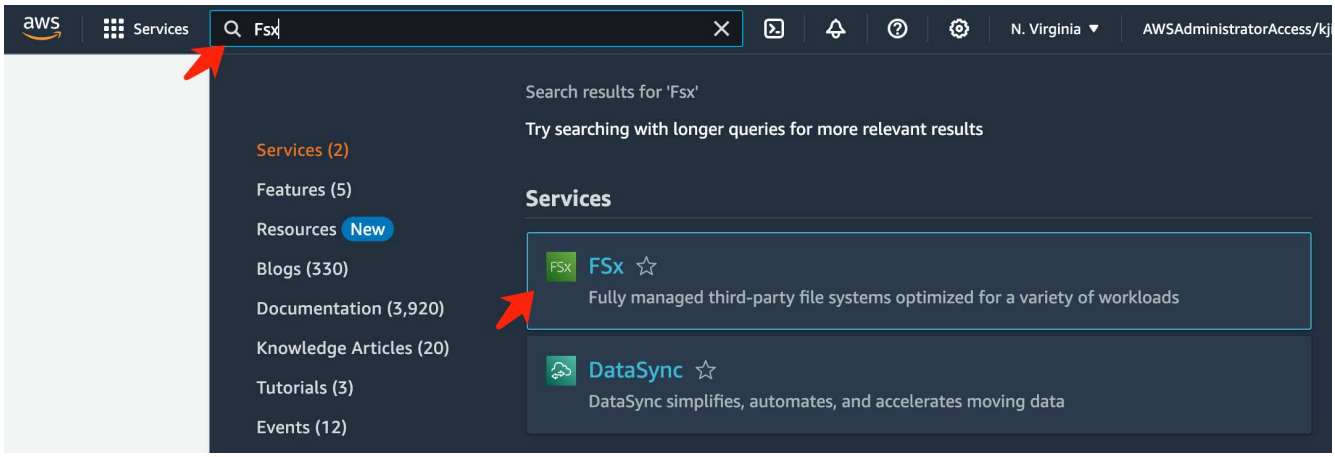


3. 在创建页面中，输入*笔记本实例名称*展开*网络*面板保留其他条目的默认值，并选择*VPC*、子网*和*安全组。 （此*VPC*和*Subnet*稍后将用于创建 FSx ONTAP文件系统）单击右下角的橙色按钮*创建笔记本实例*。

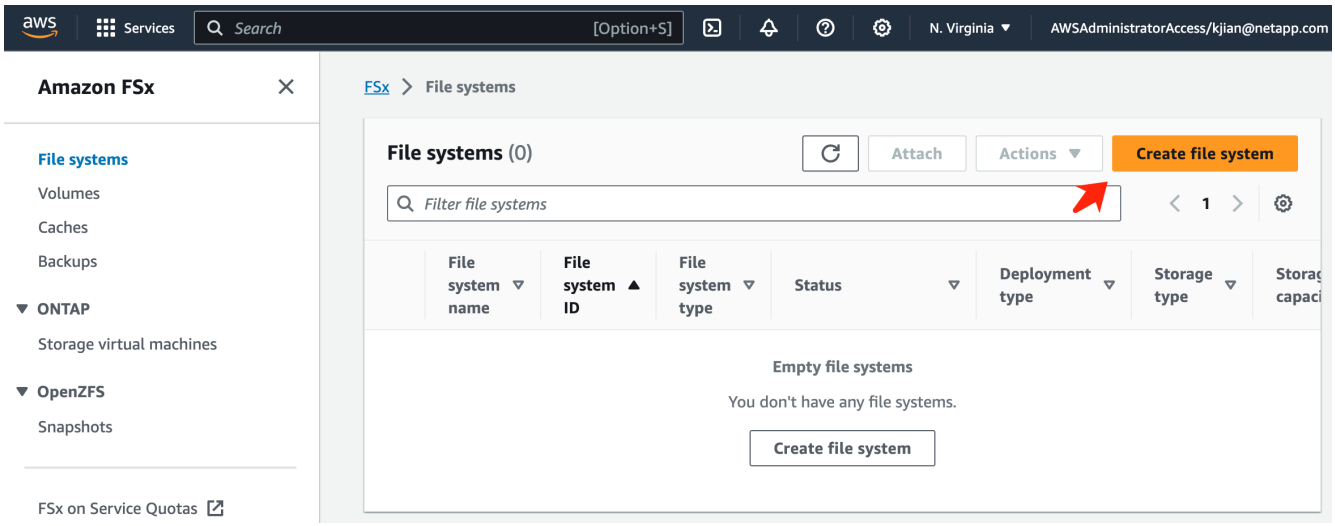


创建 **FSx ONTAP**文件系统

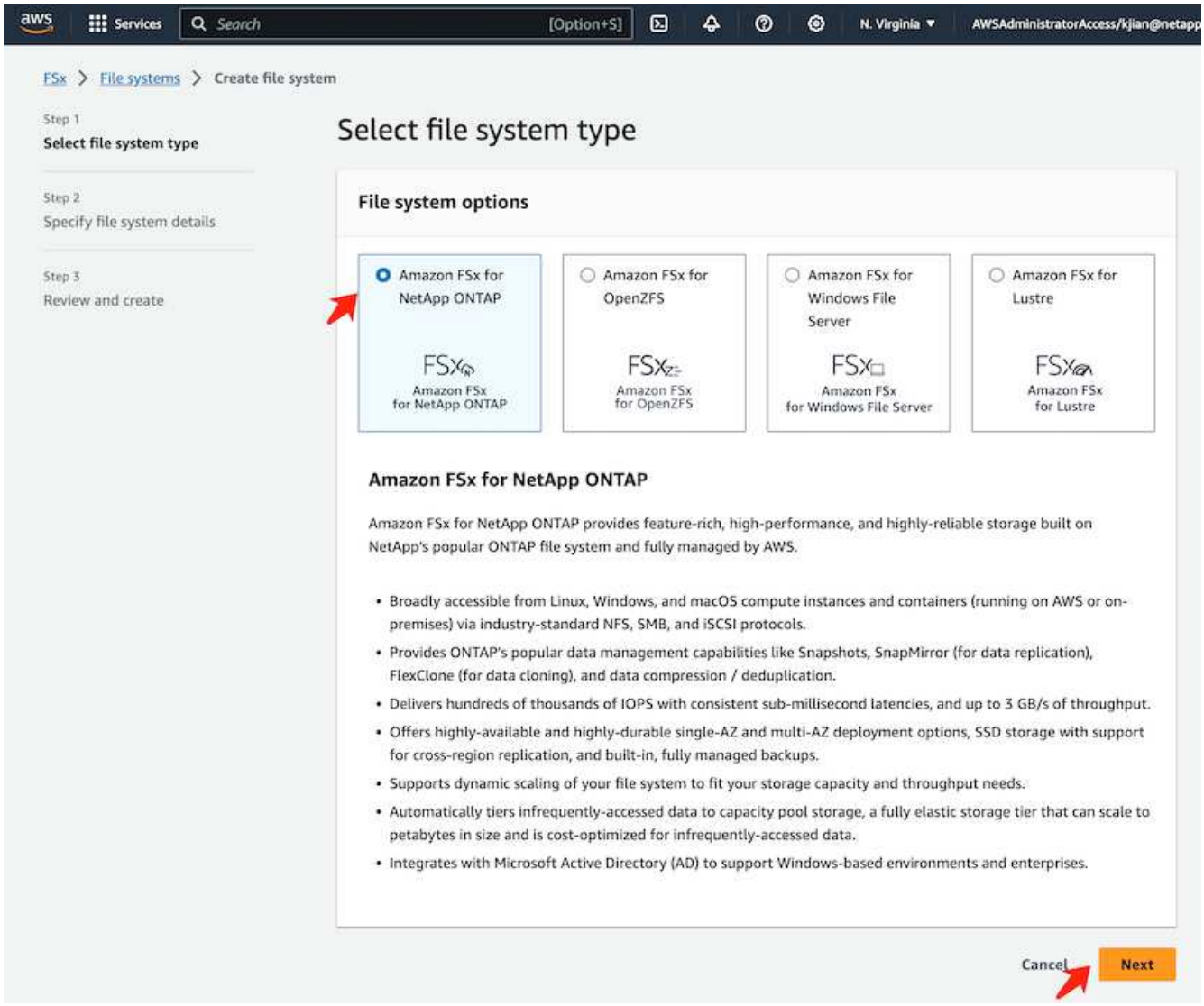
1. 打开 AWS 控制台。在搜索面板中，搜索 Fsx 并单击服务 **FSx**。



