



Amazon FSX per NetApp ONTAP (FSX ONTAP) per MLOps

NetApp Solutions

NetApp
January 09, 2025

Sommario

- Amazon FSX per NetApp ONTAP (FSX ONTAP) per MLOps 1
- Amazon FSX per NetApp ONTAP (FSX ONTAP) per MLOps 1
- Parte 1 - integrazione di Amazon FSX per NetApp ONTAP (FSX ONTAP) come bucket S3 privato in
 AWS SageMaker 1
- Parte 2 - utilizzo di AWS Amazon FSX per NetApp ONTAP (FSX ONTAP) come origine dati per il
 training sui modelli in SageMaker 15
- Parte 3 - creazione di Una pipeline MLOps semplificata (ci/CT/CD) 24

Amazon FSX per NetApp ONTAP (FSX ONTAP) per MLOps

Amazon FSX per NetApp ONTAP (FSX ONTAP) per MLOps

Questa sezione illustra l'applicazione pratica dello sviluppo di infrastrutture ai, fornendo una panoramica end-to-end della costruzione di una pipeline MLOps utilizzando FSX ONTAP. Con tre esempi completi, ti guida a soddisfare le tue esigenze MLOps tramite questa potente piattaforma per la gestione dei dati.

Autore(i):

Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

Questi articoli si concentrano su:

1. ["Parte 1 - integrazione di Amazon FSX per NetApp ONTAP \(FSX ONTAP\) come bucket S3 privato in AWS SageMaker"](#)
2. ["Parte 2 - utilizzo di Amazon FSX per NetApp ONTAP \(FSX ONTAP\) come origine dati per il training sui modelli in SageMaker"](#)
3. ["Parte 3 - creazione di Una pipeline MLOps semplificata \(ci/CT/CD\)"](#)

Al termine di questa sezione, avrete acquisito una solida comprensione di come utilizzare FSX ONTAP per semplificare i processi MLOps.

Parte 1 - integrazione di Amazon FSX per NetApp ONTAP (FSX ONTAP) come bucket S3 privato in AWS SageMaker

Questa sezione fornisce una guida alla configurazione di FSX ONTAP come bucket S3 privato tramite AWS SageMaker.

Autore(i):

Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

Introduzione

Utilizzando SageMaker come esempio, questa pagina fornisce istruzioni sulla configurazione di FSX ONTAP come bucket S3 privato.

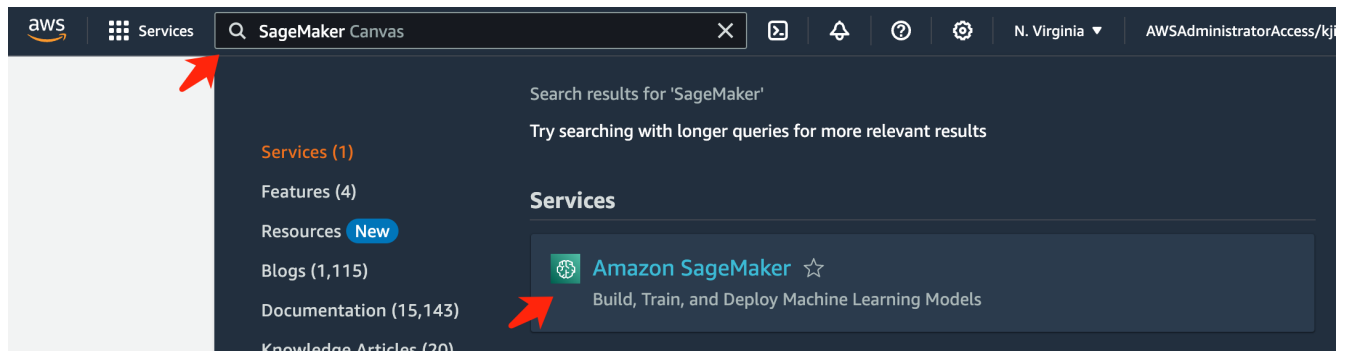
Per ulteriori informazioni su FSX ONTAP, date un'occhiata a questa presentazione (["Collegamento video"](#))

Guida dell'utente

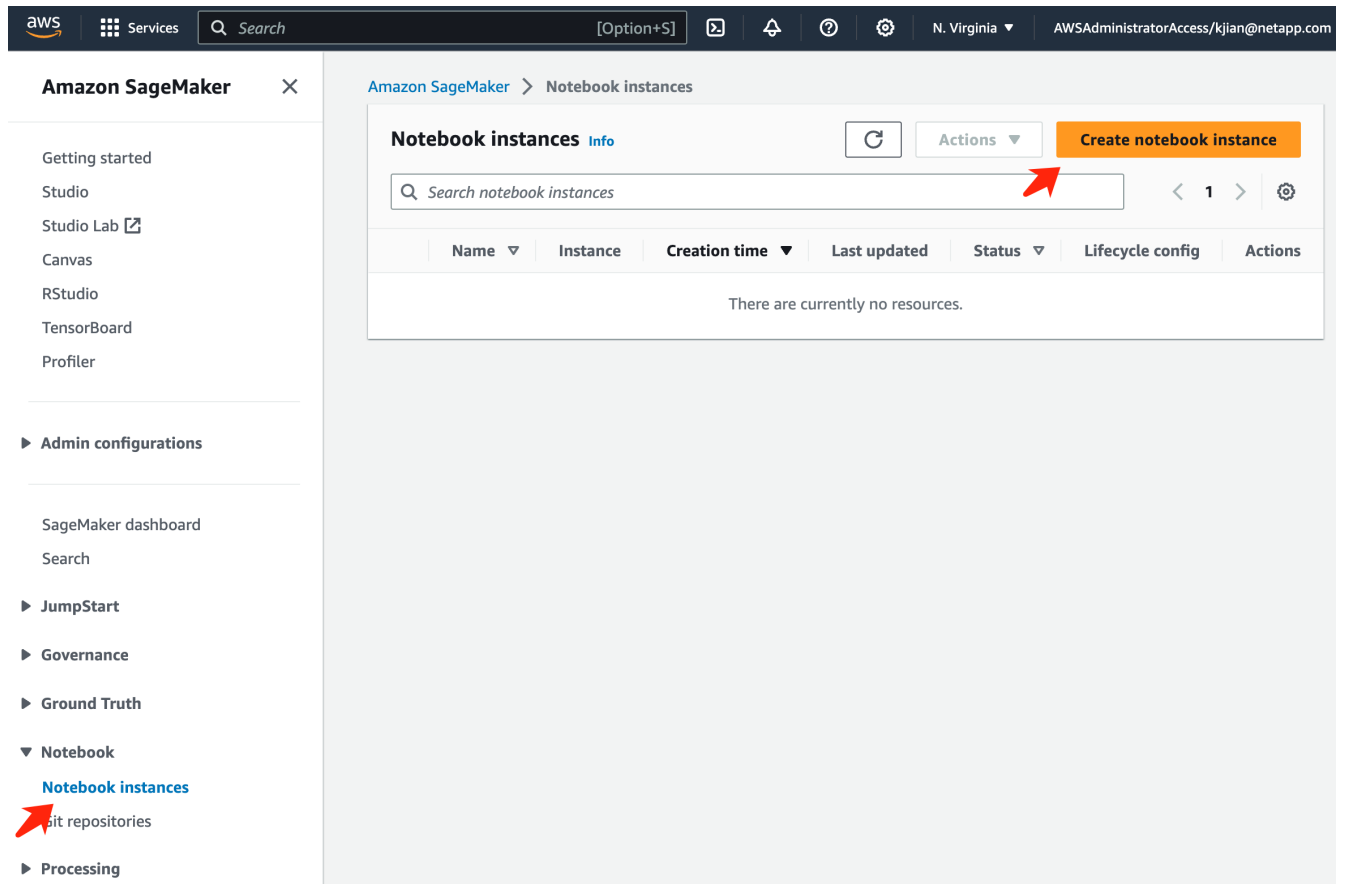
Creazione server

Creare un'istanza di notebook SageMaker

1. Apri la console AWS. Nel pannello di ricerca, cerca SageMaker e fai clic sul servizio **Amazon SageMaker**.



2. Aprire **istanze notebook** nella scheda notebook, fare clic sul pulsante arancione **Crea istanza notebook**.



3. Nella pagina di creazione, immettere il nome dell'istanza del notebook* espandere il pannello **rete** lasciare le altre voci predefinite e selezionare un gruppo **VPC**, **sottorete** e **protezione**. (Questa **VPC** e **sottorete** verranno utilizzate per creare il file system FSX ONTAP in seguito) fate clic sul pulsante arancione **Crea istanza notebook** in basso a destra.

