MLOps向けAmazon FSx for NetApp ONTAP（FSx ONTAP）

NetApp Solutions

NetApp
December 19, 2024

目次

MLOps向けAmazon FSx for NetApp ONTAP（FSx ONTAP）

MLOps向けAmazon FSx for NetApp ONTAP（FSx ONTAP）

このセクションでは、AIインフラ開発の実践的なアプリケーションについて詳しく説明し、FSx ONTAPを使用してMLOpsパイプラインを構築するためのエンドツーエンドのウォークスルーを提供します。3つの包括的な例で構成され、この強力なデータ管理プラットフォームを通じてMLOpsのニーズを満たすことができます。

作成者：
NetAppシニアデータ&アプリケーションサイエンティスト、Jian Jian（Ken）

以下の記事では、

1. "パート1 - Amazon FSx for NetApp ONTAP（FSx ONTAP）をプライベートS3バケットとしてAWS SageMakerに統合する"
2. "パート2 - SageMakerのモデルトレーニングのデータソースとしてAmazon FSx for NetApp ONTAP（FSx ONTAP）を活用"
3. "パート3 -簡易化されたMLOpsパイプラインの構築（CI/CT/CD）"

このセクションを終えると、FSx ONTAPを使用してMLOpsプロセスを合理化する方法をしっかりと理解できるようになります。

パート1 - Amazon FSx for NetApp ONTAP（FSx ONTAP）をプライベートS3バケットとしてAWS SageMakerに統合する

このセクションでは、AWS SageMakerを使用して、FSx ONTAPをプライベートS3バケットとして設定するためのガイドを提供します。

作成者：
NetAppシニアデータ&アプリケーションサイエンティスト、Jian Jian（Ken）

はじめに

このページでは、SageMakerを例に、FSx ONTAPをプライベートS3バケットとして設定するためのガイダンスを示します。

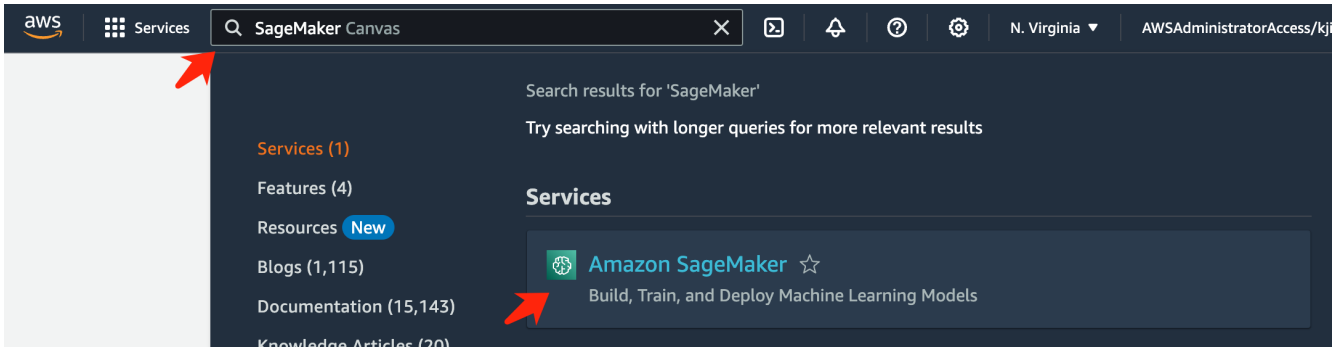
FSx ONTAPの詳細については、こちらのプレゼンテーションをご覧ください（"ビデオリンク"）

ユーザーガイド

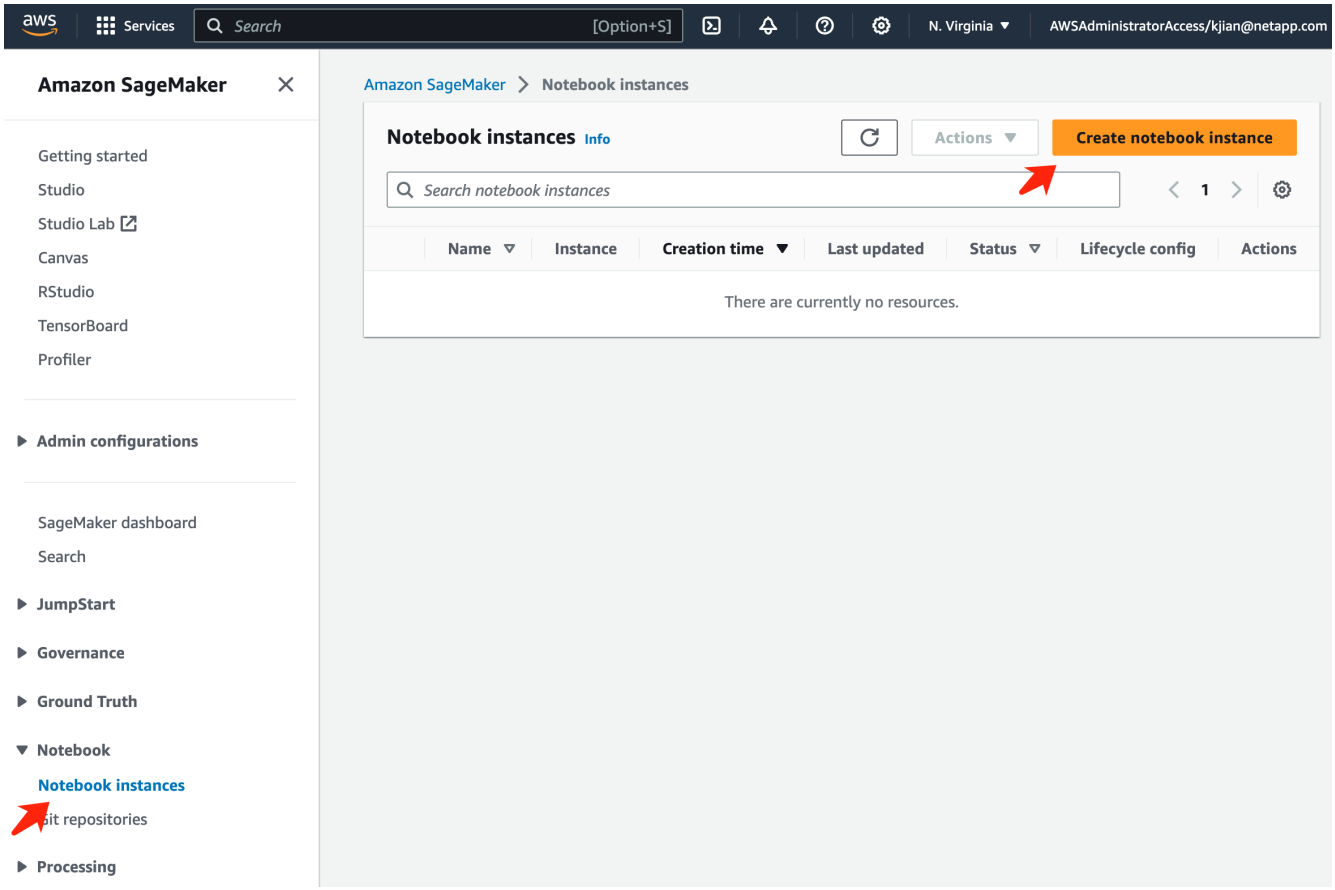
サーバの作成

SageMakerノートブックインスタンスの作成

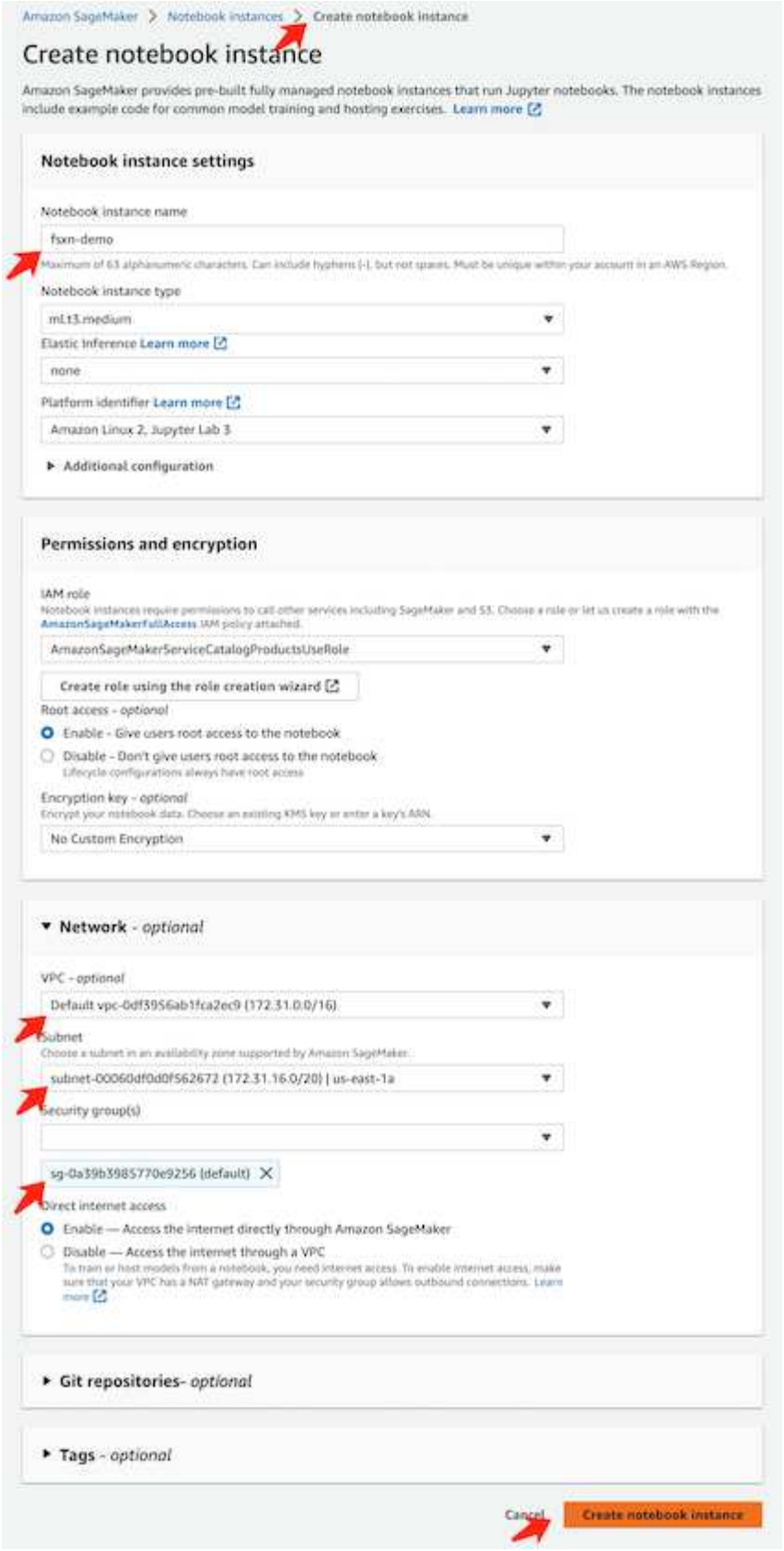
1. AWSコンソールを開きます。検索パネルで、SageMakerを検索し、サービス* Amazon SageMaker *をクリックします。



2. Notebookタブの* Notebook Instances を開き、オレンジ色の Create notebook instance *ボタンをクリックします。

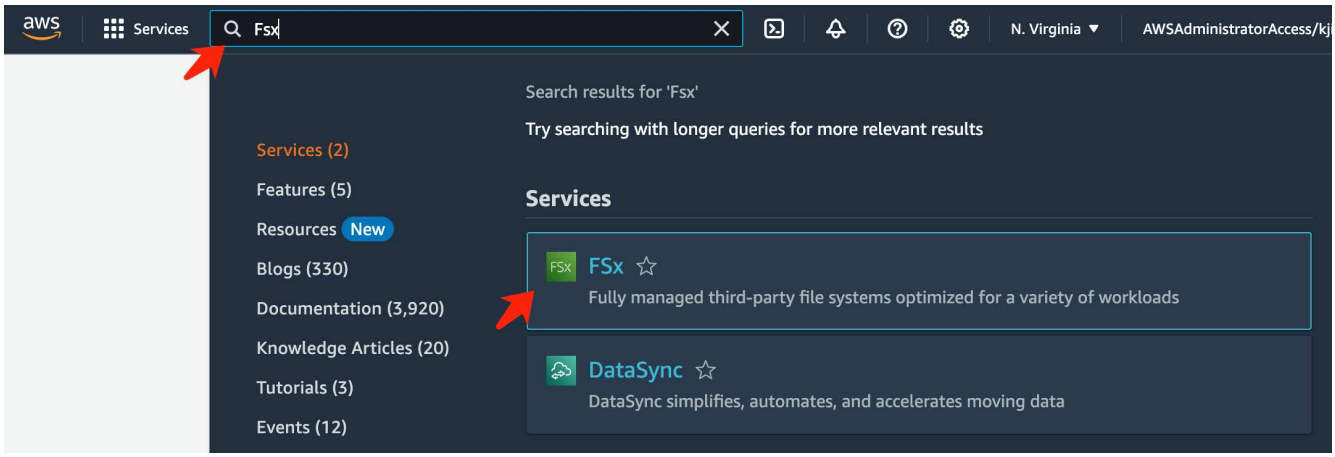


3. 作成ページで、ノートブックインスタンス名*を入力します。*ネットワーク*パネルを展開し、その他のエントリをデフォルトのままにして、 VPC*、サブネット、セキュリティグループ*を選択します。（このVPC と Subnet は後で**FSx ONTAP**ファイルシステムの作成に使用します）右下のオレンジ色のボタン[Create notebook instance]*をクリックします。

