MLOps de código aberto com NetApp

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Índice

MLOps de código aberto com NetApp

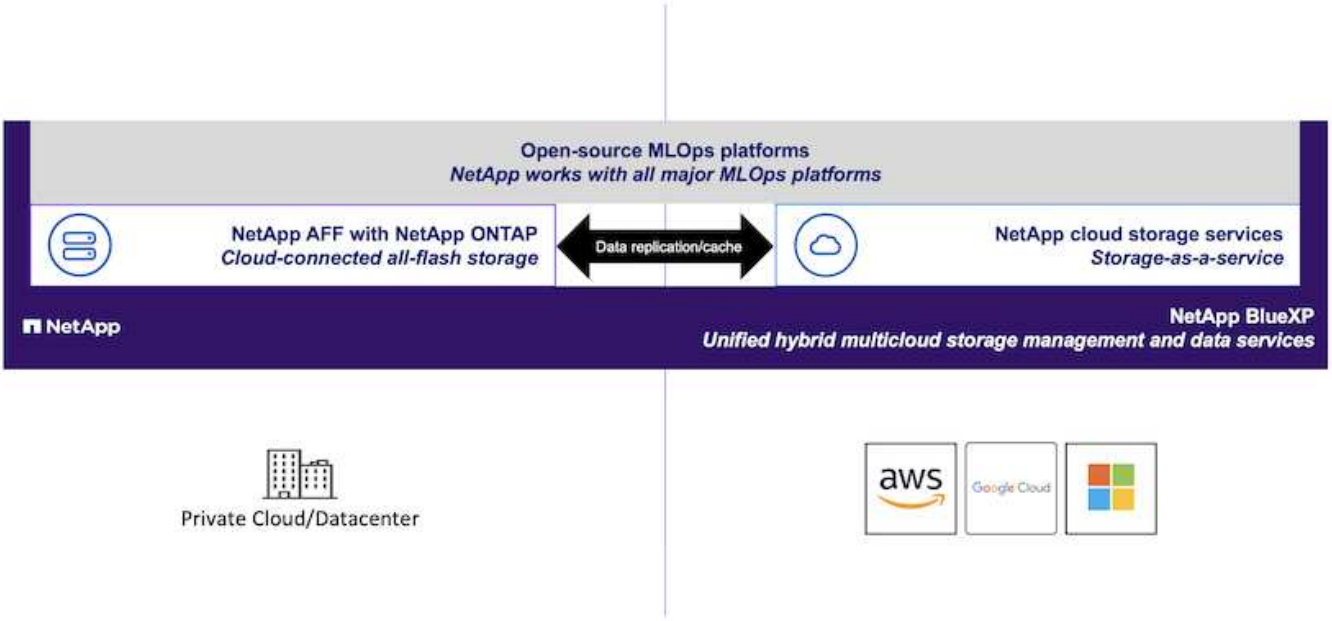
MLOps de código aberto com NetApp

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

Empresas e organizações de todos os tamanhos e de diversos setores estão recorrendo à inteligência artificial (IA) para resolver problemas do mundo real, fornecer produtos e serviços inovadores e obter vantagem em um mercado cada vez mais competitivo. Muitas organizações estão recorrendo a ferramentas MLOps de código aberto para acompanhar o ritmo acelerado de inovação no setor. Essas ferramentas de código aberto oferecem recursos avançados e de ponta, mas geralmente não levam em conta a disponibilidade e a segurança dos dados. Infelizmente, isso significa que cientistas de dados altamente qualificados são forçados a gastar uma quantidade significativa de tempo esperando para obter acesso aos dados ou esperar que operações rudimentares relacionadas a dados sejam concluídas. Ao combinar ferramentas populares de MLOps de código aberto com uma infraestrutura de dados inteligente da NetApp, as organizações podem acelerar seus pipelines de dados, o que, por sua vez, acelera suas iniciativas de IA. Eles podem extrair valor de seus dados e, ao mesmo tempo, garantir que eles permaneçam protegidos e seguros. Esta solução demonstra o emparelhamento de recursos de gerenciamento de dados da NetApp com diversas ferramentas e estruturas populares de código aberto para enfrentar esses desafios.

A lista a seguir destaca alguns dos principais recursos habilitados por esta solução:

- Os usuários podem provisionar rapidamente novos volumes de dados de alta capacidade e espaços de trabalho de desenvolvimento apoiados pelo armazenamento NetApp de alto desempenho e escalonável.
- Os usuários podem clonar quase instantaneamente volumes de dados de alta capacidade e espaços de trabalho de desenvolvimento para permitir experimentação ou iteração rápida.
- Os usuários podem salvar quase instantaneamente snapshots de volumes de dados de alta capacidade e espaços de trabalho de desenvolvimento para backup e/ou rastreabilidade/linha de base.



Um fluxo de trabalho MLOps típico incorpora espaços de trabalho de desenvolvimento, geralmente assumindo a forma de"Cadernos Jupyter" ; rastreamento de experimentos; pipelines de treinamento automatizados; pipelines de dados; e inferência/implantação. Esta solução destaca diversas ferramentas e estruturas diferentes que podem ser usadas de forma independente ou em conjunto para abordar os diferentes aspectos do fluxo de trabalho. Também demonstramos o emparelhamento de recursos de gerenciamento de dados da NetApp com cada uma dessas ferramentas. Esta solução tem como objetivo oferecer blocos de construção a partir dos quais uma organização pode construir um fluxo de trabalho MLOps personalizado, específico para seus casos de uso e requisitos.

As seguintes ferramentas/estruturas são abordadas nesta solução:

- "Apache Airflow"
- "JupyterHub"
- "Fluxo de cubo"
- "Fluxo de ML"

A lista a seguir descreve padrões comuns para implantar essas ferramentas de forma independente ou em conjunto.

- Implantar JupyterHub, MLflow e Apache Airflow em conjunto - JupyterHub para"Cadernos Jupyter" , MLflow para rastreamento de experimentos e Apache Airflow para treinamento automatizado e pipelines de dados.
- Implantar Kubeflow e Apache Airflow em conjunto - Kubeflow para"Cadernos Jupyter" , rastreamento de experimentos, pipelines de treinamento automatizados e inferência; e Apache Airflow para pipelines de dados.
- Implemente o Kubeflow como uma solução de plataforma MLOps completa para"Cadernos Jupyter" , rastreamento de experimentos, treinamento automatizado e pipelines de dados e inferência.

Visão geral da tecnologia

Esta seção se concentra na visão geral da tecnologia para OpenSource MLOps com NetApp.

Inteligência artificial

IA é uma disciplina da ciência da computação na qual os computadores são treinados para imitar as funções cognitivas da mente humana. Os desenvolvedores de IA treinam computadores para aprender e resolver problemas de uma maneira semelhante, ou até mesmo superior, aos humanos. Aprendizado profundo e aprendizado de máquina são subcampos da IA. As organizações estão adotando cada vez mais IA, ML e DL para dar suporte às suas necessidades comerciais críticas. Alguns exemplos são os seguintes:

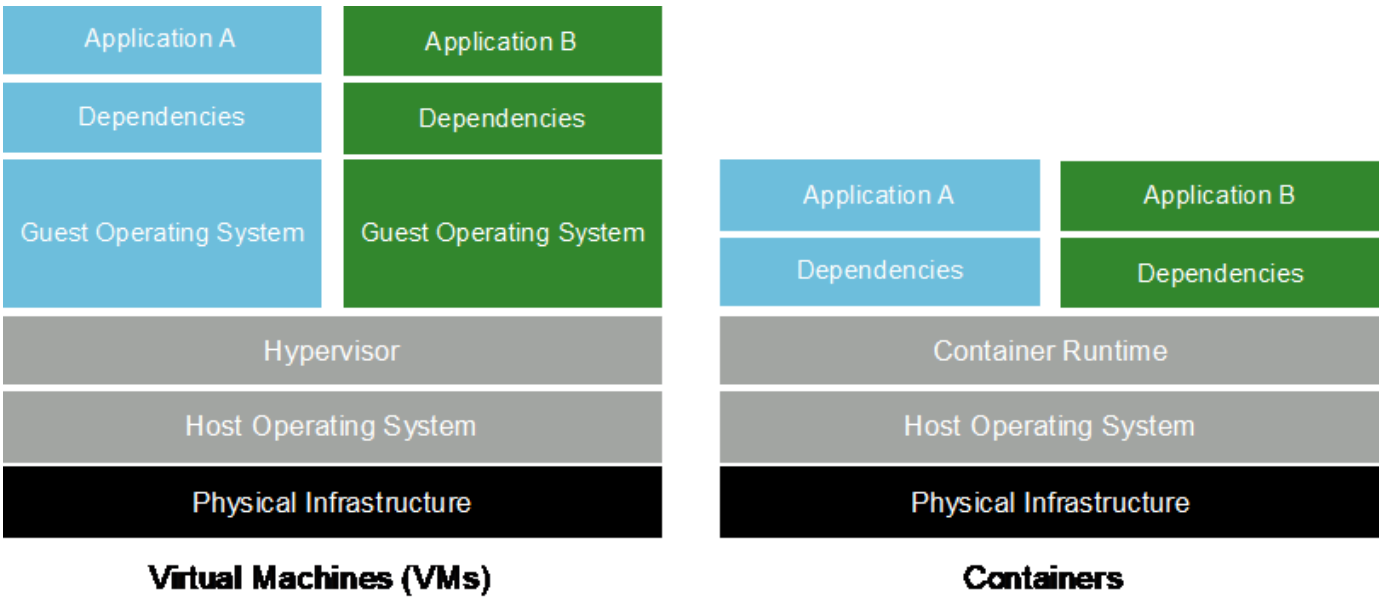
- Analisar grandes quantidades de dados para descobrir insights de negócios até então desconhecidos
- Interagindo diretamente com os clientes usando processamento de linguagem natural
- Automatizar vários processos e funções de negócios

As cargas de trabalho modernas de treinamento e inferência de IA exigem recursos de computação massivamente paralelos. Portanto, as GPUs estão sendo cada vez mais usadas para executar operações de IA porque as capacidades de processamento paralelo das GPUs são muito superiores às das CPUs de uso geral.

Recipientes

Os contêineres são instâncias isoladas do espaço do usuário que são executadas sobre um kernel de sistema operacional de host compartilhado. A adoção de contêineres está aumentando rapidamente. Os contêineres oferecem muitos dos mesmos benefícios de sandbox de aplicativos que as máquinas virtuais (VMs) oferecem. No entanto, como as camadas do hipervisor e do sistema operacional convidado das quais as VMs dependem foram eliminadas, os contêineres são muito mais leves. A figura a seguir descreve uma visualização de máquinas virtuais versus contêineres.

Os contêineres também permitem o empacotamento eficiente de dependências de aplicativos, tempos de execução e assim por diante, diretamente com um aplicativo. O formato de embalagem de contêiner mais comumente usado é o contêiner Docker. Um aplicativo que foi conteinerizado no formato de contêiner Docker pode ser executado em qualquer máquina que possa executar contêineres Docker. Isso é verdadeiro mesmo que as dependências do aplicativo não estejam presentes na máquina, porque todas as dependências são empacotadas no próprio contêiner. Para mais informações, visite o "Site do Docker" .



