



MLOps向けAWS FSx for NetApp ONTAP (FSxN)

NetApp Solutions

NetApp
April 10, 2024

This PDF was generated from https://docs.netapp.com/ja-jp/netapp-solutions/ai/mlops_fsxn_s3_integration.html on April 10, 2024. Always check docs.netapp.com for the latest.

目次

MLOps向けAWS FSx for NetApp ONTAP (FSxN)	1
パート1 - AWS FSx for NetApp ONTAP (FSxN) をプライベートS3バケットとしてAWS SageMakerに統合する	1
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AWS FSx for NetApp ONTAP (FSxN) を活用	15
パート3 -簡易化されたMLOpsパイプラインの構築 (CI/CT/CD)	24

MLOps向けAWS FSx for NetApp ONTAP (FSxN)

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)

このセクションでは、AIインフラ開発の実用的なアプリケーションについて詳しく説明し、FSxN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ワ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。

以下の記事では、

1. ["パート1 - AWS FSx for NetApp ONTAP \(FSxN\) 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AWS FSx for NetApp ONTAP \(FSxN\) 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了すると、FSxNを使用してMLOpsプロセスを合理化する方法について十分に理解できるようになります。

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

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を例として使用して、FSxN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。

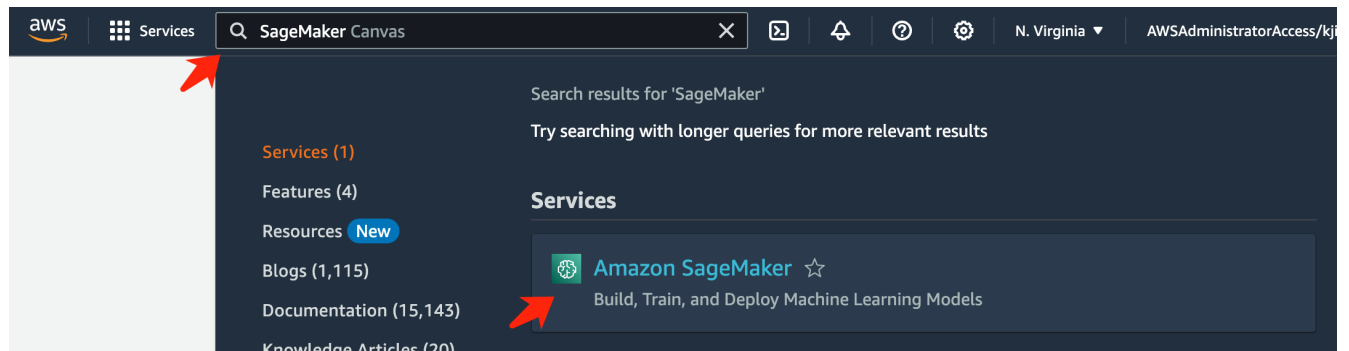
FSxNの詳細については、このプレゼンテーション (["ビデオリンク"](#))

ユーザーガイド

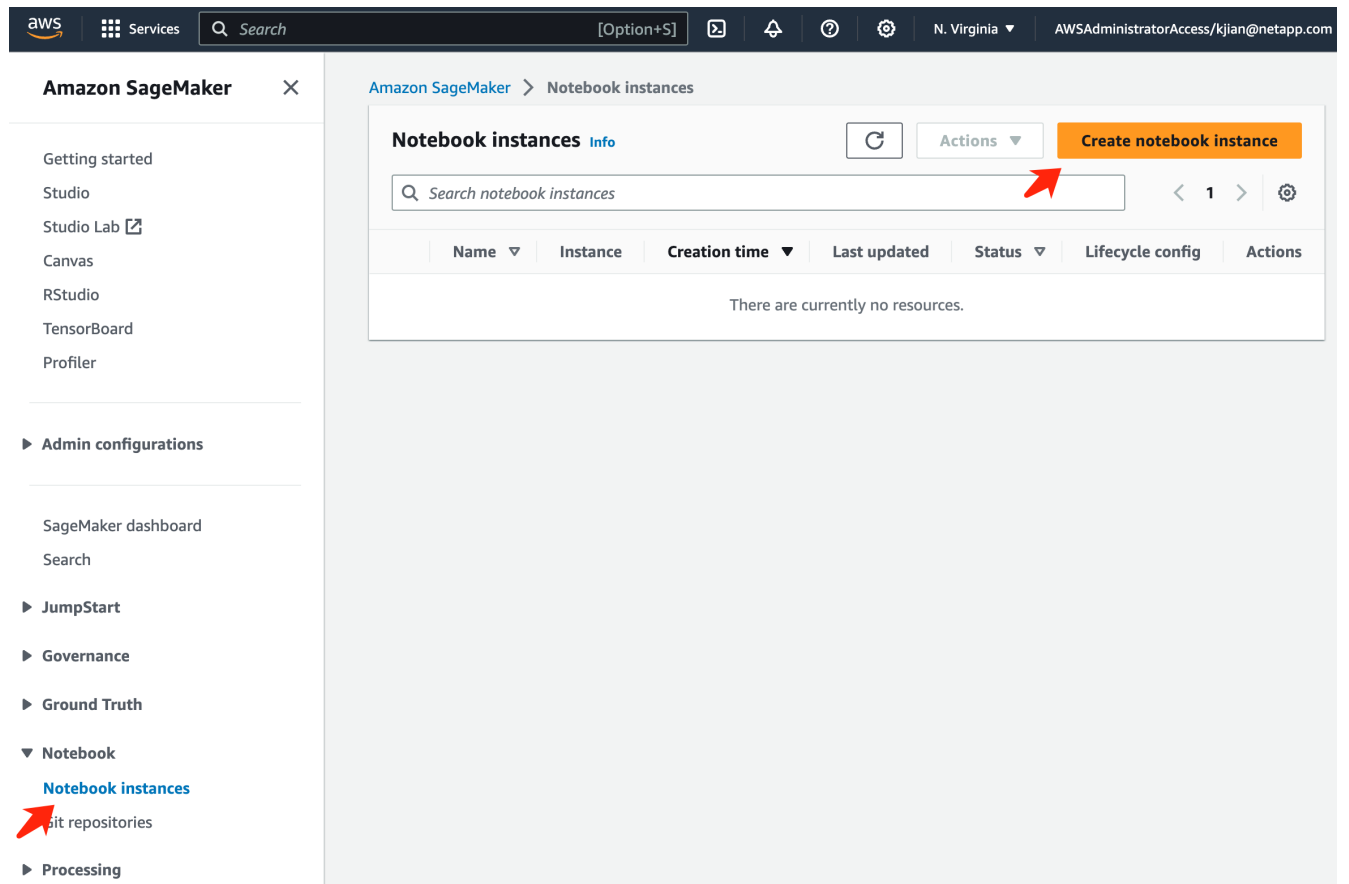
サーバの作成

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]パネルを展開します。
[その他のエントリ]はデフォルトのままにして、[VPC]、[サブネット]、および*を選択します。（このVPC と Subnet *は、あとでFSxN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。

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

fsxn-demo

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.t3.medium

Elastic Inference [Learn more](#)

none

Platform identifier [Learn more](#)

Amazon Linux 2, Jupyter Lab 3

► Additional configuration

Permissions and encryption

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMakerServiceCatalogProductsUseRole

Create role using the role creation wizard

Root access - optional

- ☒ Enable - Give users root access to the notebook
- ☐ Disable - Don't give users root access to the notebook
Lifecycle configurations always have root access

Encryption key - optional

Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption

▼ Network - optional

VPC - optional

Default vpc-0df3956ab1fca2ec9 (172.31.0.0/16)

Subnet

Choose a subnet in an availability zone supported by Amazon SageMaker.

