# 数据管道，数据湖和管理

## NetApp Solutions

NetApp
April 12, 2024

# 目录

# 数据管道，数据湖和管理

## 适用于MLOps的AWS FSx for NetApp ONTAP (FSxN)

作者：
Jian Jian (Ken)、NetApp高级数据和应用科学人员

本节将深入介绍AI基础架构开发的实际应用、提供使用FSxN构建MLOps管道的端到端逐步介绍。它包含三个全面的示例、可指导您通过这一强大的数据管理平台满足MLOps需求。

这些文章侧重于：

1. "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"
2. "第2部分—利用AWS FSx for NetApp ONTAP (FSxN)作为SageMaker模型训练的数据源"
3. "第3部分-构建简化的MLOps管道(CI/CT/CD)"

本节结束时、您将深入了解如何使用FSxN简化MLOps流程。

### 第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中

作者：
Jian Jian (Ken)、NetApp高级数据和应用科学人员

简介

本页以SageMaker为例、提供将FSxN配置为私有S3存储分段的指导。

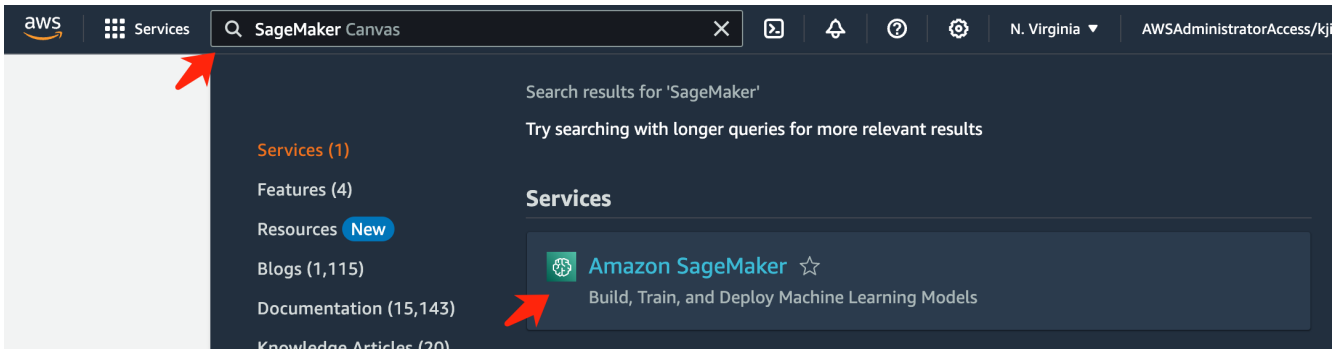
有关FSxN的更多信息、请观看此演示文稿("视频链接")

用户指南

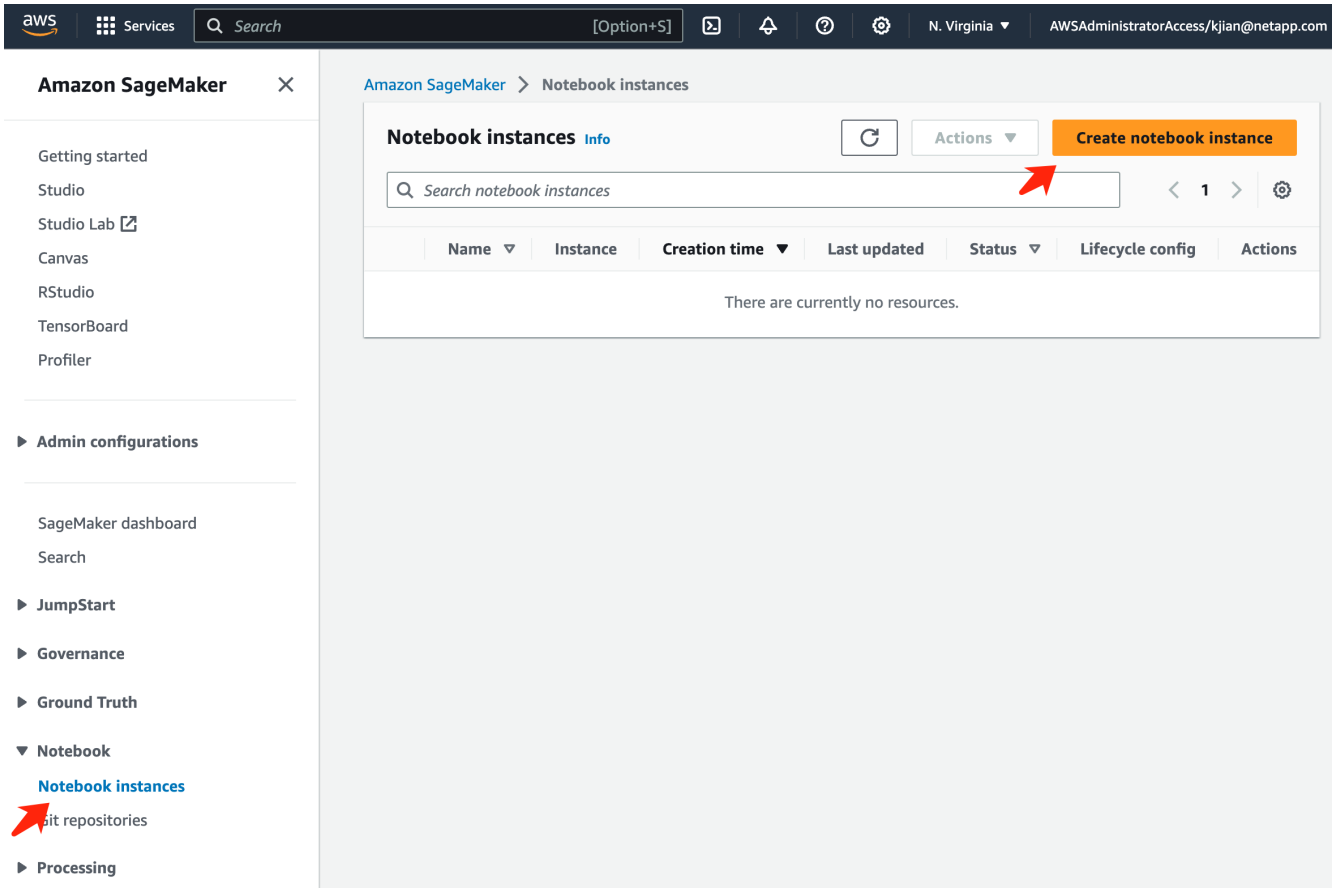
创建服务器

创建**SageMaker**笔记本实例

1. 打开AWS控制台。在搜索面板中、搜索SageMaker并单击服务*亚马逊SageMaker *。



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




3. 在创建页面中、
   输入*笔记本实例名称*
   展开*Network*面板
   保留其它条目默认值，然后选择*VPC*、**Subnet***和***Security group**。(稍后将使用此*VPC*和*Subnet*创建FSxN文件系统)
   单击右下角的橙色按钮*创建笔记本实例*。

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

Security group(s)

sg-0a39b3985770e9256 (default) ✕

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

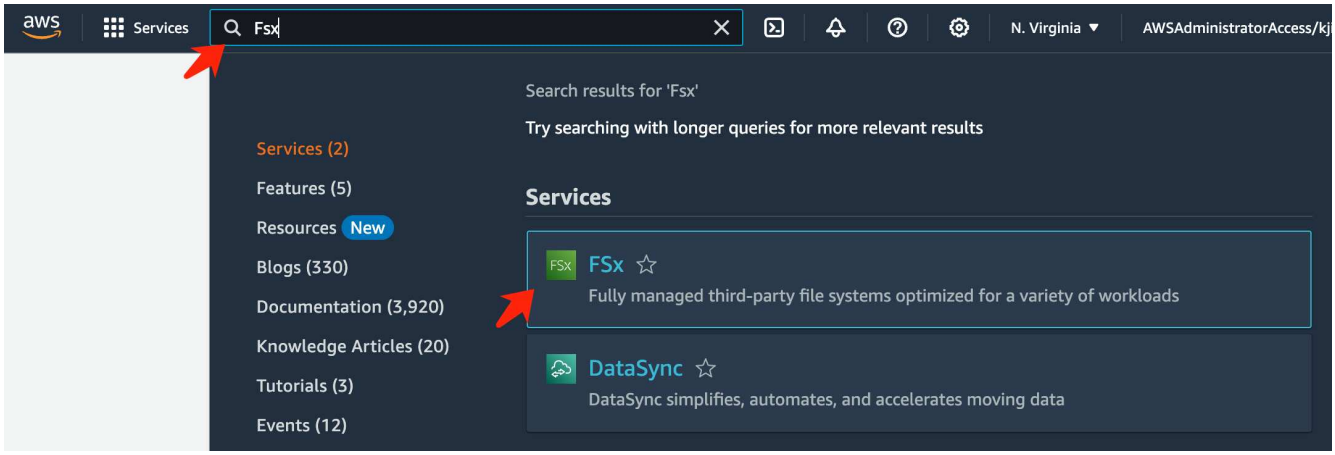
## ▶ Git repositories- *optional*

## ▶ Tags - *optional*

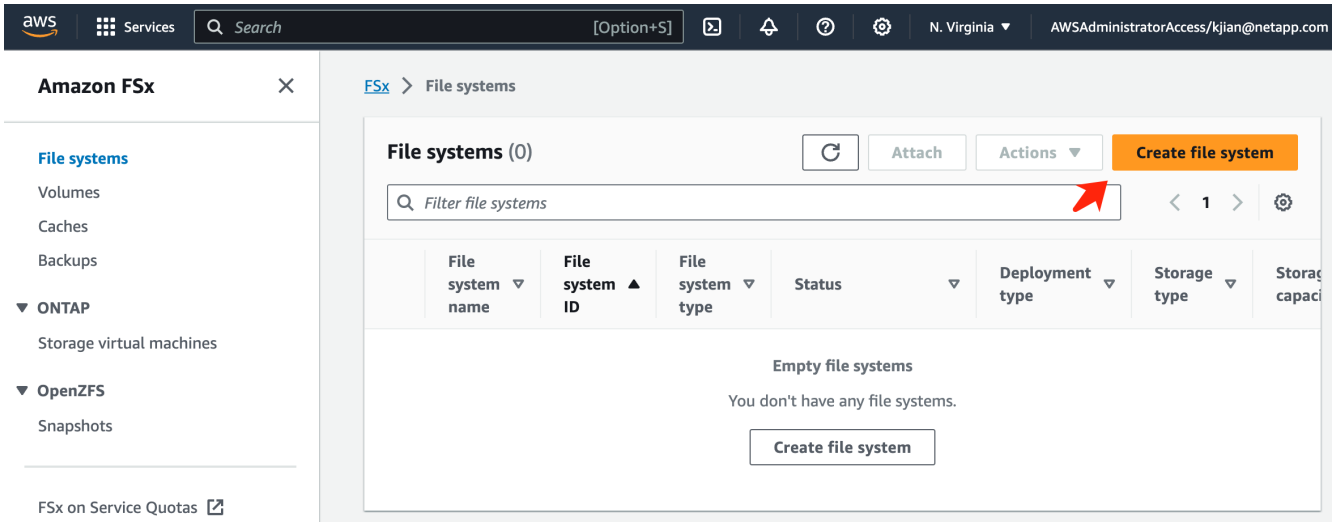
Cancel | Create notebook instance

创建**FSxN**文件系统

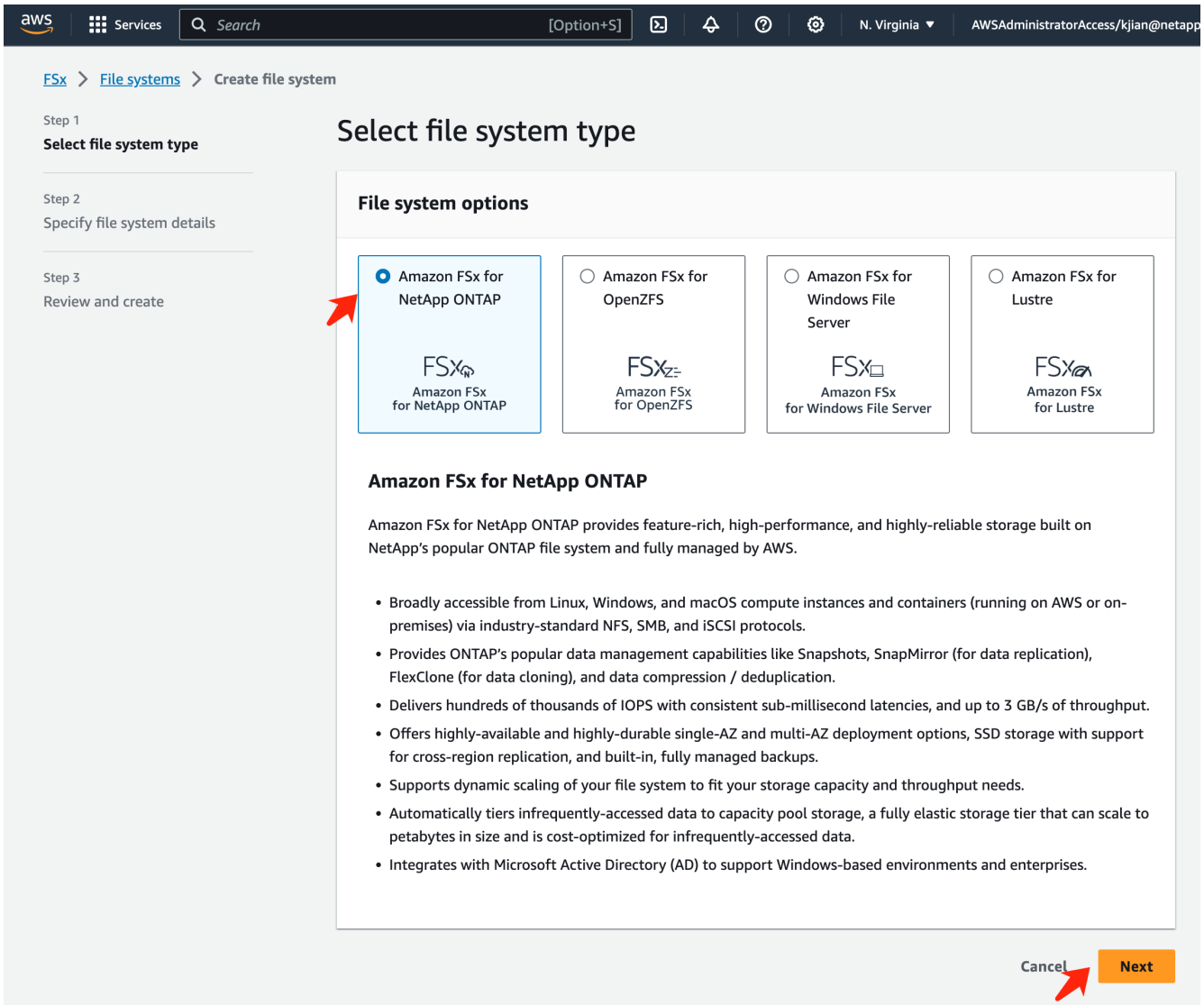
1. 打开AWS控制台。在搜索面板中，搜索FSx并单击服务*FSX*。



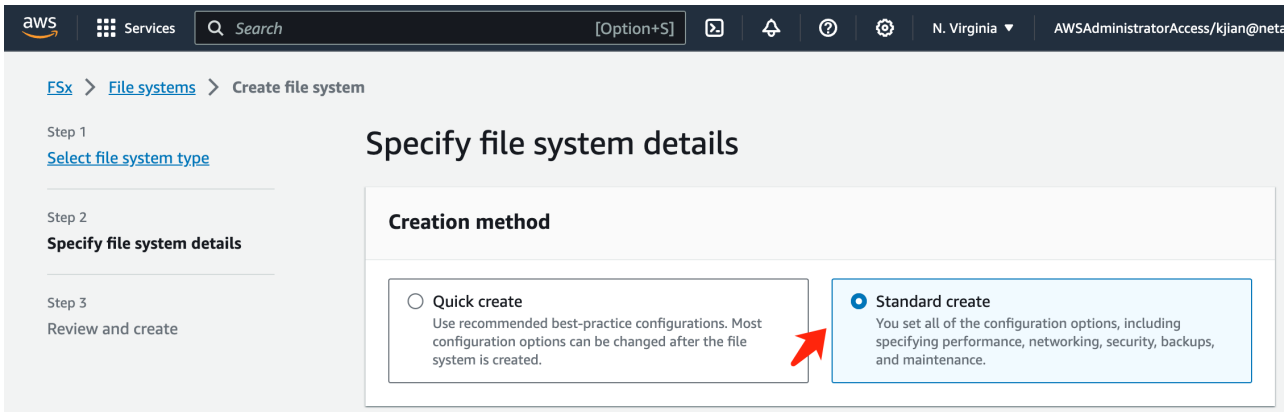

2. 单击*创建文件系统*。



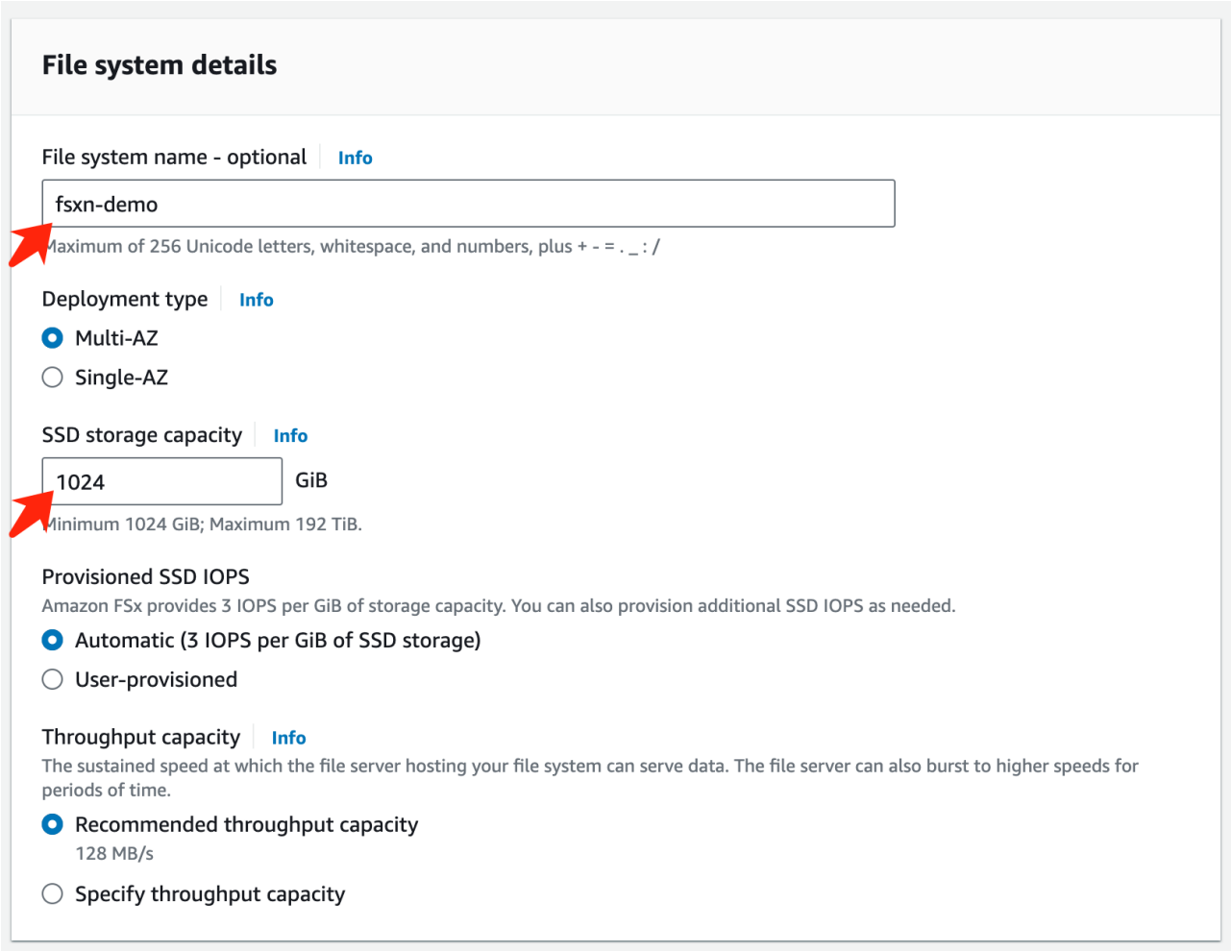

3. 选择第一张卡*FSx for FS* NetApp ONTAP，然后单击*Next*。



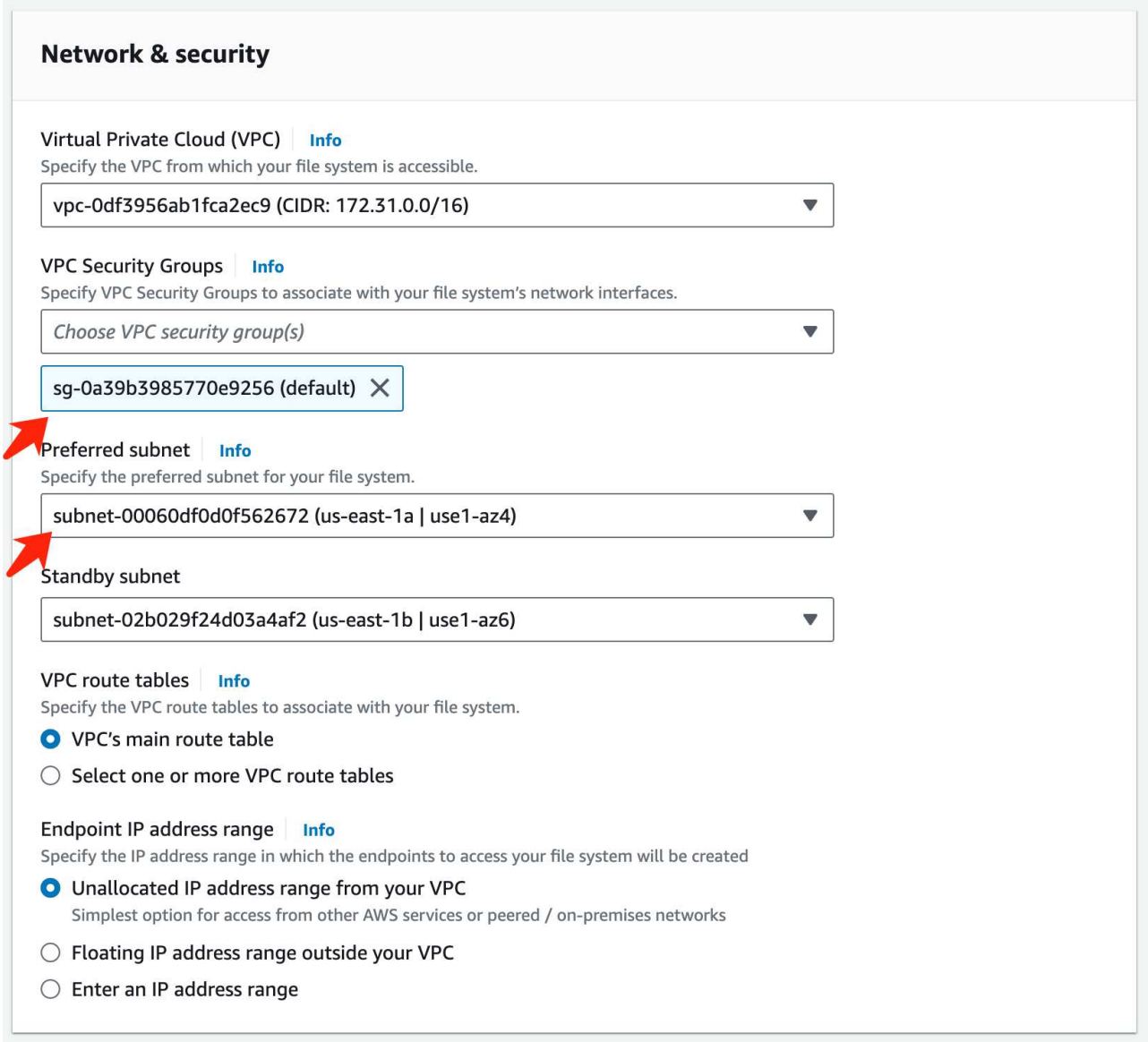

4. 在详细信息配置页面中。
   a. 选择*标准创建*选项。



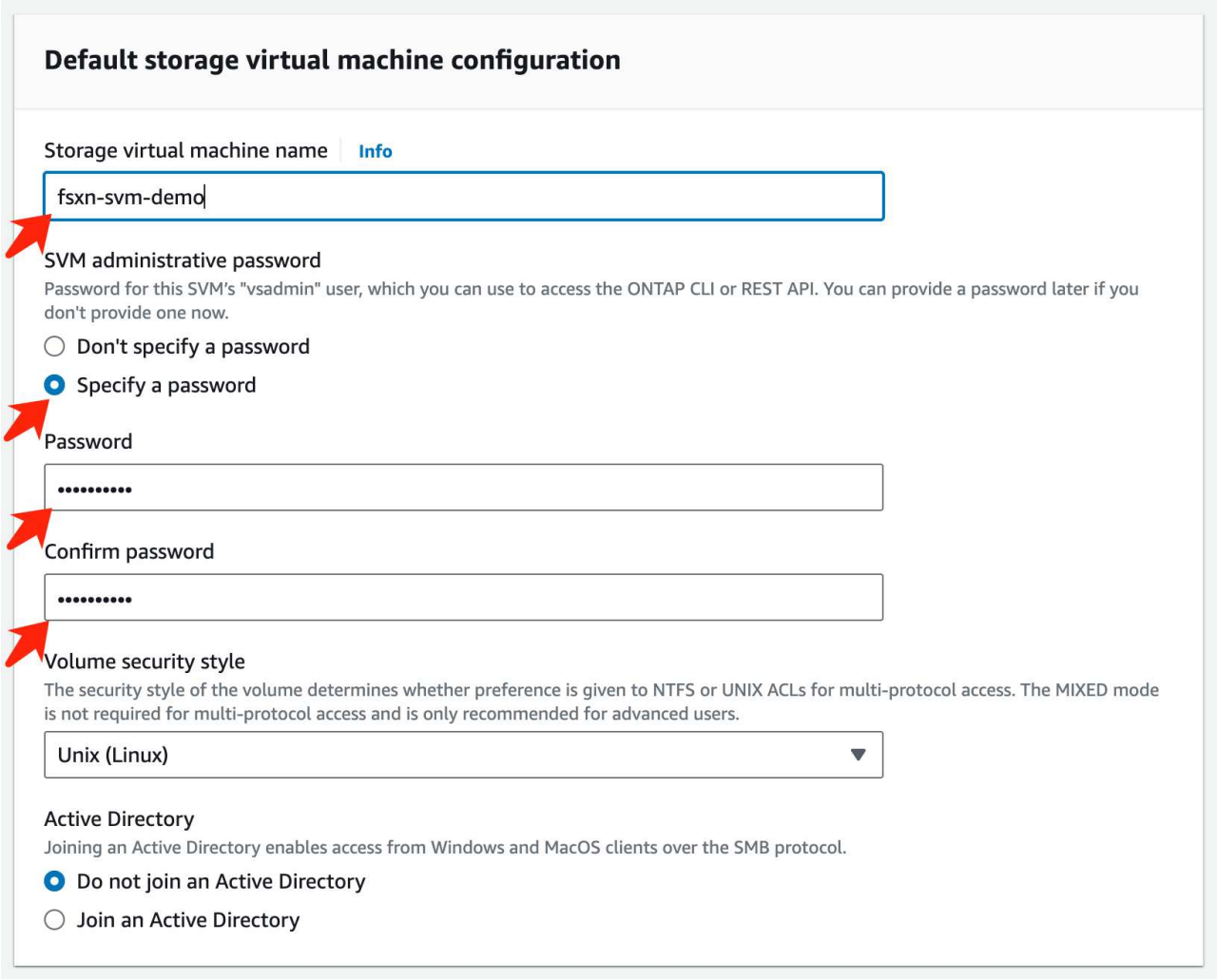

   b. 输入*文件系统名称*和* SSD存储容量*。



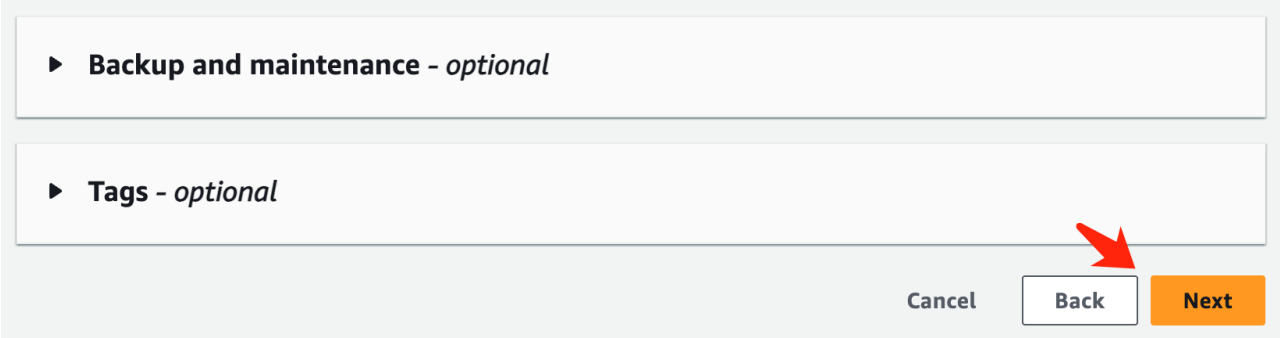

c. 确保使用与*SageMaker记事本*实例相同的*vpc*和*subnet*。



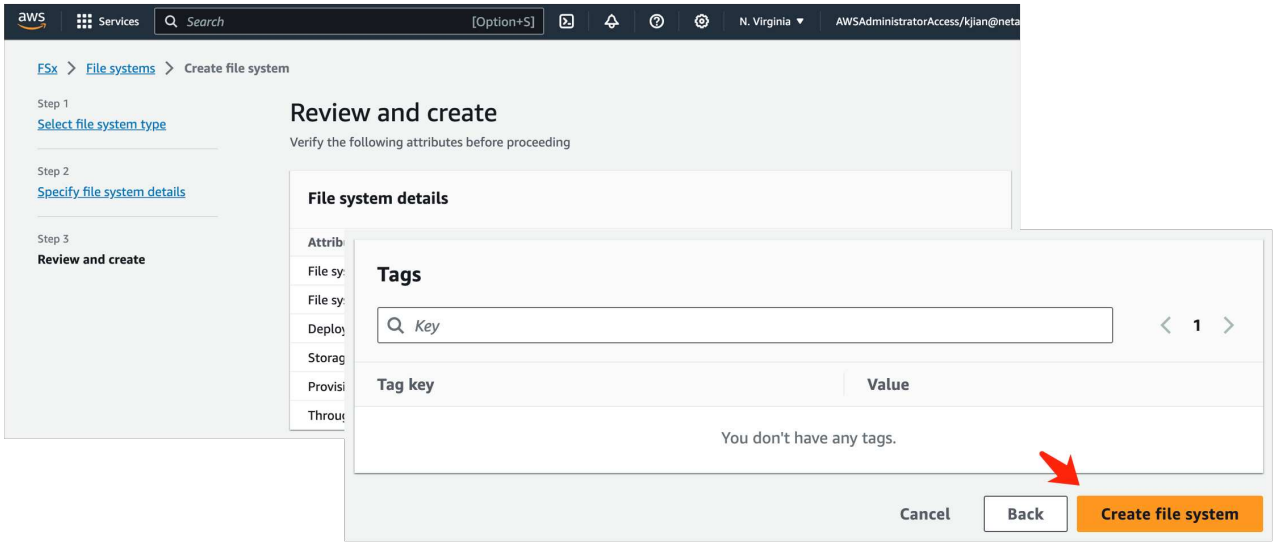

d. 输入SVM (Storage Virtual Machine)的* Storage Virtual Machine*名称和*指定密码*。



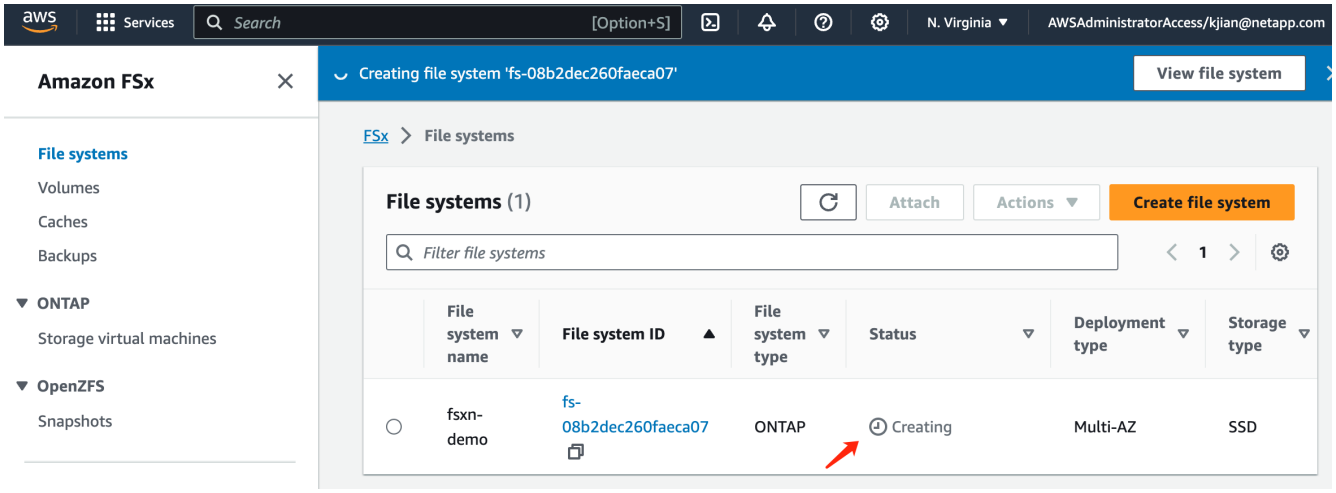

e. 保留其它条目的默认值，然后单击右下角的橙色按钮*Next*。




f. 单击查看页面右下角的橙色按钮*创建文件系统*。



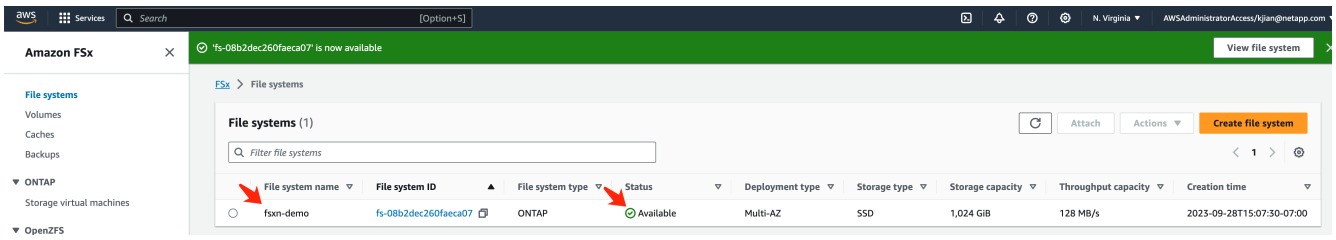

5. 启动FSx文件系统可能需要大约*20-40分钟*。



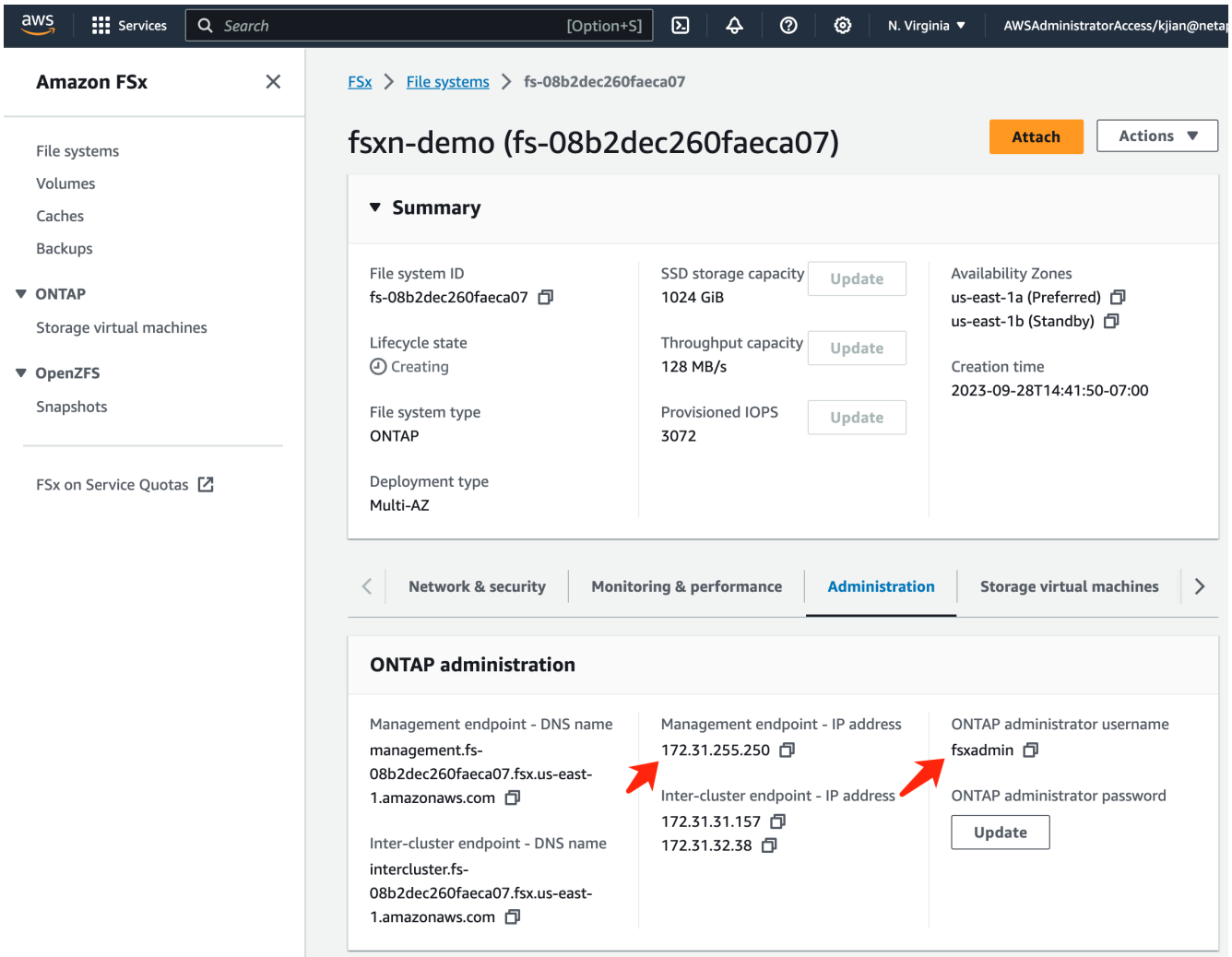

服务器配置

## ONTAP配置

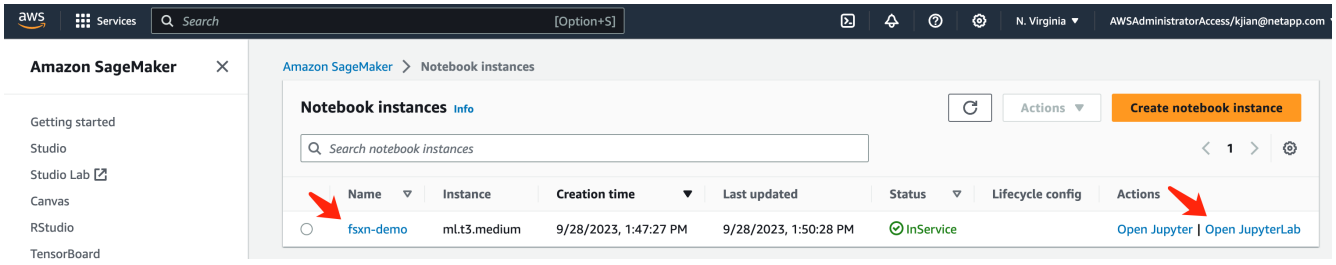
1. 打开创建的FSx文件系统。请确保状态为*可用*。



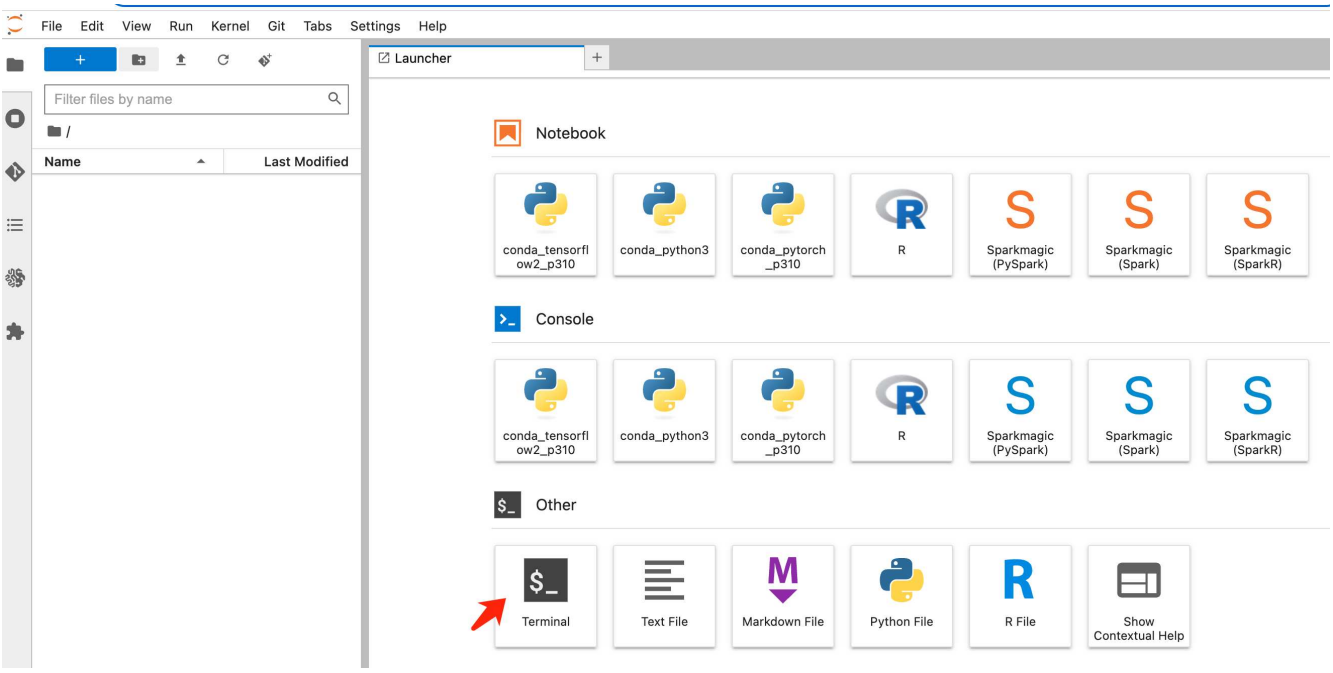

2. 选择*管理*选项卡并保留*管理端点- IP地址*和* ONTAP管理员用户名*。



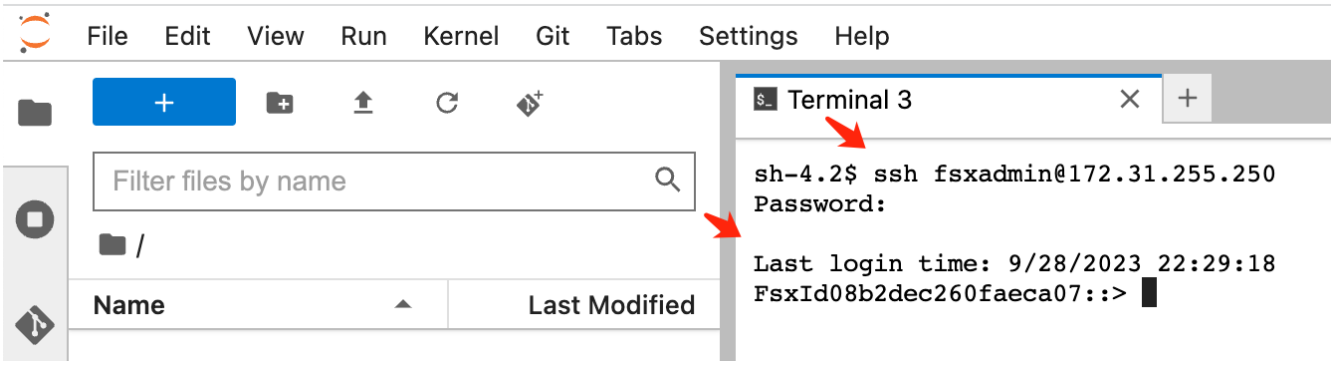

3. 打开创建的*SageMaker笔记本实例*，然后单击*Open JupyterLab*。




4. 在Jupyter Lab页面中，打开一个新的*Terminal *。

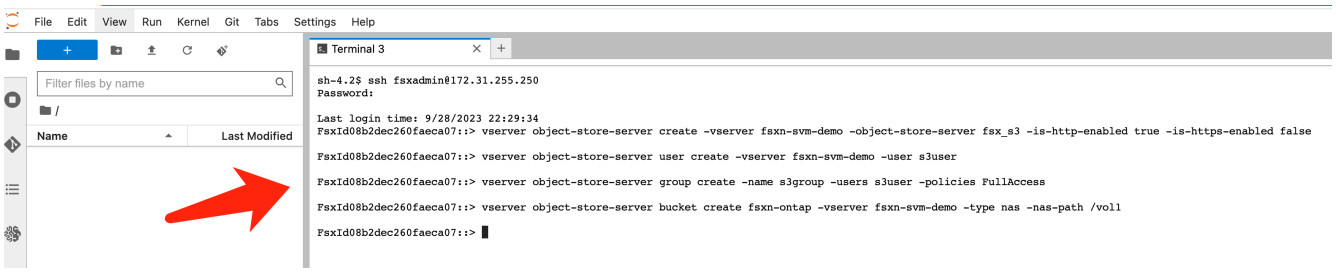



5. 输入ssh命令ssh <admin user name>@<ONTAP server IP>登录到FSxN ONTAP文件系统。(用户名和IP地址从步骤2中检索)
请使用创建*Storage Virtual Machine*时使用的密码。




6. 按以下顺序执行命令。
我们使用*fsxn-ONTAP 作为 FSxN专用S3存储分段名称*的名称。
请使用*Storage Virtual Machine name*作为*-vserver*参数。

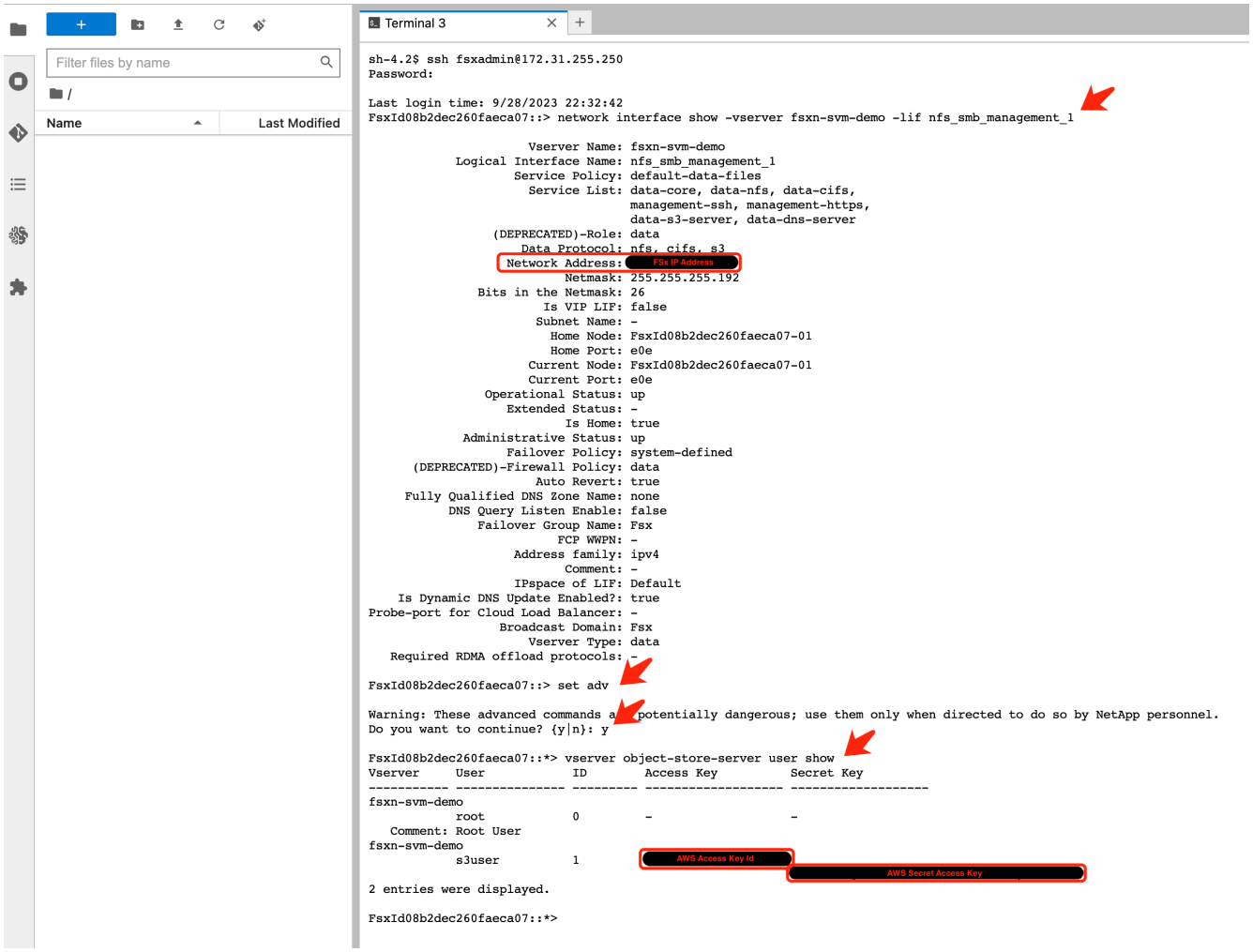
```
vserver object-store-server create -vserver fsxn-svm-demo -object-store
-server fsx_s3 -is-http-enabled true -is-https-enabled false

vserver object-store-server user create -vserver fsxn-svm-demo -user
s3user

vserver object-store-server group create -name s3group -users s3user
-policies FullAccess

vserver object-store-server bucket create fsxn-ontap -vserver fsxn-svm-
demo -type nas -nas-path /vol1
```




7. 执行以下命令以检索FSxN Private S3的端点IP和凭据。

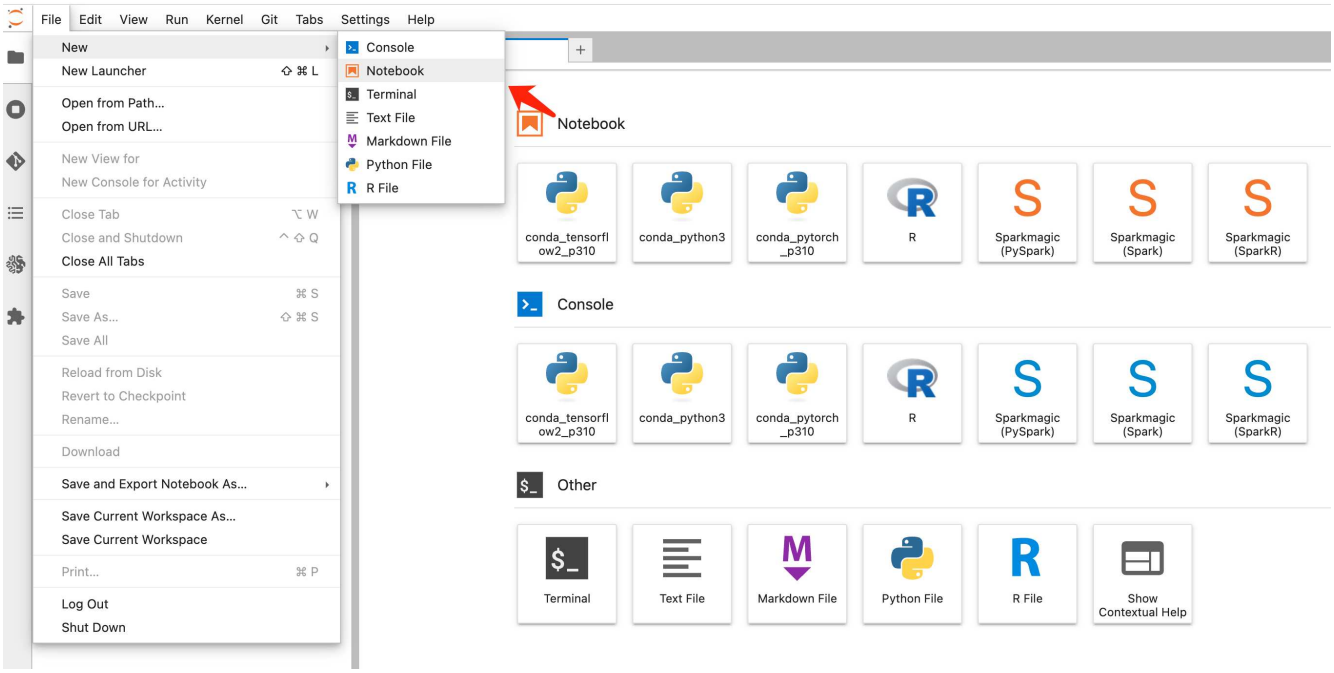
```
network interface show -vserver fsxn-svm-demo -lif nfs_smb_management_1

set adv

vserver object-store-server user show
```