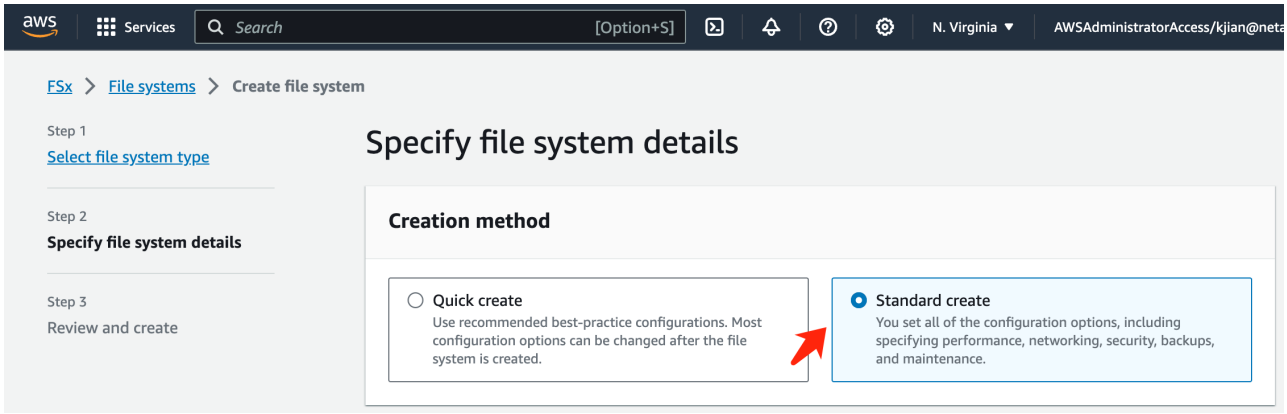
2. 单击*创建文件系统*。



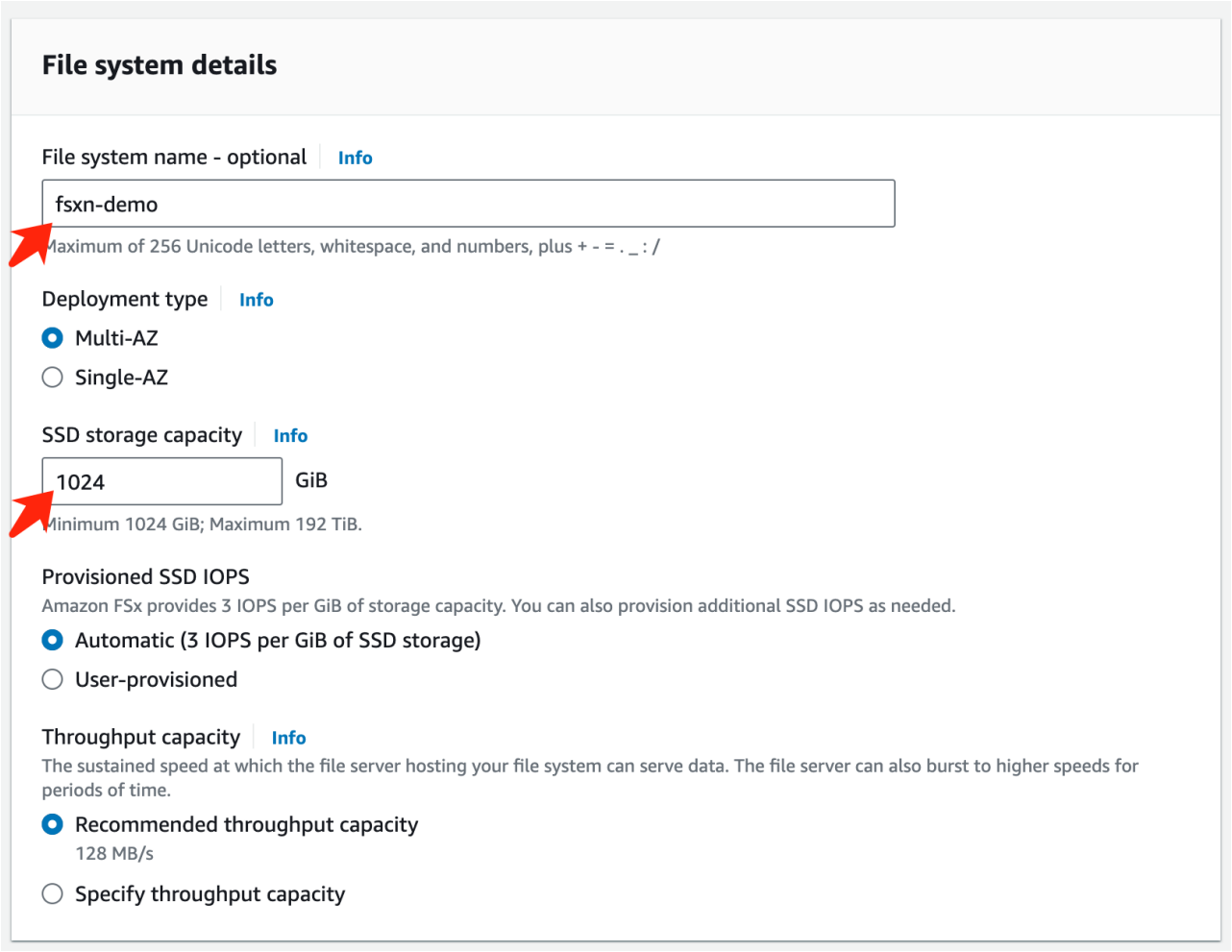
3. 选择第一张卡 **FSx ONTAP** 并单击 下一步。



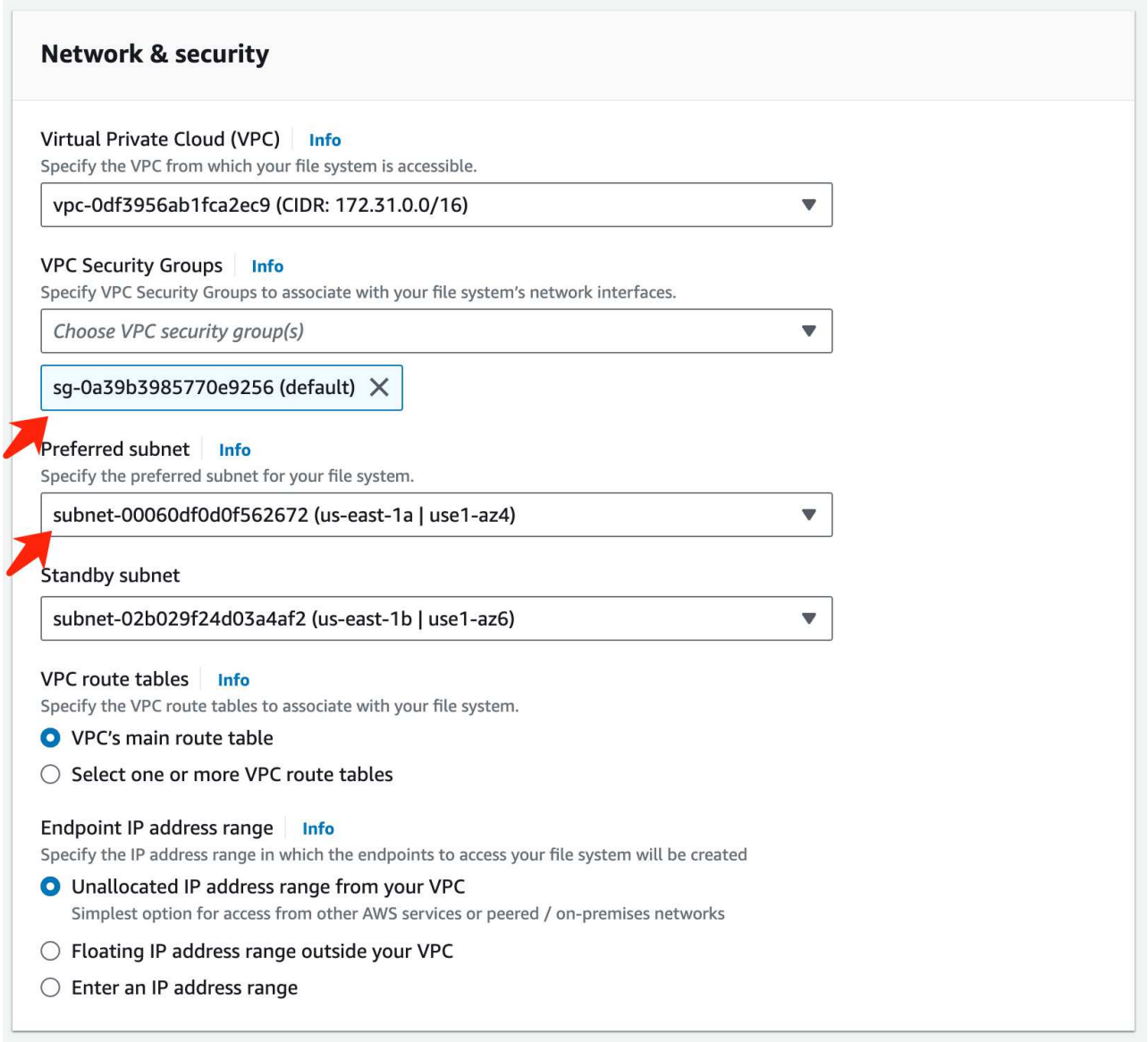
4. 在详细信息配置页面中。
   a. 选择*标准创建*选项。



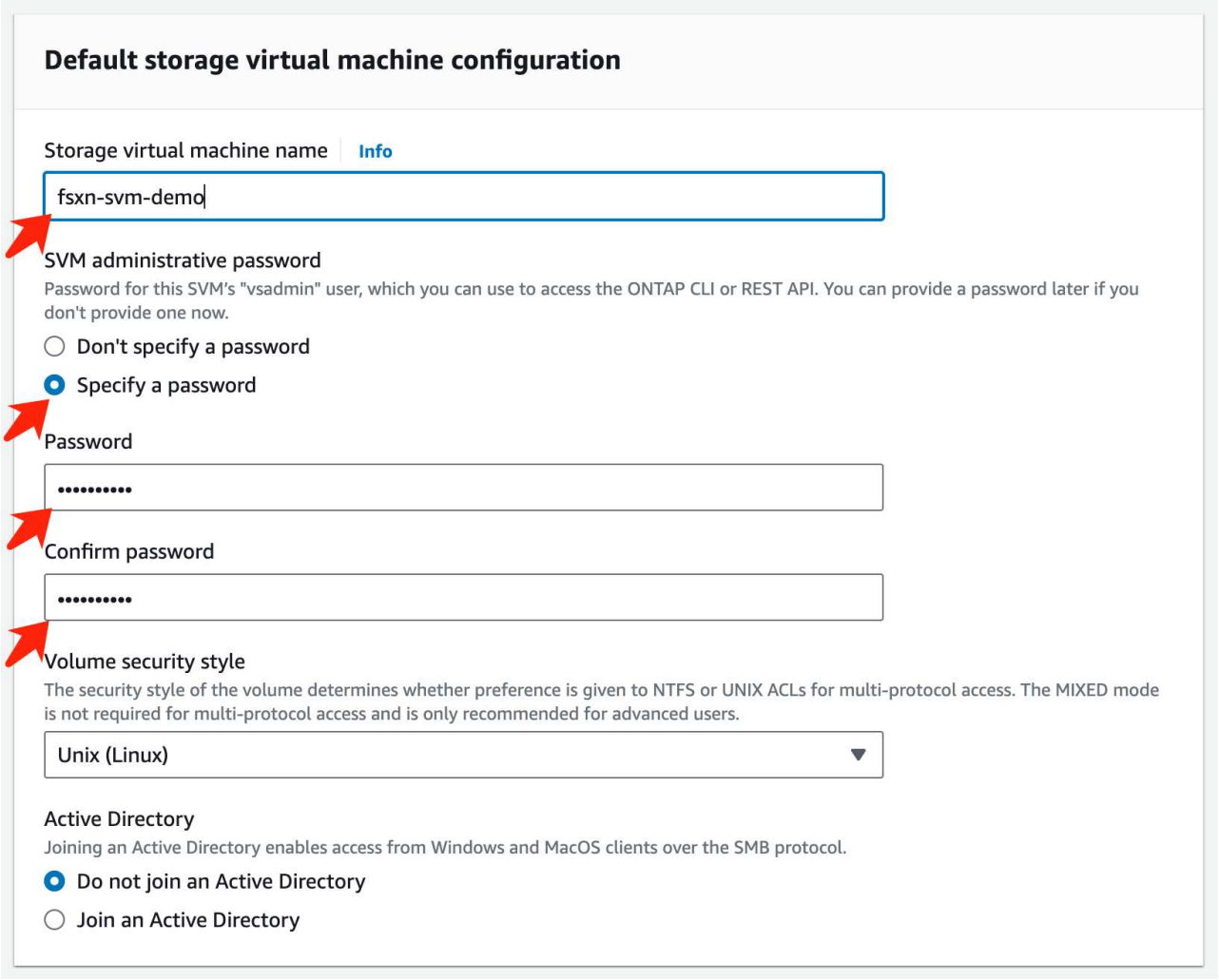
   b. 输入*文件系统名称*和*SSD存储容量*。



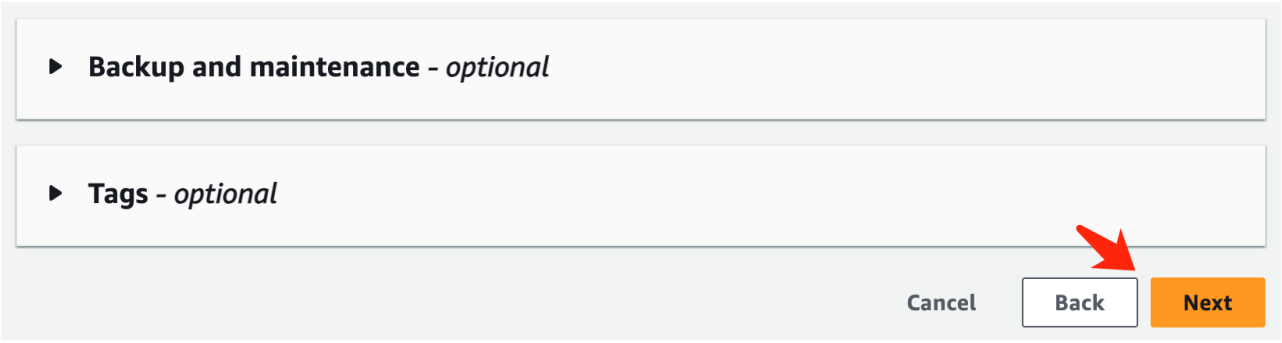
c. 确保使用与 **SageMaker Notebook** 实例相同的 **VPC** 和 **subnet**。