subnet-00060df0d0f562672 (172.31.16.0/20) | us-east-1a

Security group(s)

sg-0a39b3985770e9256 (default) 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

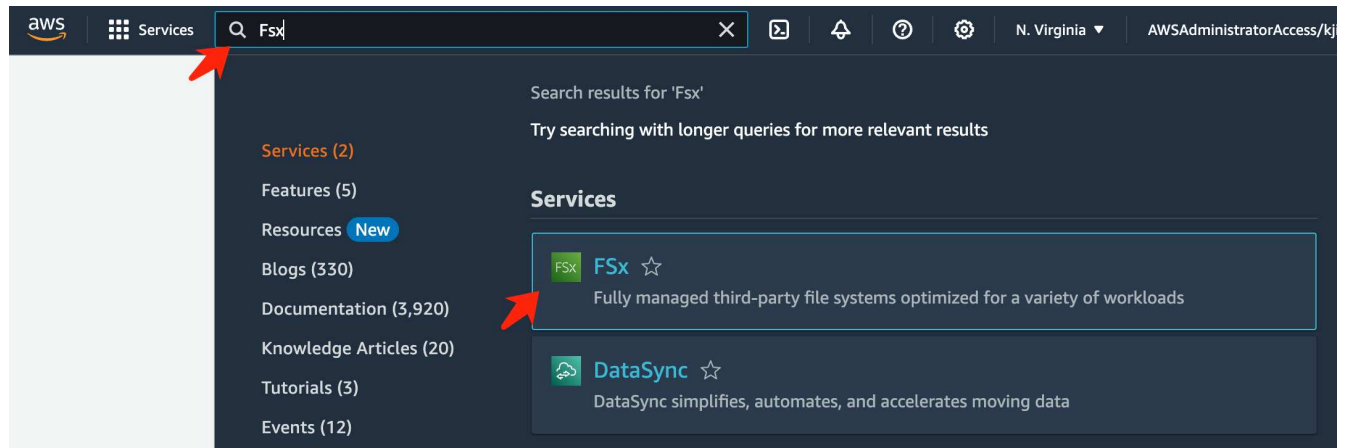
► Tags - optional

Cancel

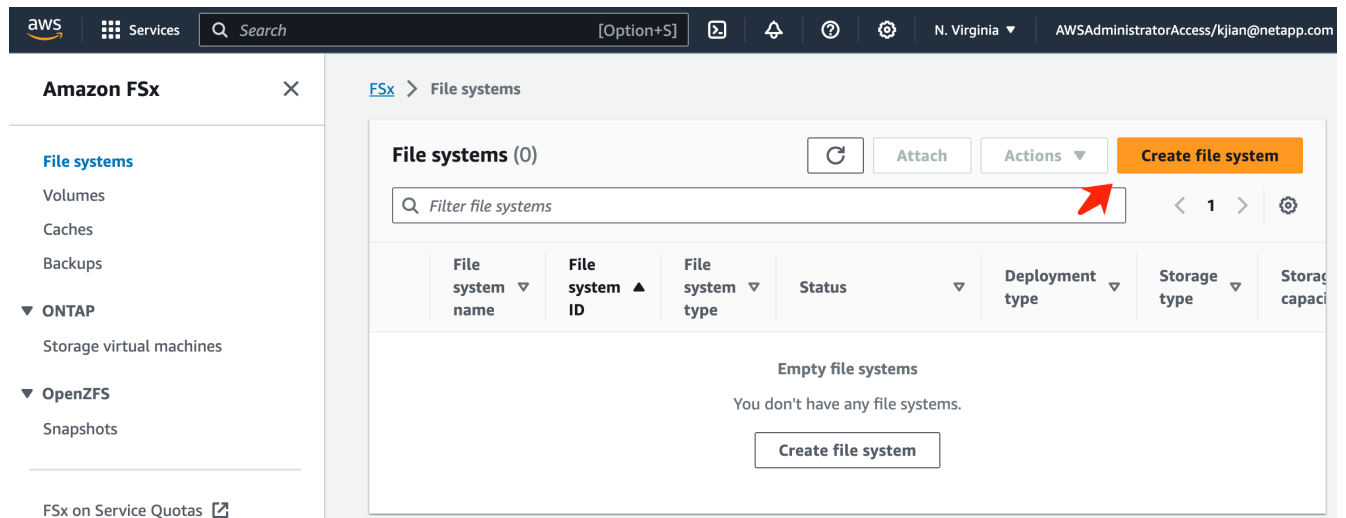
Create notebook instance

FSxN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 for NetApp ONTAP を選択し、Next *をクリックします。

aws Services Search [Option+S]

FSx > File systems > Create file system

Step 1
Select file system type

Step 2
Specify file system details

Step 3
Review and create

Select file system type

File system options

- ☒ Amazon FSx for NetApp ONTAP
- ☐ Amazon FSx for OpenZFS
- ☐ Amazon FSx for Windows File Server
- ☐ Amazon FSx for Lustre

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP provides feature-rich, high-performance, and highly-reliable storage built on NetApp's popular ONTAP file system and fully managed by AWS.

- Broadly accessible from Linux, Windows, and macOS compute instances and containers (running on AWS or on-premises) via industry-standard NFS, SMB, and iSCSI protocols.
- Provides ONTAP's popular data management capabilities like Snapshots, SnapMirror (for data replication), FlexClone (for data cloning), and data compression / deduplication.
- Delivers hundreds of thousands of IOPS with consistent sub-millisecond latencies, and up to 3 GB/s of throughput.
- Offers highly-available and highly-durable single-AZ and multi-AZ deployment options, SSD storage with support for cross-region replication, and built-in, fully managed backups.
- Supports dynamic scaling of your file system to fit your storage capacity and throughput needs.
- Automatically tiers infrequently-accessed data to capacity pool storage, a fully elastic storage tier that can scale to petabytes in size and is cost-optimized for infrequently-accessed data.
- Integrates with Microsoft Active Directory (AD) to support Windows-based environments and enterprises.

Cancel Next

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。

aws Services Search [Option+S]

FSx > File systems > Create file system

Step 1
Select file system type

Step 2
Specify file system details

Step 3
Review and create

Specify file system details

Creation method

- ☐ Quick create
Use recommended best-practice configurations. Most configuration options can be changed after the file system is created.
- ☒ Standard create
You set all of the configuration options, including specifying performance, networking, security, backups, and maintenance.

b. [File system name]*と[SSD storage capacity]*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. 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。

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. 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。

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. [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。

