Open-Source-MLOps mit NetApp

NetApp Solutions

NetApp
May 10, 2024

Inhalt

Open-Source-MLOps mit NetApp

Open-Source-MLOps mit NetApp

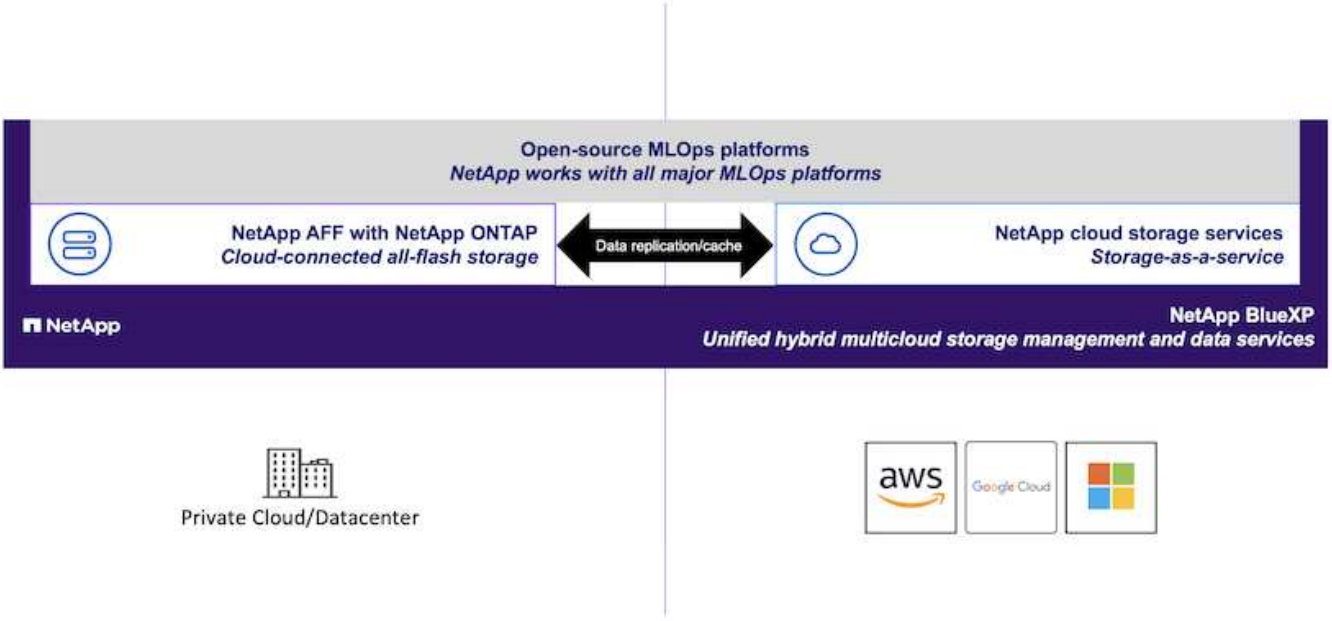
Mike Oglesby, NetApp
Mohan Acharya, NetApp

Unternehmen aller Größen und Branchen setzen zunehmend auf künstliche Intelligenz (KI), maschinelles Lernen (ML) und Deep Learning (DL), um Probleme aus der Praxis zu lösen, innovative Produkte und Services bereitzustellen und sich in einem immer härter umkämpften Markt einen Schritt voraus zu sein. Beim verstärkten Einsatz von KI, ML und DL müssen Unternehmen mit vielen Herausforderungen konfrontiert werden, darunter Workload-Skalierbarkeit und Datenverfügbarkeit. In dieser Lösung wird deutlich, wie Sie diese Herausforderungen bewältigen können, indem Sie NetApp Datenmanagementfunktionen mit gängigen Open-Source-Tools und -Frameworks kombinieren.

Diese Lösung soll mehrere verschiedene Open-Source-Tools und -Frameworks demonstrieren, die in einen MLOPS-Workflow integriert werden können. Diese unterschiedlichen Tools und Frameworks können je nach Anforderungen und Anwendungsfall gemeinsam oder alleine verwendet werden.

Die folgenden Tools/Frameworks werden in dieser Lösung behandelt:

- "Apache Airflow"
- "Kubeflow"



Technologischer Überblick

Künstliche Intelligenz

KI ist eine Informatik-Disziplin, in der Computer trainiert werden, um kognitive Funktionen des menschlichen Verstandes nachzuahmen. KI-Entwickler Schulen Computer, um Probleme so zu lernen oder zu lösen, dass sie dem Menschen ähneln oder sogar überlegen sind. Deep Learning und Machine Learning sind Unterfelder der KI. Unternehmen setzen zunehmend auf KI, ML und DL, um ihre kritischen Geschäftsanforderungen zu unterstützen. Dies sind einige Beispiele:

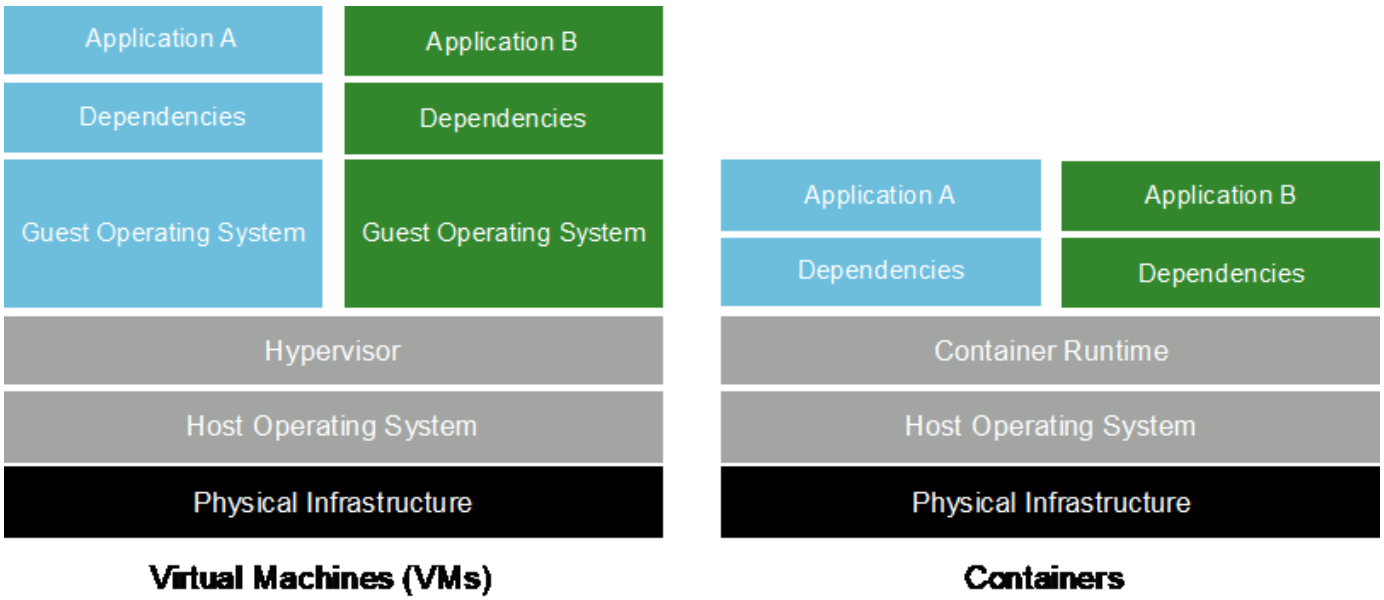
- Analyse großer Datenmengen für bislang unbekannte geschäftliche Einblicke
- Direkte Interaktion mit Kunden über natürliche Sprachverarbeitung
- Automatisierung verschiedener Geschäftsprozesse und Funktionen

Moderne KI-Trainings- und Inferenz-Workloads erfordern extrem hohe parallele Computing-Funktionen. Deshalb werden GPUs zunehmend zur Ausführung von KI-Operationen eingesetzt, da die Funktionen der parallelen Verarbeitung von GPUs denen allgemeiner CPUs überlegen sind.

Container

Container sind isolierte Instanzen von Benutzerspeicherplatz, die auf einem Kernel des Shared-Host-Betriebssystems laufen. Die Einführung von Containern nimmt immer schneller zu. Container bieten viele der gleichen Vorteile von Applikationen im Sandbox-Bereich, die Virtual Machines (VMs) bieten. Da jedoch der Hypervisor und das Gastbetriebssystem die Anzahl der VMs beseitigen, sind die Container viel schlanker. Die folgende Abbildung zeigt eine Visualisierung von Virtual Machines gegenüber Containern.

Container erlauben außerdem die effiziente Bündelung von Applikationsabhängigkeiten, Laufzeiten usw. und damit direkt mit einer Applikation. Das am häufigsten verwendete Format für Containerverpackungen ist der Docker Container. Eine Applikation, die im Docker-Container-Format gesichert wurde, kann auf jeder Maschine ausgeführt werden, die Docker Container ausführen kann. Dies gilt auch dann, wenn die Abhängigkeiten der Anwendung nicht auf der Maschine vorhanden sind, weil alle Abhängigkeiten im Container selbst verpackt sind. Weitere Informationen finden Sie auf der "Docker-Website".



Kubernetes

Kubernetes ist eine ursprünglich von Google entwickelte Open-Source-Plattform zur Container-Orchestrierung,

die jetzt von der Cloud Native Computing Foundation (CNCF) verwaltet wird. Kubernetes ermöglicht die Automatisierung von Implementierungs-, Management- und Skalierungsfunktionen für Container-Applikationen. In den letzten Jahren hat sich Kubernetes zur führenden Plattform für die Container-Orchestrierung entwickelt. Weitere Informationen finden Sie auf der "Kubernetes-Website".

NetApp Astra Trident

Astra Trident ermöglicht die Nutzung und das Management von Storage-Ressourcen über alle gängigen NetApp Storage-Plattformen hinweg, in der Public Cloud oder lokal, einschließlich ONTAP (AFF, FAS, Select, Cloud, Amazon FSX for NetApp ONTAP), Element Software (NetApp HCI, SolidFire), Azure NetApp Files Service und Cloud Volumes Service auf Google Cloud. Astra Trident ist ein CSI-konformer dynamischer Storage-Orchestrator, der sich nativ in Kubernetes integrieren lässt.

NetApp DataOps Toolkit

Der "NetApp DataOps Toolkit" Ein Python-basiertes Tool, das das Management von Workspaces für Entwicklung/Training und Inferenz-Servern mit hochperformantem, horizontal skalierbarem NetApp Storage vereinfacht Die wichtigsten Funktionen:

- Schnelle Bereitstellung neuer Workspaces mit hoher Kapazität, die auf hochperformantem, horizontal skalierbarem NetApp-Storage beruhen
- Klonen Sie Workspaces mit hoher Kapazität nahezu instan, um Experimente oder schnelle Iterationen zu ermöglichen.
- Nahezu instanziiert Snapshots von Arbeitsbereichen mit hoher Kapazität für Backups und/oder Rückverfolgbarkeit/Baselining.
- Provisionieren, Klonen und Snapshot High-Capacity-High-Performance-Daten-Volumes gleichzeitig

Kubeflow

Kubeflow ist ein Open Source AI und ML Toolkit für Kubernetes und wurde ursprünglich von Google entwickelt. Das Kubeflow-Projekt macht Implementierungen von KI- und ML-Workflows auf Kubernetes einfach, tragbar und skalierbar. Kubeflow abstrahiert die Besonderheiten von Kubernetes und ermöglicht Data Scientists, sich auf das zu konzentrieren, was sie am besten wissen — Data Science. Eine Visualisierung finden Sie in der folgenden Abbildung. Kubeflow ist eine gute Open-Source-Option für Unternehmen, die eine All-in-One-MLOPS-Plattform bevorzugen. Weitere Informationen finden Sie auf der "Kubeflow-Website".

