NetApp AI Control Plane

NetApp Solutions

NetApp
April 18, 2024

This PDF was generated from https://docs.netapp.com/de-de/netapp-solutions/ai/aicp_introduction.html on April 18, 2024. Always check docs.netapp.com for the latest.

Inhalt

NetApp AI Control Plane . . . 1
TR-4798: NetApp AI Control Plane . . . 1
Konzepte und Komponenten . . . 2
Hardware- und Software-Anforderungen . . . 11
Kubernetes Deployment . . . 12
Implementierung und Konfiguration von NetApp Trident . . . 13
Kubeflow Deployment . . . 21
Beispiel Kubeflow Operations und Tasks . . . 30
Apache Airflow Deployment . . . 38
Beispiel Für Apache Airflow Workflows . . . 42
Beispiel Für Trident-Vorgänge . . . 42
Beispiele für hochperformante Jobs für ONTAP KI-Implementierungen . . . 46
Performance-Tests . . . 57
Schlussfolgerung . . . 57

NetApp AI Control Plane

TR-4798: NetApp AI Control Plane

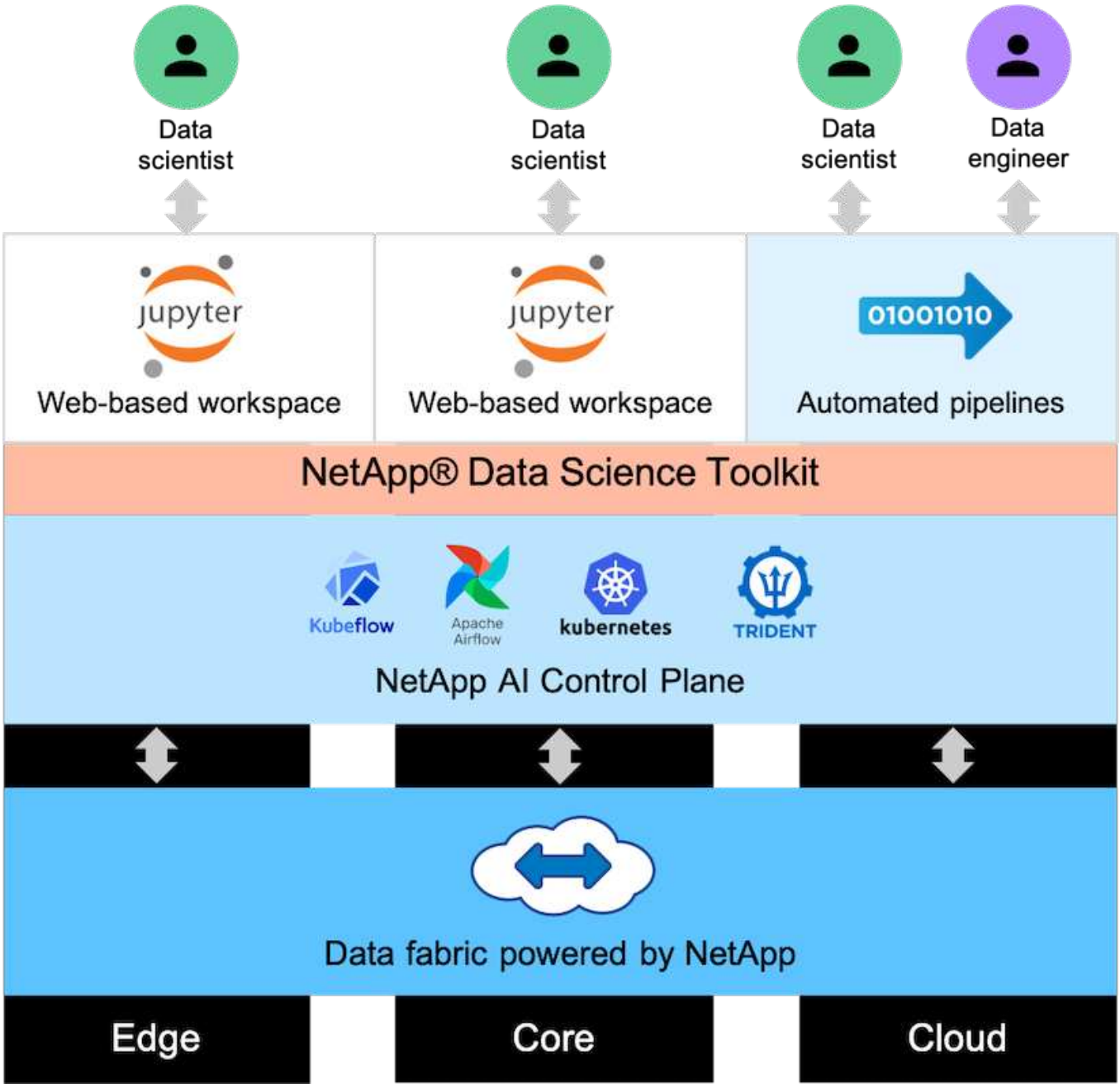
Mike Oglesby, NetApp

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

Dieser Bericht zeigt Ihnen, wie Sie einen Daten-Namespace schnell klonen können. Außerdem zeigt sie, wie Daten nahtlos über Standorte und Regionen hinweg repliziert werden, um eine zusammenhängende und einheitliche KI/ML/DL-Datenpipeline zu erstellen. Darüber hinaus werden die Trainings-Workflows für AI, ML und DL definiert und implementiert, die die nahezu sofortige Erstellung von Daten und Modellbasiden für Rückverfolgbarkeit und Versionierung enthalten. Mit dieser Lösung können Sie verfolgen, welches Modell-Training zurück zu dem genauen Datensatz führt, der zum Training und/oder zur Validierung des Modells verwendet wurde. Abschließend erfahren Sie in diesem Dokument, wie Sie Jupyter Notebook-Workspaces mit Zugriff auf riesige Datensätze schnell bereitstellen können.

Hinweis: Für verteilte Schulungen im HPC-Stil mit einer großen Anzahl von GPU-Servern, die einen gemeinsamen Zugriff auf denselben Datensatz benötigen oder wenn Sie ein paralleles Dateisystem benötigen/bevorzugen, schauen Sie sich das hier an "TR-4890". In diesem technischen Bericht wird die Vorgehensweise erläutert "Die vollständig unterstützte parallele NetApp Lösung BeeGFS von NetApp" Teil der NetApp KI-Kontrollebene. Diese Lösung wurde entwickelt, um von einer Handvoll NVIDIA DGX A100-Systemen bis zu einem voll aufgeblasen SuperPOD mit 140 Nodes skalieren zu können.

Die NetApp AI Control Plane ist auf Data Scientists und Data Engineers ausgerichtet und benötigt daher nur eine minimale NetApp oder NetApp ONTAP® Expertise. Bei dieser Lösung lassen sich Datenmanagementfunktionen mit einfachen und vertrauten Tools und Schnittstellen ausführen. Wenn Sie bereits NetApp Storage in Ihrer Umgebung haben, können Sie noch heute die NetApp AI Control Plane testen. Wenn Sie die Lösung testen möchten, aber noch nicht bereits über NetApp Storage verfügen, besuchen Sie "cloud.netapp.com", Und Sie sind innerhalb von Minuten mit einer Cloud-basierten NetApp Storage-Lösung einsatzbereit. Die folgende Abbildung bietet eine Visualisierung der Lösung.



Konzepte und Komponenten

Künstliche Intelligenz

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

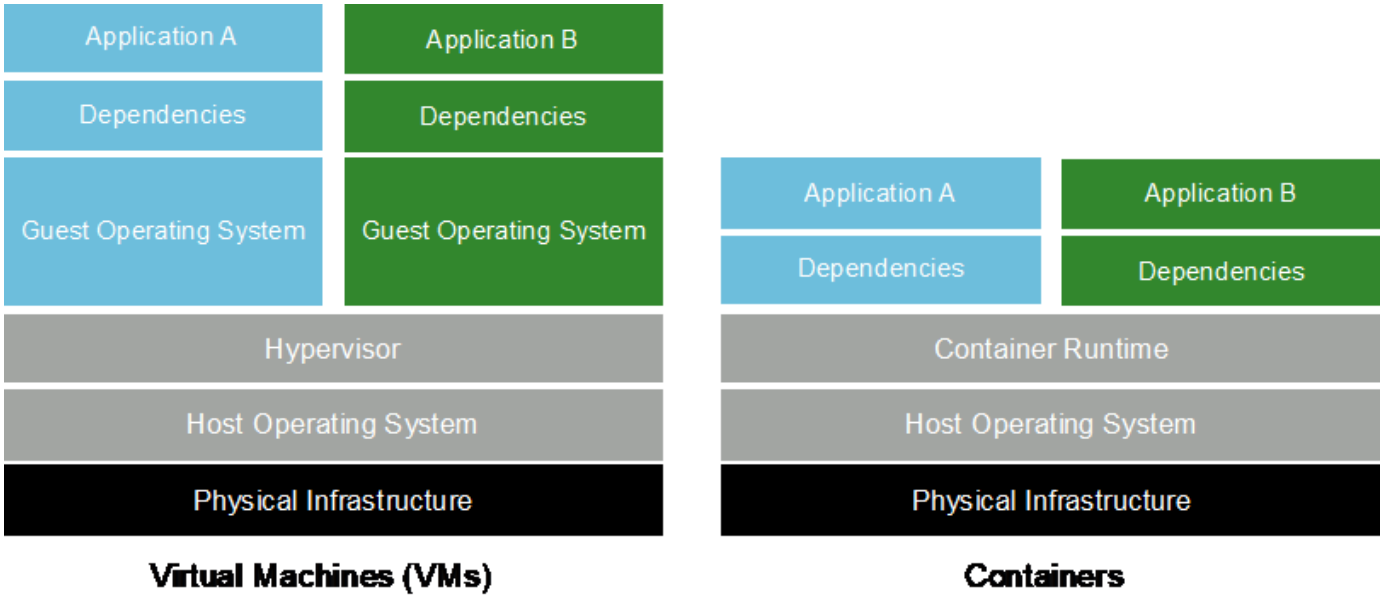
- Analyse großer Datenmengen für bislang unbekannte geschäftliche Einblicke
- Direkte Interaktion mit Kunden über natürliche Sprachverarbeitung
- Automatisierung verschiedener Geschäftsprozesse und Funktionen

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

Container

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

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



Kubernetes

Kubernetes ist eine ursprünglich von Google entwickelte Open-Source-Plattform zur Container-Orchestrierung, die jetzt von der Cloud Native Computing Foundation (CNCF) verwaltet wird. Kubernetes ermöglicht die Automatisierung von Implementierungs-, Management- und Skalierungsfunktionen für Container-Applikationen. In den letzten Jahren hat sich Kubernetes zur führenden Plattform für die Container-Orchestrierung entwickelt. Obwohl andere Container-Verpackungsformate und Laufzeiten unterstützt werden, wird Kubernetes am häufigsten als Orchestrierungssystem für Docker Container verwendet. Weitere Informationen finden Sie auf der "Kubernetes-Website".

NetApp Trident

Trident ist ein Open-Source-Storage-Orchestrator, der von NetApp entwickelt und gewartet wird. Damit wird die Erstellung, das Management und die Nutzung von persistentem Storage für Kubernetes-Workloads erheblich vereinfacht. Trident, selbst eine native Kubernetes-Applikation, wird direkt in einem Kubernetes Cluster ausgeführt. Mit Trident können Kubernetes-Benutzer (Entwickler, Data Scientists, Kubernetes Administratoren

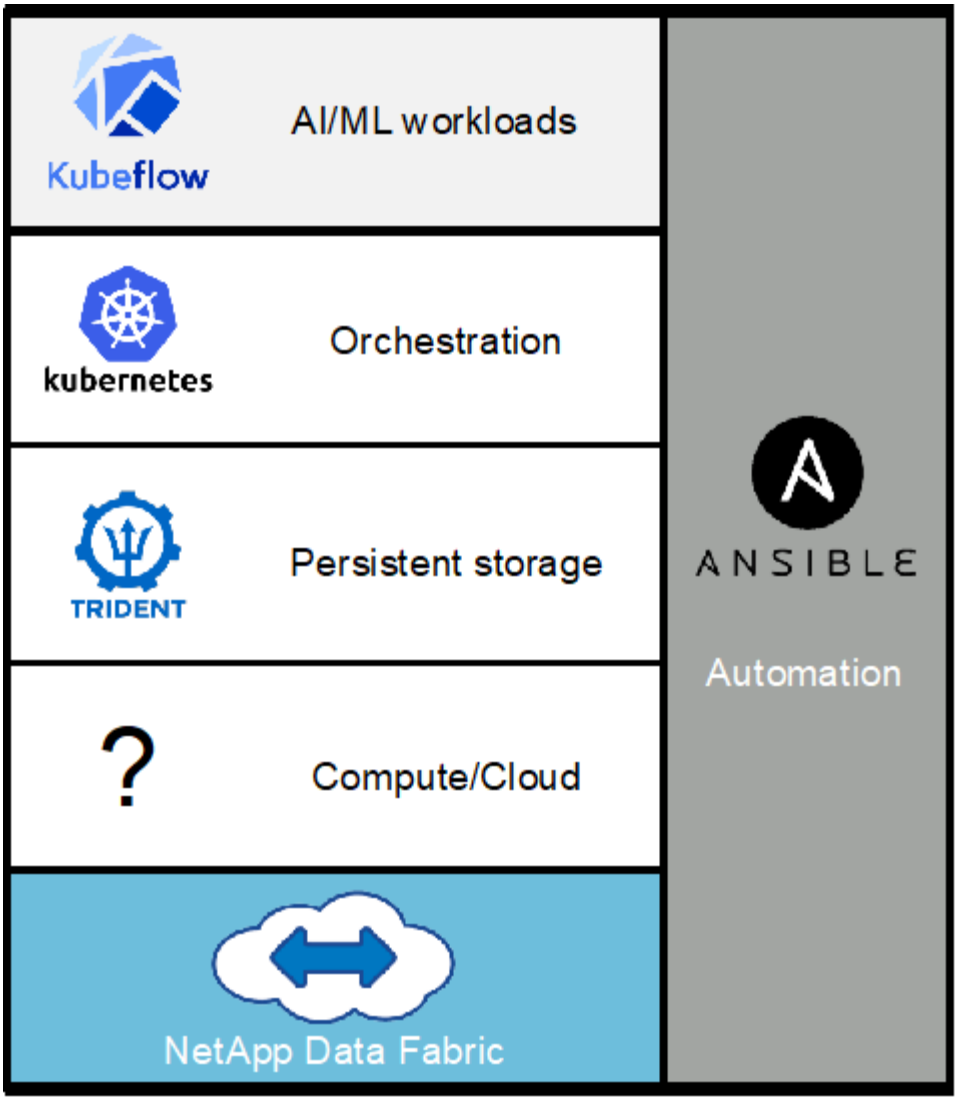
usw.) persistente Storage-Volumes im gewohnten Kubernetes-Standardformat erstellen, managen und interagieren. Gleichzeitig können sie von den erweiterten Datenmanagement-Funktionen von NetApp und der Data Fabric Strategie von NetApp profitieren. Trident abstrahiert die Komplexität von persistentem Storage und vereinfacht die Nutzung. Weitere Informationen finden Sie auf der "Trident Website".

