MLOps open source avec NetApp

NetApp Solutions

NetApp
May 10, 2024

Sommaire

MLOps open source avec NetApp

MLOps open source avec NetApp

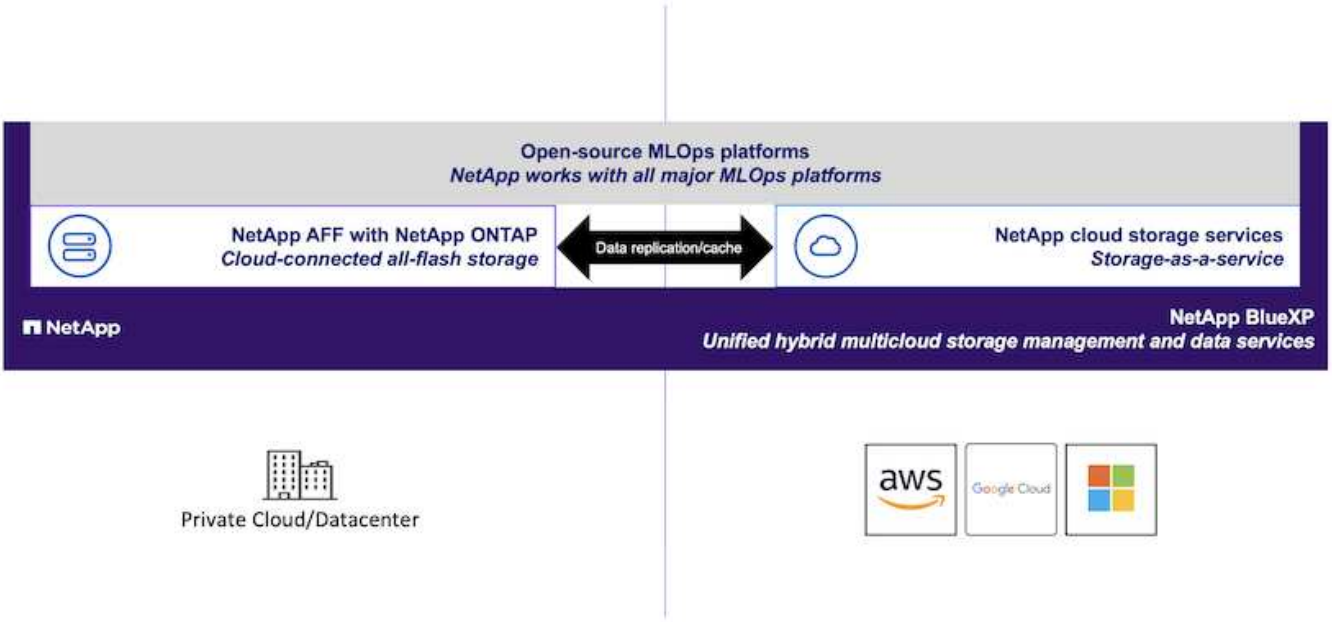
Mike Oglesby, NetApp
Mohan Acharya, NetApp

Toutes les entreprises, quelle que soit leur taille et leurs secteurs, se tournent vers l'intelligence artificielle (IA), le machine learning (ML) et le deep learning (DL) pour résoudre des problèmes concrets, proposer des produits et des services innovants et se démarquer sur un marché de plus en plus concurrentiel. Alors que les entreprises ont de plus en plus recours à l'IA, AU ML et au DL, elles sont confrontées à de nombreux défis, notamment l'évolutivité des workloads et la disponibilité des données. Cette solution vous montre comment relever ces défis en associant les fonctionnalités de gestion de données NetApp à des outils et des frameworks open source populaires.

Cette solution a pour objectif de démontrer plusieurs outils et frameworks open source pouvant être intégrés à un workflow MLOps. Ces différents outils et structures peuvent être utilisés ensemble ou eux-mêmes en fonction des exigences et du cas d'utilisation.

Cette solution propose les outils/structures suivants :

- "Débit d'air Apache"
- "Kubeflow"



Présentation de la technologie

L'intelligence artificielle

L'IA est une discipline scientifique informatique dans laquelle les ordinateurs sont entraînés pour imiter les

fonctions cognitives de l'esprit humain. Les développeurs d'IA entraînent des ordinateurs pour apprendre et résoudre les problèmes de manière similaire, voire supérieure, à celle des humains. L'apprentissage profond et l'apprentissage machine sont des sous-domaines de l'IA. Les entreprises adoptent de plus en plus l'IA, LE ML et le DL pour répondre à leurs besoins stratégiques. Voici quelques exemples :

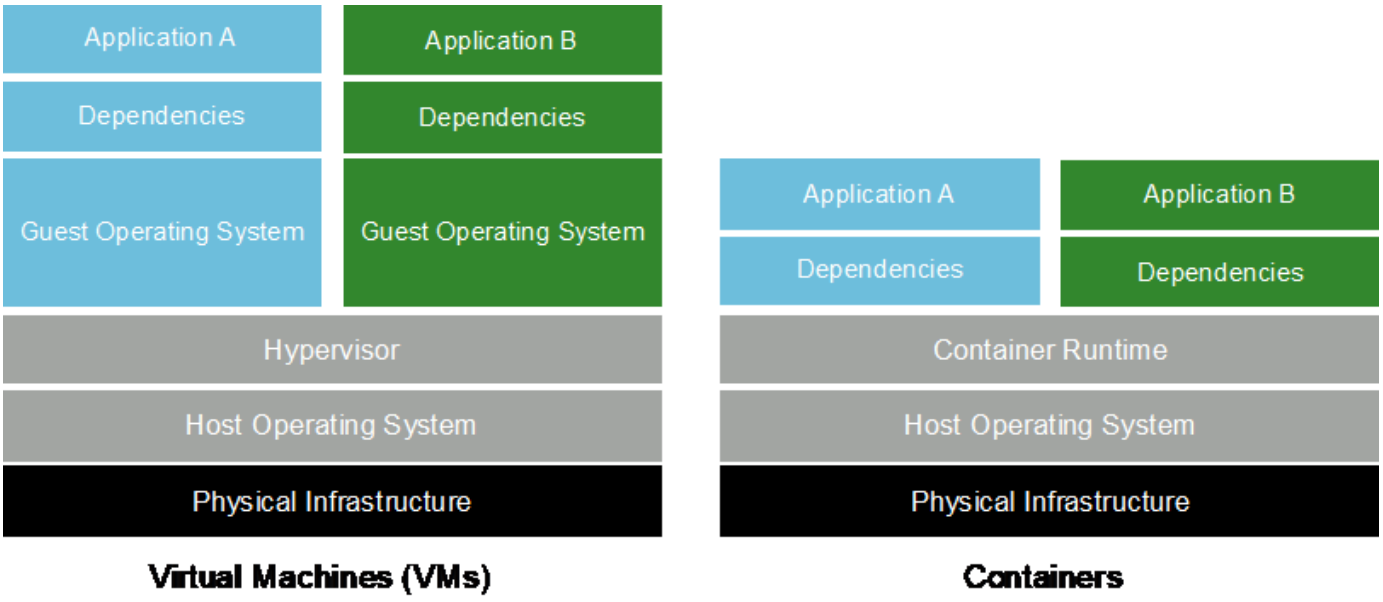
- Analyser d'importants volumes de données pour obtenir des informations stratégiques inexistantes
- Interagir directement avec les clients à l'aide du traitement du langage naturel
- Automatisation de divers processus et fonctions métier

Les workloads d'entraînement et d'inférence d'IA modernes requièrent des fonctionnalités de calcul parallèle. Par conséquent, les GPU sont de plus en plus utilisés pour exécuter les opérations d'IA, car les capacités de traitement parallèle des GPU sont largement supérieures à celles des processeurs génériques.

Conteneurs

Les conteneurs sont des instances isolées de l'espace utilisateur qui s'exécutent sur un noyau de système d'exploitation hôte partagé. Les conteneurs se généraaffichent de plus en plus rapidement. Les conteneurs offrent bon nombre des avantages de la boxe applicative offerts par les machines virtuelles. Cependant, les couches de l'hyperviseur et du système d'exploitation invité sur lesquelles reposent les machines virtuelles ont été éliminées, les conteneurs sont beaucoup plus légers. La figure suivante montre une visualisation des machines virtuelles par rapport aux conteneurs.

Les conteneurs permettent également de packaging efficace des dépendances entre applications, des durées d'exécution, etc., directement avec une application. Le format de conditionnement de conteneurs le plus utilisé est le container Docker. Une application conteneurisée dans le format de conteneur Docker peut être exécutée sur n'importe quel ordinateur capable d'exécuter des conteneurs Docker. Cela est vrai même si les dépendances de l'application ne sont pas présentes sur la machine car toutes les dépendances sont conditionnées dans le conteneur lui-même. Pour plus d'informations, consultez la "Site Web de Docker".



Kubernetes

Kubernetes est une plateforme open source d'orchestration de conteneurs distribuée, conçue à l'origine par Google, et désormais gérée par Cloud Native Computing Foundation (CNCF). Kubernetes permet d'automatiser le déploiement, la gestion et l'évolutivité des fonctions pour les applications conteneurisées.

Depuis quelques années, Kubernetes devient la plateforme dominante d'orchestration de conteneurs. Pour plus d'informations, consultez la "Site Web de Kubernetes".

NetApp Astra Trident

ASTRA Trident permet la consommation et la gestion des ressources de stockage sur toutes les plateformes de stockage NetApp populaires, dans le cloud public ou sur site, y compris ONTAP (AFF, FAS, Select, Cloud, Amazon FSX pour NetApp ONTAP), Element (NetApp HCI, SolidFire), Azure NetApp Files service et Cloud Volumes Service sur Google Cloud. ASTRA Trident est un orchestrateur de stockage dynamique conforme à CSI (Container Storage interface) qui s'intègre de manière native à Kubernetes.

Kit NetApp DataOps

Le "Kit NetApp DataOps" Est un outil Python qui simplifie la gestion des espaces de travail de développement/formation et des serveurs d'inférence pris en charge par un stockage NetApp haute performance et scale-out. Les fonctionnalités principales comprennent :

- Provisionnez rapidement de nouveaux espaces de travail de grande capacité grâce à un stockage NetApp haute performance et scale-out.
- Clonez quasi instantanément des espaces de travail de grande capacité afin de permettre l'expérimentation ou l'itération rapide.
- Sauvegardez de manière quasi instantanée des copies Snapshot d'espaces de travail haute capacité pour la sauvegarde et/ou la traçabilité/la référence.
- Provisionnement, clonage et snapshot quasi instantanés des volumes de données haute capacité haute performance.