Kubeflow-Pipelines

Kubeflow Pipelines sind ein Schlüsselbestandteil von Kubeflow. Kubeflow Pipelines sind eine Plattform und ein Standard für die Definition und Implementierung portabler und skalierbarer KI- und ML-Workflows. Weitere Informationen finden Sie im "Offizielle Dokumentation von Kubeflow".

Jupyter Notebook Server

Ein Jupyter Notebook Server ist eine Open Source Web-Anwendung, mit der Data Scientists Wiki-ähnliche Dokumente erstellen können, genannt Jupyter Notebooks, die sowohl Live-Code als auch einen beschreibenden Test enthalten. Jupyter Notebooks werden in der KI- und ML-Community häufig eingesetzt, um KI- und ML-Projekte zu dokumentieren, zu speichern und gemeinsam zu nutzen. Kubeflow vereinfacht die Bereitstellung und Bereitstellung von Jupyter Notebook-Servern auf Kubernetes. Weitere Informationen zu Jupyter Notebooks finden Sie auf der "Jupyter-Website". Weitere Informationen zu Jupyter Notebooks im Kontext von Kubeflow finden Sie im "Offizielle Dokumentation von Kubeflow".

Katib

Katib ist ein Kubernetes-natives Projekt für automatisiertes maschinelles Lernen (AutoML). Katib unterstützt Hyperparameter-Tuning, Early Stop und neuronale Architektursuche (NAS). Katib ist ein Projekt, das unabhängig von ML-Frameworks (Machine Learning) ist. Es kann Hyperparameter von Anwendungen einstellen, die in einer beliebigen Sprache des Benutzers geschrieben werden, und unterstützt nativ viele ML-Frameworks, wie TensorFlow, MXNet, PyTorch, XGBoost, und andere. Katib unterstützt viele verschiedene AutoML-Algorithmen, wie Bayesian-Optimierung, Tree of Parzen Estimators, Random Search, Covariance Matrix Adaptation Evolution Strategy, Hyperband, Efficient Neural Architecture Search, Differentiable Architecture Search und viele mehr. Weitere Informationen zu Jupyter Notebooks im Kontext von Kubeflow finden Sie im "Offizielle Dokumentation von Kubeflow".

Apache Airflow

Apache Airflow ist eine Open-Source-Workflow-Managementplattform, die programmatisches Authoring, Scheduling und Monitoring für komplexe Unternehmens-Workflows ermöglicht. Sie wird häufig zur Automatisierung von ETL- und Daten-Pipeline-Workflows verwendet, beschränkt sich jedoch nicht auf diese Arten von Workflows. Das Airflow-Projekt wurde von Airbnb gestartet, ist aber inzwischen sehr populär in der Branche und fällt nun unter die Schirmherrschaft der Apache Software Foundation. Der Luftstrom wird in Python geschrieben, Airflow-Workflows werden über Python-Skripte erstellt und Airflow wird nach dem Prinzip „Configuration as Code“ entworfen. Viele Benutzer von Airflow setzen nun Airflow auf Kubernetes aus.

Gesteuerte Acyclic-Grafiken (DAGs)

In Airflow werden Workflows als gesteuerte Acyclic Grafs (DAGs) bezeichnet. DAGs bestehen aus Aufgaben, die je nach DAG-Definition nacheinander, parallel oder kombiniert ausgeführt werden. Der Airflow Scheduler führt individuelle Aufgaben auf einem Array von Mitarbeitern aus und erfüllt dabei die in der DAG-Definition festgelegten Abhängigkeiten auf Aufgabenebene. DAGs werden über Python Skripte definiert und erstellt.

NetApp ONTAP

ONTAP 9, die jüngste Generation der Storage-Managementsoftware von NetApp, ermöglicht Unternehmen eine Modernisierung der Infrastruktur und den Übergang zu einem Cloud-fähigen Datacenter. Dank der erstklassigen Datenmanagementfunktionen lassen sich mit ONTAP sämtliche Daten mit einem einzigen Toolset managen und schützen, ganz gleich, wo sich diese Daten befinden. Zudem können Sie die Daten problemlos dorthin verschieben, wo sie benötigt werden: Zwischen Edge, Core und Cloud. ONTAP 9 umfasst zahlreiche Funktionen, die das Datenmanagement vereinfachen, geschäftskritische Daten beschleunigen und schützen und Infrastrukturfunktionen der nächsten Generation über Hybrid-Cloud-Architekturen hinweg ermöglichen.

Vereinfachtes Datenmanagement

Für den Enterprise IT-Betrieb und die Data Scientists spielt Datenmanagement eine zentrale Rolle, damit für KI-Applikationen die entsprechenden Ressourcen zum Training von KI/ML-Datensätzen verwendet werden. Die folgenden zusätzlichen Informationen über NetApp Technologien sind bei dieser Validierung nicht im Umfang enthalten, können jedoch je nach Ihrer Implementierung relevant sein.

Die ONTAP Datenmanagement-Software umfasst die folgenden Funktionen, um den Betrieb zu optimieren und zu vereinfachen und damit Ihre Gesamtbetriebskosten zu senken:

- Inline-Data-Compaction und erweiterte Deduplizierung: Data-Compaction reduziert den ungenutzten Speicherplatz in Storage-Blöcken, während Deduplizierung die effektive Kapazität deutlich steigert. Dies gilt für lokal gespeicherte Daten und für Daten-Tiering in die Cloud.
- Minimale, maximale und adaptive Quality of Service (AQoS): Durch granulare QoS-Einstellungen (Quality

of Service) können Unternehmen ihre Performance-Level für kritische Applikationen auch in Umgebungen mit vielen unterschiedlichen Workloads garantieren.

- NetApp FabricPool: Bietet automatisches Tiering von „kalten“ Daten in Private- und Public-Cloud-Storage-Optionen, einschließlich Amazon Web Services (AWS), Azure und NetApp StorageGRID Storage-Lösung. Weitere Informationen zu FabricPool finden Sie unter "TR-4598: FabricPool Best Practices".

Beschleunigung und Sicherung von Daten

ONTAP bietet überdurchschnittliche Performance und Datensicherung, erweitert diese Funktionen auf folgende Weise:

- Performance und niedrige Latenz: ONTAP bietet höchstmöglichen Durchsatz bei geringstmöglicher Latenz.
- Datensicherung ONTAP verfügt über integrierte Funktionen für die Datensicherung mit zentralem Management über alle Plattformen hinweg.
- NetApp Volume Encryption (NVE) ONTAP bietet native Verschlüsselung auf Volume-Ebene und unterstützt sowohl Onboard- als auch externes Verschlüsselungsmanagement.
- Multi-Faktor- und Multi-Faktor-Authentifizierung – ONTAP ermöglicht die gemeinsame Nutzung von Infrastrukturressourcen mit höchstmöglicher Sicherheit.

Zukunftssichere Infrastruktur

ONTAP bietet folgende Funktionen, um anspruchsvolle und sich ständig ändernde Geschäftsanforderungen zu erfüllen:

- Nahtlose Skalierung und unterbrechungsfreier Betrieb. Mit ONTAP sind das Hinzufügen von Kapazitäten zu bestehenden Controllern und das Scale-out von Clustern unterbrechungsfrei möglich. Kunden können Upgrades auf die neuesten Technologien wie NVMe und 32 GB FC ohne teure Datenmigrationen oder Ausfälle durchführen.
- Cloud-Anbindung: ONTAP ist die Storage-Managementsoftware mit der umfassendsten Cloud-Integration und bietet Optionen für softwaredefinierten Storage und Cloud-native Instanzen in allen Public Clouds.
- Integration in moderne Applikationen: ONTAP bietet Datenservices der Enterprise-Klasse für Plattformen und Applikationen der neuesten Generation, wie autonome Fahrzeuge, Smart Citys und Industrie 4.0, auf derselben Infrastruktur, die bereits vorhandene Unternehmensanwendungen unterstützt.

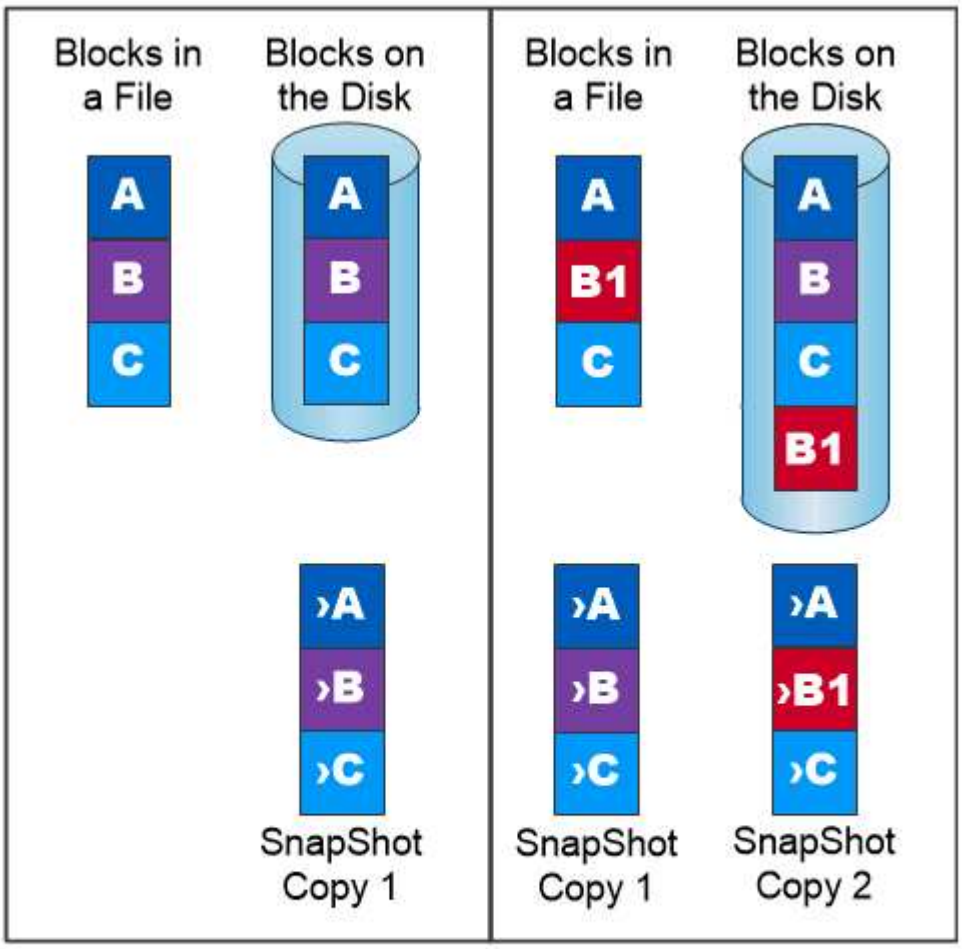
NetApp Snapshot Kopien

Eine NetApp Snapshot Kopie ist ein schreibgeschütztes, zeitpunktgenaues Image eines Volumes. Das Image verbraucht nur minimalen Speicherplatz und beeinträchtigt den Performance-Overhead, da nur Änderungen an Dateien aufgezeichnet werden, die seit der letzten Snapshot Kopie erstellt wurden, wie in der folgenden Abbildung dargestellt.

