Piano di controllo ai di NetApp

NetApp Solutions

NetApp
April 26, 2024

Sommario

Piano di controllo ai di NetApp

TR-4798: Piano di controllo ai di NetApp

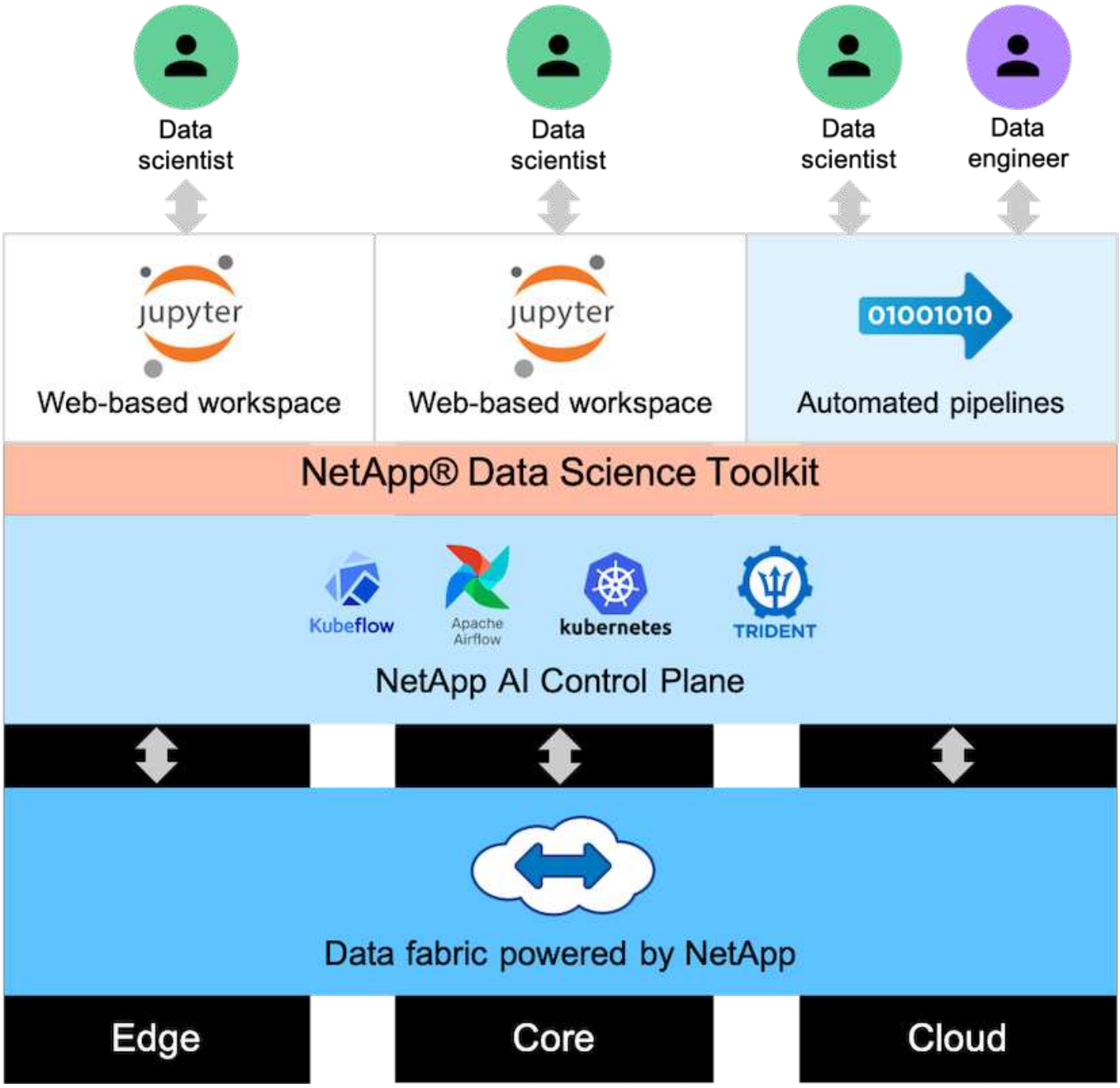
Mike Oglesby, NetApp

Aziende e organizzazioni di ogni dimensione e in molti settori stanno passando all'intelligenza artificiale (ai), all'apprendimento automatico (ML) e al deep learning (DL) per risolvere problemi reali, offrire prodotti e servizi innovativi e ottenere un vantaggio in un mercato sempre più competitivo. Man mano che le organizzazioni aumentano l'utilizzo di ai, ML e DL, devono affrontare molte sfide, tra cui la scalabilità dei workload e la disponibilità dei dati. Questo documento dimostra come affrontare queste sfide utilizzando il NetApp ai Control Plane, una soluzione che unisce le funzionalità di gestione dei dati di NetApp con i più diffusi framework e tool open-source.

Questo report mostra come clonare rapidamente uno spazio dei nomi dei dati. Mostra inoltre come replicare perfettamente i dati tra siti e regioni per creare una pipeline di dati ai/ML/DL coesa e unificata. Inoltre, ti guida attraverso la definizione e l'implementazione di workflow di training ai, ML e DL che incorporano la creazione quasi istantanea di dati e linee di base dei modelli per la tracciabilità e il controllo delle versioni. Con questa soluzione, è possibile tracciare ogni ciclo di training del modello fino all'esatto set di dati utilizzato per la formazione e/o la convalida del modello. Infine, questo documento illustra come eseguire rapidamente il provisioning delle aree di lavoro dei notebook Jupyter con accesso a set di dati di grandi dimensioni.

Nota: Per i training distribuiti in stile HPC su larga scala che coinvolgono un gran numero di server GPU che richiedono l'accesso condiviso allo stesso set di dati, o se si desidera un file system parallelo, consultare la sezione "TR-4890". Questo report tecnico descrive come includere "La soluzione di file system parallelo completamente supportata di NetApp BeeGFS" Come parte del NetApp ai Control Plane. Questa soluzione è progettata per scalare da una manciata di sistemi NVIDIA DGX A100 fino a un SuperPOD a 140 nodi completo.

Il piano di controllo ai di NetApp è rivolto a data scientist e data engineer e, di conseguenza, è necessaria una competenza minima di NetApp o NetApp ONTAP®. Con questa soluzione, le funzioni di gestione dei dati possono essere eseguite utilizzando interfacce e strumenti semplici e familiari. Se disponete già di storage NetApp nel vostro ambiente, potete testare il NetApp ai Control Plane oggi stesso. Se si desidera provare la soluzione ma non si dispone già di storage NetApp, visitare il sito "cloud.netapp.com"E potrai essere operativo con una soluzione di storage NetApp basata sul cloud in pochi minuti. La figura seguente fornisce una visualizzazione della soluzione.



Concetti e componenti

Intelligenza artificiale

L'ai è una disciplina informatica in cui i computer sono formati per imitare le funzioni cognitive della mente umana. Gli sviluppatori di ai addestrano i computer per imparare e risolvere i problemi in modo simile o addirittura superiore agli esseri umani. Il deep learning e l'apprendimento automatico sono sottocampi dell'ai. Le organizzazioni stanno adottando sempre più ai, ML e DL per supportare le loro esigenze aziendali critiche. Di seguito sono riportati alcuni esempi:

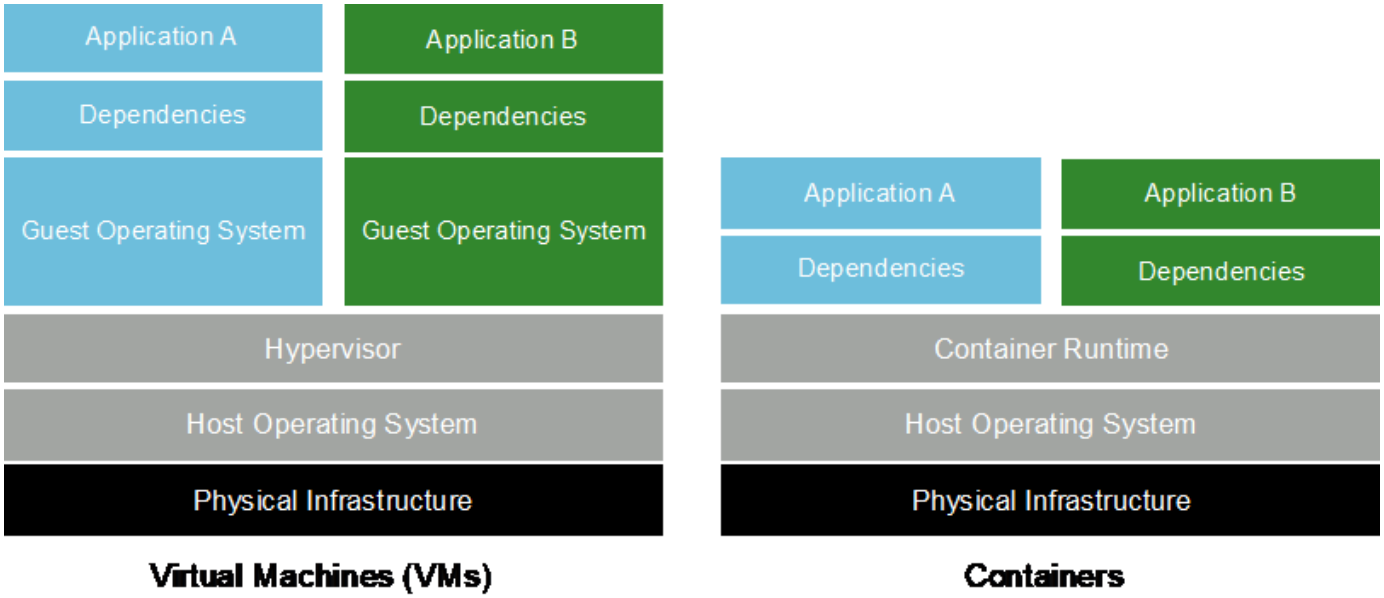
- Analisi di grandi quantità di dati per scoprire informazioni di business precedentemente sconosciute
- Interagire direttamente con i clienti utilizzando l'elaborazione del linguaggio naturale
- Automazione di vari processi e funzioni di business

I moderni carichi di lavoro di training e inferenza ai richiedono funzionalità di calcolo estremamente parallele. Pertanto, le GPU vengono sempre più utilizzate per eseguire le operazioni ai perché le funzionalità di elaborazione parallela delle GPU sono notevolmente superiori a quelle delle CPU generiche.

Container

I container sono istanze isolate dello spazio utente eseguite su un kernel del sistema operativo host condiviso. L'adozione dei container è in rapida crescita. I container offrono molti degli stessi vantaggi offerti dalle macchine virtuali (VM) per il sandboxing delle applicazioni. Tuttavia, poiché l'hypervisor e i livelli del sistema operativo guest su cui si basano le macchine virtuali sono stati eliminati, i container sono molto più leggeri. La figura seguente mostra una visualizzazione delle macchine virtuali rispetto ai container.

I container consentono inoltre un efficiente packaging delle dipendenze delle applicazioni, dei tempi di esecuzione e così via, direttamente con un'applicazione. Il formato di packaging dei container più comunemente utilizzato è Docker Container. Un'applicazione che è stata containerizzata nel formato Docker container può essere eseguita su qualsiasi computer in grado di eseguire i container Docker. Ciò è vero anche se le dipendenze dell'applicazione non sono presenti sul computer perché tutte le dipendenze sono contenute nel container stesso. Per ulteriori informazioni, visitare il "Sito web di Docker".



Kubernetes

Kubernetes è una piattaforma open source, distribuita e di orchestrazione dei container, originariamente progettata da Google e ora gestita dalla Cloud Native Computing Foundation (CNCF). Kubernetes consente l'automazione delle funzioni di implementazione, gestione e scalabilità per le applicazioni containerizzate. Negli ultimi anni, Kubernetes è emersa come piattaforma dominante per l'orchestrazione di container. Sebbene siano supportati altri formati di packaging dei container e tempi di esecuzione, Kubernetes viene spesso utilizzato come sistema di orchestrazione per i container Docker. Per ulteriori informazioni, visitare il "Sito web di Kubernetes".

Trident di NetApp

Trident è un orchestratore di storage open source sviluppato e gestito da NetApp che semplifica notevolmente la creazione, la gestione e il consumo dello storage persistente per i carichi di lavoro Kubernetes. Trident, un'applicazione nativa di Kubernetes, viene eseguita direttamente all'interno di un cluster Kubernetes. Con Trident, gli utenti di Kubernetes (sviluppatori, data scientist, amministratori di Kubernetes e così via) possono

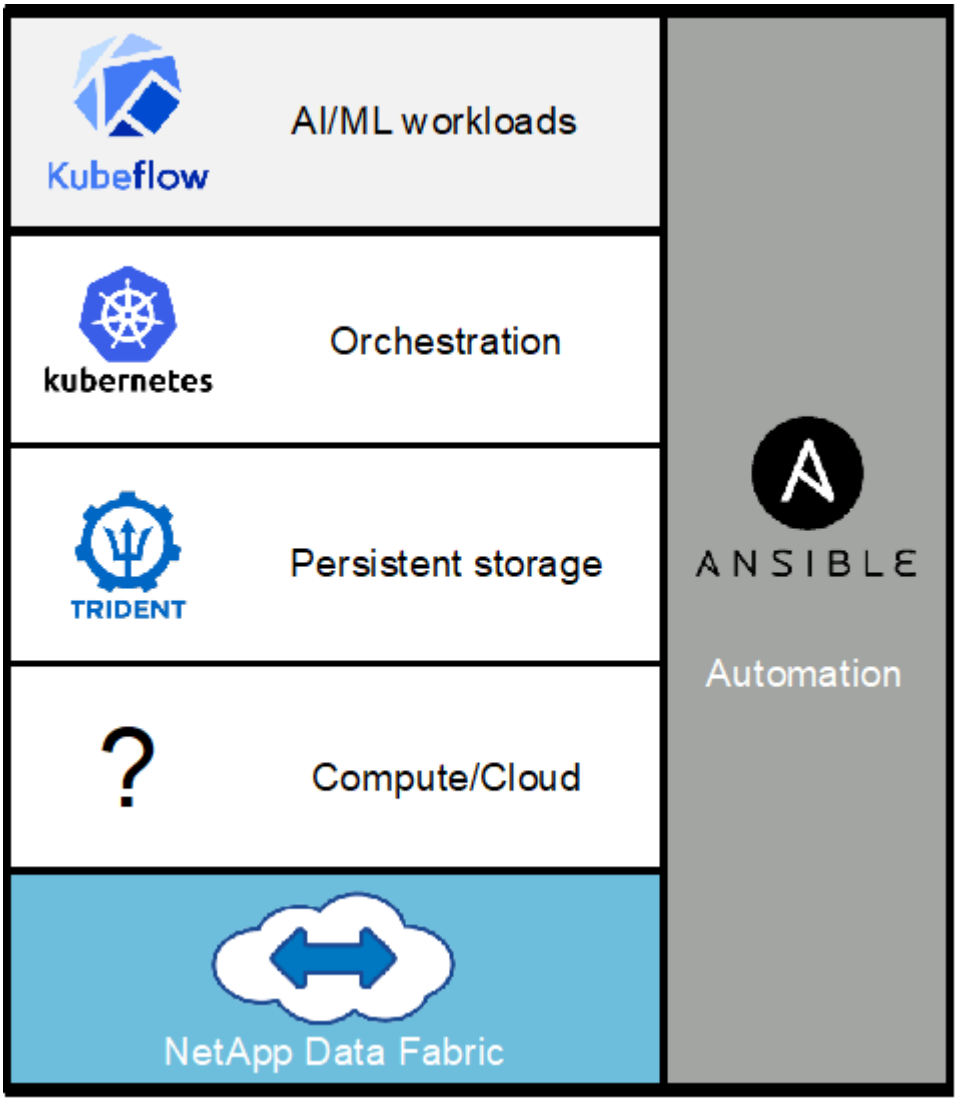
creare, gestire e interagire con volumi di storage persistenti nel formato standard di Kubernetes che già conoscono. Allo stesso tempo, possono sfruttare le funzionalità avanzate di gestione dei dati di NetApp e un data fabric basato sulla tecnologia NetApp. Trident astratta le complessità dello storage persistente e lo rende semplice da utilizzare. Per ulteriori informazioni, visitare il "Sito web di Trident".