Kubernetes

O Kubernetes é uma plataforma de orquestração de contêineres distribuída e de código aberto que foi originalmente projetada pelo Google e agora é mantida pela Cloud Native Computing Foundation (CNCF). O Kubernetes permite a automação de funções de implantação, gerenciamento e dimensionamento para aplicativos em contêineres. Nos últimos anos, o Kubernetes emergiu como a plataforma dominante de orquestração de contêineres. Para mais informações, visite o "Site do Kubernetes" .

NetApp Trident

"Trident"permite o consumo e o gerenciamento de recursos de armazenamento em todas as plataformas de armazenamento populares da NetApp , na nuvem pública ou local, incluindo ONTAP (AFF, FAS, Select, Cloud, Amazon FSx ONTAP), serviço Azure NetApp Files e Google Cloud NetApp Volumes. O Trident é um orquestrador de armazenamento dinâmico compatível com Container Storage Interface (CSI) que se integra nativamente ao Kubernetes.

Kit de ferramentas NetApp DataOps

O"Kit de ferramentas NetApp DataOps" é uma ferramenta baseada em Python que simplifica o gerenciamento de espaços de trabalho de desenvolvimento/treinamento e servidores de inferência apoiados por armazenamento NetApp de alto desempenho e escalável. Os principais recursos incluem:

- Provisione rapidamente novos espaços de trabalho de alta capacidade apoiados por armazenamento NetApp de alto desempenho e escalonável.
- Clone quase instantaneamente espaços de trabalho de alta capacidade para permitir experimentação ou iteração rápida.
- Salve instantâneos de espaços de trabalho de alta capacidade para backup e/ou rastreabilidade/linha de base.
- Provisione, clone e crie snapshots de volumes de dados de alta capacidade e alto desempenho quase instantaneamente.

Apache Airflow

O Apache Airflow é uma plataforma de gerenciamento de fluxo de trabalho de código aberto que permite a criação programática, o agendamento e o monitoramento de fluxos de trabalho empresariais complexos. Ele é frequentemente usado para automatizar fluxos de trabalho de ETL e pipeline de dados, mas não se limita a esses tipos de fluxos de trabalho. O projeto Airflow foi iniciado pelo Airbnb, mas desde então se tornou muito popular no setor e agora está sob os auspícios da Apache Software Foundation. O Airflow é escrito em Python, os fluxos de trabalho do Airflow são criados por meio de scripts Python e o Airflow é projetado sob o princípio de "configuração como código". Muitos usuários corporativos do Airflow agora executam o Airflow no Kubernetes.

Grafos Acíclicos Dirigidos (GADs)

No Airflow, os fluxos de trabalho são chamados de Gráficos Acíclicos Direcionados (DAGs). Os DAGs são compostos de tarefas executadas em sequência, em paralelo ou uma combinação dos dois, dependendo da definição do DAG. O agendador do Airflow executa tarefas individuais em uma matriz de trabalhadores, aderindo às dependências de nível de tarefa especificadas na definição do DAG. Os DAGs são definidos e criados por meio de scripts Python.

Caderno Jupyter

Os Jupyter Notebooks são documentos semelhantes a wiki que contêm código ativo e texto descritivo. Os Jupyter Notebooks são amplamente utilizados na comunidade de IA e ML como um meio de documentar, armazenar e compartilhar projetos de IA e ML. Para mais informações sobre Jupyter Notebooks, visite o "Site do Jupyter" .

Servidor de notebook Jupyter

Um Jupyter Notebook Server é um aplicativo web de código aberto que permite aos usuários criar Jupyter Notebooks.

JupyterHub

O JupyterHub é um aplicativo multiusuário que permite que um usuário individual provisione e acesse seu próprio servidor Jupyter Notebook. Para mais informações sobre o JupyterHub, visite o "Site JupyterHub" .

Fluxo de ML

MLflow é uma plataforma popular de gerenciamento de ciclo de vida de IA de código aberto. Os principais recursos do MLflow incluem rastreamento de experimentos de IA/ML e um repositório de modelos de IA/ML. Para mais informações sobre o MLflow, visite o "Site MLflow" .

Fluxo de cubo

Kubeflow é um kit de ferramentas de IA e ML de código aberto para Kubernetes que foi originalmente desenvolvido pelo Google. O projeto Kubeflow torna as implantações de fluxos de trabalho de IA e ML no Kubernetes simples, portáteis e escaláveis. O Kubeflow abstrai as complexidades do Kubernetes, permitindo que cientistas de dados se concentrem no que sabem fazer melhor: ciência de dados. Veja a figura a seguir para uma visualização. O Kubeflow é uma boa opção de código aberto para organizações que preferem uma plataforma MLOps completa. Para mais informações, visite o "Site do Kubeflow" .

Pipelines Kubeflow

Os pipelines do Kubeflow são um componente essencial do Kubeflow. Os Kubeflow Pipelines são uma plataforma e um padrão para definir e implantar fluxos de trabalho de IA e ML portáteis e escaláveis. Para mais informações, consulte o "documentação oficial do Kubeflow" .

Cadernos Kubeflow

O Kubeflow simplifica o provisionamento e a implantação de Jupyter Notebook Servers no Kubernetes. Para obter mais informações sobre Jupyter Notebooks no contexto do Kubeflow, consulte o "documentação oficial do Kubeflow" .

Katib

Katib é um projeto nativo do Kubernetes para aprendizado de máquina automatizado (AutoML). O Katib oferece suporte ao ajuste de hiperparâmetros, parada antecipada e pesquisa de arquitetura neural (NAS). Katib é um projeto agnóstico em relação a frameworks de aprendizado de máquina (ML). Ele pode ajustar hiperparâmetros de aplicativos escritos em qualquer linguagem escolhida pelos usuários e oferece suporte nativo a muitas estruturas de ML, como TensorFlow, MXNet, PyTorch, XGBoost e outras. O Katib oferece suporte a vários algoritmos AutoML, como otimização bayesiana, estimadores de árvore de Parzen, busca aleatória, estratégia de evolução de adaptação de matriz de covariância, hiperbanda, busca de arquitetura neural eficiente, busca de arquitetura diferenciável e muito mais. Para obter mais informações sobre Jupyter

Notebooks no contexto do Kubeflow, consulte o "documentação oficial do Kubeflow" .

NetApp ONTAP

ONTAP 9, a última geração de software de gerenciamento de armazenamento da NetApp, permite que as empresas modernizem a infraestrutura e façam a transição para um data center pronto para a nuvem. Aproveitando os recursos de gerenciamento de dados líderes do setor, o ONTAP permite o gerenciamento e a proteção de dados com um único conjunto de ferramentas, independentemente de onde os dados residam. Você também pode mover dados livremente para onde for necessário: na borda, no núcleo ou na nuvem. O ONTAP 9 inclui vários recursos que simplificam o gerenciamento de dados, aceleram e protegem dados críticos e permitem recursos de infraestrutura de última geração em arquiteturas de nuvem híbrida.

Simplifique o gerenciamento de dados

O gerenciamento de dados é crucial para as operações de TI corporativas e cientistas de dados, para que recursos apropriados sejam usados para aplicativos de IA e treinamento de conjuntos de dados de IA/ML. As seguintes informações adicionais sobre as tecnologias NetApp estão fora do escopo desta validação, mas podem ser relevantes dependendo da sua implantação.

O software de gerenciamento de dados ONTAP inclui os seguintes recursos para otimizar e simplificar as operações e reduzir seu custo total de operação:

- Compactação de dados em linha e desduplicação expandida. A compactação de dados reduz o desperdício de espaço dentro dos blocos de armazenamento e a desduplicação aumenta significativamente a capacidade efetiva. Isso se aplica a dados armazenados localmente e dados em camadas na nuvem.
- Qualidade de serviço mínima, máxima e adaptável (AQoS). Controles granulares de qualidade de serviço (QoS) ajudam a manter os níveis de desempenho para aplicativos críticos em ambientes altamente compartilhados.
- NetApp FabricPool. Fornece hierarquização automática de dados frios para opções de armazenamento em nuvem pública e privada, incluindo Amazon Web Services (AWS), Azure e solução de armazenamento NetApp StorageGRID . Para obter mais informações sobre FabricPool, consulte "TR-4598: Melhores práticas do FabricPool" .

Acelere e proteja os dados

O ONTAP oferece níveis superiores de desempenho e proteção de dados e estende esses recursos das seguintes maneiras:

- Desempenho e menor latência. ONTAP oferece o maior rendimento possível com a menor latência possível.
- Proteção de dados. O ONTAP fornece recursos integrados de proteção de dados com gerenciamento comum em todas as plataformas.
- Criptografia de volume NetApp (NVE). O ONTAP oferece criptografia nativa em nível de volume com suporte para gerenciamento de chaves externo e integrado.
- Multilocação e autenticação multifator. O ONTAP permite o compartilhamento de recursos de infraestrutura com os mais altos níveis de segurança.

Infraestrutura à prova do futuro

O ONTAP ajuda a atender às necessidades empresariais exigentes e em constante mudança com os seguintes recursos:

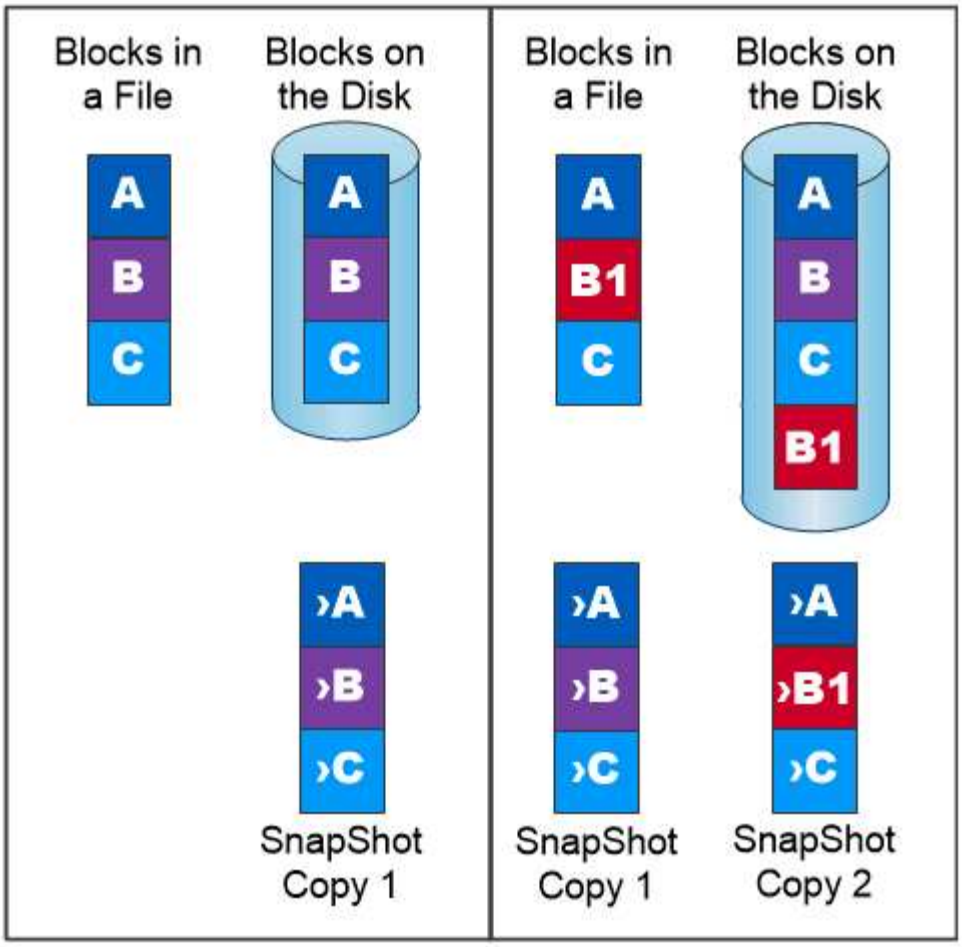
- Escalabilidade perfeita e operações não disruptivas. O ONTAP oferece suporte à adição não disruptiva de capacidade aos controladores existentes e aos clusters escaláveis. Os clientes podem atualizar para as tecnologias mais recentes sem migrações de dados dispendiosas ou interrupções.
- Conexão em nuvem. ONTAP é o software de gerenciamento de armazenamento mais conectado à nuvem, com opções para armazenamento definido por software e instâncias nativas da nuvem em todas as nuvens públicas.
- Integração com aplicações emergentes. A ONTAP oferece serviços de dados de nível empresarial para plataformas e aplicativos de última geração, como veículos autônomos, cidades inteligentes e Indústria 4.0, usando a mesma infraestrutura que dá suporte aos aplicativos empresariais existentes.