Kubeflow

Kubeflow est un kit d'IA et DE ML open source pour Kubernetes qui a été développé à l'origine par Google. Le projet Kubeflow permet de déployer des workflows d'IA et DE ML sur Kubernetes de façon simple, portable et évolutive. Kubeflow s'affranchit des complexités de Kubernetes, ce qui permet aux data Scientists de se concentrer sur ce qu'ils connaissent le mieux — data science. Voir la figure suivante pour une visualisation. Kubeflow est une bonne option open source pour les entreprises qui préfèrent une plateforme MLOps tout-en-un. Pour plus d'informations, consultez la "Site web Kubeflow".

Kubeflow pipelines

Kubeflow pipelines est un composant clé de Kubeflow. Kubeflow pipelines est une plateforme et une norme pour définir et déployer des workflows d'IA et DE ML portables et évolutifs. Pour plus d'informations, reportez-vous à la section "Documentation officielle Kubeflow".

Jupyter Notebook Server

Un Jupyter Notebook Server est une application web open source qui permet aux data Scientists de créer des documents de type wiki appelés ordinateurs portables Jupyter contenant du code en direct ainsi que des tests descriptifs. Les ordinateurs portables Jupyter sont largement utilisés dans la communauté de l'IA et DU ML afin de documenter, de stocker et de partager des projets d'IA et DE ML. Kubeflow simplifie le provisionnement et le déploiement de Jupyter Notebooks Servers sur Kubernetes. Pour plus d'informations sur les ordinateurs portables Jupyter, visitez le "Site Web de Jupyter". Pour plus d'informations sur les ordinateurs portables Jupyter dans le contexte de Kubeflow, consultez le "Documentation officielle Kubeflow".

Katib

Katib est un projet natif Kubernetes pour le machine learning automatisé (AutoML). Katib prend en charge le réglage des hyperparamètres, l'arrêt précoce et la recherche d'architecture neurale (NAS). Katib est un projet indépendant des frameworks de machine learning (ML). Il peut ajuster les hyperparamètres des applications écrits dans n'importe quel langage du choix de l'utilisateur et prendre en charge de manière native de nombreux frameworks de ML, tels que TensorFlow, MXNet, PyTorch, XGBoost, et autres. Katib prend en charge de nombreux algorithmes AutoML, tels que l'optimisation bayésienne, les estimateurs de l'arbre de Parzen, la recherche aléatoire, la stratégie d'évolution de la matrice de Covariance, l'hyperbande, la recherche efficace d'architecture neurale, la recherche d'architecture différentiable et bien d'autres encore. Pour plus d'informations sur les ordinateurs portables Jupyter dans le contexte de Kubeflow, consultez le "Documentation officielle Kubeflow".

Débit d'air Apache

Apache Airflow est une plateforme de gestion des flux de production open source qui permet de créer des programmes, de planifier et de surveiller des flux de travail d'entreprise complexes. Il est souvent utilisé pour automatiser les workflows ETL et de pipeline de traitement de données, mais ne se limite pas à ces types de flux de travail. Le projet de flux d'air a été lancé par Airbnb mais est depuis devenu très populaire dans l'industrie et est maintenant sous les auspices de la Apache Software Foundation. Le flux d'air est écrit en Python, les flux de production du flux d'air sont créés via des scripts Python, et le flux d'air est conçu selon le principe de « configuration as code ». De nombreux utilisateurs de flux d'air utilisent désormais Kubernetes.

Graphes acycliques dirigés (DAG)

Dans le flux d'air, les flux de travail sont appelés graphiques acycliques dirigés (DAG). Les GDAs sont constitués de tâches exécutées dans l'ordre, en parallèle ou une combinaison des deux, en fonction de la définition DAG. Le planificateur de flux d'air exécute des tâches individuelles sur un ensemble de travailleurs, en respectant les dépendances au niveau des tâches spécifiées dans la définition DAG. Les DAG sont définis et créés par le biais de scripts Python.

NetApp ONTAP

ONTAP 9, la dernière génération de logiciel de gestion du stockage de NetApp, permet aux entreprises de moderniser l'infrastructure et de passer à un data Center prêt pour le cloud. Avec des capacités de gestion des données à la pointe du secteur, ONTAP permet de gérer et de protéger les données avec un seul ensemble d'outils, quel que soit leur emplacement. Vous pouvez aussi déplacer vos données librement partout où elles sont nécessaires : la périphérie, le cœur ou le cloud. ONTAP 9 comprend de nombreuses fonctionnalités qui simplifient la gestion des données, accélèrent et protègent les données stratégiques, et permettent d'utiliser des fonctionnalités d'infrastructure nouvelle génération dans toutes les architectures de cloud hybride.

Gestion simplifiée

La gestion des données est cruciale pour les opérations IT et les data Scientists, de sorte que les ressources appropriées sont utilisées pour les applications d'IA et pour l'entraînement des datasets d'IA/DE ML. Les informations supplémentaires suivantes sur les technologies NetApp ne sont pas incluses dans cette validation, mais elles peuvent être pertinentes en fonction de votre déploiement.

Le logiciel de gestion des données ONTAP comprend les fonctionnalités suivantes pour rationaliser et simplifier les opérations et réduire le coût total d'exploitation :

- Compaction des données à la volée et déduplication étendue La compaction des données réduit le gaspillage d'espace à l'intérieur des blocs de stockage, et la déduplication augmente considérablement la capacité effective. Cela s'applique aux données stockées localement et à leur placement dans le cloud.

- Qualité de service (AQoS) minimale, maximale et adaptative. Les contrôles granulaires de la qualité de service (QoS) permettent de maintenir les niveaux de performance des applications stratégiques dans des environnements hautement partagés.
- NetApp FabricPool Tiering automatique des données inactives vers des options de stockage de cloud public et privé, notamment Amazon Web Services (AWS), Azure et la solution de stockage NetApp StorageGRID. Pour plus d'informations sur FabricPool, voir "Tr-4598 : meilleures pratiques de FabricPool".

Accélération et protection des données

ONTAP offre des niveaux supérieurs de performances et de protection des données et étend ces fonctionnalités aux méthodes suivantes :

- Des performances élevées et une faible latence. ONTAP offre le débit le plus élevé possible à la latence la plus faible possible.
- Protection des données. ONTAP fournit des fonctionnalités de protection des données intégrées avec une gestion commune sur toutes les plateformes.
- NetApp Volume Encryption (NVE). ONTAP offre un chiffrement natif au niveau du volume avec un support de gestion des clés interne et externe.
- Colocation et authentification multifacteur. ONTAP permet le partage des ressources d'infrastructure avec les plus hauts niveaux de sécurité.

Une infrastructure pérenne

ONTAP permet de répondre aux besoins métier en constante évolution grâce aux fonctionnalités suivantes :

- Évolutivité transparente et opérations non disruptives. ONTAP prend en charge l'ajout non disruptif de capacité aux contrôleurs et l'évolution scale-out des clusters. Les clients peuvent effectuer la mise à niveau vers les technologies les plus récentes, telles que NVMe et FC 32 Gb, sans migration des données ni panne coûteuse.
- Connexion cloud. ONTAP est le logiciel de gestion de stockage le plus connecté au cloud, avec des options de stockage Software-defined et les instances cloud natives dans tous les clouds publics.
- Intégration avec les applications émergentes ONTAP propose des services de données d'entreprise pour les plateformes et applications nouvelle génération, telles que les véhicules autonomes, les Smart cities et Industry 4.0, en utilisant la même infrastructure prenant en charge les applications d'entreprise existantes.

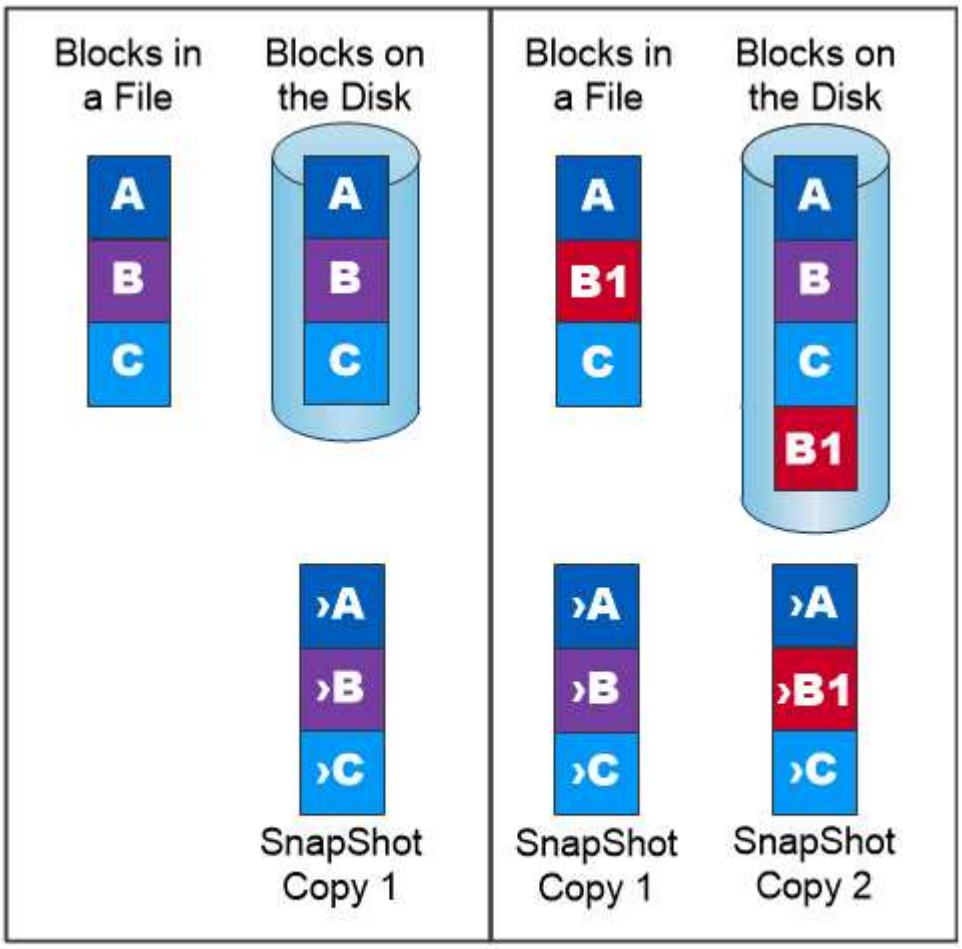
Copies NetApp Snapshot

Une copie NetApp Snapshot est une image ponctuelle en lecture seule d'un volume. La consommation d'espace de stockage de l'image est minime et l'impact sur les performances est négligeable, car elle enregistre uniquement les modifications apportées aux fichiers depuis la dernière copie Snapshot, comme illustré dans la figure ci-dessous.