Snapshot Kopien sind der zentralen ONTAP Storage-Virtualisierungstechnologie, dem Write Anywhere File Layout (WAFL), verdanken sie ihre Effizienz. Wie eine Datenbank verwendet WAFL Metadaten, um auf die tatsächlichen Datenblöcke auf der Festplatte zu verweisen. Im Gegensatz zu einer Datenbank überschreiben WAFL jedoch keine vorhandenen Blöcke. Aktualisierte Daten werden in einen neuen Block geschrieben und die Metadaten geändert. Der Grund dafür ist, dass ONTAP bei der Erstellung einer Snapshot Kopie Metadaten referenziert, statt Datenblöcke zu kopieren. Somit sind die Snapshot Kopien so effizient. So entfallen die Suchzeit, die andere Systeme beim Auffinden der zu kopierenden Blöcke sowie die Kosten für die Erstellung der Kopie selbst tragen.

Sie können eine Snapshot Kopie verwenden, um einzelne Dateien oder LUNs wiederherzustellen oder den gesamten Inhalt eines Volume wiederherzustellen. ONTAP vergleicht Zeigerinformationen in der Snapshot-

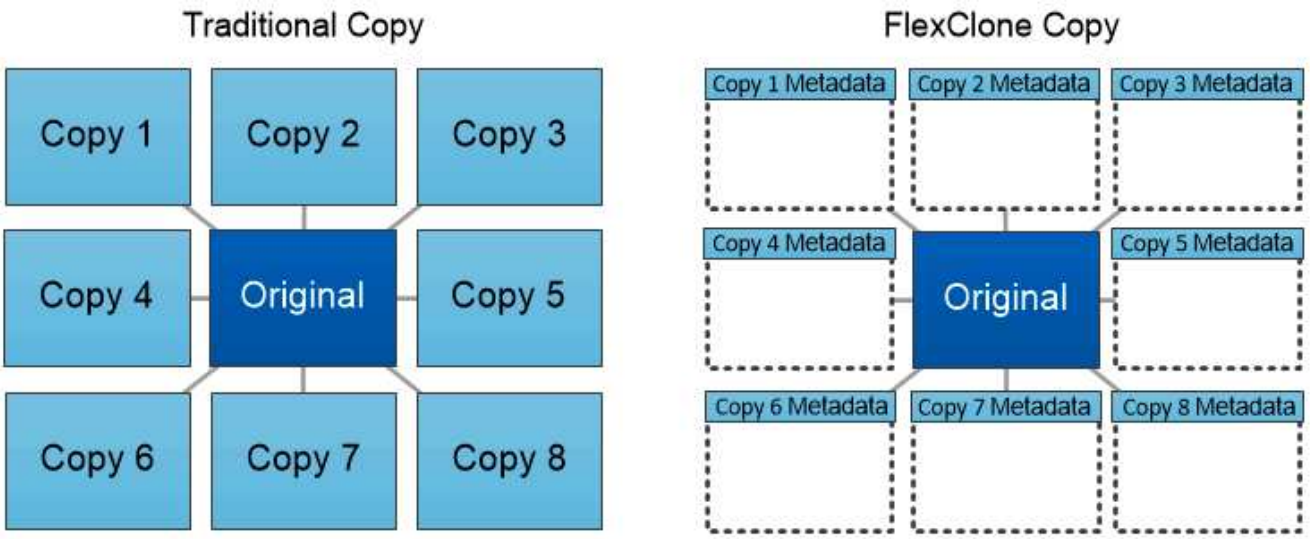
Kopie mit Daten auf der Festplatte, um das fehlende oder beschädigte Objekt ohne Ausfallzeiten und hohe Performance-Kosten zu rekonstruieren.



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

NetApp FlexClone Technologie

Die NetApp FlexClone Technologie referenziert Snapshot Metadaten, um beschreibbare, zeitpunktgenaue Kopien eines Volumes zu erstellen. Kopien verwenden Datenblöcke gemeinsam mit ihren Eltern und verbrauchen somit keinen Storage, außer was für Metadaten erforderlich ist, bis Änderungen in die Kopie geschrieben werden, wie in der folgenden Abbildung dargestellt. Bei der Erstellung herkömmlicher Kopien dauert die Erstellung von Minuten oder gar Stunden, mit FlexClone können Sie selbst die größten Datensätze nahezu sofort kopieren. Daher eignet sie sich besonders für Situationen, in denen mehrere Kopien identischer Datensätze (z. B. ein Entwicklungs-Workspace) oder temporäre Kopien eines Datensatzes benötigt werden (d. h. eine Applikation gegen einen Produktionsdatensatz testen).



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

NetApp SnapMirror Datenreplizierung

NetApp SnapMirror ist eine kostengünstige, benutzerfreundliche und einheitliche Replizierungslösung für die gesamte Data-Fabric-Strategie. Sie repliziert Daten mit hoher Geschwindigkeit über LAN oder WAN. Sie bietet hohe Datenverfügbarkeit und schnelle Datenreplizierung für alle Arten von Applikationen, einschließlich geschäftskritischer Applikationen in virtuellen und herkömmlichen Umgebungen. Durch das Replizieren und ständige Aktualisieren der sekundären Daten auf einem Storage-System von NetApp sind die Daten immer aktuell und verfügbar. Es sind keine externen Replizierungsserver erforderlich. In der folgenden Abbildung finden Sie ein Beispiel für eine Architektur, die die SnapMirror Technologie nutzt.

SnapMirror Software nutzt NetApp ONTAP Storage-Effizienzfunktionen, indem nur geänderte Datenblöcke im Netzwerk verschoben werden. Außerdem verwendet SnapMirror Software eine integrierte Netzwerkkomprimierung, um die Datenübertragung zu beschleunigen und die Auslastung der Netzwerkbandbreite um bis zu 70 % zu reduzieren. Mit der SnapMirror Technologie lässt sich ein Thin-Replication-Datenstrom erstellen, um ein einzelnes Repository zu erstellen, das sowohl den aktiven Spiegel als auch die zeitpunktgenau Kopien enthält. Auf diese Weise verringert sich der Datenverkehr im Netzwerk um bis zu 50 %.

NetApp BlueXP Kopie und Synchronisierung

BlueXP Copy and Sync ist ein NetApp Service für schnelle und sichere Datensynchronisierung. Unabhängig davon, ob Sie Dateien zwischen On-Premises-NFS- oder SMB-Dateifreigaben, NetApp StorageGRID, NetApp ONTAP S3, NetApp Cloud Volumes Service, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob mit Google Cloud Storage oder IBM Cloud Object Storage verschiebt BlueXP Copy and Sync die Dateien schnell und sicher an den gewünschten Speicherort.

Nach der Übertragung stehen die Daten an der Quelle und am Ziel vollständig zur Verfügung. BlueXP Copy and Sync kann Daten nach Bedarf synchronisieren, wenn ein Update ausgelöst wird oder Daten kontinuierlich anhand eines vordefinierten Zeitplans synchronisiert werden. Trotzdem werden mit BlueXP Copy and Sync nur die Deltas verschoben, sodass Zeit und Kosten für die Datenreplizierung minimiert werden.

BlueXP Copy and Sync ist ein Software-as-a-Service-Tool (SaaS), das sich äußerst einfach einrichten und verwenden lässt. Datentransfers, die durch BlueXP Copy und Sync ausgelöst werden, erfolgen durch

Datenmanager. Die Datenmanager von BlueXP Copy und Sync können in AWS, Azure, Google Cloud Platform oder lokal implementiert werden.

NetApp XCP

Der Client-basierte NetApp XCP Software ermöglicht Datenmigrationen zwischen beliebigen Systemen von NetApp und NetApp zu NetApp sowie Einblicke in das Filesystem. XCP ist für Skalierung ausgelegt und erreicht maximale Performance, indem alle verfügbaren Systemressourcen für umfangreiche Datensätze und hochperformante Migrationen genutzt werden. Mit XCP erhalten Sie eine vollständige Übersicht über das Dateisystem und können Berichte generieren.

NetApp XCP ist in einem einzigen Paket erhältlich, das NFS- und SMB-Protokolle unterstützt. XCP enthält eine Linux-Binärdatei für NFS-Datensätze und ein Windows Executable für SMB-Datensätze.

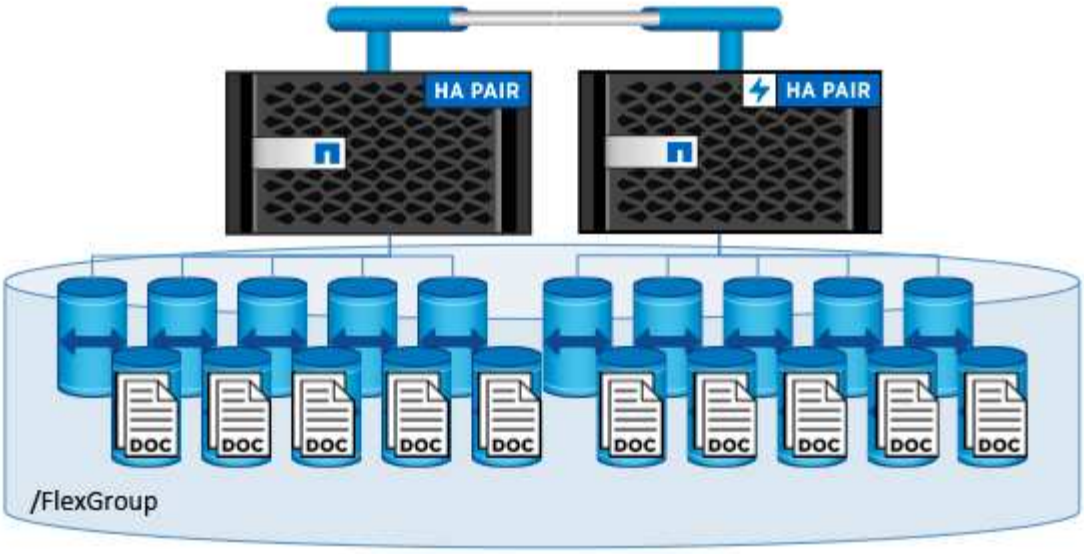
Die hostbasierte Software NetApp XCP File Analytics erkennt Dateifreigaben, führt Scans im Filesystem aus und bietet ein Dashboard für Dateianalysen. XCP File Analytics ist sowohl mit Systemen von NetApp als auch mit Systemen anderer Hersteller kompatibel und wird auf Linux- oder Windows-Hosts ausgeführt, um Analysen für NFS- und SMB-exportierte Filesysteme zu ermöglichen.

NetApp ONTAP FlexGroup Volumes

Ein Trainingsdatensatz kann eine Sammlung von möglicherweise Milliarden von Dateien sein. Dateien können Text, Audio, Video und andere Formen unstrukturierter Daten enthalten, die gespeichert und verarbeitet werden müssen, damit sie gleichzeitig gelesen werden können. Das Storage-System muss eine große Anzahl an kleinen Dateien speichern und diese parallel für sequenzielle und zufällige I/O lesen

Ein FlexGroup Volume ist ein einziger Namespace, der aus mehreren zusammengehörigen Member Volumes besteht, wie in der folgenden Abbildung dargestellt. Aus Sicht eines Storage-Administrators wird ein FlexGroup Volume wie ein NetApp FlexVol Volume gemanagt und verhält sich so wie ein NetApp Volume. Dateien in einem FlexGroup Volume werden Volumes einzelner Mitglieder zugewiesen und nicht über Volumes oder Nodes verteilt. Sie bieten folgende Möglichkeiten:

- FlexGroup Volumes bieten eine Kapazität im Petabyte-Bereich und eine planbare niedrige Latenz für Workloads mit vielen Metadaten.
- Sie unterstützen bis zu 400 Milliarden Dateien im selben Namespace
- Sowie parallelisierte Vorgänge bei NAS-Workloads über CPUs, Nodes, Aggregate und zusammengehörige FlexVol Volumes hinweg.



