MLOps de código abierto con NetApp

NetApp Solutions

NetApp
May 10, 2024

Tabla de contenidos

MLOps de código abierto con NetApp . . . 1
- MLOps de código abierto con NetApp . . . 1
- Visión general de la tecnología . . . 1
- Arquitectura . . . 9
- Configuración de Astra Trident de NetApp . . . 10
- Kubeflow . . . 15
- Flujo de aire Apache . . . 20
- Ejemplo de operaciones de Astra Trident . . . 24
- Ejemplo de trabajos de alto rendimiento para implementaciones AIPod . . . 27

MLOps de código abierto con NetApp

MLOps de código abierto con NetApp

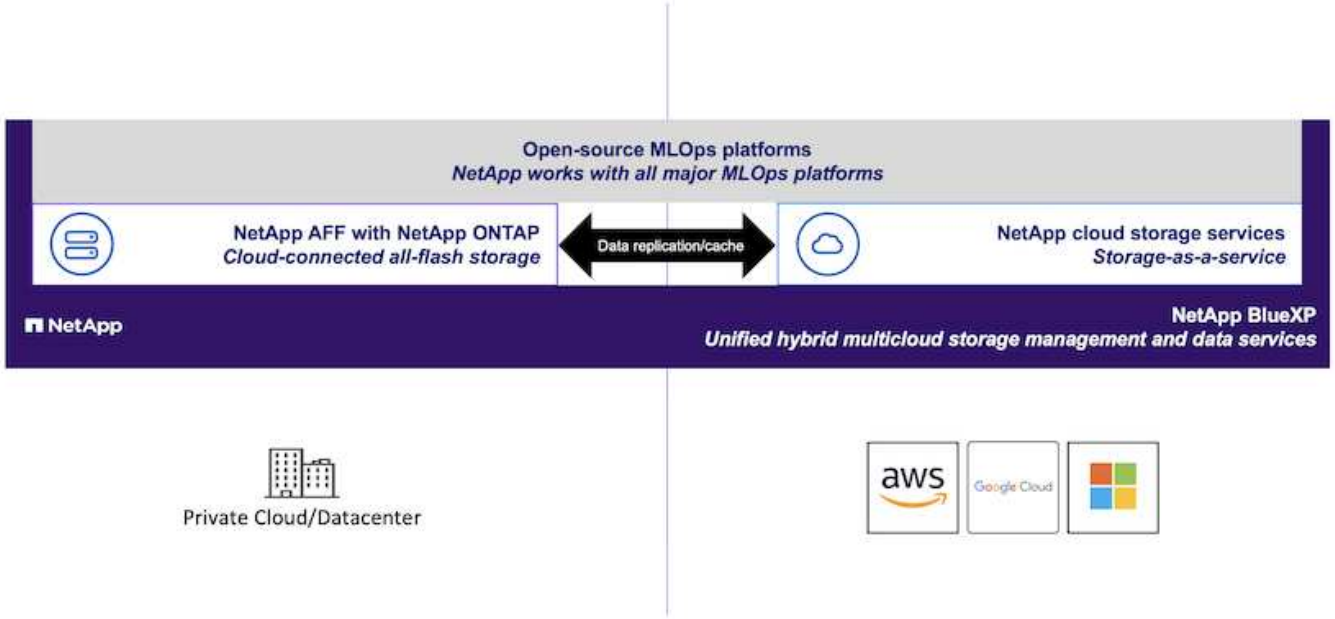
Mike Oglesby, NetApp
Mohan Acharya, NetApp

Empresas y organizaciones de todos los tamaños y sectores se están decantando por la inteligencia artificial (IA), el aprendizaje automático (ML) y el aprendizaje profundo (DL) para resolver problemas reales, ofrecer productos y servicios innovadores y obtener una ventaja en un mercado cada vez más competitivo. A medida que las organizaciones aumentan el uso de la IA, EL ML y el AP, deben hacer frente a numerosos retos, como la escalabilidad de la carga de trabajo y la disponibilidad de los datos. Esta solución demuestra cómo se pueden abordar estos retos combinando las funcionalidades de gestión de datos de NetApp con los marcos y herramientas de código abierto más populares.

La intención de esta solución es demostrar varias herramientas y marcos de código abierto diferentes que se pueden incorporar en un flujo de trabajo de MLOps. Estas diferentes herramientas y marcos pueden utilizarse de forma conjunta o por sí mismos, en función de los requisitos y el caso de uso.

En esta solución se tratan las siguientes herramientas/marcos:

- "Flujo de aire Apache"
- "Kubeflow"



Visión general de la tecnología

Inteligencia artificial

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

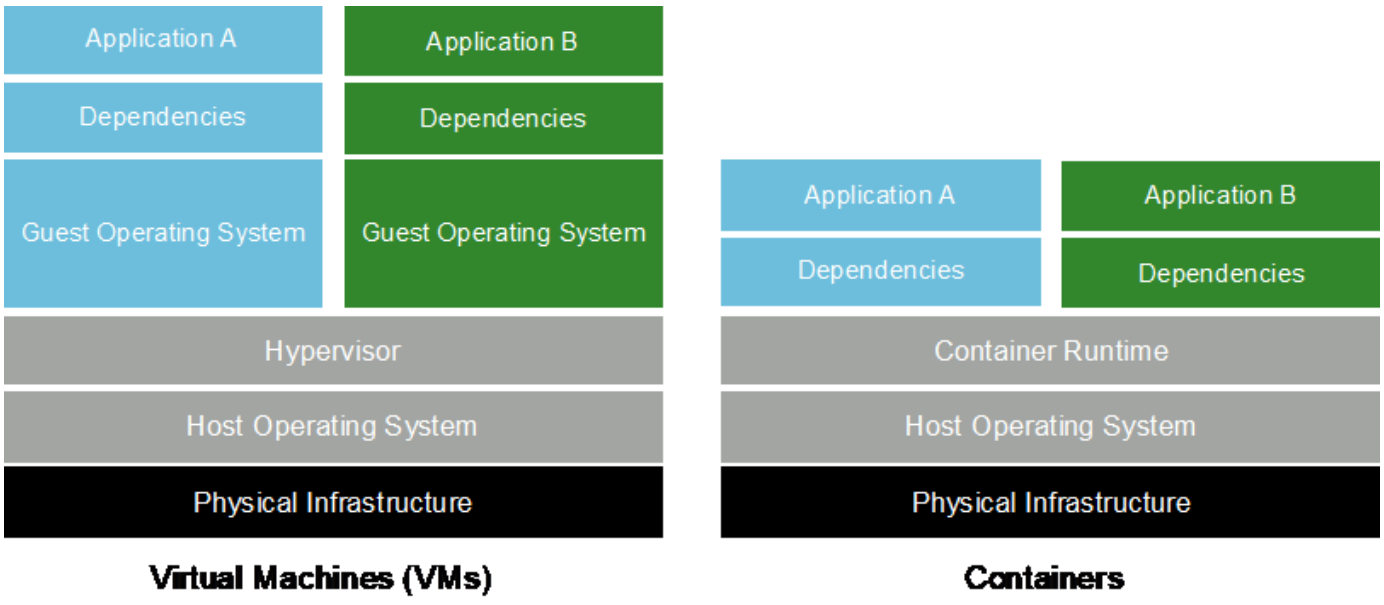
- Analizar grandes cantidades de datos para desconocer información empresarial anteriormente desconocida
- Interacción directa con los clientes mediante el procesamiento de lenguaje natural
- Automatización de diversos procesos y funciones empresariales

La formación de IA moderna y las cargas de trabajo de inferencia requieren de funcionalidades de computación en paralelo masivas. Por lo tanto, se están utilizando cada vez más GPU para ejecutar operaciones de IA, ya que las capacidades de procesamiento paralelo de las GPU son muy superiores a las de las CPU de uso general.

Contenedores

Los contenedores son instancias aisladas del espacio de usuario que se ejecutan sobre un kernel de sistema operativo host compartido. La adopción de contenedores aumenta rápidamente. Los contenedores ofrecen muchos de los mismos beneficios de uso de pruebas de espacio que las máquinas virtuales (VM). Sin embargo, debido a que se eliminan las capas de hipervisor y de sistema operativo «guest» de las que dependen las máquinas virtuales, los contenedores son mucho más ligeros. En la siguiente figura, se muestra una visualización de las máquinas virtuales en comparación con los contenedores.

Los contenedores también permiten el paquete eficiente de dependencias de aplicaciones, tiempos de ejecución, etc., directamente con una aplicación. El formato de embalaje de contenedor más utilizado es el contenedor Docker. Una aplicación que se haya contenedor 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 la "Sitio web de Docker".



Kubernetes

Kubernetes es una plataforma de orquestación de contenedores distribuida de código abierto que originalmente diseñada por Google y que ahora se mantiene mediante Cloud Native Computing Foundation (CNCF). Kubernetes permite automatizar las funciones de puesta en marcha, gestión y escalado para aplicaciones en contenedores. En los últimos años, Kubernetes se ha convertido en la plataforma de orquestación de contenedores dominante. Para obtener más información, visite la "Sitio web de Kubernetes".

Astra Trident de NetApp

Astra Trident permite el consumo y la gestión de recursos de almacenamiento en todas las plataformas de almacenamiento de NetApp más conocidas, tanto en el cloud público como en las instalaciones, incluidos ONTAP (AFF, FAS, Select, Cloud, Amazon FSx para NetApp ONTAP), el software Element (NetApp HCI, SolidFire), el servicio Azure NetApp Files y Cloud Volumes Service en Google Cloud. Astra Trident es una interfaz de almacenamiento de contenedores (CSI) que ordena el almacenamiento dinámico conforme a la normativa que se integra de forma nativa con Kubernetes.

