



MLOps 向け FSx ONTAP

NetApp artificial intelligence solutions

NetApp
February 12, 2026

目次

MLOps 向け FSx ONTAP	1
MLOps 向けAmazon FSx for NetApp ONTAP (FSx ONTAP)	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	1
はじめに	1
ユーザーガイド	1
便利なデバッグチェックリスト	14
よくある質問（2023年9月27日現在）	15
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) を活用する	15
はじめに	15
FSx ONTAPとは	15
前提条件	16
統合の概要	16
ステップバイステップの統合	17
パート 3 - 簡素化された MLOps パイプラインの構築 (CI/CT/CD)	24
はじめに	24
マニフェスト	24
前提条件	25
アーキテクチャ	25
ステップバイステップの設定	25

MLOps 向け FSx ONTAP

MLOps 向け Amazon FSx for NetApp ONTAP (FSx ONTAP)

このセクションでは、AI インフラストラクチャ開発の実際の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エンドツーエンドのチュートリアルを提供します。3つの包括的な例で構成されており、この強力なデータ管理プラットフォームを介して MLOps のニーズを満たす方法をガイドします。

これらの記事は以下に焦点を当てています。

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 でのモデルトレーニング用のデータソースとして Amazon FSx for NetApp ONTAP \(FSx ONTAP\) を活用する"](#)
3. ["パート 3 - 簡素化された MLOps パイプラインの構築 \(CI/CT/CD\)"](#)

このセクションの終わりまでに、FSx ONTAPを使用して MLOps プロセスを効率化する方法について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方法について説明します。

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方法について説明します。

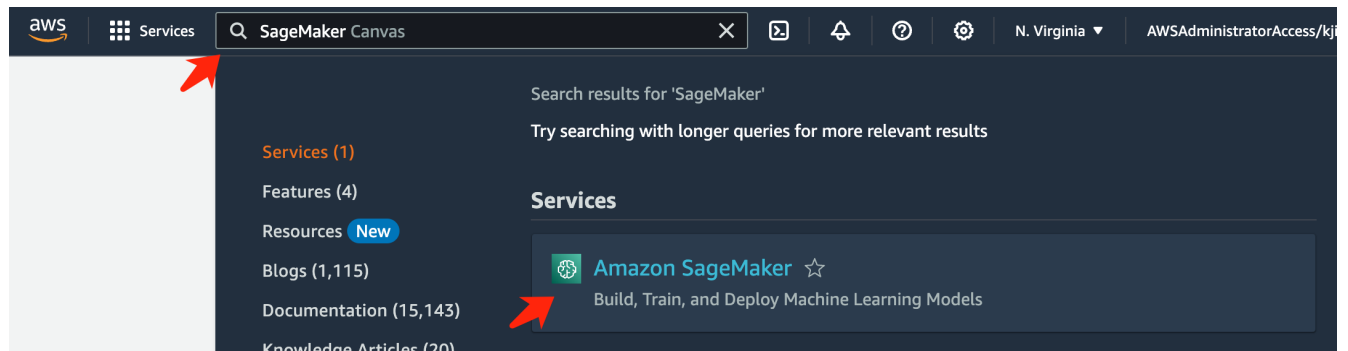
FSx ONTAPの詳細については、このプレゼンテーションをご覧ください (["ビデオリンク"](#))

ユーザーガイド

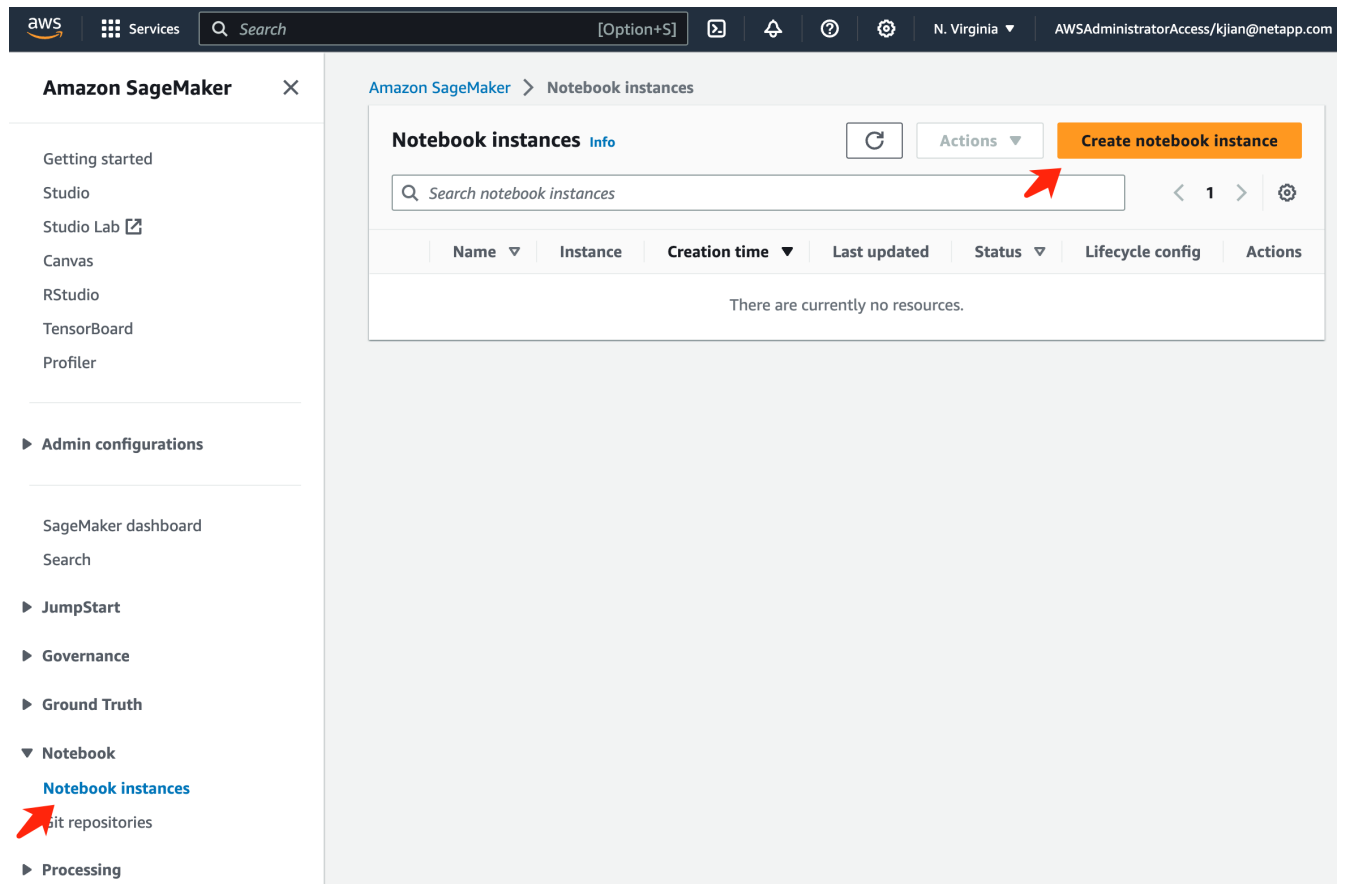
サーバーの作成

SageMaker ノートブックインスタンスを作成する

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. [ノートブック] タブの [ノートブック インスタンス] を開き、オレンジ色のボタン [ノートブック インスタンスの作成] 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*パネルを展開します。他のエントリはデフォルトのままにして、*VPC、サブネット、および*セキュリティグループ*を選択します。(この **VPC** とサブネットは、後で FSx ONTAPファイル システムを作成するために使用されます) 右下にあるオレンジ色のボタン ノートブック インスタンスの作成 をクリックします。

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-00060df0d9f562672 (172.31.16.0/20) | us-east-1a

Security group(s)
sg-0a39b3985770e9256 (default) X

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](#)

