



MLflow

NetApp Solutions

NetApp
December 19, 2024

Inhalt

- MLflow 1
- MLflow-Bereitstellung 1
- Daten-zu-Modell-Rückverfolgbarkeit mit NetApp und MLflow 3

MLflow

MLflow-Bereitstellung

In diesem Abschnitt werden die Aufgaben beschrieben, die Sie ausführen müssen, um MLflow in Ihrem Kubernetes-Cluster bereitzustellen.



MLFlow kann auf anderen Plattformen als Kubernetes implementiert werden. Die Implementierung von MLFlow auf anderen Plattformen als Kubernetes ist nicht im Umfang dieser Lösung enthalten.

Voraussetzungen

Bevor Sie die in diesem Abschnitt beschriebenen Bereitstellungsaufgaben ausführen, gehen wir davon aus, dass Sie bereits die folgenden Aufgaben ausgeführt haben:

1. Sie verfügen bereits über einen funktionierenden Kubernetes-Cluster.
2. Sie haben NetApp Trident bereits in Ihrem Kubernetes-Cluster installiert und konfiguriert. Weitere Informationen zu Trident finden Sie im "[Trident Dokumentation](#)".

Installieren Sie Helm

MLflow wird mit Helm implementiert, einem beliebten Paketmanager für Kubernetes. Bevor Sie MLflow implementieren, müssen Sie Helm auf Ihrem Kubernetes-Kontroll-Node installieren. Um Helm zu installieren, folgen Sie den "[Installationsanweisungen](#)" Anweisungen in der offiziellen Helm-Dokumentation.

Standard-Kubernetes StorageClass festlegen

Bevor Sie MLflow implementieren, müssen Sie eine Standard-StorageClass innerhalb Ihres Kubernetes-Clusters festlegen. Befolgen Sie die Anweisungen im Abschnitt, um eine Standard-StorageClass innerhalb Ihres Clusters festzulegen "[Kubeflow Deployment](#)". Wenn Sie bereits eine Standard-StorageClass innerhalb Ihres Clusters festgelegt haben, können Sie diesen Schritt überspringen.

Bereitstellen von MLflow

Sobald die Voraussetzungen erfüllt sind, können Sie mit der MLflow-Bereitstellung mit dem Helm-Diagramm beginnen.

Konfigurieren Sie die MLflow Helm Chart Deployment.

Bevor wir MLflow mit dem Helm-Diagramm bereitstellen, können wir die Bereitstellung so konfigurieren, dass sie NetApp Trident-Speicherklassen verwendet und andere Parameter entsprechend unseren Anforderungen mithilfe einer **config.yaml**-Datei ändern. Ein Beispiel für die Datei **config.yaml** finden Sie unter: <https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml>



Sie können die Trident storageClass unter dem Parameter **global.defaultStorageClass** in der Datei config.yaml einstellen (z.B. storageClass: "ontap-flexvol").

Installieren des Helm-Diagramms

Das Helm-Diagramm kann mit der benutzerdefinierten **config.yaml**-Datei für MLflow mit folgendem Befehl installiert werden:

```
helm install oci://registry-1.docker.io/bitnamicharts/mlflow -f  
config.yaml --generate-name --namespace jupyterhub
```



Der Befehl implementiert MLflow auf dem Kubernetes-Cluster in der benutzerdefinierten Konfiguration über die bereitgestellte **config.yaml**-Datei. MLflow wird im angegebenen Namespace implementiert und ein zufälliger Release-Name wird über kubernetes für die Version gegeben.

Implementierung Prüfen

Nach der Bereitstellung des Helm-Diagramms können Sie überprüfen, ob der Dienst über folgende Funktionen zugänglich ist:

```
kubectl get service -n jupyterhub
```



Ersetzen Sie **jupyterhub** durch den Namespace, den Sie während der Bereitstellung verwendet haben.