NVIDIA DeepOps

DeepOps è un progetto open source di NVIDIA che, utilizzando Ansible, automatizza l'implementazione dei cluster di server GPU in base alle Best practice. DeepOps è modulare e può essere utilizzato per varie attività di implementazione. Per questo documento e per l'esercizio di convalida descritto, DeepOps viene utilizzato per implementare un cluster Kubernetes costituito da nodi di lavoro del server GPU. Per ulteriori informazioni, visitare il "Sito Web di DeepOps".

Kubeflow

Kubeflow è un toolkit open source ai e ML per Kubernetes sviluppato originariamente da Google. Il progetto Kubeflow rende le implementazioni dei flussi di lavoro ai e ML su Kubernetes semplici, portatili e scalabili. Kubeflow astratta le complessità di Kubernetes, consentendo agli scienziati dei dati di concentrarsi su ciò che conoscono meglio—data science. Vedere la figura seguente per una visualizzazione. Kubeflow ha ottenuto notevoli risultati con la sempre maggiore standardizzazione dei reparti IT aziendali su Kubernetes. Per ulteriori informazioni, visitare il "Sito web di Kubeflow".



Pipeline Kubeflow

Le pipeline Kubeflow sono un componente chiave di Kubeflow. Le pipeline Kubeflow sono una piattaforma e uno standard per la definizione e l'implementazione di flussi di lavoro portatili e scalabili ai e ML. Per ulteriori informazioni, consultare "Documentazione ufficiale del Kubeflow".

Jupyter notebook Server

Un Jupyter notebook Server è un'applicazione web open source che consente ai data scientist di creare documenti wiki-like denominati Jupyter Notebooks che contengono codice live e test descrittivi. I notebook Jupyter sono ampiamente utilizzati nella community ai e ML come mezzo per documentare, memorizzare e condividere progetti ai e ML. Kubeflow semplifica il provisioning e l'implementazione di Jupyter notebook Server su Kubernetes. Per ulteriori informazioni sui notebook Jupyter, visitare il "Sito web di Jupyter". Per ulteriori informazioni sui notebook Jupyter nel contesto di Kubeflow, vedere "Documentazione ufficiale del Kubeflow".

Flusso d'aria Apache

Apache Airflow è una piattaforma open-source per la gestione del workflow che consente authoring, scheduling e monitoraggio programmatici per flussi di lavoro aziendali complessi. Spesso viene utilizzato per automatizzare i flussi di lavoro ETL e della pipeline di dati, ma non è limitato a questi tipi di flussi di lavoro. Il progetto Airbnb è stato avviato da Airbnb, ma da allora è diventato molto popolare nel settore e ora è sotto gli auspici della Apache Software Foundation. Il flusso d'aria è scritto in Python, i flussi di lavoro del flusso d'aria sono creati tramite script Python e il flusso d'aria è progettato in base al principio della "configurazione come codice". Molti utenti del flusso d'aria aziendale ora eseguono il flusso d'aria su Kubernetes.

Diagrammi aciclici diretti (DAG)

Nel flusso d'aria, i flussi di lavoro sono denominati diagrammi ad aciclico diretto (DAG). I dag sono costituiti da task che vengono eseguiti in sequenza, in parallelo o in una combinazione dei due, a seconda della definizione DAG. Il programma di pianificazione del flusso d'aria esegue singole attività su un array di lavoratori, rispettando le dipendenze a livello di attività specificate nella definizione DAG. I dag vengono definiti e creati tramite script Python.

NetApp ONTAP 9

NetApp ONTAP 9 è l'ultima generazione di software per la gestione dello storage NetApp che consente a aziende come la tua di modernizzare l'infrastruttura e di passare a un data center cloud-ready. Grazie alle funzionalità di gestione dei dati leader del settore, ONTAP consente di gestire e proteggere i dati con un singolo set di strumenti, indipendentemente dalla posizione in cui risiedono. Puoi anche spostare liberamente i dati ovunque ti servano: Edge, core o cloud. ONTAP 9 include numerose funzionalità che semplificano la gestione dei dati, accelerano e proteggono i tuoi dati critici e la tua infrastruttura a prova di futuro attraverso architetture di cloud ibrido.

Semplifica la gestione dei dati

La gestione dei dati è fondamentale per le operazioni IT aziendali, in modo da poter utilizzare le risorse appropriate per le applicazioni e i set di dati. ONTAP include le seguenti funzionalità per ottimizzare e semplificare le operazioni e ridurre il costo totale delle operazioni:

- **Compattazione dei dati inline e deduplica estesa.** la compattazione dei dati riduce lo spazio sprecato all'interno dei blocchi di storage e la deduplica aumenta significativamente la capacità effettiva.
- **Qualità del servizio (QoS) minima, massima e adattiva.** i controlli QoS granulari aiutano a mantenere i livelli di performance per le applicazioni critiche in ambienti altamente condivisi.

- **ONTAP FabricPool.** questa funzione offre il tiering automatico dei dati cold per le opzioni di cloud storage pubblico e privato, tra cui Amazon Web Services (AWS), Azure e lo storage basato su oggetti NetApp StorageGRID.

Accelera e proteggi i dati

ONTAP offre livelli superiori di performance e protezione dei dati ed estende queste funzionalità con le seguenti funzionalità:

- **Prestazioni elevate e bassa latenza.** ONTAP offre il throughput più elevato possibile con la latenza più bassa possibile.
- **Tecnologia NetApp ONTAP FlexGroup.** Un volume FlexGroup è un container di dati dalle performance elevate che può scalare linearmente fino a 20 PB e 400 miliardi di file, fornendo un singolo namespace che semplifica la gestione dei dati.
- **Protezione dei dati.** ONTAP offre funzionalità di protezione dei dati integrate con gestione comune su tutte le piattaforme.
- **Crittografia dei volumi NetApp.** ONTAP offre crittografia nativa a livello di volume con supporto per la gestione delle chiavi sia integrata che esterna.

Infrastruttura a prova di futuro

ONTAP 9 aiuta a soddisfare le tue esigenze di business esigenti e in continua evoluzione:

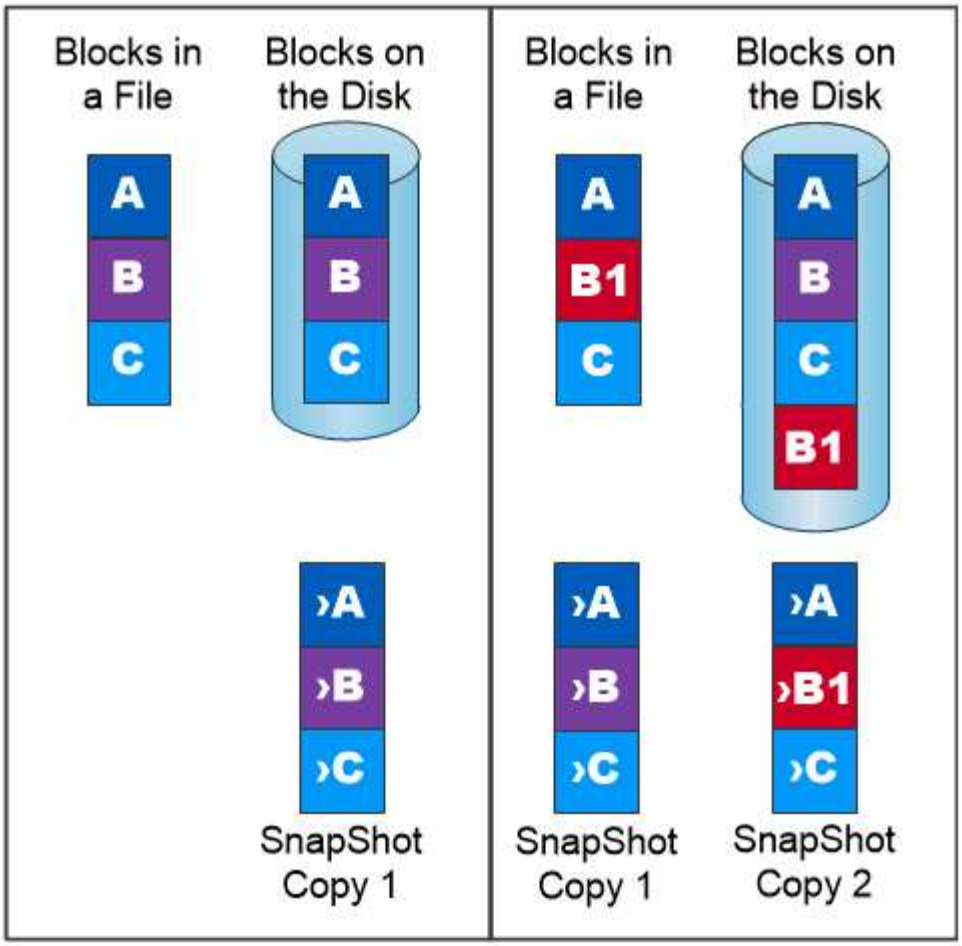
- **Scalabilità perfetta e operazioni senza interruzioni.** ONTAP supporta l'aggiunta senza interruzioni di capacità ai controller esistenti e ai cluster scale-out. Puoi eseguire l'upgrade alle tecnologie più recenti, come NVMe e 32GB FC, senza costose migrazioni dei dati o interruzioni.
- **Connessione al cloud.** ONTAP è uno dei software di gestione dello storage più connessi al cloud, con opzioni per lo storage definito tramite software (ONTAP Select) e le istanze native del cloud (NetApp Cloud Volumes Service) in tutti i cloud pubblici.
- **Integrazione con le applicazioni emergenti.** utilizzando la stessa infrastruttura che supporta le applicazioni aziendali esistenti, ONTAP offre servizi dati di livello Enterprise per piattaforme e applicazioni di prossima generazione come OpenStack, Hadoop e MongoDB.

Copie Snapshot di NetApp

Una copia Snapshot di NetApp è un'immagine point-in-time di sola lettura di un volume. L'immagine consuma uno spazio di storage minimo e comporta un overhead delle performance trascurabile, in quanto registra solo le modifiche apportate ai file creati dall'ultima copia Snapshot, come illustrato nella figura seguente.

Le copie Snapshot devono la loro efficienza alla tecnologia di virtualizzazione dello storage ONTAP principale, il layout di file Write Anywhere (WAFL). Come un database, WAFL utilizza i metadati per indicare i blocchi di dati effettivi sul disco. Tuttavia, a differenza di un database, WAFL non sovrascrive i blocchi esistenti. Scrive i dati aggiornati in un nuovo blocco e cambia i metadati. È perché ONTAP fa riferimento ai metadati quando crea una copia Snapshot, piuttosto che copiare i blocchi di dati, che le copie Snapshot sono così efficienti. In questo modo si eliminano i tempi di ricerca che altri sistemi devono affrontare per individuare i blocchi da copiare, nonché i costi di creazione della copia stessa.

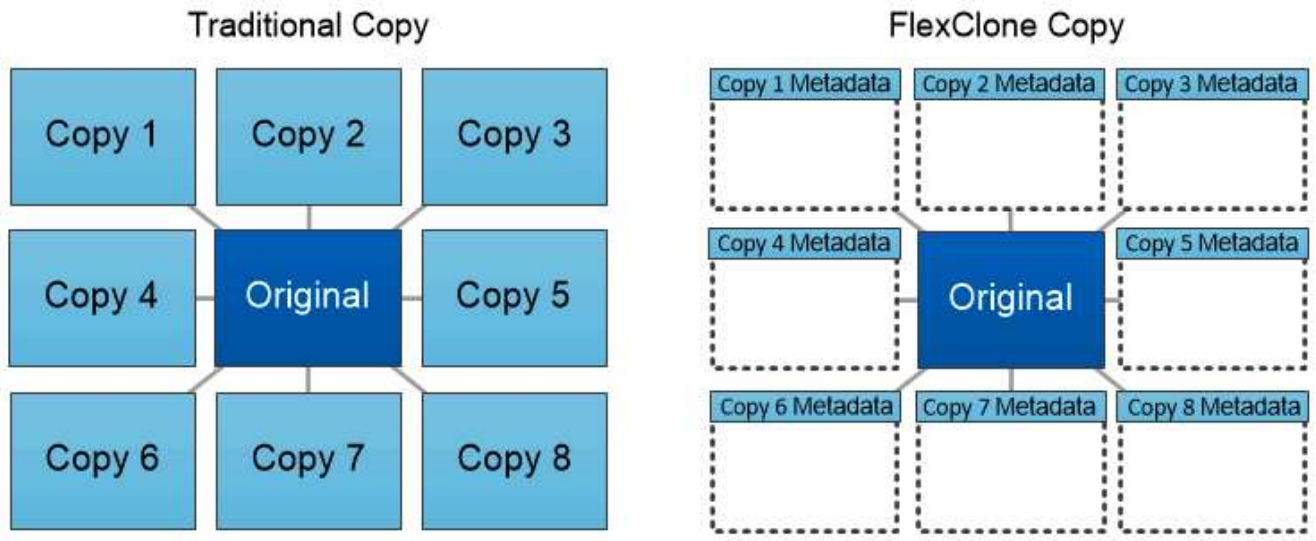
È possibile utilizzare una copia Snapshot per ripristinare singoli file o LUN o per ripristinare l'intero contenuto di un volume. ONTAP confronta le informazioni del puntatore nella copia Snapshot con i dati su disco per ricostruire l'oggetto mancante o danneggiato, senza downtime o costi di performance significativi.



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

Tecnologia NetApp FlexClone

La tecnologia NetApp FlexClone fa riferimento ai metadati Snapshot per creare copie scrivibili point-in-time di un volume. Le copie condividono i blocchi di dati con i genitori, senza consumare storage, ad eccezione di quanto richiesto per i metadati fino a quando le modifiche non vengono scritte nella copia, come illustrato nella figura seguente. Il software FlexClone consente di copiare quasi istantaneamente anche i set di dati più grandi, anche se le copie tradizionali richiedono pochi minuti o persino ore. Ciò lo rende ideale per situazioni in cui sono necessarie più copie di set di dati identici (ad esempio un'area di lavoro di sviluppo) o copie temporanee di un set di dati (test di un'applicazione rispetto a un set di dati di produzione).

