MLOps open source con NetApp

NetApp Solutions

NetApp
May 10, 2024

Sommario

MLOps open source con NetApp

MLOps open source con NetApp

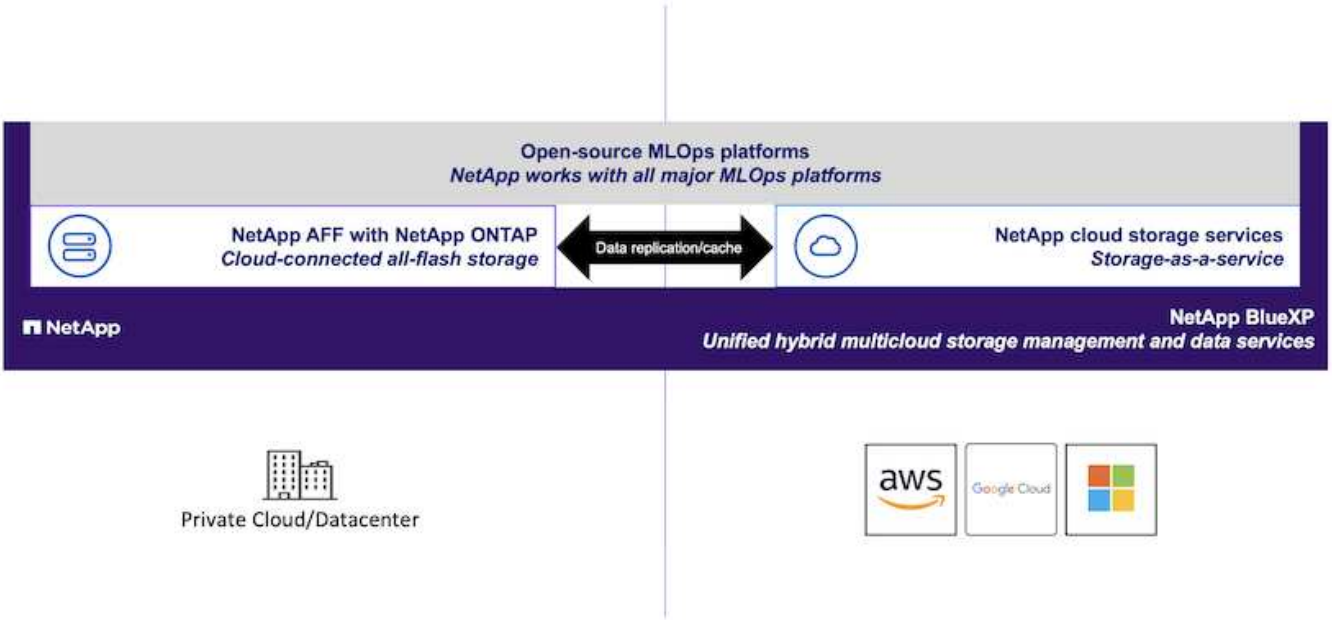
Mike Oglesby, NetApp
Mohan Acharya, NetApp

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. Questa soluzione dimostra come affrontare queste sfide abbinando le funzionalità di gestione dei dati di NetApp ai più diffusi strumenti e framework open-source.

Questa soluzione ha lo scopo di dimostrare diversi strumenti e framework open-source che possono essere incorporati in un flusso di lavoro MLOps. Questi tool e framework diversi possono essere utilizzati insieme o da soli a seconda dei requisiti e dei casi di utilizzo.

In questa soluzione vengono trattati i seguenti strumenti/framework:

- "Flusso d'aria Apache"
- "Kubeflow"



Panoramica sulla tecnologia

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:

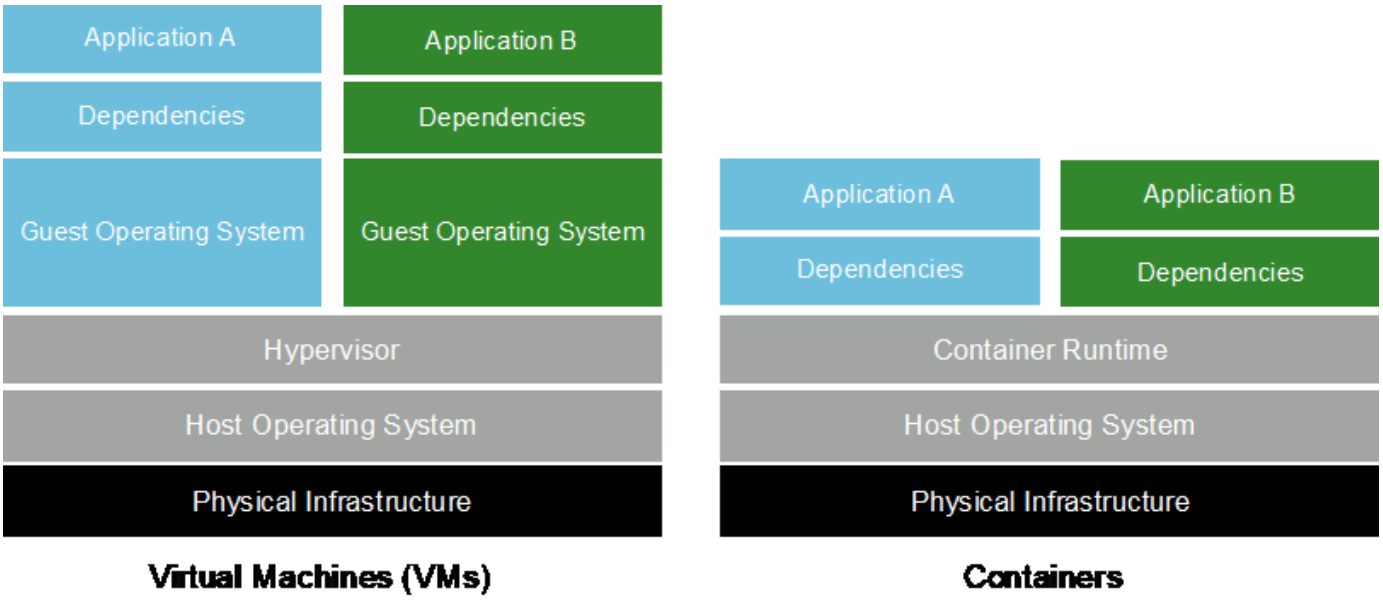
- Analisi di grandi quantità di dati per scoprire informazioni di business precedentemente sconosciute
- Interagire direttamente con i clienti utilizzando l'elaborazione del linguaggio naturale
- Automazione di vari processi e funzioni di business

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. Per

ulteriori informazioni, visitare il "Sito web di Kubernetes".

NetApp Astra Trident

Astra Trident permette il consumo e la gestione delle risorse di storage in tutte le più apprezzate piattaforme di storage NetApp, nel cloud pubblico o on-premise, incluso ONTAP (AFF, FAS, Select, cloud, Amazon FSX per NetApp ONTAP), software Element (NetApp HCI, SolidFire), servizio Azure NetApp Files e Cloud Volumes Service su Google Cloud. Astra Trident è un orchestrator di storage dinamico conforme a Container Storage Interface (CSI) che si integra in modo nativo con Kubernetes.