Folgende Dienste sollten angezeigt werden:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
mlflow-1719843029-minio	ClusterIP	10.233.22.4	<none>
80/TCP, 9001/TCP			
mlflow-1719843029-postgresql	ClusterIP	10.233.5.141	<none>
5432/TCP			
mlflow-1719843029-postgresql-hl	ClusterIP	None	<none>
5432/TCP			
mlflow-1719843029-tracking	NodePort	10.233.2.158	<none>
30002:30002/TCP			



Wir haben die config.yaml-Datei bearbeitet, um den NodePort-Dienst für den Zugriff auf MLflow auf Port 30002 zu verwenden.

Zugriff auf MLflow

Sobald alle mit MLflow verbundenen Dienste laufen, können Sie über die angegebene NodePort- oder loadbalancer-IP-Adresse darauf zugreifen (z.B. <http://10.61.181.109:30002>)

Daten-zu-Modell-Rückverfolgbarkeit mit NetApp und MLflow

Der "[NetApp DataOps Toolkit für Kubernetes](#)" kann in Verbindung mit den Experimentverfolgungsfunktionen von MLflow verwendet werden, um die Rückverfolgbarkeit von Code zu Datensatz, Datensatz zu Modell oder Arbeitsplatz zu Modell zu implementieren.

Im Beispielnotizbuch wurden folgende Bibliotheken verwendet:

Voraussetzungen

1. "[Pyfackel-Blitz](#)"
2. "[cuda_Runtime](#)"
3. "[Kuscheln](#)"
4. "[triton](#)"
5. "[Das NetApp DataOps Toolkit für Kubernetes](#)"

Um die Rückverfolgbarkeit von Code-Datensatz-Modellen oder Arbeitsbereichen zu Modellen zu implementieren, erstellen Sie einfach mithilfe des DataOps Toolkits einen Snapshot Ihres Datensatzes oder Workspace-Volumes als Teil Ihres Trainingslaufs, wie im folgenden Beispiel gezeigt, ein Codebeispiel. Mit diesem Code werden der Name des Datenvolumens und der Snapshot-Name als Tags gespeichert, die mit dem spezifischen Trainingslauf verknüpft sind, den Sie auf Ihrem MLflow-Experimentverfolgungsserver protokollieren.

```
...
from netapp_dataops.k8s import cloneJupyterLab, createJupyterLab,
deleteJupyterLab, \
listJupyterLabs, createJupyterLabSnapshot, listJupyterLabSnapshots,
restoreJupyterLabSnapshot, \
cloneVolume, createVolume, deleteVolume, listVolumes,
createVolumeSnapshot, \
deleteVolumeSnapshot, listVolumeSnapshots, restoreVolumeSnapshot

mlflow.set_tracking_uri("<your_tracking_server_uri>:<port>")
os.environ['MLFLOW_HTTP_REQUEST_TIMEOUT'] = '500' # Increase to 500
seconds
mlflow.set_experiment(experiment_id)
with mlflow.start_run() as run:
    latest_run_id = run.info.run_id
    start_time = datetime.now()

    # Preprocess the data
    preprocess(input_pdf_file_path, to_be_cleaned_input_file_path)

    # Print out sensitive data (passwords, SECRET_TOKEN, API_KEY
```

```

found)
    check_pretrain(to_be_cleaned_input_file_path)

    # Tokenize the input file
    pretrain_tokenization(to_be_cleaned_input_file_path, model_name,
tokenized_output_file_path)

    # Load the tokenizer and model
    tokenizer = GPT2Tokenizer.from_pretrained(model_name)
    model = GPT2LMHeadModel.from_pretrained(model_name)

    # Set the pad token
    tokenizer.pad_token = tokenizer.eos_token
    tokenizer.add_special_tokens({'pad_token': '[PAD]'})

    # Encode, generate, and decode the text
    with open(tokenized_output_file_path, 'r', encoding='utf-8') as
file:
        content = file.read()
        encode_generate_decode(content, decoded_output_file_path,
tokenizer=tokenizer, model=model)

    # Save the model
    model.save_pretrained(model_save_path)
    tokenizer.save_pretrained(model_save_path)

    # Finetuning here
    with open(decoded_output_file_path, 'r', encoding='utf-8') as
file:
        content = file.read()
    model.finetune(content, tokenizer=tokenizer, model=model)

    # Evaluate the model using NLTK
    output_set = Dataset.from_dict({"text": [content]})
    test_set = Dataset.from_dict({"text": [content]})
    scores = nltk_evaluation_gpt(output_set, test_set, model=model,
tokenizer=tokenizer)
    print(f"Scores: {scores}")

    # End time and elapsed time
    end_time = datetime.now()
    elapsed_time = end_time - start_time
    elapsed_minutes = elapsed_time.total_seconds() // 60
    elapsed_seconds = elapsed_time.total_seconds() % 60

    # Create DOTK snapshots for code, dataset, and model