Les copies Snapshot doivent optimiser leur efficacité par rapport à la technologie de virtualisation de base du stockage ONTAP, WAFL (Write Anywhere File Layout). Tout comme une base de données, WAFL utilise des métadonnées pour désigner des blocs de données réels sur le disque. Contrairement à une base de données, WAFL ne remplace pas les blocs existants. Il écrit les données mises à jour sur un nouveau bloc et modifie les métadonnées. C'est parce que ONTAP référence les métadonnées lorsqu'il crée une copie Snapshot, plutôt que de copier des blocs de données, ces copies sont si efficaces. Vous éliminez ainsi les temps de recherche engendrés par d'autres systèmes pour localiser les blocs à copier, et par ailleurs le coût d'une copie.

Vous pouvez utiliser une copie Snapshot pour restaurer des fichiers ou des LUN individuels, ou pour restaurer l'ensemble du contenu d'un volume. ONTAP compare les informations du pointeur de la copie Snapshot aux

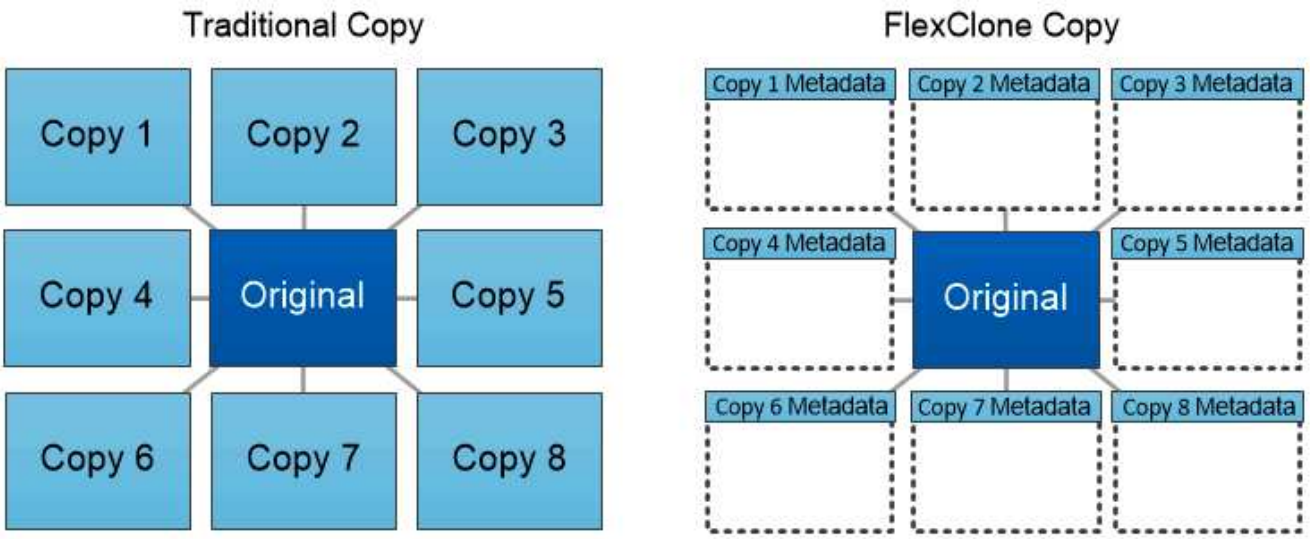
données d'un disque pour reconstruire l'objet manquant ou endommagé, sans temps d'indisponibilité ni coûts de performance significatifs.



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

Technologie NetApp FlexClone

La technologie NetApp FlexClone référence les métadonnées Snapshot pour créer des copies inscriptibles instantanées d'un volume. Les copies partagent les blocs de données avec leurs parents. Aucun stockage n'est utilisé, sauf pour les métadonnées, jusqu'à ce que les modifications soient écrites sur la copie, comme illustré dans la figure ci-dessous. Là où les copies classiques peuvent prendre des minutes, voire des heures, pour créer des copies, FlexClone vous permet de copier même les jeux de données les plus volumineux quasi instantanément. C'est pourquoi il est idéal si vous avez besoin de plusieurs copies de jeux de données identiques (un espace de travail de développement, par exemple) ou de copies temporaires d'un jeu de données (afin de tester une application par rapport à un jeu de données de production).



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

Technologie de réplication des données NetApp SnapMirror

Le logiciel NetApp SnapMirror est une solution de réplication unifiée économique et facile à utiliser dans l'environnement Data Fabric. Il réplique les données à haute vitesse sur un WAN ou un LAN. Elle vous assure haute disponibilité et une réplication rapide des données pour les applications de tous types, y compris les applications stratégiques dans les environnements classiques et virtuels. En répliquant vos données sur un ou plusieurs systèmes de stockage NetApp, puis en les mettant régulièrement à jour, vous disposez de données actualisées et accessibles dès que vous en avez besoin. Aucun serveur de réplication externe n'est requis. Voir la figure suivante pour un exemple d'architecture exploitant la technologie SnapMirror.

Le logiciel SnapMirror valorise l'efficacité du stockage NetApp ONTAP en n'envoyant que les blocs modifiés sur le réseau. Il utilise également la compression réseau intégrée pour accélérer le transfert de données et réduire l'utilisation de la bande passante jusqu'à 70 %. Avec la technologie SnapMirror, vous pouvez exploiter un flux de données de réplication fine pour créer un référentiel unique qui administre les copies du miroir actif et les copies instantanées antérieures, réduisant ainsi le trafic du réseau jusqu'à 50 %.

Copie et synchronisation NetApp BlueXP

BlueXP Copy and Sync est un service NetApp qui permet une synchronisation sûre et rapide des données. Qu'il s'agisse de transférer des fichiers entre des partages de fichiers NFS ou SMB sur site, NetApp StorageGRID, NetApp ONTAP S3, NetApp Cloud Volumes Service, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage ou IBM Cloud Object Storage, BlueXP Copy and Sync déplace les fichiers où vous le souhaitez, rapidement et en toute sécurité.

Une fois vos données transférées, elles peuvent être utilisées à la source et à la cible. BlueXP Copy and Sync peut synchroniser des données à la demande lorsqu'une mise à jour est déclenchée ou lorsque les données sont synchronisées en continu sur la base d'un calendrier prédéfini. Quoi qu'il en soit, la copie et la synchronisation BlueXP ne déplacent que les données modifiées, ce qui réduit le temps et l'argent consacrés à la réplication des données.

BlueXP Copy and Sync est un outil SaaS extrêmement simple à configurer et à utiliser. Les transferts de données déclenchés par la copie et la synchronisation BlueXP sont effectués par les courtiers de données. Les courtiers de données BlueXP Copy and Sync peuvent être déployés dans AWS, Azure, Google Cloud

Platform ou sur site.

NetApp XCP

NetApp XCP est un logiciel client pour les migrations de données et les informations relatives au système de fichiers de tout type à NetApp et NetApp à NetApp. XCP a été conçu pour évoluer et atteindre des performances maximales en exploitant toutes les ressources système disponibles pour gérer des datasets à grand volume et des migrations haute performance. XCP vous aide à obtenir une visibilité complète sur le système de fichiers avec la possibilité de générer des rapports.

NetApp XCP est disponible dans un pack unique qui prend en charge les protocoles NFS et SMB. XCP inclut un binaire Linux pour les jeux de données NFS et un exécutable Windows pour les jeux de données SMB.

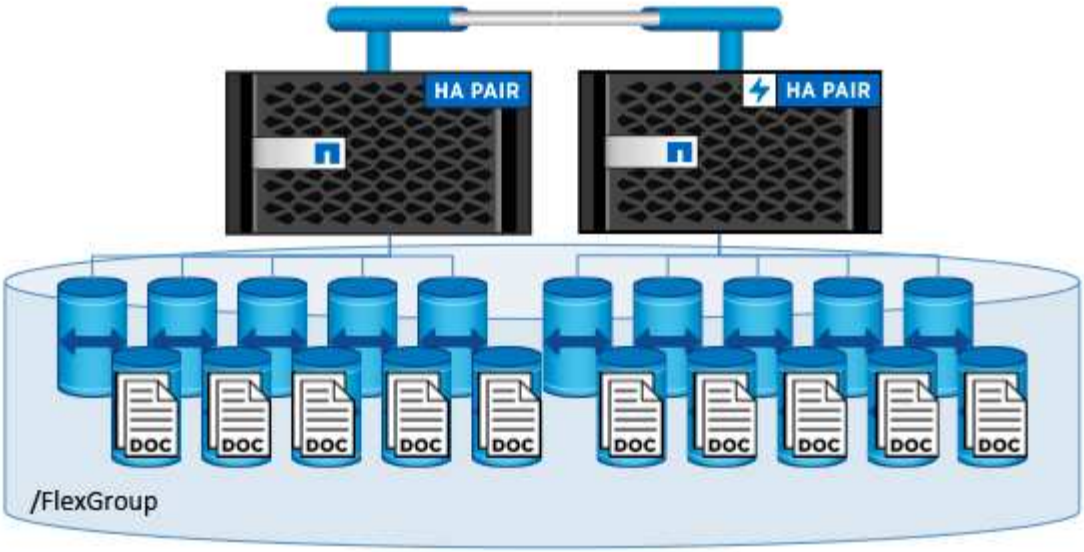
NetApp XCP File Analytics est un logiciel basé sur l'hôte qui détecte les partages de fichiers, exécute les analyses sur le système de fichiers et fournit un tableau de bord pour l'analytique des fichiers. XCP File Analytics est compatible avec les systèmes NetApp et non NetApp et s'exécute sur des hôtes Linux ou Windows pour fournir des analyses des systèmes de fichiers exportés NFS et SMB.