Der Netapp Architektur Sind

Diese Lösung ist nicht von bestimmter Hardware abhängig. Die Lösung ist mit jeder physischen NetApp Storage Appliance, jeder softwaredefinierten Instanz oder jedem Cloud-Service kompatibel, der von Trident unterstützt wird. Beispiele hierfür sind ein NetApp AFF Storage-System, Amazon FSX für NetApp ONTAP, Azure NetApp Files oder eine NetApp Cloud Volumes ONTAP Instanz. Darüber hinaus kann die Lösung auf jedem Kubernetes-Cluster implementiert werden, solange die verwendete Kubernetes-Version von Kubeflow und NetApp Astra Trident unterstützt wird. Eine Liste der von Kubeflow unterstützten Kubernetes-Versionen finden Sie im "Offizielle Dokumentation von Kubeflow". Eine Liste der von Trident unterstützten Kubernetes-Versionen finden Sie im "Trident Dokumentation". In den folgenden Tabellen finden Sie Einzelheiten zur Umgebung, die zur Validierung der Lösung verwendet wurde.

Software-Komponente	Version
Apache Airflow	2.0.1
Apache Airflow Helm Chart	8.0.8
Kubeflow	1.7, bereitgestellt über "Einsatz KF" 0.1.1
Kubernetes	1.26
NetApp Astra Trident	23.07

Unterstützung

NetApp bietet keine Enterprise-Unterstützung für Apache Airflow, Kubeflow oder Kubernetes. Wenn Sie an einer vollständig unterstützten MLOPS-Plattform interessiert sind, "Kontakt zu NetApp" Über vollständig unterstützte MLOps-Lösungen die NetApp gemeinsam mit Partnern anbietet.

NetApp Astra Trident Konfiguration

Beispiel: Astra Trident Back-Ends für NetApp AIPod Implementierungen

Bevor Sie Storage-Ressourcen innerhalb Ihres Kubernetes-Clusters mit Astra Trident dynamisch bereitstellen können, müssen Sie ein oder mehrere Trident Back-Ends erstellen. Die folgenden Beispiele stellen verschiedene Arten von Back-Ends dar, die Sie erstellen möchten, wenn Sie Komponenten dieser Lösung auf einem bereitstellen "NetApp AIPod". Weitere Informationen zu Backends finden Sie im "Astra Trident-Dokumentation".

1. NetApp empfiehlt die Erstellung eines FlexGroup fähigen Trident Back-End für Ihren AIPod.

   Die folgenden Beispielbefehle zeigen die Erstellung eines FlexGroup-fähigen Trident Back-End für eine AIPod Storage Virtual Machine (SVM). Dieses Back-End verwendet den `ontap-nas-flexgroup` Storage-Treiber: ONTAP unterstützt zwei wesentliche Daten-Volume-Typen: FlexVol und FlexGroup. Die Größe von FlexVol-Volumes ist begrenzt (ab diesem Schreibvorgang hängt die maximale Größe von der spezifischen Implementierung ab). FlexGroup Volumes hingegen lassen sich linear auf bis zu 20 PB und 400 Milliarden Dateien skalieren und sorgen in einem Single Namespace für eine erhebliche Vereinfachung des Datenmanagements. Daher sind FlexGroup-Volumes optimal für AI- und ML-Workloads, die auf große Datenmengen basieren.

   Wenn Sie mit einer geringen Menge an Daten arbeiten und statt FlexGroup Volumes FlexVol Volumes verwenden möchten, können Sie Trident Back-Ends erstellen, die den verwenden `ontap-nas` Storage-Treiber statt des `ontap-nas-flexgroup` Storage-Treiber:

```
$ cat << EOF > ./trident-backend-aipod-flexgroups-iface1.json
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": "aipod-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-aipod-flexgroups-
iface1.json -n trident
+------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+---------------------
+--------------------------------------+--------+---------+
| aipod-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+------------------------+---------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+---------------------
+--------------------------------------+--------+---------+
| aipod-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+------------------------+---------------------
+--------------------------------------+--------+---------+
```

2. NetApp empfiehlt außerdem die Erstellung eines FlexVol-fähigen Trident Back-Endes. Möglicherweise möchten Sie FlexVol Volumes zum Hosten persistenter Applikationen verwenden, zum Speichern von Ergebnissen, Ausgaben, Debug-Informationen usw. Falls Sie FlexVol Volumes verwenden möchten, müssen Sie ein oder mehrere FlexVol-aktivierte Trident-Backends erstellen. Die folgenden Beispielbefehle zeigen die Erstellung eines einzelnen FlexVol-fähigen Trident Back-Endes.

```
$ cat << EOF > ./trident-backend-aipod-flexvols.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "aipod-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-aipod-flexvols.json -n
trident
+------------------------+--------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+--------------------
+--------------------------------------+--------+---------+
| aipod-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-a9c1-
52a69657fabe | online |       0 |
+------------------------+--------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+------------------------+--------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+--------------------
+--------------------------------------+--------+---------+
| aipod-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-a9c1-
52a69657fabe | online |       0 |
| aipod-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-b263-
b6da6dec0bdd | online |       0 |
+------------------------+--------------------
+--------------------------------------+--------+---------+
```

Beispiel: Kubernetes StorageClasses for NetApp AIPod Deployments

Bevor Sie Astra Trident für die dynamische Bereitstellung von Storage-Ressourcen innerhalb Ihres Kubernetes-Clusters verwenden können, müssen Sie eine oder mehrere Kubernetes StorageClasses erstellen. Die folgenden Beispiele stellen verschiedene Typen von StorageClasses dar, die Sie erstellen möchten, wenn Sie Komponenten dieser Lösung auf einem bereitstellen "NetApp AIPod". Weitere Informationen zu

StorageClasses finden Sie im "Astra Trident-Dokumentation".

1. NetApp empfiehlt, eine StorageClass für das FlexGroup-fähige Trident Back-End zu erstellen, das Sie im Abschnitt erstellt haben "Beispiel: Astra Trident Back-Ends für NetApp AIPod Implementierungen", Schritt 1. Die folgenden Beispielbefehle zeigen die Erstellung mehrerer StorageClasses, die dem in diesem Abschnitt erstellten Beispiel-Backend entsprechen "Beispiel: Astra Trident Back-Ends für NetApp AIPod Implementierungen", Schritt 1 - eine, die verwendet "NFS über RDMA" Und eine, die nicht.

   Damit ein anhaltendes Volume nicht gelöscht wird, wenn das entsprechende PersistenzVolumeClaim (PVC) gelöscht wird, verwendet das folgende Beispiel ein `reclaimPolicy` Der Wert von `Retain`. Weitere Informationen zum `reclaimPolicy` Feld, siehe den offiziellen "Kubernetes-Dokumentation".

   Hinweis: Das folgende Beispiel StorageClasses verwendet eine maximale Übertragungsgröße von 262144. Um diese maximale Übertragungsgröße zu verwenden, müssen Sie die maximale Übertragungsgröße auf Ihrem ONTAP-System entsprechend konfigurieren. Siehe "ONTAP-Dokumentation" Entsprechende Details.

   Hinweis: Um NFS über RDMA zu verwenden, müssen Sie NFS über RDMA auf Ihrem ONTAP-System konfigurieren. Weitere Informationen finden Sie in der Dokumentation zum https://docs.netapp.com/us-en/ontap/nfs-rdma/[ONTAP.

   Hinweis: Im folgenden Beispiel wird im Feld StoragePool in der Definitionsdatei der StorageClass kein bestimmtes Backend angegeben.

```
$ cat << EOF > ./storage-class-aipod-flexgroups-retain.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: aipod-flexgroups-retain
provisioner: csi.trident.netapp.io
mountOptions: ["vers=4.1", "nconnect=16", "rsize=262144",
"wsize=262144"]
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "aipod-flexgroups-iface1:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-aipod-flexgroups-retain.yaml
storageclass.storage.k8s.io/aipod-flexgroups-retain created
$ cat << EOF > ./storage-class-aipod-flexgroups-retain-rdma.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: aipod-flexgroups-retain-rdma
provisioner: csi.trident.netapp.io
mountOptions: ["vers=4.1", "proto=rdma", "max_connect=16",
"rsize=262144", "wsize=262144"]
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "aipod-flexgroups-iface1:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-aipod-flexgroups-retain-rdma.yaml
storageclass.storage.k8s.io/aipod-flexgroups-retain-rdma created
$ kubectl get storageclass
NAME                              PROVISIONER             AGE
aipod-flexgroups-retain           csi.trident.netapp.io   0m
aipod-flexgroups-retain-rdma      csi.trident.netapp.io   0m
```

2. NetApp empfiehlt außerdem die Erstellung einer StorageClass, die dem FlexVol-fähigen Trident Back-End entspricht, das Sie im Abschnitt erstellt haben "Beispiel: Astra Trident Back-Ends für AIPod Implementierungen", Schritt 2. Die folgenden Beispielbefehle zeigen die Erstellung einer einzelnen StorageClass für FlexVol Volumes.

   Hinweis: Im folgenden Beispiel wird im Feld StoragePool in der Definitionsdatei der StorageClass kein bestimmtes Backend angegeben. Wenn Sie Kubernetes zum Verwalten von Volumes über diese StorageClass verwenden, versucht Trident, jedes verfügbare Back-End zu verwenden, das die verwendet `ontap-nas` Treiber.

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

Kubeflow

Kubeflow Deployment

In diesem Abschnitt werden die Aufgaben beschrieben, die Sie zur Bereitstellung von Kubeflow in Ihrem Kubernetes-Cluster abschließen müssen.

Voraussetzungen

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

1. Sie haben bereits einen funktionierenden Kubernetes-Cluster und Sie führen eine Version von Kubernetes aus, die von der Kubeflow-Version unterstützt wird, die Sie bereitstellen möchten. Eine Liste der unterstützten Kubernetes-Versionen finden Sie in den Abhängigkeiten für Ihre Kubeflow-Version in der "Offizielle Dokumentation von Kubeflow".
2. Sie haben NetApp Astra Trident bereits in Ihrem Kubernetes-Cluster installiert und konfiguriert. Weitere Informationen zu Astra Trident finden Sie im "Astra Trident-Dokumentation".

Standard-Kubernetes StorageClass festlegen

Bevor Sie Kubeflow implementieren, empfehlen wir, eine Standard-StorageClass in Ihrem Kubernetes-Cluster zu festlegen. Der Kubeflow-Bereitstellungsprozess versucht möglicherweise, neue persistente Volumes mit der standardmäßigen StorageClass bereitzustellen. Wenn keine StorageClass als Standard-StorageClass festgelegt ist, schlägt die Bereitstellung möglicherweise fehl. Um eine Standard-StorageClass innerhalb des Clusters festzulegen, führen Sie die folgende Aufgabe vom Sprunghost für die Bereitstellung aus. Wenn Sie bereits eine Standard-StorageClass innerhalb Ihres Clusters festgelegt haben, können Sie diesen Schritt überspringen.

1. Weisen Sie einen Ihrer vorhandenen StorageClasses als Standard-StorageClass zu. Die folgenden Beispielbefehle zeigen die Bezeichnung einer StorageClass mit dem Namen an `ontap-ai-flexvols-retain` Als Standard-StorageClass.

