MLOps open source con NetApp

NetApp artificial intelligence solutions

NetApp
December 04, 2025

Sommario

MLOps open source con NetApp

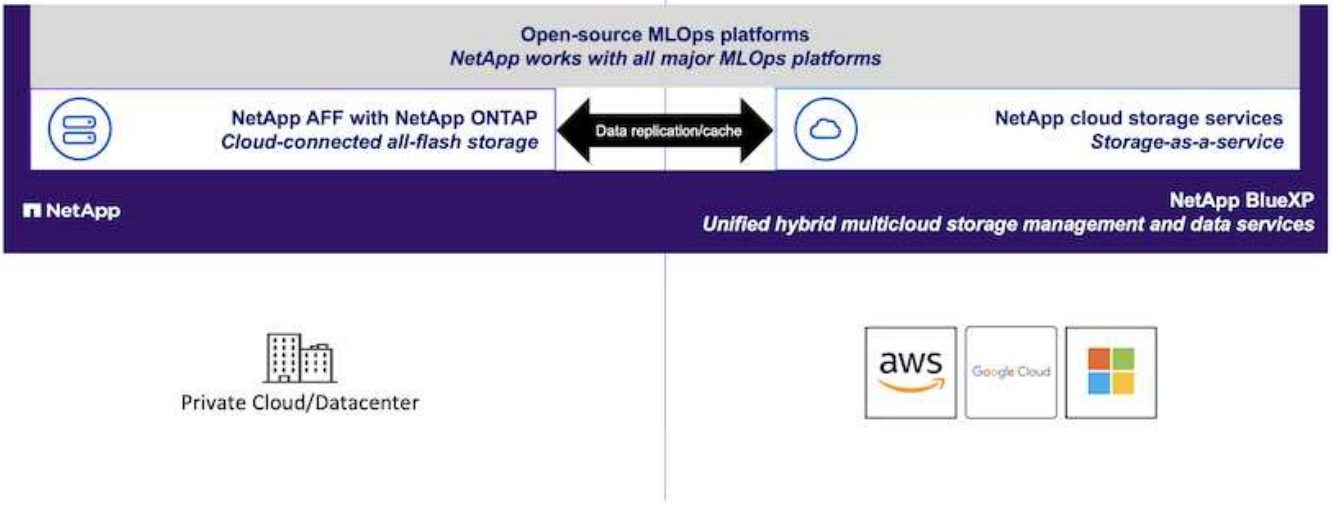
MLOps open source con NetApp

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

Aziende e organizzazioni di tutte le dimensioni e provenienti da numerosi settori si stanno rivolgendo all'intelligenza artificiale (IA) per risolvere problemi del mondo reale, offrire prodotti e servizi innovativi e ottenere un vantaggio in un mercato sempre più competitivo. Molte organizzazioni si stanno rivolgendo agli strumenti MLOps open source per stare al passo con il rapido ritmo dell'innovazione nel settore. Questi strumenti open source offrono funzionalità avanzate e all'avanguardia, ma spesso non tengono conto della disponibilità e della sicurezza dei dati. Sfortunatamente, ciò significa che gli scienziati dei dati altamente qualificati sono costretti a trascorrere molto tempo in attesa di accedere ai dati o di completare le operazioni rudimentali relative ai dati. Abbinando i popolari strumenti MLOps open source a un'infrastruttura dati intelligente di NetApp, le organizzazioni possono accelerare i propri pipeline di dati, che a loro volta accelerano le loro iniziative di intelligenza artificiale. Possono liberare il valore dei propri dati garantendone al contempo la protezione e la sicurezza. Questa soluzione dimostra l'abbinamento delle funzionalità di gestione dei dati NetApp con diversi strumenti e framework open source diffusi per affrontare queste sfide.

L'elenco seguente evidenzia alcune delle funzionalità chiave abilitate da questa soluzione:

- Gli utenti possono rapidamente fornire nuovi volumi di dati ad alta capacità e spazi di lavoro di sviluppo supportati da storage NetApp scalabile e ad alte prestazioni.
- Gli utenti possono clonare quasi istantaneamente volumi di dati ad alta capacità e spazi di sviluppo per consentire la sperimentazione o l'iterazione rapida.
- Gli utenti possono salvare quasi istantaneamente snapshot di volumi di dati ad alta capacità e di aree di lavoro di sviluppo per il backup e/o la tracciabilità/baselining.



Un tipico flusso di lavoro MLOps incorpora spazi di lavoro di sviluppo, solitamente sotto forma di"Quaderni Jupyter" ; monitoraggio degli esperimenti; pipeline di formazione automatizzate; pipeline di dati; e inferenza/distribuzione. Questa soluzione mette in evidenza diversi strumenti e framework che possono essere utilizzati in modo indipendente o combinato per affrontare i diversi aspetti del flusso di lavoro. Mostreremo inoltre l'abbinamento delle funzionalità di gestione dei dati NetApp con ciascuno di questi strumenti. Questa soluzione è pensata per offrire elementi costitutivi da cui un'organizzazione può costruire un flusso di lavoro MLOps personalizzato, specifico per i propri casi d'uso e requisiti.

Questa soluzione comprende i seguenti strumenti/framework:

- "Apache Airflow"
- "JupyterHub"
- "Kubeflow"
- "Flusso ML"

L'elenco seguente descrive i modelli comuni per l'implementazione di questi strumenti in modo indipendente o combinato.

- Distribuisci JupyterHub, MLflow e Apache Airflow insieme - JupyterHub per"Quaderni Jupyter" , MLflow per il monitoraggio degli esperimenti e Apache Airflow per la formazione automatizzata e le pipeline di dati.
- Distribuisci Kubeflow e Apache Airflow insieme - Kubeflow per"Quaderni Jupyter" , monitoraggio degli esperimenti, pipeline di formazione automatizzate e inferenza; e Apache Airflow per pipeline di dati.
- Distribuisci Kubeflow come soluzione di piattaforma MLOps all-in-one per"Quaderni Jupyter" , monitoraggio degli esperimenti, formazione automatizzata e pipeline di dati e inferenza.

Panoramica della tecnologia

Questa sezione si concentra sulla panoramica tecnologica per OpenSource MLOps con NetApp.

Intelligenza artificiale

L'intelligenza artificiale è una disciplina informatica in cui i computer vengono addestrati a imitare le funzioni cognitive della mente umana. Gli sviluppatori di intelligenza artificiale addestrano i computer ad apprendere e a risolvere i problemi in un modo simile, o addirittura superiore, a quello degli esseri umani. L'apprendimento profondo e l'apprendimento automatico sono sottocampi dell'intelligenza artificiale. Le organizzazioni stanno adottando sempre più intelligenza artificiale, apprendimento automatico e apprendimento automatico (DL) per supportare le loro esigenze aziendali critiche. Ecco alcuni esempi:

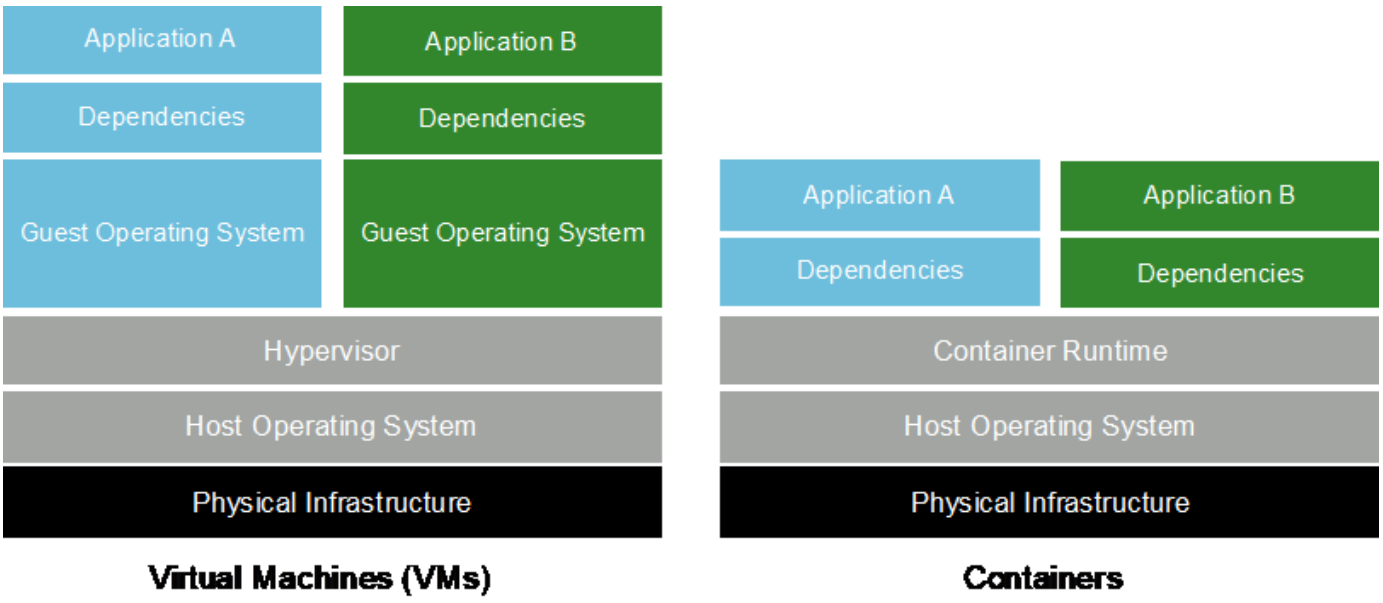
- Analizzare grandi quantità di dati per scoprire informazioni aziendali precedentemente sconosciute
- Interagire direttamente con i clienti utilizzando l'elaborazione del linguaggio naturale
- Automazione di vari processi e funzioni aziendali

I moderni carichi di lavoro di addestramento e inferenza dell'intelligenza artificiale richiedono capacità di elaborazione parallela estremamente elevate. Per questo motivo, le GPU vengono sempre più utilizzate per eseguire operazioni di intelligenza artificiale, perché le capacità di elaborazione parallela delle GPU sono di gran lunga superiori a quelle delle CPU generiche.

Contenitori

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

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



Kubernetes

Kubernetes è una piattaforma di orchestrazione di container distribuita e open source, originariamente progettata da Google e ora gestita dalla Cloud Native Computing Foundation (CNCF). Kubernetes consente l'automazione delle funzioni di distribuzione, gestione e ridimensionamento per le applicazioni containerizzate. Negli ultimi anni, Kubernetes si è affermato come la piattaforma di orchestrazione dei container dominante. Per maggiori informazioni, visita il "Sito web di Kubernetes" .

NetApp Trident

"Trident"consente l'utilizzo e la gestione delle risorse di storage su tutte le piattaforme di storage NetApp più diffuse, nel cloud pubblico o in locale, tra cui ONTAP (AFF, FAS, Select, Cloud, Amazon FSx ONTAP), il servizio Azure NetApp Files e Google Cloud NetApp Volumes. Trident è un orchestratore di storage dinamico compatibile con Container Storage Interface (CSI) che si integra in modo nativo con Kubernetes.

Kit di strumenti NetApp DataOps

IL"Kit di strumenti NetApp DataOps" è uno strumento basato su Python che semplifica la gestione degli spazi di lavoro di sviluppo/formazione e dei server di inferenza supportati da storage NetApp ad alte prestazioni e scalabile. Le principali funzionalità includono:

- Fornisci rapidamente nuovi spazi di lavoro ad alta capacità supportati da storage NetApp scalabile e ad alte prestazioni.
- Clonare quasi istantaneamente spazi di lavoro ad alta capacità per consentire la sperimentazione o l'iterazione rapida.
- Salvataggio quasi istantaneo di snapshot di spazi di lavoro ad alta capacità per backup e/o tracciabilità/baselining.
- Esegui il provisioning, la clonazione e l'istantanea di volumi di dati ad alta capacità e ad alte prestazioni in modo quasi istantaneo.