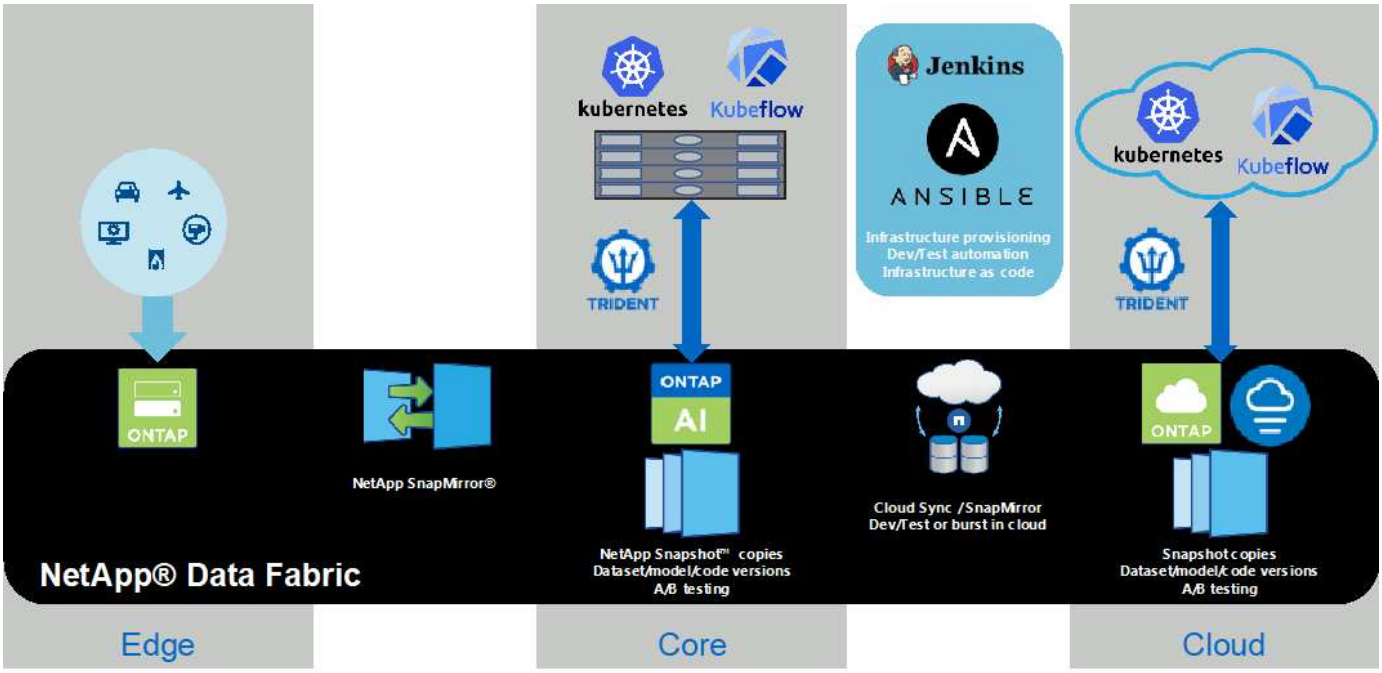


FlexClone copies share data blocks with their parents, consuming no storage except what is required for metadata.

Tecnologia NetApp SnapMirror Data Replication

Il software NetApp SnapMirror è una soluzione di replica unificata conveniente e facile da utilizzare per tutto il data fabric. Replica i dati ad alta velocità su LAN o WAN. Offre un'elevata disponibilità dei dati e una rapida replica dei dati per applicazioni di tutti i tipi, incluse le applicazioni business-critical in ambienti virtuali e tradizionali. Quando si replicano i dati su uno o più sistemi storage NetApp e si aggiornano continuamente i dati secondari, i dati vengono mantenuti aggiornati e disponibili quando necessario. Non sono richiesti server di replica esterni. Vedere la figura seguente per un esempio di architettura che sfrutta la tecnologia SnapMirror.

Il software SnapMirror sfrutta le efficienze dello storage NetApp ONTAP inviando solo i blocchi modificati sulla rete. Il software SnapMirror utilizza inoltre la compressione di rete integrata per accelerare i trasferimenti di dati e ridurre l'utilizzo della larghezza di banda di rete fino al 70%. Con la tecnologia SnapMirror, è possibile sfruttare un flusso di dati di replica con risorse limitate per creare un singolo repository che mantiene il mirror attivo e le copie point-in-time precedenti, riducendo il traffico di rete fino al 50%.



Copia e sincronizzazione di NetApp BlueXP

La copia e sincronizzazione di BlueXP è un servizio NetApp per una sincronizzazione dei dati rapida e sicura. Sia che tu debba trasferire file tra condivisioni di file NFS o SMB on-premise, NetApp StorageGRID, NetApp ONTAP S3, NetApp Cloud Volumes Service, Azure NetApp Files, AWS S3, AWS EFS, BLOB di Azure, Google Cloud Storage, o IBM Cloud Object Storage, BlueXP Copy e Sync sposta i file dove ne hai bisogno in modo rapido e sicuro.

Una volta trasferiti, i dati sono completamente disponibili per l'utilizzo sia sull'origine che sulla destinazione. La copia e sincronizzazione di BlueXP può sincronizzare i dati on-demand quando viene attivato un aggiornamento o sincronizzare costantemente i dati in base a una pianificazione predefinita. Indipendentemente, BlueXP Copy e Sync sposta solo i delta, così tempo e denaro spesi per la replica dei dati sono ridotti al minimo.

BlueXP Copy and Sync è un tool software as a service (SaaS) estremamente semplice da configurare e utilizzare. I trasferimenti dei dati attivati da BlueXP Copy e Sync sono effettuati dai broker di dati. I data broker di BlueXP Copy e Sync possono essere implementati in AWS, Azure, Google Cloud Platform o on-premise.

XCP di NetApp

NetApp XCP è un software basato su client per migrazioni di dati da qualsiasi a NetApp e da NetApp a NetApp e informazioni sui file system. XCP è progettato per scalare e ottenere le massime performance utilizzando tutte le risorse di sistema disponibili per gestire set di dati ad alto volume e migrazioni ad alte performance. XCP consente di ottenere una visibilità completa nel file system con la possibilità di generare report.

NetApp XCP è disponibile in un singolo pacchetto che supporta i protocolli NFS e SMB. XCP include un binario Linux per set di dati NFS e un eseguibile Windows per set di dati SMB.

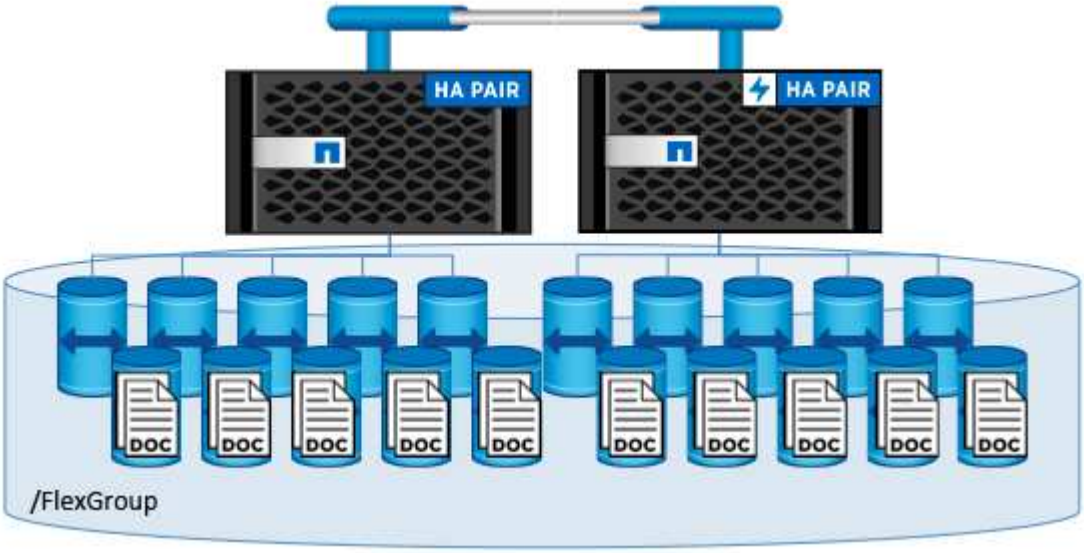
NetApp XCP file Analytics è un software basato su host che rileva le condivisioni di file, esegue scansioni sul file system e fornisce una dashboard per l'analisi dei file. XCP file Analytics è compatibile con sistemi NetApp e non NetApp ed è eseguibile su host Linux o Windows per fornire analisi per NFS e file system esportati da SMB.

NetApp ONTAP FlexGroup Volumes

Un set di dati di training può essere una raccolta di potenzialmente miliardi di file. I file possono includere testo, audio, video e altre forme di dati non strutturati che devono essere memorizzati ed elaborati per essere letti in parallelo. Il sistema di storage deve memorizzare un numero elevato di file di piccole dimensioni e leggerli in parallelo per l'i/o sequenziale e casuale

Un volume FlexGroup è un singolo namespace che comprende più volumi membri costitutivi, come illustrato nella figura seguente. Dal punto di vista dell'amministratore dello storage, un volume FlexGroup viene gestito e agisce come un volume NetApp FlexVol. I file in un volume FlexGroup vengono allocati a singoli volumi membri e non vengono sottoposti a striping tra volumi o nodi. Consentono le seguenti funzionalità:

- I volumi FlexGroup offrono diversi petabyte di capacità e bassa latenza prevedibile per carichi di lavoro con metadati elevati.
- Supportano fino a 400 miliardi di file nello stesso spazio dei nomi.
- Supportano operazioni parallelizzate nei carichi di lavoro NAS tra CPU, nodi, aggregati e volumi FlexVol costitutivi.



Requisiti hardware e software

La soluzione NetApp ai Control Plane non dipende da questo hardware specifico. La soluzione è compatibile con qualsiasi appliance di storage fisico, istanza software-defined o servizio cloud NetApp, supportato da Trident. Ad esempio, un sistema di storage NetApp AFF, Azure NetApp Files, NetApp Cloud Volumes Service, un'istanza di storage NetApp ONTAP Select definita tramite software o un'istanza di NetApp Cloud Volumes ONTAP. Inoltre, la soluzione può essere implementata su qualsiasi cluster Kubernetes purché la versione di Kubernetes utilizzata sia supportata da Kubeflow e NetApp Trident. Per un elenco delle versioni di Kubernetes supportate da Kubeflow, vedere la "Documentazione ufficiale del Kubeflow". Per un elenco delle versioni di Kubernetes supportate da Trident, vedere "Documentazione di Trident". Per informazioni dettagliate sull'ambiente utilizzato per la convalida della soluzione, consultare le tabelle seguenti.

Componente dell'infrastruttura	Quantità	Dettagli	Sistema operativo
Host di salto per l'implementazione	1	MACCHINA VIRTUALE	Ubuntu 20.04.2 LTS
Nodi master Kubernetes	1	MACCHINA VIRTUALE	Ubuntu 20.04.2 LTS
Nodi di lavoro Kubernetes	2	MACCHINA VIRTUALE	Ubuntu 20.04.2 LTS
Kubernetes nodi di lavoro GPU	2	NVIDIA DGX-1 (bare-metal)	NVIDIA DGX OS 4.0.5 (basato su Ubuntu 18.04.2 LTS)
Storage	1 coppia ha	NetApp AFF A220	NetApp ONTAP 9.7 P6

Componente software	Versione
Flusso d'aria Apache	2.0.1
Helm Chart di Apache Airflow	8.0.8
Docker	19.03.12
Kubeflow	1.2
Kubernetes	1.18.9
Trident di NetApp	21.01.2
NVIDIA DeepOps	Funzionalità di implementazione di Trident dalla filiale master al momento del commit "61898cdfda"; Tutte le altre funzionalità dalla versione 21.03

Supporto

NetApp non offre supporto Enterprise per Apache Airflow, Docker, Kubeflow, Kubernetes o NVIDIA DeepOps. Se sei interessato a una soluzione completamente supportata con funzionalità simili alla soluzione NetApp ai Control Plane, "Contatta NetApp" Informazioni sulle soluzioni ai/ML completamente supportate che NetApp offre insieme ai partner.

Implementazione di Kubernetes

Questa sezione descrive le attività da completare per implementare un cluster Kubernetes in cui implementare la soluzione NetApp ai Control Plane. Se si dispone già di un cluster Kubernetes, è possibile saltare questa sezione se si utilizza una versione di Kubernetes supportata da Kubeflow e NetApp Trident. Per un elenco delle versioni di Kubernetes supportate da Kubeflow, vedere la "Documentazione ufficiale del Kubeflow". Per un elenco delle versioni di Kubernetes supportate da Trident, vedere "Documentazione di Trident".

Per le implementazioni on-premise di Kubernetes che incorporano nodi bare-metal con GPU NVIDIA, NetApp consiglia di utilizzare il tool di implementazione DeepOps Kubernetes di NVIDIA. Questa sezione descrive l'implementazione di un cluster Kubernetes utilizzando DeepOps.

Prerequisiti

Prima di eseguire l'esercizio di implementazione descritto in questa sezione, si presuppone che siano già state eseguite le seguenti attività:

1. Sono già stati configurati nodi Kubernetes bare-metal (ad esempio, un sistema NVIDIA DGX che fa parte di un pod ai ONTAP) in base alle istruzioni di configurazione standard.
2. È stato installato un sistema operativo supportato su tutti i nodi master e worker di Kubernetes e su un host di distribuzione jump. Per un elenco dei sistemi operativi supportati da DeepOps, vedere "Sito DeepOps GitHub".

Utilizzare NVIDIA DeepOps per installare e configurare Kubernetes

Per implementare e configurare il cluster Kubernetes con NVIDIA DeepOps, eseguire le seguenti operazioni da un host di distribuzione jump:

1. Scaricare NVIDIA DeepOps seguendo le istruzioni sul "Pagina introduttiva" Sul sito NVIDIA DeepOps GitHub.
2. Implementare Kubernetes nel cluster seguendo le istruzioni sul "Pagina della Guida all'implementazione di Kubernetes" Sul sito NVIDIA DeepOps GitHub.

Implementazione e configurazione di NetApp Trident

Implementazione e configurazione di NetApp Trident

Questa sezione descrive le attività da completare per installare e configurare NetApp Trident nel cluster Kubernetes.

Prerequisiti

Prima di eseguire l'esercizio di implementazione descritto in questa sezione, si presuppone che siano già state eseguite le seguenti attività:

1. Hai già un cluster Kubernetes funzionante e stai eseguendo una versione di Kubernetes supportata da Trident. Per un elenco delle versioni supportate, vedere "Documentazione di Trident".
2. Disponete già di un'appliance di storage NetApp funzionante, di un'istanza software-defined o di un servizio di cloud storage supportato da Trident.