▶ Backup and maintenance - optional

▶ Tags - optional

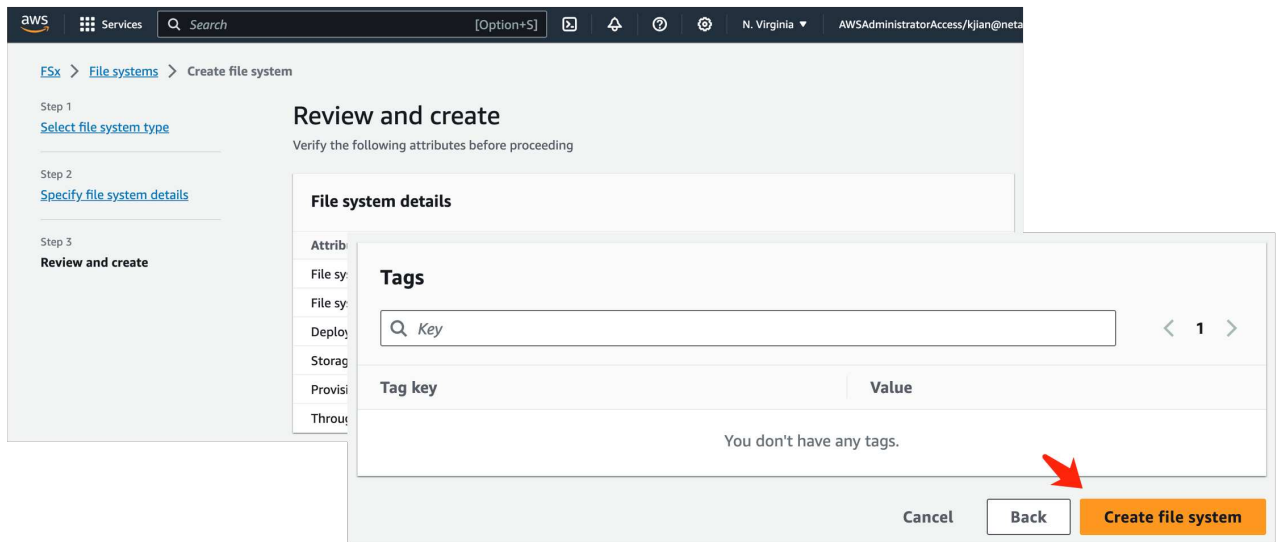
Cancel

Back

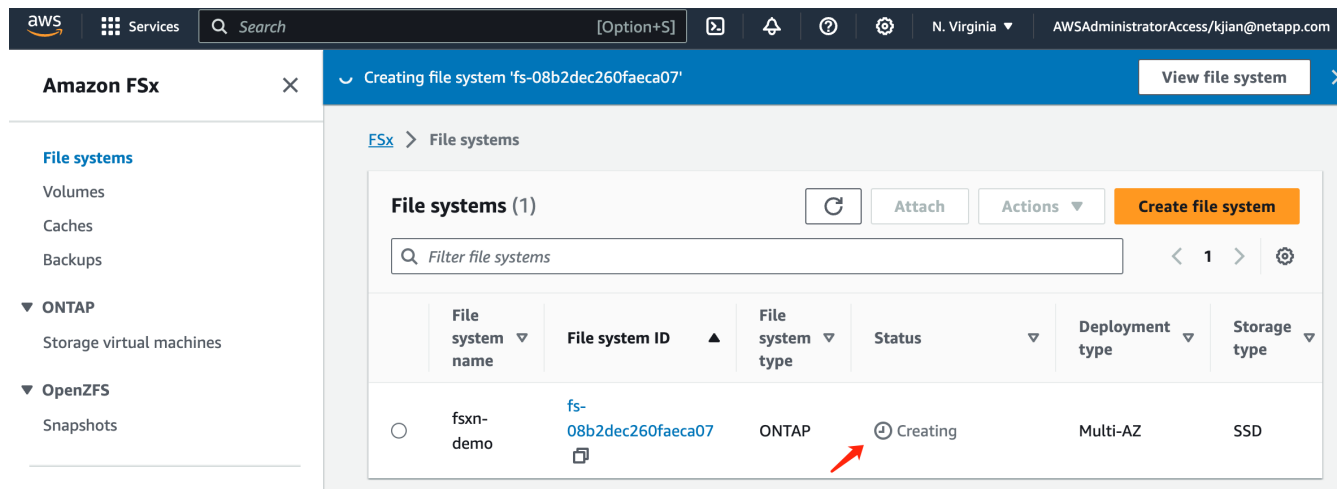
Next

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

8



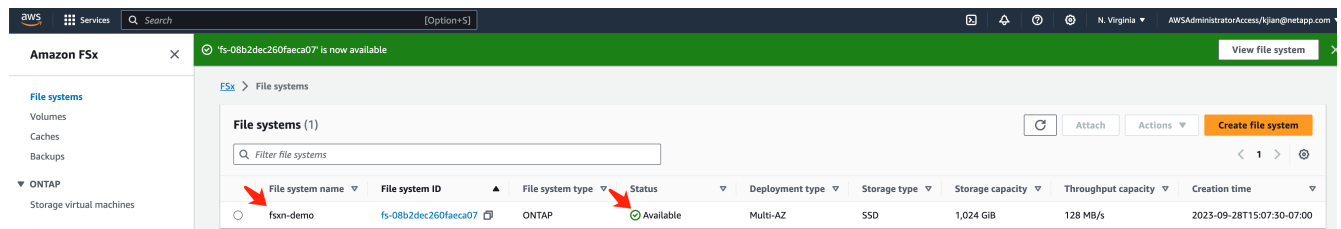
5. FSxファイルシステムのスピンアップには約* 20~40分*かかる場合があります。



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。

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

Amazon SageMaker ×

Getting started
Studio
Studio Lab [🔗](#)
Canvas
RStudio
TensorBoard

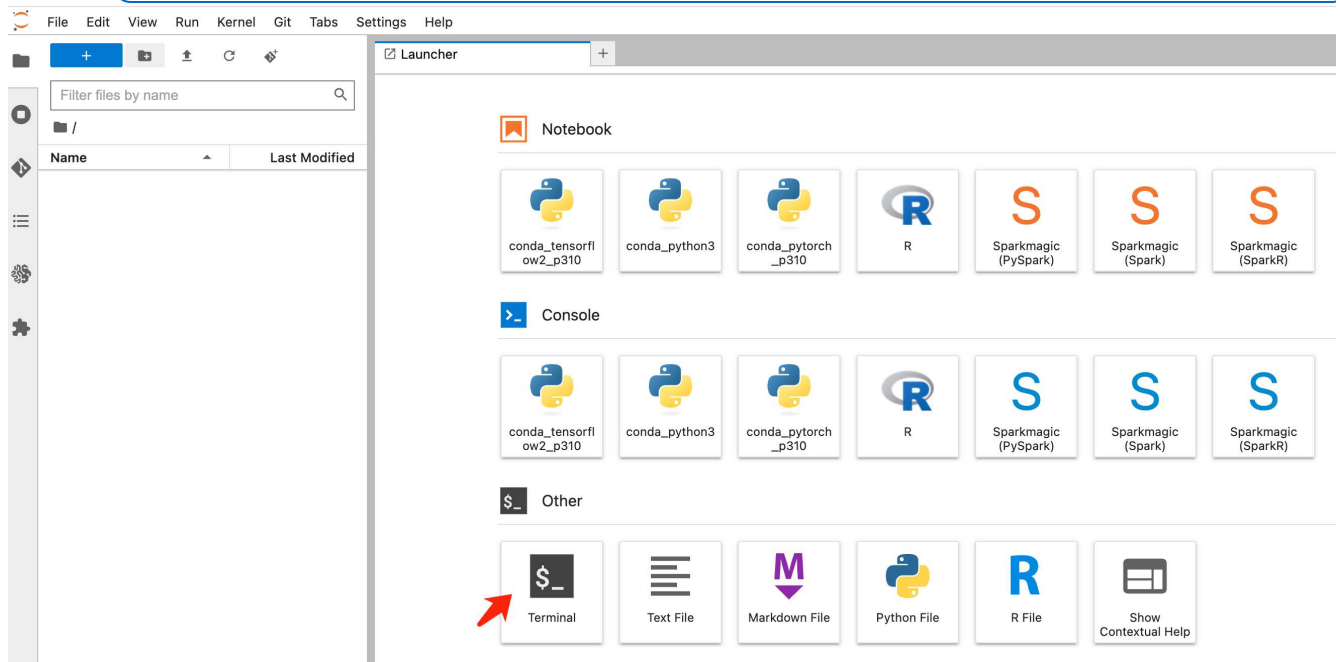
Amazon SageMaker > Notebook instances

Notebook instances [Info](#)