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-00660df0d0f562672 (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](#)

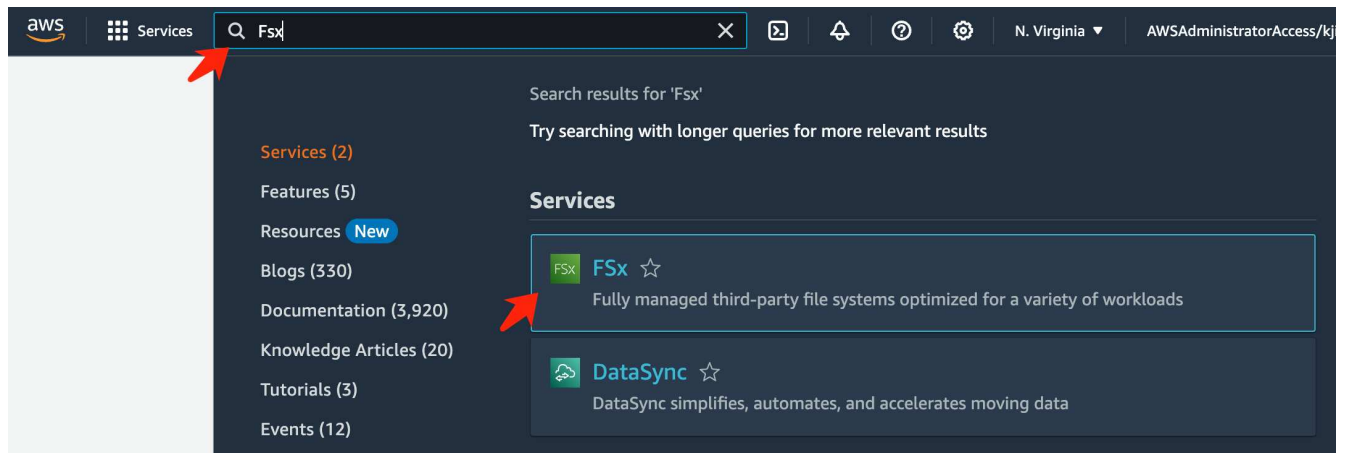
▶ Git repositories - optional

▶ Tags - optional

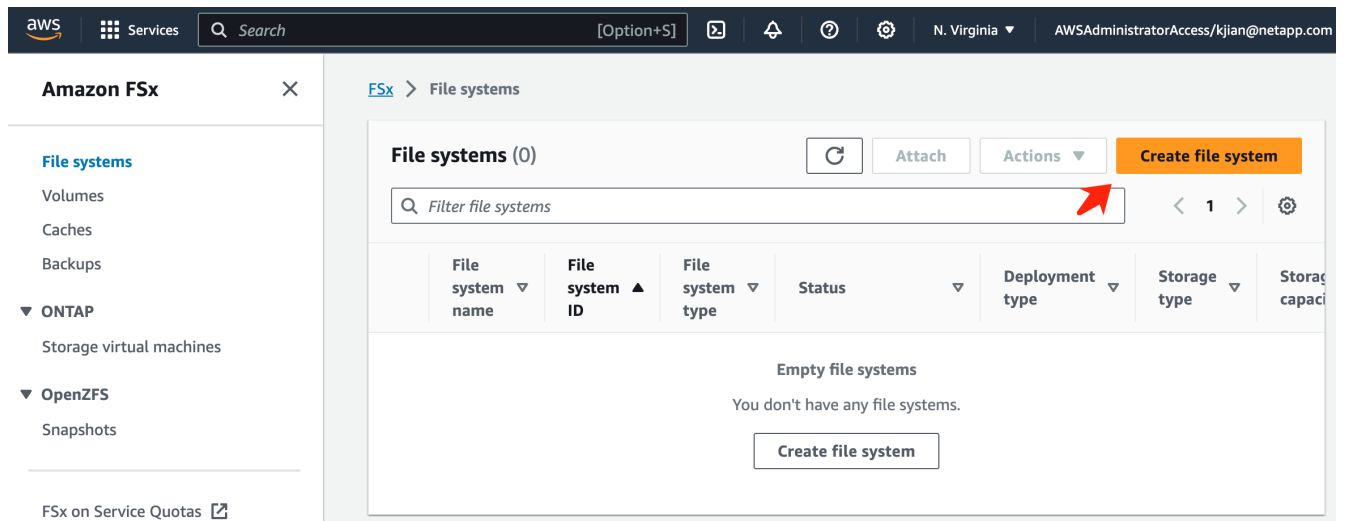
Cancel Create notebook instance

Crea un file system FSX ONTAP

1. Apri la console AWS. Nel pannello di ricerca, cercate FSX e fate clic sul servizio **FSX**.



2. Fare clic su **Crea file system**.



3. Selezionare la prima scheda **FSX ONTAP** e fare clic su **Avanti**.

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

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. Nella pagina di configurazione dei dettagli.
 - a. Selezionare l'opzione **creazione standard**.

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

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. Immettere il **nome del file system** e la **capacità di archiviazione SSD**.

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. Assicurarsi di utilizzare **VPC** e **subnet** uguali all'istanza **SageMaker notebook**.

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. Immettere il nome **Storage Virtual Machine** e **specificare una password** per la SVM (Storage Virtual Machine).

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

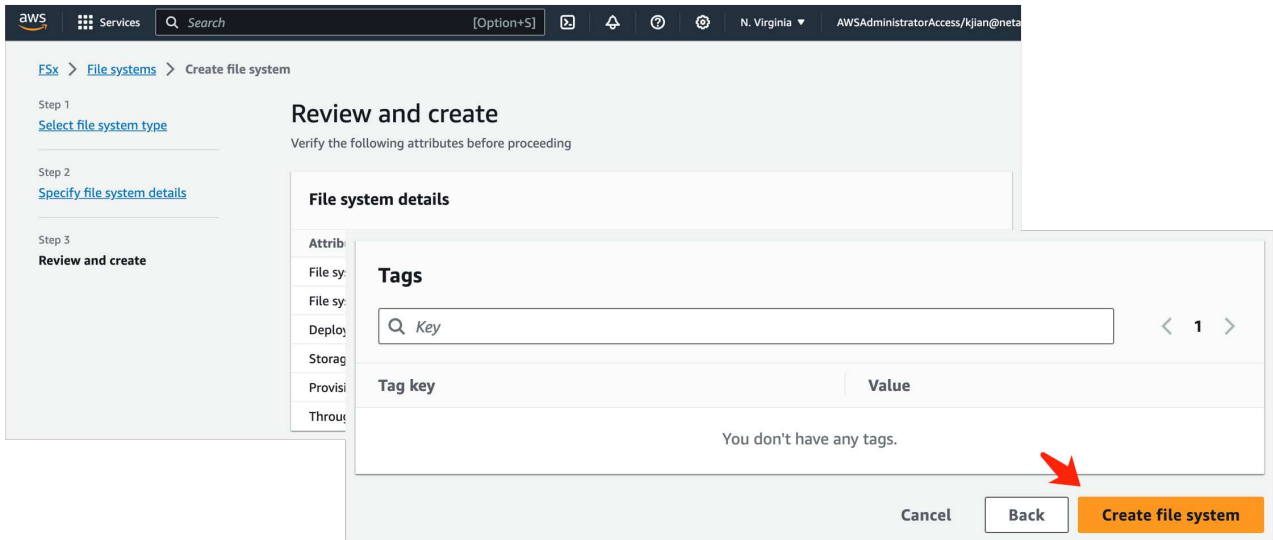
e. Lasciare le altre voci predefinite e fare clic sul pulsante arancione **Avanti** in basso a destra.

► **Backup and maintenance - optional**

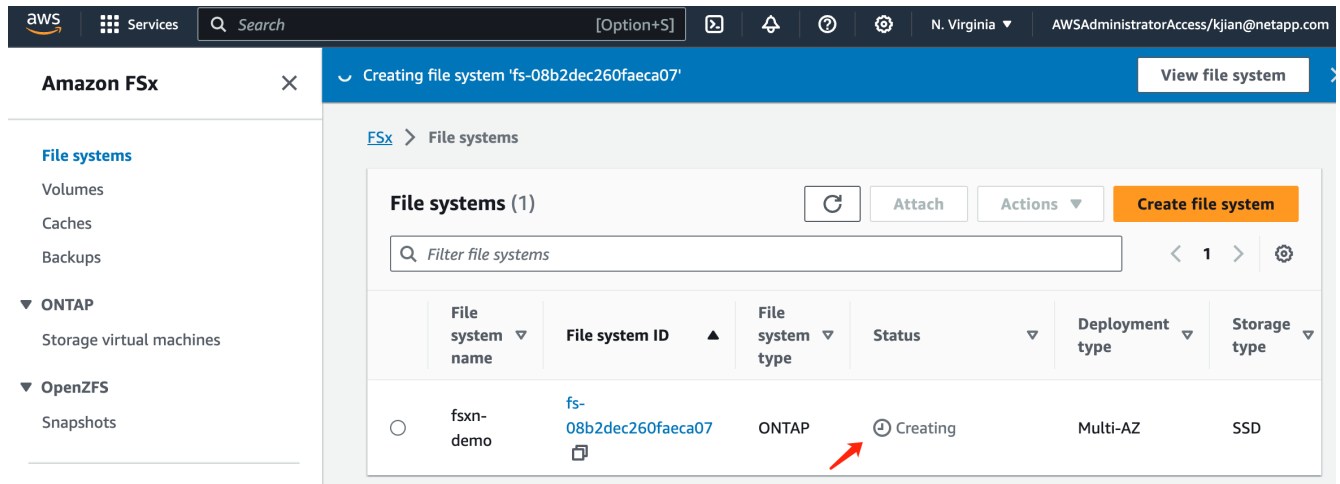
► **Tags - optional**

Cancel **Back** **Next**

f. Fare clic sul pulsante arancione **Crea file system** in basso a destra nella pagina di revisione.



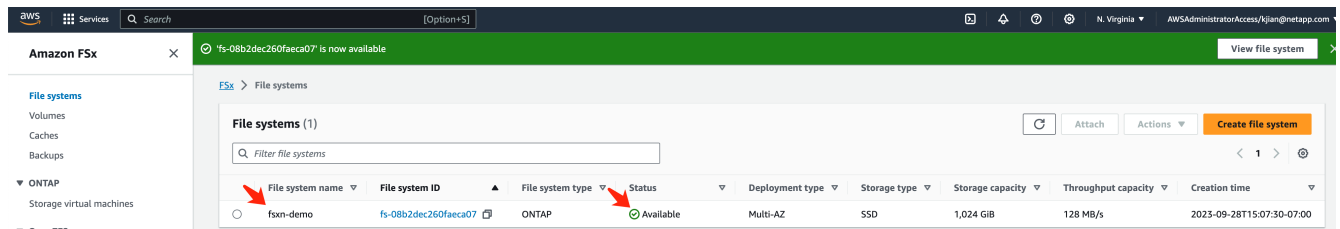
5. L'accelerazione del file system FSX può richiedere circa **20-40 minuti**.