8. 保留端点IP和凭据以供将来使用。



客户端配置

1. 在SageMaker笔记本实例中、创建新的Jupyter笔记本。




2. 使用以下代码作为解决解决方案问题的方法、将文件上传到FSxN私有S3存储分段。
有关完整的代码示例、请参阅本笔记本。
"fsxn_dema.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']
```

FSxN与SageMaker实例之间的集成到此结束。

有用的调试检查清单

- 确保SageMaker笔记本实例和FSxN文件系统位于同一个VPC中。

- 请记得在ONTAP上运行*set dev*命令，将权限级别设置为*dev*。

常见问题解答**(**截至**2023**年**9**月**27**日**)**

问：为什么在将文件上传到FSxN时、我在调用CreateMultipartUpload操作时收到错误"发生错误**(**未实施**)**：您请求的**S3**命令未实施"？

答：作为私有S3存储分段、FSxN支持上传高达100 MB的文件。使用S3协议时、大于100 MB的文件会划分为100 MB的区块、并调用"CreateMultipartUpload"函数。但是、当前实施的FSxN Private S3不支持此功能。

问：为什么在将文件上传到FSxN时、调用PutObject操作时收到错误"发生错误**(AccessDenied)**：访问被拒绝"？

答：要从SageMaker笔记本实例访问FSxN私有S3存储分段、请将AWS凭据切换到FSxN凭据。但是、要为实例授予写入权限、需要使用 临时解决策 解决方案 挂载存储分段并运行"chmod" shell命令来更改权限。

问：如何将FSxN Private S3存储分段与其他SageMaker ML服务集成？

答：遗憾的是、SageMaker服务SDK无法为专用S3存储分段指定端点。因此、FSxN S3与SageMaker服务不兼容、例如、SagMaker Data Rangler、SagMaker Clarify、SagMaker Glue、SagMaker Athena、SagMaker AutoML、 等。

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

作者：
Jian Jian (Ken)、NetApp高级数据和应用科学人员

简介

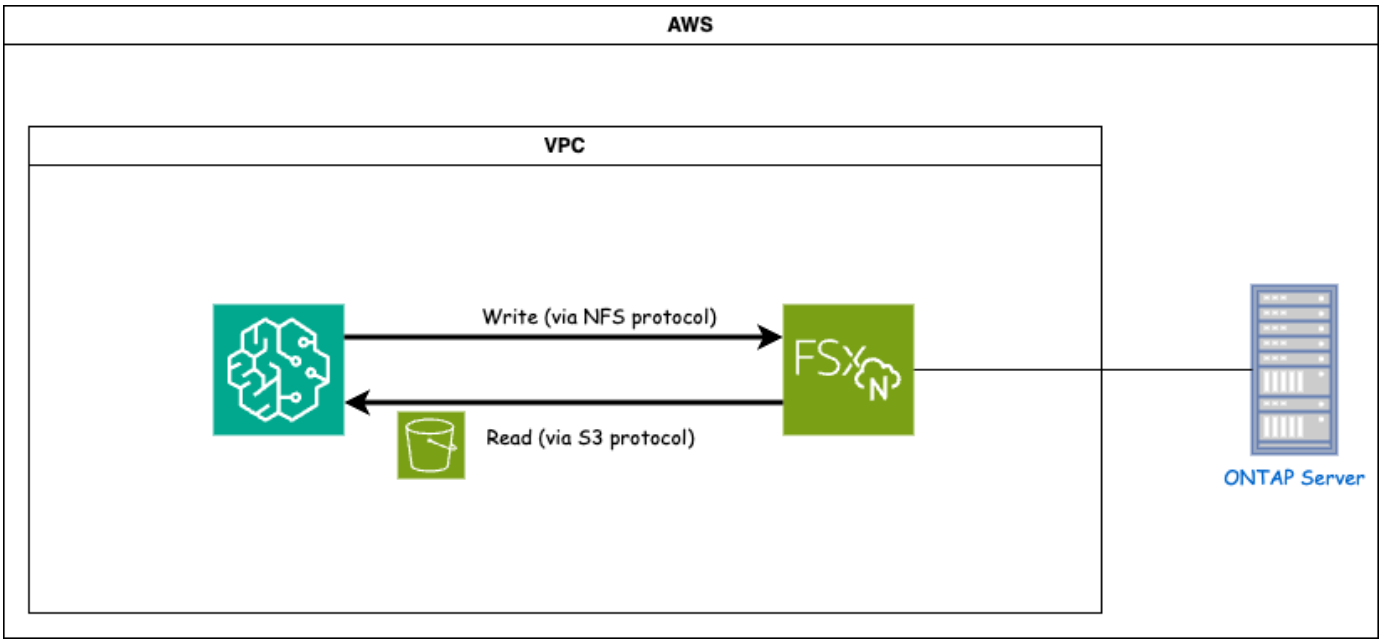
本教程提供了计算机视觉分类项目的一个实际示例、提供了在SageMaker环境中构建ML模型以利用FSxN作为数据源的实践经验。该项目侧重于使用深度学习框架PyTorch、根据轮胎图像对轮胎质量进行分类。它侧重于使用FSxN作为Amazon SageMaker中的数据源来开发机器学习模型。

什么是**FSxN**

Amazon FSx for NetApp ONTAP确实是AWS提供的一款完全托管的存储解决方案。它利用NetApp的ONTAP文件系统提供可靠的高性能存储。由于支持NFS、SMB和iSCSI等协议、因此可以从不同的计算实例和容器无缝访问。该服务旨在提供卓越的性能、确保快速高效的数据运营。它还提供高可用性和持久性、确保您的数据始终可访问并受到保护。此外、Amazon FSx for NetApp ONTAP的存储容量可扩展、使您可以根据需要轻松调整。

前提条件

网络环境



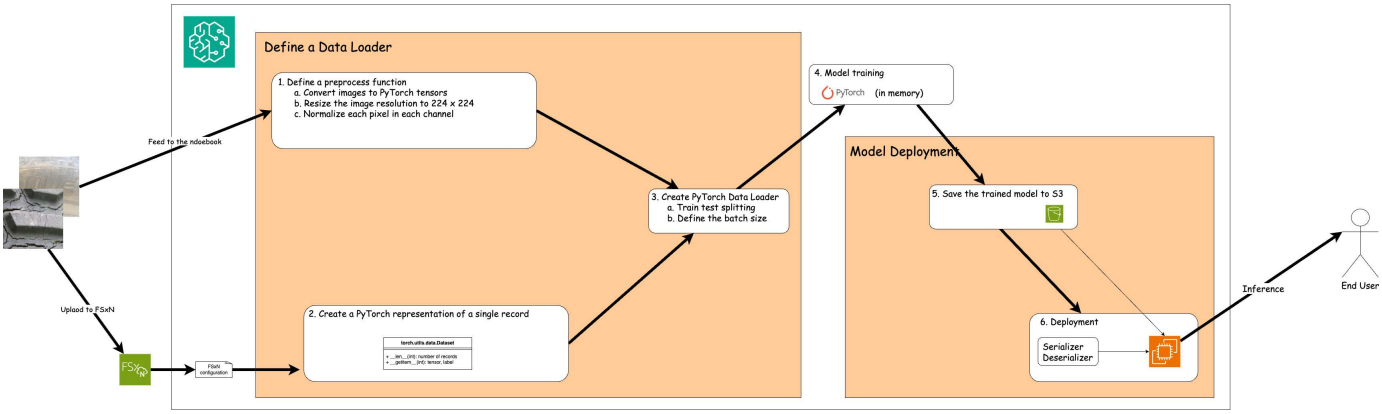

FSxN (Amazon FSx for NetApp ONTAP)是一项AWS存储服务。它包括在NetApp ONTAP系统上运行的文件系统以及与其连接的AWS托管系统虚拟机(SVM)。在提供的图中、由AWS管理的NetApp ONTAP服务器位于VPC之外。SVM充当SageMaker和NetApp ONTAP系统之间的中介、接收来自SageMaker的操作请求并将其转发到底层存储。要访问FSxN、SageMaker必须与FSxN部署位于同一个VPC中。此配置可确保SageMaker和FSxN之间的通信和数据访问。

数据访问

在实际场景中、数据科学家通常会利用存储在FSxN中的现有数据来构建机器学习模型。但是、出于演示目的、由于FSxN文件系统在创建后最初为空、因此需要手动上传训练数据。这可以通过将FSxN作为卷挂载到SageMaker来实现。成功挂载文件系统后、您可以将数据集上传到挂载位置、以便在SageMaker环境中训练模型。通过这种方法、您可以在与SageMaker合作进行模型开发和训练时利用FSxN的存储容量和功能。

数据读取过程会将FSxN配置为专用S3存储分段。要了解详细的配置说明、请参见 "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"

集成概述




使用FSxN中的训练数据在SageMaker中构建深度学习模型的工作流可概括为三个主要步骤：数据加载程序定义、模型训练和部署。总体而言、这些步骤构成了MLOps管道的基础。但是、每个步骤都涉及多个详细的子步骤、以实现全面实施。这些子步骤包括各种任务、例如数据预处理、数据集拆分、模型配置、超参数调整、模型评估、 和型号部署。这些步骤可确保在SageMaker环境中使用来自FSxN的训练数据构建和部署深度学习模型的

流程全面有效。

分步集成

数据加载程序

为了训练使用数据的PyTorch深度学习网络、我们创建了一个数据加载程序来促进数据馈送。数据加载程序不仅可以定义批大小、还可以确定用于读取和预处理批处理中每个记录的操作步骤。通过配置数据加载程序、我们可以处理批量数据处理、从而实现深度学习网络的训练。

数据加载程序由3个部分组成。

预处理功能

```
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]
    )
])
```

上述代码段演示了使用*torchVISION．transforms*模块的图像预处理转换的定义。在此图示中、创建预处理对象以应用一系列转换。首先，*ToTendor()*转换将图像转换为张图表示。随后，*Resize　224224*转换将图像的大小调整为固定的224x224像素大小。最后，*NORMDE()*转换通过减去平均值并除以沿每个通道的标准偏差来使张量值标准化。用于标准化的平均值和标准偏差值通常用于经过预先训练的神经网络模型。总之、该代码通过将图像数据转换为张量、调整图像大小和使像素值标准化来准备图像数据、以便进一步处理或输入到预先训练的模型中。

**PyTorch**数据集类

```python
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
```

此类提供了获取数据集中记录总数的功能，并定义了读取每个记录的数据的方法。在*_gottim*函数中，代码利用boto3 S3存储分段对象从FSxN中检索二进制数据。从FSxN访问数据的代码模式类似于从Amazon S3读取数据。后面的说明将深入介绍私有S3对象*bket*的创建过程。

**FSxN作为私有S3存储库**

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

要从SageMaker中的FSxN读取数据、需要创建一个处理程序、该处理程序使用S3协议指向FSxN存储。这样就可以将FSxN视为专用S3存储分段。处理程序配置包括指定FSxN SVM的IP地址、分段名称和所需凭据。有关获取这些配置项的完整说明、请参阅上的文档 "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"。

在上述示例中、b分 段对象用于实例化PyTorch DataSet对象。数据集对象将在后续章节中进一步说明。

**PyTorch数据加载程序**

```
from torch.utils.data import DataLoader
torch.manual_seed(seed)

# 1. Hyperparameters
batch_size = 64

# 2. Preparing for the dataset
dataset = FSxNImageDataset(bucket, 'dataset/tyre', preprocess=preprocess)

train, test = torch.utils.data.random_split(dataset, [1500, 356])

data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
```

在提供的示例中、指定的批大小为64、表示每个批将包含64条记录。通过将PyTorch *DataT*类、预处理功能和训练批大小相结合，我们可以获得训练所需的数据加载程序。此数据加载程序有助于在训练阶段批量迭代数据集。

模型训练

```
from torch import nn


class TyreQualityClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3,32,(3,3)),
            nn.ReLU(),
            nn.Conv2d(32,32,(3,3)),
            nn.ReLU(),
            nn.Conv2d(32,64,(3,3)),
            nn.ReLU(),
            nn.Flatten(),
            nn.Linear(64*(224-6)*(224-6),2)
        )
    def forward(self, x):
        return self.model(x)
```

```
import datetime

num_epochs = 2
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = TyreQualityClassifier()
fn_loss = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)


model.to(device)
for epoch in range(num_epochs):
    for idx, (X, y) in enumerate(data_loader):
        X = X.to(device)
        y = y.to(device)

        y_hat = model(X)

        loss = fn_loss(y_hat, y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        current_time = datetime.datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
        print(f"Current Time: {current_time} - Epoch [{epoch+1}/
{num_epochs}]- Batch [{idx + 1}] - Loss: {loss}", end='\r')
```

本规范实施标准的PyTorch培训流程。它定义了一个名为*TireQualityClassifyer*的神经网络模型，该模型使用卷积层和线性层对轮胎质量进行分类。训练循环会迭代数据批处理、并使用反向传播和优化功能来确定损失、然后更新模型的参数。此外、它还会打印当前时间、时期、批处理和损失、以供监控。

模型部署

部署

```python
import io
import os
import tarfile
import sagemaker

# 1. Save the PyTorch model to memory
buffer_model = io.BytesIO()
traced_model = torch.jit.script(model)
torch.jit.save(traced_model, buffer_model)

# 2. Upload to AWS S3
sagemaker_session = sagemaker.Session()
bucket_name_default = sagemaker_session.default_bucket()
model_name = f'tyre_quality_classifier.pth'

# 2.1. Zip PyTorch model into tar.gz file
buffer_zip = io.BytesIO()
with tarfile.open(fileobj=buffer_zip, mode="w:gz") as tar:
    # Add PyTorch pt file
    file_name = os.path.basename(model_name)
    file_name_with_extension = os.path.split(file_name)[-1]
    tarinfo = tarfile.TarInfo(file_name_with_extension)
    tarinfo.size = len(buffer_model.getbuffer())
    buffer_model.seek(0)
    tar.addfile(tarinfo, buffer_model)

# 2.2. Upload the tar.gz file to S3 bucket
buffer_zip.seek(0)
boto3.resource('s3') \
    .Bucket(bucket_name_default) \
    .Object(f'pytorch/{model_name}.tar.gz') \
    .put(Body=buffer_zip.getvalue())
```

此代码会将PyTorch模型保存到*Amazon S3*中，因为SageMaker要求将模型存储在S3中进行部署。通过将模型上传到*Amazon S3*，SageMaker便可访问模型，从而可以在已部署的模型上进行部署和引用。

```python
import time
from sagemaker.pytorch import PyTorchModel
from sagemaker.predictor import Predictor
from sagemaker.serializers import IdentitySerializer
from sagemaker.deserializers import JSONDeserializer


class TyreQualitySerializer(IdentitySerializer):
    CONTENT_TYPE = 'application/x-torch'
```

```
    def serialize(self, data):
        transformed_image = preprocess(data)
        tensor_image = torch.Tensor(transformed_image)

        serialized_data = io.BytesIO()
        torch.save(tensor_image, serialized_data)
        serialized_data.seek(0)
        serialized_data = serialized_data.read()

        return serialized_data


class TyreQualityPredictor(Predictor):
    def __init__(self, endpoint_name, sagemaker_session):
        super().__init__(
            endpoint_name,
            sagemaker_session=sagemaker_session,
            serializer=TyreQualitySerializer(),
            deserializer=JSONDeserializer(),
        )

sagemaker_model = PyTorchModel(
    model_data=f's3://{bucket_name_default}/pytorch/{model_name}.tar.gz',
    role=sagemaker.get_execution_role(),
    framework_version='2.0.1',
    py_version='py310',
    predictor_cls=TyreQualityPredictor,
    entry_point='inference.py',
    source_dir='code',
)

timestamp = int(time.time())
pytorch_endpoint_name = '{}-{}-{}'.format('tyre-quality-classifier', 'pt',
timestamp)
sagemaker_predictor = sagemaker_model.deploy(
    initial_instance_count=1,
    instance_type='ml.p3.2xlarge',
    endpoint_name=pytorch_endpoint_name
)
```

此代码有助于在SageMaker上部署PyTorch模型。它定义了一个自定义的串口器*TireQuality串 口器*，该串口器可将输入数据作为PyTorch张量进行预处理和串口处理。TireQuality谓 词*类是一个自定义的预测程序，它利用定义的串列器和*JSONDeseriiter*。该代码还会创建一个*PyTorchModel*对象，用于指定模型的S3位置、IAM角色、框架版本和引用入口点。代码会生成时间戳并根据模型和时间戳构建端点名称。最后、使用Deploy方法部署模型、并指定实例计数、实例类型和生成的端点名称。这样、可以在SageMaker上部署PyTorch模型并可用于进行推入。

参考

```
image_object = list(bucket.objects.filter('dataset/tyre'))[0].get()
image_bytes = image_object['Body'].read()

with Image.open(with Image.open(BytesIO(image_bytes)) as image:
    predicted_classes = sagemaker_predictor.predict(image)

    print(predicted_classes)
```

这是使用已部署端点执行此假定的示例。

## 第3部分-构建简化的MLOps管道(CI/CT/CD)

### 作者：
### Jian Jian (Ken)、NetApp高级数据和应用科学人员

简介

在本教程中、您将了解如何利用各种AWS服务构建一个简单的MLOps管道、其中包括持续集成(CI)、持续培训(CT)和持续部署(CD)。与传统DevOps管道不同、MLOps需要额外的注意事项才能完成运营周期。通过学习本教程、您将深入了解如何将CT整合到MLOps循环中、从而可以持续训练您的模型并无缝部署数据进行推导。本教程将指导您完成利用AWS服务建立此端到端MLOps管道的过程。

清单文件

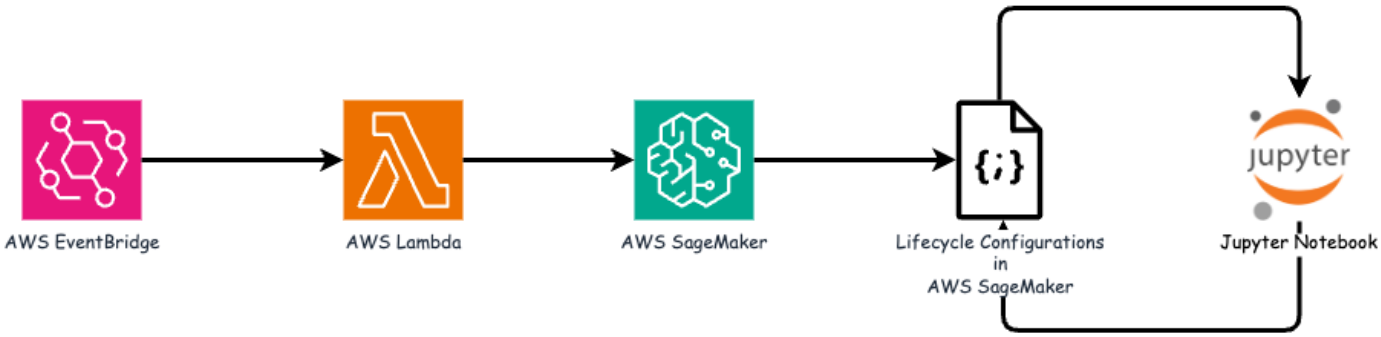
| 功能 | Name | comment |
|---|---|---|
| 数据存储 | AWS FSxN | 请参见 "第1部分—将AWS FSx for NetApp ONTAP (FSxN)作为私有S3存储分段集成到AWS SageMaker中"。 |
| 数据科学IDE | AWS SageMaker | 本教程基于中提供的Jupyter笔记本"第2部分—利用AWS FSx for NetApp ONTAP (FSxN)作为SageMaker模型训练的数据源"。 |
| 用于触发MLOps管道的功能 | AWS Lamb开发 函数 | - |
| cron作业触发器 | AWS EventBridge | - |
| 深度学习框架 | PyTorch | - |
| AWS Python SDK | 僵尸3 | - |
| 编程语言 | Python | v3.10 |

前提条件

- 一种预配置的FSxN文件系统。本教程将利用存储在FSxN中的数据进行培训。

- 一个* SageMaker笔记本电脑实例*，该实例配置为与上述FSxN文件系统共享同一个VPC。
- 在触发*AWS Lambd*函数之前，请确保*SageMaker笔记本实例*处于*STOPPED*状态。
- 要利用深度神经网络的必要GPU加速、需要使用*毫升g4dn.x大*实例类型。

架构




此MLOps管道是一种实际实施、它利用cron作业触发无服务器功能、进而执行使用生命周期回调函数注册的AWS服务。AWS EventBridge*用作cron作业。它会定期调用一个*AWS Lambad*函数，负责对模型进行重新培训和重新部署。此过程涉及到启动*AWS SageMaker笔记本*实例以执行必要的任务。

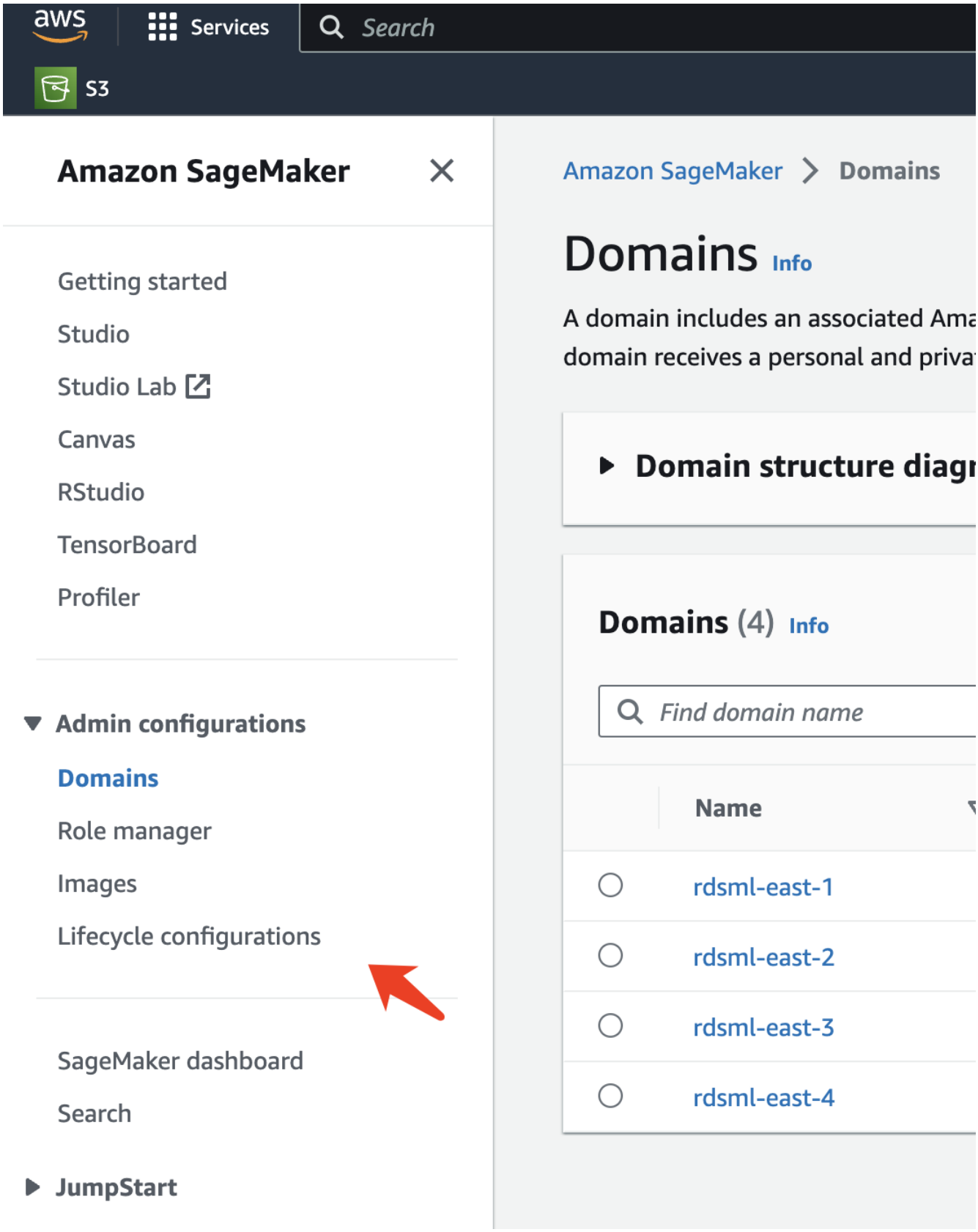
逐步配置

#### 生命周期配置

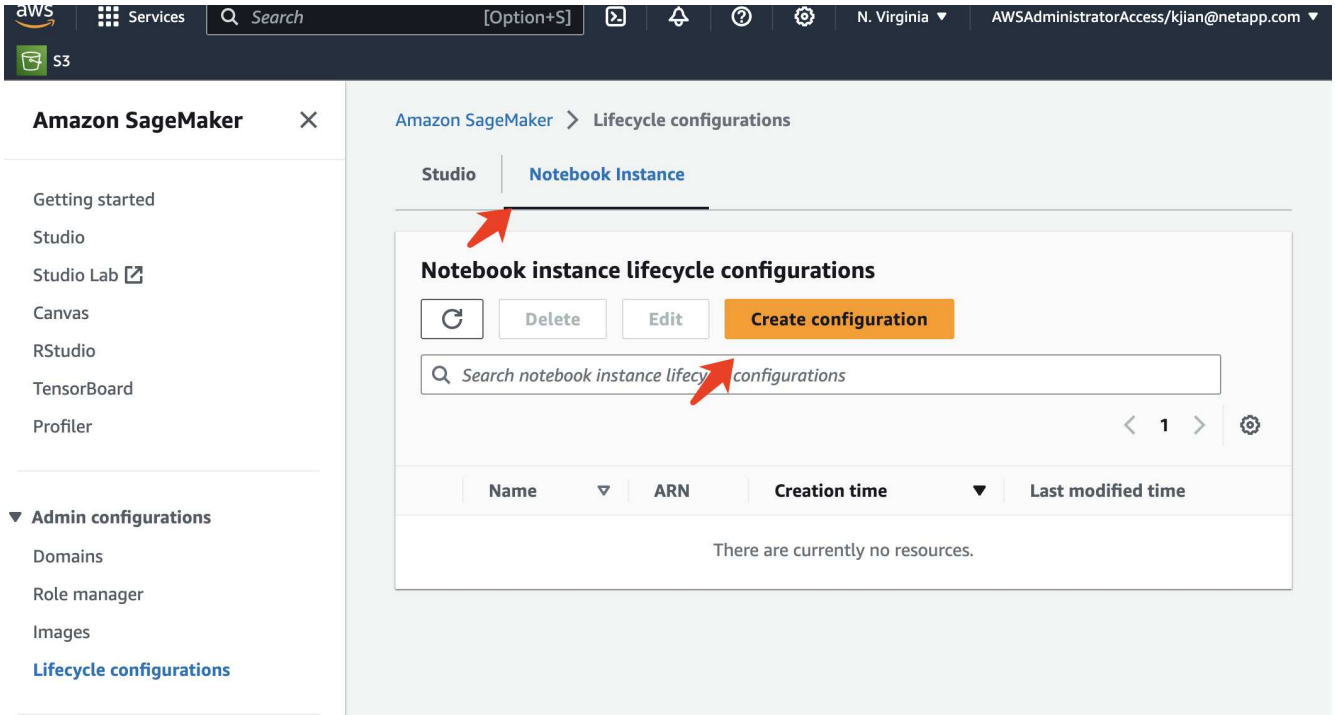
要为AWS SageMaker笔记本实例配置生命周期回调函数，应使用*Lifecycle configurations*。通过此服务，您可以定义在启动笔记本实例期间要执行的必要操作。具体而言，可以在*Lifecycle configuration*中实施shell脚本，以便在完成培训和部署过程后自动关闭笔记本实例。这是必需的配置、因为成本是MLOps中的主要考虑因素之一。

需要注意的是，需要提前设置*生命周期配置*的配置。因此、建议在继续其他MLOps管道设置之前、优先配置此方面。

1. 要设置生命周期配置，请打开*Sager*面板，然后导航到*Admin configurations*部分下的*Lifecycle configurations*。



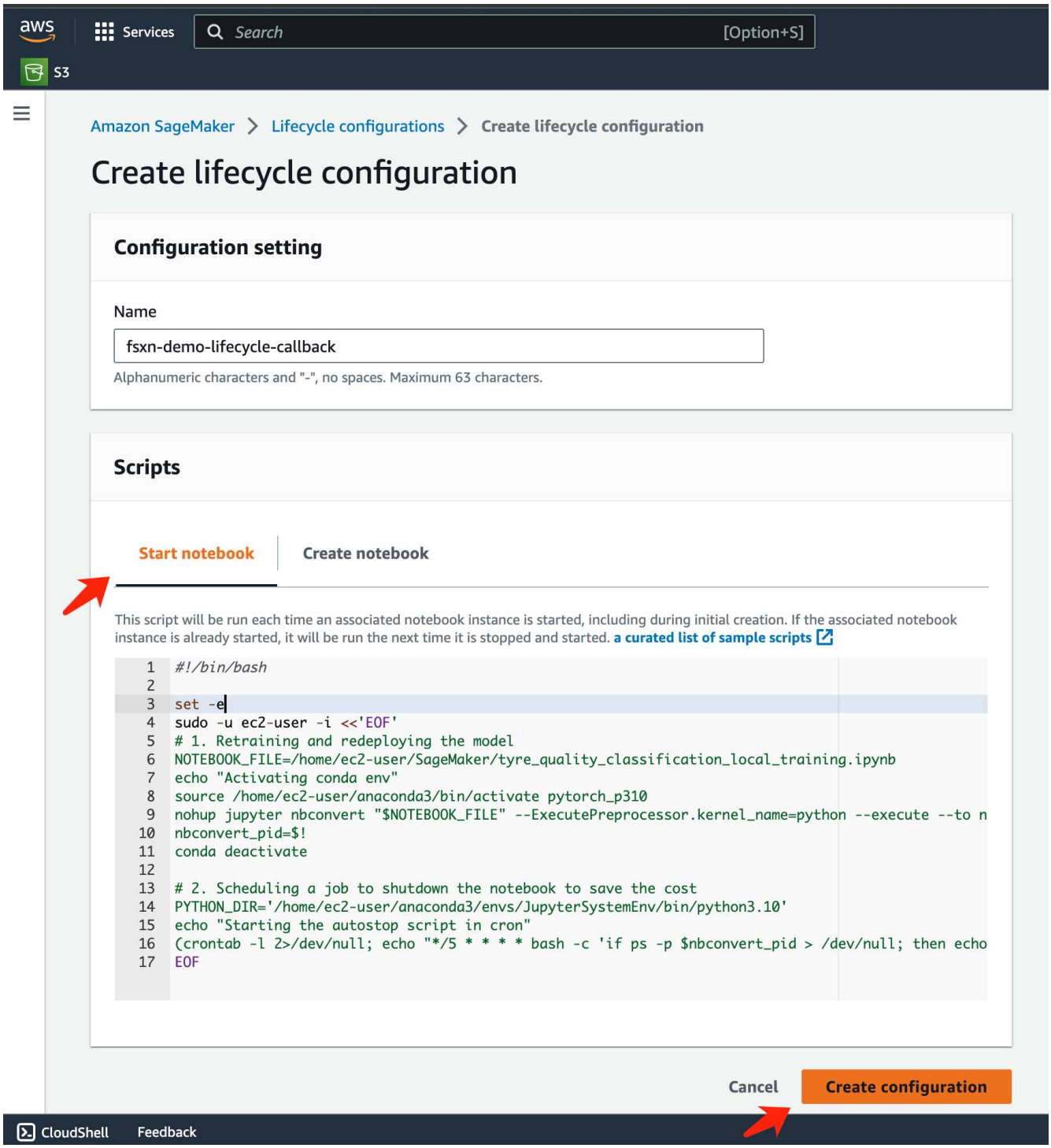

2. 选择*笔记本实例*选项卡，然后单击*创建配置*按钮



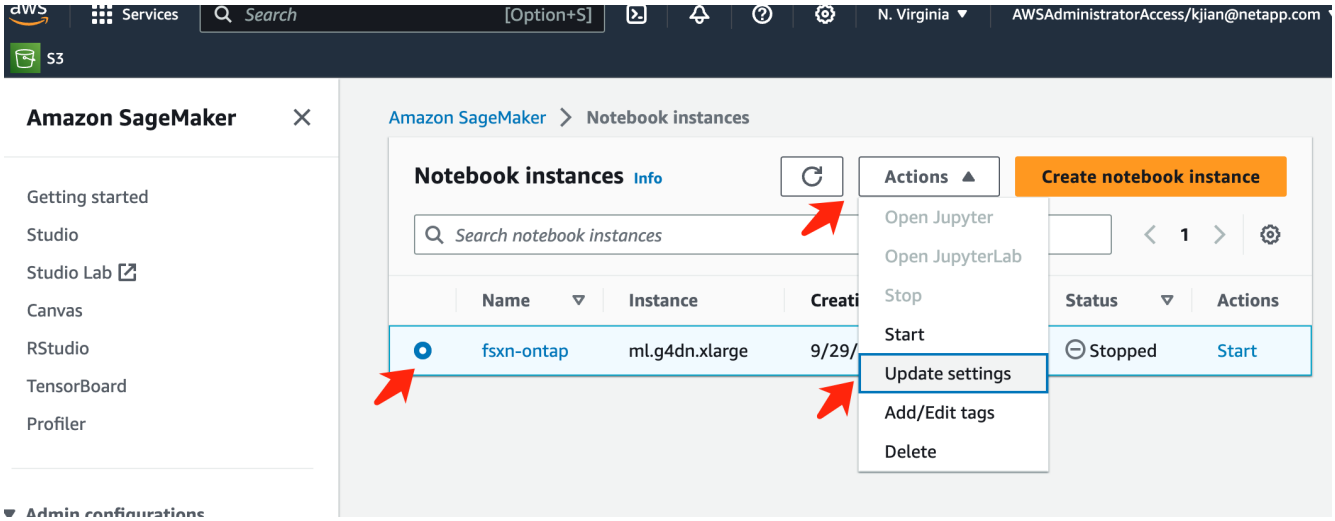

3. 将以下代码粘贴到输入区域。

```
#!/bin/bash

set -e
sudo -u ec2-user -i <<'EOF'
# 1. Retraining and redeploying the model
NOTEBOOK_FILE=/home/ec2-
user/SageMaker/tyre_quality_classification_local_training.ipynb
echo "Activating conda env"
source /home/ec2-user/anaconda3/bin/activate pytorch_p310
nohup jupyter nbconvert "$NOTEBOOK_FILE"
--ExecutePreprocessor.kernel_name=python --execute --to notebook &
nbconvert_pid=$!
conda deactivate

# 2. Scheduling a job to shutdown the notebook to save the cost
PYTHON_DIR='/home/ec2-
user/anaconda3/envs/JupyterSystemEnv/bin/python3.10'
echo "Starting the autostop script in cron"
(crontab -l 2>/dev/null; echo "*/5 * * * * bash -c 'if ps -p
$nbconvert_pid > /dev/null; then echo \"Notebook is still running.\" >>
/var/log/jupyter.log; else echo \"Notebook execution completed.\" >>
/var/log/jupyter.log; $PYTHON_DIR -c \"import boto3;boto3.client(
\'sagemaker\').stop_notebook_instance(NotebookInstanceName=get_notebook_
name())\" >> /var/log/jupyter.log; fi'") | crontab -
EOF
```

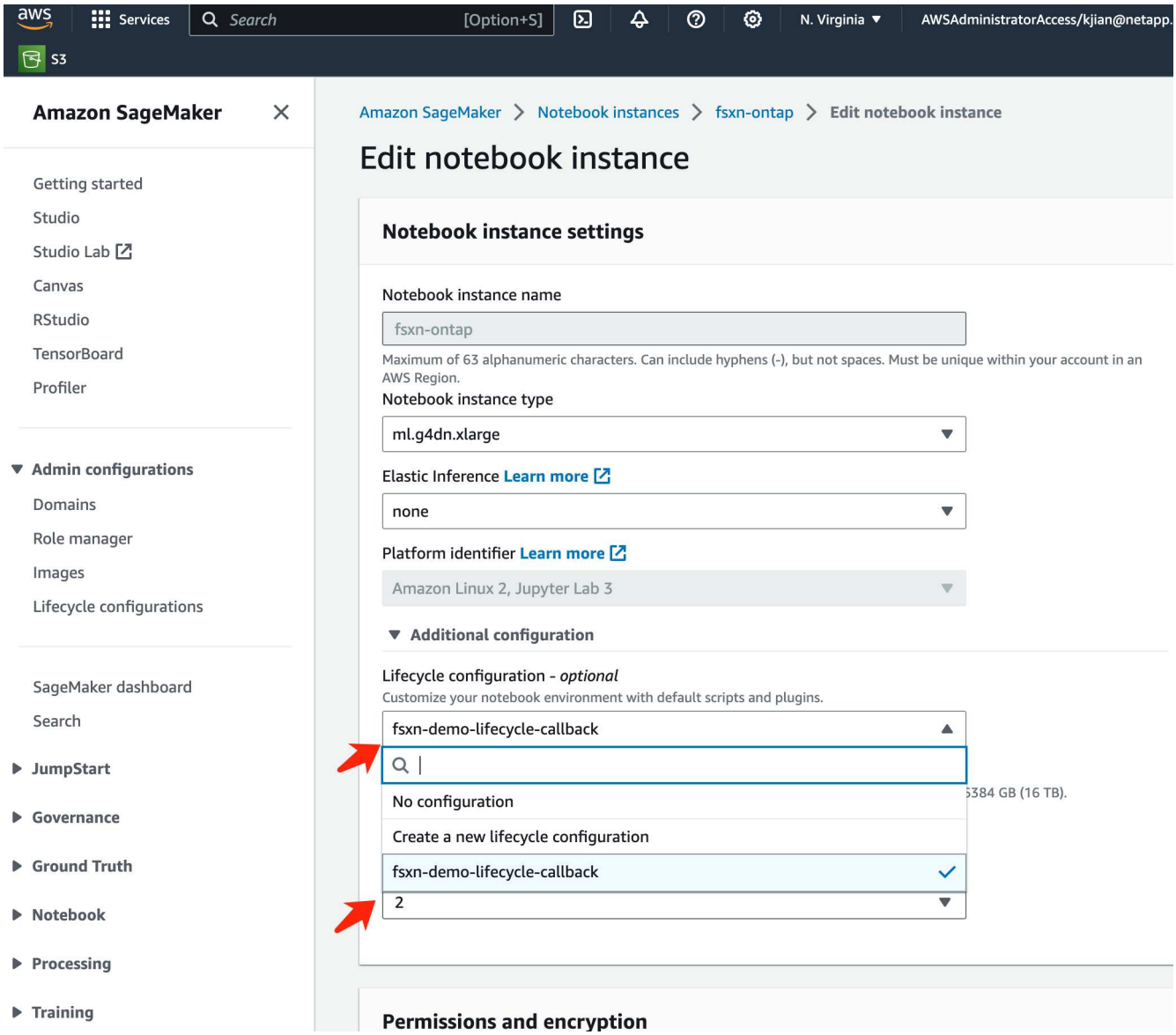
4. 此脚本执行Jupyter笔记本、该笔记本负责重新训练和重新部署模型以进行引用。执行完成后、笔记本电脑将在5分钟内自动关闭。要了解有关问题陈述和代码实施的更多信息、请参见 "第2部分—利用AWS FSx for NetApp ONTAP (FSxN)作为SageMaker模型训练的数据源"。




5. 创建后，导航到“笔记本实例”，选择目标实例，然后单击“操作”下拉列表中的*更新设置*。



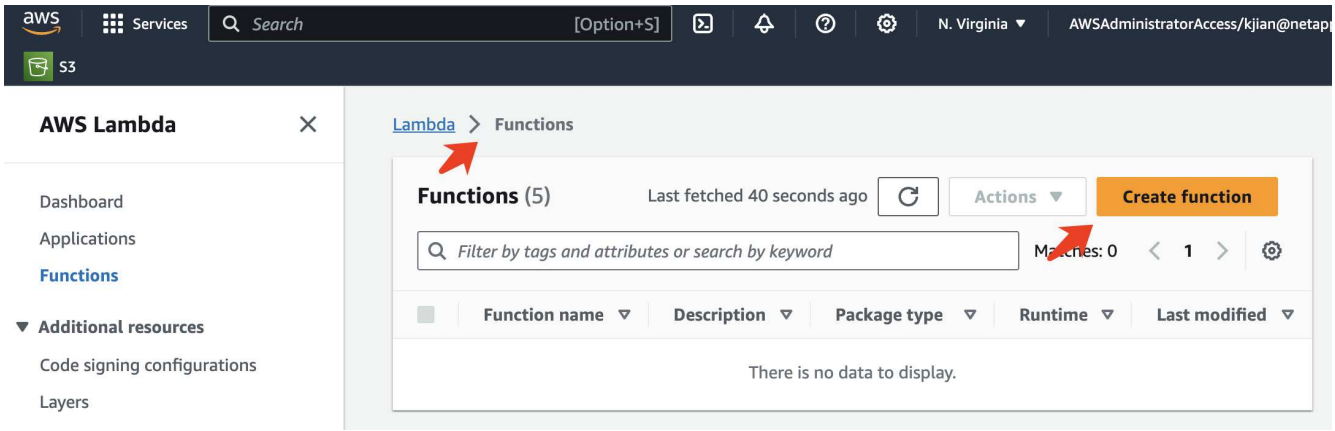

6. 选择已创建的*生命周期配置*，然后单击*更新笔记本实例*。



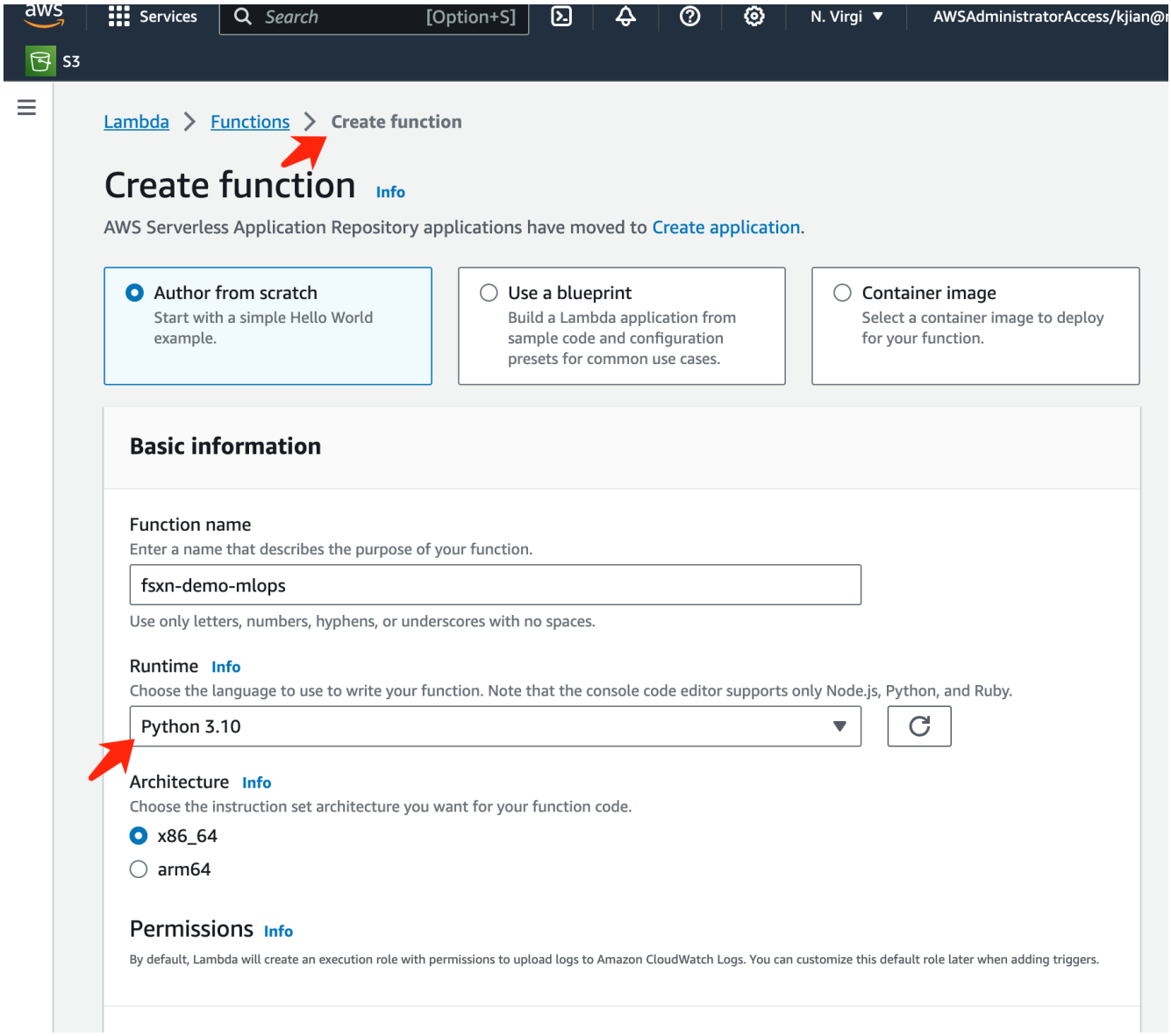
**AWS Lamb达 无服务器函数**

如前所述，**AWS Lambd*功能负责启动*AWS SageMaker**笔记本实例。

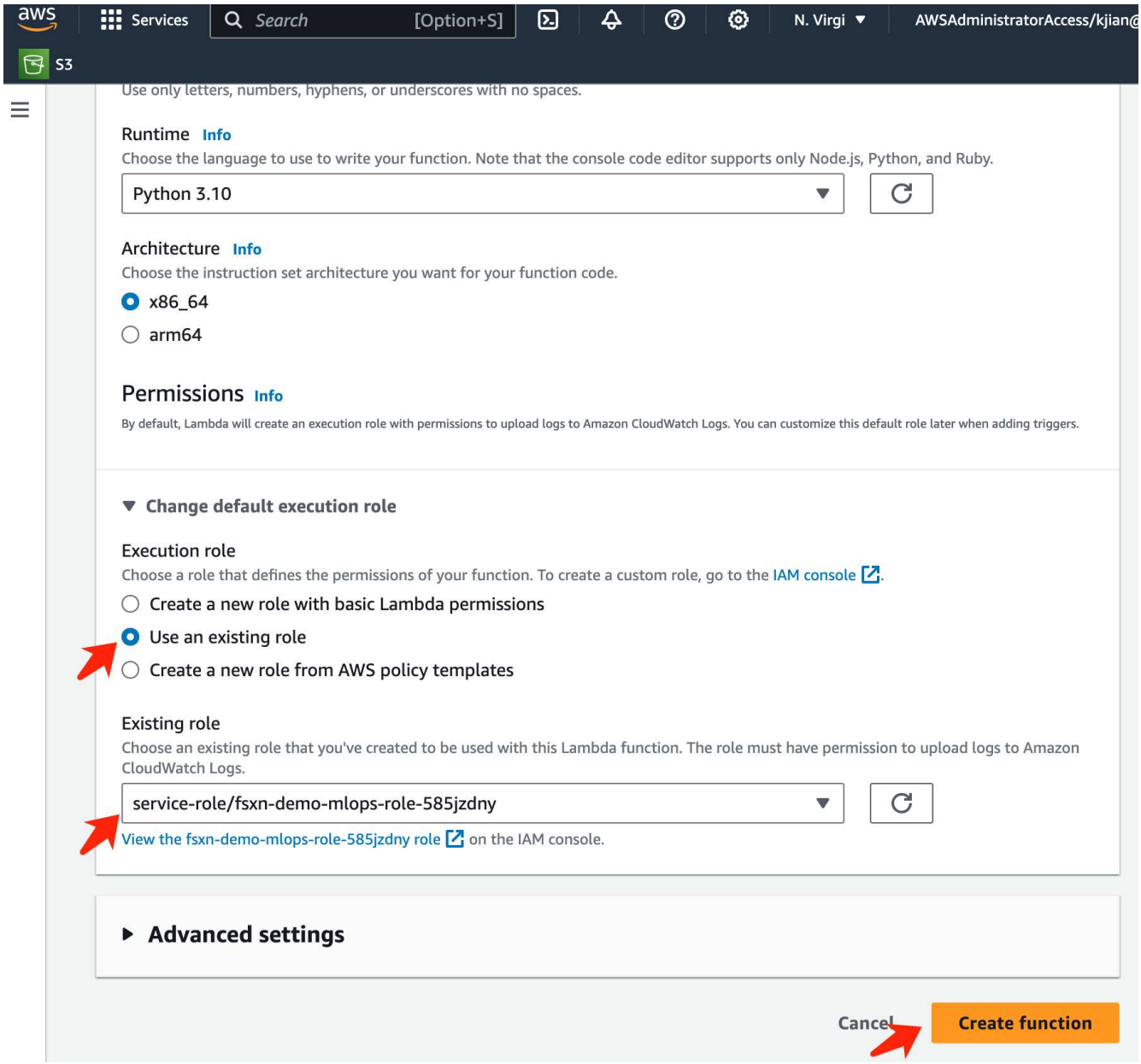
1. 要创建*AWS Lamba函数*，请导航到相应的面板，切换到*FUNCHUDENTS*选项卡，然后单击*Create FUNCHUD*。




2. 请将页面上所有必需的条目归档，并记住将运行时切换到*Python 3.10*。



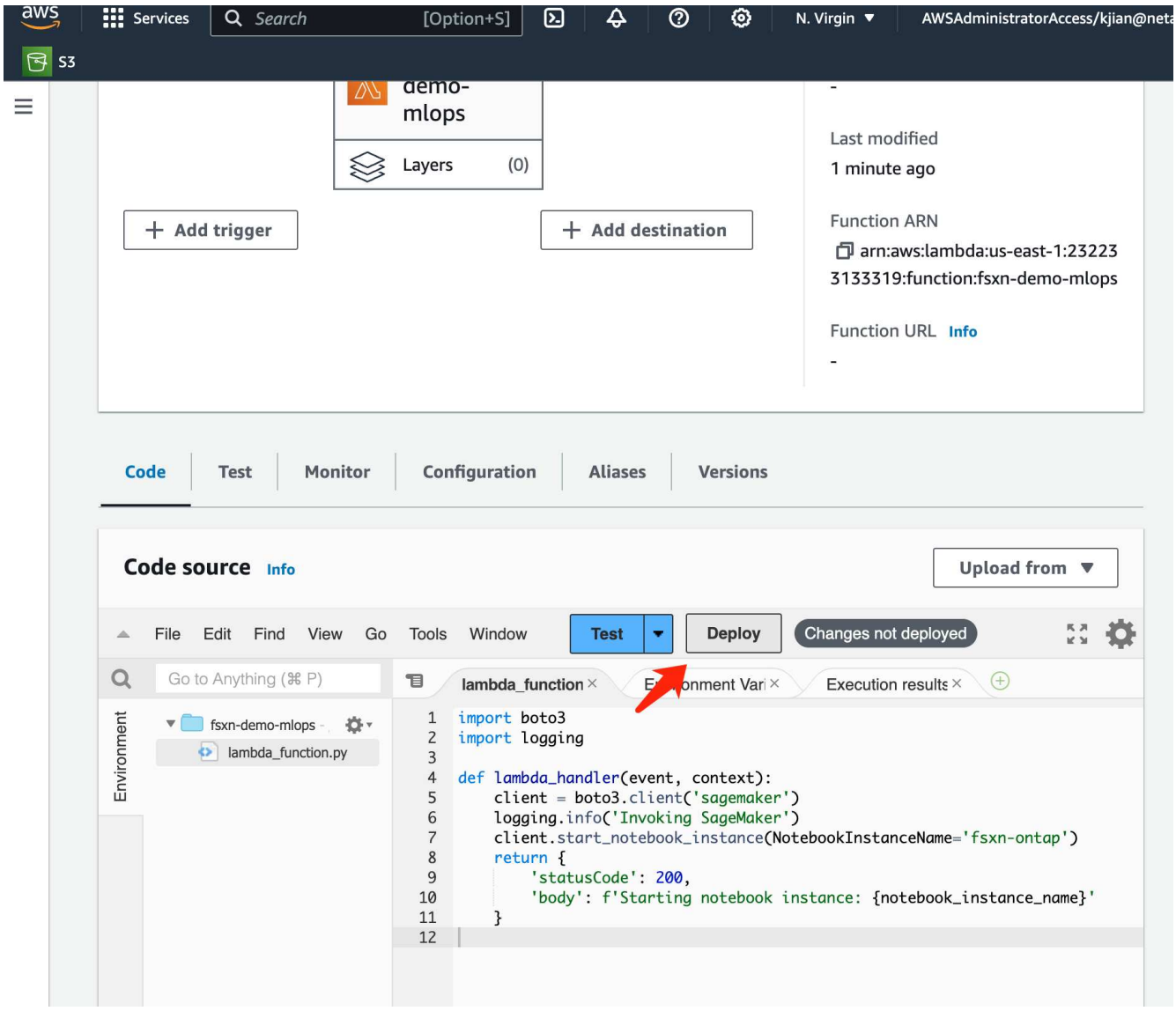

3. 请验证指定角色是否具有所需的权限*Amazon SageMakerFullAccess*，然后单击*Create Function (创建功能)*按钮。



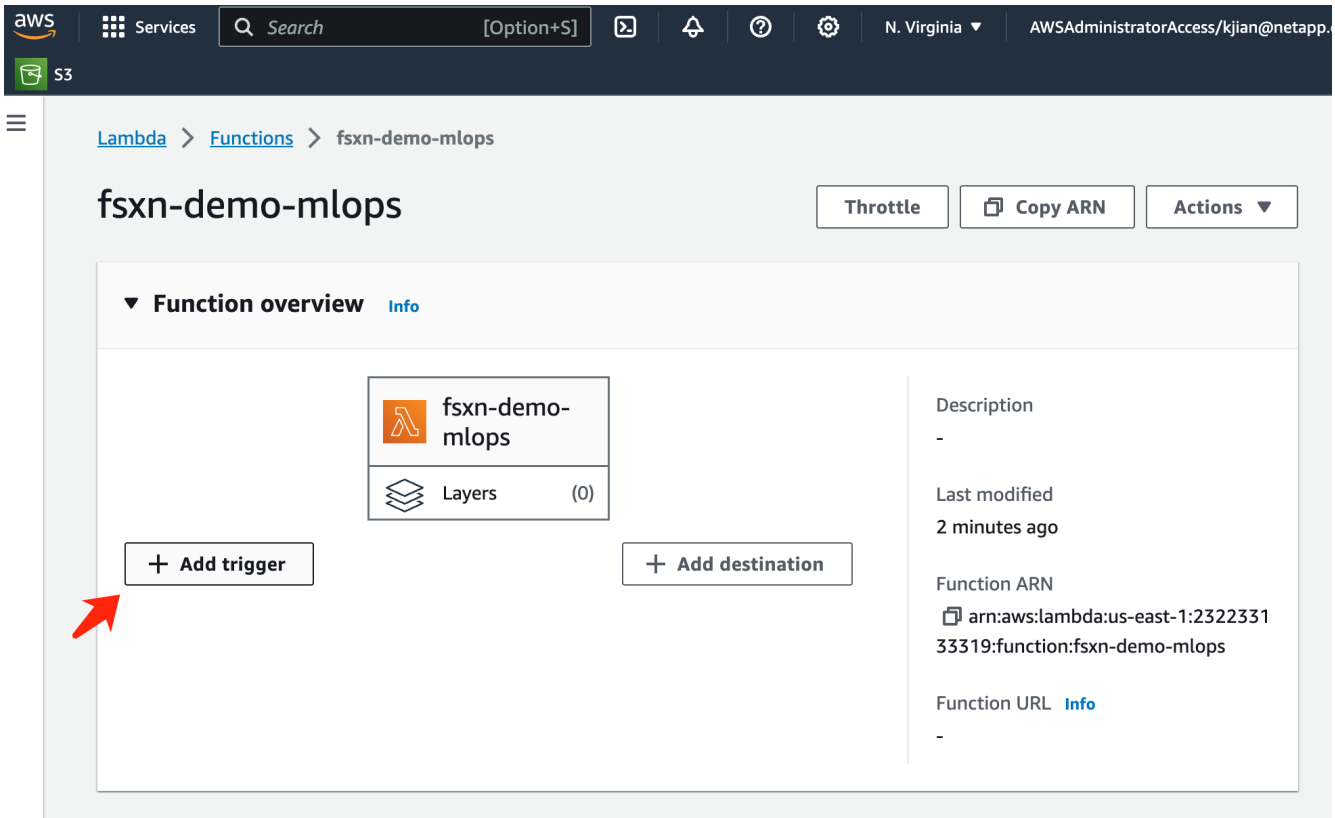

4. 选择创建的Lamb编制 函数。在代码选项卡中、将以下代码复制并粘贴到文本区域中。此代码将启动名为*fsxn-ONTAP的笔记本实例。