```

```

    snapshot = createVolumeSnapshot(pvcName="model-pvc",
namespace="default", printOutput=True)

#Log snapshot IDs to MLflow
mlflow.log_param("code_snapshot_id", snapshot)
mlflow.log_param("dataset_snapshot_id", snapshot)
mlflow.log_param("model_snapshot_id", snapshot)

# Log parameters and metrics to MLflow
mlflow.log_param("wf_start_time", start_time)
mlflow.log_param("wf_end_time", end_time)
mlflow.log_param("wf_elapsed_time_minutes", elapsed_minutes)
mlflow.log_param("wf_elapsed_time_seconds", elapsed_seconds)

mlflow.log_artifact(decoded_output_file_path.rsplit('/', 1)[0]) #
Remove the filename to log the directory
mlflow.log_artifact(model_save_path) # log the model save path

print(f"Experiment ID: {experiment_id}")
print(f"Run ID: {latest_run_id}")
print(f"Elapsed time: {elapsed_minutes} minutes and
{elapsed_seconds} seconds")

```

Der oben genannte Codeschnipsel protokolliert die Snapshot-IDs an den MLflow Experiment Tracking Server, der verwendet werden kann, um zurück zu dem spezifischen Datensatz und Modell, die für das Training des Modells verwendet wurden zurückverfolgen. Dies ermöglicht es Ihnen, zurück zu den spezifischen Datensatz und Modell, die für das Training des Modells verwendet wurden, sowie den spezifischen Code, der verwendet wurde, um die Daten vorverarbeiten, tokenize die Eingabedatei, laden Sie den Tokenizer und Modell, kodieren, generieren und dekodieren "NLTK" Sie den Text, speichern Sie das Modell, finetune das Modell, bewerten Sie das Modell mit Perplexity Scores, und protokollieren Sie die Hyperparameter und Metriken zu MLflow. Die folgende Abbildung zeigt zum Beispiel den mittleren quadratischen Fehler (MSE) eines scikit-Lernmodells für verschiedene Versuchsläufe:

[aicp mlrun mlflow sklean MLmodels MSE] | [aicp_mlrun-mlflow_sklean-MLmodels_MSEs.png](#)

Für Datenanalysen, Geschäftsbereichsleiter und Führungskräfte ist es einfach zu verstehen, welches Modell unter Ihren besonderen Einschränkungen, Einstellungen, Zeiträumen und anderen Umständen am besten funktioniert. Weitere Informationen zum Vorverarbeiten, Tokenisieren, Laden, Kodieren, Generieren, Dekodieren, Speichern, Finetunen und Evaluieren des Modells `dotk-mlflow netapp_dataops.k8s` finden Sie im Beispiel für verpackte Python im Repository.

Weitere Informationen zum Erstellen von Snapshots Ihres Datensatzes oder des JupyterLab-Arbeitsbereichs finden Sie im ["NetApp DataOps Toolkit-Seite"](#).