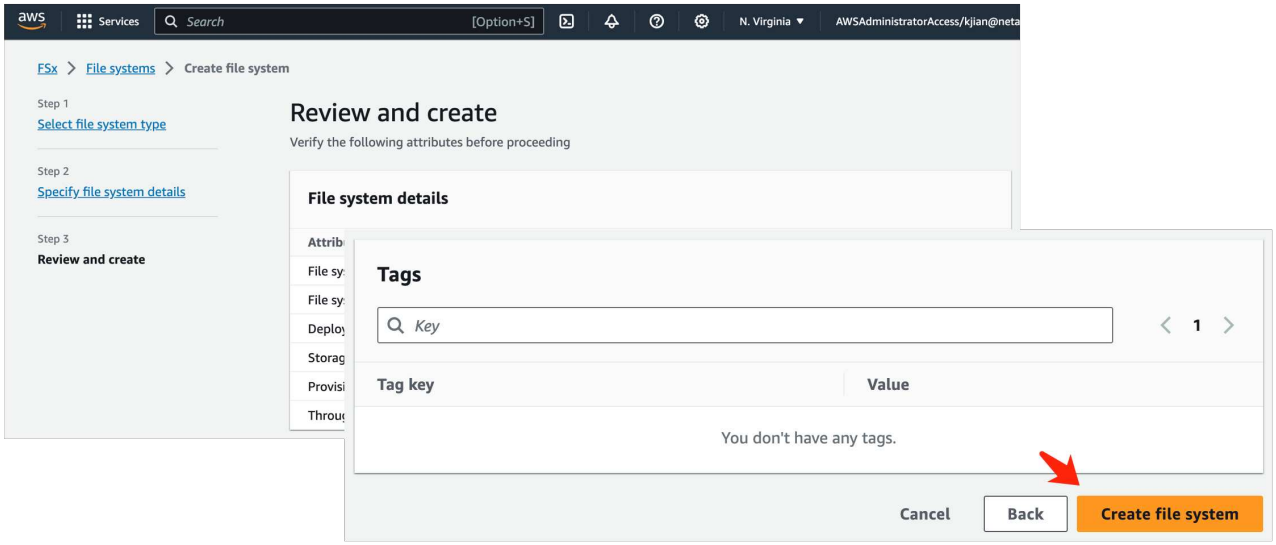
d. 为您的 SVM（存储虚拟机）输入*存储虚拟机*名称和*指定密码*。



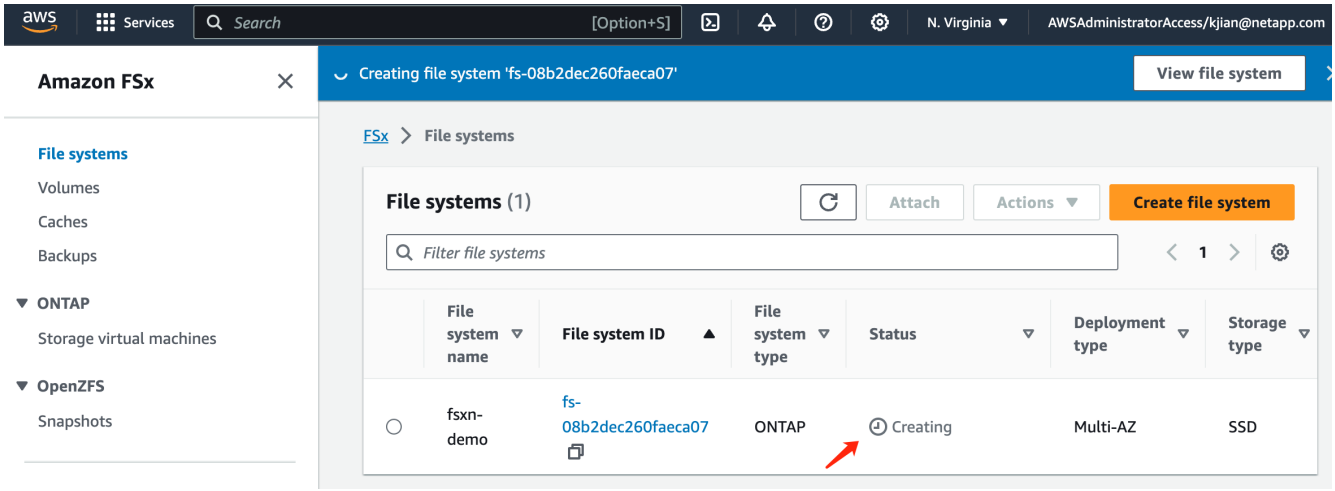
e. 保留其他条目的默认值，然后单击右下角的橙色按钮“下一步”。



f. 点击审核页面右下角的橙色按钮*创建文件系统*。



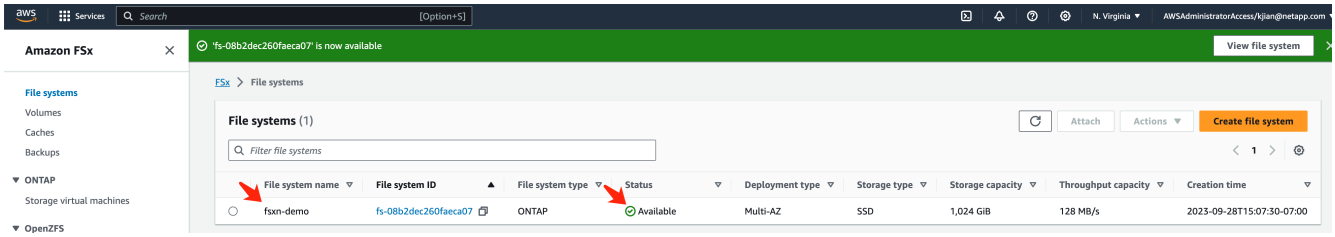
5. 启动 FSx 文件系统可能需要大约 **20-40** 分钟。



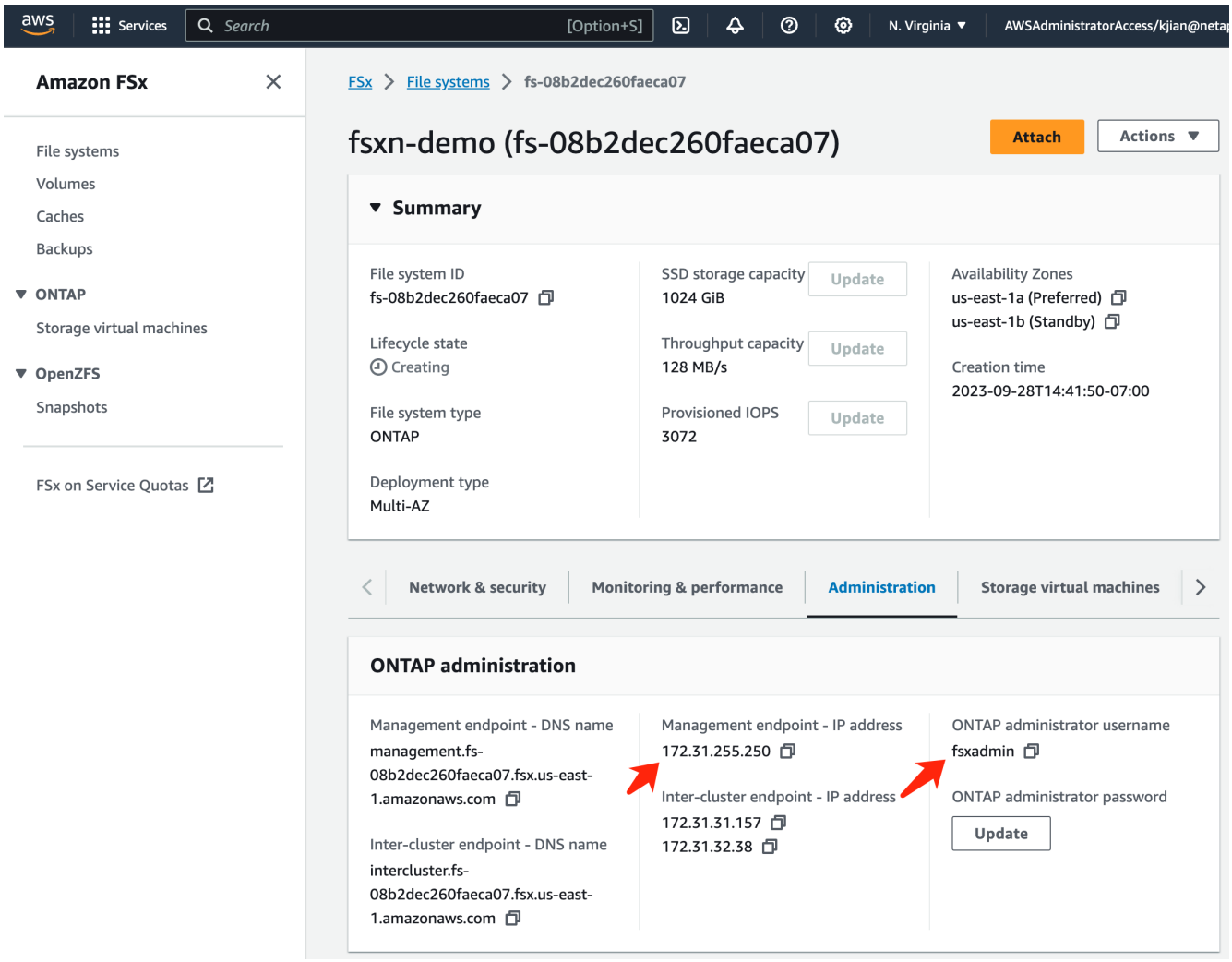
服务器配置

ONTAP 配置

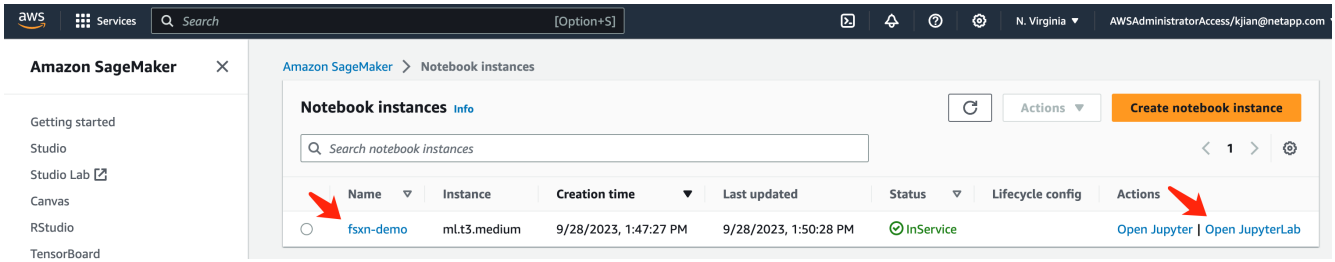
1. 打开创建的FSx文件系统。请确保状态为*可用*。



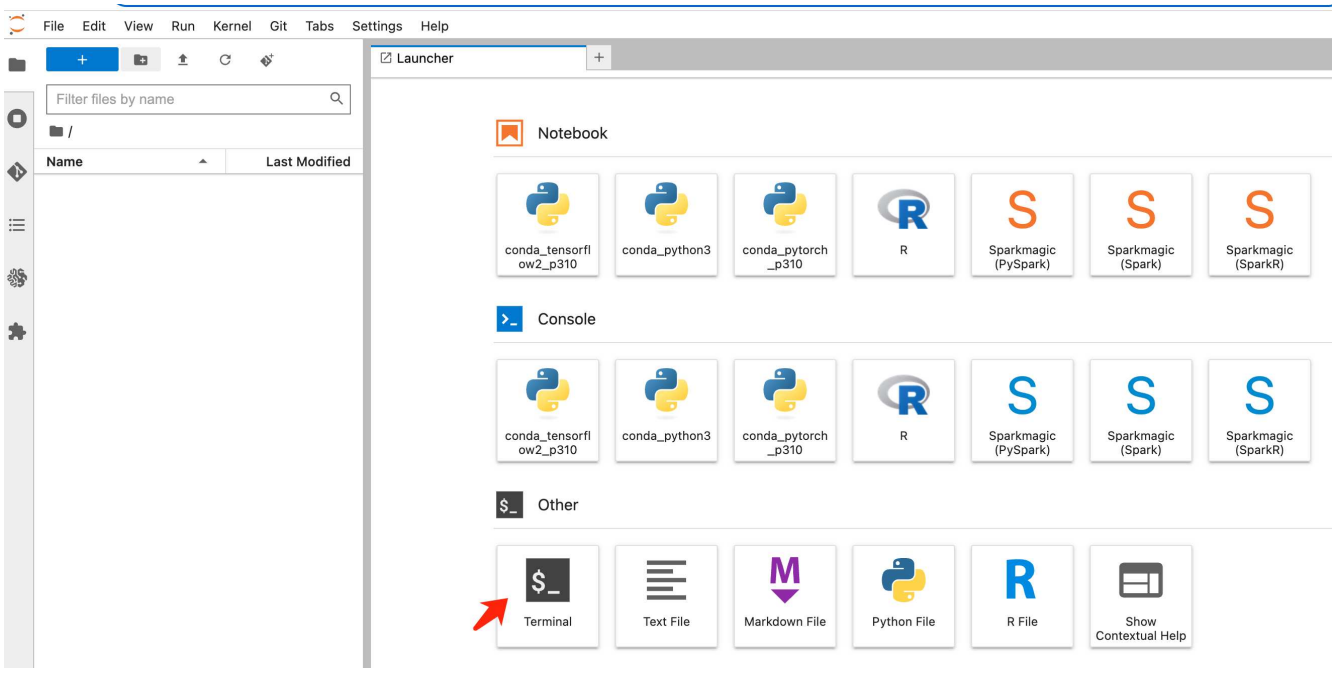
2. 选择*管理*选项卡并保留*管理端点 - IP 地址*和* ONTAP管理员用户名*。



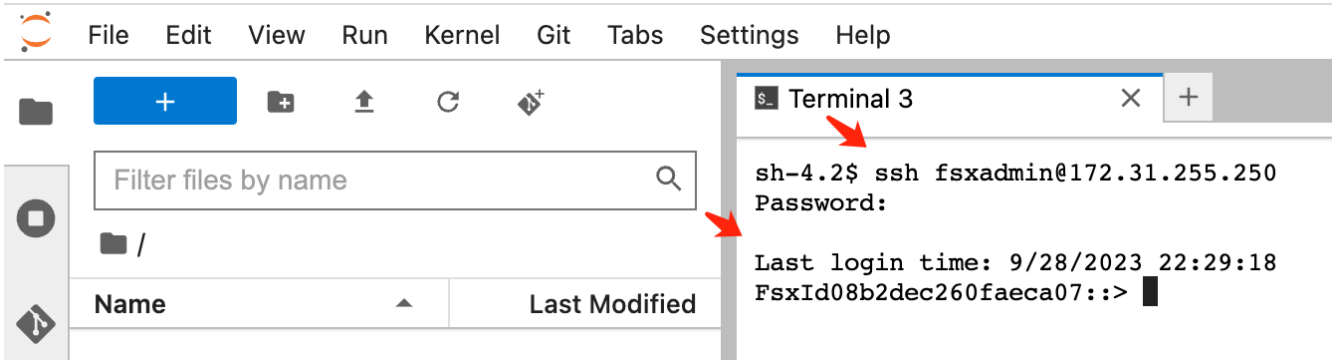
3. 打开创建的*SageMaker Notebook实例*，然后单击*打开JupyterLab*。