Installare Trident

Per installare e configurare NetApp Trident nel cluster Kubernetes, eseguire le seguenti attività dall'host di distribuzione jump:

1. Implementare Trident utilizzando uno dei seguenti metodi:
    - Se hai utilizzato NVIDIA DeepOps per implementare il cluster Kubernetes, puoi anche utilizzare NVIDIA DeepOps per implementare Trident nel cluster Kubernetes. Per implementare Trident con DeepOps, seguire "Istruzioni per l'implementazione di Trident" Sul sito NVIDIA DeepOps GitHub.
    - Se non hai utilizzato NVIDIA DeepOps per implementare il cluster Kubernetes o se preferisci semplicemente implementare Trident manualmente, puoi implementare Trident seguendo la "istruzioni per l'implementazione" Nella documentazione di Trident. Per ulteriori informazioni sulla configurazione, assicurarsi di creare almeno un backend Trident e almeno un StorageClass Kubernetes "Back-end" e.

"StorageClasses" Consulta le sottosezioni collegate nei documenti di NetApp.

> Se stai implementando la soluzione NetApp per il piano di controllo ai su un pod ai ONTAP, consulta "Esempi di backend Trident per implementazioni ai ONTAP" Per alcuni esempi di diversi backend Trident che si desidera creare e. "Esempi di storaglasses Kubernetes per implementazioni ai ONTAP" Per alcuni esempi di diverse Kubernetes StorageClasses che potresti voler creare.

Esempi di backend Trident per implementazioni ai ONTAP

Prima di utilizzare Trident per eseguire il provisioning dinamico delle risorse di storage all'interno del cluster Kubernetes, è necessario creare uno o più backend Trident. Gli esempi che seguono rappresentano diversi tipi di backend che è possibile creare se si sta implementando la soluzione per il piano di controllo ai di NetApp su un pod ai di ONTAP. Per ulteriori informazioni sui backend, consultare "Documentazione di Trident".

1. NetApp consiglia di creare un backend Trident abilitato per FlexGroup per ogni LIF di dati (interfaccia di rete logica che fornisce l'accesso ai dati) che si desidera utilizzare sul sistema NetApp AFF. In questo modo, potrai bilanciare i montaggi di volume tra le LIF

   I comandi di esempio che seguono mostrano la creazione di due backend Trident abilitati per FlexGroup per due diverse LIF di dati associate alla stessa SVM (Storage Virtual Machine) di ONTAP. Questi backend utilizzano `ontap-nas-flexgroup` driver di storage. ONTAP supporta due tipi principali di volumi di dati: FlexVol e FlexGroup. I volumi FlexVol sono limitati dalle dimensioni (al momento della scrittura, le dimensioni massime dipendono dalla distribuzione specifica). I volumi FlexGroup, invece, possono scalare linearmente fino a 20 PB e 400 miliardi di file, fornendo un singolo namespace che semplifica notevolmente la gestione dei dati. Pertanto, i volumi FlexGroup sono ottimali per i carichi di lavoro ai e ML che si basano su grandi quantità di dati.

   Se si lavora con una piccola quantità di dati e si desidera utilizzare volumi FlexVol invece di volumi FlexGroup, è possibile creare backend Trident che utilizzano `ontap-nas` driver di storage invece di `ontap-nas-flexgroup` driver di storage.

```
$ 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 consiglia inoltre di creare uno o più backend Trident abilitati per FlexVol. Se si utilizzano volumi FlexGroup per lo storage dei dataset di training, è possibile utilizzare volumi FlexVol per memorizzare risultati, output, informazioni di debug e così via. Se si desidera utilizzare i volumi FlexVol, è necessario creare uno o più backend Trident abilitati per FlexVol. I comandi di esempio che seguono mostrano la creazione di un singolo backend Trident abilitato a FlexVol che utilizza una singola LIF di dati.

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

Esempi di storage Classes Kubernetes per implementazioni ai ONTAP

Prima di utilizzare Trident per eseguire il provisioning dinamico delle risorse di storage all'interno del cluster Kubernetes, è necessario creare una o più Kubernetes StorageClasses. Gli esempi che seguono rappresentano diversi tipi di StorageClasses che potresti voler creare se stai implementando la soluzione NetApp ai Control Plane su un pod ai ONTAP. Per ulteriori informazioni su StorageClasses, vedere "Documentazione di Trident".

1. NetApp consiglia di creare una StorageClass separata per ogni backend Trident abilitato per FlexGroup creato nella sezione "Esempi di backend Trident per implementazioni ai ONTAP", fase 1. Questi StorageClasses granulari consentono di aggiungere i montaggi NFS che corrispondono a LIF specifiche (le LIF specificate al momento della creazione dei Trident Backend) come backend specifico specificato nel file delle specifiche StorageClass. I comandi di esempio che seguono mostrano la creazione di due StorageClasses che corrispondono ai due backend di esempio creati nella sezione "Esempi di backend Trident per implementazioni ai ONTAP", fase 1. Per ulteriori informazioni su StorageClasses, vedere "Documentazione di Trident".

   Per evitare che un volume persistente venga cancellato quando il PVC (PersistentVolumeClaim) corrispondente viene cancellato, nel seguente esempio viene utilizzato un `reclaimPolicy` valore di `Retain`. Per ulteriori informazioni su `reclaimPolicy` vedi il sito ufficiale "Documentazione Kubernetes".

```
$ 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 consiglia inoltre di creare un StorageClass che corrisponda al backend Trident abilitato a FlexVol creato nella sezione "Esempi di backend Trident per implementazioni ai ONTAP", punto 2. I comandi di esempio che seguono mostrano la creazione di una singola classe di storage per volumi FlexVol.

   Nell'esempio seguente, un particolare backend non viene specificato nel file di definizione StorageClass perché è stato creato un solo backend Trident abilitato a FlexVol. Quando si utilizza Kubernetes per amministrare volumi che utilizzano questo StorageClass, Trident tenta di utilizzare qualsiasi backend disponibile che utilizzi `ontap-nas` driver.

```
$ 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 consiglia inoltre di creare una classe di storage generica per i volumi FlexGroup. I seguenti comandi di esempio mostrano la creazione di una singola classe di storage generica per volumi FlexGroup.

   Si noti che un particolare backend non viene specificato nel file di definizione StorageClass. Pertanto, quando si utilizza Kubernetes per amministrare volumi che utilizzano questo StorageClass, Trident tenta di utilizzare qualsiasi backend disponibile che utilizzi `ontap-nas-flexgroup` driver.

```
$ 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
```

Implementazione di Kubeflow

In questa sezione vengono descritte le attività da completare per implementare Kubeflow nel cluster Kubernetes.

Prerequisiti

Prima di eseguire l'esercizio di implementazione descritto in questa sezione, si presuppone che siano già state eseguite le seguenti attività:

1. Hai già un cluster Kubernetes funzionante e stai eseguendo una versione di Kubernetes supportata da Kubeflow. Per un elenco delle versioni supportate, vedere "Documentazione ufficiale del Kubeflow".
2. NetApp Trident è già stato installato e configurato nel cluster Kubernetes, come descritto in "Implementazione e configurazione di Trident".

Impostare la classe di storage Kubernetes predefinita

Prima di implementare Kubeflow, è necessario specificare un StorageClass predefinito all'interno del cluster Kubernetes. Il processo di implementazione di Kubeflow tenta di eseguire il provisioning di nuovi volumi persistenti utilizzando la classe di storage predefinita. Se non viene indicato StorageClass come StorageClass predefinito, l'implementazione non riesce. Per designare una StorageClass predefinita all'interno del cluster, eseguire la seguente attività dall'host di distribuzione jump. Se è già stata designata una StorageClass predefinita all'interno del cluster, è possibile saltare questo passaggio.

1. Designare uno dei StorageClasses esistenti come StorageClass predefinito. I comandi di esempio che seguono mostrano la designazione di StorageClass denominata `ontap-ai- flexvols-retain` Come StorageClass di default.

> Il `ontap-nas-flexgroup` Il tipo di backend Trident ha una dimensione minima del PVC che è abbastanza grande. Per impostazione predefinita, Kubeflow tenta di eseguire il provisioning di PVC di dimensioni limitate a poche GB. Pertanto, non è necessario designare un StorageClass che utilizzi `ontap-nas-flexgroup` Tipo di backend come StorageClass predefinito ai fini dell'implementazione di Kubeflow.

```
$ 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
```

Utilizza NVIDIA DeepOps per implementare Kubeflow

NetApp consiglia di utilizzare il tool di implementazione Kubeflow fornito da NVIDIA DeepOps. Per implementare Kubeflow nel cluster Kubernetes utilizzando lo strumento di implementazione DeepOps, eseguire le seguenti operazioni dall'host di distribuzione jump.

> In alternativa, è possibile implementare Kubeflow manualmente seguendo la "istruzioni per l'installazione" Nella documentazione ufficiale del Kubeflow

1. Implementare Kubeflow nel cluster seguendo la "Istruzioni per l'implementazione di Kubeflow" Sul sito NVIDIA DeepOps GitHub.
2. Annotare l'URL del dashboard Kubeflow prodotto dal tool di implementazione DeepOps Kubeflow.

```
$ ./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. Verificare che tutti i pod implementati nello spazio dei nomi Kubeflow mostrino un `STATUS` di `Running` e verificare che nessun componente implementato all'interno dello spazio dei nomi sia in stato di errore. L'avvio di tutti i pod potrebbe richiedere alcuni minuti.

```
$ 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
```