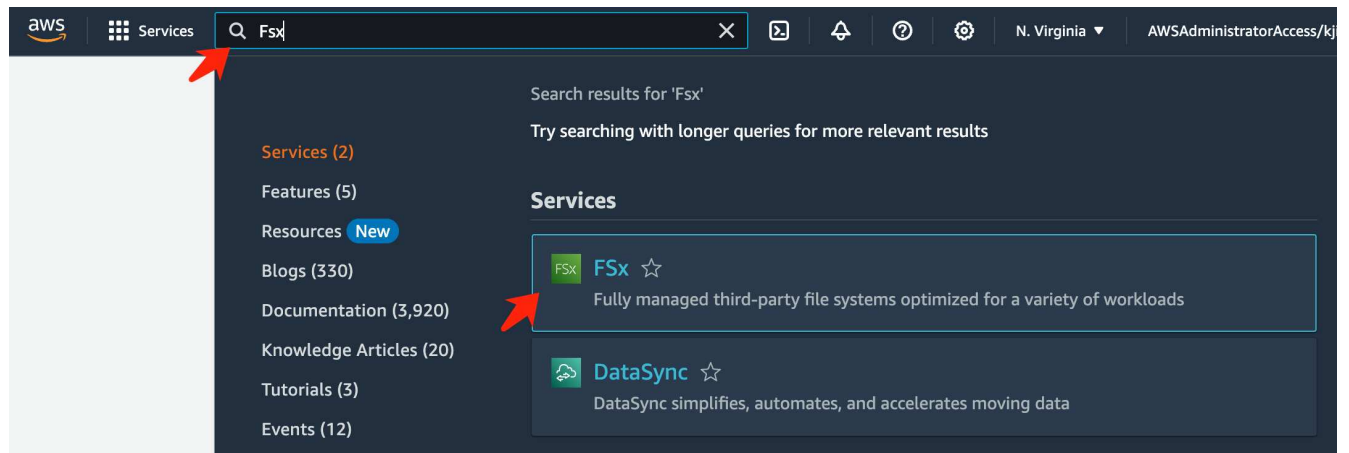
▶ Git repositories - optional

▶ Tags - optional

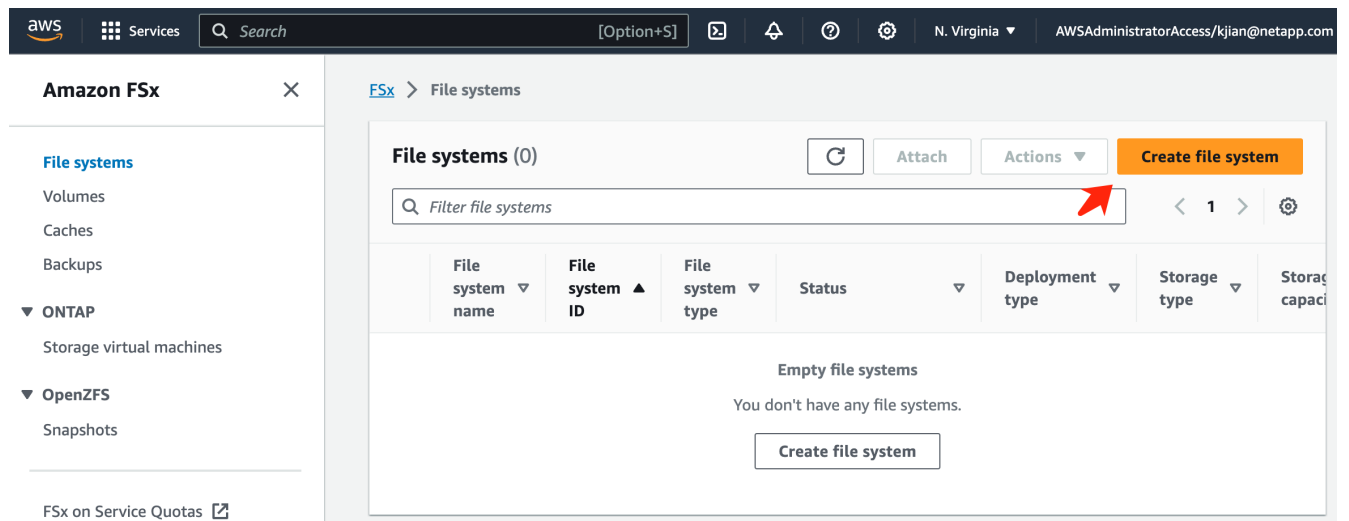
Cancel Create notebook instance

FSx ONTAPファイルシステムを作成する

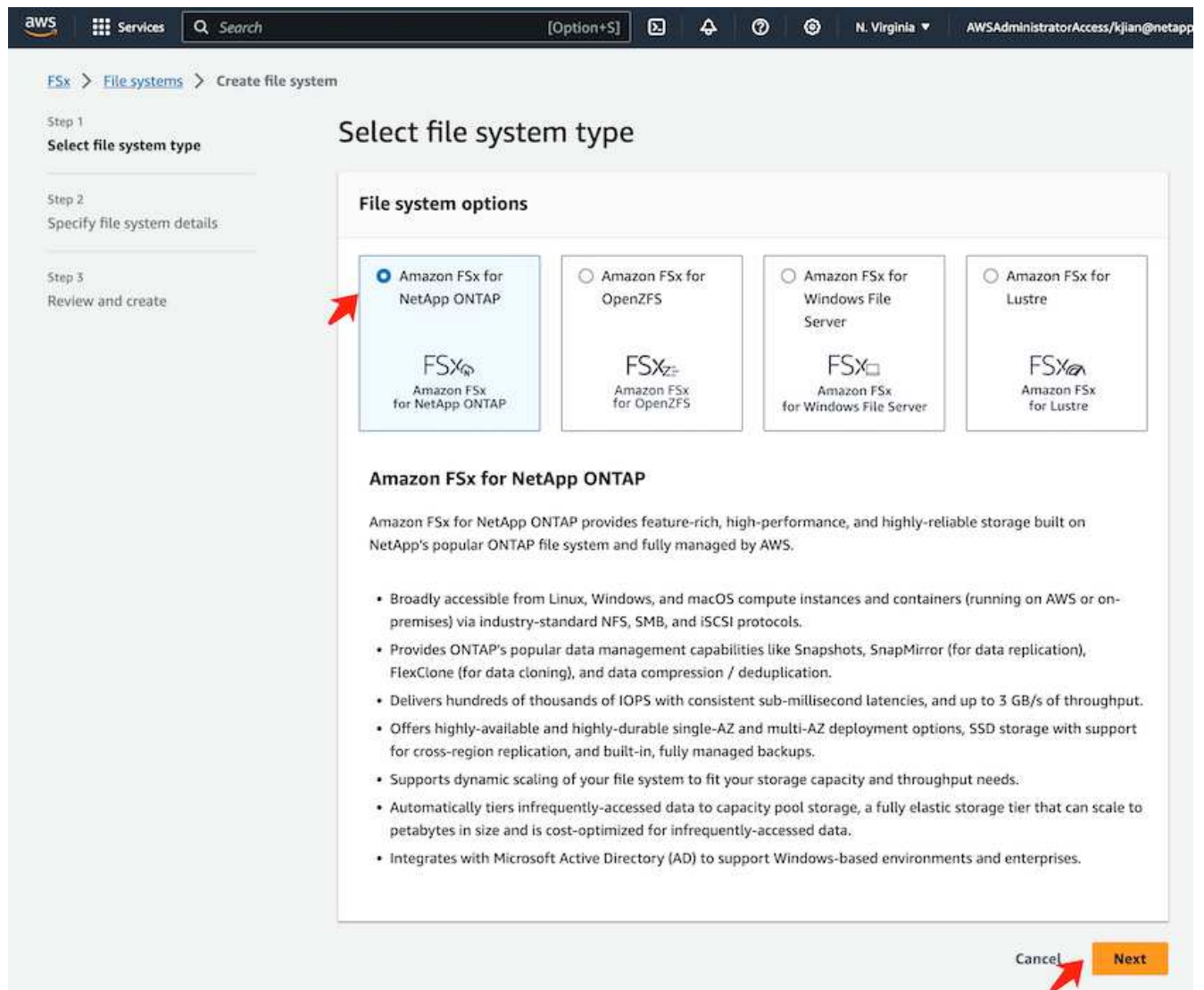
1. AWS コンソールを開きます。検索パネルで「Fsx」を検索し、サービス **FSx** をクリックします。



2. *ファイルシステムの作成*をクリックします。

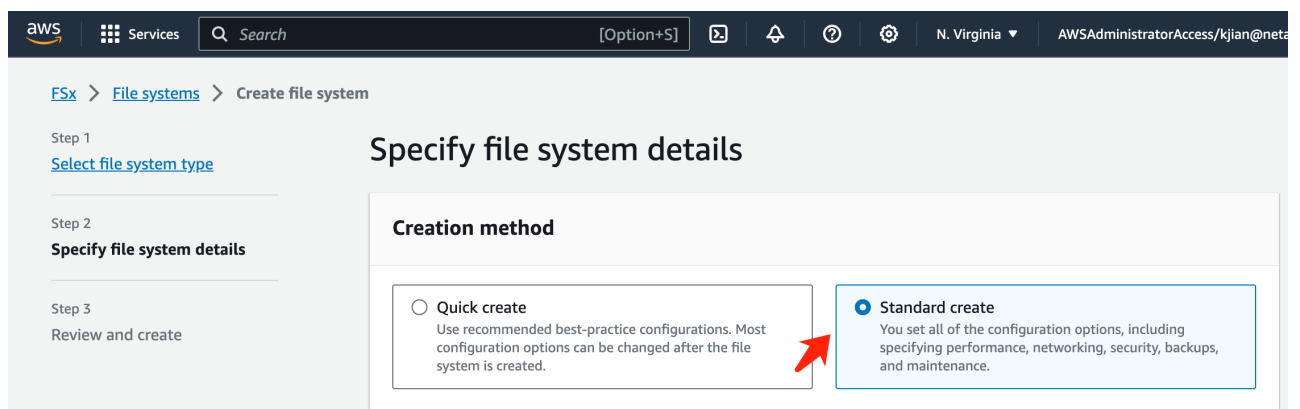


3. 最初のカード **FSx ONTAP** を選択し、次へ をクリックします。



4. 詳細設定ページで。

- a. *標準作成*オプションを選択します。



- b. *ファイルシステム名*と*SSDストレージ容量*を入力します。

File system details

File system name - optional [Info](#)

fsxn-demo

Maximum of 256 Unicode letters, whitespace, and numbers, plus + - = . _ : /

Deployment type [Info](#)

- ☒ Multi-AZ
☐ Single-AZ

SSD storage capacity [Info](#)

1024

GiB

Minimum 1024 GiB; Maximum 192 TiB.