Apache Airflow

Apache Airflow è una piattaforma open source per la gestione dei flussi di lavoro che consente la creazione, la pianificazione e il monitoraggio programmatici di flussi di lavoro aziendali complessi. Viene spesso utilizzato per automatizzare i flussi di lavoro ETL e di pipeline di dati, ma non si limita a questi tipi di flussi di lavoro. Il progetto Airflow è stato avviato da Airbnb, ma nel frattempo è diventato molto popolare nel settore e ora rientra sotto l'egida della Apache Software Foundation. Airflow è scritto in Python, i flussi di lavoro di Airflow vengono creati tramite script Python e Airflow è progettato secondo il principio della "configurazione come codice". Molti utenti aziendali di Airflow ora eseguono Airflow su Kubernetes.

Grafi aciclici diretti (DAG)

In Airflow, i flussi di lavoro sono chiamati grafi aciclici diretti (DAG). I DAG sono costituiti da attività eseguite in sequenza, in parallelo o con una combinazione delle due, a seconda della definizione del DAG. Lo scheduler Airflow esegue singole attività su una serie di worker, rispettando le dipendenze a livello di attività specificate nella definizione del DAG. I DAG vengono definiti e creati tramite script Python.

Quaderno di Jupyter

I Jupyter Notebook sono documenti simili a wiki che contengono codice live e testo descrittivo. I notebook Jupyter sono ampiamente utilizzati nella comunità AI e ML come mezzo per documentare, archiviare e condividere progetti AI e ML. Per maggiori informazioni sui Jupyter Notebooks, visita il "Sito web di Jupyter" .

Server Jupyter Notebook

Un server Jupyter Notebook è un'applicazione web open source che consente agli utenti di creare Jupyter Notebook.

JupyterHub

JupyterHub è un'applicazione multiutente che consente a un singolo utente di effettuare il provisioning e accedere al proprio server Jupyter Notebook. Per maggiori informazioni su JupyterHub, visita il "Sito web JupyterHub" .

Flusso ML

MLflow è una popolare piattaforma open source per la gestione del ciclo di vita dell'intelligenza artificiale. Le caratteristiche principali di MLflow includono il monitoraggio degli esperimenti AI/ML e un repository di modelli AI/ML. Per maggiori informazioni su MLflow, visita il "Sito web MLflow" .

Kubeflow

Kubeflow è un toolkit open source di intelligenza artificiale e apprendimento automatico per Kubernetes, originariamente sviluppato da Google. Il progetto Kubeflow rende le distribuzioni di flussi di lavoro di intelligenza artificiale e apprendimento automatico su Kubernetes semplici, portabili e scalabili. Kubeflow astrae le complessità di Kubernetes, consentendo agli scienziati dei dati di concentrarsi su ciò che conoscono meglio: la scienza dei dati. Per una visualizzazione, vedere la figura seguente. Kubeflow è una buona opzione open source per le organizzazioni che preferiscono una piattaforma MLOps all-in-one. Per maggiori informazioni, visita il "Sito web di Kubeflow" .

Pipeline Kubeflow

Le pipeline di 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 AI e ML portatili e scalabili. Per maggiori informazioni, vedere il "documentazione ufficiale di Kubeflow" .

Notebook Kubeflow

Kubeflow semplifica il provisioning e la distribuzione dei server Jupyter Notebook su Kubernetes. Per ulteriori informazioni sui Jupyter Notebook nel contesto di Kubeflow, vedere "documentazione ufficiale di Kubeflow" .

Katib

Katib è un progetto nativo di Kubernetes per l'apprendimento automatico automatizzato (AutoML). Katib supporta la sintonizzazione degli iperparametri, l'arresto anticipato e la ricerca dell'architettura neurale (NAS). Katib è un progetto indipendente dai framework di apprendimento automatico (ML). Può ottimizzare gli iperparametri delle applicazioni scritte in qualsiasi linguaggio scelto dall'utente e supporta in modo nativo molti framework ML, come TensorFlow, MXNet, PyTorch, XGBoost e altri. Katib supporta molti algoritmi AutoML diversi, come l'ottimizzazione bayesiana, gli stimatori dell'albero di Parzen, la ricerca casuale, la strategia di evoluzione dell'adattamento della matrice di covarianza, l'iperbanda, la ricerca efficiente dell'architettura neurale, la ricerca dell'architettura differenziabile e molti altri. Per ulteriori informazioni sui Jupyter Notebook nel contesto di Kubeflow, vedere "documentazione ufficiale di Kubeflow" .

NetApp ONTAP

ONTAP 9, l'ultima generazione di software di gestione dello storage di NetApp, consente alle aziende di modernizzare l'infrastruttura e passare a un data center pronto per il cloud. Sfruttando le funzionalità di

gestione dei dati leader del settore, ONTAP consente la gestione e la protezione dei dati con un unico set di strumenti, indipendentemente da dove risiedano. È inoltre possibile 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 infrastrutturali di nuova generazione nelle architetture cloud ibride.

Semplificare la gestione dei dati

La gestione dei dati è fondamentale per le operazioni IT aziendali e per gli scienziati dei dati, in modo che vengano utilizzate risorse appropriate per le applicazioni di intelligenza artificiale e per la formazione di set di dati di intelligenza artificiale/apprendimento automatico. Le seguenti informazioni aggiuntive sulle tecnologie NetApp esulano dall'ambito di questa convalida, ma potrebbero essere rilevanti a seconda della distribuzione.

Il software di gestione dati ONTAP include le seguenti funzionalità per semplificare e snellire le operazioni e ridurre i costi operativi totali:

- Compattazione dei dati in linea e deduplicazione estesa. La compattazione dei dati riduce lo spazio sprecato all'interno dei blocchi di archiviazione, mentre la deduplicazione aumenta significativamente la capacità effettiva. Ciò vale sia per i dati archiviati localmente sia per i dati archiviati a livelli nel cloud.
- Qualità del servizio minima, massima e adattiva (AQoS). I controlli granulari della qualità del servizio (QoS) aiutano a mantenere i livelli di prestazioni per le applicazioni critiche in ambienti altamente condivisi.
- NetApp FabricPool. Fornisce la suddivisione automatica dei dati inattivi in opzioni di archiviazione cloud pubbliche e private, tra cui Amazon Web Services (AWS), Azure e la soluzione di archiviazione NetApp StorageGRID . Per ulteriori informazioni su FabricPool, vedere "TR-4598: Buone pratiche FabricPool" .

Accelerare e proteggere i dati

ONTAP garantisce livelli superiori di prestazioni e protezione dei dati ed estende queste capacità nei seguenti modi:

- Prestazioni e latenza più bassa. ONTAP offre la massima capacità di trasmissione possibile con la minima latenza possibile.
- Protezione dei dati. ONTAP offre funzionalità integrate di protezione dei dati con gestione comune su tutte le piattaforme.
- Crittografia del volume NetApp (NVE). ONTAP offre la crittografia nativa a livello di volume con supporto sia per la gestione delle chiavi integrate che per quella esterna.
- Multitenancy e autenticazione multifattore. ONTAP consente la condivisione delle risorse infrastrutturali con i massimi livelli di sicurezza.

Infrastruttura a prova di futuro

ONTAP aiuta a soddisfare le esigenze aziendali più esigenti e in continua evoluzione grazie alle seguenti funzionalità:

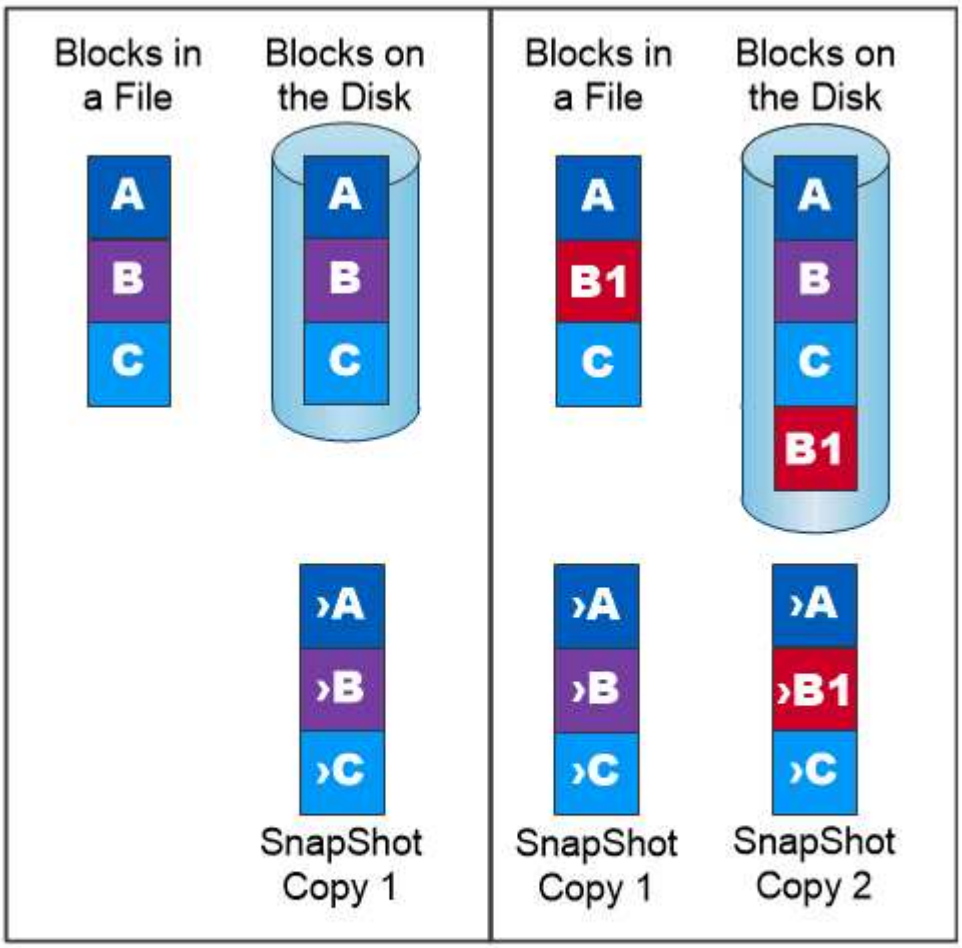
- Scalabilità fluida e operazioni senza interruzioni. ONTAP supporta l'aggiunta non distruttiva di capacità ai controller esistenti e ai cluster scalabili. I clienti possono passare alle tecnologie più recenti senza costose migrazioni di dati o interruzioni.
- Connessione cloud. ONTAP è il software di gestione dello storage più connesso al cloud, con opzioni per lo storage definito dal software e istanze cloud native in tutti i cloud pubblici.
- Integrazione con applicazioni emergenti. ONTAP offre servizi dati di livello aziendale per piattaforme e applicazioni di nuova generazione, come veicoli autonomi, città intelligenti e Industria 4.0, utilizzando la stessa infrastruttura che supporta le app aziendali esistenti.

Copie snapshot NetApp

Una copia Snapshot NetApp è un'immagine di un volume, di sola lettura e in un determinato momento. L'immagine consuma uno spazio di archiviazione minimo e comporta un sovraccarico di prestazioni trascurabile, poiché registra solo le modifiche apportate ai file creati dopo l'ultima copia Snapshot, come illustrato nella figura seguente.

Le copie snapshot devono la loro efficienza alla tecnologia di virtualizzazione dello storage ONTAP di base, Write Anywhere File Layout (WAFL). Come un database, WAFL utilizza i metadati per puntare ai 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 modifica i metadati. Le copie Snapshot sono così efficienti perché ONTAP fa riferimento ai metadati quando crea una copia Snapshot, anziché copiare blocchi di dati. In questo modo si elimina il tempo di ricerca impiegato da altri sistemi per individuare i blocchi da copiare, nonché il costo della copia stessa.