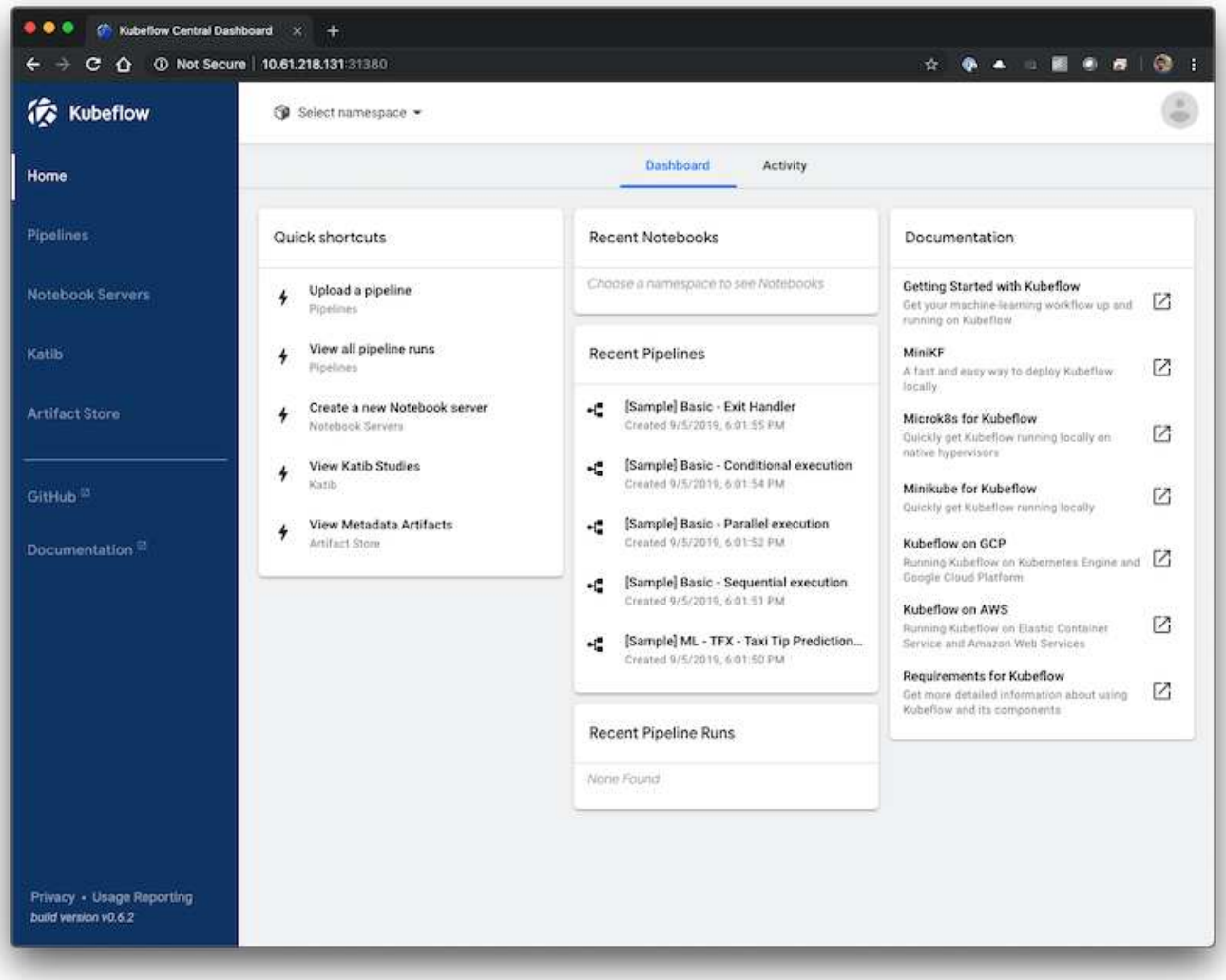
```
1           93s
deployment.apps/ml-pipeline-scheduledworkflow              1/1     1
1           93s
deployment.apps/ml-pipeline-ui                             1/1     1
1           93s
deployment.apps/ml-pipeline-viewer-controller-deployment   1/1     1
1           93s
deployment.apps/mysql                                      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. Nel browser Web, accedere alla dashboard centrale di Kubeflow accedendo all'URL annotato al punto 2.

   Il nome utente predefinito è `admin@kubeflow.org`e la password predefinita è `12341234`. Per creare altri utenti, seguire le istruzioni in "Documentazione ufficiale del Kubeflow".



Operazioni e attività Kubeflow di esempio

Questa sezione include esempi di varie operazioni e attività che è possibile eseguire utilizzando Kubeflow.

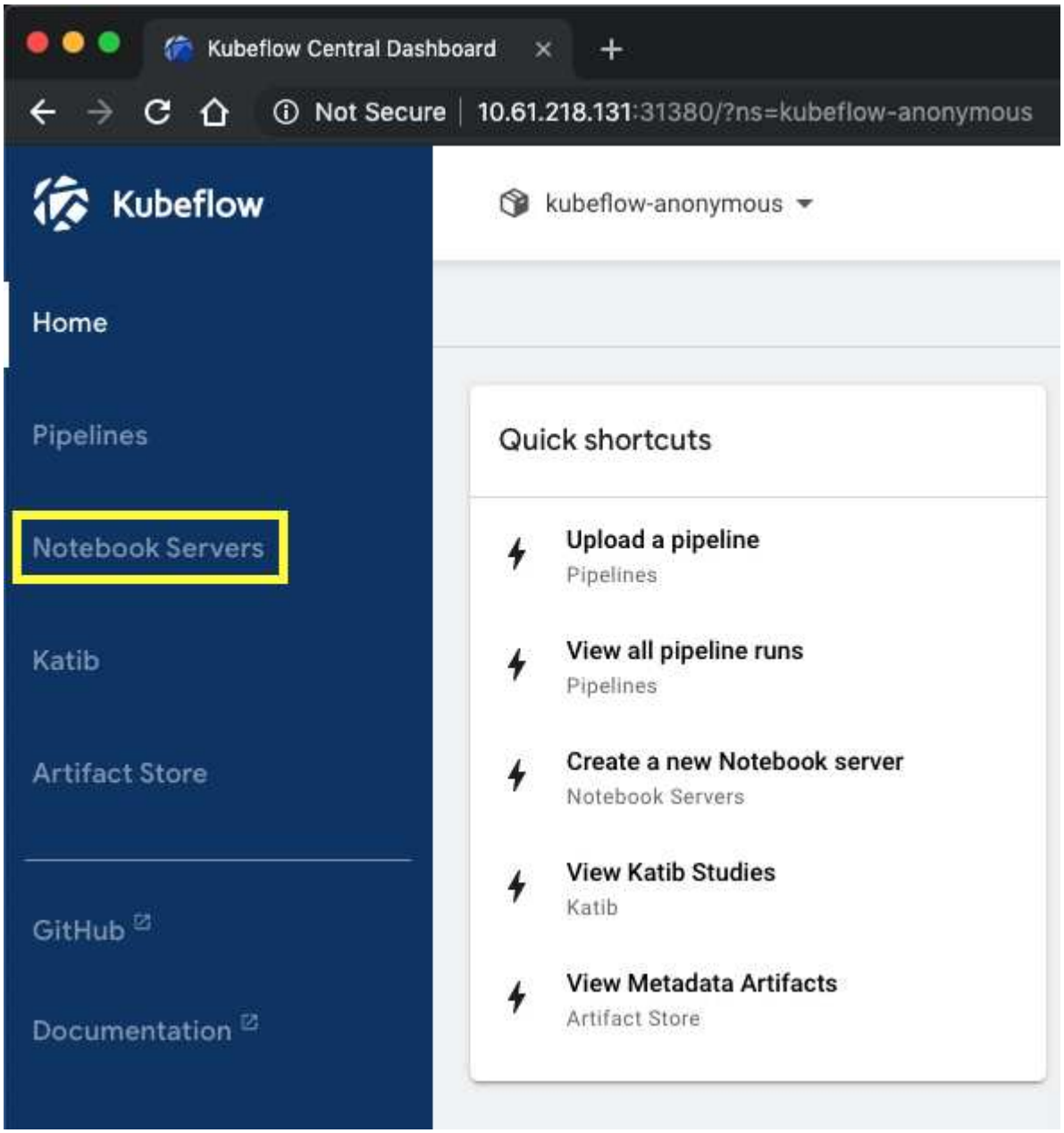
Operazioni e attività Kubeflow di esempio

Questa sezione include esempi di varie operazioni e attività che è possibile eseguire utilizzando Kubeflow.

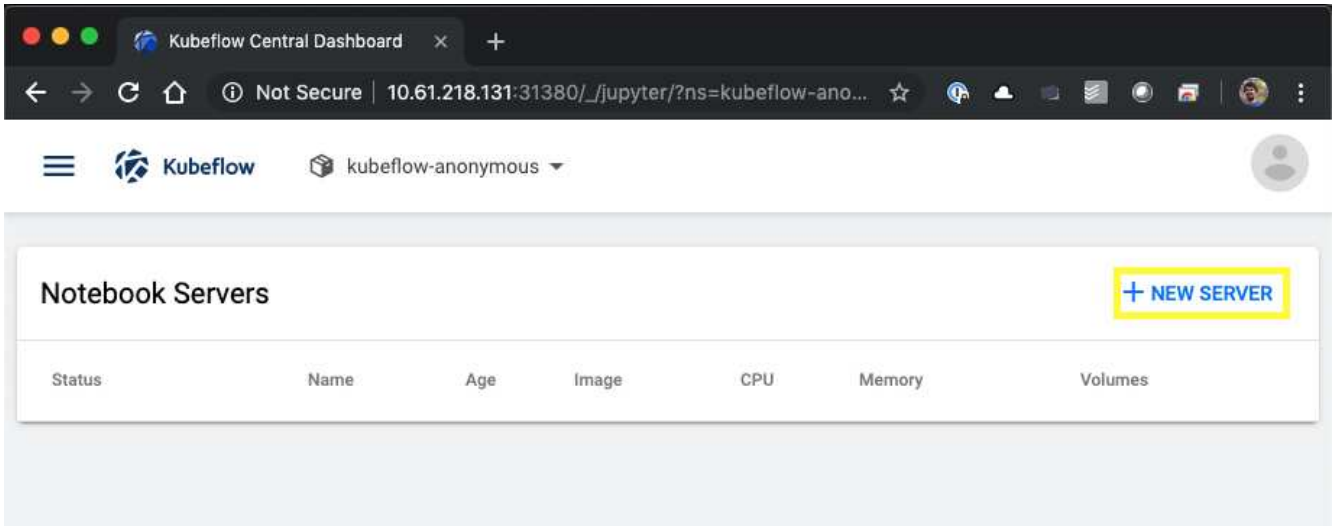
Provisioning di un’area di lavoro Jupyter notebook per l’utilizzo da parte di Data Scientist o Developer

Kubeflow è in grado di eseguire rapidamente il provisioning dei nuovi server Jupyter notebook per agire come aree di lavoro per scienziati dei dati. Per eseguire il provisioning di un nuovo server Jupyter notebook con Kubeflow, eseguire le seguenti operazioni. Per ulteriori informazioni sui notebook Jupyter all’interno del contesto Kubeflow, vedere "Documentazione ufficiale del Kubeflow".

1. Dalla dashboard centrale di Kubeflow, fare clic su notebook Servers nel menu principale per accedere alla pagina di amministrazione del server Jupyter notebook.

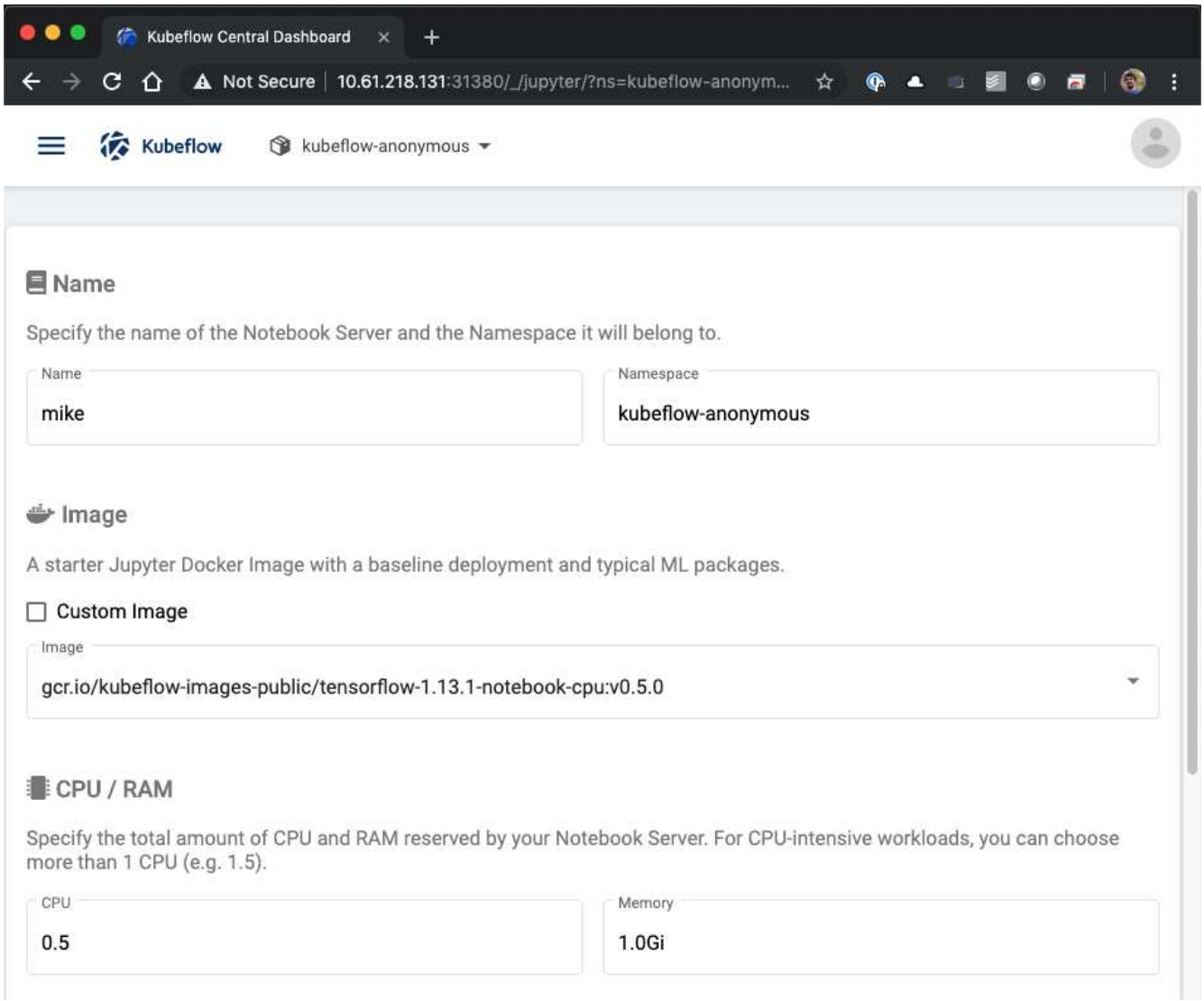


2. Fare clic su New Server (nuovo server) per eseguire il provisioning di un nuovo server Jupyter notebook.

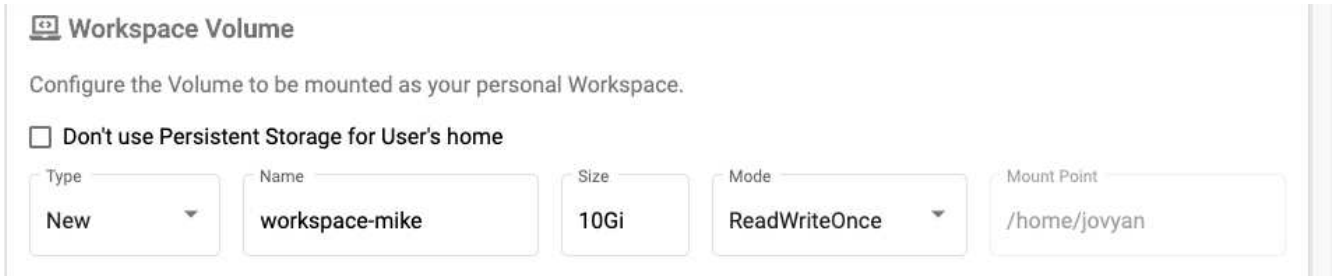


3. Assegnare un nome al nuovo server, scegliere l'immagine Docker su cui si desidera basare il server e specificare la quantità di CPU e RAM da riservare al server. Se il campo namespace è vuoto, utilizzare il menu Select namespace (Seleziona spazio dei nomi) nell'intestazione della pagina per scegliere uno spazio dei nomi. Il campo namespace viene quindi compilato automaticamente con lo spazio dei nomi scelto.

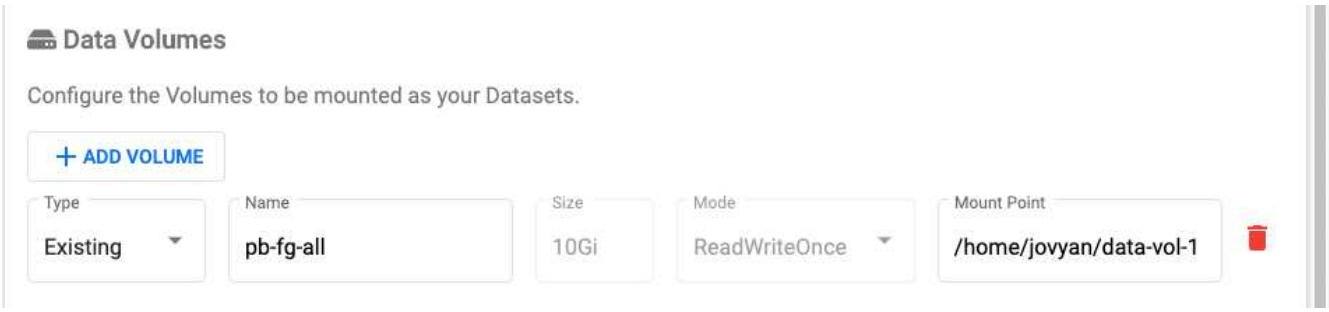
   Nell'esempio seguente, il `kubeflow-anonymous` viene scelto lo spazio dei nomi. Inoltre, vengono accettati i valori predefiniti per l'immagine Docker, la CPU e la RAM.



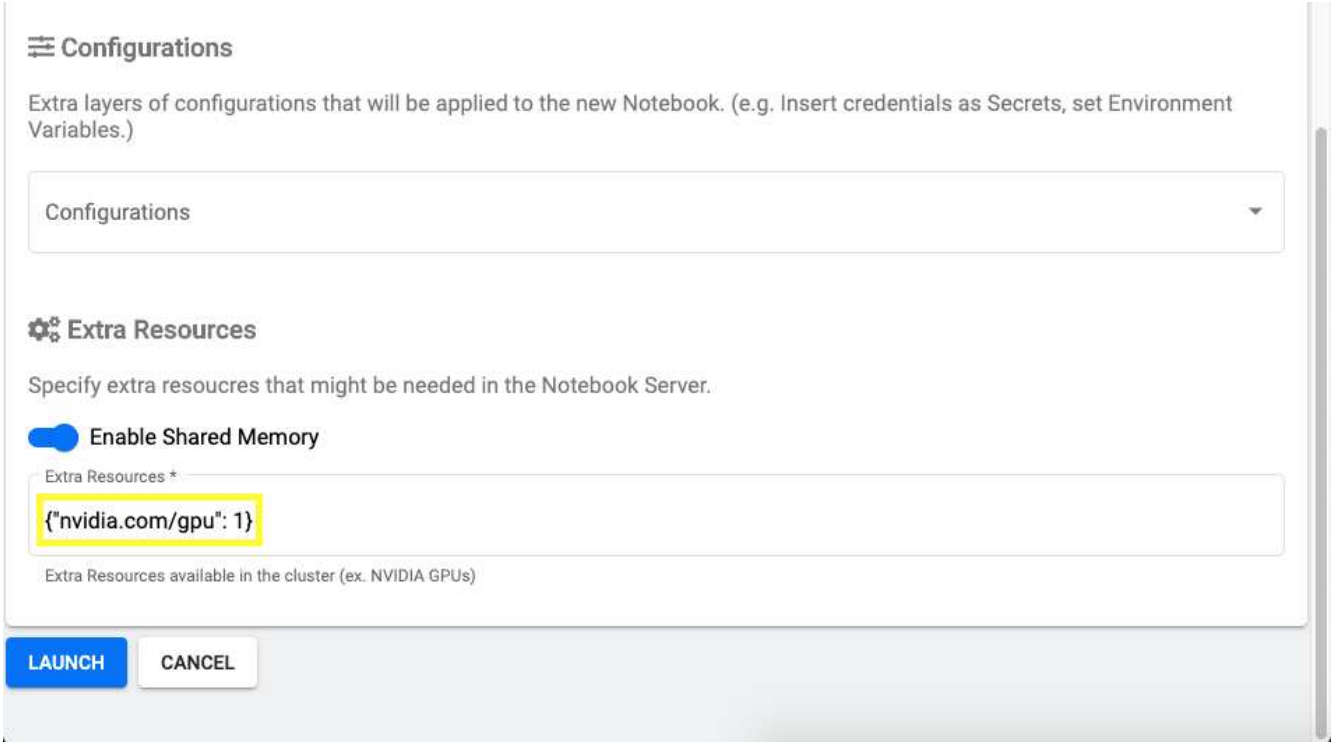
4. Specificare i dettagli del volume dello spazio di lavoro. Se si sceglie di creare un nuovo volume, il provisioning di tale volume o PVC viene eseguito utilizzando la classe di storage predefinita. Perché un StorageClass che utilizza Trident è stato designato come StorageClass predefinito nella sezione "Implementazione di Kubeflow", Il volume o PVC viene fornito con Trident. Questo volume viene montato automaticamente come area di lavoro predefinita all'interno del container Jupyter notebook Server. Tutti i notebook creati dall'utente sul server che non vengono salvati in un volume di dati separato vengono salvati automaticamente in questo volume di spazio di lavoro. Pertanto, i notebook sono persistenti durante i riavvii.



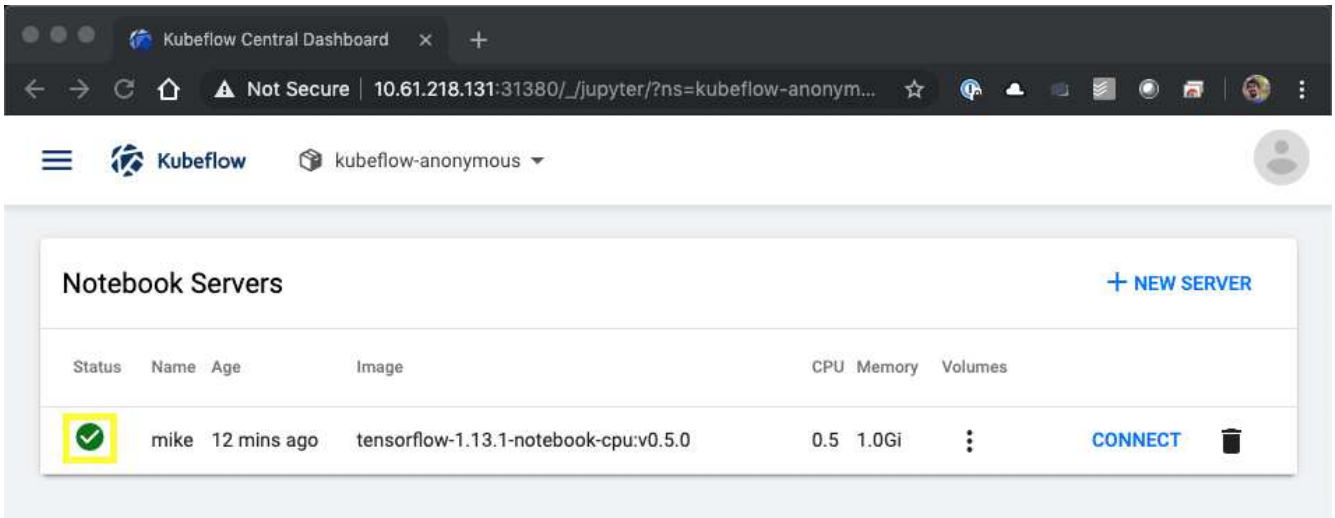
5. Aggiungere volumi di dati. Nell'esempio seguente viene specificato un PVC esistente denominato 'pb-fg-all' e viene accettato il punto di montaggio predefinito.



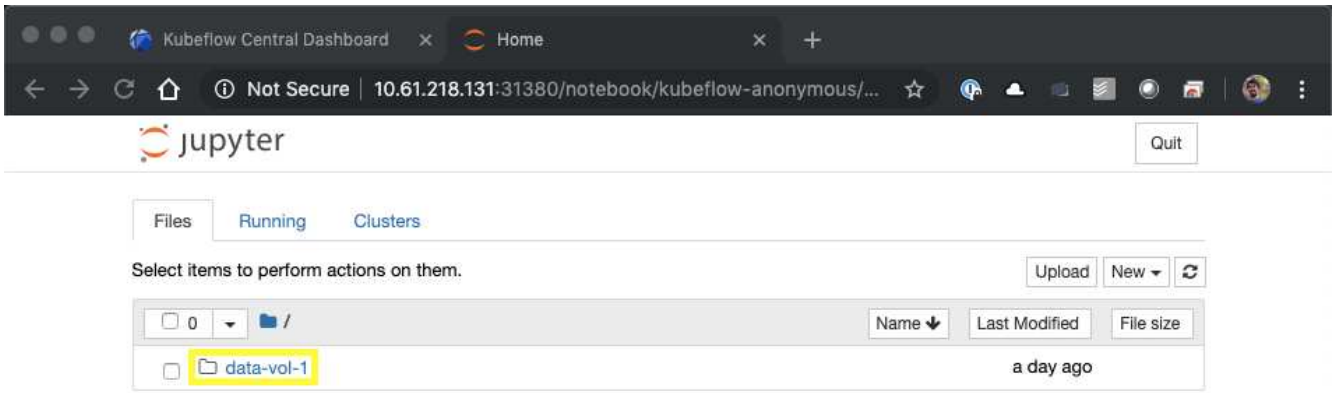
6. **Opzionale:** richiedere l'allocazione del numero desiderato di GPU al notebook server. Nell'esempio seguente, viene richiesta una GPU.

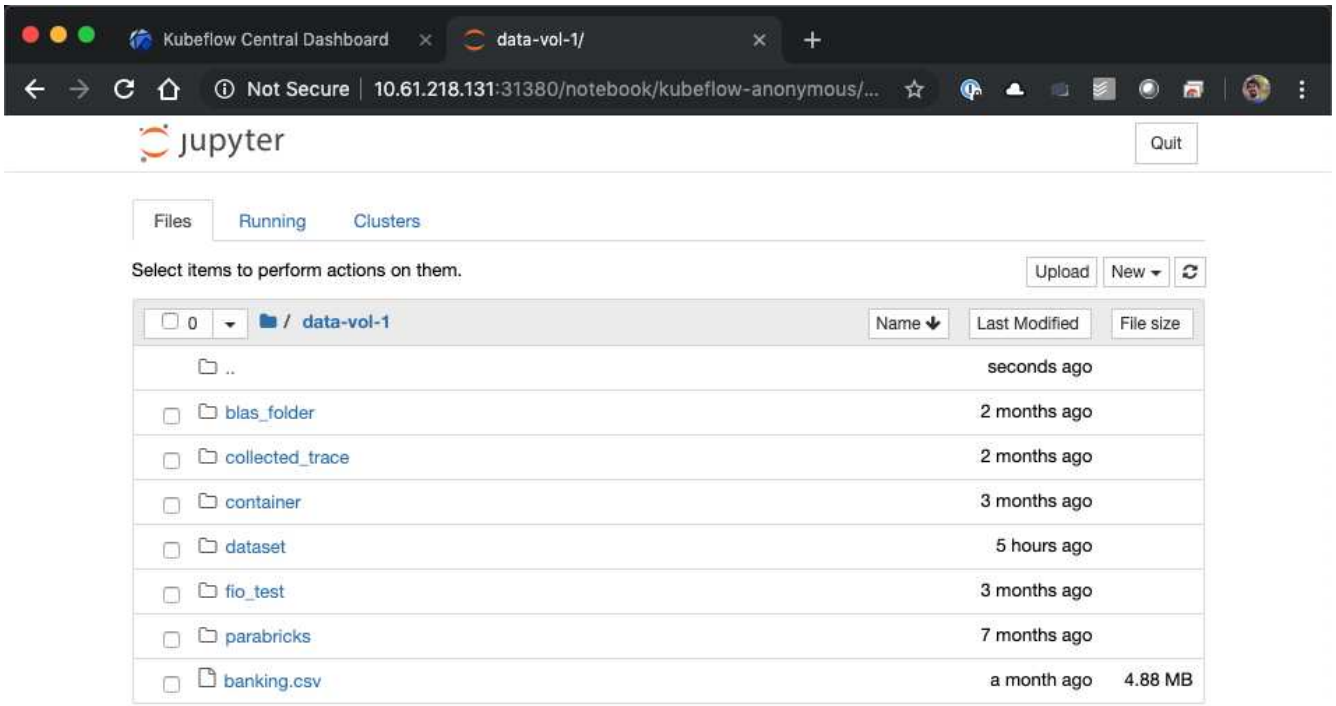


7. Fare clic su Launch (Avvia) per eseguire il provisioning del nuovo notebook server.
8. Attendere il provisioning completo del server notebook. Questa operazione può richiedere alcuni minuti se non si è mai eseguito il provisioning di un server utilizzando l'immagine Docker specificata, in quanto l'immagine deve essere scaricata. Una volta completato il provisioning del server, viene visualizzato un segno di spunta verde nella colonna Status (Stato) della pagina di amministrazione del server Jupyter notebook.



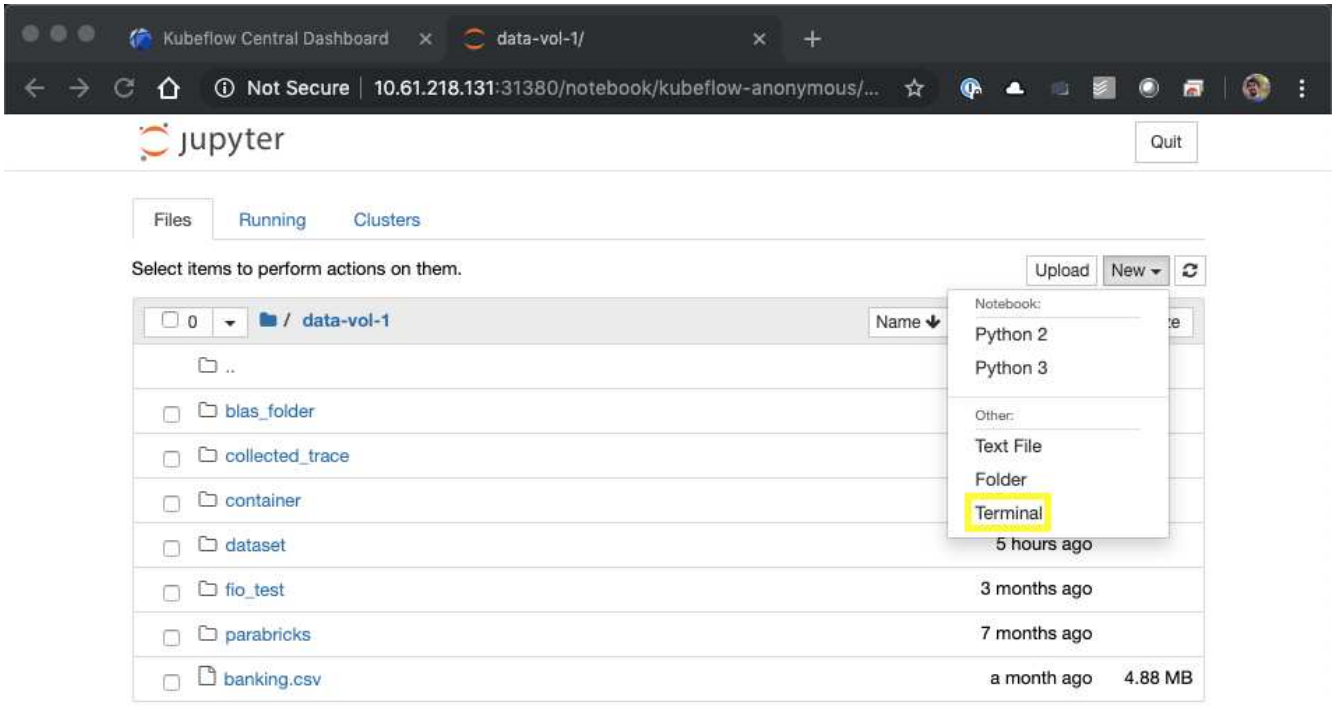
9. Fare clic su Connect (Connetti) per connettersi alla nuova interfaccia Web del server.