Kit de herramientas de operaciones de datos de NetApp

La "Kit de herramientas de operaciones de datos de NetApp" Es una herramienta basada en Python que simplifica la gestión de espacios de trabajo de desarrollo y formación y servidores de inferencia respaldados por almacenamiento NetApp de escalado horizontal y de alto rendimiento. Estas son algunas funcionalidades clave:

- Aprovisionamiento rápido de nuevos espacios de trabajo de alta capacidad respaldados por almacenamiento NetApp de escalado horizontal y alto rendimiento.
- Clone casi al instante espacios de trabajo de gran capacidad para permitir la experimentación o la iteración rápida.
- Guarde casi inmediatamente instantáneas de espacios de trabajo de gran capacidad a efectos de backup o trazabilidad/creación de bases de referencia.
- Aprovisionamiento, clonado y copias Snapshot de alta capacidad y alto rendimiento casi por vía casi instusand.

Kubeflow

Kubeflow es un kit de herramientas DE IA Y ML de código abierto para Kubernetes que fue desarrollado originalmente por Google. El proyecto Kubeflow hace que la puesta en marcha de flujos de trabajo de IA y ML en Kubernetes sea sencilla, portátil y escalable. Kubeflow abstrae las complejidades de Kubernetes, lo que permite a los científicos de datos centrarse en lo que conocen mejor — de la ciencia de datos. Consulte la siguiente figura para ver una visualización. Kubeflow es una buena opción de código abierto para organizaciones que prefieren una plataforma MLOps todo en uno. Para obtener más información, visite la "Sitio web de Kubeflow".

Tuberías de Kubeflow

Los oleoductos de Kubeflow son un componente clave de Kubeflow. Las canalizaciones de Kubeflow son una plataforma y un estándar para definir y poner en marcha flujos de trabajo DE IA Y ML escalables y portátiles. Para obtener más información, consulte "Documentación oficial de Kubeflow".

Servidor de portátiles Juppyter

Un servidor Juppyter Notebook es una aplicación web de código abierto que permite a los científicos de datos

crear documentos similares a wiki llamados portátiles Juppyter que contienen código en vivo así como pruebas descriptivas. Los portátiles Juppyter se utilizan ampliamente en la comunidad de AI Y ML como medio para documentar, almacenar y compartir proyectos de IA y ML. Kubeflow simplifica el aprovisionamiento y la puesta en marcha de servidores para portátiles Juppyter en Kubernetes. Para obtener más información sobre los Cuadernos Jupyter, visite "Sitio Web de Juppyter". Para obtener más información acerca de Jupyter Notebooks en el contexto de Kubeflow, consulte "Documentación oficial de Kubeflow".

Katib

Katib es un proyecto nativo de Kubernetes para el aprendizaje automático (AutoML). Katib admite el ajuste de hiperparámetros, la detención temprana y la búsqueda de arquitectura neuronal (NAS). Katib es el proyecto que es independiente de los marcos de aprendizaje automático (ML). Puede ajustar hiperparámetros de aplicaciones escritas en cualquier idioma de la elección de los usuarios y de forma nativa es compatible con muchos marcos de ML, como TensorFlow, MXNet, PyTorch, XGBoost, y otros. Katib soporta muchos algoritmos AutoML, como optimización bayesiana, Estimadores de Árbol de Parzen, Búsqueda aleatoria, Estrategia de Evolución de Adaptación de Matriz de Covarianza, Hiperbanda, Búsqueda de Arquitectura Neural Eficiente, Búsqueda de Arquitectura Diferenciable y muchos más. Para obtener más información acerca de Jupyter Notebooks en el contexto de Kubeflow, consulte "Documentación oficial de Kubeflow".

Flujo de aire Apache

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

Gráficos de Acíclicos dirigidos (DAG)

En el flujo de aire, 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 las dos, dependiendo de la definición DAG. El programador de flujo de aire ejecuta tareas individuales en una matriz de trabajadores y cumple con las dependencias a nivel de tarea especificadas en la definición DAG. Los DAG se definen y crean a través de scripts Python.

ONTAP de NetApp

ONTAP 9, la última generación del software de gestión del almacenamiento de NetApp, permite a las empresas modernizar su infraestructura y realizar la transición a un centro de datos preparado para el cloud. ONTAP ofrece las mejores capacidades de gestión de datos y permite la gestión y protección de los datos con un solo conjunto de herramientas, sin importar dónde residan. También puede mover los datos libremente a donde sea necesario: El perímetro, el núcleo o el cloud. ONTAP 9 incluye numerosas funciones que simplifican la gestión de datos, aceleran y protegen los datos esenciales y permiten disfrutar de funcionalidades de infraestructura de nueva generación en arquitecturas de cloud híbrido.

Simplificar la gestión de los datos

La gestión de los datos es crucial para las operaciones TECNOLÓGICAS empresariales y los científicos de datos, para que se utilicen recursos apropiados para las aplicaciones de IA y para entrenar conjuntos de datos de IA/ML. La siguiente información adicional sobre las tecnologías de NetApp no está disponible para esta validación, pero puede ser relevante en función de su puesta en marcha.

El software para la gestión de datos ONTAP incluye las siguientes funciones para mejorar y simplificar las operaciones, y reducir el coste total de funcionamiento:

- Compactación de datos inline y deduplicación expandida. La compactación de datos reduce el espacio perdido dentro de los bloques de almacenamiento, mientras que la deduplicación aumenta la capacidad efectiva de forma significativa. Esto es aplicable a los datos almacenados localmente y a los datos organizados en niveles en el cloud.
- Calidad de servicio (AQoS) mínima, máxima y adaptativa. Los controles granulares de calidad de servicio (QoS) ayudan a mantener los niveles de rendimiento para aplicaciones críticas en entornos altamente compartidos.
- FabricPool de NetApp. Proporciona la organización automática en niveles de datos fríos en opciones de almacenamiento en cloud privado como Amazon Web Services (AWS), Azure y la solución de almacenamiento StorageGRID de NetApp. Para obtener más información sobre FabricPool, consulte "TR-4598: Prácticas recomendadas de FabricPool".

Acelere y proteja sus datos

ONTAP no solo ofrece niveles de rendimiento y protección de datos superiores, sino que amplía estas capacidades de las siguientes maneras:

- Rendimiento y menor latencia. ONTAP ofrece la salida más alta posible con la menor latencia posible.
- Protección de datos. ONTAP ofrece capacidades integradas de protección de datos, con una administración común entre todas las plataformas.
- Cifrado de volúmenes de NetApp (NVE). ONTAP ofrece cifrado nativo en el nivel de volumen y permite la gestión de claves incorporada o externa.
- Multi-tenancy y autenticación multifactor. ONTAP permite compartir recursos de infraestructura con los niveles más altos de seguridad.

Infraestructura preparada para futuros retos

ONTAP ayuda a satisfacer las exigentes y siempre cambiantes necesidades de su empresa con las siguientes funciones:

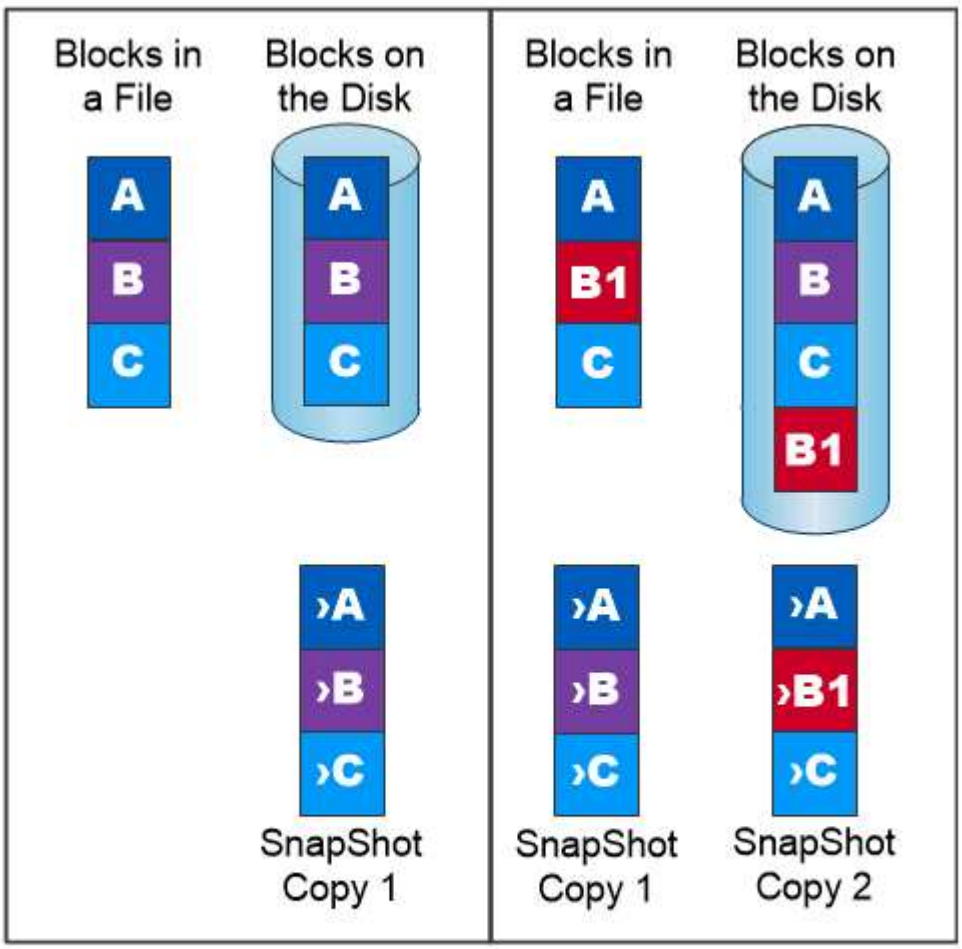
- Escalado sencillo y operaciones no disruptivas. ONTAP admite la adición no disruptiva de capacidad a las controladoras existentes y a clústeres de escalado horizontal. Los clientes pueden empezar a utilizar tecnologías punteras como NVMe y FC 32 GB, sin necesidad de realizar costosas migraciones de datos y sin cortes.
- Conexión de cloud. ONTAP es el software de gestión de almacenamiento con mejor conexión de cloud e incluye opciones de almacenamiento definido por software e instancias nativas del cloud en todos los clouds públicos.
- Integración con aplicaciones emergentes. ONTAP ofrece servicios de datos de clase empresarial para plataformas y aplicaciones de última generación, como vehículos autónomos, ciudades inteligentes e Industria 4.0, utilizando la misma infraestructura que da soporte a las aplicaciones empresariales existentes.

