MLOps Open Source avec NetApp

NetApp artificial intelligence solutions

NetApp
December 04, 2025

Sommaire

MLOps Open Source avec NetApp

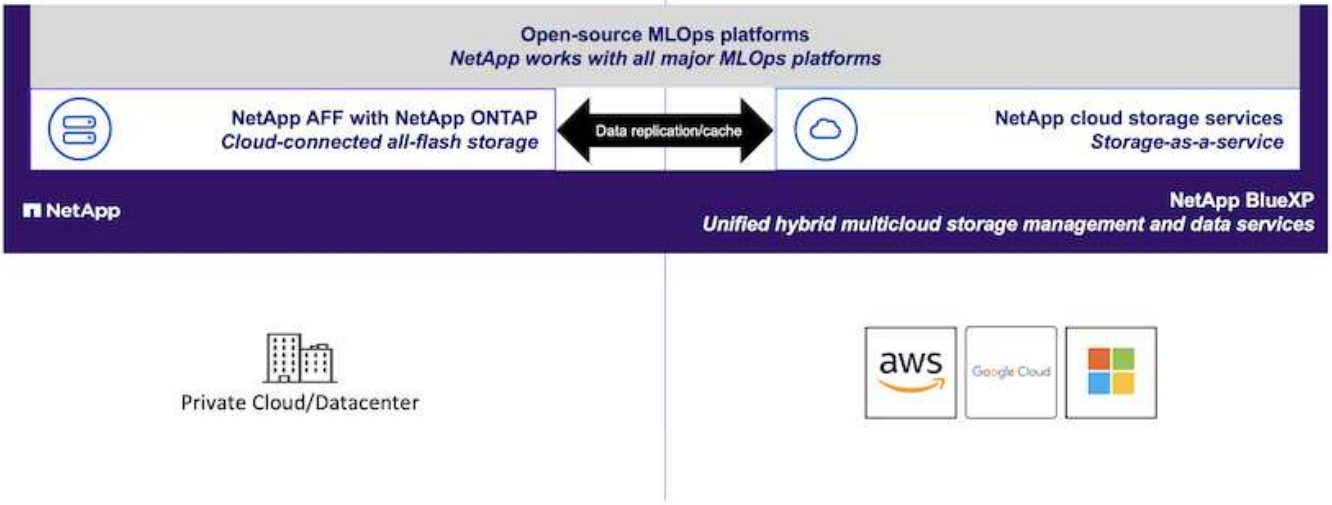
MLOps Open Source avec NetApp

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

Les entreprises et organisations de toutes tailles et de nombreux secteurs se tournent vers l'intelligence artificielle (IA) pour résoudre des problèmes du monde réel, fournir des produits et services innovants et obtenir un avantage sur un marché de plus en plus concurrentiel. De nombreuses organisations se tournent vers des outils MLOps open source afin de suivre le rythme rapide de l'innovation dans le secteur. Ces outils open source offrent des capacités avancées et des fonctionnalités de pointe, mais ne tiennent souvent pas compte de la disponibilité et de la sécurité des données. Malheureusement, cela signifie que les scientifiques de données hautement qualifiés sont obligés de passer beaucoup de temps à attendre d'avoir accès aux données ou à attendre que des opérations rudimentaires liées aux données soient terminées. En associant des outils MLOps open source populaires à une infrastructure de données intelligente de NetApp, les organisations peuvent accélérer leurs pipelines de données, ce qui, à son tour, accélère leurs initiatives d'IA. Ils peuvent exploiter pleinement leurs données tout en garantissant qu'elles restent protégées et sécurisées. Cette solution démontre l'association des capacités de gestion des données NetApp avec plusieurs outils et frameworks open source populaires afin de relever ces défis.

La liste suivante met en évidence certaines fonctionnalités clés activées par cette solution :

- Les utilisateurs peuvent rapidement provisionner de nouveaux volumes de données haute capacité et des espaces de travail de développement soutenus par un stockage NetApp hautes performances et évolutif.
- Les utilisateurs peuvent cloner presque instantanément des volumes de données de grande capacité et des espaces de travail de développement afin de permettre l'expérimentation ou l'itération rapide.
- Les utilisateurs peuvent enregistrer presque instantanément des instantanés de volumes de données de grande capacité et d'espaces de travail de développement à des fins de sauvegarde et/ou de traçabilité/base de référence.



Un flux de travail MLOps typique intègre des espaces de travail de développement, prenant généralement la forme de"Carnets Jupyter" ; suivi des expériences ; pipelines de formation automatisés ; pipelines de données ; et inférence/déploiement. Cette solution met en évidence plusieurs outils et cadres différents qui peuvent être utilisés indépendamment ou conjointement pour traiter les différents aspects du flux de travail. Nous démontrons également l'association des capacités de gestion des données NetApp avec chacun de ces outils. Cette solution vise à offrir des blocs de construction à partir desquels une organisation peut construire un flux de travail MLOps personnalisé et spécifique à ses cas d'utilisation et à ses exigences.

Les outils/frameworks suivants sont couverts dans cette solution :

- "Apache Airflow"
- "JupyterHub"
- "Kubeflow"
- "MLflow"

La liste suivante décrit les modèles courants de déploiement de ces outils indépendamment ou conjointement.

- Déployer JupyterHub, MLflow et Apache Airflow conjointement - JupyterHub pour"Carnets Jupyter" MLflow pour le suivi des expériences et Apache Airflow pour la formation automatisée et les pipelines de données.
- Déployer Kubeflow et Apache Airflow conjointement - Kubeflow pour"Carnets Jupyter" , suivi des expériences, pipelines de formation automatisés et inférence ; et Apache Airflow pour les pipelines de données.
- Déployez Kubeflow en tant que solution de plateforme MLOps tout-en-un pour"Carnets Jupyter" , suivi des expériences, formation automatisée et pipelines de données, et inférence.

Aperçu de la technologie

Cette section se concentre sur l'aperçu technologique pour OpenSource MLOps avec NetApp.

Intelligence artificielle

L'IA est une discipline informatique dans laquelle les ordinateurs sont entraînés à imiter les fonctions cognitives de l'esprit humain. Les développeurs d'IA forment les ordinateurs à apprendre et à résoudre des problèmes d'une manière similaire, voire supérieure, à celle des humains. L'apprentissage profond et l'apprentissage automatique sont des sous-domaines de l'IA. Les organisations adoptent de plus en plus l'IA, le ML et le DL pour répondre à leurs besoins commerciaux critiques. Voici quelques exemples :

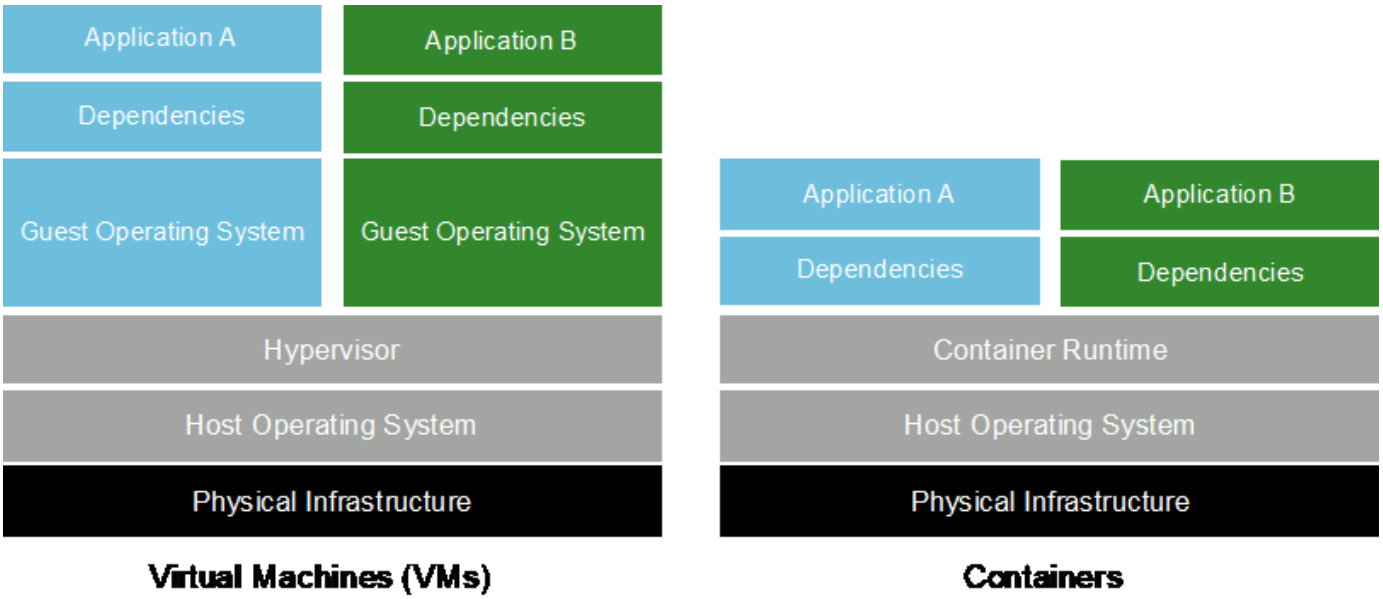
- Analyser de grandes quantités de données pour découvrir des informations commerciales jusqu'alors inconnues
- Interagir directement avec les clients en utilisant le traitement du langage naturel
- Automatisation de divers processus et fonctions métier

Les charges de travail de formation et d'inférence de l'IA moderne nécessitent des capacités de calcul massivement parallèles. Par conséquent, les GPU sont de plus en plus utilisés pour exécuter des opérations d'IA, car les capacités de traitement parallèle des GPU sont largement supérieures à celles des CPU à usage général.

Conteneurs

Les conteneurs sont des instances d'espace utilisateur isolées qui s'exécutent sur un noyau de système d'exploitation hôte partagé. L'adoption des conteneurs augmente rapidement. Les conteneurs offrent de nombreux avantages de sandboxing d'application identiques à ceux des machines virtuelles (VM). Cependant, comme les couches d'hyperviseur et de système d'exploitation invité sur lesquelles s'appuient les machines virtuelles ont été éliminées, les conteneurs sont beaucoup plus légers. La figure suivante illustre une visualisation des machines virtuelles par rapport aux conteneurs.

Les conteneurs permettent également de conditionner efficacement les dépendances des applications, les temps d'exécution, etc., directement avec une application. Le format de packaging de conteneur le plus couramment utilisé est le conteneur Docker. Une application qui a été conteneurisée au format de conteneur Docker peut être exécutée sur n'importe quelle machine 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 regroupées dans le conteneur lui-même. Pour plus d'informations, visitez le "Site Web Docker" .



Kubernetes

Kubernetes est une plate-forme d'orchestration de conteneurs open source et distribuée, conçue à l'origine par Google et désormais maintenue par la Cloud Native Computing Foundation (CNCF). Kubernetes permet l'automatisation des fonctions de déploiement, de gestion et de mise à l'échelle des applications conteneurisées. Ces dernières années, Kubernetes est devenu la plateforme d'orchestration de conteneurs dominante. Pour plus d'informations, visitez le "Site Web Kubernetes" .

NetApp Trident

"Trident"permet la consommation et la gestion des ressources de stockage sur toutes les plates-formes de stockage NetApp populaires, dans le cloud public ou sur site, y compris ONTAP (AFF, FAS, Select, Cloud, Amazon FSx ONTAP), le service Azure NetApp Files et Google Cloud NetApp Volumes. Trident est un orchestrateur de stockage dynamique compatible Container Storage Interface (CSI) qui s'intègre nativement à Kubernetes.