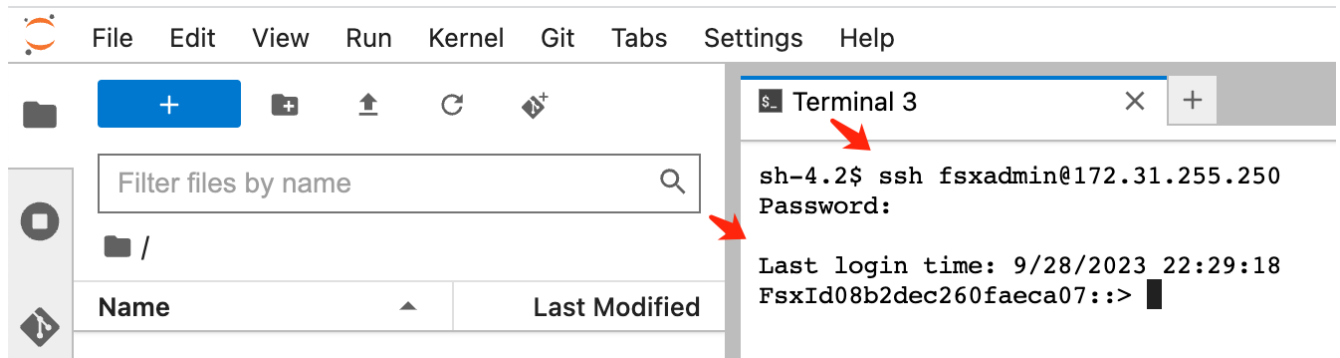
[🔄](#) [Actions](#) ▼ [Create notebook instance](#)

	Name ▼	Instance	Creation time ▼	Last updated	Status ▼	Lifecycle config	Actions
<input type="radio"/>	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. sshコマンド `ssh < admin user name >@< ONTAP server IP >`を入力して、FSxN 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で取得されます）。
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。



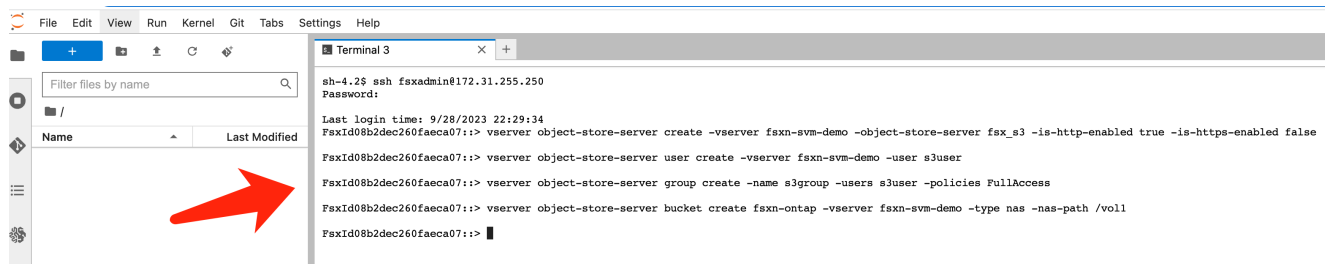
6. 次の順序でコマンドを実行します。
FSxNプライベートS3バケット名*の名前には* fsxn-ontap を使用します。
SVM *引数には Storage Virtual Machine名*を使用してください。

```
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
```



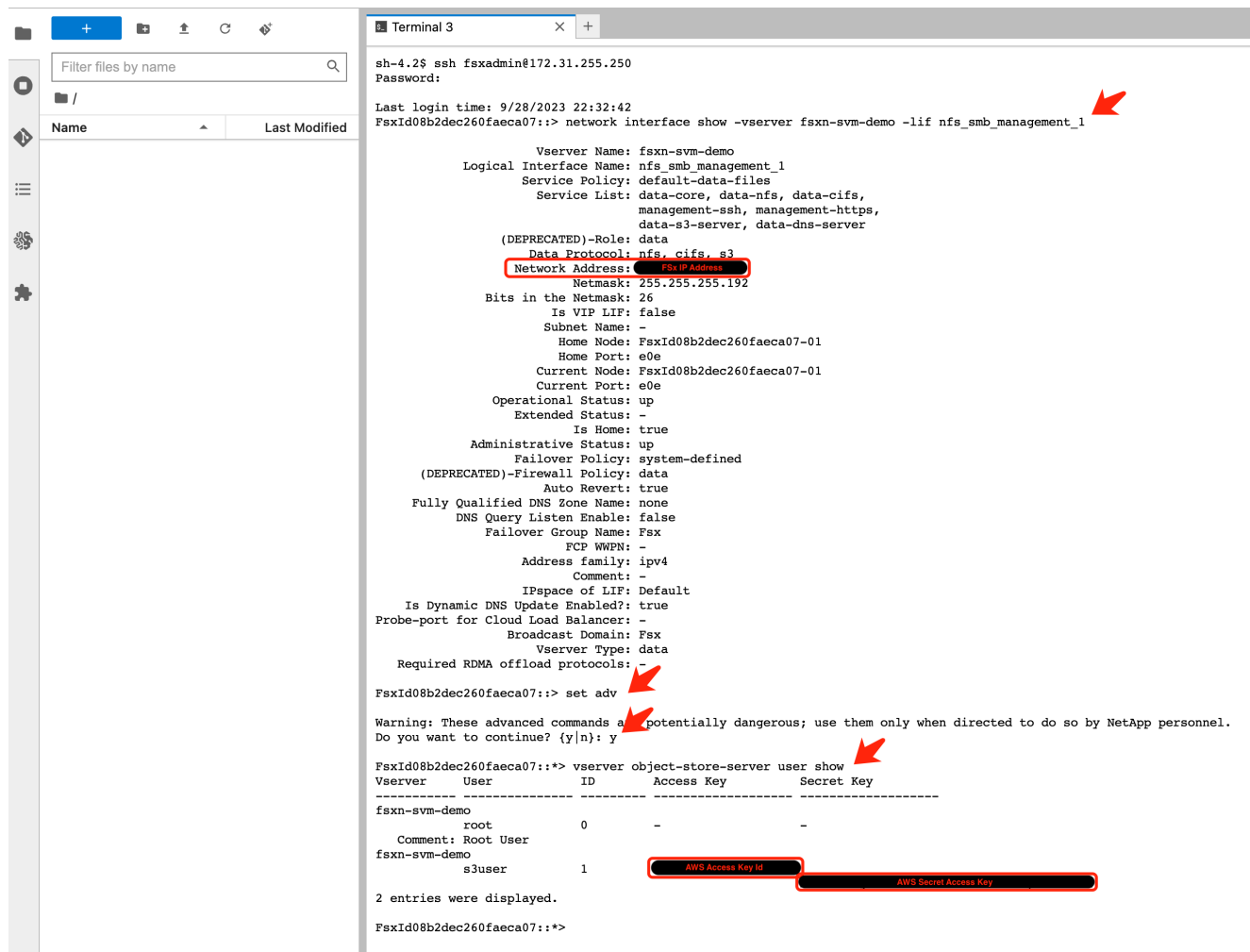
7. 次のコマンドを実行して、FSxNプライベートS3のエンドポイントIPとクレデンシャルを取得します。

```

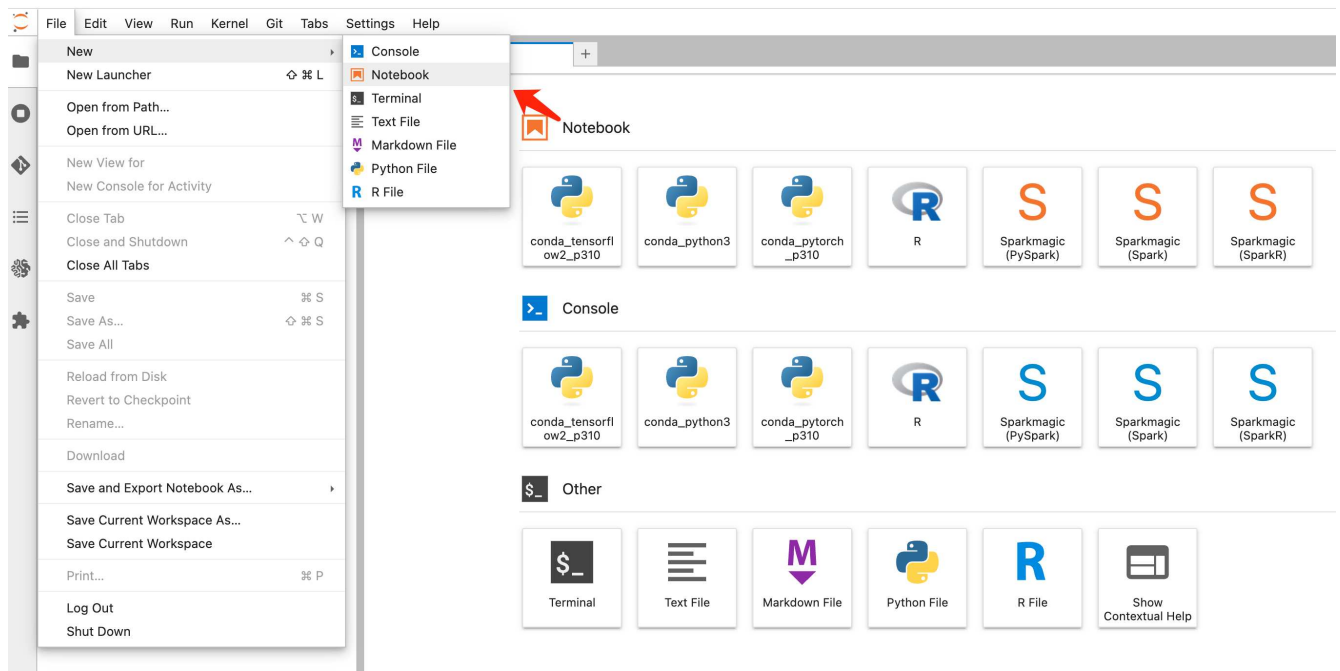
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1
set adv
vserver object-store-server user show