NVIDIA DeepOps

DeepOps ist ein Open-Source-Projekt von NVIDIA, das mithilfe von Ansible die Implementierung von GPU-Server-Clustern gemäß Best Practices automatisiert. DeepOps ist modular aufgebaut und kann für verschiedene Implementierungsaufgaben verwendet werden. DeepOps wird in diesem Dokument und der hier beschriebene Validierungsübung verwendet, um ein Kubernetes-Cluster zu implementieren, der aus GPU-Server-Worker-Nodes besteht. Weitere Informationen finden Sie auf der "DeepOps-Website".

Kubeflow

Kubeflow ist ein Open Source AI und ML Toolkit für Kubernetes und wurde ursprünglich von Google entwickelt. Das Kubeflow-Projekt macht Implementierungen von KI- und ML-Workflows auf Kubernetes einfach, tragbar und skalierbar. Kubeflow abstrahiert die Besonderheiten von Kubernetes und ermöglicht Data Scientists, sich auf das zu konzentrieren, was sie am besten kennen—Datenwissenschaft. Eine Visualisierung finden Sie in der folgenden Abbildung. Kubeflow hat seine Bedeutung gewonnen, da DIE IT-Abteilungen in Unternehmen zunehmend auf Kubernetes standardisiert haben. Weitere Informationen finden Sie auf der "Kubeflow-Website".



Kubeflow-Pipelines

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

Jupyter Notebook Server

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

Apache Airflow

Apache Airflow ist eine Open-Source-Workflow-Managementplattform, die programmatisches Authoring, Scheduling und Monitoring für komplexe Unternehmens-Workflows ermöglicht. Sie wird häufig zur Automatisierung von ETL- und Daten-Pipeline-Workflows verwendet, beschränkt sich jedoch nicht auf diese

Arten von Workflows. Das Airflow-Projekt wurde von Airbnb gestartet, ist aber inzwischen sehr populär in der Branche und fällt nun unter die Schirmherrschaft der Apache Software Foundation. Der Luftstrom wird in Python geschrieben, Airflow-Workflows werden über Python-Skripte erstellt und Airflow wird nach dem Prinzip „Configuration as Code“ entworfen. Viele Benutzer von Airflow setzen nun Airflow auf Kubernetes aus.

Gesteuerte Acyclic-Grafiken (DAGs)

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

NetApp ONTAP 9

NetApp ONTAP 9 ist die neueste Generation der Storage-Managementsoftware von NetApp. Mit ihr können Unternehmen wie Ihres ihre Infrastruktur modernisieren und auf ein Cloud-fähiges Datacenter umstellen. Dank der erstklassigen Datenmanagementfunktionen lassen sich mit ONTAP Ihre Daten mit einem einzigen Toolset managen und schützen, ganz gleich, wo sich diese Daten befinden. Zudem können Sie die Daten problemlos dorthin verschieben, wo Sie sie benötigen: Zwischen Edge, Core und Cloud. ONTAP 9 umfasst zahlreiche Funktionen, die das Datenmanagement vereinfachen, geschäftskritische Daten beschleunigen und schützen und die Infrastruktur über Hybrid-Cloud-Architekturen hinweg zukunftssicher machen.

Datenmanagement Vereinfachen

Für Ihren Enterprise-IT-Betrieb spielt Datenmanagement eine zentrale Rolle, da es die richtigen Ressourcen für Ihre Applikationen und Datensätze verwenden kann. ONTAP umfasst die folgenden Funktionen, um den Betrieb zu optimieren und zu vereinfachen und damit die Gesamtbetriebskosten zu senken:

- **Inline-Data-Compaction und erweiterte Deduplizierung.** Data-Compaction reduziert den ungenutzten Speicherplatz in Storage-Blöcken, während Deduplizierung die effektive Kapazität deutlich steigert.
- **Minimum, Maximum und anpassungsfähige Quality of Service (QoS)** durch granulare QoS-Kontrollen können geschäftskritische Applikationen auch in hochgemeinsam genutzten Umgebungen weiterhin Performance-Level aufrechterhalten.
- **ONTAP FabricPool.** Diese Funktion bietet automatisches Tiering von „kalten“ Daten in Private- und Public-Cloud-Storage-Optionen, einschließlich Amazon Web Services (AWS), Azure und objektbasiertem NetApp StorageGRID Storage.

Daten beschleunigen und sichern

ONTAP bietet überdurchschnittliche Performance und Datensicherung, erweitert durch folgende Funktionen:

- **Hohe Performance und niedrige Latenz.** ONTAP bietet den höchstmöglichen Durchsatz bei geringstmöglicher Latenz.
- **NetApp ONTAP FlexGroup-Technologie.** Ein FlexGroup Volume ist ein hochperformanter Daten-Container, der sich linear auf bis zu 20 PB und 400 Milliarden Dateien skalieren lässt und über einen Single Namespace das Datenmanagement vereinfacht.
- **Datensicherung.** ONTAP bietet integrierte Datensicherungsfunktionen mit einem einheitlichen Management über alle Plattformen hinweg.
- **NetApp Volume Encryption.** ONTAP bietet native Verschlüsselung auf Volume-Ebene mit integriertem und externem Verschlüsselungsmanagement.

Zukunftssichere Infrastruktur

ONTAP 9 hilft Unternehmen dabei, flexibel auf ständig wechselnde Geschäftsanforderungen zu reagieren:

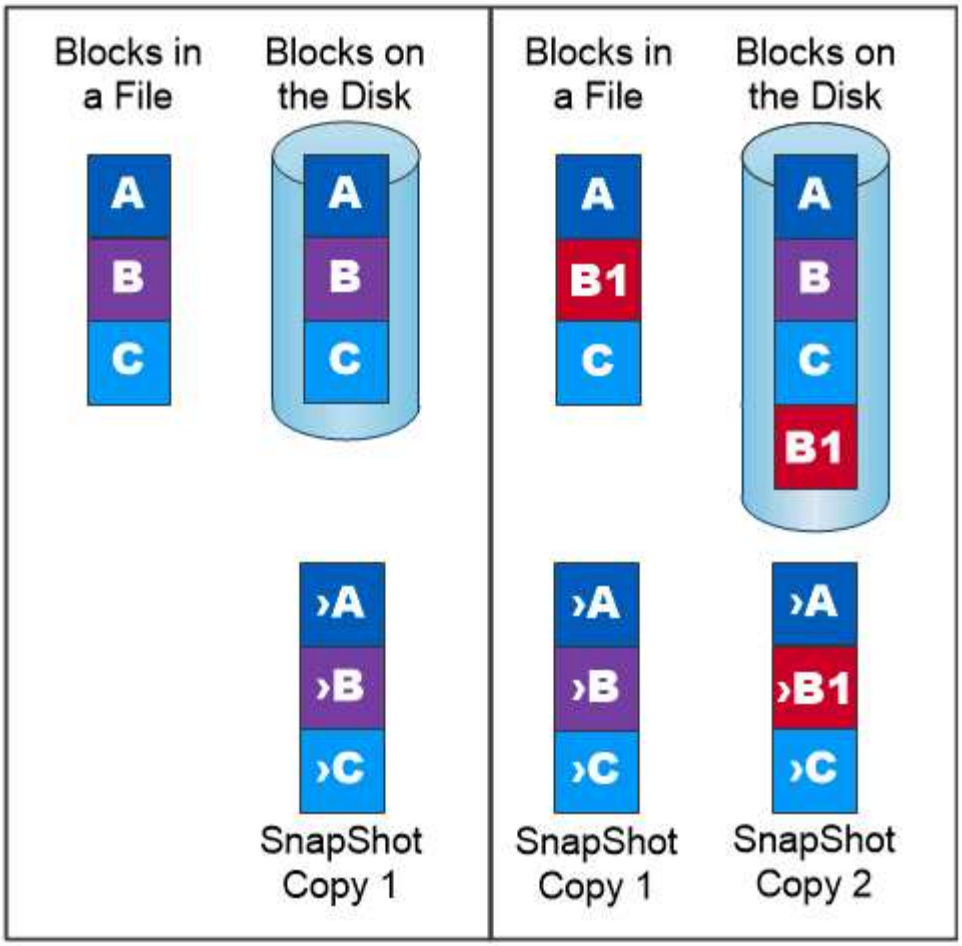
- **Nahtlose Skalierung und unterbrechungsfreier Betrieb.** ONTAP unterstützt das unterbrechungsfreie Hinzufügen von Kapazitäten zu bestehenden Controllern sowie das Scale-out von Clustern. Damit lassen sich Upgrades auf die neuesten Technologien wie NVMe und 32 GB FC ohne teure Datenmigrationen oder Ausfälle durchführen.
- **Cloud Connection.** ONTAP ist eine der Storage-Managementsoftware mit der umfassendsten Cloud-Integration und bietet Optionen für softwaredefinierten Storage (ONTAP Select) und Cloud-native Instanzen (NetApp Cloud Volumes Service) in allen Public Clouds.
- **Integration in moderne Applikationen.** durch den Einsatz derselben Infrastruktur, die bereits vorhandene Enterprise-Applikationen unterstützt, bietet ONTAP Datenservices der Enterprise-Klasse für Plattformen und Applikationen der neuesten Generation, wie OpenStack, Hadoop und MongoDB.

NetApp Snapshot Kopien

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

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

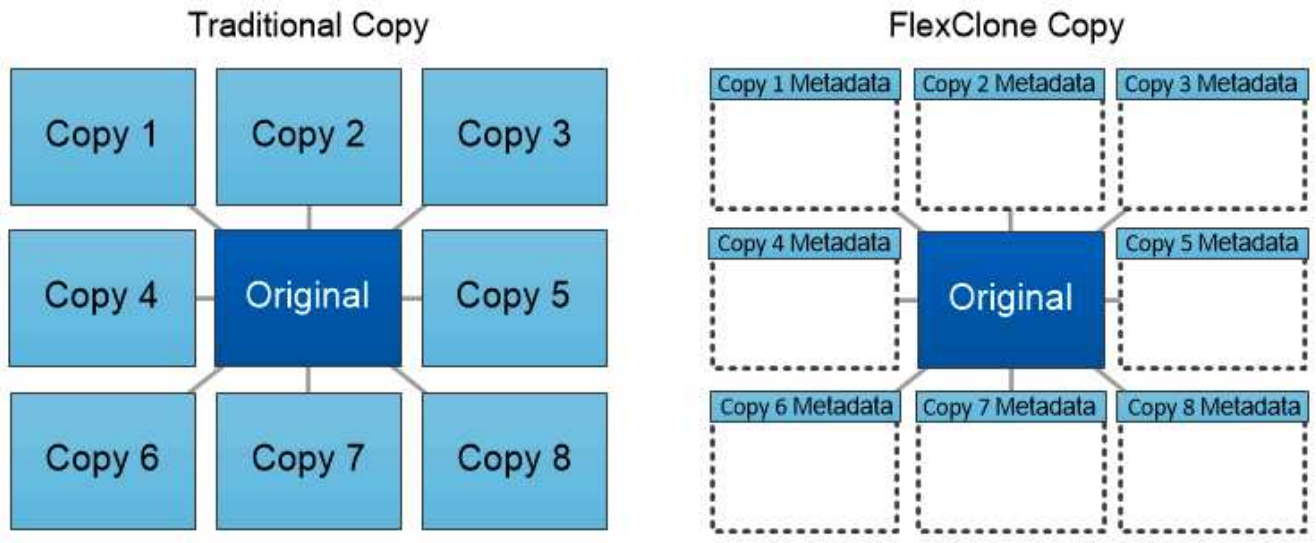
Sie können eine Snapshot Kopie verwenden, um einzelne Dateien oder LUNs wiederherzustellen oder den gesamten Inhalt eines Volume wiederherzustellen. ONTAP vergleicht Zeigerinformationen in der Snapshot-Kopie mit Daten auf der Festplatte, um das fehlende oder beschädigte Objekt ohne Ausfallzeiten und hohe Performance-Kosten zu rekonstruieren.



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

NetApp FlexClone Technologie

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

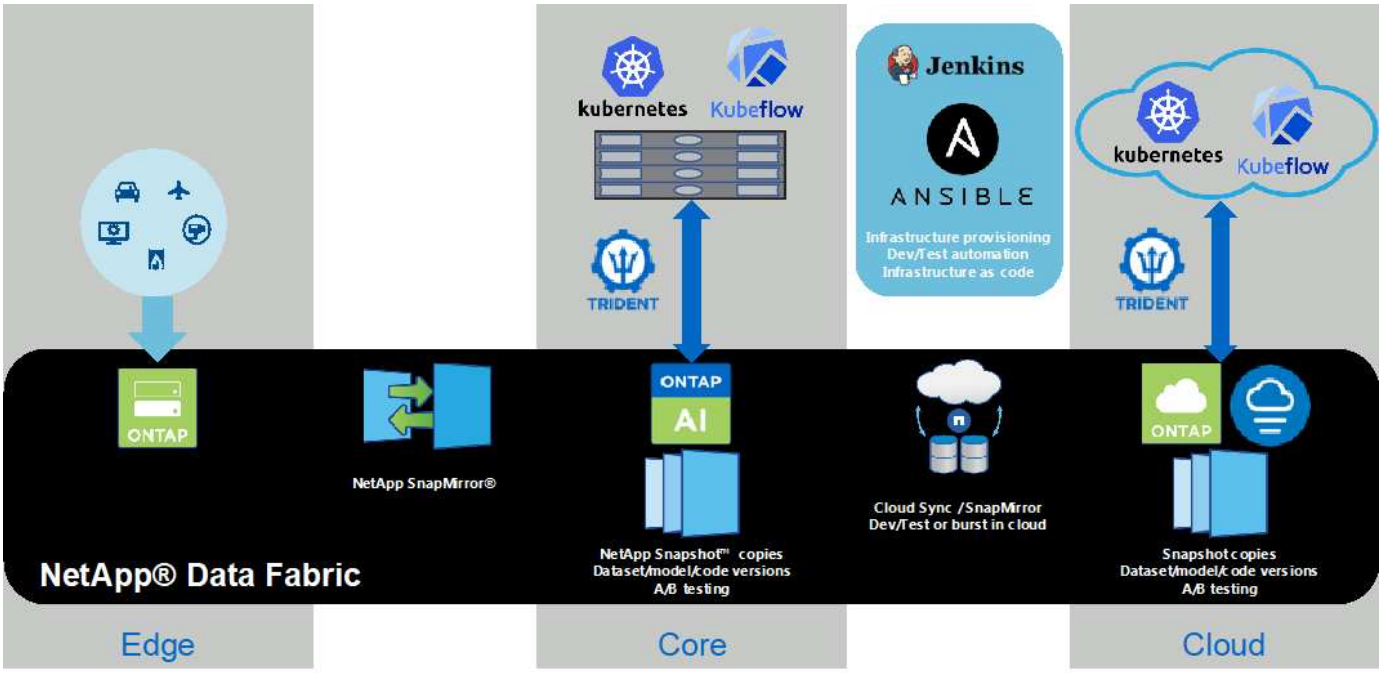


FlexClone copies share data blocks with their parents, consuming no storage except what is required for metadata.

NetApp SnapMirror Datenreplizierung

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

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



NetApp BlueXP Kopie und Synchronisierung

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

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

BlueXP Copy and Sync ist ein Software-as-a-Service-Tool (SaaS), das sich äußerst einfach einrichten und verwenden lässt. Datentransfers, die durch BlueXP Copy und Sync ausgelöst werden, erfolgen durch Datenmanager. Die Datenmanager von BlueXP Copy und Sync können in AWS, Azure, Google Cloud Platform oder lokal implementiert werden.