Provisioned SSD IOPS

Amazon FSx provides 3 IOPS per GiB of storage capacity. You can also provision additional SSD IOPS as needed.

- ☒ Automatic (3 IOPS per GiB of SSD storage)
☐ User-provisioned

Throughput capacity [Info](#)

The sustained speed at which the file server hosting your file system can serve data. The file server can also burst to higher speeds for periods of time.

- ☒ Recommended throughput capacity
128 MB/s
☐ Specify throughput capacity

c. **SageMaker Notebook** インスタンスと同じ **VPC** と サブネット を使用するようにしてください。

Network & security

Virtual Private Cloud (VPC) [Info](#)

Specify the VPC from which your file system is accessible.

vpc-0df3956ab1fca2ec9 (CIDR: 172.31.0.0/16) ▼

VPC Security Groups [Info](#)

Specify VPC Security Groups to associate with your file system's network interfaces.

Choose VPC security group(s) ▼

sg-0a39b3985770e9256 (default) ✕

Preferred subnet [Info](#)

Specify the preferred subnet for your file system.

subnet-00060df0d0f562672 (us-east-1a | use1-az4) ▼

Standby subnet

subnet-02b029f24d03a4af2 (us-east-1b | use1-az6) ▼

VPC route tables [Info](#)

Specify the VPC route tables to associate with your file system.

- ☒ VPC's main route table
- ☐ Select one or more VPC route tables

Endpoint IP address range [Info](#)

Specify the IP address range in which the endpoints to access your file system will be created

- ☒ Unallocated IP address range from your VPC
Simplest option for access from other AWS services or peered / on-premises networks
- ☐ Floating IP address range outside your VPC
- ☐ Enter an IP address range

d. ストレージ仮想マシン 名を入力し、SVM (ストレージ仮想マシン) の パスワードを指定 します。

Default storage virtual machine configuration

Storage virtual machine name

Info

fsxn-svm-demo

SVM administrative password

Password for this SVM's "vsadmin" user, which you can use to access the ONTAP CLI or REST API. You can provide a password later if you don't provide one now.

☐ Don't specify a password

☒ Specify a password

Password

.....

Confirm password

.....

Volume security style

The security style of the volume determines whether preference is given to NTFS or UNIX ACLs for multi-protocol access. The MIXED mode is not required for multi-protocol access and is only recommended for advanced users.

Unix (Linux) ▼

Active Directory

Joining an Active Directory enables access from Windows and MacOS clients over the SMB protocol.

☒ Do not join an Active Directory

☐ Join an Active Directory

e. 他のエントリはデフォルトのままにして、右下にあるオレンジ色のボタン「次へ」をクリックします。

▶ Backup and maintenance - optional

▶ Tags - optional

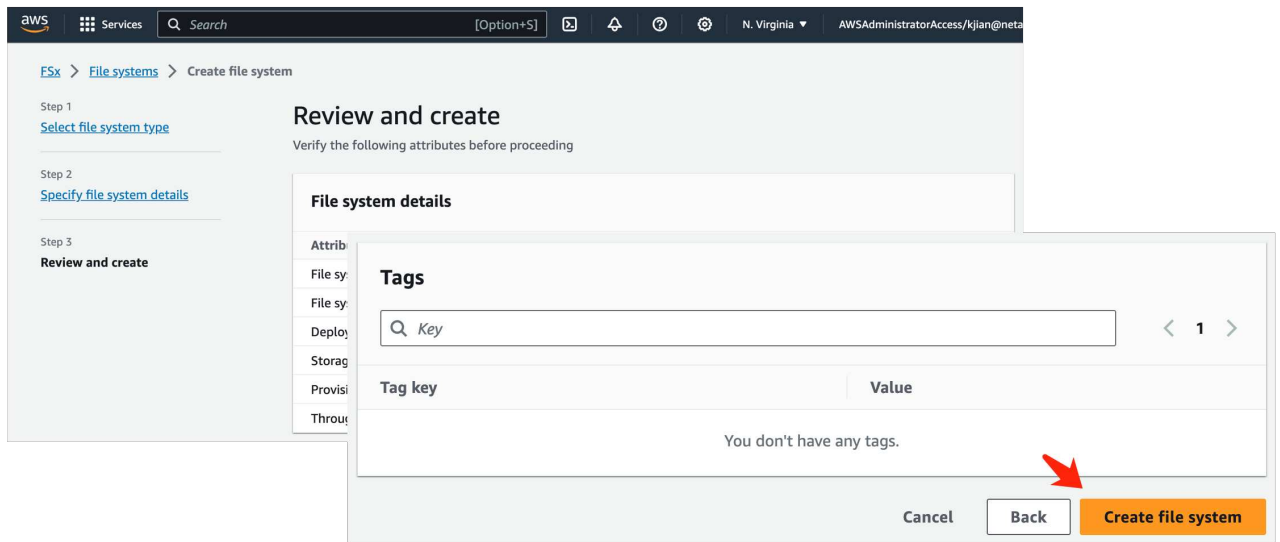
Cancel

Back

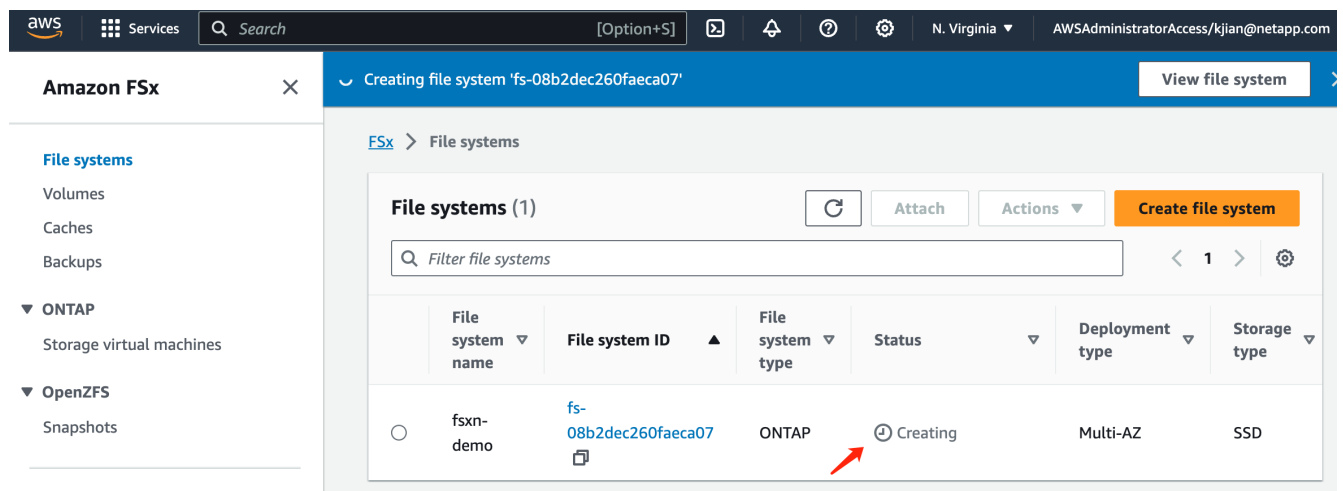
Next

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

8



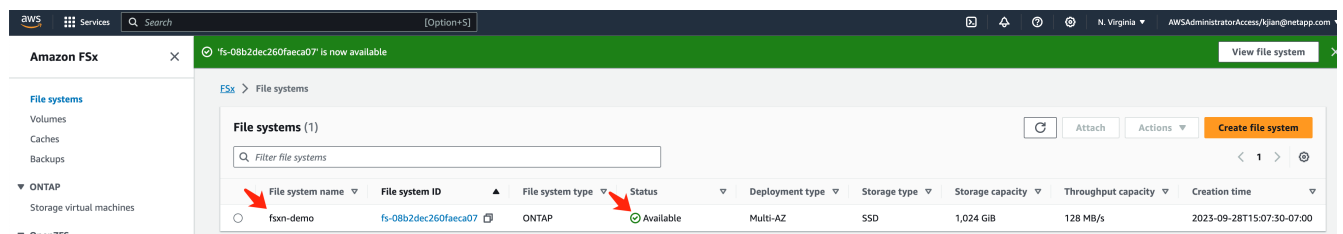
5. FSx ファイル システムの起動には約 20 ～ 40 分 かかる場合があります。



サーバー構成

ONTAP構成

1. 作成された 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管理者ユーザー名*を保持します。

aws Services Search [Option+S] N. Virginia AWSAdministratorAccess/kjian@netapp

Amazon FSx

- File systems
- Volumes
- Caches
- Backups
- ▼ ONTAP
 - Storage virtual machines
- ▼ OpenZFS
 - Snapshots
- FSx on Service Quotas

FSx > File systems > fs-08b2dec260faeca07

fsxn-demo (fs-08b2dec260faeca07)

Attach Actions

▼ Summary

File system ID fs-08b2dec260faeca07	SSD storage capacity 1024 GiB Update	Availability Zones us-east-1a (Preferred) us-east-1b (Standby)
Lifecycle state Creating	Throughput capacity 128 MB/s Update	Creation time 2023-09-28T14:41:50-07:00
File system type ONTAP	Provisioned IOPS 3072 Update	
Deployment type Multi-AZ		

< Network & security Monitoring & performance **Administration** Storage virtual machines >