```

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。



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. 解決策を回避してFSxN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には、次のコードを使用します。

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 FSxN IP address>' # Please get
this IP address from FSxN
# ----- Manual configurations -----

# Workaround
## Permission patch
!mkdir -p vol1
!sudo mount -t nfs $fsx_endpoint_ip:/vol1 /home/ec2-user/SageMaker/vol1
!sudo chmod 777 /home/ec2-user/SageMaker/vol1

## Authentication for FSxN as a Private S3 Bucket
```

```

!aws configure set aws_access_key_id $aws_access_key_id
!aws configure set aws_secret_access_key $aws_secret_access_key

## Upload file to the FSxN Private S3 Bucket
%%capture
local_file_path: str = <Your local file path>

!aws s3 cp --endpoint-url http://$fsx_endpoint_ip /home/ec2-user
/SageMaker/$local_file_path s3://$bucket_name/$local_file_path

# Read data from FSxN Private S3 bucket
## Initialize a s3 resource client
import boto3

# Get session info
region_name = boto3.session.Session().region_name

# Initialize FsxN S3 bucket object
# --- Start integrating SageMaker with FSXN ---
# This is the only code change we need to incorporate SageMaker with
FSXN
s3_client: boto3.client = boto3.resource(
    's3',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key,
    use_ssl=False,
    endpoint_url=f'http://{fsx_endpoint_ip}',
    config=boto3.session.Config(
        signature_version='s3v4',
        s3={'addressing_style': 'path'}
    )
)
# --- End integrating SageMaker with FSXN ---

## Read file byte content
bucket = s3_client.Bucket(bucket_name)

binary_data = bucket.Object(data.filename).get()['Body']

```

これで、FSxN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FSxN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: FSxN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、FSxN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、FSxN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: FSxNにファイルをアップロードするときに、「* PutObject操作を呼び出したとき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FSxNプライベートS3バケットにアクセスするには、AWSクレデンシアルをFSxNクレデンシアルに切り替えます。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: FSxN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、FSxN 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。 その他。

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AWS FSx for NetApp ONTAP (FSxN) を活用

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

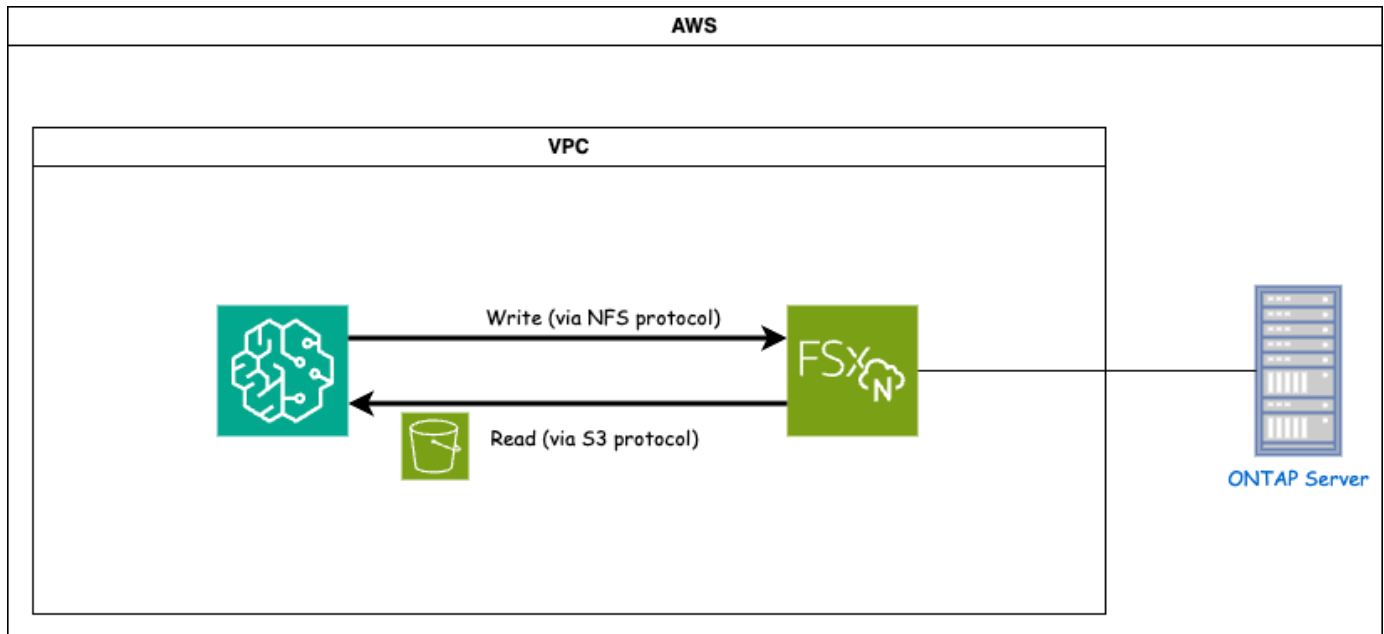
このチュートリアルでは、コンピュータビジョン分類プロジェクトの実践的な例を示し、SageMaker環境内でFSxNをデータソースとして使用するMLモデルを構築するための実践的な経験を提供します。このプロジェクトでは、ディープラーニングフレームワークであるPyTorchを使用して、タイヤの画像に基づいてタイヤの品質を分類することに焦点を当てています。Amazon SageMakerのデータソースとしてFSxNを使用した機械学習モデルの開発に重点を置いています。

FSxNとは

Amazon FSx for NetApp ONTAPは、AWSが提供するフルマネージドストレージ解決策です。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 for NetApp 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



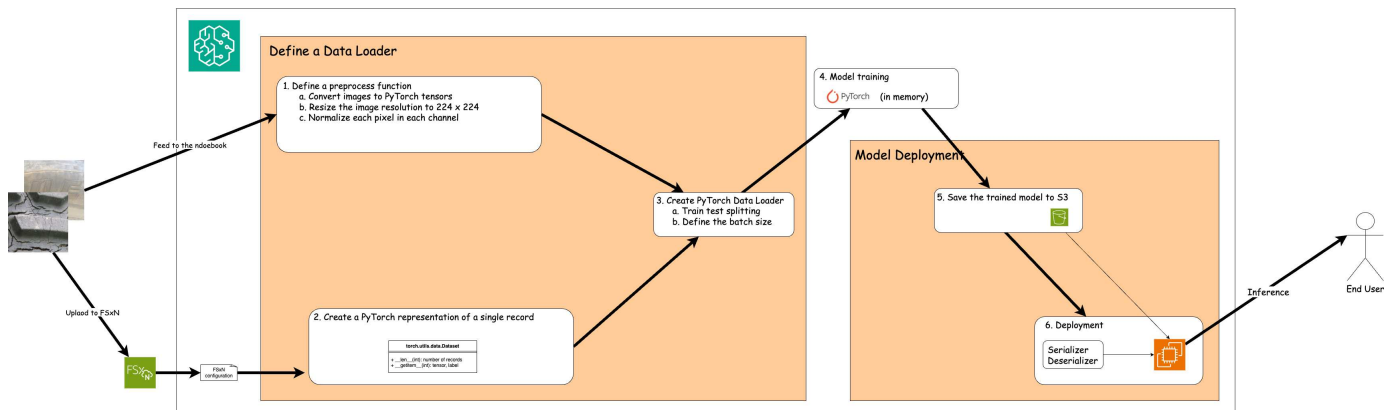
FSxN（Amazon FSx for NetApp ONTAP）は、AWSのストレージサービスです。これには、NetApp ONTAP システムで実行されているファイルシステムと、接続するAWSで管理されるSystem Virtual Machine（SVM）が含まれます。次の図では、AWSで管理されるNetApp ONTAPサーバがVPCの外部に配置されています。SVMはSageMakerとNetApp ONTAPシステムの仲介役として機能し、SageMakerからの処理要求を受け取り、基盤となるストレージに転送します。FSxNにアクセスするには、SageMakerをFSxN展開と同じVPC内に配置する必要があります。この構成により、SageMakerとFSxN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、FSxN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、FSxNファイルシステムは作成後に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、FSxN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、SageMakerと連携してモデルの開発とトレーニングを行いながら、FSxN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、FSxN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 - AWS FSx for NetApp ONTAP（FSxN）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



FSxN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、デプロイの3つの主要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、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FSxNから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、展開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

データローダー

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tutorial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使用して**FSxN**からバイナリデータを取得します。**FSxN**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。

プライベート**S3**リポジトリとしての**FSxN**

