MLOps de código abierto con NetApp

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Tabla de contenidos

MLOps de código abierto con NetApp

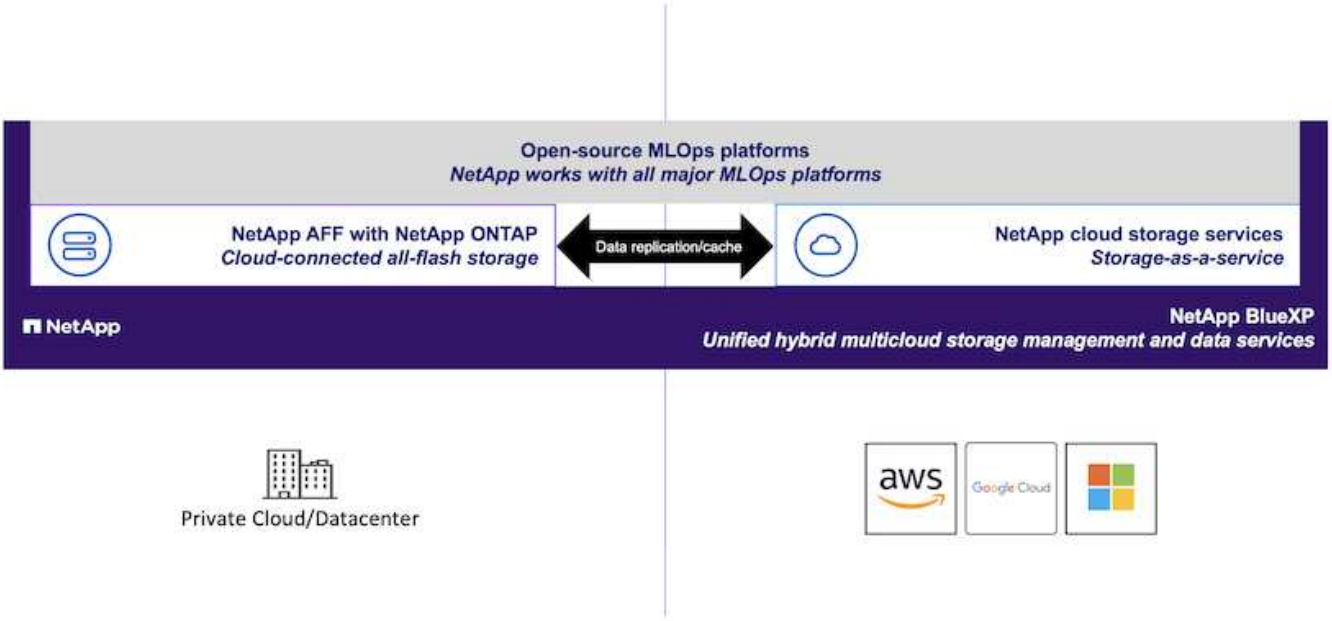
MLOps de código abierto con NetApp

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

Empresas y organizaciones de todos los tamaños y de muchas industrias están recurriendo a la inteligencia artificial (IA) para resolver problemas del mundo real, ofrecer productos y servicios innovadores y obtener una ventaja en un mercado cada vez más competitivo. Muchas organizaciones están recurriendo a herramientas MLOps de código abierto para mantenerse al día con el rápido ritmo de innovación en la industria. Estas herramientas de código abierto ofrecen capacidades avanzadas y características de última generación, pero a menudo no tienen en cuenta la disponibilidad ni la seguridad de los datos. Lamentablemente, esto significa que los científicos de datos altamente capacitados se ven obligados a pasar una cantidad significativa de tiempo esperando obtener acceso a los datos o esperando que se completen operaciones rudimentarias relacionadas con los datos. Al combinar las populares herramientas MLOps de código abierto con una infraestructura de datos inteligente de NetApp, las organizaciones pueden acelerar sus canales de datos, lo que, a su vez, acelera sus iniciativas de IA. Pueden extraer valor de sus datos y al mismo tiempo garantizar que permanezcan protegidos y seguros. Esta solución demuestra la combinación de las capacidades de gestión de datos de NetApp con varias herramientas y marcos de código abierto populares para abordar estos desafíos.

La siguiente lista destaca algunas capacidades clave que permite esta solución:

- Los usuarios pueden aprovisionar rápidamente nuevos volúmenes de datos de alta capacidad y espacios de trabajo de desarrollo respaldados por almacenamiento NetApp de alto rendimiento y escalabilidad horizontal.
- Los usuarios pueden clonar casi instantáneamente volúmenes de datos de alta capacidad y espacios de trabajo de desarrollo para permitir la experimentación o la iteración rápida.
- Los usuarios pueden guardar casi instantáneamente instantáneas de volúmenes de datos de alta capacidad y espacios de trabajo de desarrollo para realizar copias de seguridad y/o trazabilidad/establecimiento de referencia.



Un flujo de trabajo típico de MLOps incorpora espacios de trabajo de desarrollo, que generalmente toman la forma de"Cuadernos Jupyter" seguimiento de experimentos; canales de entrenamiento automatizados; canales de datos; e inferencia/implementación. Esta solución destaca varias herramientas y marcos diferentes que pueden usarse de forma independiente o en conjunto para abordar los diferentes aspectos del flujo de trabajo. También demostramos la combinación de las capacidades de gestión de datos de NetApp con cada una de estas herramientas. Esta solución está diseñada para ofrecer bloques de construcción a partir de los cuales una organización puede construir un flujo de trabajo MLOps personalizado que sea específico para sus casos de uso y requisitos.

Esta solución cubre las siguientes herramientas y marcos:

- "Flujo de aire de Apache"
- "JupyterHub"
- "Kubeflow"
- "Flujo de ml"

La siguiente lista describe patrones comunes para implementar estas herramientas de forma independiente o en conjunto.

- Implementar JupyterHub, MLflow y Apache Airflow en conjunto - JupyterHub para"Cuadernos Jupyter" , MLflow para seguimiento de experimentos y Apache Airflow para capacitación automatizada y canalización de datos.
- Implementar Kubeflow y Apache Airflow en conjunto - Kubeflow para"Cuadernos Jupyter" seguimiento de experimentos, canales de entrenamiento automatizados e inferencia; y Apache Airflow para canales de datos.
- Implemente Kubeflow como una solución de plataforma MLOps todo en uno para"Cuadernos Jupyter" , seguimiento de experimentos, entrenamiento automatizado y canalización de datos, e inferencia.

Descripción general de la tecnología

Esta sección se centra en la descripción general de la tecnología para MLOps de código abierto con NetApp.

Inteligencia artificial

La IA es una disciplina de la informática en la que se entrena a las computadoras para imitar las funciones cognitivas de la mente humana. Los desarrolladores de IA entrenan a las computadoras para aprender y resolver problemas de una manera similar, o incluso superior, a la de los humanos. El aprendizaje profundo y el aprendizaje automático son subcampos de la IA. Las organizaciones están adoptando cada vez más IA, ML y DL para respaldar sus necesidades comerciales críticas. Algunos ejemplos son los siguientes:

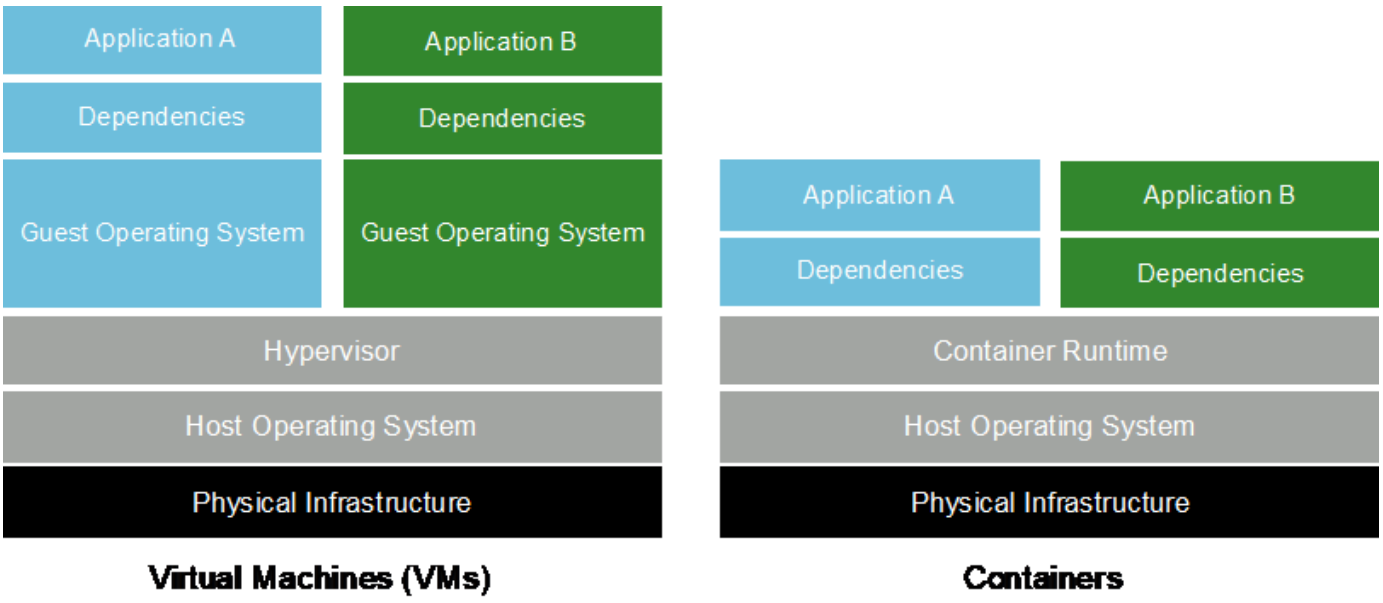
- Analizar grandes cantidades de datos para descubrir información empresarial previamente desconocida
- Interactuar directamente con los clientes mediante el procesamiento del lenguaje natural
- Automatizar diversos procesos y funciones empresariales

Las cargas de trabajo de inferencia y entrenamiento de IA modernas requieren capacidades informáticas masivamente paralelas. Por lo tanto, las GPU se utilizan cada vez más para ejecutar operaciones de IA porque las capacidades de procesamiento paralelo de las GPU son muy superiores a las de las CPU de propósito general.

Contenedores

Los contenedores son instancias de espacio de usuario aisladas que se ejecutan sobre un núcleo de sistema operativo host compartido. La adopción de contenedores está aumentando rápidamente. Los contenedores ofrecen muchos de los mismos beneficios de sandbox de aplicaciones que ofrecen las máquinas virtuales (VM). Sin embargo, debido a que se han eliminado las capas de hipervisor y sistema operativo invitado de los que dependen las máquinas virtuales, los contenedores son mucho más livianos. La siguiente figura muestra una visualización de máquinas virtuales versus contenedores.

Los contenedores también permiten el empaquetado eficiente de dependencias de aplicaciones, tiempos de ejecución, etc., directamente con una aplicación. El formato de embalaje de contenedores más utilizado es el contenedor Docker. Una aplicación que se ha contenedorizado en el formato de contenedor Docker se puede ejecutar en cualquier máquina que pueda ejecutar contenedores Docker. Esto es cierto incluso si las dependencias de la aplicación no están presentes en la máquina porque todas las dependencias están empaquetadas en el propio contenedor. Para obtener más información, visite el "Sitio web de Docker" .



Kubernetes

Kubernetes es una plataforma de orquestación de contenedores distribuida y de código abierto que fue diseñada originalmente por Google y ahora es mantenida por la Cloud Native Computing Foundation (CNCF). Kubernetes permite la automatización de funciones de implementación, administración y escalamiento para aplicaciones en contenedores. En los últimos años, Kubernetes ha surgido como la plataforma dominante de orquestación de contenedores. Para obtener más información, visite el "Sitio web de Kubernetes" .

Trident de NetApp

"Trident"permite el consumo y la gestión de recursos de almacenamiento en todas las plataformas de almacenamiento NetApp más populares, en la nube pública o en las instalaciones, incluidas ONTAP (AFF, FAS, Select, Cloud, Amazon FSx ONTAP), el servicio Azure NetApp Files y Google Cloud NetApp Volumes. Trident es un orquestador de almacenamiento dinámico compatible con la interfaz de almacenamiento de contenedores (CSI) que se integra de forma nativa con Kubernetes.

Kit de herramientas DataOps de NetApp

El"Kit de herramientas DataOps de NetApp" es una herramienta basada en Python que simplifica la gestión de espacios de trabajo de desarrollo/entrenamiento y servidores de inferencia respaldados por almacenamiento NetApp de alto rendimiento y escalabilidad horizontal. Las capacidades clave incluyen:

- Aprovisione rápidamente nuevos espacios de trabajo de alta capacidad respaldados por almacenamiento NetApp de alto rendimiento y escalabilidad horizontal.
- Clonar casi instantáneamente espacios de trabajo de alta capacidad para permitir la experimentación o la iteración rápida.
- Guarde casi instantáneamente instantáneas de espacios de trabajo de alta capacidad para realizar copias de seguridad y/o trazabilidad/establecimiento de referencia.
- Aprovisione, clone y cree instantáneas de volúmenes de datos de alto rendimiento y alta capacidad de manera casi instantánea.