NetApp ONTAP FlexGroup volumes

Un dataset d'entraînement peut être un ensemble de milliards de fichiers. Les fichiers peuvent inclure du texte, de l'audio, de la vidéo et d'autres formes de données non structurées qui doivent être stockées et traitées pour être lues en parallèle. Le système de stockage doit stocker un grand nombre de petits fichiers et doit lire ces fichiers en parallèle pour les E/S séquentielles et aléatoires

Un volume FlexGroup est un namespace unique qui comprend plusieurs volumes de membres constitutifs, comme illustré dans la figure suivante. Du point de vue de l'administrateur de stockage, un volume FlexGroup est géré et agit comme un volume NetApp FlexVol. Les fichiers du volume FlexGroup sont alloués aux volumes de membres individuels,et non répartis entre les volumes ou les nœuds. Ils présentent de nombreux atouts :

- Les volumes FlexGroup fournissent une capacité de plusieurs pétaoctets et une faible latence prévisible pour les charges de travail comportant un grand nombre de métadonnées.
- Ils prennent en charge jusqu'à 400 milliards de fichiers dans le même namespace.
- Ils prennent en charge les opérations parallélisées dans les charges de travail NAS sur les processeurs, les nœuds, les agrégats et les volumes FlexVol constitutifs.



Architecture

Cette solution ne dépend pas de matériel spécifique. La solution est compatible avec toute appliance de stockage physique, instance Software-defined ou service cloud NetApp, prise en charge par Trident. Par exemple, un système de stockage NetApp AFF, Amazon FSX pour NetApp ONTAP, Azure NetApp Files ou une instance NetApp Cloud Volumes ONTAP. De plus, la solution peut être implémentée sur n'importe quel cluster Kubernetes tant que la version Kubernetes utilisée est prise en charge par Kubeflow et NetApp Astra Trident. Pour obtenir la liste des versions Kubernetes prises en charge par Kubeflow, voir la "Documentation officielle Kubeflow". Pour obtenir la liste des versions de Kubernetes prises en charge par Trident, consultez le "Documentation Trident". Pour plus de détails sur l'environnement utilisé pour valider la solution, reportez-vous aux tableaux suivants.

Composant logiciel	Version
Débit d'air Apache	2.0.1
Graphique Helm à flux d'air Apache	8.0.8
Kubeflow	1.7, déployé via "DeployKF" 0.1.1
Kubernetes	1.26
NetApp Astra Trident	23.07

Assistance

NetApp n'offre pas de prise en charge d'entreprise pour Apache Airflow, Kubeflow ou Kubernetes. Si vous êtes intéressé par une plateforme MLOps entièrement prise en charge, "Contactez NetApp" Découvrez les solutions MLOps entièrement prises en charge proposées conjointement par NetApp et ses partenaires.

Configuration NetApp Astra Trident

Exemple de systèmes back-end Astra Trident pour les déploiements NetApp AIPod

Avant de pouvoir utiliser Astra Trident pour provisionner dynamiquement les ressources de stockage dans votre cluster Kubernetes, vous devez créer un ou plusieurs systèmes back-end Trident. Les exemples suivants représentent différents types de backends que vous pouvez créer si vous déployez des composants de cette solution sur un "Pod NetApp AIPod". Pour plus d'informations sur les systèmes back-end, reportez-vous au "Documentation Astra Trident".

1. NetApp recommande de créer un système de stockage Trident compatible FlexGroup pour votre FlexPod AIPod.

   Les exemples de commandes ci-après illustrent la création d'un back-end Trident compatible FlexGroup pour une machine virtuelle de stockage AIPod. Ce back-end utilise le `ontap-nas-flexgroup` pilote de stockage ONTAP prend en charge deux principaux types de volumes de données : FlexVol et FlexGroup. La taille des volumes FlexVol est limitée (à compter de cette écriture, la taille maximale dépend du déploiement spécifique). Au contraire, les volumes FlexGroup peuvent évoluer de manière linéaire jusqu'à 20 po et 400 milliards de fichiers, fournissant un espace de nom unique qui simplifie considérablement la gestion des données. Par conséquent, les volumes FlexGroup sont optimaux pour les workloads d'IA et DE ML qui s'appuient sur des quantités importantes de données.

   Si vous travaillez avec un volume de données réduit et que vous souhaitez utiliser des volumes FlexVol plutôt que des volumes FlexGroup, vous pouvez créer des systèmes back-end Trident qui utilisent le `ontap-nas` pilote de stockage au lieu du `ontap-nas-flexgroup` pilote de stockage

```
$ 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 recommande également de créer un système back-end Trident compatible FlexVol. Vous pouvez utiliser des volumes FlexVol pour héberger des applications permanentes, stocker des résultats, des résultats, des informations de débogage, etc. Si vous souhaitez utiliser des volumes FlexVol, vous devez créer un ou plusieurs systèmes back-end Trident compatibles avec FlexVol. Les exemples de commandes qui suivent montrent la création d'un seul back-end Trident compatible avec 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 |
+------------------------+--------------------
+--------------------------------------+--------+---------+
```

Exemple de classes de stockage Kubernetes pour les déploiements NetApp AIPod

Avant de pouvoir utiliser Astra Trident pour provisionner dynamiquement des ressources de stockage dans votre cluster Kubernetes, vous devez créer une ou plusieurs classes de stockage Kubernetes. Les exemples suivants représentent différents types de classes de stockage que vous pouvez créer si vous déployez des composants de cette solution sur un "Pod NetApp AIPod". Pour plus d'informations sur les classes de stockage,

consultez le "Documentation Astra Trident".

1. NetApp recommande de créer une classe de stockage pour le système back-end Trident compatible FlexGroup que vous avez créé dans la section "Exemple de systèmes back-end Astra Trident pour les déploiements NetApp AIPod", étape 1. Les exemples de commandes qui suivent montrent la création de plusieurs classes de stockage correspondant aux deux exemples de back-end créés dans la section "Exemple de systèmes back-end Astra Trident pour les déploiements NetApp AIPod", étape 1 - une qui utilise "NFS sur RDMA" et une qui ne le fait pas.

   Pour qu'un volume persistant ne soit pas supprimé lorsque la demande de volume persistant correspondante est supprimée, l'exemple suivant utilise une `reclaimPolicy` valeur de `Retain`. Pour plus d'informations sur le `reclaimPolicy` consultez le champ officiel "Documentation Kubernetes".

   Remarque : l'exemple suivant de classes de stockage utilise une taille de transfert maximale de 262144. Pour utiliser cette taille de transfert maximale, vous devez configurer la taille de transfert maximale sur votre système ONTAP en conséquence. Reportez-vous à la "Documentation ONTAP" pour plus d'informations.

   Remarque : pour utiliser NFS sur RDMA, vous devez configurer NFS sur RDMA sur un système ONTAP. Reportez-vous à la documentation du ONTAP de recherche de niveau 1 pour plus de détails.

   Remarque : dans l'exemple suivant, un back-end spécifique n'est pas spécifié dans le champ StoragePool du fichier de définition 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 recommande également de créer une classe de stockage correspondant au système back-end Trident activé pour FlexVol que vous avez créé dans la section "Exemple de systèmes back-end Astra Trident pour les déploiements AIPod", étape 2. Les exemples de commandes ci-dessous montrent la création d'une classe de stockage unique pour les volumes FlexVol.

   Remarque : dans l'exemple suivant, un back-end particulier n'est pas spécifié dans le champ StoragePool du fichier de définition StorageClass. Lorsque vous utilisez Kubernetes pour gérer des volumes à l'aide de cette classe de stockage, Trident tente d'utiliser n'importe quel back-end disponible qui utilise le système `ontap-nas` conducteur.

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

Kubeflow

Déploiement Kubeflow

Cette section décrit les tâches à effectuer pour déployer Kubeflow dans votre cluster Kubernetes.

Prérequis

Avant d'effectuer l'exercice de déploiement décrit dans cette section, nous supposons que vous avez déjà effectué les tâches suivantes :

1. Vous disposez déjà d'un cluster Kubernetes en fonctionnement et vous exécutez une version de Kubernetes prise en charge par la version Kubeflow que vous envisagez de déployer. Pour obtenir la liste des versions Kubernetes prises en charge, reportez-vous aux dépendances de votre version Kubeflow dans le "Documentation officielle Kubeflow".
2. Vous avez déjà installé et configuré NetApp Astra Trident dans votre cluster Kubernetes. Pour plus d'informations sur Astra Trident, consultez le "Documentation Astra Trident".

Définissez la classe de stockage Kubernetes par défaut

Avant de déployer Kubeflow, nous vous recommandons de désigner une classe de stockage par défaut dans votre cluster Kubernetes. Le processus de déploiement Kubeflow peut tenter de provisionner de nouveaux volumes persistants à l'aide de la classe de stockage par défaut. Si aucune classe de stockage n'est désignée comme classe de stockage par défaut, le déploiement risque d'échouer. Pour désigner une classe de stockage par défaut dans votre cluster, effectuez la tâche suivante à partir de l'hôte de démarrage du déploiement. Si vous avez déjà désigné une classe de stockage par défaut dans votre cluster, vous pouvez ignorer cette étape.

1. Désignez une des classes de stockage existantes comme classe de stockage par défaut. Les exemples de commandes ci-dessous montrent la désignation d'une classe de stockage nommée `ontap-ai-flexvols-retain` Comme classe de stockage par défaut.