```

seed = 77 # Random seed
bucket_name = '<Your ONTAP bucket name>' # The bucket
name in ONTAP
aws_access_key_id = '<Your ONTAP bucket key id>' # Please get
this credential from ONTAP
aws_secret_access_key = '<Your ONTAP bucket access key>' # Please get
this credential from ONTAP
fsx_endpoint_ip = '<Your FSxN IP address>' # Please get
this IP address from FSxN

```

```

import boto3

# Get session info
region_name = boto3.session.Session().region_name

# Initialize FsxN S3 bucket object
# --- Start integrating SageMaker with FSxN ---
# This is the only code change we need to incorporate SageMaker with FSxN
s3_client: boto3.client = boto3.resource(
    's3',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key,
    use_ssl=False,
    endpoint_url=f'http://{fsx_endpoint_ip}',
    config=boto3.session.Config(
        signature_version='s3v4',
        s3={'addressing_style': 'path'}
    )
)
# s3_client = boto3.resource('s3')
bucket = s3_client.Bucket(bucket_name)
# --- End integrating SageMaker with FSxN ---

```

SageMakerでFSxNからデータを読み取るために、S3プロトコルを使用してFSxNストレージを指すハンドラが作成されます。これにより、FSxNをプライベートS3バケットとして扱うことができます。ハンドラの設定では、FSxN 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を指定します。これらの設定項目の入手方法については、次のWebサイトにあるドキュメントを参照してください。["パート1 - AWS FSx for NetApp ONTAP \(FSxN\) 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。