Flujo de aire de Apache

Apache Airflow es una plataforma de gestión de flujo de trabajo de código abierto que permite la creación, programación y supervisión programática de flujos de trabajo empresariales complejos. A menudo se utiliza para automatizar flujos de trabajo de ETL y canalización de datos, pero no se limita a estos tipos de flujos de trabajo. El proyecto Airflow fue iniciado por Airbnb, pero desde entonces se ha vuelto muy popular en la industria y ahora está bajo los auspicios de The Apache Software Foundation. Airflow está escrito en Python, los flujos de trabajo de Airflow se crean a través de scripts de Python y Airflow está diseñado bajo el principio de "configuración como código". Muchos usuarios empresariales de Airflow ahora ejecutan Airflow sobre Kubernetes.

Gráficos acíclicos dirigidos (DAG)

En Airflow, los flujos de trabajo se denominan gráficos acíclicos dirigidos (DAG). Los DAG se componen de tareas que se ejecutan en secuencia, en paralelo o en una combinación de ambas, según la definición del DAG. El programador Airflow ejecuta tareas individuales en una matriz de trabajadores, cumpliendo con las dependencias a nivel de tarea que se especifican en la definición de DAG. Los DAG se definen y crean mediante scripts de Python.

Cuaderno Jupyter

Los Jupyter Notebooks son documentos tipo wiki que contienen código en vivo y texto descriptivo. Los Jupyter Notebooks se utilizan ampliamente en la comunidad de IA y ML como un medio para documentar, almacenar y compartir proyectos de IA y ML. Para obtener más información sobre Jupyter Notebooks, visite el sitio web "Sitio web de Jupyter" .

Servidor de Jupyter Notebook

Un servidor Jupyter Notebook es una aplicación web de código abierto que permite a los usuarios crear Jupyter Notebooks.

JupyterHub

JupyterHub es una aplicación multiusuario que permite a un usuario individual aprovisionar y acceder a su propio servidor Jupyter Notebook. Para obtener más información sobre JupyterHub, visite el sitio "Sitio web de JupyterHub" .

Flujo de ml

MLflow es una popular plataforma de gestión del ciclo de vida de la IA de código abierto. Las características clave de MLflow incluyen el seguimiento de experimentos de IA/ML y un repositorio de modelos de IA/ML. Para obtener más información sobre MLflow, visite el sitio web "Sitio web de MLflow" .

Kubeflow

Kubeflow es un kit de herramientas de inteligencia artificial y aprendizaje automático de código abierto para Kubernetes que fue desarrollado originalmente por Google. El proyecto Kubeflow hace que las implementaciones de flujos de trabajo de IA y ML en Kubernetes sean simples, portátiles y escalables. Kubeflow abstrae las complejidades de Kubernetes, lo que permite a los científicos de datos centrarse en lo que mejor saben: la ciencia de datos. Vea la siguiente figura para ver una visualización. Kubeflow es una buena opción de código abierto para las organizaciones que prefieren una plataforma MLOps todo en uno. Para obtener más información, visite el "Sitio web de Kubeflow" .

Tuberías de Kubeflow

Las canalizaciones de Kubeflow son un componente clave de Kubeflow. Kubeflow Pipelines es una plataforma y un estándar para definir e implementar flujos de trabajo de IA y ML portátiles y escalables. Para obtener más información, consulte la "documentación oficial de Kubeflow" .

Cuadernos de Kubeflow

Kubeflow simplifica el aprovisionamiento y la implementación de servidores Jupyter Notebook en Kubernetes. Para obtener más información sobre Jupyter Notebooks en el contexto de Kubeflow, consulte "documentación oficial de Kubeflow" .

Katib

Katib es un proyecto nativo de Kubernetes para el aprendizaje automático automatizado (AutoML). Katib admite el ajuste de hiperparámetros, la detención anticipada y la búsqueda de arquitectura neuronal (NAS). Katib es un proyecto que es agnóstico a los marcos de aprendizaje automático (ML). Puede ajustar hiperparámetros de aplicaciones escritas en cualquier lenguaje elegido por el usuario y admite de forma nativa muchos marcos de ML, como TensorFlow, MXNet, PyTorch, XGBoost y otros. Katib admite muchos algoritmos AutoML diferentes, como optimización bayesiana, estimadores de árbol de Parzen, búsqueda aleatoria,

estrategia de evolución de adaptación de matriz de covarianza, hiperbanda, búsqueda de arquitectura neuronal eficiente, búsqueda de arquitectura diferenciable y muchos más. Para obtener más información sobre Jupyter Notebooks en el contexto de Kubeflow, consulte "documentación oficial de Kubeflow" .

ONTAP de NetApp

ONTAP 9, la última generación de software de gestión de almacenamiento de NetApp, permite a las empresas modernizar la infraestructura y realizar la transición a un centro de datos preparado para la nube. Al aprovechar las capacidades de gestión de datos líderes en la industria, ONTAP permite la gestión y protección de datos con un único conjunto de herramientas, independientemente de dónde residan esos datos. También puede mover datos libremente a donde sea necesario: el borde, el núcleo o la nube. ONTAP 9 incluye numerosas características que simplifican la gestión de datos, aceleran y protegen datos críticos y habilitan capacidades de infraestructura de próxima generación en arquitecturas de nube híbrida.

Simplificar la gestión de datos

La gestión de datos es crucial para las operaciones de TI de la empresa y los científicos de datos, de modo que se utilicen los recursos adecuados para las aplicaciones de IA y el entrenamiento de conjuntos de datos de IA/ML. La siguiente información adicional sobre las tecnologías de NetApp está fuera del alcance de esta validación, pero podría ser relevante según su implementación.

El software de gestión de datos ONTAP incluye las siguientes características para optimizar y simplificar las operaciones y reducir el costo total de operación:

- Compactación de datos en línea y deduplicación ampliada. La compactación de datos reduce el espacio desperdiciado dentro de los bloques de almacenamiento y la deduplicación aumenta significativamente la capacidad efectiva. Esto se aplica a los datos almacenados localmente y a los datos almacenados en la nube.
- Calidad de servicio mínima, máxima y adaptativa (AQoS). Los controles granulares de calidad de servicio (QoS) ayudan a mantener los niveles de rendimiento de las aplicaciones críticas en entornos altamente compartidos.
- FabricPool de NetApp . Proporciona niveles automáticos de datos fríos en opciones de almacenamiento en la nube pública y privada, incluidas Amazon Web Services (AWS), Azure y la solución de almacenamiento NetApp StorageGRID . Para obtener más información sobre FabricPool, consulte "TR-4598: Prácticas recomendadas de FabricPool" .

Acelerar y proteger los datos

ONTAP ofrece niveles superiores de rendimiento y protección de datos y amplía estas capacidades de las siguientes maneras:

- Rendimiento y menor latencia. ONTAP ofrece el mayor rendimiento posible con la menor latencia posible.
- Protección de datos. ONTAP proporciona capacidades de protección de datos integradas con gestión común en todas las plataformas.
- Cifrado de volumen de NetApp (NVE). ONTAP ofrece cifrado nativo a nivel de volumen con soporte para administración de claves interna y externa.
- Autenticación multitenencia y multifactor. ONTAP permite compartir recursos de infraestructura con los más altos niveles de seguridad.

Infraestructura a prueba de futuro

ONTAP ayuda a satisfacer necesidades comerciales exigentes y en constante cambio con las siguientes

características:

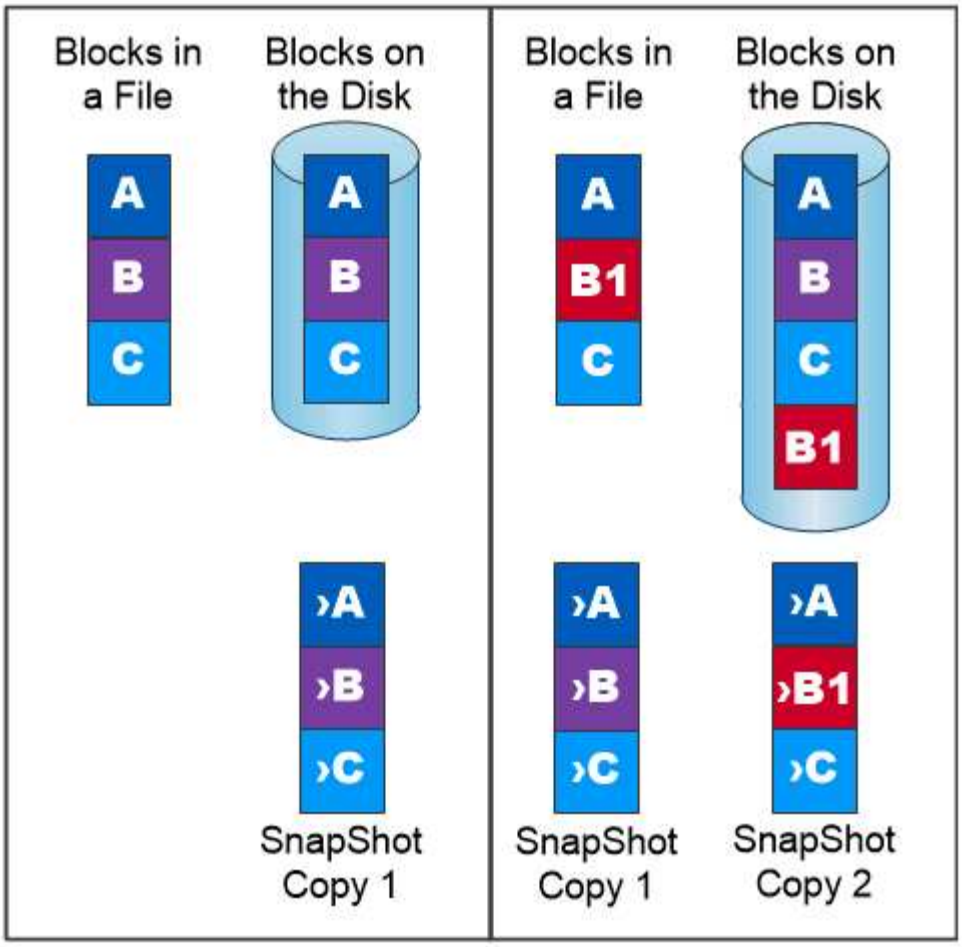
- Escalabilidad fluida y operaciones sin interrupciones. ONTAP admite la incorporación de capacidad sin interrupciones a controladores existentes y a clústeres de escalamiento horizontal. Los clientes pueden actualizar a las últimas tecnologías sin necesidad de costosas migraciones de datos ni interrupciones.
- Conexión a la nube. ONTAP es el software de gestión de almacenamiento más conectado a la nube, con opciones para almacenamiento definido por software e instancias nativas de la nube en todas las nubes públicas.
- Integración con aplicaciones emergentes. ONTAP ofrece servicios de datos de nivel empresarial para plataformas y aplicaciones de próxima generación, como vehículos autónomos, ciudades inteligentes e Industria 4.0, utilizando la misma infraestructura que respalda las aplicaciones empresariales existentes.

Copias instantáneas de NetApp

Una copia Snapshot de NetApp es una imagen de un volumen en un punto en el tiempo y de solo lectura. La imagen consume un espacio de almacenamiento mínimo y genera una sobrecarga de rendimiento insignificante porque solo registra los cambios en los archivos creados desde que se realizó la última copia de instantánea, como se muestra en la siguiente figura.

Las copias instantáneas deben su eficiencia a la tecnología central de virtualización de almacenamiento de ONTAP , Write Anywhere File Layout (WAFL). Al igual que una base de datos, WAFL utiliza metadatos para señalar bloques de datos reales en el disco. Pero, a diferencia de una base de datos, WAFL no sobrescribe los bloques existentes. Escribe datos actualizados en un nuevo bloque y cambia los metadatos. Esto se debe a que ONTAP hace referencia a metadatos cuando crea una copia instantánea, en lugar de copiar bloques de datos, que las copias instantáneas son tan eficientes. Al hacerlo así se elimina el tiempo de búsqueda que otros sistemas incurren para localizar los bloques a copiar, así como el coste de realizar la copia en sí.