In Bezug auf die trainierten Modelle wurden im `dotk-mlflow` Notebook folgende Modelle verwendet:

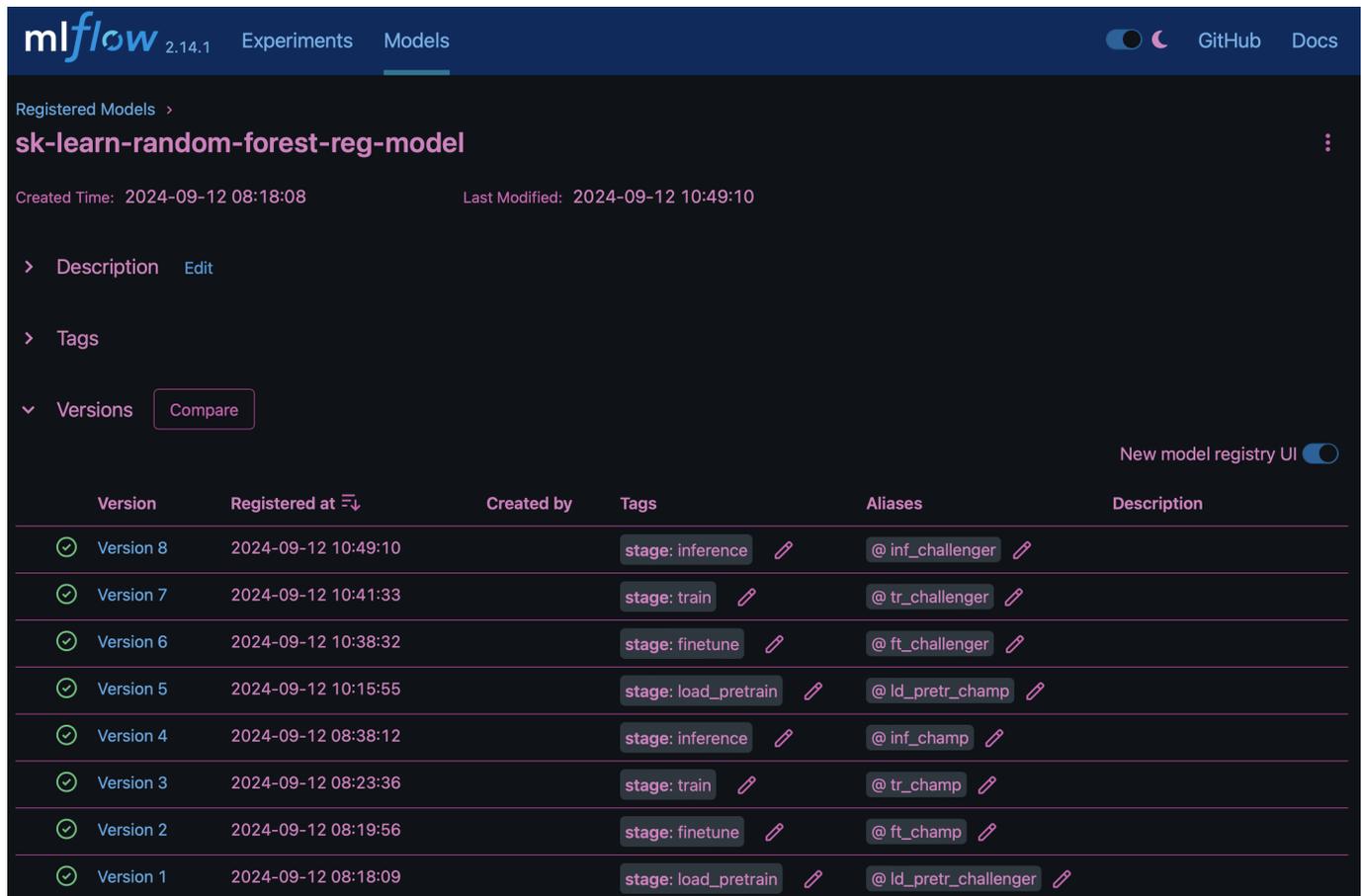
Modelle

1. ["GPT2LMHeadModel"](#): Der GPT2 Modelltransformator mit einem Sprachmodellierungskopf oben (lineare Schicht mit Gewichten, die an die Eingangsangaben gebunden sind). Es handelt sich um ein

Transformatormodell, das auf einem großen Korpus von Textdaten vortrainiert und mit einem bestimmten Datensatz abgearbeitet wurde. Wir haben das Standard-GPT2-Modell "Warnmaske" für das Batching von Eingabesequenzen mit entsprechendem Tokenizer für Ihr Wunschmodell verwendet.

