Open Source MLOps mit NetApp

NetApp artificial intelligence solutions

NetApp
December 04, 2025

Inhalt

Open Source MLOps mit NetApp

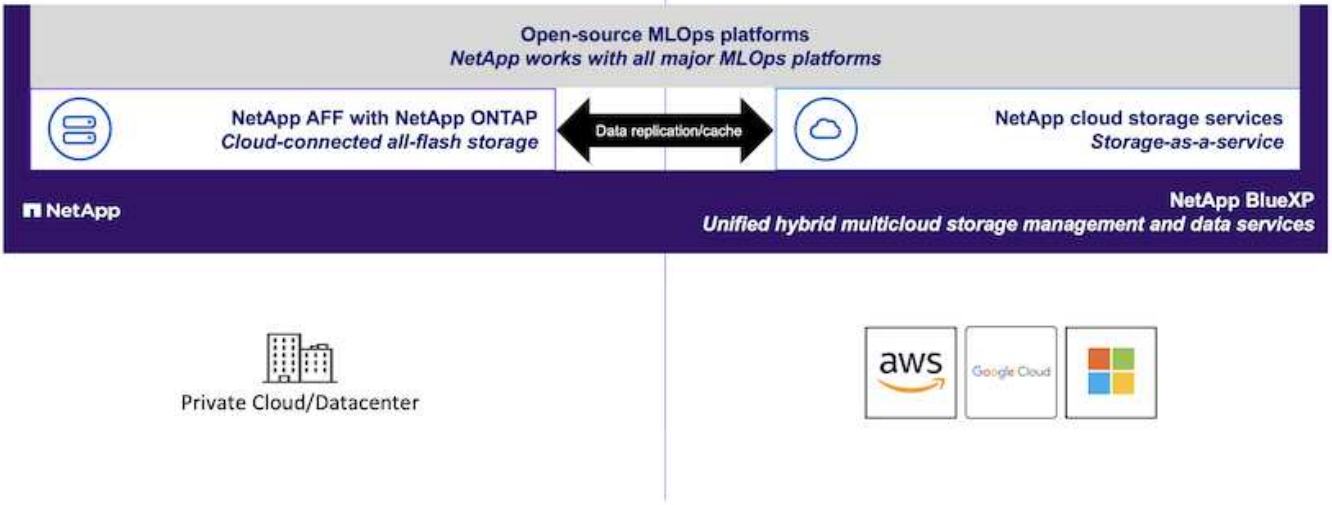
Open Source MLOps mit NetApp

Mike Oglesby, NetApp Sufian Ahmad, NetApp Rick Huang, NetApp Mohan Acharya, NetApp

Unternehmen und Organisationen aller Größen und aus vielen Branchen setzen auf künstliche Intelligenz (KI), um reale Probleme zu lösen, innovative Produkte und Dienstleistungen anzubieten und sich auf einem zunehmend wettbewerbsorientierten Markt einen Vorteil zu verschaffen. Viele Organisationen greifen auf Open-Source-MLOps-Tools zurück, um mit dem rasanten Innovationstempo in der Branche Schritt zu halten. Diese Open-Source-Tools bieten erweiterte Funktionen und hochmoderne Features, berücksichtigen jedoch häufig nicht die Datenverfügbarkeit und Datensicherheit. Leider bedeutet dies, dass hochqualifizierte Datenwissenschaftler viel Zeit damit verbringen müssen, auf den Zugriff auf Daten zu warten oder auf die Fertigstellung grundlegender datenbezogener Vorgänge. Durch die Kombination beliebter Open-Source-MLOps-Tools mit einer intelligenten Dateninfrastruktur von NetApp können Unternehmen ihre Datenpipelines beschleunigen, was wiederum ihre KI-Initiativen beschleunigt. Sie können den Wert ihrer Daten erschließen und gleichzeitig sicherstellen, dass diese geschützt und sicher bleiben. Diese Lösung demonstriert die Kombination von NetApp Datenverwaltungsfunktionen mit mehreren gängigen Open-Source-Tools und -Frameworks, um diese Herausforderungen zu bewältigen.

Die folgende Liste hebt einige wichtige Funktionen hervor, die durch diese Lösung ermöglicht werden:

- Benutzer können schnell neue Datenvolumes mit hoher Kapazität und Entwicklungsarbeitsbereiche bereitstellen, die durch leistungsstarken, skalierbaren NetApp Speicher unterstützt werden.
- Benutzer können Datenvolumes und Entwicklungsarbeitsbereiche mit hoher Kapazität nahezu augenblicklich klonen, um Experimente oder schnelle Iterationen zu ermöglichen.
- Benutzer können Snapshots von Datenvolumes mit hoher Kapazität und Entwicklungsarbeitsbereichen nahezu augenblicklich für Backups und/oder zur Rückverfolgbarkeit/Baselining speichern.



Ein typischer MLOps-Workflow umfasst Entwicklungsarbeitsbereiche, in der Regel in Form von"Jupyter-Notebooks" ; Experimentverfolgung; automatisierte Trainingspipelines; Datenpipelines; und Inferenz/Bereitstellung. Diese Lösung hebt mehrere verschiedene Tools und Frameworks hervor, die unabhängig oder zusammen verwendet werden können, um die verschiedenen Aspekte des Workflows zu berücksichtigen. Wir demonstrieren außerdem die Kombination der NetApp Datenverwaltungsfunktionen mit jedem dieser Tools. Diese Lösung soll Bausteine bieten, aus denen eine Organisation einen benutzerdefinierten MLOps-Workflow erstellen kann, der speziell auf ihre Anwendungsfälle und Anforderungen zugeschnitten ist.

Die folgenden Tools/Frameworks sind in dieser Lösung enthalten:

- "Apache Airflow"
- "JupyterHub"
- "Kubeflow"
- "MLflow"

In der folgenden Liste werden gängige Muster für die unabhängige oder kombinierte Bereitstellung dieser Tools beschrieben.

- Stellen Sie JupyterHub, MLflow und Apache Airflow gemeinsam bereit – JupyterHub für"Jupyter-Notebooks" , MLflow für die Experimentverfolgung und Apache Airflow für automatisiertes Training und Datenpipelines.
- Stellen Sie Kubeflow und Apache Airflow gemeinsam bereit – Kubeflow für"Jupyter-Notebooks" , Experimentverfolgung, automatisierte Trainingspipelines und Inferenz; und Apache Airflow für Datenpipelines.
- Stellen Sie Kubeflow als All-in-One-MLOps-Plattformlösung bereit für"Jupyter-Notebooks" , Experimentverfolgung, automatisiertes Training und Datenpipelines sowie Inferenz.

Technologieübersicht

Dieser Abschnitt konzentriert sich auf den Technologieüberblick für OpenSource MLOps mit NetApp.

Künstliche Intelligenz

KI ist eine Disziplin der Informatik, in der Computer darauf trainiert werden, die kognitiven Funktionen des menschlichen Geistes nachzuahmen. KI-Entwickler trainieren Computer, auf eine Weise zu lernen und Probleme zu lösen, die der des Menschen ähnelt oder dieser sogar überlegen ist. Deep Learning und maschinelles Lernen sind Teilgebiete der KI. Organisationen setzen zunehmend KI, ML und DL ein, um ihre kritischen Geschäftsanforderungen zu unterstützen. Einige Beispiele sind wie folgt:

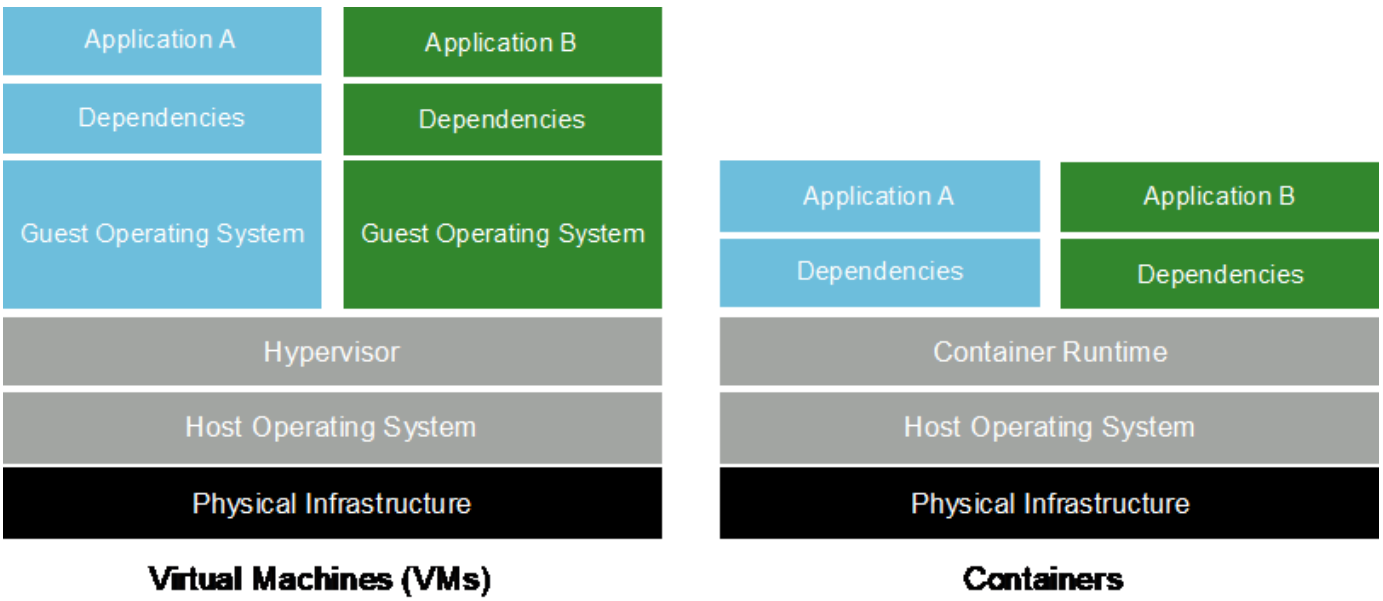
- Analyse großer Datenmengen, um bisher unbekannte Geschäftserkenntnisse zu gewinnen
- Direkte Interaktion mit Kunden durch die Verarbeitung natürlicher Sprache
- Automatisierung verschiedener Geschäftsprozesse und -funktionen

Moderne KI-Trainings- und Inferenz-Workloads erfordern massiv parallele Rechenkapazitäten. Daher werden GPUs zunehmend zur Ausführung von KI-Operationen eingesetzt, da die Parallelverarbeitungskapazitäten von GPUs denen von Allzweck-CPUs weit überlegen sind.

Behälter

Container sind isolierte Benutzerbereichsinstanzen, die auf einem gemeinsam genutzten Host-Betriebssystemkernel ausgeführt werden. Die Nutzung von Containern nimmt rasant zu. Container bieten viele der gleichen Vorteile der Anwendungs-Sandboxing-Funktion wie virtuelle Maschinen (VMs). Da jedoch die Hypervisor- und Gastbetriebssystemebenen, auf die VMs angewiesen sind, eliminiert wurden, sind Container wesentlich leichter. Die folgende Abbildung zeigt eine Visualisierung von virtuellen Maschinen im Vergleich zu Containern.

Container ermöglichen außerdem die effiziente Verpackung von Anwendungsabhängigkeiten, Laufzeiten usw. direkt mit einer Anwendung. Das am häufigsten verwendete Container-Verpackungsformat ist der Docker-Container. Eine Anwendung, die im Docker-Containerformat containerisiert wurde, kann auf jedem Computer ausgeführt werden, auf dem Docker-Container ausgeführt werden können. Dies gilt auch dann, wenn die Abhängigkeiten der Anwendung nicht auf dem Computer vorhanden sind, da alle Abhängigkeiten im Container selbst verpackt sind. Weitere Informationen finden Sie im "Docker-Website" .



Kubernetes

Kubernetes ist eine Open-Source-Plattform zur verteilten Container-Orchestrierung, die ursprünglich von Google entwickelt wurde und jetzt von der Cloud Native Computing Foundation (CNCF) gepflegt wird. Kubernetes ermöglicht die Automatisierung von Bereitstellungs-, Verwaltungs- und Skalierungsfunktionen für containerisierte Anwendungen. In den letzten Jahren hat sich Kubernetes zur dominierenden Plattform für die Container-Orchestrierung entwickelt. Weitere Informationen finden Sie im "Kubernetes-Website" .

NetApp Trident

"Trident"ermöglicht die Nutzung und Verwaltung von Speicherressourcen auf allen gängigen NetApp Speicherplattformen, in der öffentlichen Cloud oder vor Ort, einschließlich ONTAP (AFF, FAS, Select, Cloud, Amazon FSx ONTAP), Azure NetApp Files Dienst und Google Cloud NetApp Volumes. Trident ist ein Container Storage Interface (CSI)-kompatibler dynamischer Speicher-Orchestrator, der nativ in Kubernetes integriert ist.

NetApp DataOps Toolkit

Der"NetApp DataOps Toolkit" ist ein Python-basiertes Tool, das die Verwaltung von Entwicklungs-/Schulungsarbeitsbereichen und Inferenzservern vereinfacht, die durch leistungsstarken, skalierbaren NetApp Speicher unterstützt werden. Zu den wichtigsten Funktionen gehören:

- Stellen Sie schnell neue Arbeitsbereiche mit hoher Kapazität bereit, die durch leistungsstarken, skalierbaren NetApp Speicher unterstützt werden.
- Klonen Sie Arbeitsbereiche mit hoher Kapazität nahezu sofort, um Experimente oder schnelle Iterationen zu ermöglichen.
- Speichern Sie nahezu sofort Snapshots von Arbeitsbereichen mit hoher Kapazität für Backups und/oder zur Rückverfolgbarkeit/Baselining.
- Stellen Sie Datenvolumes mit hoher Kapazität und hoher Leistung nahezu sofort bereit, klonen Sie sie und erstellen Sie Snapshots davon.