NetApp XCP

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

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

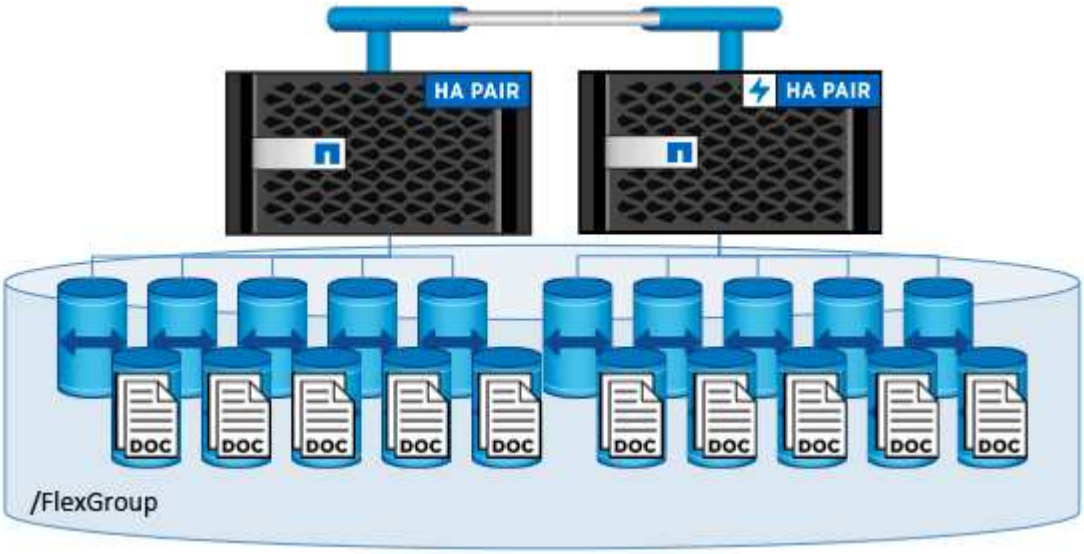
Die hostbasierte Software NetApp XCP File Analytics erkennt Dateifreigaben, führt Scans im Filesystem aus und bietet ein Dashboard für Dateianalysen. XCP File Analytics ist sowohl mit Systemen von NetApp als auch mit Systemen anderer Hersteller kompatibel und wird auf Linux- oder Windows-Hosts ausgeführt,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

NetApp ONTAP FlexGroup Volumes

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

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

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



Hardware- und Software-Anforderungen

Die NetApp AI Control Plane Lösung ist nicht von dieser spezifischen Hardware abhängig. Die Lösung ist mit jeder physischen NetApp Storage Appliance, jeder softwaredefinierten Instanz oder jedem Cloud-Service kompatibel, der von Trident unterstützt wird. Hierzu zählen beispielsweise ein NetApp AFF Storage-System, Azure NetApp Files, NetApp Cloud Volumes Service, eine softwaredefinierte Storage-Instanz von NetApp ONTAP Select oder eine NetApp Cloud Volumes ONTAP Instanz. Außerdem kann die Lösung auf jedem beliebigen Kubernetes Cluster implementiert werden, solange die verwendete Kubernetes-Version von Kubeflow und NetApp Trident unterstützt wird. Eine Liste der von Kubeflow unterstützten Kubernetes-Versionen finden Sie im "Offizielle Dokumentation von Kubeflow". Eine Liste der von Trident unterstützten Kubernetes-Versionen finden Sie im "Trident Dokumentation". In den folgenden Tabellen finden Sie

Einzelheiten zur Umgebung, die zur Validierung der Lösung verwendet wurde.

Infrastrukturkomponente	Menge	Details	Betriebssystem
Bereitstellung Jump-Host	1	VM	Ubuntu 20.04.2 LTS
Kubernetes Master-Nodes	1	VM	Ubuntu 20.04.2 LTS
Kubernetes-Worker-Nodes	2	VM	Ubuntu 20.04.2 LTS
Kubernetes-GPU-Worker-Nodes	2	NVIDIA DGX-1 (Bare-Metal)	NVIDIA DGX OS 4.0.5 (basierend auf Ubuntu 18.04.2 LTS)
Storage	1 HA-Paar	NetApp AFF A220	NetApp ONTAP 9.7 P6

Software-Komponente	Version
Apache Airflow	2.0.1
Apache Airflow Helm Chart	8.0.8
Docker	19.03.12
Kubeflow	1.2
Kubernetes	1.18.9
NetApp Trident	21.01.2
NVIDIA DeepOps	Trident-Implementierungsfunktionen von der Master-Zweigstelle bei der Übertragung "61898cdfda"; Alle anderen Funktionen von Version 21.03

Unterstützung

NetApp bietet keine Enterprise-Unterstützung für Apache Airflow, Docker, Kubeflow, Kubernetes oder NVIDIA DeepOps. Wenn Sie an einer vollständig unterstützten Lösung mit Funktionen interessiert sind, die ähnlich der NetApp KI Control Plane Lösung sind, "Kontakt zu NetApp" Über vollständig unterstützte KI/ML-Lösungen, die NetApp gemeinsam mit Partnern anbietet.

Kubernetes Deployment

In diesem Abschnitt werden die Aufgaben beschrieben, die zur Implementierung eines Kubernetes-Clusters erforderlich sind, in dem die NetApp AI Control Plane Lösung implementiert werden kann. Wenn Sie bereits einen Kubernetes Cluster besitzen, können Sie diesen Abschnitt überspringen, solange Sie eine von Kubeflow und NetApp Trident unterstützte Version von Kubernetes ausführen. Eine Liste der von Kubeflow unterstützten Kubernetes-Versionen finden Sie im "Offizielle Dokumentation von Kubeflow". Eine Liste der von Trident unterstützten Kubernetes-Versionen finden Sie im "Trident Dokumentation".

Für lokale Kubernetes-Implementierungen, die Bare-Metal-Nodes mit NVIDIA-GPU(s) einbinden, empfiehlt NetApp die Verwendung des NVIDIA DeepOps Kubernetes-Implementierungstools. In diesem Abschnitt wird

die Bereitstellung eines Kubernetes-Clusters mit DeepOps beschrieben.

Voraussetzungen

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

1. Sie haben bereits alle Bare-Metal-Kubernetes-Nodes (z. B. ein NVIDIA DGX-System, das Teil eines ONTAP AI Pods ist) gemäß den Standardkonfigurationsanweisungen konfiguriert.
2. Sie haben ein unterstütztes Betriebssystem auf allen Kubernetes Master- und Worker-Nodes und auf einem Deployment Jump-Host installiert. Eine Liste der von DeepOps unterstützten Betriebssysteme finden Sie im "DeepOps GitHub-Website".

Verwenden Sie NVIDIA DeepOps zum Installieren und Konfigurieren von Kubernetes

Um Ihren Kubernetes-Cluster mit NVIDIA DeepOps zu implementieren und zu konfigurieren, führen Sie die folgenden Aufgaben über einen Bereitstellungs-Jump-Host aus:

1. Laden Sie NVIDIA DeepOps herunter, indem Sie den Anweisungen auf der folgen "Erste Schritte" Auf der NVIDIA DeepOps GitHub Website.
2. Implementieren Sie Kubernetes in Ihrem Cluster, indem Sie die Anweisungen auf dem befolgen "Seite „Kubernetes Deployment Guide“" Auf der NVIDIA DeepOps GitHub Website.

Implementierung und Konfiguration von NetApp Trident

Implementierung und Konfiguration von NetApp Trident

In diesem Abschnitt werden die Aufgaben beschrieben, die Sie abschließen müssen, um NetApp Trident in Ihrem Kubernetes Cluster zu installieren und zu konfigurieren.

Voraussetzungen

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

1. Sie verfügen bereits über ein funktionierendes Kubernetes Cluster und führen eine von Trident unterstützte Version von Kubernetes aus. Eine Liste der unterstützten Versionen finden Sie im "Trident Dokumentation".
2. Sie verfügen bereits über eine funktionierende NetApp Storage Appliance, softwaredefinierte Instanz oder Cloud-Storage-Service, die von Trident unterstützt wird.

Installation Von Trident

Um NetApp Trident in Ihrem Kubernetes-Cluster zu installieren und zu konfigurieren, führen Sie die folgenden Aufgaben über den Jump-Host der Implementierung aus:

1. Trident lässt sich mit einer der folgenden Methoden implementieren:
    - Wenn Sie zur Implementierung Ihres Kubernetes-Clusters NVIDIA DeepOps verwendet haben, können Sie Trident auch mit NVIDIA DeepOps in Ihrem Kubernetes-Cluster implementieren. Folgen Sie zur Implementierung von Trident mit DeepOps dem "Trident Implementierungsanleitungen" Auf der NVIDIA DeepOps GitHub Website.

- Wenn Sie NVIDIA DeepOps nicht zur Implementierung Ihres Kubernetes-Clusters verwendet haben oder Trident einfach manuell implementieren möchten, können Sie Trident anhand der implementieren "Implementierungsanleitungen" In der Trident Dokumentation. Erstellen Sie mindestens ein Trident Back-End und mindestens eine Kubernetes StorageClass. Sie erhalten weitere Informationen zur Konfiguration "Back-Ends" Und "StorageClasses" Siehe verlinkte Unterabschnitte bei NetApp Docs.

> Wenn Sie die NetApp AI Control Plane Lösung auf einem ONTAP AI POD implementieren, lesen Sie "Beispiel Trident Back-Ends für ONTAP KI-Implementierungen" Beispiele für verschiedene Trident Back-Ends, die Sie möglicherweise erstellen möchten, und "Beispiel: Kubernetes Storageclasses für ONTAP AI Implementierungen" Für einige Beispiele für verschiedene Kubernetes StorageClasses, die Sie vielleicht erstellen möchten.

Beispiel Trident Back-Ends für ONTAP KI-Implementierungen

Bevor Sie Trident zur dynamischen Bereitstellung von Storage-Ressourcen innerhalb Ihres Kubernetes-Clusters verwenden können, müssen Sie ein oder mehrere Trident Back-Ends erstellen. Die folgenden Beispiele stellen unterschiedliche Arten von Backend dar, die erstellt werden sollten, wenn Sie die NetApp AI Control Plane Lösung auf einem ONTAP AI POD implementieren. Weitere Informationen zu Backends finden Sie im "Trident Dokumentation".

1. NetApp empfiehlt die Erstellung eines FlexGroup-fähigen Trident-Back-End für jede Daten-LIF (logische Netzwerkschnittstelle mit Datenzugriff), die Sie auf Ihrem NetApp AFF System verwenden möchten. So können Sie einen Ausgleich von Volume-Mounts über LIFs hinweg erzielen

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

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

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

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

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

2. NetApp empfiehlt außerdem, ein oder mehrere FlexVol-fähige Trident Back-Ends zu erstellen. Wenn Sie FlexGroup Volumes für das Training von Datensatz-Storage verwenden, könnten Sie FlexVol Volumes für die Speicherung von Ergebnissen, Ausgabe, Debug-Informationen usw. verwenden. Falls Sie FlexVol Volumes verwenden möchten, müssen Sie ein oder mehrere FlexVol-aktivierte Trident-Backends erstellen. Die folgenden Beispielbefehle zeigen die Erstellung eines einzelnen FlexVol-fähigen Trident Back-End, das eine einzelne Daten-LIF verwendet.

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

Beispiel: Kubernetes StorageClasses für ONTAP KI-Implementierungen

Bevor Trident zur dynamischen Provisionierung von Storage-Ressourcen innerhalb Ihres Kubernetes-Clusters verwendet werden kann, müssen Sie einen oder mehrere Kubernetes StorageClasses erstellen. Die folgenden Beispiele stellen verschiedene Arten

von StorageClasses dar, die Sie möglicherweise erstellen möchten, wenn Sie die NetApp AI Control Plane Lösung auf einem ONTAP AI POD implementieren. Weitere Informationen zu StorageClasses finden Sie im "Trident Dokumentation".

1. NetApp empfiehlt, für jedes FlexGroup-fähige Trident Back-End, das Sie im Abschnitt erstellt haben, eine separate StorageClass zu erstellen "Beispiel Trident Back-Ends für ONTAP KI-Implementierungen", Schritt 1. Mit diesen granularen StorageClasses können Sie NFS-Mounts hinzufügen, die bestimmten LIFs entsprechen (die LIFs, die Sie beim Erstellen der Trident Back-Ends angegeben haben), als ein bestimmtes Back-End, das in der StorageClass Spec-Datei angegeben ist. Die folgenden Beispielbefehle zeigen die Erstellung von zwei StorageClasses, die den beiden Beispiel Backends entsprechen, die im Abschnitt erstellt wurden "Beispiel Trident Back-Ends für ONTAP KI-Implementierungen", Schritt 1. Weitere Informationen zu StorageClasses finden Sie im "Trident Dokumentation".

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

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

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

   Im folgenden Beispiel wird in der StorageClass-Definitionsdatei kein bestimmtes Back-End angegeben, da nur ein FlexVol-fähiges Trident-Backend erstellt wurde. Wenn Sie Volumes mit Kubernetes verwalten, die diese StorageClass verwenden, versucht Trident, jedes verfügbare Backend zu verwenden, das den verwendet `ontap-nas` Treiber.

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

3. NetApp empfiehlt außerdem, eine generische StorageClass für FlexGroup Volumes zu erstellen. Die folgenden Beispielbefehle zeigen die Erstellung eines einzelnen allgemeinen StorageClass für FlexGroup Volumes.

   Beachten Sie, dass in der StorageClass-Definitionsdatei kein bestimmtes Backend angegeben ist. Wenn Sie daher Kubernetes für die Administration von Volumes verwenden, die diese StorageClass verwenden, versucht Trident, jedes verfügbare Back-End zu verwenden, das die verwendet `ontap-nas-flexgroup` Treiber.

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

Kubeflow Deployment

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

Voraussetzungen

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

1. Sie haben bereits einen funktionierenden Kubernetes-Cluster und führen eine Version von Kubernetes aus, die von Kubeflow unterstützt wird. Eine Liste der unterstützten Versionen finden Sie im "Offizielle Dokumentation von Kubeflow".
2. Sie haben NetApp Trident bereits in Ihrem Kubernetes Cluster installiert und konfiguriert, wie in beschrieben "Implementierung und Konfiguration von Trident".

Standard-Kubernetes StorageClass festlegen

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

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

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

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

Verwenden Sie NVIDIA DeepOps zur Implementierung von Kubeflow

NetApp empfiehlt, das von NVIDIA DeepOps zur Verfügung gestellte Kubeflow Implementierungs-Tool zu verwenden. Um Kubeflow in Ihren Kubernetes-Cluster mit dem DeepOps-Implementierungstool zu implementieren, führen Sie die folgenden Aufgaben auf dem Sprunghost der Bereitstellung aus.

> Alternativ können Sie Kubeflow manuell bereitstellen, indem Sie dem folgen "Installationsanweisungen" In der offiziellen Kubeflow Dokumentation

1. Implementieren Sie Kubeflow in Ihrem Cluster, indem Sie den folgen "Anweisungen zur Bereitstellung von Kubeflow" Auf der NVIDIA DeepOps GitHub Website.
2. Beachten Sie unten die URL des Kubeflow Dashboard, die vom DeepOps Kubeflow Bereitstellungs-Tool ausgegeben wird.