Server Configuration (Configurazione server)

Configurazione ONTAP

1. Aprire il file system FSX creato. Assicurarsi che lo stato sia **disponibile**.



2. Selezionare la scheda **Amministrazione** e mantenere **endpoint di gestione - indirizzo IP e nome utente amministratore ONTAP**.

The screenshot shows the AWS Management Console for an Amazon FSx ONTAP file system named 'fsxn-demo (fs-08b2dec260faeca07)'. The 'Administration' tab is active, showing the following details:

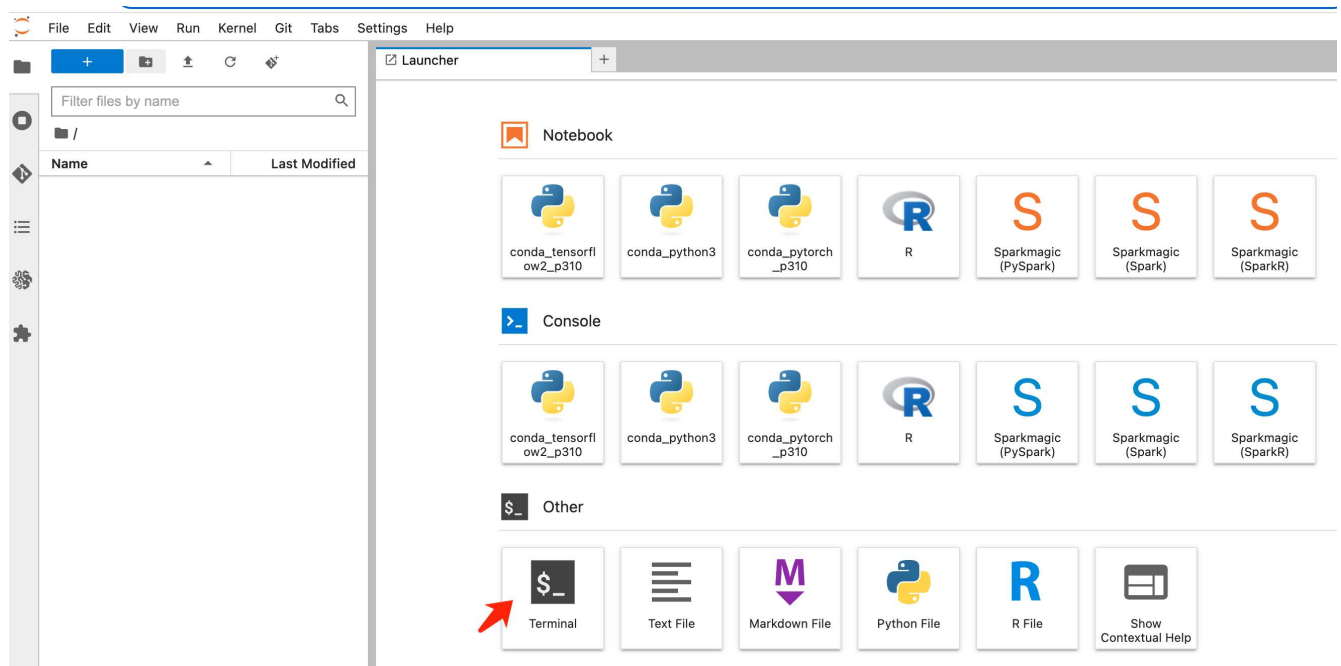
- Summary:**
 - File system ID: fs-08b2dec260faeca07
 - SSD storage capacity: 1024 GiB
 - Throughput capacity: 128 MB/s
 - Provisioned IOPS: 3072
 - Availability Zones: us-east-1a (Preferred), us-east-1b (Standby)
 - Creation time: 2023-09-28T14:50:07:00
 - Lifecycle state: Creating
 - File system type: ONTAP
 - Deployment type: Multi-AZ
- ONTAP administration:**
 - Management endpoint - DNS name: management.fs-08b2dec260faeca07.fsx.us-east-1.amazonaws.com
 - Management endpoint - IP address: 172.31.255.250
 - Inter-cluster endpoint - DNS name: intercluster.fs-08b2dec260faeca07.fsx.us-east-1.amazonaws.com
 - Inter-cluster endpoint - IP address: 172.31.32.38
 - ONTAP administrator username: fsxadmin
 - ONTAP administrator password: [Update]

3. Aprire l'istanza creata **SageMaker notebook** e fare clic su **Apri JupyterLab**.

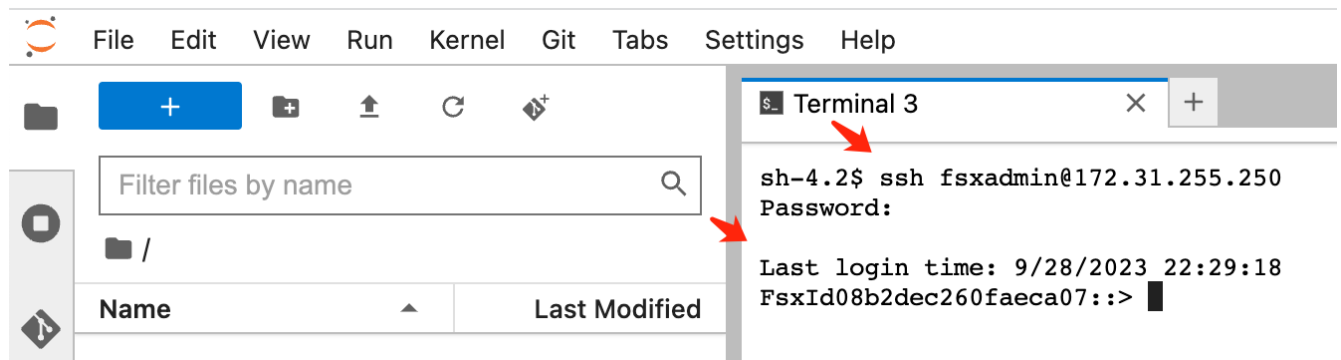
The screenshot shows the Amazon SageMaker console with the 'Notebook instances' page. A table lists the notebook instances:

| 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. Nella pagina Jupyter Lab, aprire un nuovo **terminale**.



5. Inserisci il comando `ssh ssh <nome utente admin>@<IP server ONTAP>` per accedere al file system FSX ONTAP. (Il nome utente e l'indirizzo IP vengono recuperati dal passaggio 2) utilizzare la password utilizzata per creare la **Storage Virtual Machine**.



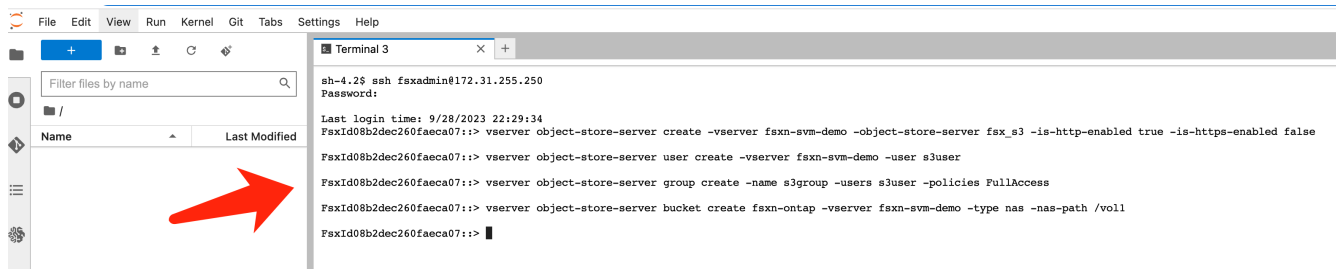
6. Eseguire i comandi nel seguente ordine. Usiamo **fsxn-ONTAP** come nome per il **FSX ONTAP private S3 bucket name**. Utilizzare **storage virtual machine name** per l'argomento **-vserver**.

```
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. Eseguire i seguenti comandi per recuperare l'IP dell'endpoint e le credenziali per FSX ONTAP private S3.