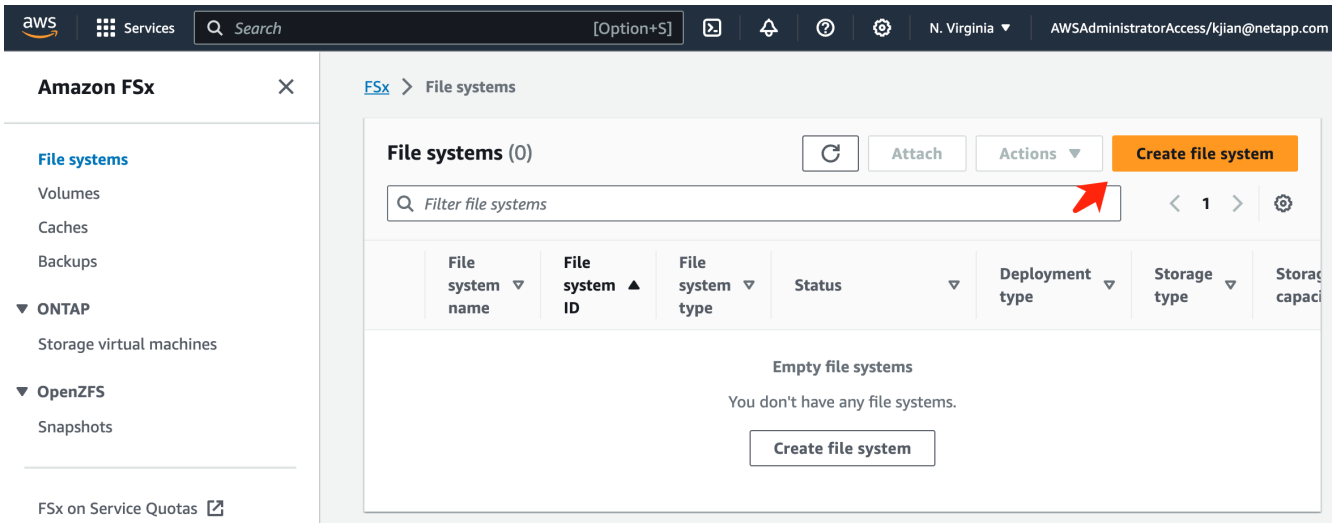


FSx ONTAPファイルシステムの作成

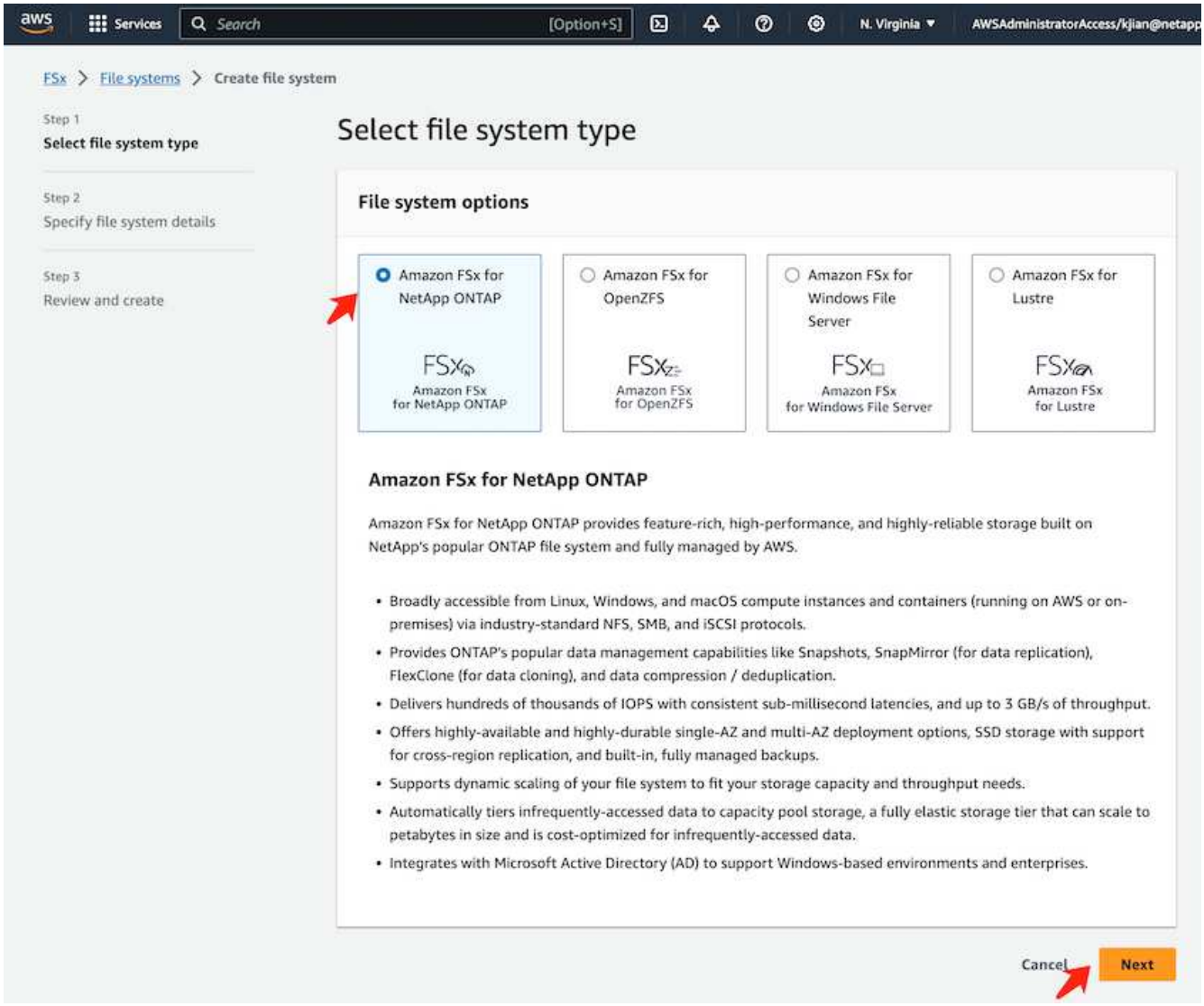
1. AWSコンソールを開きます。検索パネルでFSxを検索し、サービス* FSX *をクリックします。



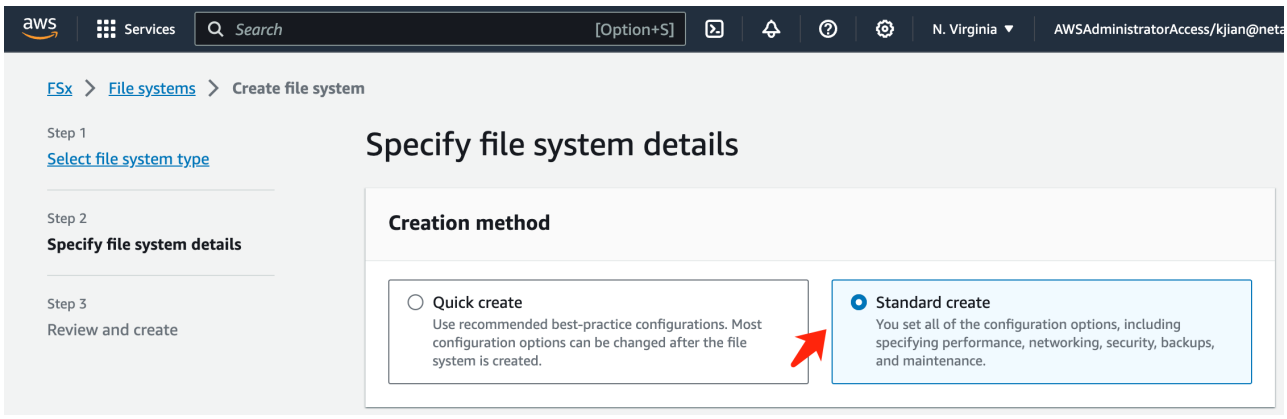
2. [ファイルシステムの作成]*をクリックします。



3. 最初のカード* FSx ONTAP を選択し、 Next *をクリックします。

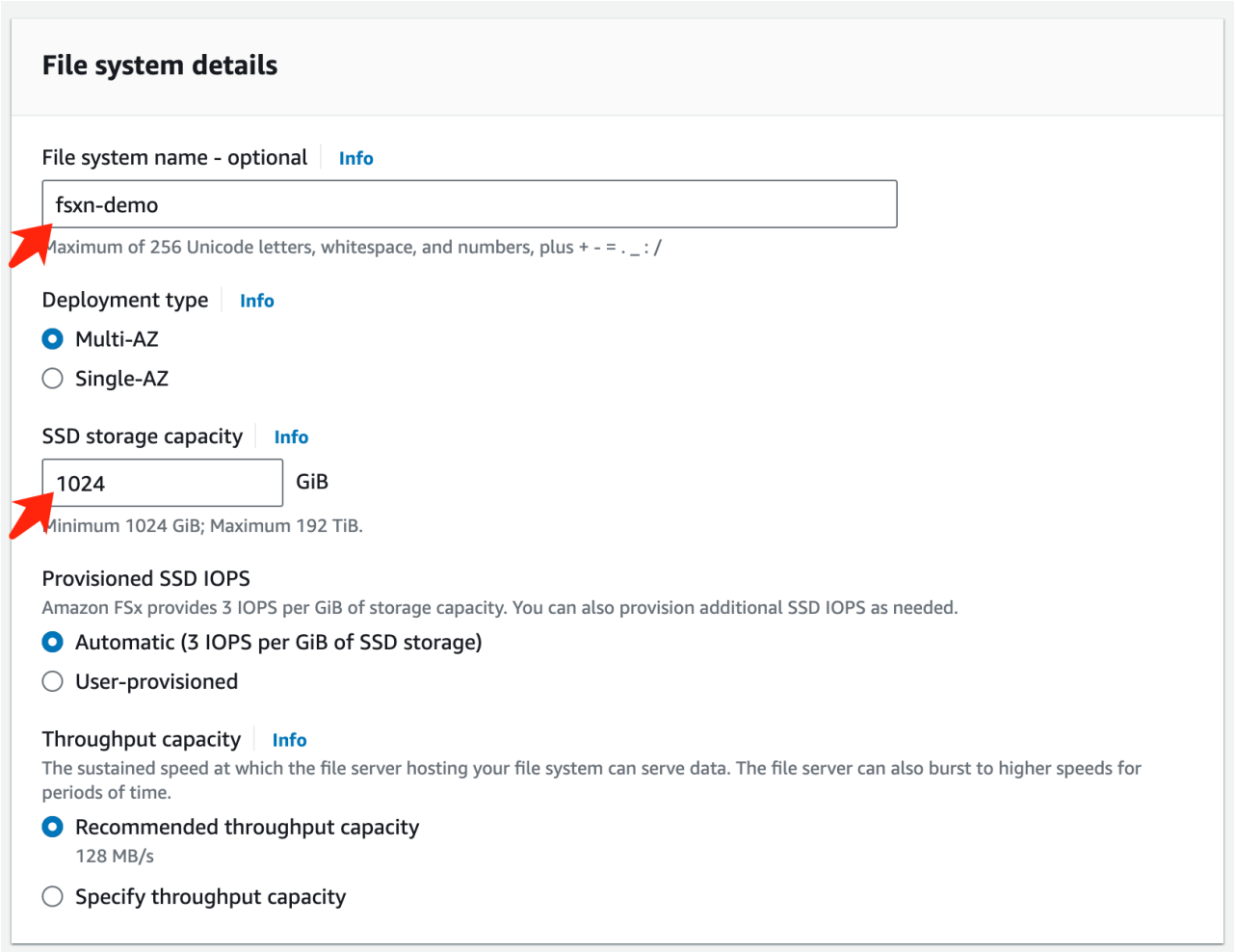


4. をクリックします。
   a. [標準作成（Standard create）]*オプションを選択します。

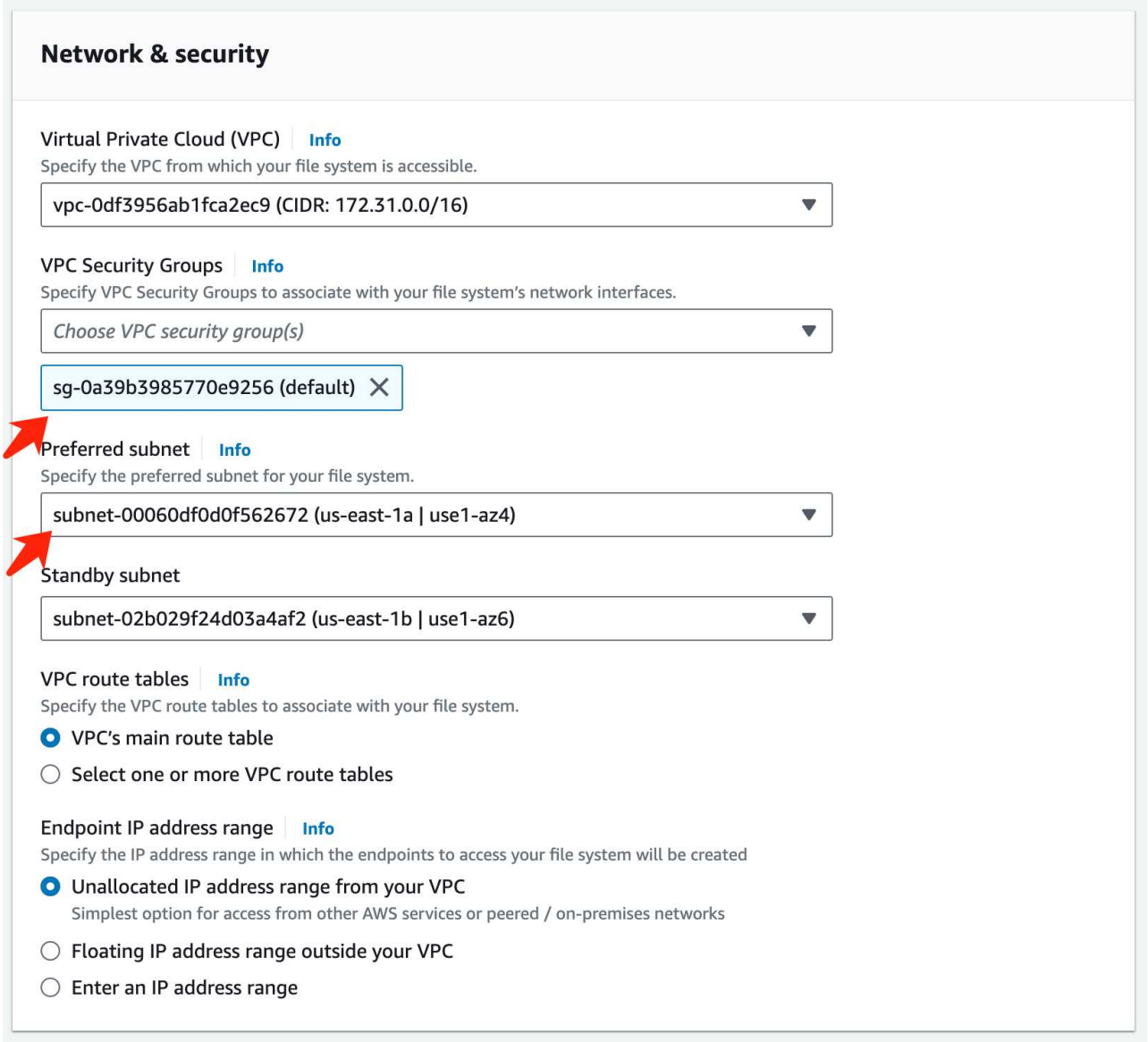


パネル"]