> Le `ontap-nas-flexgroup` Le type de volume interne Trident a une taille de volume persistant minimale et est relativement élevée. Par défaut, Kubeflow tente de provisionner des demandes de volume qui n'ont que quelques Go. Par conséquent, vous ne devez pas désigner une classe de stockage utilisant le `ontap-nas-flexgroup` Taper back-end comme classe de stockage par défaut pour le déploiement 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
```

Options de déploiement Kubeflow

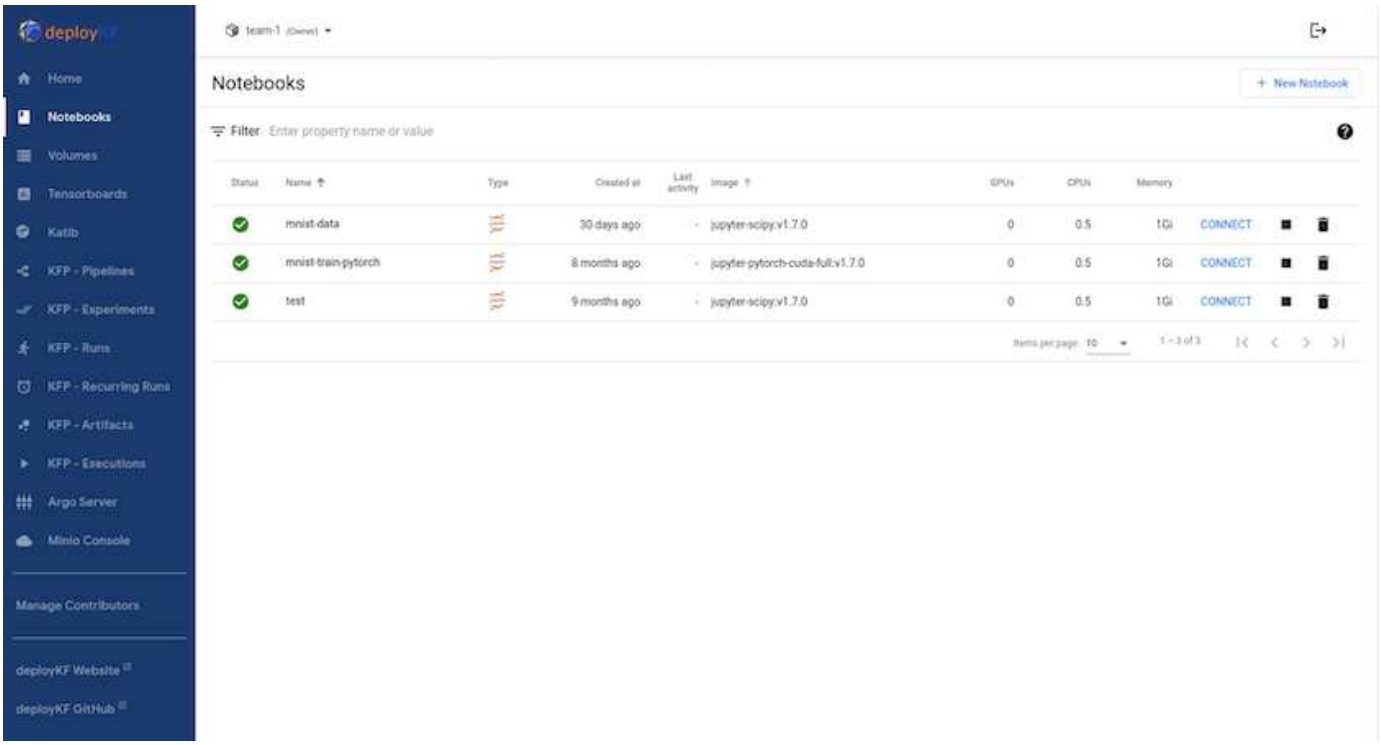
Il existe de nombreuses options de déploiement Kubeflow. Reportez-vous à la "Documentation officielle Kubeflow" pour obtenir une liste d'options de déploiement, choisissez l'option qui correspond le mieux à vos besoins.

> À des fins de validation, nous avons déployé Kubeflow 1.7 avec "DeployKF" 0.1.1.

Exemple d'opérations et de tâches Kubeflow

Provisionner un espace de travail Jupyter Notebook pour un usage Data Scientist ou Developer

Kubeflow est capable de provisionner rapidement de nouveaux serveurs Jupyter Notebook pour agir en tant qu'espaces de travail de data Scientist. Pour plus d'informations sur les ordinateurs portables Jupyter dans le contexte Kubeflow, reportez-vous au "Documentation officielle Kubeflow".



Utilisez le kit NetApp DataOps avec Kubeflow

Le "Kit NetApp Data Science Toolkit pour Kubernetes" Peut être utilisé avec Kubeflow. Grâce au kit NetApp Data Science Toolkit avec Kubeflow, il offre les avantages suivants :

- Les data Scientists peuvent effectuer des opérations avancées de gestion des données NetApp, telles que la création de snapshots et de clones, directement depuis un ordinateur portable Jupyter.
- Les opérations avancées de gestion des données NetApp, telles que la création de snapshots et de clones, peuvent être intégrées dans des workflows automatisés à l'aide de la structure Kubeflow pipelines.

Reportez-vous à la "Exemples Kubeflow" Section du référentiel GitHub pour le kit NetApp Data Science pour plus d'informations sur l'utilisation du kit avec Kubeflow.

Exemple de flux de travail - entraînez un modèle de reconnaissance d'image à l'aide de Kubeflow et du kit d'outils NetApp DataOps

Cette section décrit les étapes de la formation et du déploiement d'un réseau neuronal pour la reconnaissance des images à l'aide de Kubeflow et du kit NetApp DataOps. L'objectif est de fournir un exemple de formation intégrant le stockage NetApp.

Prérequis

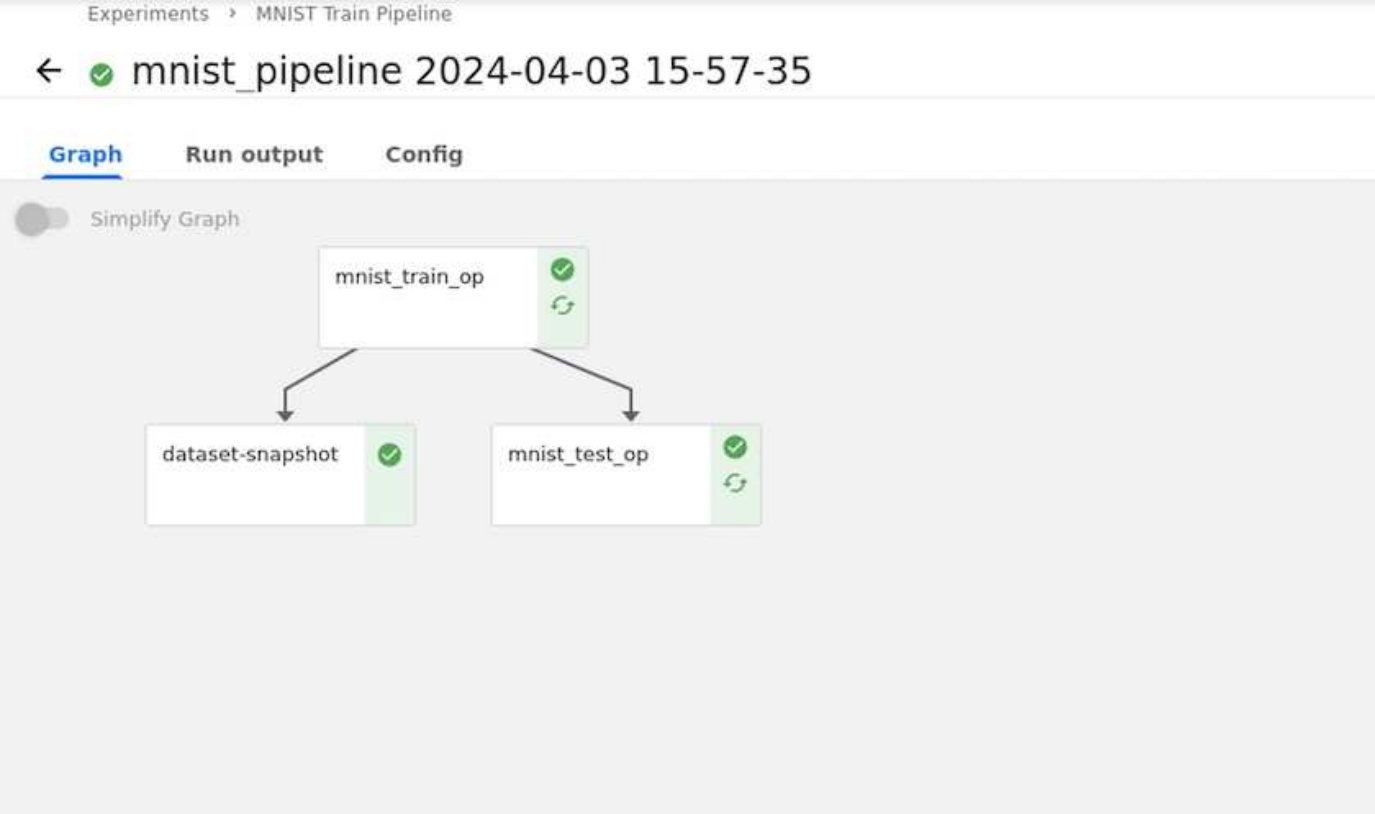
Créez un fichier Dockerfile avec les configurations requises pour les étapes de train et de test dans le pipeline Kubeflow.
Voici un exemple de fichier Dockerfile -