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

3. Vergewissern Sie sich, dass alle im Kubeflow Namespace bereitgestellten Pods A zeigen `STATUS` Von `Running` Und vergewissern Sie sich, dass sich keine im Namespace bereitgestellten Komponenten in einem Fehlerzustand befinden. Es kann mehrere Minuten dauern, bis alle Pods beginnen.

```
$ kubectl get all -n kubeflow
NAME                                                          READY
STATUS    RESTARTS   AGE
pod/admission-webhook-bootstrap-stateful-set-0                1/1
Running   0          95s
pod/admission-webhook-deployment-6b89c84c98-vrtbh             1/1
Running   0          91s
pod/application-controller-stateful-set-0                     1/1
Running   0          98s
pod/argo-ui-5dcf5d8b4f-m2wn4                                  1/1
Running   0          97s
pod/centraldashboard-cf4874ddc-7hcr8                          1/1
Running   0          97s
pod/jupyter-web-app-deployment-685b455447-gjhh7               1/1
Running   0          96s
pod/katib-controller-88c97d85c-kgq66                          1/1
Running   1          95s
pod/katib-db-8598468fd8-5jw2c                                 1/1
Running   0          95s
pod/katib-manager-574c8c67f9-wtrf5                            1/1
Running   1          95s
pod/katib-manager-rest-778857c989-fjbzn                       1/1
Running   0          95s
pod/katib-suggestion-bayesianoptimization-65df4d7455-qthmw    1/1
Running   0          94s
pod/katib-suggestion-grid-56bf69f597-98vwn                    1/1
Running   0          94s
pod/katib-suggestion-hyperband-7777b76cb9-9v6dq               1/1
Running   0          93s
```

```
pod/katib-suggestion-nasrl-77f6f9458c-2qzxq                    1/1
Running   0          93s
pod/katib-suggestion-random-77b88b5c79-l64j9                   1/1
Running   0          93s
pod/katib-ui-7587c5b967-nd629                                  1/1
Running   0          95s
pod/metacontroller-0                                           1/1
Running   0          96s
pod/metadata-db-5dd459cc-swzkm                                 1/1
Running   0          94s
pod/metadata-deployment-6cf77db994-69fk7                       1/1
Running   3          93s
pod/metadata-deployment-6cf77db994-mpbjt                       1/1
Running   3          93s
pod/metadata-deployment-6cf77db994-xg7tz                       1/1
Running   3          94s
pod/metadata-ui-78f5b59b56-qb6kr                               1/1
Running   0          94s
pod/minio-758b769d67-llvdr                                     1/1
Running   0          91s
pod/ml-pipeline-5875b9db95-g8t2k                               1/1
Running   0          91s
pod/ml-pipeline-persistenceagent-9b69ddd46-bt9r9               1/1
Running   0          90s
pod/ml-pipeline-scheduledworkflow-7b8d756c76-7x56s             1/1
Running   0          90s
pod/ml-pipeline-ui-79ffd9c76-fcwpd                             1/1
Running   0          90s
pod/ml-pipeline-viewer-controller-deployment-5fdc87f58-b2t9r   1/1
Running   0          90s
pod/mysql-657f87857d-l5k9z                                     1/1
Running   0          91s
pod/notebook-controller-deployment-56b4f59bbf-8bvnr            1/1
Running   0          92s
pod/profiles-deployment-6bc745947-mrdkh                        2/2
Running   0          90s
pod/pytorch-operator-77c97f4879-hmlrv                          1/1
Running   0          92s
pod/seldon-operator-controller-manager-0                       1/1
Running   1          91s
pod/spartakus-volunteer-5fdfddb779-l7qkm                       1/1
Running   0          92s
pod/tensorboard-6544748d94-nh8b2                               1/1
Running   0          92s
pod/tf-job-dashboard-56f79c59dd-6w59t                          1/1
Running   0          92s
```

```
pod/tf-job-operator-79cbfd6dbc-rb58c                          1/1
Running   0          91s
pod/workflow-controller-db644d554-cwrnb                       1/1
Running   0          97s
NAME                                                     TYPE
CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
service/admission-webhook-service                        ClusterIP
10.233.51.169   <none>        443/TCP             97s
service/application-controller-service                   ClusterIP
10.233.4.54     <none>        443/TCP             98s
service/argo-ui                                          NodePort
10.233.47.191   <none>        80:31799/TCP        97s
service/centraldashboard                                 ClusterIP
10.233.8.36     <none>        80/TCP              97s
service/jupyter-web-app-service                          ClusterIP
10.233.1.42     <none>        80/TCP              97s
service/katib-controller                                 ClusterIP
10.233.25.226   <none>        443/TCP             96s
service/katib-db                                         ClusterIP
10.233.33.151   <none>        3306/TCP            97s
service/katib-manager                                    ClusterIP
10.233.46.239   <none>        6789/TCP            96s
service/katib-manager-rest                               ClusterIP
10.233.55.32    <none>        80/TCP              96s
service/katib-suggestion-bayesianoptimization            ClusterIP
10.233.49.191   <none>        6789/TCP            95s
service/katib-suggestion-grid                            ClusterIP
10.233.9.105    <none>        6789/TCP            95s
service/katib-suggestion-hyperband                       ClusterIP
10.233.22.2     <none>        6789/TCP            95s
service/katib-suggestion-nasrl                           ClusterIP
10.233.63.73    <none>        6789/TCP            95s
service/katib-suggestion-random                          ClusterIP
10.233.57.210   <none>        6789/TCP            95s
service/katib-ui                                         ClusterIP
10.233.6.116    <none>        80/TCP              96s
service/metadata-db                                      ClusterIP
10.233.31.2     <none>        3306/TCP            96s
service/metadata-service                                 ClusterIP
10.233.27.104   <none>        8080/TCP            96s
service/metadata-ui                                      ClusterIP
10.233.57.177   <none>        80/TCP              96s
service/minio-service                                    ClusterIP
10.233.44.90    <none>        9000/TCP            94s
service/ml-pipeline                                      ClusterIP
10.233.41.201   <none>        8888/TCP,8887/TCP   94s
```

```
service/ml-pipeline-tensorboard-ui                   ClusterIP
10.233.36.207   <none>        80/TCP              93s
service/ml-pipeline-ui                               ClusterIP
10.233.61.150   <none>        80/TCP              93s
service/mysql                                        ClusterIP
10.233.55.117   <none>        3306/TCP            94s
service/notebook-controller-service                  ClusterIP
10.233.10.166   <none>        443/TCP             95s
service/profiles-kfam                                ClusterIP
10.233.33.79    <none>        8081/TCP            92s
service/pytorch-operator                             ClusterIP
10.233.37.112   <none>        8443/TCP            95s
service/seldon-operator-controller-manager-service   ClusterIP
10.233.30.178   <none>        443/TCP             92s
service/tensorboard                                  ClusterIP
10.233.58.151   <none>        9000/TCP            94s
service/tf-job-dashboard                             ClusterIP
10.233.4.17     <none>        80/TCP              94s
service/tf-job-operator                              ClusterIP
10.233.60.32    <none>        8443/TCP            94s
service/webhook-server-service                       ClusterIP
10.233.32.167   <none>        443/TCP             87s
NAME                                                      READY   UP-
TO-DATE   AVAILABLE   AGE
deployment.apps/admission-webhook-deployment              1/1     1
1           97s
deployment.apps/argo-ui                                   1/1     1
1           97s
deployment.apps/centraldashboard                          1/1     1
1           97s
deployment.apps/jupyter-web-app-deployment                1/1     1
1           97s
deployment.apps/katib-controller                          1/1     1
1           96s
deployment.apps/katib-db                                  1/1     1
1           97s
deployment.apps/katib-manager                             1/1     1
1           96s
deployment.apps/katib-manager-rest                        1/1     1
1           96s
deployment.apps/katib-suggestion-bayesianoptimization     1/1     1
1           95s
deployment.apps/katib-suggestion-grid                     1/1     1
1           95s
deployment.apps/katib-suggestion-hyperband                1/1     1
1           95s
```

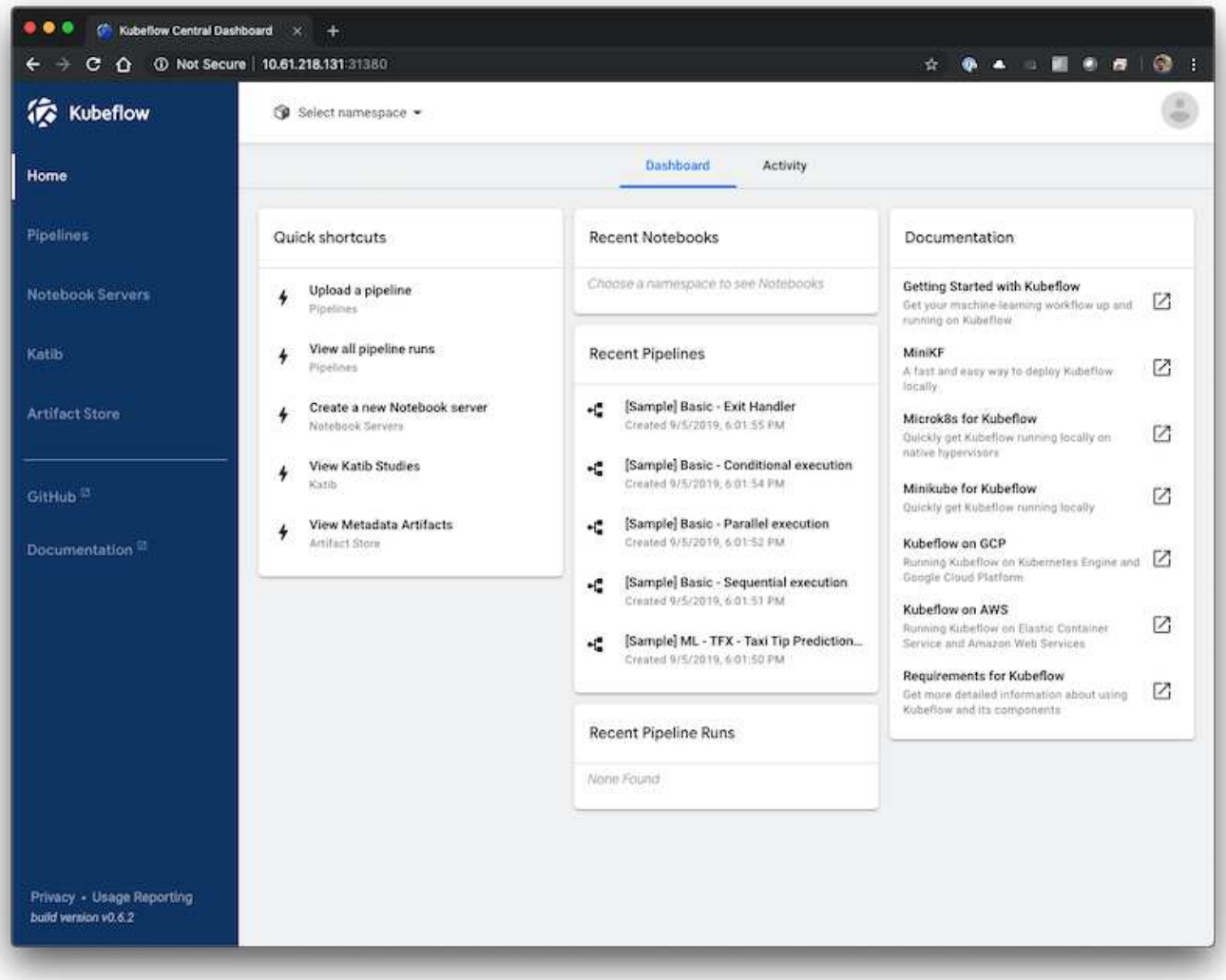
```
deployment.apps/katib-suggestion-nasrl                     1/1     1
1           95s
deployment.apps/katib-suggestion-random                    1/1     1
1           95s
deployment.apps/katib-ui                                   1/1     1
1           96s
deployment.apps/metadata-db                                1/1     1
1           96s
deployment.apps/metadata-deployment                        3/3     3
3           96s
deployment.apps/metadata-ui                                1/1     1
1           96s
deployment.apps/minio                                      1/1     1
1           94s
deployment.apps/ml-pipeline                                1/1     1
1           94s
deployment.apps/ml-pipeline-persistenceagent               1/1     1
1           93s
deployment.apps/ml-pipeline-scheduledworkflow              1/1     1
1           93s
deployment.apps/ml-pipeline-ui                             1/1     1
1           93s
deployment.apps/ml-pipeline-viewer-controller-deployment   1/1     1
1           93s
deployment.apps/mysql                                      1/1     1
1           94s
deployment.apps/notebook-controller-deployment             1/1     1
1           95s
deployment.apps/profiles-deployment                        1/1     1
1           92s
deployment.apps/pytorch-operator                           1/1     1
1           95s
deployment.apps/spartakus-volunteer                        1/1     1
1           94s
deployment.apps/tensorboard                                1/1     1
1           94s
deployment.apps/tf-job-dashboard                           1/1     1
1           94s
deployment.apps/tf-job-operator                            1/1     1
1           94s
deployment.apps/workflow-controller                        1/1     1
1           97s
NAME
DESIRED   CURRENT   READY   AGE
replicaset.apps/admission-webhook-deployment-6b89c84c98              1
1         1       97s
```

```
replicaset.apps/argo-ui-5dcf5d8b4f                                  1
1         1       97s
replicaset.apps/centraldashboard-cf4874ddc                          1
1         1       97s
replicaset.apps/jupyter-web-app-deployment-685b455447               1
1         1       97s
replicaset.apps/katib-controller-88c97d85c                          1
1         1       96s
replicaset.apps/katib-db-8598468fd8                                 1
1         1       97s
replicaset.apps/katib-manager-574c8c67f9                            1
1         1       96s
replicaset.apps/katib-manager-rest-778857c989                       1
1         1       96s
replicaset.apps/katib-suggestion-bayesianoptimization-65df4d7455    1
1         1       95s
replicaset.apps/katib-suggestion-grid-56bf69f597                    1
1         1       95s
replicaset.apps/katib-suggestion-hyperband-7777b76cb9               1
1         1       95s
replicaset.apps/katib-suggestion-nasrl-77f6f9458c                   1
1         1       95s
replicaset.apps/katib-suggestion-random-77b88b5c79                  1
1         1       95s
replicaset.apps/katib-ui-7587c5b967                                 1
1         1       96s
replicaset.apps/metadata-db-5dd459cc                                1
1         1       96s
replicaset.apps/metadata-deployment-6cf77db994                      3
3         3       96s
replicaset.apps/metadata-ui-78f5b59b56                              1
1         1       96s
replicaset.apps/minio-758b769d67                                    1
1         1       93s
replicaset.apps/ml-pipeline-5875b9db95                              1
1         1       93s
replicaset.apps/ml-pipeline-persistenceagent-9b69ddd46              1
1         1       92s
replicaset.apps/ml-pipeline-scheduledworkflow-7b8d756c76            1
1         1       91s
replicaset.apps/ml-pipeline-ui-79ffd9c76                            1
1         1       91s
replicaset.apps/ml-pipeline-viewer-controller-deployment-5fdc87f58  1
1         1       91s
replicaset.apps/mysql-657f87857d                                    1
1         1       92s
```

```
replicaset.apps/notebook-controller-deployment-56b4f59bbf            1
1         1       94s
replicaset.apps/profiles-deployment-6bc745947                        1
1         1       91s
replicaset.apps/pytorch-operator-77c97f4879                          1
1         1       94s
replicaset.apps/spartakus-volunteer-5fdfddb779                       1
1         1       94s
replicaset.apps/tensorboard-6544748d94                               1
1         1       93s
replicaset.apps/tf-job-dashboard-56f79c59dd                          1
1         1       93s
replicaset.apps/tf-job-operator-79cbfd6dbc                           1
1         1       93s
replicaset.apps/workflow-controller-db644d554                        1
1         1       97s
NAME                                                        READY   AGE
statefulset.apps/admission-webhook-bootstrap-stateful-set   1/1     97s
statefulset.apps/application-controller-stateful-set        1/1     98s
statefulset.apps/metacontroller                             1/1     98s
statefulset.apps/seldon-operator-controller-manager         1/1     92s
$ kubectl get pvc -n kubeflow
NAME             STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS               AGE
katib-mysql      Bound    pvc-b07f293e-d028-11e9-9b9d-00505681a82d
10Gi       RWO            ontap-ai-flexvols-retain   27m
metadata-mysql   Bound    pvc-b0f3f032-d028-11e9-9b9d-00505681a82d
10Gi       RWO            ontap-ai-flexvols-retain   27m
minio-pv-claim   Bound    pvc-b22727ee-d028-11e9-9b9d-00505681a82d
20Gi       RWO            ontap-ai-flexvols-retain   27m
mysql-pv-claim   Bound    pvc-b2429afd-d028-11e9-9b9d-00505681a82d
20Gi       RWO            ontap-ai-flexvols-retain   27m
```

4. Rufen Sie in Ihrem Webbrowser das zentrale Kubeflow Dashboard auf, indem Sie zur URL navigieren, die Sie in Schritt 2 angegeben haben.