ONTAP administration

Management endpoint - DNS name management.fs-08b2dec260faeca07.fsx.us-east-1.amazonaws.com	Management endpoint - IP address 172.31.255.250	ONTAP administrator username fsxadmin
Inter-cluster endpoint - DNS name intercluster.fs-08b2dec260faeca07.fsx.us-east-1.amazonaws.com	Inter-cluster endpoint - IP address 172.31.31.157 172.31.32.38	ONTAP administrator password Update

3. 作成された*SageMaker Notebookインスタンス*を開き、*JupyterLabを開く*をクリックします。

aws Services Search [Option+S] N. Virginia AWSAdministratorAccess/kjian@netapp

Amazon SageMaker

- Getting started
- Studio
- Studio Lab
- Canvas
- RStudio
- TensorBoard

Amazon SageMaker > Notebook instances

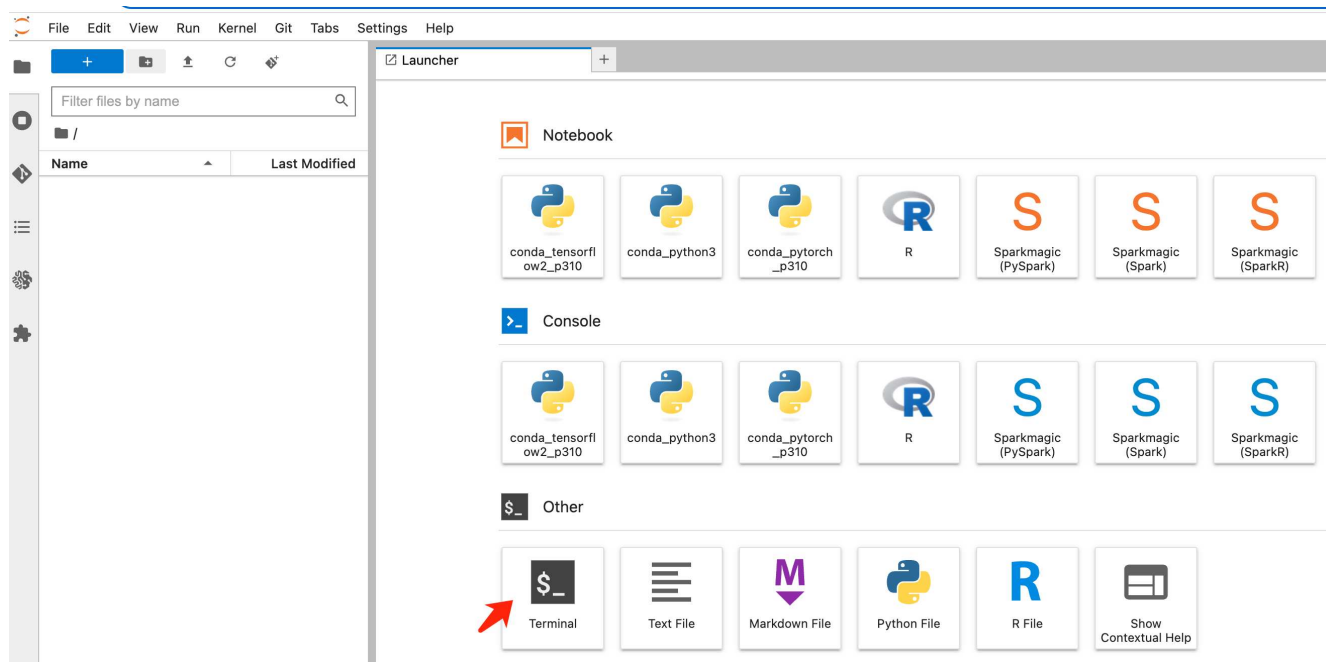
Notebook instances

Search notebook instances

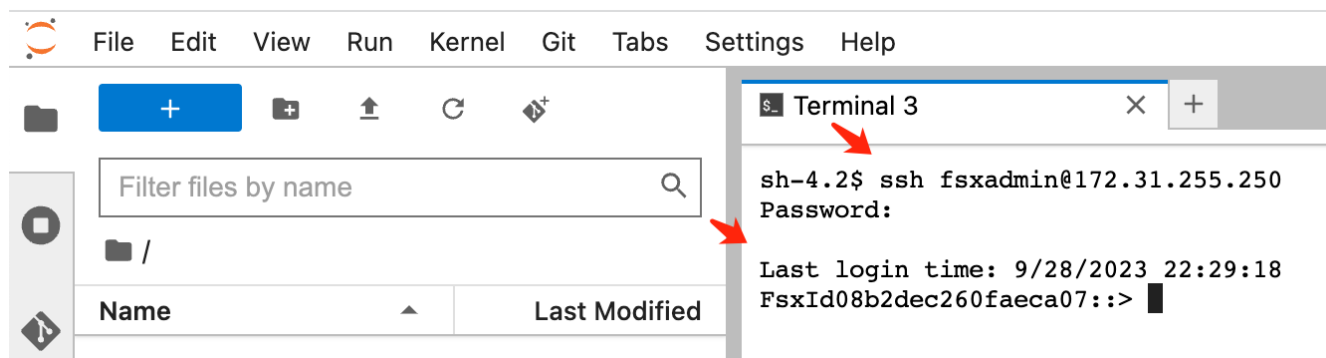
Create notebook instance

Name	Instance	Creation time	Last updated	Status	Lifecycle config	Actions
fsxn-demo	ml.t3.medium	9/28/2023, 1:47:27 PM	9/28/2023, 1:50:28 PM	InService		Open Jupyter Open JupyterLab

4. Jupyter Lab ページで、新しい ターミナル を開きます。



5. FSx ONTAPファイル システムにログインするには、ssh コマンド `ssh <admin user name>@< ONTAP server IP>` を入力します。(ユーザー名とIPアドレスは手順2で取得します) *ストレージ仮想マシン*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 の名前として **fsxn-ontap** を使用します。-vserver 引数には ストレージ仮想マシン名 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 と資格情報を取得します。

```

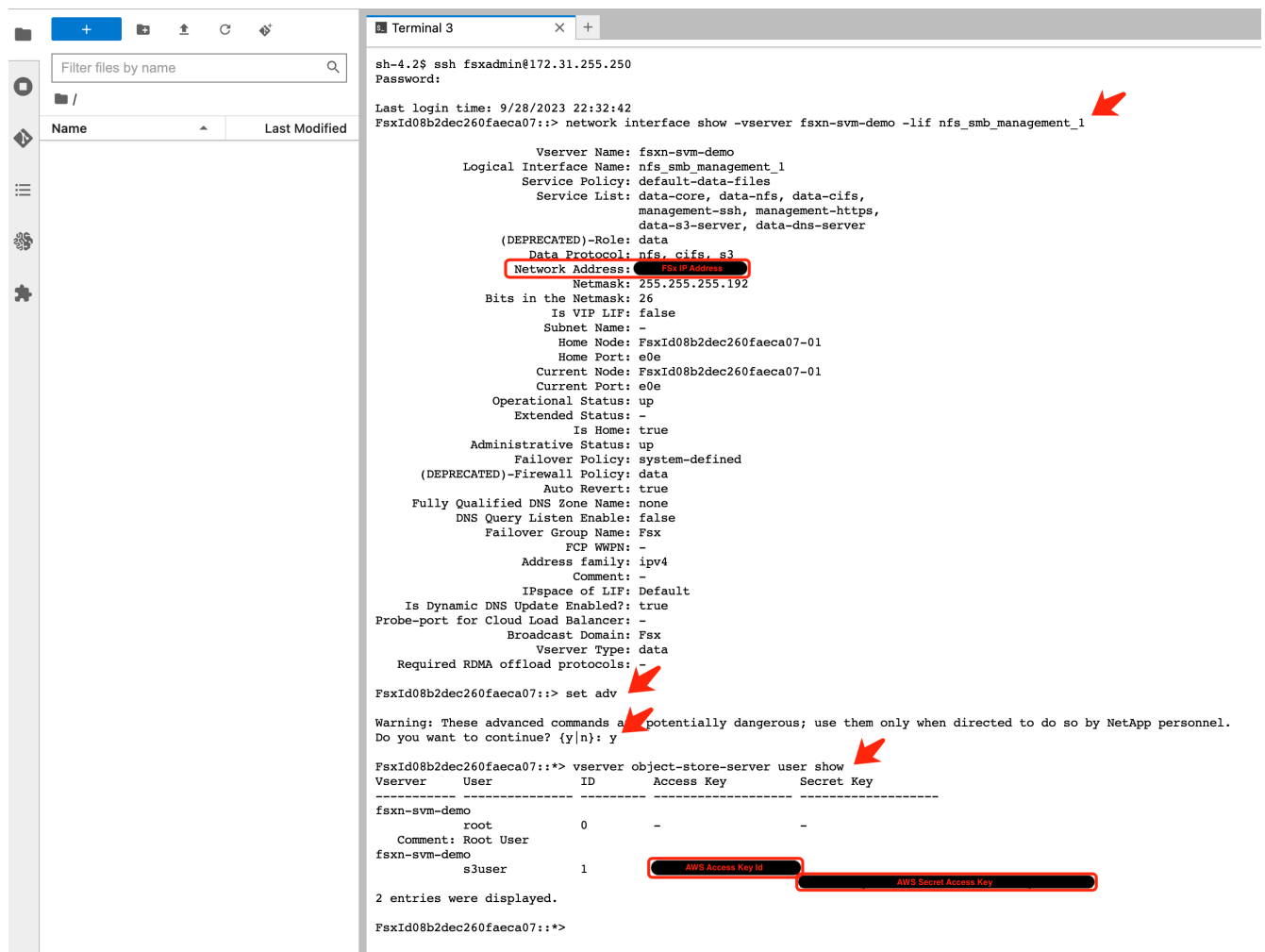
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

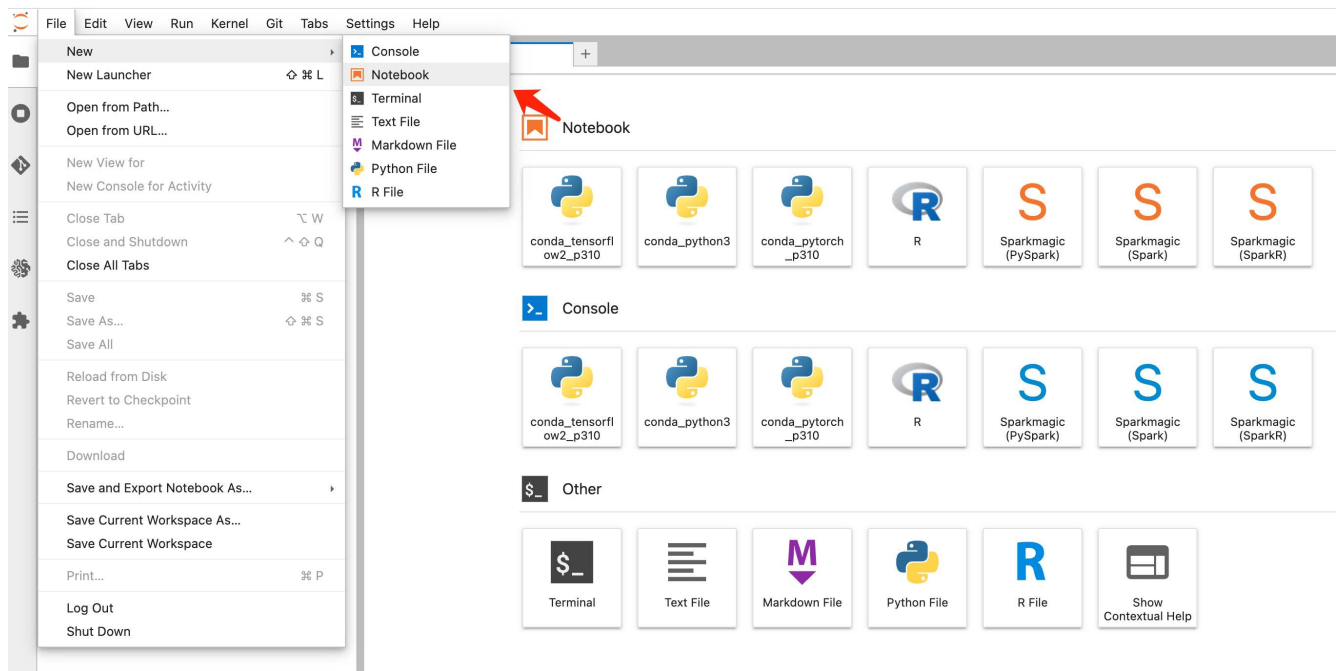
vserver object-store-server user show