Boîte à outils NetApp DataOps

Le"Boîte à outils NetApp DataOps" est un outil basé sur Python qui simplifie la gestion des espaces de travail de développement/formation et des serveurs d'inférence soutenus par un stockage NetApp hautes performances et évolutif. Les principales fonctionnalités comprennent :

- Provisionnez rapidement de nouveaux espaces de travail haute capacité soutenus par un stockage NetApp hautes performances et évolutif.
- Clonez presque instantanément des espaces de travail de grande capacité afin de permettre l'expérimentation ou l'itération rapide.
- Enregistrez presque instantanément des instantanés d'espaces de travail de grande capacité à des fins de sauvegarde et/ou de traçabilité/de référence.
- Provisionnez, clonez et capturez des instantanés de volumes de données haute capacité et hautes performances de manière quasi instantanée.

Apache Airflow

Apache Airflow est une plate-forme de gestion de flux de travail open source qui permet la création, la planification et la surveillance programmatiques de flux de travail d'entreprise complexes. Il est souvent utilisé pour automatiser les flux de travail ETL et de pipeline de données, mais il ne se limite pas à ces types de flux de travail. Le projet Airflow a été lancé par Airbnb mais est depuis devenu très populaire dans l'industrie et relève désormais des auspices de l'Apache Software Foundation. Airflow est écrit en Python, les flux de travail Airflow sont créés via des scripts Python et Airflow est conçu selon le principe de la « configuration en tant que code ». De nombreux utilisateurs d'Airflow en entreprise exécutent désormais Airflow sur Kubernetes.

Graphes acycliques dirigés (DAG)

Dans Airflow, les flux de travail sont appelés graphes acycliques dirigés (DAG). Les DAG sont constitués de tâches exécutées en séquence, en parallèle ou une combinaison des deux, selon la définition du DAG. Le planificateur Airflow exécute des tâches individuelles sur un ensemble de travailleurs, en adhérant aux dépendances au niveau des tâches spécifiées dans la définition DAG. Les DAG sont définis et créés via des scripts Python.

Carnet Jupyter

Les notebooks Jupyter sont des documents de type wiki qui contiennent du code en direct ainsi que du texte

descriptif. Les notebooks Jupyter sont largement utilisés dans la communauté de l'IA et du ML comme moyen de documenter, de stocker et de partager des projets d'IA et de ML. Pour plus d'informations sur Jupyter Notebooks, visitez le "Site Web Jupyter" .

Serveur de blocs-notes Jupyter

Un serveur Jupyter Notebook est une application Web open source qui permet aux utilisateurs de créer des Jupyter Notebooks.

JupyterHub

JupyterHub est une application multi-utilisateurs qui permet à un utilisateur individuel de provisionner et d'accéder à son propre serveur Jupyter Notebook. Pour plus d'informations sur JupyterHub, visitez le "Site Web JupyterHub" .

MLflow

MLflow est une plate-forme de gestion du cycle de vie de l'IA open source populaire. Les principales fonctionnalités de MLflow incluent le suivi des expériences AI/ML et un référentiel de modèles AI/ML. Pour plus d'informations sur MLflow, visitez le "Site Web MLflow" .

Kubeflow

Kubeflow est une boîte à outils d'IA et de ML open source pour Kubernetes qui a été initialement développée par Google. Le projet Kubeflow rend les déploiements de workflows d'IA et de ML sur Kubernetes simples, portables et évolutifs. Kubeflow fait abstraction des subtilités de Kubernetes, permettant aux scientifiques des données de se concentrer sur ce qu'ils connaissent le mieux : la science des données. Voir la figure suivante pour une visualisation. Kubeflow est une bonne option open source pour les organisations qui préfèrent une plateforme MLOps tout-en-un. Pour plus d'informations, visitez le "Site Web de Kubeflow" .

Pipelines Kubeflow

Les pipelines Kubeflow sont un composant clé de Kubeflow. Kubeflow Pipelines est une plate-forme et une norme permettant de définir et de déployer des workflows d'IA et de ML portables et évolutifs. Pour plus d'informations, consultez le "documentation officielle de Kubeflow" .

Bloc-notes Kubeflow

Kubeflow simplifie le provisionnement et le déploiement des serveurs Jupyter Notebook sur Kubernetes. Pour plus d'informations sur Jupyter Notebooks dans le contexte de Kubeflow, consultez le "documentation officielle de Kubeflow" .

Katib

Katib est un projet natif de Kubernetes pour l'apprentissage automatique automatisé (AutoML). Katib prend en charge le réglage des hyperparamètres, l'arrêt précoce et la recherche d'architecture neuronale (NAS). Katib est un projet indépendant des frameworks d'apprentissage automatique (ML). Il peut ajuster les hyperparamètres des applications écrites dans n'importe quel langage choisi par les utilisateurs et prend en charge nativement de nombreux frameworks ML, tels que TensorFlow, MXNet, PyTorch, XGBoost et autres. Katib prend en charge de nombreux algorithmes AutoML différents, tels que l'optimisation bayésienne, les estimateurs d'arbre de Parzen, la recherche aléatoire, la stratégie d'évolution d'adaptation de matrice de covariance, l'hyperbande, la recherche d'architecture neuronale efficace, la recherche d'architecture différentiable et bien d'autres. Pour plus d'informations sur Jupyter Notebooks dans le contexte de Kubeflow, consultez le "documentation officielle de Kubeflow" .

NetApp ONTAP

ONTAP 9, la dernière génération de logiciel de gestion du stockage de NetApp, permet aux entreprises de moderniser leur infrastructure et de passer à un centre de données prêt pour le cloud. En s'appuyant sur des capacités de gestion de données de pointe, ONTAP permet la gestion et la protection des données avec un seul ensemble d'outils, quel que soit l'endroit où résident ces données. Vous pouvez également déplacer librement les données là où elles sont nécessaires : vers la périphérie, le cœur ou le cloud. ONTAP 9 inclut de nombreuses fonctionnalités qui simplifient la gestion des données, accélèrent et protègent les données critiques et permettent des capacités d'infrastructure de nouvelle génération dans les architectures de cloud hybride.

Simplifier la gestion des données

La gestion des données est essentielle pour les opérations informatiques de l'entreprise et les scientifiques des données afin que les ressources appropriées soient utilisées pour les applications d'IA et la formation des ensembles de données d'IA/ML. Les informations supplémentaires suivantes sur les technologies NetApp ne sont pas couvertes par cette validation, mais 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 votre coût total d'exploitation :

- Compactage des données en ligne et déduplication étendue. La compaction des données réduit l'espace gaspillé à l'intérieur des blocs de stockage et la déduplication augmente considérablement la capacité effective. Cela s'applique aux données stockées localement et aux données hiérarchisées vers le cloud.
- Qualité de service minimale, maximale et adaptative (AQoS). Les contrôles granulaires de qualité de service (QoS) aident à maintenir les niveaux de performances des applications critiques dans les environnements hautement partagés.
- FabricPool NetApp . Fournit une hiérarchisation automatique des données froides vers des options de stockage cloud publiques et privées, notamment Amazon Web Services (AWS), Azure et la solution de stockage NetApp StorageGRID . Pour plus d'informations sur FabricPool, voir "TR-4598 : Bonnes pratiques FabricPool" .

Accélérer et protéger les données

ONTAP offre des niveaux supérieurs de performance et de protection des données et étend ces capacités des manières suivantes :

- Performances et latence réduite. ONTAP offre le débit le plus élevé possible avec 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 plates-formes.
- Chiffrement de volume NetApp (NVE). ONTAP offre un cryptage natif au niveau du volume avec prise en charge de la gestion des clés intégrée et externe.
- Authentification multi-locataire et multifactorielle. ONTAP permet le partage des ressources d'infrastructure avec les plus hauts niveaux de sécurité.

Une infrastructure à l'épreuve du temps

ONTAP permet de répondre aux besoins commerciaux exigeants et en constante évolution grâce aux fonctionnalités suivantes :

- Mise à l'échelle transparente et opérations non perturbatrices. ONTAP prend en charge l'ajout non

perturbateur de capacité aux contrôleurs existants et aux clusters évolutifs. Les clients peuvent passer aux dernières technologies sans migrations de données ni pannes coûteuses.

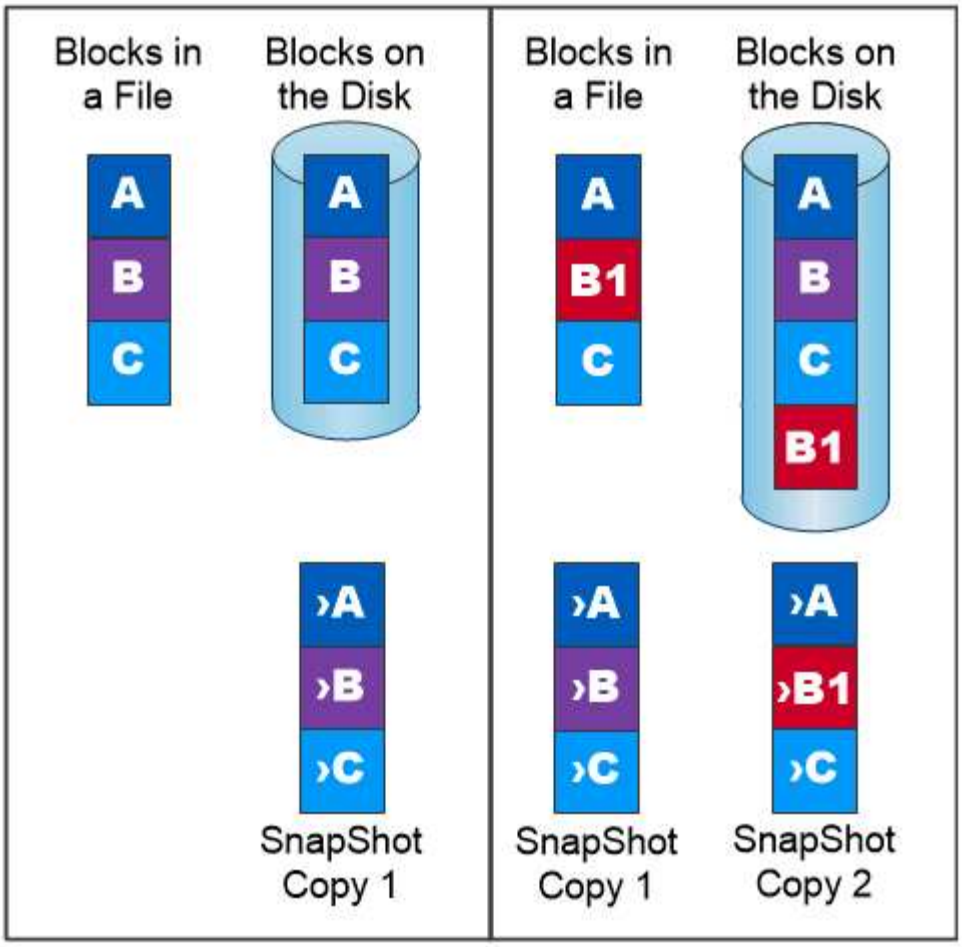
- Connexion au Cloud. ONTAP est le logiciel de gestion de stockage le plus connecté au cloud, avec des options de stockage défini par logiciel et des instances cloud natives dans tous les clouds publics.
- Intégration avec les applications émergentes. ONTAP propose des services de données de niveau entreprise pour les plates-formes et applications de nouvelle génération, telles que les véhicules autonomes, les villes intelligentes et l'industrie 4.0, en utilisant la même infrastructure qui prend en charge les applications d'entreprise existantes.

Copies instantanées NetApp

Une copie NetApp Snapshot est une image en lecture seule, à un instant T, d'un volume. L'image consomme un espace de stockage minimal et entraîne une surcharge de performances négligeable, car elle enregistre uniquement les modifications apportées aux fichiers créés depuis la dernière copie instantanée, comme illustré dans la figure suivante.