   Der Standardbenutzername lautet `admin@kubeflow.org`, Und das Standardpasswort lautet `12341234`. Um weitere Benutzer zu erstellen, befolgen Sie die Anweisungen im "Offizielle Dokumentation von Kubeflow".



Beispiel Kubeflow Operations und Tasks

Dieser Abschnitt enthält Beispiele für verschiedene Vorgänge und Aufgaben, die Sie mit Kubeflow ausführen möchten.

Beispiel Kubeflow Operations und Tasks

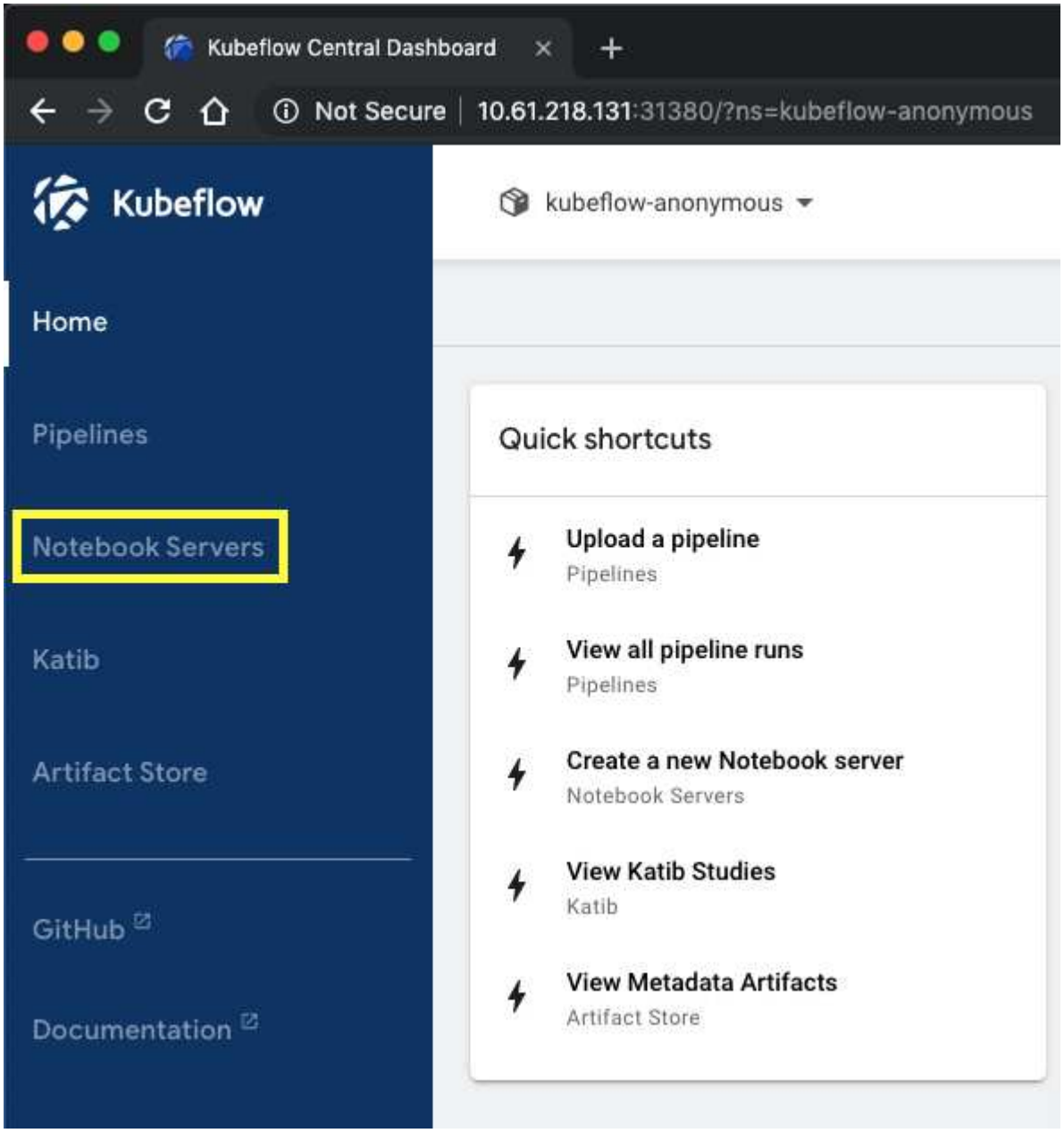
Dieser Abschnitt enthält Beispiele für verschiedene Vorgänge und Aufgaben, die Sie mit Kubeflow ausführen möchten.

Bereitstellen eines Jupyter Notebook Workspace für Data Scientist oder Entwickler

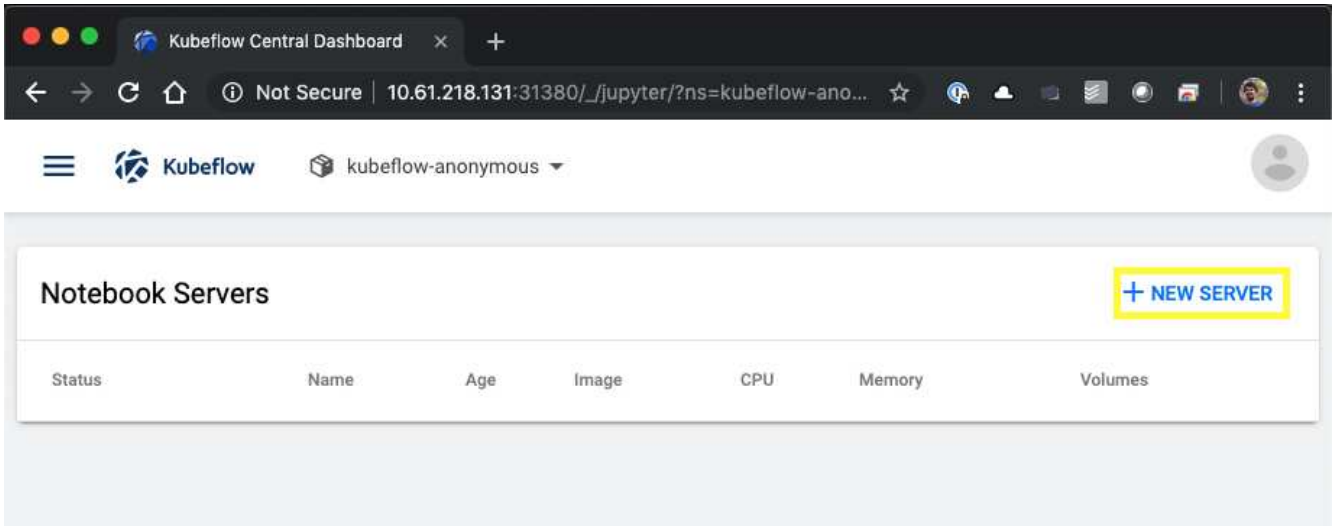
Kubeflow ist in der Lage, neue Jupyter Notebook-Server schnell als Data Scientist-Workspaces bereitzustellen. Um einen neuen Jupyter Notebook-Server mit Kubeflow bereitzustellen, führen Sie die folgenden Aufgaben aus. Weitere Informationen zu Jupyter Notebooks im Kubeflow-Kontext finden Sie im "Offizielle Dokumentation von Kubeflow".

1. Klicken Sie im zentralen Kubeflow Dashboard im Hauptmenü auf Notebook-Server, um zur Jupyter

Notebook-Serververwaltungsseite zu navigieren.

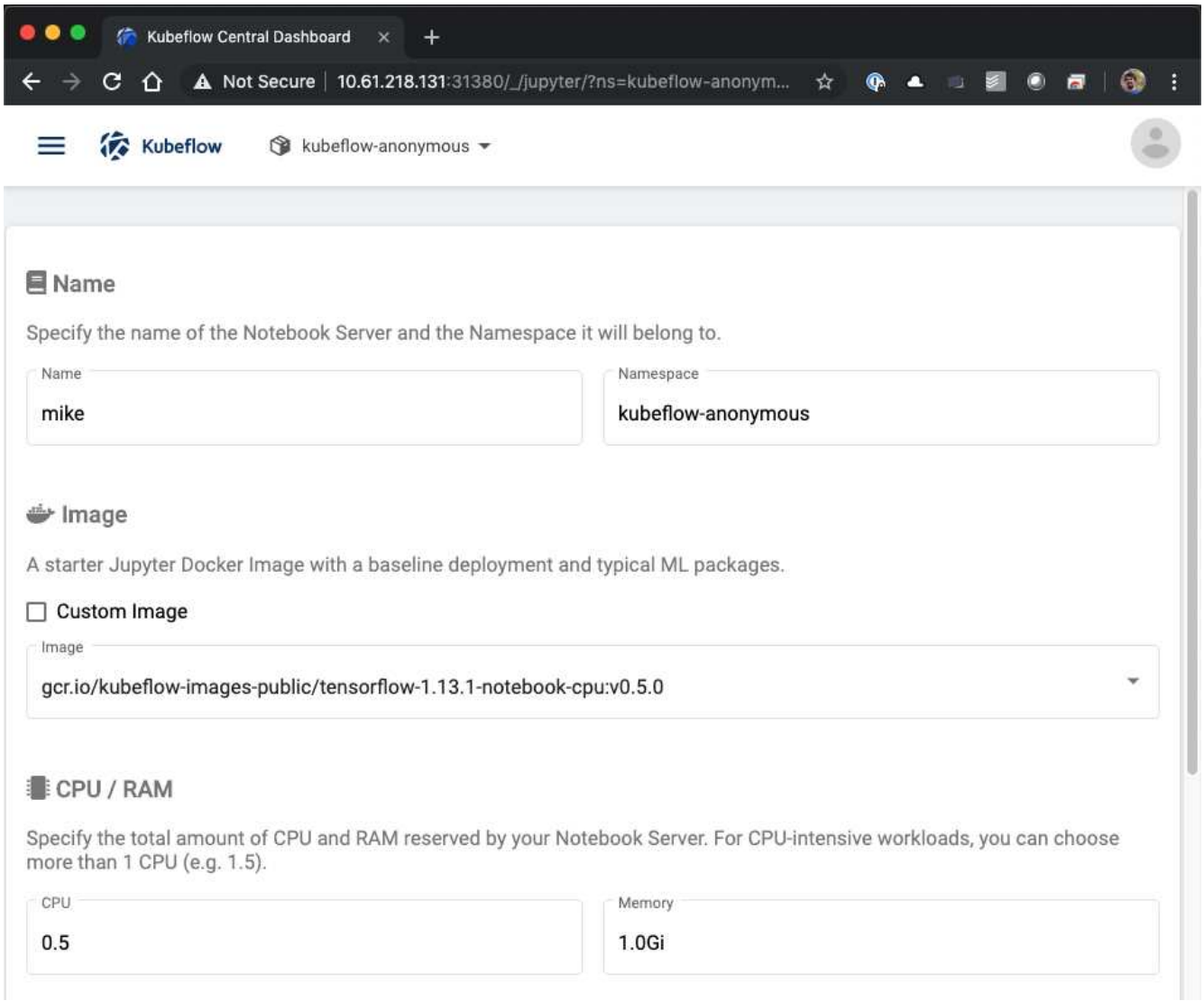


2. Klicken Sie auf Neuer Server, um einen neuen Jupyter Notebook-Server bereitzustellen.

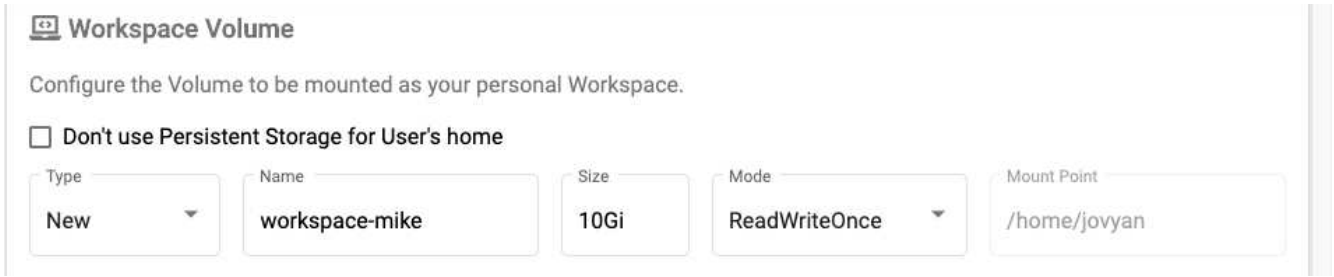


3. Geben Sie Ihrem neuen Server einen Namen, wählen Sie das Docker-Image aus, auf dem Ihr Server basieren soll, und geben Sie die CPU- und RAM-Größe an, die von Ihrem Server reserviert werden soll. Wenn das Namespace-Feld leer ist, wählen Sie über das Menü Namespace auswählen in der Seitenüberschrift einen Namespace aus. Das Namespace-Feld wird dann automatisch mit dem ausgewählten Namespace gefüllt.

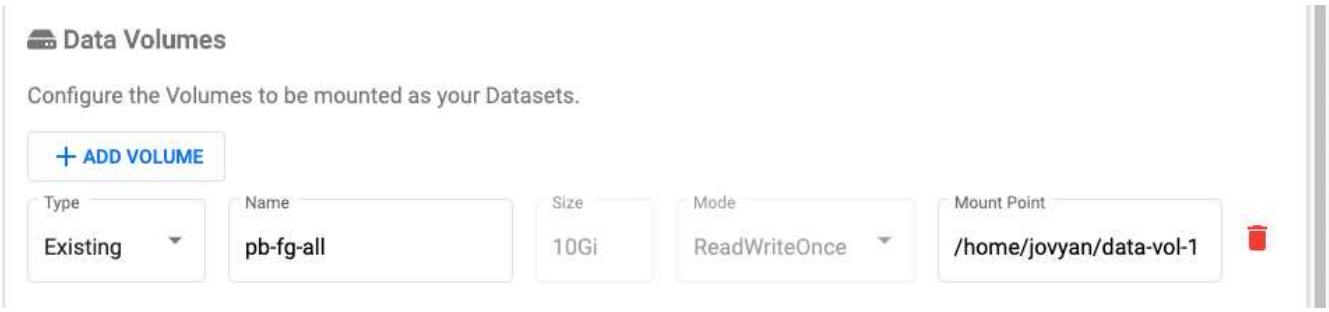
   Im folgenden Beispiel wird der verwendet `kubeflow-anonymous` Namespace ausgewählt. Darüber hinaus werden die Standardwerte für das Docker Image, die CPU und den RAM akzeptiert.