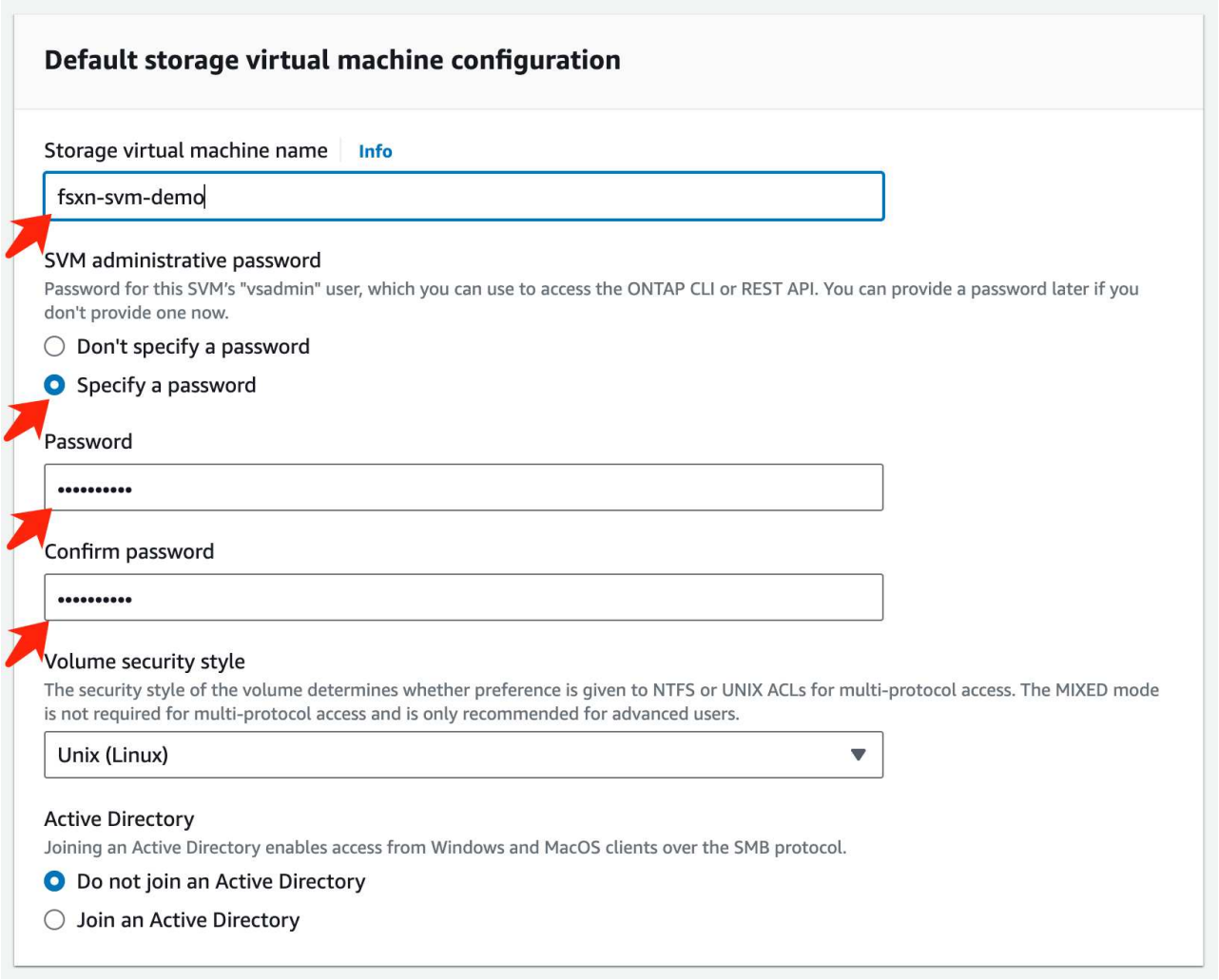
   b. [File system name]*と[SSD storage capacity]*を入力します。



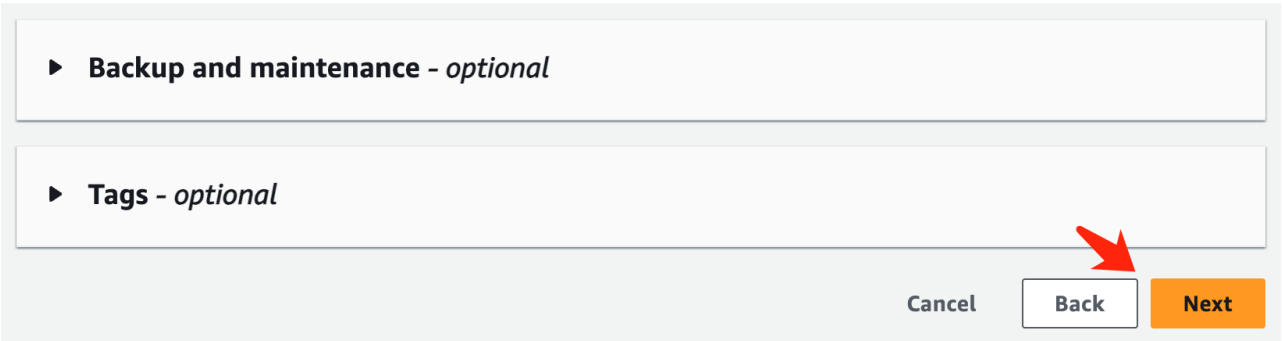
c. 必ず* VPC と*サブネット*を SageMaker Notebook *インスタンスと同じにしてください。



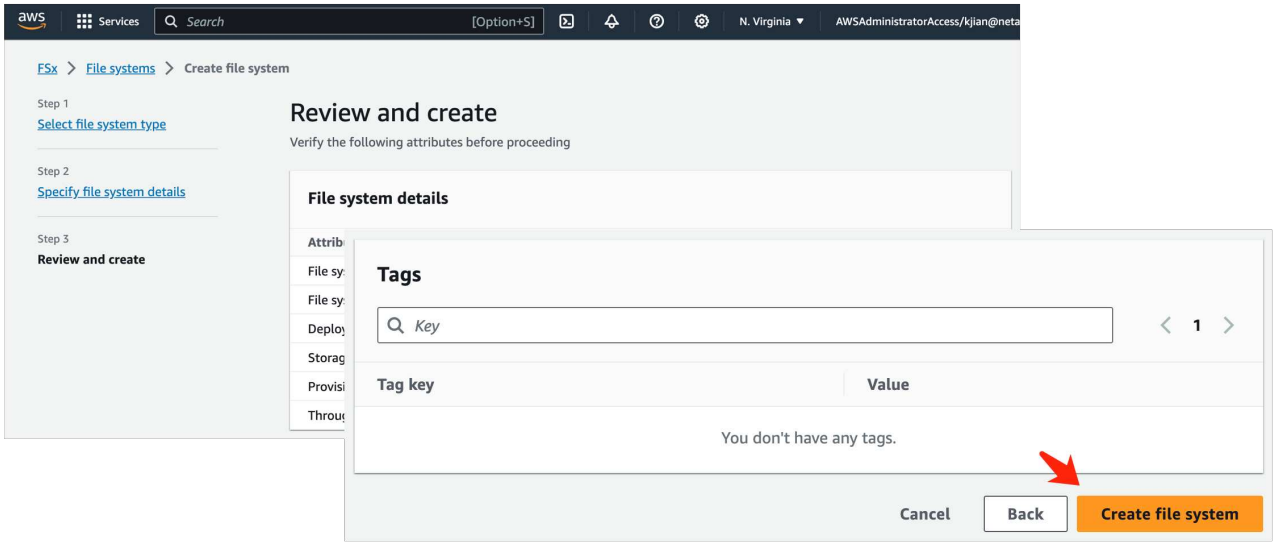
d. Storage Virtual Machine *の名前を入力し、SVM（Storage Virtual Machine）の*パスワードを*指定してください。



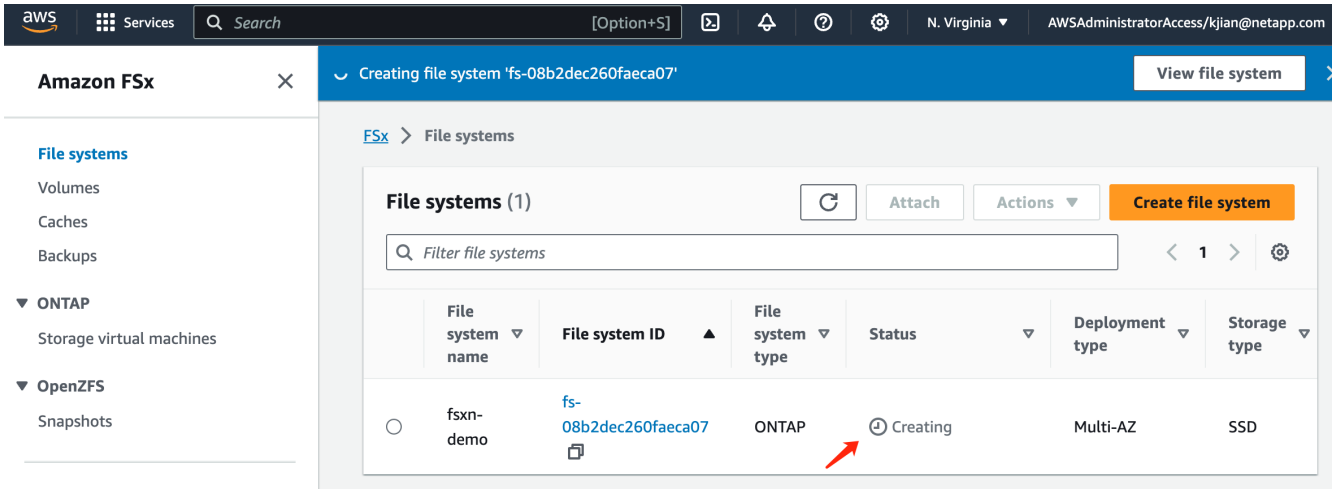
e. [その他のエントリ]はデフォルトのままにして、右下のオレンジ色のボタン*[次へ]*をクリックします。



f. レビューページの右下にあるオレンジ色の*ファイルシステムの作成*ボタンをクリックします。



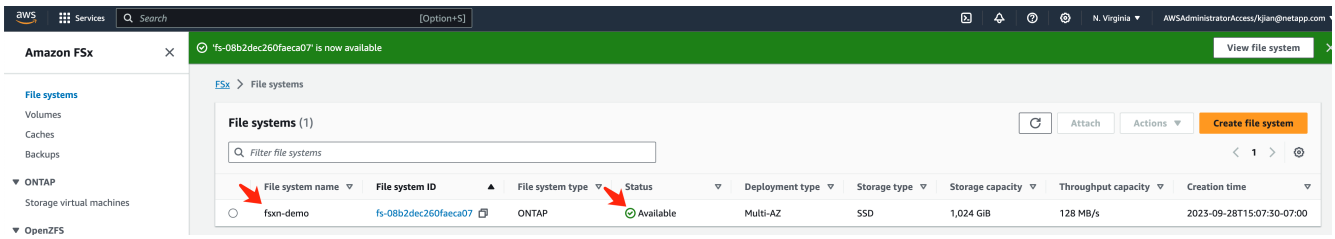
5. FSxファイルシステムのスピンアップには約* 20~40分*かかる場合があります。



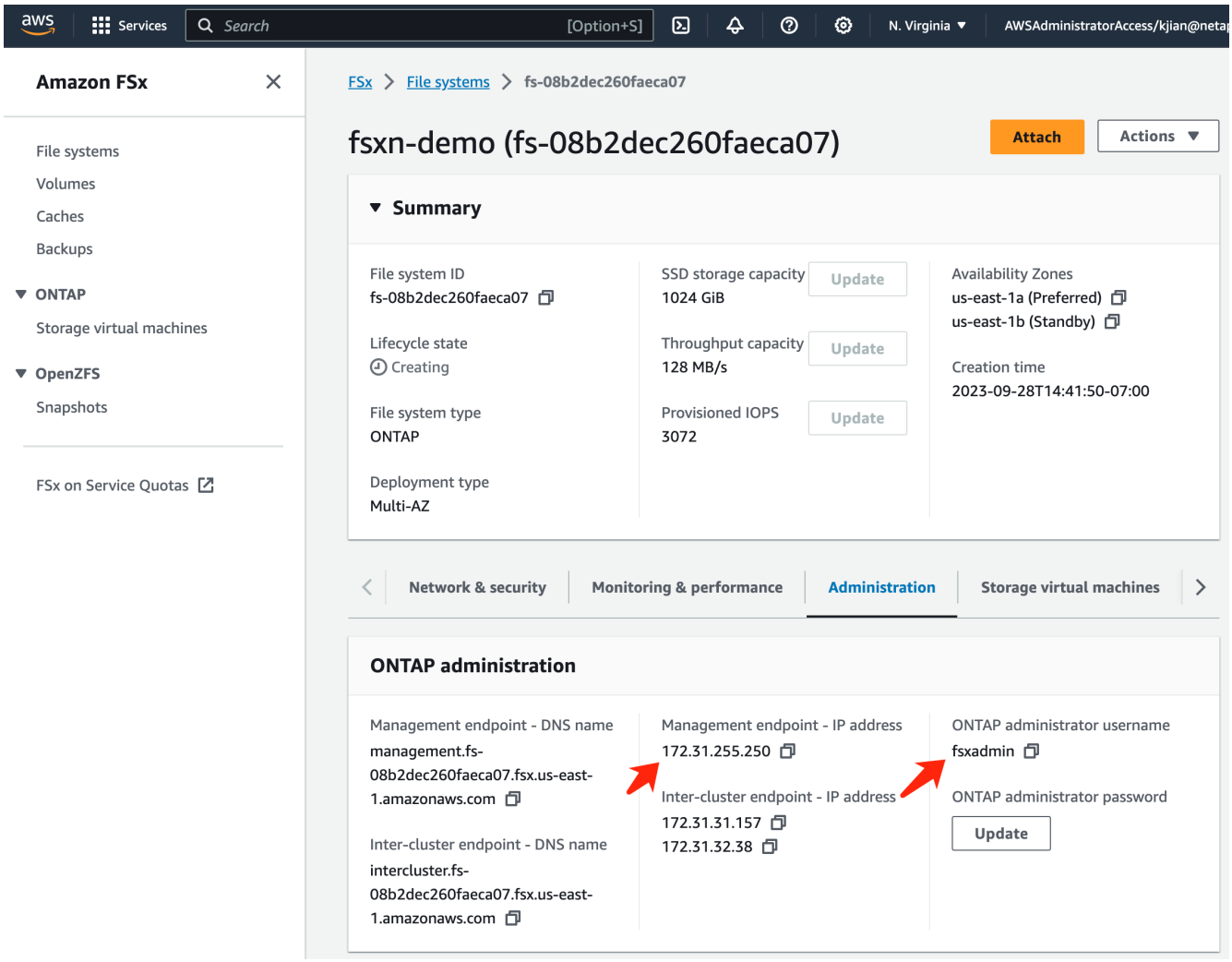
サーバの設定

ONTAPの設定

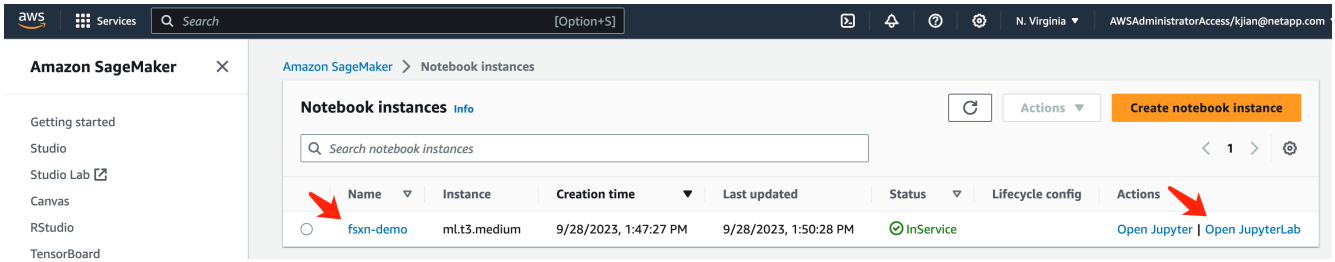
1. 作成したFSxファイルシステムを開きます。ステータスが*利用可能*であることを確認してください。