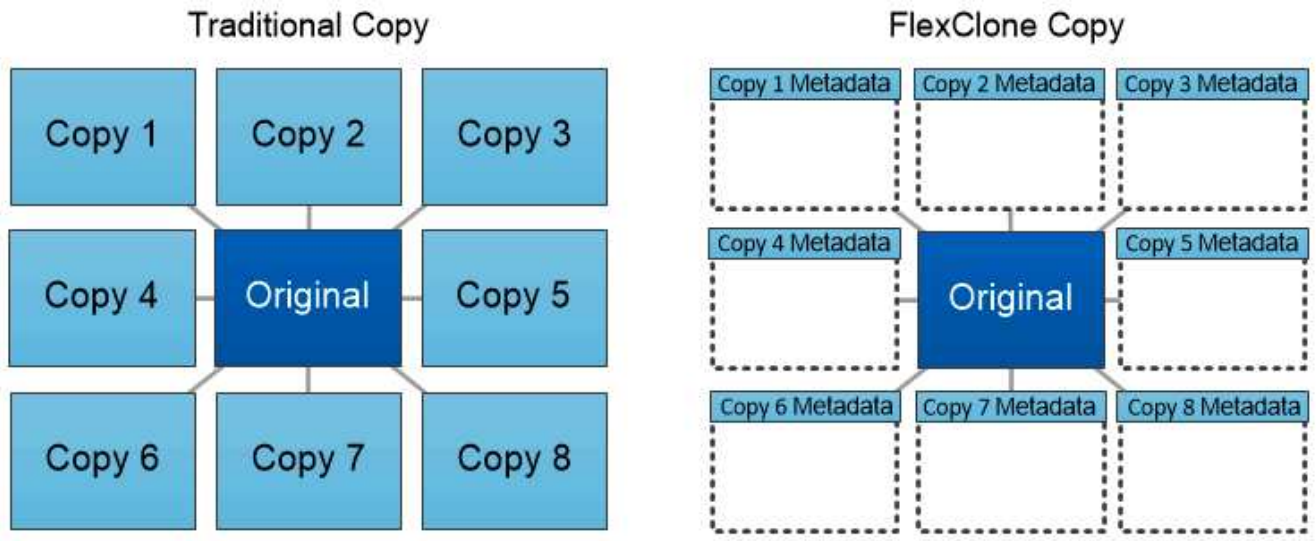
Puede utilizar una copia instantánea para recuperar archivos individuales o LUN o para restaurar todo el contenido de un volumen. ONTAP compara la información del puntero en la copia instantánea con los datos del disco para reconstruir el objeto faltante o dañado, sin tiempo de inactividad ni un costo de rendimiento significativo.



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

Tecnología FlexClone de NetApp

La tecnología NetApp FlexClone hace referencia a metadatos de Snapshot para crear copias grabables en un punto determinado del tiempo de un volumen. Las copias comparten bloques de datos con sus padres y no consumen almacenamiento excepto lo necesario para los metadatos hasta que se escriben los cambios en la copia, como se muestra en la siguiente figura. Donde las copias tradicionales pueden tardar minutos o incluso horas en crearse, el software FlexClone le permite copiar incluso los conjuntos de datos más grandes casi instantáneamente. Esto lo hace ideal para situaciones en las que se necesitan múltiples copias de conjuntos de datos idénticos (un espacio de trabajo de desarrollo, por ejemplo) o copias temporales de un conjunto de datos (probar una aplicación contra un conjunto de datos de producción).



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

Tecnología de replicación de datos SnapMirror de NetApp

El software NetApp SnapMirror es una solución de replicación unificada rentable y fácil de usar en toda la estructura de datos. Replica datos a altas velocidades a través de LAN o WAN. Le ofrece alta disponibilidad de datos y rápida replicación de datos para aplicaciones de todo tipo, incluidas aplicaciones críticas para el negocio en entornos virtuales y tradicionales. Cuando replica datos a uno o más sistemas de almacenamiento NetApp y actualiza continuamente los datos secundarios, sus datos se mantienen actualizados y están disponibles siempre que los necesite. No se requieren servidores de replicación externos. Vea la siguiente figura para ver un ejemplo de una arquitectura que aprovecha la tecnología SnapMirror .

El software SnapMirror aprovecha las eficiencias de almacenamiento de NetApp ONTAP al enviar solo los bloques modificados a través de la red. El software SnapMirror también utiliza compresión de red incorporada para acelerar las transferencias de datos y reducir la utilización del ancho de banda de la red hasta en un 70%. Con la tecnología SnapMirror , puede aprovechar un flujo de datos de replicación delgada para crear un único repositorio que mantenga tanto el espejo activo como las copias de puntos en el tiempo anteriores, lo que reduce el tráfico de red hasta en un 50%.

Copia y sincronización de NetApp BlueXP

"Copia y sincronización de BlueXP"Es un servicio de NetApp para la sincronización de datos rápida y segura. Ya sea que necesite transferir archivos entre recursos compartidos de archivos NFS o SMB locales, NetApp StorageGRID, NetApp ONTAP S3, Google Cloud NetApp Volumes, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage o IBM Cloud Object Storage, BlueXP Copy and Sync mueve los archivos donde los necesita de manera rápida y segura.

Una vez transferidos los datos, estarán totalmente disponibles para su uso tanto en el origen como en el destino. BlueXP Copy and Sync puede sincronizar datos a pedido cuando se activa una actualización o sincronizar datos de manera continua según un cronograma predefinido. De todos modos, BlueXP Copy and Sync solo mueve los deltas, por lo que se minimiza el tiempo y el dinero gastados en la replicación de datos.

BlueXP Copy and Sync es una herramienta de software como servicio (SaaS) extremadamente sencilla de configurar y utilizar. Las transferencias de datos que se activan mediante BlueXP Copy and Sync se llevan a cabo a través de corredores de datos. Los agentes de datos de copia y sincronización de BlueXP se pueden

implementar en AWS, Azure, Google Cloud Platform o en las instalaciones locales.

XCP de NetApp

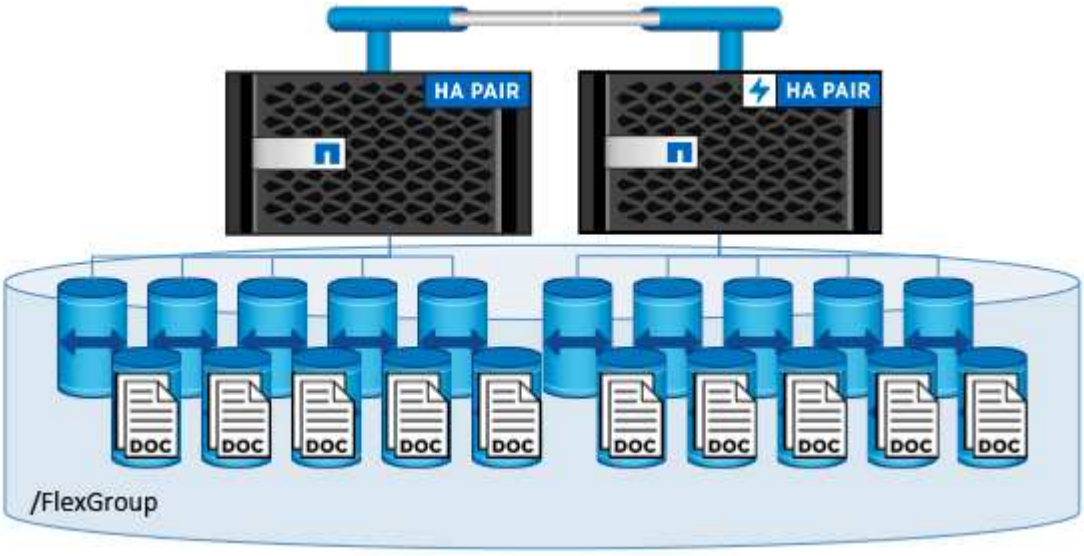
"XCP de NetApp"es un software basado en el cliente para migraciones de datos de cualquier plataforma NetApp y de NetApp a NetApp , así como para obtener información sobre sistemas de archivos. XCP está diseñado para escalar y lograr el máximo rendimiento al utilizar todos los recursos del sistema disponibles para manejar conjuntos de datos de gran volumen y migraciones de alto rendimiento. XCP le ayuda a obtener visibilidad completa del sistema de archivos con la opción de generar informes.

Volúmenes FlexGroup de NetApp ONTAP

Un conjunto de datos de entrenamiento puede ser una colección de potencialmente miles de millones de archivos. Los archivos pueden incluir texto, audio, video y otras formas de datos no estructurados que deben almacenarse y procesarse para poder leerse en paralelo. El sistema de almacenamiento debe almacenar grandes cantidades de archivos pequeños y debe leer esos archivos en paralelo para realizar E/S secuenciales y aleatorias.

Un volumen FlexGroup es un espacio de nombres único que comprende múltiples volúmenes miembros constituyentes, como se muestra en la siguiente figura. Desde el punto de vista de un administrador de almacenamiento, un volumen FlexGroup se administra y actúa como un FlexVol volume de NetApp . Los archivos de un volumen FlexGroup se asignan a volúmenes miembro individuales y no se distribuyen entre volúmenes o nodos. Permiten las siguientes capacidades:

- Los volúmenes FlexGroup proporcionan múltiples petabytes de capacidad y una latencia baja predecible para cargas de trabajo con alto contenido de metadatos.
- Admiten hasta 400 mil millones de archivos en el mismo espacio de nombres.
- Admiten operaciones paralelizadas en cargas de trabajo NAS en CPU, nodos, agregados y volúmenes FlexVol constituyentes.



Arquitectura

Esta solución no depende de hardware específico. La solución es compatible con

cualquier dispositivo de almacenamiento físico, instancia definida por software o servicio en la nube de NetApp que sea compatible con NetApp Trident. Los ejemplos incluyen un sistema de almacenamiento NetApp AFF , Amazon FSx ONTAP, Azure NetApp Files, Google Cloud NetApp Volumes o una instancia de NetApp Cloud Volumes ONTAP . Además, la solución se puede implementar en cualquier clúster de Kubernetes siempre que la versión de Kubernetes utilizada sea compatible con NetApp Trident y los demás componentes de la solución que se estén implementando. Para obtener una lista de las versiones de Kubernetes compatibles con Trident, consulte la "Documentación de Trident" . Consulte las siguientes tablas para obtener detalles sobre los entornos que se utilizaron para validar los distintos componentes de esta solución.

Entorno de validación de Apache Airflow

Componente de software	Versión
Flujo de aire de Apache	2.0.1, implementado a través de"Diagrama del timón Apache Airflow" 8.0.8
Kubernetes	1,18
Trident de NetApp	21,01

Entorno de validación de JupyterHub

Componente de software	Versión
JupyterHub	4.1.5, implementado a través de"Gráfico de Helm de JupyterHub" 3.3.7
Kubernetes	1,29
Trident de NetApp	24,02

Entorno de validación de MLflow

Componente de software	Versión
Flujo de ml	2.14.1, implementado a través de"Diagrama de Helm de MLflow" 1.4.12
Kubernetes	1,29
Trident de NetApp	24,02

Entorno de validación de Kubeflow

Componente de software	Versión
Kubeflow	1.7, implementado a través de"desplegarKF" 0.1.1
Kubernetes	1,26
Trident de NetApp	23,07

Soporte

NetApp no ofrece soporte empresarial para Apache Airflow, JupyterHub, MLflow, Kubeflow o Kubernetes. Si está interesado en una plataforma MLOps totalmente compatible,"Contactar con NetApp" sobre soluciones MLOps totalmente compatibles que NetApp ofrece conjuntamente con sus socios.

Configuración de NetApp Trident

Ejemplos de backends Trident para implementaciones de NetApp AIPod

Antes de poder usar Trident para aprovisionar dinámicamente recursos de almacenamiento dentro de su clúster de Kubernetes, debe crear uno o más backends de Trident . Los ejemplos que siguen representan diferentes tipos de backends que podría querer crear si está implementando componentes de esta solución en un"NetApp AIPod" . Para obtener más información sobre backends y, por ejemplo, backends para otras plataformas/entornos, consulte"Documentación de Trident" .

1. NetApp recomienda crear un backend Trident habilitado para FlexGroup para su AIPod.

   Los comandos de ejemplo que siguen muestran la creación de un Trident Backend habilitado para FlexGroup para una máquina virtual de almacenamiento AIPod (SVM). Este backend utiliza el `ontap-nas-flexgroup` controlador de almacenamiento. ONTAP admite dos tipos principales de volúmenes de datos: FlexVol y FlexGroup. Los volúmenes FlexVol tienen un límite de tamaño (al momento de escribir este artículo, el tamaño máximo depende de la implementación específica). Por otro lado, los volúmenes FlexGroup pueden escalar linealmente hasta 20 PB y 400 mil millones de archivos, proporcionando un único espacio de nombres que simplifica enormemente la gestión de datos. Por lo tanto, los volúmenes FlexGroup son óptimos para cargas de trabajo de IA y ML que dependen de grandes cantidades de datos.