NetApp DataOps Toolkit

Il "NetApp DataOps Toolkit" È un tool basato su Python che semplifica la gestione di spazi di lavoro di sviluppo/training e server di inferenza che sono supportati da storage NetApp scale-out dalle performance elevate. Le funzionalità principali includono:

- Fornire rapidamente nuovi spazi di lavoro ad alta capacità supportati dallo storage NetApp scale-out dalle performance elevate.
- Clonazione quasi istantanea di spazi di lavoro ad alta capacità per consentire la sperimentazione o l'iterazione rapida.
- Salva quasi istantaneamente le istantanee di spazi di lavoro ad alta capacità per il backup e/o la tracciabilità/baseline.
- Provisioning, cloning e snapshot near-instataneamente di volumi di dati ad alta capacità e performance elevate.

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 astrae le complicazioni di Kubernetes, consentendo ai data scientist di concentrarsi su ciò che sanno meglio — data science. Vedere la figura seguente per una visualizzazione. Kubeflow è una buona opzione open-source per le organizzazioni che preferiscono una piattaforma all-in-one MLOps. 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".

Katib

Katib è un progetto nativo di Kubernetes per il machine learning (AutoML) automatizzato. Katib supporta la sintonizzazione iperparametrica, l'arresto precoce e la ricerca di architetture neurali (NAS). Katib è il progetto indipendente dai framework di machine learning (ML). È in grado di regolare gli iperparametri delle applicazioni scritte in qualsiasi lingua a scelta degli utenti e supporta in modo nativo molti framework ML, come TensorFlow, MXNet, PyTorch, XGBoost, e altri. Katib supporta molti algoritmi AutoML, come l'ottimizzazione Bayesiana, gli stimatori Tree of Parzen, la ricerca casuale, la strategia di evoluzione dell'adattamento della matrice di covarianza, Hyperband, la ricerca di architettura neurale efficiente, la ricerca di architettura differenziabile e molto altro ancora. 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

ONTAP 9, l'ultima generazione di software per la gestione dello storage NetApp, consente alle aziende di modernizzare l'infrastruttura e passare a un data center predisposto per il cloud. Sfruttando le funzionalità di gestione dei dati leader del settore, ONTAP consente la gestione e la protezione dei dati con un singolo set di strumenti, indipendentemente dalla posizione dei dati. Puoi anche spostare liberamente i dati ovunque siano necessari: Edge, core o cloud. ONTAP 9 include numerose funzionalità che semplificano la gestione dei dati, accelerano e proteggono i dati critici e abilitano le funzionalità dell'infrastruttura di nuova generazione nelle architetture di cloud ibrido.

Semplifica la gestione dei dati

La gestione dei dati è fondamentale per le operazioni IT aziendali e per i data scientist, in modo che le risorse appropriate vengano utilizzate per le applicazioni ai e per la formazione dei set di dati ai/ML. Le seguenti informazioni aggiuntive sulle tecnologie NetApp non rientrano nell'ambito di questa convalida, ma potrebbero essere rilevanti a seconda dell'implementazione.

Il software per la gestione dei dati ONTAP include le seguenti funzionalità per ottimizzare e semplificare le operazioni e ridurre il costo totale delle operazioni:

- Compaction 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. Ciò vale per i dati memorizzati localmente e per i dati a più livelli nel cloud.
- Qualità del servizio (AQoS) minima, massima e adattativa. I controlli granulari della qualità del servizio

(QoS) aiutano a mantenere i livelli di performance per le applicazioni critiche in ambienti altamente condivisi.

- NetApp FabricPool. Offre il tiering automatico dei dati cold per le opzioni di cloud storage pubblico e privato, tra cui Amazon Web Services (AWS), Azure e la soluzione di storage NetApp StorageGRID. Per ulteriori informazioni su FabricPool, vedere "TR-4598: Best practice FabricPool".

Accelera e proteggi i dati

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

- Performance e latenza ridotta. ONTAP offre il throughput più elevato possibile con la latenza più bassa possibile.
- Protezione dei dati. ONTAP offre funzionalità di protezione dei dati integrate con gestione comune su tutte le piattaforme.
- NetApp Volume Encryption (NVE). ONTAP offre crittografia nativa a livello di volume con supporto per la gestione delle chiavi sia integrata che esterna.
- Multi-tenancy e autenticazione a più fattori. ONTAP consente la condivisione delle risorse dell'infrastruttura con i massimi livelli di sicurezza.

Infrastruttura a prova di futuro

ONTAP aiuta a soddisfare le esigenze di business esigenti e in continua evoluzione con le seguenti funzionalità:

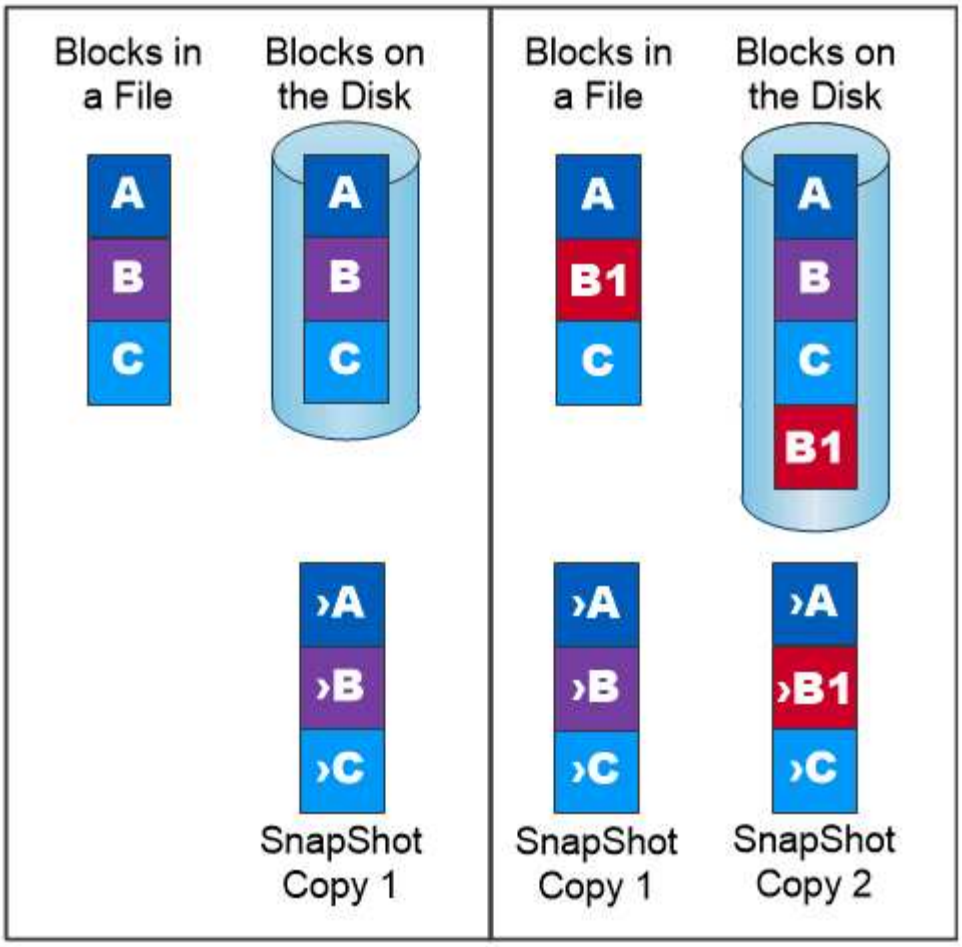
- Scalabilità perfetta e operazioni senza interruzioni. ONTAP supporta l'aggiunta senza interruzioni di capacità ai controller esistenti e ai cluster scale-out. I clienti possono eseguire l'upgrade alle tecnologie più recenti, come NVMe e 32GB FC, senza costose migrazioni dei dati o interruzioni.
- Connessione al cloud. ONTAP è il software per la gestione dello storage più connesso al cloud, con opzioni per storage software-defined e istanze native per il cloud in tutti i cloud pubblici.
- Integrazione con le applicazioni emergenti. ONTAP offre servizi dati di livello Enterprise per piattaforme e applicazioni di prossima generazione, come veicoli autonomi, città intelligenti e industria 4.0, utilizzando la stessa infrastruttura che supporta le applicazioni aziendali esistenti.