Cópias de instantâneos da NetApp

Uma cópia do NetApp Snapshot é uma imagem somente leitura e pontual de um volume. A imagem consome espaço de armazenamento mínimo e gera sobrecarga de desempenho insignificante porque ela registra apenas alterações em arquivos criados desde a última cópia do Snapshot, conforme ilustrado na figura a seguir.

As cópias de instantâneos devem sua eficiência à tecnologia de virtualização de armazenamento ONTAP , o Write Anywhere File Layout (WAFL). Assim como um banco de dados, o WAFL usa metadados para apontar para blocos de dados reais no disco. Mas, diferentemente de um banco de dados, o WAFL não substitui blocos existentes. Ele grava dados atualizados em um novo bloco e altera os metadados. É porque o ONTAP faz referência a metadados quando cria uma cópia do Snapshot, em vez de copiar blocos de dados, que as cópias do Snapshot são tão eficientes. Isso elimina o tempo de busca que outros sistemas levam para localizar os blocos a serem copiados, bem como o custo de fazer a cópia em si.

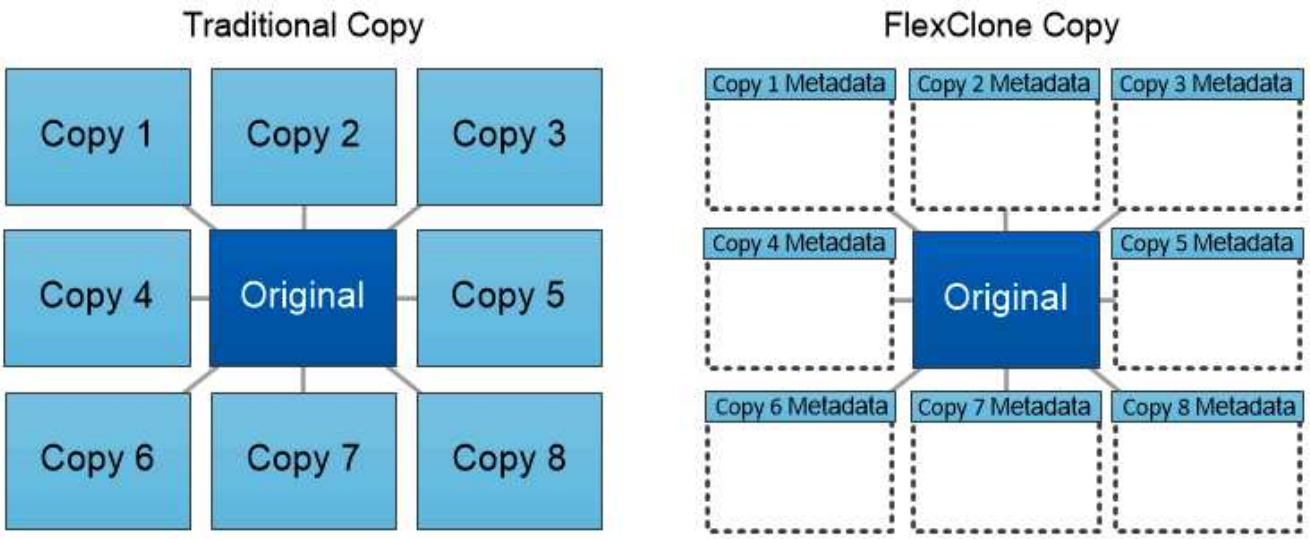
Você pode usar uma cópia de Snapshot para recuperar arquivos individuais ou LUNs ou para restaurar todo o conteúdo de um volume. O ONTAP compara as informações do ponteiro na cópia do Snapshot com os dados no disco para reconstruir o objeto ausente ou danificado, sem tempo de inatividade ou custo significativo de desempenho.



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

Tecnologia NetApp FlexClone

A tecnologia NetApp FlexClone faz referência a metadados de instantâneo para criar cópias graváveis e pontuais de um volume. As cópias compartilham blocos de dados com seus pais, não consumindo nenhum armazenamento, exceto o necessário para metadados até que as alterações sejam gravadas na cópia, conforme ilustrado na figura a seguir. Enquanto cópias tradicionais podem levar minutos ou até horas para serem criadas, o software FlexClone permite que você copie até os maiores conjuntos de dados quase instantaneamente. Isso o torna ideal para situações em que você precisa de várias cópias de conjuntos de dados idênticos (um espaço de trabalho de desenvolvimento, por exemplo) ou cópias temporárias de um conjunto de dados (testar um aplicativo em um conjunto de dados de produção).



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

Tecnologia de replicação de dados NetApp SnapMirror

O software NetApp SnapMirror é uma solução de replicação unificada, econômica e fácil de usar em toda a malha de dados. Ele replica dados em alta velocidade via LAN ou WAN. Ele oferece alta disponibilidade de dados e replicação rápida de dados para aplicativos de todos os tipos, incluindo aplicativos críticos de negócios em ambientes virtuais e tradicionais. Quando você replica dados para um ou mais sistemas de armazenamento NetApp e atualiza continuamente os dados secundários, seus dados são mantidos atualizados e estão disponíveis sempre que você precisar deles. Não são necessários servidores de replicação externos. Veja a figura a seguir para ver um exemplo de uma arquitetura que aproveita a tecnologia SnapMirror .

O software SnapMirror aproveita a eficiência do armazenamento NetApp ONTAP enviando apenas blocos alterados pela rede. O software SnapMirror também usa compactação de rede integrada para acelerar as transferências de dados e reduzir a utilização da largura de banda da rede em até 70%. Com a tecnologia SnapMirror , você pode aproveitar um fluxo fino de dados de replicação para criar um único repositório que mantém tanto o espelho ativo quanto as cópias anteriores de um ponto no tempo, reduzindo o tráfego de rede em até 50%.

Cópia e sincronização do NetApp BlueXP

"BlueXP Copiar e Sincronizar"é um serviço da NetApp para sincronização de dados rápida e segura. Se você precisa transferir arquivos entre compartilhamentos de arquivos NFS ou SMB locais, NetApp StorageGRID, NetApp ONTAP S3, Google Cloud NetApp Volumes, Azure NetApp Files, AWS S3, AWS EFS, Azure Blob, Google Cloud Storage ou IBM Cloud Object Storage, o BlueXP Copy and Sync move os arquivos para onde você precisa de forma rápida e segura.

Após seus dados serem transferidos, eles estarão totalmente disponíveis para uso tanto na origem quanto no destino. O BlueXP Copy and Sync pode sincronizar dados sob demanda quando uma atualização é acionada ou sincronizar dados continuamente com base em uma programação predefinida. De qualquer forma, o BlueXP Copy and Sync move apenas os deltas, minimizando o tempo e o dinheiro gastos na replicação de dados.

O BlueXP Copy and Sync é uma ferramenta de software como serviço (SaaS) extremamente simples de

configurar e usar. As transferências de dados acionadas pelo BlueXP Copy and Sync são realizadas por corretores de dados. Os corretores de dados BlueXP Copy and Sync podem ser implantados na AWS, Azure, Google Cloud Platform ou no local.

NetApp XCP

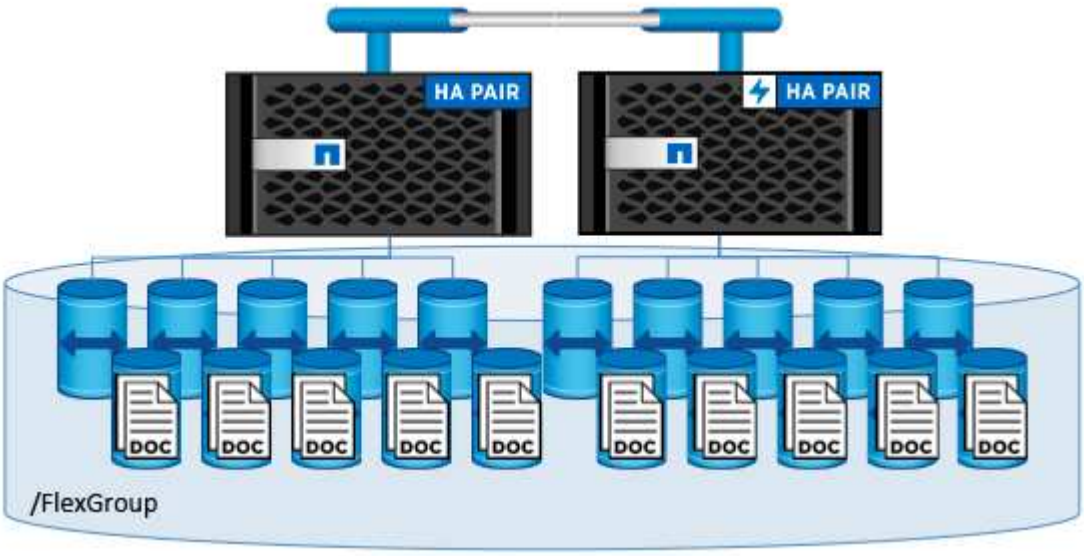
"NetApp XCP"é um software baseado em cliente para migrações de dados de qualquer para NetApp e de NetApp para NetApp e insights do sistema de arquivos. O XCP foi projetado para escalar e atingir o desempenho máximo utilizando todos os recursos do sistema disponíveis para lidar com conjuntos de dados de alto volume e migrações de alto desempenho. O XCP ajuda você a obter visibilidade completa do sistema de arquivos com a opção de gerar relatórios.

Volumes do NetApp ONTAP FlexGroup

Um conjunto de dados de treinamento pode ser uma coleção de potencialmente bilhões de arquivos. Os arquivos podem incluir texto, áudio, vídeo e outras formas de dados não estruturados que devem ser armazenados e processados para serem lidos em paralelo. O sistema de armazenamento deve armazenar um grande número de arquivos pequenos e deve ler esses arquivos em paralelo para E/S sequencial e aleatória.