   Si está trabajando con una pequeña cantidad de datos y desea utilizar volúmenes FlexVol en lugar de volúmenes FlexGroup , puede crear Trident Backends que utilicen los `ontap-nas` controlador de almacenamiento en lugar del `ontap-nas-flexgroup` controlador de almacenamiento.

```
$ 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 también recomienda crear un backend Trident habilitado para FlexVol . Es posible que desee utilizar volúmenes FlexVol para alojar aplicaciones persistentes, almacenar resultados, salida, información de depuración, etc. Si desea utilizar volúmenes FlexVol , debe crear uno o más Trident Backends habilitados para FlexVol . Los comandos de ejemplo que siguen muestran la creación de un único backend Trident habilitado para 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 |
+-------------------------+--------------------
+--------------------------------------+--------+---------+
```

Ejemplos de clases de almacenamiento de Kubernetes para implementaciones de NetApp AIPod

Antes de poder usar Trident para aprovisionar dinámicamente recursos de almacenamiento dentro de su clúster de Kubernetes, debe crear una o más StorageClasses de Kubernetes. Los ejemplos que siguen representan diferentes tipos de StorageClasses que podría querer crear si está implementando componentes de esta

solución en un"NetApp AIPod" . Para obtener más información sobre StorageClasses y, por ejemplo, StorageClasses para otras plataformas/entornos, consulte la"Documentación de Trident" .

1. NetApp recomienda crear una StorageClass para el Trident Backend habilitado para FlexGroup que creó en la sección"Ejemplos de backends Trident para implementaciones de NetApp AIPod" , paso 1. Los comandos de ejemplo que siguen muestran la creación de múltiples StorageClasses que corresponden al ejemplo Backend que se creó en la sección"Ejemplos de backends Trident para implementaciones de NetApp AIPod" , paso 1 - uno que utilice"NFS sobre RDMA" y uno que no.

   Para que un volumen persistente no se elimine cuando se elimina el PersistentVolumeClaim (PVC) correspondiente, el siguiente ejemplo utiliza un `reclaimPolicy` valor de `Retain` . Para obtener más información sobre la `reclaimPolicy` campo, ver el oficial "Documentación de Kubernetes" .

   Nota: Las siguientes clases de almacenamiento de ejemplo utilizan un tamaño de transferencia máximo de 262144. Para utilizar este tamaño de transferencia máximo, debe configurar el tamaño de transferencia máximo en su sistema ONTAP según corresponda. Consulte la"Documentación de ONTAP" Para más detalles.

   Nota: Para utilizar NFS sobre RDMA, debe configurar NFS sobre RDMA en su sistema ONTAP . Consulte la"Documentación de ONTAP" Para más detalles.

   Nota: En el siguiente ejemplo, se especifica un Backend específico en el campo storagePool en el archivo de definición 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 también recomienda crear una StorageClass que corresponda al Trident Backend habilitado para FlexVol que creó en la sección"Ejemplos de backends Trident para implementaciones de AIPod" , paso 2. Los comandos de ejemplo que siguen muestran la creación de una única StorageClass para volúmenes FlexVol .

   Nota: En el siguiente ejemplo, no se especifica un Backend particular en el campo storagePool en el archivo de definición StorageClass. Cuando utiliza Kubernetes para administrar volúmenes mediante esta StorageClass, Trident intenta usar cualquier backend disponible que use la `ontap-nas` conductor.

```
$ 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
```

Flujo de aire de Apache

Implementación de Apache Airflow

Esta sección describe las tareas que debe completar para implementar Airflow en su clúster de Kubernetes.

Es posible implementar Airflow en plataformas distintas a Kubernetes. La implementación de Airflow en plataformas distintas a Kubernetes está fuera del alcance de esta solución.

Prerrequisitos

Antes de realizar el ejercicio de implementación que se describe en esta sección, asumimos que ya ha realizado las siguientes tareas:

1. Ya tienes un clúster de Kubernetes en funcionamiento.
2. Ya ha instalado y configurado NetApp Trident en su clúster de Kubernetes. Para obtener más detalles sobre Trident, consulte la"Documentación de Trident" .

Instalar Helm

Airflow se implementa utilizando Helm, un administrador de paquetes popular para Kubernetes. Antes de implementar Airflow, debe instalar Helm en el host de salto de implementación. Para instalar Helm en el host de salto de implementación, siga las instrucciones "instrucciones de instalación" en la documentación oficial de Helm.

Establecer la clase de almacenamiento predeterminada de Kubernetes

Antes de implementar Airflow, debe designar una StorageClass predeterminada dentro de su clúster de Kubernetes. El proceso de implementación de Airflow intenta aprovisionar nuevos volúmenes persistentes utilizando la clase de almacenamiento predeterminada. Si no se designa ninguna StorageClass como StorageClass predeterminada, la implementación falla. Para designar una StorageClass predeterminada dentro de su clúster, siga las instrucciones que se describen en la"Implementación de Kubeflow" sección. Si ya ha designado una StorageClass predeterminada dentro de su clúster, puede omitir este paso.

Utilice Helm para implementar el flujo de aire

Para implementar Airflow en su clúster de Kubernetes usando Helm, realice las siguientes tareas desde el host de salto de implementación:

1. Implemente Airflow usando Helm siguiendo las instrucciones "instrucciones de implementación" para el gráfico oficial de flujo de aire en Artifact Hub. Los comandos de ejemplo que siguen muestran la implementación de Airflow usando Helm. Modificar, agregar y/o eliminar valores en el `custom-values.yaml` archivo según sea necesario dependiendo de su entorno y la configuración deseada.

```
$ 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
```

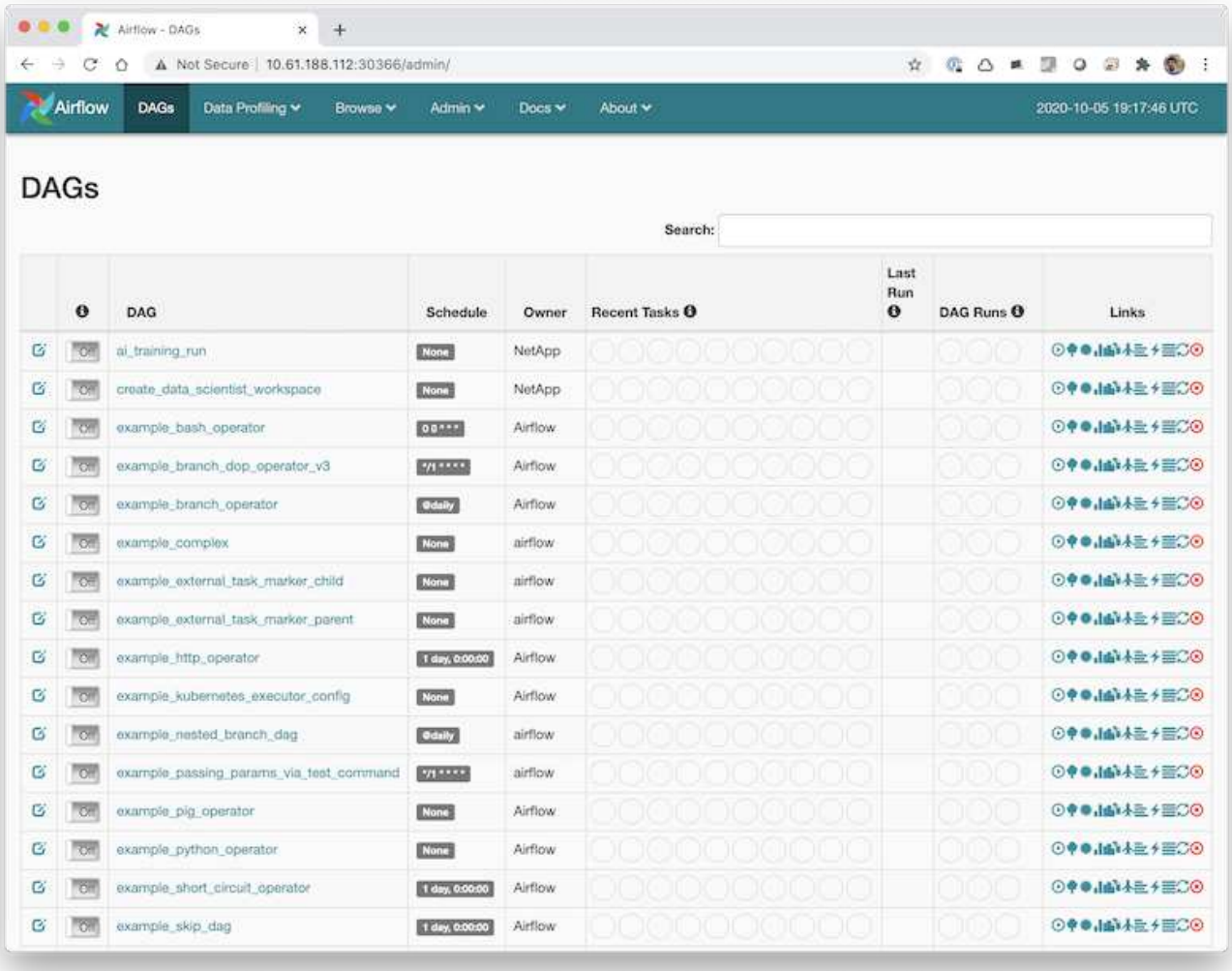
2. Confirme que todos los pods Airflow estén en funcionamiento. Es posible que pasen algunos minutos hasta que se inicien todos los pods.

```
$ 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. Obtenga la URL del servicio web Airflow siguiendo las instrucciones que se imprimieron en la consola cuando implementó Airflow usando Helm en el paso 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. Confirme que puede acceder al servicio web Airflow.



Utilice el kit de herramientas NetApp DataOps con Airflow

El "Kit de herramientas de NetApp DataOps para Kubernetes" Se puede utilizar junto con Airflow. El uso de NetApp DataOps Toolkit con Airflow le permite incorporar operaciones de administración de datos de NetApp , como la creación de instantáneas y clones, en flujos de trabajo automatizados orquestados por Airflow.

Consulte la "Ejemplos de flujo de aire" sección dentro del repositorio de GitHub de NetApp DataOps Toolkit para obtener detalles sobre el uso del kit de herramientas con Airflow.

JupyterHub

Implementación de JupyterHub

Esta sección describe las tareas que debe completar para implementar JupyterHub en su clúster de Kubernetes.

Es posible implementar JupyterHub en plataformas distintas a Kubernetes. La implementación de JupyterHub en plataformas distintas a Kubernetes está fuera del alcance de esta solución.

Prerrequisitos

Antes de realizar el ejercicio de implementación que se describe en esta sección, asumimos que ya ha realizado las siguientes tareas:

1. Ya tienes un clúster de Kubernetes en funcionamiento.
2. Ya ha instalado y configurado NetApp Trident en su clúster de Kubernetes. Para obtener más detalles sobre Trident, consulte la"Documentación de Trident" .

Instalar Helm

JupyterHub se implementa utilizando Helm, un administrador de paquetes popular para Kubernetes. Antes de implementar JupyterHub, debe instalar Helm en su nodo de control de Kubernetes. Para instalar Helm, siga las instrucciones "instrucciones de instalación" en la documentación oficial de Helm.

Establecer la clase de almacenamiento predeterminada de Kubernetes

Antes de implementar JupyterHub, debe designar una StorageClass predeterminada dentro de su clúster de Kubernetes. Para designar una StorageClass predeterminada dentro de su clúster, siga las instrucciones que se describen en la"Implementación de Kubeflow" sección. Si ya ha designado una StorageClass predeterminada dentro de su clúster, puede omitir este paso.

Implementar JupyterHub

Después de completar los pasos anteriores, ahora está listo para implementar JupyterHub. La implementación de JupyterHub requiere los siguientes pasos:

Configurar la implementación de JupyterHub

Antes de la implementación, es una buena práctica optimizar la implementación de JupyterHub para su entorno respectivo. Puede crear un archivo **config.yaml** y utilizarlo durante la implementación mediante el gráfico Helm.

Se puede encontrar un archivo **config.yaml** de ejemplo en https://github.com/jupyterhub/zero-to-jupyterhub-k8s/blob/HEAD/jupyterhub/values.yaml

En este archivo config.yaml, puede establecer el parámetro **(singleuser.storage.dynamic.storageClass)** para NetApp Trident StorageClass. Esta es la clase de almacenamiento que se utilizará para aprovisionar los volúmenes para espacios de trabajo de usuarios individuales.

Agregar volúmenes compartidos

Si desea utilizar un volumen compartido para todos los usuarios de JupyterHub, puede ajustar su **config.yaml** en consecuencia. Por ejemplo, si tiene un PersistentVolumeClaim compartido llamado jupyterhub-shared-volume, podría montarlo como /home/shared en todos los pods de usuario de la siguiente manera:

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

Este es un paso opcional, puedes ajustar estos parámetros según tus necesidades.

Implementar JupyterHub con Helm Chart

Haga que Helm conozca el repositorio de gráficos de Helm de JupyterHub.

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

Esto debería mostrar un resultado como el siguiente:

```
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!□
```

Ahora instale el gráfico configurado por su config.yaml ejecutando este comando desde el directorio que contiene su config.yaml:

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

En este ejemplo:

<helm-release-name> se establece en my-jupyterhub, que será el nombre de su versión de JupyterHub. <k8s-namespace> está configurado en my-namespace, que es el espacio de nombres donde desea instalar JupyterHub. El indicador --create-namespace se utiliza para crear el espacio de nombres si aún no existe. El indicador --values especifica el archivo config.yaml que contiene las opciones de configuración deseadas.

Comprobar implementación

Mientras se ejecuta el paso 2, puedes ver los pods que se crean desde el siguiente comando:

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

Espere a que el concentrador y el módulo proxy entren en el estado de ejecución.

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

Acceder a JupyterHub

Encuentra la IP que podemos usar para acceder a JupyterHub. Ejecute el siguiente comando hasta que la IP EXTERNA del servicio proxy público esté disponible como en la salida del ejemplo.

> Usamos el servicio NodePort en nuestro archivo config.yaml, que puedes ajustar a tu entorno según tu configuración (por ejemplo, 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
```

Para utilizar JupyterHub, ingrese la IP externa del servicio proxy público en un navegador.