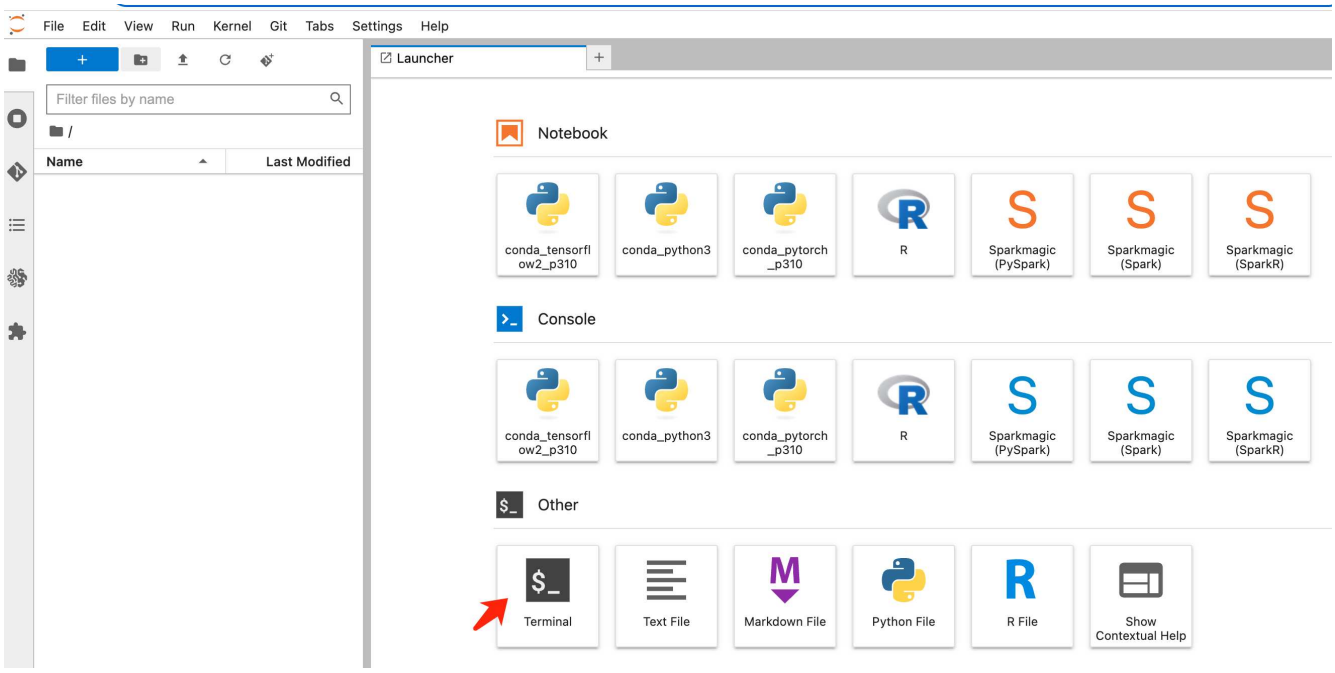
2. [管理]タブを選択し、**[管理エンドポイント- IP**アドレス]*と[ONTAP管理者のユーザー名]*のままにします。



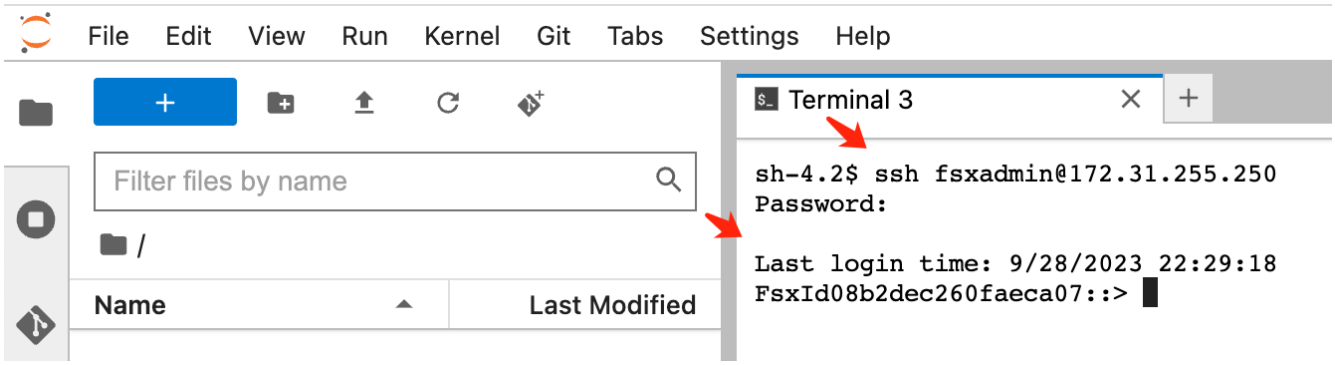
3. 作成した* SageMaker Notebookインスタンス*を開き、*[JupyterLab]*をクリックします。



4. Jupyter Labページで、新しい*ターミナル*を開きます。



5. sshコマンドssh < admin user name >@< ONTAP server IP >を入力し、FSx ONTAPファイルシステムにログインします。（ユーザ名とIPアドレスは手順2で取得）* Storage Virtual Machine *の作成時に使用したパスワードを使用してください。



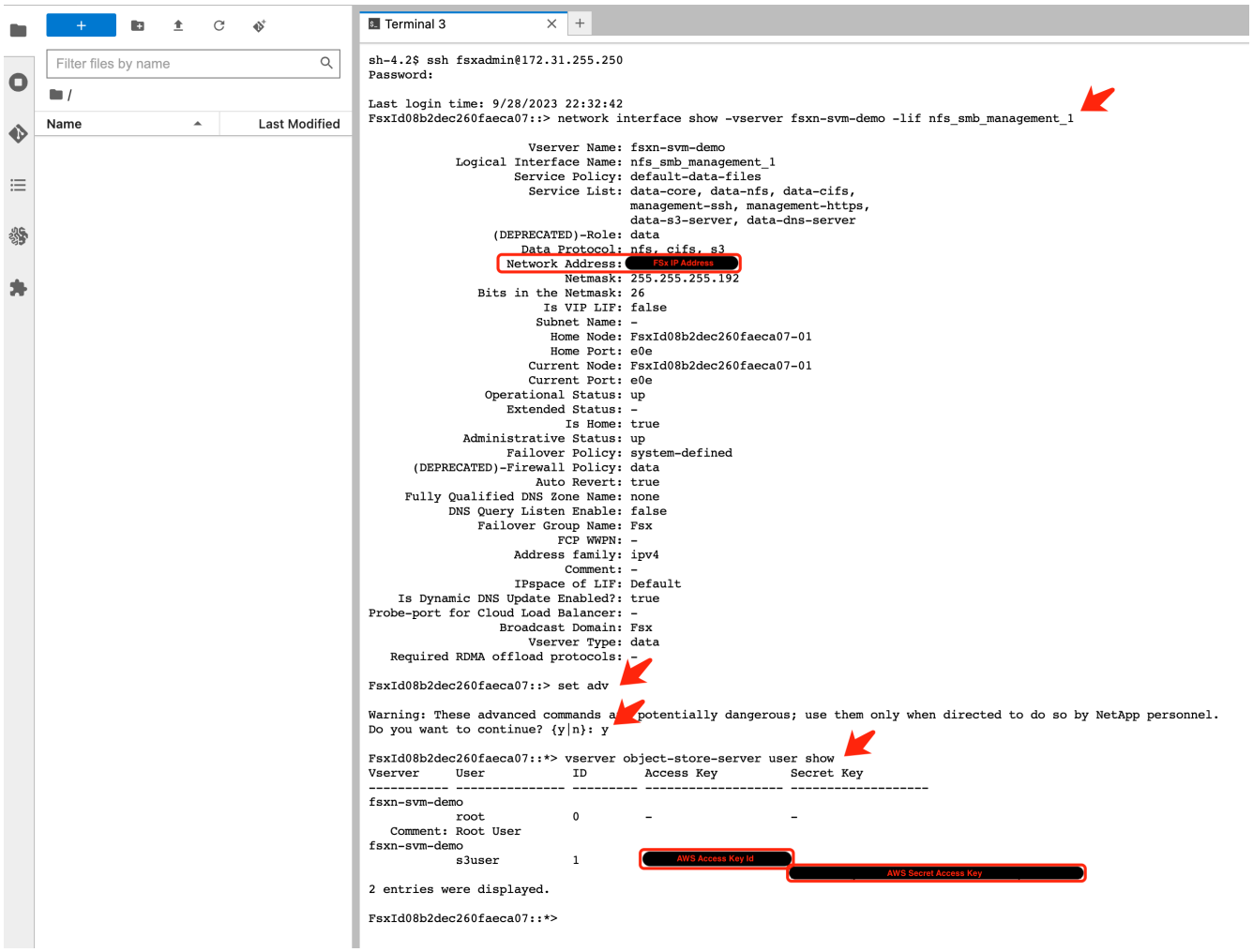
6. 次の順序でコマンドを実行します。FSx ONTAPプライベートS3バケット名*の名前には* fsxn- ONTAP を使用します。**SVM** *引数には Storage Virtual Machine名*を使用してください。

```
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. 以下のコマンドを実行して、FSx ONTAPプライベートS3のエンドポイントIPとクレデンシャルを取得します。

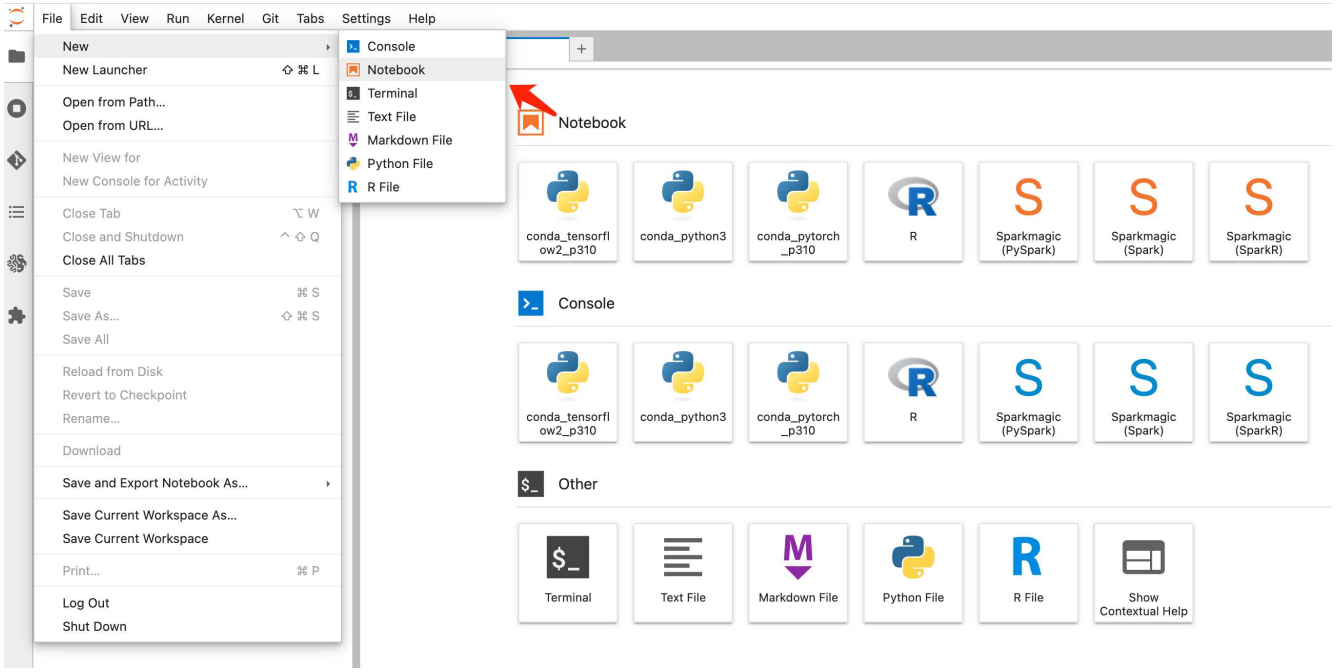
```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show
```

8. あとで使用できるように、エンドポイントのIPとクレデンシャルを保持します。



クライアント設定

1. SageMaker Notebookインスタンスで、新しいJupyterノートブックを作成します。



2. FSx ONTAPプライベートS3バケットにファイルをアップロードする回避策として、以下のコードを使用してください。包括的なコード例については、このノートブックを参照してください。"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']
```

これで、FSx ONTAPとSageMakerインスタンスの統合は終了です。

便利なデバッグチェックリスト

- SageMaker NotebookインスタンスとFSx ONTAPファイルシステムが同じVPC内にあることを確認します。

- ONTAPで* set dev コマンドを実行して、特権レベルを dev *に設定することを忘れないでください。

FAQ（2023年9月27日現在）

Q：FSX ONTAPにファイルをアップロードするときに、CreateMultipartUpload操作を呼び出したときに「エラーが発生しました（**NotImplemented**）：要求した**s3**コマンドが実装されていません」というエラーが表示されるのはなぜですか？

A：プライベートS3バケットとして、FSx ONTAPは最大100MBのファイルのアップロードをサポートしています。S3プロトコルを使用する場合、100MBを超えるファイルは100MBのチャンクに分割され、「CreateMultipartUpload」関数が呼び出されます。ただし、FSx ONTAPプライベートS3の現在の実装では、この機能はサポートされていません。

Q: FSx ONTAPにファイルをアップロードする際に、「* PutObject操作を呼び出したときにエラーが発生しました(AccessDenied)」というエラーが表示されるのはなぜですか?