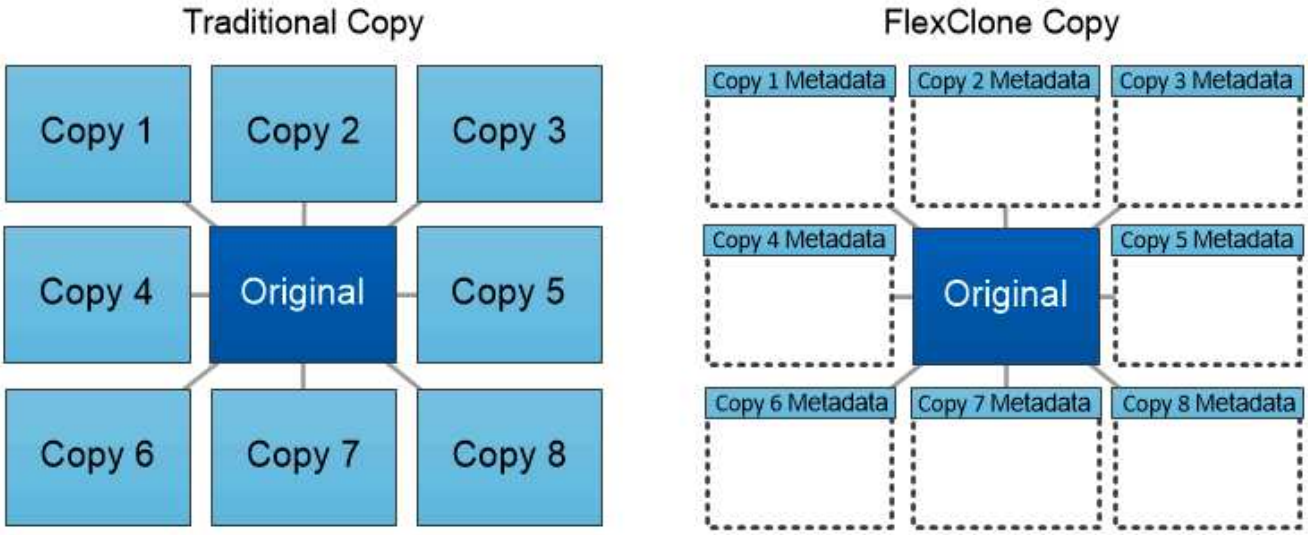
È possibile utilizzare una copia Snapshot per recuperare singoli file o LUN oppure per ripristinare l'intero contenuto di un volume. ONTAP confronta le informazioni del puntatore nella copia Snapshot con i dati sul disco per ricostruire l'oggetto mancante o danneggiato, senza tempi di inattività o costi significativi in termini di prestazioni.



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 e puntuali di un volume. Le copie condividono blocchi di dati con i loro genitori, senza consumare spazio di archiviazione, se non quello necessario per i metadati, finché le modifiche non vengono scritte sulla copia, come illustrato nella figura seguente. Mentre le copie tradizionali possono richiedere minuti o addirittura ore per essere create, il software FlexClone consente di copiare anche i set di dati più grandi in modo quasi istantaneo. Ciò lo rende ideale per le 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 (per testare 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 di replicazione dei dati NetApp SnapMirror

Il software NetApp SnapMirror è una soluzione di replica unificata conveniente e facile da usare nell'intero data fabric. Replica i dati ad alta velocità su LAN o WAN. Offre un'elevata disponibilità dei dati e una rapida replicazione dei dati per applicazioni di tutti i tipi, comprese le applicazioni aziendali critiche in ambienti sia virtuali che tradizionali. Quando si replicano i dati su uno o più sistemi di storage NetApp e si aggiornano continuamente i dati secondari, i dati vengono mantenuti aggiornati e sono disponibili ogni volta che ne hai bisogno. Non sono richiesti server di replicazione esterni. Per un esempio di architettura che sfrutta la tecnologia SnapMirror , vedere la figura seguente.

Il software SnapMirror sfrutta l'efficienza di archiviazione NetApp ONTAP inviando sulla rete solo i blocchi modificati. Il software SnapMirror utilizza anche la compressione di rete integrata per accelerare i trasferimenti di dati e ridurre l'utilizzo della larghezza di banda di rete fino al 70%. Grazie alla tecnologia SnapMirror , è possibile sfruttare un flusso di dati di replica sottile per creare un singolo repository che gestisce sia il mirror attivo sia le copie point-in-time precedenti, riducendo il traffico di rete fino al 50%.

Copia e sincronizzazione NetApp BlueXP

"BlueXP Copia e Sincronizza"è un servizio NetApp per la sincronizzazione rapida e sicura dei dati. Che tu debba trasferire file tra condivisioni file NFS o SMB locali, NetApp StorageGRID, NetApp ONTAP S3, Google Cloud NetApp Volumes, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage o IBM Cloud Object Storage, BlueXP Copy and Sync sposta i file dove ti servono in modo rapido e sicuro.

Una volta trasferiti, i dati saranno completamente disponibili per l'uso sia sulla sorgente che sulla destinazione. BlueXP Copy and Sync può sincronizzare i dati su richiesta quando viene attivato un aggiornamento oppure sincronizzare i dati in modo continuo in base a una pianificazione predefinita. In ogni caso, BlueXP Copy and Sync sposta solo i delta, riducendo al minimo il tempo e il denaro spesi per la replicazione dei dati.

BlueXP Copy and Sync è uno strumento software as a service (SaaS) estremamente semplice da configurare e utilizzare. I trasferimenti di dati attivati da BlueXP Copy and Sync vengono eseguiti da broker di dati. I broker di dati BlueXP Copy and Sync possono essere distribuiti su AWS, Azure, Google Cloud Platform o in locale.

NetApp XCP

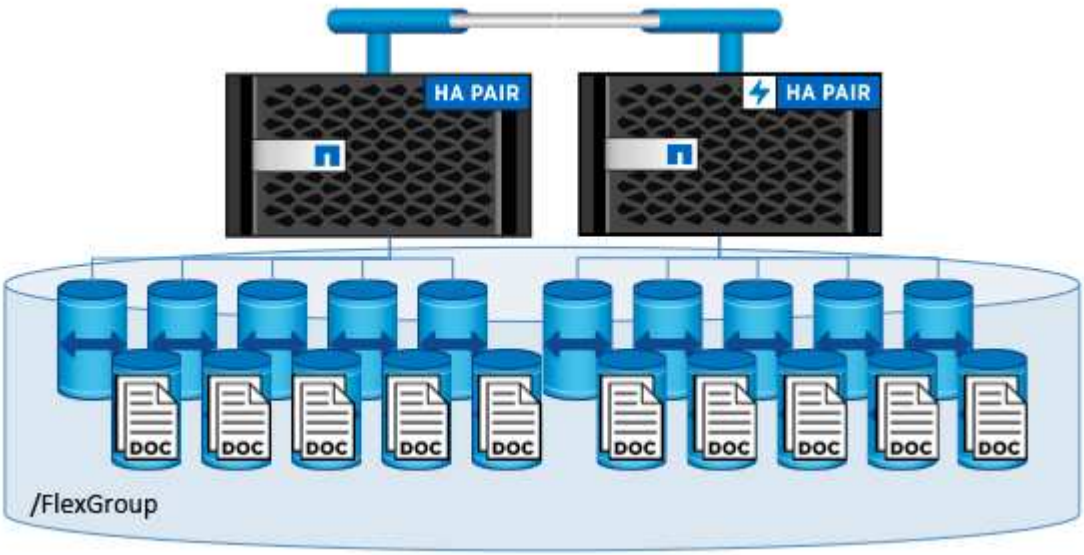
"NetApp XCP"è un software basato su client per migrazioni di dati da NetApp a NetApp e da NetApp a NetApp e per analisi del file system. XCP è progettato per scalare e raggiungere le massime prestazioni utilizzando tutte le risorse di sistema disponibili per gestire set di dati di grandi volumi e migrazioni ad alte prestazioni. XCP ti aiuta ad ottenere una visibilità completa del file system con la possibilità di generare report.

Volumi NetApp ONTAP FlexGroup

Un set di dati di addestramento può essere una raccolta di potenzialmente miliardi di file. I file possono contenere testo, audio, video e altre forme di dati non strutturati che devono essere archiviati ed elaborati per poter essere letti in parallelo. Il sistema di archiviazione deve memorizzare un gran numero di file di piccole dimensioni e deve leggere tali file in parallelo per l'I/O sequenziale e casuale.

Un volume FlexGroup è un singolo spazio dei nomi che comprende più volumi membri costituenti, come mostrato nella figura seguente. Dal punto di vista di un amministratore di storage, un volume FlexGroup viene gestito e si comporta come un FlexVol volume NetApp FlexVol. I file in un volume FlexGroup vengono assegnati ai singoli volumi membri e non vengono distribuiti tra volumi o nodi. Abilitano le seguenti funzionalità:

- I volumi FlexGroup forniscono diversi petabyte di capacità e una latenza bassa e prevedibile per carichi di lavoro con metadati elevati.
- Supportano fino a 400 miliardi di file nello stesso namespace.
- Supportano operazioni parallelizzate nei carichi di lavoro NAS su CPU, nodi, aggregati e volumi FlexVol costituenti.



Architettura

Questa soluzione non dipende da hardware specifico. La soluzione è compatibile con qualsiasi dispositivo di storage fisico NetApp , istanza definita dal software o servizio cloud supportato da NetApp Trident. Tra gli esempi figurano un sistema di archiviazione NetApp AFF , Amazon FSx ONTAP, Azure NetApp Files, Google Cloud NetApp Volumes o un'istanza NetApp Cloud Volumes ONTAP . Inoltre, la soluzione può essere implementata su qualsiasi cluster Kubernetes, a condizione che la versione di Kubernetes utilizzata sia supportata da NetApp Trident e dagli altri componenti della soluzione implementati. Per un elenco delle versioni di Kubernetes supportate da Trident, vedere "Documentazione Trident" . Per i dettagli sugli ambienti utilizzati per convalidare i vari componenti di questa soluzione, consultare le tabelle seguenti.

Ambiente di convalida Apache Airflow

Componente software	Versione
Apache Airflow	2.0.1, distribuito tramite"Grafico del timone Apache Airflow" 8.0.8
Kubernetes	1,18
NetApp Trident	21,01

Ambiente di convalida JupyterHub

Componente software	Versione
JupyterHub	4.1.5, distribuito tramite"Grafico Helm di JupyterHub" 3.3.7
Kubernetes	1,29
NetApp Trident	24,02

Ambiente di convalida MLflow

Componente software	Versione
Flusso ML	2.14.1, distribuito tramite"Grafico Helm MLflow" 1.4.12
Kubernetes	1,29
NetApp Trident	24,02

Ambiente di convalida Kubeflow

Componente software	Versione
Kubeflow	1.7, distribuito tramite"deployKF" 0.1.1
Kubernetes	1,26

Componente software	Versione
NetApp Trident	23,07

Supporto

NetApp non offre supporto aziendale per Apache Airflow, JupyterHub, MLflow, Kubeflow o Kubernetes. Se sei interessato a una piattaforma MLOps completamente supportata,"contattare NetApp" sulle soluzioni MLOps completamente supportate che NetApp offre insieme ai partner.

Configurazione NetApp Trident

Esempi di backend Trident per distribuzioni NetApp AIPod

Prima di poter utilizzare Trident per effettuare 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 potresti voler creare se stai distribuendo componenti di questa soluzione su un"NetApp AIPod" . Per ulteriori informazioni sui backend e, ad esempio, sui backend per altre piattaforme/ambienti, vedere"Documentazione Trident" .