```
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"]
```

En fonction de vos besoins, installez toutes les bibliothèques et tous les packages requis pour exécuter le programme. Avant d'entraîner le modèle de machine learning, il est supposé que vous disposez déjà d'un déploiement Kubeflow fonctionnel.

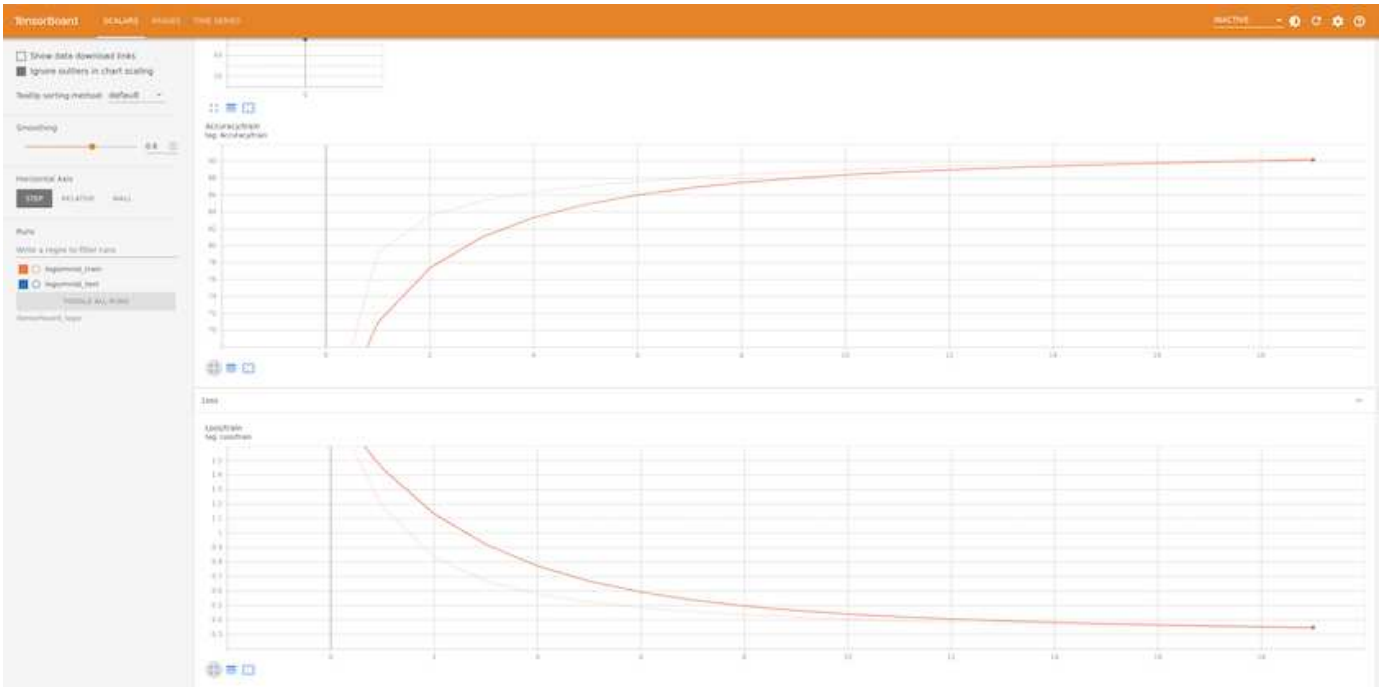
Entraîner un petit NN sur des données MNIST à l'aide de pipelines PyTorch et Kubeflow

Nous utilisons l'exemple d'un petit réseau neuronal formé sur les données MNIST. Le jeu de données MNIST se compose d'images manuscrites de chiffres compris entre 0 et 9. La taille des images est de 28 x 28 pixels. Le dataset est divisé en 60,000 images d'entraînement et 10,000 images de validation. Le réseau neuronal utilisé pour cette expérience est un réseau d'avance à 2 couches. L'entraînement est exécuté à l'aide de Kubeflow pipelines. Reportez-vous à la documentation "ici" pour en savoir plus. Notre pipeline Kubeflow intègre l'image docker de la section Prerequisites.



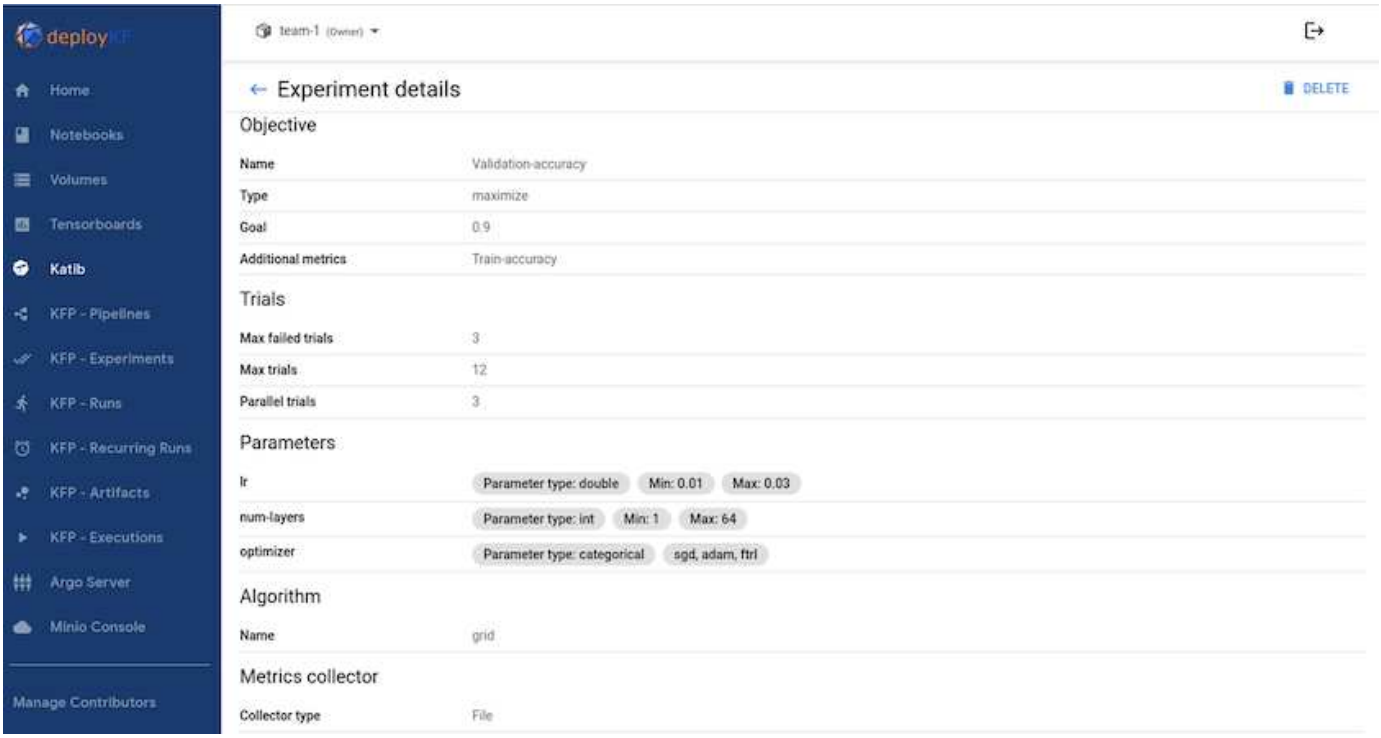
Visualisez les résultats à l'aide de Tensorboard

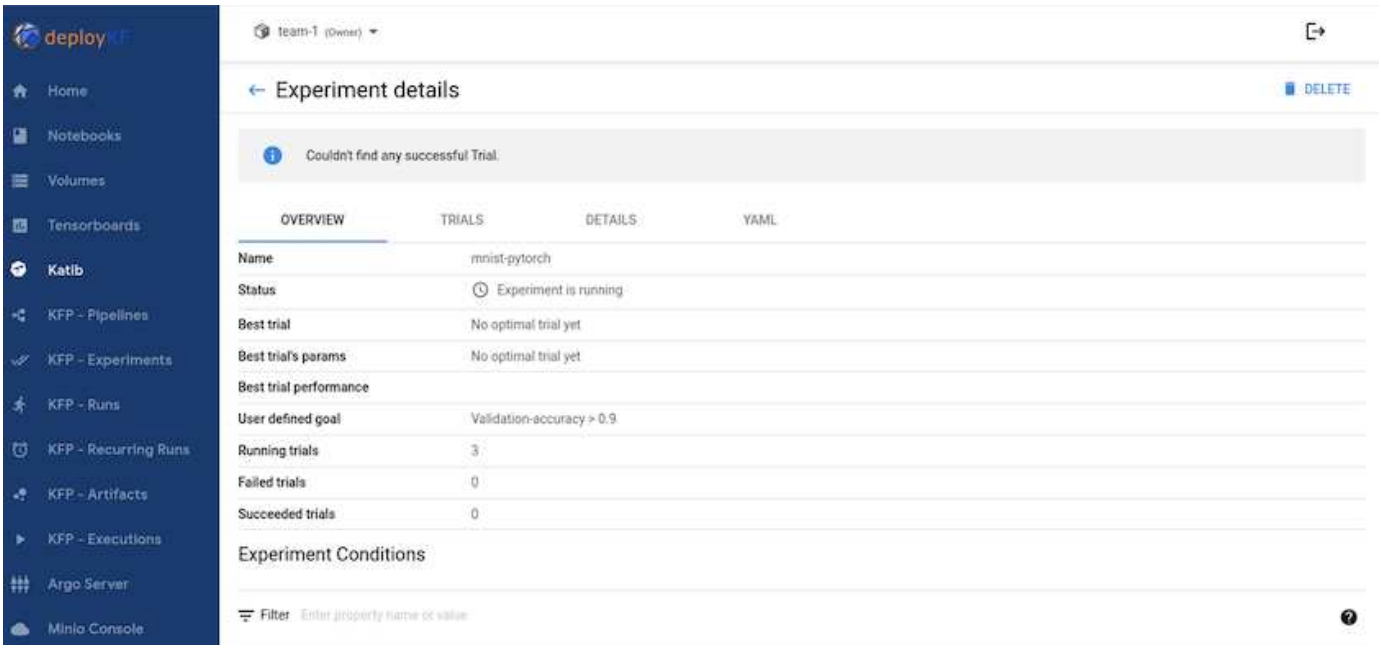
Une fois le modèle entraîné, nous pouvons visualiser les résultats à l'aide de Tensorboard. "Tensorboard" Elle est disponible en tant que fonctionnalité dans le tableau de bord Kubeflow. Vous pouvez créer un tableau de tension personnalisé pour votre travail. Un exemple ci-dessous montre le tracé de la précision de l'entraînement par rapport à nombre de séries de tests et de pertes d'entraînement par rapport à nombre de séries de tests.



Testez les hyperparamètres à l'aide de Katib

"Katib" Est un outil de Kubeflow qui peut être utilisé pour tester les hyperparamètres du modèle. Pour créer une expérience, définissez d'abord une mesure/un objectif souhaité. Il s'agit généralement de la précision du test. Une fois la mesure définie, choisissez les hyperparamètres que vous souhaitez utiliser (optimiseur/apprentissage_rate/nombre de couches). Katib effectue un balayage hyperparamètre avec les valeurs définies par l'utilisateur pour trouver la meilleure combinaison de paramètres qui répondent à la mesure souhaitée. Vous pouvez définir ces paramètres dans chaque section de l'interface utilisateur. Vous pouvez également définir un fichier **YAML** avec les spécifications nécessaires. Vous trouverez ci-dessous une illustration d'une expérience Katib -





Utilisez les instantanés NetApp pour enregistrer les données pour la traçabilité

Lors de l'entraînement du modèle, nous pouvons enregistrer un instantané du dataset d'entraînement à des fins de traçabilité. Pour ce faire, nous pouvons ajouter une étape snapshot au pipeline, comme illustré ci-dessous. Pour créer le snapshot, nous pouvons utiliser le "Kit NetApp DataOps pour 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)
```

Reportez-vous à la "Exemple de kit NetApp DataOps pour Kubeflow" pour en savoir plus.