A：SageMaker NotebookインスタンスからFSx ONTAPプライベートS3バケットにアクセスするには、AWSのクレデンシャルをFSx ONTAPのクレデンシャルに切り替えます。ただし、インスタンスに書き込み権限を付与するには、バケットをマウントし、「chmod」シェルコマンドを実行して権限を変更する 回避策 解決策が必要です。

Q：FSx ONTAPプライベートS3バケットを他のSageMaker MLサービスと統合するにはどうすればよいですか。

A:残念ながら、SageMakerサービスSDKは、プライベートS3バケットのエンドポイントを指定する方法を提供していません。そのため、FSx ONTAP S3はSagemaker Data Wrangler、Sagemaker Clarify、Sagemaker Glue、Sagemaker Athena、Sagemaker AutoMLなどのSageMakerサービスと互換性がありません。

パート**2 - SageMaker**でのモデルトレーニングのデータソースとして**AWS Amazon FSx for NetApp ONTAP**（**FSx ONTAP**）を活用

この記事は、特にタイヤ品質分類プロジェクトのために、SageMakerでPyTorchモデルをトレーニングするためにAmazon FSx for NetApp ONTAP（FSx ONTAP）を使用するためのチュートリアルです。

作成者：
NetAppシニアデータ&アプリケーションサイエンティスト、Jian Jian（Ken）

はじめに

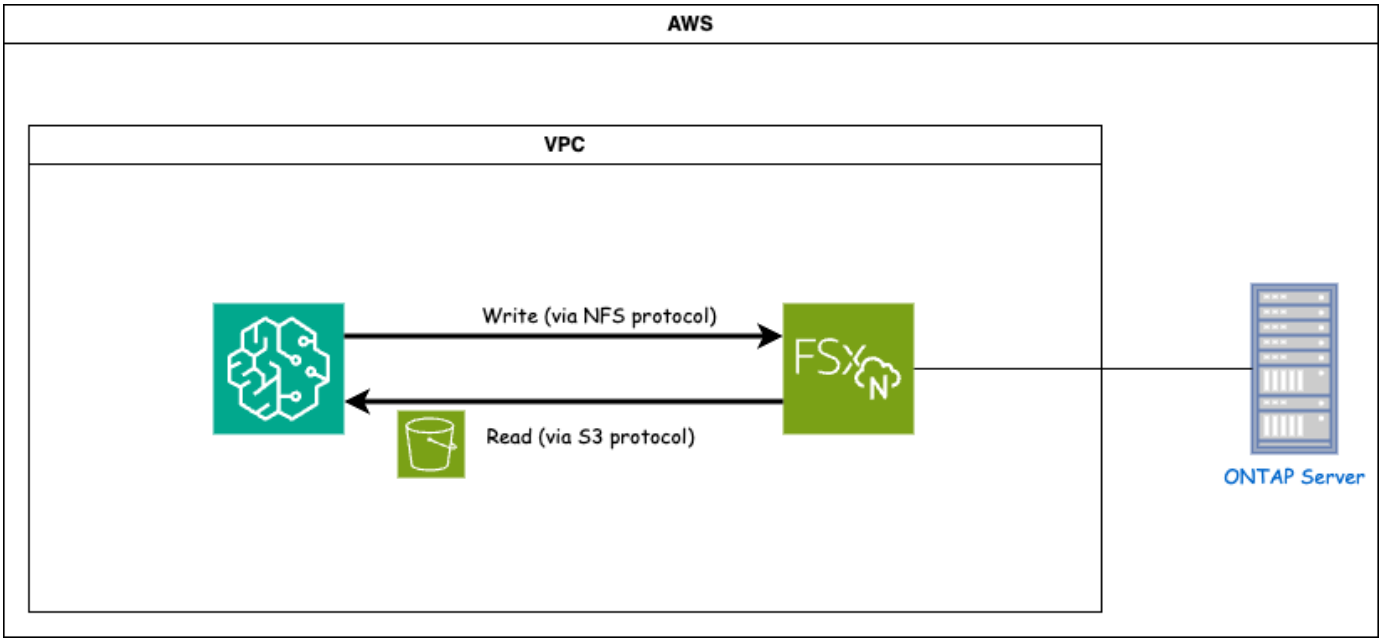
このチュートリアルでは、コンピュータビジョン分類プロジェクトの実践的な例を紹介します。FSx ONTAPをSageMaker環境内のデータソースとして使用するMLモデルを構築するための実践的な経験を提供します。このプロジェクトでは、ディープラーニングフレームワークであるPyTorchを使用して、タイヤの画像に基づいてタイヤの品質を分類することに焦点を当てています。Amazon SageMakerのデータソースとしてFSx ONTAPを使用した機械学習モデルの開発に重点を置いています。

FSx ONTAPとは

Amazon FSx ONTAPは、AWSが提供するフルマネージドのストレージソリューションです。ネットアップのONTAPファイルシステムを活用して、信頼性の高いハイパフォーマンスストレージを提供します。NFS、SMB、iSCSIなどのプロトコルをサポートしているため、さまざまなコンピューティングインスタンスやコンテナからシームレスにアクセスできます。このサービスは、卓越したパフォーマンスを提供し、高速かつ効率的なデータ運用を実現するように設計されています。また、高可用性とデータ保持性を実現し、データへのアクセスと保護を維持します。さらに、Amazon FSx ONTAPのストレージ容量は拡張性に優れているため、ニーズに合わせて簡単に調整できます。

前提条件

ネットワーク環境



FSx ONTAP（Amazon FSx ONTAP）は、AWSのストレージサービスです。これには、NetApp ONTAPシステムで実行されているファイルシステムと、接続するAWSで管理されるSystem Virtual Machine（SVM）が含まれます。次の図では、AWSで管理されるNetApp ONTAPサーバがVPCの外部に配置されています。SVMはSageMakerとNetApp ONTAPシステムの仲介役として機能し、SageMakerからの処理要求を受け取り、基盤となるストレージに転送します。FSx ONTAPにアクセスするには、SageMakerがFSx ONTAP環境と同じVPC内に配置されている必要があります。この構成により、SageMakerとFSx ONTAP間の通信とデータアクセスが保証されます。

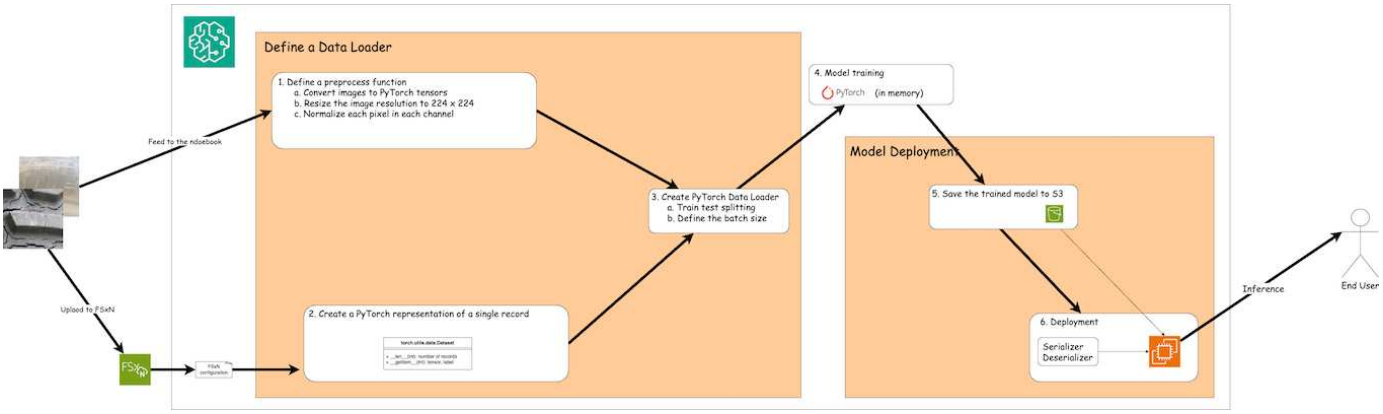
データアクセス

実際のシナリオでは、データサイエンティストは通常、FSx ONTAPに保存されている既存のデータを利用して機械学習モデルを構築します。ただし、デモ目的では、FSx ONTAPファイルシステムは作成後に最初は空になるため、トレーニングデータを手動でアップロードする必要があります。これは、FSx ONTAPをボリュームとしてSageMakerにマウントすることで実現できます。ファイルシステムが正常にマウントされたら、データセットをマウントされた場所にアップロードして、SageMaker環境内でモデルをトレーニングするためにアクセスできるようにすることができます。このアプローチでは、FSx ONTAPのストレージ容量と機能を活用しながら、SageMakerと連携してモデルの開発とトレーニングを行うことができます。

データ読み取りプロセスでは、FSx ONTAPをプライベートS3バケットとして設定します。詳細な設定手順については、を参照してください。"パート1 - Amazon FSx for NetApp ONTAP（FSx ONTAP）をプライベー

トS3バケットとしてAWS SageMakerに統合する"

統合の概要



FSx ONTAPのトレーニングデータを使用してSageMakerでディープラーニングモデルを構築するワークフローは、主に3つのステップ（Data Loaderの定義、モデルのトレーニング、導入）に要約できます。大まかに言えば、これらのステップはMLOpsパイプラインの基盤を形成します。ただし、各ステップには、包括的な実装のためのいくつかの詳細なサブステップが含まれています。これらのサブステップには、データの前処理、データセットの分割、モデルの構成、ハイパーパラメータの調整、モデルの評価など、さまざまなタスクが含まれます。 モデルの導入を支援します。これらの手順により、SageMaker環境内でFSx ONTAPのトレーニングデータを使用してディープラーニングモデルを構築し、導入するための徹底的で効果的なプロセスが保証されます。

ステップバイステップの統合

データローダ

PyTorchディープラーニングネットワークをデータでトレーニングするために、データのフィードを容易にするためのデータローダーが作成されます。データローダーは、バッチサイズを定義するだけでなく、バッチ内の各レコードを読み取って前処理するための手順も決定します。データローダーを構成することで、データの処理をバッチで処理し、ディープラーニングネットワークのトレーニングを可能にします。

データローダーは3つの部分で構成されています。

前処理機能

```
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]
    )
])
```

上記のコードスニペットは、**torchvision.transforms***モジュールを使用した画像前処理変換の定義を示しています。この**turtorial**では、一連の変換を適用するためにプリプロセスオブジェクトが作成されます。まず、***ToTensor()***変換は画像をテンソル表現に変換する。その後、***Resize(224,224***変換により、画像のサイズが**224x224**ピクセルの固定サイズに変更されます。最後に、 Normalize()*変換は、平均を減算し、各チャンネルに沿った標準偏差で割ることによってテンソル値を正規化します。正規化に使用される平均値と標準偏差値は、事前にトレーニングされたニューラルネットワークモデルで一般的に使用されます。全体的に、このコードは、画像データをテンソルに変換し、サイズを変更し、ピクセル値を正規化することで、事前にトレーニングされたモデルにさらに処理または入力できるように準備します。

PyTorchデータセットクラス

```
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
```

このクラスは、データセット内のレコードの総数を取得する機能を提供し、各レコードのデータを読み取る方法を定義します。***getItem****関数内で、コードは**boto3 S3**バケットオブジェクトを利用して、**FSx ONTAP**からバイナリデータを取得します。**FSx ONTAP**からデータにアクセスするためのコードスタイルは、**Amazon S3**からデータを読み取るのと似ています。以降の説明では、プライベート**S3**オブジェクト Bucket *の作成プロセスについて詳しく説明します。

```
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 ---
```

SageMakerでFSx ONTAPからデータを読み取るために、S3プロトコルを使用してFSx ONTAPストレージを指すハンドラが作成されます。これにより、FSx ONTAPをプライベートS3バケットとして扱うことができます。ハンドラの設定には、FSx ONTAP SVMのIPアドレス、バケット名、および必要なクレデンシャルの指定が含まれます。これらの設定項目の入手方法については、のドキュメントを参照してください"パート1 - Amazon FSx for NetApp ONTAP（FSx ONTAP）をプライベートS3バケットとしてAWS SageMakerに統合する"。

前述の例では、Bucketオブジェクトを使用してPyTorchデータセットオブジェクトをインスタンス化しています。データセットオブジェクトについては、次のセクションで詳しく説明します。

PyTorchデータローダ

```
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)
```

この例では、64のバッチサイズが指定されています。これは、各バッチに64レコードが含まれることを示しています。PyTorch * Dataset *クラス、前処理関数、およびトレーニングバッチサイズを組み合わせることで、トレーニング用のデータローダーを取得します。このデータローダーは、トレーニングフェーズ中にデータセットをバッチで反復処理するプロセスを容易にします。

モデルトレーニング

```
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')
```

このコードは標準のPyTorchトレーニングプロセスを実装しています。これは、畳み込み層と線形層を使用してタイヤの品質を分類する*TireQualityClassifier*と呼ばれるニューラルネットワークモデルを定義します。トレーニングループはデータバッチを繰り返し、損失を計算し、バックプロパゲーションと最適化を使用してモデルのパラメータを更新します。さらに、現在の時刻、エポック、バッチ、および損失を監視するために印刷します。

モデルの導入

導入

```
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())
```

このコードはPyTorchモデルを* Amazon S3 に保存します。これは、**SageMaker**が展開するためにモデルを**S3**に格納する必要があるためです。モデルを Amazon S3 *にアップロードすることで、SageMakerからアクセスできるようになり、デプロイされたモデルでのデプロイと推論が可能になります。

```
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
)
```

このコードは、SageMakerへのPyTorchモデルのデプロイを容易にします。これは、入力データをPyTorchテンソルとして前処理してシリアライズするカスタムシリアライザ*TireQualitySerializer*を定義します。**TireQualityPredictor***クラスは、定義されたシリアライザと***JSONDeserializer***を利用するカスタムプレディクタです。コードはまた、モデルの**S3**の場所、**IAM**の役割、フレームワークのバージョン、推論のエントリポイントを指定する PyTorchModel *オブジェクトを作成します。コードはタイムスタンプを生成し、モデ

ルとタイムスタンプに基づいてエンドポイント名を構築します。最後に、インスタンス数、インスタンスタイプ、生成されたエンドポイント名を指定して、deployメソッドを使用してモデルをデプロイします。これにより、PyTorchモデルをデプロイし、SageMakerで推論できるようになります。

推論

```
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)
```

次の例では、導入したエンドポイントを使用して推論を実行しています。

パート3 -簡易化されたMLOpsパイプラインの構築（CI/CT/CD）

この記事では、自動モデル再トレーニング、導入、コスト最適化に焦点を当てて、AWSサービスを使用してMLOpsパイプラインを構築するためのガイドを提供します。

- - -

作成者：
NetAppシニアデータ&アプリケーションサイエンティスト、Jian Jian（Ken）

はじめに

このチュートリアルでは、さまざまなAWSサービスを活用して、継続的統合（CI）、継続的トレーニング（CT）、継続的導入（CD）を含むシンプルなMLOpsパイプラインを構築する方法を学習します。従来のDevOpsパイプラインとは異なり、MLOpsでは運用サイクルを完了するために追加の考慮事項が必要です。このチュートリアルに従うことで、CTをMLOpsループに組み込む方法についての洞察を得ることができ、モデルの継続的なトレーニングと推論のためのシームレスな導入が可能になります。このチュートリアルでは、AWSサービスを利用してこのエンドツーエンドのMLOpsパイプラインを確立するプロセスをガイドします。