Apache Airflow

Apache Airflow ist eine Open-Source-Plattform für Workflow-Management, die die programmgesteuerte Erstellung, Planung und Überwachung komplexer Unternehmens-Workflows ermöglicht. Es wird häufig zur Automatisierung von ETL- und Datenpipeline-Workflows verwendet, ist jedoch nicht auf diese Arten von Workflows beschränkt. Das Airflow-Projekt wurde von Airbnb ins Leben gerufen, erfreut sich seitdem jedoch großer Beliebtheit in der Branche und steht nun unter der Schirmherrschaft der Apache Software Foundation. Airflow ist in Python geschrieben, Airflow-Workflows werden über Python-Skripte erstellt und Airflow ist nach dem Prinzip „Konfiguration als Code“ konzipiert. Viele Airflow-Benutzer in Unternehmen führen Airflow jetzt auf Kubernetes aus.

Gerichtete azyklische Graphen (DAGs)

In Airflow werden Workflows als gerichtete azyklische Graphen (DAGs) bezeichnet. DAGs bestehen aus Aufgaben, die je nach DAG-Definition nacheinander, parallel oder in einer Kombination aus beidem ausgeführt werden. Der Airflow-Scheduler führt einzelne Aufgaben auf einer Reihe von Workern aus und hält sich dabei an die in der DAG-Definition angegebenen Abhängigkeiten auf Aufgabenebene. DAGs werden über Python-Skripte definiert und erstellt.

Jupyter Notebook

Jupyter-Notebooks sind Wiki-ähnliche Dokumente, die sowohl Live-Code als auch beschreibenden Text enthalten. Jupyter Notebooks werden in der KI- und ML-Community häufig zum Dokumentieren, Speichern und Teilen von KI- und ML-Projekten verwendet. Weitere Informationen zu Jupyter Notebooks finden Sie auf der "Jupyter-Website" .

Jupyter Notebook Server

Ein Jupyter Notebook Server ist eine Open-Source-Webanwendung, mit der Benutzer Jupyter Notebooks erstellen können.

JupyterHub

JupyterHub ist eine Mehrbenutzeranwendung, die es einem einzelnen Benutzer ermöglicht, seinen eigenen Jupyter Notebook-Server bereitzustellen und darauf zuzugreifen. Weitere Informationen zu JupyterHub finden Sie auf der "JupyterHub-Website" .

MLflow

MLflow ist eine beliebte Open-Source-Plattform für das KI-Lebenszyklusmanagement. Zu den Hauptfunktionen von MLflow gehören die Verfolgung von KI/ML-Experimenten und ein KI/ML-Modell-Repository. Weitere Informationen zu MLflow finden Sie auf der "MLflow-Website" .

Kubeflow

Kubeflow ist ein Open-Source-KI- und ML-Toolkit für Kubernetes, das ursprünglich von Google entwickelt wurde. Das Kubeflow-Projekt macht die Bereitstellung von KI- und ML-Workflows auf Kubernetes einfach, portabel und skalierbar. Kubeflow abstrahiert die Feinheiten von Kubernetes und ermöglicht es Datenwissenschaftlern, sich auf das zu konzentrieren, was sie am besten können – Datenwissenschaft. Eine Visualisierung finden Sie in der folgenden Abbildung. Kubeflow ist eine gute Open-Source-Option für Organisationen, die eine All-in-One-MLOps-Plattform bevorzugen. Weitere Informationen finden Sie im "Kubeflow-Website" .

Kubeflow-Pipelines

Kubeflow-Pipelines sind eine Schlüsselkomponente von Kubeflow. Kubeflow Pipelines sind eine Plattform und ein Standard zum Definieren und Bereitstellen portabler und skalierbarer KI- und ML-Workflows. Weitere Informationen finden Sie im "offizielle Kubeflow-Dokumentation" .

Kubeflow-Notebooks

Kubeflow vereinfacht die Bereitstellung und Implementierung von Jupyter Notebook-Servern auf Kubernetes. Weitere Informationen zu Jupyter Notebooks im Kontext von Kubeflow finden Sie im "offizielle Kubeflow-Dokumentation" .

Katib

Katib ist ein Kubernetes-natives Projekt für automatisiertes maschinelles Lernen (AutoML). Katib unterstützt Hyperparameter-Tuning, Early Stopping und die Suche nach neuronalen Architekturen (NAS). Katib ist das Projekt, das unabhängig von Frameworks für maschinelles Lernen (ML) ist. Es kann Hyperparameter von Anwendungen optimieren, die in einer beliebigen Sprache der Wahl des Benutzers geschrieben sind, und unterstützt nativ viele ML-Frameworks wie TensorFlow, MXNet, PyTorch, XGBoost und andere. Katib unterstützt zahlreiche verschiedene AutoML-Algorithmen, wie etwa Bayes-Optimierung, Tree of Parzen-

Schätzer, Zufallssuche, 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 Kubeflow-Dokumentation" .

NetApp ONTAP

ONTAP 9, die neueste Generation der Speicherverwaltungssoftware von NetApp, ermöglicht Unternehmen die Modernisierung ihrer Infrastruktur und den Übergang zu einem Cloud-fähigen Rechenzentrum. Durch die Nutzung branchenführender Datenverwaltungsfunktionen ermöglicht ONTAP die Verwaltung und den Schutz von Daten mit einem einzigen Satz von Tools, unabhängig davon, wo sich diese Daten befinden. Sie können Daten auch frei dorthin verschieben, wo sie benötigt werden: an den Rand, in den Kern oder in die Cloud. ONTAP 9 umfasst zahlreiche Funktionen, die die Datenverwaltung vereinfachen, kritische Daten beschleunigen und schützen und Infrastrukturfunktionen der nächsten Generation in Hybrid-Cloud-Architekturen ermöglichen.

Vereinfachen Sie die Datenverwaltung

Das Datenmanagement ist für den IT-Betrieb in Unternehmen und für Datenwissenschaftler von entscheidender Bedeutung, damit für KI-Anwendungen und das Training von KI/ML-Datensätzen die richtigen Ressourcen verwendet werden. Die folgenden zusätzlichen Informationen zu NetApp -Technologien fallen nicht in den Geltungsbereich dieser Validierung, können jedoch je nach Bereitstellung relevant sein.

Die ONTAP Datenmanagementsoftware umfasst die folgenden Funktionen zur Optimierung und Vereinfachung von Abläufen und zur Senkung Ihrer Gesamtbetriebskosten:

- Inline-Datenkomprimierung und erweiterte Deduplizierung. Durch die Datenkomprimierung wird der verschwendete Speicherplatz in Speicherblöcken reduziert und durch die Deduplizierung wird die effektive Kapazität erheblich erhöht. Dies gilt für lokal gespeicherte Daten und für in der Cloud gespeicherte Daten.
- Minimale, maximale und adaptive Dienstqualität (AQoS). Durch granulare Quality of Service (QoS)-Kontrollen wird die Aufrechterhaltung des Leistungsniveaus kritischer Anwendungen in Umgebungen mit hoher gemeinsamer Nutzung unterstützt.
- NetApp FabricPool. Bietet automatisches Tiering von Cold Data für öffentliche und private Cloud-Speicheroptionen, einschließlich Amazon Web Services (AWS), Azure und der NetApp StorageGRID -Speicherlösung. Weitere Informationen zu FabricPool finden Sie unter "TR-4598: Best Practices für FabricPool" .

Beschleunigen und schützen Sie Daten

ONTAP bietet ein Höchstmaß an Leistung und Datenschutz und erweitert diese Funktionen auf folgende Weise:

- Leistung und geringere Latenz. ONTAP bietet den höchstmöglichen Durchsatz bei der geringstmöglichen Latenz.
- Datenschutz. ONTAP bietet integrierte Datenschutzfunktionen mit gemeinsamer Verwaltung auf allen Plattformen.
- NetApp Volume Encryption (NVE). ONTAP bietet native Verschlüsselung auf Volume-Ebene mit Unterstützung für integriertes und externes Schlüsselmanagement.
- Mandantenfähigkeit und Multifaktor-Authentifizierung. ONTAP ermöglicht die gemeinsame Nutzung von Infrastrukturressourcen mit höchster Sicherheit.

Zukunftssichere Infrastruktur

ONTAP unterstützt Sie mit den folgenden Funktionen bei der Erfüllung anspruchsvoller und sich ständig ändernder Geschäftsanforderungen:

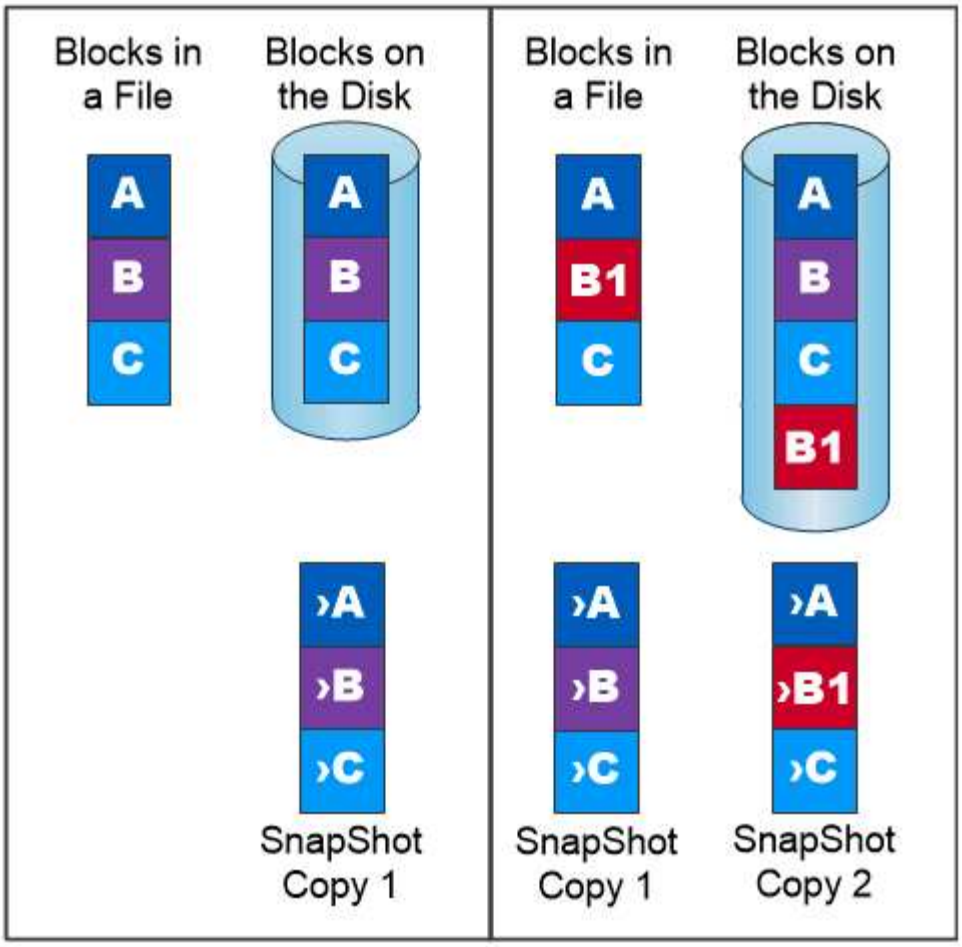
- Nahtlose Skalierung und unterbrechungsfreier Betrieb. ONTAP unterstützt die unterbrechungsfreie Kapazitätserweiterung bestehender Controller und Scale-Out-Cluster. Kunden können ohne kostspielige Datenmigrationen oder Ausfälle auf die neuesten Technologien upgraden.
- Cloud-Verbindung. ONTAP ist die Speicherverwaltungssoftware mit der stärksten Cloud-Anbindung und bietet Optionen für softwaredefinierten Speicher und Cloud-native Instanzen in allen öffentlichen Clouds.
- Integration mit neuen Anwendungen. ONTAP bietet Datendienste der Enterprise-Klasse für Plattformen und Anwendungen der nächsten Generation, wie etwa autonome Fahrzeuge, Smart Cities und Industrie 4.0, und nutzt dabei dieselbe Infrastruktur, die auch bestehende Unternehmens-Apps unterstützt.

NetApp Snapshot-Kopien

Eine NetApp Snapshot-Kopie ist ein schreibgeschütztes Point-in-Time-Image eines Volumes. Das Image verbraucht nur minimalen Speicherplatz und verursacht nur einen vernachlässigbaren Leistungsaufwand, da es nur Änderungen an Dateien aufzeichnet, die seit der letzten Snapshot-Kopie erstellt wurden, wie in der folgenden Abbildung dargestellt.

Ihre Effizienz verdanken Snapshot-Kopien der zentralen ONTAP Speichervirtualisierungstechnologie, dem Write Anywhere File Layout (WAFL). Wie eine Datenbank verwendet WAFL Metadaten, um auf tatsächliche Datenblöcke auf der Festplatte zu verweisen. Aber im Gegensatz zu einer Datenbank überschreibt WAFL keine vorhandenen Blöcke. Es schreibt aktualisierte Daten in einen neuen Block und ändert die Metadaten. Snapshot-Kopien sind deshalb so effizient, weil ONTAP beim Erstellen einer Snapshot-Kopie auf Metadaten verweist, anstatt Datenblöcke zu kopieren. Dadurch entfallen die Suchzeit, die bei anderen Systemen zum Auffinden der zu kopierenden Blöcke erforderlich ist, sowie die Kosten für die Erstellung der Kopie selbst.