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

   I comandi di esempio che seguono mostrano la creazione di un backend Trident abilitato per FlexGroup per una macchina virtuale di archiviazione AIPod (SVM). Questo Backend utilizza il `ontap-nas-flexgroup` driver di archiviazione. ONTAP supporta due tipi principali di volumi di dati: FlexVol e FlexGroup. I volumi FlexVol hanno dimensioni limitate (al momento della stesura di questo documento, la dimensione massima dipende dalla distribuzione specifica). I volumi FlexGroup , d'altro canto, possono essere scalati 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 di intelligenza artificiale e apprendimento automatico che si basano su grandi quantità di dati.

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

```
$ 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 possibile utilizzare i volumi FlexVol per ospitare applicazioni persistenti, archiviare 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 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 |
+-------------------------+--------------------
+--------------------------------------+--------+---------+
```

Esempi di classi di archiviazione Kubernetes per distribuzioni NetApp AIPod

Prima di poter utilizzare Trident per effettuare il provisioning dinamico delle risorse di storage all'interno del cluster Kubernetes, è necessario creare una o più StorageClass Kubernetes. Gli esempi che seguono rappresentano diversi tipi di StorageClasses che potresti voler creare se stai distribuendo componenti di questa soluzione su un"NetApp AIPod" . Per ulteriori informazioni su StorageClasses e, ad esempio, su StorageClasses

per altre piattaforme/ambienti, vedere"Documentazione Trident" .

1. NetApp consiglia di creare una StorageClass per il backend Trident abilitato per FlexGroup creato nella sezione"Esempi di backend Trident per distribuzioni NetApp AIPod" , passo 1. I comandi di esempio che seguono mostrano la creazione di più StorageClass che corrispondono al Backend di esempio creato nella sezione"Esempi di backend Trident per distribuzioni NetApp AIPod" , passo 1 - uno che utilizza"NFS su RDMA" e uno che non lo fa.

   Per evitare che un volume persistente venga eliminato quando viene eliminato il PersistentVolumeClaim (PVC) corrispondente, l'esempio seguente utilizza un `reclaimPolicy` valore di `Retain` . Per maggiori informazioni sul `reclaimPolicy` campo, vedere l'ufficiale "Documentazione di Kubernetes" .

   Nota: le seguenti StorageClass di esempio utilizzano una dimensione di trasferimento massima di 262144. Per utilizzare questa dimensione massima di trasferimento, è necessario configurare di conseguenza la dimensione massima di trasferimento sul sistema ONTAP . Fare riferimento al"Documentazione ONTAP" per i dettagli.

   Nota: per utilizzare NFS su RDMA, è necessario configurare NFS su RDMA sul sistema ONTAP . Fare riferimento al"Documentazione ONTAP" per i dettagli.

   Nota: nell'esempio seguente, un Backend specifico è specificato 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 una StorageClass che corrisponda al Trident Backend abilitato per FlexVol creato nella sezione"Esempi di backend Trident per distribuzioni AIPod" , passo 2. I comandi di esempio che seguono mostrano la creazione di una singola StorageClass per i volumi FlexVol .

   Nota: nell'esempio seguente, un particolare Backend non è specificato nel campo storagePool nel file di definizione StorageClass. Quando si utilizza Kubernetes per amministrare i volumi utilizzando questa StorageClass, Trident tenta di utilizzare qualsiasi backend disponibile che utilizza `ontap-nas` autista.

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

Apache Airflow

Distribuzione di Apache Airflow

Questa sezione descrive le attività che devi completare per distribuire Airflow nel tuo cluster Kubernetes.

È possibile distribuire Airflow su piattaforme diverse da Kubernetes. L'implementazione di Airflow su piattaforme diverse da Kubernetes esula dall'ambito di questa soluzione.

Prerequisiti

Prima di eseguire l'esercizio di distribuzione descritto in questa sezione, diamo per scontato che tu abbia già eseguito le seguenti attività:

1. Hai già un cluster Kubernetes funzionante.
2. Hai già installato e configurato NetApp Trident nel tuo cluster Kubernetes. Per maggiori dettagli su Trident, fare riferimento al"Documentazione Trident" .

Installa Helm

Airflow viene distribuito tramite Helm, un noto gestore di pacchetti per Kubernetes. Prima di distribuire Airflow, è necessario installare Helm sull'host di distribuzione. Per installare Helm sull'host di distribuzione jump, seguire le istruzioni "istruzioni di installazione" nella documentazione ufficiale di Helm.

Imposta la classe di archiviazione Kubernetes predefinita

Prima di distribuire Airflow, è necessario designare una StorageClass predefinita all'interno del cluster Kubernetes. Il processo di distribuzione di Airflow tenta di effettuare il provisioning di nuovi volumi persistenti

utilizzando la StorageClass predefinita. Se non viene designata alcuna StorageClass come StorageClass predefinita, la distribuzione fallisce. Per designare una StorageClass predefinita all'interno del cluster, seguire le istruzioni descritte in"Distribuzione di Kubeflow" sezione. Se hai già designato una StorageClass predefinita all'interno del tuo cluster, puoi saltare questo passaggio.

Usa Helm per distribuire il flusso d'aria

Per distribuire Airflow nel cluster Kubernetes tramite Helm, eseguire le seguenti attività dall'host di jump di distribuzione:

1. Distribuisci Airflow utilizzando Helm seguendo le istruzioni "istruzioni di distribuzione" per il grafico ufficiale del flusso d'aria su Artifact Hub. I comandi di esempio che seguono mostrano la distribuzione di Airflow tramite Helm. Modificare, aggiungere e/o rimuovere valori in `custom- values.yaml` file secondo 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
```

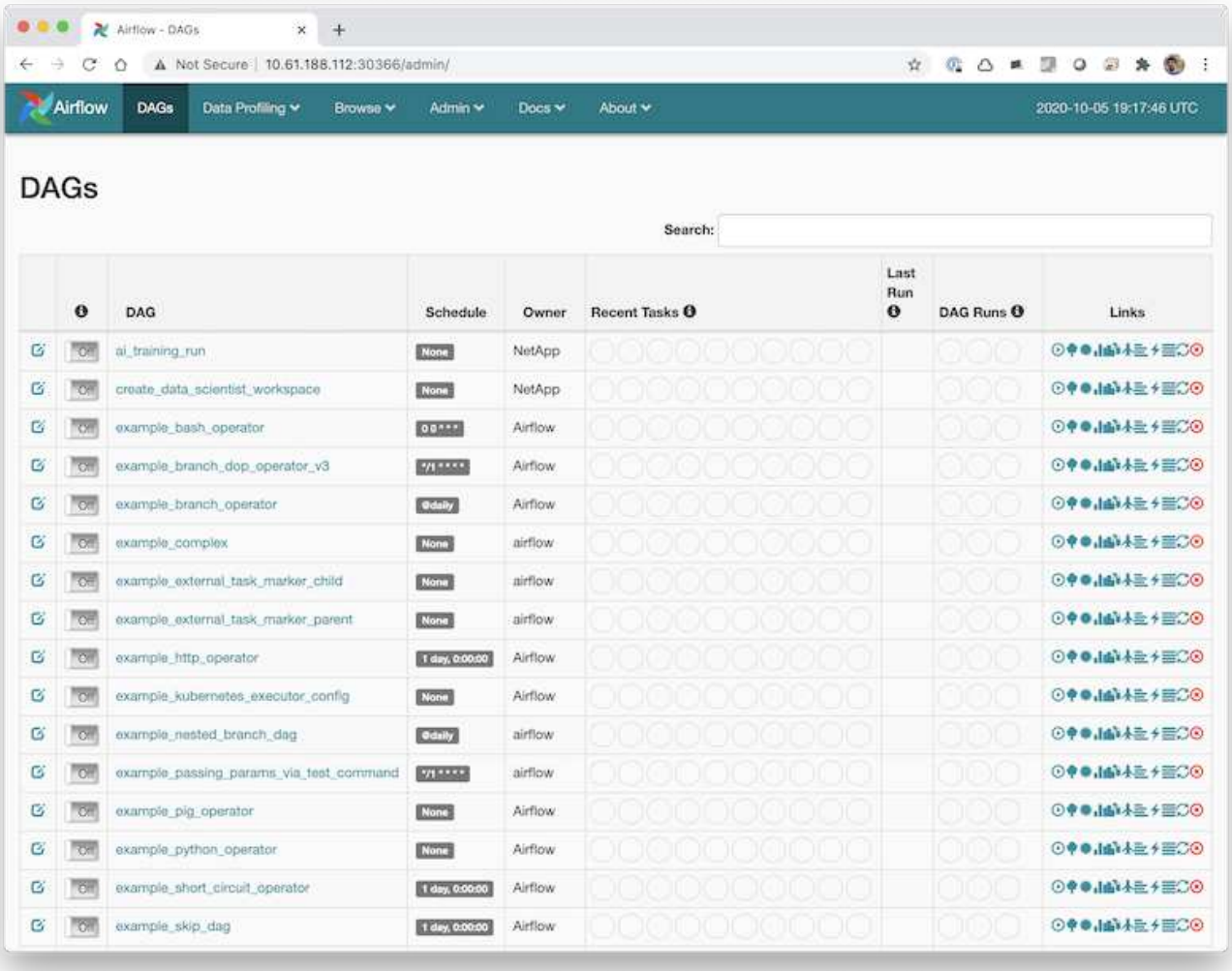
2. Verificare che tutti i pod Airflow siano attivi e funzionanti. Potrebbero essere necessari alcuni minuti prima che tutti i pod si avviino.

```
$ 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. Ottieni l'URL del servizio Web Airflow seguendo le istruzioni visualizzate sulla console quando hai distribuito Airflow tramite 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. Conferma di poter accedere al servizio web Airflow.



Utilizzare NetApp DataOps Toolkit con Airflow

IL "Kit degli strumenti NetApp DataOps per Kubernetes" può essere utilizzato insieme ad Airflow. Utilizzando NetApp DataOps Toolkit con Airflow è possibile integrare le operazioni di gestione dei dati NetApp , come la creazione di snapshot e cloni, in flussi di lavoro automatizzati orchestrati da Airflow.

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

JupyterHub

Distribuzione di JupyterHub

Questa sezione descrive le attività che devi completare per distribuire JupyterHub nel tuo cluster Kubernetes.

È possibile distribuire JupyterHub su piattaforme diverse da Kubernetes. L'implementazione di JupyterHub su piattaforme diverse da Kubernetes esula dall'ambito di questa soluzione.

Prerequisiti

Prima di eseguire l'esercizio di distribuzione descritto in questa sezione, diamo per scontato che tu abbia già eseguito le seguenti attività:

1. Hai già un cluster Kubernetes funzionante.
2. Hai già installato e configurato NetApp Trident nel tuo cluster Kubernetes. Per maggiori dettagli su Trident, fare riferimento al"Documentazione Trident" .

Installa Helm

JupyterHub viene distribuito tramite Helm, un noto gestore di pacchetti per Kubernetes. Prima di distribuire JupyterHub, devi installare Helm sul tuo nodo di controllo Kubernetes. Per installare Helm, seguire le istruzioni "istruzioni di installazione" nella documentazione ufficiale di Helm.

Imposta la classe di archiviazione Kubernetes predefinita

Prima di distribuire JupyterHub, è necessario designare una StorageClass predefinita all'interno del cluster Kubernetes. Per designare una StorageClass predefinita all'interno del cluster, seguire le istruzioni descritte in"Distribuzione di Kubeflow" sezione. Se hai già designato una StorageClass predefinita all'interno del tuo cluster, puoi saltare questo passaggio.