Les copies instantanées doivent leur efficacité à la technologie de virtualisation du stockage ONTAP de base, le Write Anywhere File Layout (WAFL). Comme une base de données, WAFL utilise des métadonnées pour pointer vers des blocs de données réels sur le disque. Mais, contrairement à une base de données, WAFL n'écrase pas les blocs existants. Il écrit les données mises à jour dans un nouveau bloc et modifie les métadonnées. C'est parce ONTAP référence les métadonnées lorsqu'il crée une copie Snapshot, plutôt que de copier des blocs de données, que les copies Snapshot sont si efficaces. Cela élimine le temps de recherche que les autres systèmes doivent accomplir pour localiser les blocs à copier, ainsi que le coût de réalisation de la copie elle-même.

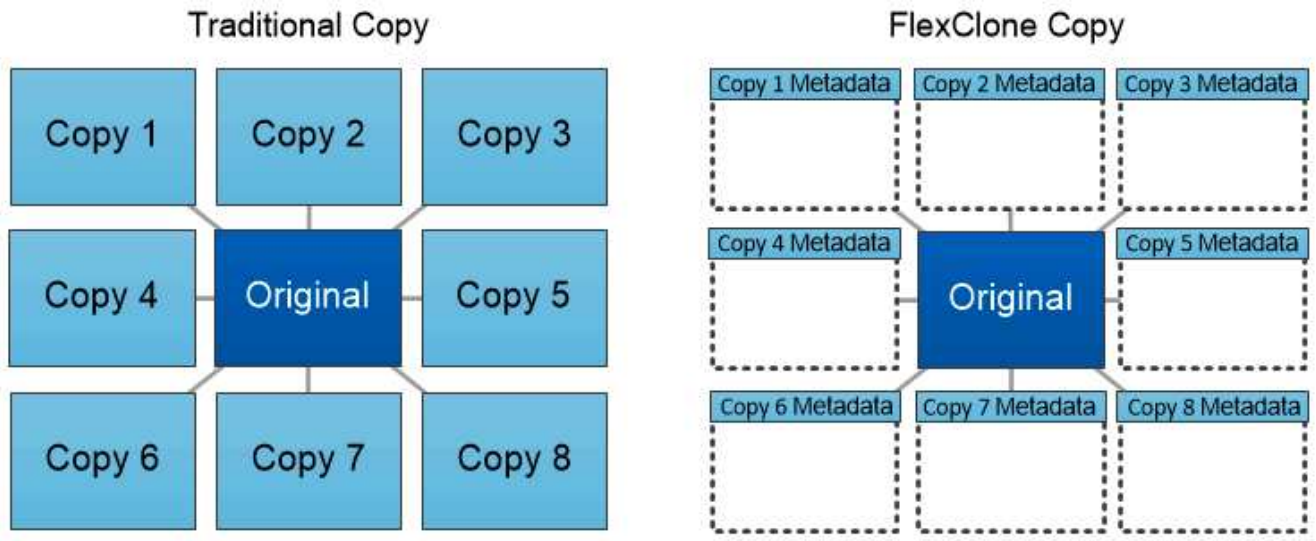
Vous pouvez utiliser une copie instantanée pour récupérer des fichiers individuels ou des LUN ou pour restaurer l'intégralité du contenu d'un volume. ONTAP compare les informations du pointeur dans la copie Snapshot avec les données sur le disque pour reconstruire l'objet manquant ou endommagé, sans temps d'arrêt ni coût de performance significatif.



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

Technologie NetApp FlexClone

La technologie NetApp FlexClone fait référence aux métadonnées Snapshot pour créer des copies inscriptibles à un instant T d'un volume. Les copies partagent des blocs de données avec leurs parents, ne consommant aucun stockage, à l'exception de ce qui est nécessaire pour les métadonnées jusqu'à ce que des modifications soient écrites sur la copie, comme illustré dans la figure suivante. Alors que la création de copies traditionnelles peut prendre des minutes, voire des heures, le logiciel FlexClone vous permet de copier presque instantanément même les plus grands ensembles de données. Cela le rend idéal pour les situations dans lesquelles vous avez besoin de plusieurs copies d'ensembles de données identiques (un espace de travail de développement, par exemple) ou de copies temporaires d'un ensemble de données (test d'une application par rapport à un ensemble de données de production).



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

Technologie de réplication de données NetApp SnapMirror

Le logiciel NetApp SnapMirror est une solution de réplication unifiée économique et facile à utiliser sur l'ensemble de la structure de données. Il réplique les données à grande vitesse sur LAN ou WAN. Il vous offre une haute disponibilité des données et une réplication rapide des données pour les applications de tous types, y compris les applications critiques pour l'entreprise dans les environnements virtuels et traditionnels. Lorsque vous répliquez des données sur un ou plusieurs systèmes de stockage NetApp et mettez à jour en permanence les données secondaires, vos données sont maintenues à jour et disponibles à tout moment. Aucun serveur de réplication externe n'est requis. Consultez la figure suivante pour un exemple d'architecture qui exploite la technologie SnapMirror .

Le logiciel SnapMirror exploite l'efficacité du stockage NetApp ONTAP en envoyant uniquement les blocs modifiés sur le réseau. Le logiciel SnapMirror utilise également la compression réseau intégrée pour accélérer les transferts de données et réduire l'utilisation de la bande passante du réseau jusqu'à 70 %. Avec la technologie SnapMirror , vous pouvez exploiter un flux de données de réplication mince pour créer un référentiel unique qui conserve à la fois le miroir actif et les copies ponctuelles antérieures, réduisant ainsi le trafic réseau jusqu'à 50 %.

Copie et synchronisation NetApp BlueXP

"Copie et synchronisation BlueXP"est un service NetApp pour une synchronisation rapide et sécurisée des données. Que vous ayez besoin de transférer des fichiers entre des partages de fichiers NFS ou SMB sur site, NetApp StorageGRID, NetApp ONTAP S3, Google Cloud NetApp Volumes, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage ou IBM Cloud Object Storage, BlueXP Copy and Sync déplace les fichiers là où vous en avez besoin rapidement et en toute sécurité.

Une fois vos données transférées, elles sont entièrement disponibles pour une utilisation sur la source et la cible. BlueXP Copy and Sync peut synchroniser les données à la demande lorsqu'une mise à jour est déclenchée ou synchroniser les données en continu selon un calendrier prédéfini. Quoi qu'il en soit, BlueXP Copy and Sync ne déplace que les deltas, donc le temps et l'argent consacrés à la réplication des données sont minimisés.

BlueXP Copy and Sync est un outil logiciel en tant que service (SaaS) extrêmement simple à configurer et à

utiliser. Les transferts de données déclenchés par BlueXP Copy and Sync sont effectués par des 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

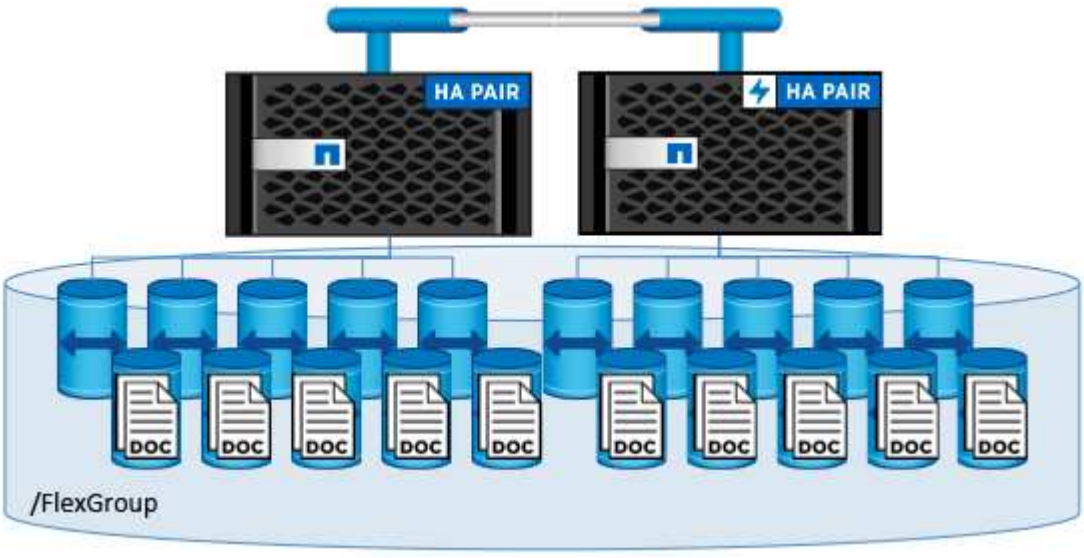
"NetApp XCP"est un logiciel basé sur le client pour les migrations de données vers NetApp et NetApp vers NetApp et les informations sur les systèmes de fichiers. XCP est conçu pour évoluer et atteindre des performances maximales en utilisant toutes les ressources système disponibles pour gérer des ensembles de données à volume élevé et des migrations hautes performances. XCP vous aide à obtenir une visibilité complète sur le système de fichiers avec la possibilité de générer des rapports.

Volumes NetApp ONTAP FlexGroup

Un ensemble de données de formation peut être une collection de milliards de fichiers potentiellement. 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 espace de noms unique qui comprend plusieurs volumes membres constitutifs, comme illustré dans la figure suivante. Du point de vue d'un administrateur de stockage, un volume FlexGroup est géré et agit comme un FlexVol volume NetApp FlexVol. Les fichiers d'un volume FlexGroup sont alloués à des volumes membres individuels et ne sont pas répartis sur des volumes ou des nœuds. Ils permettent les capacités suivantes :

- Les volumes FlexGroup offrent plusieurs pétaoctets de capacité et une faible latence prévisible pour les charges de travail à métadonnées élevées.
- Ils prennent en charge jusqu'à 400 milliards de fichiers dans le même espace de noms.
- 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 d'un matériel spécifique. La solution est compatible avec tout dispositif de stockage physique NetApp , instance définie par logiciel ou service cloud pris en charge par NetApp Trident. Les exemples incluent un système de stockage NetApp AFF , Amazon FSx ONTAP, Azure NetApp Files, Google Cloud NetApp Volumes ou une instance NetApp Cloud Volumes ONTAP . De plus, la solution peut être implémentée sur n'importe quel cluster Kubernetes à condition que la version Kubernetes utilisée soit prise en charge par NetApp Trident et les autres composants de la solution en cours d'implémentation. Pour obtenir la liste des versions de Kubernetes prises en charge par Trident, consultez le "Documentation Trident" . Consultez les tableaux suivants pour plus de détails sur les environnements qui ont été utilisés pour valider les différents composants de cette solution.

Environnement de validation Apache Airflow

Composant logiciel	Version
Apache Airflow	2.0.1, déployé via"Diagramme de flux d'air Apache Helm" 8.0.8
Kubernetes	1,18
NetApp Trident	21,01

Environnement de validation JupyterHub

Composant logiciel	Version
JupyterHub	4.1.5, déployé via"Graphique Helm de JupyterHub" 3.3.7
Kubernetes	1,29
NetApp Trident	24,02

Environnement de validation MLflow

Composant logiciel	Version
MLflow	2.14.1, déployé via"Diagramme Helm MLflow" 1.4.12
Kubernetes	1,29
NetApp Trident	24,02

Environnement de validation Kubeflow

Composant logiciel	Version
Kubeflow	1.7, déployé via"déployerKF" 0.1.1
Kubernetes	1,26

Composant logiciel	Version
NetApp Trident	23,07

Support

NetApp n'offre pas de support d'entreprise pour Apache Airflow, JupyterHub, MLflow, Kubeflow ou Kubernetes. Si vous êtes intéressé par une plateforme MLOps entièrement prise en charge,"contacter NetApp" à propos des solutions MLOps entièrement prises en charge que NetApp propose conjointement avec des partenaires.