```

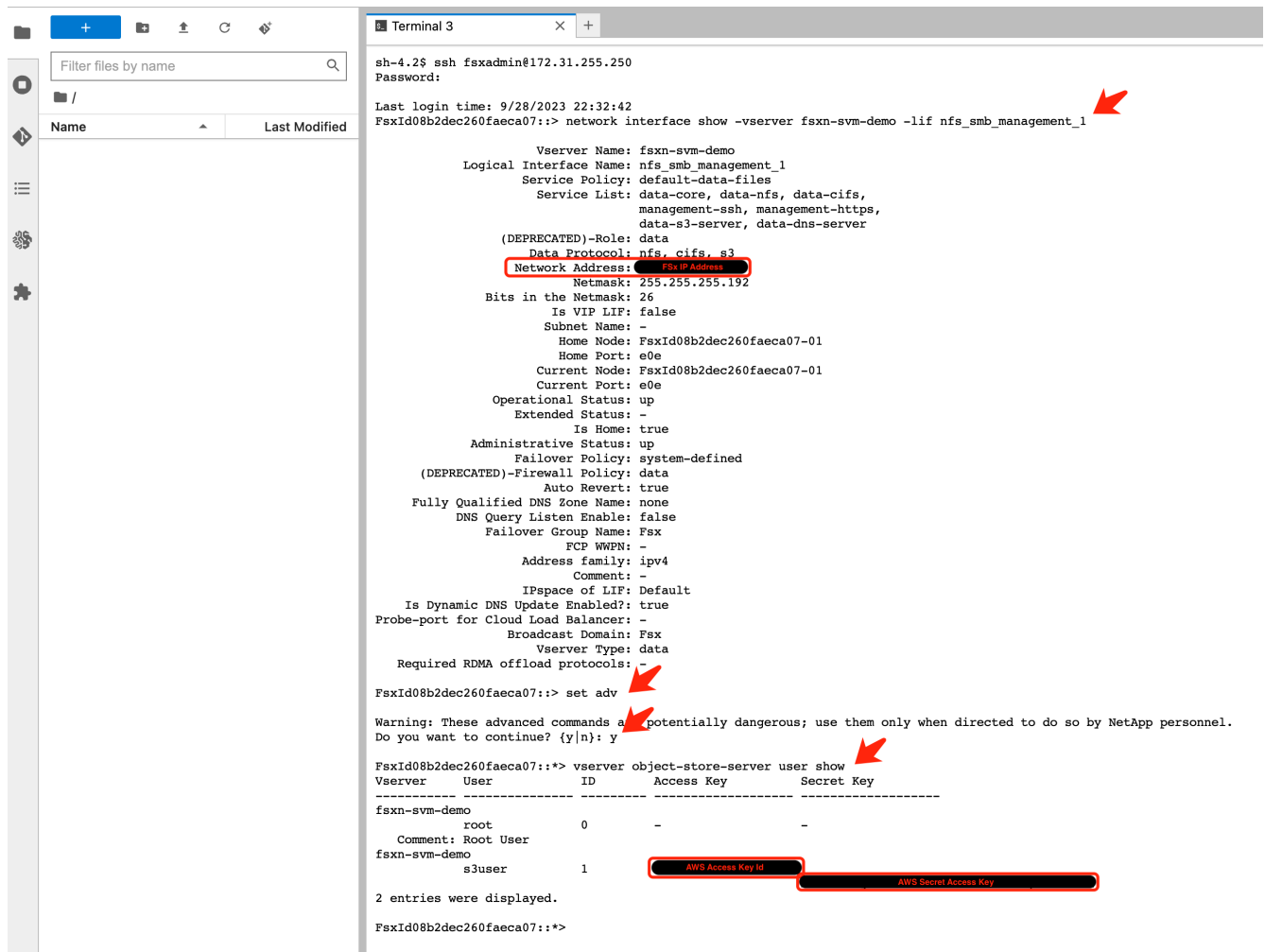
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show

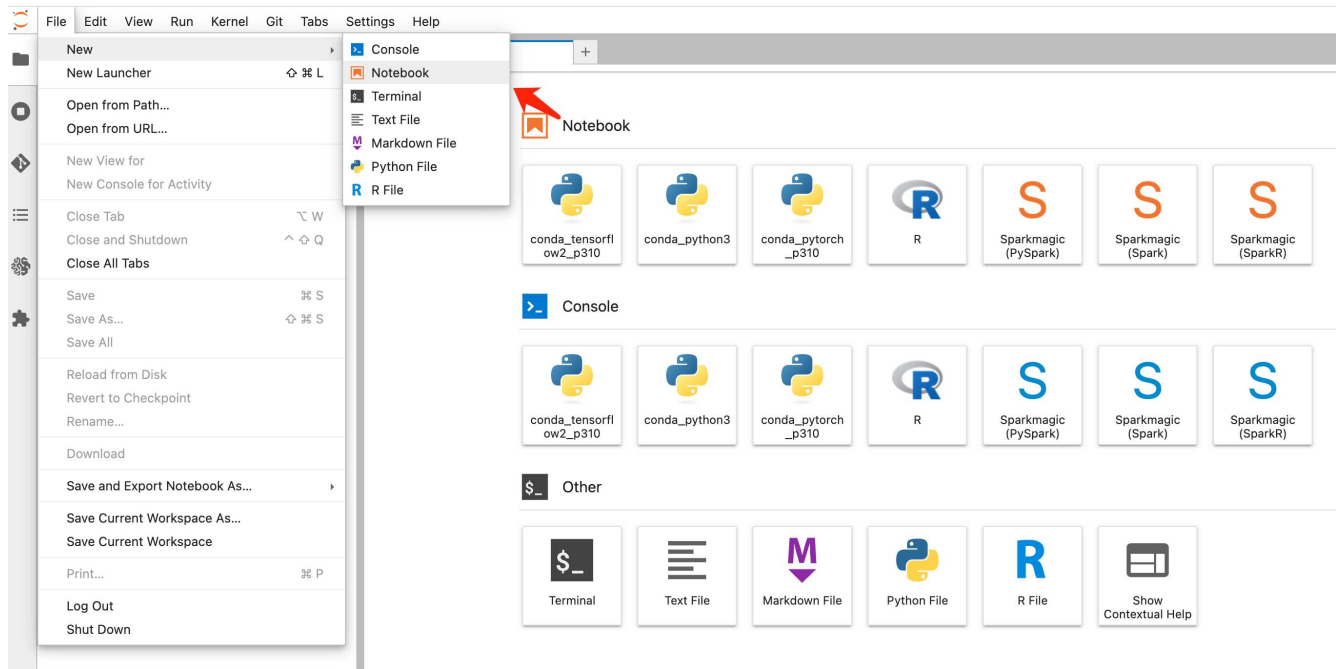
```

8. Conservare l'IP dell'endpoint e le credenziali per un utilizzo futuro.



Client Configuration (Configurazione client)

1. Nell'istanza di notebook SageMaker, creare un nuovo notebook Jupyter.



2. Utilizza il codice riportato di seguito come soluzione per caricare i file nel bucket S3 privato di FSX ONTAP. Per un esempio di codice completo, fare riferimento a questo notebook. ["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 voll
!sudo mount -t nfs $fsx_endpoint_ip:/voll /home/ec2-user/SageMaker/voll
!sudo chmod 777 /home/ec2-user/SageMaker/voll

## 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']

```

Si conclude così l'integrazione tra FSX ONTAP e l'istanza di SageMaker.

Utile elenco di controllo per il debug

- Verificare che l'istanza del notebook SageMaker e il file system FSX ONTAP si trovino nello stesso VPC.
- Ricordarsi di eseguire il comando **set dev** su ONTAP per impostare il livello di privilegio su **dev**.

FAQ (al 27 settembre 2023)

D: Perché viene visualizzato l'errore **"si è verificato un errore (NotImplemented) quando si chiama l'operazione CreateMultipartUpload: Il comando S3 richiesto non è implementato"** quando si caricano i file in FSX ONTAP?

R: Come bucket S3 privato, FSX ONTAP supporta il caricamento di file fino a 100MB KB. Quando si utilizza il protocollo S3, i file di dimensioni superiori a 100MB KB vengono divisi in 100MB blocchi e viene richiamata la funzione "CreateMultipartUpload". Tuttavia, l'attuale implementazione di FSX ONTAP private S3 non supporta questa funzione.

D: Perché viene visualizzato l'errore **"si è verificato un errore (accesso negato) quando si chiamano le operazioni PutObject: Accesso negato"** quando si caricano i file in FSX ONTAP?

R: Per accedere al bucket S3 privato FSX ONTAP da un'istanza di notebook SageMaker, passare le credenziali AWS alle credenziali FSX ONTAP. Tuttavia, la concessione del permesso di scrittura all'istanza richiede una soluzione alternativa che implica il montaggio del bucket e l'esecuzione del comando shell 'chmod' per modificare le autorizzazioni.

D: Come posso integrare il bucket S3 privato di FSX ONTAP con altri servizi ML di SageMaker?

R: Purtroppo, SageMaker Services SDK non fornisce un modo per specificare l'endpoint per il bucket S3 privato. Di conseguenza, FSX ONTAP S3 non è compatibile con i servizi SageMaker come Sagemaker Data Wrangler, Sagemaker Clarify, Sagemaker Glue, Sagemaker Athena, Sagemaker AutoML e altri.

Parte 2 - utilizzo di AWS Amazon FSX per NetApp ONTAP (FSX ONTAP) come origine dati per il training sui modelli in SageMaker

Questo articolo è un tutorial sull'utilizzo di Amazon FSX per NetApp ONTAP (FSX ONTAP) per la formazione dei modelli PyTorch in SageMaker, in particolare per un progetto di classificazione della qualità degli pneumatici.

Autore(i):

Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

Introduzione

Questo tutorial offre un esempio pratico di un progetto di classificazione della computer vision, che fornisce un'esperienza pratica nella creazione di modelli ML che utilizzano FSX ONTAP come origine dati all'interno dell'ambiente SageMaker. Il progetto si concentra sull'utilizzo di PyTorch, un framework di apprendimento approfondito, per classificare la qualità degli pneumatici in base alle immagini degli pneumatici. Enfatizza lo sviluppo di modelli di machine learning utilizzando FSX ONTAP come origine dati in Amazon SageMaker.