Distribuisci JupyterHub

Dopo aver completato i passaggi precedenti, sei pronto per distribuire JupyterHub. Per distribuire JupyterHub sono necessari i seguenti passaggi:

Configurare la distribuzione di JupyterHub

Prima della distribuzione è buona norma ottimizzare la distribuzione di JupyterHub per il proprio ambiente. È possibile creare un file **config.yaml** e utilizzarlo durante la distribuzione tramite il grafico Helm.

Un esempio di file **config.yaml** può essere trovato qui https://github.com/jupyterhub/zero-to-jupyterhub-k8s/blob/HEAD/jupyterhub/values.yaml

In questo file config.yaml è possibile impostare il parametro **(singleuser.storage.dynamic.storageClass)** per NetApp Trident StorageClass. Questa è la classe di archiviazione che verrà utilizzata per predisporre i volumi per gli spazi di lavoro dei singoli utenti.

Aggiunta di volumi condivisi

Se vuoi utilizzare un volume condiviso per tutti gli utenti di JupyterHub, puoi modificare di conseguenza il tuo **config.yaml**. Ad esempio, se hai un PersistentVolumeClaim condiviso denominato jupyterhub-shared-volume, puoi montarlo come /home/shared in tutti i pod utente come:

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

Questo è un passaggio facoltativo, puoi adattare questi parametri alle tue esigenze.

Distribuisci JupyterHub con Helm Chart

Informa Helm del repository dei grafici JupyterHub Helm.

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

Dovrebbe apparire un output simile a questo:

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

Ora installa il grafico configurato dal tuo config.yaml eseguendo questo comando dalla directory che contiene il tuo config.yaml:

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

In questo esempio:

<helm-release-name> è impostato su my-jupyterhub, che sarà il nome della tua release JupyterHub. <k8s-namespace> è impostato su my-namespace, che è lo spazio dei nomi in cui si desidera installare JupyterHub. Il flag --create-namespace viene utilizzato per creare lo spazio dei nomi se non esiste già. Il flag --values specifica il file config.yaml che contiene le opzioni di configurazione desiderate.

Controlla la distribuzione

Mentre è in esecuzione il passaggio 2, puoi vedere i pod creati tramite il seguente comando:

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

Attendi che l'hub e il proxy pod entrino nello stato In esecuzione.

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

Accedi a JupyterHub

Trova l'IP che possiamo usare per accedere a JupyterHub. Eseguire il seguente comando finché l'indirizzo EXTERNAL-IP del servizio proxy-public non è disponibile, come nell'output di esempio.

> Abbiamo utilizzato il servizio NodePort nel nostro file config.yaml; puoi adattarlo al tuo ambiente in base alla configurazione (ad esempio LoadBalancer).

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

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

Per utilizzare JupyterHub, inserisci l'IP esterno per il servizio proxy-public in un browser.

Utilizzare NetApp DataOps Toolkit con JupyterHub

IL "Kit degli strumenti NetApp DataOps per Kubernetes" può essere utilizzato insieme a JupyterHub. Utilizzando NetApp DataOps Toolkit con JupyterHub, gli utenti finali possono creare snapshot di volumi per il backup dell'area di lavoro e/o la tracciabilità del set di dati al modello direttamente da un Jupyter Notebook.

Configurazione iniziale

Prima di poter utilizzare DataOps Toolkit con JupyterHub, è necessario concedere le autorizzazioni appropriate all'account del servizio Kubernetes che JupyterHub assegna ai singoli pod utente di Jupyter Notebook Server. JupyterHub utilizza l'account di servizio specificato da `singleuser.serviceAccountName` variabile nel file di configurazione del grafico Helm di JupyterHub.

Crea un ruolo cluster per DataOps Toolkit

Per prima cosa, crea un ruolo cluster denominato "netapp-dataops" che disponga delle autorizzazioni API Kubernetes necessarie per la creazione di snapshot del volume.

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

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

Assegnare il ruolo del cluster all'account del servizio Notebook Server

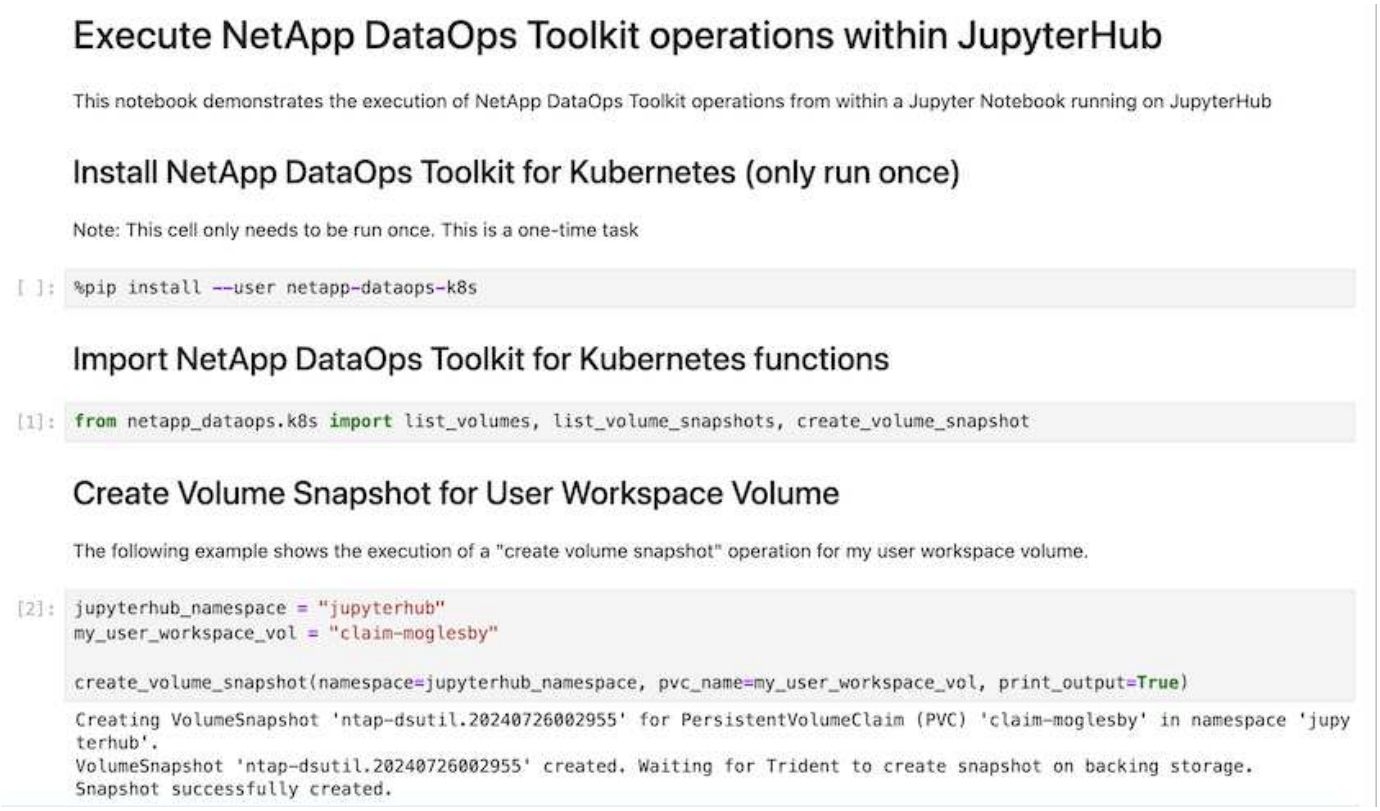
Creare un'associazione di ruolo che assegni il ruolo del cluster 'netapp-dataops-snapshots' all'account di servizio appropriato nello spazio dei nomi appropriato. Ad esempio, se hai installato JupyterHub nello spazio dei nomi 'jupyterhub' e hai specificato l'account di servizio 'predefinito' tramite `singleuser.serviceAccountName` variabile, dovresti assegnare il ruolo del cluster 'netapp-dataops-snapshots' all'account di servizio 'default' nello spazio dei nomi 'jupyterhub' come mostrato nell'esempio seguente.

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

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

Creare snapshot di volume all'interno di Jupyter Notebook

Ora gli utenti di JupyterHub possono utilizzare NetApp DataOps Toolkit per creare snapshot di volumi direttamente da un Jupyter Notebook, come mostrato nell'esempio seguente.



Importa dati in JupyterHub con NetApp SnapMirror

NetApp SnapMirror è una tecnologia di replicazione che consente di replicare i dati tra i sistemi di storage NetApp . SnapMirror può essere utilizzato per importare dati da ambienti remoti in JupyterHub.

Esempio di flusso di lavoro e demo

Fare riferimento a"questo post del blog Tech ONTAP" per un esempio dettagliato del flusso di lavoro e una demo dell'utilizzo di NetApp SnapMirror per l'acquisizione di dati in JupyterHub.

Flusso ML

Distribuzione MLflow

Questa sezione descrive le attività che devi completare per distribuire MLflow nel tuo cluster Kubernetes.

È possibile distribuire MLflow su piattaforme diverse da Kubernetes. L'implementazione di MLflow su piattaforme diverse da Kubernetes esula dall'ambito di questa soluzione.

Prerequisiti

Prima di eseguire l'esercizio di distribuzione descritto in questa sezione, diamo per scontato che tu abbia già eseguito le seguenti attività:

1. Hai già un cluster Kubernetes funzionante.
2. Hai già installato e configurato NetApp Trident nel tuo cluster Kubernetes. Per maggiori dettagli su Trident, fare riferimento al"Documentazione Trident" .

Installa Helm

MLflow viene distribuito tramite Helm, un noto gestore di pacchetti per Kubernetes. Prima di distribuire MLflow, è necessario installare Helm sul nodo di controllo Kubernetes. Per installare Helm, seguire le istruzioni "istruzioni di installazione" nella documentazione ufficiale di Helm.

Imposta la classe di archiviazione Kubernetes predefinita

Prima di distribuire MLflow, è necessario designare una StorageClass predefinita all'interno del cluster Kubernetes. Per designare una StorageClass predefinita all'interno del cluster, seguire le istruzioni descritte in"Distribuzione di Kubeflow" sezione. Se hai già designato una StorageClass predefinita all'interno del tuo cluster, puoi saltare questo passaggio.

Distribuisci MLflow

Una volta soddisfatti i prerequisiti, è possibile iniziare la distribuzione di MLflow utilizzando il grafico Helm.

Configurare la distribuzione del grafico Helm di MLflow.

Prima di distribuire MLflow utilizzando il grafico Helm, possiamo configurare la distribuzione per utilizzare NetApp Trident Storage Class e modificare altri parametri in base alle nostre esigenze utilizzando un file

config.yaml. Un esempio del file **config.yaml** può essere trovato qui: https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml

> È possibile impostare Trident storageClass nel parametro **global.defaultStorageClass** nel file config.yaml (ad esempio storageClass: "ontap-flexvol").

Installazione della tabella del timone

Il grafico Helm può essere installato con il file **config.yaml** personalizzato per MLflow utilizzando il seguente comando:

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

> Il comando distribuisce MLflow sul cluster Kubernetes nella configurazione personalizzata tramite il file **config.yaml** fornito. MLflow viene distribuito nello spazio dei nomi specificato e per la release viene assegnato un nome di rilascio casuale tramite Kubernetes.

Controlla la distribuzione

Dopo aver completato la distribuzione del grafico Helm, puoi verificare se il servizio è accessibile utilizzando:

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

> Sostituisci **jupyterhub** con lo spazio dei nomi utilizzato durante la distribuzione.

Dovresti vedere i seguenti servizi:

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

> Abbiamo modificato il file config.yaml per utilizzare il servizio NodePort per accedere a MLflow sulla porta 30002.