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.

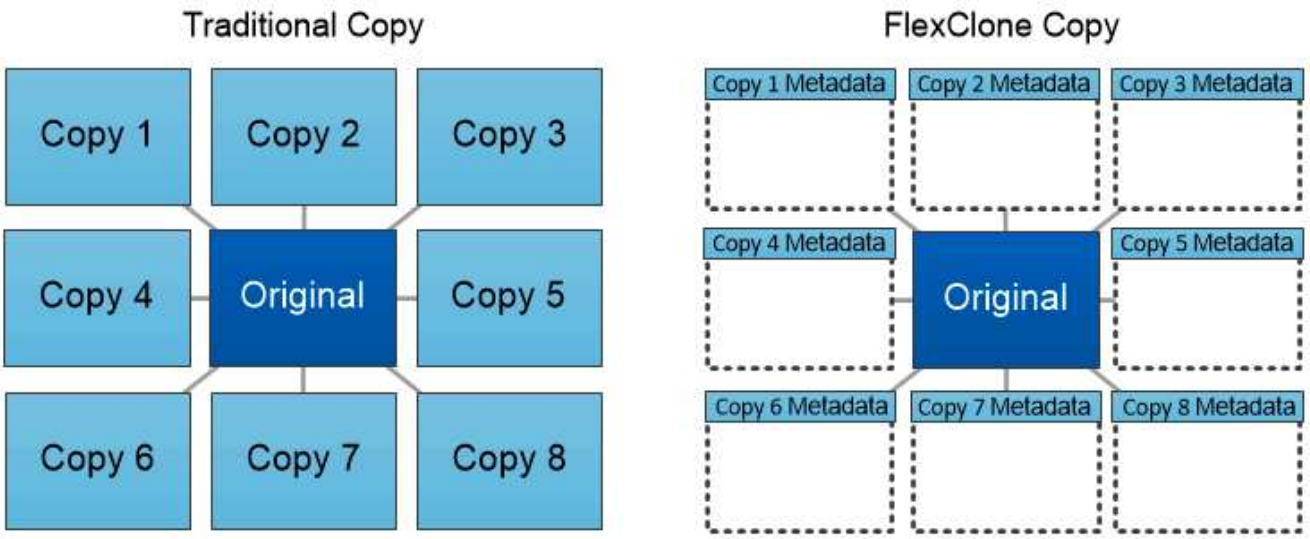
È possibile utilizzare una copia Snapshot per ripristinare singoli file o LUN o per ripristinare l'intero contenuto di un volume. ONTAP confronta le informazioni del puntatore nella copia Snapshot con i dati su disco per ricostruire l'oggetto mancante o danneggiato, senza downtime o costi di performance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

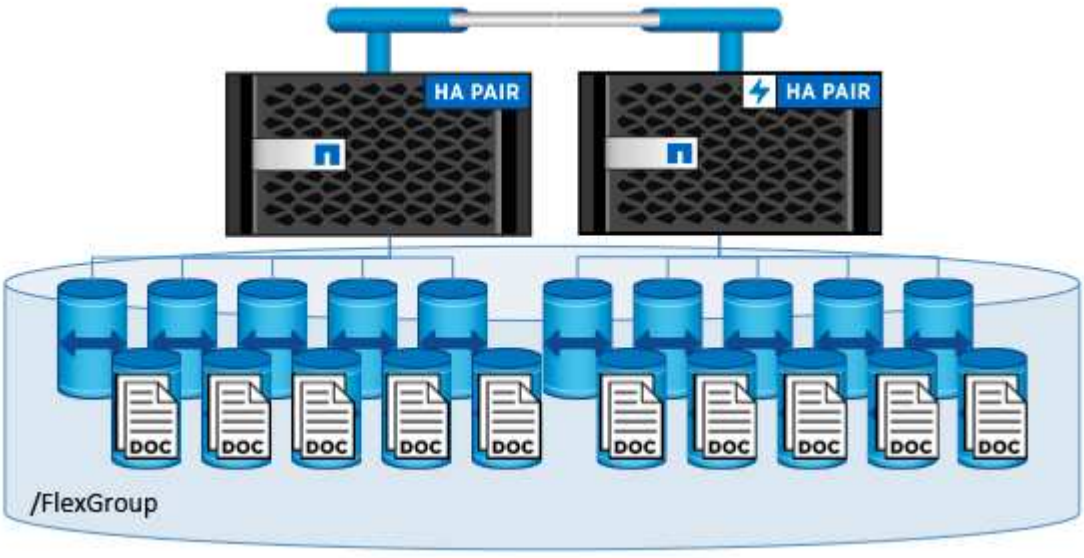
NetApp XCP file Analytics è un software basato su host che rileva le condivisioni di file, esegue scansioni sul file system e fornisce una dashboard per l'analisi dei file. XCP file Analytics è compatibile con sistemi NetApp e non NetApp ed è eseguibile su host Linux o Windows per fornire analisi per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



Architettura

Questa soluzione non dipende da hardware specifico. La soluzione è compatibile con qualsiasi appliance di storage fisico, istanza software-defined o servizio cloud NetApp, supportato da Trident. Esempi: Un sistema storage NetApp AFF, Amazon FSX per NetApp ONTAP, Azure NetApp Files 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 Astra 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 software	Versione
Flusso d'aria Apache	2.0.1
Helm Chart di Apache Airflow	8.0.8
Kubeflow	1,7, implementato tramite "DeployKF" 0.1.1
Kubernetes	1,26
NetApp Astra Trident	23,07

Supporto

NetApp non offre supporto Enterprise per Apache Airflow, Kubeflow o Kubernetes. Se sei interessato a una piattaforma MLOps completamente supportata, "Contatta NetApp" Informazioni sulle soluzioni MLOps completamente supportate offerte da NetApp insieme ai partner.