```

8. 将来使用するためにエンドポイントの IP と資格情報を保持します。



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. 以下のコードを回避策として使用して、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します。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 Fsx 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完了です。

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 にあることを確認します。

- 権限レベルを **dev** に設定するには、ONTAPで **set dev** コマンドを実行することを忘れないでください。

よくある質問（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** 操作の呼び出し時にエラーが発生しました (**NotImplemented**): 要求した **s3** コマンドは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 100 MB のファイルのアップロードをサポートします。S3 プロトコルを使用する場合、100 MB を超えるファイルは 100 MB 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ときに、「**PutObject** 操作の呼び出し時にエラーが発生しました (**AccessDenied**)。アクセスが拒否されました」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 認証情報を FSx ONTAP認証情報に切り替えます。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という回避策が必要です。

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 では、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は提供されていません。その結果、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)** を活用する

この記事は、Amazon FSx for NetApp ONTAP (FSx ONTAP) を使用して SageMaker で PyTorch モデルをトレーニングする方法について、具体的にはタイヤ品質分類プロジェクト向けのチュートリアルです。

はじめに

このチュートリアルでは、コンピュータービジョン分類プロジェクトの実用的な例を示し、SageMaker 環境内で FSx ONTAP をデータソースとして利用する ML モデルの構築に関する実践的な体験を提供します。このプロジェクトは、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、タイヤ画像に基づいてタイヤの品質を分類することに重点を置いています。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。

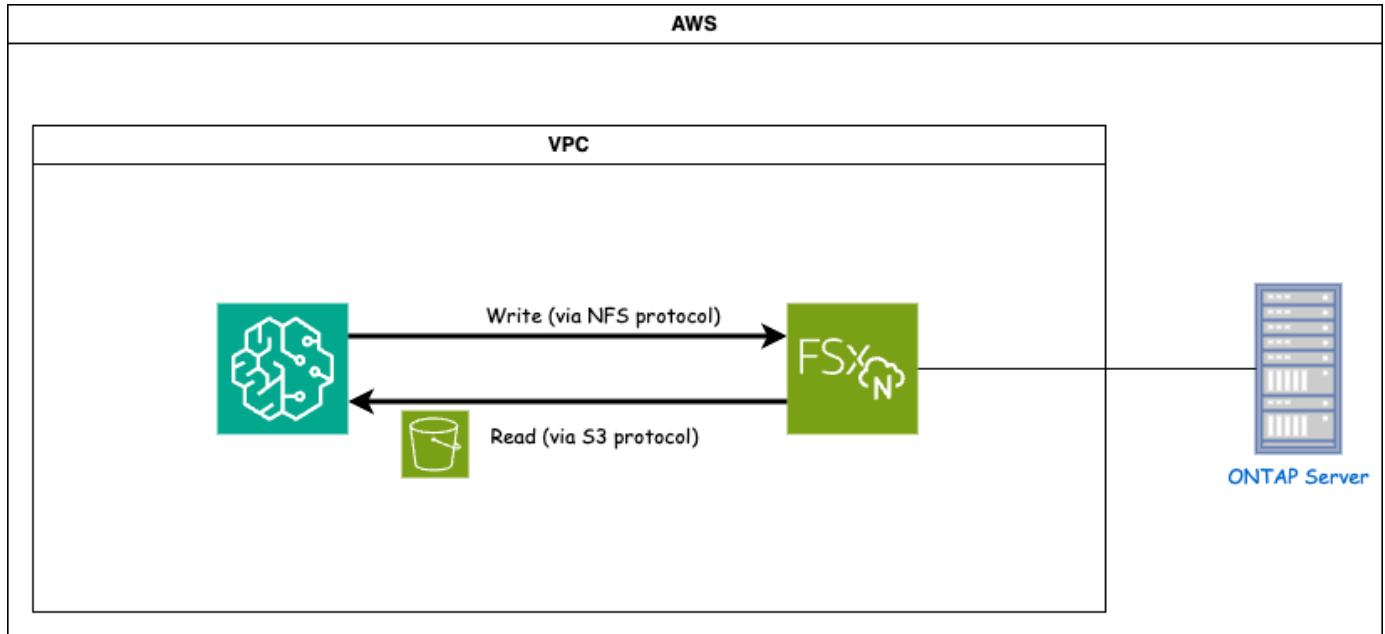
FSx ONTAPとは

Amazon FSx ONTAP は、AWS が提供する完全に管理されたストレージソリューションです。NetApp のONTAPファイルシステムを活用して、信頼性が高く高性能な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ーからのシームレスなアクセスが可能になります。このサービスは、優れたパフォーマンスを提供し、高速

かつ効率的なデータ操作を保証するように設計されています。また、高い可用性と耐久性も提供し、データのアクセスと保護が維持されます。さらに、Amazon FSx ONTAPのストレージ容量はスケーラブルなので、ニーズに応じて簡単に調整できます。