Accedi a MLflow

Una volta che tutti i servizi relativi a MLflow sono attivi e funzionanti, è possibile accedervi utilizzando l'indirizzo

IP NodePort o LoadBalancer specificato (ad esempio `http://10.61.181.109:30002`)

Tracciabilità dal set di dati al modello con NetApp e MLflow

IL "Kit degli strumenti NetApp DataOps per Kubernetes" può essere utilizzato insieme alle funzionalità di tracciamento degli esperimenti di MLflow per implementare la tracciabilità dal set di dati al modello o dall'area di lavoro al modello.

Per implementare la tracciabilità dal set di dati al modello o dall'area di lavoro al modello, è sufficiente creare uno snapshot del volume del set di dati o dell'area di lavoro utilizzando DataOps Toolkit come parte dell'esecuzione dell'addestramento, come mostrato nel seguente frammento di codice di esempio. Questo codice salverà il nome del volume di dati e il nome dello snapshot come tag associati all'esecuzione di addestramento specifica che stai registrando sul tuo server di monitoraggio degli esperimenti MLflow.

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

with mlflow.start_run() :
    ...

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

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

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

    ...
```

Kubeflow

Distribuzione di Kubeflow

Questa sezione descrive le attività che devi completare per distribuire Kubeflow nel tuo

cluster Kubernetes.

Prerequisiti

Prima di eseguire l'esercizio di distribuzione descritto in questa sezione, diamo per scontato che tu abbia già eseguito le seguenti attività:

1. Hai già un cluster Kubernetes funzionante e stai eseguendo una versione di Kubernetes supportata dalla versione di Kubeflow che intendi distribuire. Per un elenco delle versioni di Kubernetes supportate, fare riferimento alle dipendenze per la versione di Kubeflow in"documentazione ufficiale di Kubeflow" .
2. Hai già installato e configurato NetApp Trident nel tuo cluster Kubernetes. Per maggiori dettagli su Trident, fare riferimento al"Documentazione Trident" .

Imposta la classe di archiviazione Kubernetes predefinita

Prima di distribuire Kubeflow, ti consigliamo di designare una StorageClass predefinita all'interno del tuo cluster Kubernetes. Il processo di distribuzione di Kubeflow potrebbe tentare di effettuare il provisioning di nuovi volumi persistenti utilizzando la StorageClass predefinita. Se non viene designata alcuna StorageClass come StorageClass predefinita, la distribuzione potrebbe non riuscire. Per designare una StorageClass predefinita all'interno del cluster, eseguire la seguente attività dall'host di jump di distribuzione. Se hai già designato una StorageClass predefinita all'interno del tuo cluster, puoi saltare questo passaggio.

1. Designa una delle StorageClass esistenti come StorageClass predefinita. I comandi di esempio che seguono mostrano la designazione di una StorageClass denominata `ontap-ai-flexvols-retain` come StorageClass predefinito.

> IL `ontap-nas-flexgroup` Il tipo Trident Backend ha una dimensione minima del PVC piuttosto grande. Per impostazione predefinita, Kubeflow tenta di fornire PVC di dimensioni pari solo a pochi GB. Pertanto, non dovresti designare una StorageClass che utilizza `ontap-nas-flexgroup` Tipo di backend come StorageClass predefinito ai fini della distribuzione 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
```

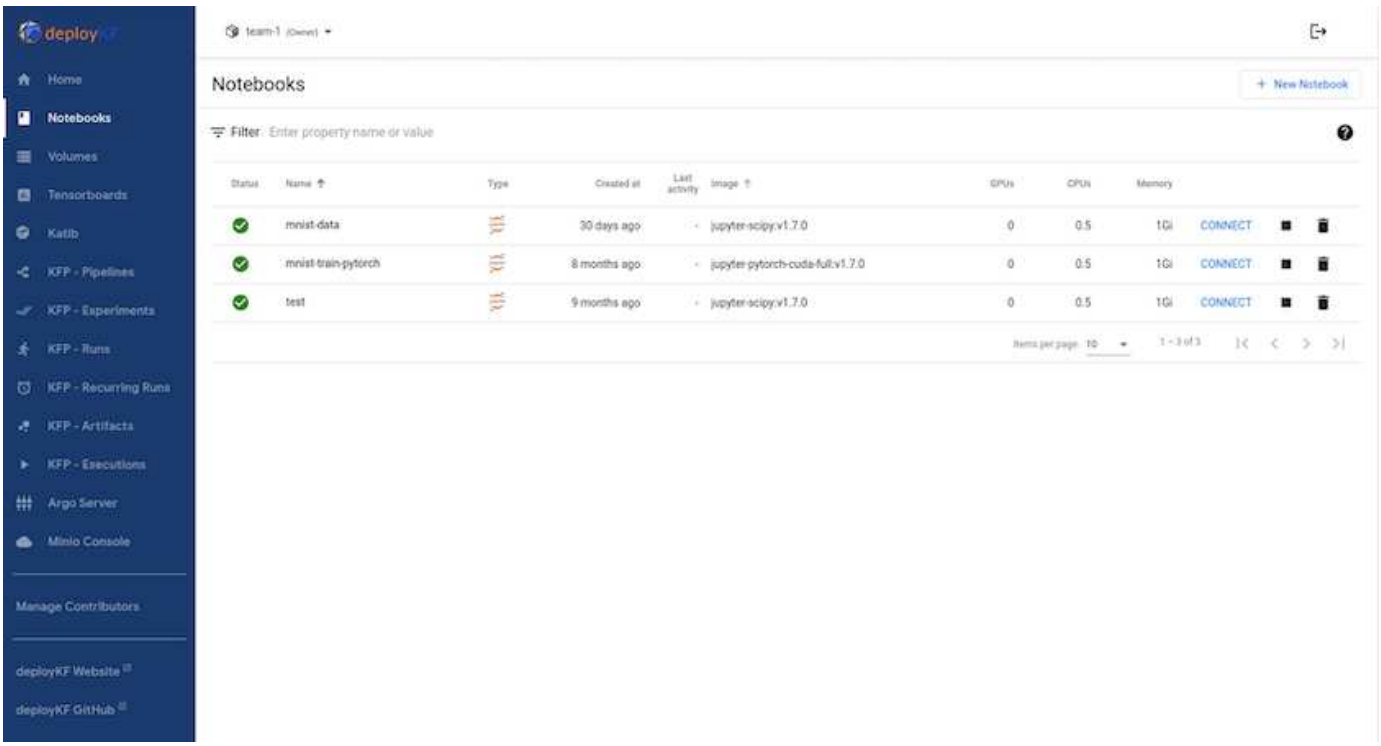
Opzioni di distribuzione di Kubeflow

Esistono molte opzioni diverse per distribuire Kubeflow. Fare riferimento al"documentazione ufficiale di Kubeflow" per un elenco delle opzioni di distribuzione e scegli l'opzione più adatta alle tue esigenze.

Per scopi di convalida, abbiamo distribuito Kubeflow 1.7 utilizzando"deployKF" 0.1.1.

Fornire un'area di lavoro Jupyter Notebook per l'uso da parte di Data Scientist o Sviluppatori

Kubeflow è in grado di fornire rapidamente nuovi server Jupyter Notebook che fungano da spazi di lavoro per gli scienziati dei dati. Per ulteriori informazioni sui notebook Jupyter nel contesto Kubeflow, vedere "documentazione ufficiale di Kubeflow" .



Utilizzare NetApp DataOps Toolkit con Kubeflow

IL "NetApp Data Science Toolkit per Kubernetes" può essere utilizzato insieme a Kubeflow. L'utilizzo di NetApp Data Science Toolkit con Kubeflow offre i seguenti vantaggi:

- Gli scienziati dei dati possono eseguire operazioni avanzate di gestione dei dati NetApp , come la creazione di snapshot e cloni, direttamente da un Jupyter Notebook.
- Le operazioni avanzate di gestione dei dati NetApp , come la creazione di snapshot e cloni, possono essere integrate in flussi di lavoro automatizzati utilizzando il framework Kubeflow Pipelines.

Fare riferimento al "Esempi di Kubeflow" sezione all'interno del repository GitHub del NetApp Data Science Toolkit per i dettagli sull'utilizzo del toolkit con Kubeflow.

Esempio di flusso di lavoro: addestrare un modello di riconoscimento delle immagini utilizzando Kubeflow e NetApp DataOps Toolkit

Questa sezione descrive i passaggi necessari per addestrare e distribuire una rete neurale per il riconoscimento delle immagini utilizzando Kubeflow e NetApp DataOps Toolkit. Questo esempio vuole mostrare un lavoro di formazione che incorpora l'archiviazione NetApp .