Configurazione di NetApp Astra Trident

Esempio di backend Astra Trident per le implementazioni di NetApp AIPod

Prima di poter utilizzare Astra Trident per il provisioning dinamico delle risorse di storage all'interno del cluster Kubernetes, devi creare uno o più backend Trident. Gli esempi che seguono rappresentano diversi tipi di backend che è possibile creare se si distribuiscono i componenti di questa soluzione su un "FlexPod NetApp". Per ulteriori informazioni sui backend, consultare "Documentazione di Astra Trident".

1. NetApp consiglia di creare un backend Trident abilitato per FlexGroup per il tuo FlexPod.

   I comandi di esempio che seguono mostrano la creazione di un backend Trident abilitato per FlexGroup per una Storage Virtual Machine (SVM) APod. Questo backend utilizza `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-aipod-flexgroups-iface1.json
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": "aipod-flexgroups-iface1",
    "managementLIF": "10.61.218.100",
    "dataLIF": "192.168.11.11",
    "svm": "ontapai_nfs",
    "username": "admin",
    "password": "ontapai"
}
EOF
$ tridentctl create backend -f ./trident-backend-aipod-flexgroups-
iface1.json -n trident
+------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+---------------------
+--------------------------------------+--------+---------+
| aipod-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+------------------------+---------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+---------------------
+--------------------------------------+--------+---------+
| aipod-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+------------------------+---------------------
+--------------------------------------+--------+---------+
```

2. NetApp consiglia inoltre di creare un backend Trident abilitato per FlexVol. È consigliabile utilizzare FlexVol Volumes per l'hosting di applicazioni persistenti, la memorizzazione dei risultati, l'output, le informazioni di debug e così via. Se si desidera utilizzare i volumi FlexVol, è necessario creare uno o più backend Trident abilitati per FlexVol. Gli esempi di comandi che seguono mostrano la creazione di un singolo backend Trident abilitato per FlexVol.

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

Esempio di Kubernetes StorageClasses per implementazioni di NetApp AIPod

Prima di poter utilizzare Astra Trident per il provisioning dinamico delle risorse di storage all'interno del cluster Kubernetes, devi creare uno o più Kubernetes StorageClasses. Gli esempi che seguono rappresentano diversi tipi di StorageClasses che si potrebbe voler creare se si distribuiscono componenti di questa soluzione su un "FlexPod NetApp". Per ulteriori informazioni su StorageClasses, vedere "Documentazione di Astra Trident".

1. NetApp consiglia di creare uno StorageClass per il back-end Trident abilitato per FlexGroup creato nella sezione "Esempio di backend Astra Trident per le implementazioni di NetApp AIPod", fase 1. I comandi di esempio che seguono mostrano la creazione di più StorageClasses che corrispondono ai due backend di esempio creati nella sezione "Esempio di backend Astra Trident per le implementazioni di NetApp AIPod", passo 1 - uno che utilizza "NFS su RDMA" e uno che non.

   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".

   Nota: Il seguente esempio StorageClasses utilizza una dimensione di trasferimento massima di 262144 GB. Per utilizzare questa dimensione di trasferimento massima, è necessario configurare di conseguenza la dimensione di trasferimento massima sul sistema ONTAP. Fare riferimento a. "Documentazione ONTAP" per ulteriori informazioni.

   Nota: Per utilizzare NFS su RDMA, è necessario configurare NFS su RDMA sul sistema ONTAP. Per ulteriori informazioni, fare riferimento alla documentazione di ONTAP.

   Nota: Nell'esempio seguente, non viene specificato un backend specifico nel campo storagePool nel file di definizione StorageClass.

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

2. NetApp consiglia inoltre di creare un StorageClass che corrisponda al backend Trident abilitato a FlexVol creato nella sezione "Esempio di backend Astra Trident per le implementazioni di AIPod", punto 2. I comandi di esempio che seguono mostrano la creazione di una singola classe di storage per volumi FlexVol.

   Nota: Nell'esempio seguente, non viene specificato un backend particolare nel campo storagePool nel file di definizione StorageClass. Quando utilizzi Kubernetes per amministrare volumi utilizzando questa StorageClass, Trident tenta di utilizzare qualsiasi backend disponibile che utilizza `ontap-nas` driver.

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

Kubeflow

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. Disponi già di un cluster Kubernetes funzionante e stai eseguendo una versione di Kubernetes supportata dalla versione di Kubeflow che intendi implementare. Per un elenco delle versioni di Kubernetes supportate, fare riferimento alle dipendenze per la versione di Kubeflow in "Documentazione ufficiale del Kubeflow".
2. Hai già installato e configurato NetApp Astra Trident nel tuo cluster Kubernetes. Per ulteriori informazioni su Astra Trident, fare riferimento alla "Documentazione di Astra Trident".

Impostare la classe di storage Kubernetes predefinita

Prima di implementare Kubeflow, ti consigliamo di indicare una classe storage predefinita all'interno del cluster Kubernetes. Il processo di implementazione di Kubeflow può tentare di eseguire il provisioning di nuovi volumi persistenti tramite StorageClass predefinito. Se nessuna StorageClass è designata come StorageClass predefinita, l'implementazione potrebbe non riuscire. 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
```