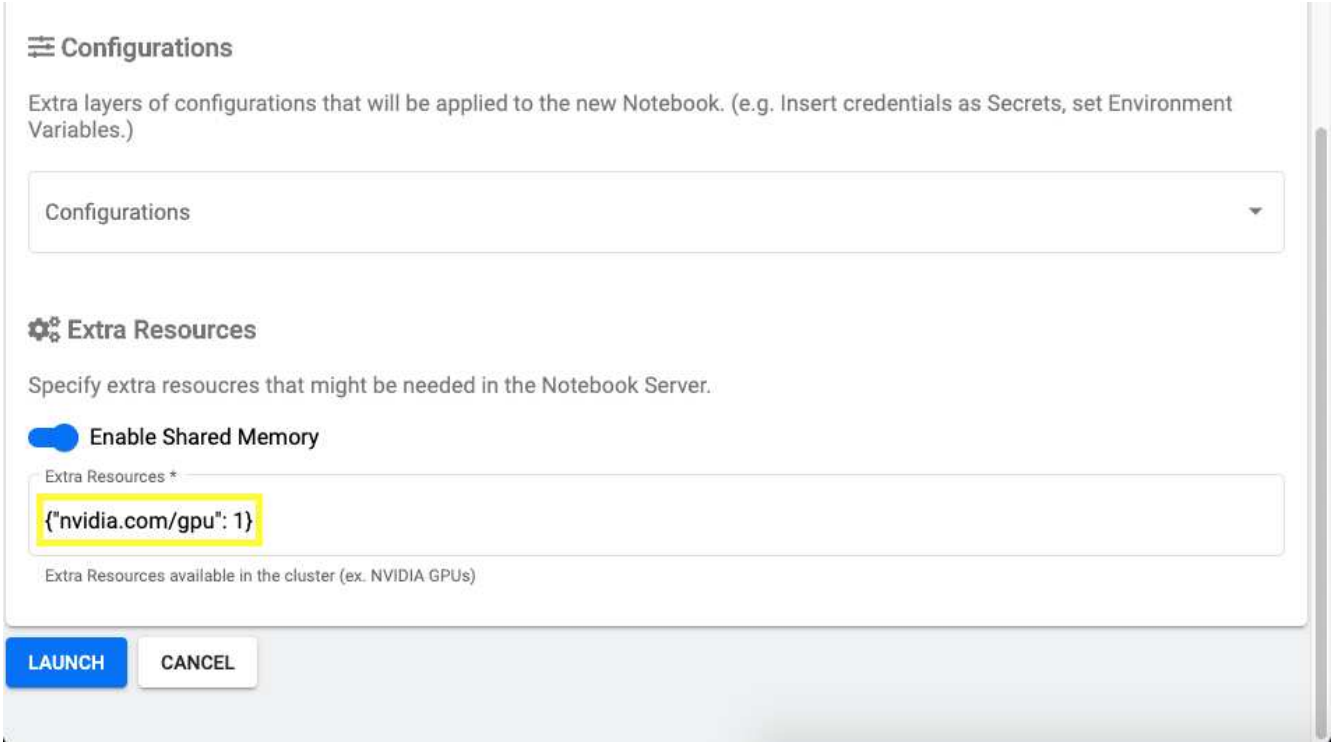
4. Geben Sie die Einzelheiten zum Workspace-Volumen an. Wenn Sie ein neues Volume erstellen möchten, wird dieses Volume oder PVC mithilfe der Standard-StorageClass bereitgestellt. Da ein StorageClass mit Trident als Standard-StorageClass in diesem Abschnitt festgelegt wurde "Kubeflow Deployment", Das Volume oder die PVC wird mit Trident bereitgestellt. Dieses Volume wird automatisch als Standardarbeitsbereich im Jupyter Notebook Server-Container bereitgestellt. Alle Notebooks, die ein Benutzer auf dem Server erstellt, die nicht in einem separaten Datenvolumen gespeichert werden, werden automatisch in diesem Workspace-Volume gespeichert. Daher bleiben die Notebooks auch bei Neustarts erhalten.



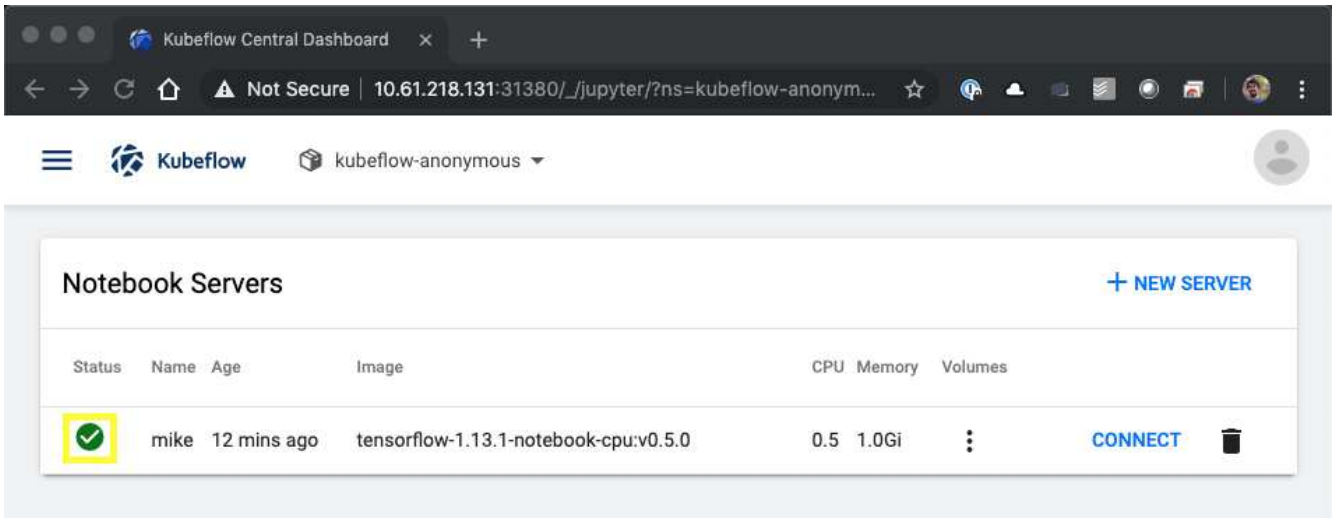
5. Hinzufügen von Daten-Volumes Im folgenden Beispiel wird ein vorhandenes PVC mit dem Namen „pb-fg-all“ angegeben und der Standard-Bereitstellungspunkt akzeptiert.



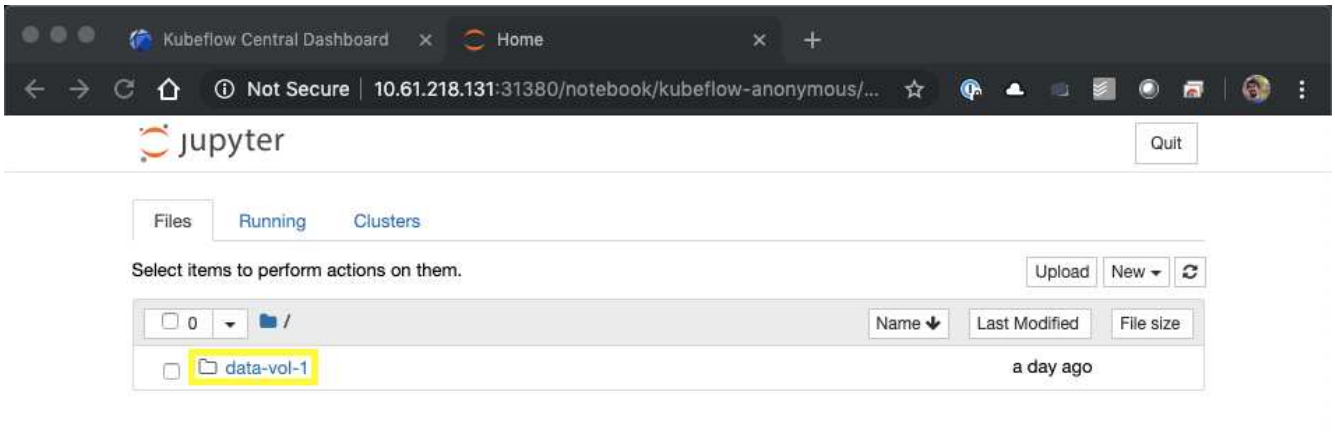
6. **Optional:** Anfrage, dass die gewünschte Anzahl von GPUs Ihrem Notebook-Server zugewiesen wird. Im folgenden Beispiel wird eine GPU angefordert.

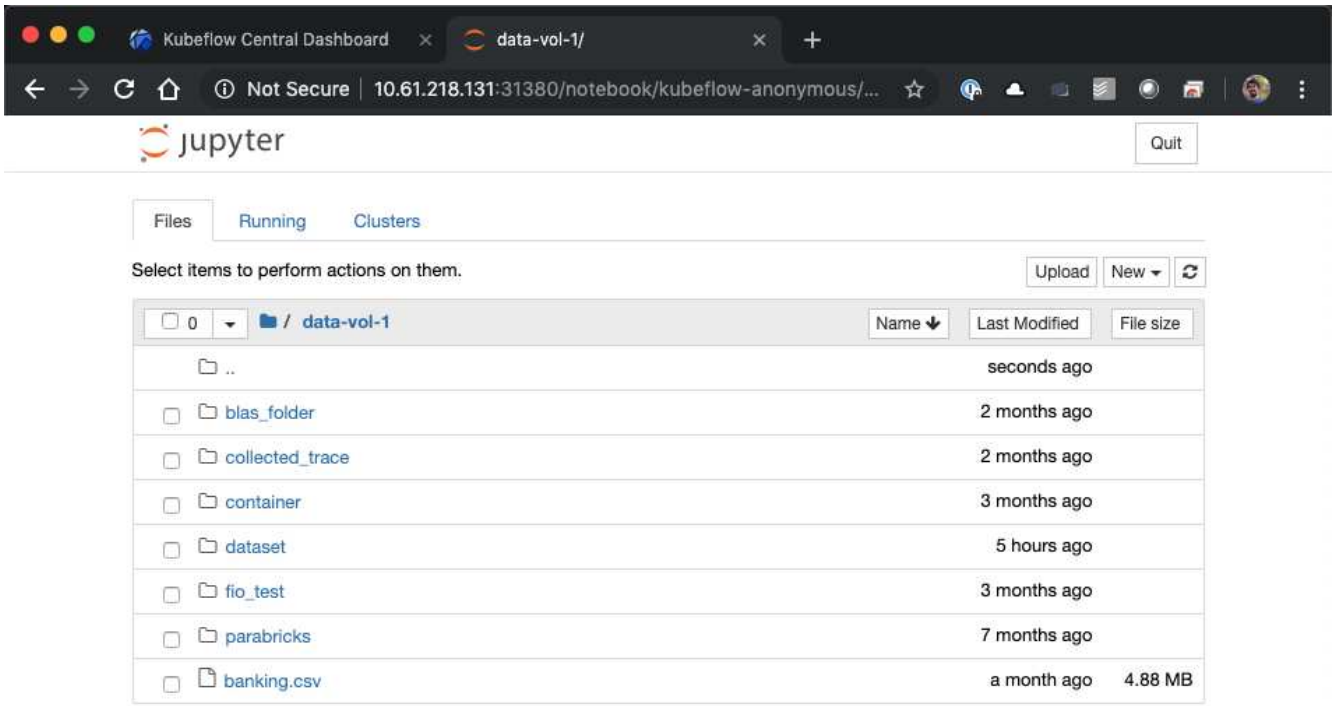


7. Klicken Sie auf Starten, um Ihren neuen Notebook-Server bereitzustellen.
8. Warten Sie, bis Ihr Notebook-Server vollständig bereitgestellt ist. Dies kann einige Minuten dauern, wenn Sie noch keinen Server mit dem von Ihnen angegebenen Docker-Image bereitgestellt haben, da das Image heruntergeladen werden muss. Wenn Ihr Server vollständig bereitgestellt wurde, wird in der Spalte Status auf der Seite Jupyter Notebook Server Administration ein grünes Häkchen angezeigt.



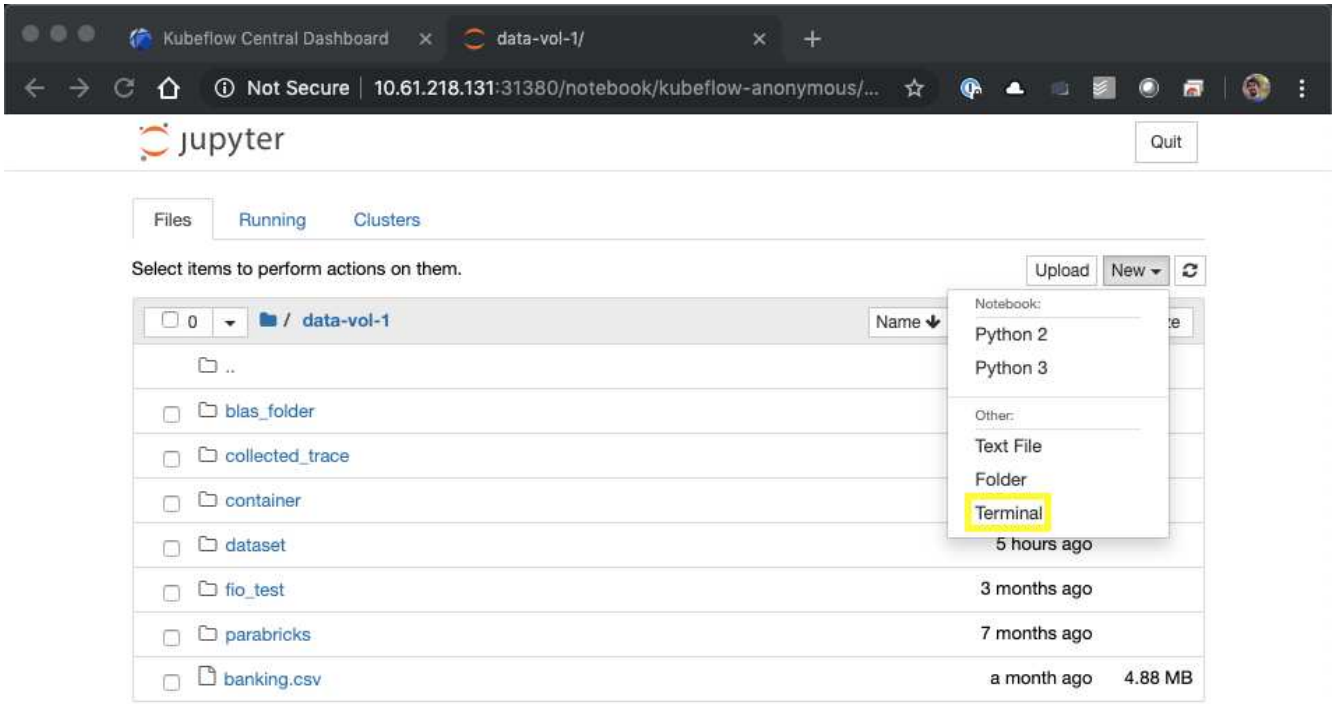
9. Klicken Sie auf Verbinden, um eine Verbindung mit der neuen Server-Webschnittstelle herzustellen.
10. Vergewissern Sie sich, dass das in Schritt 6 angegebene Datensatz-Volume auf dem Server installiert ist. Beachten Sie, dass dieses Volume standardmäßig im Standardarbeitsbereich bereitgestellt wird. Aus der Perspektive des Benutzers ist dies nur ein weiterer Ordner im Arbeitsbereich. Der Anwender, der wahrscheinlich ein Data Scientist ist und nicht ein Infrastruktur-Experte ist, muss nicht über Storage-Fachkenntnisse verfügen, um dieses Volumen zu nutzen.