Copias Snapshot de NetApp

Una copia Snapshot de NetApp es una imagen puntual de solo lectura de un volumen. La imagen consume un espacio de almacenamiento mínimo y tiene una sobrecarga del rendimiento mínima, ya que solo registra los cambios que se han realizado en los archivos creados desde que se realizó la última copia Snapshot, como se muestra en la siguiente figura.

Las copias Snapshot deben su eficiencia a la tecnología de virtualización del almacenamiento central de ONTAP, el sistema de archivos de escritura en cualquier lugar (WAFL). Al igual que una base de datos, WAFL utiliza metadatos para apuntar a los bloques de datos reales en el disco. Sin embargo, a diferencia de una base de datos, WAFL no sobrescribe los bloques existentes. Escribe los datos actualizados en un bloque nuevo y cambia los metadatos. Porque ONTAP hace referencia a los metadatos cuando crea una copia Snapshot, en lugar de copiar bloques de datos, es tan eficiente que las copias Snapshot. Al hacerlo, se elimina el tiempo de búsqueda que otros sistemas incurren en la localización de los bloques a copiar, así como el costo de hacer la copia misma.

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

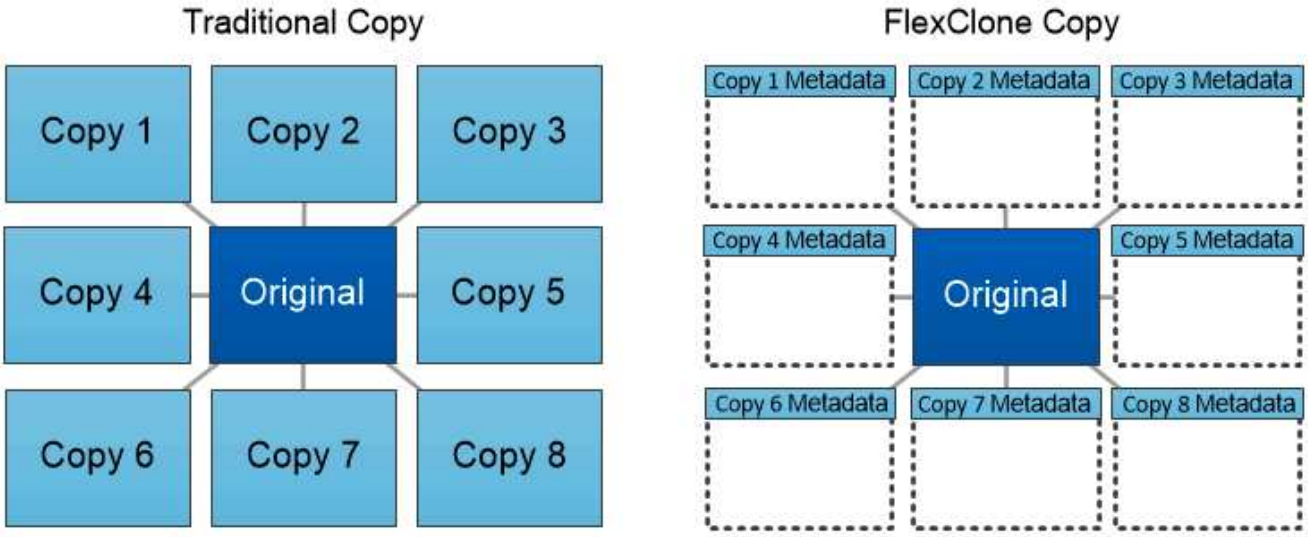


A Snapshot copy records only changes to the active file system since the last Snapshot copy.

Tecnología FlexClone de NetApp

La tecnología FlexClone de NetApp hace referencia a los metadatos de Snapshot para crear copias puntuales editables de un volumen. Las copias comparten bloques de datos con sus padres, sin consumir almacenamiento excepto lo que se necesita para los metadatos hasta que se escriben los cambios en la copia, como se muestra en la siguiente figura. Cuando se pueden crear copias tradicionales en minutos o

incluso horas, el software FlexClone le permite copiar incluso los conjuntos de datos más grandes de forma casi instantánea. Esto lo convierte en la opción ideal para las situaciones en las que necesita varias 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 de SnapMirror de NetApp

El software SnapMirror de NetApp es una solución de replicación unificada rentable y fácil de usar para todo Data Fabric. Replica datos a altas velocidades mediante LAN o WAN. Le proporciona una alta disponibilidad de datos y una rápida replicación de datos para todo tipo de aplicaciones, incluidas aplicaciones vitales para el negocio en entornos tanto virtuales como tradicionales. Al replicar datos en uno o varios sistemas de almacenamiento de NetApp y actualizar continuamente los datos secundarios, estos están siempre al día y disponibles cuando los necesite. No se requieren servidores de replicación externos. Consulte la figura siguiente para ver un ejemplo de una arquitectura que aprovecha la tecnología SnapMirror.

El software SnapMirror aprovecha las eficiencias del almacenamiento de ONTAP de NetApp y envía únicamente los bloques cambiados a través de la red. El software SnapMirror también usa la compresión de red incorporada para acelerar las transferencias de datos y reducir la utilización de ancho de banda hasta un 70 %. Con la tecnología SnapMirror, puede aprovechar un flujo de datos de thin replication para crear un único almacén que mantenga los reflejos activos y las copias de momentos específicos anteriores, lo que reduce el tráfico de red hasta un 50 %.

Copia y sincronización de NetApp BlueXP

La copia y sincronización de BlueXP es un servicio de NetApp que ofrece una sincronización de datos rápida y segura. Ya tenga que transferir archivos entre recursos compartidos de archivos NFS o SMB en las instalaciones, NetApp StorageGRID, NetApp ONTAP S3, NetApp Cloud Volumes Service, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage, o IBM Cloud Object Storage, BlueXP Copy and Sync mueve los archivos a donde los necesitas de forma rápida y segura.

Una vez transferidos los datos, estarán completamente disponibles para su uso tanto en origen como en destino. BlueXP Copy and Sync puede sincronizar los datos bajo demanda al activar una actualización o sincronizar continuamente los datos en función de una programación predefinida. Independientemente de ello,

BlueXP Copy y Sync solo mueve los diferenciales, por lo que se reducen al mínimo el tiempo y el dinero que se invierten en la replicación de datos.

Copia y sincronización de BlueXP es una herramienta de software como servicio (SaaS) extremadamente sencilla de configurar y usar. Las transferencias de datos activadas por BlueXP Copy and Sync se llevan a cabo por agentes de datos. Los agentes de datos de BlueXP Copy y Sync se pueden poner en marcha en AWS, Azure, Google Cloud Platform o en las instalaciones.

XCP de NetApp

XCP de NetApp es el software basado en cliente para migraciones de datos y análisis del sistema de archivos entre NetApp y NetApp. XCP se ha diseñado para escalar y lograr el máximo rendimiento utilizando todos los recursos del sistema disponibles para gestionar conjuntos de datos de gran volumen y migraciones de alto rendimiento. XCP le ayuda a obtener una visibilidad completa del sistema de archivos con la opción de generar informes.

XCP de NetApp está disponible en un único paquete compatible con los protocolos NFS y SMB. XCP incluye un binario de Linux para conjuntos de datos NFS y un ejecutable de Windows para conjuntos de datos SMB.

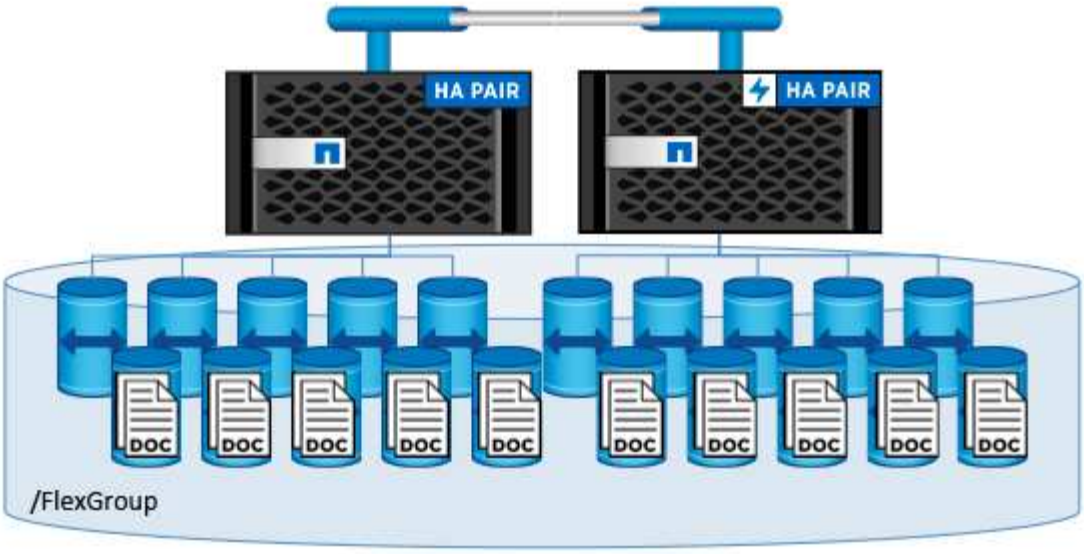
XCP File Analytics de NetApp es un software basado en host que detecta recursos compartidos de archivos, ejecuta análisis en el sistema de archivos y proporciona una consola para el análisis de archivos. XCP File Analytics es compatible con los sistemas NetApp y de otros proveedores, y se ejecuta en hosts Linux o Windows para proporcionar análisis en sistemas de archivos NFS y exportados SMB.