```
import boto3
import logging

def lambda_handler(event, context):
    client = boto3.client('sagemaker')
    logging.info('Invoking SageMaker')
    client.start_notebook_instance(NotebookInstanceName='fsxn-ontap')
    return {
        'statusCode': 200,
        'body': f'Starting notebook instance: {notebook_instance_name}'
    }
```

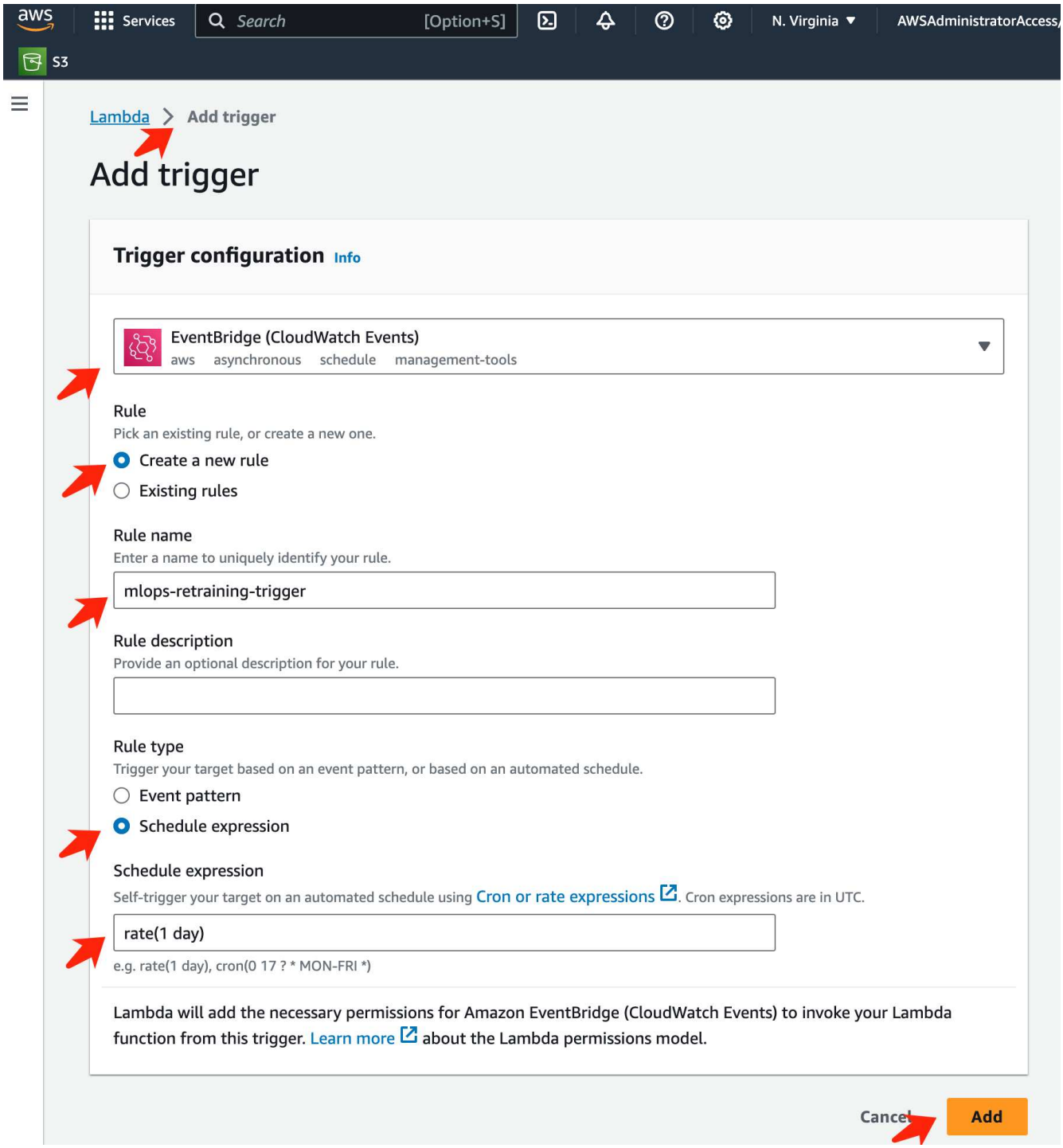
5. 单击*DEPLE*按钮以应用此代码更改。




6. 要指定如何触发此AWS Lambar函数、请单击添加触发器按钮。




7. 从下拉菜单中选择EventBridge、然后单击标有创建新规则的单选按钮。在计划表达式字段中、输入 `rate(1 day)`，然后单击添加按钮以创建此新的cron作业规则并将其应用于AWS Lamb另 一个函数。




每天完成两步配置后，**AWS Lambd***功能将启动***SageMaker**笔记本，使用*FSxN*存储库中的数据执行模型重新训练，将更新的模型重新部署到生产环境，并自动关闭*SageMaker笔记本实例*以优化成本。这可确保模型保持最新。

开发MLOps管道的教程到此结束。

# 采用Domino数据实验室和NetApp的混合多云MLOps

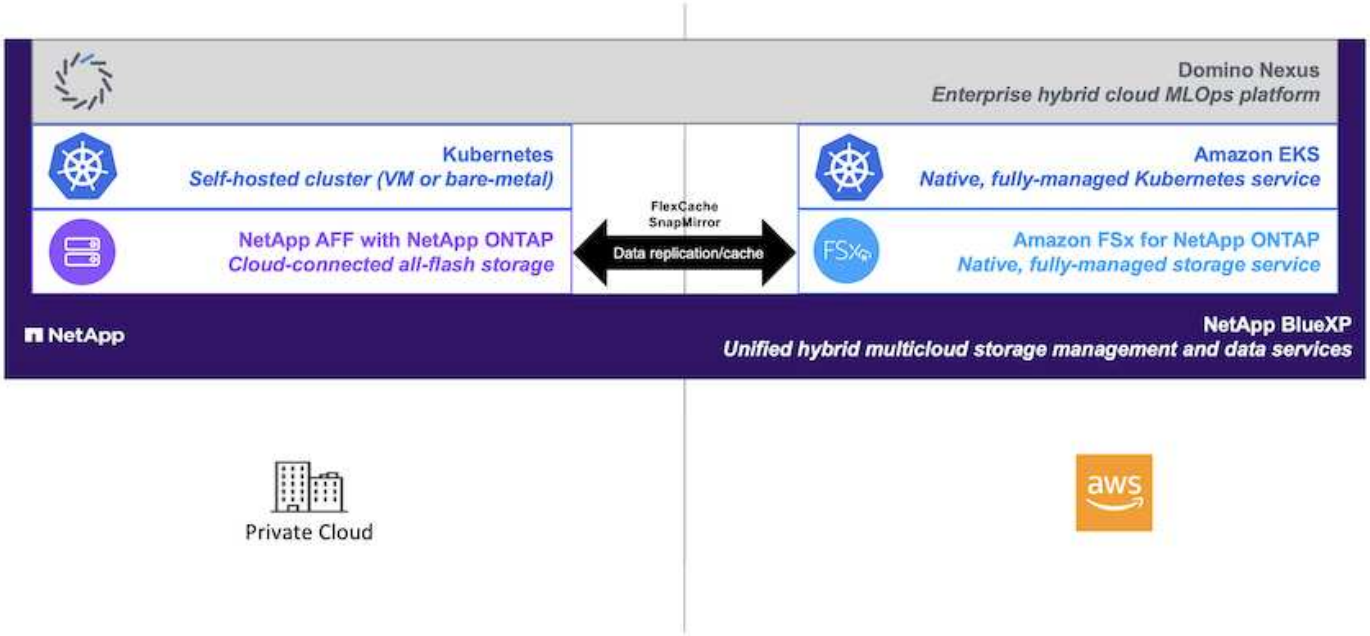
## 采用Domino数据实验室和NetApp的混合多云MLOps

NetApp 公司 Mike Oglesby

目前、全球各地的企业都在采用AI来实现业务和流程转型。因此、AI就绪的计算基础架构往往短缺。为了充分利用不同区域、数据中心和云之间的可用计算环境、企业纷纷采用混合多云MLOps架构、从而平衡成本、可用性和性能。

Domino Data Lab的Domino Nexus是一个统一的MLOps控制平台、支持您跨任何云、区域或内部环境中的任何计算集群运行数据科学和机器学习工作负载。它统一了整个企业中的数据科学孤岛、让您有一个地方来构建、部署和监控模型。同样、NetApp的混合云数据管理功能使您能够将数据带到工作和工作空间中、无论这些数据在何处运行。将Domino Nexus与NetApp配对后、您可以灵活地跨环境计划工作负载、而无需担心数据可用性。换言之、您可以将工作负载和数据发送到相应的计算环境、从而加快AI部署速度、同时遵守有关数据隐私和控制权的法规。

此解决方案演示了如何部署统一的MLOps控制平面、其中包括内部Kubelnetes集群和在Amazon Web Services (AWS)中运行的Elapic Kubelnetes Service (EKS)集群。




## 技术概述

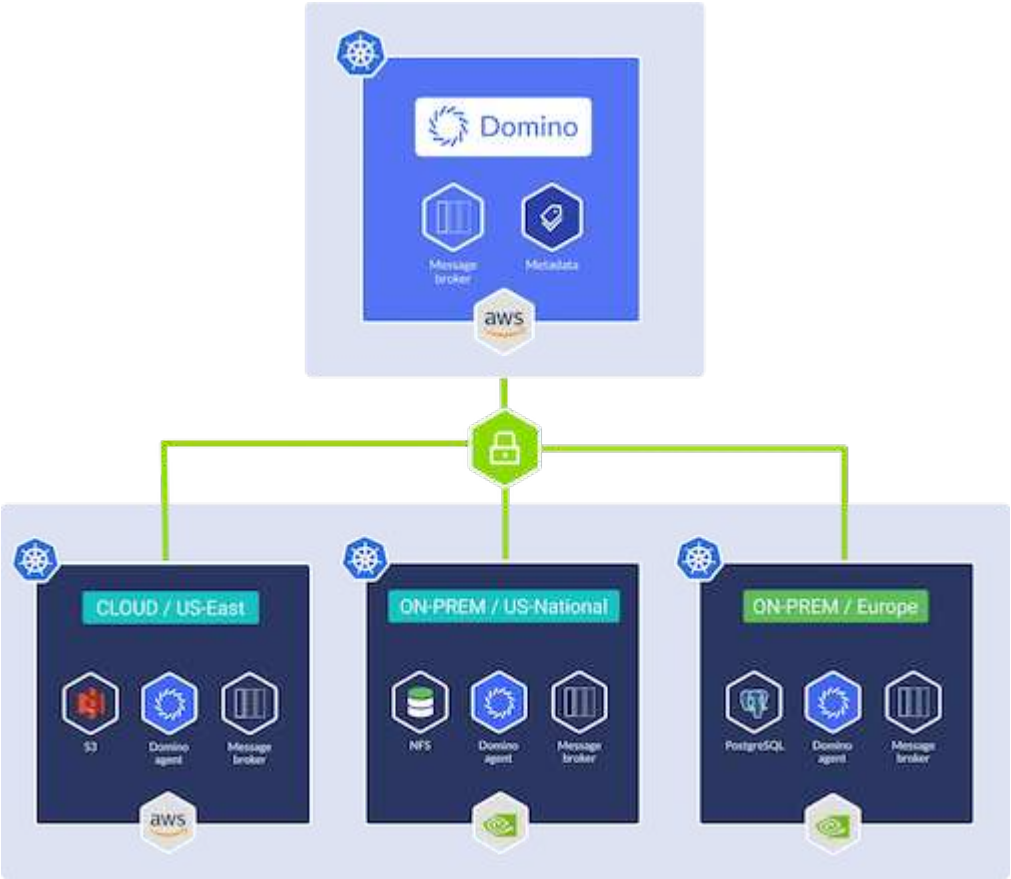
### Domino数据实验室

Domino Data Lab凭借其领先的企业级AI平台为模型驱动型企业提供支持、该平台受到超过20%的财富100强企业的信赖。Domino可加快数据科学工作的开发和部署速度、同时增强协作和监管。有了Domino、全世界的企业可以开发更好的药物、种植更具生产力的作物、建造更好的汽车等等。Domino成立于2013年、由Coatue Management、Great Hill Partners、高地资本、Sequoia Capital和其他主要投资者提供支持。

Domino支持企业及其数据科学家在一个统一的端到端平台上快速、负责任且经济高效地构建、部署和管理AI。团队可以在任何环境中访问所需的所有数据、工具、计算、模型和项目、因此他们可以进行协作、重复利用过去的工作、跟踪生产中的模型以提高准确性、采用最佳实践进行标准化、以及让AI成为负责任和受监管的企业。

- *开放且灵活：*访问最广泛的开放源代码和商业工具及基础架构生态系统，获得最佳创新，不受制于供应商。

- *记录系统：*整个企业的人工智能运营和知识中心、支持最佳实践、跨职能协作、加快创新速度和提高效率。
- *集成：*集成工作流和自动化—专为企业流程、控制和监管而构建—可满足您的合规性和法规要求。
- *混合多云：*在靠近数据的位置运行AI工作负载—内部环境、混合环境、任何云或多云—以降低成本、优化性能和合规性。




### Domino Nexus

Domino Nexus是一个单一管理平台、支持您跨任何云、区域或内部环境中的任何计算集群运行数据科学和机器学习工作负载。它统一了整个企业中的数据科学孤岛、让您有一个地方来构建、部署和监控模型。

### NetApp BlueXP

NetApp BlueXP将NetApp的所有存储和数据服务统一到一个工具中、让您可以构建、保护和管理混合多云数据资产。它可以跨内部环境和云环境为存储和数据服务提供统一的体验、并通过AIIOPS的强大功能实现运营精简性、同时还具有当今云主导环境所需的灵活使用参数和集成保护。

### NetApp ONTAP

ONTAP 9是NetApp推出的最新一代存储管理软件、可帮助企业打造现代化的基础架构并过渡到云就绪数据中心。借助行业领先的数据管理功能，无论数据位于何处， ONTAP 都可以通过一组工具来管理和保护数据。您还可以将数据自由移动到需要的任何位置：边缘，核心或云。ONTAP 9包含许多功能、可简化数据管理、加快和保护关键数据、并在混合云架构中实现下一代基础架构功能。

#### 简化数据管理

数据管理对于企业IT运营和数据科学家至关重要、这样才能将适当的资源用于AI应用程序和训练AI/ML数据集。以下有关NetApp技术的追加信息 不在此验证范围内、但可能与您的部署相关。

ONTAP 数据管理软件包括以下功能、可简化操作并降低总运营成本：

- 实时数据缩减和扩展的重复数据删除。数据缩减可减少存储块中浪费的空间、重复数据删除可显著提高有效容量。此适用场景数据存储在本地，并分层到云。
- 最低、最高和自适应服务质量(AQoS)。精细的服务质量(QoS)控制有助于在高度共享的环境中保持关键应用程序的性能水平。
- NetApp FabricPool。可将冷数据自动分层到公有 和私有云存储选项、包括Amazon Web Services (AWS)、Azure和NetApp StorageGRID Storage解决方案。有关 FabricPool 的详细信息，请参见 "TR-4598：FabricPool 最佳实践"。

#### 加速和保护数据

ONTAP 可提供卓越的性能和数据保护、并通过以下方式扩展这些功能：

- 性能和更低的延迟。ONTAP 可提供尽可能高的吞吐量和尽可能低的延迟。
- 数据保护ONTAP 可提供内置数据保护功能、并在所有平台之间进行通用管理。
- NetApp卷加密(NVE)。ONTAP 提供原生 卷级加密、并支持板载和外部密钥管理。
- 多租户和多因素身份验证。ONTAP 支持以最高的安全性级别共享基础架构资源。

#### Future-Proof 基础架构

ONTAP 可通过以下功能满足不断变化的苛刻业务需求：

- 无缝扩展和无中断运行。ONTAP 支持无中断地向现有控制器和横向扩展集群添加容量。客户可以升级到 NVMe 和 32 Gb FC 等最新技术，而无需进行成本高昂的数据迁移或中断。
- 云连接。ONTAP是云互联程度最高的存储管理软件、可在所有公有云中选择软件定义的存储和云原生实例。
- 与新兴应用程序集成。ONTAP 通过使用支持现有企业应用程序的相同基础架构、为下一代平台和应用程序(例如自动驾驶汽车、智能城市和行业4.0)提供企业级数据服务。

### 适用于 **NetApp ONTAP** 的 **Amazon FSX**

Amazon FSx for NetApp ONTAP是第一方完全托管的AWS服务、可提供基于NetApp流行的ONTAP文件系统构建的高度可靠、可扩展、高性能和功能丰富的文件存储。FSX for ONTAP 将NetApp文件系统的常见特性、性能、功能和API操作与完全托管的AWS服务的灵活性、可扩展性和精简性相结合。

### NetApp Astra Trident

Asta Trident支持在公有云或内部环境中的所有常见NetApp存储平台上使用和管理存储资源、包括ONTAP (AFF、FAS、Select、云、 Amazon FSx for NetApp ONTAP)、Element软件(NetApp HCI、SolidFire)、Azure NetApp Files服务以及Google Cloud上的Cloud Volumes Service。Asta Trident是一款符合容器存储接口(CSI)的动态存储编排程序、可与Kubbernetes本机集成。

### Kubernetes

Kubernetes 是一款开源分布式容器编排平台，最初由 Google 设计，现在由 Cloud 原生计算基金会（ CNCF ）维护。Kubnetes支持容器化应用程序的部署、管理和扩展功能自动化、是企业环境中的主要容器流程编排平台。

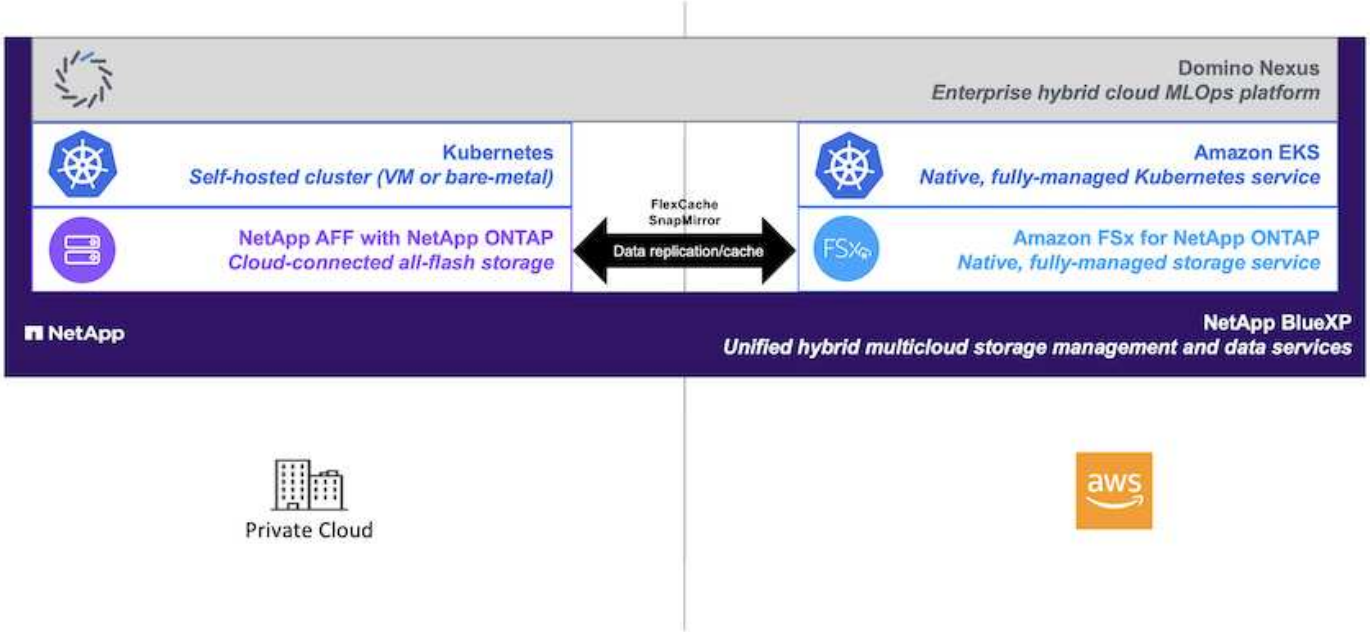
### Amazon Elelic Kubelnetes Service (EKS)

Amazon El生 性Kubelnetes Service (Amazon EKS)是AWS云中的托管Kubelnetes服务。Amazon EKS会自动管理Kubersnetes控制平台节点的可用性和可扩展性、这些节点负责计划容器、管理应用程序可用性、存储集群数据以及其他关键任务。借助Amazon EKS、您可以利用AWS基础架构的所有性能、扩展性、可靠性和可用性、以及与AWS网络和安全服务的集成。

## 架构

此解决方案将Domino Nexus的混合多云工作负载计划功能与NetApp数据服务相结合、创建统一的混合云MLOps平台。有关详细信息、请参见下表。

| 组件 | **Name** | **environment** |
|---|---|---|
| MLOps控制平台 | "采用Domino Nexus的Domino企业级AI平台" | AWS |
| MLOps平台计算环境 | "Domino Nexus数据平面" | AWS、内部数据中心 |
| 内部计算平台 | "Kubernetes" 使用 "NetApp Astra Trident" | 内部数据中心 |
| 云计算平台 | "Amazon Elelic Kubelnetes Service (EKS)" 使用 "NetApp Astra Trident" | AWS |
| 内部数据平台 | "NetApp存储设备" 由提供支持 "NetApp ONTAP" | 内部数据中心 |
| 云数据平台 | "适用于 NetApp ONTAP 的 Amazon FSX" | AWS |




## 初始设置

本节介绍在整合内部数据中心和AWS的混合环境中将Domino Nexus与NetApp数据服务结合使用所需执行的初始设置任务。

### 前提条件

在执行本节所述的步骤之前、我们假定您已执行以下任务：

- 您已部署和配置内部NetApp ONTAP存储平台。有关详细信息，请参见 "NetApp 产品文档"。
- 您已在AWS中配置Amazon FSx for NetApp ONTAP实例。有关详细信息，请参见 "Amazon FSx for NetApp ONTAP商品页面"。
- 您已在内部数据中心中配置了Kubbernetes集群。有关详细信息，请参见 "Domino管理指南"。
- 您已在AWS中配置Amazon EKS集群。有关详细信息，请参见 "Domino管理指南"。
- 您已在内部部署的Kubbernetes集群中安装NetApp Asta三端存储。此外、您还配置了此三项技术实例、以便在配置和管理存储资源时使用内部NetApp ONTAP存储平台。有关详细信息，请参见 "NetApp Astra Trident文档"。
- 您已在Amazon EKS集群中安装NetApp Asta三端磁盘。此外、您还配置了此TRIDent实例、以便在配置和管理存储资源时使用Amazon FSx for NetApp ONTAP实例。有关详细信息，请参见 "NetApp Astra Trident 文档"。
- 您必须在内部数据中心和AWS中的虚拟私有云(Virtual Private Cloud、VPC)之间建立双向网络连接。有关实施此功能的各种选项的更多详细信息、请参见 "Amazon虚拟专用网络(VPN)文档"。

### 在AWS中安装Domino Enterprise AI Platform

要在AWS中安装Domino Enterprise MLOps平台、请按照中所述的说明进行操作 "Domino管理指南"。您必须在先前配置的同一个Amazon EKS集群中部署Domino。此外、必须已在此EKS集群中安装和配置NetApp Asta三端磁盘、并且必须在Domino/yml安装配置文件中指定一个三端磁盘管理的存储类作为共享存储类。

请参见 "Domino安装配置参考指南" 有关如何在Domino/yml安装配置文件中指定共享存储类的详细信息。

"技术报告TR-4952" 介绍如何使用Amazon FSx for NetApp ONTAP在AWS中部署Domino、对于解决出现的任何问题、这可能是一个有用的参考。

#### 启用Domino Nexus

接下来、您必须启用Domino Nexus。请参见 "Domino管理指南" 了解详细信息。

#### 在内部数据中心部署Domino数据平面

接下来、您必须在内部数据中心部署Domino数据平面。您必须将此数据平面部署在先前配置的内部Kubbernetes集群中。此外、必须已在此Kubbernetes集群中安装和配置NetApp Asta三端存储。请参见 "Domino管理指南" 了解详细信息。

## 将现有NetApp卷公开给Domino

本节介绍向Domino MLOps平台公开现有NetApp ONTAP NFS卷所需执行的任务。这些步骤同样适用于内部和AWS。

#### 为什么要将NetApp ONTAP卷公开给Domino?

将NetApp卷与Domino结合使用具有以下优势：

- 您可以利用NetApp ONTAP的横向扩展功能对超大型数据集执行工作负载。
- 您可以跨多个计算节点执行工作负载、而无需将数据复制到各个节点。
- 您可以利用NetApp的混合多云数据移动和同步功能跨多个数据中心和/或云访问数据。
- 您希望能够在其他数据中心或云中快速轻松地创建数据缓存。

#### 公开Asta Trident未配置的现有NFS卷

如果现有的NetApp ONTAP NFS卷不是由Astra Trident配置的、请按照本小节中概述的步骤进行操作。

##### 在Kubbernetes中创建PV和PVC

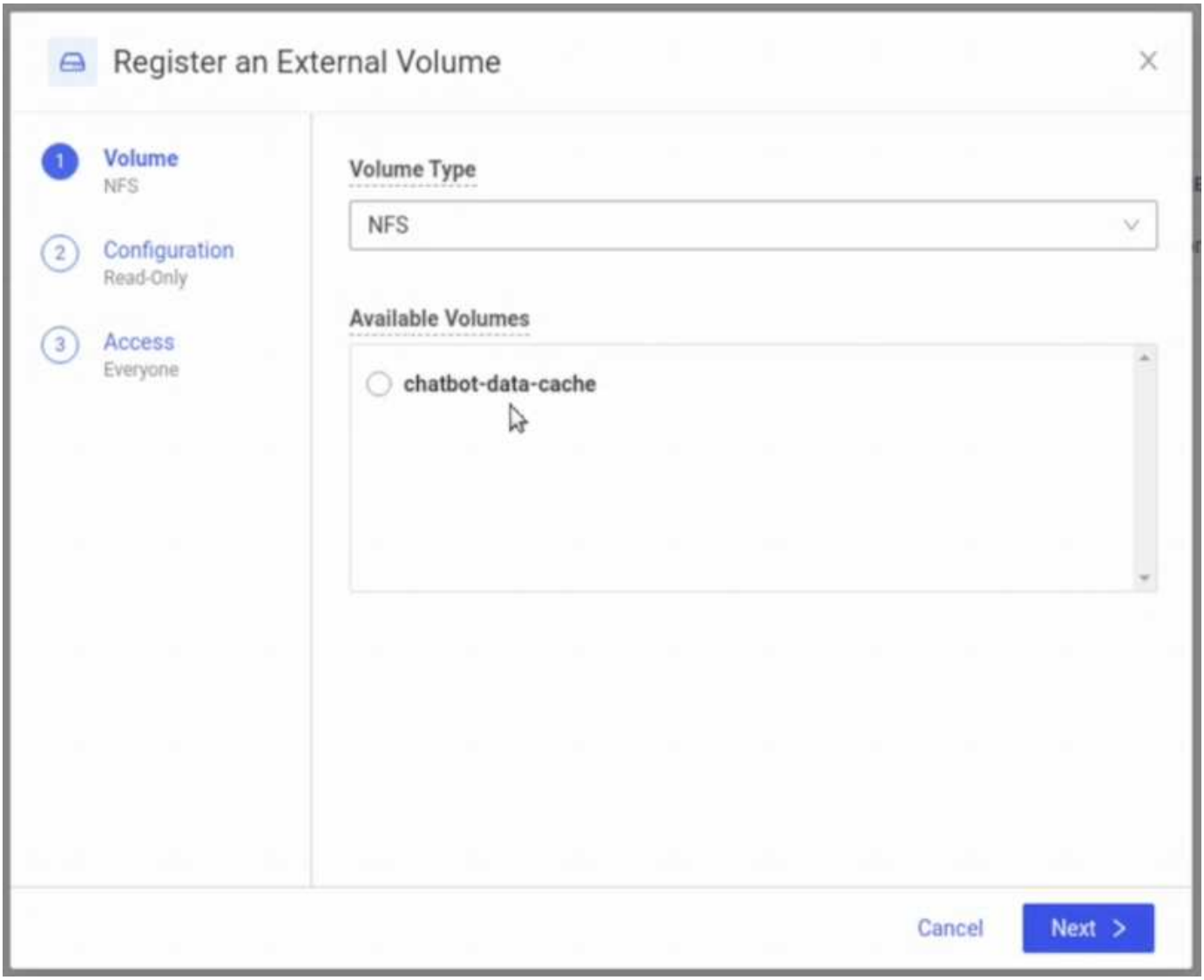
对于内部部署卷、请在内部Kubbernetes集群中创建PV和PVC。对于Amazon FSx for NetApp ONTAP卷、在Amazon EKS中创建PV和PVC。

首先、您必须在Kubbernetes集群中创建永久性卷(PV)和永久性卷请求(PVC)。要创建PV和PVC、请使用 "NFS PV/PVC示例" 并更新这些值以反映您的环境。请务必为指定正确的值 `namespace`， `nfs.path`，和 `nfs.server` 字段。此外、我们建议为您的PV和PVC提供唯一名称、以表示相应ONTAP NFS卷上存储的数据的性质。例如、如果卷包含制造缺陷的图像、您可以将PV命名为 `pv-mfg-defect-images`和PVC、 `pvc-mfg-defect-images`。

##### 在Domino中注册外部数据卷

接下来、您必须在Domino中注册外部数据卷。要注册外部数据卷、请参见 "说明" 在Domino管理指南中。注册

卷时、请务必从"卷类型"下拉菜单中选择"NFS"。选择"NFS"后、您应在"可用卷"列表中看到您的PVC。




公开**Asta Trident**配置的现有卷

如果现有卷是由Asta Trident配置的、请按照本小节中概述的步骤进行操作。

#### 编辑现有**PVC**

如果您的卷是由Asta Trident配置的、则您已经拥有与您的卷对应的永久性卷请求(PVC)。要将此卷公开给Domino、必须编辑PVC并将以下标签添加到中的标签列表中 `metadata.labels` 字段：