Opzioni di implementazione di Kubeflow

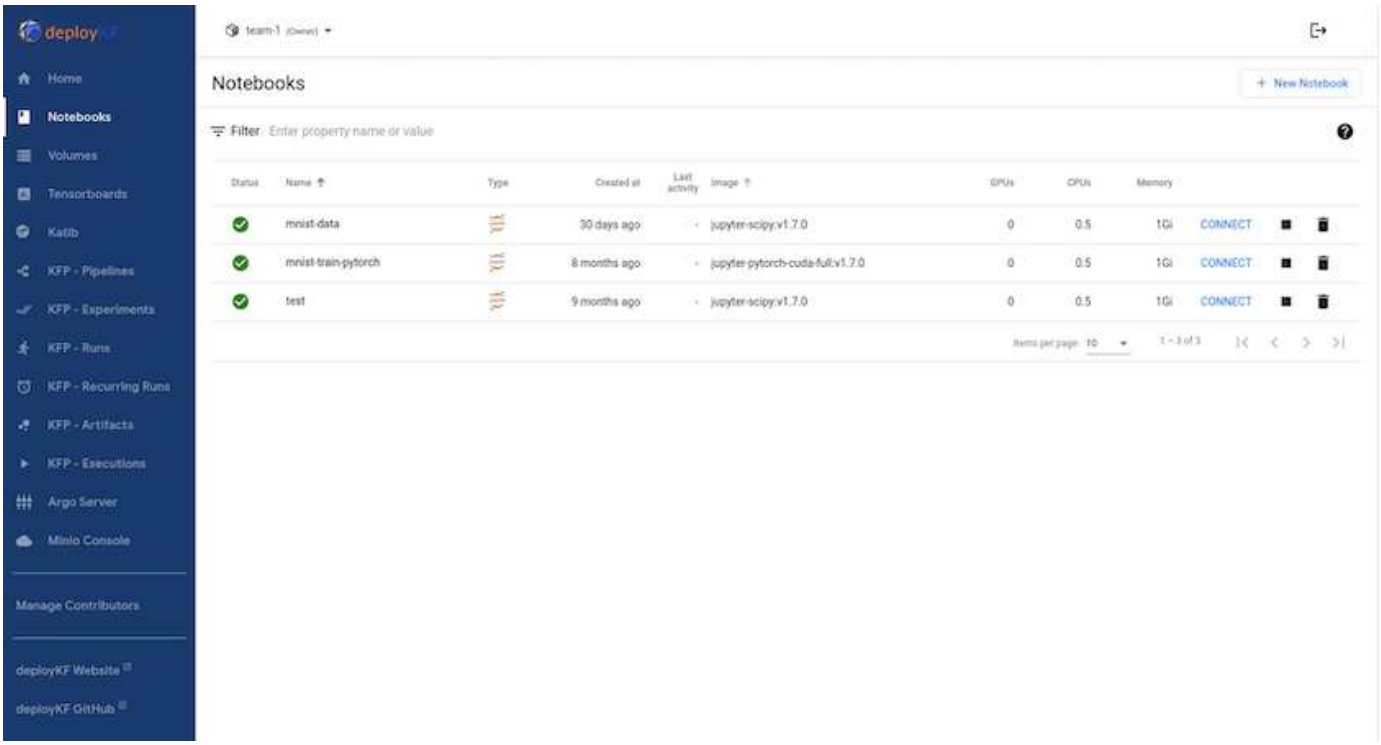
Ci sono molte opzioni diverse per implementare Kubeflow. Fare riferimento a. "Documentazione ufficiale del Kubeflow" per un elenco delle opzioni di distribuzione e scegliere l'opzione più adatta alle proprie esigenze.

> A scopo di convalida, abbiamo implementato Kubeflow 1,7 utilizzando "DeployKF" 0,1.1.

Operazioni e attività Kubeflow di esempio

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 ulteriori informazioni sui notebook Jupyter all'interno del contesto Kubeflow, vedere "Documentazione ufficiale del Kubeflow".



USA il toolkit DataOps di NetApp con Kubeflow

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, come la creazione di snapshot e cloni, direttamente dall'interno di un notebook Jupyter.
- Le operazioni avanzate di gestione dei dati di NetApp, come la creazione di snapshot e cloni, possono essere incorporate nei workflow automatizzati utilizzando il framework Kubeflow Pipelines.

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.

Esempio di flusso di lavoro - Traduci un modello di riconoscimento delle immagini utilizzando Kubeflow e il toolkit DataOps di NetApp

In questa sezione vengono descritte le fasi della formazione e dell'implementazione di una rete neurale per il riconoscimento delle immagini utilizzando Kubeflow e il toolkit NetApp DataOps. Lo scopo è quello di mostrare un lavoro di formazione che incorpora lo storage NetApp.

Prerequisiti

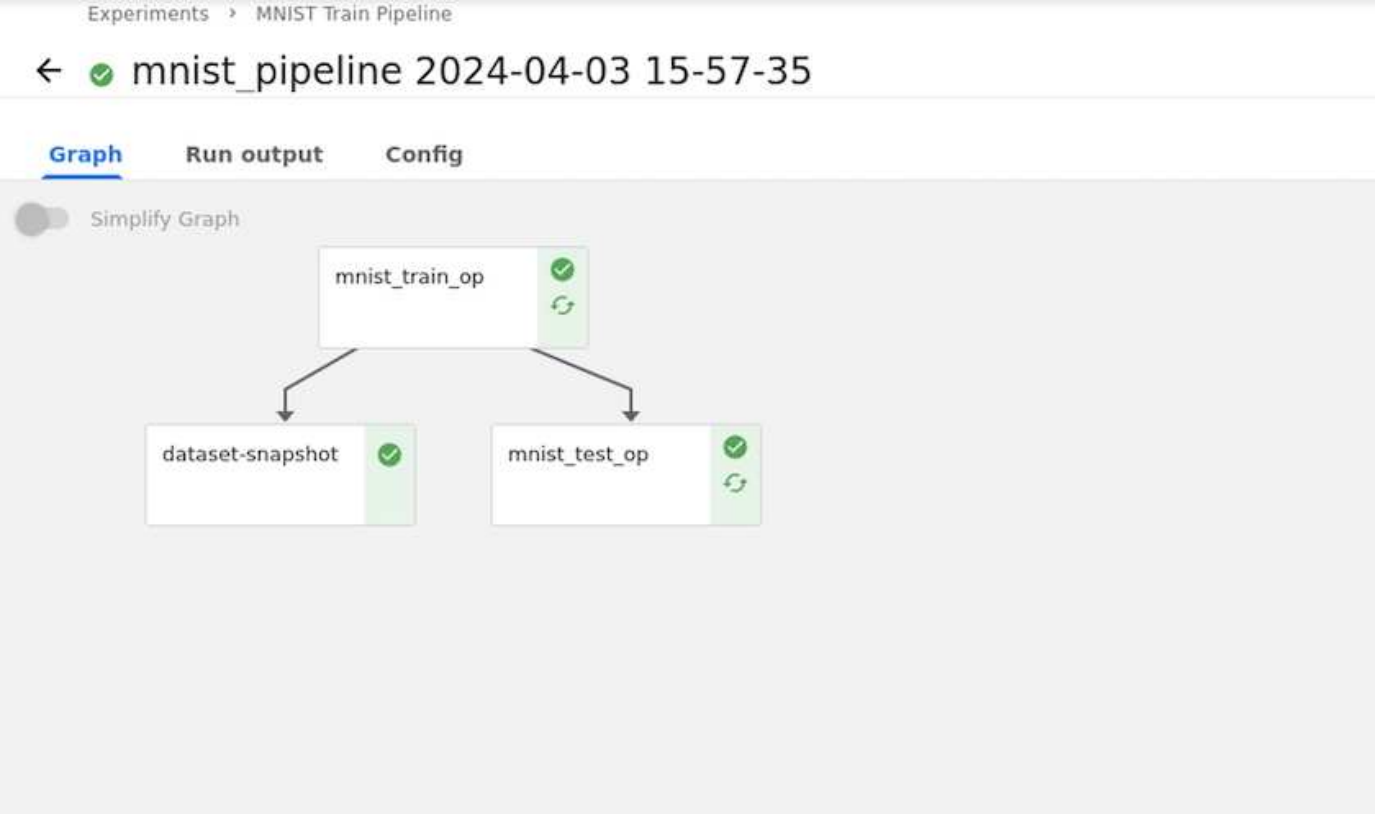
Creare un Dockerfile con le configurazioni necessarie da utilizzare per i passaggi di treno e test all'interno della pipeline Kubeflow.
Ecco un esempio di un Dockerfile -

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

A seconda dei requisiti, installare tutte le librerie e i pacchetti necessari per eseguire il programma. Prima di addestrare il modello di apprendimento automatico, si presuppone che si disponga già di una distribuzione Kubeflow funzionante.