Utilice el kit de herramientas NetApp DataOps con JupyterHub

El "Kit de herramientas de NetApp DataOps para Kubernetes" se puede utilizar junto con JupyterHub. El uso del kit de herramientas NetApp DataOps con JupyterHub permite a los usuarios finales crear instantáneas de volumen para realizar copias de seguridad del espacio de trabajo o para la trazabilidad del conjunto de datos al modelo directamente desde un Jupyter Notebook.

Configuración inicial

Antes de poder usar DataOps Toolkit con JupyterHub, debe otorgar los permisos adecuados a la cuenta de servicio de Kubernetes que JupyterHub asigna a los pods de Jupyter Notebook Server de usuarios individuales. JupyterHub utiliza la cuenta de servicio especificada por el `singleuser.serviceAccountName` variable en su archivo de configuración del gráfico Helm de JupyterHub.

Crear un rol de clúster para el kit de herramientas DataOps

Primero, cree una función de clúster denominada "netapp-dataops" que tenga los permisos de API de Kubernetes necesarios para crear instantáneas de volumen.

```
$ 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
```

Asignar rol de clúster a la cuenta de servicio del servidor Notebook

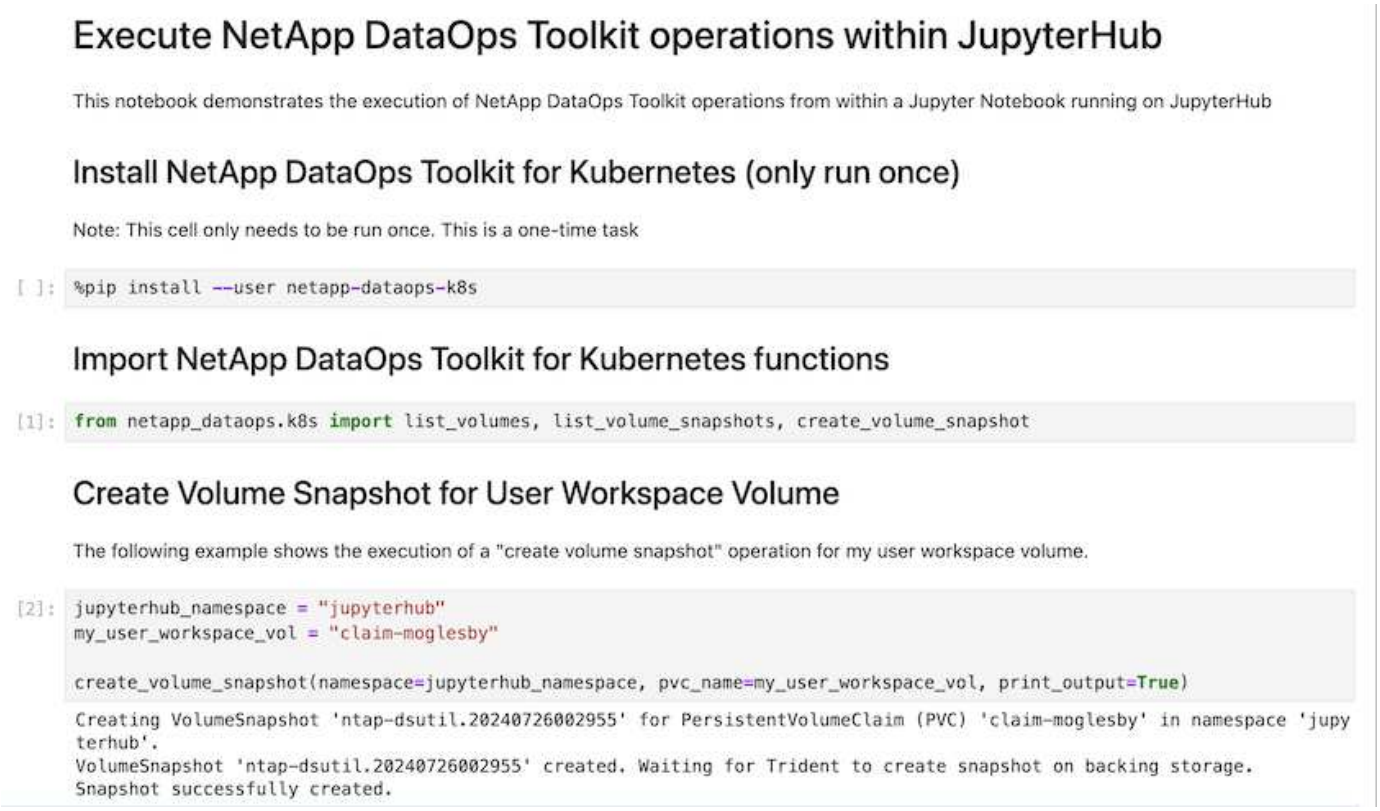
Cree un enlace de rol que asigne el rol de clúster 'netapp-dataops-snapshots' a la cuenta de servicio adecuada en el espacio de nombres apropiado. Por ejemplo, si instaló JupyterHub en el espacio de nombres 'jupyterhub' y especificó la cuenta de servicio 'predeterminada' a través de `singleuser.serviceAccountName` variable, asignaría la función de clúster 'netapp-dataops-snapshots' a la cuenta de servicio 'predeterminada' en el espacio de nombres 'jupyterhub' como se muestra en el siguiente ejemplo.

```
$ 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
```

Crear instantáneas de volumen dentro de Jupyter Notebook

Ahora, los usuarios de JupyterHub pueden usar NetApp DataOps Toolkit para crear instantáneas de volumen directamente desde un Jupyter Notebook como se muestra en el siguiente ejemplo.



Ingerir datos en JupyterHub con NetApp SnapMirror

NetApp SnapMirror es una tecnología de replicación que le permite replicar datos entre sistemas de almacenamiento NetApp . SnapMirror se puede utilizar para ingerir datos de entornos remotos en JupyterHub.

Ejemplo de flujo de trabajo y demostración

Referirse a"Esta publicación del blog de Tech ONTAP" para obtener un ejemplo detallado de flujo de trabajo y una demostración del uso de NetApp SnapMirror para ingerir datos en JupyterHub.

Flujo de ml

Implementación de MLflow

Esta sección describe las tareas que debe completar para implementar MLflow en su clúster de Kubernetes.

> Es posible implementar MLflow en plataformas distintas a Kubernetes. La implementación de MLflow en plataformas distintas a Kubernetes está fuera del alcance de esta solución.

Prerrequisitos

Antes de realizar el ejercicio de implementación que se describe en esta sección, asumimos que ya ha realizado las siguientes tareas:

1. Ya tienes un clúster de Kubernetes en funcionamiento.
2. Ya ha instalado y configurado NetApp Trident en su clúster de Kubernetes. Para obtener más detalles sobre Trident, consulte la"Documentación de Trident" .

Instalar Helm

MLflow se implementa utilizando Helm, un administrador de paquetes popular para Kubernetes. Antes de implementar MLflow, debe instalar Helm en su nodo de control de Kubernetes. Para instalar Helm, siga las instrucciones "instrucciones de instalación" en la documentación oficial de Helm.

Establecer la clase de almacenamiento predeterminada de Kubernetes

Antes de implementar MLflow, debe designar una StorageClass predeterminada dentro de su clúster de Kubernetes. Para designar una StorageClass predeterminada dentro de su clúster, siga las instrucciones que se describen en la"Implementación de Kubeflow" sección. Si ya ha designado una StorageClass predeterminada dentro de su clúster, puede omitir este paso.

Implementar MLflow

Una vez que se cumplan los requisitos previos, puede comenzar con la implementación de MLflow utilizando el gráfico de Helm.

Configurar la implementación del gráfico Helm de MLflow.

Antes de implementar MLflow utilizando el gráfico Helm, podemos configurar la implementación para usar la

clase de almacenamiento NetApp Trident y cambiar otros parámetros para adaptarlos a nuestras necesidades utilizando un archivo **config.yaml**. Puedes encontrar un ejemplo de archivo **config.yaml** en: https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml

Puede configurar la clase de almacenamiento Trident en el parámetro **global.defaultStorageClass** en el archivo config.yaml (por ejemplo, clase de almacenamiento: "ontap-flexvol").

Instalación del cuadro de mandos Helm

El gráfico Helm se puede instalar con el archivo **config.yaml** personalizado para MLflow usando el siguiente comando:

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

El comando implementa MLflow en el clúster de Kubernetes en la configuración personalizada a través del archivo **config.yaml** proporcionado. MLflow se implementa en el espacio de nombres indicado y se proporciona un nombre de versión aleatorio a través de Kubernetes para la versión.

Comprobar implementación

Una vez finalizada la implementación del gráfico de Helm, puedes comprobar si el servicio es accesible mediante:

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

Reemplace **jupyterhub** con el espacio de nombres que utilizó durante la implementación.

Debería ver los siguientes servicios:

```
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
```

Editamos el archivo config.yaml para usar el servicio NodePort para acceder a MLflow en el puerto 30002.

Acceso a MLflow

Una vez que todos los servicios relacionados con MLflow estén en funcionamiento, puede acceder a él utilizando la dirección IP de NodePort o LoadBalancer indicada (por ejemplo, `http://10.61.181.109:30002`)

Trazabilidad del conjunto de datos al modelo con NetApp y MLflow

El "Kit de herramientas de NetApp DataOps para Kubernetes" se puede utilizar junto con las capacidades de seguimiento de experimentos de MLflow para implementar la trazabilidad del conjunto de datos al modelo o del espacio de trabajo al modelo.

Para implementar la trazabilidad del conjunto de datos al modelo o del espacio de trabajo al modelo, simplemente cree una instantánea de su conjunto de datos o volumen del espacio de trabajo utilizando el kit de herramientas DataOps como parte de su ejecución de entrenamiento, como se muestra en el siguiente fragmento de código de ejemplo. Este código guardará el nombre del volumen de datos y el nombre de la instantánea como etiquetas asociadas con la ejecución de entrenamiento específica que está registrando en su servidor de seguimiento de experimentos de 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

Implementación de Kubeflow

Esta sección describe las tareas que debe completar para implementar Kubeflow en su clúster de Kubernetes.

Prerrequisitos

Antes de realizar el ejercicio de implementación que se describe en esta sección, asumimos que ya ha realizado las siguientes tareas:

1. Ya tienes un clúster de Kubernetes en funcionamiento y estás ejecutando una versión de Kubernetes compatible con la versión de Kubeflow que deseas implementar. Para obtener una lista de las versiones compatibles de Kubernetes, consulte las dependencias para su versión de Kubeflow en"documentación oficial de Kubeflow" .
2. Ya ha instalado y configurado NetApp Trident en su clúster de Kubernetes. Para obtener más detalles sobre Trident, consulte la"Documentación de Trident" .

Establecer la clase de almacenamiento predeterminada de Kubernetes

Antes de implementar Kubeflow, recomendamos designar una StorageClass predeterminada dentro de su clúster de Kubernetes. El proceso de implementación de Kubeflow puede intentar aprovisionar nuevos volúmenes persistentes utilizando la clase de almacenamiento predeterminada. Si no se designa ninguna StorageClass como StorageClass predeterminada, la implementación puede fallar. Para designar una StorageClass predeterminada dentro de su clúster, realice la siguiente tarea desde el host de salto de implementación. Si ya ha designado una StorageClass predeterminada dentro de su clúster, puede omitir este paso.