ONTAP FlexGroup Volumes de NetApp

Un conjunto de datos de entrenamiento puede ser una colección con hasta miles de millones de archivos. Pueden ser archivos de texto, de audio, de vídeo o cualquier otra forma de datos no estructurados que deban almacenarse y procesarse para su lectura en paralelo. El sistema de almacenamiento debe almacenar un gran número de archivos pequeños y debe leerlos en paralelo, con una entrada y salida secuencial o aleatoria

Un volumen FlexGroup es un espacio de nombres único que comprende varios volúmenes miembro constituyentes, tal y como se muestra en la siguiente figura. Desde el punto de vista de un administrador de almacenamiento, un volumen FlexGroup se gestiona y actúa como un volumen FlexVol de NetApp. Los archivos de un volumen de FlexGroup se asignan a volúmenes miembro individuales y no están repartidos en volúmenes o nodos. Ofrecen las siguientes capacidades:

- Los volúmenes FlexGroup proporcionan varios petabytes de capacidad y una baja latencia predecible para cargas de trabajo con una gran cantidad de metadatos.
- Permiten un máximo de 400 000 millones de archivos en un mismo espacio de nombres.
- Admiten operaciones en paralelo para cargas de trabajo NAS entre varias 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 cloud de NetApp compatible con Trident. Entre los ejemplos se incluyen un sistema de almacenamiento de NetApp AFF, Amazon FSx para NetApp ONTAP, Azure NetApp Files o una instancia de NetApp Cloud Volumes ONTAP. Además, la solución se puede implementar en cualquier clúster de Kubernetes, siempre y cuando la versión de Kubernetes utilizada sea compatible con Kubeflow y Astra Trident de NetApp. Si desea ver una lista de las versiones de Kubernetes compatibles con ubeflow, consulte la "Documentación oficial de Kubeflow". Si desea ver una lista de las versiones de Kubernetes compatibles con Trident, consulte "Documentación de Trident". Consulte las siguientes tablas para obtener información detallada sobre el entorno que se utilizó para validar la solución.

Componente de software	Versión
Flujo de aire Apache	2.0.1
Tabla de Helm para flujo de aire de Apache	8.0.8
Kubeflow	1,7, puesto en marcha mediante "Desplegar KF" 0.1.1
Kubernetes	1,26
Astra Trident de NetApp	23,07

Soporte técnico

NetApp no ofrece compatibilidad empresarial para Apache Airflow, Kubeflow o Kubernetes. Si está interesado en una plataforma MLOps totalmente compatible, "Póngase en contacto con NetApp" Acerca de las soluciones de MLOps totalmente compatibles que NetApp ofrece conjuntamente con partners.

Configuración de Astra Trident de NetApp

Ejemplo de backend Astra Trident para implementaciones de AIPod de NetApp

Para poder usar Astra Trident para aprovisionar de forma dinámica recursos de almacenamiento dentro de tu clúster de Kubernetes, debes crear uno o varios back-ends de Trident. Los siguientes ejemplos representan diferentes tipos de backend que puede que desee crear si va a implementar componentes de esta solución en un "AIPod de NetApp". Para obtener más información acerca de backends, consulte "Documentación de Astra Trident".

1. NetApp recomienda crear un back-end de Trident habilitado para FlexGroup para su AIPod.

   Los comandos de ejemplo siguientes muestran la creación de un back-end de Trident habilitado para FlexGroup para una máquina virtual de almacenamiento (SVM) de AIPod. Este backend utiliza el `ontap-nas-flexgroup` controlador de almacenamiento. ONTAP admite dos tipos de volúmenes de datos principales: FlexVol y FlexGroup. Los volúmenes FlexVol tienen un tamaño limitado (a partir de la escritura, el tamaño máximo depende de la implementación específica). Por otro lado, los volúmenes FlexGroup se pueden escalar de forma lineal hasta 20 PB y 400 000 millones de archivos y, además, ofrecen un espacio de nombres único que simplifica enormemente la gestión de los 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 usar volúmenes de FlexVol en lugar de volúmenes de FlexGroup, puede crear Back-ends de Trident que utilizan `ontap-nas` controlador de almacenamiento en lugar de `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 back-end de Trident habilitado para FlexVol. Es posible que desee usar FlexVol Volumes para alojar aplicaciones persistentes, almacenar resultados, resultados, información de depuración, etc. Si se desean usar volúmenes de FlexVol, se deben crear uno o varios Back-ends de Trident habilitados para FlexVol. Los siguientes comandos de ejemplo muestran la creación de un único back-end de 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 puestas en marcha de AIPod de NetApp

Para poder usar Astra Trident para aprovisionar de forma dinámica recursos de almacenamiento dentro de su clúster de Kubernetes, debe crear uno o varios StorageClasses de Kubernetes. Los ejemplos siguientes representan los diferentes tipos de StorageClasses que puede que desee crear si va a implementar componentes de

esta solución en un "AIPod de NetApp". Si desea obtener más información sobre las clases de almacenamiento, consulte "Documentación de Astra Trident".

1. NetApp recomienda crear una clase de almacenamiento para el back-end de Trident habilitado para FlexGroup que haya creado en la sección "Ejemplo de backend Astra Trident para implementaciones de AIPod de NetApp", paso 1. Los comandos de ejemplo que siguen muestran la creación de varias clases de almacenamiento que corresponden a los dos backend de ejemplo que se crearon en la sección "Ejemplo de backend Astra Trident para implementaciones de AIPod de NetApp", paso 1 - uno que utiliza "NFS sobre RDMA" y uno que no lo hace.

   Para que no se elimine un volumen persistente cuando se elimine la reclamación de volumen persistente (RVP) correspondiente, en el siguiente ejemplo se utiliza un `reclaimPolicy` valor de `Retain`. Para obtener más información acerca de `reclaimPolicy` consulte el funcionario "Documentación de Kubernetes".

   Nota: El siguiente ejemplo StorageClasses utiliza un tamaño de transferencia máximo de 262144. Para utilizar este tamaño máximo de transferencia, debe configurar el tamaño máximo de transferencia en el sistema ONTAP de forma acorde. Consulte la "Documentación de ONTAP" para obtener más detalles.

   Nota: Para utilizar NFS a través de RDMA, debe configurar NFS a través de RDMA en el sistema ONTAP. Consulte la documentación de linkhttps://docs.netapp.com/us-en/ontap/nfs-rdma/[ONTAP] para obtener más información.

   Nota: En el siguiente ejemplo, no se especifica un backend específico en el campo storagePool del archivo de definición de 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 un StorageClass que se corresponda con el back-end Trident habilitado para FlexVol que ha creado en la sección "Ejemplo de backend de Astra Trident para implementaciones de AIPod", paso 2. Los comandos de ejemplo siguientes muestran la creación de un solo tipo de almacenamiento para volúmenes FlexVol.

   Nota: En el siguiente ejemplo, no se especifica un backend particular en el campo storagePool del archivo de definición de StorageClass. Cuando se usa Kubernetes para administrar volúmenes mediante este StorageClass, Trident intenta utilizar cualquier back-end disponible que utilice el `ontap-nas` controlador.

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

Kubeflow

Despliegue de Kubeflow

En esta sección se describen las tareas que debe completar para poner en marcha Kubeflow en su clúster de Kubernetes.

Requisitos previos

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

1. Ya tiene un clúster de Kubernetes en funcionamiento y está ejecutando una versión de Kubernetes que es compatible con la versión de Kubeflow que desea implementar. Para ver una lista de las versiones de Kubernetes compatibles, consulte las dependencias de su versión de Kubeflow en la "Documentación oficial de Kubeflow".
2. Ya ha instalado y configurado Astra Trident de NetApp en su clúster de Kubernetes. Si quiere más información sobre Astra Trident, consulte la "Documentación de Astra Trident".

Establezca el tipo de almacenamiento de Kubernetes predeterminado

Antes de implementar Kubeflow, recomendamos designar un StorageClass predeterminado en su clúster de Kubernetes. El proceso de puesta en marcha de Kubeflow puede intentar aprovisionar nuevos volúmenes persistentes mediante el StorageClass predeterminado. Si no se designa ningún StorageClass como el StorageClass predeterminado, es posible que falle la implementación. Para designar un StorageClass predeterminado en el clúster, realice la siguiente tarea desde el host de salto de implementación. Si ya ha designado un tipo de almacenamiento predeterminado en el clúster, puede omitir este paso.

1. Designe una de las clases de almacenamiento existentes como clase de almacenamiento predeterminada.

Los comandos de ejemplo siguientes muestran la designación de un StorageClass llamado `ontap-ai-flexvols-retain` Como el tipo de almacenamiento predeterminado.