```
"dominodatalab.com/external-data-volume": "Generic"
```

#### 在**Domino**中注册外部数据卷

接下来、您必须在Domino中注册外部数据卷。要注册外部数据卷、请参见 "说明" 在Domino管理指南中。注册卷时、请务必从"卷类型"下拉菜单中选择"通用"。选择"通用"后、您应在"可用卷"列表中看到您的PVC。

## 在不同环境中访问相同的数据

本节介绍在不同计算环境中访问相同数据所需执行的任务。在Domino MLOps平台中、计算环境称为"数据平面"。 如果您的数据驻留在一个数据平面中的NetApp卷上、但您需要在另一个数据平面中访问该数据、请按照本节中所述的任务进行操作。这种情形通常称为"突发"、如果目标环境为云、则称为"云突发"。 处理有限或超额预订的计算资源时、通常需要此功能。例如、如果您的内部计算集群订阅过量、您可能希望将工作负载计划到云、以便立即启动。

对于访问位于不同数据平面中的NetApp卷、建议使用两种方法。这些选项将在下面的小节中进行概述。根据您的特定要求、选择其中一个选项。下表介绍了这两个选项的优点和缺点。

| 选项 | 优势 | 缺点 |
| --- | --- | --- |
| 选项1 -缓存 | -工作流更简单<br>-能够根据需要缓存一部分数据<br>-能够将数据写回源<br>-没有要管理的远程副本 | -随着缓存水合、初始数据访问延迟会增加。 |
| 选项2 -镜像 | -源卷的完整副本<br>-缓存融合不会增加延迟(镜像操作完成后) | -必须等待镜像操作完成、然后才能访问数据<br>-必须管理远程副本<br>-无法回写源 |

### 选项**1** -为驻留在其他数据平面中的卷创建缓存

使用 "NetApp FlexCache 技术"，则可以为驻留在其他数据平面中的NetApp卷创建缓存。例如、如果您的内部数据平面中有一个NetApp卷、而您需要在AWS数据平面中访问该卷、则可以在AWS中为该卷创建缓存。本节概述了为驻留在其他数据平面中的NetApp卷创建缓存所需执行的任务。

#### 在目标环境中创建**FlexCache**卷

> 如果目标环境是您的内部数据中心、则需要在内部ONTAP系统上创建FlexCache卷。如果目标环境为AWS、则需要在Amazon FSx for NetApp ONTAP实例上创建FlexCache卷。

首先、必须在目标环境中创建FlexCache卷。

建议使用BlueXP创建FlexCache卷。要使用BlueXP创建FlexCache卷、请按照中所述的说明进行操作 "BlueXP卷缓存文档"。

如果您不想使用BlueXP、则可以使用ONTAP系统管理器或ONTAP命令行界面创建FlexCache卷。要使用System Manager创建FlexCache卷、请参阅中概述的说明 "ONTAP 文档"。要使用ONTAP命令行界面创建FlexCache卷、请参阅中概述的说明 "ONTAP 文档"。

如果要自动执行此过程、可以使用 "BlueXP API"， "ONTAP REST API"或 "ONTAP的"Ans征选""。

> System Manager在Amazon FSx for NetApp ONTAP中不可用。

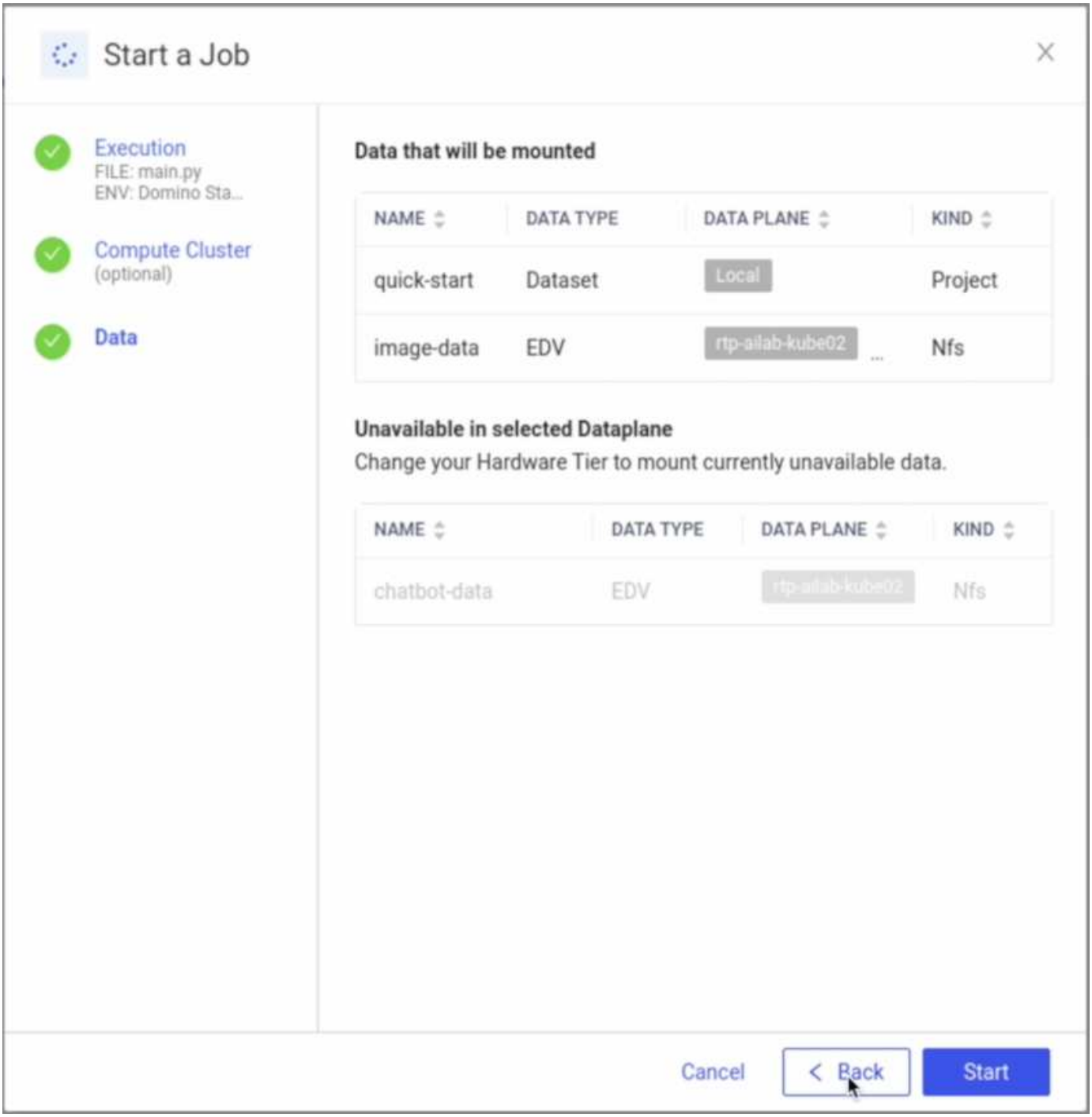
#### 将**FlexCache**卷公开给**Domino**

接下来、必须将FlexCache卷公开给Domino MLOps平台。要向Domino公开FlexCache卷、请按照的"公开未

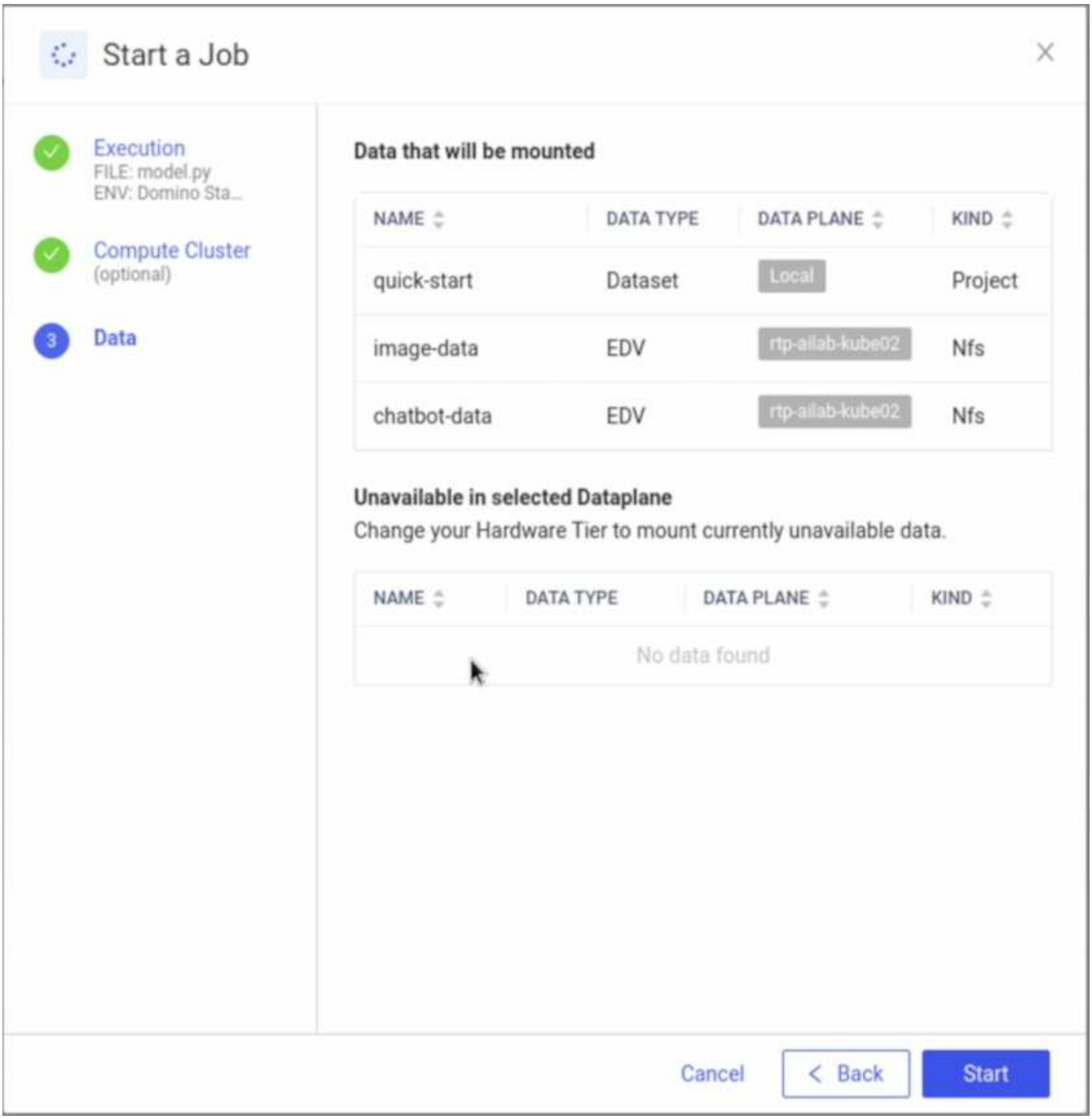
由Astra Trident配置的现有NFS卷"子部分中所述的说明进行操作 ""向Domino公开现有NetApp卷"部分" 解决方案。

现在、您可以在目标数据平面中启动作业和工作空间时挂载FlexCache卷、如以下屏幕截图所示。

创建**FlexCache**卷之前




将**FlexCache**卷公开给**Domino**之后




选项**2 -**备份驻留在其他数据平面中的卷

使用 "NetApp SnapMirror数据复制技术"，则可以创建驻留在其他数据平面中的NetApp卷的副本。例如、如果您的内部数据平面中有一个NetApp卷、而您需要在AWS数据平面中访问该卷、则可以在AWS中创建该卷的副本。本节概述了为驻留在其他数据平面中的NetApp卷创建副本所需执行的任务。

创建 **SnapMirror** 关系

首先、必须在源卷与目标环境中的新目标卷之间创建SnapMirror关系。请注意、目标卷将在创建SnapMirror关系的过程中创建。

建议使用BlueXP创建SnapMirror关系。要使用BlueXP创建SnapMirror关系、请按照中所述的说明进行操作

"BlueXP复制文档"。

如果您不想使用BlueXP、则可以使用ONTAP系统管理器或ONTAP命令行界面创建SnapMirror关系。要创建与System Manager的SnapMirror关系、请参阅中概述的说明 "ONTAP 文档"。要使用ONTAP命令行界面创建SnapMirror关系、请参阅中概述的说明 "ONTAP 文档"。

如果要自动执行此过程、可以使用 "BlueXP API"， "ONTAP REST API"或 "ONTAP的"Ans征选""。

System Manager在Amazon FSx for NetApp ONTAP中不可用。

#### 中断 **SnapMirror** 关系

接下来、您必须中断SnapMirror关系、才能激活目标卷以进行数据访问。请等待初始复制完成、然后再执行此步骤。

您可以通过在BlueXP、ONTAP系统管理器或ONTAP命令行界面中检查镜像状态来确定复制是否已完成。复制完成后、镜像状态将为"snapMirrored"。

建议使用BlueXP中断SnapMirror关系。要中断与BlueXP的SnapMirror关系、请按照中所述的说明进行操作 "BlueXP复制文档"。

如果您不想使用BlueXP、则可以使用ONTAP系统管理器或ONTAP命令行界面中断SnapMirror关系。要中断与System Manager的SnapMirror关系、请参阅中概述的说明 "ONTAP 文档"。要中断与ONTAP命令行界面的SnapMirror关系、请参阅中概述的说明 "ONTAP 文档"。

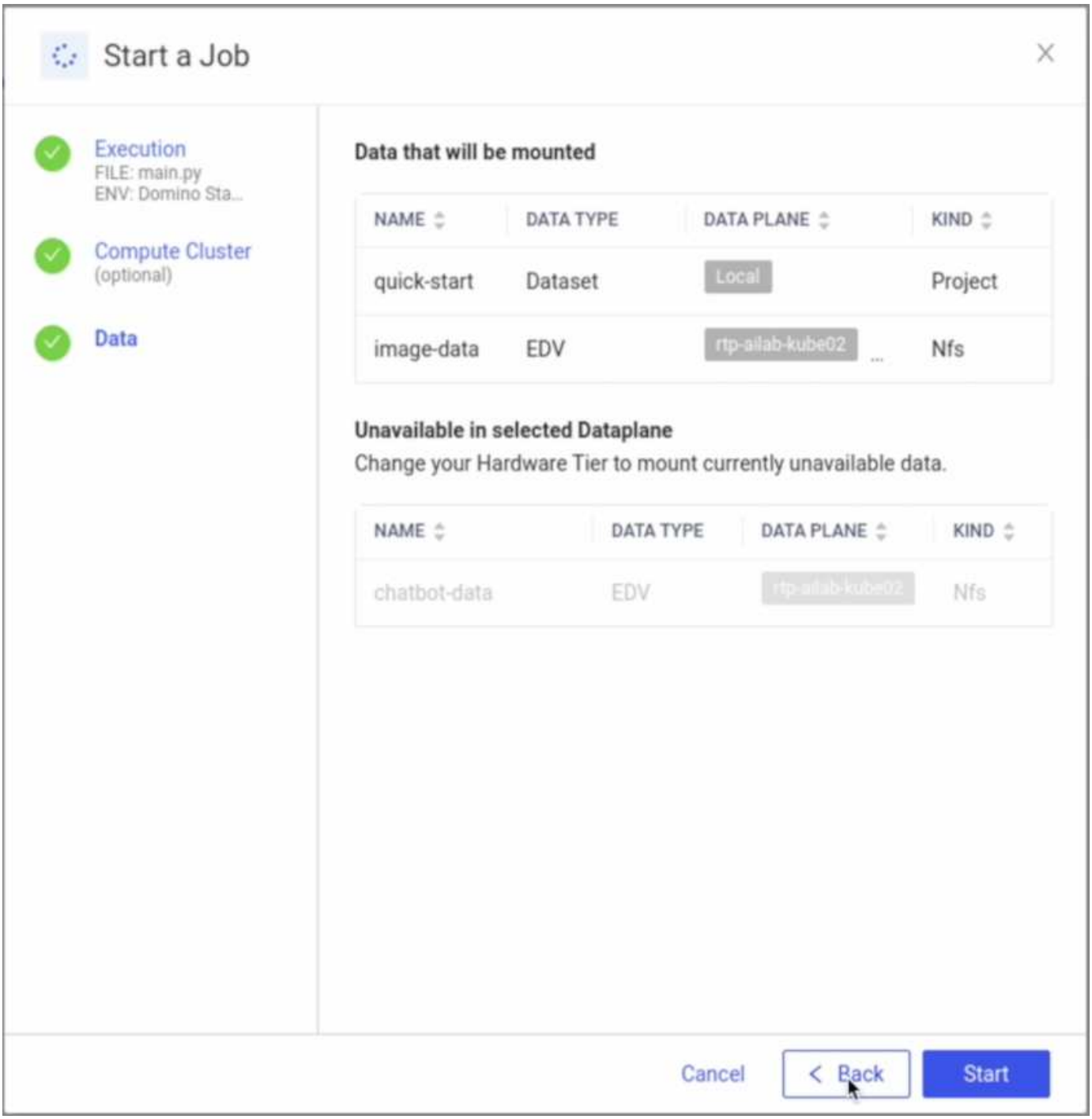
如果要自动执行此过程、可以使用 "BlueXP API"， "ONTAP REST API"或 "ONTAP的"Ans征选""。

#### 向**Domino**公开目标卷

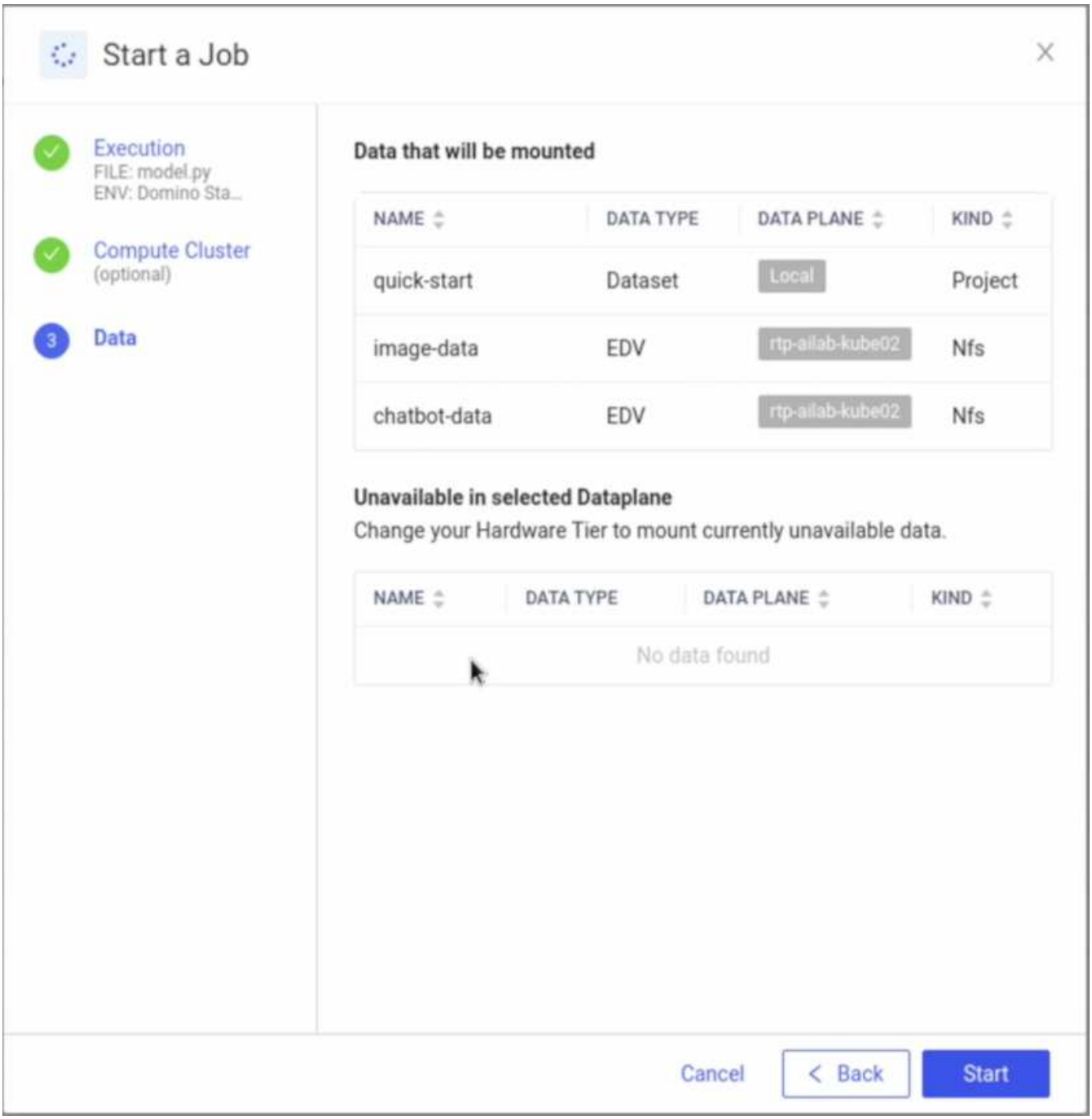
接下来、您必须将目标卷公开给Domino MLOps平台。要向Domino公开目标卷、请按照的"公开未由Astra Trident配置的现有NFS卷"子部分中所述的说明进行操作 ""向Domino公开现有NetApp卷"部分" 解决方案。

现在、您可以在目标数据平面中启动作业和工作空间时挂载目标卷、如以下屏幕截图所示。

创建**SnapMirror**关系之前




向**Domino**公开目标卷之后




## 从何处查找追加信息

要了解有关本文档中所述信息的更多信息，请参见以下文档和 / 或网站：

- Domino数据实验室

  "https://domino.ai"

- Domino Nexus

  "https://domino.ai/platform/nexus"

- NetApp BlueXP

  "https://bluexp.netapp.com"

- NetApp ONTAP 数据管理软件

  "https://www.netapp.com/data-management/ontap-data-management-software/"

- NetApp AI解决方案

  "https://www.netapp.com/artificial-intelligence/"

致谢

- Domino Data Lab技术联盟SA主管Jsh Mineroff
- Domino Data Lab现场首席技术官Nicolas Jablonski
- NetApp公司解决方案架构师Arjunan先生
- NetApp技术联盟合作伙伴全球联盟总监Brian Young

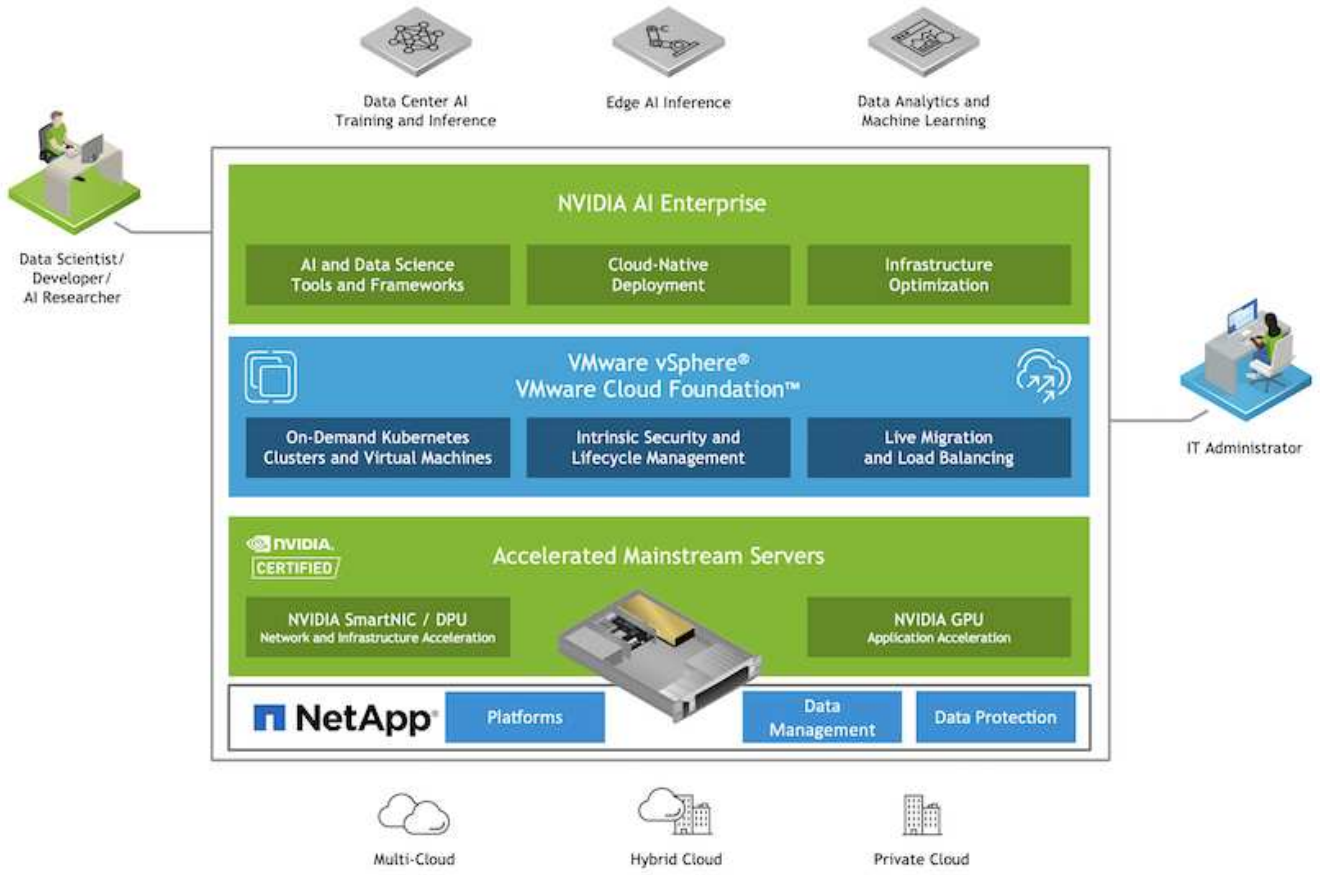
# 采用NetApp和VMware的NVIDIA AI Enterprise

## 采用NetApp和VMware的NVIDIA AI Enterprise

NetApp 公司 Mike Oglesby

对于IT架构师和管理员来说、AI工具可能非常复杂、而且不熟悉。此外、许多AI平台还没有为企业做好准备。由NetApp和VMware提供支持的NVIDIA AI Enterprise旨在提供简化的企业级AI架构。

NVIDIA AI Enterprise是一款端到端云原生AI和数据分析软件套件、经过NVIDIA优化、认证和支持、可在采用NVIDIA认证系统的VMware vSphere上运行。此软件有助于在现代混合云环境中轻松快速地部署、管理和扩展AI工作负载。由NetApp和VMware提供支持的NVIDIA AI Enterprise通过一个简单熟悉的软件包提供企业级AI工作负载和数据管理。




## 技术概述

### NVIDIA AI Enterprise

NVIDIA AI Enterprise是一款端到端云原生AI和数据分析软件套件、经过NVIDIA优化、认证和支持、可在采用NVIDIA认证系统的VMware vSphere上运行。此软件有助于在现代混合云环境中轻松快速地部署、管理和扩展AI工作负载。

### NVIDIA GPU Cloud （NGC）

NVIDIA NGC提供了一个GPU优化软件目录、供AI从业者开发其AI解决方案。此外、还可以访问各种AI服务、包括用于模型培训的NVIDIA Base Command、用于部署和监控模型的NVIDIA Baset Command以及用于安全访问和管理专有AI软件的NGC私有注册表。此外、NVIDIA AI Enterprise客户还可以通过NGC门户申请支持。

### VMware vSphere

VMware vSphere是VMware的虚拟化平台、可将数据中心转变为包括CPU、存储和网络资源在内的聚合计算基础架构。vSphere将这些基础架构作为一个统一的操作环境进行管理、并为管理员提供用于管理参与该环境的数据中心的工具。

vSphere的两个核心组件是ESXi和vCenter Server。ESXi是一个虚拟化平台、管理员可以在此平台上创建和运行虚拟机和虚拟设备。vCenter Server是一项服务、管理员可以通过此服务管理连接到网络和池主机资源的多个主机。

### NetApp ONTAP

ONTAP 9是NetApp推出的最新一代存储管理软件、可帮助企业打造现代化的基础架构并过渡到云就绪数据中心。借助行业领先的数据管理功能，无论数据位于何处， ONTAP 都可以通过一组工具来管理和保护数据。您还可以将数据自由移动到需要的任何位置：边缘，核心或云。ONTAP 9包含许多功能、可简化数据管理、加快和保护关键数据、并在混合云架构中实现下一代基础架构功能。

#### 简化数据管理

数据管理对于企业IT运营和数据科学家至关重要、这样才能将适当的资源用于AI应用程序和训练AI/ML数据集。以下有关NetApp技术的追加信息 不在此验证范围内、但可能与您的部署相关。

ONTAP 数据管理软件包括以下功能、可简化操作并降低总运营成本：

- 实时数据缩减和扩展的重复数据删除。数据缩减可减少存储块中浪费的空间、重复数据删除可显著提高有效容量。此适用场景数据存储在本地，并分层到云。
- 最低、最高和自适应服务质量(AQoS)。精细的服务质量(QoS)控制有助于在高度共享的环境中保持关键应用程序的性能水平。
- NetApp FabricPool。可将冷数据自动分层到公有 和私有云存储选项、包括Amazon Web Services (AWS)、Azure和NetApp StorageGRID Storage解决方案。有关 FabricPool 的详细信息，请参见 "TR-4598：FabricPool 最佳实践"。

#### 加速和保护数据

ONTAP 可提供卓越的性能和数据保护、并通过以下方式扩展这些功能：

- 性能和更低的延迟。ONTAP 可提供尽可能高的吞吐量和尽可能低的延迟。
- 数据保护ONTAP 可提供内置数据保护功能、并在所有平台之间进行通用管理。
- NetApp卷加密(NVE)。ONTAP 提供原生 卷级加密、并支持板载和外部密钥管理。
- 多租户和多因素身份验证。ONTAP 支持以最高的安全性级别共享基础架构资源。

#### Future-Proof 基础架构

ONTAP 可通过以下功能满足不断变化的苛刻业务需求：

- 无缝扩展和无中断运行。ONTAP 支持无中断地向现有控制器和横向扩展集群添加容量。客户可以升级到 NVMe 和 32 Gb FC 等最新技术，而无需进行成本高昂的数据迁移或中断。
- 云连接。ONTAP 是云互联程度最高的存储管理软件、可在所有公有 云中选择软件定义的存储(ONTAP Select)和云原生实例(NetApp Cloud Volumes Service)。
- 与新兴应用程序集成。ONTAP 通过使用支持现有企业应用程序的相同基础架构、为下一代平台和应用程序(例如自动驾驶汽车、智能城市和行业4.0)提供企业级数据服务。

### NetApp DataOps 工具包

NetApp DataOps工具包是一款基于Python的工具、可简化开发/培训工作空间和推理服务器的管理、这些工作空间和服务器由高性能横向扩展NetApp存储提供支持。主要功能包括：

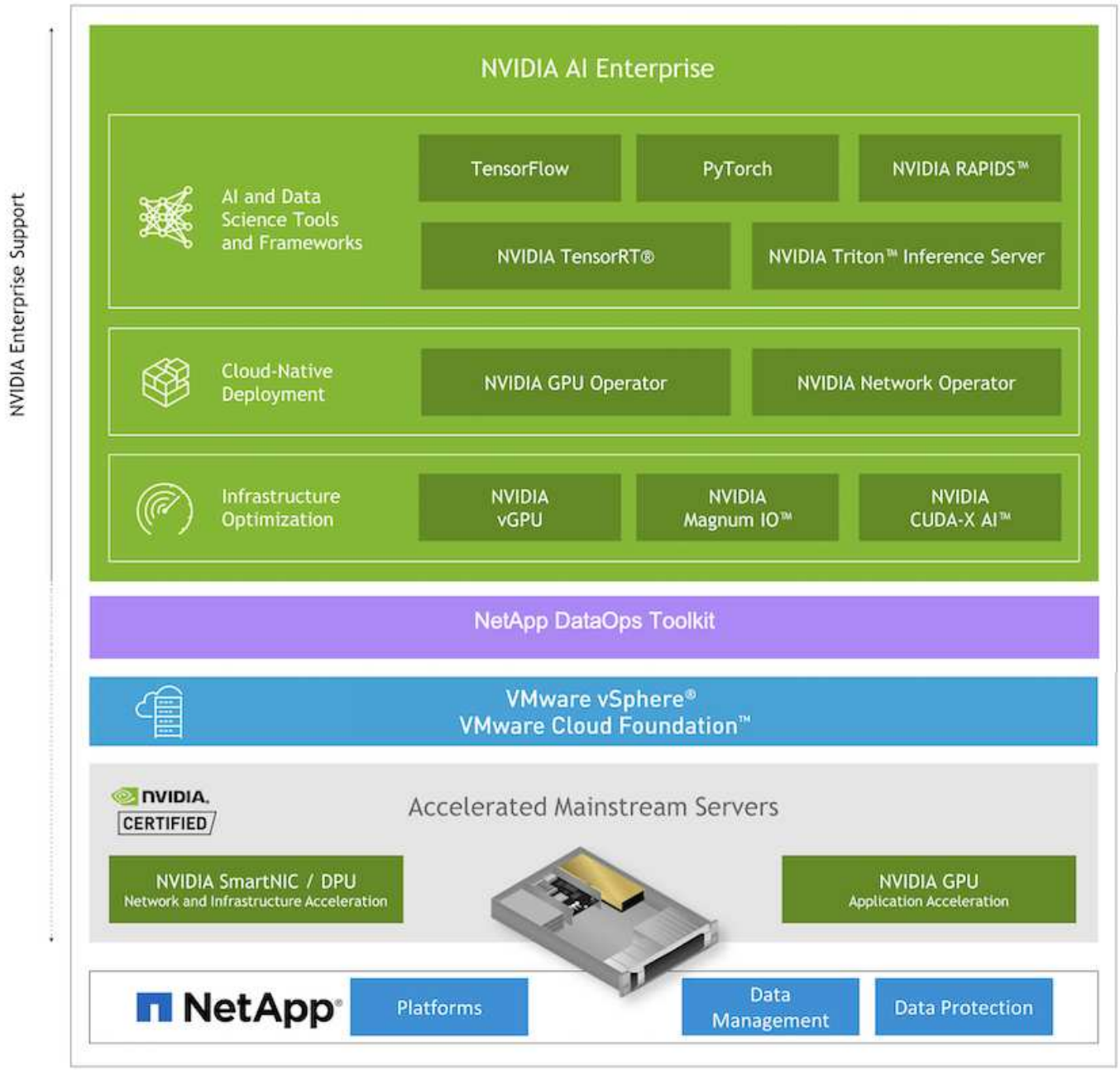
- 快速配置新的高容量JupyterLab工作空间、这些工作空间以高性能横向扩展NetApp存储为后盾。
- 快速配置由企业级NetApp存储提供支持的新NVIDIA Triton推理服务器实例。

- 可近乎即时地克隆高容量JupyterLab工作空间、以便进行实验或快速迭代。
- 可近乎即时地保存高容量JupyterLab工作空间的快照、以实现备份和/或可追溯性/基线化。
- 近乎即时地配置、克隆和快照高容量、高性能数据卷。

## 架构

此解决方案 基于经验证且熟悉的架构构建、该架构采用NetApp、VMware和NVIDIA认证系统。有关详细信息、请参见下表。

| 组件 | 详细信息 |
| --- | --- |
| AI和数据分析软件 | "适用于VMware的NVIDIA AI Enterprise" |
| 虚拟化平台 | "VMware vSphere" |
| 计算平台 | "NVIDIA认证系统" |
| 数据管理平台 | "NetApp ONTAP" |




## 初始设置

本节介绍在NetApp和VMware中使用NVIDIA AI Enterprise时需要执行的初始设置任务。

### 前提条件

在执行本节所述的步骤之前、我们假定您已部署VMware vSphere和NetApp ONTAP。请参见 "NVIDIA AI企业产品支持表" 有关受支持的vSphere版本的详细信息。请参见 "NetApp和VMware解决方案 文档" 有关使用NetApp ONTAP 部署VMware vSphere的详细信息。

### 安装**NVIDIA AI Enterprise Host**软件

要安装NVIDIA AI Entrprise主机软件、请按照中第1-4节所述的说明进行操作 "NVIDIA AI Enterprise快速入门指南"。

## 使用NVIDIA NGC软件

本节介绍在NVIDIA AI Enterprise环境中使用NVIDIA NGC企业软件需要执行的任务。

### 设置

本节介绍在NVIDIA AI Enterprise环境中使用NVIDIA NGC企业软件所需执行的初始设置任务。

#### 前提条件

在执行本节所述的步骤之前、我们假定您已按照中所述的说明部署NVIDIA AI Entrprise主机软件 "初始设置" 页面。

#### 使用vGPU创建Ubuntu子虚拟机

首先、您必须使用vGPU创建Ubuntu 20.04子虚拟机。要使用vGPU创建Ubuntu 20.04子虚拟机、请按照中概述的说明进行操作 "NVIDIA AI Enterprise部署指南"。

#### 下载并安装NVIDIA子软件

接下来、您必须在上一步创建的子虚拟机中安装所需的NVIDIA子系统软件。要在子虚拟机中下载并安装所需的NVIDIA子软件、请按照中第5.1-5.4节所述的说明进行操作 "NVIDIA AI Enterprise快速入门指南"。

> 在执行第5.4节所述的验证任务时、您可能需要使用不同的CUDA容器映像版本标记、因为自编写本指南以来、CUDA容器映像已进行了更新。在我们的验证中、我们使用了"NVIDIA/CUDA：11.0.3-base-ubuntu20.04"。

#### 下载AI/分析框架容器

接下来、您必须从NVIDIA NGC下载所需的AI或分析框架容器映像、以便它们可以在子虚拟机中使用。要在子虚拟机中下载框架容器、请按照中所述的说明进行操作 "NVIDIA AI Enterprise部署指南"。

#### 安装和配置NetApp DataOps工具包

接下来、您必须在子虚拟机中安装适用于传统环境的NetApp DataOps工具包。NetApp DataOps工具包可用于直接从子虚拟机中的终端管理ONTAP 系统上的横向扩展数据卷。要在子虚拟机中安装NetApp DataOps工具包、请执行以下任务。

1. 安装pip。

```
$ sudo apt update
$ sudo apt install python3-pip
$ python3 -m pip install netapp-dataops-traditional
```

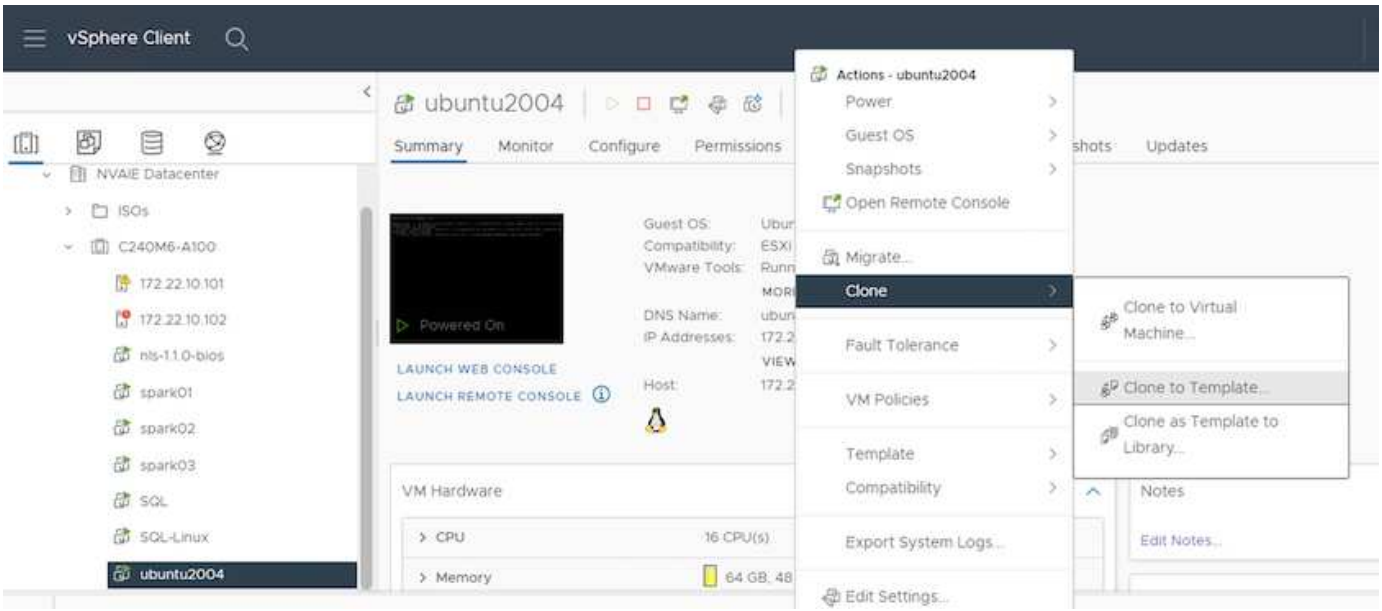
2. 从子虚拟机终端中注销、然后重新登录。
3. 配置NetApp DataOps工具包。要完成此步骤、您需要有关ONTAP 系统的API访问详细信息。您可能需要从存储管理员处获取这些信息。

```
$ netapp_dataops_cli.py config

Enter ONTAP management LIF hostname or IP address (Recommendation: Use
SVM management interface): 172.22.10.10
Enter SVM (Storage VM) name: NVAIE-client
Enter SVM NFS data LIF hostname or IP address: 172.22.13.151
Enter default volume type to use when creating new volumes
(flexgroup/flexvol) [flexgroup]:
Enter export policy to use by default when creating new volumes
[default]:
Enter snapshot policy to use by default when creating new volumes
[none]:
Enter unix filesystem user id (uid) to apply by default when creating
new volumes (ex. '0' for root user) [0]:
Enter unix filesystem group id (gid) to apply by default when creating
new volumes (ex. '0' for root group) [0]:
Enter unix filesystem permissions to apply by default when creating new
volumes (ex. '0777' for full read/write permissions for all users and
groups) [0777]:
Enter aggregate to use by default when creating new FlexVol volumes:
aff_a400_01_NVME_SSD_1
Enter ONTAP API username (Recommendation: Use SVM account): admin
Enter ONTAP API password (Recommendation: Use SVM account):
Verify SSL certificate when calling ONTAP API (true/false): false
Do you intend to use this toolkit to trigger BlueXP Copy and Sync
operations? (yes/no): no
Do you intend to use this toolkit to push/pull from S3? (yes/no): no
Created config file: '/home/user/.netapp_dataops/config.json'.
```

创建子虚拟机模板

最后、您必须根据子虚拟机创建VM模板。您可以使用此模板快速创建子虚拟机、以便使用NVIDIA NGC软件。

要基于来宾VM创建VM模板、请登录到VMware vSphere、然后右键单击来宾VM名称、选择"克隆"、选择"克隆到模板…"、然后按照向导进行操作。




示例用例**—TensorFlow**培训作业

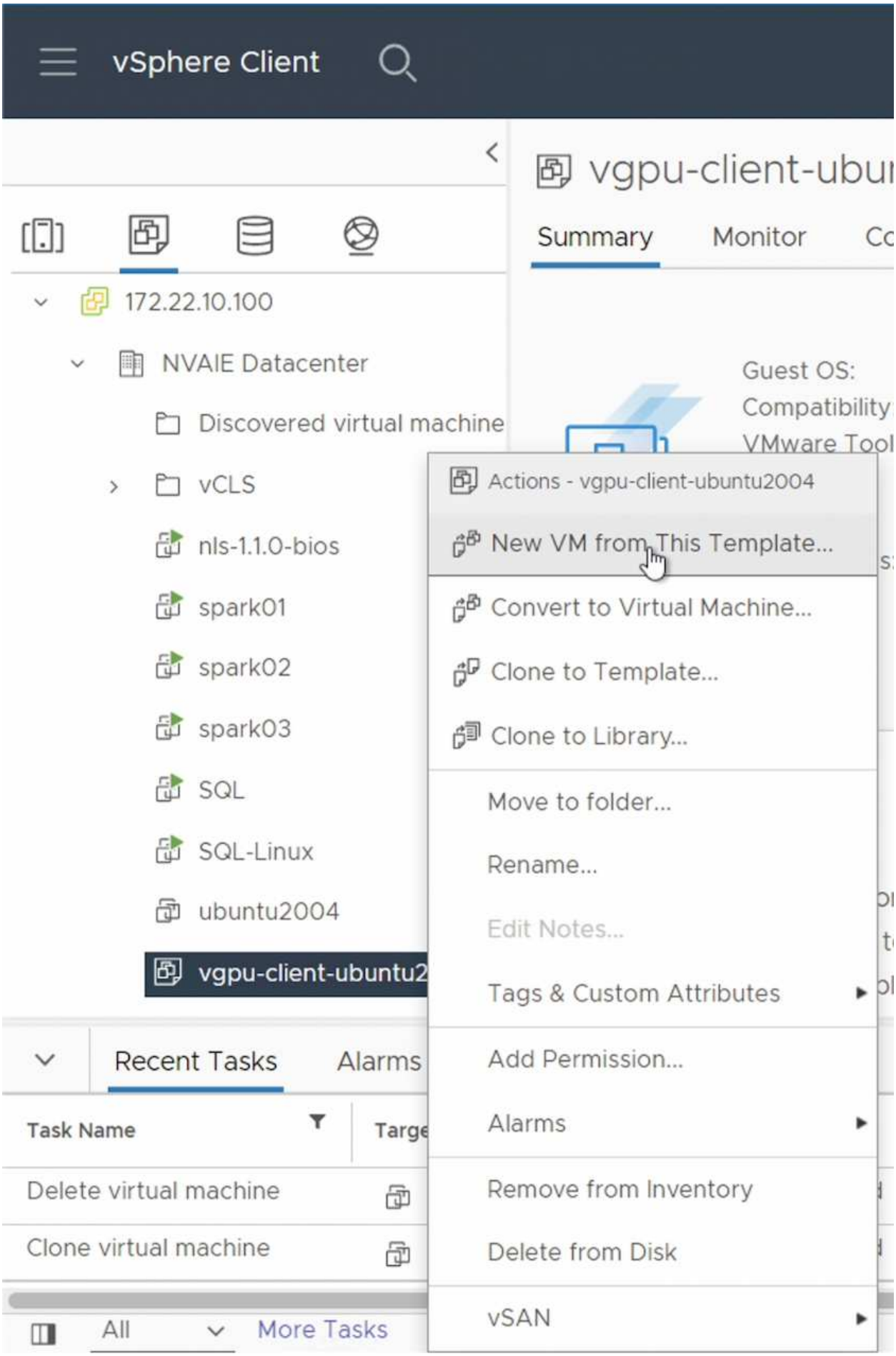
## 本节介绍在NVIDIA AI Enterprise环境中执行TensorFlow培训作业所需执行的任务。

前提条件

在执行本节所述的步骤之前、我们假定您已按照中所述的说明创建了子虚拟机模板 "设置" 页面。

使用模板创建子虚拟机

首先、您必须使用上一节中创建的模板创建新的子虚拟机。要使用模板创建新的子虚拟机、请登录到VMware vSphere、然后右键单击模板名称、选择"从此模板新建虚拟机…"、然后按照向导进行操作。


vSphere Client
vgpu-client-ubu
Summary
Monitor
Co
172.22.10.100
NVAIE Datacenter
Discovered virtual machine
vCLS
nls-1.1.0-bios
spark01
spark02
spark03
SQL
SQL-Linux
ubuntu2004
vgpu-client-ubuntu2
Guest OS:
Compatibility
VMware Tool
Actions - vgpu-client-ubuntu2004
New VM from This Template...
Convert to Virtual Machine...
Clone to Template...
Clone to Library...
Move to folder...
Rename...
Edit Notes...
Tags & Custom Attributes
Add Permission...
Alarms
Remove from Inventory
Delete from Disk
vSAN
Recent Tasks
Alarms
Task Name
Targe
Delete virtual machine
Clone virtual machine
All
More Tasks

#### 创建和挂载数据卷

接下来、您必须创建一个新的数据卷、用于存储培训数据集。您可以使用NetApp DataOps工具包快速创建新的数据卷。以下命令示例显示了如何创建容量为2 TB的名为"imagenet"的卷。

```
$ netapp_dataops_cli.py create vol -n imagenet -s 2TB
```

在为数据卷填充数据之前、必须先将其挂载到子虚拟机中。您可以使用NetApp DataOps工具包快速挂载数据卷。下面的示例命令显示了上一步创建的卷的布线。