Débit d'air Apache

Déploiement du flux d'air Apache

Cette section décrit les tâches à effectuer pour déployer l'air dans votre cluster Kubernetes.

> Il est possible de déployer du flux d'air sur d'autres plateformes que Kubernetes. Le déploiement de flux d'air sur des plateformes autres que Kubernetes ne fait pas partie du cadre de cette solution.

Prérequis

Avant d'effectuer l'exercice de déploiement décrit dans cette section, nous supposons que vous avez déjà effectué les tâches suivantes :

1. Vous disposez déjà d'un cluster Kubernetes fonctionnel.
2. Vous avez déjà installé et configuré NetApp Astra Trident dans votre cluster Kubernetes. Pour plus d'informations sur Astra Trident, consultez le "Documentation Astra Trident".

Installer Helm

Le flux d'air est déployé à l'aide de Helm, un gestionnaire de packages populaire pour Kubernetes. Avant de déployer le flux d'air, vous devez installer Helm sur l'hôte de démarrage à déploiement rapide. Pour installer Helm sur l'hôte de saut de déploiement, suivez le "instructions d'installation" Dans la documentation officielle Helm.

Définissez la classe de stockage Kubernetes par défaut

Avant de déployer le flux d'air, vous devez désigner une classe de stockage par défaut dans votre cluster Kubernetes. Le processus de déploiement du flux d'air tente de provisionner de nouveaux volumes persistants à l'aide de la classe de stockage par défaut. Si aucune classe de stockage n'est désignée comme classe de stockage par défaut, le déploiement échoue. Pour désigner une classe de stockage par défaut au sein de votre cluster, suivez les instructions décrites dans le "Déploiement Kubeflow" section. Si vous avez déjà désigné une classe de stockage par défaut dans votre cluster, vous pouvez ignorer cette étape.

Utilisez Helm pour déployer le flux d'air

Pour déployer le flux d'air dans votre cluster Kubernetes à l'aide de Helm, effectuez les tâches suivantes à partir de l'hôte saut de déploiement :

1. Déployer le flux d'air à l'aide de Helm en suivant le "instructions de déploiement" Pour le tableau de débit d'air officiel sur le concentrateur d'artefacts. Les exemples de commandes qui suivent montrent le déploiement du flux d'air à l'aide de Helm. Modifiez, ajoutez et/ou supprimez des valeurs dans `custom-values.yaml` fichier selon votre environnement et la configuration de votre choix.

```
$ 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
```

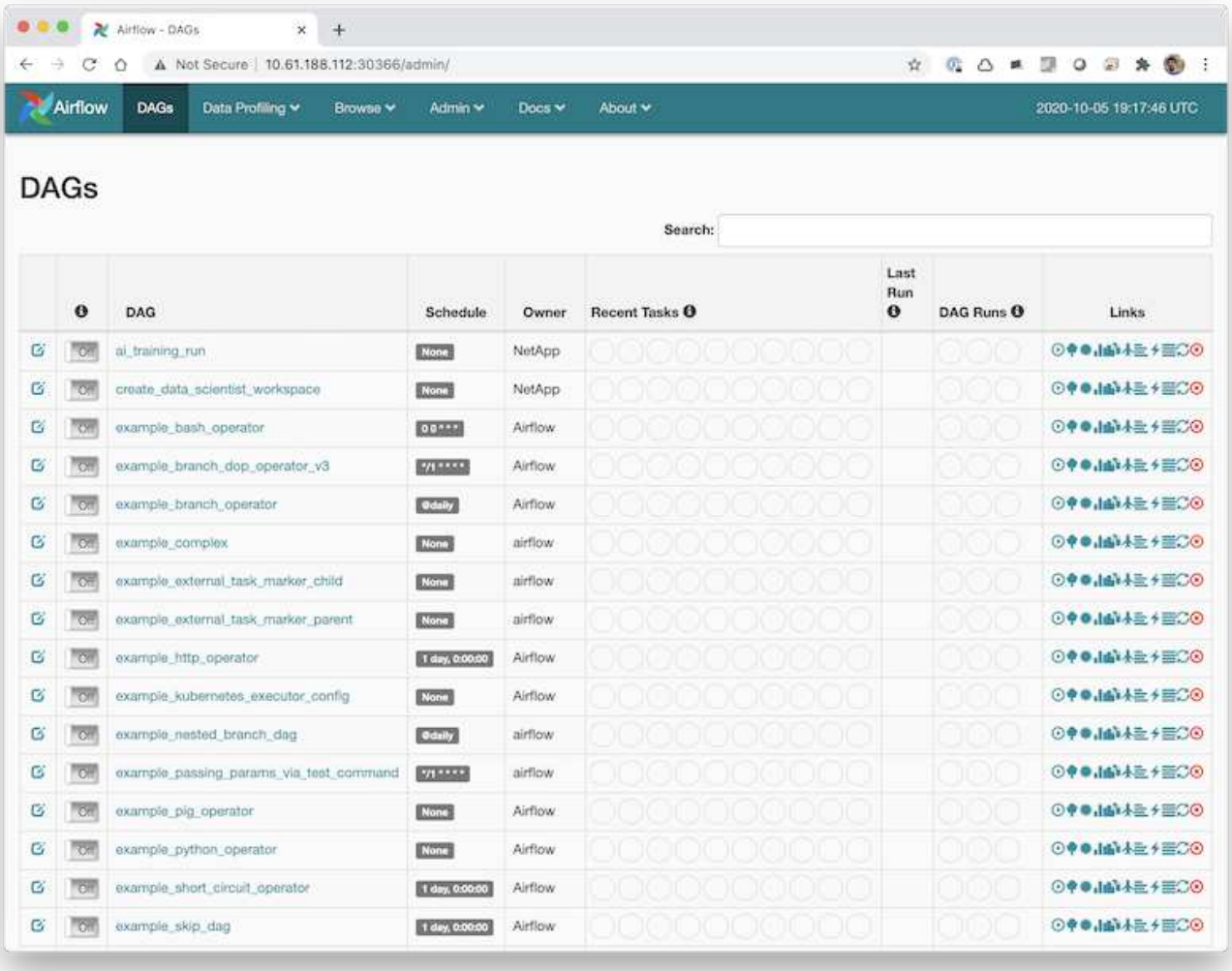
2. Assurez-vous que tous les modules de ventilation sont opérationnels. Le démarrage des modules peut prendre quelques minutes.

```
$ 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. Pour obtenir l'URL du service Web de flux d'air, suivez les instructions qui ont été imprimées sur la console lorsque vous avez déployé du flux d'air à l'aide de Helm à l'étape 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. Confirmez que vous pouvez accéder au service Web de débit d'air.



Utilisez le kit NetApp DataOps avec circulation de l'air

Le "Kit NetApp DataOps pour Kubernetes" Peut être utilisé en conjonction avec le débit d'air. L'utilisation du kit NetApp DataOps avec Airflow vous permet d'incorporer des opérations de gestion des données NetApp, telles que la création de snapshots et de clones, dans des workflows automatisés qui sont orchestrés par le flux d'air.

Reportez-vous à la "Exemples de débit d'air" Section du référentiel GitHub du kit NetApp DataOps pour obtenir des informations détaillées sur l'utilisation du kit avec circulation de l'air.

Exemple d'opérations Astra Trident

Cette section contient des exemples d'opérations que vous pouvez exécuter avec Astra Trident.

Importer un volume existant

Si votre système/plateforme de stockage NetApp contient des volumes que vous souhaitez monter sur des conteneurs au sein de votre cluster Kubernetes, mais qui ne sont pas liés aux demandes de volume persistant

dans le cluster, vous devez importer ces volumes. Vous pouvez utiliser la fonctionnalité d'importation de volumes Trident pour importer ces volumes.

Les exemples de commandes qui suivent montrent l'importation d'un volume nommé `pb_fg_all`. Pour plus d'informations sur les ESV, consultez le "Documentation officielle Kubernetes". Pour plus d'informations sur la fonctionnalité d'importation de volume, reportez-vous à la section "Documentation Trident".

An `accessModes` valeur de `ReadOnlyMany` Est spécifié dans les fichiers de spécifications PVC d'exemple. Pour plus d'informations sur le `accessMode` voir "Documentation officielle 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
```

Provisionner un nouveau volume

Vous pouvez utiliser Trident pour provisionner un nouveau volume sur votre système ou plateforme de stockage NetApp.

Provisionnez un nouveau volume à l'aide de kubectl

Les exemples de commandes suivants montrent le provisionnement d'un nouveau volume FlexVol à l'aide de kubectl.