- "Phi-2": Phi-2 ist ein Transformator mit 2.7 Milliarden Parametern. Es wurde mit denselben Datenquellen wie Phi-1.5 trainiert, ergänzt um eine neue Datenquelle, die aus verschiedenen synthetischen NLP-Texten und gefilterten Websites besteht (aus Sicherheitsgründen und Bildungswert).
- "XLNet (Modell mit basierter Größe)": XLNet-Modell auf Englisch vortrainiert. Es wurde in dem Papier "XLNet: Generalisierte autoregressive Vorschulung für das Sprachverständnis" von Yang et al. eingeführt und erstmals in diesem veröffentlicht "Repository".

Das Ergebnis "Modellregistrierung in MLflow" enthält die folgenden zufälligen Forest-Modelle, Versionen und Tags:



Version	Registered at	Created by	Tags	Aliases	Description
Version 8	2024-09-12 10:49:10		stage: inference	@inf_challenger	
Version 7	2024-09-12 10:41:33		stage: train	@tr_challenger	
Version 6	2024-09-12 10:38:32		stage: finetune	@ft_challenger	
Version 5	2024-09-12 10:15:55		stage: load_pretrain	@ld_pretr_champ	
Version 4	2024-09-12 08:38:12		stage: inference	@inf_champ	
Version 3	2024-09-12 08:23:36		stage: train	@tr_champ	
Version 2	2024-09-12 08:19:56		stage: finetune	@ft_champ	
Version 1	2024-09-12 08:18:09		stage: load_pretrain	@ld_pretr_challenger	

Um das Modell über Kubernetes auf einem Inferenzserver bereitzustellen, führen Sie einfach das folgende Jupyter Notebook aus. Beachten Sie, dass `dotk-mlflow` wir in diesem Beispiel anstelle des Pakets die Architektur des zufälligen Forest Regression-Modells ändern, um den mittleren quadratischen Fehler (MSE) im Anfangsmodell zu minimieren und daher mehrere Versionen dieses Modells in unserer Modellregistrierung zu erstellen.

```
from mlflow.models import Model
mlflow.set_tracking_uri("http://<tracking_server_URI_with_port>")
experiment_id='<your_specified_exp_id>'

# Alternatively, you can load the Model object from a local MLmodel file
```

```

# modell = Model.load("~/path/to/my/MLmodel")

from sklearn.datasets import make_regression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

import mlflow
import mlflow.sklearn
from mlflow.models import infer_signature

# Create a new experiment and get its ID
experiment_id = mlflow.create_experiment(experiment_id)

# Or fetch the ID of the existing experiment
# experiment_id =
mlflow.get_experiment_by_name("<your_specified_exp_id>").experiment_id

with mlflow.start_run(experiment_id=experiment_id) as run:
    X, y = make_regression(n_features=4, n_informative=2, random_state=0,
shuffle=False)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
    params = {"max_depth": 2, "random_state": 42}
    model = RandomForestRegressor(**params)
    model.fit(X_train, y_train)

    # Infer the model signature
    y_pred = model.predict(X_test)
    signature = infer_signature(X_test, y_pred)

    # Log parameters and metrics using the MLflow APIs
    mlflow.log_params(params)
    mlflow.log_metrics({"mse": mean_squared_error(y_test, y_pred)})

    # Log the sklearn model and register as version 1
    mlflow.sklearn.log_model(
        sk_model=model,
        artifact_path="sklearn-model",
        signature=signature,
        registered_model_name="sk-learn-random-forest-reg-model",
    )