10. Verificare che il volume del set di dati specificato al punto 6 sia montato sul server. Si noti che questo volume viene montato nell'area di lavoro predefinita per impostazione predefinita. Dal punto di vista dell'utente, questa è solo un'altra cartella all'interno dello spazio di lavoro. L'utente, che è probabilmente un data scientist e non un esperto di infrastruttura, non deve possedere alcuna esperienza di storage per utilizzare questo volume.

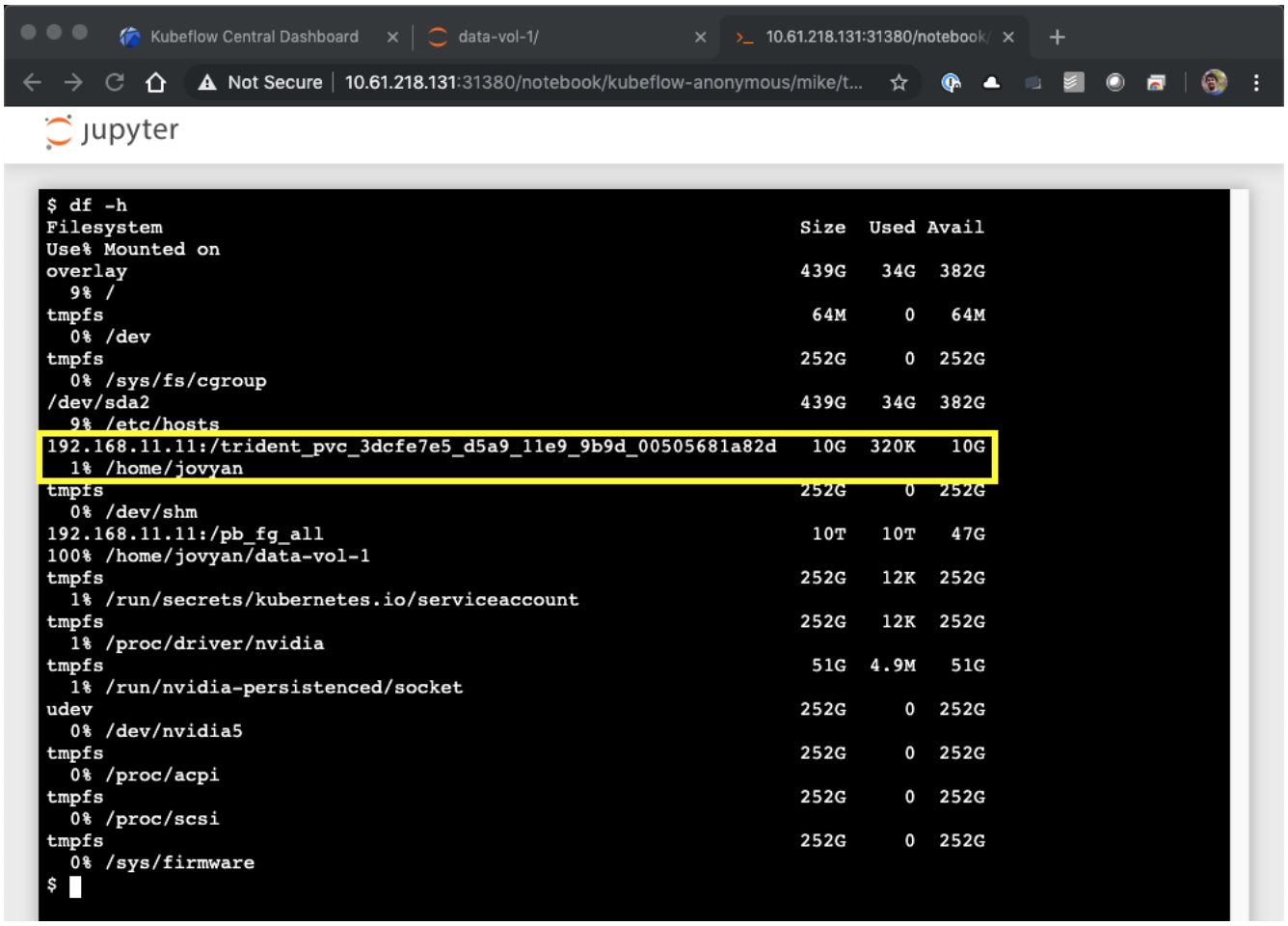




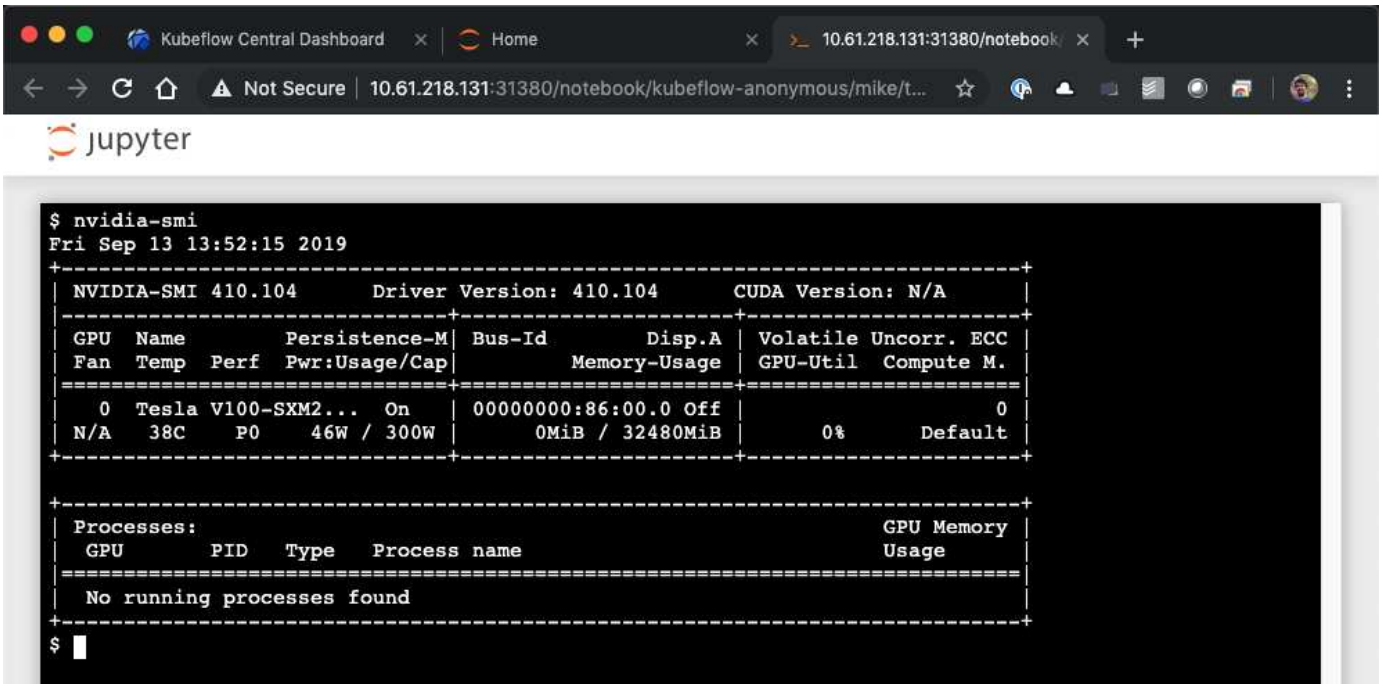
11. Aprire un terminale e, supponendo che sia stato richiesto un nuovo volume nel passaggio 5, eseguire `df -h` Per confermare che un nuovo volume persistente con provisioning Trident è montato come area di lavoro predefinita.

    La directory predefinita dello spazio di lavoro è la directory di base che viene visualizzata quando si accede per la prima volta all'interfaccia Web del server. Pertanto, tutti gli artefatti creati utilizzando l'interfaccia Web vengono memorizzati su questo volume persistente con provisioning Trident.





12. Utilizzando il terminale, eseguire `nvidia-smi` Per confermare che il numero corretto di GPU è stato allocato al notebook server. Nell'esempio seguente, una GPU è stata allocata al notebook server come richiesto nel passaggio 7.



Esempi di notebook e pipeline

Il "NetApp Data Science Toolkit per Kubernetes" Utilizzabile in combinazione con Kubeflow. L'utilizzo del NetApp Data Science Toolkit con Kubeflow offre i seguenti vantaggi:

- I data scientist possono eseguire operazioni avanzate di gestione dei dati NetApp direttamente da un Jupyter notebook.
- Le operazioni avanzate di gestione dei dati NetApp possono essere incorporate nei flussi di lavoro automatizzati utilizzando il framework Kubeflow Pipeline.

Fare riferimento a. "Esempi di Kubeflow" Sezione all'interno del repository GitHub del NetApp Data Science Toolkit per informazioni dettagliate sull'utilizzo del toolkit con Kubeflow.

Implementazione di Apache Airflow

NetApp consiglia di eseguire Apache Airflow su Kubernetes. Questa sezione descrive le attività da completare per implementare il flusso d'aria nel cluster Kubernetes.

> È possibile implementare il flusso d'aria su piattaforme diverse da Kubernetes. L'implementazione del flusso d'aria su piattaforme diverse da Kubernetes non rientra nell'ambito di questa soluzione.

Prerequisiti

Prima di eseguire l'esercizio di implementazione descritto in questa sezione, si presuppone che siano già state eseguite le seguenti attività:

1. Hai già un cluster Kubernetes funzionante.
2. NetApp Trident è già stato installato e configurato nel cluster Kubernetes, come descritto nella sezione "implementazione e configurazione di NetApp Trident".

Installare Helm

Il flusso d'aria viene implementato utilizzando Helm, un popolare gestore di pacchetti per Kubernetes. Prima di implementare il flusso d'aria, è necessario installare Helm sull'host di distribuzione jump. Per installare Helm sull'host di distribuzione jump, seguire la "istruzioni per l'installazione" Nella documentazione ufficiale di Helm.

Impostare la classe di storage Kubernetes predefinita

Prima di implementare il flusso d'aria, è necessario specificare un StorageClass predefinito all'interno del cluster Kubernetes. Il processo di implementazione del flusso d'aria tenta di eseguire il provisioning di nuovi volumi persistenti utilizzando la classe di storage predefinita. Se non viene indicato StorageClass come StorageClass predefinito, l'implementazione non riesce. Per designare una StorageClass predefinita all'interno del cluster, seguire le istruzioni riportate nella sezione "Implementazione di Kubeflow". Se è già stata designata una StorageClass predefinita all'interno del cluster, è possibile saltare questo passaggio.

USA Helm per implementare il flusso d'aria

Per implementare il flusso d'aria nel cluster Kubernetes utilizzando Helm, eseguire le seguenti operazioni

dall'host di distribuzione jump:

1. Implementare il flusso d'aria utilizzando Helm seguendo il "istruzioni per l'implementazione" Per il diagramma ufficiale del flusso d'aria sull'Artifact Hub. I comandi di esempio che seguono mostrano l'implementazione del flusso d'aria con Helm. Modificare, aggiungere e/o rimuovere i valori in `custom-values.yaml` file in base alle necessità, a seconda dell'ambiente e della configurazione desiderata.