> Der `ontap-nas-flexgroup` Der Trident Back-End-Typ hat eine ziemlich große PVC-Mindestgröße. Standardmäßig versucht Kubeflow, PVCs bereitzustellen, die nur wenige GB groß sind. Daher sollten Sie keine StorageClass angeben, die den verwendet `ontap-nas-flexgroup` Back-End-Typ als Standard StorageClass für die Zwecke der Kubeflow-Bereitstellung.

```
$ 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
```

Kubeflow Implementierungsoptionen

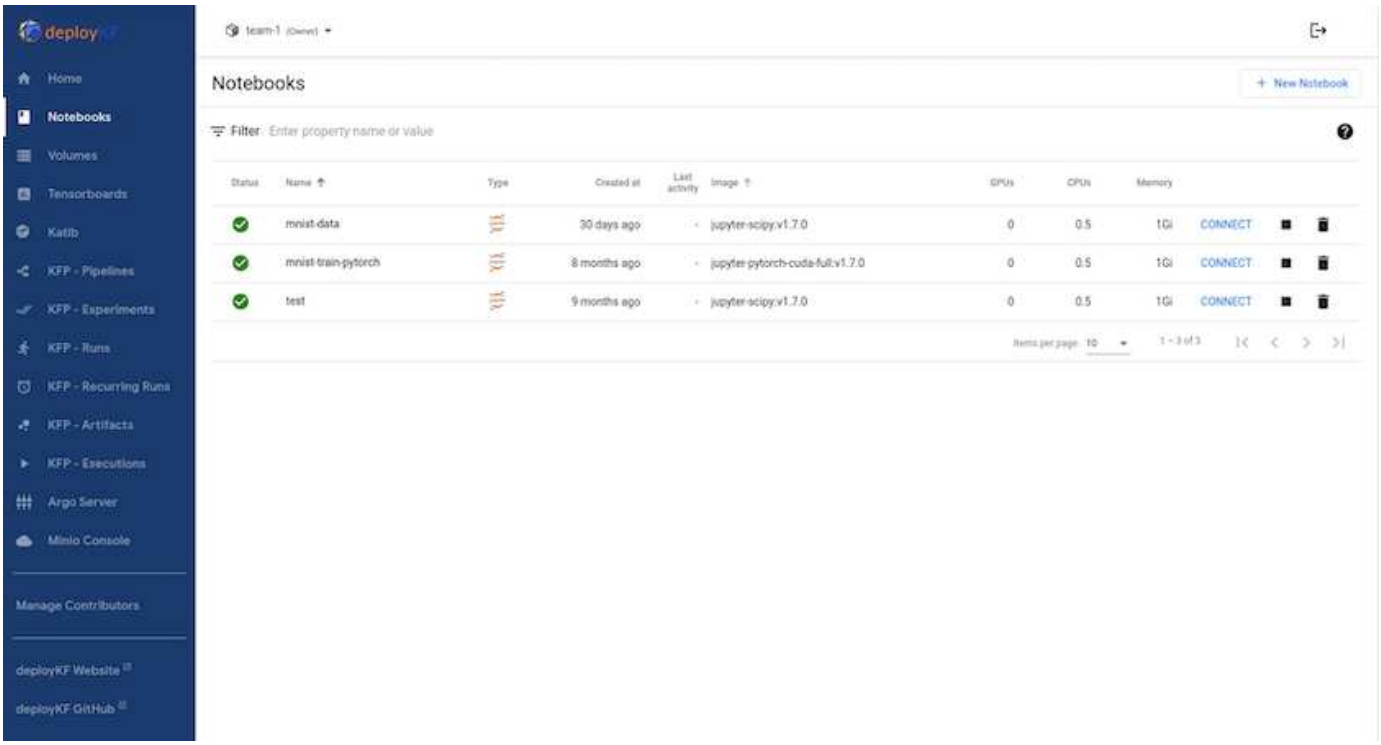
Es gibt viele verschiedene Optionen für die Bereitstellung von Kubeflow. Siehe "Offizielle Dokumentation von Kubeflow" Sie erhalten eine Liste mit Implementierungsoptionen und wählen die Option, die am besten zu Ihren Anforderungen passt.

> Für die Validierung haben wir Kubeflow 1.7 mit implementiert "Einsatz KF" 0.1.1.

Beispiel Kubeflow Operations und Tasks

Bereitstellen eines Jupyter Notebook Workspace für Data Scientist oder Entwickler

Kubeflow ist in der Lage, neue Jupyter Notebook-Server schnell als Data Scientist-Workspaces bereitzustellen. Weitere Informationen zu Jupyter Notebooks im Kubeflow-Kontext finden Sie im "Offizielle Dokumentation von Kubeflow".



Verwenden Sie das NetApp DataOps Toolkit mit Kubeflow

Der "Das NetApp Data Science Toolkit für Kubernetes" Kann in Verbindung mit Kubeflow verwendet werden. Die Verwendung des NetApp Data Science Toolkit und Kubeflow bietet folgende Vorteile:

- Data Scientists können fortschrittliche NetApp-Datenmanagement-Vorgänge, wie das Erstellen von Snapshots und Klonen, direkt von einem Jupyter Notebook aus durchführen.
- Erweiterte NetApp Datenmanagement-Operationen wie das Erstellen von Snapshots und Klonen können mithilfe des Kubeflow Pipelines-Frameworks in automatisierte Workflows integriert werden.

Siehe "Kubeflow Beispiele" Im Abschnitt zum NetApp Data Science Toolkit GitHub Repository finden Sie weitere Informationen zur Verwendung des Toolkit mit Kubeflow.

Beispiel-Workflow – Trainieren eines Bilderkennungsmodells mit Kubeflow und dem NetApp DataOps Toolkit

In diesem Abschnitt werden die Schritte beschrieben, die bei der Schulung und Bereitstellung eines Neuronalen Netzwerks zur Bilderkennung mit Kubeflow und dem NetApp DataOps Toolkit erforderlich sind. Dies soll als Beispiel dienen, um eine Trainingsaufgabe zu zeigen, die NetApp Storage integriert.

Voraussetzungen

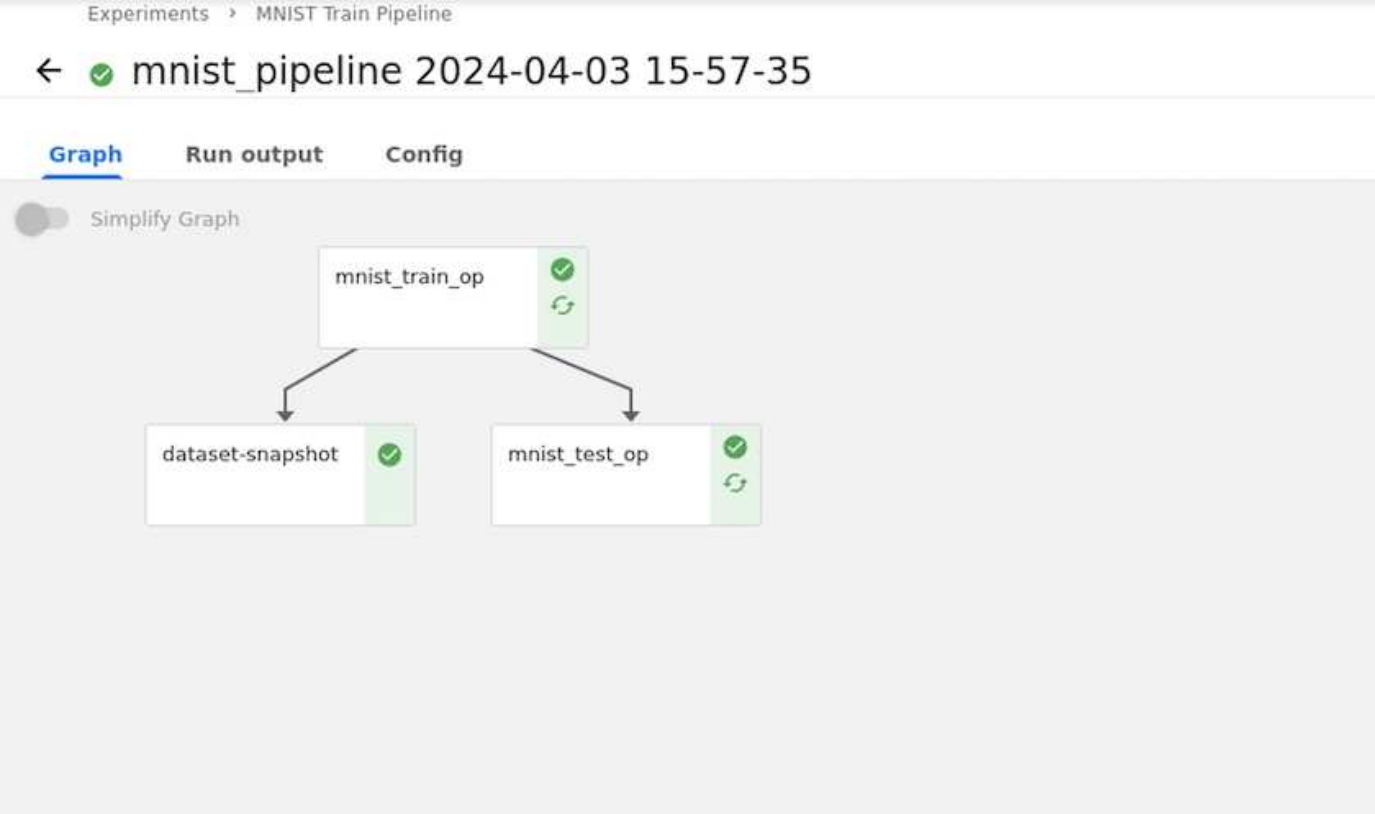
Erstellen Sie eine Dockdatei mit den erforderlichen Konfigurationen für die Zug- und Testschritte innerhalb der Kubeflow-Pipeline.
Hier ist ein Beispiel für eine Dockerdatei -

```
FROM pytorch/pytorch:latest
RUN pip install torchvision numpy scikit-learn matplotlib tensorboard
WORKDIR /app
COPY . /app
COPY train_mnist.py /app/train_mnist.py
CMD ["python", "train_mnist.py"]
```

Installieren Sie je nach Ihren Anforderungen alle erforderlichen Bibliotheken und Pakete, die zum Ausführen des Programms erforderlich sind. Bevor Sie das Machine-Learning-Modell trainieren, wird davon ausgegangen, dass Sie bereits über eine funktionierende Kubeflow-Implementierung verfügen.