```
$ sudo -E netapp_dataops_cli.py mount vol -n imagenet -m ~/imagenet
```

#### 填充数据卷

配置并挂载新卷后、可以从源位置检索培训数据集并将其放置在新卷上。这通常涉及从S3或Hadoop数据湖中提取数据、有时还需要数据工程师的帮助。

#### 执行**TensorFlow**培训作业

现在、您已准备好执行TensorFlow培训作业。要执行TensorFlow培训作业、请执行以下任务。

1. 提取NVIDIA NGC企业TensorFlow容器映像。

   ```
   $ sudo docker pull nvcr.io/nvaie/tensorflow-2-1:22.05-tf1-nvaie-2.1-py3
   ```

2. 启动NVIDIA NGC企业版TensorFlow容器的实例。使用"-v"选项将数据卷连接到容器。

   ```
   $ sudo docker run --gpus all -v ~/imagenet:/imagenet -it --rm
   nvcr.io/nvaie/tensorflow-2-1:22.05-tf1-nvaie-2.1-py3
   ```

3. 在容器中执行TensorFlow培训计划。下面的示例命令显示了容器映像中包含的示例RESNET-50培训计划的执行情况。

   ```
   $ python ./nvidia-examples/cnn/resnet.py --layers 50 -b 64 -i 200 -u
   batch --precision fp16 --data_dir /imagenet/data
   ```

## 从何处查找追加信息

### 要了解有关本文档中所述信息的更多信息，请参见以下文档和 / 或网站：

- NetApp ONTAP 数据管理软件—ONTAP 信息库

  http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286

- NetApp DataOps 工具包

  "https://github.com/NetApp/netapp-dataops-toolkit"

- 采用VMware的NVIDIA AI Enterprise

  https://www.nvidia.com/en-us/data-center/products/ai-enterprise/vmware/^]

致谢

- Bobby Oommen、高级NetApp经理
- NetApp系统管理员Ramesh Isaac
- NetApp技术营销工程师Roney Daniel

# TR-4851：适用于自动驾驶工作负载的NetApp StorageGRID 数据湖—解决方案 设计

NetApp公司David Arnette

TR-4851展示了NetApp StorageGRID 对象存储是机器学习(ML)和深度学习(DL)软件开发的数据存储库和管理系统。本白皮书介绍了自动驾驶汽车软件开发的数据流和要求、以及简化数据生命周期的StorageGRID 功能。此 解决方案 适用场景 是ML和DL开发流程中典型的任何多阶段数据管道工作流。

"TR-4851：适用于自动驾驶工作负载的NetApp StorageGRID 数据湖—解决方案 设计"

# NetApp AI 控制平台

## TR-4798： NetApp AI 控制平台

NetApp 公司 Mike Oglesby

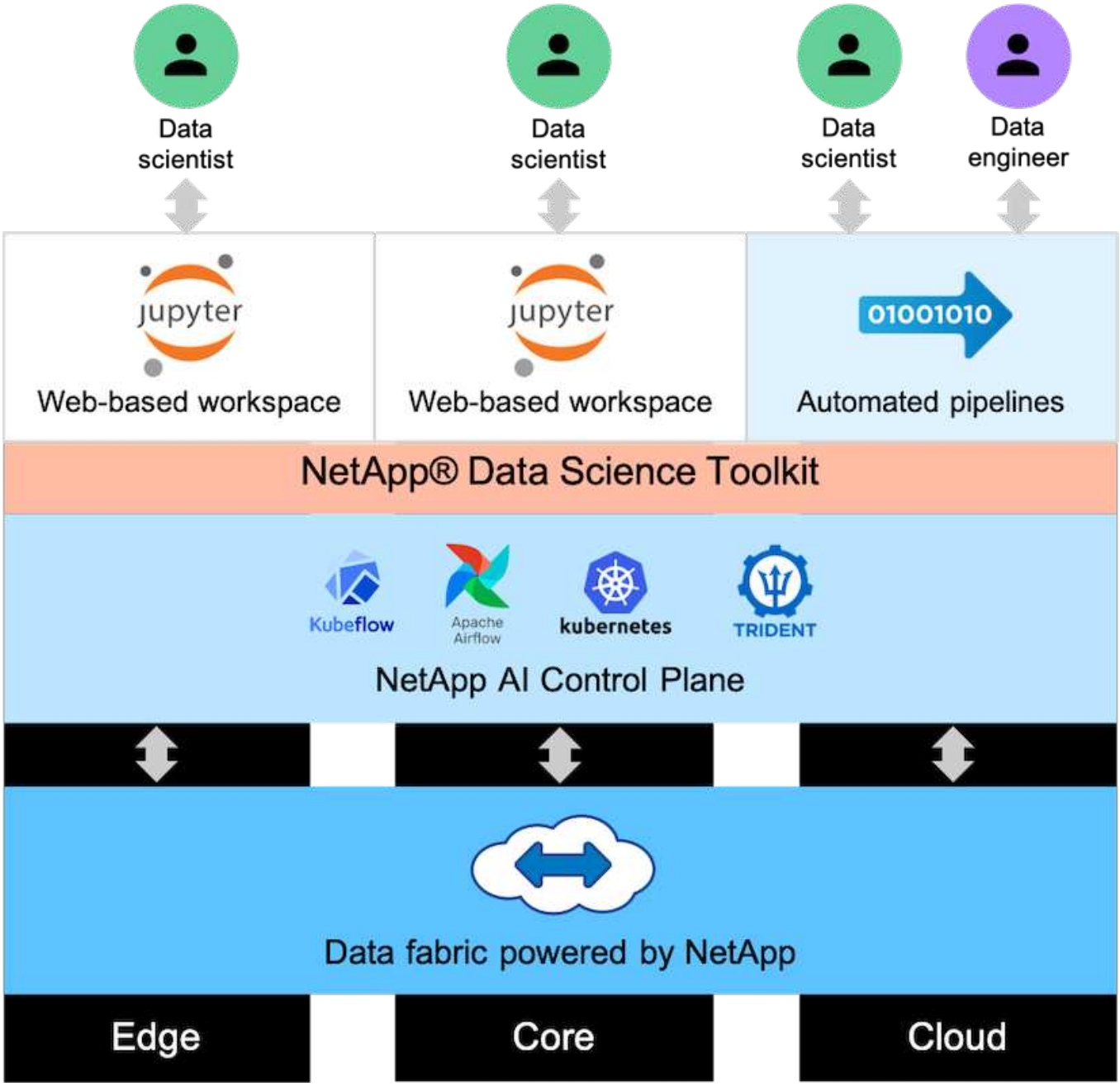
各行各业各种规模的企业和组织都在转向人工智能（AI），机器学习（ML）和深度学习（DL），以解决实际问题，提供创新产品和服务，并在竞争日益激烈的市场中占据优势。随着企业越来越多地使用 AI， ML 和 DL，他们面临着许多挑战，包括工作负载可扩展性和数据可用性。本文档演示了如何使用 NetApp AI 控制平台来应对这些挑战，NetApp AI 控制平台是一种将 NetApp 数据管理功能与常见开源工具和框架配对的解决方案。

此报告介绍如何快速克隆数据命名空间。此外，还将向您展示如何在站点和区域之间无缝复制数据，以创建统一的统一 AI/ML/DL 数据管道。此外，它还会指导您完成 AI， ML 和 DL 培训工作流的定义和实施，这些工作流可近乎即时地创建数据和模型基线，以实现可追溯性和版本控制。使用此解决方案，您可以跟踪每个模型训练返回到用于训练和 / 或验证模型的确切数据集。最后，本文档将向您介绍如何快速配置 Jupyter 笔记本电脑工作空间，以便访问海量数据集。

注意：对于涉及大量需要共享访问同一数据集的 GPU 服务器的 HPC 模式大规模分布式培训，或者如果您需要 / 更喜欢并行文件系统，请查看 "TR-4890"。本技术报告介绍了如何包括 "NetApp 完全支持的并行文件系统解决

方案 BeeGFS" 作为 NetApp AI 控制平台的一部分。此解决方案可从少数 NVIDIA DGX A100 系统扩展到全闪满的 140 节点 SuperPOD 。

NetApp AI 控制平台面向数据科学家和数据工程师，因此只需极少的 NetApp 或 NetApp ONTAP ® 专业知识即可。借助此解决方案，可以使用简单熟悉的工具和界面来执行数据管理功能。如果您的环境中已有 NetApp 存储，您可以立即测试运行 NetApp AI Control 平台。如果您要测试解决方案驱动器，但尚未安装 NetApp 存储，请访问 "cloud.netapp.com"，您只需几分钟即可使用基于云的 NetApp 存储解决方案启动并运行。下图显示了解决方案的可视化视图。




## 概念和组件

### 人工智能

AI 是一门计算机科学学科，其中计算机经过训练，可以模拟人类思维的认知功能。AI 开发人员培训计算机，以便以类似于甚至优于人类的方式学习和解决问题。深度学习和机器学习是 AI 的子领域。企业越来越多地采用 AI

， ML 和 DL 来满足其关键业务需求。以下是一些示例：

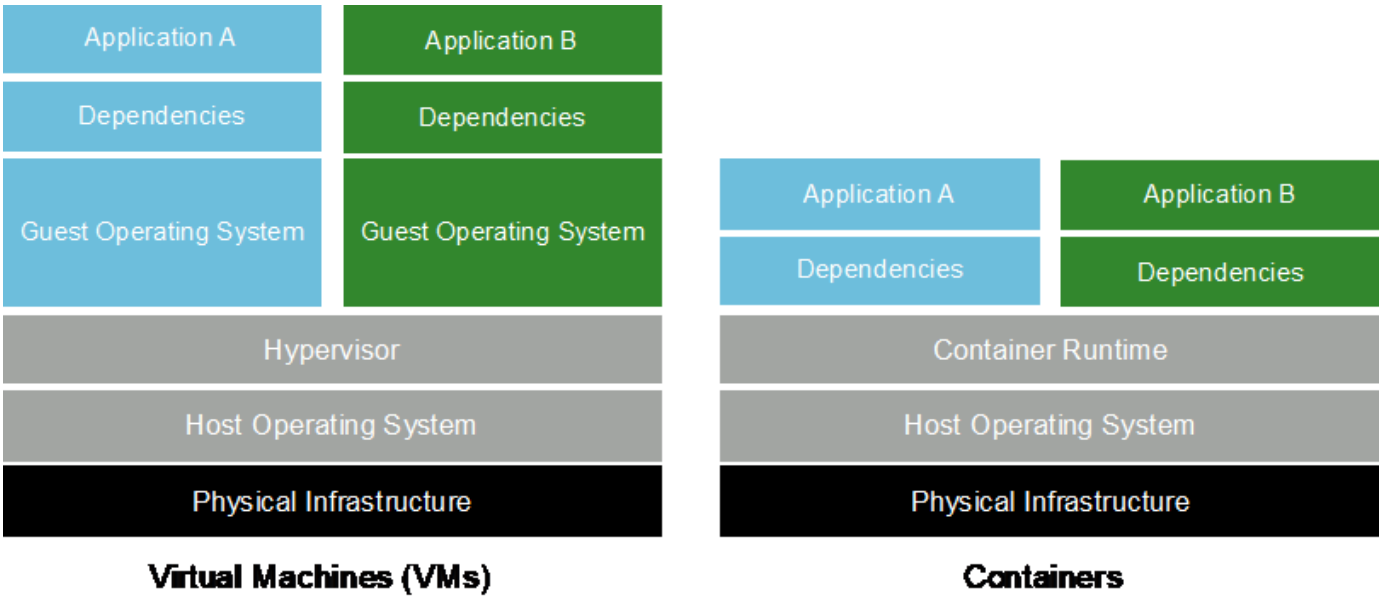
- 分析大量数据以挖掘以前未知的业务洞察力
- 使用自然语言处理直接与客户互动
- 自动化执行各种业务流程和功能

现代 AI 训练和推理工作负载需要大规模并行计算功能。因此， GPU 越来越多地用于执行 AI 操作，因为 GPU 的并行处理功能远远优于通用 CPU 。

容器

容器是在共享主机操作系统内核上运行的隔离用户空间实例。容器的采用率正在快速增长。容器可提供许多与虚拟机（ VM ）相同的应用程序沙盒优势。但是，由于虚拟机所依赖的虚拟机管理程序和子操作系统层已被消除，因此容器的重量要轻得多。下图展示了虚拟机与容器的可视化情况。

此外，还可以通过容器直接将应用程序依赖关系，运行时间等内容高效地打包到应用程序中。最常用的容器打包格式是 Docker 容器。已采用 Docker 容器格式进行容器化的应用程序可以在可以运行 Docker 容器的任何计算机上执行。即使计算机上不存在应用程序的依赖关系，也是如此，因为所有依赖关系都打包在容器中。有关详细信息，请访问 "Docker 网站"。




### Kubernetes

Kubernetes 是一款开源分布式容器编排平台，最初由 Google 设计，现在由 Cloud 原生计算基金会（ CNCF ）维护。Kubernetes 可以为容器化应用程序实现部署，管理和扩展功能的自动化。近年来， Kubernetes 已成为主导容器业务流程平台。虽然支持其他容器打包格式和运行时间，但 Kubernetes 最常用作 Docker 容器的业务流程系统。有关详细信息，请访问 "Kubernetes 网站"。

### NetApp Trident

Trident 是一款由 NetApp 开发和维护的开源存储编排程序，可大大简化 Kubernetes 工作负载的永久性存储的创建，管理和使用。Trident 本身是 Kubernetes 本机应用程序，直接在 Kubernetes 集群中运行。借助 Trident ，Kubernetes 用户（开发人员，数据科学家， Kubernetes 管理员等）可以采用他们已熟悉的标准 Kubernetes 格式创建，管理永久性存储卷并与其交互。同时，他们还可以利用 NetApp 的高级数据管理功能以及由 NetApp 技术提供支持的数据网络结构。Trident 可将持久存储的复杂性抽象化，并使其易于使用。有关详细信息，请访问

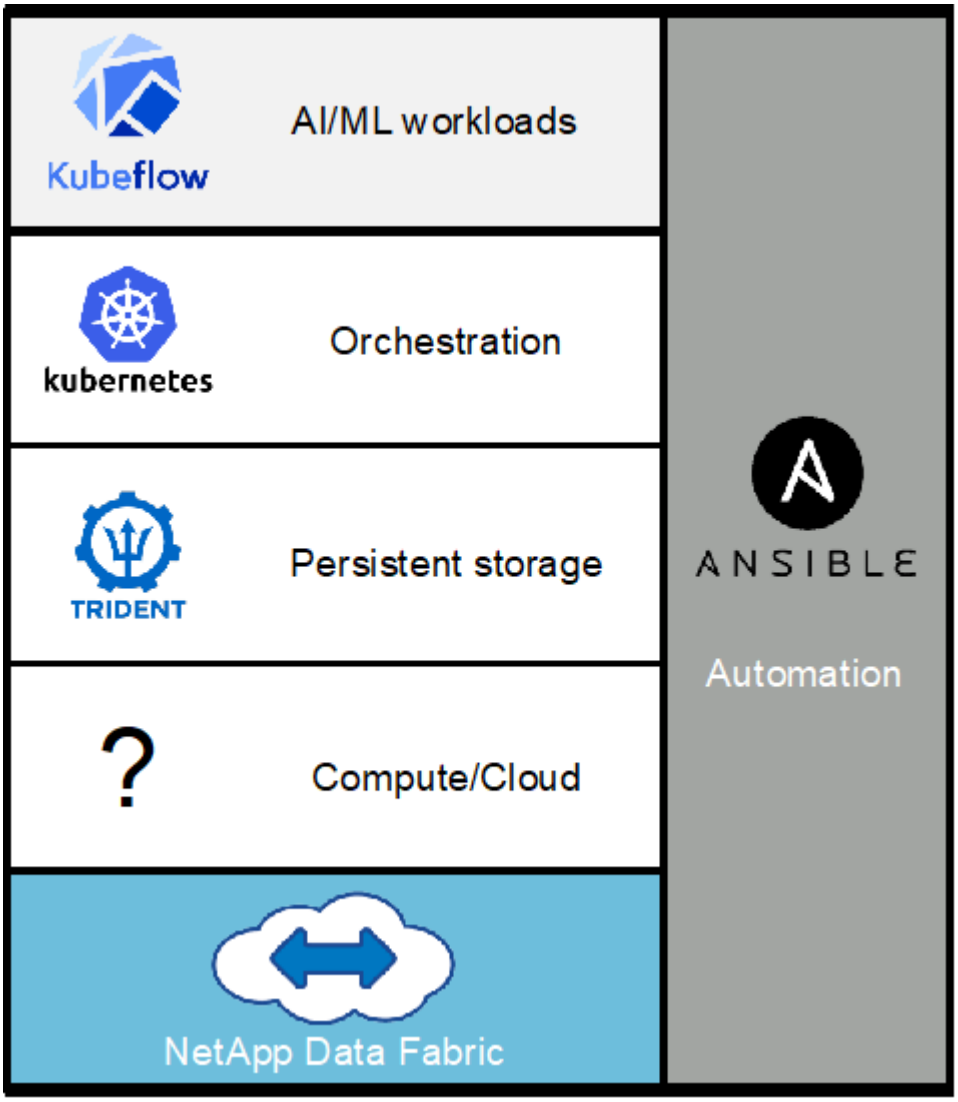
"Trident 网站"。

### NVIDIA DeepOps

DeepOps 是 NVIDIA 的一个开源项目，通过使用 Ansible ，可以根据最佳实践自动部署 GPU 服务器集群。DeepOps 采用模块化设计，可用于执行各种部署任务。在本文档及其所述的验证练习中， DeepOps 用于部署一个由 GPU 服务器辅助节点组成的 Kubernetes 集群。有关详细信息，请访问 "DeepOps 网站"。

### Kubeflow

Kubeflow 是一款适用于 Kubernetes 的开源 AI 和 ML 工具包，最初由 Google 开发。通过 Kubeflow 项目，可以在 Kubernetes 上轻松，便携且可扩展地部署 AI 和 ML 工作流。Kubeflow 将 Kubernetes 的错综复杂之处抽象出来，让数据科学家能够专注于他们最了解的—数据科学。有关可视化效果，请参见下图。随着企业 IT 部门越来越多地在 Kubernetes 上实现标准化， Kubeflow 的吸引力也越来越大。有关详细信息，请访问 "Kubeflow 网站"。




#### KubeFlow 管道

Kubeflow 管道是 Kubeflow 的一个关键组件。Kubeflow 管道是一个平台和标准，用于定义和部署可移植且可扩展的 AI 和 ML 工作流。有关详细信息，请参见 "Kubeflow 官方文档"。

#### Jupyter 笔记本电脑服务器

Jupyter 笔记本电脑服务器是一款开源 Web 应用程序，数据科学家可以利用它创建类似于维基的文档，这些文档称为 Jupyter 笔记本电脑，其中包含实时代码以及描述性测试。Jupyter 笔记本电脑在 AI 和 ML 社区中广泛使用，可用于记录，存储和共享 AI 和 ML 项目。Kubeflow 可简化 Kubernetes 上 Jupyter 笔记本电脑服务器的配置和部署。有关 Jupyter 笔记本电脑的详细信息，请访问 "Jupyter 网站"。有关 Kubeflow 环境中 Jupyter 笔记本电脑的详细信息，请参见 "Kubeflow 官方文档"。

### Apache 气流

Apache Airflow 是一个开源工作流管理平台，支持对复杂的企业工作流进行编程创作，计划和监控。它通常用于自动执行 ETL 和数据管道工作流，但不限于这些类型的工作流。气流项目由 Airbnb 发起，但此后在业内备受欢迎，现在由 Apache 软件基金会赞助。气流采用 Python 编写，气流工作流通过 Python 脚本创建，气流是按照 " 配置即代码 " 原则设计的。 现在，许多企业级气流用户都在 Kubernetes 上运行 Airflow 。

#### 定向循环图（DAG）

在气流模式下，工作流称为定向环比图（ DAG ）。DAG 由按顺序，并行或两者的组合执行的任务组成，具体取决于 DAG 定义。气流计划程序会根据 DAG 定义中指定的任务级别依赖关系对一组工作负载执行各个任务。DAG 是通过 Python 脚本定义和创建的。

### NetApp ONTAP 9.

NetApp ONTAP 9 是 NetApp 推出的最新一代存储管理软件，可帮助像您这样的企业打造现代化的基础架构并过渡到云就绪数据中心。借助行业领先的数据管理功能，无论数据位于何处， ONTAP 都可以通过一组工具来管理和保护数据。您还可以将数据自由移动到任何需要的位置：边缘，核心或云。ONTAP 9 包含许多功能，可简化数据管理，加快和保护关键数据，并在混合云架构中打造适应未来需求的基础架构。

#### 简化数据管理

数据管理对于企业 IT 运营至关重要，这样您就可以为应用程序和数据集使用适当的资源。ONTAP 包括以下功能，可简化运营并降低总运营成本：

- * 实时数据缩减和扩展的重复数据删除。 * 数据缩减可减少存储块中浪费的空间，重复数据删除可显著提高有效容量。
- * 最低，最高和自适应服务质量（ QoS ）。 * 细粒度 QoS 控制有助于在高度共享的环境中保持关键应用程序的性能水平。
- * ONTAP FabricPool 。 * 此功能可将冷数据自动分层到公有和私有云存储选项，包括 Amazon Web Services （ AWS ）， Azure 和 NetApp StorageGRID 基于对象的存储。

#### 加速和保护数据

ONTAP 可提供卓越的性能和数据保护，并通过以下功能扩展这些功能：

- * 高性能和低延迟。 * ONTAP 以尽可能低的延迟提供最高的吞吐量。
- * NetApp ONTAP FlexGroup 技术 * 。 FlexGroup 卷是一种高性能数据容器，可线性扩展到高达 20 PB 和 4000 亿个文件，从而提供一个可简化数据管理的命名空间。
- * 数据保护。 * ONTAP 提供内置数据保护功能，并在所有平台之间进行通用管理。
- * NetApp 卷加密。 * ONTAP 提供原生卷级加密，并支持板载和外部密钥管理。

ONTAP 9 可帮助您满足不断变化的苛刻业务需求：

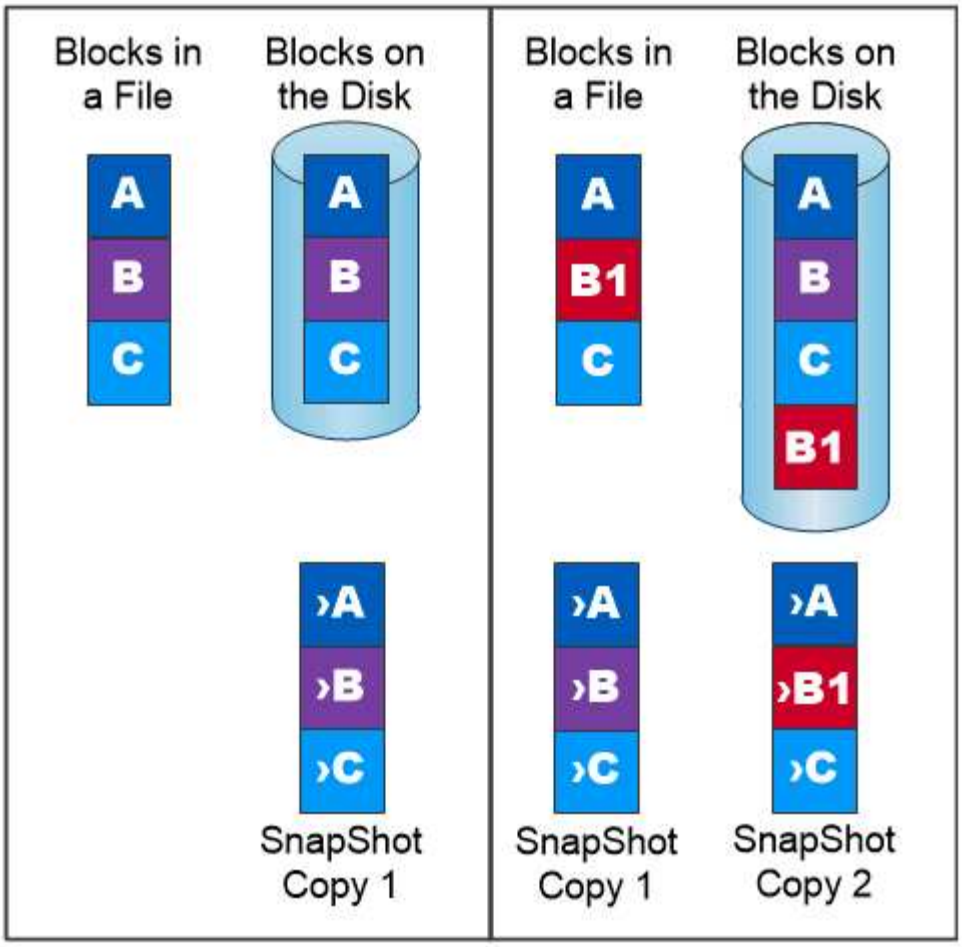
- * 无缝扩展和无中断运行。 * ONTAP 支持向现有控制器和横向扩展集群无中断添加容量。您可以升级到 NVMe 和 32 Gb FC 等最新技术，而无需进行昂贵的数据迁移或中断。
- * 云连接。 * ONTAP 是云连接最广泛的存储管理软件之一，可在所有公有云中选择软件定义的存储（ ONTAP Select ）和云原生实例（ NetApp Cloud Volumes Service ）。
- * 与新兴应用程序集成。 * 通过使用支持现有企业级应用程序的相同基础架构， ONTAP 可为 OpenStack ， Hadoop 和 MongoDB 等下一代平台和应用程序提供企业级数据服务。

### **NetApp Snapshot** 副本

NetApp Snapshot 副本是卷的只读时间点映像。该映像占用的存储空间极少，并且性能开销极低，因为它仅记录自创建上次 Snapshot 副本以来创建的文件所做的更改，如下图所示。

Snapshot 副本的效率归功于核心 ONTAP 存储虚拟化技术—任意位置写入文件布局（ Write Anywhere File Layout ， WAFL ）。与数据库一样， WAFL 使用元数据指向磁盘上的实际数据块。但是，与数据库不同， WAFL 不会覆盖现有块。它会将更新后的数据写入新块并更改元数据。这是因为 ONTAP 在创建 Snapshot 副本时引用元数据，而不是复制数据块，因此 Snapshot 副本的效率非常高。这样做可以避免其他系统在查找要复制的块时花费寻道时间，并避免创建副本本身的成本。

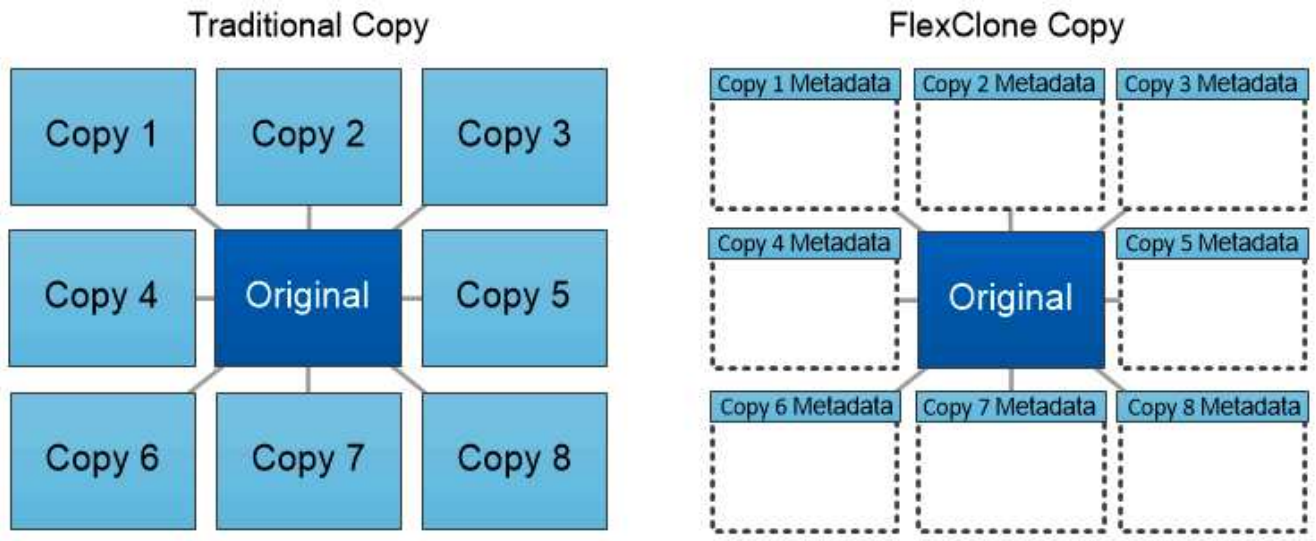
您可以使用 Snapshot 副本恢复单个文件或 LUN ，或者还原卷的整个内容。ONTAP 会将 Snapshot 副本中的指针信息与磁盘上的数据进行比较，以重建缺少或损坏的对象，而不会造成停机或高昂的性能成本。




*A Snapshot copy records only changes to the active file system since the last Snapshot copy.*

**NetApp FlexClone** 技术

NetApp FlexClone 技术会引用 Snapshot 元数据来创建卷的可写时间点副本。副本与其父级共享数据块，在将更改写入副本之前，除了元数据所需的存储外，不会占用任何其他存储，如下图所示。传统副本可能需要几分钟甚至几小时才能创建，而 FlexClone 软件可以让您几乎即时复制最大的数据集。因此，如果您需要相同数据集的多个副本（例如，开发工作空间）或数据集的临时副本（针对生产数据集测试应用程序），则这种情况是理想之选。

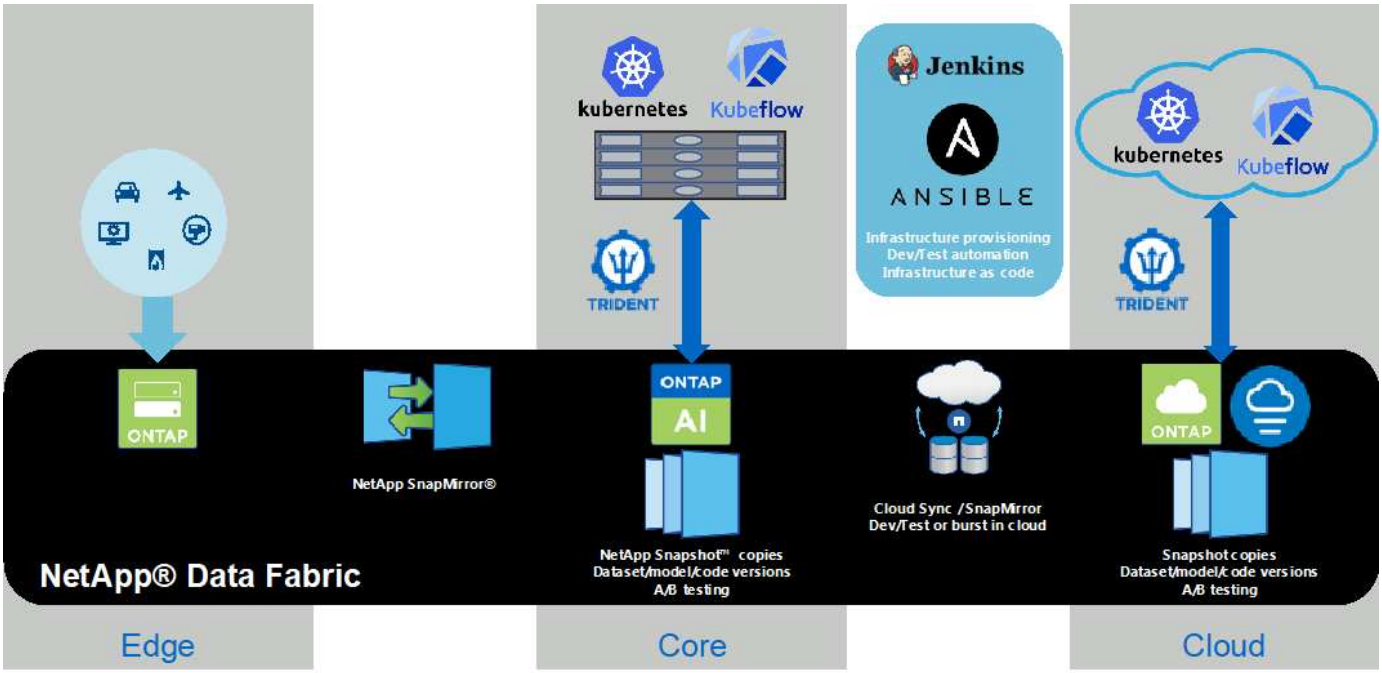



*FlexClone copies share data blocks with their parents, consuming no storage except what is required for metadata.*

**NetApp SnapMirror** 数据复制技术

NetApp SnapMirror 软件是一款经济高效且易于使用的统一复制解决方案，可跨数据网络结构实现。它可以通过 LAN 或 WAN 高速复制数据。它可以为各种类型的应用程序提供高数据可用性和快速数据复制，包括虚拟和传统环境中的业务关键型应用程序。在将数据复制到一个或多个 NetApp 存储系统并持续更新二级数据时，您的数据将保持最新，并可随时使用。不需要外部复制服务器。有关利用 SnapMirror 技术的架构示例，请参见下图。

SnapMirror 软件通过仅通过网络发送更改的块来利用 NetApp ONTAP 的存储效率。SnapMirror 软件还可使用内置网络压缩来加快数据传输速度，并将网络带宽利用率降低多达 70% 。借助 SnapMirror 技术，您可以利用一个精简复制数据流创建一个存储库，同时维护活动镜像和先前的时间点副本，从而将网络流量减少多达 50% 。



**NetApp BlueXP复制和同步**

BlueXP复制和同步是一项NetApp服务、用于快速安全地同步数据。无论您是需要在内部NFS还是SMB文件共享、NetApp StorageGRID、NetApp ONTAP S3、NetApp Cloud Volumes Service、Azure NetApp Files、AWS S3、AWS EFS、Azure Blb、 Google Cloud Storage或IBM Cloud Object Storage、BlueXP Copy and Sync可将文件快速安全地移动到您需要的位置。

数据传输完成后，即可在源和目标上完全使用。BlueXP复制和同步功能可以在触发更新时按需同步数据、也可以根据预定义的计划持续同步数据。不管怎样、BlueXP复制和同步功能只会移动增量、因此可以最大限度地减少数据复制所需的时间和资金。

BlueXP Copy and Sync是一款软件即服务(SaaS)工具、设置和使用极其简单。由BlueXP复制和同步触发的数据传输由数据代理执行。BlueXP复制和同步数据代理可以部署在AWS、Azure、Google Cloud Platform或内部环境中。

**NetApp XCP**

NetApp XCP 是一款基于客户端的软件，可用于任意到 NetApp 以及 NetApp 到 NetApp 的数据迁移和文件系统洞察。XCP 旨在通过利用所有可用系统资源来处理大容量数据集和高性能迁移来实现扩展和最大性能。XCP 可通过生成报告的选项帮助您全面了解文件系统。

NetApp XCP 可通过一个软件包提供，该软件包支持 NFS 和 SMB 协议。XCP 包括一个适用于 NFS 数据集的 Linux 二进制文件和一个适用于 SMB 数据集的 Windows 可执行文件。

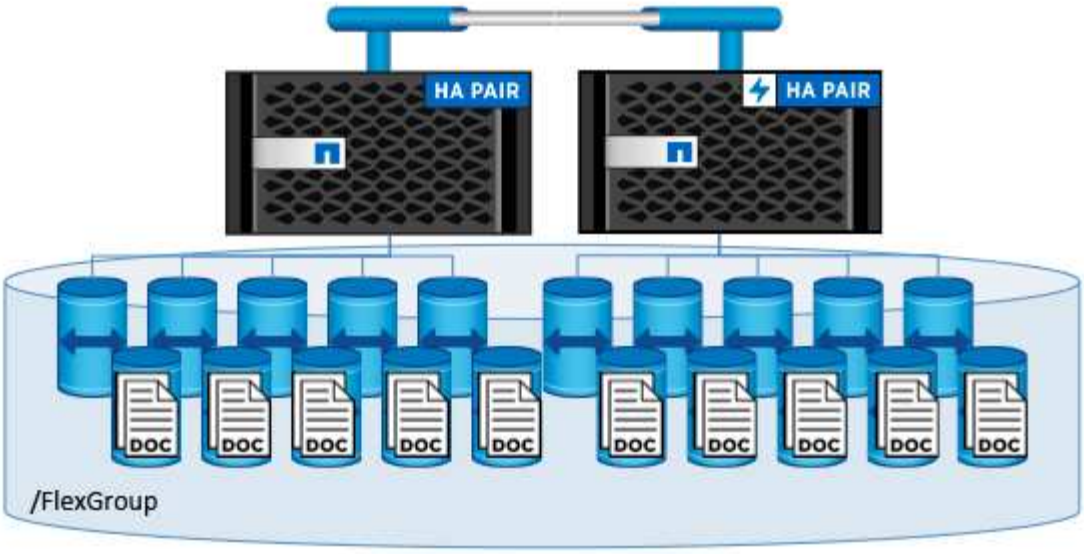
NetApp XCP 文件分析是一款基于主机的软件，可检测文件共享，对文件系统运行扫描并提供用于文件分析的信息板。XCP 文件分析与 NetApp 和非 NetApp 系统兼容，并可在 Linux 或 Windows 主机上运行，以便为 NFS 和 SMB 导出的文件系统提供分析。

**NetApp ONTAP FlexGroup 卷**

培训数据集可以是一组可能包含数十亿个文件的集合。文件可以包括文本，音频，视频以及其他形式的非结构化数据，这些数据必须进行存储和处理才能并行读取。存储系统必须存储大量小文件，并且必须并行读取这些文件，以便执行顺序和随机 I/O

FlexGroup 卷是一个包含多个成分卷的命名空间，如下图所示。从存储管理员的角度来看， FlexGroup 卷是一个受管卷，其作用类似于 NetApp FlexVol 卷。FlexGroup 卷中的文件将分配给各个成员卷，并且不会在卷或节点之间进行条带化。它们支持以下功能：

- FlexGroup 卷可为高元数据工作负载提供多 PB 的容量和可预测的低延迟。
- 它们在同一命名空间中最多支持 4000 亿个文件。
- 它们支持在 CPU ，节点，聚合和成分卷之间的 NAS 工作负载中执行并行操作 FlexVol 。




## 硬件和软件要求

NetApp AI 控制平面解决方案不依赖于此特定硬件。解决方案与 Trident 支持的任何 NetApp 物理存储设备，软件定义的实例或云服务兼容。例如， NetApp AFF 存储系统，Azure NetApp Files ， NetApp Cloud Volumes Service ， NetApp ONTAP Select 软件定义的存储实例或 NetApp Cloud Volumes ONTAP 实例。此外，只要 Kubernetes 和 NetApp Trident 支持所使用的 Kubernetes 版本，即可在任何 Kubernetes 集群上实施解决方案。有关 Kubeflow 支持的 Kubernetes 版本列表，请参见 "Kubeflow 官方文档"。有关 Trident 支持的 Kubernetes 版本列表，请参见 "Trident 文档"。有关用于验证解决方案的环境的详细信息，请参见下表。

| 基础架构组件 | 数量 | 详细信息 | 操作系统 |
|---|---|---|---|
| 部署跳转主机 | 1. | 虚拟机 | Ubuntu 20.04.2 LTS |
| Kubernetes 主节点 | 1. | 虚拟机 | Ubuntu 20.04.2 LTS |
| Kubernetes 工作节点 | 2. | 虚拟机 | Ubuntu 20.04.2 LTS |
| Kubernetes GPU 工作节点 | 2. | NVIDIA DGX-1 （裸机） | NVIDIA DGX OS 4.0.5 （基于 Ubuntu 18.04.2 LTS ） |
| 存储 | 1 个 HA 对 | NetApp AFF A220 | NetApp ONTAP 9.7 P6 |

| 软件组件 | **version** |
|---|---|
| Apache 气流 | 2.0.1 |
| Apache 气流 Helm 图表 | 8.0.8 |
| Docker | 19.03.12 |
| Kubeflow | 1.2 |
| Kubernetes | 1.18.9 |

| 软件组件 | **version** |
| --- | --- |
| NetApp Trident | 21.01.2 |
| NVIDIA DeepOps | 提交时主分支的 Trident 部署功能 "61898cdfda"；从 21.03 版开始提供所有其他功能 |

支持

NetApp 不为 Apache Airflow ， Docker ， Kubeflow ， Kubernetes 或 NVIDIA DeepOps 提供企业级支持。如果您对具有与 NetApp AI 控制平台解决方案类似的功能的完全受支持的解决方案感兴趣， "请联系 NetApp" 关于 NetApp 与合作伙伴共同提供的完全受支持的 AI/ML 解决方案。

## Kubernetes 部署

本节介绍部署 Kubernetes 集群以实施 NetApp AI 控制平面解决方案时必须完成的任务。如果您已有 Kubernetes 集群，则只要您运行的是 Kubernetes 和 NetApp Trident 支持的 Kubernetes 版本，就可以跳过本节。有关 Kubeflow 支持的 Kubernetes 版本列表，请参见 "Kubeflow 官方文档"。有关 Trident 支持的 Kubernetes 版本列表，请参见 "Trident 文档"。

对于采用采用 NVIDIA GPU 的裸机节点的内部 Kubernetes 部署， NetApp 建议使用 NVIDIA 的 DeepOps Kubernetes 部署工具。本节概述了使用 DeepOps 部署 Kubernetes 集群的过程。

前提条件

在执行本节所述的部署练习之前，我们假定您已执行以下任务：

1. 您已按照标准配置说明配置任何裸机 Kubernetes 节点（例如，属于 ONTAP AI POD 的 NVIDIA DGX 系统）。
2. 您已在所有 Kubernetes 主节点和工作节点以及部署跳转主机上安装受支持的操作系统。有关 DeepOps 支持的操作系统列表，请参见 "DeepOps GitHub 站点"。

使用 **NVIDIA DeepOps** 安装和配置 **Kubernetes**

要使用 NVIDIA DeepOps 部署和配置 Kubernetes 集群，请从部署跳转主机执行以下任务：

1. 按照上的说明下载 NVIDIA DeepOps "Getting Started 页面" 在 NVIDIA DeepOps GitHub 站点上。
2. 按照上的说明在集群中部署 Kubernetes "Kubernetes 部署指南页面" 在 NVIDIA DeepOps GitHub 站点上。

## NetApp Trident 部署和配置

**NetApp Trident** 部署和配置

本节介绍在 Kubernetes 集群中安装和配置 NetApp Trident 时必须完成的任务。

前提条件

在执行本节所述的部署练习之前，我们假定您已执行以下任务：

1. 您已有一个有效的 Kubernetes 集群，并且正在运行 Trident 支持的 Kubernetes 版本。有关支持的版本列表，请参见 "Trident 文档"。
2. 您已有一个可正常工作的 NetApp 存储设备，软件定义的实例或云存储服务， Trident 支持此服务。

#### 安装 Trident

要在 Kubernetes 集群中安装和配置 NetApp Trident ，请从部署跳转主机执行以下任务：

1. 使用以下方法之一部署 Trident ：
   - 如果您使用 NVIDIA DeepOps 部署 Kubernetes 集群，则也可以使用 NVIDIA DeepOps 在 Kubernetes 集群中部署 Trident 。要使用 DeepOps 部署 Trident ，请按照 "Trident 部署说明" 在 NVIDIA DeepOps GitHub 站点上。
   - 如果您未使用 NVIDIA DeepOps 部署 Kubernetes 集群，或者您只是希望手动部署 Trident ，则可以按照部署 Trident "部署说明" 在 Trident 文档中。有关如何配置的详细信息、请务必至少创建一个Trident后端和一个Kubernetes StorageClass "后端" 和 "StorageClasses" 请参见NetApp文档中链接的小节。

> 如果要在 ONTAP AI POD 上部署 NetApp AI 控制平台解决方案，请参见 "ONTAP AI 部署的 Trident 后端示例" 有关您可能希望创建和的不同 Trident 后端的一些示例，请参见 "适用于 ONTAP AI 部署的 Kubernetes Storageclasses 示例" 有关可能要创建的不同 Kubernetes StorageClasses 的一些示例，请参见。

#### ONTAP AI 部署的 Trident 后端示例

在使用 Trident 在 Kubernetes 集群中动态配置存储资源之前，必须先创建一个或多个 Trident 后端。以下示例展示了在 ONTAP AI POD 上部署 NetApp AI 控制平台解决方案时可能需要创建的不同类型的后端。有关后端的详细信息，请参见 "Trident 文档"。

1. NetApp 建议为要在 NetApp AFF 系统上使用的每个数据 LIF （提供数据访问的逻辑网络接口）创建一个启用了 FlexGroup 的 Trident 后端。这样，您可以在 LIF 之间平衡卷挂载

   以下示例命令显示了为与同一 ONTAP Storage Virtual Machine （ SVM ）关联的两个不同数据 LIF 创建两个启用了 FlexGroup 的 Trident 后端。这些后端使用 `ontap-nas-flexgroup` 存储驱动程序。ONTAP 支持两种主要数据卷类型： FlexVol 和 FlexGroup 。FlexVol 卷具有大小限制（截至本文撰写时，最大大小取决于特定部署）。另一方面， FlexGroup 卷可以线性扩展到高达 20 PB 和 4000 亿个文件，从而提供一个可显著简化数据管理的命名空间。因此， FlexGroup 卷最适合依赖大量数据的 AI 和 ML 工作负载。

   如果您使用的是少量数据，并且希望使用 FlexVol 卷而不是 FlexGroup 卷，则可以创建使用 `ontap-NAS` 存储驱动程序而不是 `ontap-nas-flexgroup` 存储驱动程序的 Trident 后端。

```
$ cat << EOF > ./trident-backend-ontap-ai-flexgroups-iface1.json
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": "ontap-ai-flexgroups-iface1",
    "managementLIF": "10.61.218.100",
    "dataLIF": "192.168.11.11",
    "svm": "ontapai_nfs",
```

```
    "username": "admin",
    "password": "ontapai"
}
EOF
$ tridentctl create backend -f ./trident-backend-ontap-ai-flexgroups-
iface1.json -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |
UUID                 | STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
$ cat << EOF > ./trident-backend-ontap-ai-flexgroups-iface2.json
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": "ontap-ai-flexgroups-iface2",
    "managementLIF": "10.61.218.100",
    "dataLIF": "192.168.12.12",
    "svm": "ontapai_nfs",
    "username": "admin",
    "password": "ontapai"
}
EOF
$ tridentctl create backend -f ./trident-backend-ontap-ai-flexgroups-
iface2.json -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |
UUID                 | STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexgroups-iface2 | ontap-nas-flexgroup | 61814d48-c770-436b-
9cb4-cf7ee661274d | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |
UUID                 | STATE  | VOLUMES |
+----------------------------+---------------------
```

```
+---------------------------------------+--------+---------+
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
| ontap-ai-flexgroups-iface2 | ontap-nas-flexgroup | 61814d48-c770-436b-
9cb4-cf7ee661274d | online |       0 |
+----------------------------+---------------------
+---------------------------------------+--------+---------+
```

2. NetApp 还建议创建一个或多个启用了 FlexVol 的 Trident 后端。如果您使用 FlexGroup 卷来训练数据集存储，则可能需要使用 FlexVol 卷来存储结果，输出，调试信息等。如果要使用 FlexVol 卷，必须创建一个或多个启用了 FlexVol 的 Trident 后端。下面的示例命令显示了如何创建一个使用单个数据 LIF 且已启用 FlexVol 的 Trident 后端。

```
$ cat << EOF > ./trident-backend-ontap-ai-flexvols.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-ai-flexvols",
    "managementLIF": "10.61.218.100",
    "dataLIF": "192.168.11.11",
    "svm": "ontapai_nfs",
    "username": "admin",
    "password": "ontapai"
}
EOF
$ tridentctl create backend -f ./trident-backend-ontap-ai-flexvols.json -n
trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-
a9c1-52a69657fabe | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-
a9c1-52a69657fabe | online |       0 |
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
| ontap-ai-flexgroups-iface2 | ontap-nas-flexgroup | 61814d48-c770-436b-
9cb4-cf7ee661274d | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
```

适用于 **ONTAP AI** 部署的 **Kubernetes StorageClasses** 示例

在使用 Trident 在 Kubernetes 集群中动态配置存储资源之前，必须创建一个或多个 Kubernetes StorageClasses 。以下示例展示了在 ONTAP AI POD 上部署 NetApp AI 控制平面解决方案时可能需要创建的不同类型的 StorageClasses 。有关 StorageClasses 的详

细信息，请参见 "Trident 文档"。

1. NetApp 建议为在一节中创建的每个启用了 FlexGroup 的 Trident 后端创建一个单独的 StorageClass "ONTAP AI 部署的 Trident 后端示例"，步骤 1 。通过这些粒度级 StorageClasses ，您可以将与特定 LIF （创建 Trident 后端时指定的 LIF ）相对应的 NFS 挂载添加为 StorageClass 规范文件中指定的特定后端。下面的示例命令显示了两个 StorageClasses 的创建过程，这两个 StorageClasses 对应于在部分中创建的两个示例后端 "ONTAP AI 部署的 Trident 后端示例"，步骤 1 。有关 StorageClasses 的详细信息，请参见 "Trident 文档"。

   为了在删除相应的 PersistentVolumeClaim （ PVC ）时不删除永久性卷，以下示例使用了 `reClaimPolicy` 值 `Retain` 。有关 `re"claimPolicy` " 字段的详细信息，请参见相关官员 "Kubernetes 文档"。

```
$ cat << EOF > ./storage-class-ontap-ai-flexgroups-retain-iface1.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexgroups-retain-iface1
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "ontap-ai-flexgroups-iface1:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexgroups-retain-
iface1.yaml
storageclass.storage.k8s.io/ontap-ai-flexgroups-retain-iface1 created
$ cat << EOF > ./storage-class-ontap-ai-flexgroups-retain-iface2.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexgroups-retain-iface2
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "ontap-ai-flexgroups-iface2:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexgroups-retain-
iface2.yaml
storageclass.storage.k8s.io/ontap-ai-flexgroups-retain-iface2 created
$ kubectl get storageclass
NAME                                PROVISIONER         AGE
ontap-ai-flexgroups-retain-iface1   netapp.io/trident   0m
ontap-ai-flexgroups-retain-iface2   netapp.io/trident   0m
```

2. NetApp 还建议创建一个与您在部分中创建的启用了 FlexVol 的 Trident 后端对应的 StorageClass "ONTAP

AI 部署的 Trident 后端示例"，步骤 2 。下面的示例命令显示了为 FlexVol 卷创建一个 StorageClass 的过程。

在以下示例中，未在 StorageClass 定义文件中指定特定后端，因为仅创建了一个启用了 FlexVol 的 Trident 后端。使用 Kubernetes 管理使用此 StorageClass 的卷时， Trident 会尝试使用使用 `ontap-NAS` 驱动程序的任何可用后端。

```
$ cat << EOF > ./storage-class-ontap-ai-flexvols-retain.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexvols-retain
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexvols-retain.yaml
storageclass.storage.k8s.io/ontap-ai-flexvols-retain created
$ kubectl get storageclass
NAME                                PROVISIONER         AGE
ontap-ai-flexgroups-retain-iface1   netapp.io/trident   1m
ontap-ai-flexgroups-retain-iface2   netapp.io/trident   1m
ontap-ai-flexvols-retain            netapp.io/trident   0m
```

3. NetApp 还建议为 FlexGroup 卷创建通用存储类。以下示例命令显示了如何为 FlexGroup 卷创建一个通用 StorageClass 。

   请注意， StorageClass 定义文件中未指定特定后端。因此，在使用 Kubernetes 管理使用此 StorageClass 的卷时， Trident 会尝试使用使用 `ontap-nas-flexgroup` 驱动程序的任何可用后端。

```
$ cat << EOF > ./storage-class-ontap-ai-flexgroups-retain.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexgroups-retain
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas-flexgroup"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexgroups-retain.yaml
storageclass.storage.k8s.io/ontap-ai-flexgroups-retain created
$ kubectl get storageclass
NAME                                  PROVISIONER         AGE
ontap-ai-flexgroups-retain            netapp.io/trident   0m
ontap-ai-flexgroups-retain-iface1     netapp.io/trident   2m
ontap-ai-flexgroups-retain-iface2     netapp.io/trident   2m
ontap-ai-flexvols-retain              netapp.io/trident   1m
```

## Kubeflow 部署

本节介绍在 Kubernetes 集群中部署 Kubeflow 必须完成的任务。

### 前提条件

在执行本节所述的部署练习之前，我们假定您已执行以下任务：

1. 您已有一个有效的 Kubernetes 集群，并且正在运行 Kubernetes 支持的版本。有关支持的版本列表，请参见 "Kubeflow 官方文档"。
2. 您已在 Kubernetes 集群中安装和配置 NetApp Trident ，如中所述 "Trident 部署和配置"。

### 设置默认 **Kubernetes StorageClass**

在部署 Kubeflow 之前，您必须在 Kubernetes 集群中指定一个默认 StorageClass 。Kubeflow 部署过程会尝试使用默认 StorageClass 配置新的永久性卷。如果未将任何 StorageClass 指定为默认 StorageClass ，则部署将失败。要在集群中指定默认 StorageClass ，请从部署跳转主机执行以下任务。如果已在集群中指定默认 StorageClass ，则可以跳过此步骤。

1. 将现有 StorageClasses 之一指定为默认 StorageClass 。以下示例命令显示了名为 `ontap-ai-FlexVols-Retain` 的 StorageClass 的默认 StorageClass 。

> `ontap-nas-flexgroup` Trident 后端类型的最小 PVC 大小相当大。默认情况下， KubeFlow 会尝试配置大小只有少数几 GB 的 PVC 。因此，在部署 Kubeflow 时，不应将利用 `ontap-nas-flexgroup` 后端类型的 StorageClass 指定为默认 StorageClass 。

```
$ kubectl get sc
NAME                                  PROVISIONER             AGE
ontap-ai-flexgroups-retain            csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface1     csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface2     csi.trident.netapp.io   25h
ontap-ai-flexvols-retain              csi.trident.netapp.io   3s
$ kubectl patch storageclass ontap-ai-flexvols-retain -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
storageclass.storage.k8s.io/ontap-ai-flexvols-retain patched
$ kubectl get sc
NAME                                   PROVISIONER             AGE
ontap-ai-flexgroups-retain             csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface1      csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface2      csi.trident.netapp.io   25h
ontap-ai-flexvols-retain (default)     csi.trident.netapp.io   54s
```

### 使用 **NVIDIA DeepOps** 部署 **Kubeflow**

NetApp 建议使用 NVIDIA DeepOps 提供的 Kubeflow 部署工具。要使用 DeepOps 部署工具在 Kubernetes 集群中部署 Kubeflow ，请从部署跳转主机执行以下任务。

> 或者，您也可以按照手动方式部署 Kubeflow "安装说明" 在 Kubeflow 官方文档中

1. 按照中的说明在集群中部署 Kubeflow "Kubeflow 部署说明" 在 NVIDIA DeepOps GitHub 站点上。
2. 记下 DeepOps Kubeflow 部署工具输出的 Kubeflow 信息板 URL 。

```
$ ./scripts/k8s/deploy_kubeflow.sh -x
…
INFO[0007] Applied the configuration Successfully!
filename="cmd/apply.go:72"
Kubeflow app installed to: /home/ai/kubeflow
It may take several minutes for all services to start. Run 'kubectl get
pods -n kubeflow' to verify
To remove (excluding CRDs, istio, auth, and cert-manager), run:
./scripts/k8s_deploy_kubeflow.sh -d
To perform a full uninstall : ./scripts/k8s_deploy_kubeflow.sh -D
Kubeflow Dashboard (HTTP NodePort): http://10.61.188.111:31380
```

3. 确认在 Kubeflow 命名空间中部署的所有 Pod 均显示 `Ststatus` of `running` ，并确认命名空间中部署的任何组件均未处于错误状态。可能需要几分钟时间，才能启动所有 Pod 。

```
$ kubectl get all -n kubeflow
NAME                                                          READY
```

```
STATUS    RESTARTS   AGE
pod/admission-webhook-bootstrap-stateful-set-0                  1/1
Running   0          95s
pod/admission-webhook-deployment-6b89c84c98-vrtbh               1/1
Running   0          91s
pod/application-controller-stateful-set-0                       1/1
Running   0          98s
pod/argo-ui-5dcf5d8b4f-m2wn4                                    1/1
Running   0          97s
pod/centraldashboard-cf4874ddc-7hcr8                            1/1
Running   0          97s
pod/jupyter-web-app-deployment-685b455447-gjhh7                 1/1
Running   0          96s
pod/katib-controller-88c97d85c-kgq66                            1/1
Running   1          95s
pod/katib-db-8598468fd8-5jw2c                                   1/1
Running   0          95s
pod/katib-manager-574c8c67f9-wtrf5                              1/1
Running   1          95s
pod/katib-manager-rest-778857c989-fjbzn                         1/1
Running   0          95s
pod/katib-suggestion-bayesianoptimization-65df4d7455-qthmw      1/1
Running   0          94s
pod/katib-suggestion-grid-56bf69f597-98vwn                      1/1
Running   0          94s
pod/katib-suggestion-hyperband-7777b76cb9-9v6dq                 1/1
Running   0          93s
pod/katib-suggestion-nasrl-77f6f9458c-2qzxq                     1/1
Running   0          93s
pod/katib-suggestion-random-77b88b5c79-l64j9                    1/1
Running   0          93s
pod/katib-ui-7587c5b967-nd629                                   1/1
Running   0          95s
pod/metacontroller-0                                            1/1
Running   0          96s
pod/metadata-db-5dd459cc-swzkm                                  1/1
Running   0          94s
pod/metadata-deployment-6cf77db994-69fk7                        1/1
Running   3          93s
pod/metadata-deployment-6cf77db994-mpbjt                        1/1
Running   3          93s
pod/metadata-deployment-6cf77db994-xg7tz                        1/1
Running   3          94s
pod/metadata-ui-78f5b59b56-qb6kr                                1/1
Running   0          94s
pod/minio-758b769d67-llvdr                                      1/1
```

```
Running   0          91s
pod/ml-pipeline-5875b9db95-g8t2k                                1/1
Running   0          91s
pod/ml-pipeline-persistenceagent-9b69ddd46-bt9r9                1/1
Running   0          90s
pod/ml-pipeline-scheduledworkflow-7b8d756c76-7x56s              1/1
Running   0          90s
pod/ml-pipeline-ui-79ffd9c76-fcwpd                              1/1
Running   0          90s
pod/ml-pipeline-viewer-controller-deployment-5fdc87f58-b2t9r   1/1
Running   0          90s
pod/mysql-657f87857d-l5k9z                                      1/1
Running   0          91s
pod/notebook-controller-deployment-56b4f59bbf-8bvnr             1/1
Running   0          92s
pod/profiles-deployment-6bc745947-mrdkh                         2/2
Running   0          90s
pod/pytorch-operator-77c97f4879-hmlrv                           1/1
Running   0          92s
pod/seldon-operator-controller-manager-0                        1/1
Running   1          91s
pod/spartakus-volunteer-5fdfddb779-l7qkm                        1/1
Running   0          92s
pod/tensorboard-6544748d94-nh8b2                                1/1
Running   0          92s
pod/tf-job-dashboard-56f79c59dd-6w59t                           1/1
Running   0          92s
pod/tf-job-operator-79cbfd6dbc-rb58c                            1/1
Running   0          91s
pod/workflow-controller-db644d554-cwrnb                         1/1
Running   0          97s
NAME                                                   TYPE
CLUSTER-IP      EXTERNAL-IP   PORT(S)              AGE
service/admission-webhook-service                      ClusterIP
10.233.51.169   <none>        443/TCP              97s
service/application-controller-service                 ClusterIP
10.233.4.54     <none>        443/TCP              98s
service/argo-ui                                        NodePort
10.233.47.191   <none>        80:31799/TCP         97s
service/centraldashboard                               ClusterIP
10.233.8.36     <none>        80/TCP               97s
service/jupyter-web-app-service                        ClusterIP
10.233.1.42     <none>        80/TCP               97s
service/katib-controller                               ClusterIP
10.233.25.226   <none>        443/TCP              96s
service/katib-db                                       ClusterIP
```

```
10.233.33.151   <none>        3306/TCP            97s
service/katib-manager                                ClusterIP
10.233.46.239   <none>        6789/TCP            96s
service/katib-manager-rest                           ClusterIP
10.233.55.32    <none>        80/TCP              96s
service/katib-suggestion-bayesianoptimization        ClusterIP
10.233.49.191   <none>        6789/TCP            95s
service/katib-suggestion-grid                        ClusterIP
10.233.9.105    <none>        6789/TCP            95s
service/katib-suggestion-hyperband                   ClusterIP
10.233.22.2     <none>        6789/TCP            95s
service/katib-suggestion-nasrl                       ClusterIP
10.233.63.73    <none>        6789/TCP            95s
service/katib-suggestion-random                      ClusterIP
10.233.57.210   <none>        6789/TCP            95s
service/katib-ui                                     ClusterIP
10.233.6.116    <none>        80/TCP              96s
service/metadata-db                                  ClusterIP
10.233.31.2     <none>        3306/TCP            96s
service/metadata-service                             ClusterIP
10.233.27.104   <none>        8080/TCP            96s
service/metadata-ui                                  ClusterIP
10.233.57.177   <none>        80/TCP              96s
service/minio-service                                ClusterIP
10.233.44.90    <none>        9000/TCP            94s
service/ml-pipeline                                  ClusterIP
10.233.41.201   <none>        8888/TCP,8887/TCP   94s
service/ml-pipeline-tensorboard-ui                   ClusterIP
10.233.36.207   <none>        80/TCP              93s
service/ml-pipeline-ui                               ClusterIP
10.233.61.150   <none>        80/TCP              93s
service/mysql                                        ClusterIP
10.233.55.117   <none>        3306/TCP            94s
service/notebook-controller-service                  ClusterIP
10.233.10.166   <none>        443/TCP             95s
service/profiles-kfam                                ClusterIP
10.233.33.79    <none>        8081/TCP            92s
service/pytorch-operator                             ClusterIP
10.233.37.112   <none>        8443/TCP            95s
service/seldon-operator-controller-manager-service   ClusterIP
10.233.30.178   <none>        443/TCP             92s
service/tensorboard                                  ClusterIP
10.233.58.151   <none>        9000/TCP            94s
service/tf-job-dashboard                             ClusterIP
10.233.4.17     <none>        80/TCP              94s
service/tf-job-operator                              ClusterIP
```

```
10.233.60.32    <none>        8443/TCP            94s
service/webhook-server-service                       ClusterIP
10.233.32.167   <none>        443/TCP             87s
NAME                                                          READY   UP-
TO-DATE   AVAILABLE   AGE
deployment.apps/admission-webhook-deployment                  1/1     1
1           97s
deployment.apps/argo-ui                                       1/1     1
1           97s
deployment.apps/centraldashboard                              1/1     1
1           97s
deployment.apps/jupyter-web-app-deployment                    1/1     1
1           97s
deployment.apps/katib-controller                              1/1     1
1           96s
deployment.apps/katib-db                                      1/1     1
1           97s
deployment.apps/katib-manager                                 1/1     1
1           96s
deployment.apps/katib-manager-rest                            1/1     1
1           96s
deployment.apps/katib-suggestion-bayesianoptimization         1/1     1
1           95s
deployment.apps/katib-suggestion-grid                         1/1     1
1           95s
deployment.apps/katib-suggestion-hyperband                    1/1     1
1           95s
deployment.apps/katib-suggestion-nasrl                        1/1     1
1           95s
deployment.apps/katib-suggestion-random                       1/1     1
1           95s
deployment.apps/katib-ui                                      1/1     1
1           96s
deployment.apps/metadata-db                                   1/1     1
1           96s
deployment.apps/metadata-deployment                           3/3     3
3           96s
deployment.apps/metadata-ui                                   1/1     1
1           96s
deployment.apps/minio                                         1/1     1
1           94s
deployment.apps/ml-pipeline                                   1/1     1
1           94s
deployment.apps/ml-pipeline-persistenceagent                  1/1     1
1           93s
deployment.apps/ml-pipeline-scheduledworkflow                 1/1     1
```

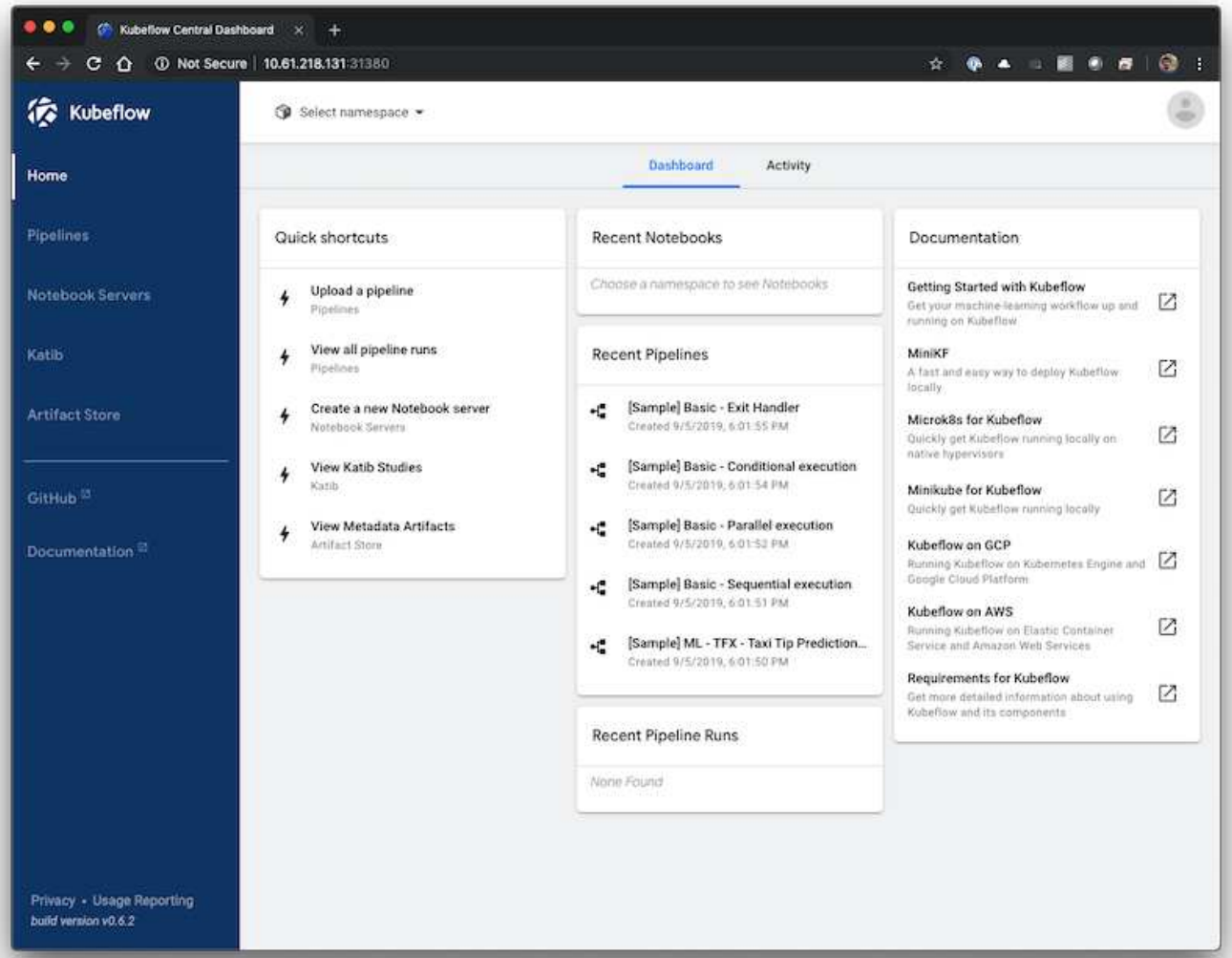
```
1           93s
deployment.apps/ml-pipeline-ui                                1/1     1
1           93s
deployment.apps/ml-pipeline-viewer-controller-deployment   1/1     1
1           93s
deployment.apps/mysql                                         1/1     1
1           94s
deployment.apps/notebook-controller-deployment                1/1     1
1           95s
deployment.apps/profiles-deployment                           1/1     1
1           92s
deployment.apps/pytorch-operator                              1/1     1
1           95s
deployment.apps/spartakus-volunteer                           1/1     1
1           94s
deployment.apps/tensorboard                                   1/1     1
1           94s
deployment.apps/tf-job-dashboard                              1/1     1
1           94s
deployment.apps/tf-job-operator                               1/1     1
1           94s
deployment.apps/workflow-controller                           1/1     1
1           97s
NAME
DESIRED   CURRENT   READY   AGE
replicaset.apps/admission-webhook-deployment-6b89c84c98              1
1         1       97s
replicaset.apps/argo-ui-5dcf5d8b4f                                   1
1         1       97s
replicaset.apps/centraldashboard-cf4874ddc                           1
1         1       97s
replicaset.apps/jupyter-web-app-deployment-685b455447                1
1         1       97s
replicaset.apps/katib-controller-88c97d85c                           1
1         1       96s
replicaset.apps/katib-db-8598468fd8                                  1
1         1       97s
replicaset.apps/katib-manager-574c8c67f9                             1
1         1       96s
replicaset.apps/katib-manager-rest-778857c989                        1
1         1       96s
replicaset.apps/katib-suggestion-bayesianoptimization-65df4d7455     1
1         1       95s
replicaset.apps/katib-suggestion-grid-56bf69f597                     1
1         1       95s
replicaset.apps/katib-suggestion-hyperband-7777b76cb9                1
```

```
1         1       95s
replicaset.apps/katib-suggestion-nasrl-77f6f9458c                    1
1         1       95s
replicaset.apps/katib-suggestion-random-77b88b5c79                   1
1         1       95s
replicaset.apps/katib-ui-7587c5b967                                  1
1         1       96s
replicaset.apps/metadata-db-5dd459cc                                 1
1         1       96s
replicaset.apps/metadata-deployment-6cf77db994                       3
3         3       96s
replicaset.apps/metadata-ui-78f5b59b56                               1
1         1       96s
replicaset.apps/minio-758b769d67                                     1
1         1       93s
replicaset.apps/ml-pipeline-5875b9db95                               1
1         1       93s
replicaset.apps/ml-pipeline-persistenceagent-9b69ddd46               1
1         1       92s
replicaset.apps/ml-pipeline-scheduledworkflow-7b8d756c76             1
1         1       91s
replicaset.apps/ml-pipeline-ui-79ffd9c76                             1
1         1       91s
replicaset.apps/ml-pipeline-viewer-controller-deployment-5fdc87f58   1
1         1       91s
replicaset.apps/mysql-657f87857d                                     1
1         1       92s
replicaset.apps/notebook-controller-deployment-56b4f59bbf            1
1         1       94s
replicaset.apps/profiles-deployment-6bc745947                        1
1         1       91s
replicaset.apps/pytorch-operator-77c97f4879                          1
1         1       94s
replicaset.apps/spartakus-volunteer-5fdfddb779                       1
1         1       94s
replicaset.apps/tensorboard-6544748d94                               1
1         1       93s
replicaset.apps/tf-job-dashboard-56f79c59dd                          1
1         1       93s
replicaset.apps/tf-job-operator-79cbfd6dbc                           1
1         1       93s
replicaset.apps/workflow-controller-db644d554                        1
1         1       97s
NAME                                                        READY   AGE
statefulset.apps/admission-webhook-bootstrap-stateful-set   1/1     97s
statefulset.apps/application-controller-stateful-set        1/1     98s
```