4. 在 Jupyter Lab 页面中，打开一个新的*终端*。



5. 输入 ssh 命令 ssh <管理员用户名>@< ONTAP服务器 IP> 登录 FSx ONTAP文件系统。 （用户名和 IP 地址从步骤 2 中检索）请使用创建 存储虚拟机 时使用的密码。



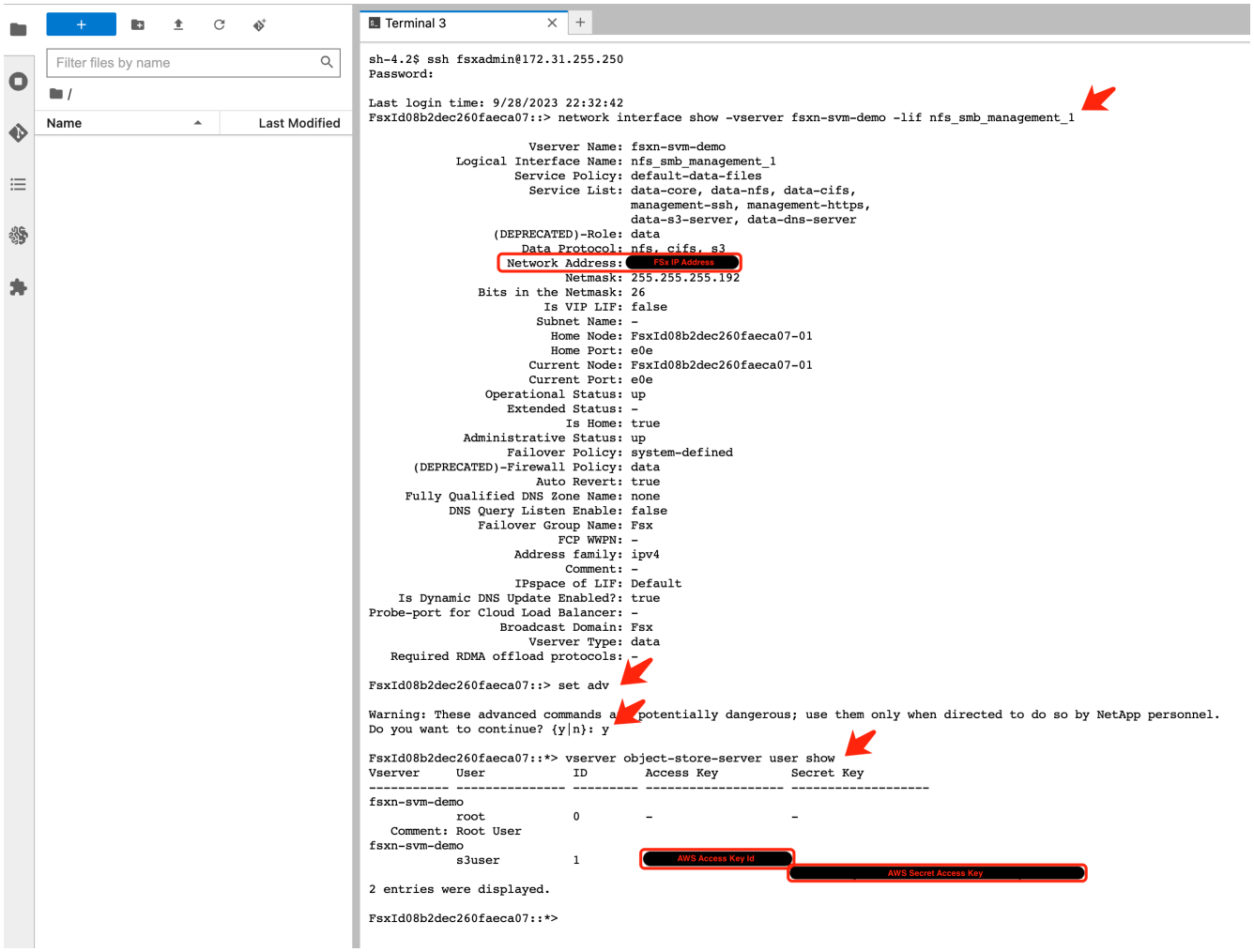
6. 按以下顺序执行命令。我们使用 **fsxn-ontap** 作为 **FSx ONTAP**私有 **S3** 存储桶名称 的名称。请使用*存储虚拟机名称*作为*-vserver* 参数。

```
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. 执行以下命令来检索 FSx ONTAP私有 S3 的端点 IP 和凭据。

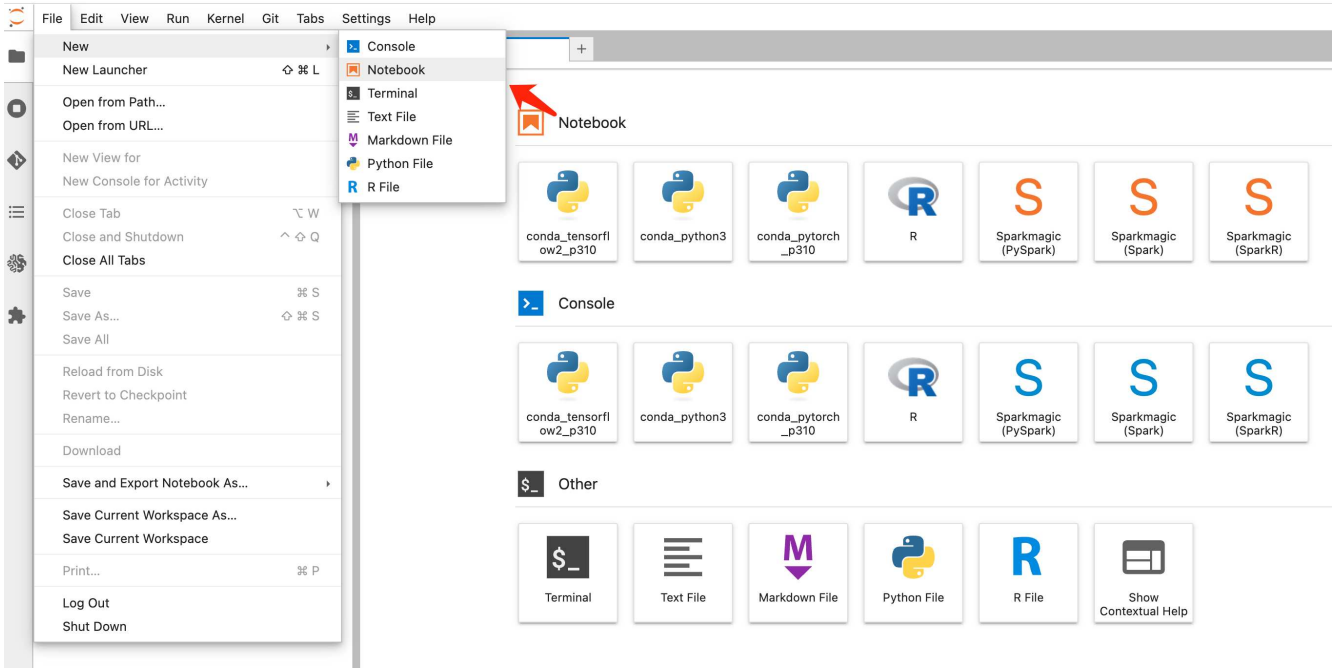
```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show
```

8. 保留端点 IP 和凭证以供将来使用。



客户端配置

1. 在 SageMaker Notebook 实例中，创建一个新的 Jupyter 笔记本。



2. 使用以下代码作为解决方案将文件上传到 FSx ONTAP私有 S3 存储桶。有关全面的代码示例，请参阅此笔记本。"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']
```

这完成了 FSx ONTAP和 SageMaker 实例之间的集成。

有用的调试清单

- 确保 SageMaker Notebook 实例和 FSx ONTAP文件系统位于同一个 VPC 中。
- 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：为什么我在将文件上传到 FSx ONTAP时收到错误“调用 **CreateMultipartUpload** 操作时发生错误（未实现）：您请求的 **s3** 命令未实现”？

答：作为私有 S3 存储桶，FSx ONTAP支持上传最大 100MB 的文件。使用S3协议时，大于100MB的文件会被分成100MB的块，并调用‘CreateMultipartUpload’函数。但是，FSx ONTAP私有 S3 的当前实现不支持此功能。

问：为什么在将文件上传到 FSx ONTAP时出现错误“调用 **PutObject** 操作时发生错误（**AccessDenied**）：访问被拒绝”？

答：要从 SageMaker Notebook 实例访问 FSx ONTAP私有 S3 存储桶，请将 AWS 凭证切换到 FSx ONTAP凭证。但是，授予实例写入权限需要一种解决方法，即安装存储桶并运行“chmod”shell 命令来更改权限。

问：如何将 FSx ONTAP私有 S3 存储桶与其他 SageMaker ML 服务集成？

答：遗憾的是，SageMaker 服务 SDK 没有提供指定私有 S3 存储桶端点的方法。因此，FSx ONTAP S3 与 Sagemaker Data Wrangler、Sagemaker Clarify、Sagemaker Glue、Sagemaker Athena、Sagemaker AutoML 等 SageMaker 服务不兼容。

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

本文是关于使用Amazon FSx for NetApp ONTAP (FSx ONTAP) 在 SageMaker 中训练 PyTorch 模型的教程，具体针对轮胎质量分类项目。

简介

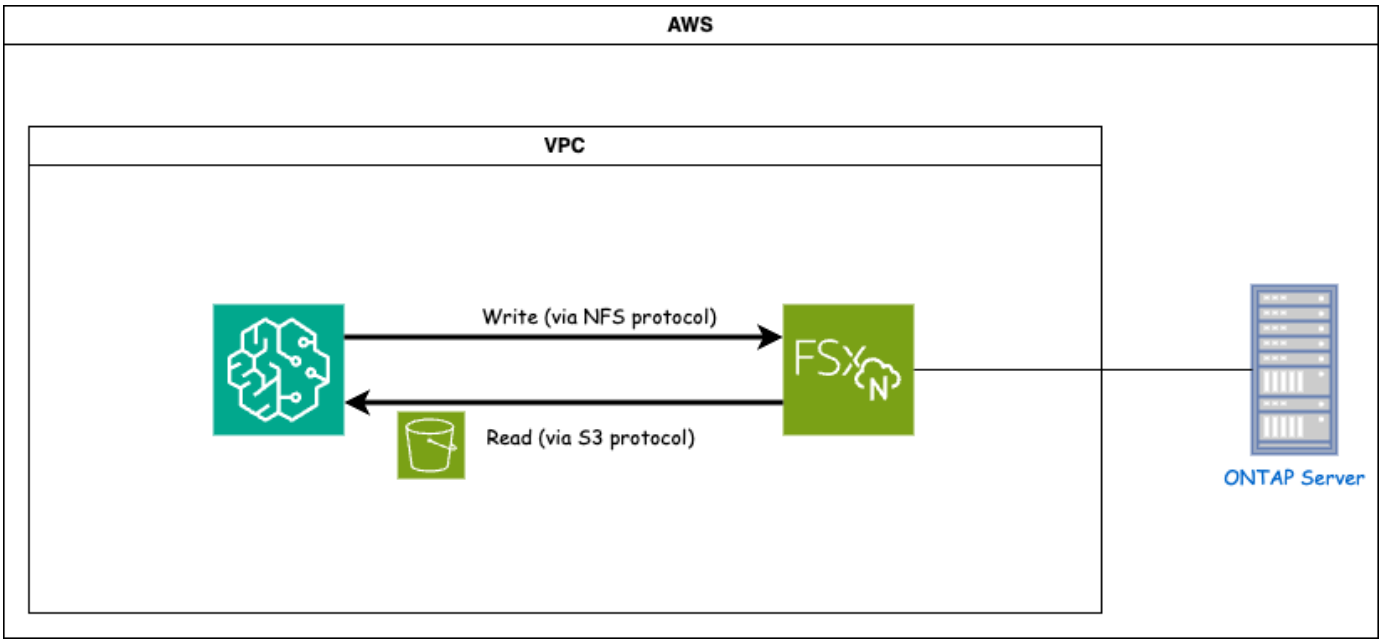
本教程提供了一个计算机视觉分类项目的实际示例，提供了在 SageMaker 环境中利用 FSx ONTAP作为数据源构建 ML 模型的实践经验。该项目专注于使用深度学习框架 PyTorch 根据轮胎图像对轮胎质量进行分类。它强调使用 FSx ONTAP作为 Amazon SageMaker 中的数据源来开发机器学习模型。