Um volume FlexGroup é um único namespace que compreende vários volumes de membros constituintes, conforme mostrado na figura a seguir. Do ponto de vista de um administrador de armazenamento, um volume FlexGroup é gerenciado e age como um FlexVol volume NetApp FlexVol. Os arquivos em um volume FlexGroup são alocados para volumes de membros individuais e não são distribuídos entre volumes ou nós. Eles permitem os seguintes recursos:

- Os volumes FlexGroup fornecem vários petabytes de capacidade e baixa latência previsível para cargas de trabalho com muitos metadados.
- Eles suportam até 400 bilhões de arquivos no mesmo namespace.
- Eles oferecem suporte a operações paralelizadas em cargas de trabalho NAS em CPUs, nós, agregados e volumes FlexVol constituintes.



Arquitetura

Esta solução não depende de hardware específico. A solução é compatível com qualquer dispositivo de armazenamento físico, instância definida por software ou serviço de nuvem da NetApp suportado pelo NetApp Trident. Exemplos incluem um sistema de armazenamento NetApp AFF , Amazon FSx ONTAP, Azure NetApp Files, Google Cloud NetApp Volumes ou uma instância NetApp Cloud Volumes ONTAP . Além disso, a solução pode ser implementada em qualquer cluster do Kubernetes, desde que a versão do Kubernetes usada seja compatível com o NetApp Trident e os outros componentes da solução que estão sendo implementados. Para obter uma lista de versões do Kubernetes suportadas pelo Trident, consulte "Documentação do Trident" . Consulte as tabelas a seguir para obter detalhes sobre os ambientes que foram usados para validar os vários componentes desta solução.

Ambiente de validação do Apache Airflow

Componente de software	Versão
Apache Airflow	2.0.1, implantado via"Gráfico do Apache Airflow Helm" 8.0.8
Kubernetes	1,18
NetApp Trident	21,01

Ambiente de validação JupyterHub

Componente de software	Versão
JupyterHub	4.1.5, implantado via"Gráfico do JupyterHub Helm" 3.3.7
Kubernetes	1,29
NetApp Trident	24,02

Ambiente de Validação MLflow

Componente de software	Versão
Fluxo de ML	2.14.1, implantado via"Gráfico Helm do MLflow" 1.4.12
Kubernetes	1,29
NetApp Trident	24,02

Ambiente de Validação Kubeflow

Componente de software	Versão
Fluxo de cubo	1.7, implantado via"implantarKF" 0.1.1

Componente de software	Versão
Kubernetes	1,26
NetApp Trident	23,07

Apoiar

A NetApp não oferece suporte empresarial para Apache Airflow, JupyterHub, MLflow, Kubeflow ou Kubernetes. Se você estiver interessado em uma plataforma MLOps totalmente suportada,"entre em contato com a NetApp" sobre soluções MLOps totalmente suportadas que a NetApp oferece em conjunto com parceiros.

Configuração do NetApp Trident

Exemplo de backends Trident para implantações NetApp AIPod

Antes de usar o Trident para provisionar dinamicamente recursos de armazenamento no seu cluster Kubernetes, você deve criar um ou mais Trident Backends. Os exemplos a seguir representam diferentes tipos de backends que você pode querer criar se estiver implantando componentes desta solução em um"NetApp AIPod" . Para obter mais informações sobre backends e, por exemplo, backends para outras plataformas/ambientes, consulte o"Documentação do Trident" .

1. A NetApp recomenda a criação de um Trident Backend habilitado para FlexGroup para seu AIPod.

   Os comandos de exemplo a seguir mostram a criação de um Trident Backend habilitado para FlexGroup para uma máquina virtual de armazenamento AIPod (SVM). Este Backend usa o `ontap-nas-flexgroup` driver de armazenamento. O ONTAP suporta dois tipos principais de volume de dados: FlexVol e FlexGroup. Os volumes FlexVol têm tamanho limitado (no momento em que este artigo foi escrito, o tamanho máximo depende da implantação específica). Os volumes FlexGroup , por outro lado, podem ser dimensionados linearmente para até 20 PB e 400 bilhões de arquivos, fornecendo um único namespace que simplifica muito o gerenciamento de dados. Portanto, os volumes FlexGroup são ideais para cargas de trabalho de IA e ML que dependem de grandes quantidades de dados.

   Se você estiver trabalhando com uma pequena quantidade de dados e quiser usar volumes FlexVol em vez de volumes FlexGroup , você pode criar Trident Backends que usam o `ontap-nas` driver de armazenamento em vez do `ontap-nas-flexgroup` driver de armazenamento.

```
$ 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. A NetApp também recomenda a criação de um Trident Backend habilitado para FlexVol . Você pode querer usar volumes FlexVol para hospedar aplicativos persistentes, armazenar resultados, saídas, informações de depuração e assim por diante. Se quiser usar volumes FlexVol , você deve criar um ou mais Trident Backends habilitados para FlexVol . Os comandos de exemplo a seguir mostram a criação de um único Trident Backend 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 |
+-------------------------+--------------------
+--------------------------------------+--------+---------+
```

Exemplo de classes de armazenamento do Kubernetes para implantações do NetApp AIPod

Antes de usar o Trident para provisionar dinamicamente recursos de armazenamento no seu cluster Kubernetes, você deve criar uma ou mais Kubernetes StorageClasses. Os exemplos a seguir representam diferentes tipos de StorageClasses que você pode querer criar se estiver implantando componentes desta solução em um"NetApp AIPod" .

Para obter mais informações sobre StorageClasses e, por exemplo, StorageClasses para outras plataformas/ambientes, consulte o"Documentação do Trident" .

1. A NetApp recomenda a criação de um StorageClass para o FlexGroup-enabled Trident Backend que você criou na seção"Exemplo de backends Trident para implantações NetApp AIPod" , passo 1. Os comandos de exemplo a seguir mostram a criação de várias StorageClasses que correspondem ao Backend de exemplo que foi criado na seção"Exemplo de backends Trident para implantações NetApp AIPod" , passo 1 - aquele que utiliza"NFS sobre RDMA" e um que não.

   Para que um volume persistente não seja excluído quando o PersistentVolumeClaim (PVC) correspondente for excluído, o exemplo a seguir usa um `reclaimPolicy` valor de `Retain` . Para mais informações sobre o `reclaimPolicy` campo, veja o oficial "Documentação do Kubernetes" .

   Observação: os StorageClasses de exemplo a seguir usam um tamanho máximo de transferência de 262144. Para usar esse tamanho máximo de transferência, você deve configurar o tamanho máximo de transferência no seu sistema ONTAP adequadamente. Consulte o"Documentação do ONTAP" para mais detalhes.

   Observação: para usar o NFS sobre RDMA, você deve configurar o NFS sobre RDMA no seu sistema ONTAP . Consulte o"Documentação do ONTAP" para mais detalhes.

   Observação: no exemplo a seguir, um Backend específico é especificado no campo storagePool no arquivo de definição 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. A NetApp também recomenda a criação de um StorageClass que corresponda ao FlexVol-enabled Trident Backend que você criou na seção"Exemplo de backends Trident para implantações de AIPod" , passo 2. Os comandos de exemplo a seguir mostram a criação de uma única StorageClass para volumes FlexVol .

   Observação: no exemplo a seguir, um Backend específico não é especificado no campo storagePool no arquivo de definição StorageClass. Quando você usa o Kubernetes para administrar volumes usando este StorageClass, o Trident tenta usar qualquer backend disponível que use o `ontap-nas` motorista.

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

Apache Airflow

Implantação do Apache Airflow

Esta seção descreve as tarefas que você deve concluir para implantar o Airflow no seu cluster Kubernetes.

É possível implantar o Airflow em outras plataformas além do Kubernetes. A implantação do Airflow em plataformas diferentes do Kubernetes está fora do escopo desta solução.

Pré-requisitos

Antes de executar o exercício de implantação descrito nesta seção, presumimos que você já tenha executado as seguintes tarefas:

1. Você já tem um cluster Kubernetes funcional.
2. Você já instalou e configurou o NetApp Trident no seu cluster Kubernetes. Para mais detalhes sobre o Trident, consulte o"Documentação do Trident" .

Instalar o Helm

O Airflow é implantado usando o Helm, um gerenciador de pacotes popular para Kubernetes. Antes de implantar o Airflow, você deve instalar o Helm no host de salto de implantação. Para instalar o Helm no host de salto de implantação, siga as instruções "instruções de instalação" na documentação oficial do Helm.

Definir classe de armazenamento padrão do Kubernetes

Antes de implantar o Airflow, você deve designar um StorageClass padrão dentro do seu cluster Kubernetes. O processo de implantação do Airflow tenta provisionar novos volumes persistentes usando o StorageClass

padrão. Se nenhuma StorageClass for designada como StorageClass padrão, a implantação falhará. Para designar uma StorageClass padrão em seu cluster, siga as instruções descritas no"Implantação do Kubeflow" seção. Se você já designou uma StorageClass padrão dentro do seu cluster, pode pular esta etapa.

Use o Helm para implantar o Airflow

Para implantar o Airflow no seu cluster Kubernetes usando o Helm, execute as seguintes tarefas no host de salto de implantação:

1. Implante o Airflow usando o Helm seguindo o "instruções de implantação" para o gráfico oficial do Airflow no Artifact Hub. Os comandos de exemplo a seguir mostram a implantação do Airflow usando o Helm. Modifique, adicione e/ou remova valores no `custom- values.yaml` arquivo conforme necessário, dependendo do seu ambiente e da configuração desejada.

```
$ 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
```