Configuration de NetApp Trident

Exemples de backends Trident pour les déploiements NetApp AIPod

Avant de pouvoir utiliser Trident pour provisionner dynamiquement des ressources de stockage au sein de votre cluster Kubernetes, vous devez créer un ou plusieurs backends Trident . Les exemples qui suivent représentent différents types de backends que vous souhaiterez peut-être créer si vous déployez des composants de cette solution sur un"NetApp AIPod" . Pour plus d'informations sur les backends, et par exemple sur les backends pour d'autres plateformes/environnements, consultez le"Documentation Trident" .

1. NetApp recommande de créer un backend Trident compatible FlexGroup pour votre AIPod.

   Les exemples de commandes qui suivent montrent la création d'un backend Trident compatible FlexGroup pour une machine virtuelle de stockage AIPod (SVM). Ce backend utilise le `ontap-nas-flexgroup` pilote de stockage. ONTAP prend en charge deux principaux types de volumes de données : FlexVol et FlexGroup. Les volumes FlexVol sont limités en taille (au moment de la rédaction de cet article, la taille maximale dépend du déploiement spécifique). Les volumes FlexGroup , en revanche, peuvent évoluer de manière linéaire jusqu'à 20 Po et 400 milliards de fichiers, fournissant un espace de noms unique qui simplifie considérablement la gestion des données. Par conséquent, les volumes FlexGroup sont optimaux pour les charges de travail d'IA et de ML qui reposent sur de grandes quantités de données.

   Si vous travaillez avec une petite quantité de données et que vous souhaitez utiliser des volumes FlexVol au lieu de volumes FlexGroup , vous pouvez créer des backends 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 backend Trident compatible FlexVol . Vous souhaiterez peut-être utiliser des volumes FlexVol pour héberger des applications persistantes, stocker des résultats, des sorties, des informations de débogage, etc. Si vous souhaitez utiliser des volumes FlexVol , vous devez créer un ou plusieurs backends Trident compatibles FlexVol . Les exemples de commandes qui suivent montrent la création d'un seul backend Trident compatible 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 |
+------------------------+--------------------
+--------------------------------------+--------+---------+
```

Exemples de classes de stockage Kubernetes pour les déploiements NetApp AIPod

Avant de pouvoir utiliser Trident pour provisionner dynamiquement des ressources de stockage au sein de votre cluster Kubernetes, vous devez créer une ou plusieurs StorageClasses Kubernetes. Les exemples qui suivent représentent différents types de StorageClasses que vous souhaiterez peut-être créer si vous déployez des composants

de cette solution sur un"NetApp AIPod" . Pour plus d'informations sur les StorageClasses, et par exemple sur les StorageClasses pour d'autres plates-formes/environnements, consultez le"Documentation Trident" .

1. NetApp recommande de créer une StorageClass pour le backend Trident compatible FlexGroup que vous avez créé dans la section"Exemples de backends Trident pour les déploiements NetApp AIPod" , étape 1. Les exemples de commandes qui suivent montrent la création de plusieurs StorageClasses qui correspondent à l'exemple de Backend qui a été créé dans la section"Exemples de backends Trident pour les déploiements NetApp AIPod" , étape 1 - une étape qui utilise"NFS sur RDMA" et un qui ne le fait pas.

   Pour qu'un volume persistant ne soit pas supprimé lorsque le PersistentVolumeClaim (PVC) correspondant est supprimé, l'exemple suivant utilise un `reclaimPolicy` valeur de `Retain` . Pour plus d'informations sur le `reclaimPolicy` terrain, voir le site officiel "Documentation Kubernetes" .

   Remarque : les exemples de StorageClasses suivants utilisent une taille de transfert maximale de 262 144. Pour utiliser cette taille de transfert maximale, vous devez configurer la taille de transfert maximale sur votre système ONTAP en conséquence. Se référer à la"Documentation ONTAP" pour plus de détails.

   Remarque : pour utiliser NFS sur RDMA, vous devez configurer NFS sur RDMA sur votre système ONTAP . Se référer à la"Documentation ONTAP" pour plus de détails.

   Remarque : dans l'exemple suivant, un backend spécifique est 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 StorageClass qui correspond au backend Trident compatible FlexVol que vous avez créé dans la section"Exemples de backends Trident pour les déploiements AIPod" , étape 2. Les exemples de commandes qui suivent montrent la création d'une seule StorageClass pour les volumes FlexVol .

   Remarque : dans l'exemple suivant, aucun backend particulier n'est spécifié dans le champ storagePool du fichier de définition StorageClass. Lorsque vous utilisez Kubernetes pour administrer des volumes à l'aide de cette StorageClass, Trident tente d'utiliser tout backend disponible qui utilise le `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
```

Apache Airflow

Déploiement d'Apache Airflow

Cette section décrit les tâches que vous devez effectuer pour déployer Airflow dans votre cluster Kubernetes.

Il est possible de déployer Airflow sur d'autres plateformes que Kubernetes. Le déploiement d'Airflow sur des plateformes autres que Kubernetes n'entre pas dans le 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 Trident dans votre cluster Kubernetes. Pour plus de détails sur Trident, reportez-vous au"Documentation Trident" .

Installer Helm

Airflow est déployé à l'aide de Helm, un gestionnaire de packages populaire pour Kubernetes. Avant de déployer Airflow, vous devez installer Helm sur l'hôte de saut de déploiement. Pour installer Helm sur l'hôte de saut de déploiement, suivez les instructions "instructions d'installation" dans la documentation officielle de Helm.

Définir la classe de stockage Kubernetes par défaut

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

Utilisez Helm pour déployer Airflow

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

1. Déployez Airflow à l'aide de Helm en suivant les "instructions de déploiement" pour le tableau officiel du flux d'air sur l'Artifact Hub. Les exemples de commandes qui suivent montrent le déploiement d'Airflow à l'aide de Helm. Modifier, ajouter et/ou supprimer des valeurs dans le `custom- values.yaml` fichier selon vos besoins en fonction de votre environnement et de la configuration souhaitée.

```
$ 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
```

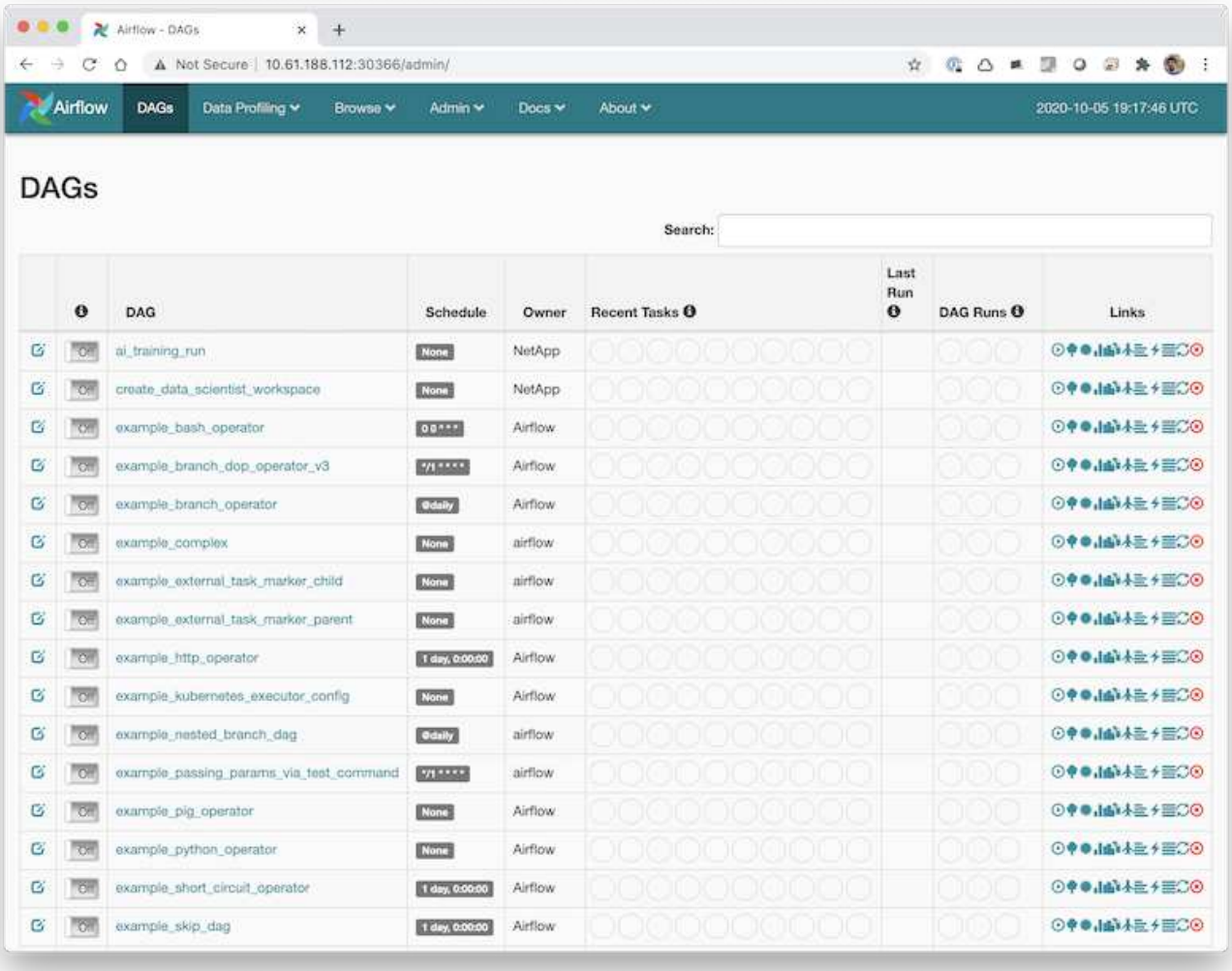
2. Vérifiez que tous les modules Airflow sont opérationnels. Le démarrage de tous les modules peut prendre quelques minutes.

```
$ 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. Obtenez l'URL du service Web Airflow en suivant les instructions imprimées sur la console lorsque vous avez déployé Airflow 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 Airflow.



Utilisez la boîte à outils NetApp DataOps avec Airflow

Le "Boîte à outils NetApp DataOps pour Kubernetes" peut être utilisé en conjonction avec Airflow. L'utilisation de NetApp DataOps Toolkit avec Airflow vous permet d'intégrer des opérations de gestion de données NetApp , telles que la création d'instantanés et de clones, dans des flux de travail automatisés orchestrés par Airflow.

Se référer à la "Exemples de flux d'air" section dans le référentiel GitHub NetApp DataOps Toolkit pour plus de détails sur l'utilisation de la boîte à outils avec Airflow.

JupyterHub

Déploiement de JupyterHub

Cette section décrit les tâches que vous devez effectuer pour déployer JupyterHub dans votre cluster Kubernetes.

Il est possible de déployer JupyterHub sur d'autres plateformes que Kubernetes. Le déploiement de JupyterHub sur des plateformes autres que Kubernetes n'entre pas dans le 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 Trident dans votre cluster Kubernetes. Pour plus de détails sur Trident, reportez-vous au"Documentation Trident" .

Installer Helm

JupyterHub est déployé à l'aide de Helm, un gestionnaire de packages populaire pour Kubernetes. Avant de déployer JupyterHub, vous devez installer Helm sur votre nœud de contrôle Kubernetes. Pour installer Helm, suivez les instructions "instructions d'installation" dans la documentation officielle de Helm.

Définir la classe de stockage Kubernetes par défaut

Avant de déployer JupyterHub, vous devez désigner une StorageClass par défaut dans votre cluster Kubernetes. Pour désigner une StorageClass par défaut au sein de votre cluster, suivez les instructions décrites dans le"Déploiement de Kubeflow" section. Si vous avez déjà désigné une StorageClass par défaut dans votre cluster, vous pouvez ignorer cette étape.

Déployer JupyterHub

Après avoir terminé les étapes ci-dessus, vous êtes maintenant prêt à déployer JupyterHub. Le déploiement de JupyterHub nécessite les étapes suivantes :

Configurer le déploiement de JupyterHub

Avant le déploiement, il est recommandé d'optimiser le déploiement de JupyterHub pour votre environnement respectif. Vous pouvez créer un fichier **config.yaml** et l'utiliser lors du déploiement à l'aide du graphique Helm.

Un exemple de fichier **config.yaml** peut être trouvé à l'adresse https://github.com/jupyterhub/zero-to-jupyterhub-k8s/blob/HEAD/jupyterhub/values.yaml

Dans ce fichier config.yaml, vous pouvez définir le paramètre **(singleuser.storage.dynamic.storageClass)** pour NetApp Trident StorageClass. Il s'agit de la classe de stockage qui sera utilisée pour provisionner les volumes pour les espaces de travail des utilisateurs individuels.

Ajout de volumes partagés

Si vous souhaitez utiliser un volume partagé pour tous les utilisateurs JupyterHub, vous pouvez ajuster votre **config.yaml** en conséquence. Par exemple, si vous avez un PersistentVolumeClaim partagé appelé jupyterhub-shared-volume, vous pouvez le monter en tant que /home/shared dans tous les pods utilisateur comme suit :

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

Il s'agit d'une étape facultative, vous pouvez ajuster ces paramètres selon vos besoins.

Déployer JupyterHub avec Helm Chart

Informez Helm du référentiel de graphiques JupyterHub Helm.

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

Cela devrait afficher une sortie comme :