> La `ontap-nas-flexgroup` El tipo de backend de Trident tiene un tamaño de RVP mínimo que es bastante grande. De manera predeterminada, Kubeflow intenta suministrar EVs que son sólo unos pocos GBS en tamaño. Por lo tanto, no debe designar un StorageClass que utilice `ontap-nas-flexgroup` Tipo back-end como StorageClass predeterminado para la 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

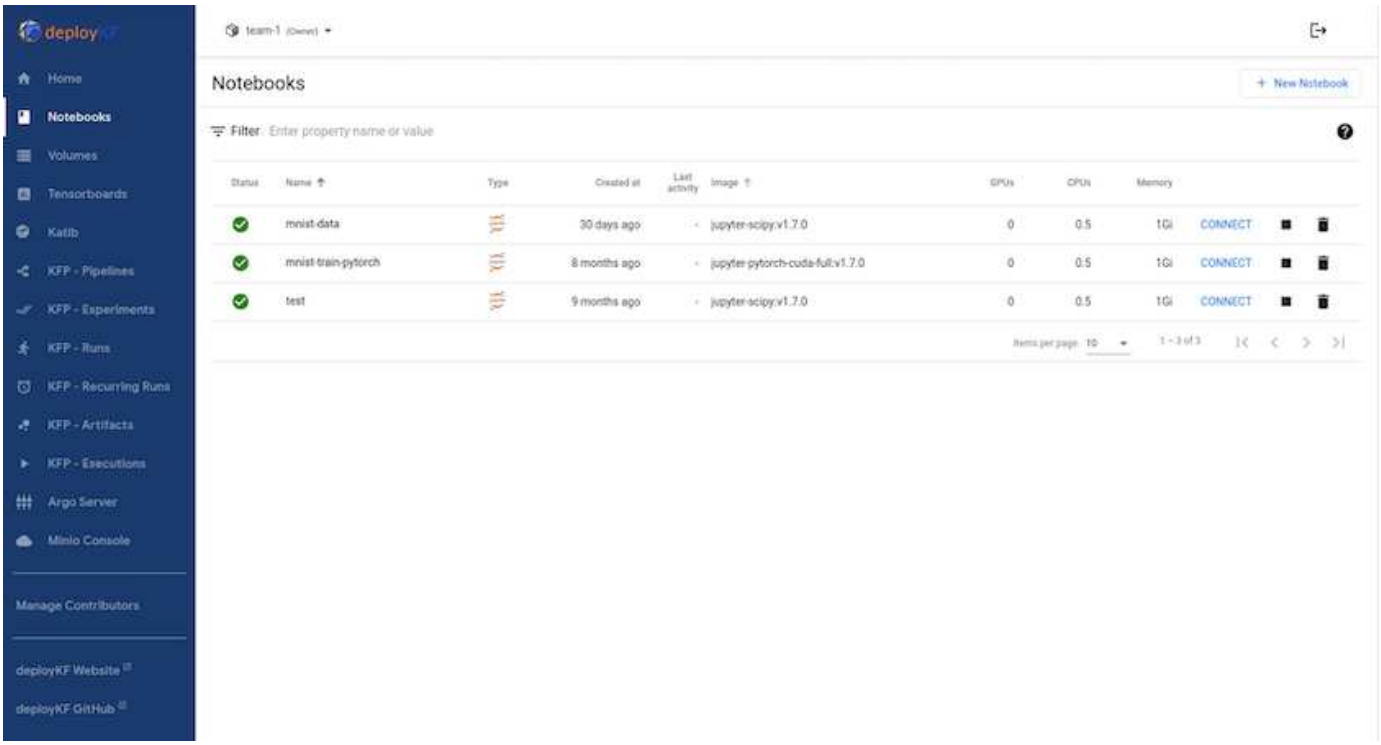
Hay muchas opciones diferentes para implementar Kubeflow. Consulte la "Documentación oficial de Kubeflow" para acceder a una lista de opciones de puesta en marcha y elija la opción que mejor se ajuste a sus necesidades.

> Para fines de validación, implementamos Kubeflow 1,7 utilizando "Desplegar KF" 0,1.1.

Ejemplo de operaciones y tareas de Kubeflow

Aprovisione un espacio de trabajo para portátiles Juppyter para uso científico de datos o desarrollador

Kubeflow es capaz de suministrar rápidamente nuevos servidores Jupyter Notebook para actuar como espacios de trabajo de científicos de datos. Para obtener más información acerca de Jupyter Notebooks dentro del contexto de Kubeflow, consulte "Documentación oficial de Kubeflow".



Use el kit de herramientas NetApp DataOps con Kubeflow

La "Kit de herramientas para la ciencia de datos de NetApp para Kubernetes" Se puede utilizar junto con Kubeflow. El uso del kit de herramientas para la ciencia de datos de NetApp con Kubeflow ofrece las siguientes ventajas:

- Los científicos de datos pueden llevar a cabo operaciones avanzadas de gestión de datos de NetApp, como la creación de copias Snapshot y clones, directamente desde un portátil de Jupyter.
- Las operaciones avanzadas de gestión de datos de NetApp, como la creación de snapshots y clones, se pueden incorporar en flujos de trabajo automatizados mediante el marco de Kubeflow Pipelines.

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

Flujo de trabajo de ejemplo: Formación de un modelo de reconocimiento de imágenes mediante Kubeflow y el kit de herramientas de DataOps de NetApp

En esta sección se describen los pasos que deben seguirse para entrenar y poner en marcha una red neuronal para el reconocimiento de imágenes mediante Kubeflow y el kit de herramientas NetApp DataOps. Esta información pretende servir de ejemplo para mostrar un trabajo de formación que incorpora almacenamiento NetApp.