```

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*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.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())

```

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ル化するカスタムシリアライザ*TyreQualitySerializer*を定義します。**TyreQualityPredictor***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、推論のエントリー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）

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

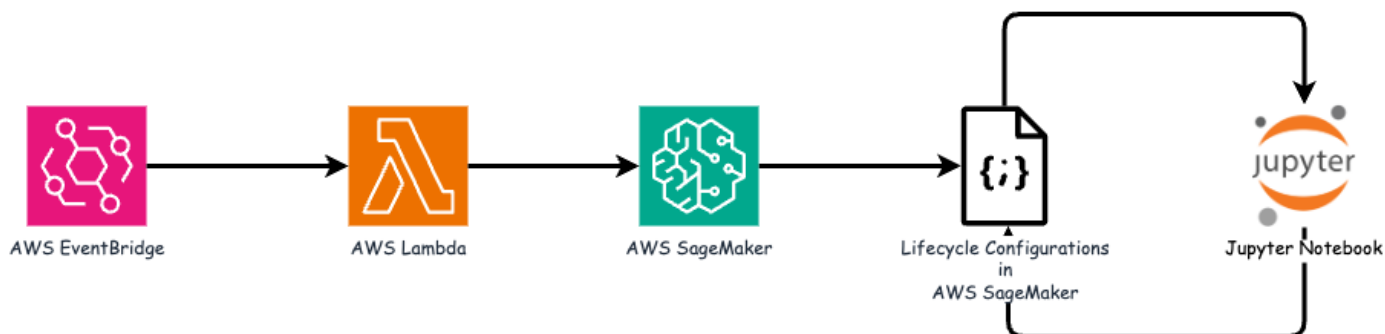
機能性	名前	コメント（Comment）
データストレージ	AWS FSxN	を参照してください" パート1 - AWS FSx for NetApp ONTAP（FSxN）をプライベートS3バケットとしてAWS SageMakerに統合する "。
データサイエンスIDE	AWS SageMaker	このチュートリアルは、で紹介されているJupyterノートブックをベースにしています。" パート2 - SageMakerのモデルトレーニングのデータソースとしてAWS FSx for NetApp ONTAP（FSxN）を活用 "。

機能性	名前	コメント (Comment)
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

- 事前設定されたFSxNファイルシステム。このチュートリアルでは、FSxN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FSxNファイルシステムと同じVPCを共有するように構成された* 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*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。

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 notebook instance. Each domain receives a personal and private Amazon SageMaker notebook instance.

► Domain structure diagram

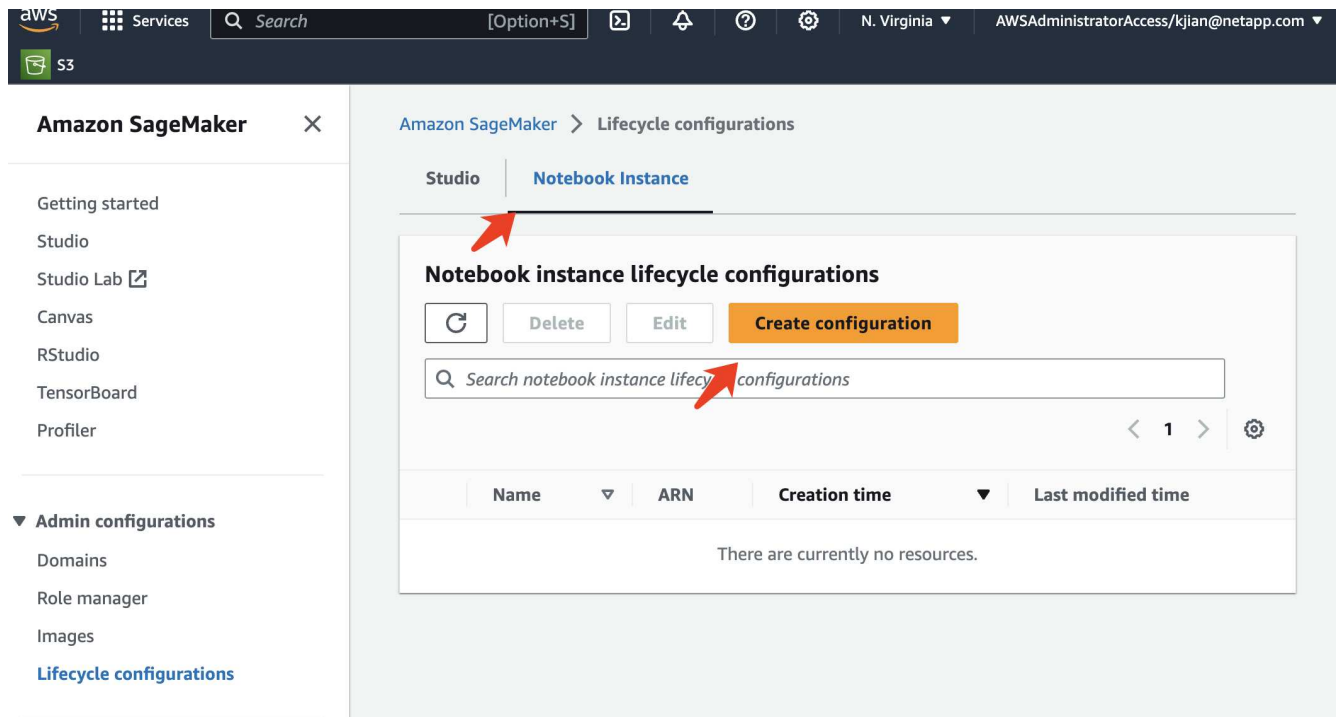
Domains (4)

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. [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AWS FSx for NetApp ONTAP \(FSxN\) を活用](#)"。

Configuration setting

Name

fsxn-demo-lifecycle-callback

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

Scripts

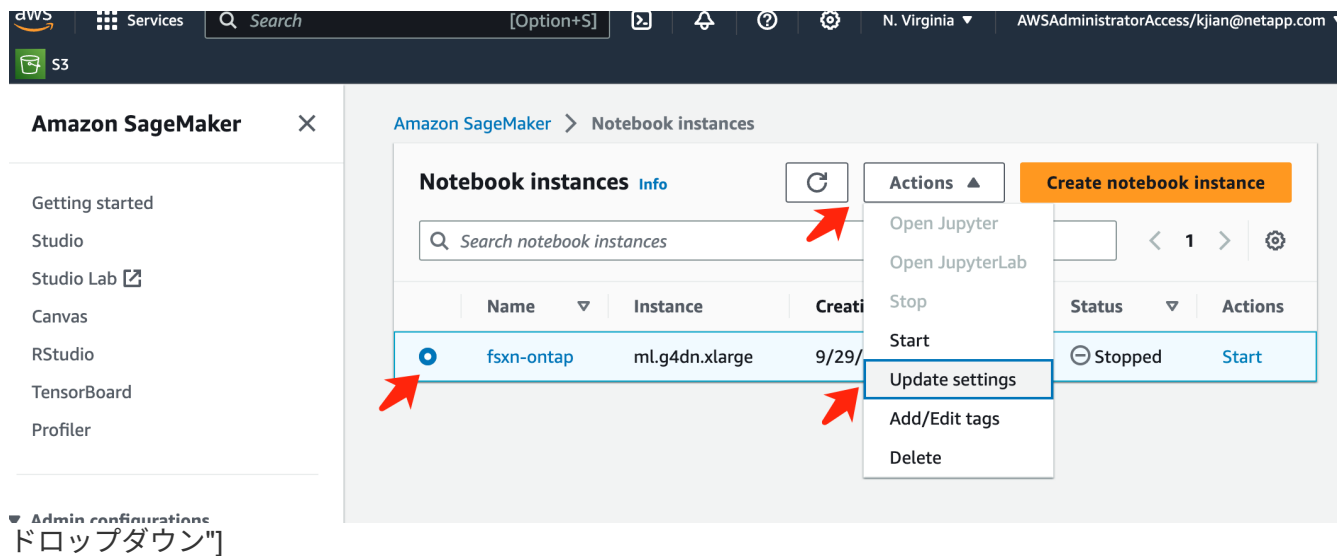
Start notebook | Create notebook

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](#)

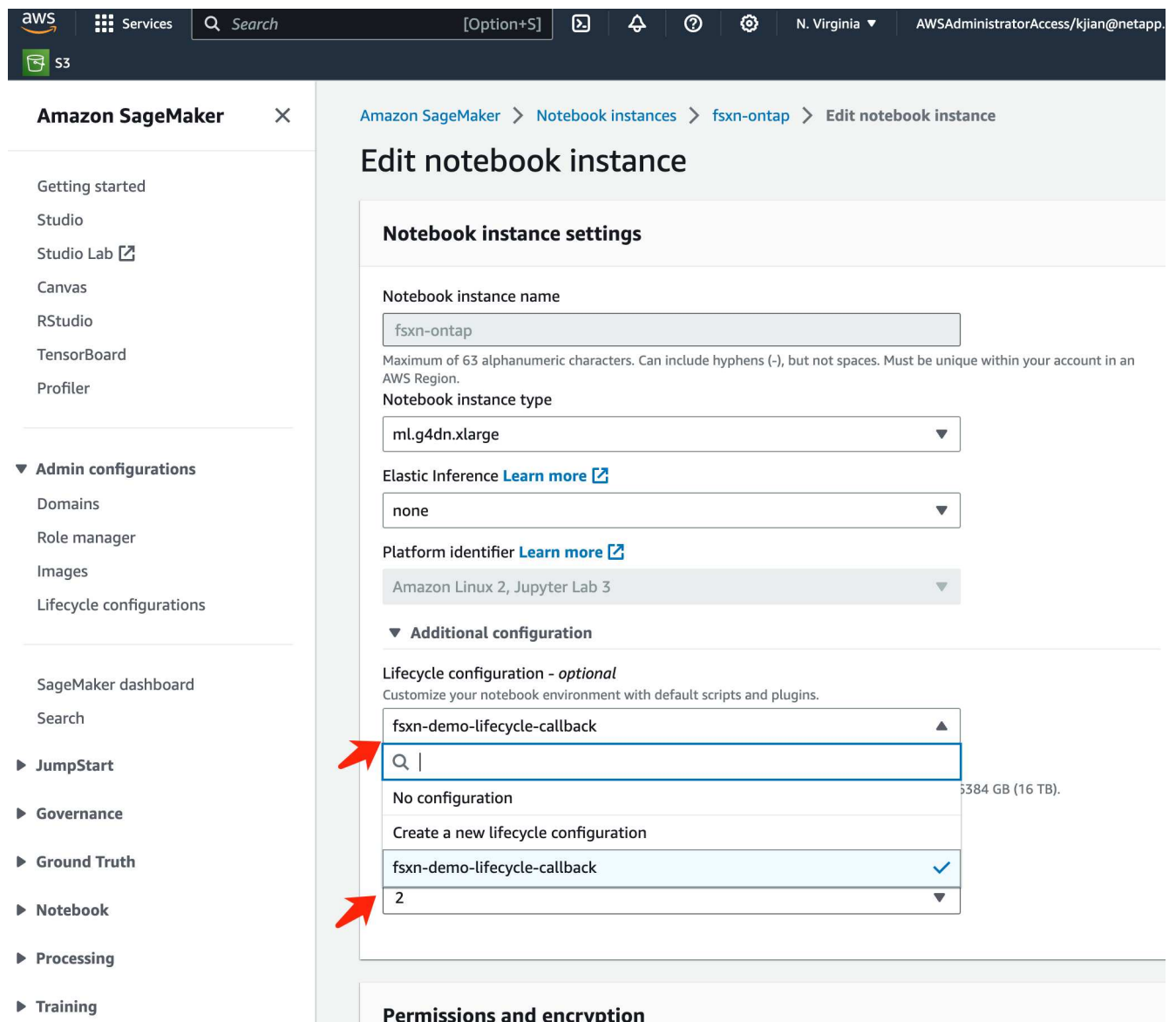
```
1 #!/bin/bash
2
3 set -e
4 sudo -u ec2-user -i <<'EOF'
5 # 1. Retraining and redeploying the model
6 NOTEBOOK_FILE=/home/ec2-user/SageMaker/tyre_quality_classification_local_training.ipynb
7 echo "Activating conda env"
8 source /home/ec2-user/anaconda3/bin/activate torch_p310
9 nohup jupyter nbconvert "$NOTEBOOK_FILE" --ExecutePreprocessor.kernel_name=python --execute --to nbconvert_pid=$!
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、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。