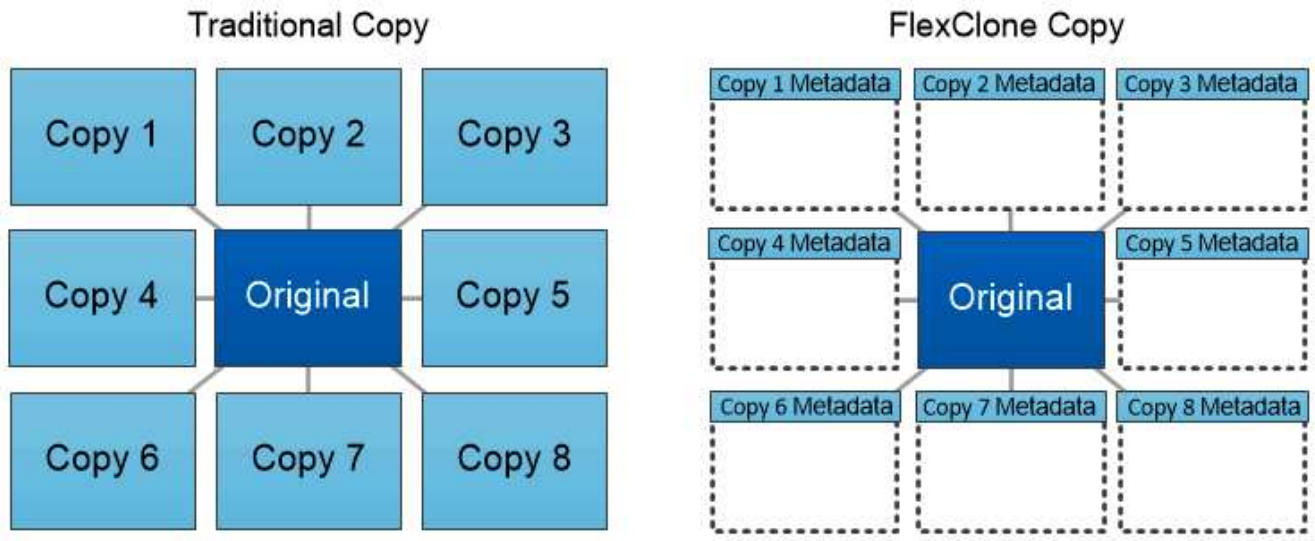
Sie können eine Snapshot-Kopie verwenden, um einzelne Dateien oder LUNs wiederherzustellen oder den gesamten Inhalt eines Volumes wiederherzustellen. ONTAP vergleicht Zeigerinformationen in der Snapshot-Kopie mit Daten auf der Festplatte, um das fehlende oder beschädigte Objekt ohne Ausfallzeiten oder erhebliche Leistungseinbußen 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 Point-in-Time-Kopien eines Volumes zu erstellen. Kopien teilen sich Datenblöcke mit ihren Eltern und verbrauchen keinen Speicherplatz außer dem, der für Metadaten benötigt wird, bis Änderungen in die Kopie geschrieben werden, wie in der folgenden Abbildung dargestellt. Während die Erstellung herkömmlicher Kopien Minuten oder sogar Stunden dauern kann, können Sie mit der FlexClone -Software selbst die größten Datensätze nahezu augenblicklich kopieren. Dadurch eignet es sich ideal für Situationen, in denen Sie mehrere Kopien identischer Datensätze (z. B. einen Entwicklungsarbeitsbereich) oder temporäre Kopien eines Datensatzes (Testen einer Anwendung anhand eines Produktionsdatensatzes) benötigen.



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

NetApp SnapMirror Datenreplikationstechnologie

Die NetApp SnapMirror -Software ist eine kostengünstige, benutzerfreundliche, einheitliche Replikationslösung für die gesamte Datenstruktur. Es repliziert Daten mit hoher Geschwindigkeit über LAN oder WAN. Es bietet Ihnen hohe Datenverfügbarkeit und schnelle Datenreplikation für Anwendungen aller Art, einschließlich geschäftskritischer Anwendungen in virtuellen und herkömmlichen Umgebungen. Wenn Sie Daten auf ein oder mehrere NetApp -Speichersysteme replizieren und die sekundären Daten kontinuierlich aktualisieren, bleiben Ihre Daten aktuell und stehen Ihnen jederzeit zur Verfügung. Es sind keine externen Replikationsserver erforderlich. In der folgenden Abbildung sehen Sie ein Beispiel für eine Architektur, die die SnapMirror -Technologie nutzt.

Die SnapMirror -Software nutzt die Speichereffizienz von NetApp ONTAP , indem sie nur geänderte Blöcke über das Netzwerk sendet. Die SnapMirror -Software verwendet außerdem eine integrierte Netzwerkkomprimierung, um die Datenübertragung zu beschleunigen und die Netzwerkbandbreitenauslastung um bis zu 70 % zu reduzieren. Mit der SnapMirror -Technologie können Sie einen Thin-Replication-Datenstrom nutzen, um ein einzelnes Repository zu erstellen, das sowohl den aktiven Spiegel als auch frühere Point-in-Time-Kopien verwaltet und so den Netzwerkverkehr um bis zu 50 % reduziert.

NetApp BlueXP Kopieren und Synchronisieren

"BlueXP Kopieren und Synchronisieren"ist ein NetApp -Dienst für die schnelle und sichere Datensynchronisierung. Unabhängig davon, ob Sie Dateien zwischen lokalen NFS- oder SMB-Dateifreigaben, NetApp StorageGRID, NetApp ONTAP S3, Google Cloud NetApp Volumes, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage oder IBM Cloud Object Storage übertragen müssen, verschiebt BlueXP Copy and Sync die Dateien schnell und sicher dorthin, wo Sie sie benötigen.

Nachdem Ihre Daten übertragen wurden, stehen sie sowohl auf der Quelle als auch auf dem Ziel vollständig zur Verwendung zur Verfügung. BlueXP Copy and Sync kann Daten bei Bedarf synchronisieren, wenn ein Update ausgelöst wird, oder Daten kontinuierlich basierend auf einem vordefinierten Zeitplan synchronisieren. Unabhängig davon verschiebt BlueXP Copy and Sync nur die Deltas, sodass der Zeit- und Kostenaufwand für die Datenreplikation minimiert wird.

BlueXP Copy and Sync ist ein Software-as-a-Service-Tool (SaaS), das extrem einfach einzurichten und zu

verwenden ist. Datenübertragungen, die durch BlueXP Copy and Sync ausgelöst werden, werden von Datenbrokern durchgeführt. BlueXP Copy and Sync-Datenbroker können in AWS, Azure, Google Cloud Platform oder vor Ort bereitgestellt werden.

NetApp XCP

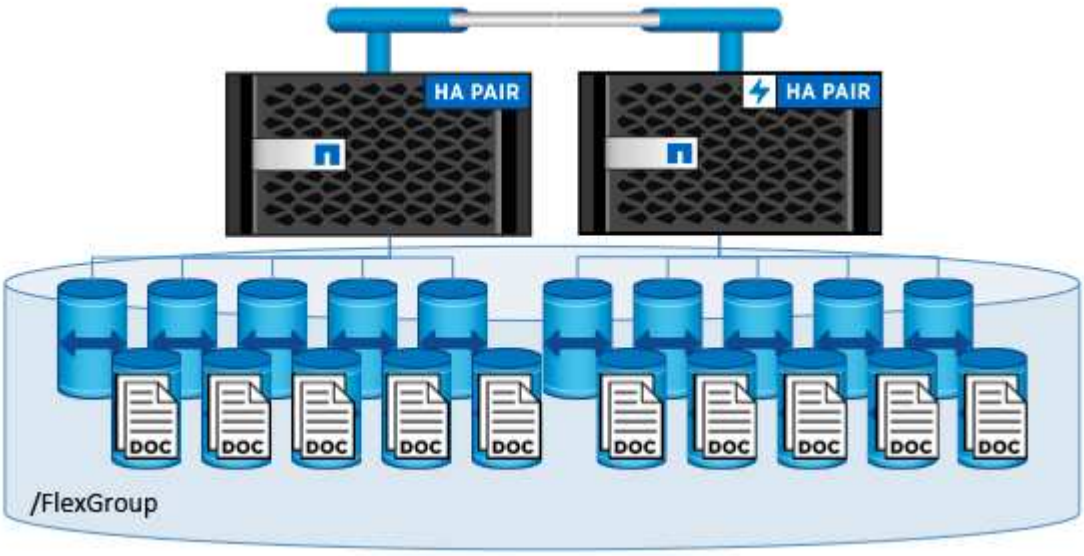
"NetApp XCP"ist eine clientbasierte Software für Datenmigrationen von beliebigen zu NetApp und von NetApp zu NetApp sowie Einblicke in Dateisysteme. XCP ist auf Skalierbarkeit und maximale Leistung ausgelegt, indem alle verfügbaren Systemressourcen genutzt werden, um große Datensätze und leistungsstarke Migrationen zu verarbeiten. XCP hilft Ihnen, einen vollständigen Einblick in das Dateisystem zu erhalten und bietet die Möglichkeit, Berichte zu erstellen.

NetApp ONTAP FlexGroup Volumes

Ein Trainingsdatensatz kann eine Sammlung von potenziell Milliarden von Dateien sein. Dateien können Text, Audio, Video und andere Formen unstrukturierter Daten enthalten, die gespeichert und verarbeitet werden müssen, um parallel gelesen werden zu können. Das Speichersystem muss eine große Anzahl kleiner Dateien speichern und diese Dateien für sequenzielle und zufällige E/A-Vorgänge parallel lesen.

Ein FlexGroup -Volume ist ein einzelner Namespace, der aus mehreren Mitgliedsvolumes besteht, wie in der folgenden Abbildung dargestellt. Aus Sicht eines Speicheradministrators wird ein FlexGroup -Volume wie ein NetApp FlexVol volume verwaltet und verhält sich wie dieses. Dateien in einem FlexGroup -Volume werden einzelnen Mitgliedsvolumes zugewiesen und nicht über Volumes oder Knoten verteilt. Sie ermöglichen die folgenden Funktionen:

- FlexGroup -Volumes bieten mehrere Petabyte an Kapazität und vorhersehbar niedrige Latenz für Workloads mit vielen Metadaten.
- Sie unterstützen bis zu 400 Milliarden Dateien im selben Namespace.
- Sie unterstützen parallelisierte Vorgänge in NAS-Workloads über CPUs, Knoten, Aggregate und einzelne FlexVol Volumes hinweg.



Architektur

Diese Lösung ist nicht von bestimmter Hardware abhängig. Die Lösung ist mit allen physischen Speichergeräten, softwaredefinierten Instanzen und Cloud-Diensten von NetApp kompatibel, die von NetApp Trident unterstützt werden. Beispiele hierfür sind ein NetApp AFF Speichersystem, Amazon FSx ONTAP, Azure NetApp Files, Google Cloud NetApp Volumes oder eine NetApp Cloud Volumes ONTAP -Instanz. Darüber hinaus kann die Lösung auf jedem Kubernetes-Cluster implementiert werden, solange die verwendete Kubernetes-Version von NetApp Trident und den anderen implementierten Lösungskomponenten unterstützt wird. Eine Liste der von Trident unterstützten Kubernetes-Versionen finden Sie im "Trident -Dokumentation" . In den folgenden Tabellen finden Sie Einzelheiten zu den Umgebungen, die zur Validierung der verschiedenen Komponenten dieser Lösung verwendet wurden.

Apache Airflow-Validierungsumgebung

Softwarekomponente	Version
Apache Airflow	2.0.1, bereitgestellt über"Apache Airflow Helm-Diagramm" 8.0.8
Kubernetes	1,18
NetApp Trident	21,01

JupyterHub-Validierungsumgebung

Softwarekomponente	Version
JupyterHub	4.1.5, bereitgestellt über"JupyterHub Helm-Diagramm" 3.3.7
Kubernetes	1,29
NetApp Trident	24,02

MLflow-Validierungsumgebung

Softwarekomponente	Version
MLflow	2.14.1, bereitgestellt über"MLflow Helm-Diagramm" 1.4.12
Kubernetes	1,29
NetApp Trident	24,02

Kubeflow-Validierungsumgebung

Softwarekomponente	Version
Kubeflow	1.7, bereitgestellt über"deployKF" 0.1.1

Softwarekomponente	Version
Kubernetes	1,26
NetApp Trident	23,07

Support

NetApp bietet keinen Enterprise-Support für Apache Airflow, JupyterHub, MLflow, Kubeflow oder Kubernetes. Wenn Sie an einer vollständig unterstützten MLOps-Plattform interessiert sind,"NetApp kontaktieren" über vollständig unterstützte MLOps-Lösungen, die NetApp gemeinsam mit Partnern anbietet.

NetApp Trident -Konfiguration

Beispiele für Trident -Backends für NetApp AIPod -Bereitstellungen

Bevor Sie Trident zur dynamischen Bereitstellung von Speicherressourcen in Ihrem Kubernetes-Cluster verwenden können, müssen Sie ein oder mehrere Trident -Backends erstellen. Die folgenden Beispiele stellen verschiedene Arten von Backends dar, die Sie möglicherweise erstellen möchten, wenn Sie Komponenten dieser Lösung auf einem"NetApp AIPod" . Weitere Informationen zu Backends und beispielsweise Backends für andere Plattformen/Umgebungen finden Sie im"Trident -Dokumentation" .