Requisitos previos

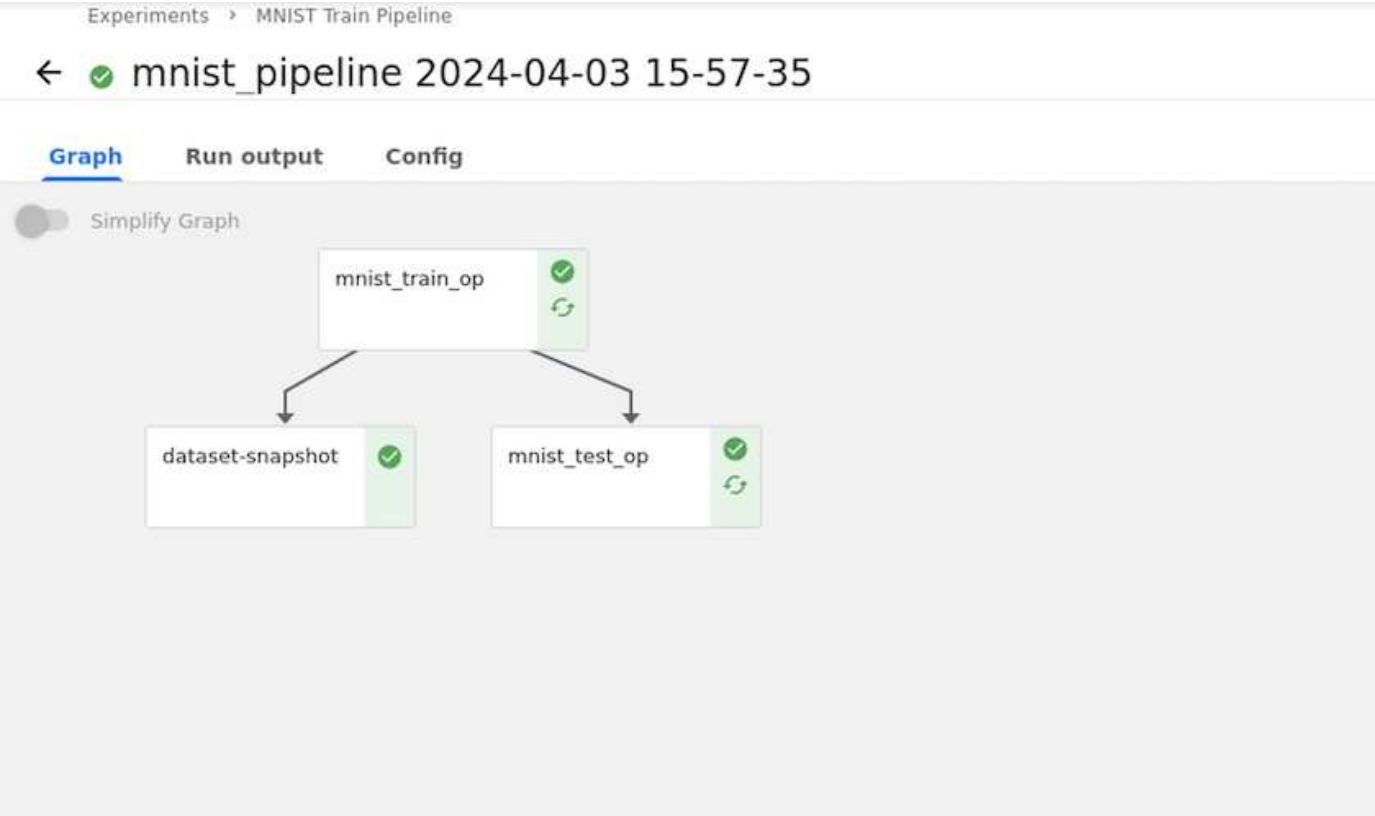
Cree un Dockerfile con las configuraciones necesarias para usar en los pasos de tren y prueba dentro de la tubería de Kubeflow.
Aquí hay un ejemplo de un Dockerfile -

```
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 asume que ya tiene un despliegue de Kubeflow en funcionamiento.

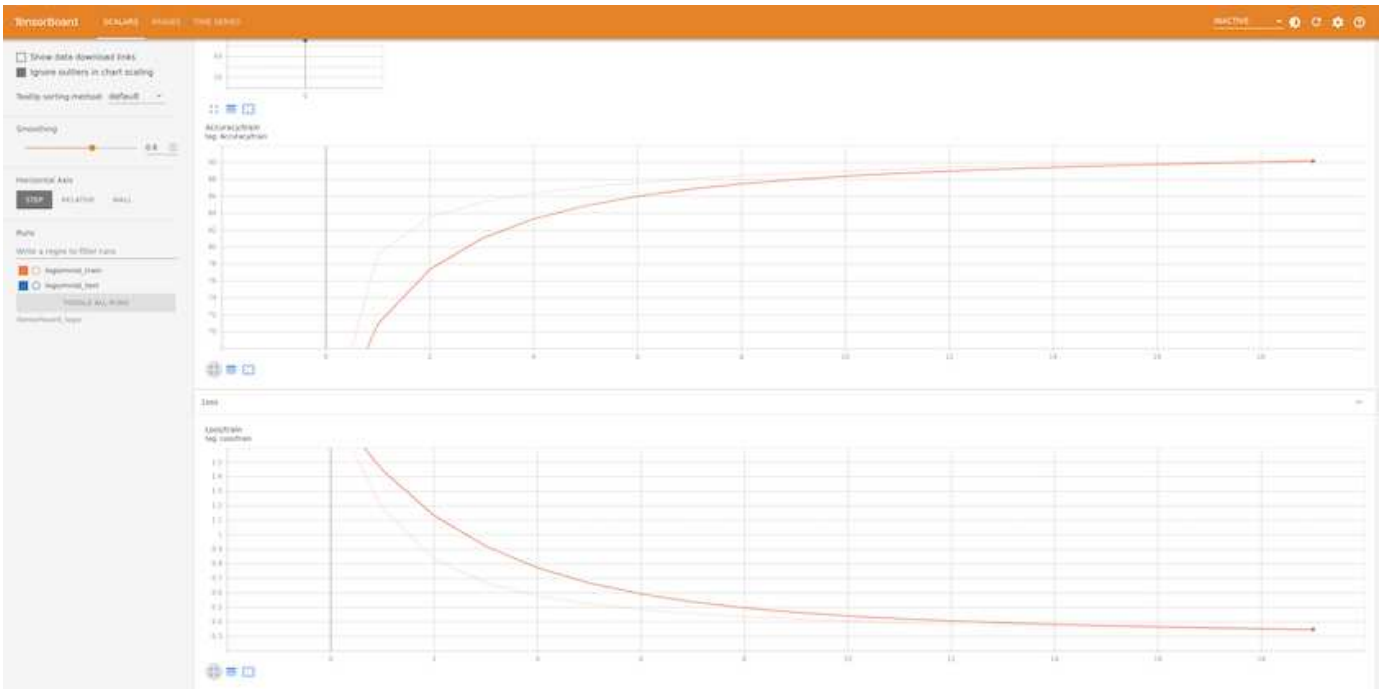
Entrena un pequeño NN en datos MNIST usando PyTorch y Kubeflow Pipelines

Utilizamos el ejemplo de una pequeña red neuronal entrenada en 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 tren y 10.000 imágenes de validación. La red neuronal utilizada para este experimento es una red de avance de 2 capas. El entrenamiento se ejecuta mediante Tuberías Kubeflow. Consulte la documentación "aquí" si quiere más información. Nuestro Pipeline Kubeflow incorpora la imagen de docker de la sección Requisitos previos.



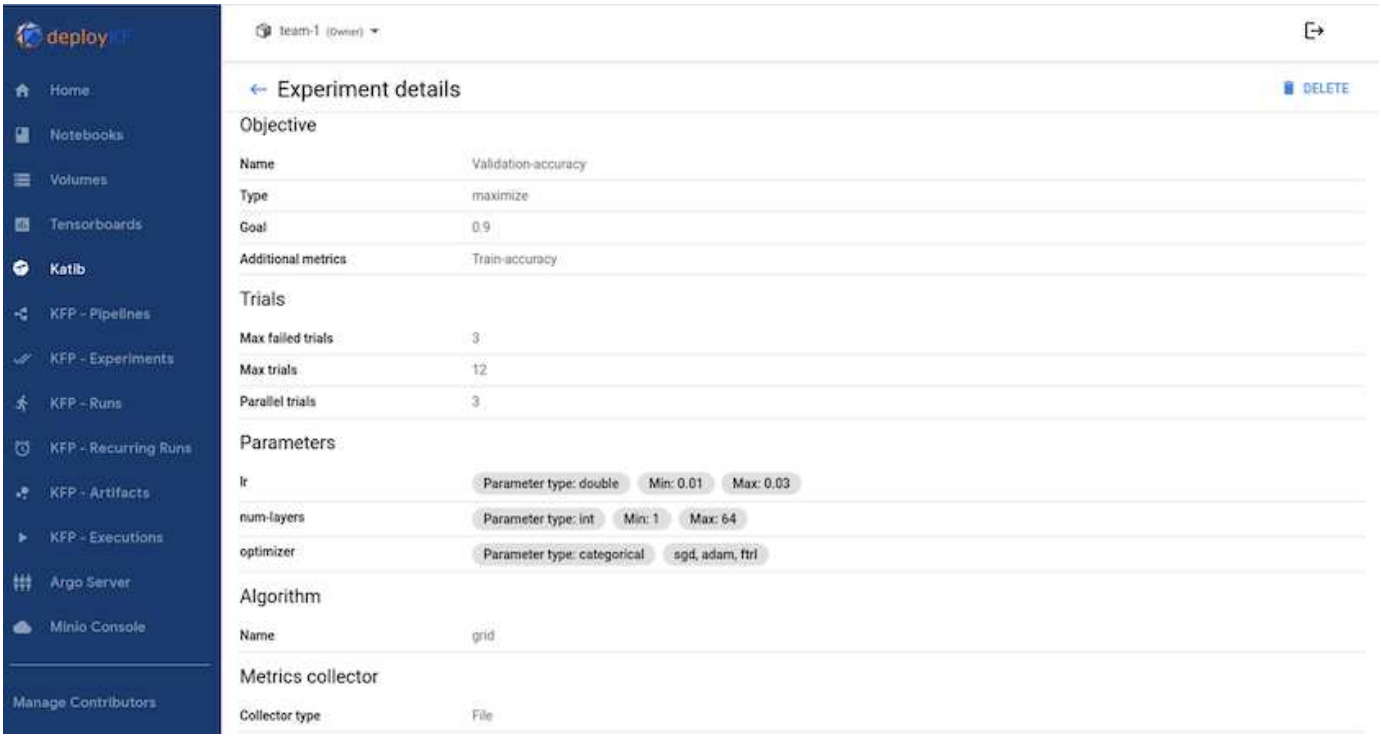
Visualiza los resultados usando Tensorboard

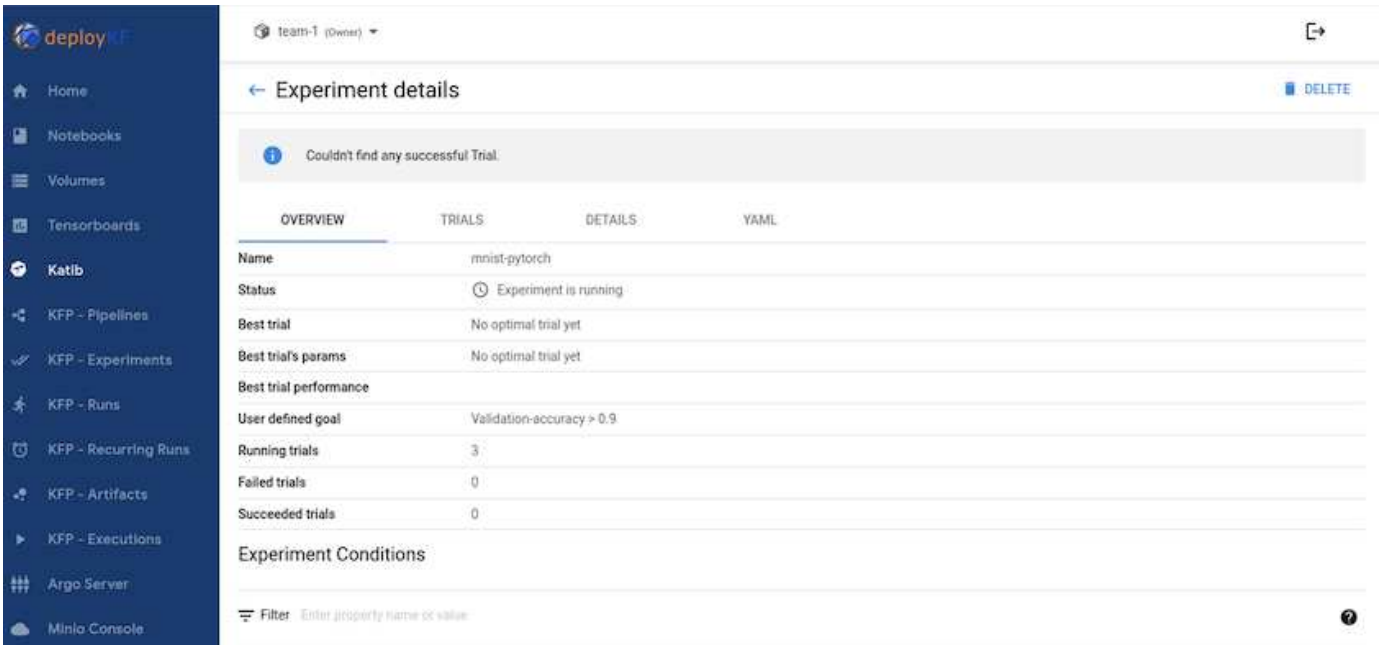
Una vez entrenado el modelo, podemos visualizar los resultados usando Tensorboard. "Tablero de placas" Está disponible como una característica en el Panel de control de Kubeflow. Puede crear un tablero de tenencia personalizado para el trabajo. Un ejemplo a continuación muestra el trazado de la precisión del entrenamiento frente a número de épocas y pérdida de entrenamiento con respecto a número de épocas.



Experimenta 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, defina primero una métrica/objetivo deseado. Esta es generalmente la precisión de la prueba. Una vez definida la métrica, elija hiperparámetros con los que desea jugar (optimizador/learning_rate/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, puede definir un archivo **YAML** con las especificaciones necesarias. A continuación se muestra una ilustración de un experimento Katib -





Utilice Instantáneas de NetApp para guardar datos y rastrearlos

Durante el entrenamiento del modelo, puede que desee guardar una instantánea del conjunto de datos de entrenamiento para la trazabilidad. Para ello, podemos agregar un paso de snapshot a la canalización, tal como se muestra a continuación. Para crear la instantánea, podemos usar la "Kit de herramientas Data OPS de NetApp 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 del kit de herramientas NetApp DataOps para Kubeflow" si quiere más información.

Flujo de aire Apache

Distribución del flujo de aire de Apache

En esta sección se describen las tareas que debe completar para poner en marcha el flujo de aire en el clúster de Kubernetes.

> Es posible poner en marcha un flujo de aire en plataformas distintas a Kubernetes. Esta solución no cubre la posibilidad de poner en marcha un flujo de aire en plataformas distintas a Kubernetes.

Requisitos previos

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

1. Ya tiene un clúster de Kubernetes en funcionamiento.
2. Ya ha instalado y configurado Astra Trident de NetApp en su clúster de Kubernetes. Si quiere más información sobre Astra Trident, consulte la "Documentación de Astra Trident".

Instale el Helm

El flujo de aire se pone en marcha con Helm, un conocido administrador de paquetes para Kubernetes. Antes de implementar el flujo de aire, debe instalar Helm en el host de salto de la implementación. Para instalar Helm en el host de salto de despliegue, siga la "instrucciones de instalación" En la documentación oficial de Helm.

Establezca el tipo de almacenamiento de Kubernetes predeterminado

Antes de implementar el flujo de aire, debe designar un tipo de almacenamiento predeterminado en el clúster de Kubernetes. El proceso de implementación de flujo de aire intenta aprovisionar nuevos volúmenes persistentes mediante el tipo de almacenamiento predeterminado. Si no se designa StorageClass como clase de almacenamiento predeterminado, la implementación falla. Para designar una clase de almacenamiento predeterminada en el clúster, siga las instrucciones descritas en "Despliegue de Kubeflow" sección. Si ya ha designado un tipo de almacenamiento predeterminado en el clúster, puede omitir este paso.

Utilice Helm para desplegar el flujo de aire

Para poner en marcha el flujo de aire en su clúster de Kubernetes con Helm, realice las siguientes tareas desde el host de salto de implementación:

1. Despliegue el flujo de aire con Helm siguiendo la "instrucciones de puesta en funcionamiento" Para el diagrama de flujo de aire oficial en el Hub de artefactos. Los comandos de ejemplo siguientes muestran la implementación del flujo de aire con Helm. Modifique, agregue o elimine valores en la `custom-values.yaml` fichero según sea necesario en función de su entorno y de 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
```