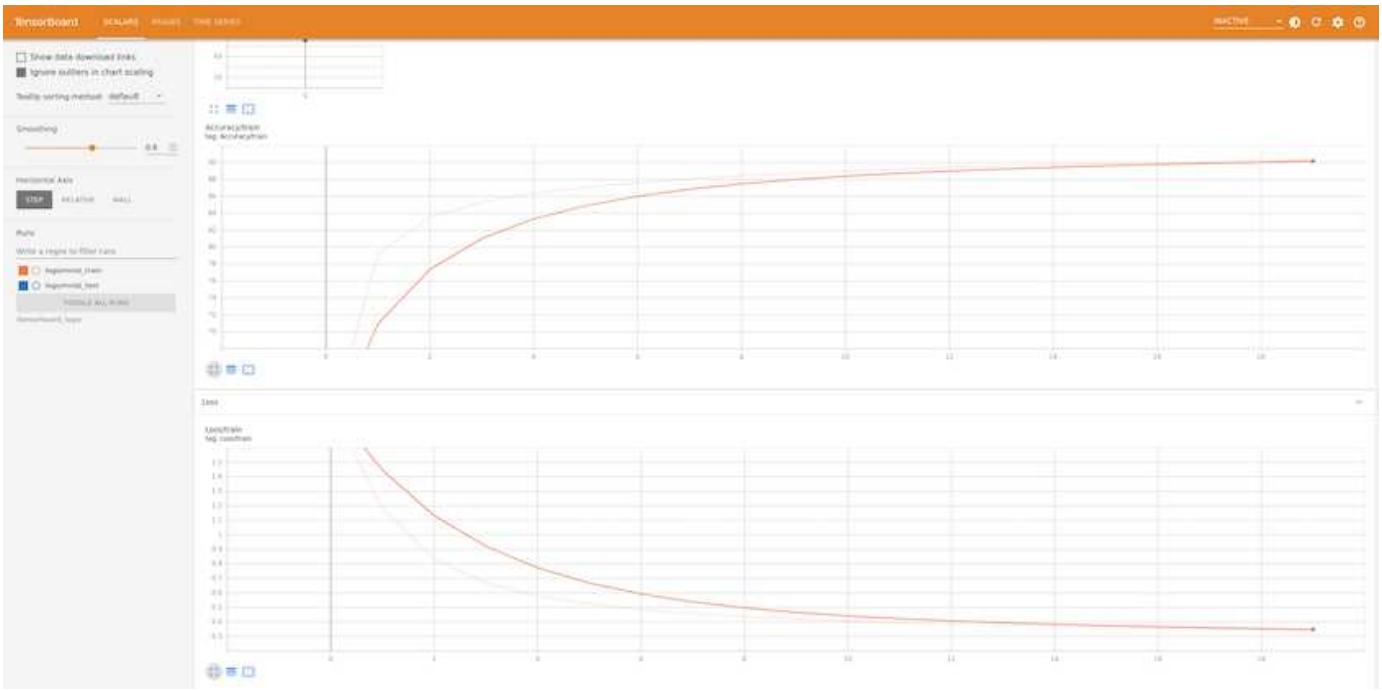
Trainieren Sie einen kleinen NN auf MNIST-Daten mit PyTorch- und Kubeflow-Pipelines

Wir verwenden das Beispiel eines kleinen neuronalen Netzwerks, das auf MNIST-Daten trainiert ist. Der MNIST-Datensatz besteht aus handschriftlichen Bildern von Ziffern von 0 bis 9. Die Bilder sind 28x28 Pixel groß. Der Datensatz ist in 60,000 Zug-Bilder und 10,000 Validierungsbilder unterteilt. Das für dieses Experiment verwendete Neuronale Netzwerk ist ein 2-schichtiges Feedforward-Netzwerk. Das Training wird mit Kubeflow Pipelines durchgeführt. Siehe Dokumentation "Hier" Finden Sie weitere Informationen. Unsere Kubeflow-Pipeline enthält das Docker-Image aus dem Abschnitt Voraussetzungen.



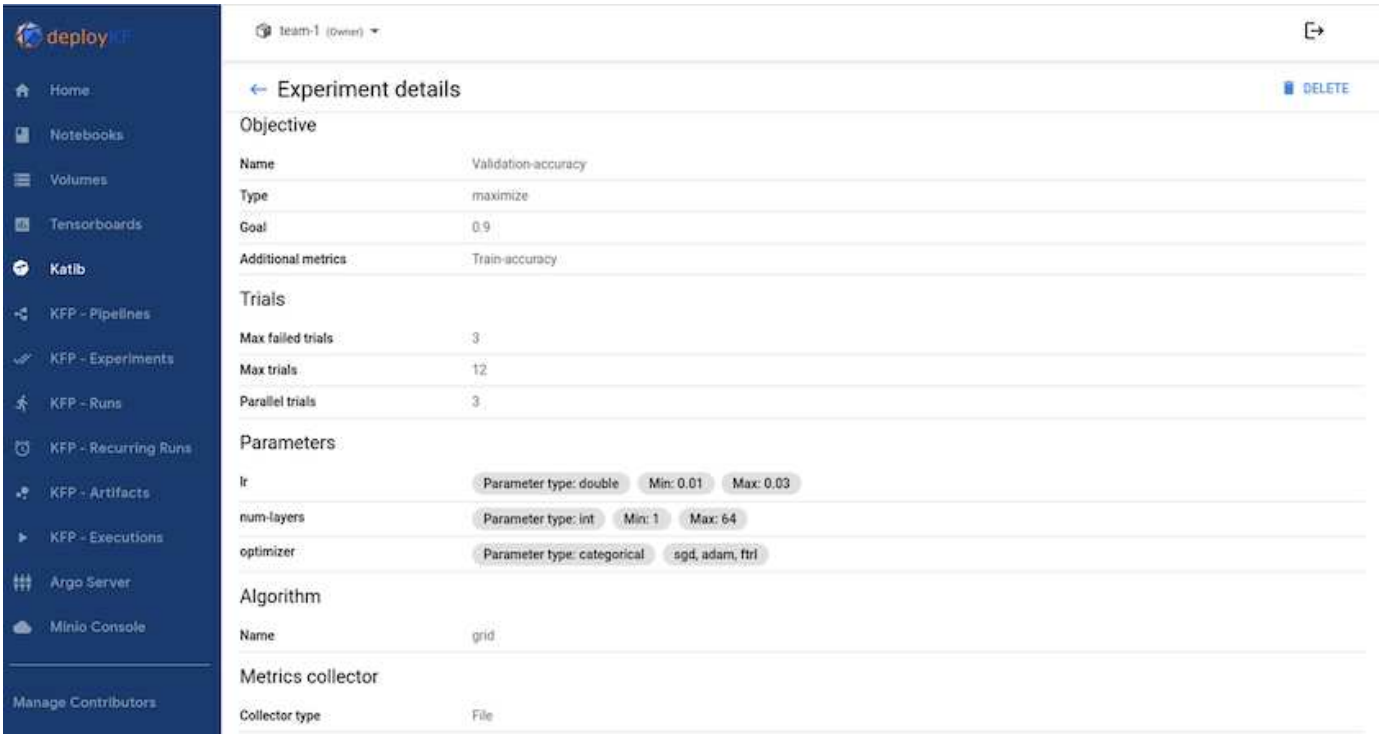
Visualisieren Sie Die Ergebnisse Mit Tensorboard

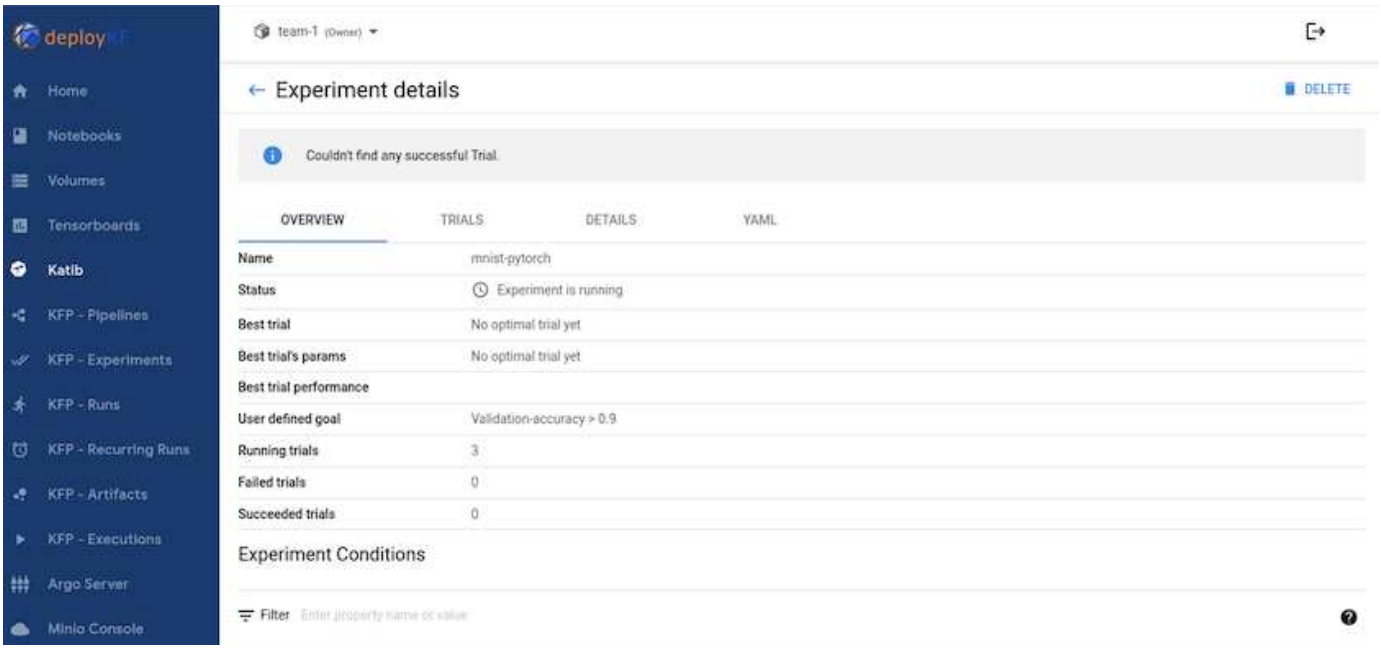
Sobald das Modell trainiert ist, können wir die Ergebnisse mit Tensorboard visualisieren. "Tensorboard" Ist als Funktion im Kubeflow Dashboard verfügbar. Sie können ein individuelles Tensorboard für Ihren Job erstellen. Ein Beispiel unten zeigt die Darstellung der Trainingsgenauigkeit vs. Anzahl der Epochen und Trainingsverluste vs Anzahl der Epochen.



Experimentieren Sie mit Hyperparametern mit Katib

"Katib" Ist ein Werkzeug innerhalb von Kubeflow, das verwendet werden kann, um mit den Modell-Hyperparametern zu experimentieren. Um ein Experiment zu erstellen, definieren Sie zunächst eine gewünschte Metrik/ein Ziel. Dies ist in der Regel die Prüfgenauigkeit. Nachdem die Metrik definiert wurde, wählen Sie Hyperparameter aus, mit denen Sie spielen möchten (Optimizer/Learning_Rate/Anzahl der Ebenen). Katib führt einen Hyperparameter-Sweep mit den benutzerdefinierten Werten durch, um die beste Kombination von Parametern zu finden, die die gewünschte Metrik erfüllen. Sie können diese Parameter in jedem Abschnitt der Benutzeroberfläche definieren. Alternativ können Sie eine **YAML**-Datei mit den erforderlichen Spezifikationen definieren. Unten ist eine Illustration eines Katib-Experiments -





Verwenden Sie NetApp-Snapshots, um Daten für die Rückverfolgbarkeit zu speichern

Während des Modelltrainings möchten wir eventuell eine Momentaufnahme des Trainingsdatensatzes zur Rückverfolgbarkeit speichern. Zu diesem Zweck können wir der Pipeline wie unten gezeigt einen Snapshot-Schritt hinzufügen. Zum Erstellen des Snapshots können wir den verwenden "NetApp DataOps Toolkit für Kubernetes".

```
@dsl.pipeline(
    name = 'MNIST Classification Pipeline',
    description = 'Train a simple NN for classification'
)
def mnist_pipeline():
    mnist_train_task = mnist_train_op()
    mnist_train_task.apply(
        kfp.onprem.mount_pvc('mnist-data', 'mnist-data-vol', '/mnt/data/')
    )

    mnist_test_task = mnist_test_op()
    mnist_test_task.apply(
        kfp.onprem.mount_pvc('mnist-data', 'mnist-data-vol', '/mnt/data/')
    )

    volume_snapshot_name = "mnist-pytorch-snapshot"
    dataset_snapshot = dsl.ContainerOp(
        name="dataset-snapshot",
        image="python:3.9",
        command=["/bin/bash", "-c"],
        arguments=["\
            python3 -m pip install netapp-dataops-k8s && \
            echo '" + volume_snapshot_name + "' > /volume_snapshot_name.txt && \
            netapp_dataops_k8s_cli.py create volume-snapshot --pvc-name=" + "mnist-data" + " --snapshot-name=" + str(volume_snapshot_name) + " --namespace={{workflow.namespace}}"],
        file_outputs={"volume_snapshot_name": "/volume_snapshot_name.txt"}
    )
    mnist_test_task.after(mnist_train_task)
    dataset_snapshot.after(mnist_train_task)
```

Siehe "Beispiel für das NetApp DataOps Toolkit für Kubeflow" Finden Sie weitere Informationen.