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

   Die folgenden Beispielbefehle zeigen die Erstellung eines FlexGroup-fähigen Trident -Backends für eine AIPod -Speicher-Virtual-Machine (SVM). Dieses Backend verwendet die `ontap-nas-flexgroup` Speichertreiber. ONTAP unterstützt zwei Hauptdatenvolumentypen: FlexVol und FlexGroup. FlexVol -Volumes sind größenbeschränkt (zum Zeitpunkt der Erstellung dieses Dokuments hängt die maximale Größe von der jeweiligen Bereitstellung ab). FlexGroup -Volumes hingegen können linear auf bis zu 20 PB und 400 Milliarden Dateien skaliert werden und bieten einen einzigen Namespace, der die Datenverwaltung erheblich vereinfacht. Daher sind FlexGroup Volumes optimal für KI- und ML-Workloads, die auf großen Datenmengen basieren.

   Wenn Sie mit einer kleinen Datenmenge arbeiten und FlexVol -Volumes anstelle von FlexGroup Volumes verwenden möchten, können Sie Trident -Backends erstellen, die die `ontap-nas` Speichertreiber anstelle des `ontap-nas-flexgroup` Speichertreiber.

```
$ 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 Backends. Möglicherweise möchten Sie FlexVol -Volumes zum Hosten persistenter Anwendungen, zum Speichern von Ergebnissen, Ausgaben, Debug-Informationen usw. verwenden. Wenn Sie FlexVol Volumes verwenden möchten, müssen Sie ein oder mehrere FlexVol-fähige Trident Backends erstellen. Die folgenden Beispielbefehle zeigen die Erstellung eines einzelnen FlexVol-fähigen Trident Backends.

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

Beispiele für Kubernetes-Speicherklassen für NetApp AIPod Bereitstellungen

Bevor Sie Trident verwenden können, um Speicherressourcen in Ihrem Kubernetes-Cluster dynamisch bereitzustellen, müssen Sie eine oder mehrere Kubernetes-StorageClasses erstellen. Die folgenden Beispiele stellen verschiedene Arten von StorageClasses dar, die Sie möglicherweise erstellen möchten, wenn Sie Komponenten dieser Lösung auf einem"NetApp AIPod" . Weitere Informationen zu StorageClasses und

beispielsweise StorageClasses für andere Plattformen/Umgebungen finden Sie im"Trident -Dokumentation" .

1. NetApp empfiehlt die Erstellung einer StorageClass für das FlexGroup-fähige Trident Backend, das Sie im Abschnitt"Beispiele für Trident -Backends für NetApp AIPod -Bereitstellungen" , Schritt 1. Die folgenden Beispielbefehle zeigen die Erstellung mehrerer StorageClasses, die dem Beispiel-Backend entsprechen, das im Abschnitt erstellt wurde."Beispiele für Trident -Backends für NetApp AIPod -Bereitstellungen" , Schritt 1 - einer, der nutzt"NFS über RDMA" und eine, die das nicht tut.

   Damit ein persistentes Volume nicht gelöscht wird, wenn der entsprechende PersistentVolumeClaim (PVC) gelöscht wird, verwendet das folgende Beispiel ein `reclaimPolicy` Wert von `Retain` . Weitere Informationen zum `reclaimPolicy` Feld, siehe die offizielle "Kubernetes-Dokumentation" .

   Hinweis: Die folgenden Beispiel-StorageClasses verwenden eine maximale Übertragungsgröße von 262144. Um diese maximale Übertragungsgröße zu verwenden, müssen Sie die maximale Übertragungsgröße auf Ihrem ONTAP System entsprechend konfigurieren. Weitere Informationen finden Sie im"ONTAP-Dokumentation" für Details.

   Hinweis: Um NFS über RDMA zu verwenden, müssen Sie NFS über RDMA auf Ihrem ONTAP System konfigurieren. Weitere Informationen finden Sie im"ONTAP-Dokumentation" für Details.

   Hinweis: Im folgenden Beispiel wird im Feld „storagePool“ in der StorageClass-Definitionsdatei ein 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 Backend entspricht, das Sie im Abschnitt"Beispiele für Trident -Backends für AIPod -Bereitstellungen" , Schritt 2. Die folgenden Beispielbefehle zeigen die Erstellung einer einzelnen StorageClass für FlexVol -Volumes.

   Hinweis: Im folgenden Beispiel ist im Feld „storagePool“ in der StorageClass-Definitionsdatei kein bestimmtes Backend angegeben. Wenn Sie Kubernetes verwenden, um Volumes mit dieser StorageClass zu verwalten, versucht Trident , jedes verfügbare Backend zu verwenden, das die `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
```

Apache Airflow

Apache Airflow-Bereitstellung

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

> Es ist möglich, Airflow auf anderen Plattformen als Kubernetes bereitzustellen. Die Bereitstellung von Airflow auf anderen Plattformen als Kubernetes liegt außerhalb des Umfangs dieser Lösung.

Voraussetzungen

Bevor Sie die in diesem Abschnitt beschriebene Bereitstellungsübung durchführen, gehen wir davon aus, dass Sie die folgenden Aufgaben bereits 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 Einzelheiten zu Trident finden Sie im"Trident -Dokumentation" .

Helm installieren

Airflow wird mit Helm bereitgestellt, einem beliebten Paketmanager für Kubernetes. Bevor Sie Airflow bereitstellen, müssen Sie Helm auf dem Bereitstellungs-Jump-Host installieren. Um Helm auf dem Bereitstellungs-Jump-Host zu installieren, folgen Sie den "Installationsanweisungen" in der offiziellen Helm-Dokumentation.

Standardmäßige Kubernetes-Speicherklasse festlegen

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

Verwenden Sie Helm, um Airflow bereitzustellen

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

1. Stellen Sie Airflow mit Helm bereit, indem Sie den Anweisungen folgen "Bereitstellungsanweisungen" für das offizielle Airflow-Diagramm im Artifact Hub. Die folgenden Beispielbefehle zeigen die Bereitstellung von Airflow mit Helm. Ändern, Hinzufügen und/oder Entfernen von Werten in der `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
```

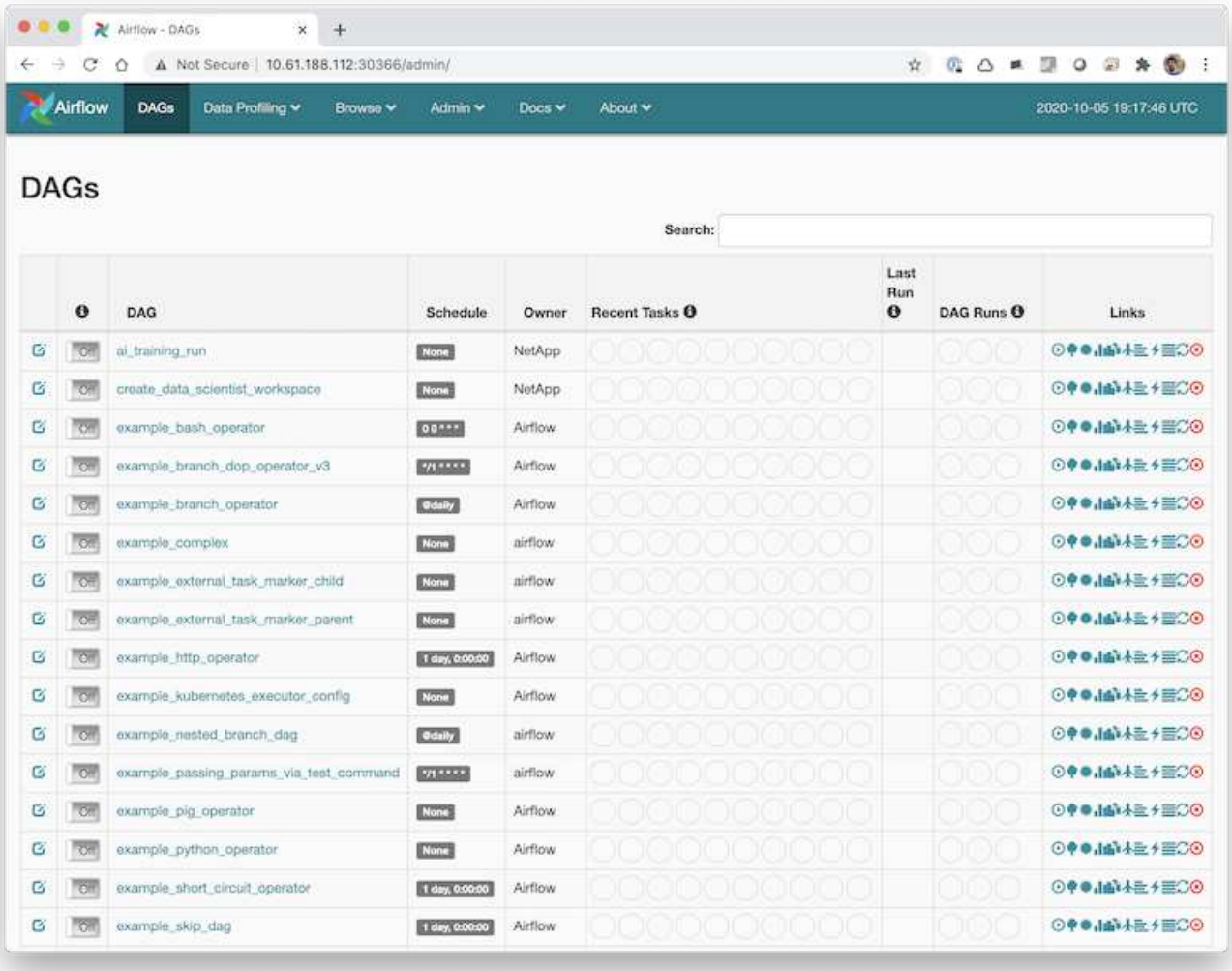
2. Bestätigen Sie, dass alle Airflow-Pods betriebsbereit sind. Es kann einige Minuten dauern, bis alle Pods gestartet sind.

```
$ 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-Webdienstes ab, indem Sie den Anweisungen folgen, die auf der Konsole angezeigt wurden, als Sie Airflow in Schritt 1 mit Helm bereitgestellt haben.

```
$ 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. Bestätigen Sie, dass Sie auf den Airflow-Webdienst zugreifen können.



Verwenden Sie das NetApp DataOps Toolkit mit Airflow

Der "NetApp DataOps Toolkit für Kubernetes" kann in Verbindung mit Airflow verwendet werden. Durch die Verwendung des NetApp DataOps Toolkit mit Airflow können Sie NetApp Datenverwaltungsvorgänge, wie das Erstellen von Snapshots und Klonen, in automatisierte Workflows integrieren, die von Airflow orchestriert werden.

Weitere Informationen finden Sie im "Beispiele für Luftströme" Weitere Informationen zur Verwendung des Toolkits mit Airflow finden Sie im Abschnitt „NetApp DataOps Toolkit GitHub-Repository“.

JupyterHub

JupyterHub-Bereitstellung

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

Es ist möglich, JupyterHub auf anderen Plattformen als Kubernetes bereitzustellen. Die Bereitstellung von JupyterHub auf anderen Plattformen als Kubernetes liegt außerhalb des Umfangs dieser Lösung.

Voraussetzungen

Bevor Sie die in diesem Abschnitt beschriebene Bereitstellungsübung durchführen, gehen wir davon aus, dass Sie die folgenden Aufgaben bereits 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 Einzelheiten zu Trident finden Sie im"Trident -Dokumentation" .

Helm installieren

JupyterHub wird mit Helm bereitgestellt, einem beliebten Paketmanager für Kubernetes. Bevor Sie JupyterHub bereitstellen, müssen Sie Helm auf Ihrem Kubernetes-Steuerknoten installieren. Um Helm zu installieren, folgen Sie den "Installationsanweisungen" in der offiziellen Helm-Dokumentation.

Standardmäßige Kubernetes-Speicherklasse festlegen

Bevor Sie JupyterHub bereitstellen, müssen Sie eine Standard-StorageClass in Ihrem Kubernetes-Cluster festlegen. Um eine Standard-StorageClass innerhalb Ihres Clusters festzulegen, folgen Sie den Anweisungen im"Kubeflow-Bereitstellung" Abschnitt. Wenn Sie in Ihrem Cluster bereits eine Standard-StorageClass festgelegt haben, können Sie diesen Schritt überspringen.

JupyterHub bereitstellen

Nachdem Sie die oben genannten Schritte abgeschlossen haben, können Sie JupyterHub nun bereitstellen. Für die Bereitstellung von JupyterHub sind die folgenden Schritte erforderlich:

JupyterHub-Bereitstellung konfigurieren

Vor der Bereitstellung empfiehlt es sich, die JupyterHub-Bereitstellung für Ihre jeweilige Umgebung zu optimieren. Sie können eine **config.yaml**-Datei erstellen und sie während der Bereitstellung mithilfe des Helm-Diagramms verwenden.

Eine Beispieldatei **config.yaml** finden Sie unter https://github.com/jupyterhub/zero-to-jupyterhub-k8s/blob/HEAD/jupyterhub/values.yaml

In dieser config.yaml-Datei können Sie den Parameter **(singleuser.storage.dynamic.storageClass)** für die NetApp Trident StorageClass festlegen. Dies ist die Speicherklasse, die zum Bereitstellen der Volumes für einzelne Benutzerarbeitsbereiche verwendet wird.

Hinzufügen freigegebener Volumes

Wenn Sie ein gemeinsames Volume für alle JupyterHub-Benutzer verwenden möchten, können Sie Ihre **config.yaml** entsprechend anpassen. Wenn Sie beispielsweise über einen freigegebenen PersistentVolumeClaim namens „jupyterhub-shared-volume“ verfügen, können Sie ihn in allen Benutzer-Pods wie folgt als /home/shared mounten:

```
singleuser:
  storage:
    extraVolumes:
      - name: jupyterhub-shared
        persistentVolumeClaim:
          claimName: jupyterhub-shared-volume
    extraVolumeMounts:
      - name: jupyterhub-shared
        mountPath: /home/shared
```

Dies ist ein optionaler Schritt. Sie können diese Parameter an Ihre Bedürfnisse anpassen.

JupyterHub mit Helm Chart bereitstellen

Machen Sie Helm auf das JupyterHub Helm-Diagramm-Repository aufmerksam.