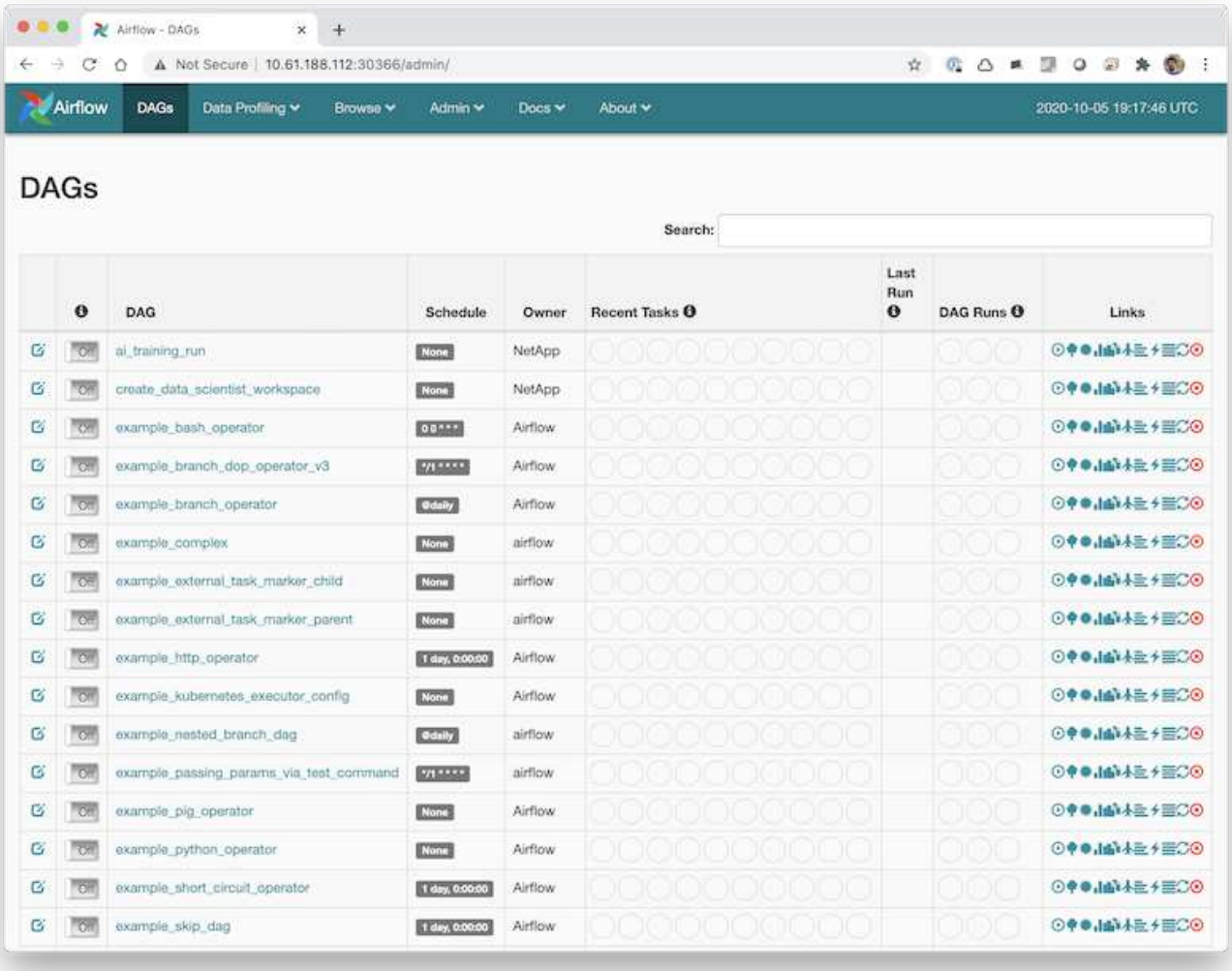
2. Confirme que todos los pods de flujo de aire estén activos y en funcionamiento. Puede tardar varios minutos en comenzar todos los pods.

```
$ 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 de flujo de aire siguiendo las instrucciones que se imprimieron en la consola cuando implementó el flujo de aire con 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 de flujo de aire.



Use el kit de herramientas de operaciones de datos de NetApp con flujo de aire

La "Kit de herramientas Data OPS de NetApp para Kubernetes" Se puede utilizar junto con el flujo de aire. El uso del kit de herramientas de DataOps de NetApp con flujo de aire le permite incorporar operaciones de gestión de datos de NetApp, como la creación de snapshots y clones, en flujos de trabajo automatizados orquestados por flujo de aire.

Consulte la "Ejemplos de flujo de aire" En el repositorio de GitHub del kit de herramientas de DataOps de NetApp para obtener más información sobre cómo usar el kit de herramientas con flujo de aire.

Ejemplo de operaciones de Astra Trident

En esta sección se incluyen ejemplos de distintas operaciones que puede que desee realizar con Astra Trident.

Importe un volumen existente

Si hay volúmenes existentes en su sistema/plataforma de almacenamiento de NetApp que desea montar en contenedores dentro de su clúster de Kubernetes, pero que no están ligados a las RVP en el clúster, debe

importar estos volúmenes. Es posible usar la funcionalidad de importación de volúmenes de Trident para importar estos volúmenes.

Los comandos de ejemplo siguientes muestran la importación de un volumen denominado `pb_fg_all`. Para obtener más información acerca de las EVs, consulte "Documentación oficial sobre Kubernetes". Para obtener más información sobre la funcionalidad de importación de volúmenes, consulte "Documentación de Trident".

An `accessModes` valor de `ReadOnlyMany` Se especifica en los archivos de especificaciones de PVC de ejemplo. Para obtener más información acerca de `accessMode` consulte "Documentación oficial sobre 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
```