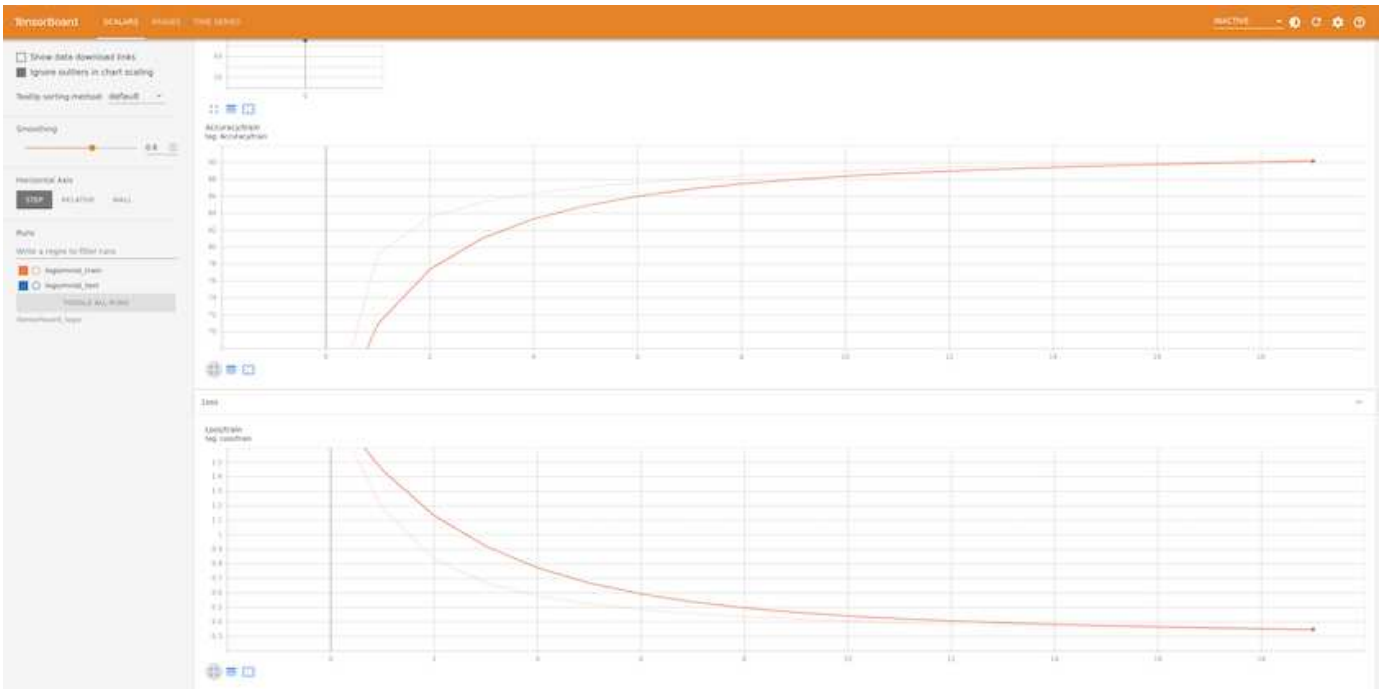
Addestrare un NN piccolo sui dati MNIST utilizzando le tubazioni PyTorch e Kubeflow

Utilizziamo l'esempio di una piccola rete neurale formata su dati MNIST. Il set di dati MNIST è composto da immagini scritte a mano di cifre da 0 a 9. Le immagini sono di dimensioni 28x28 pixel. Il set di dati è diviso in 60.000 immagini del treno e 10.000 immagini di convalida. La rete neurale utilizzata per questo esperimento è una rete feedforward a 2 strati. La formazione viene eseguita utilizzando le pipeline Kubeflow. Consultare la documentazione "qui" per ulteriori informazioni. La nostra pipeline Kubeflow incorpora l'immagine docker della sezione Prerequisiti.



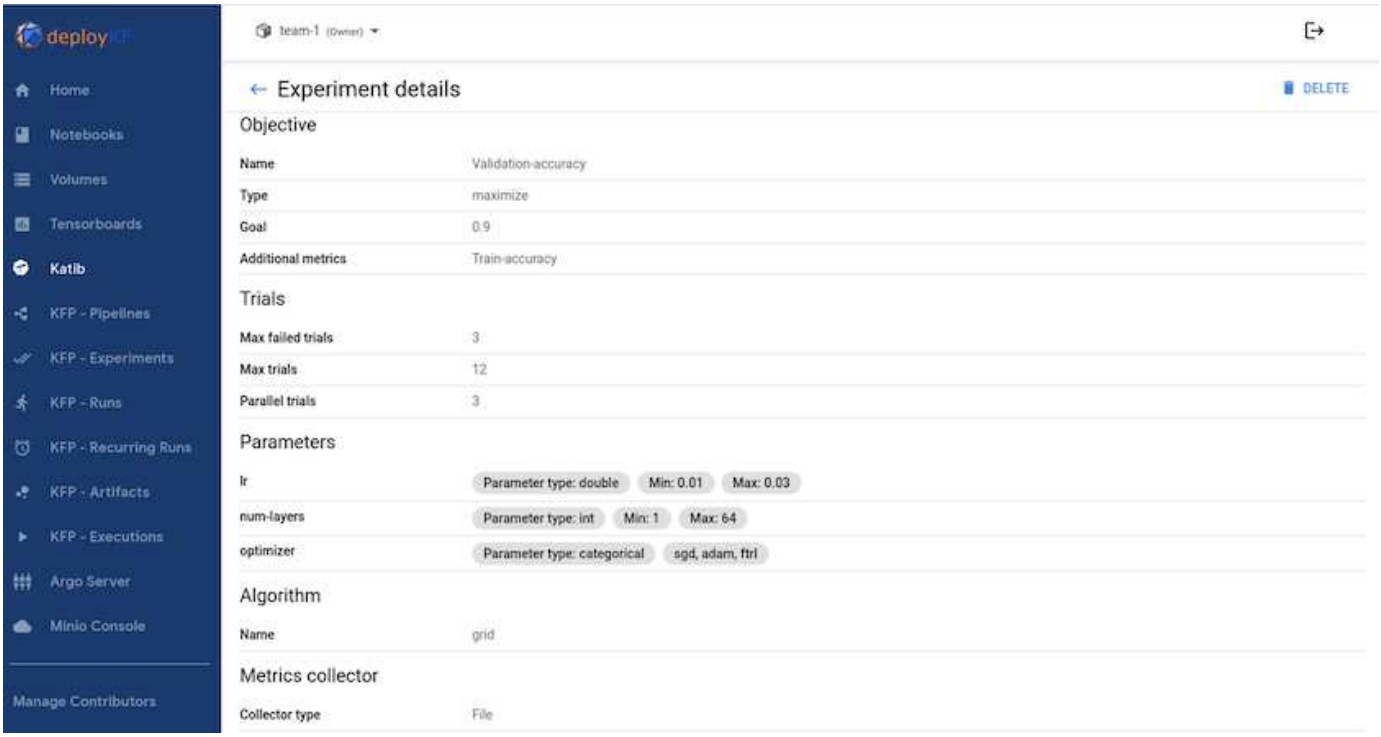
Visualizzare i risultati utilizzando Tensorboard