An `accessModes` valeur de `ReadWriteMany` Est spécifié dans l'exemple de fichier de définition de PVC suivant. Pour plus d'informations sur le `accessMode` voir "Documentation officielle 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
```

Provisionnez un nouveau volume à l'aide du kit NetApp DataOps

Vous pouvez également utiliser le kit NetApp DataOps pour Kubernetes pour provisionner un nouveau volume sur votre système ou plateforme de stockage NetApp. Le kit NetApp DataOps pour Kubernetes utilise Trident pour provisionner les volumes, mais simplifie le processus pour l'utilisateur. Reportez-vous à la "documentation" pour plus d'informations.

Exemple de tâches hautes performances pour les déploiements AIPod

Exécutez un workload d'IA à un seul nœud

Pour exécuter une tâche d'IA et DE ML à un seul nœud dans votre cluster Kubernetes, effectuez les tâches suivantes à partir de l'hôte de démarrage du déploiement. Trident vous permet de faire rapidement et facilement un volume de données, potentiellement contenant des pétaoctets de données, accessibles pour un workload Kubernetes. Pour rendre ce volume de données accessible depuis un pod Kubernetes, il vous suffit de spécifier une demande de volume persistant dans la définition du pod.

Dans cette section, vous devez déjà avoir conteneurisé (au format du conteneur Docker) le workload d'IA et DE ML spécifique que vous essayez d'exécuter dans votre cluster Kubernetes.

1. L'exemple de commandes suivant montre la création d'un travail Kubernetes pour un workload de banc d'essai TensorFlow qui utilise le dataset ImageNet. Pour plus d'informations sur le dataset ImageNet, reportez-vous à la "Site Web ImageNET".

   Cet exemple de tâche nécessite huit GPU, soit un nœud worker doté d'au moins huit GPU. Cet exemple de tâche peut être envoyée dans un cluster pour lequel un nœud worker doté d'au moins huit GPU n'est pas présent ou est actuellement occupé par une autre charge de travail. Si c'est le cas, le travail reste à l'état en attente jusqu'à ce qu'un nœud de travail soit disponible.

   En outre, pour optimiser la bande passante de stockage, le volume contenant les données d'entraînement requises est monté deux fois dans le pod que cette tâche crée. Un autre volume est également monté dans le pod. Ce deuxième volume sera utilisé pour stocker les résultats et les mesures. Ces volumes sont référencés dans la définition de travail en utilisant les noms des ESV. Pour plus d'informations sur les tâches Kubernetes, consultez le "Documentation officielle Kubernetes".

   An `emptyDir` volume avec un `medium` valeur de `Memory` est monté sur `/dev/shm` dans le pod créé par cet exemple de travail. La taille par défaut du `/dev/shm` Le volume virtuel créé automatiquement par le runtime des conteneurs Docker peut parfois manquer pour TensorFlow. Montage d'un `emptyDir` comme dans l'exemple suivant, le volume est suffisamment grand `/dev/shm` volume virtuel. Pour plus d'informations sur `emptyDir` les volumes, voir "Documentation officielle Kubernetes".

   Le conteneur unique spécifié dans cet exemple de définition de travail est donné un `securityContext > privileged` valeur de `true`. Cette valeur signifie que le conteneur dispose d'un accès racine sur l'hôte. Cette annotation est utilisée dans ce cas, car la charge de travail spécifique exécutée nécessite un accès racine. Plus précisément, une opération de mise en cache claire exécutée par la charge de travail nécessite un accès racine. Si cela est ou non `privileged: true` l'annotation est nécessaire dépend des exigences de la charge de travail spécifique que vous exécutez.

```
$ 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. Vérifiez que le travail que vous avez créé à l'étape 1 fonctionne correctement. L'exemple de commande suivant confirme qu'un seul pod a été créé pour le travail, comme spécifié dans la définition du travail, et que ce pod s'exécute actuellement sur l'un des nœuds workers 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. Vérifiez que le travail que vous avez créé à l'étape 1 s'est terminé avec succès. L'exemple de commandes suivant confirme que le travail a été terminé avec succès.

```
$ 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. **Facultatif:** nettoyer les artefacts de travail. Les exemples de commandes suivants montrent la suppression de l'objet de travail créé à l'étape 1.

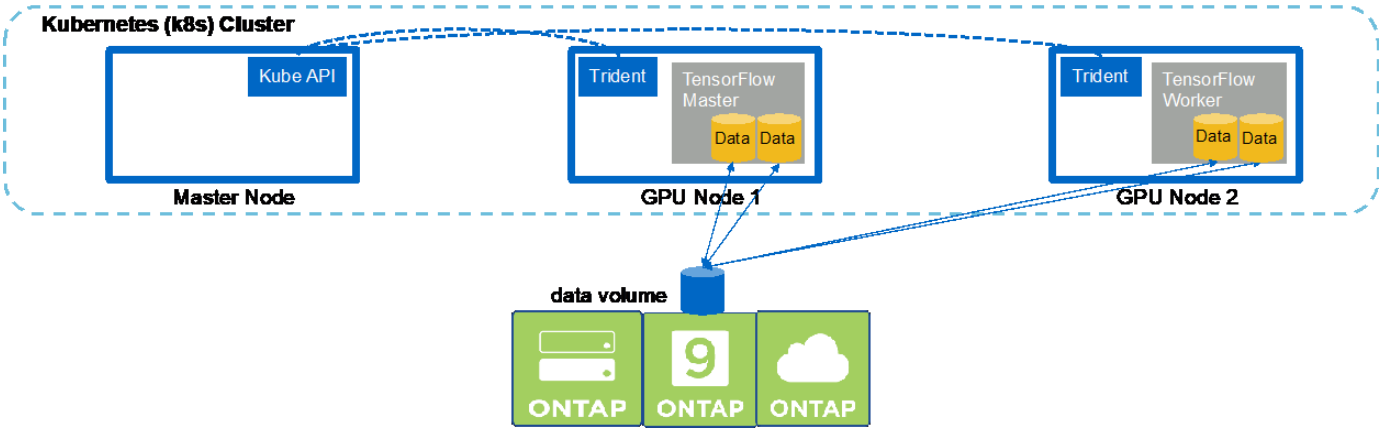
   Lorsque vous supprimez l'objet travail, Kubernetes supprime automatiquement les pods associés.

```
$ 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.
```

Exécutez un workload d'IA distribué synchrone

Pour exécuter une tâche d'IA et DE ML à plusieurs nœuds synchrones dans votre cluster Kubernetes, exécutez les tâches suivantes sur l'hôte de démarrage du déploiement. Ce processus vous permet de exploiter les données stockées sur un volume NetApp et d'utiliser plus de GPU que n'en fournir un seul nœud de travail. Reportez-vous à la figure suivante pour obtenir une description d'une tâche d'IA distribuée synchrone.

Les tâches distribuées synchrones permettent d'améliorer les performances et la précision de l'entraînement par rapport aux tâches distribuées asynchrones. Un examen des avantages et inconvénients des emplois synchrones par rapport aux emplois asynchrones est hors du champ d'application de ce document.



1. Les commandes d'exemple suivantes montrent la création d'un travailleur qui participe à l'exécution distribuée synchrone du même travail de banc d'essai TensorFlow qui a été exécuté sur un seul nœud dans l'exemple de la section "Exécutez un workload d'IA à un seul nœud". Dans cet exemple spécifique, seul un travailleur est déployé car le travail est exécuté sur deux nœuds worker.

Cet exemple de déploiement utilisateur nécessite huit GPU, et peut donc s'exécuter sur un seul nœud worker GPU doté d'au moins huit GPU. Si les nœuds workers GPU disposent de plus de huit GPU, afin d'optimiser les performances, il est possible que vous souhaitiez augmenter ce nombre afin qu'il soit égal au nombre de GPU dont bénéficient les nœuds workers. Pour en savoir plus sur les déploiements Kubernetes, rendez-vous sur le "Documentation officielle Kubernetes".

Un déploiement Kubernetes est créé dans cet exemple, car ce travailleur conteneurisé ne s'en serait jamais achevé seul. C'est pourquoi il n'est pas logique de le déployer via la construction de tâches Kubernetes. Si votre travailleur est conçu ou écrit de manière à le compléter seul, il peut être judicieux d'utiliser la structure de travail pour déployer votre travailleur.

Le pod spécifié dans cet exemple de spécification de déploiement est donné un `hostNetwork` valeur de `true`. Cette valeur signifie que le pod utilise la pile réseau du nœud du worker hôte au lieu de la pile de réseau virtuel que Kubernetes crée habituellement pour chaque pod. Cette annotation est utilisée dans ce cas car la charge de travail spécifique repose sur Open MPI, NCCL et Horovod pour exécuter la charge de travail de façon synchrone distribuée. Par conséquent, elle nécessite l'accès à la pile réseau de l'hôte. Une discussion sur Open MPI, NCCL et Horovod n'est pas dans le cadre du présent document. Si cela est ou non `hostNetwork: true` l'annotation est nécessaire dépend des exigences de la charge de travail spécifique que vous exécutez. Pour plus d'informations sur le `hostNetwork` voir "Documentation officielle 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. Confirmez que le déploiement de collaborateur que vous avez créé à l'étape 1 a été lancé avec succès. Les exemples de commandes suivants confirment qu'un seul pod worker a été créé pour le déploiement, comme indiqué dans la définition du déploiement, et que ce pod s'exécute actuellement sur l'un des nœuds workers 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. Créez un travail Kubernetes pour un master qui démarre, participe et suit l'exécution du travail multinœud synchrone. Les commandes d'exemple suivantes créent un master qui démarre, participe à et assure le suivi de l'exécution distribuée synchrone du même travail de banc d'essai TensorFlow qui a été exécuté

sur un seul nœud dans l'exemple de la section "Exécutez un workload d'IA à un seul nœud".

Cet exemple de tâche maître demande huit GPU, puis peut être exécuté sur un seul nœud worker GPU doté d'au moins huit GPU. Si les nœuds workers GPU disposent de plus de huit GPU, afin d'optimiser les performances, il est possible que vous souhaitiez augmenter ce nombre afin qu'il soit égal au nombre de GPU dont bénéficient les nœuds workers.

Le pod maître spécifié dans cet exemple de définition de travail est donné un `hostNetwork` valeur de `true`, tout comme le pod de travailleur a été donné `hostNetwork` valeur de `true` à l'étape 1. Voir l'étape 1 pour plus de détails sur la raison pour laquelle cette valeur est nécessaire.

```
$ 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. Vérifiez que le travail principal que vous avez créé à l'étape 3 fonctionne correctement. L'exemple de commande suivant confirme qu'un module maître unique a été créé pour le travail, comme indiqué dans la définition du travail, et que ce pod s'exécute actuellement sur l'un des nœuds workers GPU. Vous devriez également voir que le pod de worker que vous avez initialement vu à l'étape 1 est toujours en cours d'exécution et que les pods master et worker exécutent sur différents nœuds.

```
$ 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. Confirmez que le travail principal que vous avez créé à l'étape 3 s'est terminé avec succès. L'exemple de commandes suivant confirme que le travail a été terminé avec succès.

```
$ 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. Supprimez le déploiement de collaborateur lorsque vous n'en avez plus besoin. L'exemple de commandes suivant montre la suppression de l'objet de déploiement de travail qui a été créé à l'étape 1.

   Lorsque vous supprimez l'objet de déploiement worker, Kubernetes supprime automatiquement les pods workers associés.

```
$ 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. **Facultatif:** nettoyez les artefacts du travail principal. Les exemples de commandes suivants montrent la suppression de l'objet de travail maître créé à l'étape 3.

   Lorsque vous supprimez l'objet de travail maître, Kubernetes supprime automatiquement les modules maîtres associés.

```
$ 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.
```

Informations sur le copyright

Informations sur les marques commerciales