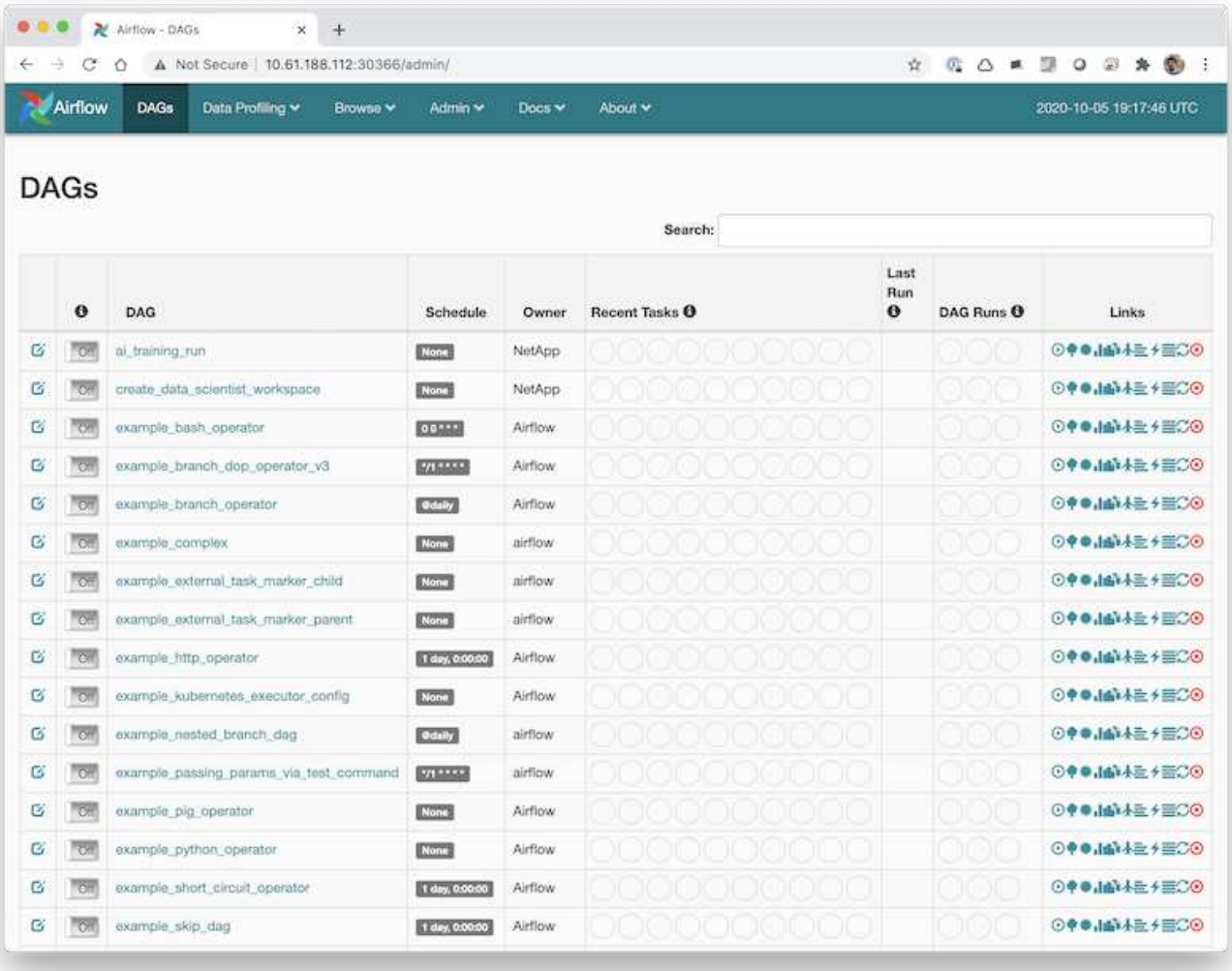
2. Confirme se todos os pods Airflow estão funcionando. Pode levar alguns minutos para que todos os pods sejam iniciados.

```
$ 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. Obtenha a URL do serviço web do Airflow seguindo as instruções impressas no console quando você implantou o Airflow usando o Helm na etapa 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 se você consegue acessar o serviço web Airflow.



Use o NetApp DataOps Toolkit com o Airflow

O "Kit de ferramentas NetApp DataOps para Kubernetes" pode ser usado em conjunto com o Airflow. Usar o NetApp DataOps Toolkit com o Airflow permite incorporar operações de gerenciamento de dados do NetApp , como criar snapshots e clones, em fluxos de trabalho automatizados orquestrados pelo Airflow.

Consulte o "Exemplos de fluxo de ar" seção no repositório GitHub do NetApp DataOps Toolkit para obter detalhes sobre como usar o kit de ferramentas com o Airflow.

JupyterHub

Implantação do JupyterHub

Esta seção descreve as tarefas que você deve concluir para implantar o JupyterHub no seu cluster Kubernetes.

É possível implantar o JupyterHub em plataformas diferentes do Kubernetes. A implantação do JupyterHub em plataformas diferentes do Kubernetes está fora do escopo desta solução.

Pré-requisitos

Antes de executar o exercício de implantação descrito nesta seção, presumimos que você já tenha executado as seguintes tarefas:

1. Você já tem um cluster Kubernetes funcional.
2. Você já instalou e configurou o NetApp Trident no seu cluster Kubernetes. Para mais detalhes sobre o Trident, consulte o"Documentação do Trident" .

Instalar o Helm

O JupyterHub é implantado usando o Helm, um gerenciador de pacotes popular para Kubernetes. Antes de implantar o JupyterHub, você deve instalar o Helm no seu nó de controle do Kubernetes. Para instalar o Helm, siga as instruções "instruções de instalação" na documentação oficial do Helm.

Definir classe de armazenamento padrão do Kubernetes

Antes de implantar o JupyterHub, você deve designar um StorageClass padrão dentro do seu cluster Kubernetes. Para designar uma StorageClass padrão em seu cluster, siga as instruções descritas no"Implantação do Kubeflow" seção. Se você já designou uma StorageClass padrão dentro do seu cluster, pode pular esta etapa.

Implantar o JupyterHub

Depois de concluir as etapas acima, você estará pronto para implantar o JupyterHub. A implantação do JupyterHub requer as seguintes etapas:

Configurar a implantação do JupyterHub

Antes da implantação, é uma boa prática otimizar a implantação do JupyterHub para seu respectivo ambiente. Você pode criar um arquivo **config.yaml** e utilizá-lo durante a implantação usando o gráfico Helm.

Um exemplo de arquivo **config.yaml** pode ser encontrado em https://github.com/jupyterhub/zero-to-jupyterhub-k8s/blob/HEAD/jupyterhub/values.yaml

Neste arquivo config.yaml, você pode definir o parâmetro **(singleuser.storage.dynamic.storageClass)** para o NetApp Trident StorageClass. Esta é a classe de armazenamento que será usada para provisionar os volumes para espaços de trabalho de usuários individuais.

Adicionando volumes compartilhados

Se quiser usar um volume compartilhado para todos os usuários do JupyterHub, você pode ajustar seu **config.yaml** adequadamente. Por exemplo, se você tiver um PersistentVolumeClaim compartilhado chamado jupyterhub-shared-volume, você pode montá-lo como /home/shared em todos os pods de usuário como:

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

Esta é uma etapa opcional, você pode ajustar esses parâmetros conforme suas necessidades.

Implantar o JupyterHub com o Helm Chart

Informe ao Helm sobre o repositório de gráficos do JupyterHub Helm.

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

Isso deve mostrar uma saída como esta:

```
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!□
```

Agora instale o gráfico configurado pelo seu config.yaml executando este comando no diretório que contém seu config.yaml:

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

Neste exemplo:

<helm-release-name> é definido como my-jupyterhub, que será o nome da sua versão do JupyterHub. <k8s-namespace> é definido como my-namespace, que é o namespace onde você deseja instalar o JupyterHub. O sinalizador --create-namespace é usado para criar o namespace se ele ainda não existir. O sinalizador --values especifica o arquivo config.yaml que contém as opções de configuração desejadas.

Verificar implantação

Enquanto a etapa 2 estiver em execução, você poderá ver os pods sendo criados a partir do seguinte comando:

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

Aguarde até que o hub e o pod proxy entrem no estado Em execução.

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

Acesse o JupyterHub

Encontre o IP que podemos usar para acessar o JupyterHub. Execute o seguinte comando até que o EXTERNAL-IP do serviço proxy-public esteja disponível, como na saída de exemplo.

> Usamos o serviço NodePort em nosso arquivo config.yaml, você pode ajustar para seu ambiente com base em sua configuração (por exemplo, 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 usar o JupyterHub, insira o IP externo do serviço proxy-público em um navegador.

Use o NetApp DataOps Toolkit com o JupyterHub

O "Kit de ferramentas NetApp DataOps para Kubernetes" pode ser usado em conjunto com o JupyterHub. O uso do NetApp DataOps Toolkit com o JupyterHub permite que os usuários finais criem instantâneos de volume para backup do espaço de trabalho e/ou rastreabilidade do conjunto de dados para o modelo diretamente de um Jupyter Notebook.

Configuração inicial

Antes de poder usar o DataOps Toolkit com o JupyterHub, você deve conceder permissões apropriadas à conta de serviço do Kubernetes que o JupyterHub atribui aos pods individuais do Jupyter Notebook Server. O JupyterHub usa a conta de serviço especificada pelo `singleuser.serviceAccountName` variável no seu arquivo de configuração do gráfico do JupyterHub Helm.

Criar função de cluster para o DataOps Toolkit

Primeiro, crie uma função de cluster chamada 'netapp-dataops' que tenha as permissões de API do Kubernetes necessárias para criar instantâneos de volume.

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

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

Atribuir função de cluster à conta de serviço do servidor de notebook

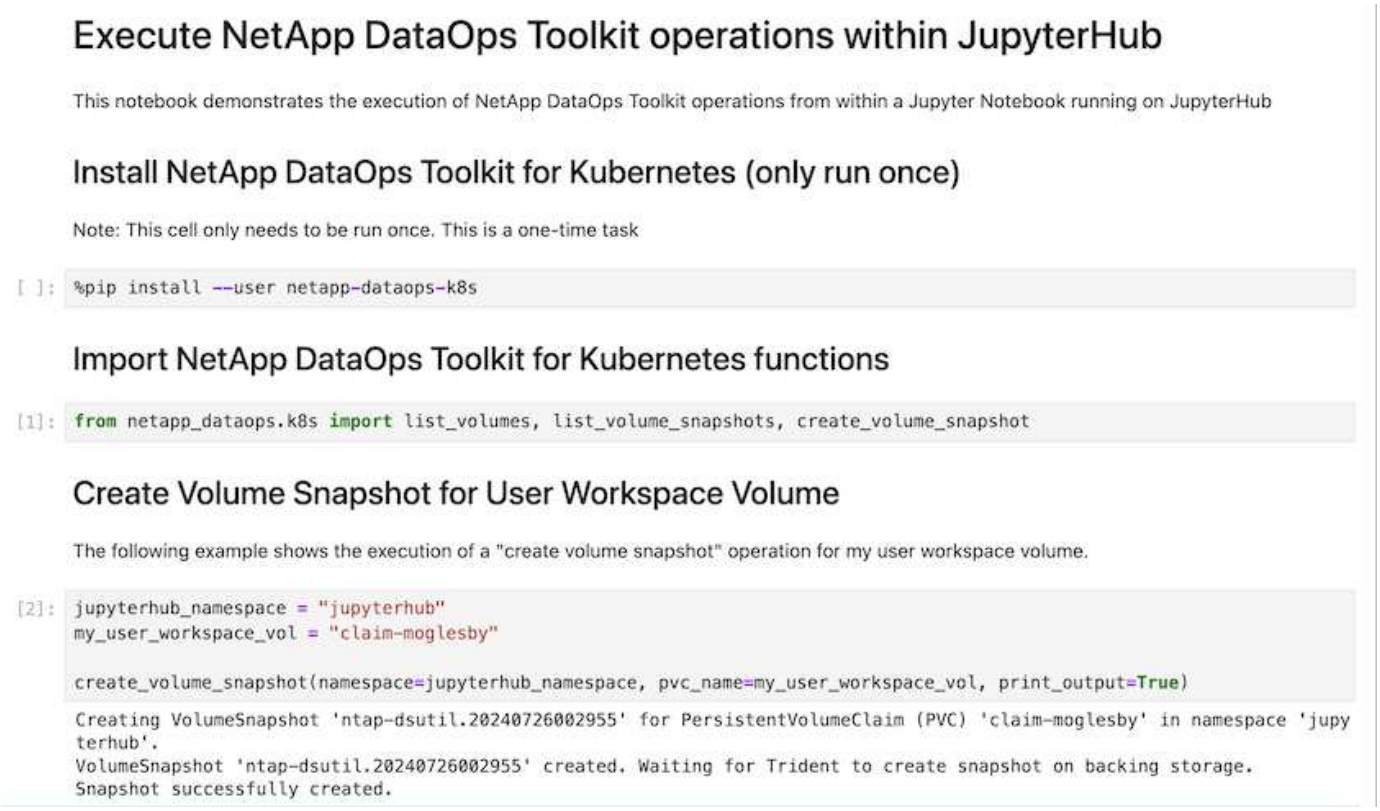
Crie uma associação de função que atribua a função de cluster 'netapp-dataops-snapshots' à conta de serviço apropriada no namespace apropriado. Por exemplo, se você instalou o JupyterHub no namespace 'jupyterhub' e especificou a conta de serviço 'padrão' por meio do `singleuser.serviceAccountName` variável, você atribuiria a função de cluster 'netapp-dataops-snapshots' à conta de serviço 'default' no namespace 'jupyterhub', conforme mostrado no exemplo a seguir.