1. Designe una de sus StorageClasses existentes como la StorageClass predeterminada. Los comandos de ejemplo que siguen muestran la designación de una StorageClass denominada `ontap-ai-flexvols-retain` como la clase de almacenamiento predeterminada.

> El `ontap-nas-flexgroup` El tipo Trident Backend tiene un tamaño mínimo de PVC que es bastante grande. De forma predeterminada, Kubeflow intenta aprovisionar PVC que tengan solo unos pocos GB de tamaño. Por lo tanto, no debe designar una StorageClass que utilice el `ontap-nas-flexgroup` Tipo de backend como StorageClass predeterminado para los fines de implementación 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
```

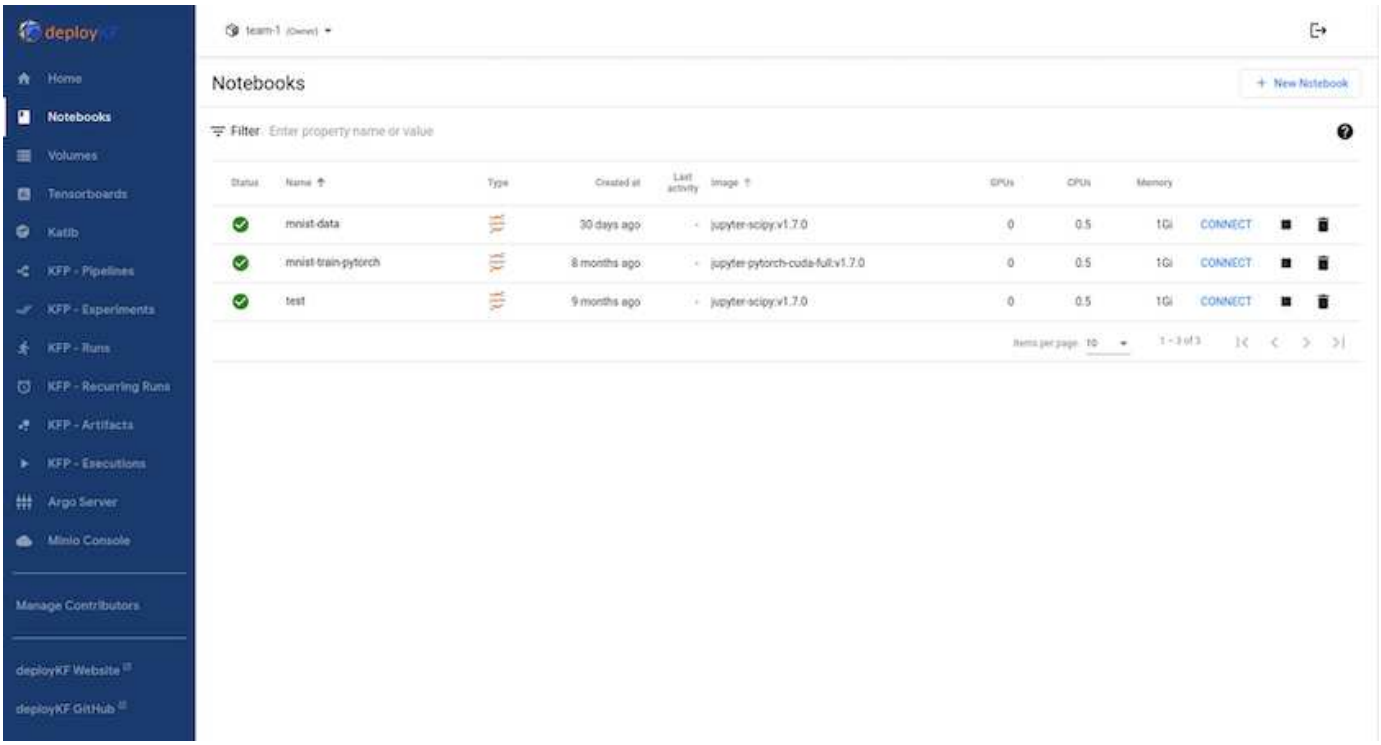
Opciones de implementación de Kubeflow

Hay muchas opciones diferentes para implementar Kubeflow. Consulte la"documentación oficial de Kubeflow" para obtener una lista de opciones de implementación y elegir la opción que mejor se adapte a sus necesidades.

Para fines de validación, implementamos Kubeflow 1.7 usando"desplegarKF" 0.1.1.

Proporcionar un espacio de trabajo de Jupyter Notebook para uso de desarrolladores o científicos de datos

Kubeflow es capaz de aprovisionar rápidamente nuevos servidores Jupyter Notebook para que actúen como espacios de trabajo de científicos de datos. Para obtener más información sobre Jupyter Notebooks dentro del contexto de Kubeflow, consulte la "documentación oficial de Kubeflow" .



Utilice el kit de herramientas NetApp DataOps con Kubeflow

El "Kit de herramientas de ciencia de datos de NetApp para Kubernetes" se puede utilizar junto con Kubeflow. El uso del kit de herramientas de ciencia de datos de NetApp con Kubeflow proporciona los siguientes beneficios:

- Los científicos de datos pueden realizar operaciones avanzadas de gestión de datos de NetApp , como la creación de instantáneas y clones, directamente desde un Jupyter Notebook.
- Las operaciones avanzadas de gestión de datos de NetApp , como la creación de instantáneas y clones, se pueden incorporar a flujos de trabajo automatizados mediante el marco Kubeflow Pipelines.

Consulte la "Ejemplos de Kubeflow" sección dentro del repositorio de GitHub de NetApp Data Science Toolkit para obtener detalles sobre el uso del kit de herramientas con Kubeflow.

Ejemplo de flujo de trabajo: Entrenamiento de un modelo de reconocimiento de imágenes mediante Kubeflow y el kit de herramientas DataOps de NetApp

Esta sección describe los pasos necesarios para entrenar e implementar una red neuronal para reconocimiento de imágenes utilizando Kubeflow y NetApp DataOps Toolkit. Esto pretende servir como ejemplo para mostrar un trabajo de capacitación que incorpora almacenamiento NetApp .

Prerrequisitos

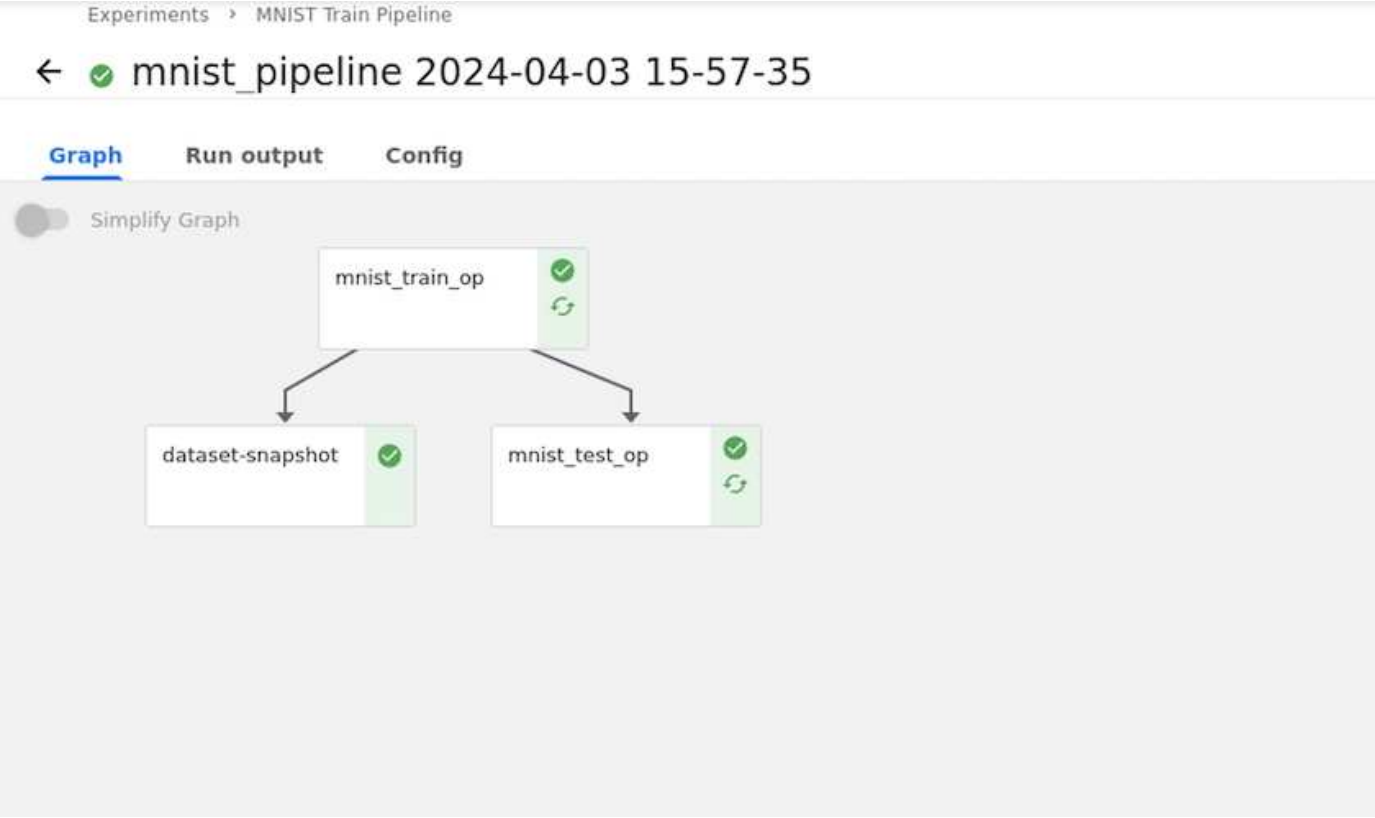
Cree un Dockerfile con las configuraciones necesarias para utilizar en los pasos de entrenamiento y prueba dentro del pipeline de Kubeflow. Aquí hay un ejemplo de un Dockerfile:

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

Dependiendo de sus requisitos, instale todas las bibliotecas y paquetes necesarios para ejecutar el programa. Antes de entrenar el modelo de aprendizaje automático, se supone que ya tienes una implementación de Kubeflow en funcionamiento.

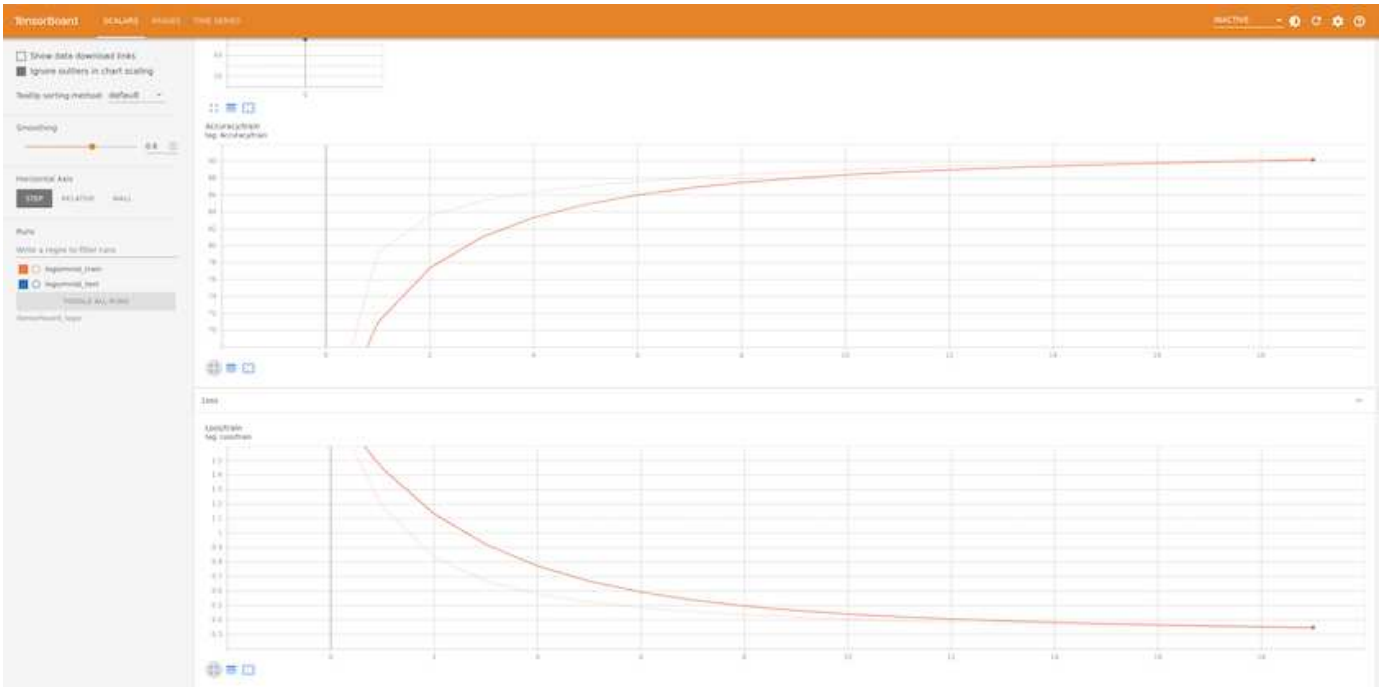
Entrene una red neuronal pequeña con datos MNIST usando PyTorch y Kubeflow Pipelines

Utilizamos el ejemplo de una pequeña red neuronal entrenada con datos MNIST. El conjunto de datos MNIST consta de imágenes escritas a mano de dígitos del 0 al 9. Las imágenes tienen un tamaño de 28x28 píxeles. El conjunto de datos se divide en 60.000 imágenes de trenes y 10.000 imágenes de validación. La red neuronal utilizada para este experimento es una red de propagación hacia adelante de dos capas. El entrenamiento se ejecuta utilizando Kubeflow Pipelines. Consulte la documentación "aquí" Para más información. Nuestra canalización de Kubeflow incorpora la imagen de Docker de la sección Requisitos previos.



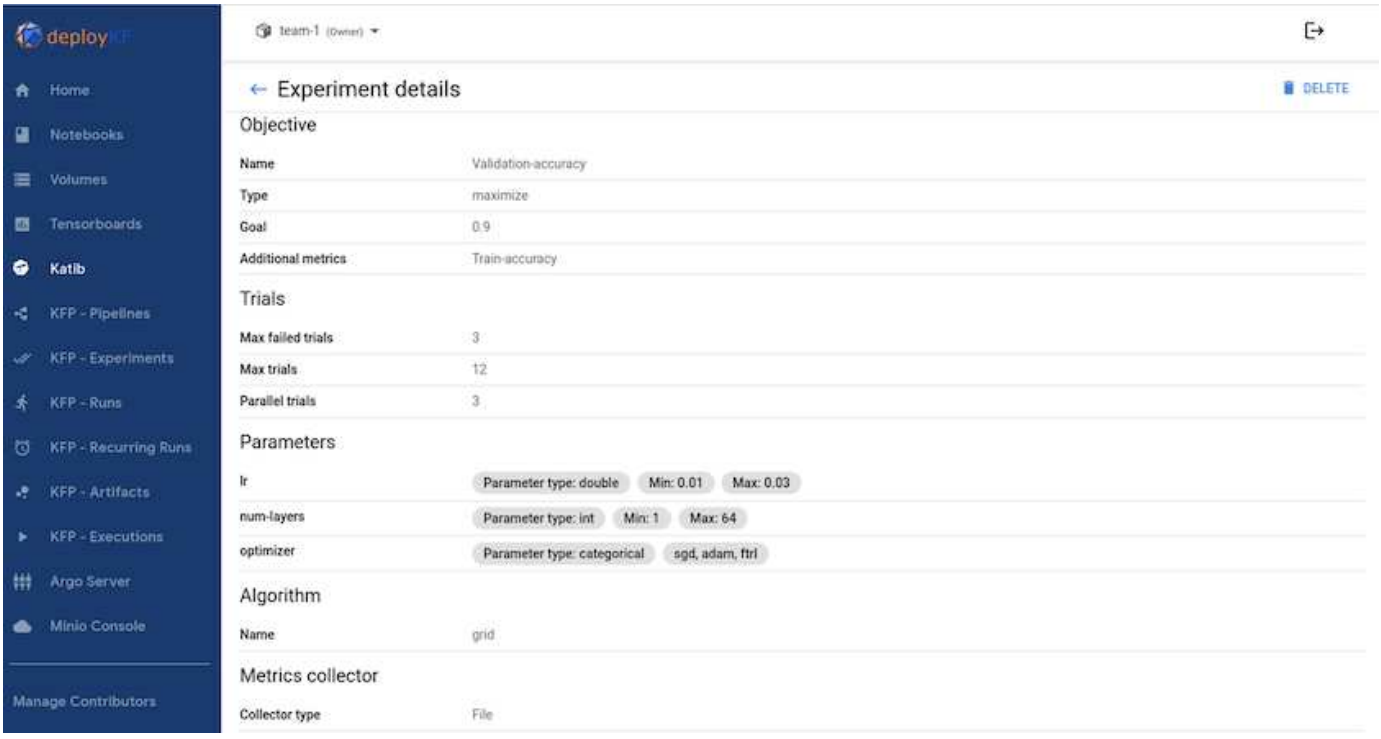
Visualizar resultados con Tensorboard

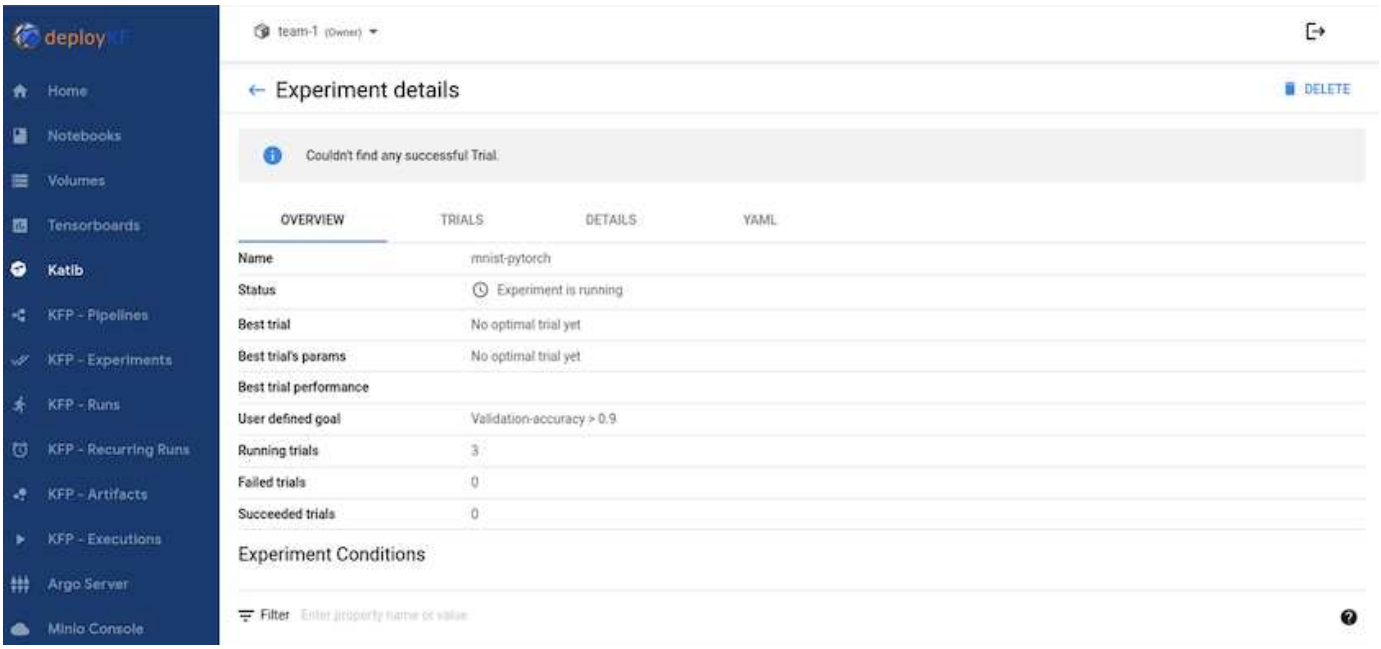
Una vez entrenado el modelo, podemos visualizar los resultados utilizando Tensorboard. "Tablero tensor" Está disponible como una función en el panel de Kubeflow. Puede crear un tensorboard personalizado para su trabajo. A continuación se muestra un ejemplo de la gráfica de la precisión del entrenamiento frente al número de épocas y de la pérdida de entrenamiento frente al número de épocas.



Experimente con hiperparámetros usando Katib

"Katib"es una herramienta dentro de Kubeflow que se puede utilizar para experimentar con los hiperparámetros del modelo. Para crear un experimento, primero defina una métrica/objetivo deseado. Generalmente esta es la precisión de la prueba. Una vez definida la métrica, elija los hiperparámetros con los que desea experimentar (optimizador/tasa de aprendizaje/número de capas). Katib realiza un barrido de hiperparámetros con los valores definidos por el usuario para encontrar la mejor combinación de parámetros que satisfagan la métrica deseada. Puede definir estos parámetros en cada sección de la interfaz de usuario. Alternativamente, podría definir un archivo **YAML** con las especificaciones necesarias. A continuación se muestra una ilustración de un experimento de Katib:





Utilice instantáneas de NetApp para guardar datos para trazabilidad

Durante el entrenamiento del modelo, es posible que queramos guardar una instantánea del conjunto de datos de entrenamiento para facilitar la trazabilidad. Para ello, podemos agregar un paso de instantánea a la canalización como se muestra a continuación. Para crear la instantánea, podemos utilizar el "Kit de herramientas de NetApp DataOps para 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)
```