Apache Airflow

Apache Airflow Deployment

In diesem Abschnitt werden die Aufgaben beschrieben, die Sie zur Implementierung von Airflow in Ihrem Kubernetes-Cluster ausführen müssen.

> Es ist möglich, Airflow auf anderen Plattformen als Kubernetes bereitzustellen. Die Implementierung von Airflow 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 Astra Trident bereits in Ihrem Kubernetes-Cluster installiert und konfiguriert. Weitere Informationen zu Astra Trident finden Sie im "Astra Trident-Dokumentation".

Installieren Sie Helm

Der Luftstrom wird über Helm, einen beliebten Paketmanager für Kubernetes, implementiert. Bevor Sie Airflow bereitstellen, müssen Sie Helm auf dem Bereitstellungs-Jump-Host installieren. Um Helm auf dem Sprunghost für die Bereitstellung zu installieren, folgen Sie dem "Installationsanweisungen" In der offiziellen Helm-Dokumentation.

Standard-Kubernetes StorageClass festlegen

Bevor Sie Airflow bereitstellen, müssen Sie eine Standard-StorageClass in Ihrem Kubernetes-Cluster zuweisen. Der Airflow-Implementierungsprozess versucht, mithilfe der Standard-StorageClass neue persistente Volumes bereitzustellen. Wenn keine StorageClass als Standard-StorageClass festgelegt ist, schlägt die Bereitstellung fehl. Befolgen Sie die Anweisungen in, um eine Standard-StorageClass innerhalb Ihres Clusters festzulegen "Kubeflow Deployment" Abschnitt. Wenn Sie bereits eine Standard-StorageClass innerhalb Ihres Clusters festgelegt haben, können Sie diesen Schritt überspringen.

Verwenden Sie Helm zum Bereitstellen des Luftstroms

Um Airflow mithilfe von Helm in Ihren Kubernetes-Cluster zu implementieren, führen Sie die folgenden Aufgaben vom Bereitstellungs-Jump-Host aus:

1. Setzen Sie den Luftstrom mithilfe von Helm ein, indem Sie den folgen "Implementierungsanleitungen" Für das offizielle Airflow-Diagramm auf dem Artefakt-Hub. Die folgenden Beispielbefehle zeigen die Bereitstellung von Airflow mit Helm. Ändern, Hinzufügen und/oder Entfernen von Werten im `custom-values.yaml` Datei nach Bedarf, abhängig von Ihrer Umgebung und der gewünschten Konfiguration.

```
$ 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
```

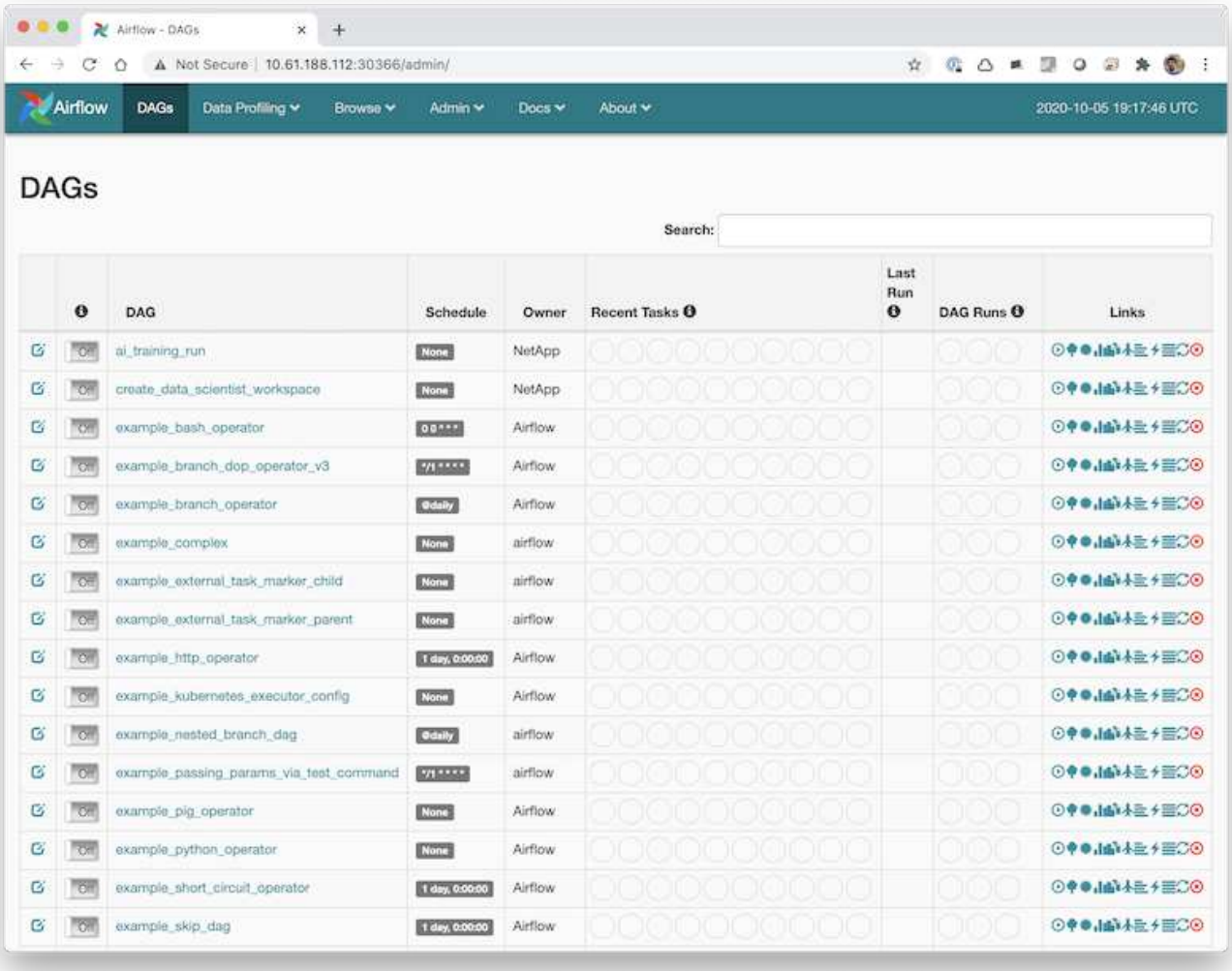
2. Vergewissern Sie sich, dass alle Airflow-Pods betriebsbereit sind. Es kann ein paar Minuten dauern, bis alle Pods beginnen.

```
$ 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. Rufen Sie die URL des Airflow Webservice ab, indem Sie die Anweisungen befolgen, die bei der Bereitstellung von Airflow mit Hilfe von Helm in Schritt 1 an der Konsole gedruckt wurden.

```
$ 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. Vergewissern Sie sich, dass Sie auf den Airflow Webservice zugreifen können.



Verwenden Sie das NetApp DataOps Toolkit mit Airflow

Der "NetApp DataOps Toolkit für Kubernetes" Kann in Verbindung mit Airflow verwendet werden. Die Verwendung des NetApp DataOps Toolkits mit Airflow ermöglicht die Integration von NetApp Datenmanagement-Vorgängen wie die Erstellung von Snapshots und Klonen in automatisierte, von Airflow orchestrierte Workflows.

Siehe "Beispiele Für Luftströmungen" Abschnitt im NetApp DataOps Toolkit GitHub Repository. Dort finden Sie weitere Informationen zur Verwendung des Toolkits mit Airflow.

Beispiel: Astra Trident Operations

Dieser Abschnitt enthält Beispiele für verschiedene Vorgänge, die Sie mit Astra Trident ausführen können.

Importieren eines vorhandenen Volumes

Wenn bereits vorhandene Volumes auf dem NetApp Storage-System/auf der Plattform vorhanden sind, die Sie in Containern innerhalb des Kubernetes Clusters einbinden möchten, die jedoch nicht an PVCs im Cluster

gebunden sind, müssen Sie diese Volumes importieren. Sie können diese Volumes mit der Trident-Funktion für den Volume-Import importieren.

Die folgenden Beispielbefehle zeigen den Import eines Volumes mit dem Namen an `pb_fg_all`. Weitere Informationen zu PVCs finden Sie im "Offizielle Kubernetes-Dokumentation". Weitere Informationen zur Importfunktion des Volumes finden Sie im "Trident Dokumentation".

An `accessModes` Der Wert von `ReadOnlyMany` Ist in den Beispiel-PVC-Spezifikationsdateien angegeben. Weitere Informationen zum `accessMode` Feld, siehe "Offizielle Kubernetes-Dokumentation".

```
$ 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
|
+---------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
$ 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
|
+-----------------------------------+---------
+-----------------------------------+----------
+---------------------------------------+--------+---------+
$ 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
```

Bereitstellung eines neuen Volumes

Mit Trident können Sie ein neues Volume auf Ihrem NetApp Storage-System oder Ihrer NetApp Plattform bereitstellen.

Stellen Sie ein neues Volume mit kubectl bereit

Die folgenden Beispielbefehle zeigen die Bereitstellung eines neuen FlexVol-Volumes mit kubectl.

An `accessModes` Der Wert von `ReadWriteMany` Wird in der folgenden Beispiel-PVC-Definitionsdatei angegeben. Weitere Informationen zum `accessMode` Feld, siehe "Offizielle Kubernetes-Dokumentation".

```
$ 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
tensorflow-results                  Bound     default-tensorflow-results-
2fd60   1073741824       RWX            ontap-ai-flexvols-retain
25h
```

Stellen Sie ein neues Volume mit dem NetApp DataOps Toolkit bereit

Außerdem können Sie mit dem NetApp DataOps Toolkit für Kubernetes ein neues Volume auf Ihrem Storage-System oder Ihrer NetApp Plattform bereitstellen. Das NetApp DataOps Toolkit für Kubernetes verwendet Trident zur Bereitstellung von Volumes, vereinfacht jedoch den Prozess für die Benutzer. Siehe "Dokumentation" Entsprechende Details.

Beispiel für hochperformante Jobs für AIPod Implementierungen

Single-Node-KI-Workload ausführen

Um einen Single-Node-KI- und -ML-Job in Ihrem Kubernetes-Cluster auszuführen, führen Sie die folgenden Aufgaben vom Bereitstellungs-Jump-Host aus: Mit Trident lässt sich ein Daten-Volume schnell und einfach erstellen, das möglicherweise mehrere Petabyte an Daten enthält und damit für Kubernetes-Workloads zugänglich ist. Wenn ein solches Daten-Volume über einen Kubernetes Pod zugänglich sein soll, geben Sie in der Pod-Definition einfach ein PVC an.

In diesem Abschnitt wird vorausgesetzt, dass Sie bereits den spezifischen KI- und ML-Workload in einem Container (im Docker Container-Format) bereitstellen, den Sie in Ihrem Kubernetes-Cluster ausführen möchten.

1. Die folgenden Beispielbefehle zeigen die Erstellung eines Kubernetes-Jobs für einen TensorFlow Benchmark-Workload, bei dem der ImageNet-Datensatz verwendet wird. Weitere Informationen zum ImageNet-Datensatz finden Sie im "ImageNet-Website".

   Dieses Beispieljob fordert acht GPUs an und kann daher auf einem einzelnen GPU-Worker-Node mit acht oder mehr GPUs ausgeführt werden. Dieser Beispieljob kann in einem Cluster eingereicht werden, für den ein Worker-Node mit acht oder mehr GPUs nicht vorhanden ist oder derzeit mit einem anderen Workload belegt ist. Wenn dies der Fall ist, bleibt der Job in einem ausstehenden Status, bis ein solcher Worker-Node verfügbar ist.