```
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!□
```

Installez maintenant le graphique configuré par votre config.yaml en exécutant cette commande à partir du répertoire qui contient votre config.yaml :

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

Dans cet exemple :

<helm-release-name> est défini sur my-jupyterhub, qui sera le nom de votre version JupyterHub. <k8s-namespace> est défini sur my-namespace, qui est l'espace de noms dans lequel vous souhaitez installer JupyterHub. L'indicateur --create-namespace est utilisé pour créer l'espace de noms s'il n'existe pas déjà. L'indicateur --values spécifie le fichier config.yaml qui contient les options de configuration souhaitées.

Vérifier le déploiement

Pendant que l'étape 2 est en cours d'exécution, vous pouvez voir les pods créés à partir de la commande suivante :

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

Attendez que le hub et le pod proxy entrent dans l'état d'exécution.

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

Accéder à JupyterHub

Trouvez l'IP que nous pouvons utiliser pour accéder au JupyterHub. Exécutez la commande suivante jusqu'à ce que l'adresse IP EXTERNE du service proxy-public soit disponible comme dans l'exemple de sortie.

> Nous avons utilisé le service NodePort dans notre fichier config.yaml, vous pouvez l'ajuster en fonction de votre environnement en fonction de votre configuration (par exemple 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
```

Pour utiliser JupyterHub, saisissez l'adresse IP externe du service proxy-public dans un navigateur.

Utilisez la boîte à outils NetApp DataOps avec JupyterHub

Le "Boîte à outils NetApp DataOps pour Kubernetes" peut être utilisé en conjonction avec JupyterHub. L'utilisation de NetApp DataOps Toolkit avec JupyterHub permet aux utilisateurs finaux de créer des instantanés de volume pour la sauvegarde de l'espace de travail et/ou la traçabilité de l'ensemble de données vers le modèle directement à partir d'un bloc-notes Jupyter.

Configuration initiale

Avant de pouvoir utiliser DataOps Toolkit avec JupyterHub, vous devez accorder les autorisations appropriées au compte de service Kubernetes que JupyterHub attribue aux pods Jupyter Notebook Server des utilisateurs individuels. JupyterHub utilise le compte de service spécifié par le `singleuser.serviceAccountName` variable dans votre fichier de configuration de graphique JupyterHub Helm.

Créer un rôle de cluster pour la boîte à outils DataOps

Tout d'abord, créez un rôle de cluster nommé « netapp-dataops » qui dispose des autorisations d'API Kubernetes requises pour créer des instantanés de 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
```

Attribuer un rôle de cluster au compte de service du serveur Notebook

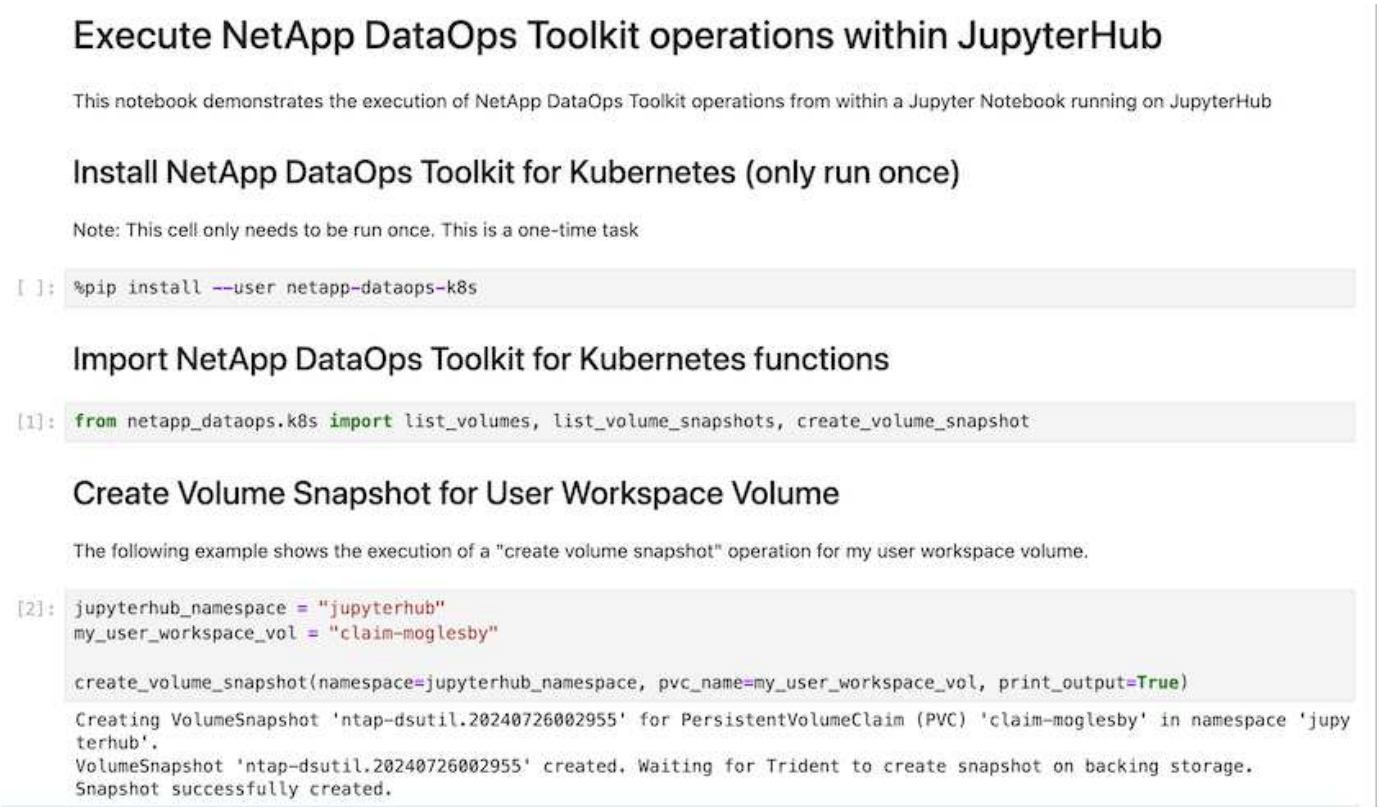
Créez une liaison de rôle qui attribue le rôle de cluster « netapp-dataops-snapshots » au compte de service approprié dans l'espace de noms approprié. Par exemple, si vous avez installé JupyterHub dans l'espace de noms « jupyterhub » et que vous avez spécifié le compte de service « par défaut » via le `singleuser.serviceAccountName` variable, vous attribueriez le rôle de cluster « netapp-dataops-snapshots » au compte de service « default » dans l'espace de noms « jupyterhub », comme indiqué dans l'exemple suivant.

```
$ 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
```

Créer des instantanés de volume dans Jupyter Notebook

Désormais, les utilisateurs de JupyterHub peuvent utiliser NetApp DataOps Toolkit pour créer des instantanés de volume directement à partir d'un Jupyter Notebook, comme illustré dans l'exemple suivant.



Ingérer des données dans JupyterHub avec NetApp SnapMirror

NetApp SnapMirror est une technologie de réplication qui vous permet de répliquer des données entre des systèmes de stockage NetApp . SnapMirror peut être utilisé pour ingérer des données provenant d'environnements distants dans JupyterHub.

Exemple de flux de travail et démonstration

Se référer à"cet article de blog Tech ONTAP" pour un exemple détaillé de flux de travail et une démonstration de l'utilisation de NetApp SnapMirror pour ingérer des données dans JupyterHub.

MLflow

Déploiement de MLflow

Cette section décrit les tâches que vous devez effectuer pour déployer MLflow dans votre cluster Kubernetes.

> Il est possible de déployer MLflow sur d'autres plateformes que Kubernetes. Le déploiement de MLflow sur des plateformes autres que Kubernetes n'entre pas dans le 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 Trident dans votre cluster Kubernetes. Pour plus de détails sur Trident, reportez-vous au"Documentation Trident" .

Installer Helm

MLflow est déployé à l'aide de Helm, un gestionnaire de packages populaire pour Kubernetes. Avant de déployer MLflow, vous devez installer Helm sur votre nœud de contrôle Kubernetes. Pour installer Helm, suivez les instructions "instructions d'installation" dans la documentation officielle de Helm.

Définir la classe de stockage Kubernetes par défaut

Avant de déployer MLflow, vous devez désigner une StorageClass par défaut dans votre cluster Kubernetes. Pour désigner une StorageClass par défaut au sein de votre cluster, suivez les instructions décrites dans le"Déploiement de Kubeflow" section. Si vous avez déjà désigné une StorageClass par défaut au sein de votre cluster, vous pouvez ignorer cette étape.

Déployer MLflow

Une fois les prérequis remplis, vous pouvez commencer le déploiement de MLflow à l'aide du graphique Helm.

Configurer le déploiement du graphique Helm MLflow.

Avant de déployer MLflow à l'aide du graphique Helm, nous pouvons configurer le déploiement pour utiliser la classe de stockage NetApp Trident et modifier d'autres paramètres en fonction de nos besoins à l'aide d'un

fichier **config.yaml**. Un exemple de fichier **config.yaml** peut être trouvé à l'adresse : https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml

> Vous pouvez définir la classe de stockage Trident sous le paramètre **global.defaultStorageClass** dans le fichier config.yaml (par exemple storageClass : « ontap-flexvol »).

Installation du Helm Chart

Le graphique Helm peut être installé avec le fichier **config.yaml** personnalisé pour MLflow à l'aide de la commande suivante :

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

> La commande déploie MLflow sur le cluster Kubernetes dans la configuration personnalisée via le fichier **config.yaml** fourni. MLflow est déployé dans l'espace de noms donné et un nom de version aléatoire est donné via Kubernetes pour la version.

Vérifier le déploiement

Une fois le déploiement du graphique Helm terminé, vous pouvez vérifier si le service est accessible en utilisant :

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

> Remplacez **jupyterhub** par l'espace de noms que vous avez utilisé lors du déploiement.

Vous devriez voir les services suivants :