```
statefulset.apps/metacontroller                                  1/1     98s
statefulset.apps/seldon-operator-controller-manager              1/1     92s
$ kubectl get pvc -n kubeflow
NAME                STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS                AGE
katib-mysql         Bound    pvc-b07f293e-d028-11e9-9b9d-00505681a82d
10Gi       RWO              ontap-ai-flexvols-retain   27m
metadata-mysql      Bound    pvc-b0f3f032-d028-11e9-9b9d-00505681a82d
10Gi       RWO              ontap-ai-flexvols-retain   27m
minio-pv-claim      Bound    pvc-b22727ee-d028-11e9-9b9d-00505681a82d
20Gi       RWO              ontap-ai-flexvols-retain   27m
mysql-pv-claim      Bound    pvc-b2429afd-d028-11e9-9b9d-00505681a82d
20Gi       RWO              ontap-ai-flexvols-retain   27m
```

4. 在 Web 浏览器中，通过导航到步骤 2 中记下的 URL 来访问 Kubeflow 中央信息板。

   默认用户名为 `admin@kubeflow.org` ，默认密码为 `1231234` 。要创建其他用户，请按照中的说明进行操作 "Kubeflow 官方文档"。



## Kubeflow 操作和任务示例

本节包括您可能希望使用 Kubeflow 执行的各种操作和任务的示例。

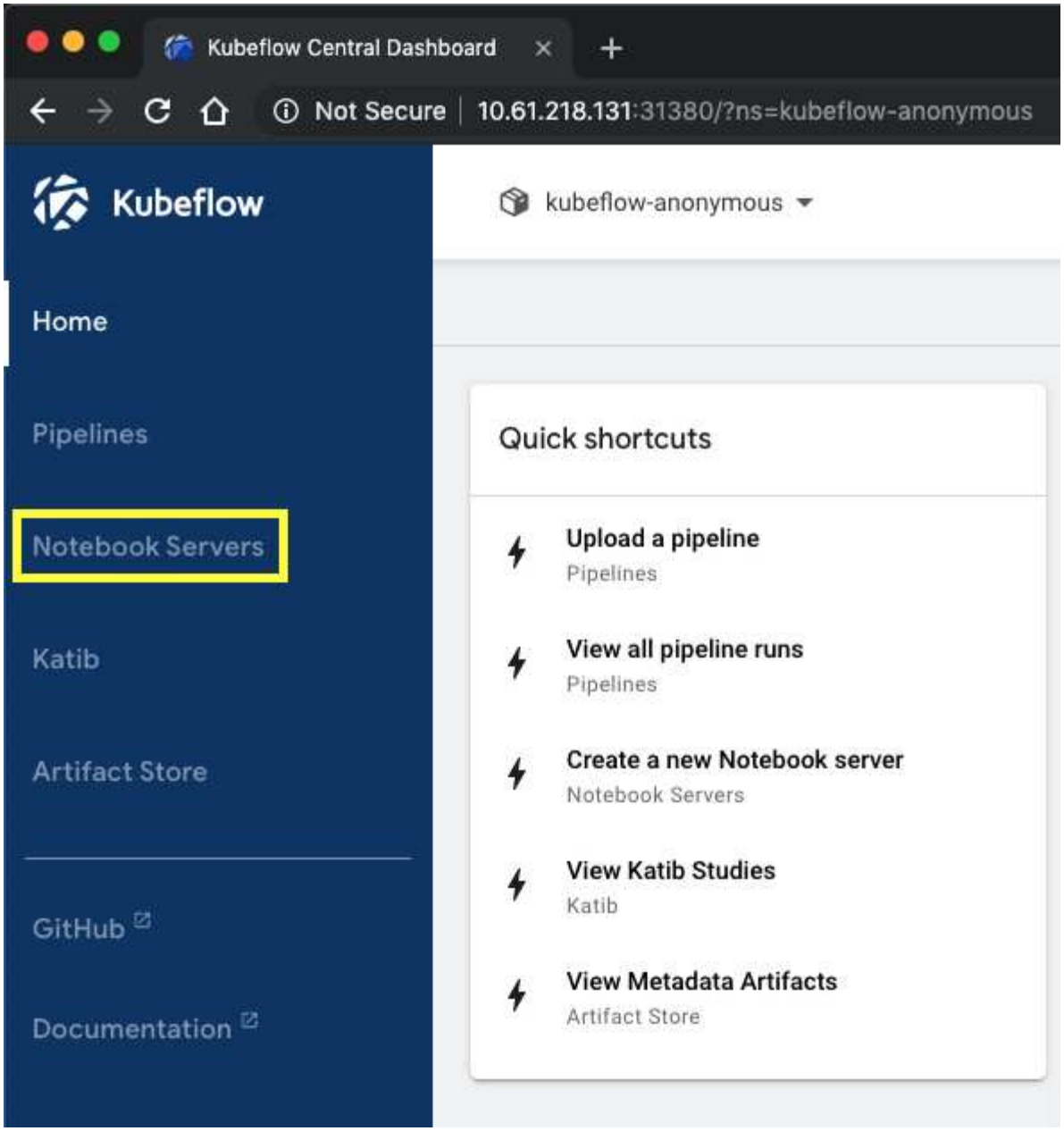
### Kubeflow 操作和任务示例

本节包括您可能希望使用 Kubeflow 执行的各种操作和任务的示例。

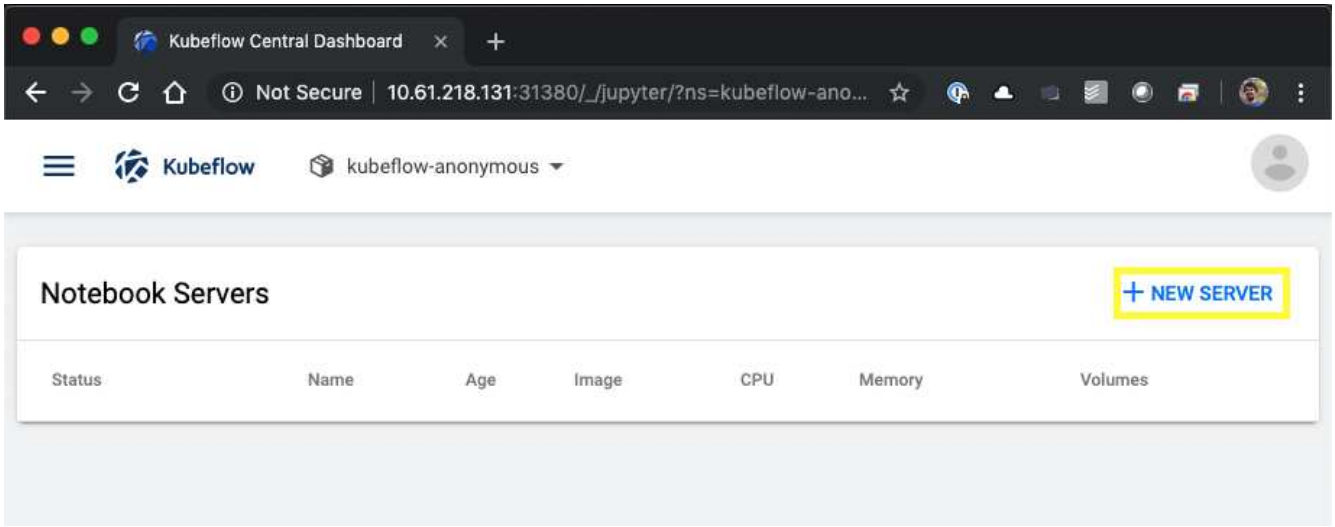
#### 为数据科学家或开发人员配置 Jupyter 笔记本电脑工作空间

Kubeflow 能够快速配置新的 Jupyter 笔记本电脑服务器，以充当数据科学家工作空间。要使用 Kubeflow 配置新的 Jupyter 笔记本电脑服务器，请执行以下任务。有关 Kubeflow 上下文中 Jupyter 笔记本电脑的详细信息，请参见 "Kubeflow 官方文档"。

1. 从 Kubeflow 中央信息板中，单击主菜单中的 Notebook Servers 以导航到 Jupyter 笔记本电脑服务器管理页面。

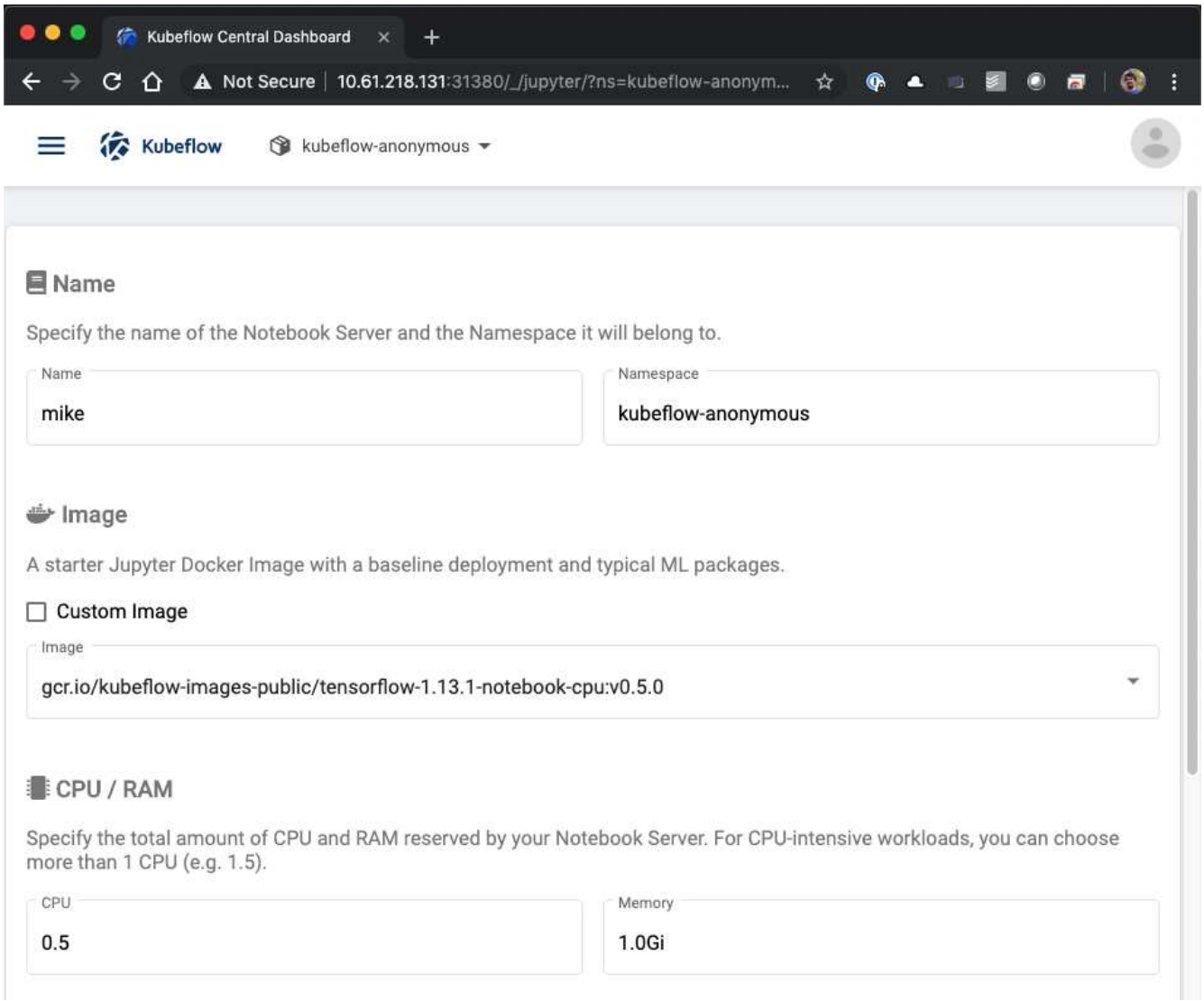



2. 单击新服务器以配置新的 Jupyter 笔记本电脑服务器。

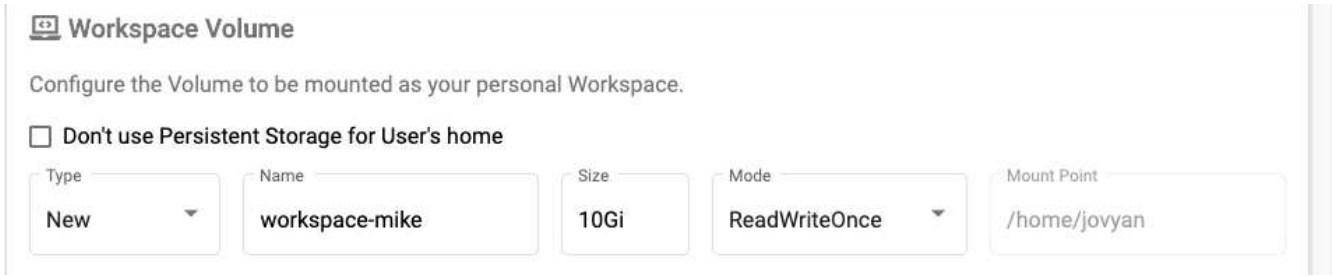



3. 为新服务器指定一个名称，选择希望服务器基于的 Docker 映像，并指定服务器要预留的 CPU 和 RAM 量。如果命名空间字段为空，请使用页面标题中的选择命名空间菜单选择命名空间。然后， Namespace 字段将自动填充所选命名空间。

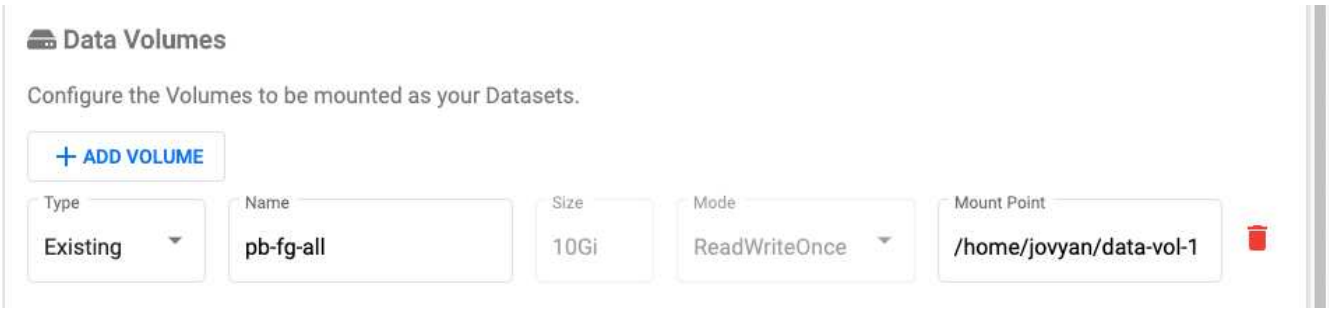
   在以下示例中，选择了 `kubeflow-anonymous` 命名空间。此外，还接受 Docker 映像， CPU 和 RAM 的默认值。



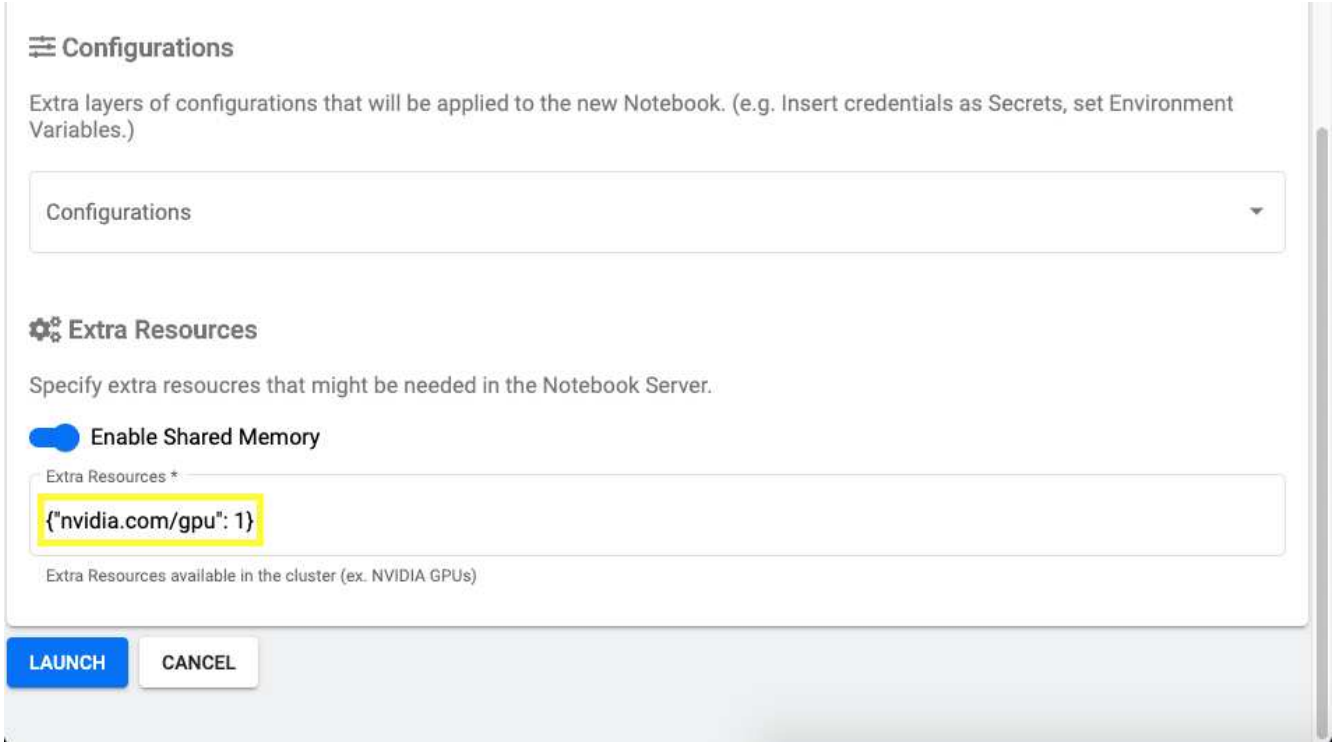

4. 指定工作空间卷详细信息。如果选择创建新卷，则会使用默认 StorageClass 配置该卷或 PVC 。因为在部分中，使用 Trident 的 StorageClass 被指定为默认 StorageClass "Kubeflow 部署"，卷或 PVC 配置有 Trident 。此卷会自动挂载为 Jupyter 笔记本电脑服务器容器中的默认工作空间。用户在服务器上创建但未保存到单独数据卷的任何笔记本电脑将自动保存到此工作空间卷。因此，这些笔记本电脑在重新启动后会持久存在。



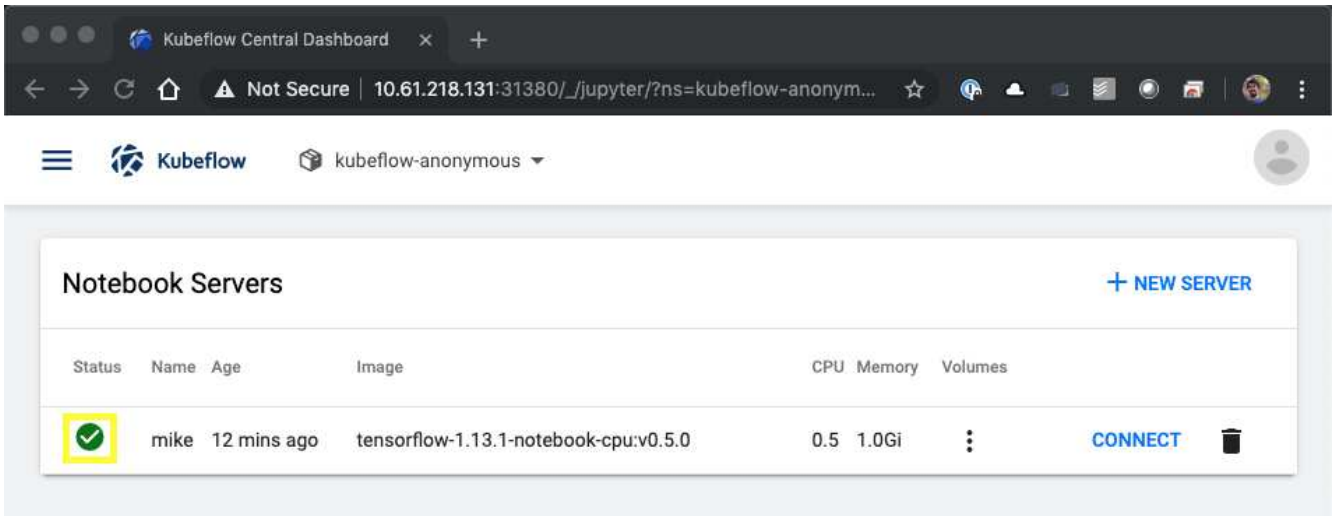

5. 添加数据卷。以下示例指定了一个名为 "pt-fg-all" 的现有 PVC 并接受默认挂载点。



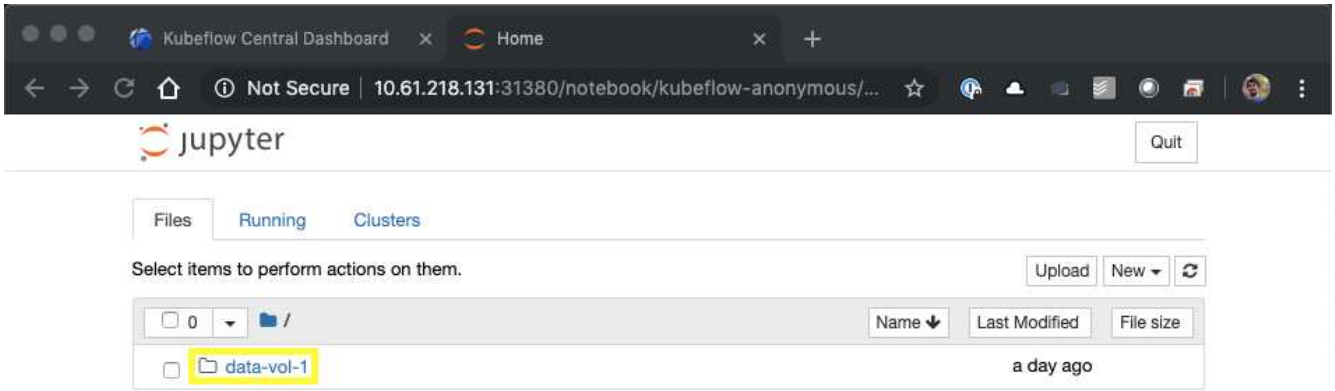

6. * 可选： * 请求将所需数量的 GPU 分配给您的笔记本服务器。在以下示例中，请求一个 GPU 。

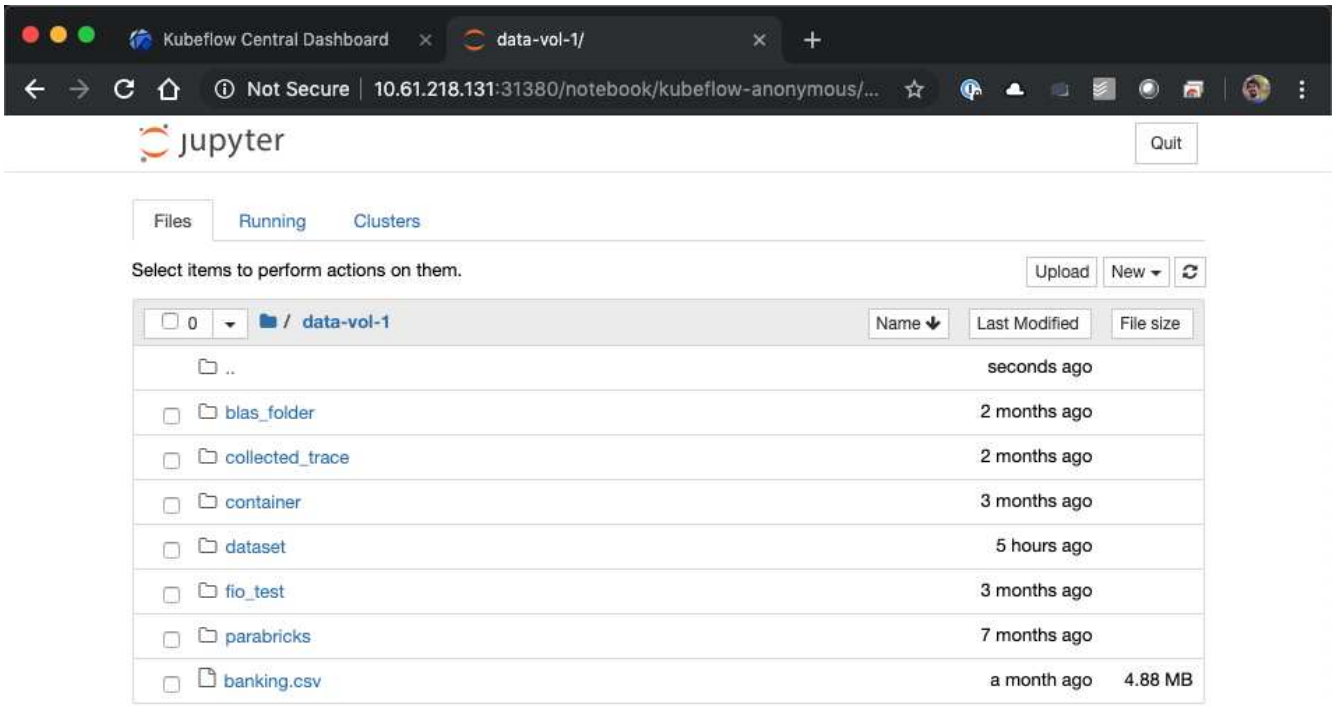



7. 单击启动以配置新的笔记本电脑服务器。
8. 等待笔记本电脑服务器完全配置完毕。如果您从未使用您指定的 Docker 映像配置服务器，则可能需要几分钟的时间，因为需要下载此映像。服务器配置完成后， Jupyter 笔记本电脑服务器管理页面上的状态列会显示一个绿色复选标记。



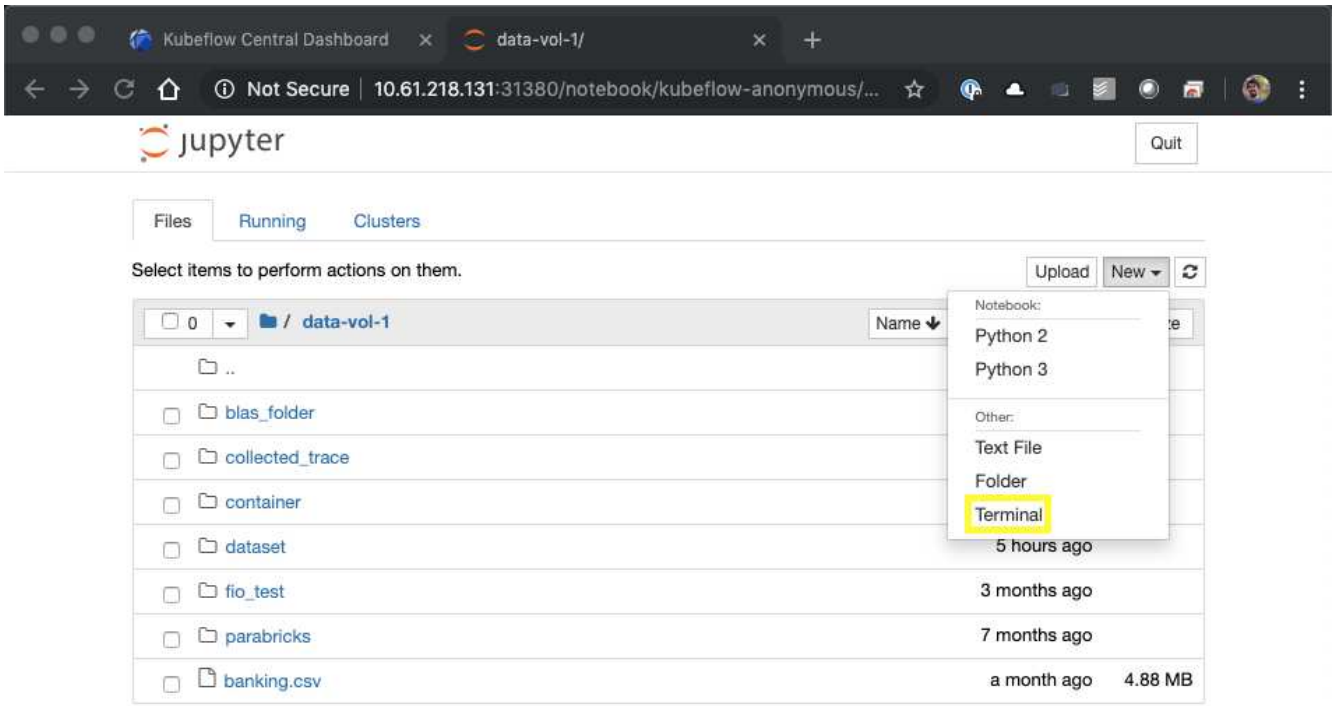

9. 单击连接以连接到新的服务器 Web 界面。
10. 确认步骤 6 中指定的数据集卷已挂载到服务器上。请注意，默认情况下，此卷会挂载在默认工作空间中。从用户的角度来看，这只是工作空间中的另一个文件夹。用户可能是数据科学家，而不是基础架构专家，因此使用此卷无需具备任何存储专业知识。






11. 打开一个终端，假设步骤 5 中请求了一个新卷，请执行 `df -h` 以确认已挂载新的 Trident 配置的永久性卷作为默认工作空间。

    默认工作空间目录是首次访问服务器的 Web 界面时显示的基目录。因此，使用 Web 界面创建的任何项目都会存储在此 Trident 配置的永久性卷上。



```
$ df -h
Filesystem                                                      Size  Used Avail
Use% Mounted on
overlay                                                         439G   34G  382G
  9% /
tmpfs                                                            64M     0   64M
  0% /dev
tmpfs                                                           252G     0  252G
  0% /sys/fs/cgroup
/dev/sda2                                                       439G   34G  382G
  9% /etc/hosts
192.168.11.11:/trident_pvc_3dcfe7e5_d5a9_11e9_9b9d_00505681a82d   10G  320K   10G
  1% /home/jovyan
tmpfs                                                           252G     0  252G
  0% /dev/shm
192.168.11.11:/pb_fg_all                                         10T   10T   47G
100% /home/jovyan/data-vol-1
tmpfs                                                           252G   12K  252G
  1% /run/secrets/kubernetes.io/serviceaccount
tmpfs                                                           252G   12K  252G
  1% /proc/driver/nvidia
tmpfs                                                            51G  4.9M   51G
  1% /run/nvidia-persistenced/socket
udev                                                            252G     0  252G
  0% /dev/nvidia5
tmpfs                                                           252G     0  252G
  0% /proc/acpi
tmpfs                                                           252G     0  252G
  0% /proc/scsi
tmpfs                                                           252G     0  252G
  0% /sys/firmware
$
```

12. 使用终端运行 `nvidia-smi` 以确认为笔记本电脑服务器分配了正确数量的 GPU 。在以下示例中，已按照步骤 7 中的请求为笔记本电脑服务器分配一个 GPU 。

```
$ nvidia-smi
Fri Sep 13 13:52:15 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.104      Driver Version: 410.104      CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:86:00.0 Off |                    0 |
| N/A   38C    P0    46W / 300W |      0MiB / 32480MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
$
```

示例笔记本电脑和管道

。 "适用于 Kubernetes 的 NetApp 数据科学工具包" 可与 Kubeflow 结合使用。将 NetApp 数据科学工具包与 Kubeflow 结合使用具有以下优势：

- 数据科学家可以直接在 Jupyter 笔记本电脑中执行高级 NetApp 数据管理操作。
- 可以使用 Kubeflow 管道框架将高级 NetApp 数据管理操作整合到自动化工作流中。

请参见 "Kubeflow 示例" 有关将工具包与 Kubeflow 结合使用的详细信息，请参见 NetApp Data Science Toolkit GitHub 存储库中的一节。

## Apache Airflow 部署

NetApp 建议在 Kubernetes 顶部运行 Apache Airflow 。本节介绍在 Kubernetes 集群中部署气流时必须完成的任务。

> 可以在 Kubernetes 以外的平台上部署 Airflow 。在 Kubernetes 以外的平台上部署气流不在此解决方案的范围内。

前提条件

在执行本节所述的部署练习之前，我们假定您已执行以下任务：

1. 您已有一个工作正常的 Kubernetes 集群。
2. 您已按照《 NetApp Trident 部署和配置》一节中所述在 Kubernetes 集群中安装和配置 NetApp Trident 。

安装 **Helm**

Airflow 可使用 Kubernetes 常用的软件包管理器 Helm 进行部署。在部署气流之前，必须在部署跳转主机上安装 Helm 。要在部署跳转主机上安装 Helm ，请按照 "安装说明" 在官方 Helm 文档中。

设置默认 **Kubernetes StorageClass**

在部署 Airflow 之前，您必须在 Kubernetes 集群中指定一个默认 StorageClass 。气流部署过程会尝试使用默认 StorageClass 配置新的永久性卷。如果未将任何 StorageClass 指定为默认 StorageClass ，则部署将失败。要在集群中指定默认 StorageClass ，请按照一节中所述的说明进行操作 "Kubeflow 部署"。如果已在集群中指定默认 StorageClass ，则可以跳过此步骤。

使用 **Helm** 部署气流

要使用 Helm 在 Kubernetes 集群中部署气流，请从部署跳转主机执行以下任务：

1. 按照说明使用 Helm 部署气流 "部署说明" 用于 Artifact Hub 上的官方气流图表。下面的示例命令显示了如何使用 Helm 部署气流。根据您的环境和所需配置，根据需要修改，添加和 / 或删除 `custom-values.yaml` 文件中的值。

```
$ cat << EOF > custom-values.yaml
##################################
```

```
# Airflow - Common Configs
####################################
airflow:
  ## the airflow executor type to use
  ##
  executor: "CeleryExecutor"
  ## environment variables for the web/scheduler/worker Pods (for
airflow configs)
  ##
  #
####################################
# Airflow - WebUI Configs
####################################
web:
  ## configs for the Service of the web Pods
  ##
  service:
    type: NodePort
####################################
# Airflow - Logs Configs
####################################
logs:
  persistence:
    enabled: true
####################################
# Airflow - DAGs Configs
####################################
dags:
  ## configs for the DAG git repository & sync container
  ##
  gitSync:
    enabled: true
    ## url of the git repository
    ##
    repo: "git@github.com:mboglesby/airflow-dev.git"
    ## the branch/tag/sha1 which we clone
    ##
    branch: master
    revision: HEAD
    ## the name of a pre-created secret containing files for ~/.ssh/
    ##
    ## NOTE:
    ## - this is ONLY RELEVANT for SSH git repos
    ## - the secret commonly includes files: id_rsa, id_rsa.pub,
known_hosts
    ## - known_hosts is NOT NEEDED if `git.sshKeyscan` is true
```

```
    ##
    sshSecret: "airflow-ssh-git-secret"
    ## the name of the private key file in your `git.secret`
    ##
    ## NOTE:
    ## - this is ONLY RELEVANT for PRIVATE SSH git repos
    ##
    sshSecretKey: id_rsa
    ## the git sync interval in seconds
    ##
    syncWait: 60
EOF
$ helm install airflow airflow-stable/airflow -n airflow --version 8.0.8
--values ./custom-values.yaml
...
Congratulations. You have just deployed Apache Airflow!
1. Get the Airflow Service URL by running these commands:
   export NODE_PORT=$(kubectl get --namespace airflow -o
jsonpath="{.spec.ports[0].nodePort}" services airflow-web)
   export NODE_IP=$(kubectl get nodes --namespace airflow -o
jsonpath="{.items[0].status.addresses[0].address}")
   echo http://$NODE_IP:$NODE_PORT/
2. Open Airflow in your web browser
```

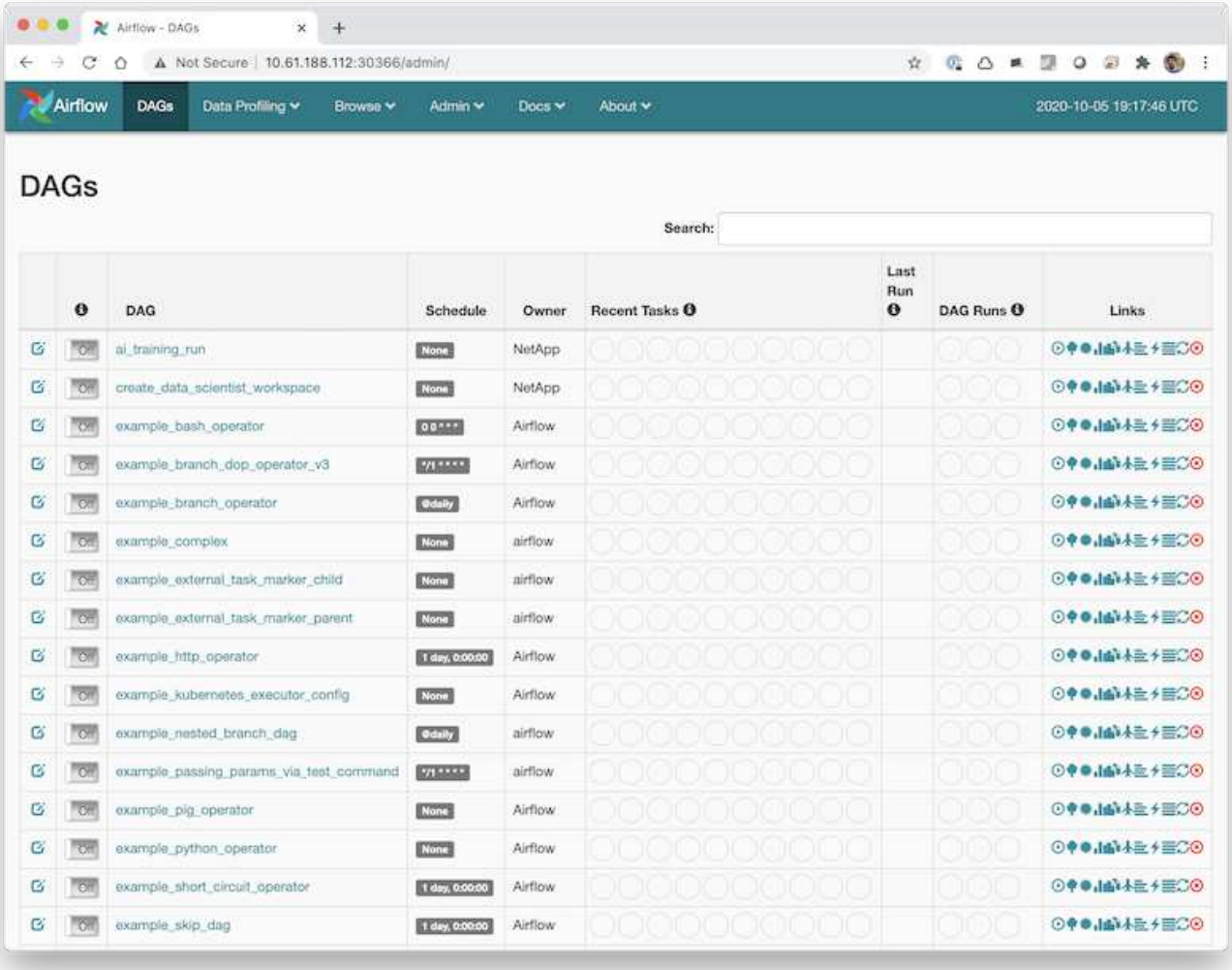
2. 确认所有气流 Pod 均已启动且正在运行。所有 POD 可能需要几分钟的时间才能启动。

```
$ kubectl -n airflow get pod
NAME                                READY   STATUS    RESTARTS   AGE
airflow-flower-b5656d44f-h8qjk      1/1     Running   0          2h
airflow-postgresql-0                1/1     Running   0          2h
airflow-redis-master-0              1/1     Running   0          2h
airflow-scheduler-9d95fcdf9-clf4b   2/2     Running   2          2h
airflow-web-59c94db9c5-z7rg4        1/1     Running   0          2h
airflow-worker-0                    2/2     Running   2          2h
```

3. 按照步骤 1 中使用 Helm 部署 Airflow 时控制台上印有的说明获取 Airflow Web 服务 URL 。

```
$ export NODE_PORT=$(kubectl get --namespace airflow -o
jsonpath="{.spec.ports[0].nodePort}" services airflow-web)
$ export NODE_IP=$(kubectl get nodes --namespace airflow -o
jsonpath="{.items[0].status.addresses[0].address}")
$ echo http://$NODE_IP:$NODE_PORT/
```

4. 确认您可以访问 Airflow Web 服务。




## Apache 气流工作流示例

。 "适用于 Kubernetes 的 NetApp 数据科学工具包" 可与气流结合使用。通过将 NetApp 数据科学工具包与 Airflow 结合使用，您可以将 NetApp 数据管理操作整合到由 Airflow 协调的自动化工作流中。

请参见 "气流示例" 有关在 Airflow 中使用工具包的详细信息，请参见 NetApp Data Science Toolkit GitHub 存储库中的一节。

## Trident 操作示例

本节包括您可能希望使用 Trident 执行的各种操作的示例。

### 导入现有卷

如果您的 NetApp 存储系统 / 平台上有要挂载到 Kubernetes 集群中的容器上但未与集群中的 PVC 绑定的现有卷，则必须导入这些卷。您可以使用 Trident 卷导入功能导入这些卷。

以下示例命令显示了为在本节示例中创建的每个 Trident 后端导入相同卷 `PB_FG_ALL` 两次 "ONTAP AI 部署的

Trident 后端示例"，步骤 1 。通过以这种方式导入同一卷两次，您可以在不同 LIF 之间多次挂载此卷（现有 FlexGroup 卷），如一节所述 "ONTAP AI 部署的 Trident 后端示例"，步骤 1 。有关 PVCs 的详细信息，请参见 "Kubernetes 官方文档"。有关卷导入功能的详细信息，请参见 "Trident 文档"。

在示例 PVC 规范文件中指定了 `accessModes` 值 `ReadOnlyMany` 。有关 `accessMode` 字段的详细信息，请参见 "Kubernetes 官方文档"。

> 以下示例导入命令中指定的后端名称与在本节的示例中创建的后端相对应 "ONTAP AI 部署的 Trident 后端示例"，步骤 1 。以下示例 PVC 定义文件中指定的 StorageClass 名称与在本节的示例中创建的 StorageClasses 相对应 "适用于 ONTAP AI 部署的 Kubernetes StorageClasses 示例"，步骤 1 。

```
$ cat << EOF > ./pvc-import-pb_fg_all-iface1.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pb-fg-all-iface1
  namespace: default
spec:
  accessModes:
    - ReadOnlyMany
  storageClassName: ontap-ai-flexgroups-retain-iface1
EOF
$ tridentctl import volume ontap-ai-flexgroups-iface1 pb_fg_all -f ./pvc-
import-pb_fg_all-iface1.yaml -n trident
+--------------------------------+--------
+-----------------------------------+----------
+-----------------------------------------------+--------+----------+
|          NAME                  |  SIZE  |       STORAGE CLASS
| PROTOCOL |             BACKEND UUID                          | STATE  |
MANAGED |
+--------------------------------+--------
+-----------------------------------+----------
+-----------------------------------------------+--------+----------+
| default-pb-fg-all-iface1-7d9f1 | 10 TiB | ontap-ai-flexgroups-retain-
iface1 | file     | b74cbddb-e0b8-40b7-b263-b6da6dec0bdd | online | true
|
+--------------------------------+--------
+-----------------------------------+----------
+-----------------------------------------------+--------+----------+
$ cat << EOF > ./pvc-import-pb_fg_all-iface2.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pb-fg-all-iface2
  namespace: default
spec:
```

```
  accessModes:
    - ReadOnlyMany
  storageClassName: ontap-ai-flexgroups-retain-iface2
EOF
$ tridentctl import volume ontap-ai-flexgroups-iface2 pb_fg_all -f ./pvc-
import-pb_fg_all-iface2.yaml -n trident
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
|          NAME                  |  SIZE  |       STORAGE CLASS
| PROTOCOL |             BACKEND UUID                        | STATE  |
MANAGED |
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
| default-pb-fg-all-iface2-85aee | 10 TiB | ontap-ai-flexgroups-retain-
iface2 | file     | 61814d48-c770-436b-9cb4-cf7ee661274d | online | true
|
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
$ tridentctl get volume -n trident
+-----------------------------------+---------
+-----------------------------------+----------
+--------------------------------------+--------+---------+
|               NAME               |  SIZE   |           STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------+---------
+-----------------------------------+----------
+--------------------------------------+--------+---------+
| default-pb-fg-all-iface1-7d9f1   | 10 TiB  | ontap-ai-flexgroups-retain-
iface1 | file     | b74cbddb-e0b8-40b7-b263-b6da6dec0bdd | online | true
|
| default-pb-fg-all-iface2-85aee   | 10 TiB  | ontap-ai-flexgroups-retain-
iface2 | file     | 61814d48-c770-436b-9cb4-cf7ee661274d | online | true
|
+-----------------------------------+---------
+-----------------------------------+----------
+--------------------------------------+--------+---------+
$ kubectl get pvc
NAME                 STATUS   VOLUME                                   CAPACITY
ACCESS MODES   STORAGECLASS                        AGE
pb-fg-all-iface1     Bound    default-pb-fg-all-iface1-7d9f1
10995116277760   ROX            ontap-ai-flexgroups-retain-iface1   25h
pb-fg-all-iface2     Bound    default-pb-fg-all-iface2-85aee
10995116277760   ROX            ontap-ai-flexgroups-retain-iface2   25h
```

配置新卷

您可以使用 Trident 在 NetApp 存储系统或平台上配置新卷。以下示例命令显示了新 FlexVol 卷的配置。在此示例中，使用在一节的示例中创建的 StorageClass 配置卷 "适用于 ONTAP AI 部署的 Kubernetes StorageClasses 示例"，步骤 2 。

在以下示例 PVC 定义文件中指定了 `accessModes` 值 `ReadWriteMany` 。有关 `accessMode` 字段的详细信息，请参见 "Kubernetes 官方文档"。

```
$ cat << EOF > ./pvc-tensorflow-results.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: tensorflow-results
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-ai-flexvols-retain
EOF
$ kubectl create -f ./pvc-tensorflow-results.yaml
persistentvolumeclaim/tensorflow-results created
$ kubectl get pvc
NAME                                STATUS    VOLUME
CAPACITY         ACCESS MODES   STORAGECLASS                        AGE
pb-fg-all-iface1                    Bound     default-pb-fg-all-iface1-7d9f1
10995116277760   ROX            ontap-ai-flexgroups-retain-iface1   26h
pb-fg-all-iface2                    Bound     default-pb-fg-all-iface2-85aee
10995116277760   ROX            ontap-ai-flexgroups-retain-iface2   26h
tensorflow-results                  Bound     default-tensorflow-results-
2fd60   1073741824       RWX            ontap-ai-flexvols-retain
25h
```

## ONTAP AI 部署的高性能作业示例

本节包括在 ONTAP AI POD 上部署 Kubernetes 时可以执行的各种高性能作业的示例。

### ONTAP AI 部署的高性能作业示例

本节包括在 ONTAP AI POD 上部署 Kubernetes 时可以执行的各种高性能作业的示例。

#### 执行单节点 AI 工作负载

要在 Kubernetes 集群中执行单节点 AI 和 ML 作业，请从部署跳转主机执行以下任务。借

助 Trident ，您可以快速轻松地使可能包含数 PB 数据的数据卷可供 Kubernetes 工作负载访问。要使此类数据卷可从 Kubernetes Pod 中访问，只需在 Pod 定义中指定 PVC 即可。此步骤是 Kubernetes 本机操作；不需要 NetApp 专业知识。

> 本节假定您已将尝试在 Kubernetes 集群中执行的特定 AI 和 ML 工作负载容器化（采用 Docker 容器格式）。

1. 以下示例命令显示了如何为使用 ImageNet 数据集的 TensorFlow 基准工作负载创建 Kubernetes 作业。有关 ImageNet 数据集的详细信息，请参见 "ImageNet 网站"。

   此示例作业请求八个 GPU ，因此可以在具有八个或更多 GPU 的单个 GPU 工作节点上运行。此示例作业可以在集群中提交，其中包含八个或更多 GPU 的工作节点不存在或当前占用另一个工作负载。如果是，则此作业将保持待定状态，直到此类辅助节点变为可用为止。

   此外，为了最大程度地提高存储带宽，包含所需训练数据的卷会在该作业创建的 POD 中挂载两次。此外，还会在 Pod 中挂载另一个卷。第二个卷将用于存储结果和指标。这些卷在作业定义中使用 PVC 的名称进行引用。有关 Kubernetes 作业的详细信息，请参见 "Kubernetes 官方文档"。

   在本示例作业创建的 Pod 中， `edium` 值为 `Memory` 的` emtyDir m卷将挂载到 ` `/dev/shm` 。Docker 容器运行时自动创建的 ` /dev/shm` 虚拟卷的默认大小有时可能不足以满足 TensorFlow 的需求。按以下示例所示挂载 `emptyDir` 卷可提供足够大的 ` /dev/shm` 虚拟卷。有关 `emptyDir` 卷的详细信息，请参见 "Kubernetes 官方文档"。

   在此示例作业定义中指定的单个容器将获得 `securityContext > privileged` 值 `true` 。此值表示容器在主机上具有有效的 root 访问权限。在这种情况下使用此标注是因为要执行的特定工作负载需要 root 访问权限。具体而言，工作负载执行的清除缓存操作需要 root 访问权限。是否需要此 特权： `true` 标注取决于您要执行的特定工作负载的要求。

```
$ cat << EOF > ./netapp-tensorflow-single-imagenet.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-single-imagenet
spec:
  backoffLimit: 5
  template:
    spec:
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
```

```
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--dgx_version=dgx1", "--
num_devices=8"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-single-imagenet.yaml
job.batch/netapp-tensorflow-single-imagenet created
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-single-imagenet           0/1           24s        24s
```

2. 确认您在步骤 1 中创建的作业正在正确运行。以下示例命令确认已按照作业定义中的指定为此作业创建了一个 POD ，并且此 POD 当前正在其中一个 GPU 工作节点上运行。

```
$ kubectl get pods -o wide
NAME                                                READY   STATUS
RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-single-imagenet-m7x92             1/1     Running     0
3m    10.233.68.61    10.61.218.154   <none>
```

3. 确认您在步骤 1 中创建的作业已成功完成。以下示例命令确认作业已成功完成。

```
$ kubectl get jobs
NAME                                          COMPLETIONS   DURATION
AGE
netapp-tensorflow-single-imagenet             1/1           5m42s
10m
$ kubectl get pods
NAME                                                  READY   STATUS
RESTARTS   AGE
netapp-tensorflow-single-imagenet-m7x92               0/1     Completed
0          11m
$ kubectl logs netapp-tensorflow-single-imagenet-m7x92
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-
PERMISSIONS in file gds_dstore.c at line 702
[netapp-tensorflow-single-imagenet-m7x92:00008] PMIX ERROR: NO-
PERMISSIONS in file gds_dstore.c at line 711
Total images/sec = 6530.59125
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 1 -H localhost:1 bash -c 'sync; echo 1 >
/proc/sys/vm/drop_caches'
==========================================
mpirun -allow-run-as-root -np 8 -H localhost:8 -bind-to none -map-by
slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000
--datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_105450_tensorflow_horovod_rdma_resnet50_gpu_8_256_b500_ima
genet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

4. * 可选： * 清理作业项目。以下示例命令显示了在步骤 1 中创建的作业对象的删除情况。

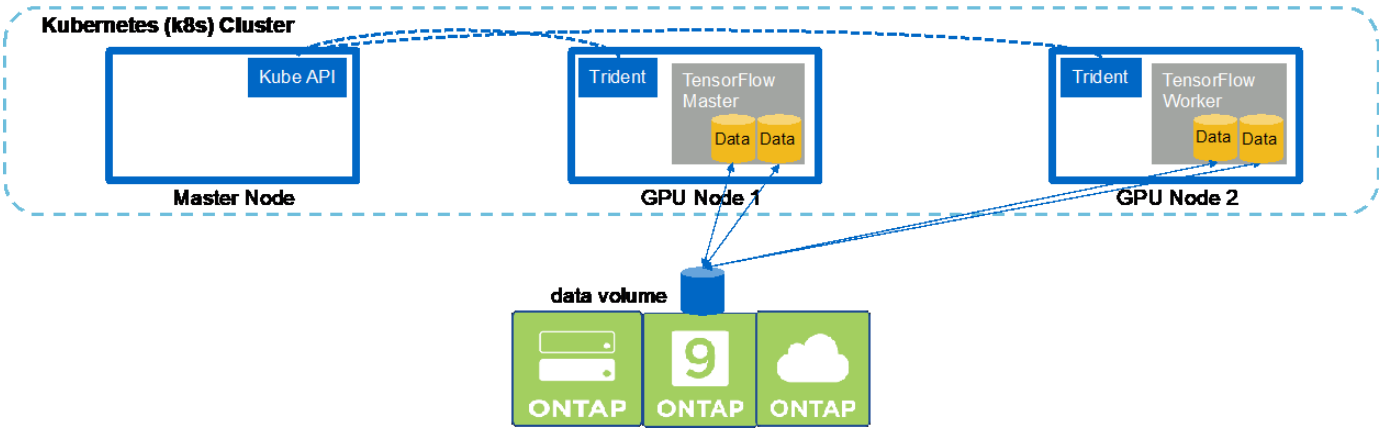
   删除作业对象时， Kubernetes 会自动删除任何关联的 Pod 。

```
$ kubectl get jobs
NAME                                                COMPLETIONS   DURATION
AGE
netapp-tensorflow-single-imagenet                   1/1           5m42s
10m
$ kubectl get pods
NAME                                                      READY   STATUS
RESTARTS   AGE
netapp-tensorflow-single-imagenet-m7x92                   0/1     Completed
0          11m
$ kubectl delete job netapp-tensorflow-single-imagenet
job.batch "netapp-tensorflow-single-imagenet" deleted
$ kubectl get jobs
No resources found.
$ kubectl get pods
No resources found.
```