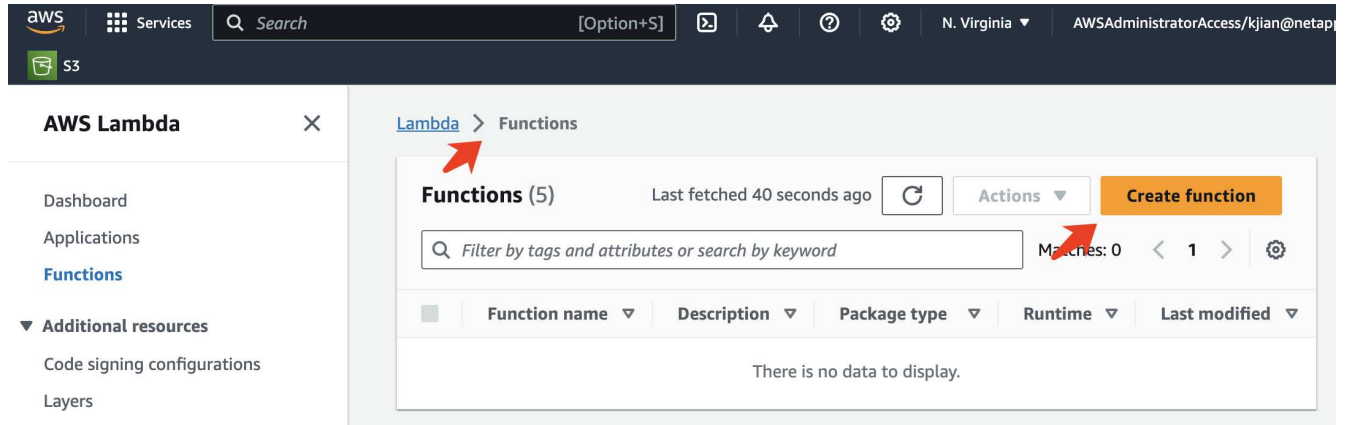
6. 作成した* Lifecycle configuration を選択し、Update notebook instance *をクリックします。



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。

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、* Create Function * 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**

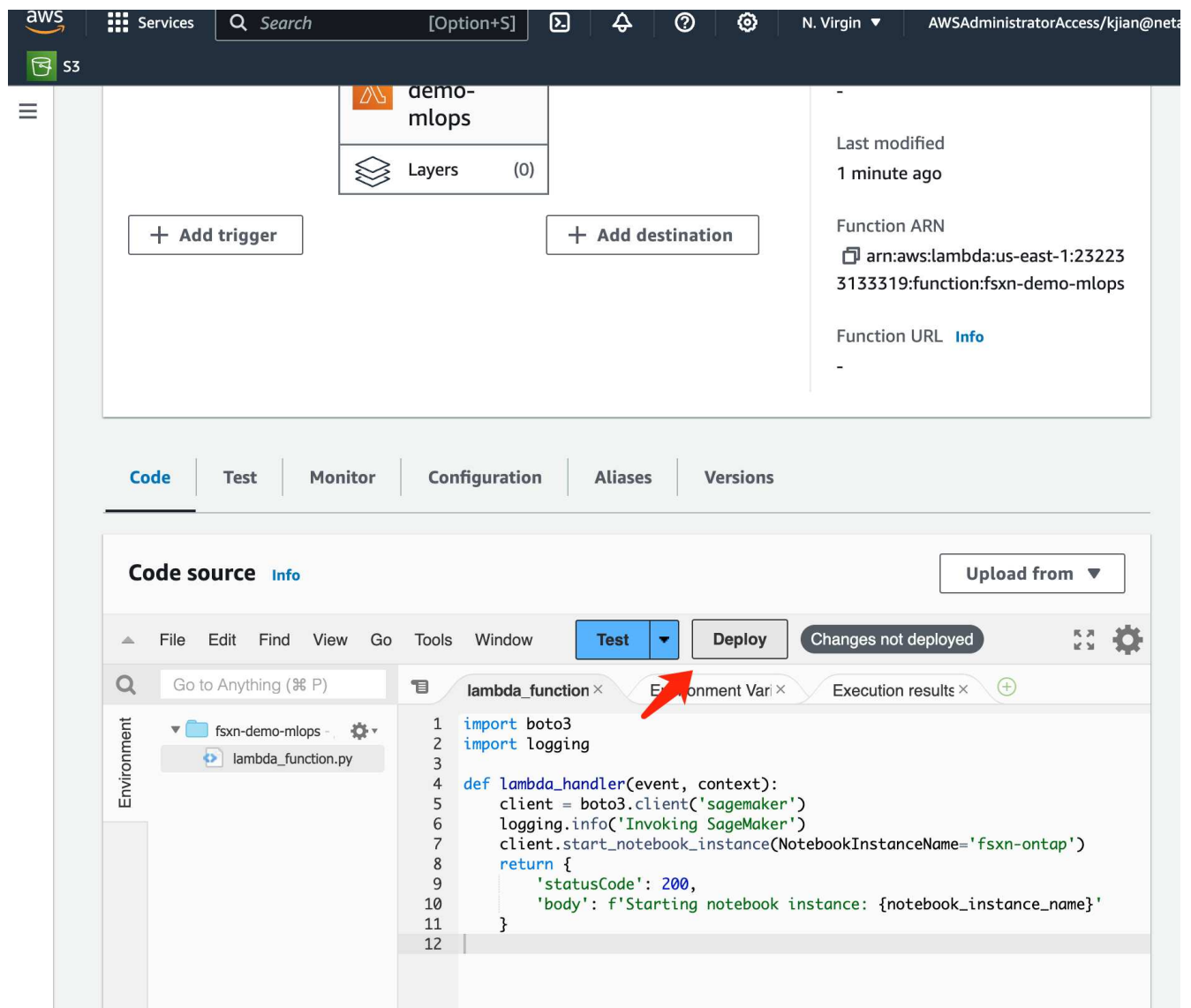
Cancel Create function

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。

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、[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。

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、* AWS Lambda関数*が毎日* SageMaker Notebook を開始し、FSxN 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

Copyright © 2024 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。