```

Das Ausführungsergebnis Ihrer Jupyter Notebook-Zelle sollte dem folgenden ähneln, wobei das Modell als Version 3 in der Modellregistrierung registriert wird:

```
Registered model 'sk-learn-random-forest-reg-model' already exists.  
Creating a new version of this model...  
2024/09/12 15:23:36 INFO mlflow.store.model_registry.abstract_store:  
Waiting up to 300 seconds for model version to finish creation. Model  
name: sk-learn-random-forest-reg-model, version 3  
Created version '3' of model 'sk-learn-random-forest-reg-model'.
```

In der Modellregistrierung ist es möglich, nach dem Speichern der gewünschten Modelle, Versionen und Tags auf den spezifischen Datensatz, das Modell und den Code zurückzugreifen, der zum Trainieren des Modells verwendet wurde, sowie auf den spezifischen Code, der zum Verarbeiten der Daten verwendet wurde, den Tokenizer und das Modell zu laden, den Text zu kodieren, zu generieren und zu dekodieren, das Modell zu speichern, das Modell mit NLTK Perplexity Scores oder anderen geeigneten Hypertab-Parametern auszuwerten `snapshot_id`'s and your chosen metrics to MLflow by choosing the correct experiment under ``mlrun`.

📁 / ... / mlruns / 0 /

Name	Last Modified ▲
📁 d1b60feef95498e9f9650f4717cfd00	yesterday
📁 e594a0bd159e4a0385b70b2fcc245...	10 days ago
📁 ffd362a7b84741238fea758f61ca933b	10 days ago
📁 594a2ea3f02f474bb40d0483c10c4...	10 days ago
📁 a819b9464f6c4825b1775d3bc89aa...	12 days ago
📁 a109af0d7dc04d23b91f3fec93c939...	12 days ago
📁 d956baa62bbf4fdf9a7b7692e85ca2...	12 days ago
📁 859487a119c344dbbf4053674e9fd...	12 days ago
📁 a13bd03ab91a4a7f847d953ce96f8f...	12 days ago
📁 7e0df1fb96af499a83e55ef132ed86dd	12 days ago
📁 21af6c1b1afe4b2fab4c7902ac6cfefc	12 days ago
📁 da632683d23b48bd825196fdc7b72...	12 days ago
📁 b5977cc431c14376a0b0daa5d72ab...	13 days ago
📁 8bdb272a3837439da6e9d4cc72844...	13 days ago
📁 18748118e867465c8974f553533d5...	13 days ago
📁 db986997f61e4abe9a63c96c69973f...	13 days ago
📁 7f67342558f54b2ca0687c321754f2...	13 days ago
📁 0a52680a927d4368bc493f68c927e...	14 days ago
📁 3d68fa51a58248538d32ca72be8cb...	14 days ago
📁 f86685b664ed4f5199f5666519b41...	14 days ago
📁 b8814d8376fc4c6aafa6c471496c0a7f	14 days ago
📁 ab7c1e29302345cda9ad8c5a1635d...	14 days ago
📁 652808daa03045198556af61d4d26...	14 days ago
📁 e1d6df0c834c487bb103dd3553ab4...	14 days ago
📁 7a28f96275814ce6bce1a2f17774c9...	14 days ago

In ähnlicher Weise `phi-2_finetuned_model torch` können wir für unsere, deren quantifizierte Gewichte über GPU oder vGPU mithilfe der Bibliothek berechnet wurden, die folgenden Zwischenartefakte prüfen, die die Performance-Optimierung, Skalierbarkeit (Durchsatz/SLA-Garantie) und Kostensenkung des gesamten Workflows ermöglichen:

📁 / ... / 21af6c1b1afe4b2fab4c7902ac6cfefc / params /

Name	Last Modified
📄 lr_scheduler_kwargs	12 days ago
📄 lr_scheduler_type	12 days ago
📄 optim_target_modules	12 days ago
📄 max_steps	12 days ago
📄 neptune_noise_alpha	12 days ago
📄 num_train_epochs	12 days ago
📄 include_num_input_tokens_seen	12 days ago
📄 max_grad_norm	12 days ago
📄 adam_epsilon	12 days ago
📄 include_tokens_per_second	12 days ago
📄 adam_beta2	12 days ago
📄 split_batches	12 days ago
📄 adam_beta1	12 days ago
📄 dispatch_batches	12 days ago
📄 torch_compile_mode	12 days ago
📄 weight_decay	12 days ago
📄 torch_compile_backend	12 days ago
📄 learning_rate	12 days ago
📄 torch_compile	12 days ago
📄 torch_empty_cache_steps	12 days ago
📄 ddp_timeout	12 days ago
📄 eval_delay	12 days ago
📄 eval_accumulation_steps	12 days ago
📄 ray_scope	12 days ago
📄 gradient_accumulation_steps	12 days ago
📄 torchdynamo	12 days ago