```
$ 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
```

Crie instantâneos de volume no Jupyter Notebook

Agora, os usuários do JupyterHub podem usar o NetApp DataOps Toolkit para criar snapshots de volume diretamente de um Jupyter Notebook, conforme mostrado no exemplo a seguir.



Ingerir dados no JupyterHub com o NetApp SnapMirror

O NetApp SnapMirror é uma tecnologia de replicação que permite replicar dados entre sistemas de armazenamento NetApp . O SnapMirror pode ser usado para ingerir dados de ambientes remotos no JupyterHub.

Exemplo de fluxo de trabalho e demonstração

Consulte"esta postagem do blog Tech ONTAP" para um exemplo detalhado de fluxo de trabalho e demonstração do uso do NetApp SnapMirror para ingerir dados no JupyterHub.

Fluxo de ML

Implantação do MLflow

Esta seção descreve as tarefas que você deve concluir para implantar o MLflow no seu cluster Kubernetes.

> É possível implantar o MLflow em outras plataformas além do Kubernetes. A implantação do MLflow em plataformas diferentes do Kubernetes está fora do escopo desta solução.

Pré-requisitos

Antes de executar o exercício de implantação descrito nesta seção, presumimos que você já tenha executado as seguintes tarefas:

1. Você já tem um cluster Kubernetes funcional.
2. Você já instalou e configurou o NetApp Trident no seu cluster Kubernetes. Para mais detalhes sobre o Trident, consulte o"Documentação do Trident" .

Instalar o Helm

O MLflow é implantado usando o Helm, um gerenciador de pacotes popular para Kubernetes. Antes de implantar o MLflow, você deve instalar o Helm no seu nó de controle do Kubernetes. Para instalar o Helm, siga as instruções "instruções de instalação" na documentação oficial do Helm.

Definir classe de armazenamento padrão do Kubernetes

Antes de implantar o MLflow, você deve designar uma StorageClass padrão dentro do seu cluster Kubernetes. Para designar uma StorageClass padrão em seu cluster, siga as instruções descritas no"Implantação do Kubeflow" seção. Se você já designou uma StorageClass padrão dentro do seu cluster, pode pular esta etapa.

Implantar MLflow

Depois que os pré-requisitos forem atendidos, você poderá começar a implantação do MLflow usando o gráfico do Helm.

Configurar a implantação do gráfico do MLflow Helm.

Antes de implantar o MLflow usando o gráfico Helm, podemos configurar a implantação para usar a classe de armazenamento NetApp Trident e alterar outros parâmetros para atender às nossas necessidades usando um

arquivo **config.yaml**. Um exemplo do arquivo **config.yaml** pode ser encontrado em: https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml

> Você pode definir o Trident storageClass no parâmetro **global.defaultStorageClass** no arquivo config.yaml (por exemplo, storageClass: "ontap-flexvol").

Instalando o Helm Chart

O gráfico Helm pode ser instalado com o arquivo **config.yaml** personalizado para MLflow usando o seguinte comando:

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

> O comando implanta o MLflow no cluster Kubernetes na configuração personalizada por meio do arquivo **config.yaml** fornecido. O MLflow é implantado no namespace fornecido e um nome de versão aleatório é fornecido via kubernetes para a versão.

Verificar implantação

Após a implantação do gráfico Helm, você pode verificar se o serviço está acessível usando:

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

> Substitua **jupyterhub** pelo namespace que você usou durante a implantação.

Você deverá ver os seguintes serviços:

```
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 o arquivo config.yaml para usar o serviço NodePort para acessar o MLflow na porta 30002.

Acesse o MLflow

Depois que todos os serviços relacionados ao MLflow estiverem ativos e em execução, você poderá acessá-lo

usando o endereço IP NodePort ou LoadBalancer fornecido (por exemplo `http://10.61.181.109:30002`)

Rastreabilidade de conjunto de dados para modelo com NetApp e MLflow

O "Kit de ferramentas NetApp DataOps para Kubernetes" pode ser usado em conjunto com os recursos de rastreamento de experimentos do MLflow para implementar a rastreabilidade do conjunto de dados para o modelo ou do espaço de trabalho para o modelo.

Para implementar a rastreabilidade de conjunto de dados para modelo ou de espaço de trabalho para modelo, basta criar um instantâneo do seu conjunto de dados ou volume de espaço de trabalho usando o DataOps Toolkit como parte da sua execução de treinamento, conforme mostrado no seguinte trecho de código de exemplo. Este código salvará o nome do volume de dados e o nome do instantâneo como tags associadas à execução de treinamento específica que você está registrando no seu servidor de rastreamento de experimentos do 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)

    ...
```

Fluxo de cubo

Implantação do Kubeflow

Esta seção descreve as tarefas que você deve concluir para implantar o Kubeflow no seu cluster Kubernetes.

Pré-requisitos

Antes de executar o exercício de implantação descrito nesta seção, presumimos que você já tenha executado as seguintes tarefas:

1. Você já tem um cluster Kubernetes funcional e está executando uma versão do Kubernetes compatível com a versão do Kubeflow que pretende implantar. Para obter uma lista de versões do Kubernetes com suporte, consulte as dependências da sua versão do Kubeflow no"documentação oficial do Kubeflow" .
2. Você já instalou e configurou o NetApp Trident no seu cluster Kubernetes. Para mais detalhes sobre o Trident, consulte o"Documentação do Trident" .

Definir classe de armazenamento padrão do Kubernetes

Antes de implantar o Kubeflow, recomendamos designar um StorageClass padrão no seu cluster Kubernetes. O processo de implantação do Kubeflow pode tentar provisionar novos volumes persistentes usando o StorageClass padrão. Se nenhuma StorageClass for designada como StorageClass padrão, a implantação poderá falhar. Para designar um StorageClass padrão dentro do seu cluster, execute a seguinte tarefa no host de salto de implantação. Se você já designou uma StorageClass padrão dentro do seu cluster, pode pular esta etapa.

1. Designe uma das suas StorageClasses existentes como a StorageClass padrão. Os comandos de exemplo a seguir mostram a designação de uma StorageClass denominada `ontap-ai-flexvols-retain` como StorageClass padrão.

O `ontap-nas-flexgroup` O tipo Trident Backend tem um tamanho mínimo de PVC que é bastante grande. Por padrão, o Kubeflow tenta provisionar PVCs com apenas alguns GBs de tamanho. Portanto, você não deve designar uma StorageClass que utilize o `ontap-nas-flexgroup` Tipo de backend como StorageClass padrão para fins de implantação do 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
```

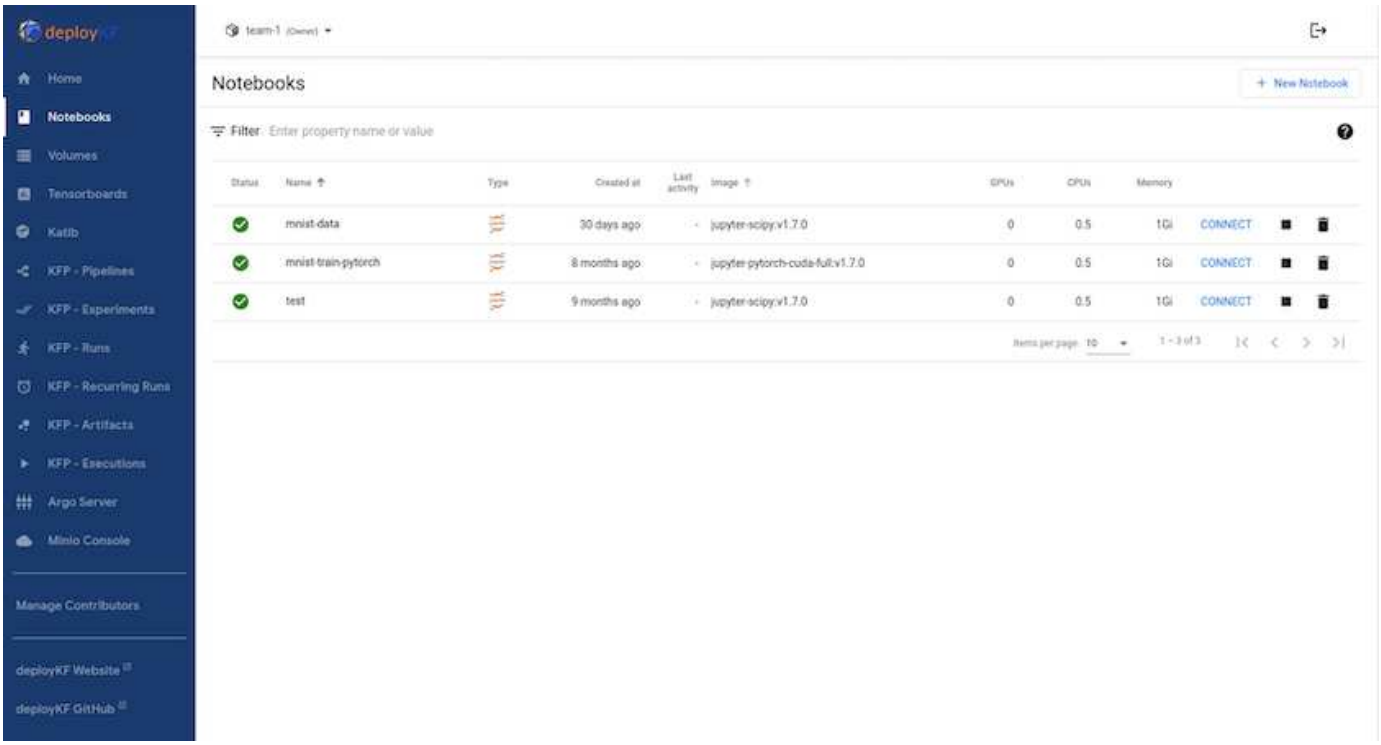
Opções de implantação do Kubeflow

Há muitas opções diferentes para implantar o Kubeflow. Consulte o"documentação oficial do Kubeflow" para obter uma lista de opções de implantação e escolher a opção mais adequada às suas necessidades.

Para fins de validação, implantamos o Kubeflow 1.7 usando"implantarKF" 0.1.1.

Provisionar um espaço de trabalho do Jupyter Notebook para uso de cientistas de dados ou desenvolvedores

O Kubeflow é capaz de provisionar rapidamente novos servidores Jupyter Notebook para atuarem como espaços de trabalho de cientistas de dados. Para obter mais informações sobre Jupyter Notebooks no contexto do Kubeflow, consulte o "documentação oficial do Kubeflow" .