```
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
```

> Nous avons modifié le fichier config.yaml pour utiliser le service NodePort pour accéder à MLflow sur le port 30002.

Accéder à MLflow

Une fois que tous les services liés à MLflow sont opérationnels, vous pouvez y accéder en utilisant l'adresse IP NodePort ou LoadBalancer donnée (par exemple `http://10.61.181.109:30002`)

Traçabilité des ensembles de données aux modèles avec NetApp et MLflow

Le "Boîte à outils NetApp DataOps pour Kubernetes" peut être utilisé en conjonction avec les capacités de suivi des expériences de MLflow afin de mettre en œuvre la traçabilité de l'ensemble de données au modèle ou de l'espace de travail au modèle.

Pour implémenter la traçabilité de l'ensemble de données au modèle ou de l'espace de travail au modèle, créez simplement un instantané de votre ensemble de données ou de votre volume d'espace de travail à l'aide de la boîte à outils DataOps dans le cadre de votre exécution de formation, comme illustré dans l'exemple de code suivant. Ce code enregistrera le nom du volume de données et le nom de l'instantané en tant que balises associées à l'exécution d'entraînement spécifique que vous enregistrez sur votre serveur de suivi d'expérience 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

Déploiement de Kubeflow

Cette section décrit les tâches que vous devez 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 fonctionnel et vous exécutez une version de Kubernetes prise en charge par la version de Kubeflow que vous envisagez de déployer. Pour obtenir la liste des versions de Kubernetes prises en charge, reportez-vous aux dépendances de votre version de Kubeflow dans le"documentation officielle de Kubeflow" .
2. Vous avez déjà installé et configuré NetApp Trident dans votre cluster Kubernetes. Pour plus de détails sur Trident, reportez-vous au"Documentation Trident" .

Définir la classe de stockage Kubernetes par défaut

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

1. Désignez l'une de vos StorageClasses existantes comme StorageClass par défaut. Les exemples de commandes qui suivent montrent la désignation d'une StorageClass nommée `ontap-ai-flexvols-retain` comme StorageClass par défaut.

> Le `ontap-nas-flexgroup` Le type de backend Trident a une taille de PVC minimale assez grande. Par défaut, Kubeflow tente de provisionner des PVC dont la taille ne dépasse pas quelques Go. Par conséquent, vous ne devez pas désigner une StorageClass qui utilise le `ontap-nas-flexgroup` Type de backend comme StorageClass par défaut pour les besoins du déploiement de 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
```

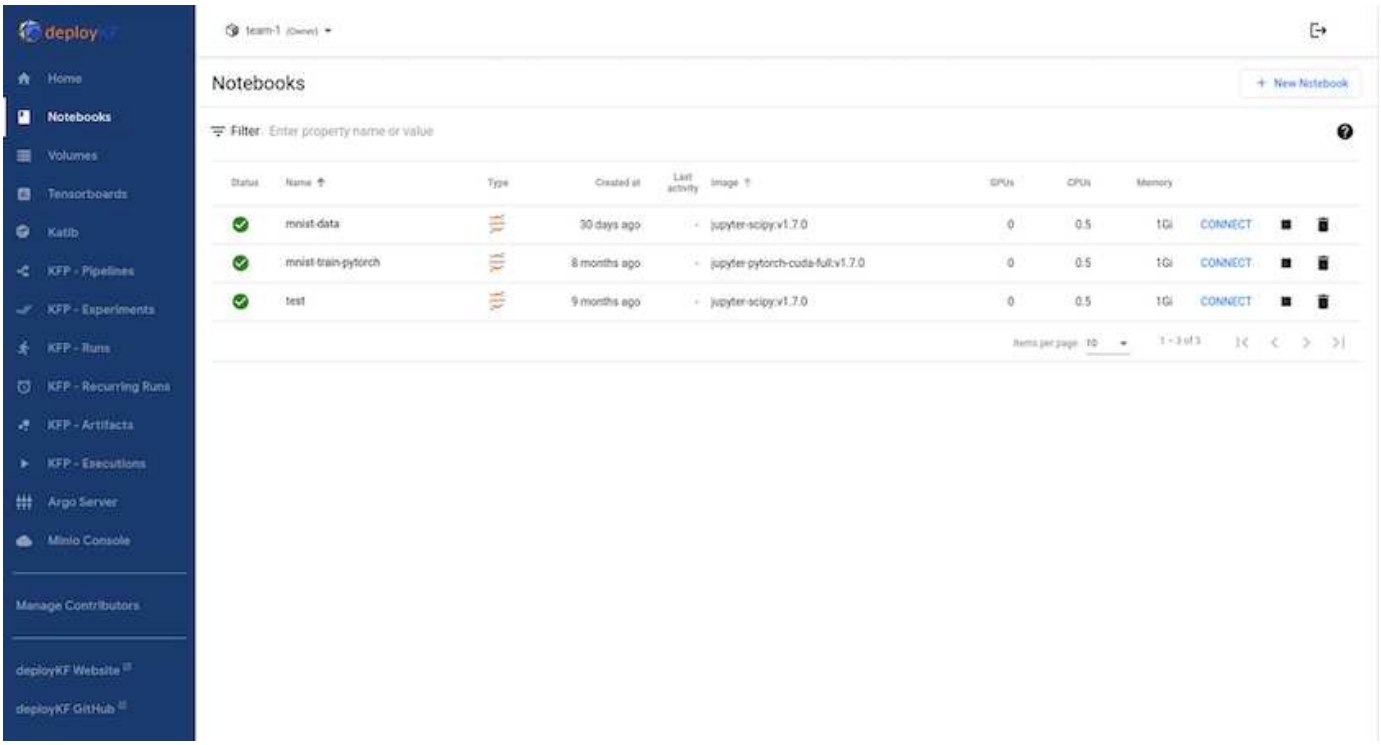
Options de déploiement de Kubeflow

Il existe de nombreuses options différentes pour déployer Kubeflow. Se référer à la"documentation officielle de Kubeflow" pour obtenir une liste des options de déploiement, et choisissez l'option qui correspond le mieux à vos besoins.

ⓘ À des fins de validation, nous avons déployé Kubeflow 1.7 en utilisant"déployerKF" 0.1.1.

Fournir un espace de travail Jupyter Notebook pour une utilisation par un data scientist ou un développeur

Kubeflow est capable de provisionner rapidement de nouveaux serveurs Jupyter Notebook pour agir comme espaces de travail de data scientist. Pour plus d'informations sur les notebooks Jupyter dans le contexte de Kubeflow, consultez le "documentation officielle de Kubeflow" .



Utilisez la boîte à outils NetApp DataOps avec Kubeflow

Le "Boîte à outils NetApp Data Science pour Kubernetes" peut être utilisé en conjonction avec Kubeflow. L'utilisation de NetApp Data Science Toolkit avec Kubeflow offre les avantages suivants :

- Les scientifiques des données peuvent effectuer des opérations avancées de gestion des données NetApp , telles que la création d'instantanés et de clones, directement à partir d'un bloc-notes Jupyter.
- Les opérations avancées de gestion des données NetApp , telles que la création d'instantanés et de clones, peuvent être intégrées dans des flux de travail automatisés à l'aide du framework Kubeflow Pipelines.

Se référer à la "Exemples de Kubeflow" section dans le référentiel GitHub NetApp Data Science Toolkit pour plus de détails sur l'utilisation de la boîte à outils avec Kubeflow.

Exemple de workflow : Entraîner un modèle de reconnaissance d'images à l'aide de Kubeflow et de la boîte à outils NetApp DataOps

Cette section décrit les étapes impliquées dans la formation et le déploiement d'un réseau neuronal pour la reconnaissance d'images à l'aide de Kubeflow et de NetApp DataOps Toolkit. Ceci est destiné à servir d'exemple pour montrer un travail de formation qui intègre le stockage NetApp .