前提条件

ネットワーク環境



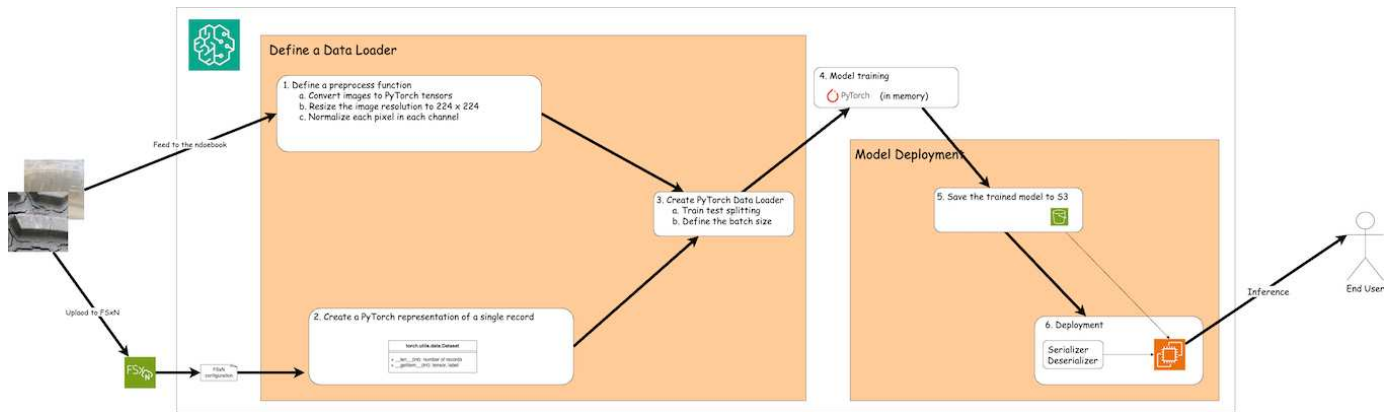
FSx ONTAP (Amazon FSx ONTAP) は AWS ストレージサービスです。これには、NetApp ONTAPシステム上で実行されるファイルシステムと、それに接続する AWS 管理のシステム仮想マシン (SVM) が含まれます。提供された図では、AWS によって管理される NetApp ONTAP サーバーは VPC の外部に配置されています。SVM は SageMaker と NetApp ONTAP システム間の仲介役として機能し、SageMaker から操作要求を受信して、基盤となるストレージに転送します。FSx ONTAP にアクセスするには、SageMaker を FSx ONTAP デプロイメントと同じ VPC 内に配置する必要があります。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活用して機械学習モデルを構築します。ただし、デモンストレーションの目的では、FSx ONTAP ファイルシステムは作成後最初は空であるため、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ると、マウントされた場所にデータセットをアップロードして、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なります。このアプローチにより、モデルの開発とトレーニングに SageMaker を使用しながら、FSx ONTAP のストレージ容量と機能を活用できます。

データ読み取りプロセスでは、FSx ONTAP をプライベート S3 バケットとして構成する必要があります。詳細な設定手順については、以下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、データ ローダーの定義、モデルのトレーニング、およびデプロイメントという 3 つの主なステップに要約できます。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、包括的な実装のために、各ステップにはいくつかの詳細なサブステップが含まれます。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、モデルの展開などのさまざまなタスクが含まれます。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 モデルを構築および展開するための徹底的かつ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 Loader

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が可能になります。

データ ローダーは 3 つの部分で構成されます。

前処理関数

```
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。このチュートリアルでは、一連の変換を適用するための前処理オブジェクトが作成されます。まず、**ToTensor()** 変換により、画像をテンソル表現に変換します。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、各チャンネルに

沿って平均を減算し、標準偏差で割ることで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、事前トレーニング済みのニューラル ネットワーク モデルでよく使用されます。全体として、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、画像データをさらに処理したり、事前トレーニング済みモデルに入力したり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合計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メソッドを定義します。**getitem** 関数内で、コードは boto3 S3 バケット オブジェクトを使用して、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場合と似ています。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。

プライベートS3リポジトリとしてのFSx ONTAP

```

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 Fsx 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ーが作成されます。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。ハンドラーの設定には、FSx ONTAP SVM の IP アドレス、バケット名、および必要な資格情報の指定が含まれます。これらの構成項目の取得に関する詳細な説明については、次の文書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"](#)。

上記の例では、バケット 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、後続のセクションでさらに詳しく説明します。

```

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)

```

提供されている例では、バッチ サイズ 64 が指定されており、各バッチに 64 個の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 **TyreQualityClassifier** 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 バッチを反復処理し、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、監視の目的で現在の時刻、エポック、バッチ、損失を出力します。

モデルの展開

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.splitext(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())

```

SageMaker ではモデルをデプロイするために S3 に保存する必要があるため、コードは PyTorch モデルを **Amazon S3** に保存します。モデルを **Amazon S3** にアップロードすると、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ル化するカスタム シリアライザー **TyreQualitySerializer** を定義します。**TyreQualityPredictor** クラスは、定義されたシリアライザーと **JSONDeserializer** を利用するカスタム予測子です。このコードは、モデルの S3 の場所、IAM ロール、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ための **PyTorchModel** オブジェクトも作成します。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がデプロイされます。これにより、PyTorch モデルをデプロイし、SageMaker で推論にアクセス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)
```

これは、デプロイされたエンドポイントを使用して推論を行う例です。

パート 3 - 簡素化された MLOps パイプラインの構築 (CI/CT/CD)

この記事では、自動化されたモデルの再トレーニング、デプロイ、コストの最適化に焦点を当て、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。

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、継続的インテグレーション (CI)、継続的トレーニング (CT)、継続的デプロイメント (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 ループに組み込み、モデルの継続的なトレーニングと推論のシームレスな展開を可能にする方法について理解を深めることができます。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について説明します。

マニフェスト

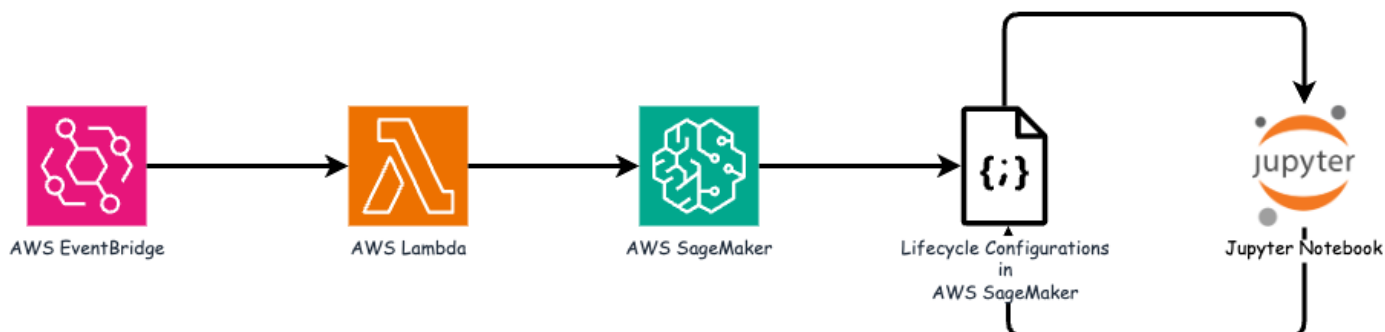
機能	Name	コメント
データストレージ	AWS FSx ONTAP	。"パート 1 - Amazon FSx for NetApp ONTAP (FSx ONTAP) をプライベート S3 バケットとして AWS SageMaker に統合する"。
データサイエンスIDE	AWS セージメーカー	このチュートリアルは、"パート 2 - SageMaker でのモデルトレーニング用のデータソースとして Amazon FSx for NetApp ONTAP (FSx ONTAP) を活用する"。

機能	Name	コメント
MLOpsパイプラインをトリガーする関数	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	パイトーチ	-
AWS Python SDK	ボト3	-
プログラミング言語	Python	バージョン3.10

前提条件

- 事前設定された FSx ONTAPファイル システム。このチュートリアルでは、トレーニング プロセスに FSx ONTAPに保存されているデータを利用します。
- 上記の 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 Notebook** 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 状態であ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活用するには、**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ー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 を定期的呼び出します。このプロセスでは、必要なタスクを実行するために **AWS SageMaker Notebook** インスタンスを起動する必要があります。

ステップバイステップの設定

ライフサイクル構成

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*を利用します。このサービスを使用すると、ノートブック インスタンスを起動するときに実行する必要なアクションを定義できます。具体的には、*ライフサイクル構成*内にシェル スクリプトを実装して、トレーニングとデプロイメントの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 ではコストが主要な考慮事項の 1 つであるため、これは必須の構成です。

*ライフサイクル構成*の構成を事前に設定しておく必要があることに注意することが重要です。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、この側面の構成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、管理構成 セクションの ライフサイクル 構成 に移動します。

aws

Services

Q Search

S3

Amazon SageMaker

×

Getting started

Studio

Studio Lab

Canvas

RStudio

TensorBoard

Profiler

▼ Admin configurations

Domains

Role manager

Images

Lifecycle configurations

SageMaker dashboard

Search

► JumpStart

Amazon SageMaker > Domains

Domains Info

A domain includes an associated Amazon SageMaker Studio instance. Each domain receives a personal and private Amazon SageMaker Studio instance.