Use o NetApp DataOps Toolkit com o Kubeflow

O "Kit de ferramentas de ciência de dados da NetApp para Kubernetes" pode ser usado em conjunto com o Kubeflow. O uso do NetApp Data Science Toolkit com o Kubeflow oferece os seguintes benefícios:

- Cientistas de dados podem executar operações avançadas de gerenciamento de dados do NetApp , como criar snapshots e clones, diretamente de um Jupyter Notebook.
- Operações avançadas de gerenciamento de dados do NetApp , como criação de snapshots e clones, podem ser incorporadas em fluxos de trabalho automatizados usando a estrutura do Kubeflow Pipelines.

Consulte o "Exemplos de Kubeflow" seção no repositório GitHub do NetApp Data Science Toolkit para obter detalhes sobre como usar o kit de ferramentas com o Kubeflow.

Exemplo de fluxo de trabalho - Treinar um modelo de reconhecimento de imagem usando o Kubeflow e o NetApp DataOps Toolkit

Esta seção descreve as etapas envolvidas no treinamento e na implantação de uma rede neural para reconhecimento de imagem usando o Kubeflow e o NetApp DataOps Toolkit. O objetivo é servir como exemplo para mostrar um trabalho de treinamento que incorpora armazenamento NetApp .

Pré-requisitos

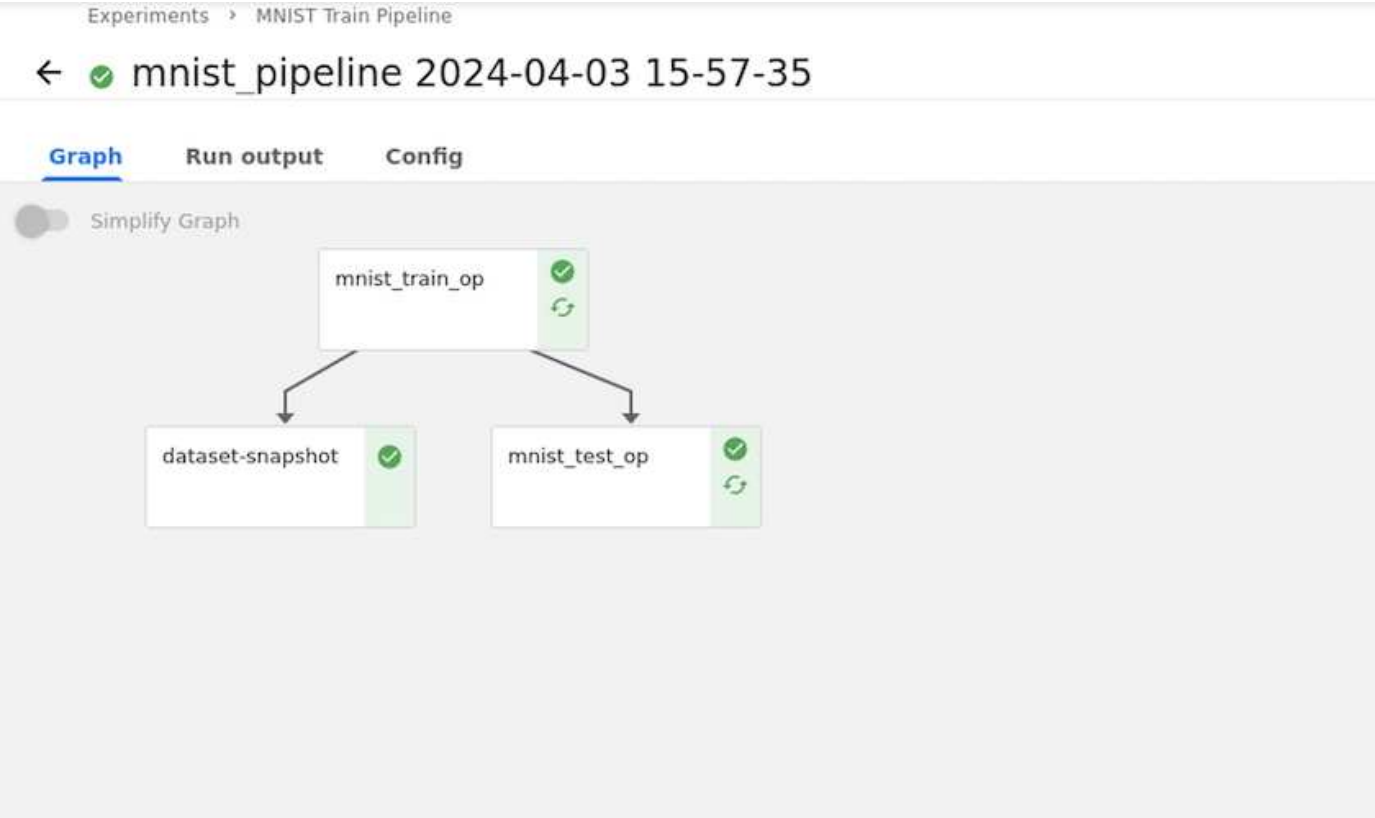
Crie um Dockerfile com as configurações necessárias para usar nas etapas de treinamento e teste dentro do pipeline do Kubeflow. Aqui está um exemplo de um Dockerfile -

```
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"]
```

Dependendo de suas necessidades, instale todas as bibliotecas e pacotes necessários para executar o programa. Antes de treinar o modelo de Machine Learning, presume-se que você já tenha uma implantação funcional do Kubeflow.

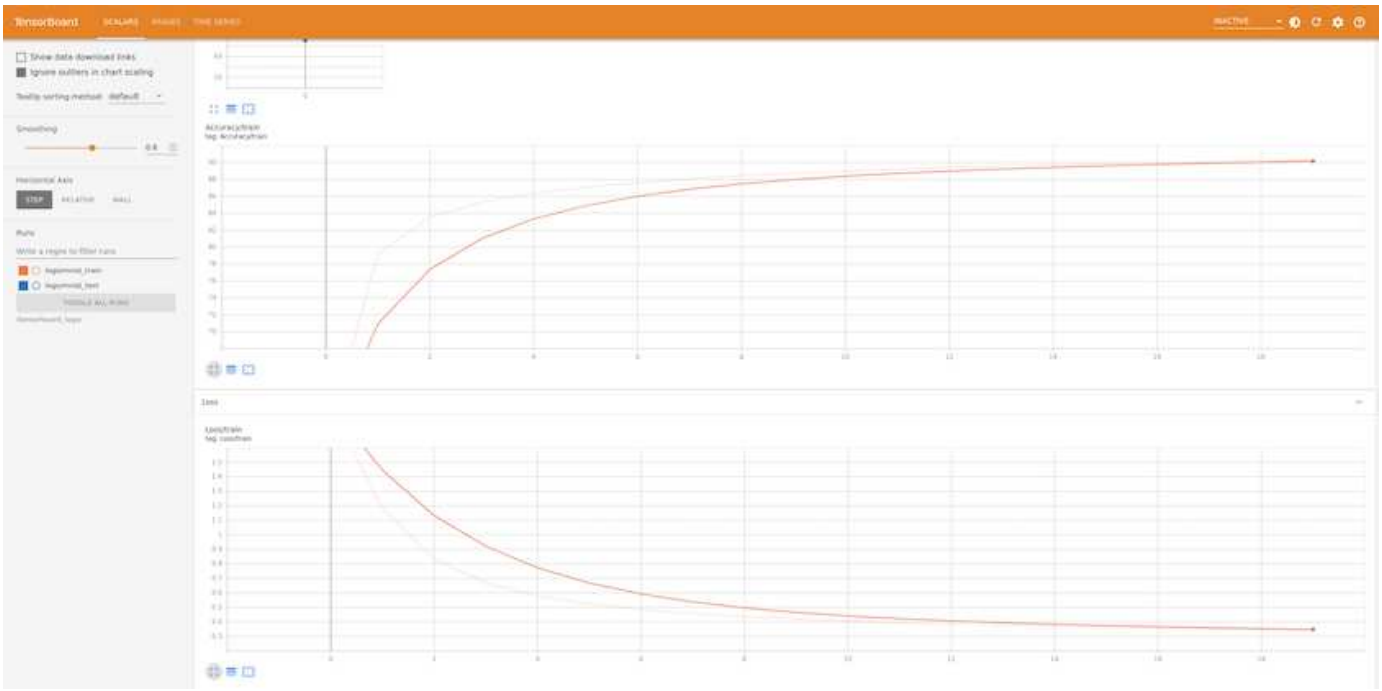
Treine uma pequena NN em dados MNIST usando PyTorch e pipelines Kubeflow

Usamos o exemplo de uma pequena Rede Neural treinada em dados MNIST. O conjunto de dados MNIST consiste em imagens manuscritas de dígitos de 0 a 9. As imagens têm 28x28 pixels de tamanho. O conjunto de dados é dividido em 60.000 imagens de trens e 10.000 imagens de validação. A rede neural usada neste experimento é uma rede feedforward de duas camadas. O treinamento é executado usando o Kubeflow Pipelines. Consulte a documentação "aqui" para maiores informações. Nosso pipeline do Kubeflow incorpora a imagem do Docker da seção Pré-requisitos.



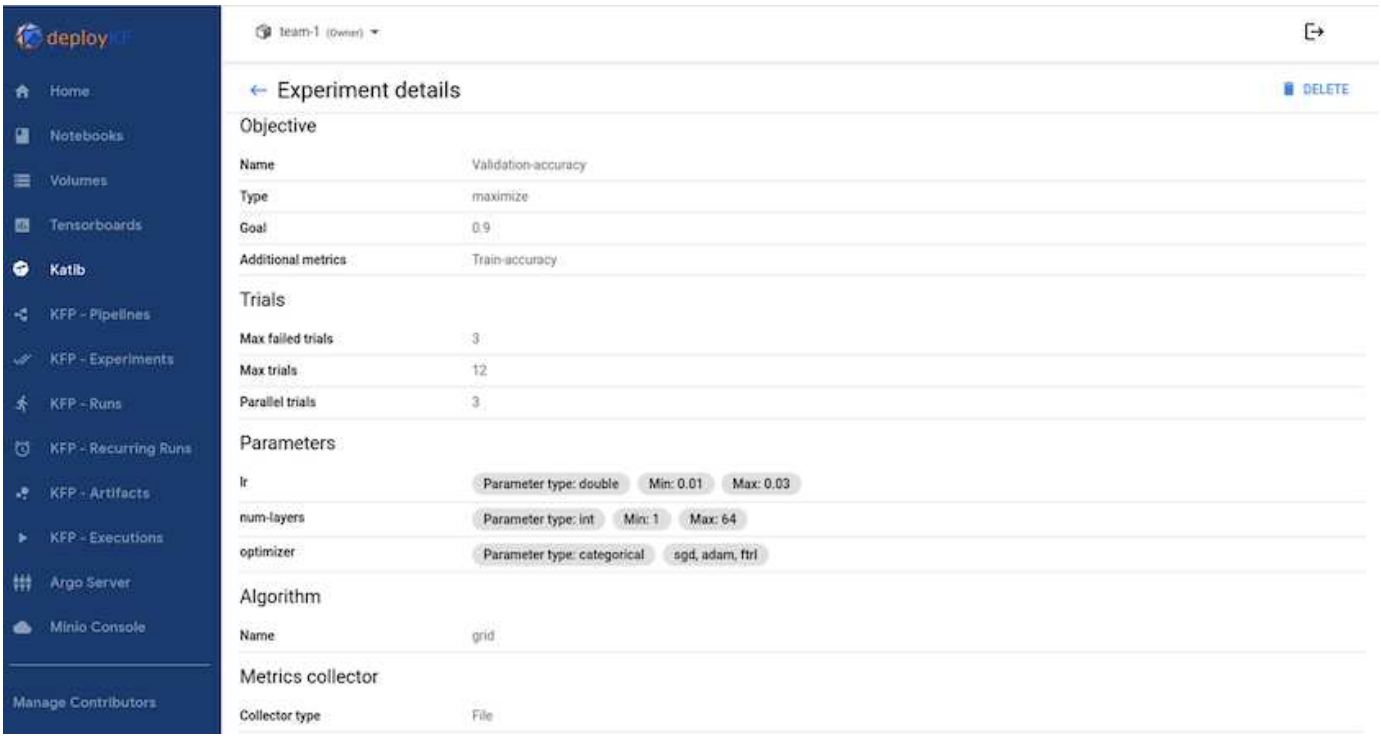
Visualize resultados usando o Tensorboard