マニフェスト

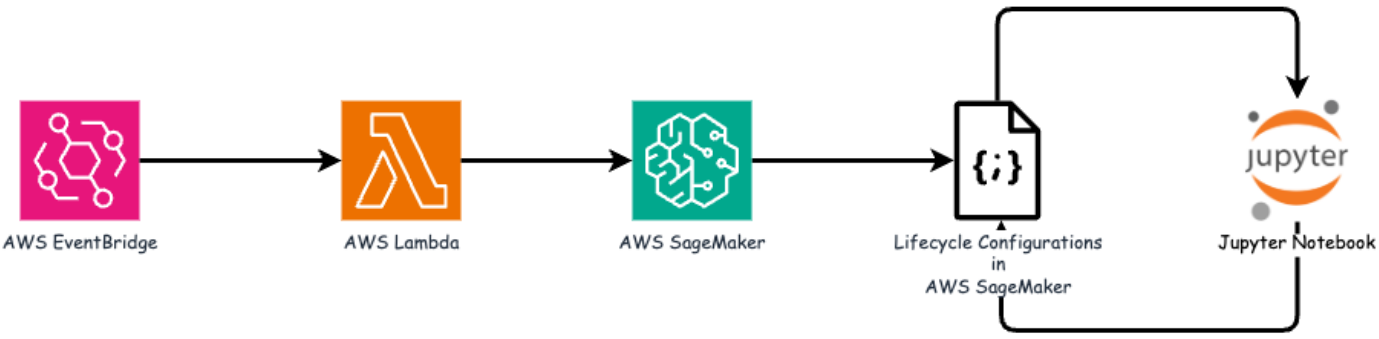
機能性	名前	コメント（ **Comment** ）
データストレージ	AWS FSX ONTAP の略	を参照してください "パート1 - Amazon FSx for NetApp ONTAP（FSx ONTAP）をプライベートS3バケットとしてAWS SageMakerに統合する"。

機能性	名前	コメント（ **Comment** ）
データサイエンスIDE	AWS SageMaker	このチュートリアルは、に示すJupyterノートブックに基づいて"パート2 - SageMakerのモデルトレーニングのデータソースとしてAmazon FSx for NetApp ONTAP（FSx ONTAP）を活用"います。
MLOpsパイプラインをトリガーする機能	AWS Lambda関数	-
cronジョブトリガー	AWSイベントブリッジ	-
ディープラーニングフレームワーク	PyTorch	-
AWS Python SDK	ボット3	-
プログラミング言語	Python	v3.10

前提条件

- 構成済みのFSx ONTAPファイルシステム。このチュートリアルでは、FSx ONTAPに保存されているデータをトレーニングプロセスに利用します。
- 前述のFSx ONTAPファイルシステムと同じVPCを共有するように設定された* SageMakerノートブックインスタンス*。
- * AWS Lambda関数*をトリガーする前に、* SageMaker Notebookインスタンス*が*停止*ステータスになっていることを確認してください。
- ディープニューラルネットワークの計算に必要なGPUアクセラレーションを利用するには、* ml.g4dn.xlarge *インスタンスタイプが必要です。

アーキテクチャ



このMLOpsパイプラインは、cronジョブを利用してサーバレス関数をトリガーし、ライフサイクルコールバック関数に登録されたAWSサービスを実行する実用的な実装です。AWS EventBridge は**cron**ジョブとして機能します。モデルの再トレーニングと再デプロイを担当する AWS Lambda関数*を定期的に呼び出します。このプロセスでは、* AWS SageMaker Notebook *インスタンスをスピンアップして必要なタスクを実行します。

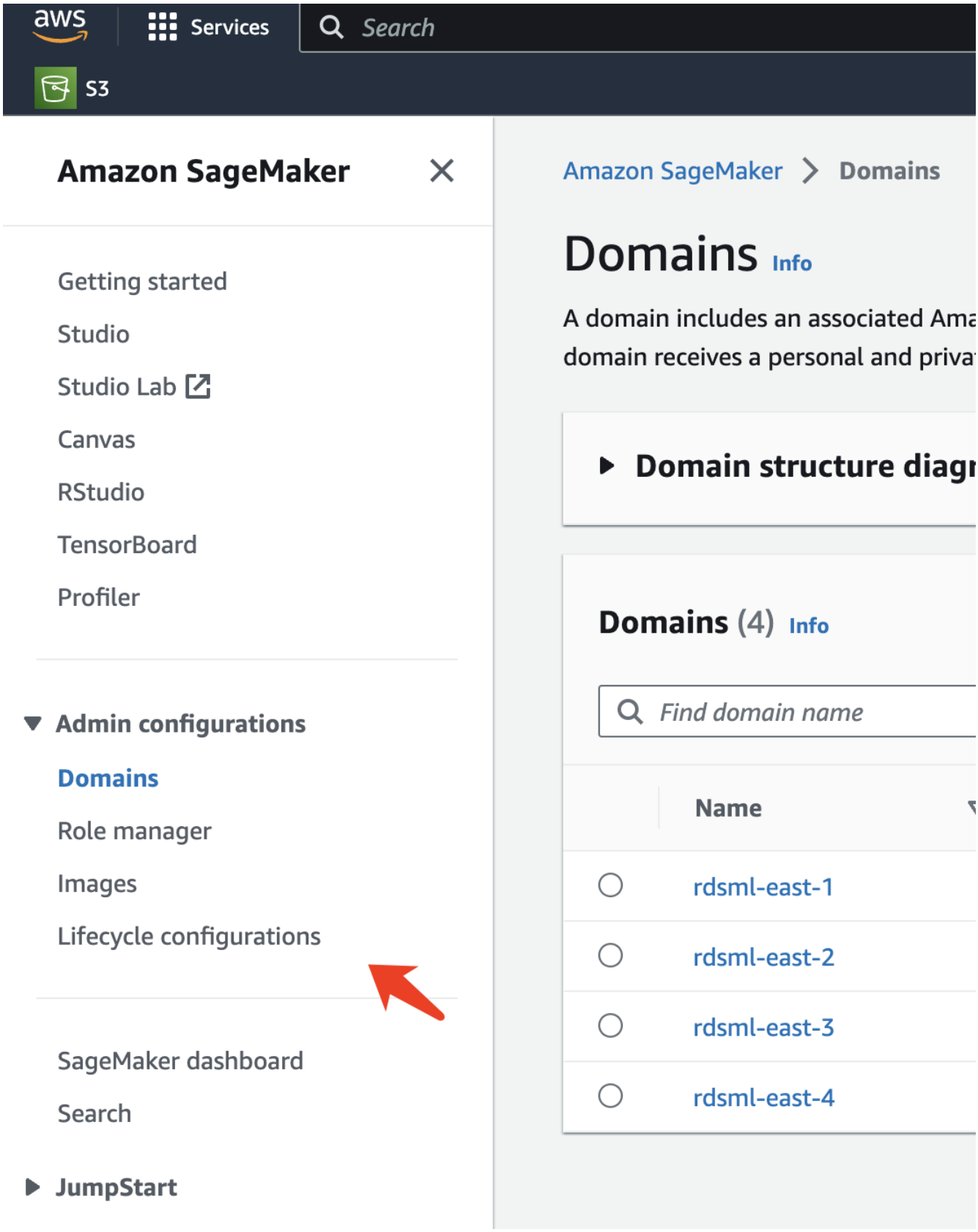
ステップバイステップ構成

ライフサイクル設定

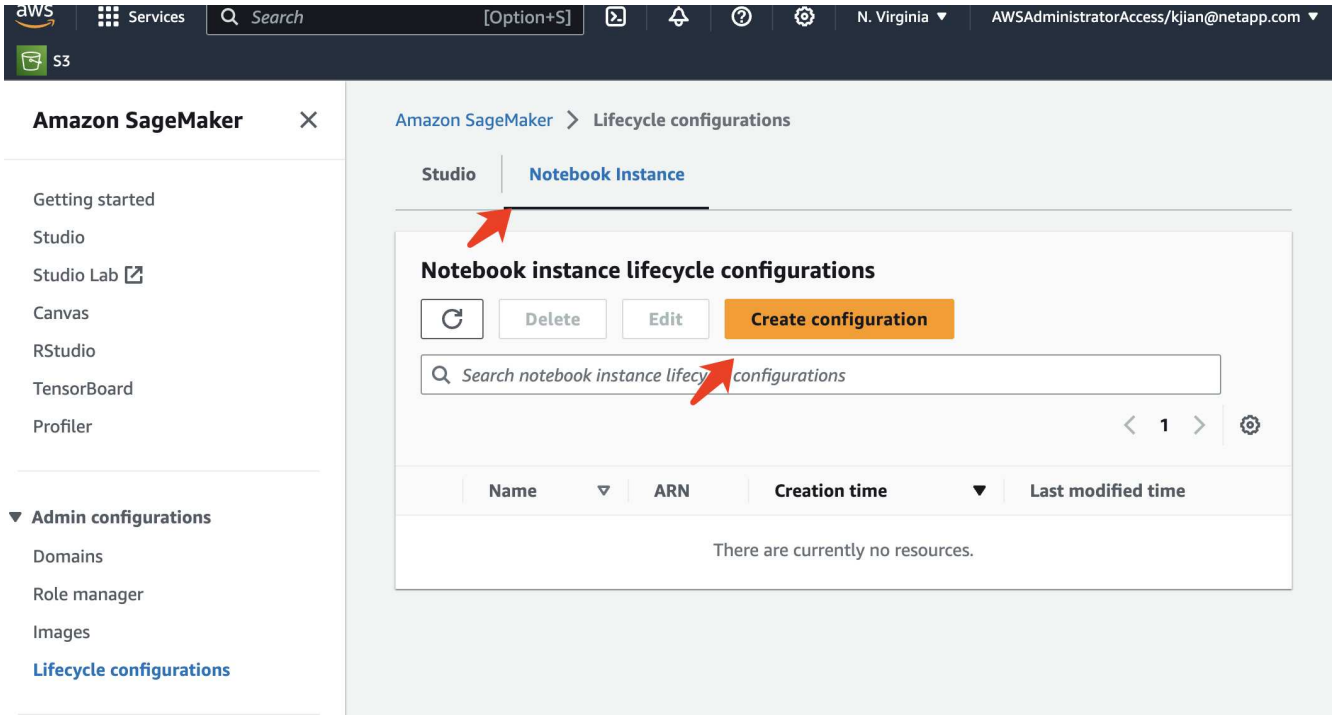
AWS SageMaker Notebookインスタンスのライフサイクルコールバック関数を設定するには、ライフサイクル設定*を使用します。このサービスでは、ノートブックインスタンスをスピンアップするときに実行する必要のあるアクションを定義できます。具体的には、 Lifecycle configurations *内にシェルスクリプトを実装して、トレーニングおよび展開プロセスが完了するとノートブックインスタンスを自動的にシャットダウンすることができます。MLOpsではコストが重要な考慮事項の1つであるため、これは必須の設定です。

重要なのは、*ライフサイクル構成*の構成は事前に設定する必要があることです。したがって、他のMLOpsパイプラインのセットアップに進む前に、この側面の設定を優先することをお勧めします。

1. ライフサイクル構成を設定するには、* Sagemaker パネルを開き、 Admin configurations セクションの Lifecycle configurations *に移動します。



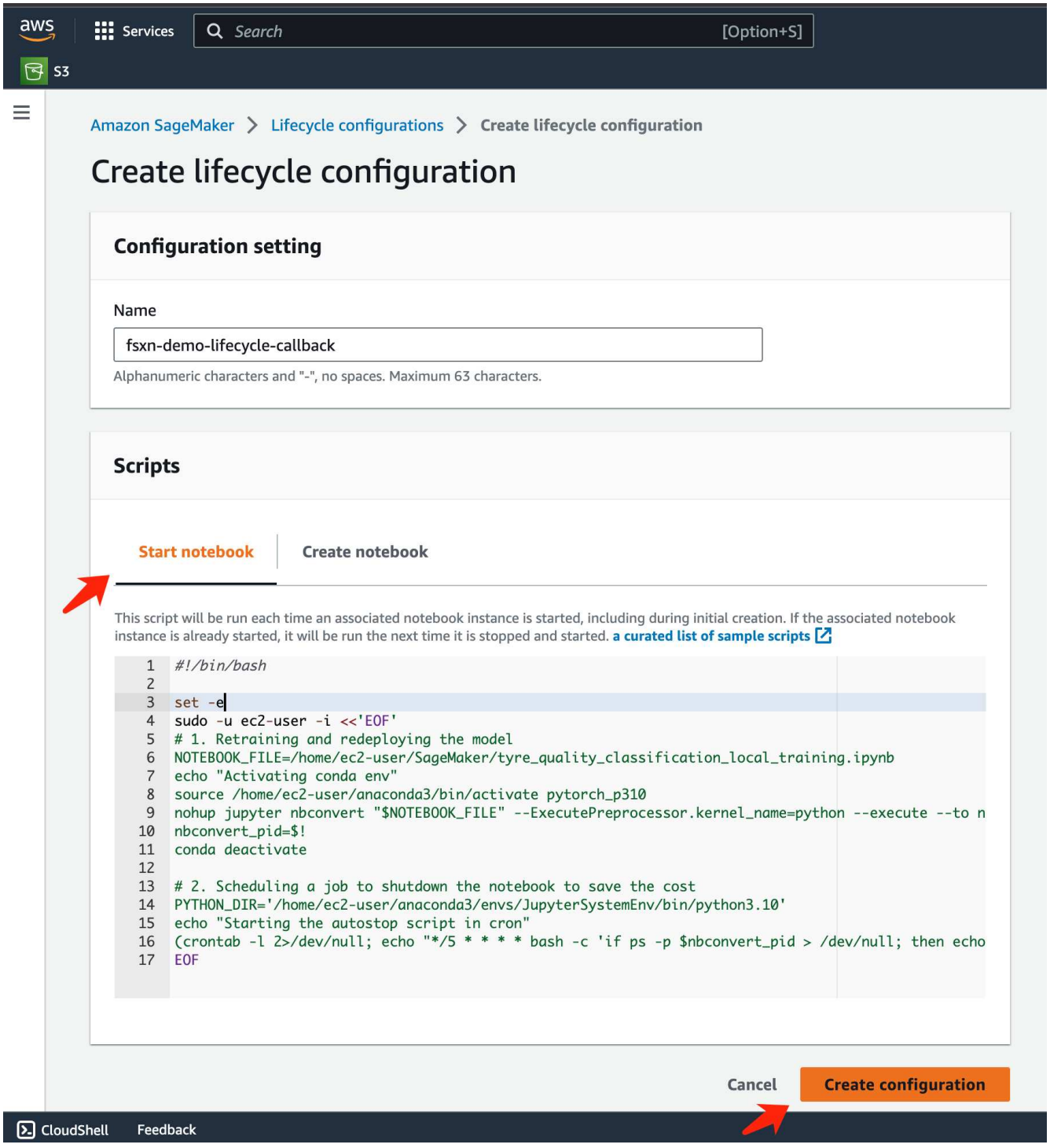
2. [Notebook Instance]タブを選択し、[Create configuration]ボタンをクリックします。