```
helm repo add jupyterhub https://hub.jupyter.org/helm-chart/
helm repo update
```

Dies sollte eine Ausgabe wie die folgende anzeigen:

```
Hang tight while we grab the latest from your chart repositories...
...Skip local chart repository
...Successfully got an update from the "stable" chart repository
...Successfully got an update from the "jupyterhub" chart repository
Update Complete. □ Happy Helming!□
```

Installieren Sie nun das von Ihrer config.yaml konfigurierte Diagramm, indem Sie diesen Befehl aus dem Verzeichnis ausführen, das Ihre config.yaml enthält:

```
helm upgrade --cleanup-on-fail \
  --install my-jupyterhub jupyterhub/jupyterhub \
  --namespace my-namespace \
  --create-namespace \
  --values config.yaml
```

In diesem Beispiel:

<helm-release-name> ist auf my-jupyterhub eingestellt, was der Name Ihrer JupyterHub-Version sein wird. <k8s-namespace> ist auf my-namespace eingestellt, also den Namespace, in dem Sie JupyterHub installieren möchten. Das Flag --create-namespace wird verwendet, um den Namespace zu erstellen, falls er noch nicht vorhanden ist. Das Flag --values gibt die Datei config.yaml an, die Ihre gewünschten Konfigurationsoptionen enthält.

Bereitstellung prüfen

Während Schritt 2 ausgeführt wird, können Sie sehen, wie die Pods mit dem folgenden Befehl erstellt werden:

```
kubectl get pod --namespace <k8s-namespace>
```

Warten Sie, bis der Hub und der Proxy-Pod in den Status „Laufen“ wechseln.

```
NAME                      READY     STATUS    RESTARTS   AGE
hub-5d4ffd57cf-k68z8      1/1       Running   0          37s
proxy-7cb9bc4cc-9bdlp     1/1       Running   0          37s
```

Zugriff auf JupyterHub

Suchen Sie die IP, die wir für den Zugriff auf den JupyterHub verwenden können. Führen Sie den folgenden Befehl aus, bis die EXTERNE IP des öffentlichen Proxy-Dienstes wie in der Beispielausgabe verfügbar ist.

Wir haben den NodePort-Dienst in unserer Datei config.yaml verwendet. Sie können ihn basierend auf Ihrem Setup (z. B. LoadBalancer) an Ihre Umgebung anpassen.

```
kubectl --namespace <k8s-namespace> get service proxy-public
```

```
NAME           TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)
AGE
proxy-public   NodePort   10.51.248.230   104.196.41.97   80:30000/TCP
1m
```

Um JupyterHub zu verwenden, geben Sie die externe IP für den Proxy-Public-Dienst in einen Browser ein.

Verwenden Sie das NetApp DataOps Toolkit mit JupyterHub

Der "NetApp DataOps Toolkit für Kubernetes" kann in Verbindung mit JupyterHub verwendet werden. Durch die Verwendung des NetApp DataOps Toolkit mit JupyterHub können Endbenutzer Volume-Snapshots für die Sicherung des Arbeitsbereichs und/oder die Rückverfolgbarkeit von Datensätzen zu Modellen direkt aus einem Jupyter-Notebook erstellen.

Ersteinrichtung

Bevor Sie das DataOps Toolkit mit JupyterHub verwenden können, müssen Sie dem Kubernetes-Dienstkonto, das JupyterHub einzelnen Jupyter Notebook Server-Pods zuweist, die entsprechenden Berechtigungen erteilen. JupyterHub verwendet das Dienstkonto, das durch den `singleuser.serviceAccountName` Variable in Ihrer JupyterHub Helm-Diagramm-Konfigurationsdatei.

Clusterrolle für DataOps Toolkit erstellen

Erstellen Sie zunächst eine Clusterrolle mit dem Namen „netapp-dataops“, die über die erforderlichen Kubernetes-API-Berechtigungen zum Erstellen von Volume-Snapshots verfügt.

```
$ vi clusterrole-netapp-dataops-snapshots.yaml
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-dataops-snapshots
rules:
- apiGroups: [""]
  resources: ["persistentvolumeclaims", "persistentvolumeclaims/status",
"services"]
  verbs: ["get", "list"]
- apiGroups: ["snapshot.storage.k8s.io"]
  resources: ["volumesnapshots", "volumesnapshots/status",
"volumesnapshotcontents", "volumesnapshotcontents/status"]
  verbs: ["get", "list", "create"]

$ kubectl create -f clusterrole-netapp-dataops-snapshots.yaml
clusterrole.rbac.authorization.k8s.io/netapp-dataops-snapshots created
```

Zuweisen der Clusterrolle zum Notebook-Server-Dienstkonto

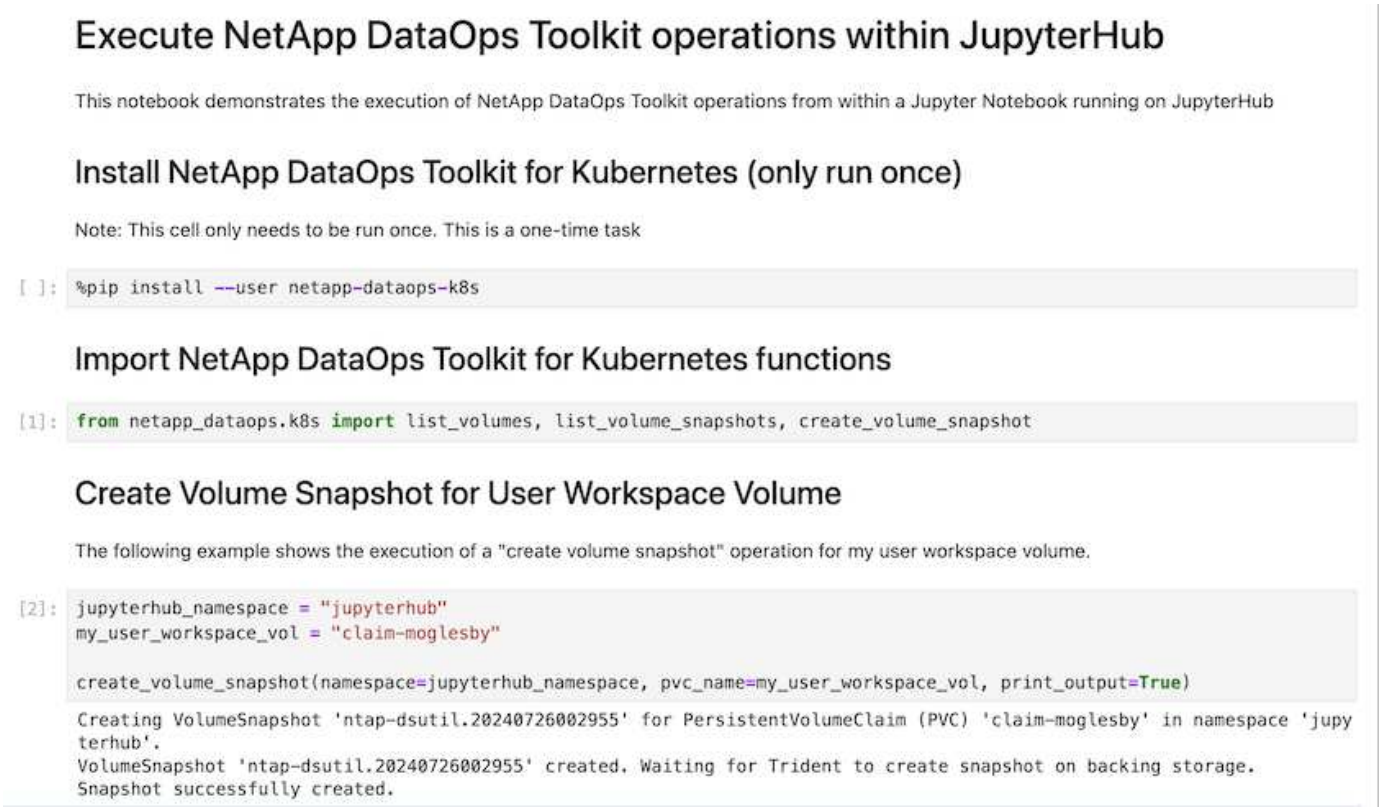
Erstellen Sie eine Rollenbindung, die die Clusterrolle „netapp-dataops-snapshots“ dem entsprechenden Dienstkonto im entsprechenden Namespace zuweist. Wenn Sie beispielsweise JupyterHub im Namespace „jupyterhub“ installiert haben und das Dienstkonto „default“ über die `singleuser.serviceAccountName` Mit dieser Variable würden Sie die Clusterrolle „netapp-dataops-snapshots“ dem Dienstkonto „default“ im Namespace „jupyterhub“ zuweisen, wie im folgenden Beispiel gezeigt.

```
$ vi rolebinding-jupyterhub-netapp-dataops-snapshots.yaml
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: jupyterhub-netapp-dataops-snapshots
  namespace: jupyterhub # Replace with you JupyterHub namespace
subjects:
- kind: ServiceAccount
  name: default # Replace with your JupyterHub
singleuser.serviceAccountName
  namespace: jupyterhub # Replace with you JupyterHub namespace
roleRef:
  kind: ClusterRole
  name: netapp-dataops-snapshots
  apiGroup: rbac.authorization.k8s.io

$ kubectl create -f ./rolebinding-jupyterhub-netapp-dataops-snapshots.yaml
rolebinding.rbac.authorization.k8s.io/jupyterhub-netapp-dataops-snapshots
created
```

Erstellen Sie Volume-Snapshots innerhalb des Jupyter-Notebooks

Jetzt können JupyterHub-Benutzer das NetApp DataOps Toolkit verwenden, um Volume-Snapshots direkt aus einem Jupyter-Notebook heraus zu erstellen, wie im folgenden Beispiel gezeigt.



Daten mit NetApp SnapMirror in JupyterHub einlesen

NetApp SnapMirror ist eine Replikationstechnologie, mit der Sie Daten zwischen NetApp Speichersystemen replizieren können. SnapMirror kann verwendet werden, um Daten aus Remote-Umgebungen in JupyterHub zu übernehmen.

Beispiel-Workflow und Demo

Siehe"dieser Tech ONTAP -Blogbeitrag" für einen detaillierten Beispiel-Workflow und eine Demo zur Verwendung von NetApp SnapMirror zum Einspeisen von Daten in JupyterHub.

MLflow

MLflow-Bereitstellung

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

> Es ist möglich, MLflow auf anderen Plattformen als Kubernetes bereitzustellen. Die Bereitstellung von MLflow auf anderen Plattformen als Kubernetes liegt außerhalb des Umfangs dieser Lösung.

Voraussetzungen

Bevor Sie die in diesem Abschnitt beschriebene Bereitstellungsübung durchführen, gehen wir davon aus, dass Sie die folgenden Aufgaben bereits 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 Einzelheiten zu Trident finden Sie im"Trident -Dokumentation" .

Helm installieren

MLflow wird mit Helm bereitgestellt, einem beliebten Paketmanager für Kubernetes. Bevor Sie MLflow bereitstellen, müssen Sie Helm auf Ihrem Kubernetes-Steuerknoten installieren. Um Helm zu installieren, folgen Sie den "Installationsanweisungen" in der offiziellen Helm-Dokumentation.

Standardmäßige Kubernetes-Speicherklasse festlegen

Bevor Sie MLflow bereitstellen, müssen Sie eine Standard-StorageClass in Ihrem Kubernetes-Cluster festlegen. Um eine Standard-StorageClass innerhalb Ihres Clusters festzulegen, folgen Sie den Anweisungen im"Kubeflow-Bereitstellung" Abschnitt. Wenn Sie in Ihrem Cluster bereits eine Standard-StorageClass festgelegt haben, können Sie diesen Schritt überspringen.

MLflow bereitstellen

Sobald die Voraussetzungen erfüllt sind, können Sie mit der MLflow-Bereitstellung mithilfe des Helm-Diagramms beginnen.

Konfigurieren Sie die Bereitstellung des MLflow Helm-Diagramms.

Bevor wir MLflow mithilfe des Helm-Diagramms bereitstellen, können wir die Bereitstellung so konfigurieren, dass die NetApp Trident Storage Class verwendet wird, und mithilfe einer **config.yaml**-Datei andere Parameter an unsere Anforderungen anpassen. Ein Beispiel für eine **config.yaml**-Datei finden Sie unter: https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml

Sie können die Trident -Speicherklasse unter dem Parameter **global.defaultStorageClass** in der Datei config.yaml festlegen (z. B. Speicherklasse: „ontap-flexvol“).

Installieren des Helm-Diagramms

Das Helm-Diagramm kann mit der benutzerdefinierten Datei **config.yaml** für MLflow mithilfe des folgenden Befehls installiert werden:

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

Der Befehl stellt MLflow über die bereitgestellte Datei **config.yaml** in der benutzerdefinierten Konfiguration auf dem Kubernetes-Cluster bereit. MLflow wird im angegebenen Namespace bereitgestellt und für die Version wird über Kubernetes ein zufälliger Versionsname vergeben.

Bereitstellung prüfen

Nachdem die Bereitstellung des Helm-Diagramms abgeschlossen ist, können Sie mit folgendem Befehl überprüfen, ob auf den Dienst zugegriffen werden kann:

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

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

Sie sollten die folgenden Dienste sehen:

```
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)             AGE
mlflow-1719843029-minio             ClusterIP   10.233.22.4     <none>
80/TCP,9001/TCP     25d
mlflow-1719843029-postgresql        ClusterIP   10.233.5.141    <none>
5432/TCP            25d
mlflow-1719843029-postgresql-hl     ClusterIP   None            <none>
5432/TCP            25d
mlflow-1719843029-tracking          NodePort    10.233.2.158    <none>
30002:30002/TCP     25d
```

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

Zugriff auf MLflow

Sobald alle mit MLflow verbundenen Dienste betriebsbereit sind, können Sie über die angegebene NodePort- oder LoadBalancer-IP-Adresse darauf zugreifen (z. B. `http://10.61.181.109:30002`)

Rückverfolgbarkeit vom Datensatz zum Modell mit NetApp und MLflow

Der "NetApp DataOps Toolkit für Kubernetes" kann in Verbindung mit den Experimentverfolgungsfunktionen von MLflow verwendet werden, um die Rückverfolgbarkeit vom Datensatz zum Modell oder vom Arbeitsbereich zum Modell zu implementieren.

Um die Rückverfolgbarkeit vom Datensatz zum Modell oder vom Arbeitsbereich zum Modell zu implementieren, erstellen Sie im Rahmen Ihres Trainingslaufs einfach einen Snapshot Ihres Datensatzes oder Arbeitsbereichsvolumens mithilfe des DataOps Toolkit, wie im folgenden Beispielcodeausschnitt gezeigt. Dieser Code speichert den Datenvolumennamen und den Snapshot-Namen als Tags, die mit dem spezifischen Trainingslauf verknüpft sind, den Sie auf Ihrem MLflow-Experiment-Tracking-Server protokollieren.

```
...
from netapp_dataops.k8s import create_volume_snapshot

with mlflow.start_run() :
    ...

    namespace = "my_namespace" # Kubernetes namespace in which dataset
volume PVC resides
    dataset_volume_name = "project1" # Name of PVC corresponding to
dataset volume
    snapshot_name = "run1" # Name to assign to your new snapshot

    # Create snapshot
    create_volume_snapshot(
        namespace=namespace,
        pvc_name=dataset_volume_name,
        snapshot_name=snapshot_name,
        printOutput=True
    )

    # Log data volume name and snapshot name as "tags"
    # associated with this training run in mlflow.
    mlflow.set_tag("data_volume_name", dataset_volume_name)
    mlflow.set_tag("snapshot_name", snapshot_name)

    ...
```

Kubeflow

Kubeflow-Bereitstellung

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

Voraussetzungen

Bevor Sie die in diesem Abschnitt beschriebene Bereitstellungsübung durchführen, gehen wir davon aus, dass Sie die folgenden Aufgaben bereits ausgeführt haben:

1. Sie verfügen bereits über einen funktionierenden Kubernetes-Cluster und 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 im"offizielle Kubeflow-Dokumentation" .
2. Sie haben NetApp Trident bereits in Ihrem Kubernetes-Cluster installiert und konfiguriert. Weitere Einzelheiten zu Trident finden Sie im"Trident -Dokumentation" .

Standardmäßige Kubernetes-Speicherklasse festlegen

Bevor Sie Kubeflow bereitstellen, empfehlen wir, eine Standard-StorageClass in Ihrem Kubernetes-Cluster festzulegen. Der Kubeflow-Bereitstellungsprozess versucht möglicherweise, neue persistente Volumes mithilfe der Standard-StorageClass bereitzustellen. Wenn keine StorageClass als Standard-StorageClass festgelegt ist, kann die Bereitstellung fehlschlagen. Um eine Standard-StorageClass innerhalb Ihres Clusters festzulegen, führen Sie die folgende Aufgabe vom Bereitstellungs-Jump-Host aus. Wenn Sie in Ihrem Cluster bereits eine Standard-StorageClass festgelegt haben, können Sie diesen Schritt überspringen.

1. Legen Sie eine Ihrer vorhandenen StorageClasses als Standard-StorageClass fest. Die folgenden Beispielbefehle zeigen die Bezeichnung einer StorageClass namens `ontap-ai-flexvols-retain` als Standard-Speicherklasse.

> Der `ontap-nas-flexgroup` Der Typ Trident Backend hat eine ziemlich große Mindest-PVC-Größe. Standardmäßig versucht Kubeflow, PVCs bereitzustellen, die nur wenige GB groß sind. Daher sollten Sie keine StorageClass festlegen, die die `ontap-nas-flexgroup` Backend-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
```

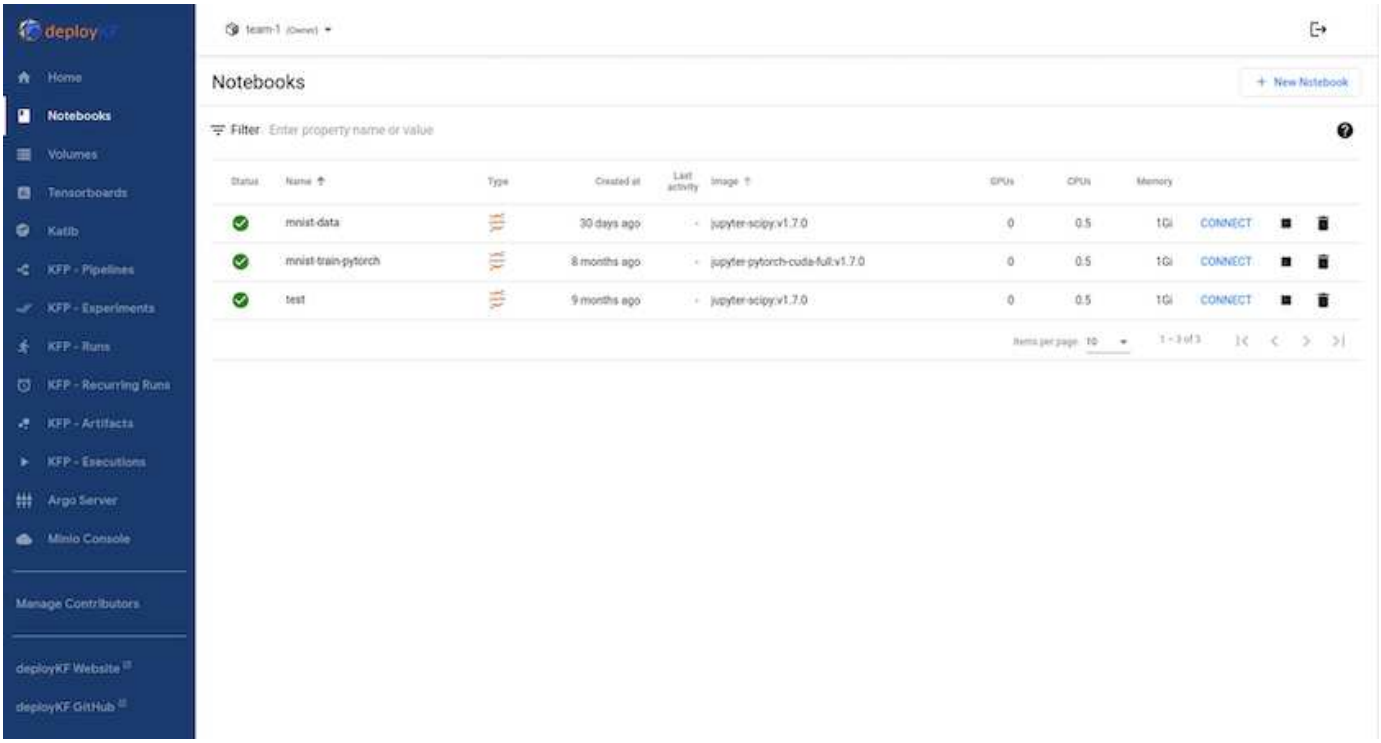
Kubeflow-Bereitstellungsoptionen

Es gibt viele verschiedene Optionen für die Bereitstellung von Kubeflow. Weitere Informationen finden Sie im"offizielle Kubeflow-Dokumentation" Klicken Sie auf „Weiter“, um eine Liste der Bereitstellungsoptionen anzuzeigen, und wählen Sie die Option aus, die Ihren Anforderungen am besten entspricht.

> Zu Validierungszwecken haben wir Kubeflow 1.7 bereitgestellt mit"deployKF" 0.1.1.

Bereitstellen eines Jupyter Notebook-Arbeitsbereichs für die Verwendung durch Datenwissenschaftler oder Entwickler

Kubeflow kann schnell neue Jupyter Notebook-Server bereitstellen, die als Arbeitsbereiche für Datenwissenschaftler fungieren. Weitere Informationen zu Jupyter Notebooks im Kubeflow-Kontext finden Sie im "offizielle Kubeflow-Dokumentation" .



Verwenden Sie das NetApp DataOps Toolkit mit Kubeflow

Der "NetApp Data Science Toolkit für Kubernetes" kann in Verbindung mit Kubeflow verwendet werden. Die Verwendung des NetApp Data Science Toolkit mit Kubeflow bietet die folgenden Vorteile:

- Datenwissenschaftler können erweiterte NetApp Datenverwaltungsvorgänge, wie das Erstellen von Snapshots und Klonen, direkt aus einem Jupyter-Notebook heraus durchführen.
- Erweiterte NetApp Datenverwaltungsvorgänge, wie das Erstellen von Snapshots und Klonen, können mithilfe des Kubeflow Pipelines-Frameworks in automatisierte Workflows integriert werden.

Weitere Informationen finden Sie im "Kubeflow-Beispiele" Weitere Informationen zur Verwendung des Toolkits mit Kubeflow finden Sie im Abschnitt „NetApp Data Science Toolkit GitHub-Repository“.

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

In diesem Abschnitt werden die Schritte zum Trainieren und Bereitstellen eines neuronalen Netzwerks für die Bilderkennung mit Kubeflow und dem NetApp DataOps Toolkit beschrieben. Dies soll als Beispiel für einen Trainingsjob dienen, der NetApp -Speicher einbezieht.