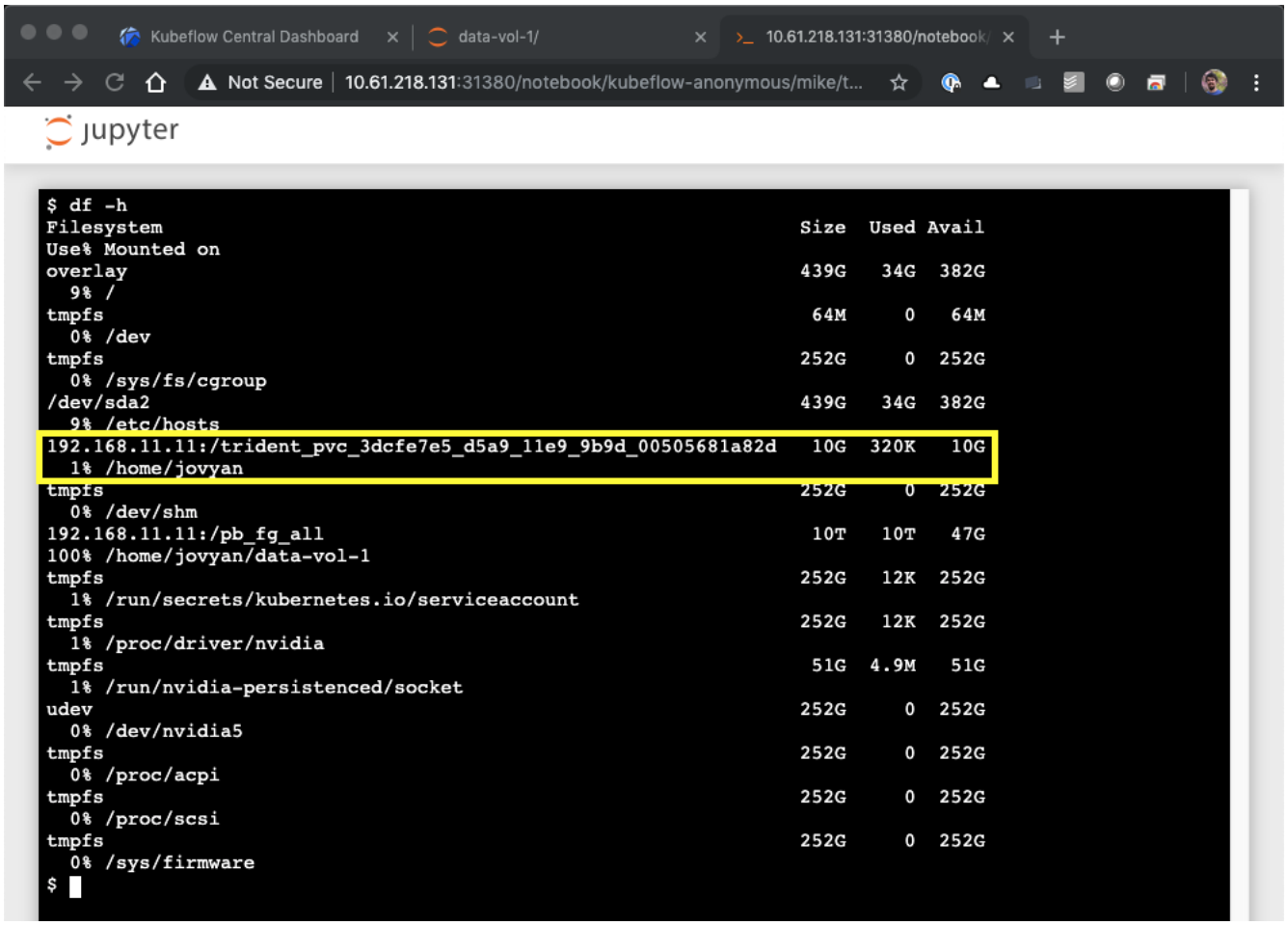




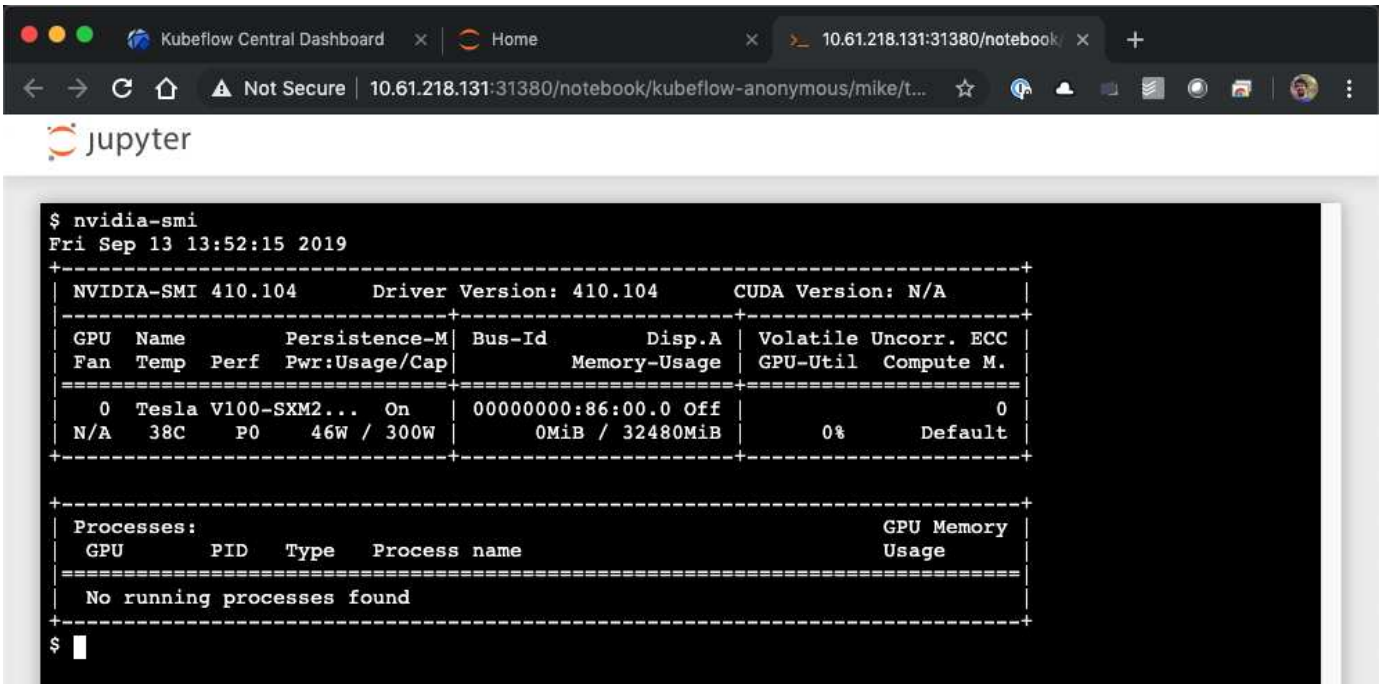
11. Öffnen Sie ein Terminal und, vorausgesetzt, dass ein neues Volume in Schritt 5 angefordert wurde, führen Sie aus `df -h` So bestätigen Sie, dass ein neues persistentes Volume mit Trident-Bereitstellung als Standardarbeitsbereich gemountet wird.

    Das Standard-Workspace-Verzeichnis ist das Basisverzeichnis, mit dem Sie angezeigt werden, wenn Sie zum ersten Mal auf die Web-Schnittstelle des Servers zugreifen. Auf diesem persistenten Volume durch Trident werden daher alle mit der Web-Schnittstelle erstellten Artefakte gespeichert.





12. Führen Sie mit dem Terminal aus `nvidia-smi` Um zu bestätigen, dass dem Notebook-Server die korrekte Anzahl an GPUs zugewiesen wurde. Im folgenden Beispiel wurde dem Notebook-Server eine GPU zugewiesen, wie in Schritt 7 angefordert.



Beispiel für Notebooks und Pipelines

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

- Data Scientists können erweiterte NetApp Datenmanagement-Vorgänge direkt über ein Jupyter Notebook ausführen.
- Erweiterte Datenmanagement-Vorgänge von NetApp können mithilfe des Kubeflow Pipelines in automatisierte Workflows integriert werden.

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

Apache Airflow Deployment

NetApp empfiehlt, Apache Airflow zusätzlich zu Kubernetes auszuführen. In diesem Abschnitt werden die Aufgaben beschrieben, die Sie zur Implementierung von Airflow in Ihrem Kubernetes-Cluster ausführen müssen.

> Es ist möglich, Airflow auf anderen Plattformen als Kubernetes bereitzustellen. Die Implementierung von Airflow auf anderen Plattformen als Kubernetes ist nicht im Umfang dieser Lösung enthalten.

Voraussetzungen

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

1. Sie verfügen bereits über einen funktionierenden Kubernetes-Cluster.
2. Sie haben NetApp Trident bereits in Ihrem Kubernetes Cluster installiert und konfiguriert, wie im Abschnitt „NetApp Trident Deployment and Configuration“ beschrieben.

Installieren Sie Helm

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

Standard-Kubernetes StorageClass festlegen

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

Verwenden Sie Helm zum Bereitstellen des Luftstroms

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

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

```
$ cat << EOF > custom-values.yaml
####################################
# Airflow - Common Configs
####################################
airflow:
  ## the airflow executor type to use
  ##
  executor: "CeleryExecutor"
  ## environment variables for the web/scheduler/worker Pods (for
airflow configs)
  ##
  #
####################################
# Airflow - WebUI Configs
####################################
web:
  ## configs for the Service of the web Pods
  ##
  service:
    type: NodePort
####################################
# Airflow - Logs Configs
####################################
logs:
  persistence:
    enabled: true
####################################
# Airflow - DAGs Configs
####################################
dags:
  ## configs for the DAG git repository & sync container
  ##
  gitSync:
    enabled: true
    ## url of the git repository
    ##
    repo: "git@github.com:mboglesby/airflow-dev.git"
```

```
    ## the branch/tag/sha1 which we clone
    ##
    branch: master
    revision: HEAD
    ## the name of a pre-created secret containing files for ~/.ssh/
    ##
    ## NOTE:
    ## - this is ONLY RELEVANT for SSH git repos
    ## - the secret commonly includes files: id_rsa, id_rsa.pub,
known_hosts
    ## - known_hosts is NOT NEEDED if `git.sshKeyscan` is true
    ##
    sshSecret: "airflow-ssh-git-secret"
    ## the name of the private key file in your `git.secret`
    ##
    ## NOTE:
    ## - this is ONLY RELEVANT for PRIVATE SSH git repos
    ##
    sshSecretKey: id_rsa
    ## the git sync interval in seconds
    ##
    syncWait: 60
EOF
$ helm install airflow airflow-stable/airflow -n airflow --version 8.0.8
--values ./custom-values.yaml
...
Congratulations. You have just deployed Apache Airflow!
1. Get the Airflow Service URL by running these commands:
   export NODE_PORT=$(kubectl get --namespace airflow -o
jsonpath="{.spec.ports[0].nodePort}" services airflow-web)
   export NODE_IP=$(kubectl get nodes --namespace airflow -o
jsonpath="{.items[0].status.addresses[0].address}")
   echo http://$NODE_IP:$NODE_PORT/
2. Open Airflow in your web browser
```

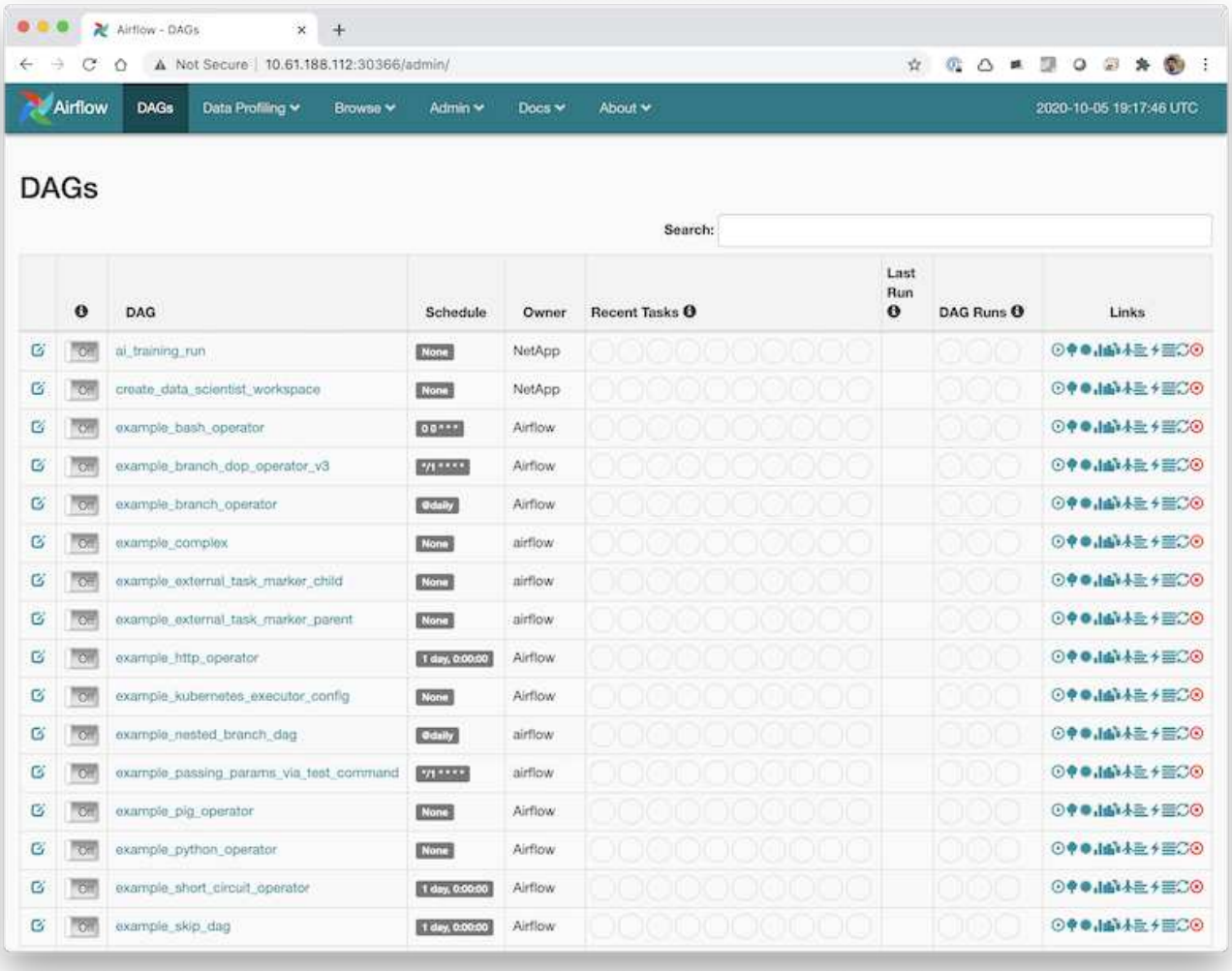
2. Vergewissern Sie sich, dass alle Airflow-Pods betriebsbereit sind. Es kann ein paar Minuten dauern, bis alle Pods beginnen.