Prérequis

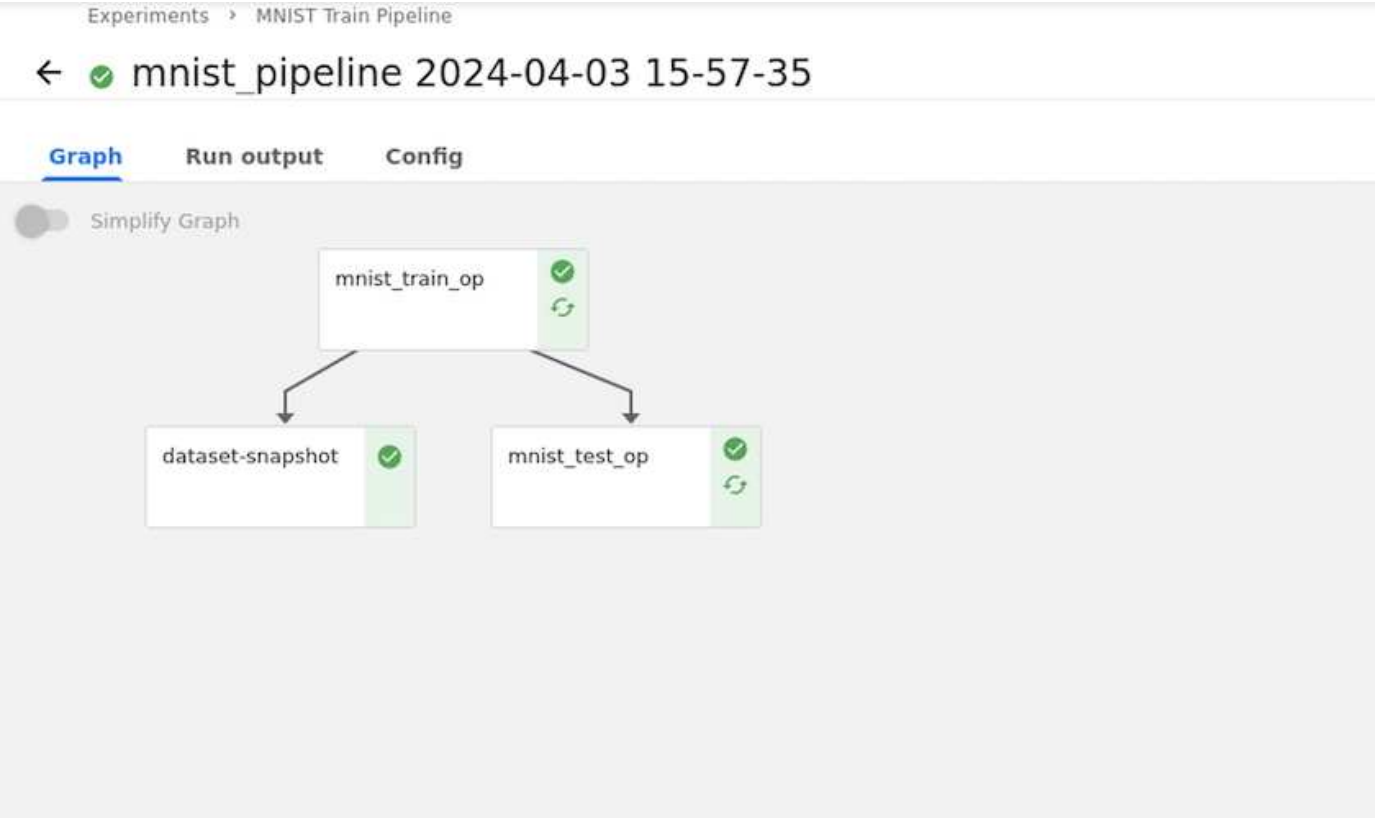
Créez un Dockerfile avec les configurations requises à utiliser pour les étapes de train et de test dans le pipeline Kubeflow. Voici un exemple de Dockerfile -

```
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 packages requis pour exécuter le programme. Avant de former le modèle d'apprentissage automatique, il est supposé que vous disposez déjà d'un déploiement Kubeflow fonctionnel.

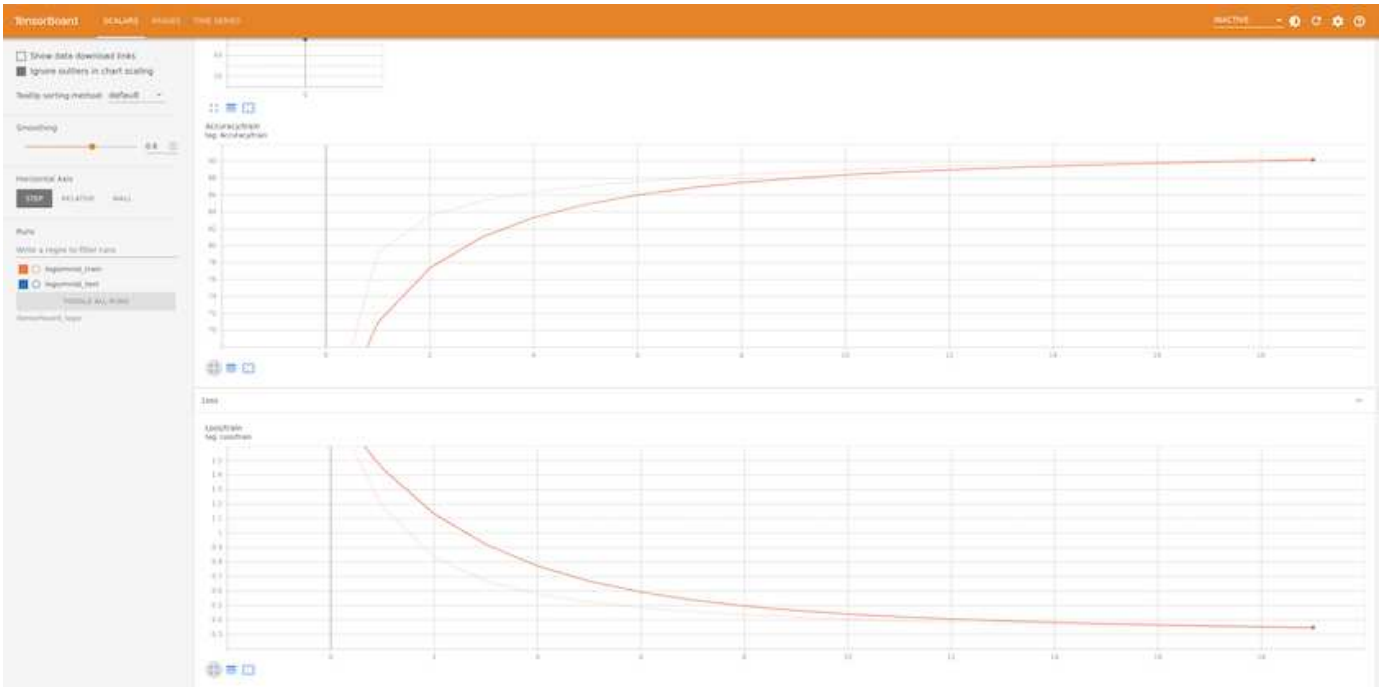
Entraîner un petit NN sur des données MNIST à l'aide de pipelines PyTorch et Kubeflow

Nous utilisons l'exemple d'un petit réseau neuronal formé sur des données MNIST. L'ensemble de données MNIST se compose d'images manuscrites de chiffres de 0 à 9. Les images ont une taille de 28x28 pixels. L'ensemble de données est divisé en 60 000 images de train et 10 000 images de validation. Le réseau neuronal utilisé pour cette expérience est un réseau à propagation directe à 2 couches. La formation est exécutée à l'aide de Kubeflow Pipelines. Se référer à la documentation "ici" pour plus d'informations. Notre pipeline Kubeflow intègre l'image Docker de la section Prérequis.



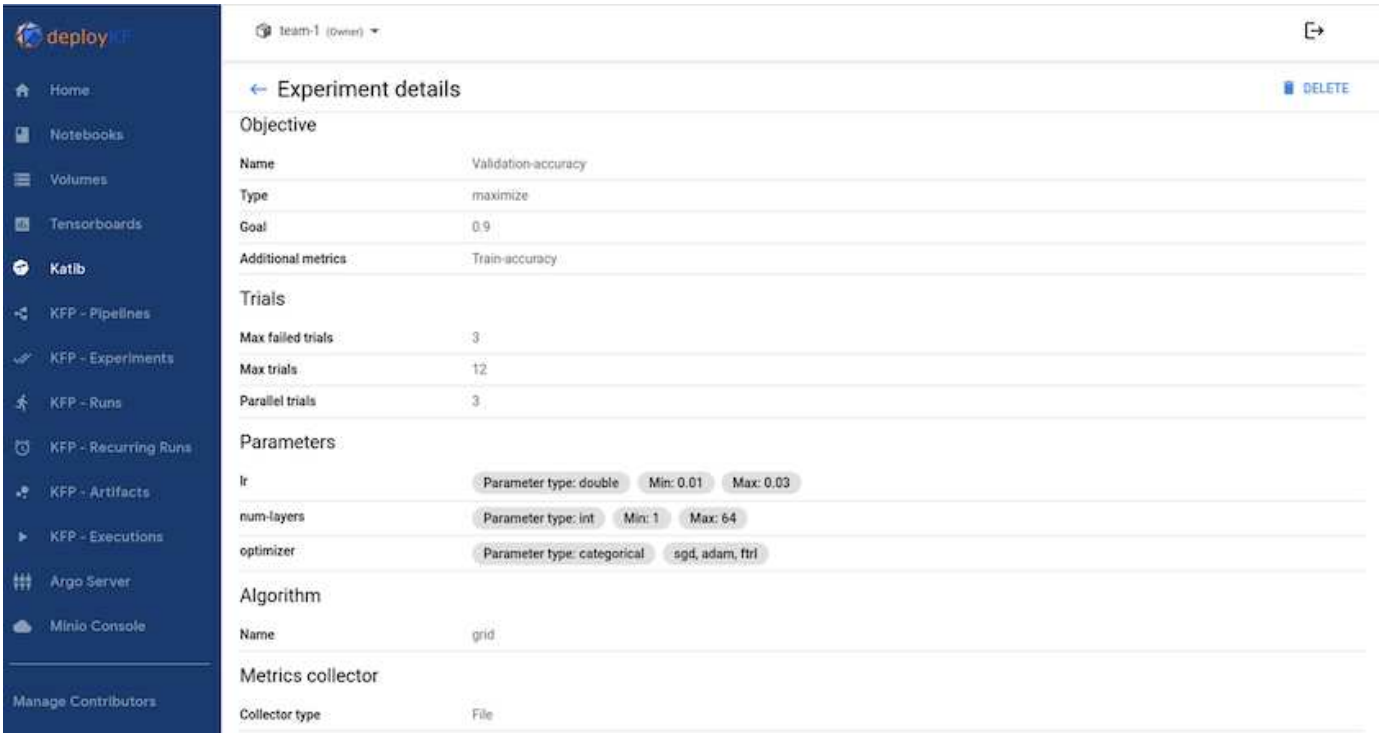
Visualiser les résultats à l'aide de Tensorboard

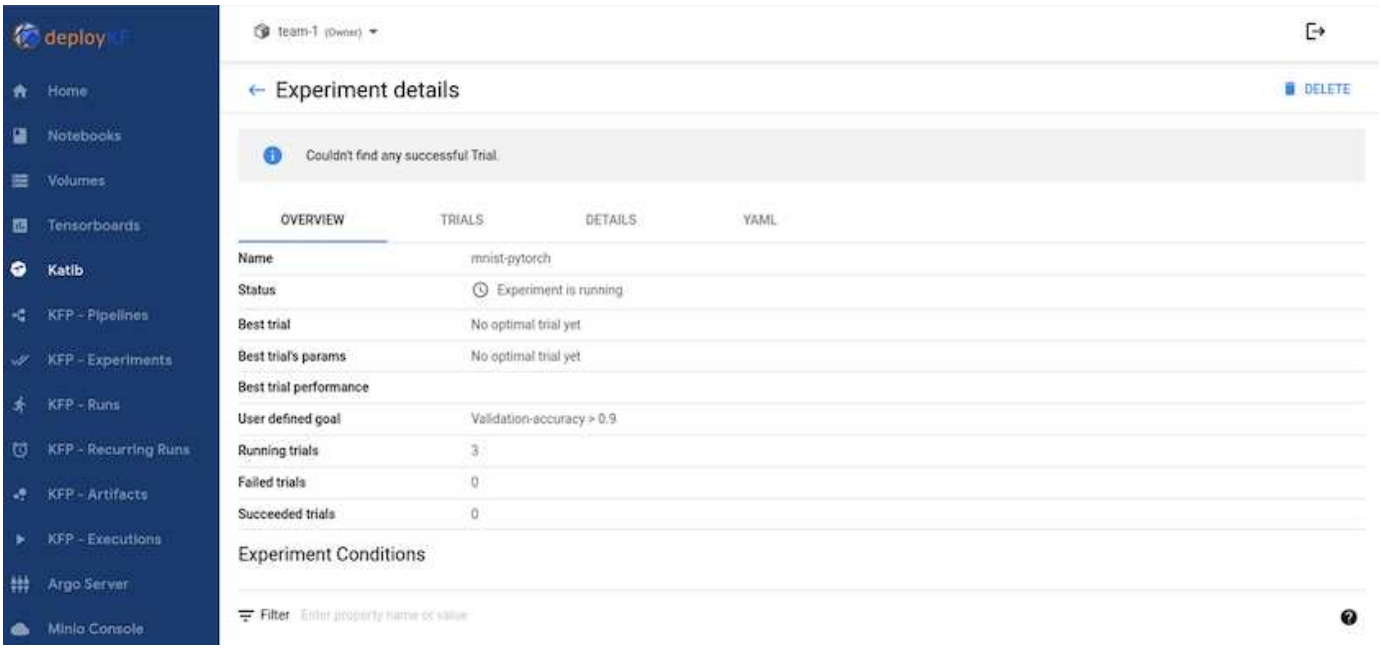
Une fois le modèle formé, nous pouvons visualiser les résultats à l'aide de Tensorboard. "Panneau Tensorboard" est disponible en tant que fonctionnalité sur le tableau de bord Kubeflow. Vous pouvez créer un tensorboard personnalisé pour votre travail. Un exemple ci-dessous montre le graphique de la précision de l'entraînement par rapport au nombre d'époques et de la perte d'entraînement par rapport au nombre d'époques.



Expérimenter avec des hyperparamètres à l'aide de Katib

"Katib"est un outil au sein de Kubeflow qui peut être utilisé pour expérimenter les hyperparamètres du modèle. Pour créer une expérience, définissez d'abord une métrique/un objectif souhaité. Il s'agit généralement de la précision du test. Une fois la métrique définie, choisissez les hyperparamètres avec lesquels vous souhaitez jouer (optimiseur/taux d'apprentissage/nombre de couches). Katib effectue un balayage d'hyperparamètres avec les valeurs définies par l'utilisateur pour trouver la meilleure combinaison de paramètres qui satisfont la métrique souhaitée. Vous pouvez définir ces paramètres dans chaque section de l'interface utilisateur. Alternativement, vous pouvez définir un fichier **YAML** avec les spécifications nécessaires. Ci-dessous, une illustration d'une expérience Katib -





Utilisez les instantanés NetApp pour enregistrer les données à des fins de traçabilité

Pendant la formation du modèle, nous souhaiterons peut-être enregistrer un instantané de l'ensemble de données de formation à des fins de traçabilité. Pour ce faire, nous pouvons ajouter une étape d'instantané au pipeline comme indiqué ci-dessous. Pour créer l'instantané, nous pouvons utiliser le "Boîte à outils 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)
```

Se référer à la "Exemple de boîte à outils NetApp DataOps pour Kubeflow" pour plus d'informations.