Voraussetzungen

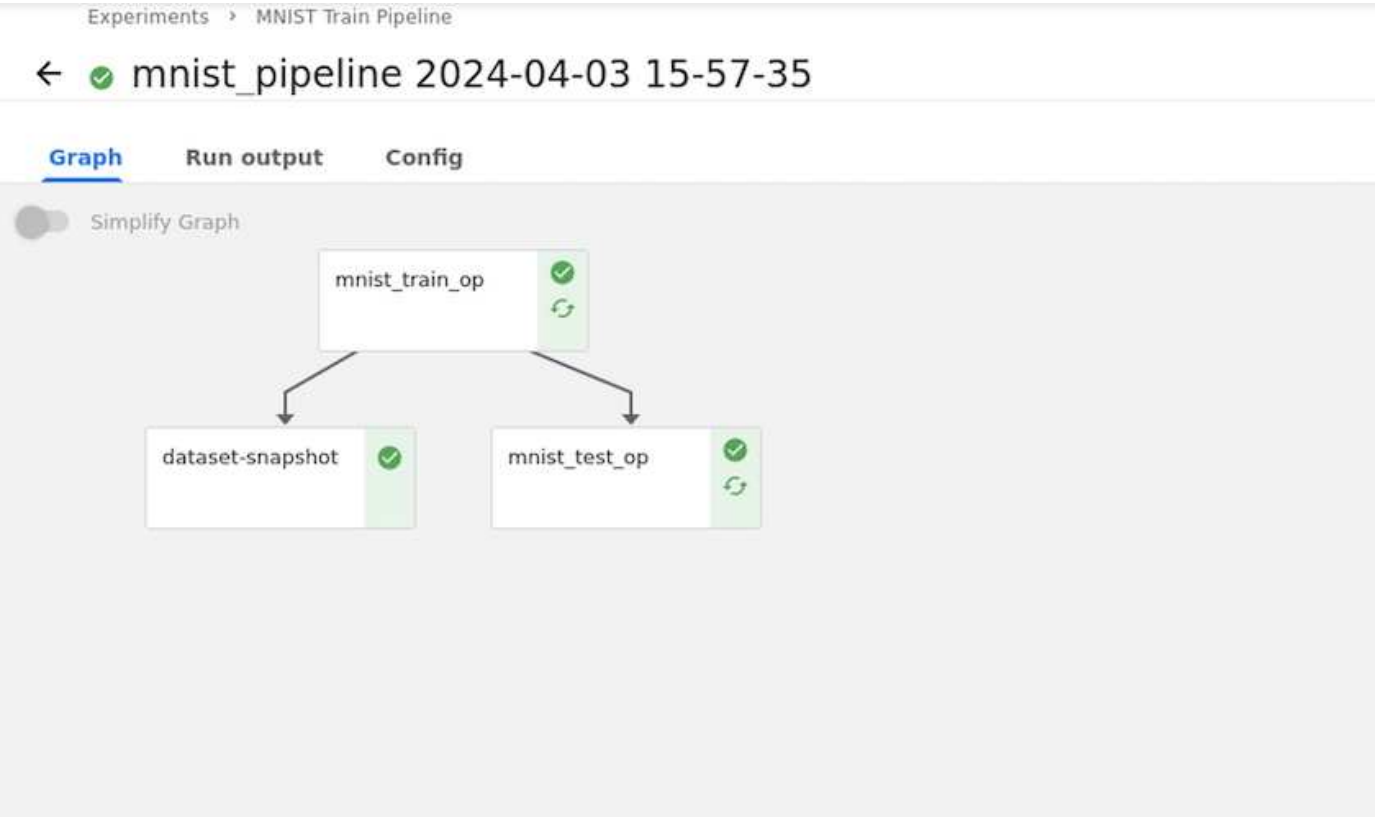
Erstellen Sie eine Docker-Datei mit den erforderlichen Konfigurationen für die Trainings- und Testschritte innerhalb der Kubeflow-Pipeline. Hier ist ein Beispiel für eine Docker-Datei -

```
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-Bereitstellung verfügen.

Trainieren Sie ein kleines neuronales Netzwerk anhand von MNIST-Daten mithilfe von PyTorch- und Kubeflow-Pipelines

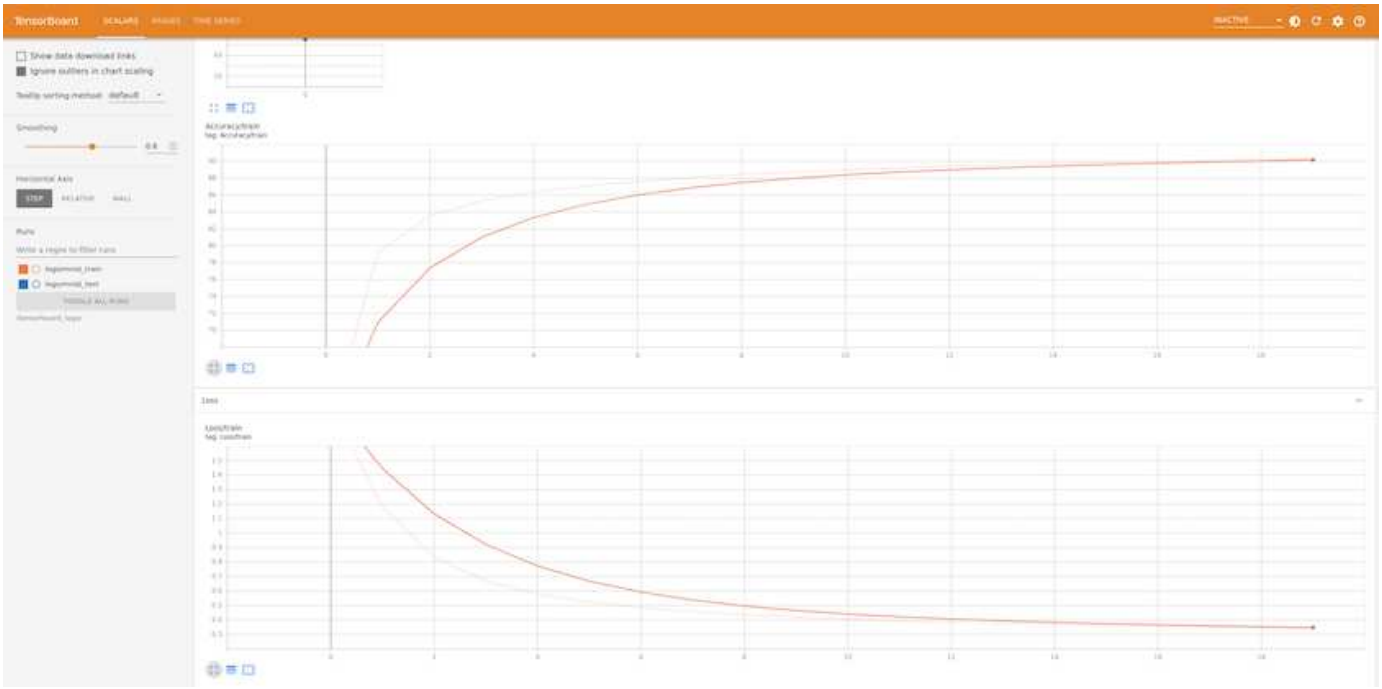
Wir verwenden das Beispiel eines kleinen neuronalen Netzwerks, das mit MNIST-Daten trainiert wurde. Der MNIST-Datensatz besteht aus handschriftlichen Bildern der Ziffern 0-9. Die Bilder haben eine Größe von 28 x 28 Pixel. Der Datensatz ist in 60.000 Zugbilder und 10.000 Validierungsbilder unterteilt. Das für dieses Experiment verwendete neuronale Netzwerk ist ein zweischichtiges Feedforward-Netzwerk. Das Training wird mithilfe von Kubeflow Pipelines durchgeführt. Weitere Informationen finden Sie in der Dokumentation "hier," für weitere Informationen. Unsere Kubeflow-Pipeline enthält das Docker-Image aus dem Abschnitt „Voraussetzungen“.



Visualisieren Sie Ergebnisse mit Tensorboard

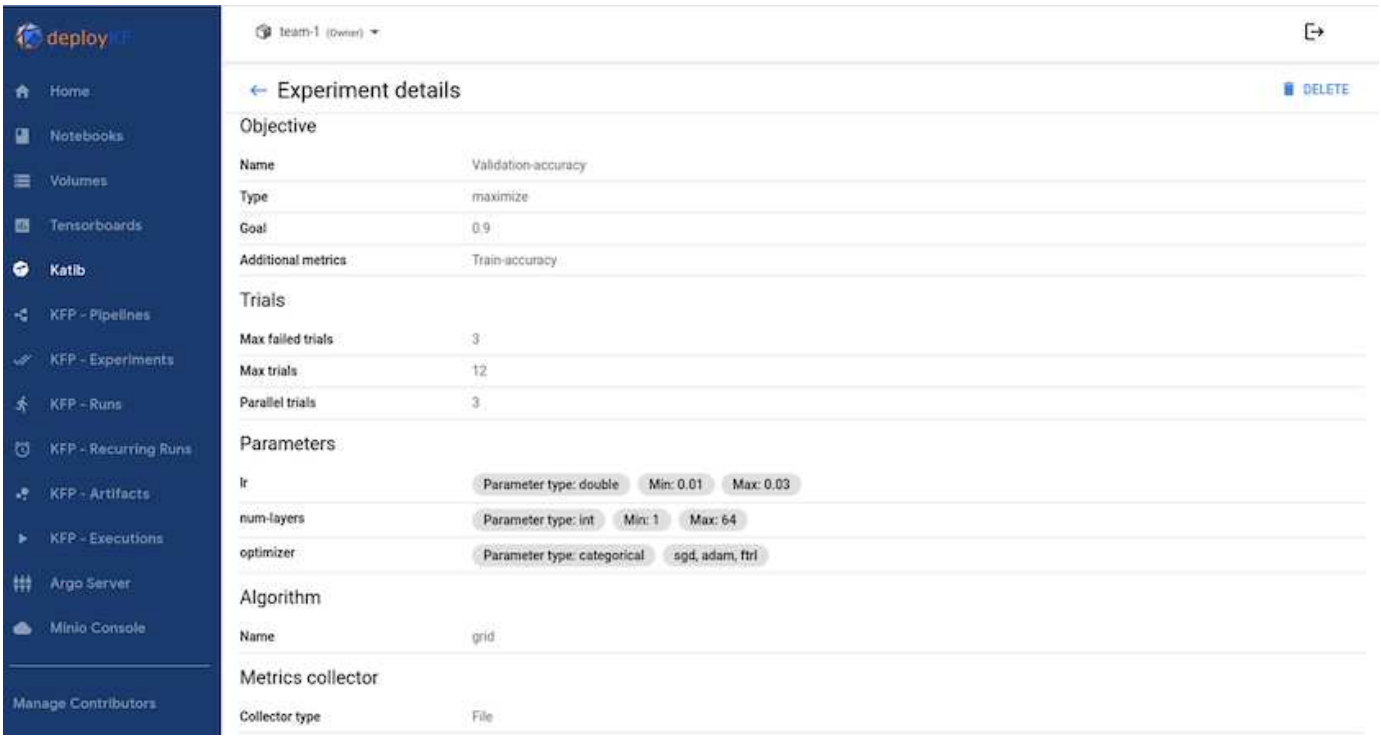
Sobald das Modell trainiert ist, können wir die Ergebnisse mit Tensorboard visualisieren. "Tensorboard" ist als Funktion im Kubeflow-Dashboard verfügbar. Sie können ein benutzerdefiniertes Tensorboard für Ihren Job erstellen. Ein Beispiel unten zeigt die Darstellung der Trainingsgenauigkeit im Vergleich zur Anzahl der

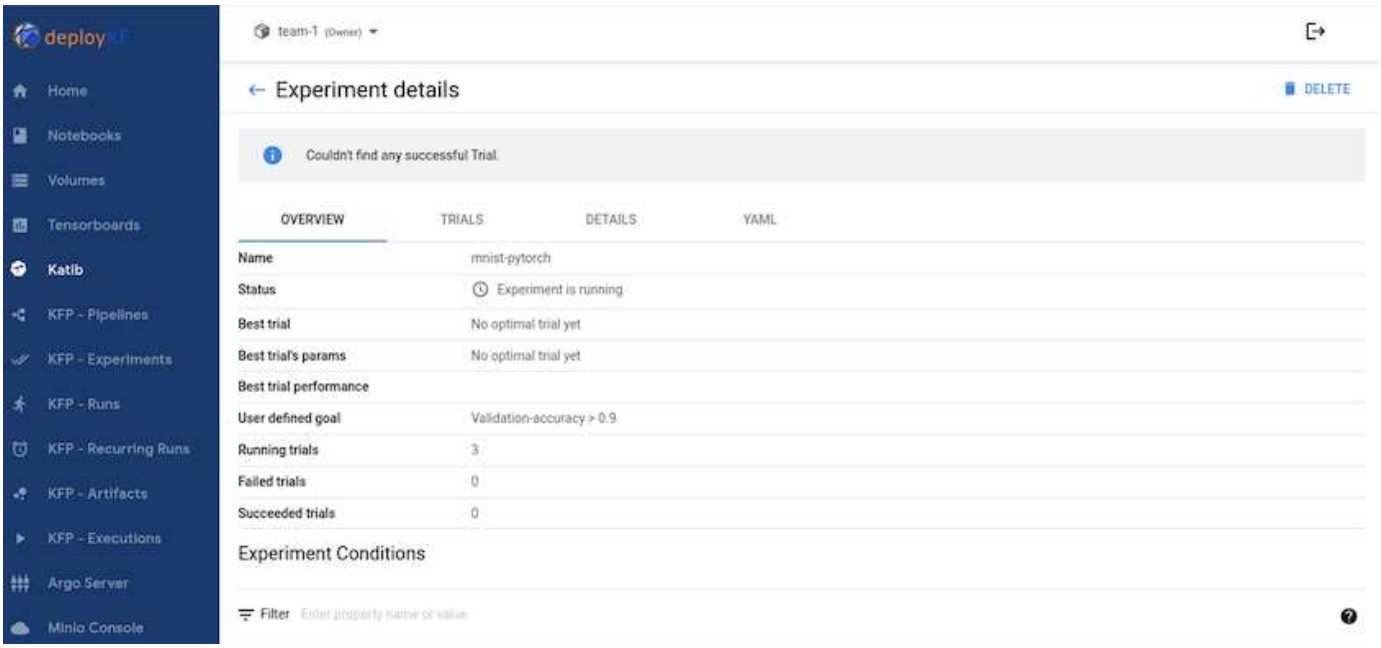
Epochen und des Trainingsverlusts im Vergleich zur Anzahl der Epochen.



Experimentieren Sie mit Hyperparametern mit Katib

"Katib"ist ein Tool innerhalb von Kubeflow, mit dem mit den Hyperparametern des Modells experimentiert werden kann. Um ein Experiment zu erstellen, definieren Sie zuerst eine gewünschte Metrik/ein gewünschtes Ziel. Dies ist normalerweise die Testgenauigkeit. Sobald die Metrik definiert ist, wählen Sie Hyperparameter aus, mit denen Sie herumspielen möchten (Optimierer/Lernrate/Anzahl der Schichten). Katib führt einen Hyperparameter-Sweep mit den benutzerdefinierten Werten durch, um die beste Parameterkombination zu finden, die die gewünschte Metrik erfüllt. Sie können diese Parameter in jedem Abschnitt der Benutzeroberfläche definieren. Alternativ können Sie eine **YAML**-Datei mit den erforderlichen Spezifikationen definieren. Unten sehen Sie eine Illustration eines Katib-Experiments -





Verwenden Sie NetApp Snapshots zum Speichern von Daten zur Rückverfolgbarkeit

Während des Modelltrainings möchten wir möglicherweise zur Rückverfolgbarkeit einen Snapshot des Trainingsdatensatzes speichern. Dazu können wir der Pipeline einen Snapshot-Schritt hinzufügen, wie unten gezeigt. Um den Snapshot zu erstellen, können wir die "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)
```

Weitere Informationen finden Sie im "NetApp DataOps Toolkit-Beispiel für Kubeflow" für weitere Informationen.

Beispiel für Trident -Operationen

Dieser Abschnitt enthält Beispiele für verschiedene Vorgänge, die Sie möglicherweise mit Trident durchführen möchten.