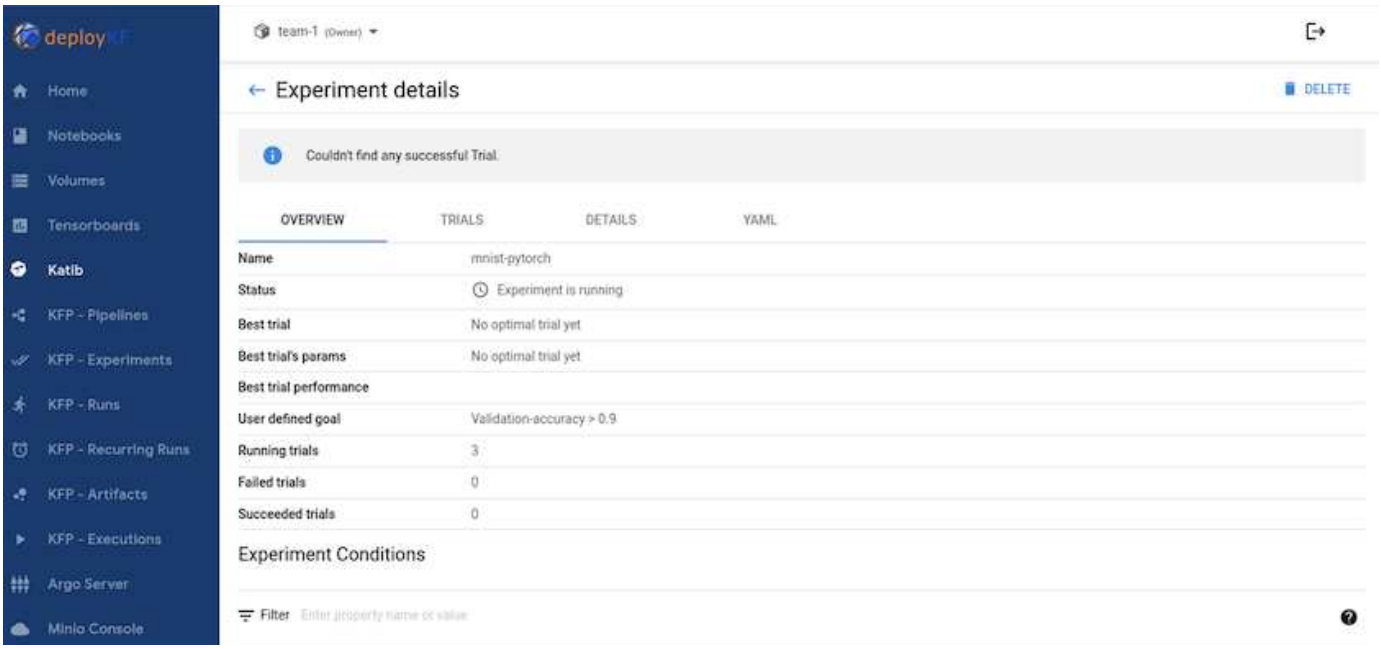
Una volta addestrato il modello, possiamo visualizzare i risultati utilizzando Tensorboard. "Tensorboard" È disponibile come funzione nella dashboard Kubeflow. È possibile creare una scheda tensoriale personalizzata per il proprio lavoro. Un esempio riportato di seguito mostra il grafico della precisione della formazione rispetto al numero di epoche e perdita di formazione vs numero di epoche.



Sperimenta con Hyperparameters usando Katib

"Katib" È uno strumento all'interno di Kubeflow che può essere utilizzato per sperimentare gli iperparametri del modello. Per creare un esperimento, definire prima una metrica/obiettivo desiderato. Questa è solitamente la precisione del test. Una volta definita la metrica, scegliete gli iperparametri con cui volete giocare (optimizer/learning_rate/number of layers). Katib esegue una scansione iperparametrica con i valori definiti dall'utente per trovare la migliore combinazione di parametri che soddisfano la metrica desiderata. È possibile definire questi parametri in ciascuna sezione dell'interfaccia utente. In alternativa, è possibile definire un file **YAML** con le specifiche necessarie. Qui sotto è un'illustrazione di un esperimento di Katib -





Utilizzare le istantanee NetApp per salvare i dati per la tracciabilità

Durante il training sui modelli, potremmo voler salvare un'istantanea del set di dati di training per la tracciabilità. A tale scopo, possiamo aggiungere un passo snapshot alla pipeline, come illustrato di seguito. Per creare l'istantanea, è possibile utilizzare "NetApp DataOps Toolkit per Kubernetes".

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

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

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

Fare riferimento a. "Esempio di toolkit DataOps NetApp per Kubeflow" per ulteriori informazioni.