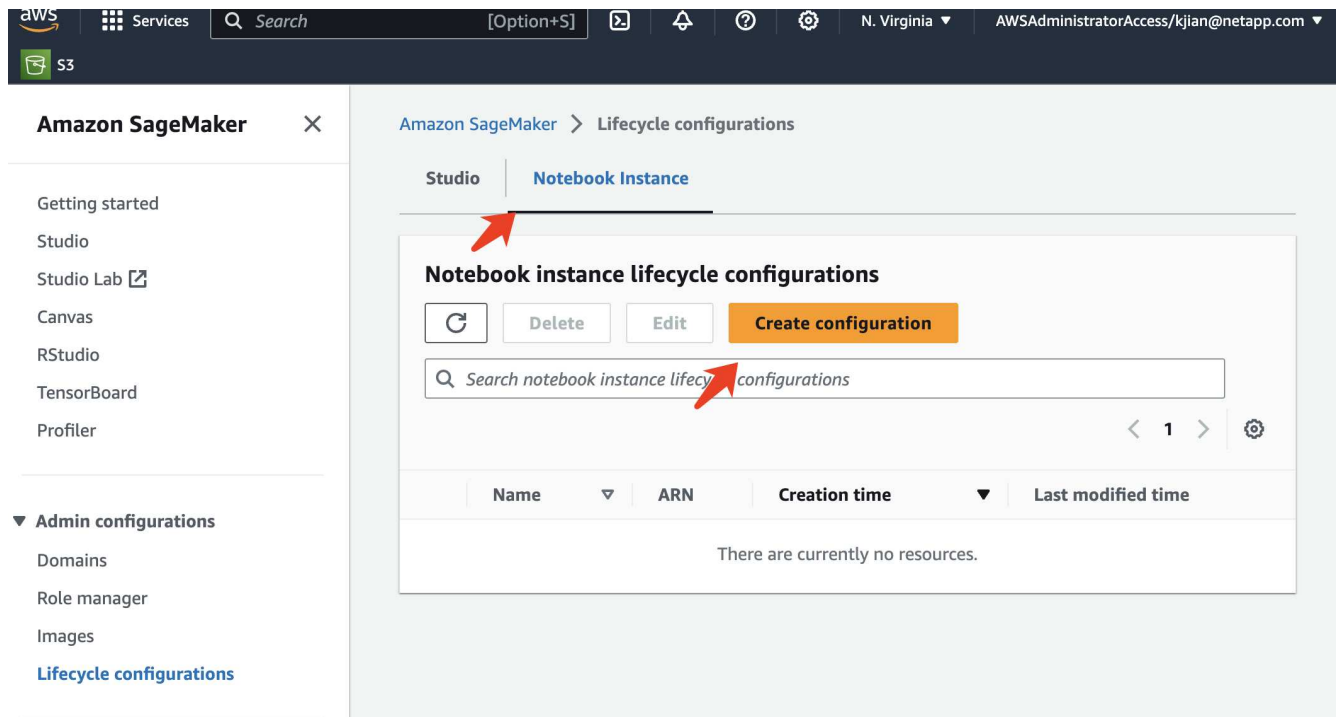
► Domain structure diagram

Domains (4) Info

Q Find domain name

	Name
<input type="radio"/>	rdsml-east-1
<input type="radio"/>	rdsml-east-2
<input type="radio"/>	rdsml-east-3
<input type="radio"/>	rdsml-east-4