什么是 **FSx ONTAP**

Amazon FSx ONTAP确实是 AWS 提供的完全托管的存储解决方案。它利用 NetApp 的ONTAP文件系统提供可靠且高性能的存储。通过支持 NFS、SMB 和 iSCSI 等协议，它允许从不同的计算实例和容器进行无缝访问。该服务旨在提供卓越的性能，确保快速高效的数据操作。它还具有高可用性和耐用性，确保您的数据保持可访问和受保护。此外， Amazon FSx ONTAP的存储容量是可扩展的，您可以根据需要轻松调整。

前提条件

网络环境



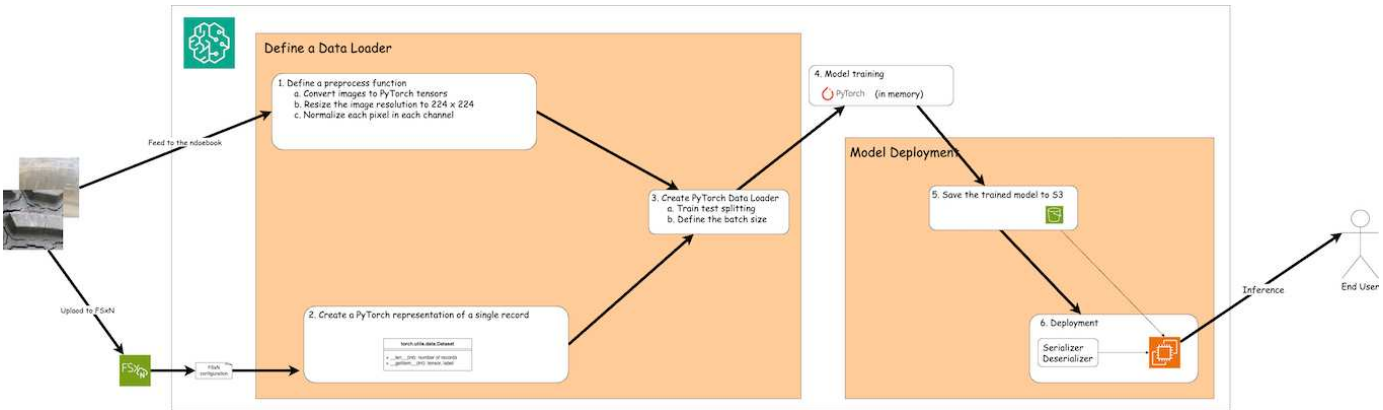
FSx ONTAP （Amazon FSx ONTAP）是一项 AWS 存储服务。它包括在NetApp ONTAP系统上运行的文件系统和连接到它的 AWS 管理的系统虚拟机 (SVM)。在提供的图中，AWS 管理的NetApp ONTAP服务器位于 VPC 外部。 SVM作为SageMaker和NetApp ONTAP系统之间的中介，接收来自SageMaker的操作请求并转发到底层存储。要访问 FSx ONTAP，SageMaker 必须放置在与 FSx ONTAP部署相同的 VPC 内。此配置可确保 SageMaker 和 FSx ONTAP之间的通信和数据访问。

数据访问

在现实场景中，数据科学家通常利用 FSx ONTAP中存储的现有数据来构建他们的机器学习模型。但是，出于演示目的，由于 FSx ONTAP文件系统在创建后最初是空的，因此需要手动上传训练数据。这可以通过将 FSx ONTAP作为卷安装到 SageMaker 来实现。文件系统成功挂载后，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。这种方法允许您利用 FSx ONTAP的存储容量和功能，同时与 SageMaker 进行模型开发和训练。

数据读取过程涉及将 FSx ONTAP配置为私有 S3 存储桶。详细配置说明请参考"第 1 部分 - 将Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为私有 S3 存储桶集成到 AWS SageMaker"

集成概述



使用 FSx ONTAP中的训练数据在 SageMaker 中构建深度学习模型的工作流程可以概括为三个主要步骤：数据加载器定义、模型训练和部署。从高层次来看，这些步骤构成了 MLOps 管道的基础。然而，为了全面实施，每

个步骤都涉及几个详细的子步骤。这些子步骤涵盖数据预处理、数据集拆分、模型配置、超参数调整、模型评估和模型部署等各种任务。这些步骤确保在 SageMaker 环境中使用来自 FSx ONTAP的训练数据构建和部署深度学习模型的流程全面有效。

逐步集成

数据**Loader**

为了使用数据训练 PyTorch 深度学习网络，创建了一个数据加载器以方便数据的输入。数据加载器不仅定义批次大小，还确定读取和预处理批次中每个记录的过程。通过配置数据加载器，我们可以批量处理数据，从而实现深度学习网络的训练。

数据加载器由3部分组成。

预处理函数

```
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]
    )
])
```

上面的代码片段演示了使用 **torchvision.transforms** 模块定义图像预处理转换。在本教程中，创建预处理对象来应用一系列转换。首先，**ToTensor()** 转换将图像转换为张量表示。随后，**Resize 224,224** 转换将图像调整为固定大小 224x224 像素。最后，**Normalize()** 转换通过减去平均值并除以每个通道的标准差来对张量值进行归一化。用于标准化的平均值和标准差值通常用于预训练的神经网络模型。总的来说，这段代码通过将图像数据转换为张量、调整其大小以及规范化像素值来准备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
```

该类提供获取数据集中记录总数的功能，并定义读取每条记录数据的方法。在 ***getitem*** 函数中，代码利用 boto3 S3 bucket 对象从 FSx ONTAP检索二进制数据。从 FSx ONTAP访问数据的代码样式类似于从 Amazon S3 读取数据。后续讲解深入探讨私有 S3 对象 **bucket** 的创建过程。

FSx ONTAP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 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 ---
```

为了从 SageMaker 中的 FSx ONTAP读取数据，需要创建一个使用 S3 协议指向 FSx ONTAP存储的处理程序。这使得 FSx ONTAP可以被视为私有 S3 存储桶。处理程序配置包括指定 FSx ONTAP SVM 的 IP 地址、存储桶名称和必要的凭据。有关获取这些配置项的详细说明，请参阅以下文档："第 1 部分 - 将Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为私有 S3 存储桶集成到 AWS SageMaker" 。

在上面的例子中，bucket对象用于实例化PyTorch数据集对象。数据集对象将在后续章节中进一步解释。

PyTorch 数据Loader

```
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，指定批次大小为 64，表示每个批次将包含 64 条记录。通过结合 PyTorch **Dataset** 类、预处理函数和训练批次大小，我们获得了用于训练的数据加载器。该数据加载器有助于在训练阶段分批迭代数据集的过程。

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')
```

此代码实现了标准的 PyTorch 训练流程。它定义了一个名为*TyreQualityClassifier*的神经网络模型，使用卷积层和线性层来对轮胎质量进行分类。训练循环迭代数据批次，计算损失，并使用反向传播和优化更新模型的参数。此外，它还打印当前时间、纪元、批次和损失以供监控目的。

模型部署

部署

```
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())
```

代码将 PyTorch 模型保存到 **Amazon S3**，因为 SageMaker 要求将模型存储在 S3 中以便部署。通过将模型上传到 **Amazon S3**，SageMaker 就可以访问它，从而允许对已部署的模型进行部署和推理。

```
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序列化器 **TyreQualitySerializer**，它将输入数据预处理并序列化为 PyTorch 张量。 **TyreQualityPredictor** 类是一个自定义预测器，它利用定义的序列化器和 **JSONDeserializer**。该代码还创建了一个 **PyTorchModel** 对象来指定模型的 S3 位置、IAM 角色、框架版本和推理的入口点。代码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，指定实例数量、实例类型和生成的端点名称。这使得 PyTorch 模型可以在 SageMaker 上部署并进行推理。

```
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进行推理的示例。

第 3 部分 - 构建简化的 MLOps 管道 (CI/CT/CD)

本文提供了使用 AWS 服务构建 MLOps 管道的指南，重点关注自动模型再训练、部署和成本优化。

简介

在本教程中，您将学习如何利用各种 AWS 服务构建一个包含持续集成 (CI)、持续训练 (CT) 和持续部署 (CD) 的简单 MLOps 管道。与传统的 DevOps 管道不同，MLOps 需要额外的考虑才能完成操作周期。通过学习本教程，您将深入了解如何将 CT 纳入 MLOps 循环，从而实现模型的持续训练和推理的无缝部署。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。

显现

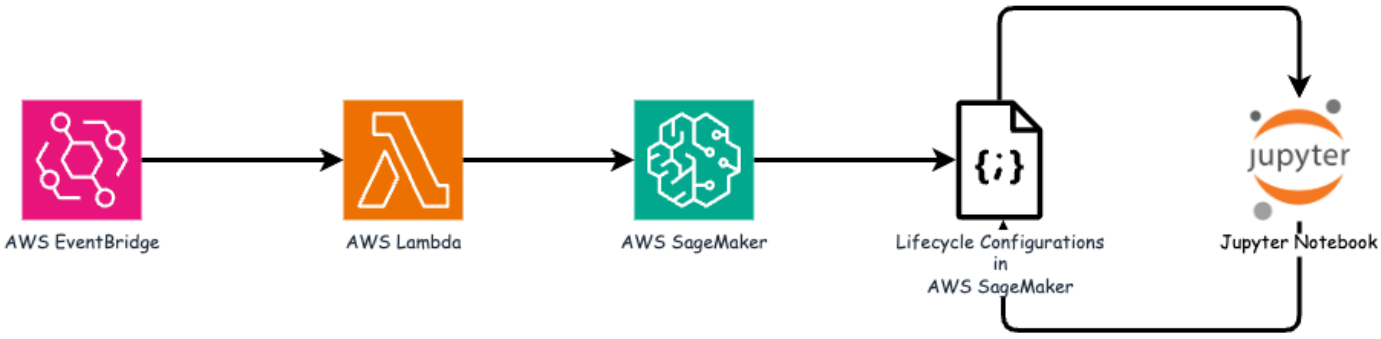
功能	名称	注释
数据存储	AWS FSx ONTAP	请参阅"第 1 部分 - 将Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为私有 S3 存储桶集成到 AWS SageMaker" 。
数据科学 IDE	AWS SageMaker	本教程基于"第 2 部分 - 利用Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为 SageMaker 模型训练的数据源"。
触发 MLOps 管道的函数	AWS Lambda 函数	-
Cron 作业触发器	AWS EventBridge	-
深度学习框架	PyTorch	-
AWS Python 开发工具包	boto3	-
编程语言	Python	v3.10

前提条件

- 预配置的 FSx ONTAP文件系统。本教程利用 FSx ONTAP中存储的数据进行训练过程。

- 配置为与上面提到的 FSx ONTAP文件系统共享相同 VPC 的 **SageMaker Notebook** 实例。
- 在触发 **AWS Lambda** 函数 之前，请确保 **SageMaker Notebook** 实例 处于 已停止 状态。
- 需要 **ml.g4dn.xlarge** 实例类型来利用深度神经网络计算所需的 GPU 加速。

架构



这个 MLOps 管道是一个实际的实现，它利用 cron 作业来触发无服务器功能，进而执行使用生命周期回调函数注册的 AWS 服务。 **AWS EventBridge** 充当 cron 作业。它定期调用负责重新训练和重新部署模型的 **AWS Lambda** 函数。此过程涉及启动 **AWS SageMaker Notebook** 实例以执行必要的任务。