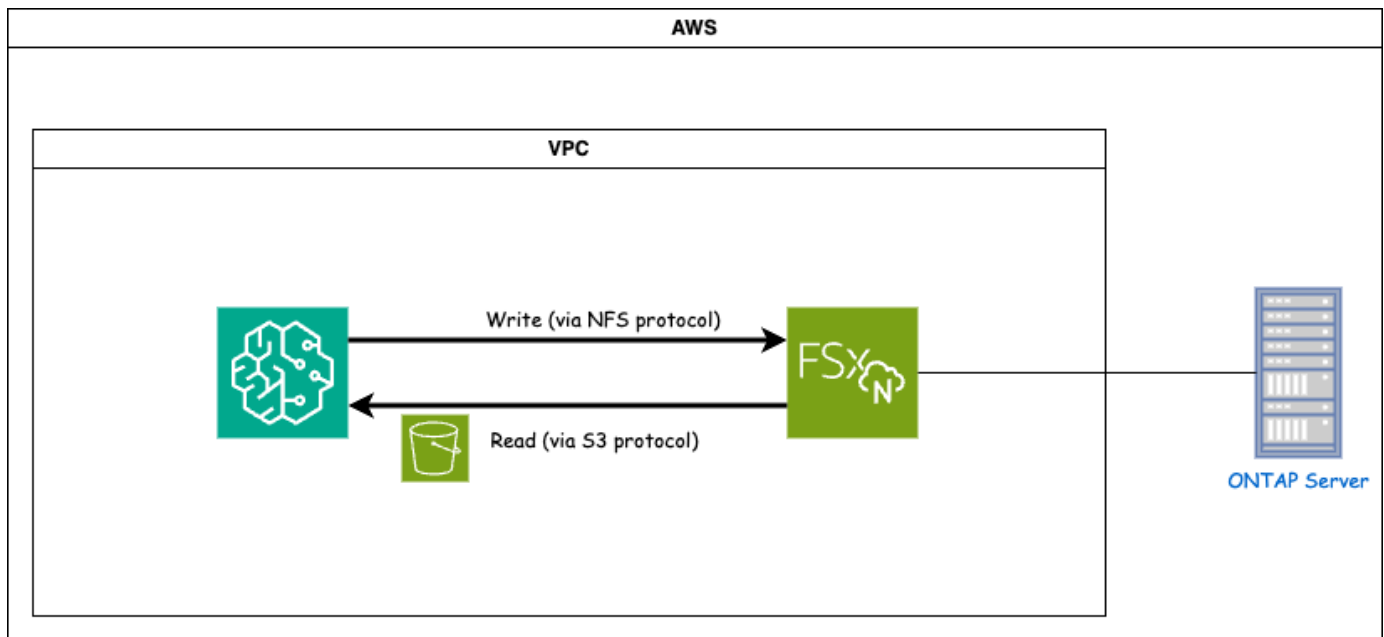
Che cos'è FSX ONTAP

Amazon FSX ONTAP è in realtà una soluzione storage completamente gestita offerta da AWS. Sfrutta il file system ONTAP di NetApp per fornire storage affidabile e dalle performance elevate. Grazie al supporto per protocolli come NFS, SMB e iSCSI, permette l'accesso perfetto da diversi container e istanze di calcolo. Il servizio è progettato per offrire performance eccezionali, garantendo operazioni sui dati rapide ed efficienti. Inoltre, offre high Availability e durata elevata per garantire che i tuoi dati rimangano accessibili e protetti. Inoltre, la capacità storage di Amazon FSX ONTAP è scalabile e ti permette di regolarla facilmente in base alle

tue esigenze.

Prerequisito

Ambiente di rete



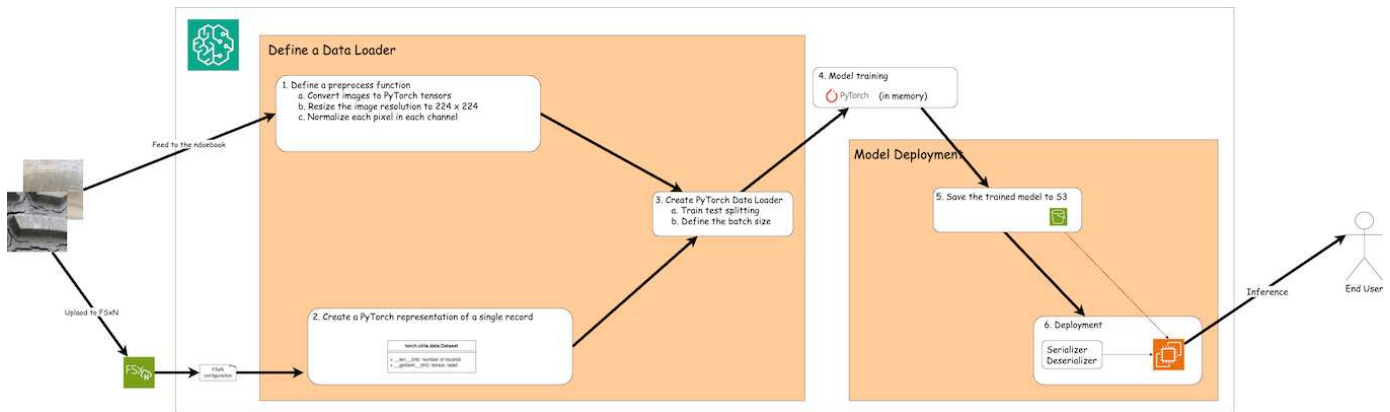
FSX ONTAP (Amazon FSX ONTAP) è un servizio di storage AWS. Include un file system in esecuzione sul sistema NetApp ONTAP e una SVM (System Virtual Machine) gestita da AWS che si connette all'IT. Nel diagramma fornito, il server NetApp ONTAP gestito da AWS si trova all'esterno del VPC. La SVM funge da intermediario tra SageMaker e il sistema NetApp ONTAP, ricevendo le richieste operative da SageMaker e inoltrandole allo storage sottostante. Per accedere a FSX ONTAP, SageMaker deve essere posizionato nello stesso VPC della distribuzione di FSX ONTAP. Questa configurazione garantisce la comunicazione e l'accesso ai dati tra SageMaker e FSX ONTAP.

Accesso ai dati

Negli scenari reali, i data scientist utilizzano in genere i dati esistenti memorizzati in FSX ONTAP per creare i propri modelli di machine learning. Tuttavia, per scopi dimostrativi, poiché il file system FSX ONTAP è inizialmente vuoto dopo la creazione, è necessario caricare manualmente i dati di formazione. Questo può essere ottenuto montando FSX ONTAP come volume su SageMaker. Una volta montato correttamente il file system, è possibile caricare il set di dati nella posizione montata, rendendolo accessibile per l'addestramento dei modelli all'interno dell'ambiente SageMaker. Questo approccio ti consente di sfruttare la capacità di storage e le funzionalità di FSX ONTAP, lavorando contemporaneamente con SageMaker per lo sviluppo e il training dei modelli.

Il processo di lettura dei dati implica la configurazione di FSX ONTAP come bucket S3 privato. Per istruzioni dettagliate sulla configurazione, fare riferimento alla ["Parte 1 - integrazione di Amazon FSX per NetApp ONTAP \(FSX ONTAP\)\) come bucket S3 privato in AWS SageMaker"](#)

Panoramica sull'integrazione



Il flusso di lavoro dell'utilizzo dei dati di formazione in FSX ONTAP per creare un modello di apprendimento approfondito in SageMaker può essere riassunto in tre fasi principali: Definizione di Data Loader, formazione dei modelli e distribuzione. Ad alto livello, questi passaggi costituiscono la base di una pipeline MLOps. Tuttavia, ogni fase prevede diverse fasi secondarie dettagliate per un'implementazione completa. Queste fasi secondarie comprendono varie attività come la pre-elaborazione dei dati, la suddivisione del dataset, la configurazione del modello, la regolazione dell'iperparametro, la valutazione del modello, e distribuzione dei modelli. Questi passaggi garantiscono un processo completo ed efficace per la creazione e la distribuzione di modelli di apprendimento approfondito utilizzando i dati di formazione di FSX ONTAP all'interno dell'ambiente SageMaker.

Integrazione step-by-step

Caricatore dati

Per addestrare una rete di apprendimento profondo PyTorch con i dati, viene creato un caricatore dati per facilitare l'alimentazione dei dati. Il caricatore dati non solo definisce la dimensione del batch, ma determina anche la procedura di lettura e pre-elaborazione di ciascun record all'interno del batch. Configurando il data loader, possiamo gestire l'elaborazione dei dati in batch, consentendo la formazione della rete di deep learning.

Il caricatore dati è composto da 3 parti.

Funzione di pre-elaborazione

```

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]
    )
])

```

Il frammento di codice riportato sopra illustra la definizione delle trasformazioni di pre-elaborazione delle immagini utilizzando il modulo **torchvision.Transforms**. In questa tutorial, l'oggetto di pre-elaborazione viene creato per applicare una serie di trasformazioni. In primo luogo, la trasformazione **ToTensor()** converte

l'immagine in una rappresentazione tensoriale. Successivamente, la trasformazione **Ridimensiona((224.224))** ridimensiona l'immagine a una dimensione fissa di 224x224 pixel. Infine, la trasformazione **Normalize()** normalizza i valori del tensore sottraendo la media e dividendo per la deviazione standard lungo ciascun canale. I valori di deviazione media e standard utilizzati per la normalizzazione sono comunemente impiegati in modelli di rete neurale pre-addestrati. Nel complesso, questo codice prepara i dati dell'immagine per un'ulteriore elaborazione o immissione in un modello pre-addestrato convertendoli in un tensore, ridimensionandoli e normalizzando i valori dei pixel.

Classe dataset PyTorch

```
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
```

Questa classe fornisce funzionalità per ottenere il numero totale di record nell'insieme di dati e definisce il metodo di lettura dei dati per ogni record. All'interno della funzione **getitem**, il codice utilizza l'oggetto bucket boto3 S3 per recuperare i dati binari da FSX ONTAP. Lo stile del codice per l'accesso ai dati da FSX ONTAP è simile alla lettura dei dati da Amazon S3. La spiegazione successiva si sofferma sul processo di creazione dell'oggetto S3 privato **bucket**.

FSX ONTAP come repository S3 privato

```
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 ---
```

Per leggere i dati da FSX ONTAP in SageMaker, viene creato un gestore che punta allo storage FSX ONTAP utilizzando il protocollo S3. In questo modo FSX ONTAP può essere trattato come un bucket S3 privato. La configurazione del gestore include l'indicazione dell'indirizzo IP della SVM di FSX ONTAP, del nome del bucket e delle credenziali necessarie. Per una spiegazione completa su come ottenere questi elementi di configurazione, fare riferimento al documento all'indirizzo ["Parte 1 - integrazione di Amazon FSX per NetApp ONTAP \(FSX ONTAP\) come bucket S3 privato in AWS SageMaker"](#).

Nell'esempio sopra menzionato, l'oggetto bucket viene utilizzato per creare un'istanza dell'oggetto dataset PyTorch. L'oggetto dataset verrà ulteriormente spiegato nella sezione successiva.