Bei einem einzelnen Experiment mit Scikit-learn und MLflow zeigt die folgende Abbildung die generierten Artefakte, conda Umgebung, MLmodel Datei und MLmodel Verzeichnis an:

📁 / ... / artifacts / sklearn-model /

Name	Last Modified	File Size
Y: python_env.yaml	yesterday	114 B
Y: conda.yaml	yesterday	250 B
📄 requirements.txt	yesterday	125 B
📄 MLmodel	yesterday	735 B
📄 model.pkl	yesterday	73.9 KB

Kunden können Tags angeben, z. B. „Standard“, „Phase“, „Prozess“, „Engpass“, um verschiedene Eigenschaften ihrer AI-Workflow-Durchläufe `contributors` zu organisieren, die neuesten Ergebnisse zu notieren oder den Entwicklerfortschritt des Data Science-Teams zu verfolgen. Wenn für das Standard-Tag " ", Ihre gespeicherten `mlflow.log-model.history`, `mlflow.runName`, `mlflow.source.type` `mlflow.source.name` und `mlflow.user` unter JupyterHub aktuell aktiven Datei Navigator Registerkarte:

📁 / ... / d1b60feef95498e9f9650f4717cfd00 / tags /

Name	Last Modified	File Size
📄 mlflow.log-model.history	yesterday	806 B
📄 mlflow.runName	yesterday	18 B
📄 mlflow.source.type	yesterday	5 B
📄 mlflow.source.name	yesterday	61 B
📄 mlflow.user	yesterday	4 B

Schließlich haben die Benutzer ihren eigenen angegebenen Jupyter Workspace, der versioniert und in einer Persistent Volume Claim (PVC) im Kubernetes-Cluster gespeichert wird. Die folgende Abbildung zeigt den Jupyter Workspace, der das `netapp_dataops.k8s` Python-Paket enthält, und die Ergebnisse eines erfolgreich erstellten `VolumeSnapshot`:

Import NetApp DataOps Toolkit for Kubernetes functions

```
[1]: from netapp_dataops.k8s import list_volumes, list_volume_snapshots, create_volume_snapshot
```

Create Volume Snapshot for User Workspace Volume

The following example shows the execution of a "create volume snapshot" operation for my user workspace volume.

```
[2]: jupyterhub_namespace = "jupyterhub"
my_user_workspace_vol = "claim-moglesby"

create_volume_snapshot(namespace=jupyterhub_namespace, pvc_name=my_user_workspace_vol, print_output=True)

Creating VolumeSnapshot 'ntap-dsutil.20240726002955' for PersistentVolumeClaim (PVC) 'claim-moglesby' in namespace 'jupyterhub'.
VolumeSnapshot 'ntap-dsutil.20240726002955' created. Waiting for Trident to create snapshot on backing storage.
Snapshot successfully created.
```

Unsere branchenweit bewährten Snapshot® und andere Technologien wurden für Datensicherung, Verschiebung und effiziente Komprimierung der Enterprise-Klasse eingesetzt. Weitere KI-Anwendungsfälle finden Sie in der ["NetApp AIPOd"](#) Dokumentation.