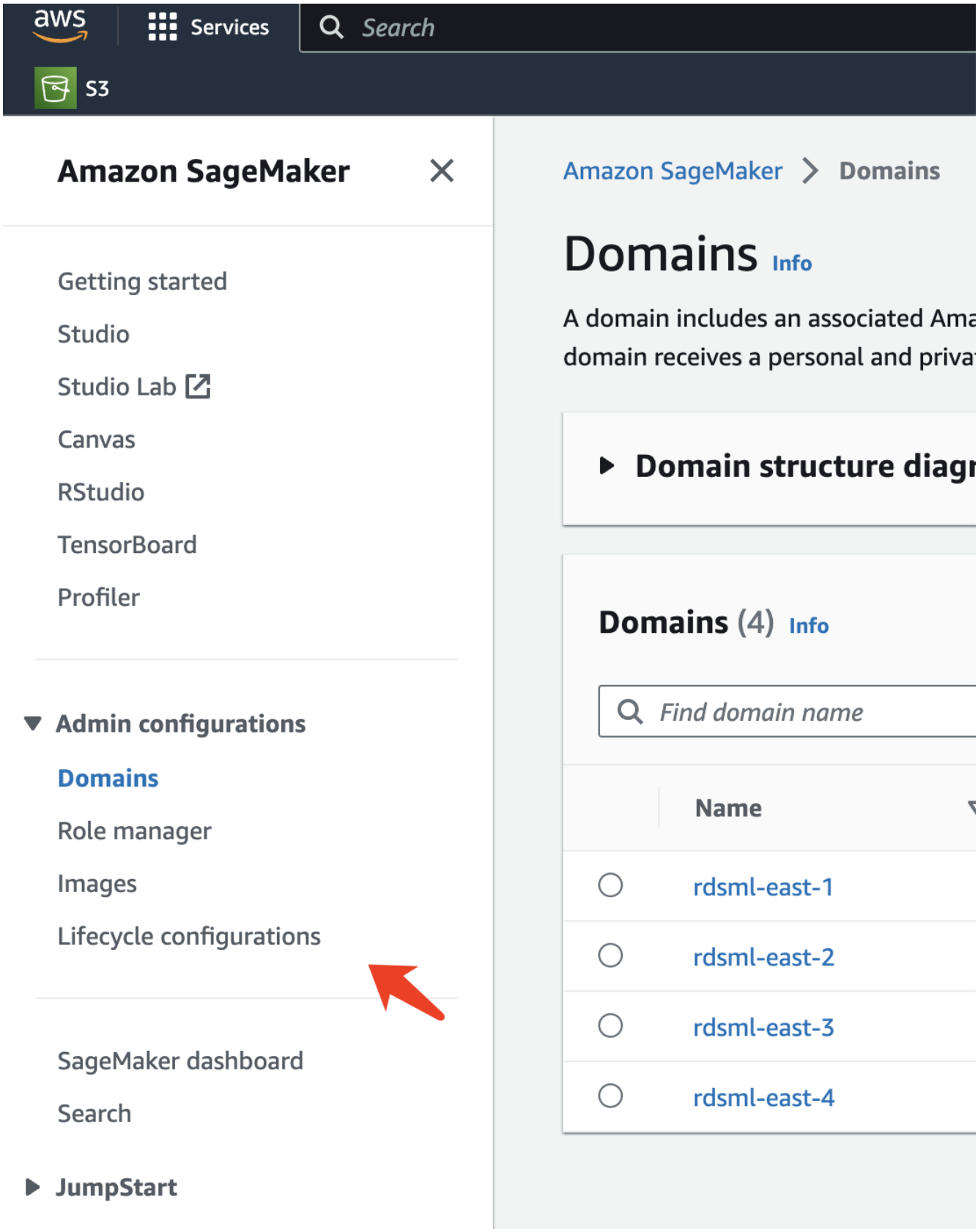
分步配置

生命周期配置

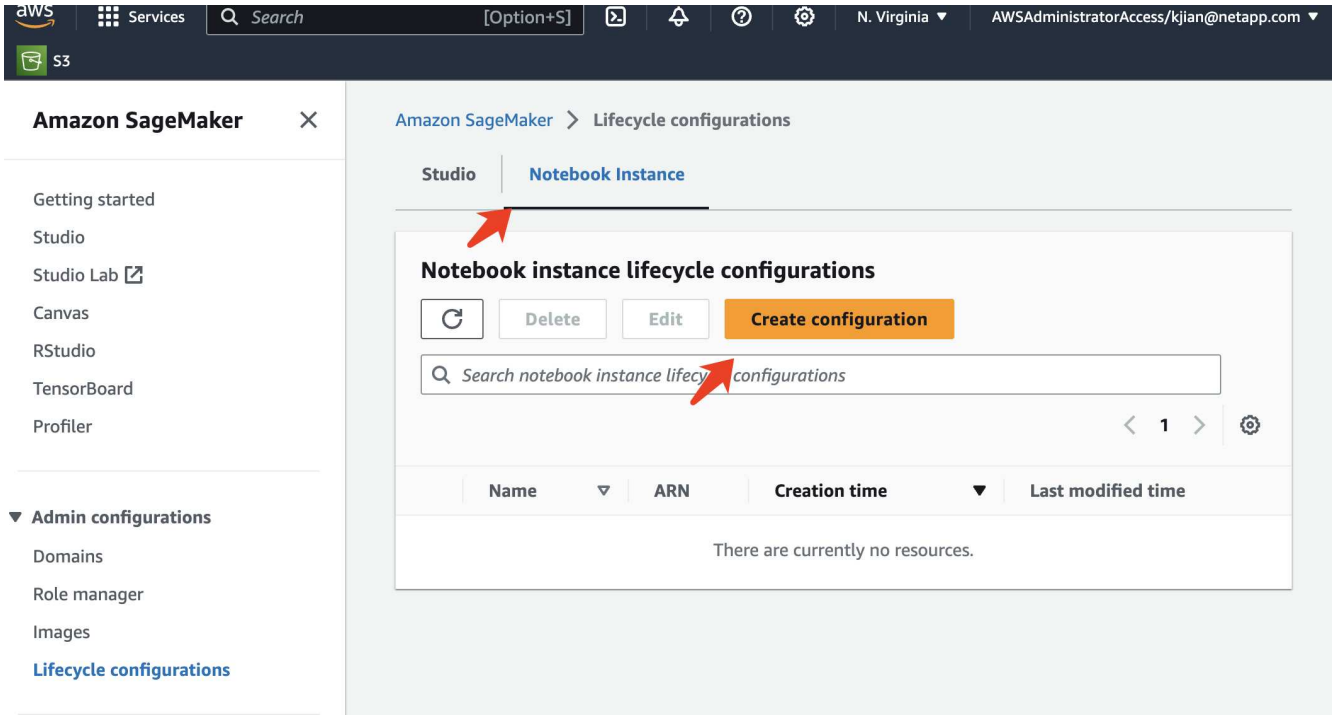
要为 AWS SageMaker Notebook 实例配置生命周期回调函数，您需要使用*生命周期配置*。此服务允许您定义启动笔记本实例时需要执行的必要操作。具体来说，可以在*生命周期配置*中实现一个 shell 脚本，以便在训练和部署过程完成后自动关闭笔记本实例。这是必需的配置，因为成本是 MLOps 中的主要考虑因素之一。

值得注意的是，*生命周期配置*的配置需要提前设置。因此，建议在继续其他 MLOps 管道设置之前优先配置这一方面。

1. 要设置生命周期配置，请打开 **Sagemaker** 面板并导航到 **Admin configuration** 部分下的 **Lifecycle configuration**。



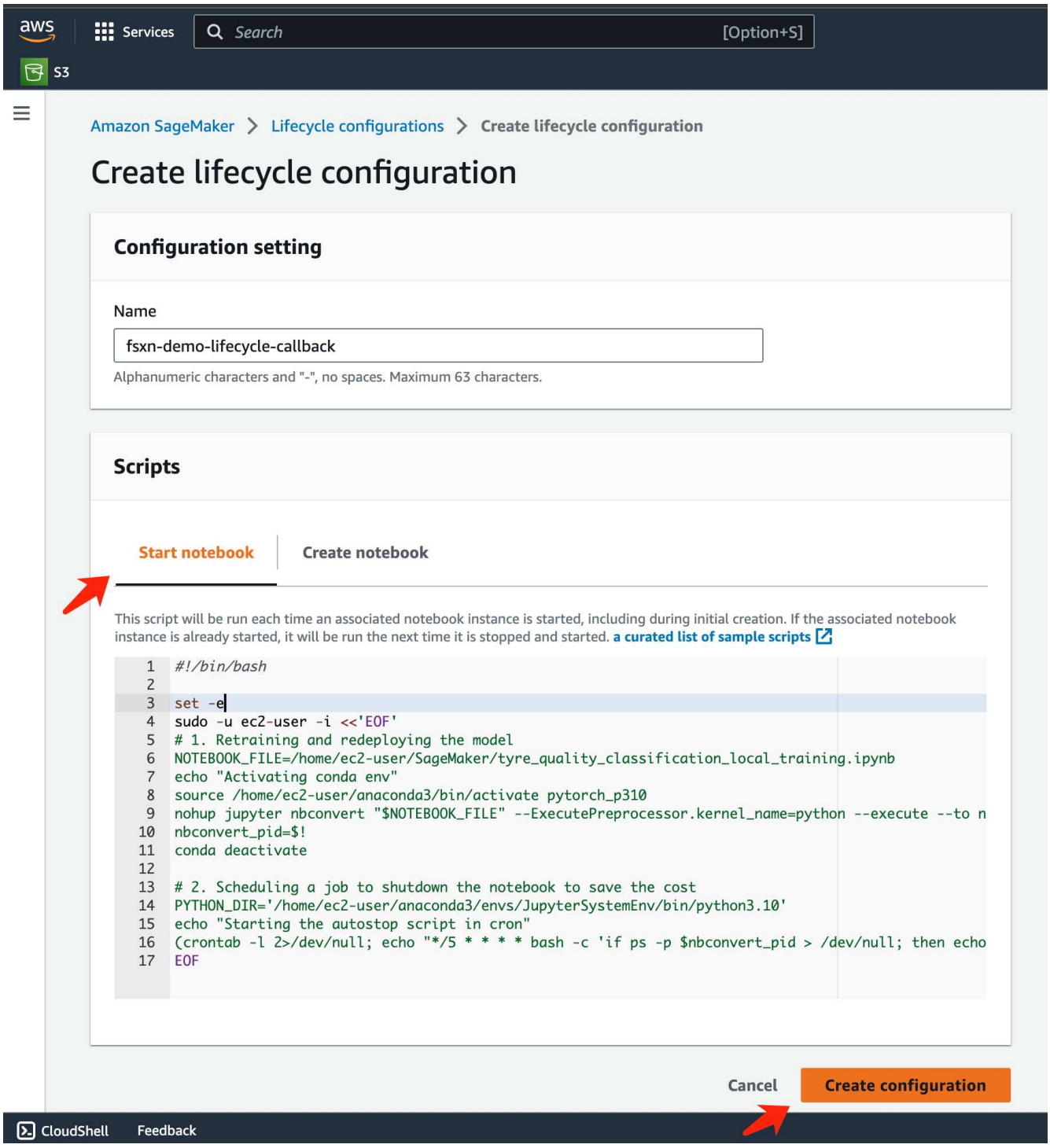
2. 选择“笔记本实例”选项卡，然后单击“创建配置”按钮



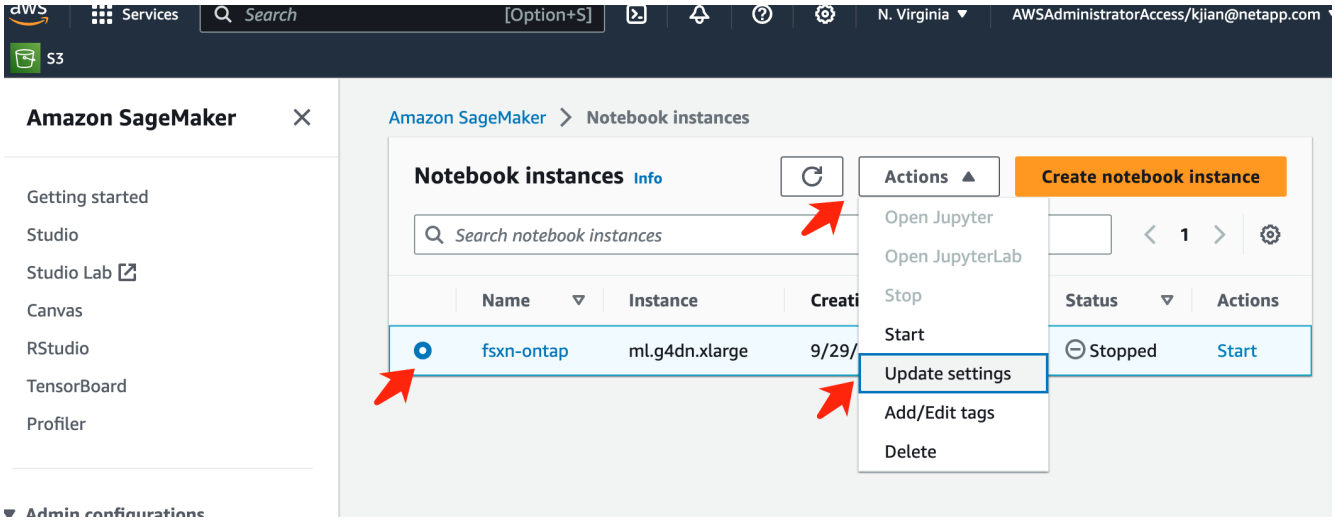
3. 将以下代码粘贴到输入区域。

```
#!/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
```