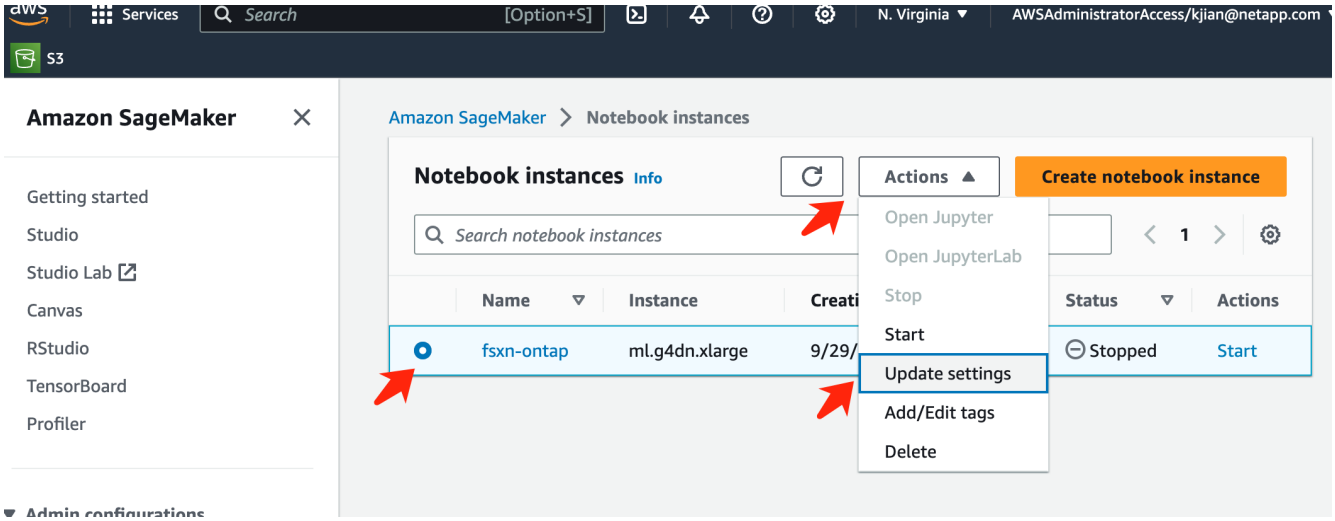
3. 次のコードを入力領域に貼り付けます。

```
#!/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. このスクリプトは、推論のためのモデルの再トレーニングと再配置を処理するJupyter Notebookを実行します。実行が完了すると、ノートブックは5分以内に自動的にシャットダウンされます。問題点とコード実装の詳細については、を参照してください"パート2 - SageMakerのモデルトレーニングのデータソースとしてAmazon FSx for NetApp ONTAP（FSx ONTAP）を活用"。

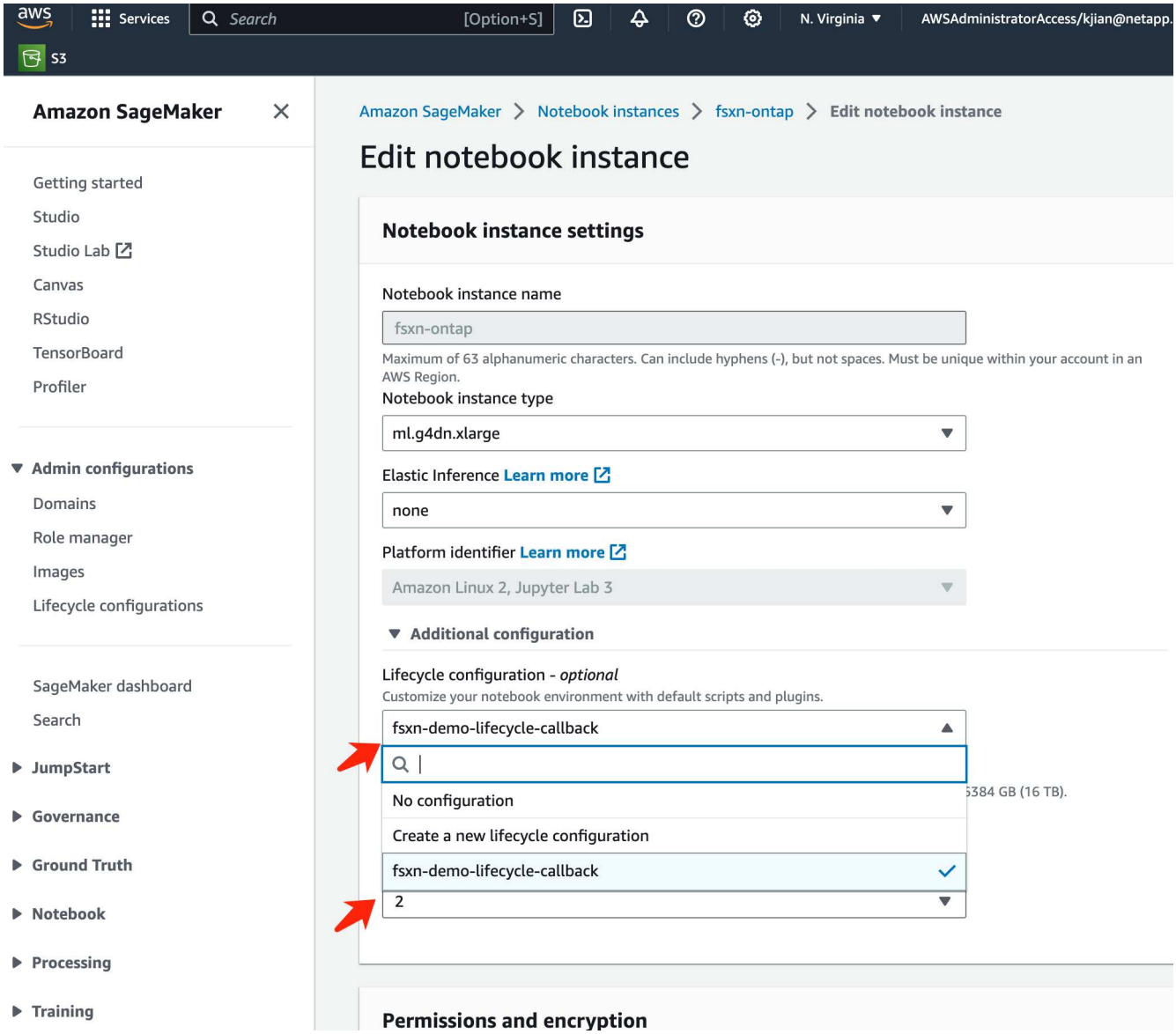


5. 作成後、Notebook Instances（ノートブックインスタンス）に移動してターゲットインスタンスを選択し、Actions（アクション）ドロップダウンの* Update settings（設定の更新）*をクリックします。



ドロップダウン"]

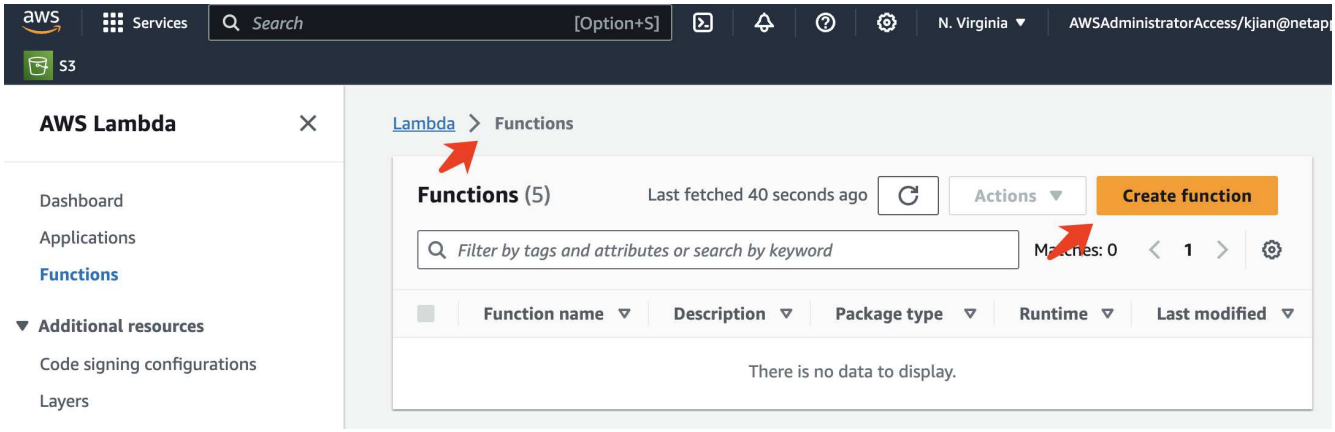
6. 作成した* Lifecycle configuration を選択し、 Update notebook instance *をクリックします。



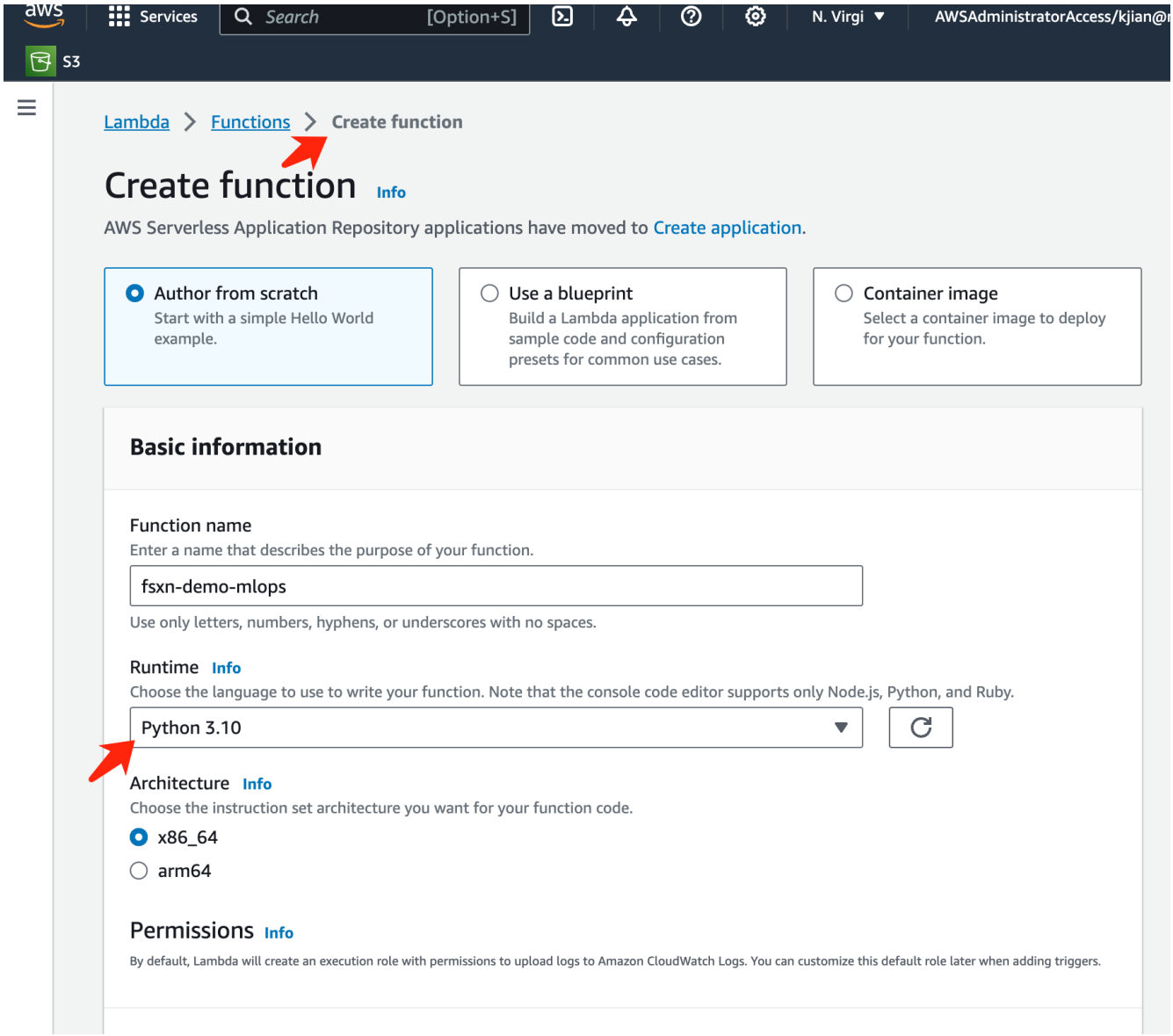
AWS Lambdaサーバレス関数

前述したように、* AWS Lambda関数*は* AWS SageMaker Notebookインスタンス*のスピンアップを担当します。

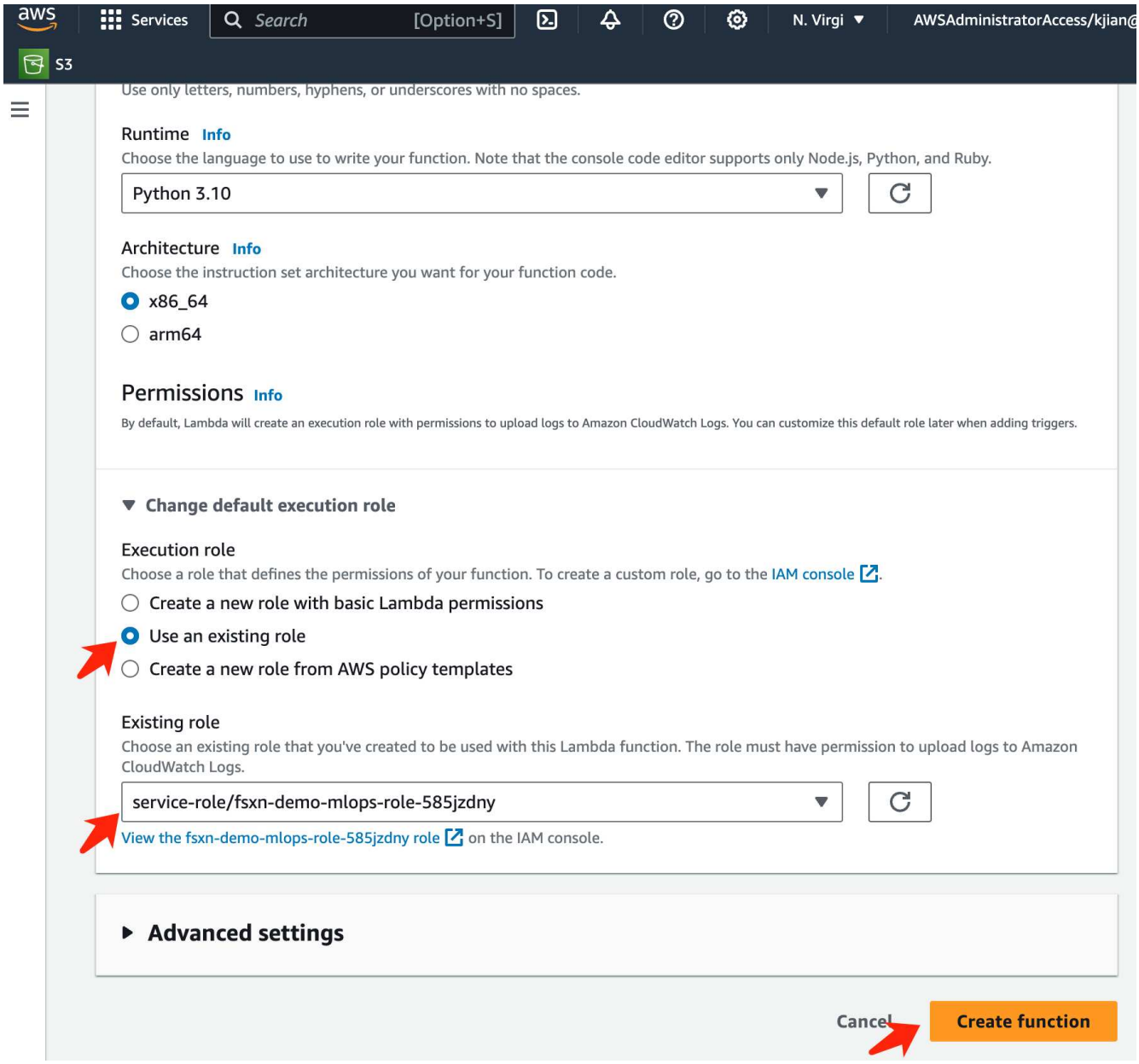
1. AWS Lambda Function を作成するには、該当するパネルに移動し、 Functions タブに切り替えて Create Function *をクリックします。



2. ページに必要なすべてのエントリをファイルし、ランタイムを*Python 3.10*に切り替えることを忘れないでください。



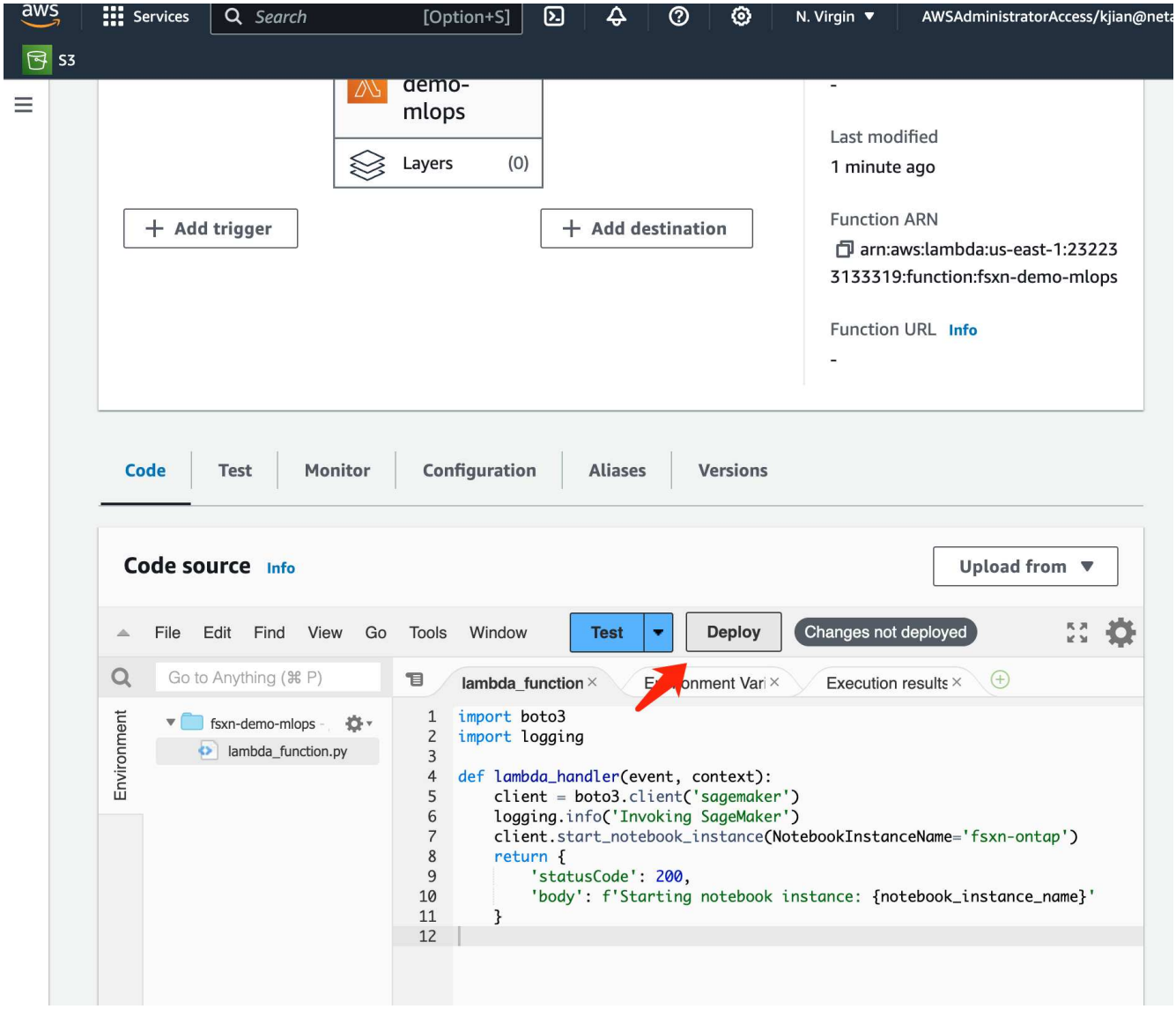
3. 指定されたロールに必要な権限* AmazonSageMakerFullAccess*があることを確認し、* Create Function * ボタンをクリックしてください。



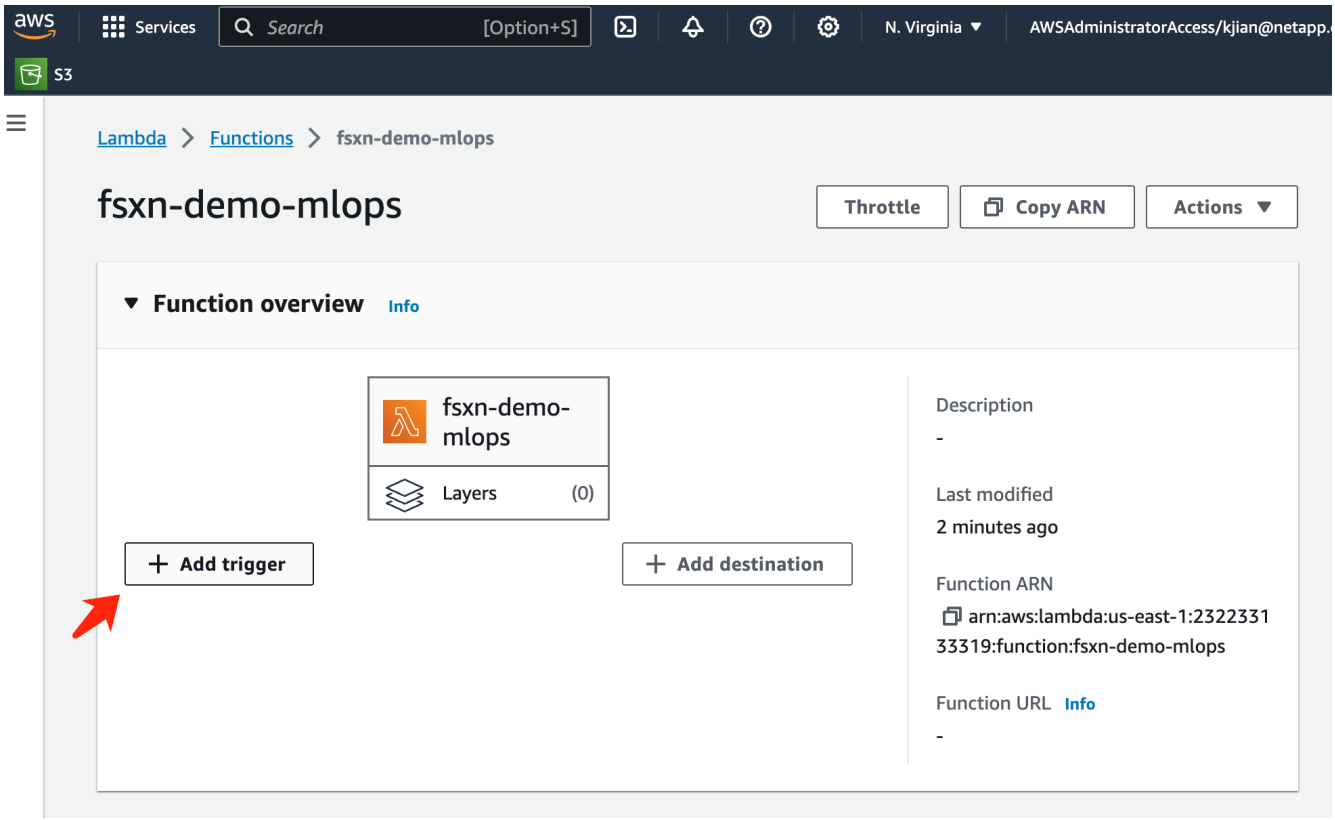
4. 作成したLambda関数を選択します。[コード]タブで、次のコードをコピーしてテキスト領域に貼り付けます。このコードは、* fsxn-ontap *という名前のノートブックインスタンスを起動します。