Exemple d'opérations Trident

Cette section comprend des exemples de diverses opérations que vous souhaiterez peut-être effectuer avec Trident.

Importer un volume existant

S'il existe des volumes existants sur votre système/plateforme de stockage NetApp que vous souhaitez monter sur des conteneurs au sein de votre cluster Kubernetes, mais qui ne sont pas liés aux PVC du cluster, vous devez importer ces volumes. Vous pouvez utiliser la fonctionnalité d'importation de volume 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 PVC, consultez le "documentation officielle de Kubernetes" . Pour plus d'informations sur la fonctionnalité d'importation de volume, consultez le "Documentation Trident" .

Un `accessModes` valeur de `ReadOnlyMany` est spécifié dans les exemples de fichiers de spécifications PVC. Pour plus d'informations sur le `accessMode` champ, voir le "documentation officielle de 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 plate-forme de stockage NetApp .

Provisionner un nouveau volume à l'aide de kubectl

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

Un `accessModes` valeur de `ReadWriteMany` est spécifié dans l'exemple de fichier de définition PVC suivant. Pour plus d'informations sur le `accessMode` champ, voir le "documentation officielle de 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
```

Provisionner un nouveau volume à l'aide de la boîte à outils NetApp DataOps

Vous pouvez également utiliser NetApp DataOps Toolkit pour Kubernetes pour provisionner un nouveau volume sur votre système ou plateforme de stockage NetApp . La boîte à outils NetApp DataOps pour Kubernetes utilise Trident pour provisionner les volumes mais simplifie le processus pour l'utilisateur. Se référer à la"documentation" pour plus de détails.

Exemples de tâches hautes performances pour les déploiements AIPod

Exécuter une charge de travail d'IA à nœud unique

Pour exécuter une tâche d'IA et de ML à nœud unique dans votre cluster Kubernetes, effectuez les tâches suivantes à partir de l'hôte de saut de déploiement. Avec Trident, vous pouvez rapidement et facilement rendre un volume de données, contenant potentiellement des pétaoctets de données, accessible à une charge de travail Kubernetes. Pour rendre un tel volume de données accessible depuis un pod Kubernetes, spécifiez simplement un PVC dans la définition du pod.

Cette section suppose que vous avez déjà conteneurisé (au format de conteneur Docker) la charge de travail IA et ML spécifique que vous tentez d'exécuter dans votre cluster Kubernetes.

1. Les exemples de commandes suivants montrent la création d'une tâche Kubernetes pour une charge de travail de référence TensorFlow qui utilise l'ensemble de données ImageNet. Pour plus d'informations sur l'ensemble de données ImageNet, consultez le "Site Web ImageNet" .

   Cet exemple de tâche nécessite huit GPU et peut donc s'exécuter sur un seul nœud de travail GPU doté de huit GPU ou plus. Cet exemple de travail peut être soumis dans un cluster pour lequel un nœud de travail comportant huit GPU ou plus n'est pas présent ou est actuellement occupé par une autre charge de travail. Si tel est le cas, le travail reste dans un état en attente jusqu'à ce qu'un tel nœud de travail soit disponible.

   De plus, afin de maximiser la bande passante de stockage, le volume contenant les données de formation nécessaires est monté deux fois dans le pod créé par cette tâche. Un autre volume est également monté dans la nacelle. Ce deuxième volume servira à stocker les résultats et les métriques. Ces volumes sont référencés dans la définition du travail en utilisant les noms des PVC. Pour plus d'informations sur les tâches Kubernetes, consultez le "documentation officielle de Kubernetes" .

   Un `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 l'environnement d'exécution du conteneur Docker peut parfois être insuffisant pour les besoins de TensorFlow. Montage d'un `emptyDir` le volume comme dans l'exemple suivant fournit un volume suffisamment grand `/dev/shm` volume virtuel. Pour plus d'informations sur `emptyDir` volumes, voir le "documentation officielle de Kubernetes" .

   Le conteneur unique spécifié dans cet exemple de définition de tâche reçoit un `securityContext > privileged` valeur de `true` . Cette valeur signifie que le conteneur dispose effectivement d'un accès root sur l'hôte. Cette annotation est utilisée dans ce cas car la charge de travail spécifique en cours d'exécution nécessite un accès root. Plus précisément, une opération de vidage du cache effectuée par la charge de travail nécessite un accès root. Que cela soit ou non `privileged: true` L'annotation est nécessaire en fonction 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. Confirmez que la tâche que vous avez créée à l'étape 1 s'exécute 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 est actuellement en cours d'exécution sur l'un des nœuds de travail 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. Confirmez que la tâche que vous avez créée à l'étape 1 se termine avec succès. Les exemples de commandes suivants confirment que le travail s'est 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.

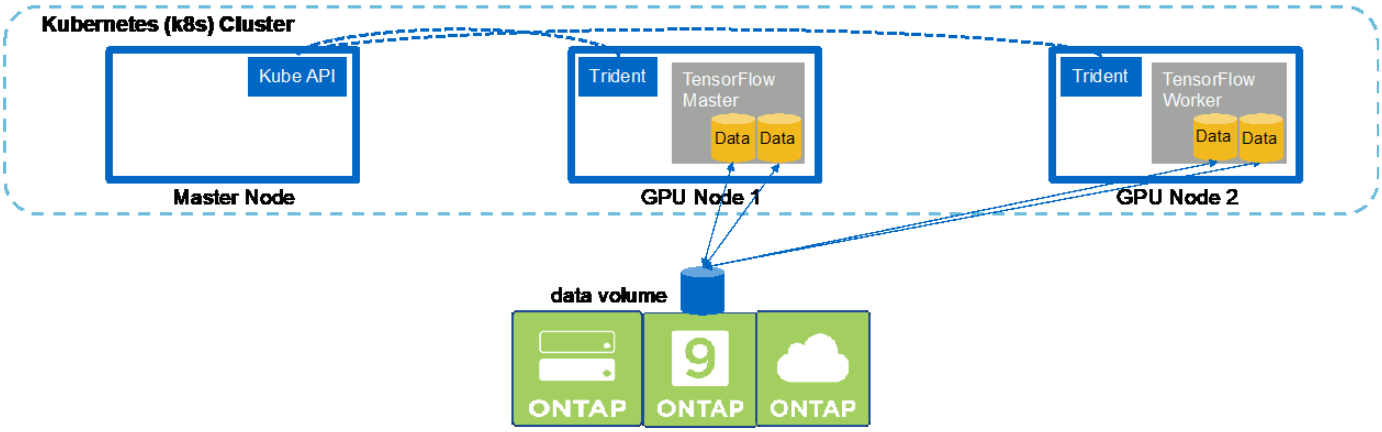
   Lorsque vous supprimez l'objet de travail, Kubernetes supprime automatiquement tous les pods associés.

```
$ 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.
```

Exécuter une charge de travail d'IA distribuée synchrone

Pour exécuter une tâche IA et ML multinœud synchrone dans votre cluster Kubernetes, effectuez les tâches suivantes sur l'hôte de saut de déploiement. Ce processus vous permet de tirer parti des données stockées sur un volume NetApp et d'utiliser plus de GPU qu'un seul nœud de travail ne peut en fournir. Consultez la figure suivante pour une représentation d'un travail d'IA distribué synchrone.

Les tâches distribuées synchrones peuvent aider à augmenter les performances et la précision de la formation par rapport aux tâches distribuées asynchrones. Une discussion sur les avantages et les inconvénients des tâches synchrones par rapport aux tâches asynchrones n'entre pas dans le cadre de ce document.



1. Les exemples de commandes suivants montrent la création d'un worker qui participe à l'exécution distribuée synchrone du même travail de référence TensorFlow qui a été exécuté sur un seul nœud dans l'exemple de la section"Exécuter une charge de travail d'IA à nœud unique" . Dans cet exemple spécifique, un seul travailleur est déployé car le travail est exécuté sur deux nœuds de travail.

Cet exemple de déploiement de travailleur nécessite huit GPU et peut donc s'exécuter sur un seul nœud de travail GPU doté de huit GPU ou plus. Si vos nœuds de travail GPU comportent plus de huit GPU, pour optimiser les performances, vous souhaiterez peut-être augmenter ce nombre pour qu'il soit égal au nombre de GPU dont disposent vos nœuds de travail. Pour plus d'informations sur les déploiements Kubernetes, consultez le "documentation officielle de Kubernetes" .

Un déploiement Kubernetes est créé dans cet exemple car ce travailleur conteneurisé spécifique ne se terminerait jamais tout seul. Par conséquent, il n'est pas logique de le déployer en utilisant la construction de tâche Kubernetes. Si votre worker est conçu ou écrit pour s'exécuter de manière autonome, il peut être judicieux d'utiliser la construction de tâche pour déployer votre worker.

Le pod spécifié dans cet exemple de spécification de déploiement reçoit un `hostNetwork` valeur de `true` . Cette valeur signifie que le pod utilise la pile réseau du nœud de travail hôte au lieu de la pile réseau virtuelle que Kubernetes crée habituellement pour chaque pod. Cette annotation est utilisée dans ce cas car la charge de travail spécifique s'appuie sur Open MPI, NCCL et Horovod pour exécuter la charge de travail de manière distribuée synchrone. Par conséquent, il nécessite un accès à la pile réseau de l'hôte. Une discussion sur Open MPI, NCCL et Horovod sort du cadre de ce document. Que cela soit ou non `hostNetwork: true` L'annotation est nécessaire en fonction des exigences de la charge de travail spécifique que vous exécutez. Pour plus d'informations sur le `hostNetwork` champ, voir le "documentation officielle de 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 travail que vous avez créé à l'étape 1 a été lancé avec succès. Les exemples de commandes suivants confirment qu'un seul pod de travail a été créé pour le déploiement, comme indiqué dans la définition de déploiement, et que ce pod est actuellement en cours d'exécution sur l'un des nœuds de travail 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 une tâche Kubernetes pour un maître qui démarre, participe et suit l'exécution de la tâche multinœud synchrone. Les exemples de commandes suivants créent un maître qui lance, participe et suit l'exécution distribuée synchrone du même travail de référence TensorFlow qui a été exécuté sur un seul

nœud dans l'exemple de la section"Exécuter une charge de travail d'IA à nœud unique" .

Cet exemple de tâche principale demande huit GPU et peut donc s'exécuter sur un seul nœud de travail GPU doté de huit GPU ou plus. Si vos nœuds de travail GPU comportent plus de huit GPU, pour optimiser les performances, vous souhaiterez peut-être augmenter ce nombre pour qu'il soit égal au nombre de GPU dont disposent vos nœuds de travail.

Le pod maître spécifié dans cet exemple de définition de tâche reçoit un `hostNetwork` valeur de `true` , tout comme le groupe de travailleurs a reçu un `hostNetwork` valeur de `true` à l'étape 1. Consultez 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. Confirmez que le travail principal que vous avez créé à l'étape 3 s'exécute correctement. L'exemple de commande suivant confirme qu'un seul pod maître a été créé pour le travail, comme indiqué dans la définition du travail, et que ce pod est actuellement en cours d'exécution sur l'un des nœuds de travail GPU. Vous devriez également voir que le pod de travail que vous avez vu à l'origine à l'étape 1 est toujours en cours d'exécution et que les pods maître et de travail s'exécutent sur des nœuds différents.

```
$ 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 se termine avec succès. Les exemples de commandes suivants confirment que le travail s'est 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 du travailleur lorsque vous n'en avez plus besoin. Les exemples de commandes suivants montrent la suppression de l'objet de déploiement Worker créé à l'étape 1.

   Lorsque vous supprimez l'objet de déploiement de travail, Kubernetes supprime automatiquement tous les pods de travail 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 principal créé à l'étape 3.

   Lorsque vous supprimez l'objet de tâche principal, Kubernetes supprime automatiquement tous les pods principaux 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.
```