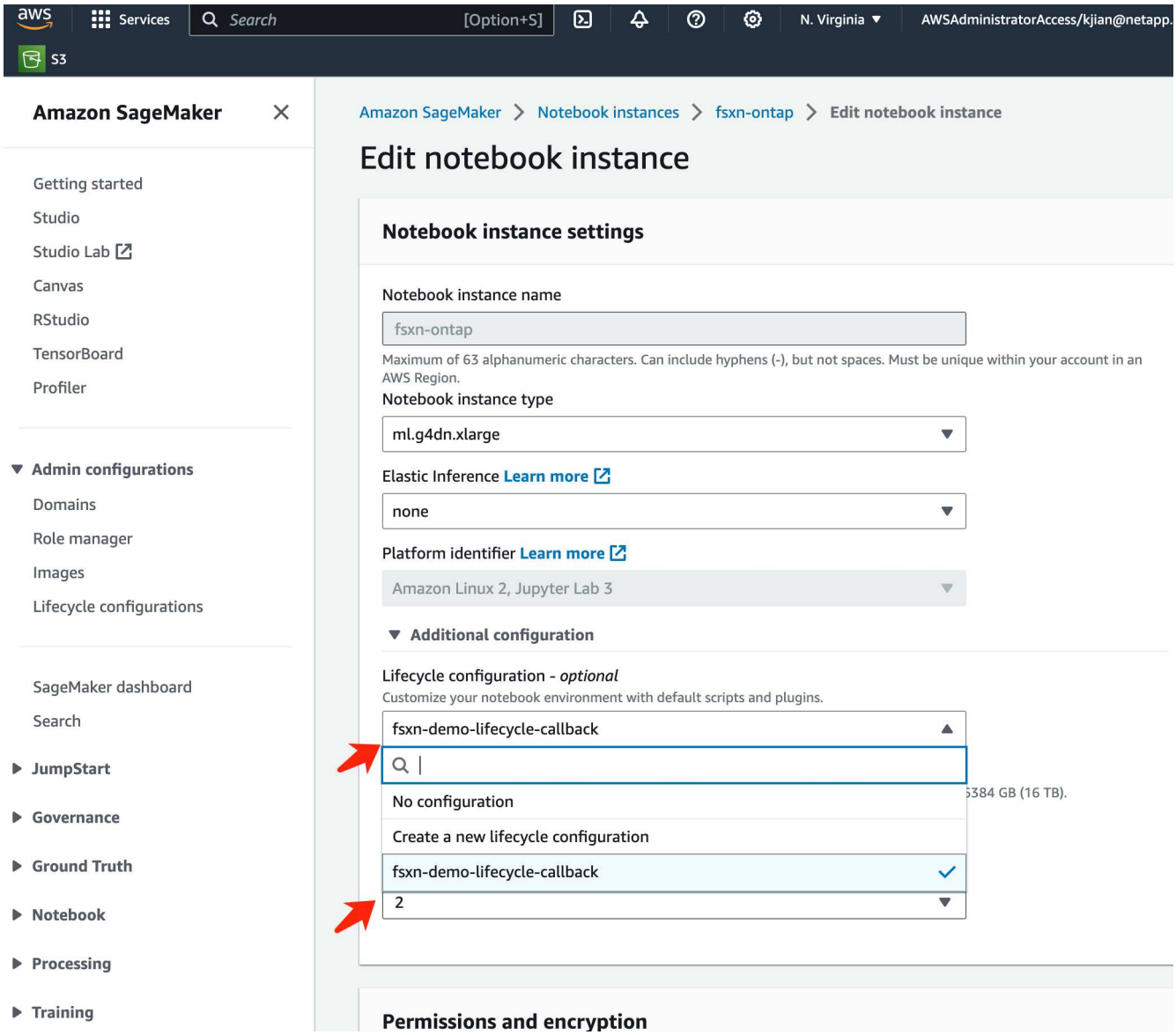
4. 该脚本执行 Jupyter Notebook，用于处理模型的重新训练和重新部署以进行推理。执行完成后，笔记本将在5分钟内自动关机。要了解有关问题陈述和代码实现的更多信息，请参阅"第 2 部分 - 利用Amazon FSx for NetApp ONTAP (FSx ONTAP) 作为 SageMaker 模型训练的数据源"。



5. 创建后，导航到 Notebook 实例，选择目标实例，然后单击“操作”下拉菜单下的“更新设置”。



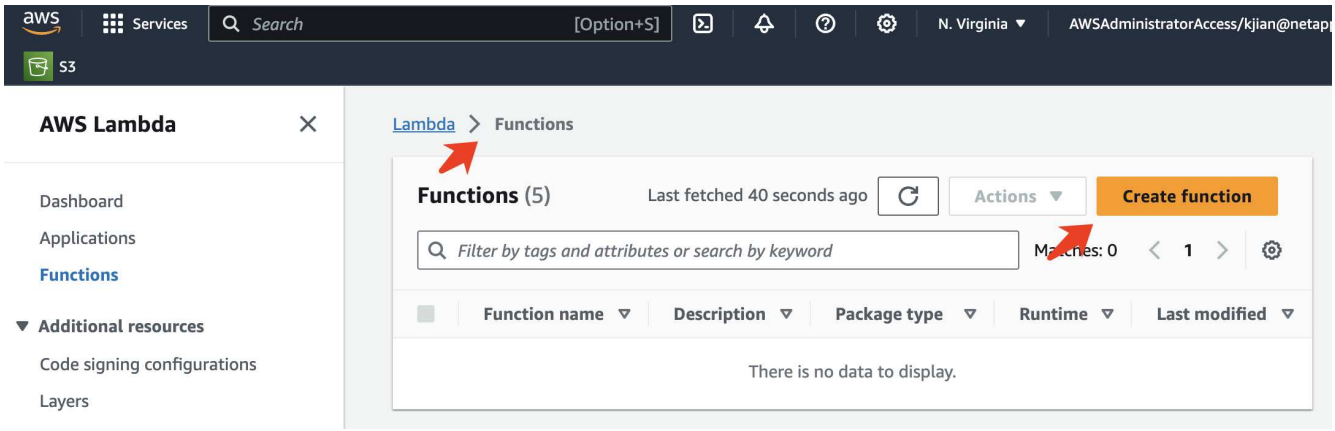
6. 选择创建的*生命周期配置*，然后单击*更新笔记本实例*。



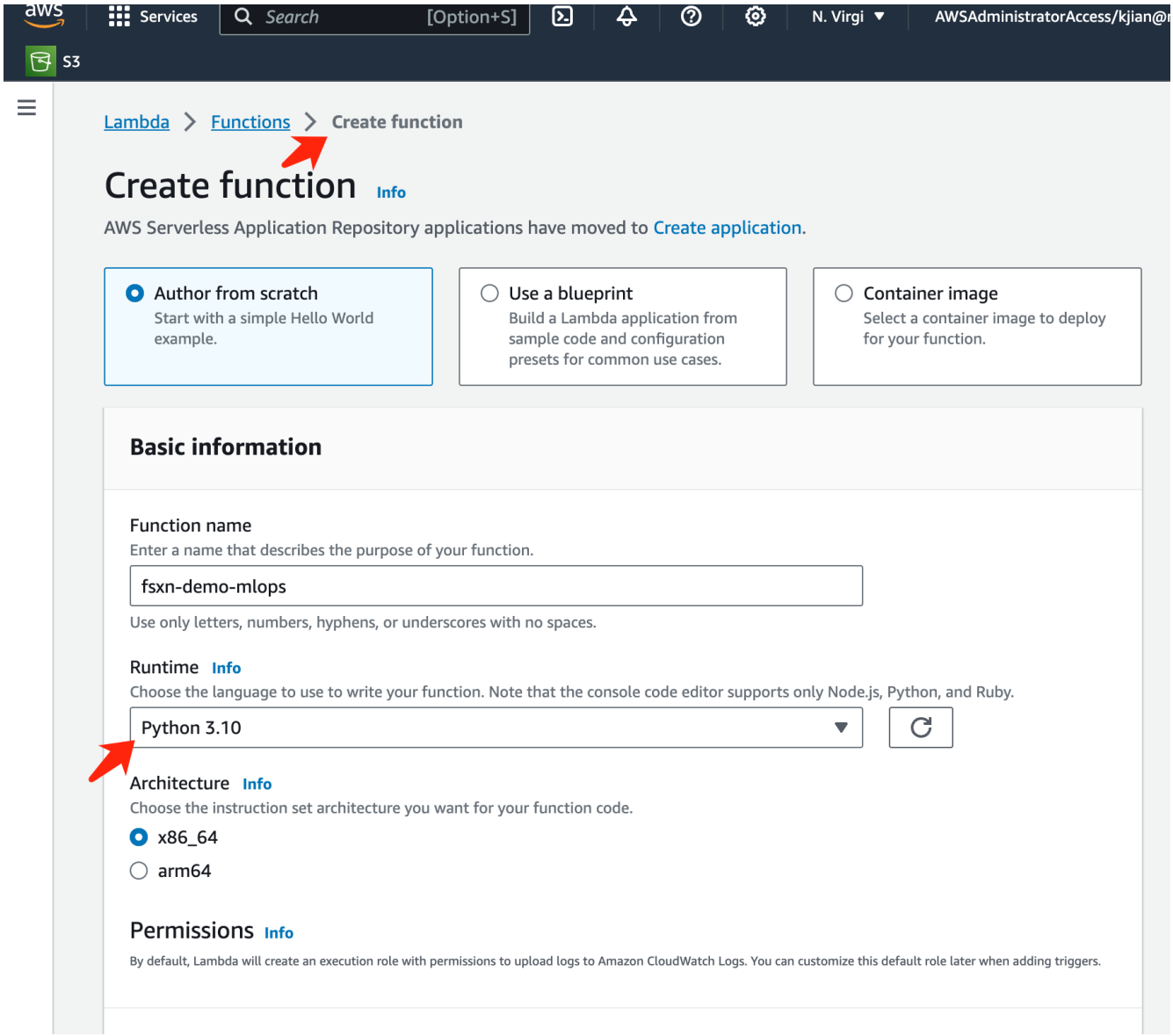
AWS Lambda 无服务器函数

如前所述，**AWS Lambda** 函数*负责启动 ***AWS SageMaker Notebook** 实例。

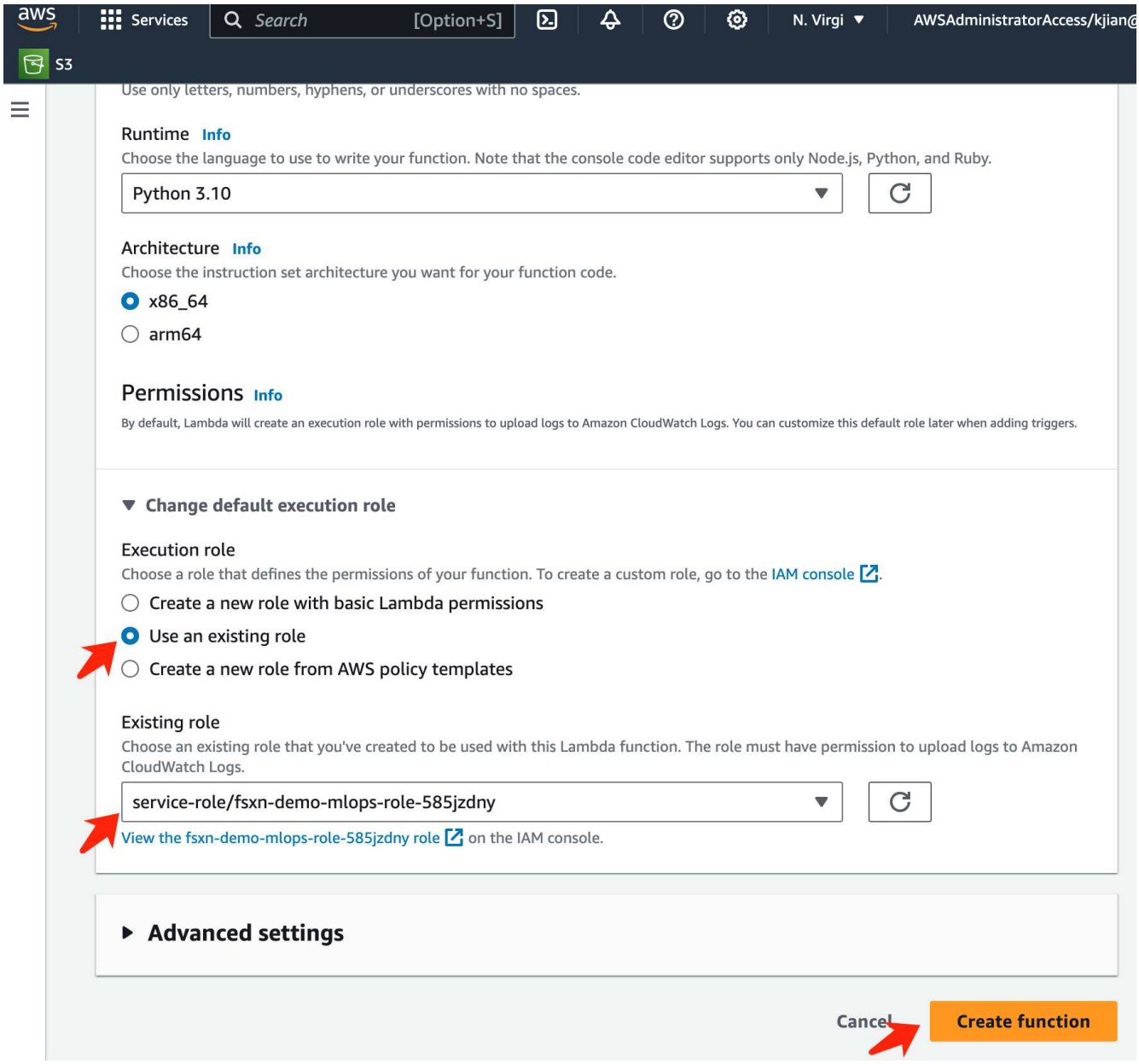
1. 要创建 **AWS Lambda** 函数，请导航到相应的面板，切换到 函数 选项卡，然后单击 创建函数。