```
$ 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. Verificare che tutti i pod del flusso d'aria siano in funzione. L'avvio di tutti i pod potrebbe richiedere alcuni minuti.

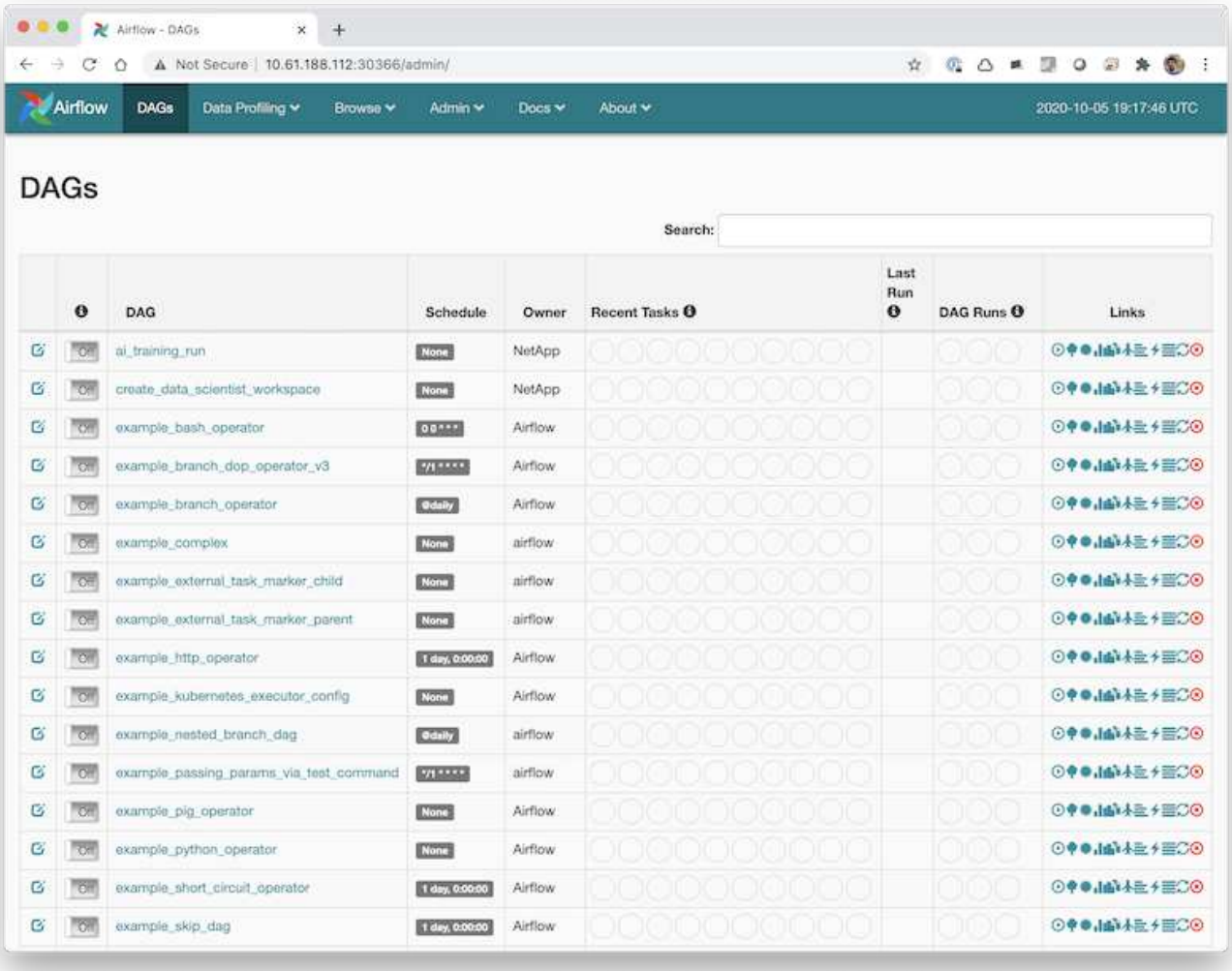
```
$ kubectl -n airflow get pod
NAME                                READY   STATUS    RESTARTS   AGE
airflow-flower-b5656d44f-h8qjk      1/1     Running   0          2h
airflow-postgresql-0                1/1     Running   0          2h
airflow-redis-master-0              1/1     Running   0          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. Ottenere l'URL del servizio Web Airflow seguendo le istruzioni stampate sulla console quando si

implementa Airflow utilizzando Helm nel passaggio 1.

```
$ 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. Verificare che sia possibile accedere al servizio Web Airflow.



Esempio di flussi di lavoro Apache Airflow

Il "NetApp Data Science Toolkit per Kubernetes" Utilizzabile in combinazione con il flusso d'aria. L'utilizzo del NetApp Data Science Toolkit con Airflow consente di incorporare le operazioni di gestione dei dati NetApp in flussi di lavoro automatizzati orchestrati dal flusso d'aria.

Fare riferimento a. "Esempi di flusso d'aria" Sezione all'interno del repository GitHub del NetApp Data Science

Toolkit per informazioni dettagliate sull'utilizzo del toolkit con flusso d'aria.

Esempio di operazioni Trident

Questa sezione include esempi di varie operazioni che è possibile eseguire con Trident.

Importare un volume esistente

Se nel sistema/piattaforma di storage NetApp sono presenti volumi che si desidera montare su container all'interno del cluster Kubernetes, ma che non sono legati ai PVC nel cluster, è necessario importare questi volumi. È possibile utilizzare la funzionalità di importazione dei volumi Trident per importare questi volumi.

I comandi di esempio seguenti mostrano l'importazione dello stesso volume, denominato `pb_fg_all`, Due volte, una per ogni backend Trident creato nell'esempio nella sezione "Esempi di backend Trident per implementazioni ai ONTAP", fase 1. L'importazione dello stesso volume due volte in questo modo consente di montare il volume (un volume FlexGroup esistente) più volte su diverse LIF, come descritto nella sezione "Esempi di backend Trident per implementazioni ai ONTAP", fase 1. Per ulteriori informazioni sui PVC, vedere "Documentazione ufficiale di Kubernetes". Per ulteriori informazioni sulla funzionalità di importazione dei volumi, vedere "Documentazione di Trident".

An `accessModes` valore di `ReadOnlyMany` È specificato nei file delle specifiche PVC di esempio. Per ulteriori informazioni su `accessMode` vedere il campo "Documentazione ufficiale di Kubernetes".

> I nomi di backend specificati nei comandi di importazione di esempio riportati di seguito corrispondono ai backend creati nell'esempio della sezione "Esempi di backend Trident per implementazioni ai ONTAP", fase 1. I nomi StorageClass specificati nei seguenti file di definizione PVC di esempio corrispondono ai StorageClasses creati nell'esempio nella sezione "Esempi di storage Classes Kubernetes per implementazioni ai ONTAP", fase 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
```

Provisioning di un nuovo volume

È possibile utilizzare Trident per eseguire il provisioning di un nuovo volume sul sistema o sulla piattaforma di storage NetApp. I seguenti comandi di esempio mostrano il provisioning di un nuovo volume FlexVol. In questo esempio, il provisioning del volume viene eseguito utilizzando StorageClass creato nell'esempio della sezione "Esempi di storage Classes Kubernetes per implementazioni ai ONTAP", punto 2.

An `accessModes` valore di `ReadWriteMany` Viene specificato nel seguente file di definizione PVC di esempio. Per ulteriori informazioni su `accessMode` vedere il campo "Documentazione ufficiale di Kubernetes".