Flusso d'aria Apache

Implementazione di Apache Airflow

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. Hai già installato e configurato NetApp Astra Trident nel tuo cluster Kubernetes. Per ulteriori informazioni su Astra Trident, fare riferimento alla "Documentazione di Astra 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, segui le istruzioni riportate nella "Implementazione di Kubeflow" sezione. 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
```

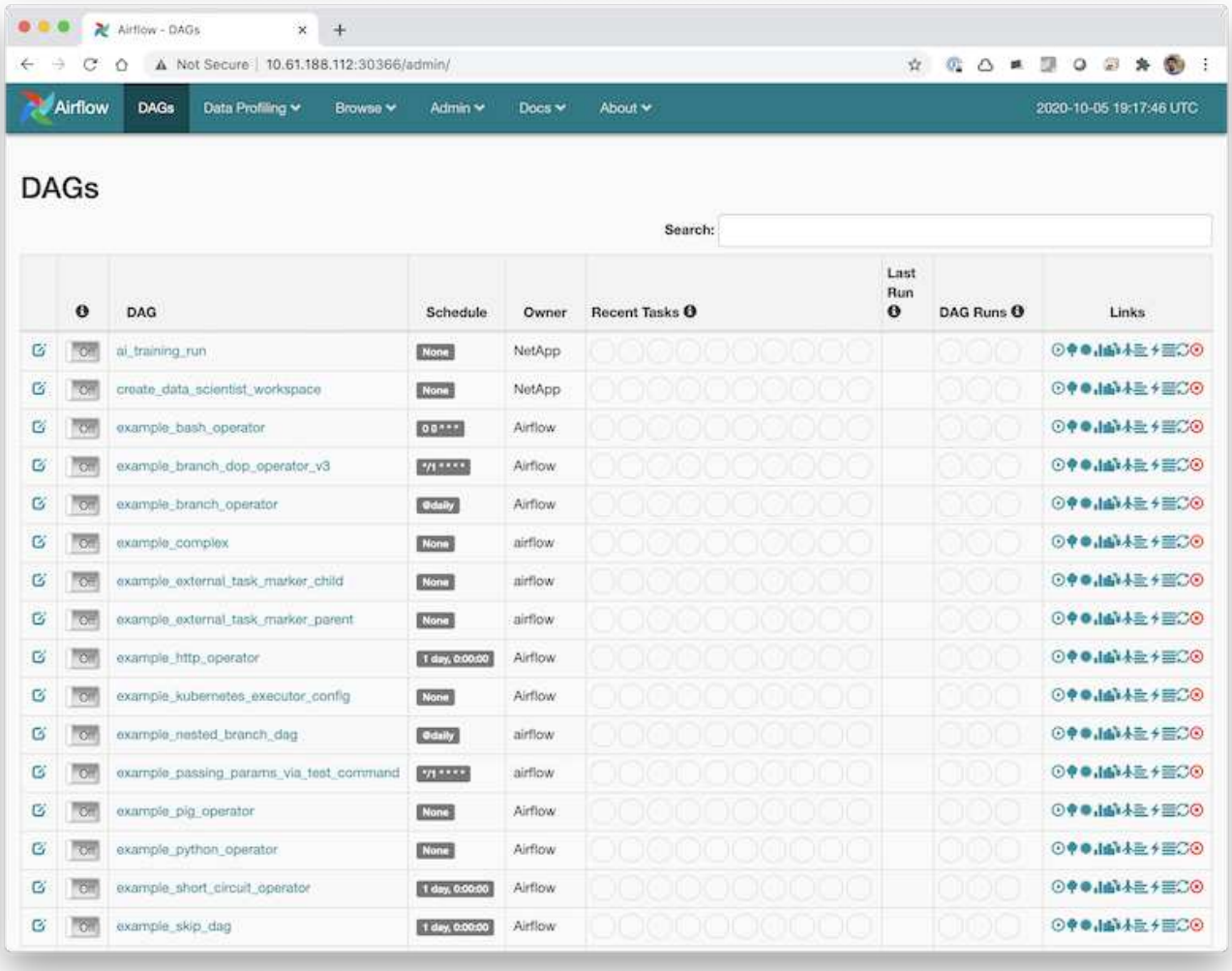
2. Verificare che tutti i pod del flusso d'aria siano in funzione. L'avvio di tutti i pod potrebbe richiedere alcuni minuti.

```
$ 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.



USA il toolkit DataOps di NetApp con Airflow

Il "NetApp DataOps Toolkit per Kubernetes" Utilizzabile in combinazione con il flusso d'aria. L'utilizzo del toolkit NetApp DataOps con flusso d'aria consente di integrare le operazioni di gestione dei dati NetApp, come la creazione di snapshot e cloni, in workflow automatizzati orchestrati da flussi d'aria.

Fare riferimento a. "Esempi di flusso d'aria" Sezione all'interno del repository GitHub del toolkit DataOps di NetApp per i dettagli sull'utilizzo del toolkit con flusso d'aria.

Esempio di operazioni Astra Trident

Questa sezione include esempi di varie operazioni che potresti eseguire con Astra 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.

Gli esempi di comandi che seguono mostrano l'importazione di un volume denominato `pb_fg_all`. 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".

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

```
iface1 | file     | b74cbddb-e0b8-40b7-b263-b6da6dec0bdd | online | true
|
+-----------------------------------+---------
+-----------------------------------+----------
+---------------------------------------+--------+---------+
$ kubectl get pvc
NAME                  STATUS   VOLUME                                  CAPACITY
ACCESS MODES   STORAGECLASS                        AGE
pb-fg-all-iface1      Bound    default-pb-fg-all-iface1-7d9f1
10995116277760   ROX            ontap-ai-flexgroups-retain-iface1   25h
```

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.

Eseguire il provisioning di un nuovo volume utilizzando kubectl

I seguenti comandi di esempio mostrano il provisioning di un nuovo volume FlexVol utilizzando kubectl.

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

Provisioning di un nuovo volume con il toolkit NetApp DataOps

Puoi anche utilizzare il toolkit NetApp DataOps per Kubernetes per il provisioning di un nuovo volume sul sistema storage o sulla piattaforma NetApp. NetApp DataOps Toolkit for Kubernetes utilizza Trident per eseguire il provisioning dei volumi ma semplifica il processo per l'utente. Fare riferimento a. "documentazione" per ulteriori informazioni.

Esempi di processi ad alte prestazioni per le implementazioni di AIPod

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.

> 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.

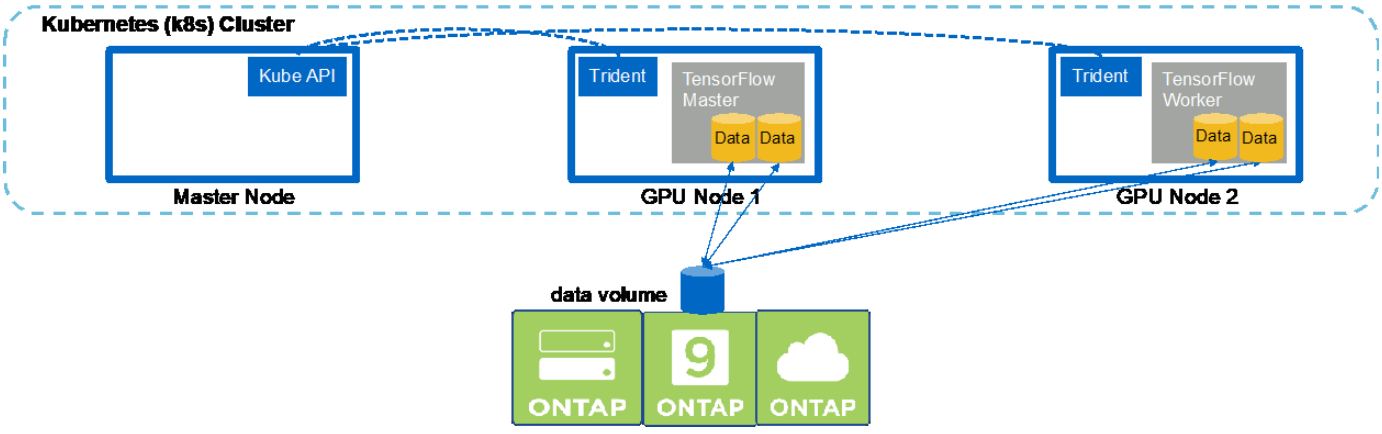
   Quando si elimina l'oggetto di lavoro, Kubernetes elimina automaticamente tutti i pod associati.

```
$ 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.
```

Informazioni sul copyright

Informazioni sul marchio commerciale