```
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}'
    }
```

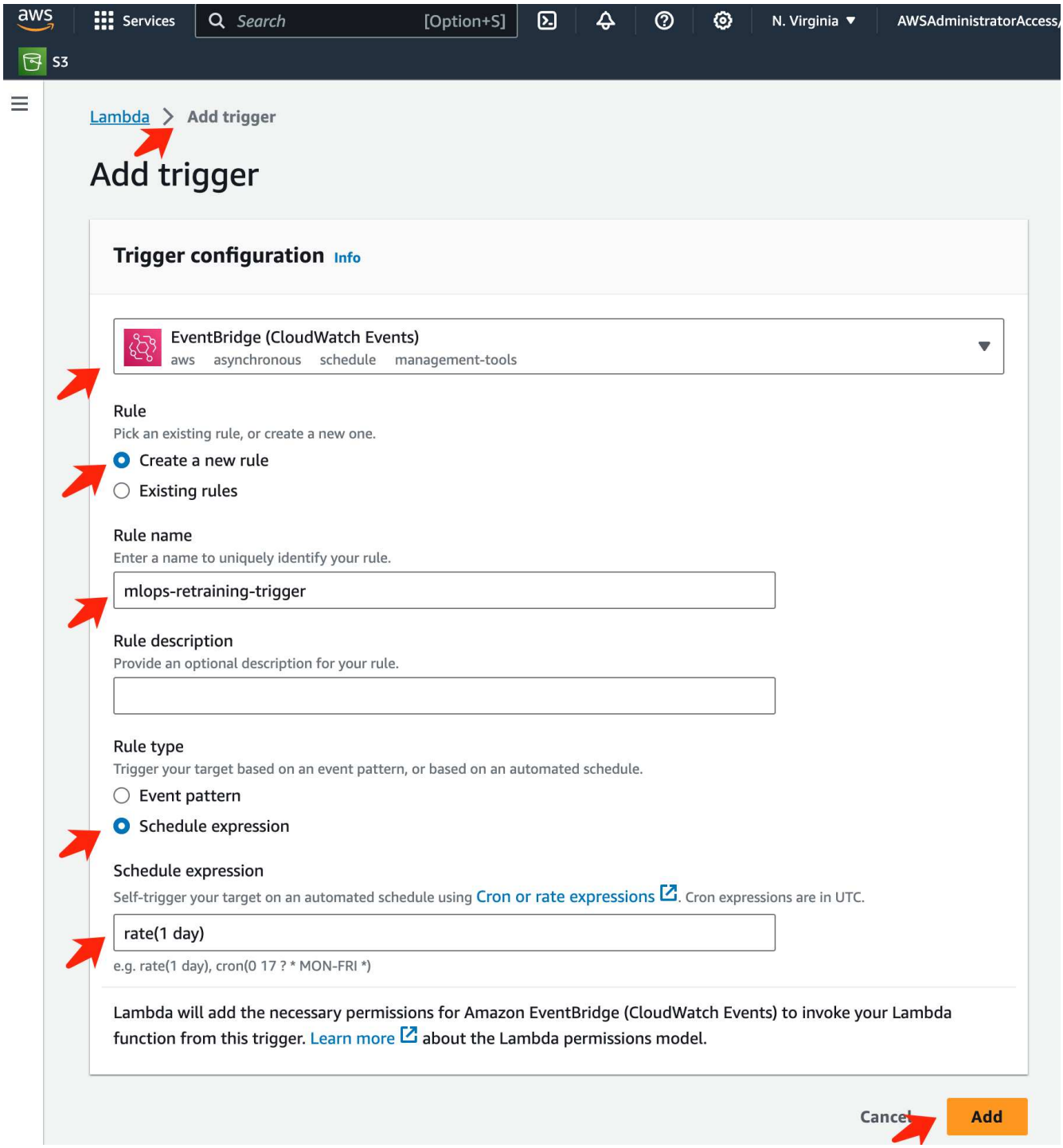
5. このコード変更を適用するには、*配布*ボタンをクリックします。



6. このAWS Lambda関数をトリガーする方法を指定するには、Add Triggerボタンをクリックします。



7. ドロップダウンメニューから[EventBridge]を選択し、[Create a new rule]というラベルの付いたラジオボタンをクリックします。[スケジュール式]フィールドに、次のように入力します。 `rate(1 day)`をクリックし、[追加]ボタンをクリックして、この新しいcronジョブルールを作成し、AWS Lambda関数に適用します。



2段階の設定が完了すると、* AWS Lambda関数*が毎日* SageMaker Notebook を開始し、 FSx ONTAP リポジトリのデータを使用してモデルの再トレーニングを実行し、更新されたモデルを本番環境に再導入し、SageMaker Notebook *インスタンスを自動的にシャットダウンしてコストを最適化します。これにより、モデルが常に最新の状態に保たれます。

これで、MLOpsパイプラインを開発するためのチュートリアルは終了です。

著作権に関する情報

商標に関する情報