   Zusätzlich wird das Volume, das die erforderlichen Trainingsdaten enthält, zur Maximierung der Storage-Bandbreite zweimal innerhalb des POD angehängt, das diesen Job erstellt. Ein weiteres Volume wird ebenfalls im POD gemountet. Dieses zweite Volume wird zur Speicherung der Ergebnisse und Kennzahlen verwendet. Diese Volumes werden in der Jobdefinition unter Verwendung der Namen der VES referenziert. Weitere Informationen zu Kubernetes-Jobs finden Sie im "Offizielle Kubernetes-Dokumentation".

   An `emptyDir` Volumen mit einem `medium` Der Wert von `Memory` Ist in angehängt `/dev/shm` In dem POD, den dieser Beispieljob erzeugt. Die Standardgröße des `/dev/shm` Das virtuelle Volume, das automatisch durch die Docker Container-Laufzeit erstellt wird, kann manchmal nicht ausreichen, um die Anforderungen von TensorFlow zu erfüllen. Montage an `emptyDir` Das Volumen wie im folgenden Beispiel bietet eine ausreichend große Menge `/dev/shm` Virtuelles Volume: Finden Sie weitere Informationen zu `emptyDir` Volumes, siehe "Offizielle Kubernetes-Dokumentation".

   Der einzelne Container, der in dieser Beispieljobdefinition angegeben wird, wird als angegeben `securityContext > privileged` Der Wert von `true`. Dieser Wert bedeutet, dass der Container effektiv Root-Zugriff auf dem Host hat. Diese Annotation wird in diesem Fall verwendet, da der spezifische Workload ausgeführt wird und den Root-Zugriff erfordert. Insbesondere für einen Vorgang mit klarem Cache, der von dem Workload ausgeführt wird, ist ein Root-Zugriff erforderlich. Ob oder nicht `privileged: true` Eine Anmerkung ist erforderlich, abhängig von den Anforderungen des spezifischen Workloads, die Sie ausführen.

```
$ 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. Vergewissern Sie sich, dass der in Schritt 1 erstellte Job korrekt ausgeführt wird. Der folgende Beispielbefehl bestätigt, dass für den Job ein einzelner Pod erstellt wurde, wie in der Jobdefinition angegeben, und dass dieser Pod derzeit auf einem der GPU-Worker-Nodes ausgeführt wird.

```
$ 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. Bestätigen Sie, dass der in Schritt 1 erstellte Job erfolgreich abgeschlossen wurde. Mit den folgenden Beispielbefehlen wird bestätigt, dass der Job erfolgreich abgeschlossen wurde.

```
$ 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. **Optional:** Aufräumen von Auftragsartefakten. Die folgenden Beispielbefehle zeigen das Löschen des in Schritt 1 erstellten Jobobjekts.

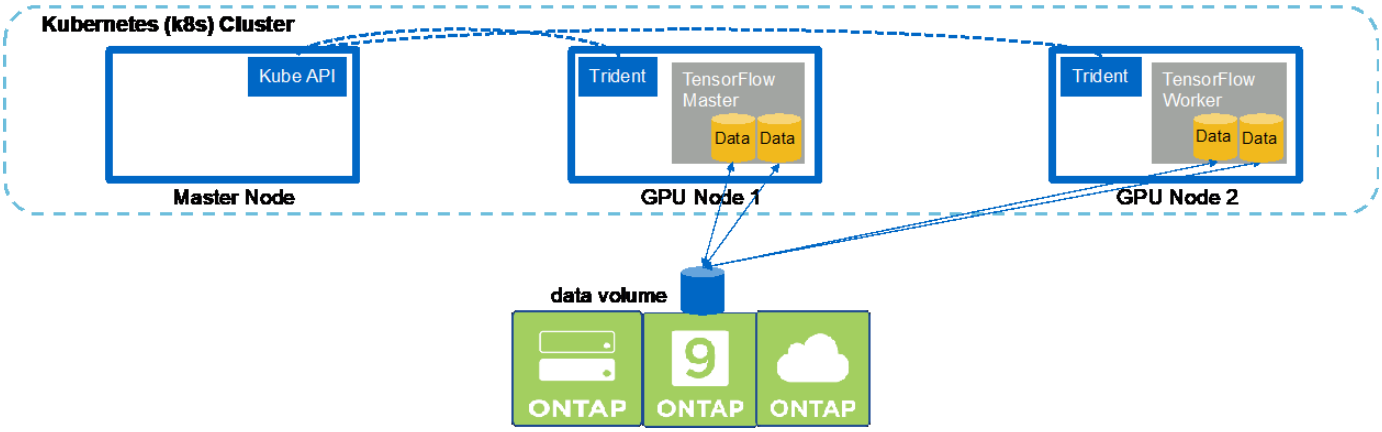
   Wenn Sie das Jobobjekt löschen, löscht Kubernetes automatisch alle zugehörigen Pods.

```
$ 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.
```

Ausführung eines synchronen, verteilten KI-Workloads

Um einen synchronen AI- und ML-Job mit mehreren Nodes in Ihrem Kubernetes-Cluster auszuführen, führen Sie die folgenden Aufgaben auf dem Jump-Host der Implementierung durch. Mit diesem Prozess können Sie auf einem NetApp Volume gespeicherte Daten nutzen und mehr GPUs verwenden, als ein einzelner Worker-Node bieten kann. Die folgende Abbildung zeigt einen synchronen, verteilten KI-Job.

Synchrone, verteilte Jobs können dazu beitragen, die Performance und die Trainingsgenauigkeit im Vergleich zu asynchronen, verteilten Jobs zu steigern. Eine Diskussion über die vor- und Nachteile von synchronen Jobs gegenüber asynchronen Jobs geht nicht in dieses Dokument über.



1. Die folgenden Beispielbefehle zeigen die Erstellung eines Workers, der an der synchronen, verteilten Ausführung desselben TensorFlow Benchmark-Jobs beteiligt ist, der auf einem einzelnen Node im Beispiel im Abschnitt ausgeführt wurde "Single-Node-KI-Workload ausführen". In diesem speziellen Beispiel wird nur ein einziger Worker bereitgestellt, da der Job über zwei Worker-Nodes ausgeführt wird.

Dieses Beispiel für die Implementierung eines Mitarbeiters fordert acht GPUs an und kann so auf einem einzelnen GPU-Worker-Node mit acht oder mehr GPUs ausgeführt werden. Wenn Ihre GPU-Worker-Nodes mehr als acht GPUs aufweisen, um die Performance zu maximieren, könnte die Anzahl dieser GPUs erhöht werden, die der Funktion der Worker-Nodes entsprechen. Weitere Informationen über Kubernetes-Implementierungen finden Sie im "Offizielle Kubernetes-Dokumentation".

In diesem Beispiel wird eine Kubernetes-Implementierung erstellt, da dieser spezifische Container-Worker niemals eigenständig abgeschlossen sein würde. Daher macht es keinen Sinn, es durch die Verwendung des Kubernetes Job-Konstrukts zu implementieren. Wenn Ihr Mitarbeiter für sich selbst konzipiert oder geschrieben wurde, ist es möglicherweise sinnvoll, das Job-Konstrukt für die Bereitstellung Ihres Mitarbeiters zu verwenden.

Dem POD, der in diesem Beispiel der Implementierungsspezifikation angegeben ist, wird eine zugewiesen `hostNetwork` Der Wert von `true`. Dieser Wert bedeutet, dass der Pod den Netzwerk-Stack des Host-Mitarbeiters-Nodes anstelle des virtuellen Netzwerk-Stacks verwendet, den Kubernetes normalerweise für jeden Pod erstellt. Diese Annotation wird in diesem Fall verwendet, da der spezifische Workload auf Open MPI, NCCL und Horovod angewiesen ist, um den Workload in einer synchronen, verteilten Art und Weise auszuführen. Daher ist für diese Lösung der Zugriff auf den Host-Netzwerk-Stack erforderlich. Eine Diskussion über Open MPI, NCCL und Horovod geht nicht in dieses Dokument über. Ob oder nicht `hostNetwork: true` Eine Anmerkung ist erforderlich, abhängig von den Anforderungen des spezifischen Workloads, die Sie ausführen. Weitere Informationen zum `hostNetwork` Feld, siehe "Offizielle Kubernetes-Dokumentation".

```
$ 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. Bestätigen Sie, dass die in Schritt 1 erstellte Workers-Bereitstellung erfolgreich gestartet wurde. Die folgenden Beispielbefehle bestätigen, dass ein Pod für einzelne Mitarbeiter für die Implementierung erstellt wurde, wie in der Implementierungsdefinition angegeben, und dass dieser Pod derzeit auf einem der GPU-Worker-Nodes ausgeführt wird.

```
$ 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. Kubernetes-Job für einen Master erstellen, der zu Beginn startet, an dem er teilnimmt und die Ausführung des synchronen Jobs mit mehreren Nodes verfolgt. Die folgenden Beispielbefehle erzeugen einen Master, der abstartet, an dem teilnimmt und die synchrone, verteilte Ausführung desselben TensorFlow-Benchmark-Jobs verfolgt, der auf einem einzelnen Node im Beispiel im Abschnitt ausgeführt wurde "Single-Node-KI-Workload ausführen".

   Dieses Beispiel-Master-Job fordert acht GPUs an, wodurch auf einem einzelnen GPU-Worker-Node mit acht oder mehr GPUs ausgeführt werden kann. Wenn Ihre GPU-Worker-Nodes mehr als acht GPUs aufweisen, um die Performance zu maximieren, könnte die Anzahl dieser GPUs erhöht werden, die der Funktion der Worker-Nodes entsprechen.

   Der Master-Pod, der in dieser Beispiel-Jobdefinition angegeben wird, wird A zugewiesen `hostNetwork` Der Wert von `true`, So wie der Arbeiter POD wurde ein `hostNetwork` Der Wert von `true` In Schritt 1. Weitere Informationen dazu, warum dieser Wert notwendig ist, finden Sie in Schritt 1.

```
$ 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. Vergewissern Sie sich, dass der in Schritt 3 erstellte Master-Job korrekt ausgeführt wird. Der folgende Beispielbefehl bestätigt, dass für den Job ein einzelner Master-Pod erstellt wurde, wie in der Jobdefinition angegeben, und dass dieser Pod derzeit auf einem der GPU-Worker-Nodes ausgeführt wird. Sie sollten auch sehen, dass der Worker Pod, den Sie ursprünglich in Schritt 1 gesehen haben, noch läuft und dass die Master- und Worker-Pods auf unterschiedlichen Nodes ausgeführt werden.

```
$ 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. Vergewissern Sie sich, dass der in Schritt 3 erstellte Masterjob erfolgreich abgeschlossen wurde. Mit den folgenden Beispielbefehlen wird bestätigt, dass der Job erfolgreich abgeschlossen wurde.

```
$ 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. Löschen Sie die Mitarbeiterbereitstellung, wenn Sie sie nicht mehr benötigen. Die folgenden Beispielbefehle zeigen das Löschen des in Schritt 1 erstellten Workers Deployment-Objekts.

Wenn Sie das Bereitstellungsobjekt für Mitarbeiter löschen, löscht Kubernetes automatisch alle zugehörigen „Worker“-Pods.

```
$ 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. **Optional:** Säubern Sie die Master Job Artefakte. Die folgenden Beispielbefehle zeigen das Löschen des in Schritt 3 erstellten Master-Jobobjekts.

   Wenn Sie das Master-Job-Objekt löschen, löscht Kubernetes automatisch alle zugehörigen Master-Pods.

```
$ 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.
```