执行同步分布式 **AI** 工作负载

要在 Kubernetes 集群中执行同步多节点 AI 和 ML 作业，请在部署跳转主机上执行以下任务。通过此过程，您可以利用存储在 NetApp 卷上的数据，并使用单个工作节点所能提供的 GPU 。有关同步分布式 AI 作业的描述，请参见下图。

> 与异步分布式作业相比，同步分布式作业有助于提高性能和训练准确性。本文档不会讨论同步作业与异步作业的利弊。




1. 以下示例命令显示了创建一名员工参与同步分布式执行本节中示例中在单个节点上执行的同一 TensorFlow 基准测试作业的过程 "执行单节点 AI 工作负载"。在此特定示例中，仅部署一个员工，因为此作业会在两个员工节点上执行。

   此示例员工部署请求八个 GPU ，因此可以在一个 GPU 工作节点上运行，该节点具有八个或更多 GPU 。如果您的 GPU 工作节点具有八个以上的 GPU ，则为了最大限度地提高性能，您可能需要增加此数量，使其等于您的工作节点所具有的 GPU 数量。有关 Kubernetes 部署的详细信息，请参见 "Kubernetes 官方文档"。

在此示例中创建了 Kubernetes 部署，因为此特定容器化员工永远不会自行完成。因此，使用 Kubernetes 作业构造来部署它毫无意义。如果员工的设计或编写是为了自己完成，则可以使用此作业构建来部署员工。

在此示例部署规范中指定的 Pod 的值为 `hostNetwork` 值 `true` 。此值表示 Pod 使用主机工作节点的网络堆栈，而不是 Kubernetes 通常为每个 Pod 创建的虚拟网络堆栈。在这种情况下使用此标注是因为特定工作负载依靠 Open MPI ， NCCL 和 Horovod 以同步分布式方式执行工作负载。因此，它需要访问主机网络堆栈。有关 Open MPI ， NCCL 和 Horovod 的讨论不在本文档的讨论范围之内。是否需要此 `hostNetwork： true` 标注取决于要执行的特定工作负载的要求。有关 `hostNetwork` 字段的详细信息，请参见 "Kubernetes 官方文档"。

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-worker.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: netapp-tensorflow-multi-imagenet-worker
spec:
  replicas: 1
  selector:
    matchLabels:
      app: netapp-tensorflow-multi-imagenet-worker
  template:
    metadata:
      labels:
        app: netapp-tensorflow-multi-imagenet-worker
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["bash", "/netapp/scripts/start-slave-multi.sh",
"22122"]
        resources:
          limits:
            nvidia.com/gpu: 8
```

```
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-worker.yaml
deployment.apps/netapp-tensorflow-multi-imagenet-worker created
$ kubectl get deployments
NAME                                      DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           4s
```

2. 确认您在第 1 步中创建的员工部署已成功启动。以下示例命令确认已为部署创建了一个辅助 POD ，如部署定义所示，并且此 POD 当前正在其中一个 GPU 辅助节点上运行。

```
$ kubectl get pods -o wide
NAME                                                        READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running   0          60s   10.61.218.154   10.61.218.154   <none>
$ kubectl logs netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725
22122
```

3. 为启动，参与并跟踪同步多节点作业执行的主节点创建 Kubernetes 作业。以下示例命令创建一个主节点，用于启动，参与和跟踪在一节中的示例中对单个节点执行的相同 TensorFlow 基准测试作业的同步分布式执行 "执行单节点 AI 工作负载"。

   此示例主作业请求八个 GPU ，因此可以在具有八个或更多 GPU 的单个 GPU 工作节点上运行。如果您的 GPU 工作节点具有八个以上的 GPU ，则为了最大限度地提高性能，您可能需要增加此数量，使其等于您的工作节点所具有的 GPU 数量。

   在本示例作业定义中指定的主 Pod 的值为 `hostNetwork` 值为 `true` ，就像在步骤 1 中为工作 Pod 提供了 `hostNetwork` 值 `true` 一样。有关为何需要此值的详细信息，请参见第 1 步。

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-master.yaml
apiVersion: batch/v1
kind: Job
```

```
metadata:
  name: netapp-tensorflow-multi-imagenet-master
spec:
  backoffLimit: 5
  template:
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--port=22122", "--
num_devices=16", "--dgx_version=dgx1", "--
nodes=10.61.218.152,10.61.218.154"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-master.yaml
job.batch/netapp-tensorflow-multi-imagenet-master created
$ kubectl get jobs
```

```
NAME                                          COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   0/1           25s        25s
```

4. 确认您在步骤 3 中创建的主作业正在正确运行。以下示例命令确认已为作业创建了一个主 Pod ，如作业定义所示，并且此 Pod 当前正在其中一个 GPU 工作节点上运行。您还应看到，您最初在步骤 1 中看到的辅助 POD 仍在运行，并且主节点和辅助节点正在不同的节点上运行。

```
$ kubectl get pods -o wide
NAME                                                          READY
STATUS    RESTARTS   AGE
IP              NODE            NOMINATED NODE
netapp-tensorflow-multi-imagenet-master-ppwwj               1/1
Running   0          45s   10.61.218.152   10.61.218.152   <none>
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running   0          26m   10.61.218.154   10.61.218.154   <none>
```

5. 确认您在步骤 3 中创建的主作业已成功完成。以下示例命令确认作业已成功完成。

```
$ kubectl get jobs
NAME                                          COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      9m18s
$ kubectl get pods
NAME                                                          READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj               0/1
Completed   0          9m38s
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          35m
$ kubectl logs netapp-tensorflow-multi-imagenet-master-ppwwj
[10.61.218.152:00008] WARNING: local probe returned unhandled
shell:unknown assuming bash
rm: cannot remove '/lib': Is a directory
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
Total images/sec = 12881.33875
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 2 -H 10.61.218.152:1,10.61.218.154:1 -mca
pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" bash -c 'sync; echo 1 >
```

```
/proc/sys/vm/drop_caches'
=========================================
mpirun -allow-run-as-root -np 16 -H 10.61.218.152:8,10.61.218.154:8
-bind-to none -map-by slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH
-mca pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -x
NCCL_IB_HCA=mlx5 -x NCCL_NET_GDR_READ=1 -x NCCL_IB_SL=3 -x
NCCL_IB_GID_INDEX=3 -x
NCCL_SOCKET_IFNAME=enp5s0.3091,enp12s0.3092,enp132s0.3093,enp139s0.3094
-x NCCL_IB_CUDA_SUPPORT=1 -mca orte_base_help_aggregate 0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000 --
datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_161609_tensorflow_horovod_rdma_resnet50_gpu_16_256_b500_im
agenet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

6. 如果您不再需要此员工部署，请将其删除。以下示例命令显示了删除在步骤 1 中创建的工作部署对象的过程。

   删除 worker 部署对象时， Kubernetes 会自动删除任何关联的 worker Pod 。

```
$ kubectl get deployments
NAME                                        DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           43m
$ kubectl get pods
NAME                                                        READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj              0/1
Completed   0          17m
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          43m
$ kubectl delete deployment netapp-tensorflow-multi-imagenet-worker
deployment.extensions "netapp-tensorflow-multi-imagenet-worker" deleted
$ kubectl get deployments
No resources found.
$ kubectl get pods
NAME                                              READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj   0/1     Completed   0
18m
```

7. * 可选： * 清理主作业项目。以下示例命令显示了删除在步骤 3 中创建的主作业对象的过程。

   删除主作业对象时， Kubernetes 会自动删除任何关联的主 Pod 。

```
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      19m
$ kubectl get pods
NAME                                              READY   STATUS
RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj   0/1     Completed   0
19m
$ kubectl delete job netapp-tensorflow-multi-imagenet-master
job.batch "netapp-tensorflow-multi-imagenet-master" deleted
$ kubectl get jobs
No resources found.
$ kubectl get pods
No resources found.
```

### 性能测试

在创建此解决方案时，我们执行了简单的性能比较。我们使用 Kubernetes 执行了多个标准 NetApp AI 基准测试作业，并将基准测试结果与使用简单 Docker run 命令执行的执行情况进行了比较。我们没有发现任何明显的性能差异。因此，我们得出的结论是，使用 Kubernetes 编排容器化 AI 培训作业不会对性能产生负面影响。有关性能比较结果，请参见下表。

| 基准测试 | 数据集 | **Docker** 运行（映像 / 秒） | **Kubernetes** （图像 / 秒） |
|---|---|---|---|
| 单节点 TensorFlow | 合成数据 | 6,667.2475 | 6,661.93125 |
| 单节点 TensorFlow | ImageNet | 6,570.2025 | 6,530.59125 |
| 同步分布式双节点 TensorFlow | 合成数据 | 13,213.70625 | 13,218.288125 |
| 同步分布式双节点 TensorFlow | ImageNet | 12,941.69125 | 12,881.33875 |

### 结论

各行各业各种规模的企业和组织都在转向人工智能（ AI ），机器学习（ ML ）和深度学习（ DL ），以解决实际问题，提供创新产品和服务，并在竞争日益激烈的市场中占据优势。随着企业越来越多地使用 AI ， ML 和 DL ，他们面临着许多挑战，包括工作负载可扩展性和数据可用性。可以通过使用 NetApp AI 控制平台解决方案来应对这些挑战。

通过此解决方案，您可以快速克隆数据命名空间。此外，您还可以定义和实施 AI ， ML 和 DL 培训工作流，这些工作流可近乎即时地创建数据和模型基线，以实现可追溯性和版本控制。使用此解决方案，您可以跟踪每个模型训练返回到模型经过训练和 / 或验证的确切数据集。最后，借助此解决方案，您可以快速配置 Jupyter 笔记本电脑工作空间，以访问海量数据集。

由于此解决方案面向数据科学家和数据工程师，因此只需极少的 NetApp 或 NetApp ONTAP 专业知识即可。借助此解决方案，可以使用简单熟悉的工具和界面来执行数据管理功能。此外，此解决方案还利用完全开源和免费的组件。因此，如果您的环境中已有 NetApp 存储，则可以立即实施此解决方案。如果您要测试此解决方案的驱动器，但尚未安装 NetApp 存储，请访问 "cloud.netapp.com"，您可以随时使用基于云的 NetApp 存储解决方案启动和运行。

# 与 Iguazio 的 MLRun 管道

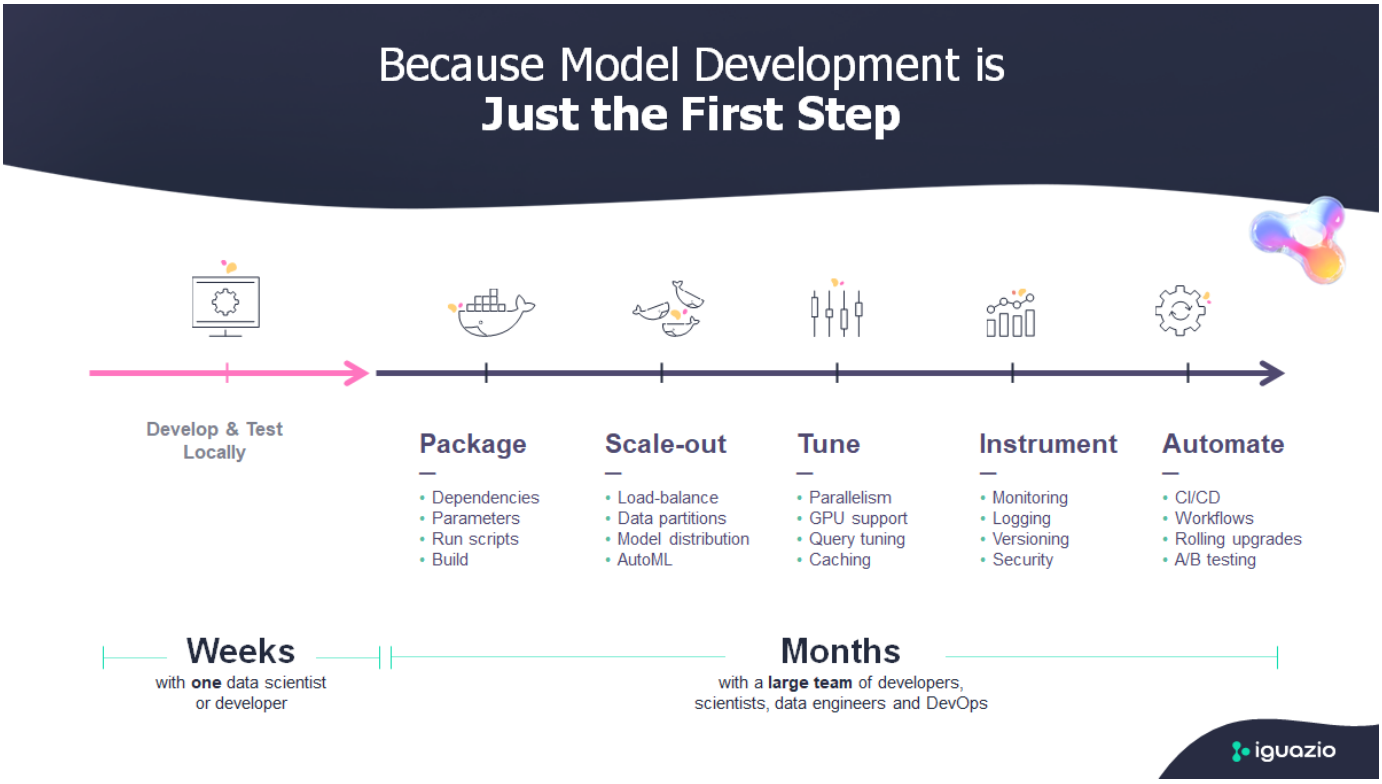
## TR-4834：《NetApp 和 Iguazio for MLRun 管道》

Rick Huang， David Arnette， NetApp Marcelo LitovskY， Iguazio

本文档介绍了使用 NetApp ONTAP AI ， NetApp AI 控制平台， NetApp Cloud Volumes 软件和 Iguazio 数据科学平台的 MLRun 管道的详细信息。我们使用的是 Nercio 无服务器功能， Kubernetes 永久性卷， NetApp Cloud Volumes ， NetApp Snapshot 副本， Grafana 信息板， 以及 Iguazio 平台上的其他服务，用于构建端到端数据管道，以模拟网络故障检测。我们集成了 Iguazio 和 NetApp 技术，可在内部和云端实现快速的模型部署

，数据复制和生产监控功能。

数据科学家的工作重点应放在机器学习（ML）和人工智能（AI）模型的培训和调整上。但是，根据 Google 的研究，数据科学家花费了 ~80% 的时间来研究如何使其模型能够与企业应用程序结合使用并大规模运行，如以下描述 AI/ML 工作流中模型开发的图像所示。




要管理端到端 AI/ML 项目，需要更广泛地了解企业组件。虽然 DevOps 已接管这些类型的组件的定义，集成和部署，但机器学习操作的目标是类似的流程，其中包括 AI/ML 项目。要了解端到端 AI/ML 管道在企业中涉及的内容，请参见以下所需组件列表：

- 存储
- 网络
- 数据库
- 文件系统
- 容器
- 持续集成和持续部署（CI/CD）管道
- 开发集成开发环境（IDE）
- 安全性
- 数据访问策略
- 硬件
- 云
- 虚拟化
- 数据科学工具集和库

在本白皮书中，我们展示了 NetApp 与 Iguazio 之间的合作关系如何显著简化端到端 AI/ML 管道的开发。这种简化可以加快所有 AI/ML 应用程序的上市速度。

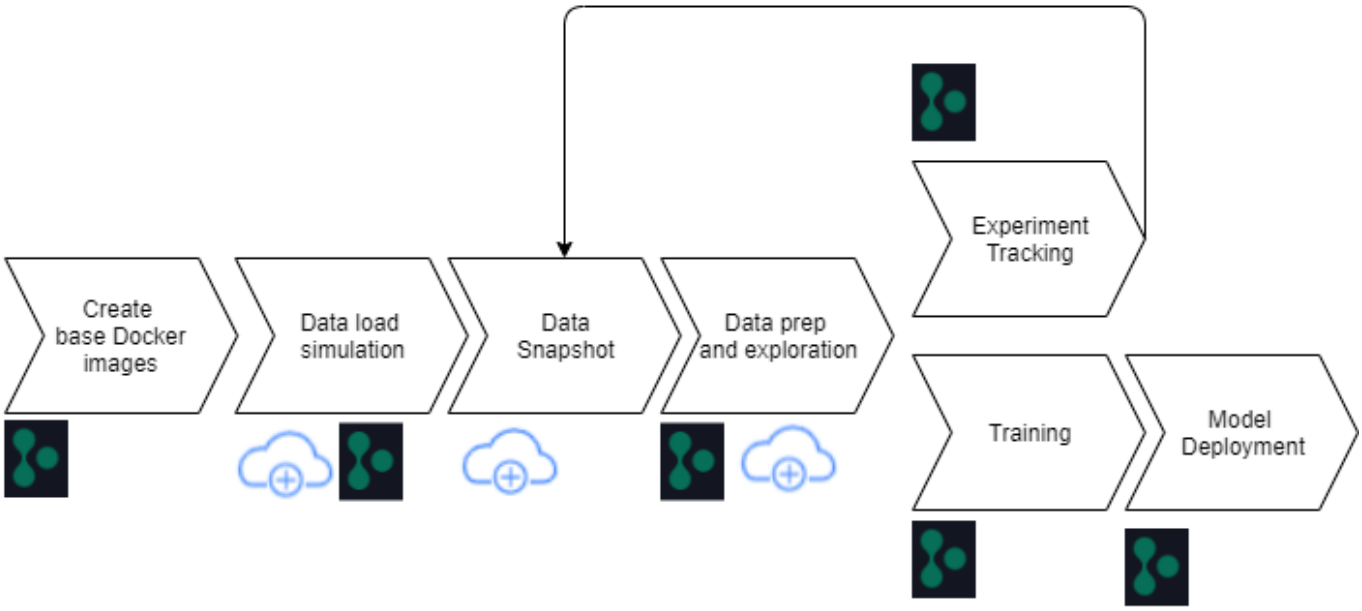
目标受众

数据科学领域涉及信息技术和业务领域的多个学科。

- 数据科学家需要灵活地使用自己选择的工具和库。
- 数据工程师需要了解数据的流动方式及其所在位置。
- DevOps 工程师需要使用工具将新的 AI/ML 应用程序集成到其 CI/CD 管道中。
- 业务用户希望能够访问 AI/ML 应用程序。我们介绍了 NetApp 和 Iguazio 如何帮助这些角色为我们的平台带来业务价值。

解决方案概述

此解决方案遵循 AI/ML 应用程序的生命周期。我们从数据科学家的工作开始，定义准备数据以及训练和部署模型所需的不同步骤。接下来，我们将完成创建完整管道所需的工作，该管道能够跟踪项目，试验执行并部署到 Kubeflow 。为了完成整个周期，我们将管道与 NetApp Cloud Volumes 集成，以启用数据版本控制，如下图所示。




iguazio

Cloud Volumes ONTAP

## 技术概述

**NetApp** 概述

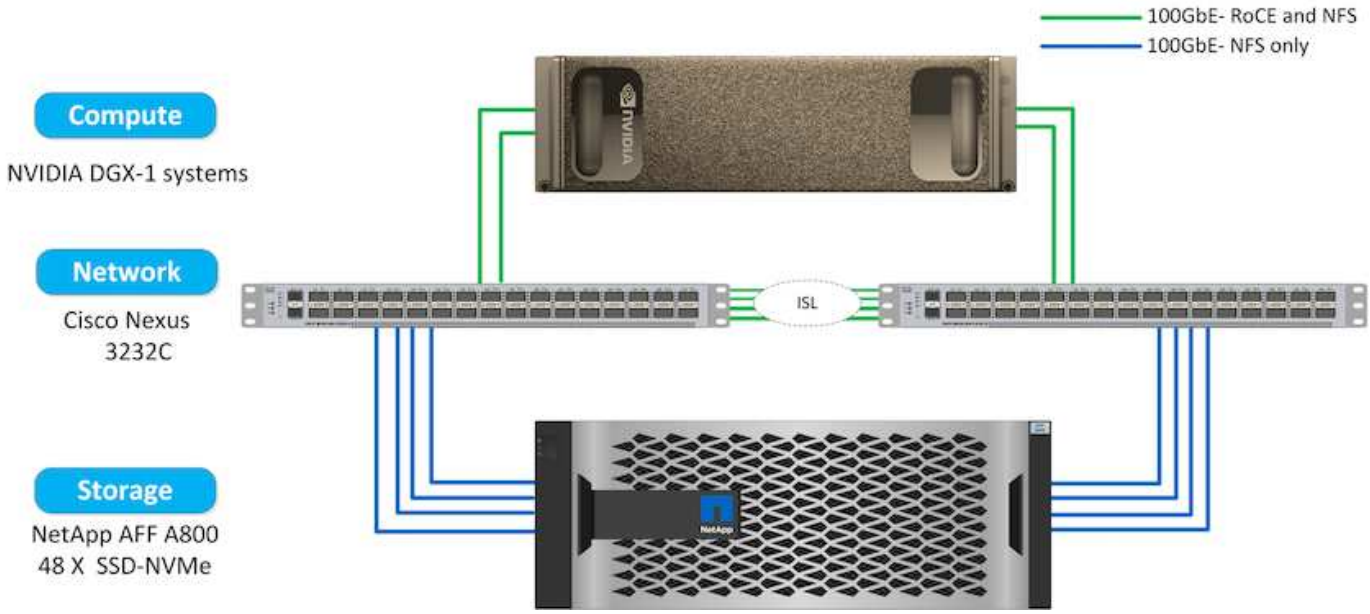
NetApp 是混合云数据管理领域的权威企业。NetApp 提供全套混合云数据服务，可简化云和内部环境中的应用程序和数据管理，加速数字化转型。NetApp 与我们的合作伙伴携手，赋予全球企业充分释放数据的全部潜能，

以扩大客户接触点，促进更大的创新并优化运营的能力。

### NetApp ONTAP AI

NetApp ONTAP AI 由 NVIDIA DGX 系统和 NetApp 云互联全闪存存储提供支持，可可靠地简化数据流，并加快从边缘到核心再到云的数据网络结构的分析，培训和推理速度。它为 IT 组织提供了一个架构，可提供以下优势：

- 消除设计复杂性
- 支持独立扩展计算和存储
- 支持客户从小规模入手，无缝扩展
- 为各种性能和成本点提供一系列存储选项 NetApp ONTAP AI 提供融合基础架构堆栈，其中包括 NVIDIA DGX-1 ，一个 petaflop 级 AI 系统和 NVIDIA Mellanox 高性能以太网交换机，可统一 AI 工作负载，简化部署并加快 ROI 。在本技术报告中，我们将 ONTAP AI 与一个 DGX-1 和一个 NetApp AFF A800 存储系统结合使用。下图显示了此验证中使用的 DGX-1 系统的 ONTAP AI 拓扑。




### NetApp AI 控制平台

借助 NetApp AI 控制平台，您可以借助解决方案充分发挥 AI 和 ML 的潜能，该平台可提供极高的可扩展性，简化的部署以及无中断的数据可用性。AI 控制平面解决方案将 Kubernetes 和 Kubeflow 与 NetApp 支持的数据网络结构相集成。Kubernetes 是适用于云原生部署的行业标准容器编排平台，可实现工作负载的可扩展性和可移动性。Kubeflow 是一款开源机器学习平台，可简化管理和部署，使开发人员能够在更短的时间内完成更多的数据科学工作。NetApp 支持的 Data Fabric 可提供无与伦比的数据可用性和可移植性，确保您的数据可通过管道从边缘到核心再到云进行访问。本技术报告在 MLRun 管道中使用 NetApp AI 控制平台。下图显示了 Kubernetes 集群管理页面，您可以在其中为每个集群设置不同的端点。我们将 NFS 永久性卷连接到 Kubernetes 集群，下图显示了连接到集群的永久性卷，其中 "NetApp Trident" 提供持久存储支持和数据管理功能。

Kubernetes Clusters

Discover Cluster

4 Kubernetes Clusters

kubernetes

https://3.20.111.39:6443
Cluster Endpoint

v1.15.5
Cluster Version

19.07.1
Trident Version

0
Working Environments

kubernetes

https://172.31.14.31:6443
Cluster Endpoint

v1.15.5
Cluster Version

19.07.1
Trident Version

1
Working Environments

## Persistent Volumes for Kubernetes

### Connected with Kubernetes Cluster

Cloud Volumes ONTAP is connected to 1 Kubernetes cluster. View Cluster

You can connect another Kubernetes cluster to this Cloud Volumes ONTAP system. If the Kubernetes cluster is in a different network than Cloud Volumes ONTAP, specify a custom export policy to provide access to clients.

### Kubernetes Cluster

Select Kubernetes Cluster

kubernetes

### Custom Export Policy *(Optional)*

Custom Export Policy

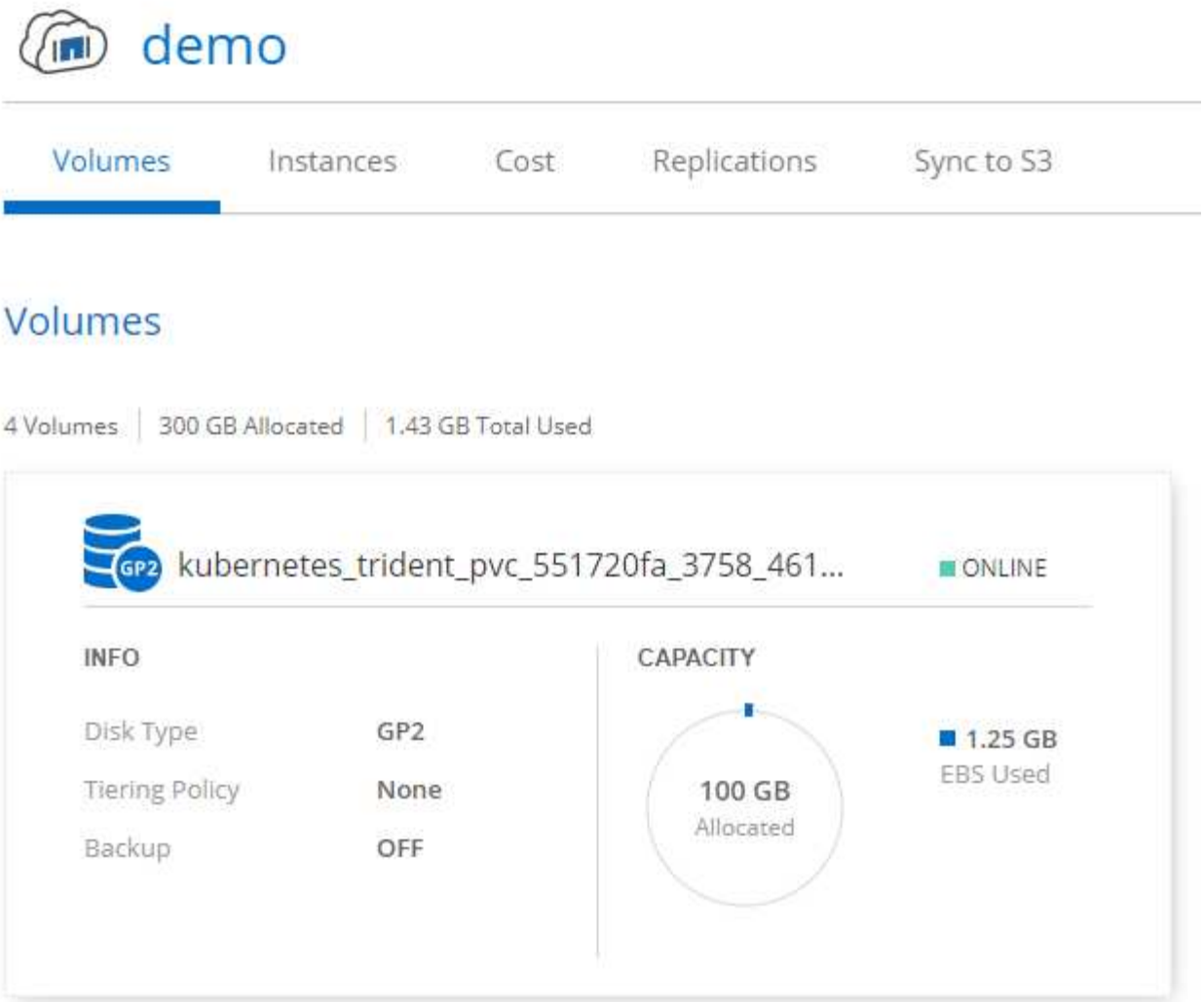
172.31.0.0/16

☑ Set as default storage class

◉ NFS ○ iSCSI

Connect

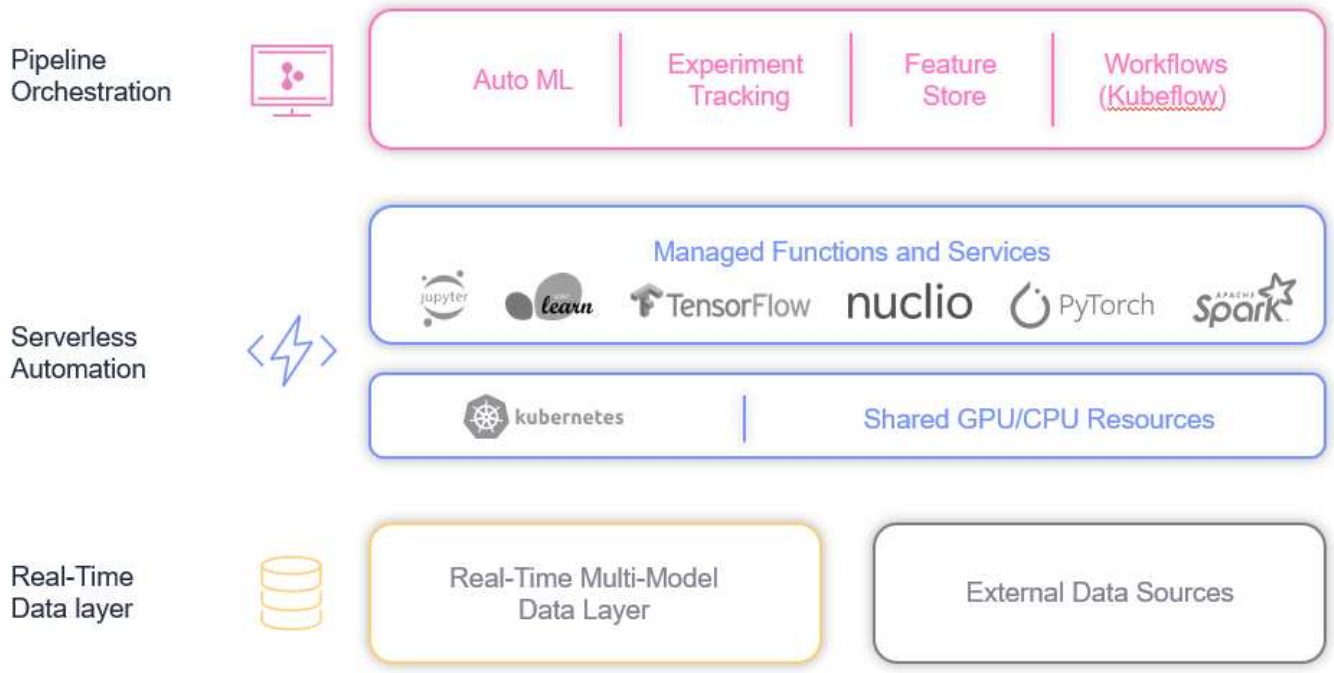
Cancel




**Iguazio** 概述

Iguazio 数据科学平台是一个完全集成且安全的数据科学平台即服务（ PaaS ），可简化开发，加快性能，促进协作并解决运营难题。此平台包含以下组件， Iguazio 数据科学平台如下图所示：

- 数据科学工作台，包括 Jupyter 笔记本电脑，集成分析引擎和 Python 软件包
- 通过试验跟踪和自动化管道功能进行模型管理
- 通过可扩展的 Kubernetes 集群管理数据和 ML 服务
- Nutrio ，一种实时无服务器功能框架
- 一个速度极快且安全的数据层，支持 SQL ， NoSQL ，时间序列数据库，文件（简单对象）和流式传输
- 与 NetApp ， Amazon S3 ， HDFS ， SQL 数据库以及流式传输或消息传送协议等第三方数据源集成
- 基于 Grafana 的实时信息板




## 软件和硬件要求

网络配置：

以下是在云中设置的网络配置要求：

- Iguazio 集群和 NetApp Cloud Volumes 必须位于同一个虚拟私有云中。
- 云管理器必须有权访问 Iguazio 应用程序节点上的端口 6443 。
- 我们在本技术报告中使用了 Amazon Web Services 。但是，用户可以选择在任何云提供商中部署解决方案。为了便于在采用 NVIDIA DGX-1 的 ONTAP AI 中进行内部测试，我们使用了 Iguazio 托管的 DNS 服务。

客户端必须能够访问动态创建的 DNS 域。如果需要，客户可以使用自己的 DNS 。

### 硬件要求

您可以在自己的集群中安装 Iguazio 内部部署。我们已使用 NVIDIA DGX-1 系统验证了 NetApp ONTAP AI 中的解决方案。下表列出了用于测试此解决方案的硬件。

| 硬件 | 数量 |
| --- | --- |
| DGX-1 系统 | 1. |
| NetApp AFF A800 系统 | 1 个高可用性（HA）对，包括 2 个控制器和 48 个 NVMe SSD （3.8 TB 或更高） |
| Cisco Nexus 3232C 网络交换机 | 2. |

下表列出了内部测试所需的软件组件：

| 软件 | 版本或其他信息 |
| --- | --- |
| NetApp ONTAP 数据管理软件 | 9.7 |
| Cisco NX-OS 交换机固件 | 7.0 （3） I6 （1） |
| NVIDIA DGX 操作系统 | 4.4 — Ubuntu 18.04 LTS |
| Docker 容器平台 | 19.03.5 |
| 容器版本 | 20.01-tF1-py2. |
| 机器学习框架 | TensorFlow 1.15.0 |
| Iguazio | 版本 2.8+ |
| ESX 服务器 | 6.5 |

此解决方案已通过 Iguazio 2.5 版和适用于 AWS 的 NetApp Cloud Volumes ONTAP 的全面测试。Iguazio 集群和 NetApp 软件均在 AWS 上运行。

| 软件 | 版本或类型 |
| --- | --- |
| Iguazio | 版本 2.8+ |
| 应用程序节点 | m5.4xlarge |
| 数据节点 | I3.4 x 大型 |

## 网络设备故障预测用例摘要

本用例基于亚洲电信领域的 Iguazio 客户。每年有 10 万个企业客户和 125 万个网络中断事件，因此迫切需要预测并采取主动行动，防止网络故障影响客户。此解决方案为他们提供了以下优势：

- 对网络故障进行预测性分析
- 与票证系统集成
- 通过主动采取措施，防止因实施 Iguazio 而导致网络故障， 60% 的故障都是主动预防的。

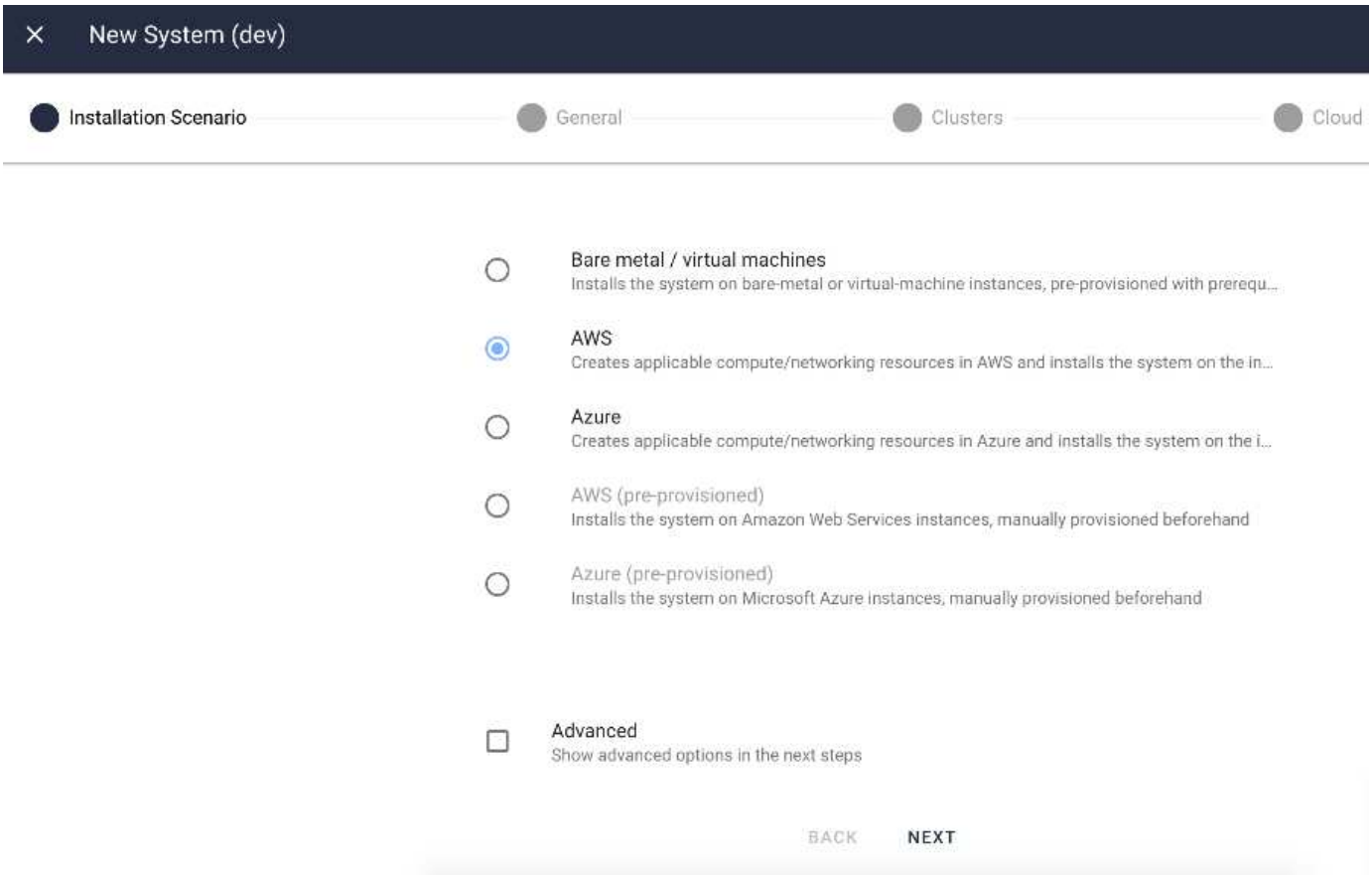
## 设置概述

Iguazio 可以安装在内部或云提供商上。

### 安装 **Iguazio**

配置可以作为一项服务来完成，并由 Iguazio 或客户进行管理。在这两种情况下， Iguazio 都会提供一个部署应用程序（ Provazio ）来部署和管理集群。

有关内部安装，请参见 "NVA-1121" 用于计算，网络和存储设置。Iguazio 的内部部署由 Iguazio 提供，客户无需支付额外费用。请参见 "此页面" DNS 和 SMTP 服务器配置。Provazio 安装页面如下所示。



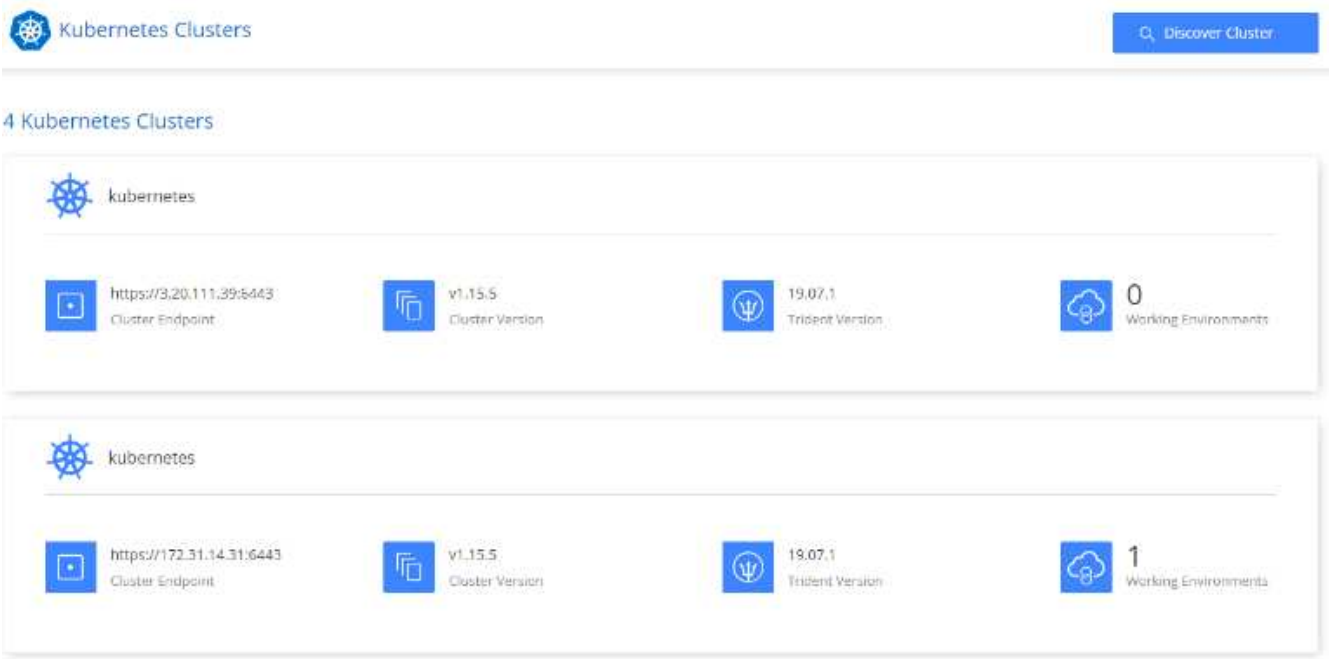

正在配置 **Kubernetes** 集群

## 本节将分别分为两部分，分别用于云和内部部署。

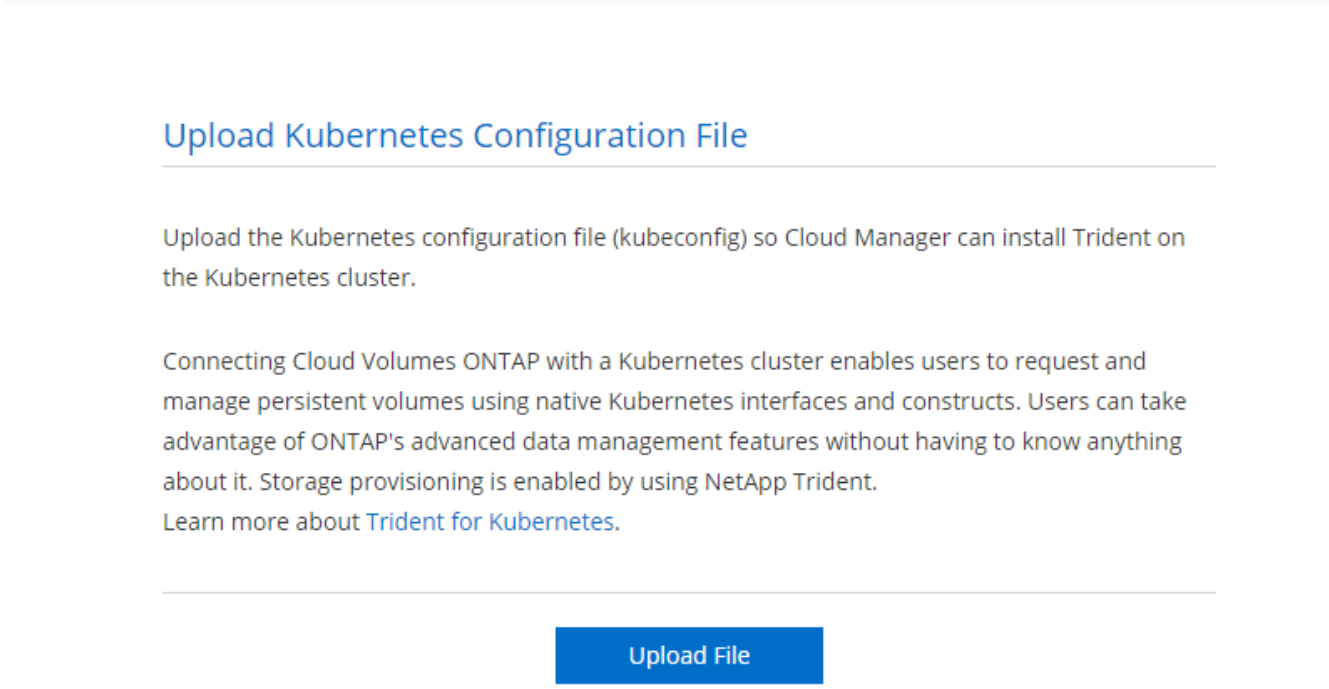
云部署 **Kubernetes** 配置

通过 NetApp Cloud Manager ，您可以定义与 Iguazio Kubernetes 集群的连接。要使卷可用， Trident 需要访问集群中的多个资源。

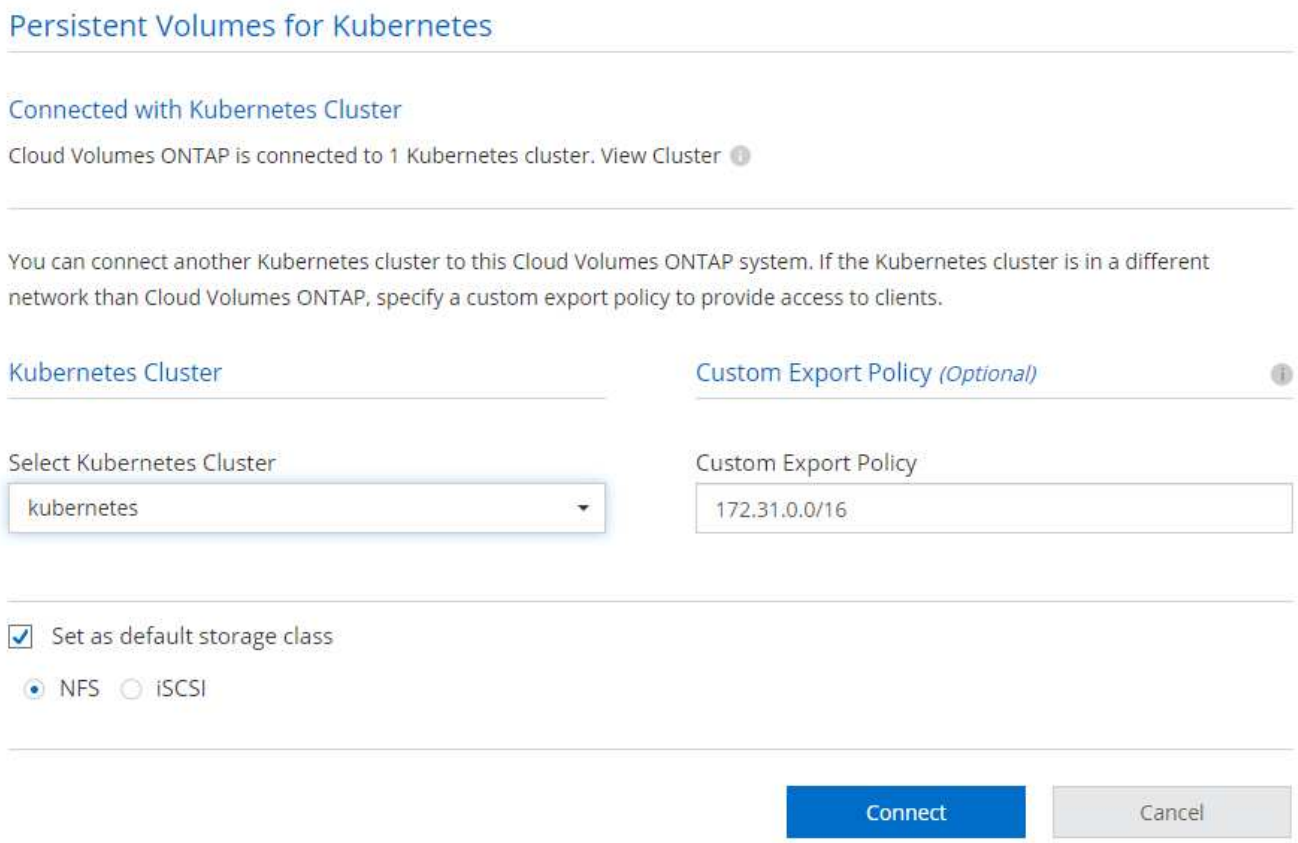
1. 要启用访问，请从一个 Iguazio 节点获取 Kubernetes 配置文件。该文件位于 ` /home/Iguazio/.Kube/config 下。` 将此文件下载到桌面。
2. 转至 Discover Cluster 以进行配置。




3. 上传 Kubernetes 配置文件。请参见下图。




4. 部署 Trident 并将卷与集群相关联。有关定义永久性卷并将其分配给 Iguazio 集群的信息，请参见下图。此过程将在 Iguazio 的 Kubernetes 集群中创建永久性卷（PV）。在使用它之前，您必须定义永久性卷声明（PVC）。




内部部署 **Kubernetes** 配置

有关 NetApp Trident 的内部安装，请参见 "TR-4798" 了解详细信息。配置 Kubernetes 集群并安装 NetApp Trident 后，您可以将 Trident 连接到 Iguazio 集群以启用 NetApp 数据管理功能，例如为数据和型号创建 Snapshot 副本。

定义永久性卷声明

1. 将以下 YAML 保存到文件中，以创建类型为 Basic 的 PVC 。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: netapp-file
```

2. 将 YAML 文件应用于您的 Iguazio Kubernetes 集群。

```
Kubectl -n default-tenant apply -f <your yaml file>
```

#### 将 NetApp 卷附加到 Jupyter 笔记本电脑

Iguazio 提供多种托管服务，为数据科学家提供完整的端到端堆栈，用于开发和部署 AI/ML 应用程序。有关这些组件的详细信息，请参见 "Iguazio 应用程序服务和工具概述"。

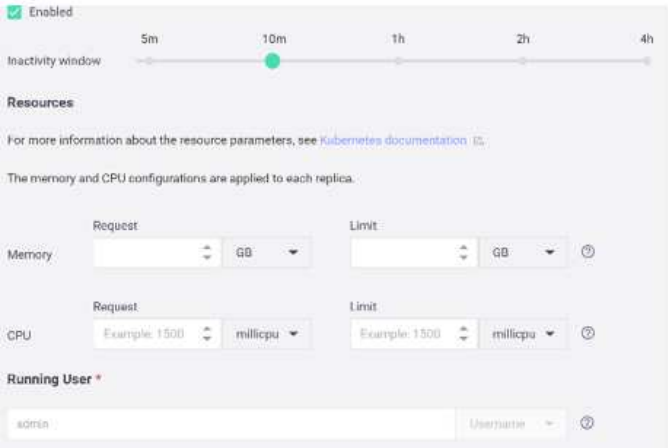
其中一项托管服务是 Jupyter Notebook 。每个开发人员都可以使用开发所需的资源自行部署一个笔记本容器。要授予他们对 NetApp Cloud Volume 的访问权限，您可以将卷分配给容器，并在下图中显示了永久性卷声明的正在运行的用户和环境变量设置。