Importieren eines vorhandenen Volumes

Wenn auf Ihrem NetApp Speichersystem/Ihrer NetApp-Speicherplattform Volumes vorhanden sind, die Sie auf Containern innerhalb Ihres Kubernetes-Clusters mounten möchten, die aber nicht an PVCs im Cluster gebunden sind, müssen Sie diese Volumes importieren. Sie können die Volume-Importfunktion von Trident verwenden, um diese Volumes zu importieren.

Die folgenden Beispielbefehle zeigen den Import eines Volumes mit dem Namen `pb_fg_all` . Weitere Informationen zu PVCs finden Sie im "offizielle Kubernetes-Dokumentation" . Weitere Informationen zur Funktion zum Importieren von Volumes finden Sie im "Trident -Dokumentation" .

Ein `accessModes` Wert von `ReadOnlyMany` ist in den Beispiel-PVC-Spezifikationsdateien angegeben. Weitere Informationen zum `accessMode` Feld finden Sie im "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
```

Bereitstellen eines neuen Volumes

Sie können Trident verwenden, um ein neues Volume auf Ihrem NetApp Speichersystem oder Ihrer NetApp-Plattform bereitzustellen.

Bereitstellen eines neuen Volumes mit kubectl

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

Ein `accessModes` Wert von `ReadWriteMany` ist in der folgenden Beispiel-PVC-Definitionsdatei angegeben. Weitere Informationen zum `accessMode` Feld finden Sie im "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
```

Bereitstellen eines neuen Volumes mit dem NetApp DataOps Toolkit

Sie können auch das NetApp DataOps Toolkit für Kubernetes verwenden, um ein neues Volume auf Ihrem NetApp -Speichersystem oder Ihrer NetApp-Plattform bereitzustellen. Das NetApp DataOps Toolkit für Kubernetes nutzt Trident zur Bereitstellung von Volumes, vereinfacht den Prozess für den Benutzer jedoch. Weitere Informationen finden Sie im"Dokumentation" für Details.

Beispiele für Hochleistungsjobs für AIPod Bereitstellungen

Ausführen einer Single-Node-KI-Workload

Um einen KI- und ML-Job mit einem Knoten in Ihrem Kubernetes-Cluster auszuführen, führen Sie die folgenden Aufgaben vom Bereitstellungs-Jump-Host aus. Mit Trident können Sie schnell und einfach ein Datenvolumen, das möglicherweise Petabyte an Daten enthält, für eine Kubernetes-Workload zugänglich machen. Um ein solches Datenvolumen innerhalb eines Kubernetes-Pods zugänglich zu machen, geben Sie einfach einen PVC in der Pod-Definition an.

In diesem Abschnitt wird davon ausgegangen, dass Sie die spezifische KI- und ML-Workload, die Sie in Ihrem Kubernetes-Cluster ausführen möchten, bereits containerisiert haben (im Docker-Containerformat).

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

   Dieser Beispieljob erfordert acht GPUs und kann daher auf einem einzelnen GPU-Workerknoten ausgeführt werden, der über acht oder mehr GPUs verfügt. Dieser Beispieljob könnte in einem Cluster übermittelt werden, für den kein Worker-Knoten mit acht oder mehr GPUs vorhanden ist oder der derzeit mit einer anderen Arbeitslast belegt ist. Wenn dies der Fall ist, verbleibt der Job im ausstehenden Status, bis ein solcher Worker-Knoten verfügbar wird.

   Um die Speicherbandbreite zu maximieren, wird das Volume, das die benötigten Trainingsdaten enthält, außerdem zweimal innerhalb des Pods gemountet, den dieser Job erstellt. Ein weiteres Volumen ist ebenfalls in der Kapsel montiert. Dieses zweite Volume wird zum Speichern von Ergebnissen und Metriken verwendet. Auf diese Datenträger wird in der Jobdefinition mithilfe der Namen der PVCs verwiesen. Weitere Informationen zu Kubernetes-Jobs finden Sie im "offizielle Kubernetes-Dokumentation" .

   Ein `emptyDir` Volumen mit einem `medium` Wert von `Memory` ist montiert an `/dev/shm` im Pod, den dieser Beispieljob erstellt. Die Standardgröße des `/dev/shm` Das von der Docker-Container-Laufzeit automatisch erstellte virtuelle Volume kann für die Anforderungen von TensorFlow manchmal nicht ausreichen. Montage eines `emptyDir` Volumen wie im folgenden Beispiel bietet eine ausreichend große `/dev/shm` virtuelles Volume. Weitere Informationen zu `emptyDir` Bände finden Sie im "offizielle Kubernetes-Dokumentation" .

   Der einzelne Container, der in dieser Beispiel-Jobdefinition angegeben ist, erhält eine `securityContext > privileged` Wert von `true` . Dieser Wert bedeutet, dass der Container effektiv Root-Zugriff auf dem Host hat. Diese Anmerkung wird in diesem Fall verwendet, da für die spezifische Arbeitslast, die ausgeführt wird, Root-Zugriff erforderlich ist. Insbesondere erfordert ein Cache-Löschvorgang, den die Arbeitslast durchführt, Root-Zugriff. Ob dies `privileged: true` Ob eine Annotation erforderlich ist, hängt von den Anforderungen der spezifischen Arbeitslast ab, 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. Bestätigen Sie, dass der in Schritt 1 erstellte Job ordnungsgemäß ausgeführt wird. Der folgende Beispielbefehl bestätigt, dass ein einzelner Pod für den Job erstellt wurde, wie in der Jobdefinition angegeben, und dass dieser Pod derzeit auf einem der GPU-Workerknoten 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. Die folgenden Beispielbefehle bestätigen, dass der Auftrag 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:** Bereinigen Sie Jobartefakte. Die folgenden Beispielbefehle zeigen das Löschen des in Schritt 1 erstellten Jobobjekts.

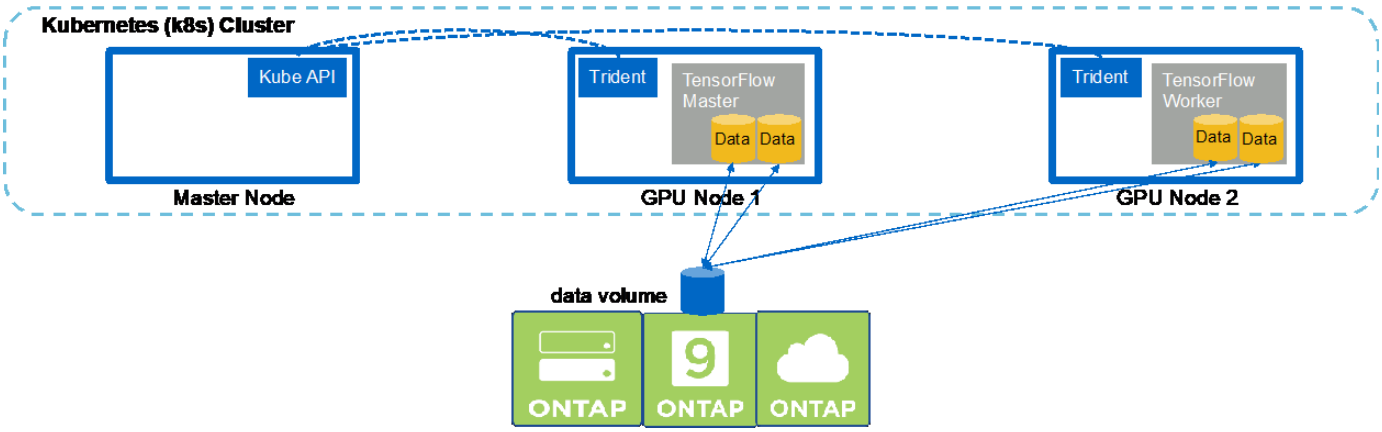
   Wenn Sie das Jobobjekt löschen, löscht Kubernetes automatisch alle zugehörigen Pods.

```
$ 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.
```

Ausführen einer synchronen verteilten KI-Workload

Um einen synchronen Multinode-KI- und ML-Job in Ihrem Kubernetes-Cluster auszuführen, führen Sie die folgenden Aufgaben auf dem Bereitstellungs-Jump-Host aus. Mit diesem Prozess können Sie die auf einem NetApp -Volume gespeicherten Daten nutzen und mehr GPUs verwenden, als ein einzelner Worker-Knoten bereitstellen kann. In der folgenden Abbildung sehen Sie eine Darstellung eines synchronen verteilten KI-Jobs.

Synchron verteilte Jobs können im Vergleich zu asynchron verteilten Jobs dazu beitragen, die Leistung und Trainingsgenauigkeit zu verbessern. Eine Diskussion der Vor- und Nachteile synchroner Jobs gegenüber asynchronen Jobs geht über den Rahmen dieses Dokuments hinaus.



1. Die folgenden Beispielbefehle zeigen die Erstellung eines Workers, der an der synchronen verteilten Ausführung desselben TensorFlow-Benchmark-Jobs teilnimmt, der im Beispiel im Abschnitt auf einem einzelnen Knoten ausgeführt wurde."Ausführen einer Single-Node-KI-Workload" . In diesem speziellen

Beispiel wird nur ein einziger Worker bereitgestellt, da der Job auf zwei Worker-Knoten ausgeführt wird.

Dieses Beispiel einer Worker-Bereitstellung erfordert acht GPUs und kann daher auf einem einzelnen GPU-Worker-Knoten ausgeführt werden, der über acht oder mehr GPUs verfügt. Wenn Ihre GPU-Workerknoten über mehr als acht GPUs verfügen, sollten Sie diese Zahl zur Leistungsmaximierung erhöhen, sodass sie der Anzahl der GPUs Ihrer Workerknoten entspricht. Weitere Informationen zu Kubernetes-Bereitstellungen finden Sie im "offizielle Kubernetes-Dokumentation" .

In diesem Beispiel wird eine Kubernetes-Bereitstellung erstellt, da dieser spezielle Container-Worker von alleine nie fertig werden würde. Daher ist es nicht sinnvoll, es mithilfe der Kubernetes-Jobkonstruktion bereitzustellen. Wenn Ihr Worker so konzipiert oder geschrieben ist, dass er die Aufgabe selbstständig erledigt, kann es sinnvoll sein, zum Bereitstellen Ihres Workers die Jobkonstruktion zu verwenden.

Der in dieser Beispiel-Bereitstellungsspezifikation angegebene Pod erhält eine `hostNetwork` Wert von `true` . Dieser Wert bedeutet, dass der Pod den Netzwerkstapel des Host-Workerknotens anstelle des virtuellen Netzwerkstapels verwendet, den Kubernetes normalerweise für jeden Pod erstellt. Diese Anmerkung wird in diesem Fall verwendet, da die spezifische Arbeitslast auf Open MPI, NCCL und Horovod angewiesen ist, um die Arbeitslast synchron und verteilt auszuführen. Daher ist Zugriff auf den Host-Netzwerkstapel erforderlich. Eine Diskussion über Open MPI, NCCL und Horovod geht über den Rahmen dieses Dokuments hinaus. Ob dies `hostNetwork: true` Ob eine Annotation erforderlich ist, hängt von den Anforderungen der spezifischen Arbeitslast ab, die Sie ausführen. Weitere Informationen zum `hostNetwork` Feld finden Sie im "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 Worker-Bereitstellung erfolgreich gestartet wurde. Die folgenden Beispielbefehle bestätigen, dass ein einzelner Worker-Pod für die Bereitstellung erstellt wurde, wie in der Bereitstellungsdefinition angegeben, und dass dieser Pod derzeit auf einem der GPU-Worker-Knoten 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. Erstellen Sie einen Kubernetes-Job für einen Master, der den synchronen Multinode-Job startet, daran teilnimmt und seine Ausführung verfolgt. Die folgenden Beispielbefehle erstellen einen Master, der die synchrone verteilte Ausführung desselben TensorFlow-Benchmark-Jobs startet, daran teilnimmt und verfolgt, der im Beispiel im Abschnitt auf einem einzelnen Knoten ausgeführt wurde."Ausführen einer Single-Node-KI-Workload" .

   Dieser beispielhafte Masterjob fordert acht GPUs an und kann daher auf einem einzelnen GPU-Workerknoten ausgeführt werden, der über acht oder mehr GPUs verfügt. Wenn Ihre GPU-Workerknoten über mehr als acht GPUs verfügen, sollten Sie diese Zahl zur Leistungsmaximierung erhöhen, sodass sie der Anzahl der GPUs Ihrer Workerknoten entspricht.

   Der in dieser Beispiel-Jobdefinition angegebene Master-Pod erhält einen `hostNetwork` Wert von `true` , genau wie die Arbeiterkapsel eine `hostNetwork` Wert von `true` in Schritt 1. Einzelheiten dazu, warum dieser Wert erforderlich 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. Bestätigen Sie, dass der in Schritt 3 erstellte Masterjob ordnungsgemäß 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-Knoten ausgeführt wird. Sie sollten auch sehen, dass der Worker-Pod, den Sie ursprünglich in Schritt 1 gesehen haben, noch ausgeführt wird und dass die Master- und Worker-Pods auf verschiedenen Knoten 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. Bestätigen Sie, dass der in Schritt 3 erstellte Masterjob erfolgreich abgeschlossen wurde. Die folgenden Beispielbefehle bestätigen, dass der Auftrag 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 Worker-Bereitstellung, wenn Sie sie nicht mehr benötigen. Die folgenden Beispielbefehle zeigen das Löschen des in Schritt 1 erstellten Worker-Bereitstellungsobjekts.

Wenn Sie das Worker-Bereitstellungsobjekt 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:** Bereinigen Sie die Master-Job-Artefakte. Die folgenden Beispielbefehle zeigen das Löschen des in Schritt 3 erstellten Master-Job-Objekts.

   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.
```