Il caricatore dati PyTorch

```
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)
```

Nell'esempio fornito, viene specificata una dimensione batch di 64, che indica che ogni batch conterrà 64 record. Combinando la classe PyTorch **dataset**, la funzione di pre-elaborazione e la dimensione del batch di training, otteniamo il caricatore dati per la formazione. Questo caricatore dati facilita il processo di iterazione del set di dati in batch durante la fase di training.

Training sui modelli

```
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')

```

Questo codice implementa un processo di formazione PyTorch standard. Definisce un modello di rete neurale chiamato **TyreQualityClassifier** utilizzando strati convoluzionali e uno strato lineare per classificare la qualità dei pneumatici. Il ciclo di training itera i batch di dati, calcola la perdita e aggiorna i parametri del modello utilizzando la backpropagation e l'ottimizzazione. Inoltre, stampa l'ora corrente, l'epoca, il batch e la perdita a scopo di monitoraggio.

Implementazione dei modelli

Implementazione

```

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())

```

Il codice salva il modello PyTorch in **Amazon S3** perché SageMaker richiede che il modello venga memorizzato in S3 per la distribuzione. Caricando il modello su **Amazon S3**, diventa accessibile a SageMaker, consentendo la distribuzione e l'inferenza sul modello distribuito.

```

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
)

```

Questo codice facilita la distribuzione di un modello PyTorch su SageMaker. Definisce un serializzatore personalizzato, **TyreQualitySerializer**, che preelabora e serializza i dati di input come un tensor PyTorch. La classe **TyreQualityPredictor** è un predittore personalizzato che utilizza il serializzatore definito e un **JSONDeserializer**. Il codice crea inoltre un oggetto **PyTorchModel** per specificare la posizione S3 del modello, il ruolo IAM, la versione del framework e il punto di ingresso per l'inferenza. Il codice genera un

indicatore data e ora e costruisce un nome endpoint in base al modello e all'indicatore data e ora. Infine, il modello viene distribuito utilizzando il metodo Deploy, specificando il numero di istanze, il tipo di istanza e il nome dell'endpoint generato. In questo modo, il modello PyTorch può essere distribuito e accessibile per l'inferenza su SageMaker.

Inferenza

```
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)
```

Questo è l'esempio di utilizzo dell'endpoint distribuito per l'inferenza.

Parte 3 - creazione di Una pipeline MLOps semplificata (ci/CT/CD)

Questo articolo fornisce una guida alla creazione di una pipeline MLOps con servizi AWS, concentrandosi su ritraining, implementazione e ottimizzazione dei costi automatizzati dei modelli.

Autore(i):

Jian Jian (Ken), Senior Data & Applied Scientist, NetApp

Introduzione

In questo tutorial scoprirai come sfruttare i vari servizi AWS per costruire una semplice pipeline MLOps che comprende Continuous Integration (ci), Continuous Training (CT) e Continuous Deployment (CD). A differenza delle tradizionali pipeline DevOps, gli MLOps richiedono ulteriori considerazioni per completare il ciclo operativo. Seguendo questo tutorial, si acquisiranno informazioni sull'integrazione della TC nel loop MLOps, consentendo una formazione continua dei modelli e una distribuzione perfetta per l'inferenza. Il tutorial ti guiderà attraverso il processo di utilizzo dei servizi AWS per creare questa pipeline MLOps end-to-end.

Manifesto

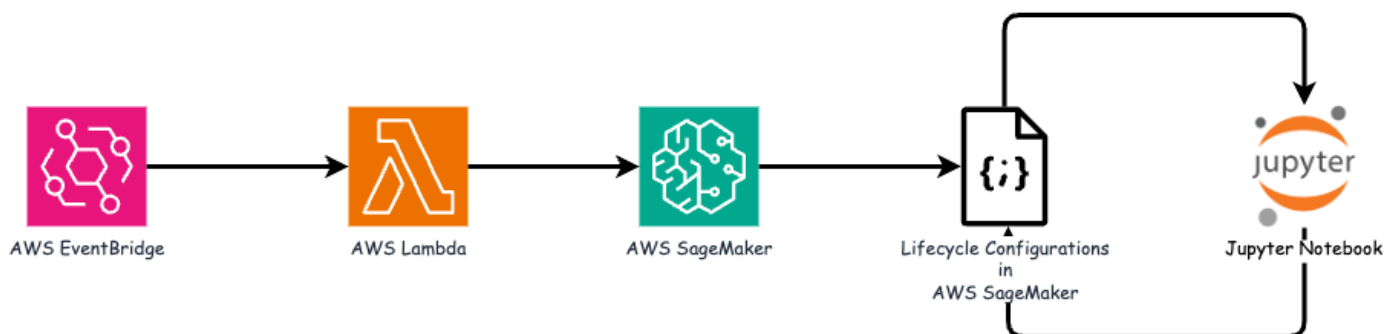
| Funzionalità | Nome | Commento |
|------------------|---------------|---|
| Storage dei dati | ONTAP AWS FSX | Fare riferimento alla "Parte 1 - integrazione di Amazon FSX per NetApp ONTAP (FSX ONTAP) come bucket S3 privato in AWS SageMaker" . |

| Funzionalità | Nome | Commento |
|---|---------------------|---|
| IDE di data science | AWS SageMaker | Questo tutorial si basa sul notebook Jupyter presentato in "Parte 2 - utilizzo di Amazon FSX per NetApp ONTAP (FSX ONTAP) come origine dati per il training sui modelli in SageMaker" . |
| Funzione per attivare la pipeline MLOps | Funzione AWS Lambda | - |
| Trigger di job cron | AWS EventBridge | - |
| Framework di deep learning | PyTorch | - |
| SDK AWS Python | boto3 | - |
| Linguaggio di programmazione | Python | v3,10 |

Prerequisito

- Un file system FSX ONTAP preconfigurato. Questa esercitazione utilizza i dati memorizzati in FSX ONTAP per il processo di formazione.
- Un'istanza **SageMaker notebook** configurata per condividere lo stesso VPC del file system FSX ONTAP menzionato sopra.
- Prima di attivare la funzione **AWS Lambda**, assicurarsi che l'istanza **SageMaker notebook** sia nello stato **Stopped**.
- Il tipo di istanza **ml.g4dn.xlarge** è necessario per sfruttare l'accelerazione GPU necessaria per i calcoli delle reti neurali profonde.

Architettura



Questa pipeline MLOps è un'implementazione pratica che utilizza un job cron per attivare una funzione senza server, che a sua volta esegue un servizio AWS registrato con una funzione di callback del ciclo di vita. Il **AWS EventBridge** agisce come job cron. Richiama periodicamente una funzione **AWS Lambda** responsabile del riaddestramento e della redistribuzione del modello. Questo processo comporta la creazione dell'istanza **AWS SageMaker notebook** per eseguire le attività necessarie.

Configurazione dettagliata

Configurazioni del ciclo di vita

Per configurare la funzione di callback del ciclo di vita per l'istanza del notebook AWS SageMaker, utilizzare **configurazioni del ciclo di vita**. Questo servizio consente di definire le azioni necessarie da eseguire durante la rotazione dell'istanza del notebook. In particolare, è possibile implementare uno script della shell all'interno delle **configurazioni del ciclo di vita** per arrestare automaticamente l'istanza del notebook una volta completati i processi di formazione e distribuzione. Si tratta di una configurazione richiesta, in quanto il costo è una delle considerazioni principali di MLOps.

È importante notare che la configurazione per **configurazioni del ciclo di vita** deve essere impostata in anticipo. Pertanto, si consiglia di assegnare una priorità alla configurazione di questo aspetto prima di procedere con l'altra impostazione della pipeline MLOps.

1. Per impostare una configurazione del ciclo di vita, aprire il pannello **Sagemaker** e passare a **configurazioni del ciclo di vita** nella sezione **configurazioni amministratore**.

aws Services 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 S3 bucket.

► **Domain structure diagram**

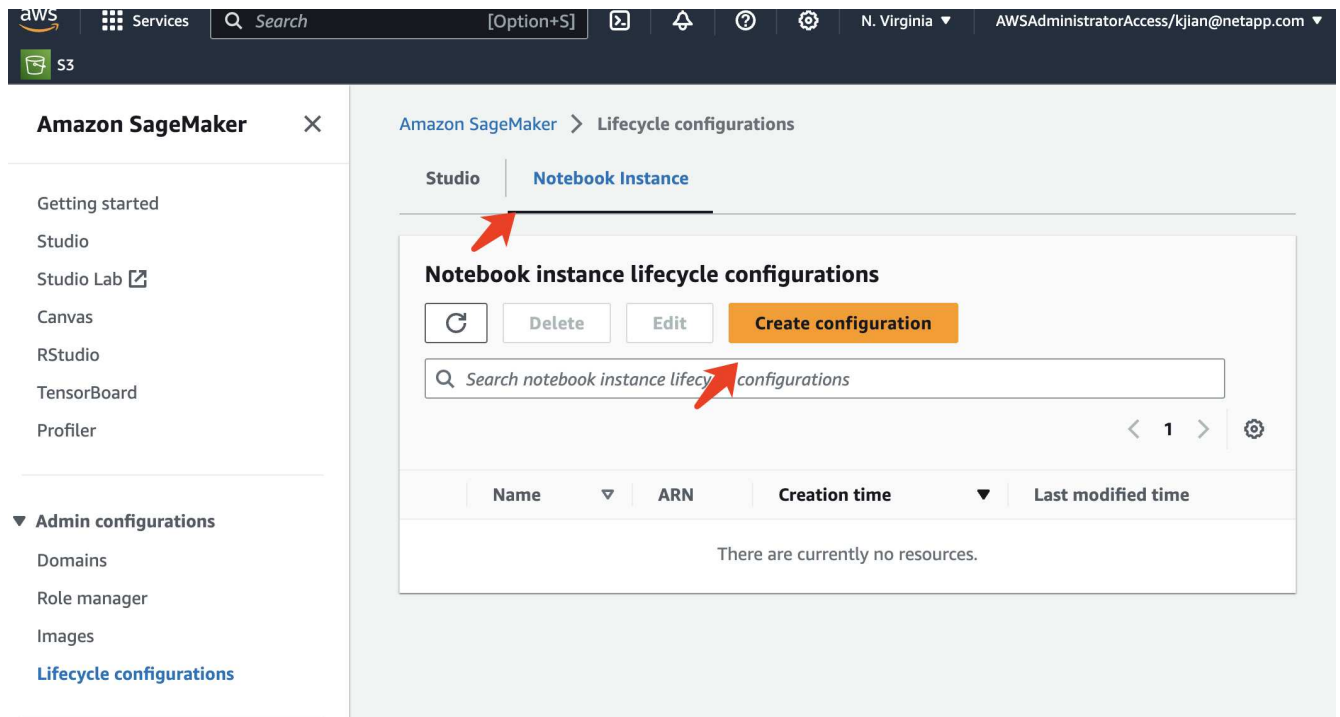
Domains (4) [Info](#)