Consulte la "Ejemplo de NetApp DataOps Toolkit para Kubeflow" Para más información.

Ejemplo de operaciones Trident

Esta sección incluye ejemplos de varias operaciones que quizás desee realizar con Trident.

Importar un volumen existente

Si hay volúmenes existentes en su sistema/plataforma de almacenamiento NetApp que desea montar en contenedores dentro de su clúster de Kubernetes, pero que no están vinculados a PVC en el clúster, entonces debe importar estos volúmenes. Puede utilizar la funcionalidad de importación de volumen de Trident para

importar estos volúmenes.

Los comandos de ejemplo que siguen muestran la importación de un volumen llamado `pb_fg_all` . Para obtener más información sobre los PVC, consulte la "documentación oficial de Kubernetes" . Para obtener más información sobre la funcionalidad de importación de volumen, consulte la "Documentación de Trident" .

Un `accessModes` valor de `ReadOnlyMany` se especifica en los archivos de especificaciones de PVC de ejemplo. Para obtener más información sobre la `accessMode` campo, ver el "documentación oficial 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
```

Proporcionar un nuevo volumen

Puede utilizar Trident para aprovisionar un nuevo volumen en su sistema o plataforma de almacenamiento NetApp .

Aprovisionar un nuevo volumen usando kubectl

Los siguientes comandos de ejemplo muestran el aprovisionamiento de un nuevo FlexVol volume mediante kubectl.

Un `accessModes` valor de `ReadWriteMany` se especifica en el siguiente archivo de definición de PVC de ejemplo. Para obtener más información sobre la `accessMode` campo, ver el "documentación oficial 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
```

Aprovisionar un nuevo volumen mediante el kit de herramientas NetApp DataOps

También puede utilizar NetApp DataOps Toolkit para Kubernetes para aprovisionar un nuevo volumen en su sistema o plataforma de almacenamiento NetApp . El kit de herramientas NetApp DataOps para Kubernetes utiliza Trident para aprovisionar volúmenes, pero simplifica el proceso para el usuario. Consulte la"documentación" Para más detalles.

Ejemplos de trabajos de alto rendimiento para implementaciones de AIPod

Ejecutar una carga de trabajo de IA de un solo nodo

Para ejecutar un trabajo de IA y ML de un solo nodo en su clúster de Kubernetes, realice las siguientes tareas desde el host de salto de implementación. Con Trident, puede hacer que un volumen de datos, que potencialmente contenga petabytes de datos, sea accesible a una carga de trabajo de Kubernetes de manera rápida y sencilla. Para que dicho volumen de datos sea accesible desde un pod de Kubernetes, simplemente especifique una PVC en la definición del pod.

Esta sección asume que ya ha contenedorizado (en el formato de contenedor Docker) la carga de trabajo de IA y ML específica que está intentando ejecutar en su clúster de Kubernetes.

1. Los siguientes comandos de ejemplo muestran la creación de un trabajo de Kubernetes para una carga de trabajo de referencia de TensorFlow que utiliza el conjunto de datos ImageNet. Para obtener más información sobre el conjunto de datos ImageNet, consulte "Sitio web de ImageNet" .

   Este trabajo de ejemplo solicita ocho GPU y, por lo tanto, puede ejecutarse en un solo nodo de trabajo de GPU que tenga ocho o más GPU. Este trabajo de ejemplo podría enviarse en un clúster en el que no hay un nodo de trabajo con ocho o más GPU presente o que actualmente está ocupado con otra carga de trabajo. Si es así, el trabajo permanece en estado pendiente hasta que dicho nodo de trabajo esté disponible.