Copyright-Informationen

Copyright © 2024 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

Software, die von urheberrechtlich geschütztem NetApp Material abgeleitet wird, unterliegt der folgenden Lizenz und dem folgenden Haftungsausschluss:

DIE VORLIEGENDE SOFTWARE WIRD IN DER VORLIEGENDEN FORM VON NETAPP ZUR VERFÜGUNG GESTELLT, D. H. OHNE JEGLICHE EXPLIZITE ODER IMPLIZITE GEWÄHRLEISTUNG, EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE STILLSCHWEIGENDE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DIE HIERMIT AUSGESCHLOSSEN WERDEN. NETAPP ÜBERNIMMT KEINERLEI HAFTUNG FÜR DIREKTE, INDIREKTE, ZUFÄLLIGE, BESONDERE, BEISPIELHAFT SCHÄDEN ODER FOLGESCHÄDEN (EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE BESCHAFFUNG VON ERSATZWAREN ODER -DIENSTLEISTUNGEN, NUTZUNGS-, DATEN- ODER GEWINNVERLUSTE ODER UNTERBRECHUNG DES GESCHÄFTSBETRIEBS), UNABHÄNGIG DAVON, WIE SIE VERURSACHT WURDEN UND AUF WELCHER HAFTUNGSTHEORIE SIE BERUHEN, OB AUS VERTRAGLICH FESTGELEGTER HAFTUNG, VERSCHULDENSUNABHÄNGIGER HAFTUNG ODER DELIKTSHAFTUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER AUF ANDEREM WEGE), DIE IN IRGEND EINER WEISE AUS DER NUTZUNG DIESER SOFTWARE RESULTIEREN, SELBST WENN AUF DIE MÖGLICHKEIT DERARTIGER SCHÄDEN HINGEWIESEN WURDE.

NetApp behält sich das Recht vor, die hierin beschriebenen Produkte jederzeit und ohne Vorankündigung zu ändern. NetApp übernimmt keine Verantwortung oder Haftung, die sich aus der Verwendung der hier beschriebenen Produkte ergibt, es sei denn, NetApp hat dem ausdrücklich in schriftlicher Form zugestimmt. Die Verwendung oder der Erwerb dieses Produkts stellt keine Lizenzierung im Rahmen eines Patentrechts, Markenrechts oder eines anderen Rechts an geistigem Eigentum von NetApp dar.

Das in diesem Dokument beschriebene Produkt kann durch ein oder mehrere US-amerikanische Patente, ausländische Patente oder anhängige Patentanmeldungen geschützt sein.

ERLÄUTERUNG ZU „RESTRICTED RIGHTS“: Nutzung, Vervielfältigung oder Offenlegung durch die US-Regierung unterliegt den Einschränkungen gemäß Unterabschnitt (b)(3) der Klausel „Rights in Technical Data – Noncommercial Items“ in DFARS 252.227-7013 (Februar 2014) und FAR 52.227-19 (Dezember 2007).

Die hierin enthaltenen Daten beziehen sich auf ein kommerzielles Produkt und/oder einen kommerziellen Service (wie in FAR 2.101 definiert) und sind Eigentum von NetApp, Inc. Alle technischen Daten und die Computersoftware von NetApp, die unter diesem Vertrag bereitgestellt werden, sind gewerblicher Natur und wurden ausschließlich unter Verwendung privater Mittel entwickelt. Die US-Regierung besitzt eine nicht ausschließliche, nicht übertragbare, nicht unterlizenzierbare, weltweite, limitierte unwiderrufliche Lizenz zur Nutzung der Daten nur in Verbindung mit und zur Unterstützung des Vertrags der US-Regierung, unter dem die Daten bereitgestellt wurden. Sofern in den vorliegenden Bedingungen nicht anders angegeben, dürfen die Daten ohne vorherige schriftliche Genehmigung von NetApp, Inc. nicht verwendet, offengelegt, vervielfältigt, geändert, aufgeführt oder angezeigt werden. Die Lizenzrechte der US-Regierung für das US-Verteidigungsministerium sind auf die in DFARS-Klausel 252.227-7015(b) (Februar 2014) genannten Rechte beschränkt.

Markeninformationen

NETAPP, das NETAPP Logo und die unter <http://www.netapp.com/TM> aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen und Produktnamen können Marken der jeweiligen Eigentümer sein.