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. Selezionare la scheda **istanza notebook** e fare clic sul pulsante **Crea configurazione**



3. Incollare il codice riportato di seguito nell'area di immissione.

```
#!/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 "*/* * * * 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. Questo script esegue il notebook Jupyter, che gestisce il riaddestramento e la redistribuzione del modello per l'inferenza. Al termine dell'esecuzione, il notebook si spegne automaticamente entro 5 minuti. Per ulteriori informazioni sull'istruzione Problem e sull'implementazione del codice, fare riferimento a ["Parte 2 - utilizzo di Amazon FSX per NetApp ONTAP \(FSX ONTAP\) come origine dati per il training sui modelli in SageMaker"](#).

Configuration setting

Name

fsxn-demo-lifecycle-callback

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

Scripts

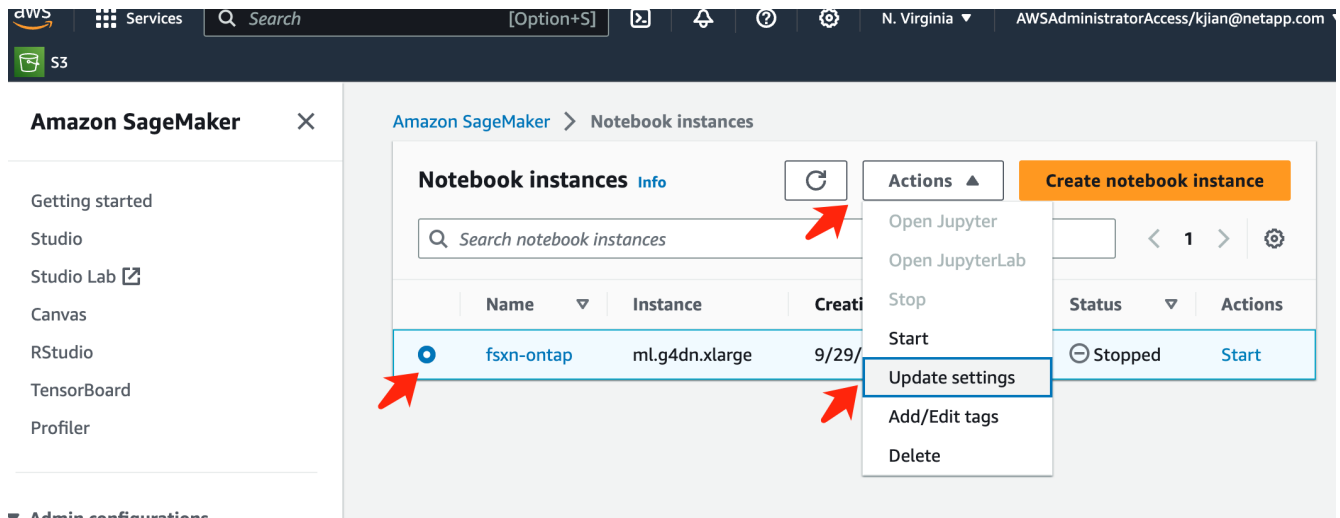
Start notebook | Create notebook

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](#)

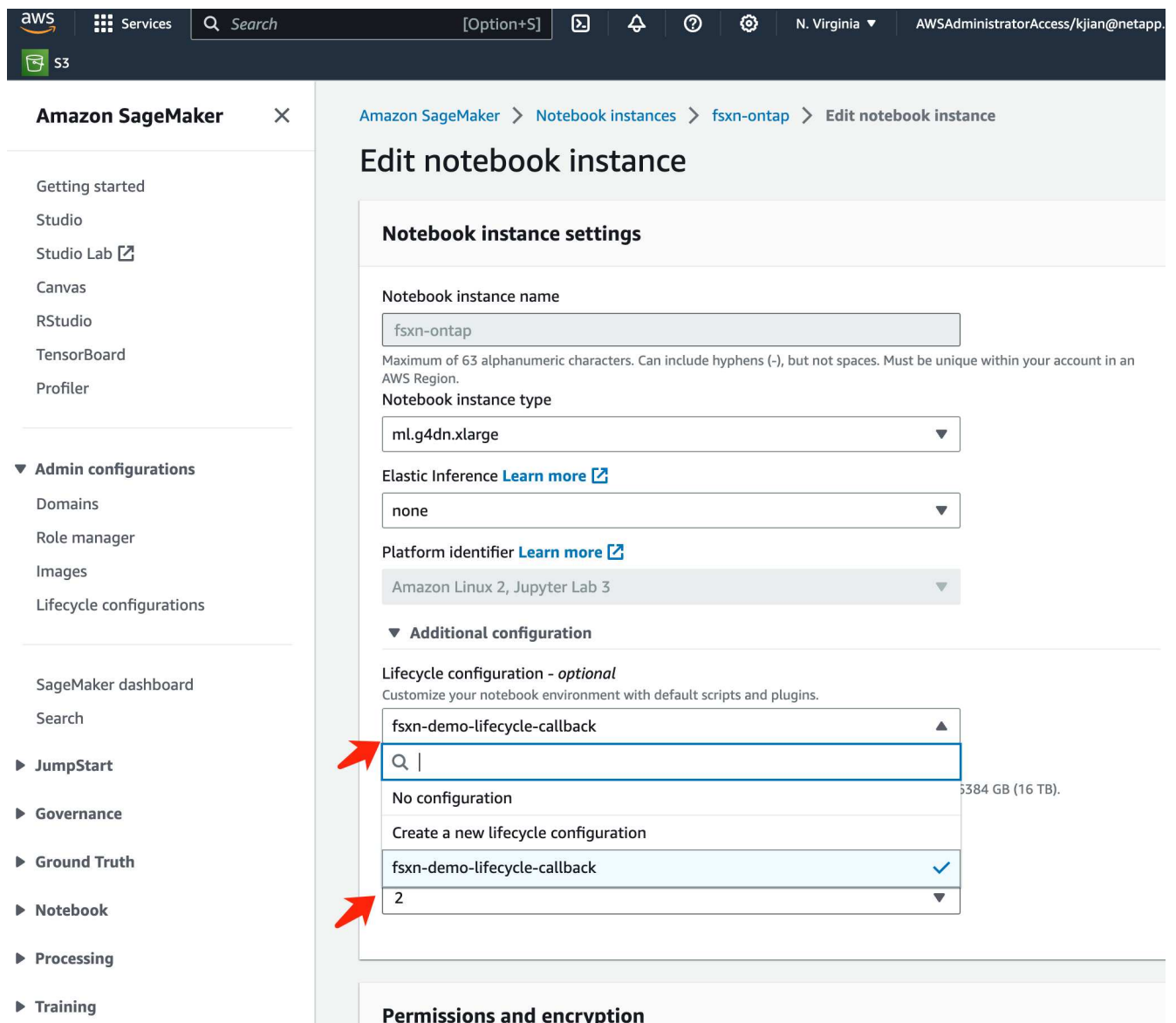
```
1 #!/bin/bash
2
3 set -e
4 sudo -u ec2-user -i <<'EOF'
5 # 1. Retraining and redeploying the model
6 NOTEBOOK_FILE=/home/ec2-user/SageMaker/tyre_quality_classification_local_training.ipynb
7 echo "Activating conda env"
8 source /home/ec2-user/anaconda3/bin/activate pytorch_p310
9 nohup jupyter nbconvert "$NOTEBOOK_FILE" --ExecutePreprocessor.kernel_name=python --execute --to n
10 nbconvert_pid=$!
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. Dopo la creazione, accedere a istanze notebook, selezionare l'istanza di destinazione e fare clic su **Aggiorna impostazioni** nel menu a discesa azioni.



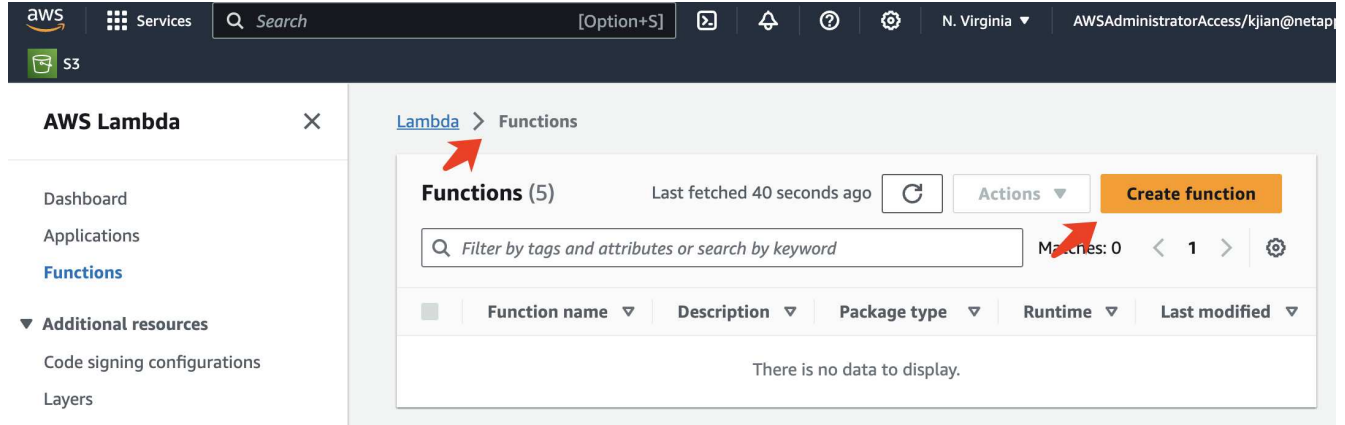
6. Selezionare la configurazione **Lifecycle** creata e fare clic su **Aggiorna istanza notebook**.