Aprovisione un nuevo volumen

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

Aprovisione un nuevo volumen mediante kubectl

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

An `accessModes` valor de `ReadWriteMany` Se especifica en el siguiente archivo de definición de PVC de ejemplo. Para obtener más información acerca de `accessMode` consulte "Documentación oficial sobre 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
```

Aprovisione un nuevo volumen con el kit de herramientas de DataOps de NetApp

También puedes usar el kit de herramientas de DataOps de NetApp para Kubernetes a fin de aprovisionar un volumen nuevo en tu sistema de almacenamiento o plataforma de NetApp. El kit de herramientas DataOps de NetApp para Kubernetes utiliza Trident para aprovisionar volúmenes, pero simplifica el proceso para el usuario. Consulte la "documentación" para obtener más detalles.

Ejemplo de trabajos de alto rendimiento para implementaciones AIPod

Ejecute una carga de trabajo de IA de un solo nodo

Para ejecutar una tarea DE IA y ML de un solo nodo en el clúster de Kubernetes, realice las siguientes tareas desde el host de puesta en marcha. Con Trident, puede crear de forma rápida y sencilla un volumen de datos, con potencialmente petabytes de datos al que se puede acceder una carga de trabajo de Kubernetes. Para que un volumen de datos de este tipo sea accesible desde un pod de Kubernetes, solo tiene que especificar una RVP en la definición del pod.

En esta sección se supone que ya ha realizado un contenedor (en el formato de contenedor de Docker) con la carga de trabajo específica DE IA y ML que intenta 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 prueba de ImageNET que utiliza el conjunto de datos de TensorFlow. Para obtener más información acerca del 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 con ocho o más GPU. Este trabajo de ejemplo se puede enviar en un clúster para el que no hay un nodo de trabajo con ocho o más GPU o esté ocupado actualmente 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 en el pod que crea este trabajo. Otro volumen también se monta en el pod. Este segundo volumen se utilizará para almacenar resultados y métricas. Estos volúmenes se hacen referencia en la definición de trabajo utilizando los nombres de las RVP. Para obtener más información sobre los trabajos de Kubernetes, consulte "Documentación oficial sobre Kubernetes".

   An `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 `/dev/shm` El volumen virtual que se crea automáticamente mediante el tiempo de ejecución del contenedor Docker puede en ocasiones ser insuficiente para las necesidades de TensorFlow. Montaje de un `emptyDir` volumen como en el ejemplo siguiente proporciona un tamaño suficiente `/dev/shm` volumen virtual. Para obtener más información acerca de `emptyDir` volúmenes, consulte "Documentación oficial sobre Kubernetes".

   El contenedor único que se especifica en esta definición de trabajo de ejemplo se proporciona un `securityContext > privileged` valor de `true`. Este valor significa que el contenedor tiene acceso raíz en el host de forma efectiva. Esta anotación se utiliza en este caso porque la carga de trabajo específica que se está ejecutando requiere acceso raíz. Específicamente, una operación de caché clara que ejecuta la carga de trabajo requiere acceso raíz. Si esto o no `privileged: true` la anotación es necesaria depende de los requisitos de la carga de trabajo específica que se 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 ha creado en el paso 1 se está ejecutando correctamente. El siguiente comando de ejemplo confirma que se creó un solo pod para el trabajo, tal como se especifica en la definición de trabajos, y que este pod se ejecuta 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 ha creado en el paso 1 se ha completado correctamente. Los siguientes

comandos de ejemplo confirman que el trabajo se ha completado 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 artefactos de trabajo. Los siguientes comandos de ejemplo muestran la eliminación del objeto de trabajo creado en el paso 1.

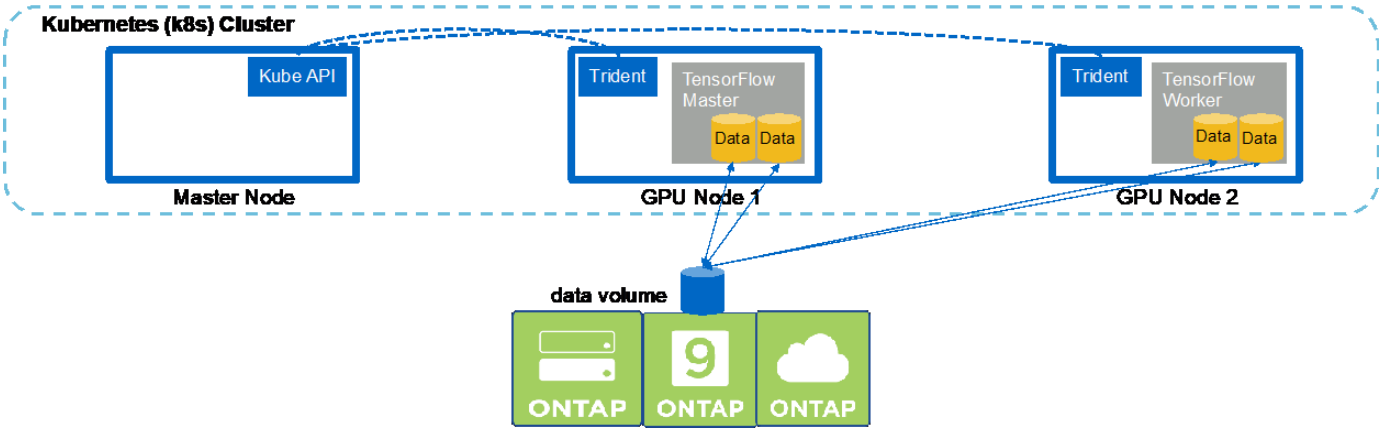
   Cuando se elimina el objeto de trabajo, Kubernetes elimina automáticamente todos los pods asociados.

```
$ 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.
```

Ejecute una carga de trabajo de IA distribuida síncrona

Para ejecutar un trabajo de IA y ML multinodo síncrono en un clúster de Kubernetes, lleve a cabo las siguientes tareas en el host de saltos de la puesta en marcha. Este proceso le permite aprovechar los datos almacenados en un volumen de NetApp y utilizar más GPU de las que puede proporcionar un único nodo de trabajo. Consulte la siguiente figura para obtener una descripción de un trabajo de IA distribuido sincrónica.

Los trabajos distribuidos síncronos pueden ayudar a aumentar el rendimiento y la precisión de la formación en comparación con los trabajos distribuidos de manera asíncrona. Un análisis de los pros y los contras de los trabajos síncronos frente a los trabajos asincrónicos está fuera del alcance de este documento.



1. En los siguientes comandos de ejemplo, se muestra la creación de un trabajador que participa en la ejecución síncrona y distribuida de la misma tarea de prueba de rendimiento TensorFlow que se ejecutó en un solo nodo en el ejemplo de la sección "Ejecute una carga de trabajo de IA de un solo nodo". En este ejemplo específico, sólo se implementa un único trabajador porque el trabajo se ejecuta en dos nodos de

trabajo.

Esta puesta en marcha de trabajo de ejemplo solicita ocho GPU y, por lo tanto, puede ejecutarse en un único nodo de trabajo de GPU con ocho o más GPU. Si los nodos de trabajo de la GPU tienen más de ocho GPU, para maximizar el rendimiento, es posible que desee aumentar este número para ser igual al número de GPU que disponen los nodos de trabajo. Para obtener más información sobre las puestas en marcha de Kubernetes, consulte "Documentación oficial sobre Kubernetes".

En este ejemplo se crea una puesta en marcha de Kubernetes, ya que este trabajador en contenedor específico nunca lo completaría por sí solo. Por lo tanto, no tiene sentido implementarlo usando la construcción de trabajos de Kubernetes. Si su trabajador está diseñado o escrito para completar por sí solo, entonces podría tener sentido utilizar la construcción del trabajo para desplegar a su trabajador.

El POD especificado en esta especificación de implementación de ejemplo recibe una `hostNetwork` valor de `true`. Este valor significa que el pod utiliza la pila de red del nodo de trabajo del host en lugar de la pila de red virtual que Kubernetes suele crear para cada pod. Esta anotación se utiliza en este caso porque la carga de trabajo específica depende de Open MPI, NCCL y Horovod para ejecutar la carga de trabajo de forma síncrona distribuida. Por lo tanto, requiere acceso a la pila de red del host. Un debate sobre Open MPI, NCCL y Horovod está fuera del alcance de este documento. Si esto o no `hostNetwork: true` la anotación es necesaria depende de los requisitos de la carga de trabajo específica que se esté ejecutando. Para obtener más información acerca de `hostNetwork` consulte "Documentación oficial sobre 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 el despliegue del trabajador que creó en el paso 1 se inició correctamente. Los siguientes comandos de ejemplo confirman que se creó un solo pod de trabajadores para la implementación, tal y como se indica en la definición de la puesta en marcha, y que este pod se ejecuta 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 en un maestro que se inicia, participe y realice un seguimiento de la ejecución de un trabajo de varios nodos síncronos. Los siguientes comandos de ejemplo crean un maestro que inicia sesión, participa en y realiza un seguimiento de la ejecución síncrona distribuida de la misma tarea de prueba de rendimiento TensorFlow que se ejecutó en un solo nodo del ejemplo de la sección "Ejecute 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 único nodo de trabajo de GPU con ocho o más GPU. Si los nodos de trabajo de la GPU tienen más de ocho GPU, para maximizar el rendimiento, es posible que desee aumentar este número para ser igual al número de GPU que disponen los nodos de trabajo.

   El POD maestro especificado en esta definición de trabajo de ejemplo recibe una `hostNetwork` valor de `true`, así como se le dio a la cápsula de trabajo un `hostNetwork` valor de `true` en el paso 1. Consulte el paso 1 para obtener más información acerca de 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 está ejecutando correctamente. El siguiente comando de ejemplo confirma que se creó un único pod maestro para el trabajo, tal como se indica en la definición de trabajos, y que este pod se ejecuta actualmente en uno de los nodos de trabajo de la GPU. También debe ver que el pod de trabajo que originalmente vio en el paso 1 sigue en ejecución y que los pods maestro y trabajador se ejecutan en diferentes nodos.

```
$ 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 ha creado en el paso 3 se ha completado correctamente. Los siguientes comandos de ejemplo confirman que el trabajo se ha completado 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. Elimine la implementación del trabajador cuando ya no la necesite. Los siguientes comandos de ejemplo muestran la eliminación del objeto de implementación de trabajo que se creó en el paso 1.

Cuando se elimina el objeto de implementación de trabajo, 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:** Limpie 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 se 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.
```

Información de copyright

Información de la marca comercial