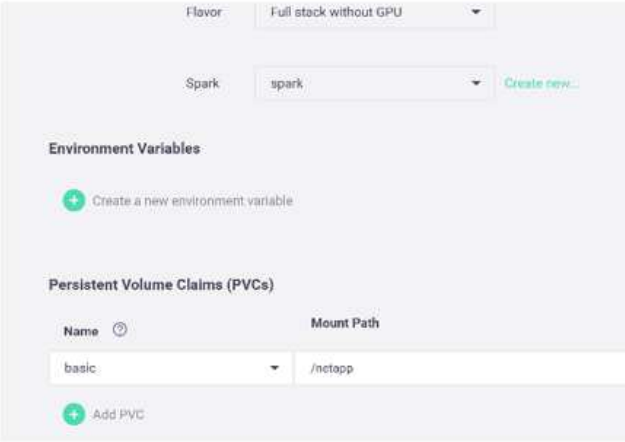
对于内部配置，您可以参见 "TR-4798" 在 Trident 设置中启用 NetApp ONTAP 数据管理功能，例如为数据或模型创建 Snapshot 副本以进行版本控制。在 Trident 后端配置文件中添加以下行，以使 Snapshot 目录可见：

```
{
    …
    "defaults": {
        "snapshotDir": "true"
    }
}
```

您必须以 JSON 格式创建 Trident 后端配置文件，然后运行以下命令 "Trident 命令" 要参考此指南：

```
tridentctl create backend -f <backend-file>
```




## 部署应用程序

以下各节介绍了如何安装和部署应用程序。

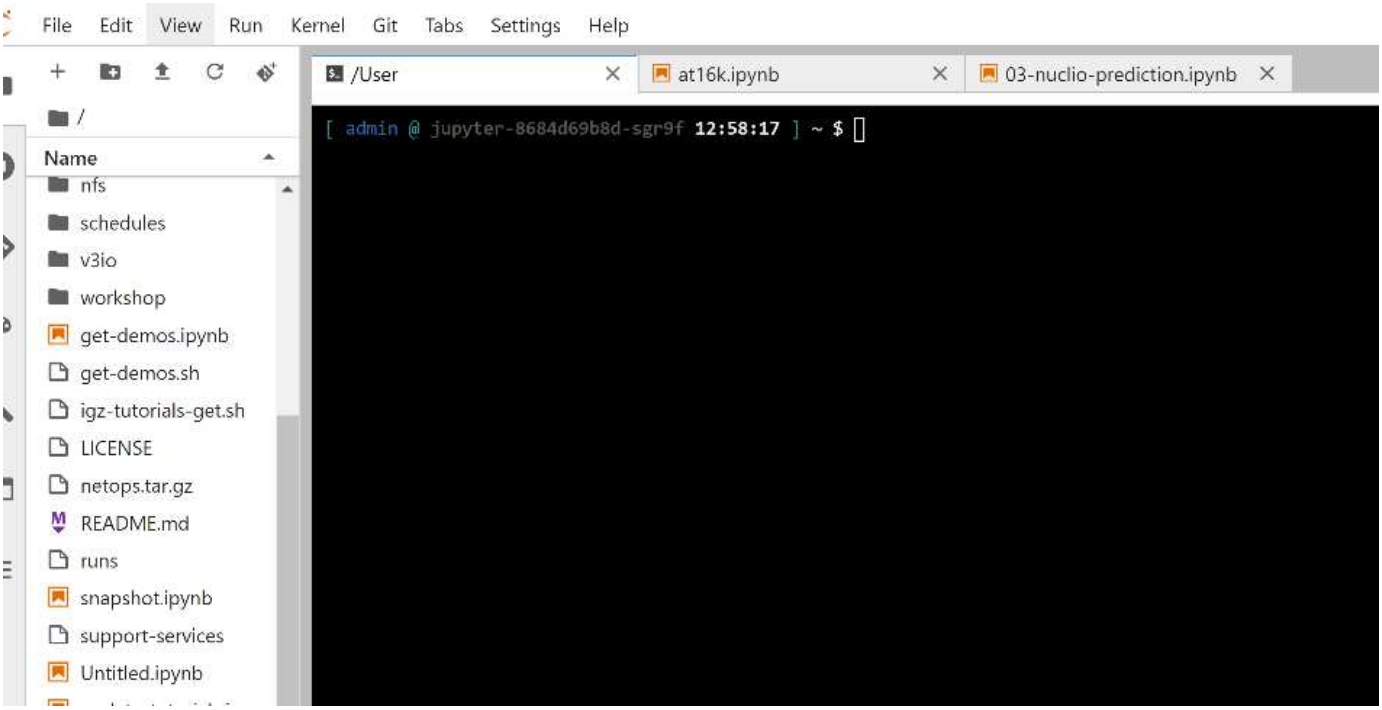
#### 从 GitHub 获取代码

既然 NetApp Cloud Volume 或 NetApp Trident 卷可供 Iguazio 集群和开发人员环境使用，

您就可以开始查看该应用程序了。

用户拥有自己的工作空间（目录）。在每个笔记本电脑上，用户目录的路径为 ` /User` 。Iguazio 平台负责管理目录。如果按照上述说明进行操作，则可以在 ` /NetApp` 目录中找到 NetApp Cloud 卷。

使用 Jupyter 终端从 GitHub 获取代码。




在 Jupyter 终端提示符处，克隆项目。

```
cd /User
git clone .
```

现在，您应在 Jupyter 工作空间的文件树中看到 `NetOps" - "NetApp` " 文件夹。

配置工作环境

## 将 `Notebook set_env-example.ipynb` 复制为 `set_env.ipynb` 。打开并编辑 `set_env.ipynb` 。此笔记本电脑可为凭据，文件位置和执行驱动程序设置变量。

如果按照上述说明进行操作，则只需执行以下步骤即可：

1. 从 Iguazio 服务信息板获取此值： `docker_regRegistry`

   示例： `docker-registry.default-tenant.app.clusterq.iguaziodev.com:80`

2. 将 `admin` 更改为您的 Iguazio 用户名：

   `IGZ_container_path = "/" 用户 /admin"`

下面是 ONTAP 系统连接详细信息。包括安装 Trident 时生成的卷名称。以下设置适用于内部 ONTAP 集群：

```
ontapClusterMgmtHostname = '0.0.0.0'
ontapClusterAdminUsername = 'USER'
ontapClusterAdminPassword = 'PASSWORD'
sourceVolumeName = 'SOURCE VOLUME'
```

以下设置适用于 Cloud Volumes ONTAP：

```
MANAGER=ontapClusterMgmtHostname
svm='svm'
email='email'
password=ontapClusterAdminPassword
weid="weid"
volume=sourceVolumeName
```

#### 创建基本 Docker 映像

构建 ML 管道所需的一切都包含在 Iguazio 平台中。开发人员可以定义运行管道和从 Jupyter Notebook 执行映像创建所需的 Docker 映像的规格。打开笔记本 `creation- images.ipynb` 并运行所有单元格。

此笔记本可创建两个我们在管道中使用的映像。

- `igiio/NetApp.` 用于处理 ML 任务。

**Create image for training pipeline**

```
[4]: fn.build_config(image=docker_registry+'/iguazio/netapp', commands=['pip install \
     v3io_frames fsspec>=0.3.3 PyYAML==5.1.2 pyarrow==0.15.1 pandas==0.25.3 matplotlib seaborn yellowb
     fn.deploy()
```

- `NetApp/` 渠道 。包含用于处理 NetApp Snapshot 副本的实用程序。

**Create image for Ontap utilitites**

```
[9]: fn.build_config(image=docker_registry + '/netapp/pipeline:latest',commands=['apt -y update','pip install v3io_frames netapp_ontap'
     fn.deploy()
```

#### 查看各个 Jupyter 笔记本电脑

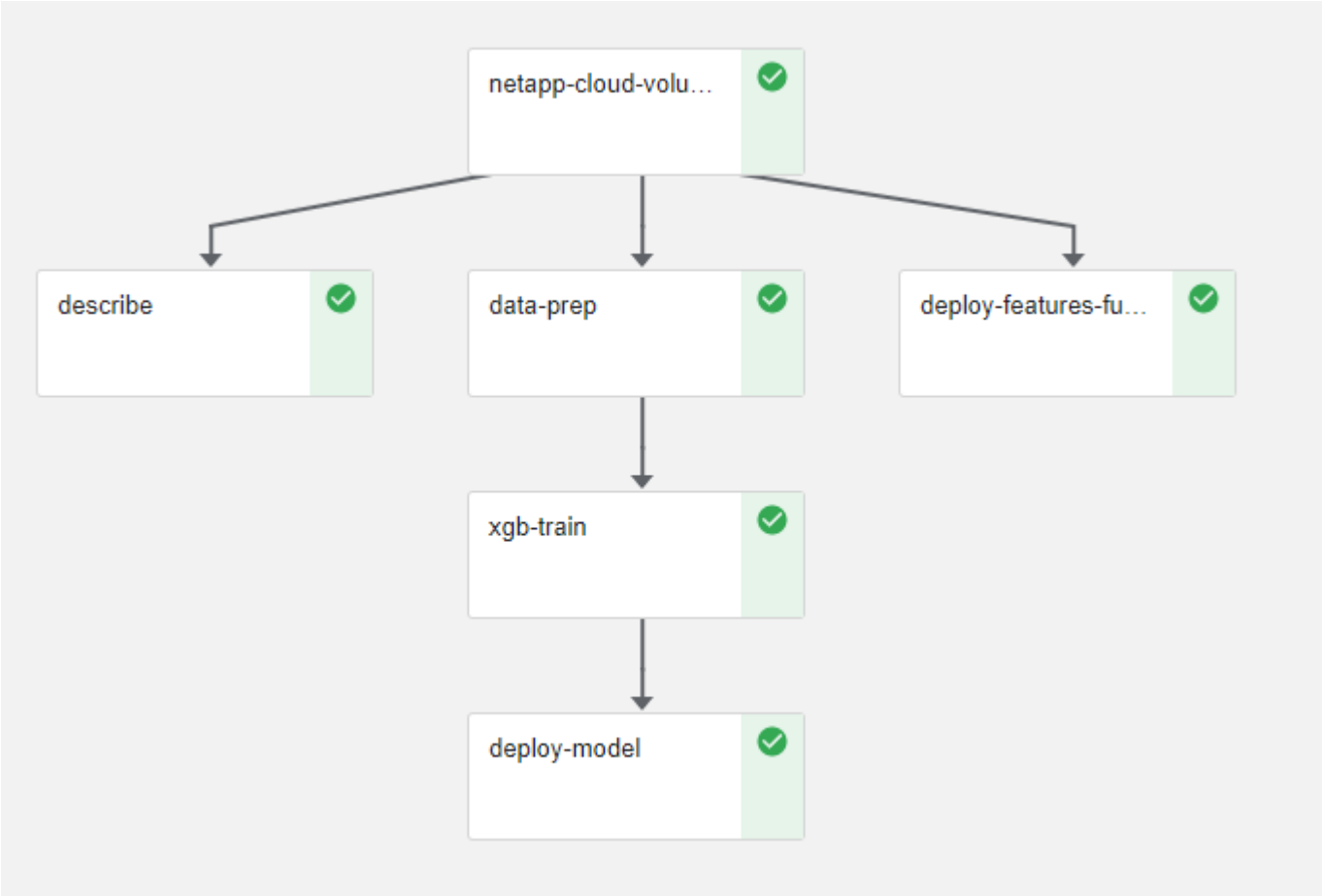
下表列出了我们用于构建此任务的库和框架。所有这些组件均已与 Iguazio 基于角色的访问和安全控制完全集成。

| 库 / 框架 | Description |
|---|---|
| MLRun | 由 Iguazio 管理的，用于组装，执行和监控 ML/AI 管道。 |

| 库 / 框架 | Description |
| --- | --- |
| Nutriio | 与 Iguazio 集成的无服务器功能框架。也可作为由 Iguazio 管理的开源项目提供。 |
| Kubeflow | 基于 Kubernetes 的框架，用于部署管道。这也是一个开源项目， Iguazio 为此做出了贡献。它与 Iguazio 集成，可提高安全性，并与基础架构的其余部分集成。 |
| Docker | Docker 注册表作为服务在 Iguazio 平台中运行。您也可以更改此设置以连接到注册表。 |
| NetApp Cloud Volumes | 通过在 AWS 上运行的 Cloud Volumes ，我们可以访问大量数据，并可以创建 Snapshot 副本来版本用于培训的数据集。 |
| Trident | Trident 是一个由 NetApp 管理的开源项目。它有助于与 Kubernetes 中的存储和计算资源集成。 |

我们使用多台笔记本电脑来构建 ML 管道。在将每台笔记本电脑整合到管道中之前，可以对其进行单独测试。我们将按照此演示应用程序的部署流程分别介绍每台笔记本电脑。

理想的结果是，通过管道根据数据的 Snapshot 副本训练模型，并部署模型进行推理。下图显示了已完成的 MLRun 管道的结构图。



本节介绍如何使用 Nutrio 无服务器功能生成网络设备数据。此用例是从部署了管道并使用 Iguazio 服务监控和预测网络设备故障的 Iguazio 客户端改编而来的。

我们模拟了来自网络设备的数据。执行 Jupyter 笔记本 `data-generator.ipynb` 可创建一个每 10 分钟运行一次的无服务器功能，并使用新数据生成一个 Parquet 文件。要部署此功能，请运行此笔记本中的所有单元。请参见 "Nutrio 网站" 查看此笔记本中任何不熟悉的组件。

生成函数时，将忽略包含以下注释的单元格。假设笔记本电脑中的每个单元都属于该功能的一部分。导入 Nuclio 模块以启用 ` %nuclio 幻影` 。

```
# nuclio: ignore
import nuclio
```

在函数规范中，我们定义了函数的执行环境，触发方式以及使用的资源。

```
spec = nuclio.ConfigSpec(config={"spec.triggers.inference.kind":"cron",

"spec.triggers.inference.attributes.interval" :"10m",
                                "spec.readinessTimeoutSeconds" : 60,
                                "spec.minReplicas" : 1},……
```

初始化 `init_context` 函数时， Noclio 框架会调用该函数。

```
def init_context(context):
    ….
```

在函数初始化时，将调用不在函数中的任何代码。调用该命令时，系统将执行处理程序功能。您可以更改处理程序的名称并在函数规范中指定它。

```
def handler(context, event):
            …
```

您可以在部署之前从笔记本电脑测试此功能。

```
%%time
# nuclio: ignore
init_context(context)
event = nuclio.Event(body='')
output = handler(context, event)
output
```

该功能可以从笔记本电脑部署，也可以从 CI/CD 管道部署（修改此代码）。

```
addr = nuclio.deploy_file(name='generator',project='netops',spec=spec,
tag='v1.1')
```

渠道笔记本电脑

这些笔记本电脑不能单独执行此设置。这只是对每台笔记本电脑的回顾。我们在管道中调用了这些命令。要分别执行这些操作，请查看 MLRun 文档，将其作为 Kubernetes 作业执行。

**Snap_CV.ipynb**

此笔记本电脑在管道开始时处理 Cloud Volume Snapshot 副本。它会将卷的名称传递到管道环境。此笔记本会调用 shell 脚本来处理 Snapshot 副本。在管道中运行时，执行上下文包含可帮助查找执行所需的所有文件的变量。编写此代码时，开发人员不必担心执行此代码的容器中的文件位置。如后面所述，此应用程序会随其所有依赖项一起部署，而是通过管道参数的定义来提供执行上下文。

```
command = os.path.join(context.get_param('APP_DIR'),"snap_cv.sh")
```

创建的 Snapshot 副本位置将放置在 MLRun 上下文中，供管道中的步骤使用。

```
context.log_result('snapVolumeDetails',snap_path)
```

接下来的三台笔记本电脑将并行运行。

**data-prep.ipynb**

必须将原始指标转换为功能，才能进行模型培训。此笔记本电脑可从 Snapshot 目录读取原始指标，并将模型培训的功能写入 NetApp 卷。

在管道环境中运行时，输入 `DATA_DIR` 包含 Snapshot 副本位置。

```
metrics_table = os.path.join(str(mlruncontext.get_input('DATA_DIR',
os.getenv('DATA_DIR','/netpp'))),
                              mlruncontext.get_param('metrics_table',
os.getenv('metrics_table','netops_metrics_parquet')))
```

描述 **.ipynb**

为了直观显示传入指标，我们部署了一个管道步骤，该步骤可提供通过 Kubeflow 和 MLRun UI 提供的图解和图形。每个执行都有自己版本的此可视化工具。

```
ax.set_title("features correlation")
plt.savefig(os.path.join(base_path, "plots/corr.png"))
context.log_artifact(PlotArtifact("correlation",  body=plt.gcf()),
local_path="plots/corr.html")
```

#### deploy-feature-feature.ipynb

我们会持续监控指标以查找异常。此笔记本电脑可创建一个无服务器功能，用于生成对传入指标运行预测所需的功能。此笔记本电脑将调用函数的创建。功能代码位于笔记本电脑 `data-prep.ipynb` 中。请注意，我们使用同一笔记本电脑作为管道中的一个步骤。

#### 训练 .ipynb

创建功能后，我们将触发模型培训。此步骤的输出为要用于推理的模型。我们还会收集统计信息，以跟踪每个执行情况（实验）。

例如，以下命令会将准确性得分输入到该实验的上下文中。此值在 Kubeflow 和 MLRun 中可见。

```
context.log_result('accuracy',score)
```

#### deploy-inftion-Function.ipynb

管道中的最后一步是将模型部署为无服务器功能，以实现持续推理。此笔记本电脑将调用在 `nuclio-inference - Function .ipynb` 中定义的无服务器功能的创建过程。

##### 审核和构建管道

通过将所有笔记本电脑整合到一个管道中，可以持续运行实验，根据新指标重新评估模型的准确性。首先，打开 `pipeline.ipynb` 笔记本电脑。我们将详细介绍 NetApp 和 Iguazio 如何简化此 ML 管道的部署。

我们使用 MLRun 为管道的每个步骤提供上下文并处理资源分配。MLRun API 服务在 Iguazio 平台中运行，是与 Kubernetes 资源交互的点。每个开发人员都不能直接请求资源； API 负责处理这些请求并启用访问控制。

```
# MLRun API connection definition
mlconf.dbpath = 'http://mlrun-api:8080'
```

此管道可以与 NetApp Cloud Volumes 和内部卷配合使用。我们构建此演示的目的是使用 Cloud Volumes ，但您可以在代码中看到在内部运行的选项。

```
# Initialize the NetApp snap fucntion once for all functions in a notebook
if [ NETAPP_CLOUD_VOLUME ]:
    snapfn =
code_to_function('snap',project='NetApp',kind='job',filename="snap_cv.ipyn
b").apply(mount_v3io())
    snap_params = {
    "metrics_table" : metrics_table,
    "NETAPP_MOUNT_PATH" : NETAPP_MOUNT_PATH,
    'MANAGER' : MANAGER,
    'svm' : svm,
    'email': email,
    'password': password ,
    'weid': weid,
    'volume': volume,
    "APP_DIR" : APP_DIR
       }
else:
    snapfn =
code_to_function('snap',project='NetApp',kind='job',filename="snapshot.ipy
nb").apply(mount_v3io())
….
snapfn.spec.image = docker_registry + '/netapp/pipeline:latest'
snapfn.spec.volume_mounts =
[snapfn.spec.volume_mounts[0],netapp_volume_mounts]
      snapfn.spec.volumes = [ snapfn.spec.volumes[0],netapp_volumes]
```

将 Jupyter 笔记本电脑转变为 Kubeflow 步骤所需的第一个操作是将代码转换为函数。功能具有运行该笔记本电脑所需的所有规格。向下滚动笔记本电脑时，您可以看到我们为管道中的每个步骤定义了一个函数。

| 属于笔记本电脑 | **Description** |
| --- | --- |
| <code_to_Function> （ MLRun 模块的一部分） | 函数名称： project name 。用于组织所有项目项目项目。此信息会显示在 MLRun UI 中。好的。在这种情况下，是 Kubernetes 作业。这可以是 dask ， MPI ， spark8s 等。有关详细信息，请参见 MLRun 文档。文件笔记本的名称。此位置也可以是 Git （ HTTP ）中的一个位置。 |
| 图像 | 我们在此步骤中使用的 Docker 映像的名称。我们先前使用 create-image.ipynb 笔记本创建了此版本。 |
| volume_mounts 和 volumes | 有关在运行时挂载 NetApp Cloud Volume 的详细信息。 |

我们还定义了步骤的参数。

```
params={   "FEATURES_TABLE":FEATURES_TABLE,
           "SAVE_TO" : SAVE_TO,
           "metrics_table" : metrics_table,
           'FROM_TSDB': 0,
           'PREDICTIONS_TABLE': PREDICTIONS_TABLE,
           'TRAIN_ON_LAST': '1d',
           'TRAIN_SIZE':0.7,
           'NUMBER_OF_SHARDS' : 4,
           'MODEL_FILENAME' : 'netops.v3.model.pickle',
           'APP_DIR' : APP_DIR,
           'FUNCTION_NAME' : 'netops-inference',
           'PROJECT_NAME' : 'netops',
           'NETAPP_SIM' : NETAPP_SIM,
           'NETAPP_MOUNT_PATH': NETAPP_MOUNT_PATH,
           'NETAPP_PVC_CLAIM' : NETAPP_PVC_CLAIM,
           'IGZ_CONTAINER_PATH' : IGZ_CONTAINER_PATH,
           'IGZ_MOUNT_PATH' : IGZ_MOUNT_PATH
            }
```

在为所有步骤定义了函数之后，您可以构建管道。我们使用 `kfp` 模块来定义此定义。使用 MLRun 与自行构建之间的区别在于编码的简化和缩短。

我们定义的函数将使用 MLRun 的 `as_step` 函数转换为步骤组件。

#### Snapshot 步骤定义

启动 Snapshot 功能，输出并将 v3io 作为源进行挂载：

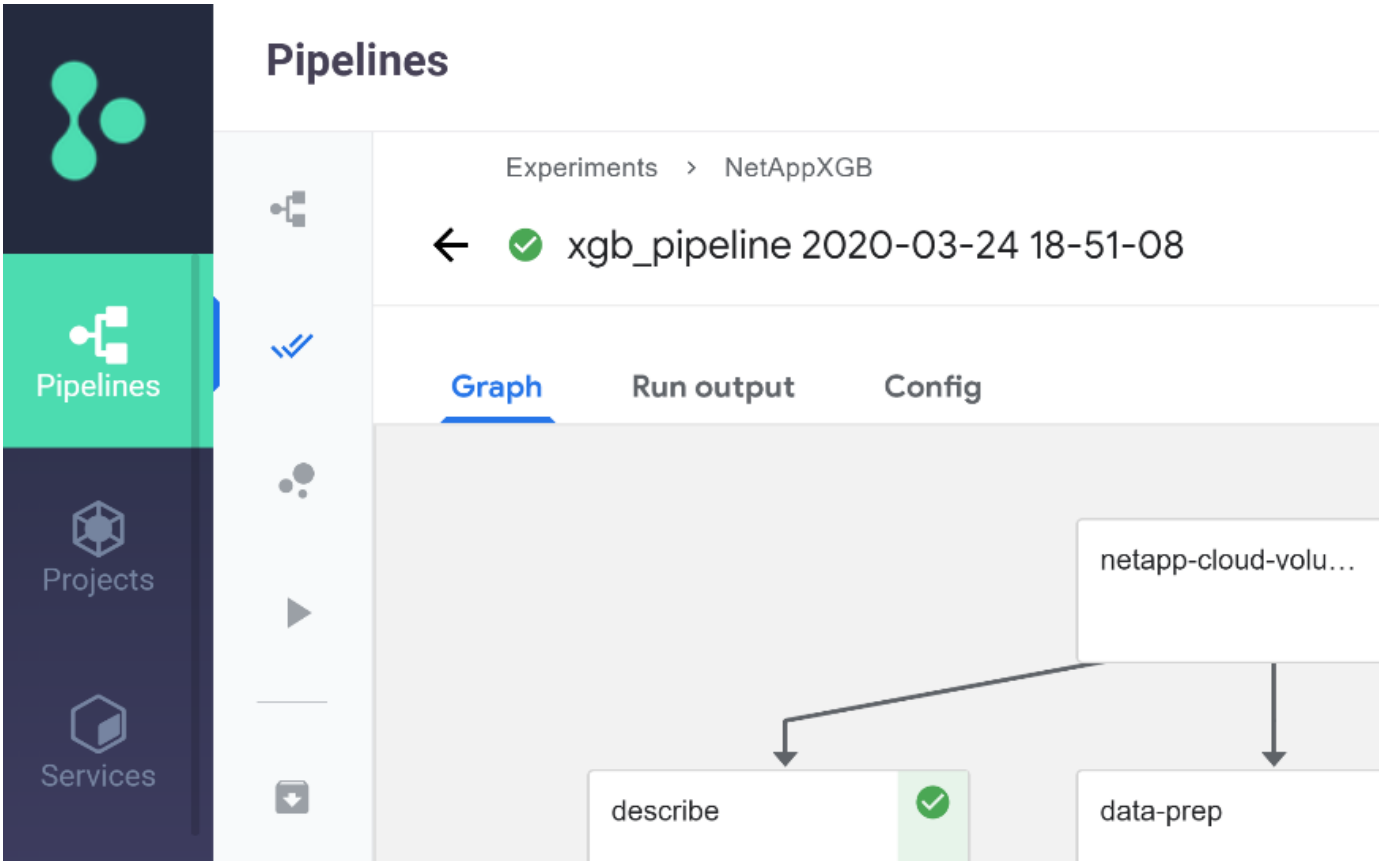
```
snap = snapfn.as_step(NewTask(handler='handler',params=snap_params),
name='NetApp_Cloud_Volume_Snapshot',outputs=['snapVolumeDetails','training
_parquet_file']).apply(mount_v3io())
```

| **Parameters** | 详细信息 |
| --- | --- |
| newtask | newtask 是函数 run 的定义。 |
| （ MLRun 模块） | 处理程序。要调用的 Python 函数的名称。我们在笔记本中使用了名称处理程序，但这不是必需的。参数。我们传递给执行的参数。在代码中，我们使用 context.get_param （‘parameter｝ ）来获取值。 |
| as_step | NameKubeflow 管道步骤的名称。输出。这些值是步骤在完成时添加到词典中的值。查看 snap_CV.ipynb 笔记本电脑。mount_v3io （）。此操作将为执行管道的用户配置挂载 /User 的步骤。 |

```
prep = data_prep.as_step(name='data-prep',
handler='handler',params=params,
                          inputs = {'DATA_DIR':
snap.outputs['snapVolumeDetails']} ,

out_path=artifacts_path).apply(mount_v3io()).after(snap)
```

| Parameters | 详细信息 |
| --- | --- |
| 输入 | 您可以将上一步的输出传递到步骤。在这种情况下，snap.outputs"snapVolumeDetails" 是我们在快照步骤中创建的 Snapshot 副本的名称。 |
| 输出路径 | 一个位置，用于放置使用 MLRun 模块 log_tools. 生成的项目。 |

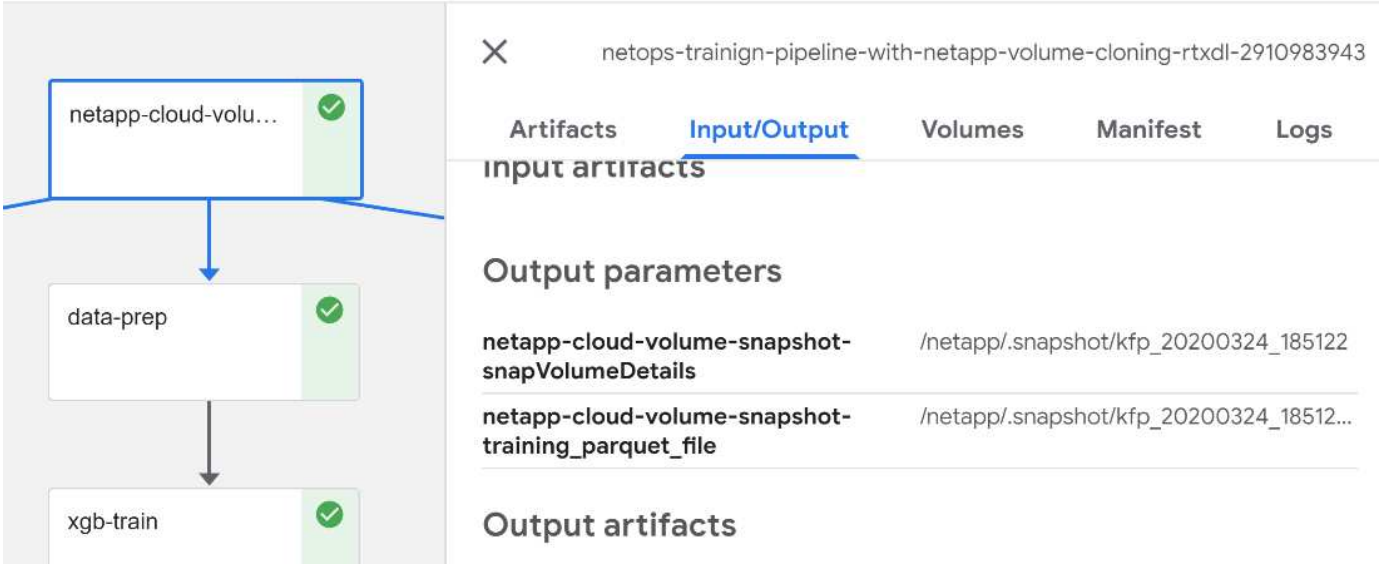
您可以从上至下运行 `pipvip.ipynb` 。然后，您可以转到 Iguazio 信息板中的管道选项卡来监控进度，如 Iguazio 信息板管道选项卡中所示。



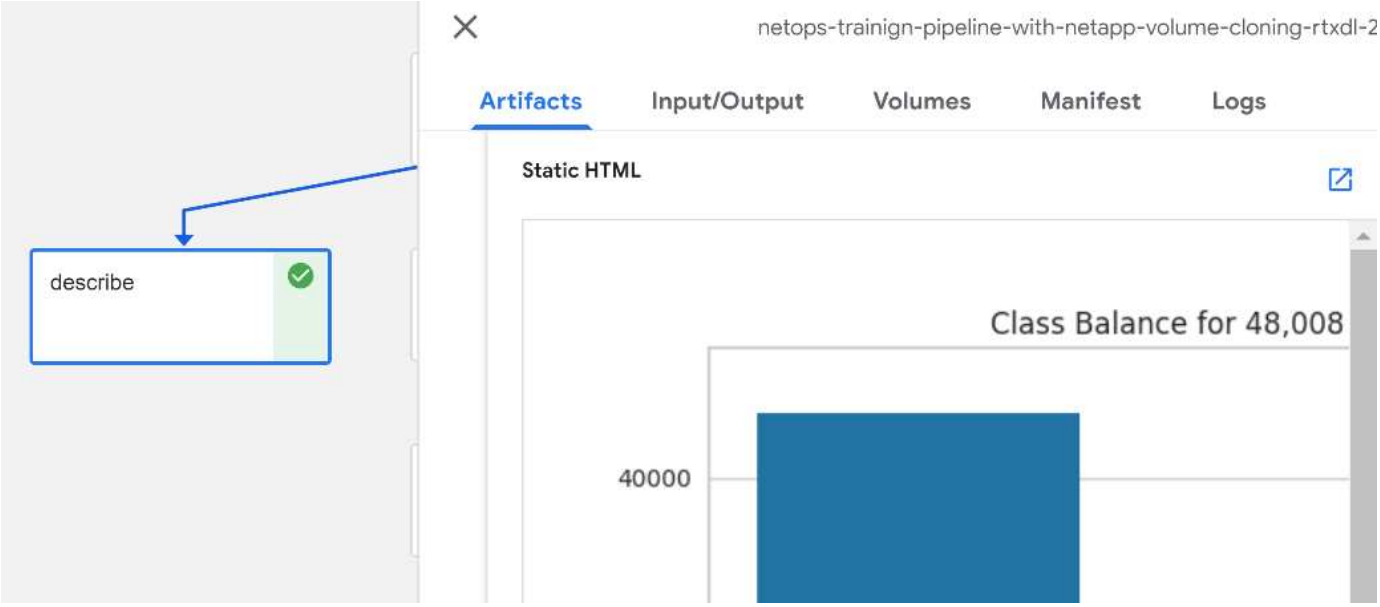

由于我们在每次运行中都记录了训练步骤的准确性，因此我们在每个实验中都有一个准确性记录，如训练准确性记录所示。

| Run name | Status | Duration | Pipeline Version | Recurring ... | Start time | accuracy |
|---|---|---|---|---|---|---|
| xgb_pipeline 2020-03-24 18-51-... | ✓ | 0:08:43 | [View pipeline] | - | 3/24/2020, 2:51:09 PM | 0.985 |
| xgb_pipeline 2020-03-19 13-31-... | ✓ | 0:08:14 | [View pipeline] | - | 3/19/2020, 9:31:19 AM | 0.980 |
| xgb_pipeline 2020-03-18 12-56-... | ✓ | 0:08:11 | [View pipeline] | - | 3/18/2020, 8:56:08 AM | 0.990 |
| xgb_pipeline 2020-03-17 19-49-... | ✓ | 0:08:03 | [View pipeline] | - | 3/17/2020, 3:49:31 PM | 0.985 |
| xgb_pipeline 2020-03-17 18-34-... | ✓ | 0:05:54 | [View pipeline] | - | 3/17/2020, 2:34:56 PM | 0.980 |
| xgb_pipeline 2020-03-17 17-34-... | ✓ | 0:04:48 | [View pipeline] | - | 3/17/2020, 1:34:16 PM | 0.982 |
| xgb_pipeline 2020-03-17 17-01-... | ✓ | 0:05:25 | [View pipeline] | - | 3/17/2020, 1:01:58 PM | 0.987 |
| xgb_pipeline 2020-03-16 16-47-... | ✓ | 0:06:08 | [View pipeline] | - | 3/16/2020, 12:47:19 ... | 0.983 |
| xgb_pipeline 2020-03-16 13-57-... | ✓ | 0:05:18 | [View pipeline] | - | 3/16/2020, 9:57:03 AM | 0.980 |

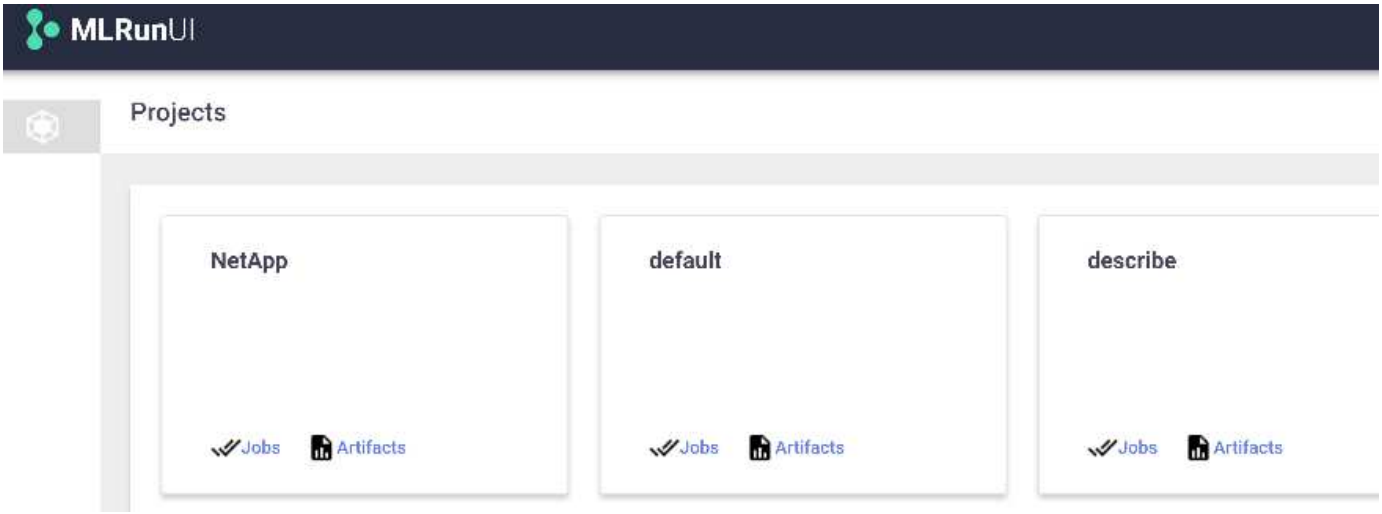
如果选择 Snapshot 步骤，则可以看到用于运行此实验的 Snapshot 副本的名称。



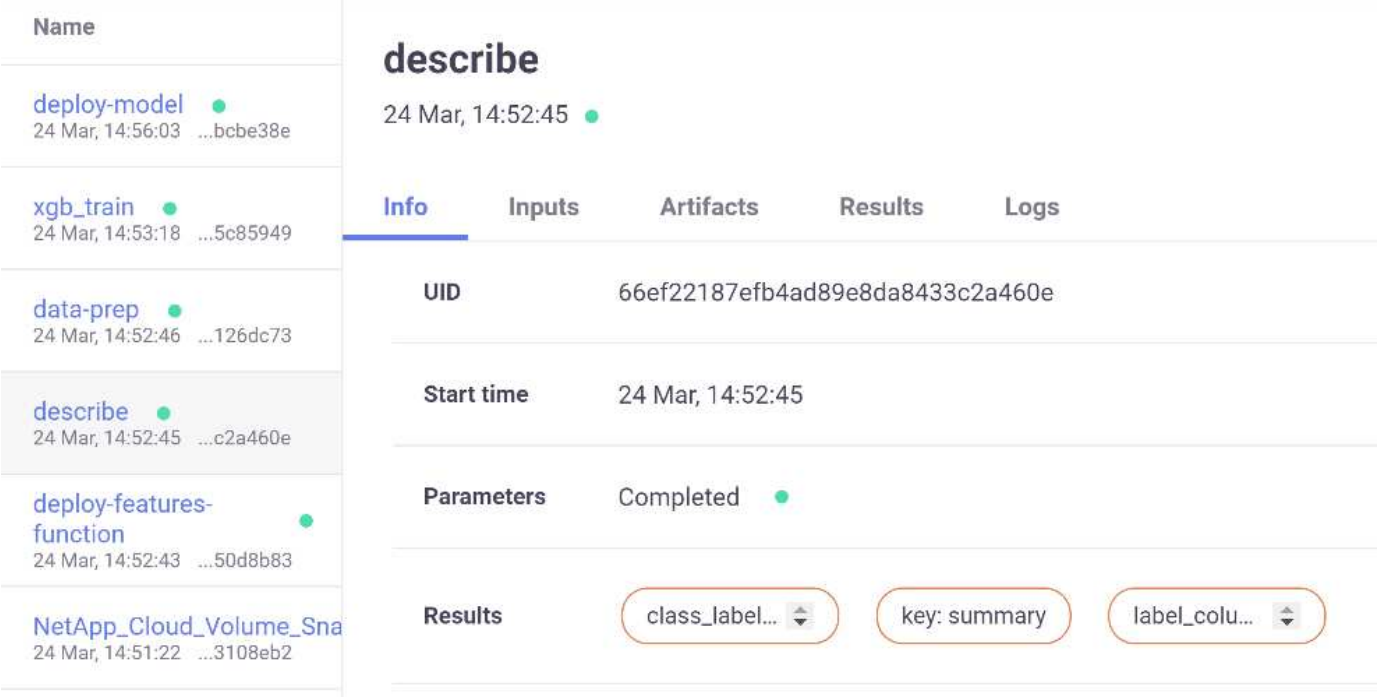

所述步骤具有可视化项目，可用于浏览我们使用的指标。您可以展开以查看完整图，如下图所示。




此外， MLRun API 数据库还会跟踪按项目组织的每个运行的输入，输出和项目。下图显示了每个运行的输入，输出和项目示例。




对于每个作业，我们会存储更多详细信息。



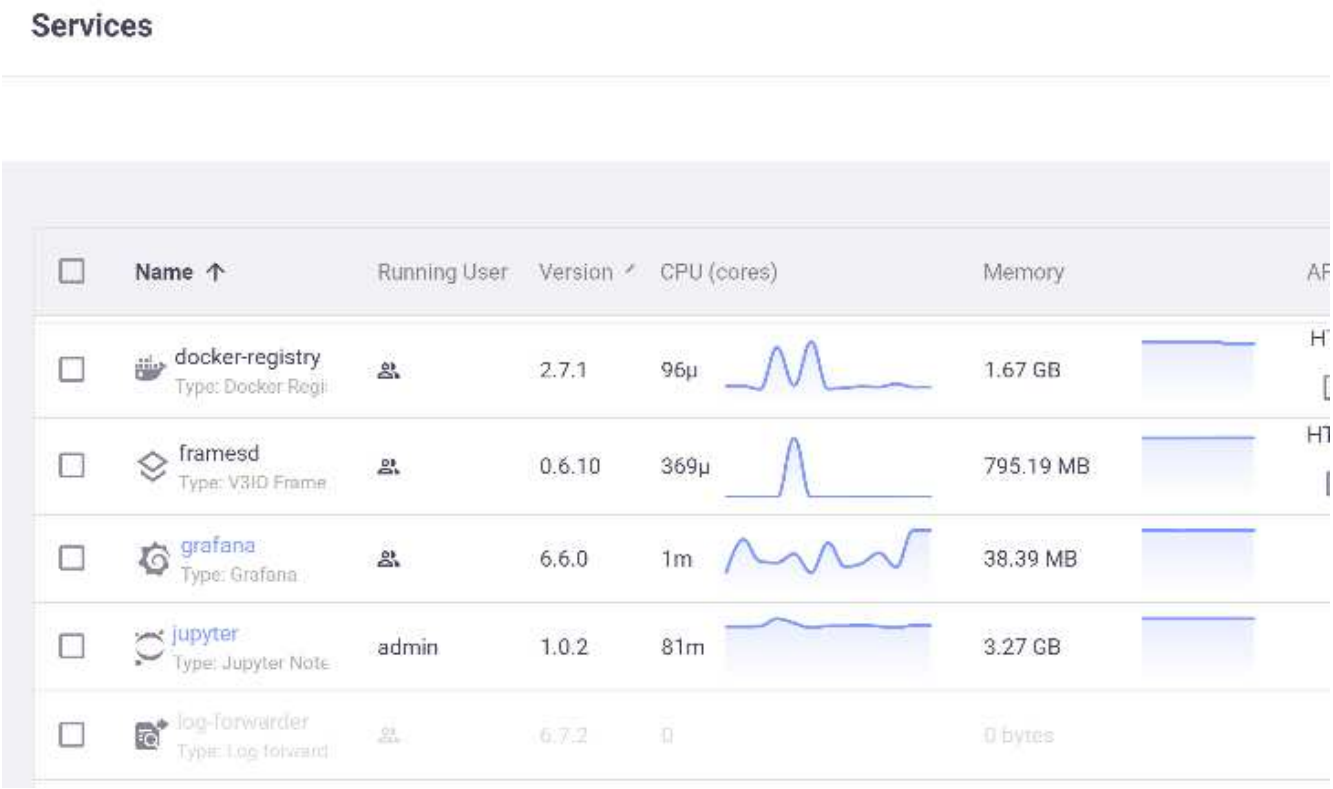

有关 MLRun 的信息比本文档中介绍的信息更多。可以将 AL 项目（包括步骤和功能的定义）保存到 API 数据库中，并进行版本控制，也可以单独调用或作为完整项目调用。此外，还可以保存项目并将其推送到 Git 以供日后使用。我们建议您在中了解更多信息 "MLRun GitHub 站点"。

部署 **Grafana** 信息板

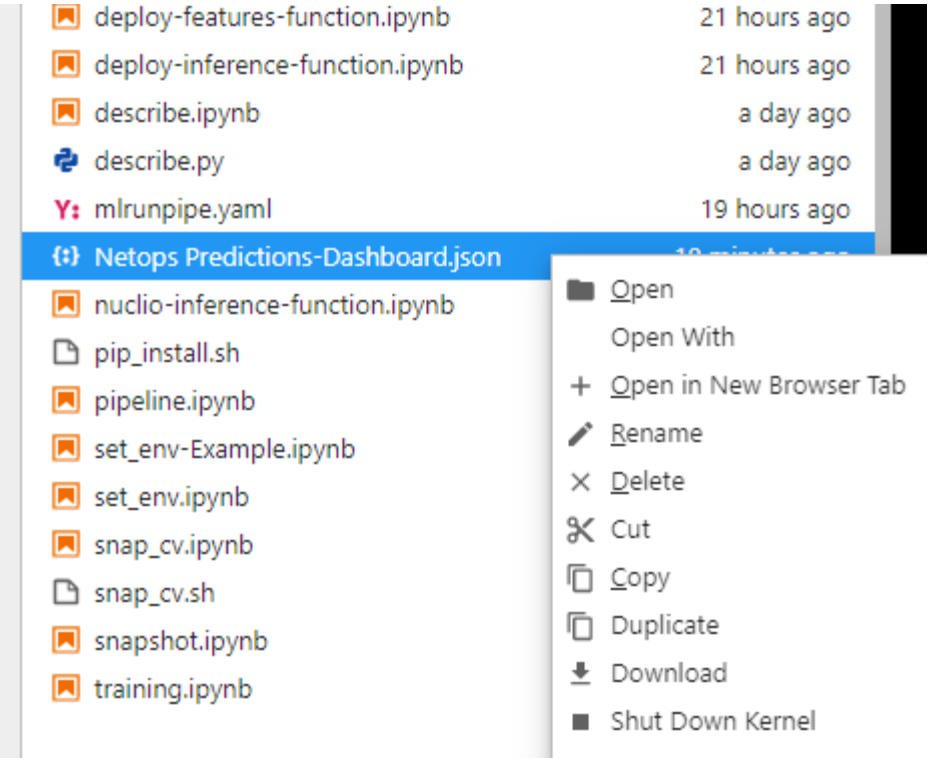
部署完所有内容后，我们会对新数据运行推断。这些型号可预测网络设备故障。预测结果存储在 Iguazio 时间序列表中。您可以在与 Iguazio 的安全和数据访问策略集成的平台中使用 Grafana 来查看结果。

您可以通过将提供的 JSON 文件导入到集群中的 Grafana 接口来部署信息板。

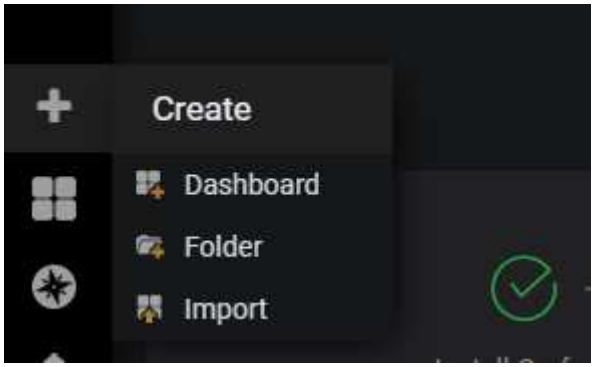
1. 要验证 Grafana 服务是否正在运行，请查看服务下的。



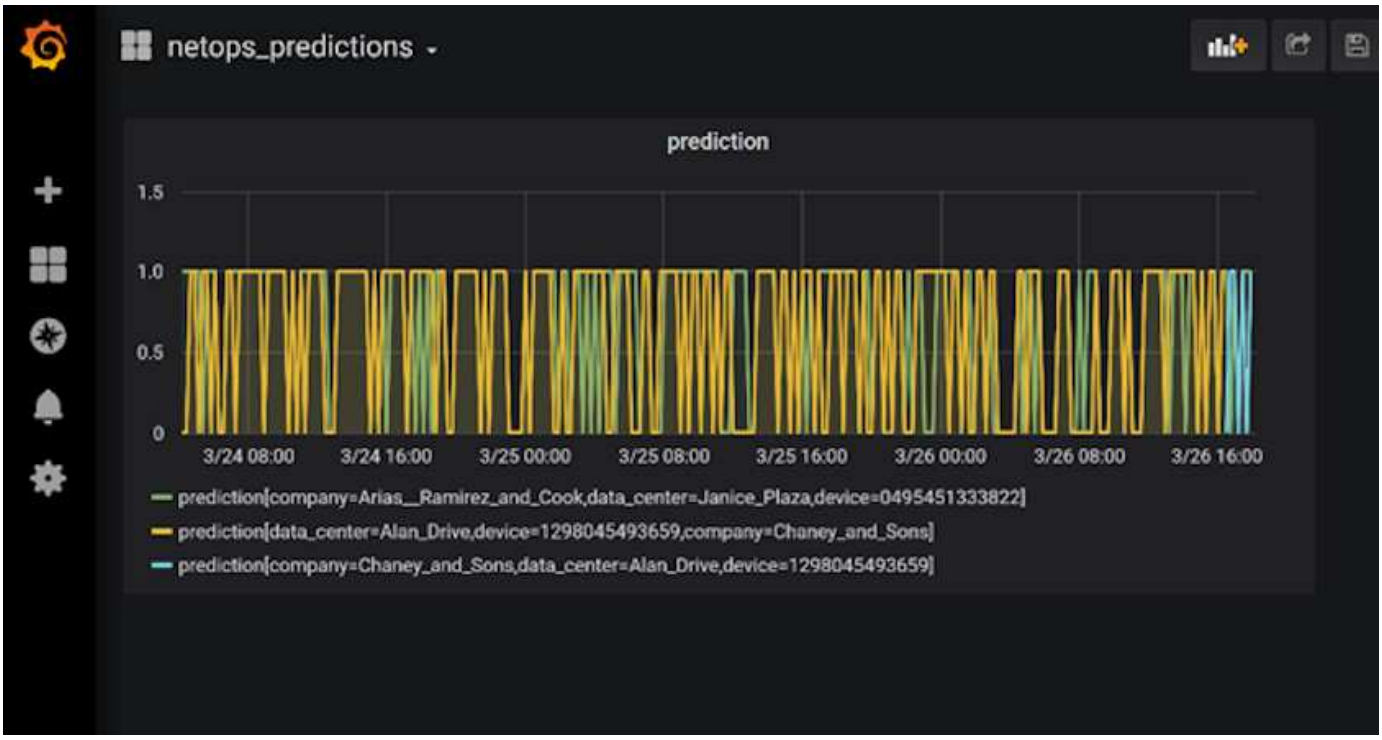

2. 如果不存在此实例，请从服务部分部署此实例：
   a. 单击新建服务。
   b. 从列表中选择 Grafana 。
   c. 接受默认值。
   d. 单击下一步。
   e. 输入您的用户 ID 。
   f. 单击 Save Service 。
   g. 单击顶部的 Apply Changes 。
3. 要部署信息板，请通过 Jupyter 界面下载文件 `NetopsPredictions-Dashboard.json` 。




4. 从服务部分打开 Grafana 并导入信息板。




5. 单击 Upload `* 。 json` File ，然后选择先前下载的文件（`NetopsPredictions-Dashboard.json`）。上传完成后，将显示信息板。




部署清理功能

当您生成大量数据时，保持数据干净有序非常重要。为此，请使用 `cleanup.ipynb` 笔记本部署清理功能。

优势

NetApp 和 Iguazio 通过构建 Kubeflow ， Apache Spark 和 TensorFlow 等基本框架以及 Docker 和 Kubernetes 等业务流程工具，加快并简化 AI 和 ML 应用程序的部署。通过统一端到端数据管道， NetApp 和 Iguazio 可以减少许多高级计算工作负载固有的延迟和复杂性，从而有效地缩小开发和运营之间的差距。在培训阶段，数据科学家可以对大型数据集运行查询，并与授权用户安全地共享数据和算法模型。容器化模型准备好投入生产后，您可以轻松地将其从开发环境迁移到操作环境。

## 结论

在构建自己的 AI/ML 管道时，配置架构中组件的集成，管理，安全性和可访问性是一项极具挑战性的任务。让开发人员访问和控制其环境也带来了另一组挑战。

NetApp 与 Iguazio 的结合将这些技术作为托管服务整合在一起，加快了技术采用速度，并缩短了新 AI/ML 应用程序的上市时间。

# TR-4915：利用E系列和BeeGFS移动数据、实现人工智能和分析工作流

NetApp公司Cody Harryman和Ryan Rodine

TR-4915介绍了如何将数据从任何数据存储库移动到由NetApp E系列SAN存储提供支持的BeeGFS文件系统。对于人工智能(AI)和机器学习(ML)应用程序、客户可能会经常需要将超过数PB数据的大型数据集移动到BeeGFS集群中以进行模型开发。本文档探讨如何使

用NetApp XCP和NetApp BlueXP复制和同步工具来实现此目的。

"TR-4915：利用E系列和BeeGFS移动数据、实现人工智能和分析工作流"