```
$ 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
```

Esempi di opportunità di lavoro ad alte performance per le implementazioni ai di ONTAP

Questa sezione include esempi di vari job dalle performance elevate che possono essere eseguiti quando Kubernetes viene implementato su un pod ai ONTAP.

Esempi di opportunità di lavoro ad alte performance per le implementazioni ai di ONTAP

Questa sezione include esempi di vari job dalle performance elevate che possono essere eseguiti quando Kubernetes viene implementato su un pod ai ONTAP.

Eseguire un carico di lavoro ai a nodo singolo

Per eseguire un processo ai e ML a nodo singolo nel cluster Kubernetes, eseguire le seguenti operazioni dall'host di distribuzione jump. Con Trident, è possibile rendere un volume di dati, potenzialmente contenente petabyte di dati, accessibile a un carico di lavoro Kubernetes in modo rapido e semplice. Per rendere un volume di dati accessibile

dall'interno di un pod Kubernetes, è sufficiente specificare un PVC nella definizione del pod. Si tratta di un'operazione nativa di Kubernetes, senza richiedere alcuna esperienza NetApp.

> In questa sezione si presuppone che sia già stato containerizzato (nel formato Docker Container) il carico di lavoro ai e ML specifico che si sta tentando di eseguire nel cluster Kubernetes.

1. I seguenti comandi di esempio mostrano la creazione di un lavoro Kubernetes per un carico di lavoro di benchmark TensorFlow che utilizza il dataset ImageNet. Per ulteriori informazioni sul set di dati ImageNet, vedere "Sito Web ImageNet".

   Questo processo di esempio richiede otto GPU e quindi può essere eseguito su un singolo nodo di lavoro GPU che dispone di otto o più GPU. Questo job di esempio potrebbe essere inviato in un cluster per il quale un nodo di lavoro con otto o più GPU non è presente o è attualmente occupato con un altro workload. In tal caso, il lavoro rimane in uno stato in sospeso fino a quando tale nodo di lavoro non diventa disponibile.

   Inoltre, per massimizzare la larghezza di banda dello storage, il volume contenente i dati di training necessari viene montato due volte all'interno del pod creato da questo lavoro. Nel pod è montato anche un altro volume. Questo secondo volume verrà utilizzato per memorizzare risultati e metriche. Questi volumi vengono referenziati nella definizione del lavoro utilizzando i nomi dei PVC. Per ulteriori informazioni sui job Kubernetes, consultare "Documentazione ufficiale di Kubernetes".

   An `emptyDir` volume con a. `medium` valore di `Memory` è montato su `/dev/shm` nel pod creato da questo lavoro di esempio. La dimensione predefinita di `/dev/shm` Il volume virtuale creato automaticamente dal runtime del container Docker può talvolta essere insufficiente per le esigenze di TensorFlow. Montaggio di un `emptyDir` il volume come nell'esempio seguente fornisce un volume sufficientemente grande `/dev/shm` volume virtuale. Per ulteriori informazioni su `emptyDir` volumes (volumi), vedere "Documentazione ufficiale di Kubernetes".

   Al singolo contenitore specificato in questa definizione di lavoro di esempio viene assegnato un `securityContext > privileged` valore di `true`. Questo valore significa che il container dispone effettivamente dell'accesso root sull'host. Questa annotazione viene utilizzata in questo caso perché il carico di lavoro specifico che viene eseguito richiede l'accesso root. In particolare, un'operazione di cancellazione della cache eseguita dal carico di lavoro richiede l'accesso root. Che sia o meno così `privileged: true` l'annotazione è necessaria a seconda dei requisiti del carico di lavoro specifico che si sta eseguendo.

```
$ 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. Verificare che il lavoro creato al punto 1 sia in esecuzione correttamente. Il seguente comando di esempio conferma che è stato creato un singolo pod per il lavoro, come specificato nella definizione del lavoro, e che questo pod è attualmente in esecuzione su uno dei nodi di lavoro GPU.

```
$ 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. Verificare che il lavoro creato al passo 1 sia stato completato correttamente. I seguenti comandi di esempio confermano che il lavoro è stato completato correttamente.

```
$ 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. **Opzionale:** eliminare gli artefatti del lavoro. I seguenti comandi di esempio mostrano l'eliminazione dell'oggetto di lavoro creato al passo 1.

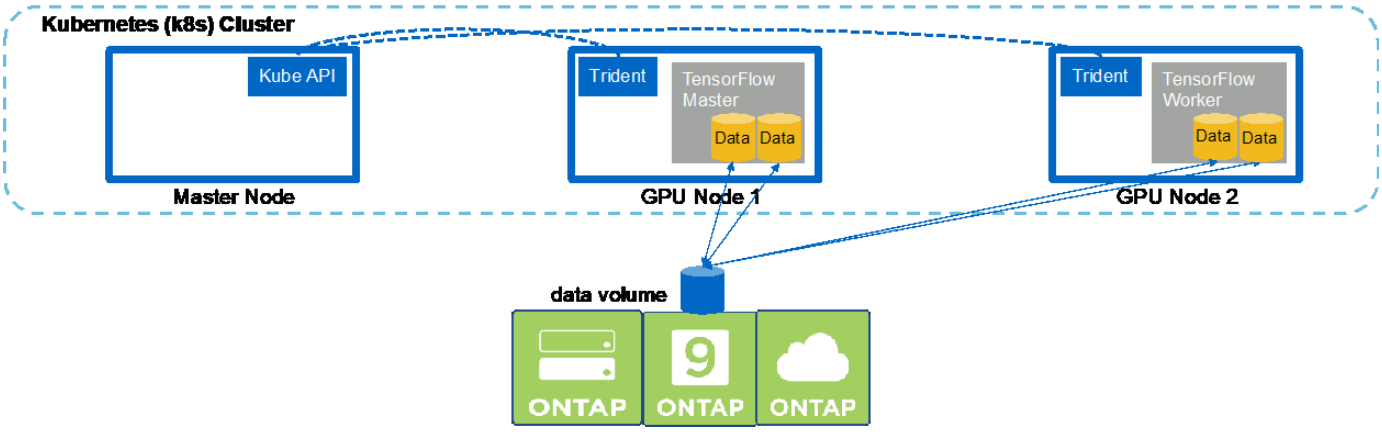
   Quando si elimina l'oggetto di lavoro, Kubernetes elimina automaticamente tutti i pod associati.

```
$ 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.
```

Eseguire un carico di lavoro ai distribuito sincrono

Per eseguire un processo ai e ML multinodo sincrono nel cluster Kubernetes, eseguire le seguenti operazioni sull'host di distribuzione jump. Questo processo consente di sfruttare i dati memorizzati su un volume NetApp e di utilizzare più GPU di quelle che un singolo nodo di lavoro può fornire. Vedere la figura seguente per un'illustrazione di un lavoro di ai distribuito sincrono.

> I lavori distribuiti sincroni possono contribuire ad aumentare la precisione delle performance e della formazione rispetto ai lavori distribuiti asincroni. Una discussione sui pro e contro dei lavori sincroni rispetto ai lavori asincroni non rientra nell'ambito di questo documento.



1. I seguenti comandi di esempio mostrano la creazione di un worker che partecipa all'esecuzione distribuita sincrona dello stesso job di benchmark TensorFlow eseguito su un singolo nodo nell'esempio della sezione "Eseguire un carico di lavoro ai a nodo singolo". In questo esempio specifico, viene implementato solo un singolo worker perché il lavoro viene eseguito su due nodi di lavoro.

In questo esempio, l'implementazione di lavoro richiede otto GPU e può quindi essere eseguita su un singolo nodo di lavoro GPU che dispone di otto o più GPU. Se i nodi di lavoro GPU dispongono di più di otto GPU, per massimizzare le performance, è possibile aumentare questo numero in modo da essere uguale al numero di GPU presenti nei nodi di lavoro. Per ulteriori informazioni sulle implementazioni di Kubernetes, vedere "Documentazione ufficiale di Kubernetes".

In questo esempio viene creata un'implementazione di Kubernetes perché questo specifico lavoratore containerizzato non viene mai completato da solo. Pertanto, non ha senso implementarlo utilizzando il costrutto di lavoro Kubernetes. Se il tuo lavoratore è stato progettato o scritto per essere completato da solo, potrebbe essere opportuno utilizzare il costrutto di lavoro per implementare il tuo lavoratore.

Al pod specificato in questa specifica di implementazione di esempio viene assegnato un `hostNetwork` valore di `true`. Questo valore significa che il pod utilizza lo stack di rete del nodo di lavoro host invece dello stack di rete virtuale creato da Kubernetes per ciascun pod. Questa annotazione viene utilizzata in questo caso perché il carico di lavoro specifico si basa su Open MPI, NCCL e Horovod per eseguire il carico di lavoro in maniera sincrona e distribuita. Pertanto, richiede l'accesso allo stack di rete host. Una discussione su Open MPI, NCCL e Horovod non rientra nell'ambito di questo documento. Che sia o meno così `hostNetwork: true` l'annotazione è necessaria a seconda dei requisiti del carico di lavoro specifico che si sta eseguendo. Per ulteriori informazioni su `hostNetwork` vedere il campo "Documentazione ufficiale di Kubernetes".

```
$ 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. Verificare che l'implementazione worker creata al punto 1 sia stata avviata correttamente. I seguenti comandi di esempio confermano che è stato creato un singolo pod di lavoro per l'implementazione, come indicato nella definizione di implementazione, e che questo pod è attualmente in esecuzione su uno dei nodi di lavoro GPU.

```
$ 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. Creare un lavoro Kubernetes per un master che inizia, partecipa e tiene traccia dell'esecuzione del lavoro sincrono a più nodi. I seguenti comandi di esempio creano un master che inizia, partecipa e tiene traccia dell'esecuzione distribuita sincrona dello stesso job di benchmark TensorFlow eseguito su un singolo nodo

nell'esempio nella sezione "Eseguire un carico di lavoro ai a nodo singolo".

Questo processo master di esempio richiede otto GPU e può quindi essere eseguito su un singolo nodo di lavoro GPU che dispone di otto o più GPU. Se i nodi di lavoro GPU dispongono di più di otto GPU, per massimizzare le performance, è possibile aumentare questo numero in modo da essere uguale al numero di GPU presenti nei nodi di lavoro.

Al pod master specificato in questa definizione di lavoro di esempio viene assegnato un `hostNetwork` valore di `true`, proprio come al pod di lavoro è stato assegnato un `hostNetwork` valore di `true` nella fase 1. Per ulteriori informazioni sul motivo per cui questo valore è necessario, vedere il passaggio 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. Verificare che il lavoro principale creato al punto 3 sia in esecuzione correttamente. Il seguente comando di esempio conferma che è stato creato un singolo pod master per il lavoro, come indicato nella definizione del lavoro, e che questo pod è attualmente in esecuzione su uno dei nodi di lavoro GPU. Inoltre, il pod di lavoro inizialmente visto al punto 1 è ancora in esecuzione e i pod master e di lavoro sono in esecuzione su nodi diversi.

```
$ 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. Verificare che il lavoro principale creato al punto 3 sia stato completato correttamente. I seguenti comandi di esempio confermano che il lavoro è stato completato correttamente.

```
$ 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. Eliminare l'implementazione dei lavoratori quando non è più necessaria. I seguenti comandi di esempio mostrano l'eliminazione dell'oggetto di implementazione worker creato nel passaggio 1.

   Quando si elimina l'oggetto di implementazione worker, Kubernetes elimina automaticamente tutti i worker pod associati.

```
$ 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. **Opzionale:** eliminare gli artefatti del job master. I seguenti comandi di esempio mostrano l'eliminazione dell'oggetto di lavoro master creato nel passaggio 3.

   Quando si elimina l'oggetto di lavoro master, Kubernetes elimina automaticamente tutti i pod master associati.

```
$ 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.
```

Test delle performance

Come parte della creazione di questa soluzione, abbiamo eseguito un semplice confronto delle performance. Utilizzando Kubernetes, abbiamo eseguito diversi processi di benchmarking ai standard di NetApp e abbiamo confrontato i risultati del benchmark con le esecuzioni eseguite utilizzando un semplice comando di esecuzione di Docker. Non sono state riscontrate differenze significative in termini di performance. Pertanto, abbiamo concluso che l'utilizzo di Kubernetes per orchestrare i lavori di training ai containerizzati non influisce negativamente sulle performance. Consulta la tabella seguente per i risultati del nostro confronto delle performance.

Benchmark	Dataset	Esecuzione Docker (immagini/sec)	Kubernetes (immagini/sec)
TensorFlow a nodo singolo	Dati sintetici	6,667.2475	6,661.93125
TensorFlow a nodo singolo	ImageNet	6,570.2025	6,530.59125
Synchronous Distributed Two-Node TensorFlow	Dati sintetici	13,213.70625	13,218.288125
Synchronous Distributed Two-Node TensorFlow	ImageNet	12,941.69125	12,881.33875

Conclusione

Aziende e organizzazioni di tutte le dimensioni e in tutti i settori stanno passando all'intelligenza artificiale (ai), all'apprendimento automatico (ML) e al deep learning (DL) per risolvere problemi reali, offrire prodotti e servizi innovativi e ottenere un vantaggio in un mercato sempre più competitivo. Man mano che le organizzazioni aumentano l'utilizzo di ai, ML e DL, devono affrontare molte sfide, tra cui la scalabilità dei workload e la disponibilità dei dati. Queste sfide possono essere affrontate utilizzando la soluzione NetApp ai Control Plane.

Questa soluzione consente di clonare rapidamente uno spazio dei nomi dei dati. Inoltre, consente di definire e implementare flussi di lavoro di training ai, ML e DL che incorporano la creazione quasi istantanea di dati e linee di base dei modelli per la tracciabilità e il controllo delle versioni. Con questa soluzione, è possibile tracciare ogni singolo modello di training fino ai set di dati esatti con cui il modello è stato addestrato e/o validato. Infine, questa soluzione consente di eseguire rapidamente il provisioning degli spazi di lavoro dei notebook Jupyter con accesso a set di dati di grandi dimensioni.

Poiché questa soluzione è rivolta a data scientist e data engineer, è necessaria una competenza minima di NetApp o NetApp ONTAP. Con questa soluzione, le funzioni di gestione dei dati possono essere eseguite utilizzando interfacce e strumenti semplici e familiari. Inoltre, questa soluzione utilizza componenti completamente open-source e liberi. Pertanto, se disponete già di storage NetApp nel vostro ambiente, potete implementare questa soluzione oggi stesso. Se si desidera provare questa soluzione ma non si dispone già di storage NetApp, visitare il sito "cloud.netapp.com"E potrai essere operativo con una soluzione di storage NetApp basata sul cloud in pochissimo tempo.