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

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

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

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



Beispiel Für Apache Airflow Workflows

Der "Das NetApp Data Science Toolkit für Kubernetes" Kann in Verbindung mit Airflow verwendet werden. Mit dem NetApp Data Science Toolkit in Airflow können Sie NetApp Datenmanagement-Operationen in automatisierte Workflows einbinden, die mit Airflow orchestriert sind.

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

Beispiel Für Trident-Vorgänge

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

Importieren eines vorhandenen Volumes

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

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

Die folgenden Beispielbefehle zeigen den Import des gleichen Volumes mit dem Namen `pb_fg_all`, Zweimal einmal pro Trident Back-End, das im Beispiel im Abschnitt erstellt wurde "Beispiel Trident Back-Ends für ONTAP KI-Implementierungen", Schritt 1. Wenn Sie dasselbe Volume zweimal auf diese Weise importieren, können Sie das Volume (ein vorhandenes FlexGroup Volume) mehrmals zwischen verschiedenen LIFs mounten, wie im Abschnitt beschrieben "Beispiel Trident Back-Ends für ONTAP KI-Implementierungen", Schritt 1. Weitere Informationen zu PVCs finden Sie im "Offizielle Kubernetes-Dokumentation". Weitere Informationen zur Importfunktion des Volumes finden Sie im "Trident Dokumentation".

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

> Die Backend-Namen, die in den folgenden Beispiel-Importbefehlen angegeben sind, entsprechen den Backends, die im Beispiel im Abschnitt erstellt wurden "Beispiel Trident Back-Ends für ONTAP KI-Implementierungen", Schritt 1. Die StorageClass-Namen, die im folgenden Beispiel PVC-Definitionsdateien angegeben sind, entsprechen den StorageClasses, die im Beispiel im Abschnitt erstellt wurden "Beispiel: Kubernetes StorageClasses für ONTAP KI-Implementierungen", Schritt 1.

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

```
+-------------------------------------------+--------+---------+
$ cat << EOF > ./pvc-import-pb_fg_all-iface2.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pb-fg-all-iface2
  namespace: default
spec:
  accessModes:
    - ReadOnlyMany
  storageClassName: ontap-ai-flexgroups-retain-iface2
EOF
$ tridentctl import volume ontap-ai-flexgroups-iface2 pb_fg_all -f ./pvc-
import-pb_fg_all-iface2.yaml -n trident
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
|          NAME                  |  SIZE  |       STORAGE CLASS
| PROTOCOL |             BACKEND UUID                        | STATE  |
MANAGED |
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
| default-pb-fg-all-iface2-85aee | 10 TiB | ontap-ai-flexgroups-retain-
iface2 | file     | 61814d48-c770-436b-9cb4-cf7ee661274d | online | true
|
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
$ tridentctl get volume -n trident
+-----------------------------------+---------
+-----------------------------------+----------
+---------------------------------------+--------+---------+
|               NAME                |  SIZE   |           STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------+---------
+-----------------------------------+----------
+---------------------------------------+--------+---------+
| default-pb-fg-all-iface1-7d9f1   | 10 TiB  | ontap-ai-flexgroups-retain-
iface1 | file     | b74cbddb-e0b8-40b7-b263-b6da6dec0bdd | online | true
|
| default-pb-fg-all-iface2-85aee   | 10 TiB  | ontap-ai-flexgroups-retain-
iface2 | file     | 61814d48-c770-436b-9cb4-cf7ee661274d | online | true
|
+-----------------------------------+---------
+-----------------------------------+----------
```

```
+-------------------------------------+--------+---------+
$ kubectl get pvc
NAME                  STATUS   VOLUME                                   CAPACITY
ACCESS MODES   STORAGECLASS                        AGE
pb-fg-all-iface1      Bound    default-pb-fg-all-iface1-7d9f1
10995116277760   ROX            ontap-ai-flexgroups-retain-iface1   25h
pb-fg-all-iface2      Bound    default-pb-fg-all-iface2-85aee
10995116277760   ROX            ontap-ai-flexgroups-retain-iface2   25h
```

Bereitstellung eines neuen Volumes

Mit Trident können Sie ein neues Volume auf Ihrem NetApp Storage-System oder Ihrer NetApp Plattform bereitstellen. Mit den folgenden Beispielbefehlen wird die Bereitstellung eines neuen FlexVol Volumes angezeigt. In diesem Beispiel wird das Volume mit der StorageClass bereitgestellt, die im Beispiel im Abschnitt erstellt wurde "Beispiel: Kubernetes StorageClasses für ONTAP KI-Implementierungen", Schritt 2.

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

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

Beispiele für hochperformante Jobs für ONTAP KI-Implementierungen

Dieser Abschnitt enthält Beispiele für verschiedene hochperformante Jobs, die ausgeführt werden können, wenn Kubernetes auf einem ONTAP AI POD implementiert wird.

Beispiele für hochperformante Jobs für ONTAP KI-Implementierungen

Dieser Abschnitt enthält Beispiele für verschiedene hochperformante Jobs, die ausgeführt werden können, wenn Kubernetes auf einem ONTAP AI POD implementiert wird.

Single-Node-KI-Workload ausführen

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

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

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

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

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

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

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

```
$ cat << EOF > ./netapp-tensorflow-single-imagenet.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-single-imagenet
spec:
  backoffLimit: 5
  template:
    spec:
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--dgx_version=dgx1", "--
num_devices=8"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
```

```
          name: results
        securityContext:
          privileged: true
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-single-imagenet.yaml
job.batch/netapp-tensorflow-single-imagenet created
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-single-imagenet           0/1           24s        24s
```

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

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

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

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

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

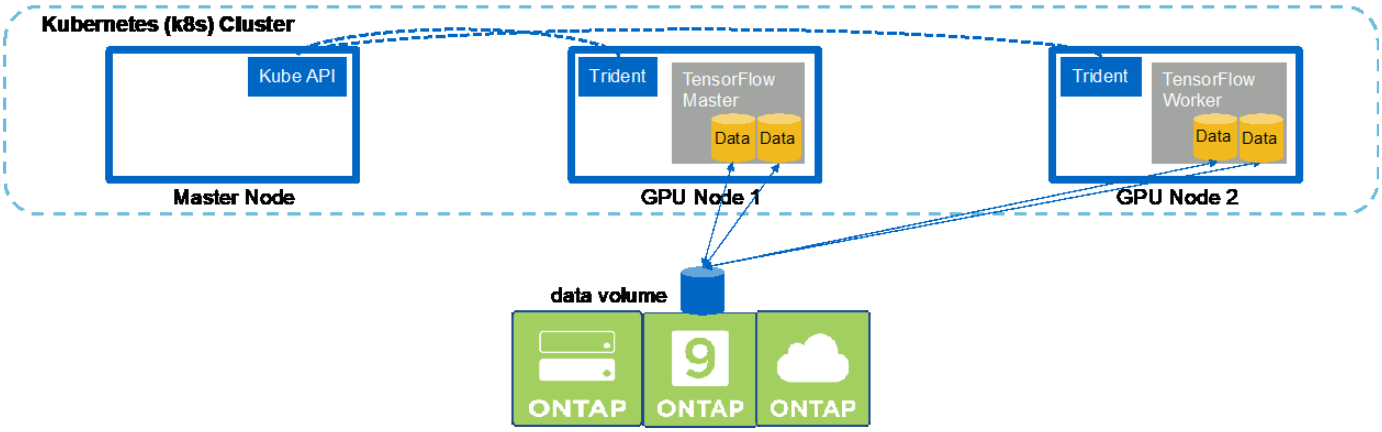
   Wenn Sie das Jobobjekt löschen, löscht Kubernetes automatisch alle zugehörigen Pods.

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

Ausführung eines synchronen, verteilten KI-Workloads

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

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



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

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

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

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

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-worker.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: netapp-tensorflow-multi-imagenet-worker
spec:
  replicas: 1
  selector:
    matchLabels:
      app: netapp-tensorflow-multi-imagenet-worker
  template:
    metadata:
      labels:
        app: netapp-tensorflow-multi-imagenet-worker
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
```

```
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["bash", "/netapp/scripts/start-slave-multi.sh",
"22122"]
        resources:
          limits:
            nvidia.com/gpu: 8
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-worker.yaml
deployment.apps/netapp-tensorflow-multi-imagenet-worker created
$ kubectl get deployments
NAME                                      DESIRED   CURRENT   UP-TO-DATE
AVAILABLE   AGE
netapp-tensorflow-multi-imagenet-worker   1         1         1
1           4s
```

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

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

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

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

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

```
$ cat << EOF > ./netapp-tensorflow-multi-imagenet-master.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-tensorflow-multi-imagenet-master
spec:
  backoffLimit: 5
  template:
    spec:
      hostNetwork: true
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
      - name: testdata-iface1
        persistentVolumeClaim:
          claimName: pb-fg-all-iface1
      - name: testdata-iface2
        persistentVolumeClaim:
          claimName: pb-fg-all-iface2
      - name: results
        persistentVolumeClaim:
          claimName: tensorflow-results
      containers:
      - name: netapp-tensorflow-py2
        image: netapp/tensorflow-py2:19.03.0
        command: ["python", "/netapp/scripts/run.py", "--
dataset_dir=/mnt/mount_0/dataset/imagenet", "--port=22122", "--
num_devices=16", "--dgx_version=dgx1", "--
nodes=10.61.218.152,10.61.218.154"]
        resources:
          limits:
            nvidia.com/gpu: 8
```

```
        volumeMounts:
        - mountPath: /dev/shm
          name: dshm
        - mountPath: /mnt/mount_0
          name: testdata-iface1
        - mountPath: /mnt/mount_1
          name: testdata-iface2
        - mountPath: /tmp
          name: results
        securityContext:
          privileged: true
      restartPolicy: Never
EOF
$ kubectl create -f ./netapp-tensorflow-multi-imagenet-master.yaml
job.batch/netapp-tensorflow-multi-imagenet-master created
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   0/1           25s        25s
```

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

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

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

```
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      9m18s
$ kubectl get pods
NAME                                                          READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj               0/1
Completed   0          9m38s
```

```
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          35m
$ kubectl logs netapp-tensorflow-multi-imagenet-master-ppwwj
[10.61.218.152:00008] WARNING: local probe returned unhandled
shell:unknown assuming bash
rm: cannot remove '/lib': Is a directory
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
Total images/sec = 12881.33875
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 2 -H 10.61.218.152:1,10.61.218.154:1 -mca
pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" bash -c 'sync; echo 1 >
/proc/sys/vm/drop_caches'
==========================================
mpirun -allow-run-as-root -np 16 -H 10.61.218.152:8,10.61.218.154:8
-bind-to none -map-by slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH
-mca pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -x
NCCL_IB_HCA=mlx5 -x NCCL_NET_GDR_READ=1 -x NCCL_IB_SL=3 -x
NCCL_IB_GID_INDEX=3 -x
NCCL_SOCKET_IFNAME=enp5s0.3091,enp12s0.3092,enp132s0.3093,enp139s0.3094
-x NCCL_IB_CUDA_SUPPORT=1 -mca orte_base_help_aggregate 0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000 --
datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_161609_tensorflow_horovod_rdma_resnet50_gpu_16_256_b500_im
agenet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

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

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

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

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

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

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

Performance-Tests

Bei der Erstellung dieser Lösung haben wir einen einfachen Performance-Vergleich durchgeführt. Wir haben mithilfe von Kubernetes mehrere standardmäßige NetApp KI-Benchmark-Jobs ausgeführt. Die Benchmark-Ergebnisse wurden mit Ausführungsvorgängen verglichen, die über einen einfachen Docker Run-Befehl ausgeführt wurden. Wir haben keine spürbaren Unterschiede in der Performance gesehen. Daher kamen wir zu dem Schluss, dass die Verwendung von Kubernetes zur Orchestrierung von Training-Jobs in Container-KI keine negative Auswirkung auf die Performance hat. Die Ergebnisse unseres Performance-Vergleichs finden Sie in der folgenden Tabelle.

Benchmark	Datensatz	Docker Run (Images/Sek.)	Kubernetes (Bilder/Sek.)
TensorFlow mit einem Node	Synthetische Daten	6,667.2475	6,661.93125
TensorFlow mit einem Node	ImageNet	6,570.2025	6,530.59125
Synchrones verteiltes TensorFlow mit zwei Knoten	Synthetische Daten	13,213.70625	13,218.288125
Synchrones verteiltes TensorFlow mit zwei Knoten	ImageNet	12,941.69125	12,881.33875

Schlussfolgerung

Unternehmen und Unternehmen aller Größen und Branchen setzen zunehmend auf künstliche Intelligenz (KI), Machine Learning (ML) und Deep Learning (DL), um Probleme aus der Praxis zu lösen, innovative Produkte und Services zu entwickeln und sich in einem immer härter umkämpften Markt einen Schritt voraus zu sein. Beim verstärkten Einsatz von KI, ML und DL müssen Unternehmen mit vielen Herausforderungen konfrontiert werden, darunter Workload-Skalierbarkeit und Datenverfügbarkeit. Diese Herausforderungen können durch den Einsatz der NetApp AI Control Plane Lösung bewältigt werden.

Mit dieser Lösung können Sie einen Daten-Namespace schnell klonen. Darüber hinaus können Kunden AI-, ML- und DL-Training-Workflows definieren und implementieren, die die nahezu sofortige Erstellung von Daten und Modellbasiden für Rückverfolgbarkeit und Versionierung enthalten. Mit dieser Lösung können Sie hinter jedem einzelnen Modell-Training zurück zu den genauen Datensätzen führen, mit denen das Modell geschult und/oder validiert wurde. Und schließlich können Sie mit dieser Lösung schnell Jupyter Notebook-Workspaces mit Zugriff auf riesige Datensätze bereitstellen.

Da diese Lösung auf Data Scientists und Data Engineers ausgerichtet ist, sind minimale Fachkenntnisse von NetApp oder NetApp ONTAP erforderlich. Bei dieser Lösung lassen sich Datenmanagementfunktionen mit einfachen und vertrauten Tools und Schnittstellen ausführen. Darüber hinaus nutzt diese Lösung vollständig Open-Source- und freie Komponenten. Wenn Sie also bereits NetApp Storage in Ihrer Umgebung haben,

können Sie diese Lösung heute implementieren. Wenn Sie diese Lösung testen möchten, aber noch nicht bereits über NetApp Storage verfügen, besuchen Sie "cloud.netapp.com", Und Sie sind innerhalb kürzester Zeit mit einer Cloud-basierten NetApp Storage-Lösung einsatzbereit.

Copyright-Informationen

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

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

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

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

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

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

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

Markeninformationen

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