   Además, para maximizar el ancho de banda de almacenamiento, el volumen que contiene los datos de entrenamiento necesarios se monta dos veces dentro del pod que crea este trabajo. Otro volumen también está montado en la cápsula. Este segundo volumen se utilizará para almacenar resultados y métricas. Estos volúmenes se referencian en la definición del trabajo mediante los nombres de las PVC. Para obtener más información sobre los trabajos de Kubernetes, consulte "documentación oficial de Kubernetes" .

   Un `emptyDir` volumen con un `medium` valor de `Memory` está montado en `/dev/shm` en el pod que crea este trabajo de ejemplo. El tamaño predeterminado de la `/dev/shm` El volumen virtual que crea automáticamente el entorno de ejecución del contenedor Docker a veces puede ser insuficiente para las necesidades de TensorFlow. Montaje de un `emptyDir` El volumen como en el siguiente ejemplo proporciona un volumen suficientemente grande. `/dev/shm` volumen virtual. Para obtener más información sobre `emptyDir` volúmenes, véase el "documentación oficial de Kubernetes" .

   Al contenedor único que se especifica en esta definición de trabajo de ejemplo se le asigna un `securityContext > privileged` valor de `true` . Este valor significa que el contenedor efectivamente tiene acceso root en el host. Esta anotación se utiliza en este caso porque la carga de trabajo específica que se está ejecutando requiere acceso root. Específicamente, una operación de limpieza de caché que realiza la carga de trabajo requiere acceso root. Sea esto o no `privileged: true` La anotación es necesaria dependiendo de los requisitos de la carga de trabajo específica que esté ejecutando.

```
$ 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. Confirme que el trabajo que creó en el paso 1 se esté ejecutando correctamente. El siguiente comando de ejemplo confirma que se creó un solo pod para el trabajo, como se especifica en la definición del trabajo, y que este pod se está ejecutando actualmente en uno de los nodos de trabajo de la 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. Confirme que el trabajo que creó en el paso 1 se complete exitosamente. Los siguientes comandos de ejemplo confirman que el trabajo se completó correctamente.

```
$ 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. **Opcional:** Limpiar los artefactos del trabajo. Los siguientes comandos de ejemplo muestran la eliminación del objeto de trabajo que se creó en el paso 1.

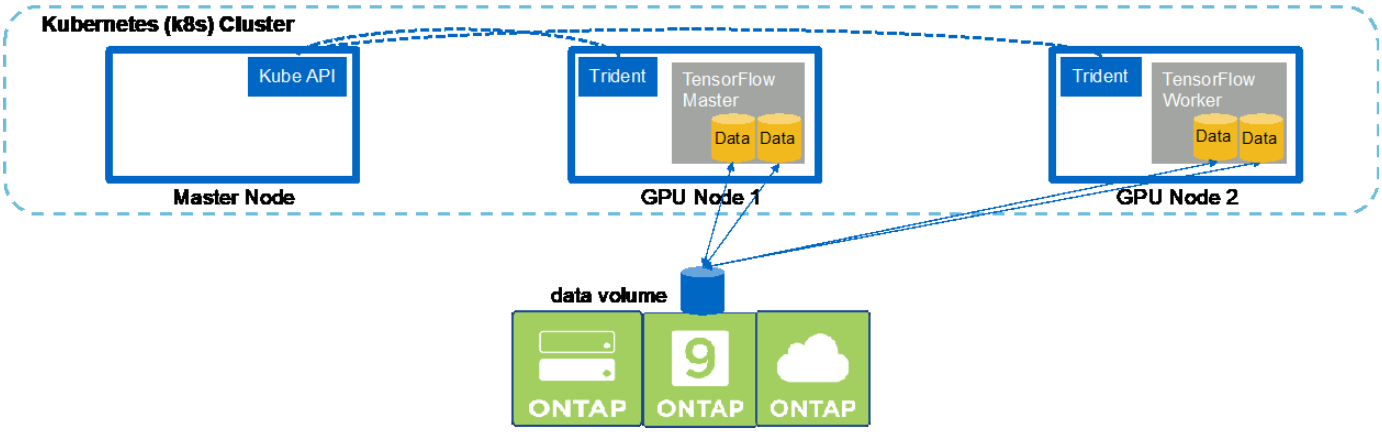
   Cuando elimina el objeto de trabajo, Kubernetes elimina automáticamente todos los pods asociados.

```
$ 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.
```

Ejecutar una carga de trabajo de IA distribuida sincrónica

Para ejecutar un trabajo de IA y ML sincrónico de múltiples nodos en su clúster de Kubernetes, realice las siguientes tareas en el host de salto de implementación. Este proceso le permite aprovechar los datos almacenados en un volumen de NetApp y utilizar más GPU de las que un solo nodo de trabajo puede proporcionar. Vea la siguiente figura para ver una representación de un trabajo de IA distribuido sincrónico.

Los trabajos distribuidos sincrónicos pueden ayudar a aumentar el rendimiento y la precisión del entrenamiento en comparación con los trabajos distribuidos asincrónicos. Una discusión de los pros y contras de los trabajos sincrónicos versus los trabajos asincrónicos está fuera del alcance de este documento.



1. Los siguientes comandos de ejemplo muestran la creación de un trabajador que participa en la ejecución distribuida sincrónica del mismo trabajo de referencia de TensorFlow que se ejecutó en un solo nodo en el ejemplo de la sección"Ejecutar una carga de trabajo de IA de un solo nodo" . En este ejemplo específico, solo se implementa un único trabajador porque el trabajo se ejecuta en dos nodos de trabajo.

Esta implementación de trabajador de ejemplo solicita ocho GPU y, por lo tanto, puede ejecutarse en un solo nodo de trabajador de GPU que tenga ocho o más GPU. Si sus nodos de trabajo de GPU cuentan con más de ocho GPU, para maximizar el rendimiento, es posible que desee aumentar este número para que sea igual a la cantidad de GPU que cuentan sus nodos de trabajo. Para obtener más información sobre las implementaciones de Kubernetes, consulte "documentación oficial de Kubernetes" .

En este ejemplo se crea una implementación de Kubernetes porque este trabajador en contenedor específico nunca se completaría por sí solo. Por lo tanto, no tiene sentido implementarlo utilizando la construcción del trabajo de Kubernetes. Si su trabajador está diseñado o escrito para completarse por sí solo, entonces podría tener sentido utilizar la construcción del trabajo para implementar su trabajador.

Al pod que se especifica en esta especificación de implementación de ejemplo se le asigna un `hostNetwork` valor de `true` . Este valor significa que el pod utiliza la pila de red del nodo de trabajo host en lugar de la pila de red virtual que Kubernetes normalmente crea para cada pod. Esta anotación se utiliza en este caso porque la carga de trabajo específica depende de Open MPI, NCCL y Horovod para ejecutarla de manera distribuida sincrónica. Por lo tanto, requiere acceso a la pila de red del host. Una discusión sobre Open MPI, NCCL y Horovod está fuera del alcance de este documento. Sea esto o no `hostNetwork: true` La anotación es necesaria dependiendo de los requisitos de la carga de trabajo específica que esté ejecutando. Para obtener más información sobre la `hostNetwork` campo, ver el "documentación oficial 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. Confirme que la implementación del trabajador que creó en el paso 1 se inició correctamente. Los siguientes comandos de ejemplo confirman que se creó un solo pod de trabajo para la implementación, como se indica en la definición de implementación, y que este pod se está ejecutando actualmente en uno de los nodos de trabajo de la 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. Cree un trabajo de Kubernetes para un maestro que inicie, participe y realice un seguimiento de la ejecución del trabajo multinodo sincrónico. Los siguientes comandos de ejemplo crean un maestro que inicia, participa y rastrea la ejecución distribuida sincrónica del mismo trabajo de referencia de TensorFlow

que se ejecutó en un solo nodo en el ejemplo de la sección"Ejecutar una carga de trabajo de IA de un solo nodo" .

Este trabajo maestro de ejemplo solicita ocho GPU y, por lo tanto, puede ejecutarse en un solo nodo de trabajo de GPU que tenga ocho o más GPU. Si sus nodos de trabajo de GPU cuentan con más de ocho GPU, para maximizar el rendimiento, es posible que desee aumentar este número para que sea igual a la cantidad de GPU que cuentan sus nodos de trabajo.

Al pod maestro que se especifica en esta definición de trabajo de ejemplo se le asigna un `hostNetwork` valor de `true` , justo cuando se le dio un puesto al trabajador `hostNetwork` valor de `true` en el paso 1. Consulte el paso 1 para obtener detalles sobre por qué es necesario este valor.

```
$ 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. Confirme que el trabajo maestro que creó en el paso 3 se esté ejecutando correctamente. El siguiente comando de ejemplo confirma que se creó un solo pod maestro para el trabajo, como se indica en la definición del trabajo, y que este pod se está ejecutando actualmente en uno de los nodos de trabajo de la GPU. También debería ver que el pod de trabajo que vio originalmente en el paso 1 todavía está ejecutándose y que los pods maestro y de trabajo se están ejecutando en nodos diferentes.

```
$ 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. Confirme que el trabajo maestro que creó en el paso 3 se complete exitosamente. Los siguientes comandos de ejemplo confirman que el trabajo se completó correctamente.

```
$ kubectl get jobs
NAME                                        COMPLETIONS   DURATION   AGE
netapp-tensorflow-multi-imagenet-master   1/1           5m50s      9m18s
$ kubectl get pods
NAME                                                              READY
STATUS      RESTARTS   AGE
netapp-tensorflow-multi-imagenet-master-ppwwj                     0/1
Completed   0          9m38s
netapp-tensorflow-multi-imagenet-worker-654fc7f486-v6725   1/1
Running     0          35m
```

```
$ kubectl logs netapp-tensorflow-multi-imagenet-master-ppwwj
[10.61.218.152:00008] WARNING: local probe returned unhandled
shell:unknown assuming bash
rm: cannot remove '/lib': Is a directory
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.154:00033] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 702
[10.61.218.152:00008] PMIX ERROR: NO-PERMISSIONS in file gds_dstore.c at
line 711
Total images/sec = 12881.33875
================ Clean Cache !!! ==================
mpirun -allow-run-as-root -np 2 -H 10.61.218.152:1,10.61.218.154:1 -mca
pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" bash -c 'sync; echo 1 >
/proc/sys/vm/drop_caches'
========================================
mpirun -allow-run-as-root -np 16 -H 10.61.218.152:8,10.61.218.154:8
-bind-to none -map-by slot -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH
-mca pml ob1 -mca btl ^openib -mca btl_tcp_if_include enp1s0f0 -x
NCCL_IB_HCA=mlx5 -x NCCL_NET_GDR_READ=1 -x NCCL_IB_SL=3 -x
NCCL_IB_GID_INDEX=3 -x
NCCL_SOCKET_IFNAME=enp5s0.3091,enp12s0.3092,enp132s0.3093,enp139s0.3094
-x NCCL_IB_CUDA_SUPPORT=1 -mca orte_base_help_aggregate 0 -mca
plm_rsh_agent ssh -mca plm_rsh_args "-p 22122" python
/netapp/tensorflow/benchmarks_190205/scripts/tf_cnn_benchmarks/tf_cnn_be
nchmarks.py --model=resnet50 --batch_size=256 --device=gpu
--force_gpu_compatible=True --num_intra_threads=1 --num_inter_threads=48
--variable_update=horovod --batch_group_size=20 --num_batches=500
--nodistortions --num_gpus=1 --data_format=NCHW --use_fp16=True
--use_tf_layers=False --data_name=imagenet --use_datasets=True
--data_dir=/mnt/mount_0/dataset/imagenet
--datasets_parallel_interleave_cycle_length=10
--datasets_sloppy_parallel_interleave=False --num_mounts=2
--mount_prefix=/mnt/mount_%d --datasets_prefetch_buffer_size=2000 --
datasets_use_prefetch=True --datasets_num_private_threads=4
--horovod_device=gpu >
/tmp/20190814_161609_tensorflow_horovod_rdma_resnet50_gpu_16_256_b500_im
agenet_nodistort_fp16_r10_m2_nockpt.txt 2>&1
```

6. Elimina la implementación del trabajador cuando ya no la necesites. Los siguientes comandos de ejemplo muestran la eliminación del objeto de implementación de trabajador que se creó en el paso 1.

   Cuando elimina el objeto de implementación del trabajador, Kubernetes elimina automáticamente todos los

pods de trabajador asociados.

```
$ 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. **Opcional:** Limpiar los artefactos del trabajo maestro. Los siguientes comandos de ejemplo muestran la eliminación del objeto de trabajo maestro que se creó en el paso 3.

   Cuando elimina el objeto de trabajo maestro, Kubernetes elimina automáticamente todos los pods maestros asociados.

```
$ 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.
```