Prerequisiti

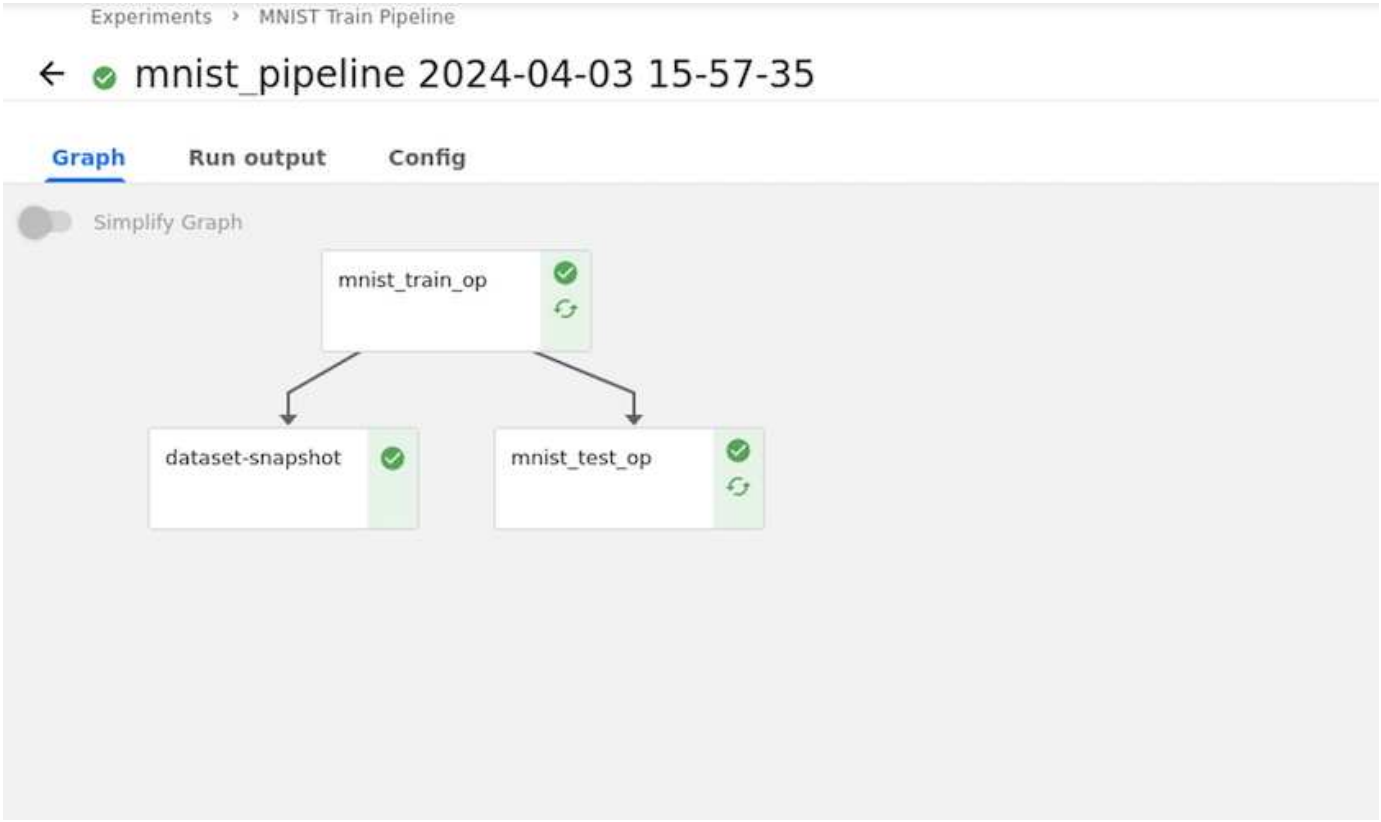
Creare un Dockerfile con le configurazioni richieste da utilizzare per i passaggi di training e test all'interno della pipeline Kubeflow. Ecco un esempio di Dockerfile:

```
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 delle tue esigenze, installa tutte le librerie e i pacchetti necessari per eseguire il programma. Prima di addestrare il modello di Machine Learning, si presuppone che si disponga già di una distribuzione Kubeflow funzionante.

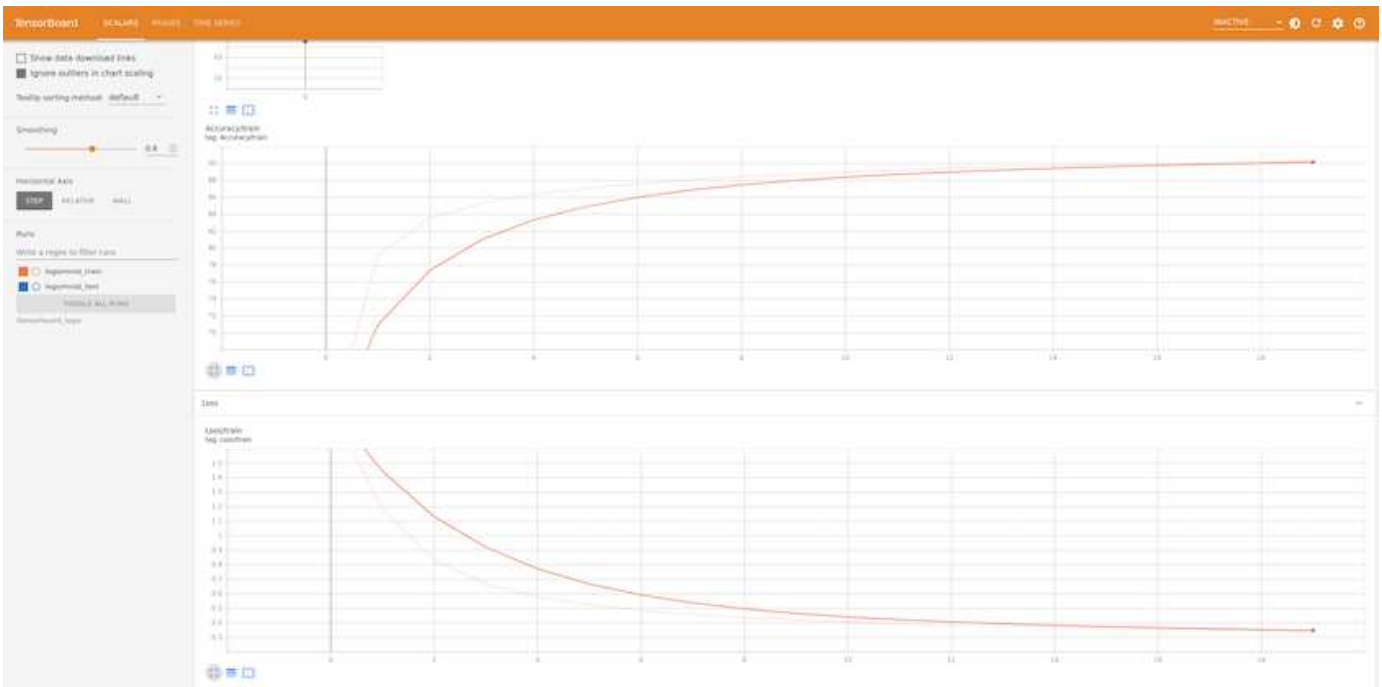
Addestrare una piccola rete neurale su dati MNIST utilizzando pipeline PyTorch e Kubeflow

Utilizziamo l'esempio di una piccola rete neurale addestrata sui dati MNIST. Il set di dati MNIST è costituito da immagini manoscritte di cifre da 0 a 9. Le immagini hanno una dimensione di 28x28 pixel. Il set di dati è suddiviso in 60.000 immagini di 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 Kubeflow Pipelines. Fare riferimento alla documentazione "Qui" per maggiori informazioni. La nostra pipeline Kubeflow incorpora l'immagine Docker dalla sezione Prerequisiti.



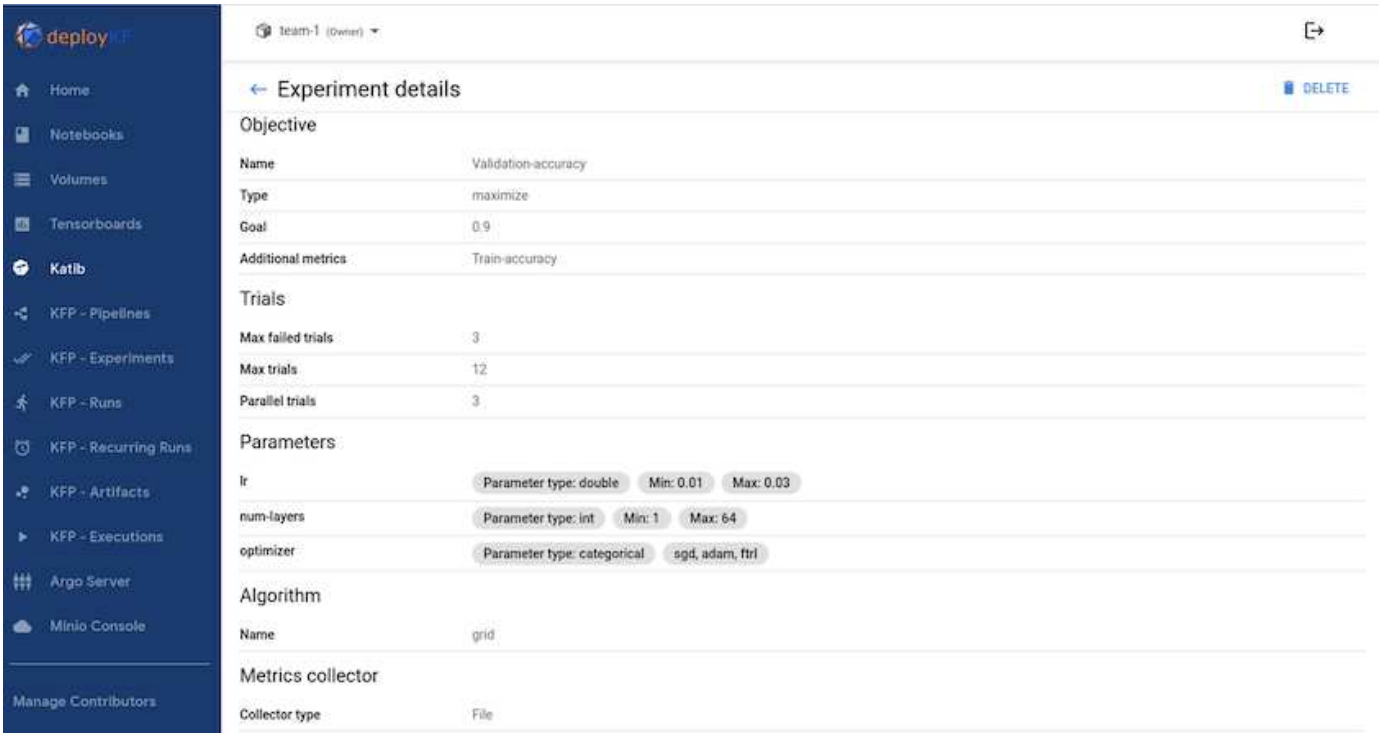
Visualizza i risultati utilizzando Tensorboard

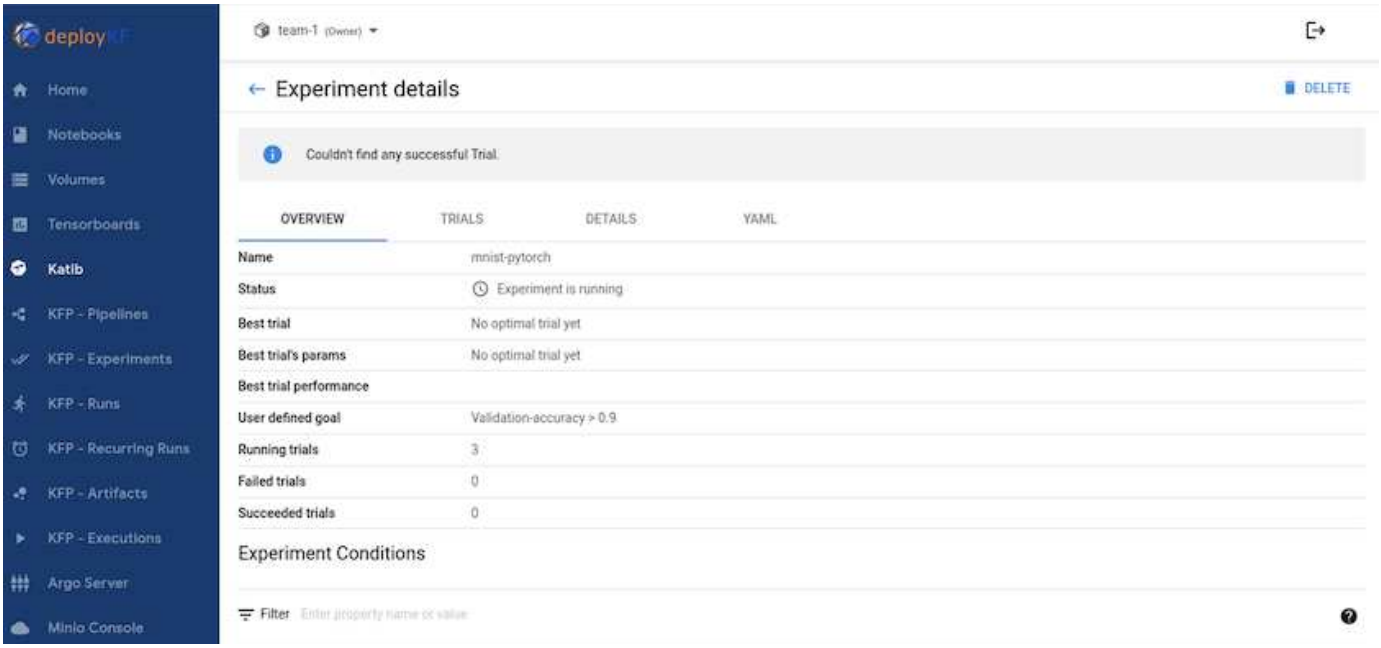
Una volta addestrato il modello, possiamo visualizzare i risultati utilizzando Tensorboard. "Tensorboard" è disponibile come funzionalità nella dashboard di Kubeflow. Puoi creare un tensorboard personalizzato per il tuo lavoro. Di seguito è riportato un esempio che mostra il grafico dell'accuratezza dell'addestramento rispetto al numero di epoche e della perdita dell'addestramento rispetto al numero di epoche.



Sperimentare con gli iperparametri utilizzando Katib

"Katib"è uno strumento all'interno di Kubeflow che può essere utilizzato per sperimentare con gli iperparametri del modello. Per creare un esperimento, definisci prima una metrica/un obiettivo desiderato. Di solito questa è la precisione del test. Una volta definita la metrica, scegli gli iperparametri con cui vuoi sperimentare (ottimizzatore/tasso di apprendimento/numero di livelli). Katib esegue una scansione degli iperparametri con i valori definiti dall'utente per trovare la migliore combinazione di parametri che soddisfi la metrica desiderata. È possibile definire questi parametri in ogni sezione dell'interfaccia utente. In alternativa, è possibile definire un file **YAML** con le specifiche necessarie. Di seguito è riportata un'illustrazione di un esperimento di Katib:





Utilizzare gli snapshot NetApp per salvare i dati per la tracciabilità

Durante l'addestramento del modello, potremmo voler salvare un'istantanea del set di dati di addestramento per la tracciabilità. Per fare ciò, possiamo aggiungere un passaggio di snapshot alla pipeline come mostrato di seguito. Per creare lo snapshot, possiamo usare il "Kit degli strumenti NetApp DataOps 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 al "Esempio di NetApp DataOps Toolkit per Kubeflow" per maggiori informazioni.

Esempio di operazioni Trident

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

Importa un volume esistente

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

I comandi di esempio che seguono mostrano l'importazione di un volume denominato `pb_fg_all` . Per maggiori informazioni sui PVC, vedere "documentazione ufficiale di Kubernetes" . Per ulteriori informazioni sulla funzionalità di importazione del volume, vedere "Documentazione Trident" .

UN `accessModes` valore di `ReadOnlyMany` è specificato nei file di specifiche PVC di esempio. Per maggiori informazioni sul `accessMode` campo, vedere il "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
```

Fornire un nuovo volume

Puoi utilizzare Trident per predisporre un nuovo volume sul tuo sistema o piattaforma di storage NetApp .