Funzione serverless di AWS Lambda

Come accennato in precedenza, la funzione **AWS Lambda** è responsabile della creazione dell'istanza **AWS SageMaker notebook**.

1. Per creare una funzione **AWS Lambda**, accedere al pannello corrispondente, passare alla scheda **funzioni** e fare clic su **Crea funzione**.



2. Si prega di archiviare tutte le voci necessarie nella pagina e ricordarsi di cambiare il Runtime a **Python 3,10**.

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.

3. Verificare che il ruolo designato disponga dell'autorizzazione richiesta **AmazonSageMakerFullAccess** e fare clic sul pulsante **Crea funzione**.

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-585jzdney
[View the fsxn-demo-mlops-role-585jzdney role](#) on the IAM console.

► **Advanced settings**

4. Selezionare la funzione Lambda creata. Nella scheda Codice, copiare e incollare il seguente codice nell'area di testo. Questo codice avvia l'istanza del notebook denominata **fsxn-ontap**.

```
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. Fare clic sul pulsante **Deploy** per applicare questa modifica di codice.

The screenshot shows the AWS Lambda console interface. At the top, the navigation bar includes the AWS logo, 'Services', a search bar, and the user's name 'N. Virgin'. The main content area displays the configuration for a Lambda function named 'demo-mlops'. On the left, there are buttons for '+ Add trigger' and '+ Add destination'. On the right, the function's details are shown: 'Last modified 1 minute ago', 'Function ARN: arn:aws:lambda:us-east-1:232233133319:function:fsxn-demo-mlops', and 'Function URL: Info'. Below the configuration, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code source' tab is selected, showing a code editor with the following Python code:

```
1 import boto3
2 import logging
3
4 def lambda_handler(event, context):
5     client = boto3.client('sagemaker')
6     logging.info('Invoking SageMaker')
7     client.start_notebook_instance(NotebookInstanceName='fsxn-ontap')
8     return {
9         'statusCode': 200,
10        'body': f'Starting notebook instance: {notebook_instance_name}'
11    }
12
```

A red arrow points to the 'Deploy' button in the code editor's toolbar. The toolbar also includes 'Test', 'Changes not deployed', and a settings icon. The code editor's environment pane on the left shows the file 'lambda_function.py'.

6. Per specificare come attivare questa funzione AWS Lambda, fare clic sul pulsante Add Trigger (Aggiungi trigger).

The screenshot shows the AWS Lambda console interface for a function named 'fsxn-demo-mlops'. At the top, there is a navigation bar with the AWS logo, 'Services', a search bar, and the user's session information. Below the navigation bar, the breadcrumb trail reads 'Lambda > Functions > fsxn-demo-mlops'. The main heading is 'fsxn-demo-mlops', followed by three buttons: 'Throttle', 'Copy ARN', and 'Actions'. The 'Function overview' section is expanded, showing a card for the function with the AWS Lambda icon, the name 'fsxn-demo-mlops', and a 'Layers (0)' section. Below the card are two buttons: '+ Add trigger' and '+ Add destination'. A red arrow points to the '+ Add trigger' button. To the right of the card, there is a metadata panel with the following information: 'Description' (empty), 'Last modified' (2 minutes ago), 'Function ARN' (arn:aws:lambda:us-east-1:232233133319:function:fsxn-demo-mlops), and 'Function URL' (empty).

7. Selezionare EventBridge dal menu a discesa, quindi fare clic sul pulsante di opzione Crea una nuova regola. Nel campo espressione pianificazione, immettere `rate(1 day)`, Quindi fare clic sul pulsante Aggiungi per creare e applicare questa nuova regola del job cron alla funzione AWS Lambda.

The screenshot shows the AWS Lambda console interface for adding a trigger. The breadcrumb navigation is 'Lambda > Add trigger'. The main heading is 'Add trigger'. Under 'Trigger configuration', the selected provider is 'EventBridge (CloudWatch Events)'. The 'Rule' section has 'Create a new rule' selected. The 'Rule name' is 'mlops-retraining-trigger'. The 'Rule type' is 'Schedule expression', and the 'Schedule expression' is 'rate(1 day)'. At the bottom right, there are 'Cancel' and 'Add' buttons.

Dopo aver completato la configurazione in due fasi, su base giornaliera, la funzione **AWS Lambda** avvierà il notebook **SageMaker**, eseguirà il riaddestramento del modello utilizzando i dati del repository **FSX ONTAP**, ridistribuirà il modello aggiornato nell'ambiente di produzione e spegnerà automaticamente l'istanza del notebook **SageMaker** per ottimizzare i costi. In questo modo, il modello rimane aggiornato.

Questo conclude il tutorial per lo sviluppo di una pipeline MLOps.

Informazioni sul copyright

Copyright © 2025 NetApp, Inc. Tutti i diritti riservati. Stampato negli Stati Uniti d'America. Nessuna porzione di questo documento soggetta a copyright può essere riprodotta in qualsiasi formato o mezzo (grafico, elettronico o meccanico, inclusi fotocopie, registrazione, nastri o storage in un sistema elettronico) senza previo consenso scritto da parte del detentore del copyright.

Il software derivato dal materiale sottoposto a copyright di NetApp è soggetto alla seguente licenza e dichiarazione di non responsabilità:

IL PRESENTE SOFTWARE VIENE FORNITO DA NETAPP "COSÌ COM'È" E SENZA QUALSIVOGLIA TIPO DI GARANZIA IMPLICITA O ESPRESSA FRA CUI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, GARANZIE IMPLICITE DI COMMERCIALIZZABILITÀ E IDONEITÀ PER UNO SCOPO SPECIFICO, CHE VENGONO DECLINATE DAL PRESENTE DOCUMENTO. NETAPP NON VERRÀ CONSIDERATA RESPONSABILE IN ALCUN CASO PER QUALSIVOGLIA DANNO DIRETTO, INDIRETTO, ACCIDENTALE, SPECIALE, ESEMPLARE E CONSEGUENZIALE (COMPRESI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, PROCUREMENT O SOSTITUZIONE DI MERCI O SERVIZI, IMPOSSIBILITÀ DI UTILIZZO O PERDITA DI DATI O PROFITTI OPPURE INTERRUZIONE DELL'ATTIVITÀ AZIENDALE) CAUSATO IN QUALSIVOGLIA MODO O IN RELAZIONE A QUALUNQUE TEORIA DI RESPONSABILITÀ, SIA ESSA CONTRATTUALE, RIGOROSA O DOVUTA A INSOLVENZA (COMPRESA LA NEGLIGENZA O ALTRO) INSORTA IN QUALSIASI MODO ATTRAVERSO L'UTILIZZO DEL PRESENTE SOFTWARE ANCHE IN PRESENZA DI UN PREAVVISO CIRCA L'EVENTUALITÀ DI QUESTO TIPO DI DANNI.

NetApp si riserva il diritto di modificare in qualsiasi momento qualunque prodotto descritto nel presente documento senza fornire alcun preavviso. NetApp non si assume alcuna responsabilità circa l'utilizzo dei prodotti o materiali descritti nel presente documento, con l'eccezione di quanto concordato espressamente e per iscritto da NetApp. L'utilizzo o l'acquisto del presente prodotto non comporta il rilascio di una licenza nell'ambito di un qualche diritto di brevetto, marchio commerciale o altro diritto di proprietà intellettuale di NetApp.

Il prodotto descritto in questa guida può essere protetto da uno o più brevetti degli Stati Uniti, esteri o in attesa di approvazione.

LEGENDA PER I DIRITTI SOTTOPOSTI A LIMITAZIONE: l'utilizzo, la duplicazione o la divulgazione da parte degli enti governativi sono soggetti alle limitazioni indicate nel sottoparagrafo (b)(3) della clausola Rights in Technical Data and Computer Software del DFARS 252.227-7013 (FEB 2014) e FAR 52.227-19 (DIC 2007).

I dati contenuti nel presente documento riguardano un articolo commerciale (secondo la definizione data in FAR 2.101) e sono di proprietà di NetApp, Inc. Tutti i dati tecnici e il software NetApp forniti secondo i termini del presente Contratto sono articoli aventi natura commerciale, sviluppati con finanziamenti esclusivamente privati. Il governo statunitense ha una licenza irrevocabile limitata, non esclusiva, non trasferibile, non cedibile, mondiale, per l'utilizzo dei Dati esclusivamente in connessione con e a supporto di un contratto governativo statunitense in base al quale i Dati sono distribuiti. Con la sola esclusione di quanto indicato nel presente documento, i Dati non possono essere utilizzati, divulgati, riprodotti, modificati, visualizzati o mostrati senza la previa approvazione scritta di NetApp, Inc. I diritti di licenza del governo degli Stati Uniti per il Dipartimento della Difesa sono limitati ai diritti identificati nella clausola DFARS 252.227-7015(b) (FEB 2014).

Informazioni sul marchio commerciale

NETAPP, il logo NETAPP e i marchi elencati alla pagina <http://www.netapp.com/TM> sono marchi di NetApp, Inc. Gli altri nomi di aziende e prodotti potrebbero essere marchi dei rispettivi proprietari.