2. 请在页面上提交所有必需的条目，并记得将运行时切换为 **Python 3.10**。



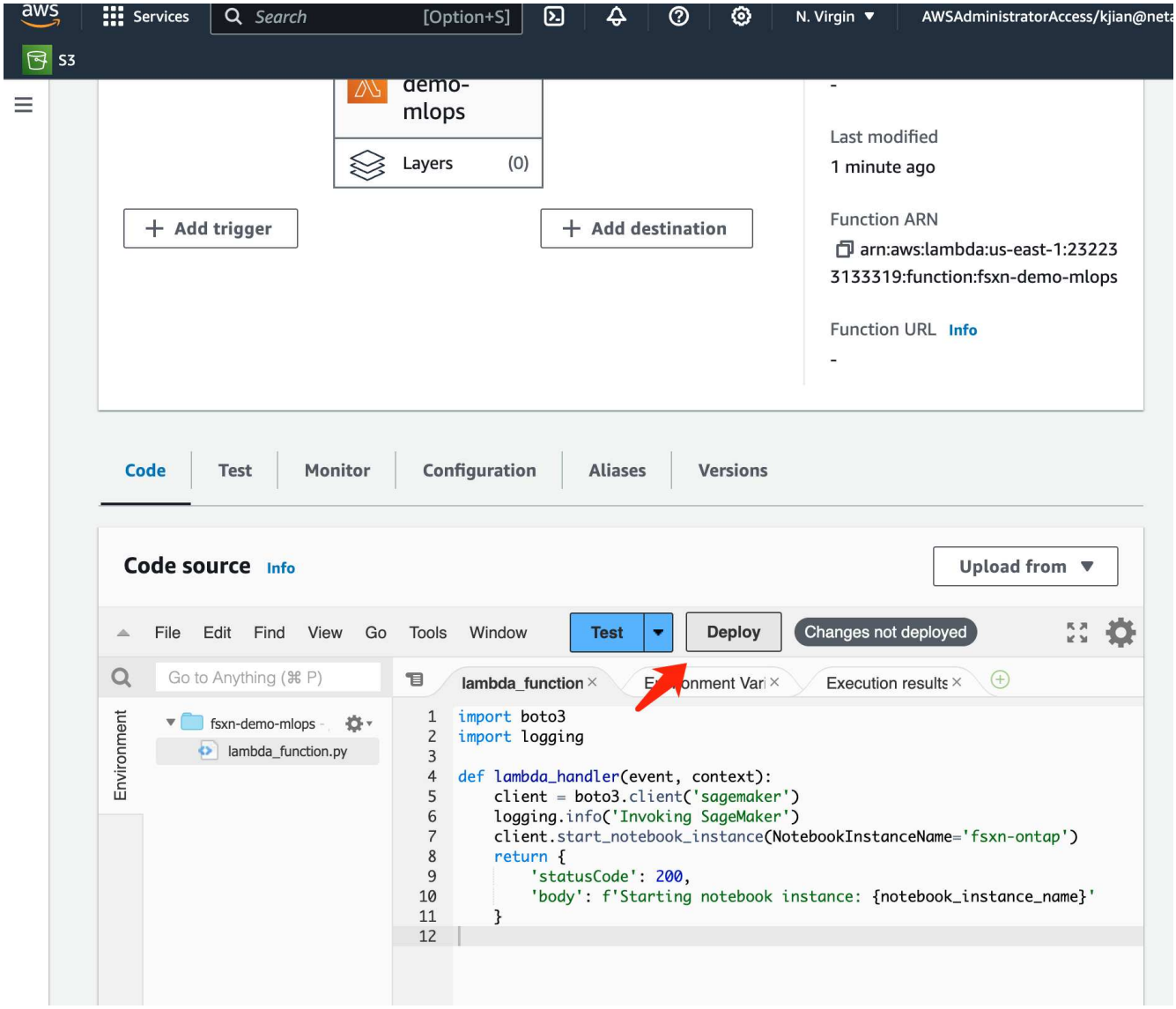
3. 请验证指定的角色是否具有所需的权限*AmazonSageMakerFullAccess*，然后单击*创建功能*按钮。



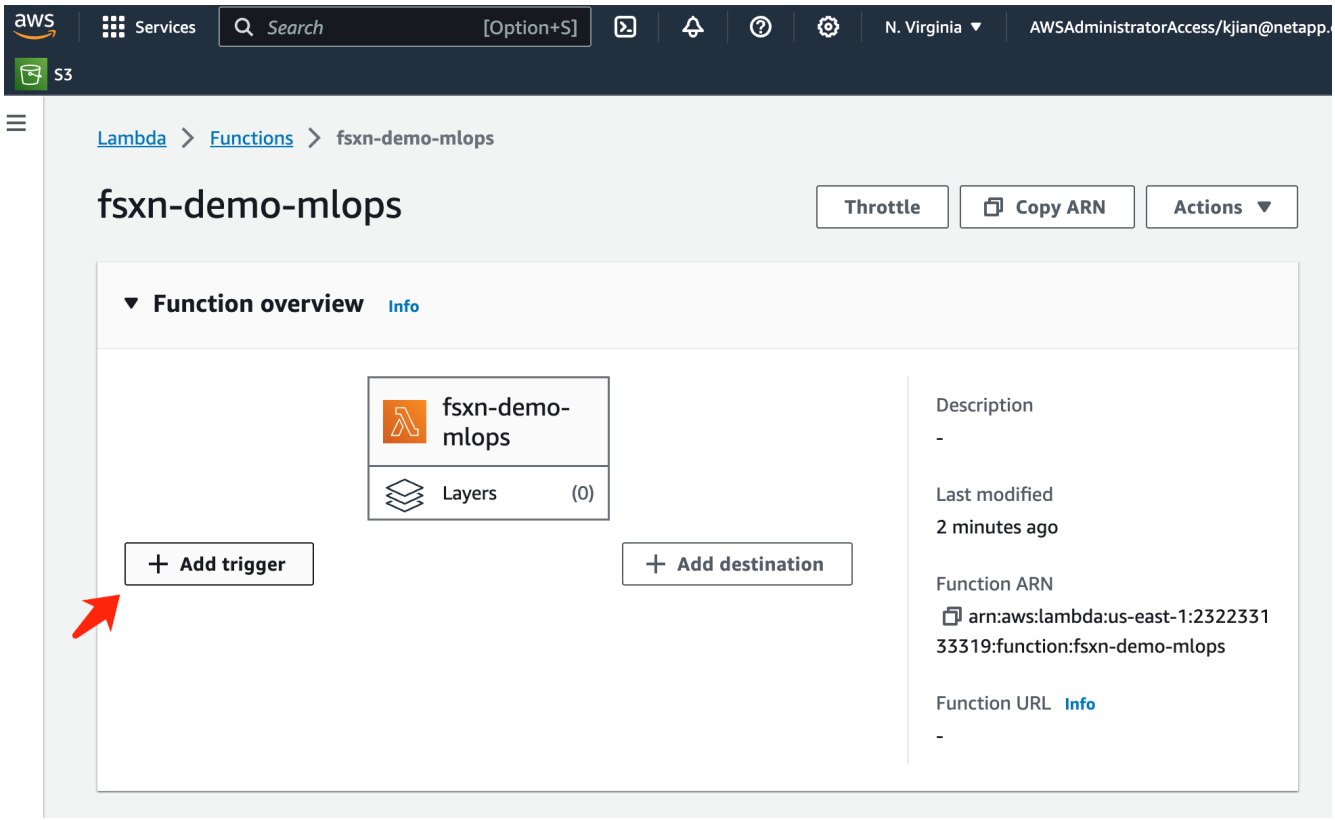
4. 选择创建的Lambda函数。在代码选项卡中，将以下代码复制并粘贴到文本区域中。此代码启动名为 **fsxn-ontap** 的笔记本实例。

```
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}'
    }
```

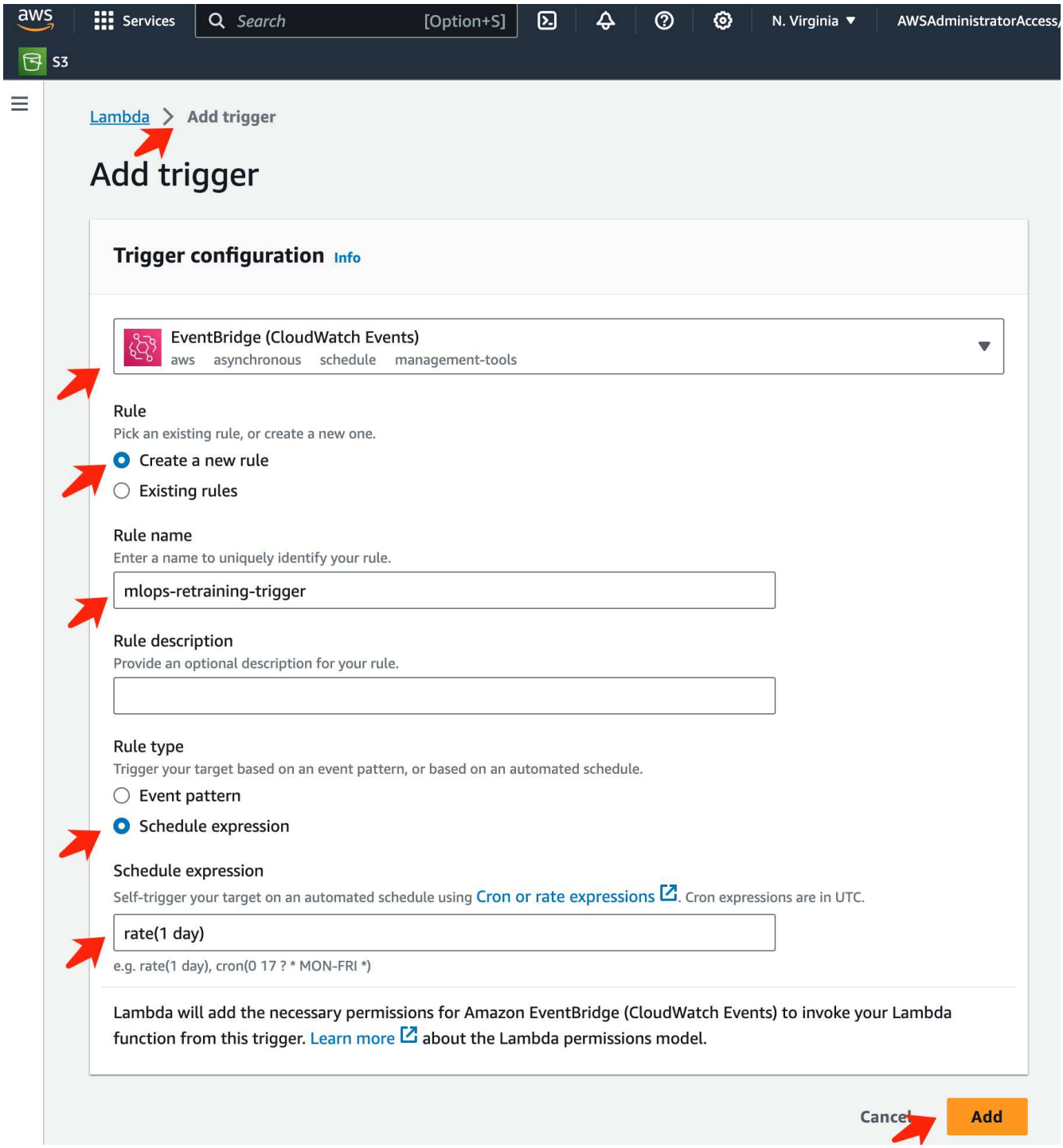
5. 单击“部署”按钮以应用此代码更改。



6. 要指定如何触发此 AWS Lambda 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并将此新的 cron 作业规则应用于 AWS Lambda 函数。



完成这两步配置后，每天，**AWS Lambda** 函数*都会启动 ***SageMaker Notebook**，使用 **FSx ONTAP** 存储库中的数据执行模型重新训练，将更新后的模型重新部署到生产环境，并自动关闭 *SageMaker Notebook 实例*以优化成本。这确保模型保持最新。

开发 MLOps 管道的教程到此结束。