2. *ノートブックインスタンス*タブを選択し、*構成の作成*ボタンをクリックします。



3. 以下のコードを入力エリアに貼り付けます。

```
#!/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、推論用モデルの再トレーニングと再デプロイメントを処理する 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します。問題の説明とコードの実装の詳細については、以下を参照してください。"[パート 2 - SageMaker でのモデルトレーニング用のデータソースとして Amazon FSx for NetApp ONTAP \(FSx ONTAP\) を活用する](#)"。

Configuration setting

Name

fsxn-demo-lifecycle-callback

Alphanumeric characters and "-", no spaces. Maximum 63 characters.

Scripts

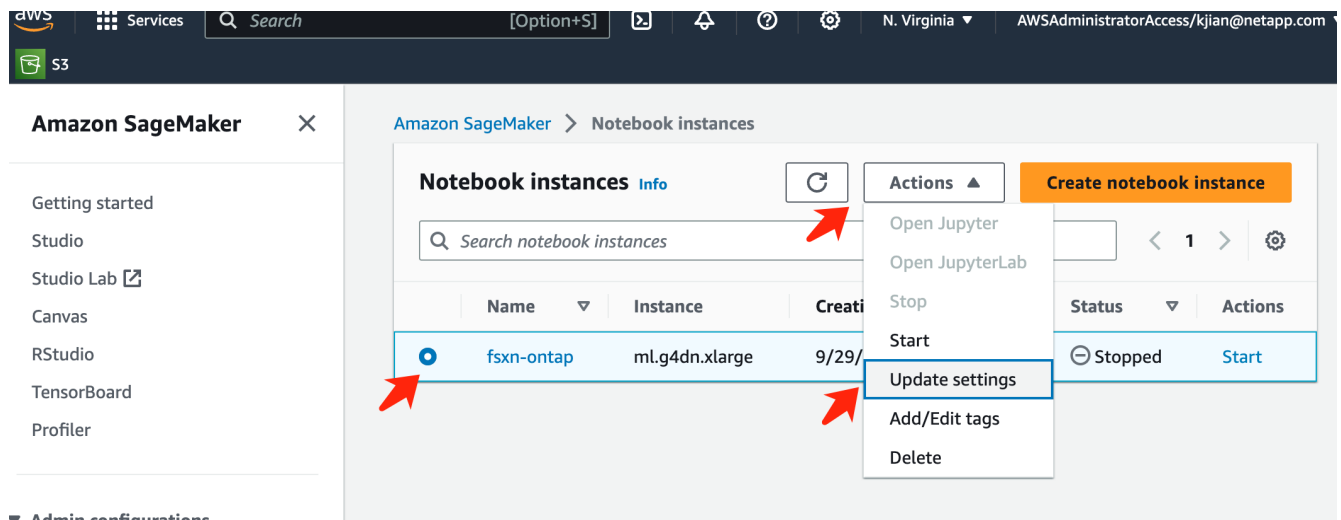
Start notebook | Create notebook

This script will be run each time an associated notebook instance is started, including during initial creation. If the associated notebook instance is already started, it will be run the next time it is stopped and started. [a curated list of sample scripts](#)

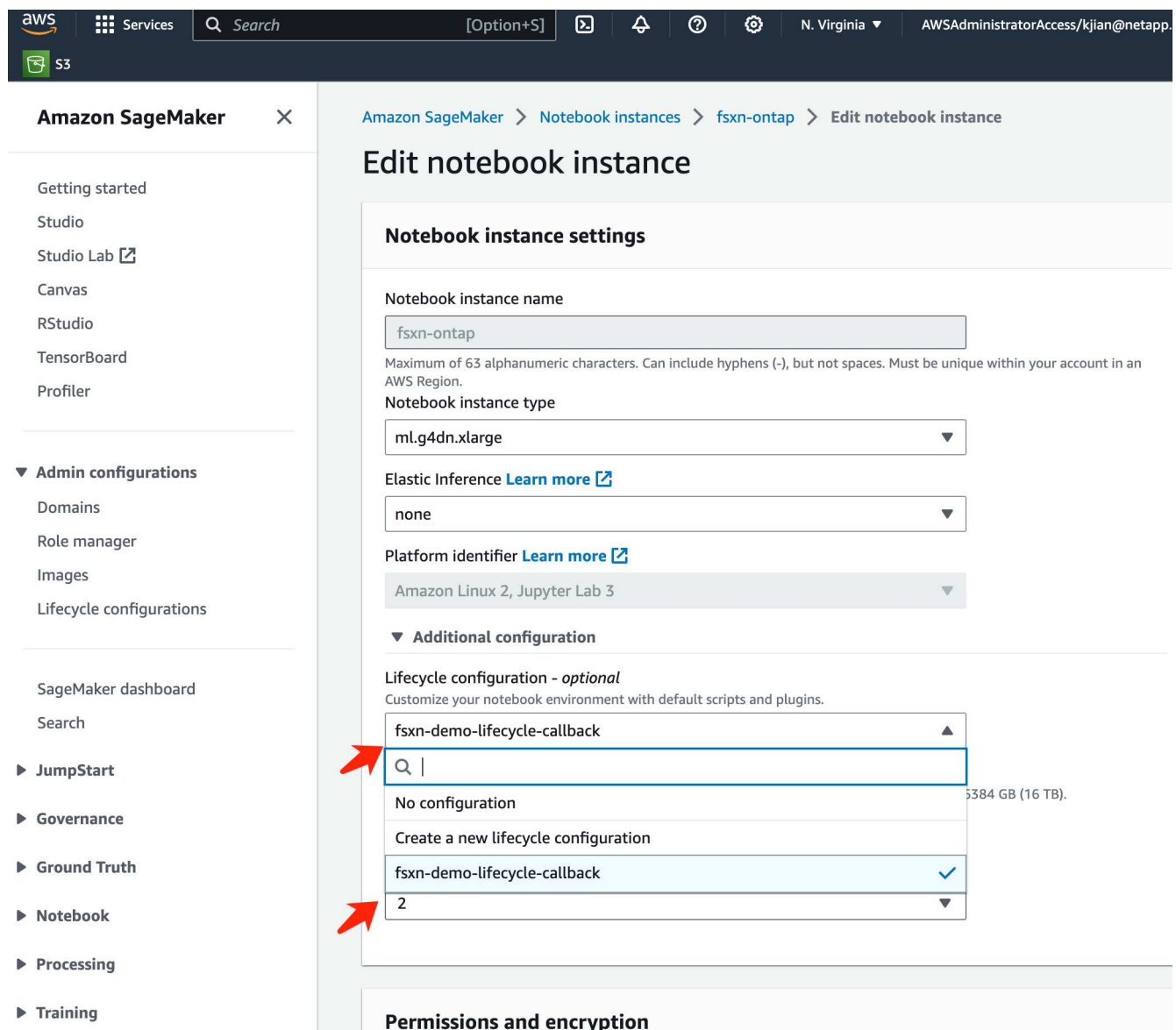
```
1 #!/bin/bash
2
3 set -e
4 sudo -u ec2-user -i <<'EOF'
5 # 1. Retraining and redeploying the model
6 NOTEBOOK_FILE=/home/ec2-user/SageMaker/tyre_quality_classification_local_training.ipynb
7 echo "Activating conda env"
8 source /home/ec2-user/anaconda3/bin/activate torch_p310
9 nohup jupyter nbconvert "$NOTEBOOK_FILE" --ExecutePreprocessor.kernel_name=python --execute --to nbconvert_pid=$!
10 nbconvert_pid=$!
11 conda deactivate
12
13 # 2. Scheduling a job to shutdown the notebook to save the cost
14 PYTHON_DIR='/home/ec2-user/anaconda3/envs/JupyterSystemEnv/bin/python3.10'
15 echo "Starting the autostop script in cron"
16 (crontab -l 2>/dev/null; echo "*/5 * * * * bash -c 'if ps -p $nbconvert_pid > /dev/null; then echo"
17 EOF
```

Cancel **Create configuration**

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



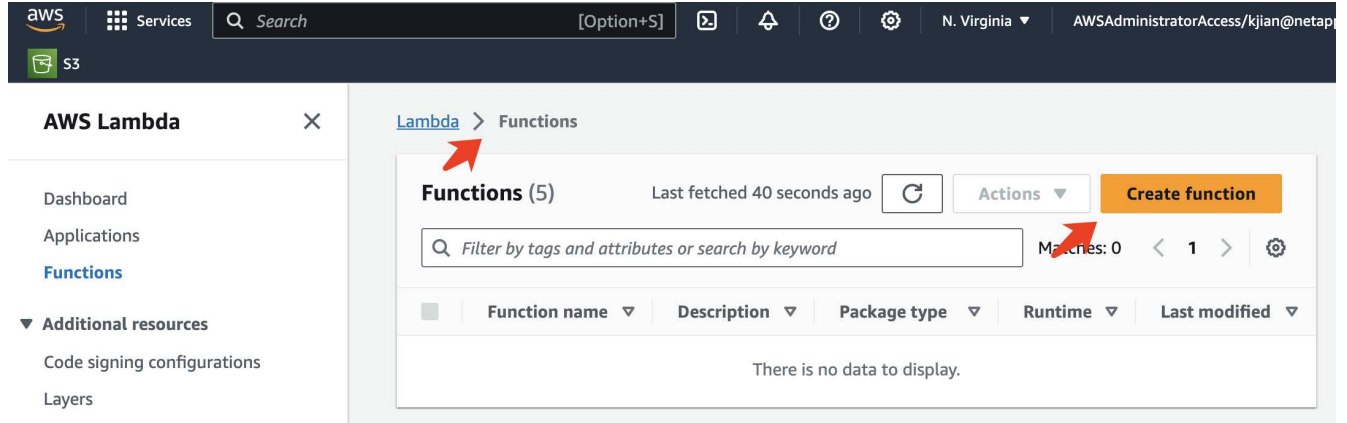
6. 作成した*ライフサイクル構成*を選択し、*ノートブックインスタンスの更新*をクリックします。



AWS Lambda サーバーレス関数

前述のように、**AWS Lambda 関数***は ***AWS SageMaker Notebook** インスタンス を起動する役割を担っています。

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



2. ページ上のすべての必須エントリを入力し、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。

aws Services Search [Option+S] N. Virgi AWSAdministratorAccess/kjian@

S3

Lambda > Functions > Create function

Create function [Info](#)

AWS Serverless Application Repository applications have moved to [Create application](#).

☒ **Author from scratch**
Start with a simple Hello World example.

☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

fsxn-demo-mlops

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.10

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ x86_64

☐ arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

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

aws Services Search [Option+S] N. Virgi AWSAdministratorAccess/kjian@

S3

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.10

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **Change default execution role**

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions
☒ Use an existing role
☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
service-role/fsxn-demo-mlops-role-585jzdny
[View the fsxn-demo-mlops-role-585jzdny role](#) on the IAM console.

► **Advanced settings**

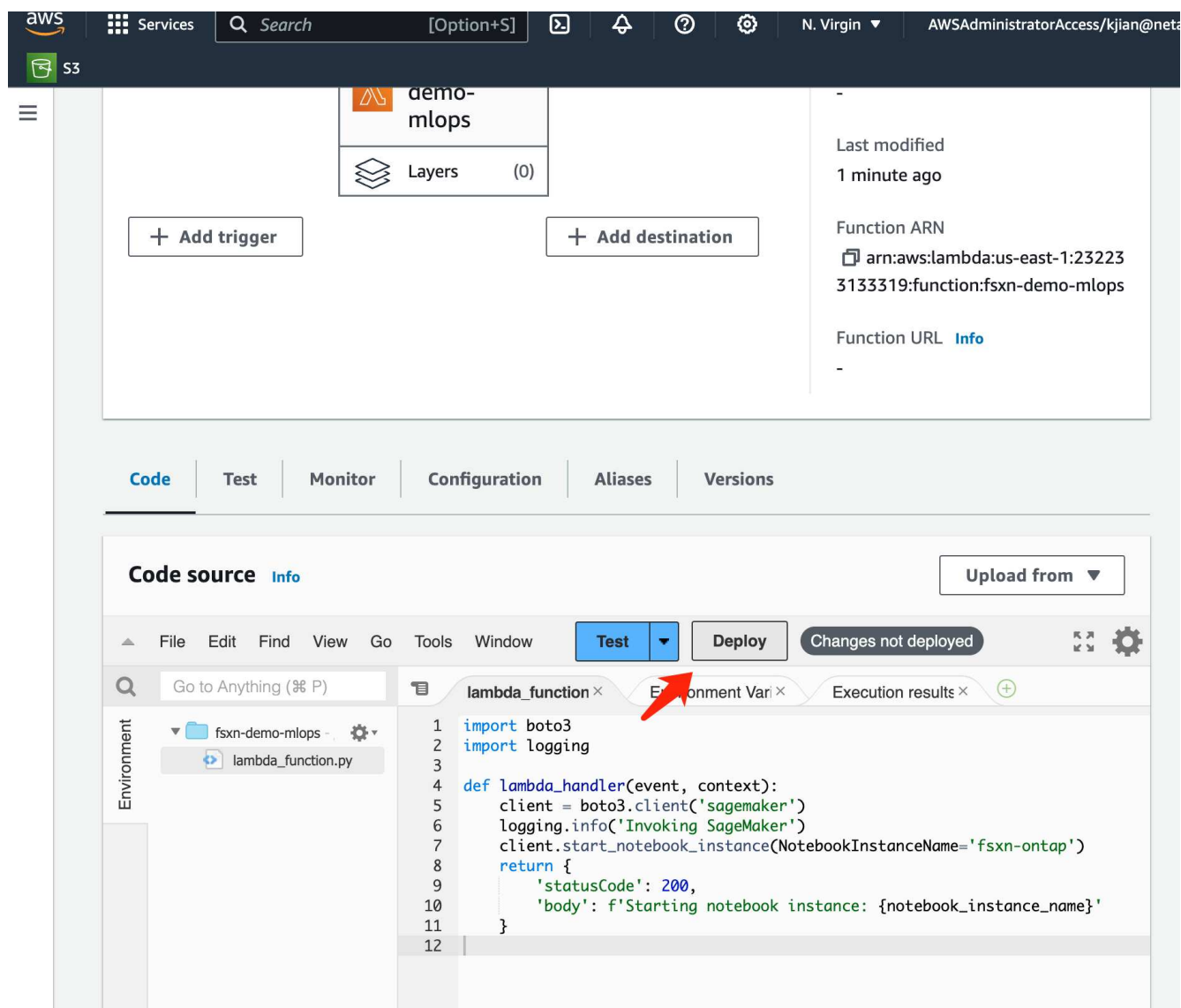
Cancel Create function

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. このコードの変更を適用するには、[デプロイ] 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、「トリガーの追加」ボタンをクリックします。

aws Services Search [Option+S] N. Virginia AWSAdministratorAccess/kjian@netapp.


S3


Lambda > Functions > fsxn-demo-mlops

fsxn-demo-mlops

Throttle Copy ARN Actions

▼ Function overview Info

 fsxn-demo-mlops

 Layers (0)

+ Add trigger + Add destination

Description -

Last modified 2 minutes ago

Function ARN
arn:aws:lambda:us-east-1:232233133319:function:fsxn-demo-mlops

Function URL Info -

7. ドロップダウン メニューから EventBridge を選択し、「新しいルールの作成」というラジオ 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。

aws Services Search [Option+S] N. Virginia AWSAdministratorAccess

S3

[Lambda](#) > Add trigger

Add trigger

Trigger configuration [Info](#)

EventBridge (CloudWatch Events)
aws asynchronous schedule management-tools

Rule
Pick an existing rule, or create a new one.

☒ Create a new rule
☐ Existing rules

Rule name
Enter a name to uniquely identify your rule.

mlops-retraining-trigger

Rule description
Provide an optional description for your rule.

Rule type
Trigger your target based on an event pattern, or based on an automated schedule.

☐ Event pattern
☒ Schedule expression

Schedule expression
Self-trigger your target on an automated schedule using [Cron or rate expressions](#). Cron expressions are in UTC.

rate(1 day)

e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

Lambda will add the necessary permissions for Amazon EventBridge (CloudWatch Events) to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add

2 段階の設定が完了すると、毎日、**AWS Lambda** 関数 が **SageMaker Notebook** を起動し、**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デプロイし、**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、モデルが最新の状態に保たれます。

これで、MLOps パイプラインの開発に関するチュートリアルは終了です。

著作権に関する情報

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。