Fornire un nuovo volume utilizzando kubectl

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

UN `accessModes` valore di `ReadWriteMany` è specificato nel seguente file di definizione PVC di esempio. Per maggiori informazioni sul `accessMode` campo, vedere il "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
```

Fornire un nuovo volume utilizzando NetApp DataOps Toolkit

Puoi anche utilizzare NetApp DataOps Toolkit per Kubernetes per effettuare il provisioning di un nuovo volume sul tuo sistema o piattaforma di storage NetApp . NetApp DataOps Toolkit per Kubernetes utilizza Trident per il provisioning dei volumi, semplificando però il processo per l'utente. Fare riferimento al"documentazione" per i dettagli.

Esempi di lavori ad alte prestazioni per distribuzioni AIPod

Eseguire un carico di lavoro AI a nodo singolo

Per eseguire un job di intelligenza artificiale e apprendimento automatico a nodo singolo nel cluster Kubernetes, esegui le seguenti attività dall'host di jump di distribuzione. Con Trident puoi rendere accessibile in modo rapido e semplice un volume di dati, potenzialmente contenente petabyte di dati, a un carico di lavoro Kubernetes. Per rendere accessibile tale volume di dati dall'interno di un pod Kubernetes, è sufficiente specificare un PVC nella definizione del pod.

Questa sezione presuppone che tu abbia già containerizzato (nel formato container Docker) il carico di lavoro AI e ML specifico che stai tentando di eseguire nel tuo cluster Kubernetes.

1. I seguenti comandi di esempio mostrano la creazione di un job 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 di ImageNet" .

   Questo esempio di lavoro richiede otto GPU e pertanto può essere eseguito su un singolo nodo worker GPU dotato di otto o più GPU. Questo esempio di lavoro potrebbe essere inviato in un cluster per il quale non è presente un nodo worker con otto o più GPU oppure è attualmente occupato da un altro carico di lavoro. In tal caso, il processo rimane in sospeso finché non diventa disponibile un nodo worker.

   Inoltre, per massimizzare la larghezza di banda di archiviazione, il volume che contiene i dati di formazione necessari viene montato due volte all'interno del pod creato da questo processo. Un altro volume è montato nel contenitore. Questo secondo volume verrà utilizzato per archiviare risultati e metriche. Questi volumi sono indicati nella definizione del lavoro utilizzando i nomi dei PVC. Per ulteriori informazioni sui lavori Kubernetes, vedere "documentazione ufficiale di Kubernetes" .

   UN `emptyDir` volume con un `medium` valore di `Memory` è montato su `/dev/shm` nel pod creato da questo lavoro di esempio. La dimensione predefinita del `/dev/shm` Il volume virtuale creato automaticamente dal runtime del contenitore Docker può talvolta essere insufficiente per le esigenze di TensorFlow. Montare un `emptyDir` volume come nell'esempio seguente fornisce un sufficientemente grande `/dev/shm` volume virtuale. Per maggiori informazioni su `emptyDir` volumi, vedere il "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 indica che il contenitore ha effettivamente accesso root sull'host. Questa annotazione viene utilizzata in questo caso perché il carico di lavoro specifico in esecuzione richiede l'accesso root. Nello specifico, un'operazione di cancellazione della cache eseguita dal carico di lavoro richiede l'accesso root. Che questo sia o meno `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 processo creato nel passaggio 1 venga eseguito correttamente. Il seguente comando di esempio conferma che è stato creato un singolo pod per il job, come specificato nella definizione del job, e che questo pod è attualmente in esecuzione su uno dei nodi worker 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 processo creato nel passaggio 1 venga 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. **Facoltativo:** Pulisci gli artefatti del lavoro. I seguenti comandi di esempio mostrano l'eliminazione dell'oggetto processo creato nel passaggio 1.

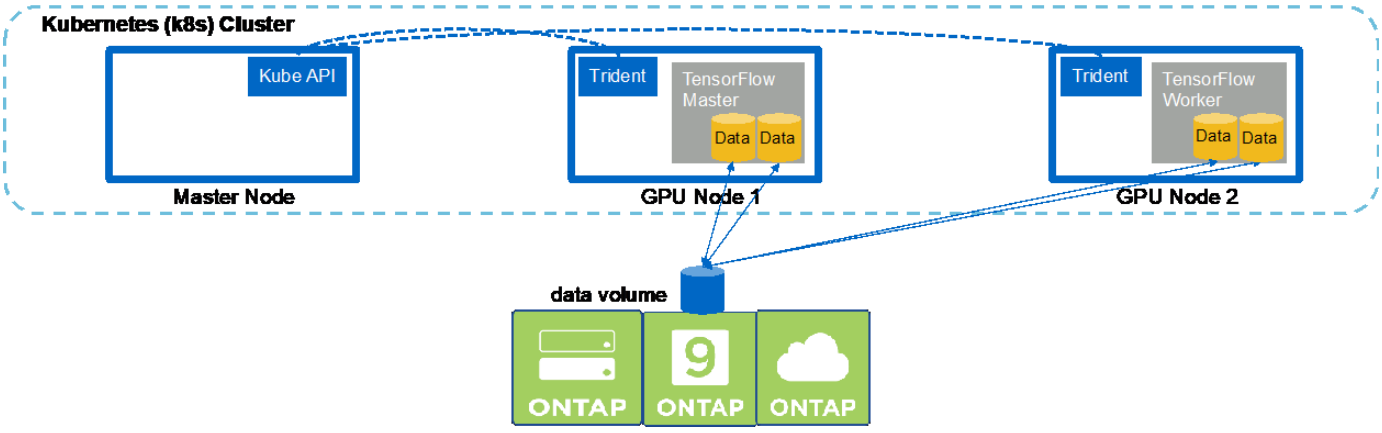
   Quando elimini l'oggetto job, Kubernetes elimina automaticamente tutti i pod associati.

```
$ 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 di intelligenza artificiale distribuito sincrono

Per eseguire un processo di intelligenza artificiale e apprendimento automatico multinodo sincrono nel cluster Kubernetes, eseguire le seguenti attività sull'host di jump di distribuzione. Questo processo consente di sfruttare i dati archiviati su un volume NetApp e di utilizzare più GPU di quelle che un singolo nodo worker può fornire. Per una rappresentazione di un lavoro di intelligenza artificiale distribuita sincrona, vedere la figura seguente.

I lavori distribuiti sincroni possono contribuire ad aumentare le prestazioni e la precisione della formazione rispetto ai lavori distribuiti asincroni. Una discussione sui pro e contro dei lavori sincroni rispetto a quelli asincroni esula dallo scopo di questo documento.



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

Questa distribuzione di worker di esempio richiede otto GPU e può quindi essere eseguita su un singolo nodo worker GPU che dispone di otto o più GPU. Se i nodi worker GPU dispongono di più di otto GPU, per massimizzare le prestazioni potresti voler aumentare questo numero in modo che corrisponda al numero di GPU presenti nei nodi worker. Per ulteriori informazioni sulle distribuzioni di Kubernetes, vedere "documentazione ufficiale di Kubernetes" .

In questo esempio viene creata una distribuzione Kubernetes perché questo specifico worker containerizzato non verrebbe mai completato da solo. Pertanto, non ha senso distribuirlo utilizzando la struttura dei job di Kubernetes. Se il tuo worker è progettato o scritto per completarsi da solo, allora potrebbe essere sensato utilizzare la struttura del lavoro per distribuire il tuo worker.

Al pod specificato in questa specifica di distribuzione di esempio viene assegnato un `hostNetwork` valore di `true` . Questo valore indica che il pod utilizza lo stack di rete del nodo worker host anziché lo stack di rete virtuale che Kubernetes crea solitamente per ogni 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 modo sincrono e distribuito. Pertanto, richiede l'accesso allo stack di rete host. Una discussione su Open MPI, NCCL e Horovod esula dallo scopo di questo documento. Che questo sia o meno `hostNetwork: true` l'annotazione è necessaria a seconda dei requisiti del carico di lavoro specifico che si sta eseguendo. Per maggiori informazioni sul `hostNetwork` campo, vedere il "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. Verifica che la distribuzione del worker creata nel passaggio 1 sia stata avviata correttamente. I seguenti comandi di esempio confermano che è stato creato un singolo pod worker per la distribuzione, come indicato nella definizione di distribuzione, e che questo pod è attualmente in esecuzione su uno dei nodi worker 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 job Kubernetes per un master che avvia, partecipa e monitora l'esecuzione del job multinodo sincrono. I seguenti comandi di esempio creano un master che avvia, partecipa e tiene traccia dell'esecuzione sincrona distribuita dello stesso processo di benchmark TensorFlow eseguito su un singolo

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

Questo esempio di master job richiede otto GPU e può quindi essere eseguito su un singolo nodo worker GPU dotato di otto o più GPU. Se i nodi worker GPU dispongono di più di otto GPU, per massimizzare le prestazioni potresti voler aumentare questo numero in modo che corrisponda al numero di GPU presenti nei nodi worker.

Al pod master specificato in questa definizione di lavoro di esempio viene assegnato un `hostNetwork` valore di `true` , proprio come è stato dato al pod dei lavoratori un `hostNetwork` valore di `true` nel passaggio 1. Per maggiori dettagli 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 processo master creato nel passaggio 3 sia in esecuzione correttamente. Il seguente comando di esempio conferma che è stato creato un singolo pod master per il job, come indicato nella definizione del job, e che questo pod è attualmente in esecuzione su uno dei nodi worker GPU. Dovresti anche vedere che il pod worker che hai visto originariamente nel passaggio 1 è ancora in esecuzione e che i pod master e worker 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 processo master creato nel passaggio 3 venga 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. Elimina la distribuzione del worker quando non ti serve più. I seguenti comandi di esempio mostrano l'eliminazione dell'oggetto di distribuzione del worker creato nel passaggio 1.

   Quando elimini l'oggetto di distribuzione del worker, Kubernetes elimina automaticamente tutti i pod worker 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. **Facoltativo:** ripulisci gli artefatti del lavoro master. I seguenti comandi di esempio mostrano l'eliminazione dell'oggetto master del processo creato nel passaggio 3.

   Quando elimini l'oggetto master job, Kubernetes elimina automaticamente tutti i master pod 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