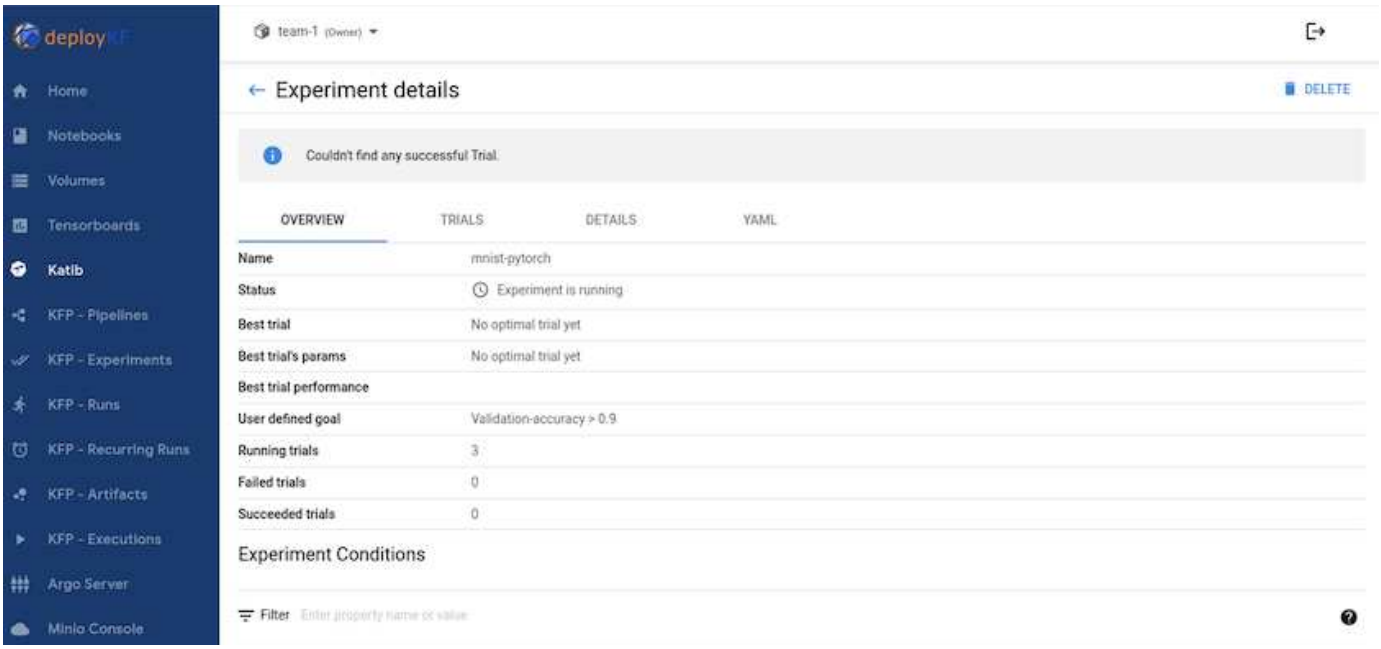
Depois que o modelo estiver treinado, podemos visualizar os resultados usando o Tensorboard. "Tensorboard" está disponível como um recurso no Painel do Kubeflow. Você pode criar um tensorboard personalizado para seu trabalho. O exemplo abaixo mostra o gráfico da precisão do treinamento versus número de épocas e perda de treinamento versus número de épocas.



Experimento com hiperparâmetros usando Katib

"Katib"é uma ferramenta dentro do Kubeflow que pode ser usada para experimentar os hiperparâmetros do modelo. Para criar um experimento, defina primeiro uma métrica/meta desejada. Geralmente essa é a precisão do teste. Depois que a métrica for definida, escolha os hiperparâmetros com os quais você gostaria de brincar (otimizador/taxa de aprendizado/número de camadas). Katib faz uma varredura de hiperparâmetros com os valores definidos pelo usuário para encontrar a melhor combinação de parâmetros que satisfaçam a métrica desejada. Você pode definir esses parâmetros em cada seção da interface do usuário. Como alternativa, você pode definir um arquivo **YAML** com as especificações necessárias. Abaixo está uma ilustração de um experimento Katib -





Use instantâneos do NetApp para salvar dados para rastreabilidade

Durante o treinamento do modelo, podemos querer salvar um instantâneo do conjunto de dados de treinamento para rastreabilidade. Para fazer isso, podemos adicionar uma etapa de instantâneo ao pipeline, conforme mostrado abaixo. Para criar o snapshot, podemos usar o "Kit de ferramentas 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 o "Exemplo do NetApp DataOps Toolkit para Kubeflow" para maiores informações.

Exemplo de Operações Trident

Esta seção inclui exemplos de várias operações que você pode querer executar com o Trident.

Importar um volume existente

Se houver volumes existentes no seu sistema/plataforma de armazenamento NetApp que você deseja montar em contêineres dentro do seu cluster Kubernetes, mas que não estão vinculados a PVCs no cluster, você deverá importar esses volumes. Você pode usar a funcionalidade de importação de volumes do Trident para

importar esses volumes.

Os comandos de exemplo a seguir mostram a importação de um volume denominado `pb_fg_all` . Para mais informações sobre PVCs, consulte o "documentação oficial do Kubernetes" . Para obter mais informações sobre a funcionalidade de importação de volume, consulte o "Documentação do Trident" .

Um `accessModes` valor de `ReadOnlyMany` é especificado nos arquivos de especificação de PVC de exemplo. Para mais informações sobre o `accessMode` campo, veja o "documentação oficial do 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
```

Provisionar um novo volume

Você pode usar o Trident para provisionar um novo volume no seu sistema de armazenamento ou plataforma NetApp .

Provisionar um novo volume usando kubectl

Os comandos de exemplo a seguir mostram o provisionamento de um novo FlexVol volume usando kubectl.

Um `accessModes` valor de `ReadWriteMany` é especificado no seguinte arquivo de definição de PVC de exemplo. Para mais informações sobre o `accessMode` campo, veja o "documentação oficial do 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
```

Provisionar um novo volume usando o NetApp DataOps Toolkit

Você também pode usar o NetApp DataOps Toolkit for Kubernetes para provisionar um novo volume no seu sistema de armazenamento ou plataforma NetApp . O NetApp DataOps Toolkit para Kubernetes utiliza o Trident para provisionar volumes, mas simplifica o processo para o usuário. Consulte o"documentação" para mais detalhes.

Exemplo de trabalhos de alto desempenho para implantações de AIPod

Executar uma carga de trabalho de IA de nó único

Para executar uma tarefa de IA e ML de nó único no seu cluster Kubernetes, execute as seguintes tarefas no host de salto de implantação. Com o Trident, você pode tornar um volume de dados, potencialmente contendo petabytes de dados, acessível de forma rápida e fácil a uma carga de trabalho do Kubernetes. Para tornar esse volume de dados acessível de dentro de um pod do Kubernetes, basta especificar um PVC na definição do pod.

Esta seção pressupõe que você já tenha conteinerizado (no formato de contêiner do Docker) a carga de trabalho específica de IA e ML que está tentando executar no seu cluster Kubernetes.

1. Os comandos de exemplo a seguir mostram a criação de um trabalho do Kubernetes para uma carga de trabalho de benchmark do TensorFlow que usa o conjunto de dados ImageNet. Para obter mais informações sobre o conjunto de dados ImageNet, consulte o "Site ImageNet" .

   Este trabalho de exemplo solicita oito GPUs e, portanto, pode ser executado em um único nó de trabalho de GPU que apresenta oito ou mais GPUs. Este trabalho de exemplo pode ser enviado em um cluster para o qual um nó de trabalho com oito ou mais GPUs não está presente ou está atualmente ocupado com outra carga de trabalho. Se for esse o caso, o trabalho permanecerá em estado pendente até que o nó de trabalho fique disponível.

   Além disso, para maximizar a largura de banda de armazenamento, o volume que contém os dados de treinamento necessários é montado duas vezes dentro do pod criado por essa tarefa. Outro volume também é montado no pod. Este segundo volume será usado para armazenar resultados e métricas. Esses volumes são referenciados na definição do trabalho usando os nomes dos PVCs. Para obter mais informações sobre os trabalhos do Kubernetes, consulte o "documentação oficial do Kubernetes" .

   Um `emptyDir` volume com um `medium` valor de `Memory` é montado em `/dev/shm` no pod que este trabalho de exemplo cria. O tamanho padrão do `/dev/shm` O volume virtual criado automaticamente pelo tempo de execução do contêiner do Docker às vezes pode ser insuficiente para as necessidades do TensorFlow. Montando um `emptyDir` volume como no exemplo a seguir fornece um suficientemente grande `/dev/shm` volume virtual. Para mais informações sobre `emptyDir` volumes, veja o "documentação oficial do Kubernetes" .

   O único contêiner especificado neste exemplo de definição de trabalho recebe um `securityContext > privileged` valor de `true` . Este valor significa que o contêiner efetivamente tem acesso root no host. Esta anotação é usada neste caso porque a carga de trabalho específica que está sendo executada requer acesso root. Especificamente, uma operação de limpeza de cache executada pela carga de trabalho requer acesso root. Se isso é verdade ou não `privileged: true` a anotação é necessária depende dos requisitos da carga de trabalho específica que você está executando.

```
$ 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 se o trabalho que você criou na etapa 1 está sendo executado corretamente. O comando de exemplo a seguir confirma que um único pod foi criado para o trabalho, conforme especificado na definição do trabalho, e que esse pod está atualmente em execução em um dos nós de trabalho da 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 se o trabalho que você criou na etapa 1 foi concluído com sucesso. Os comandos de exemplo a seguir confirmam que o trabalho foi concluído com sucesso.

```
$ 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:** Limpe artefatos de trabalho. Os comandos de exemplo a seguir mostram a exclusão do objeto de trabalho que foi criado na etapa 1.

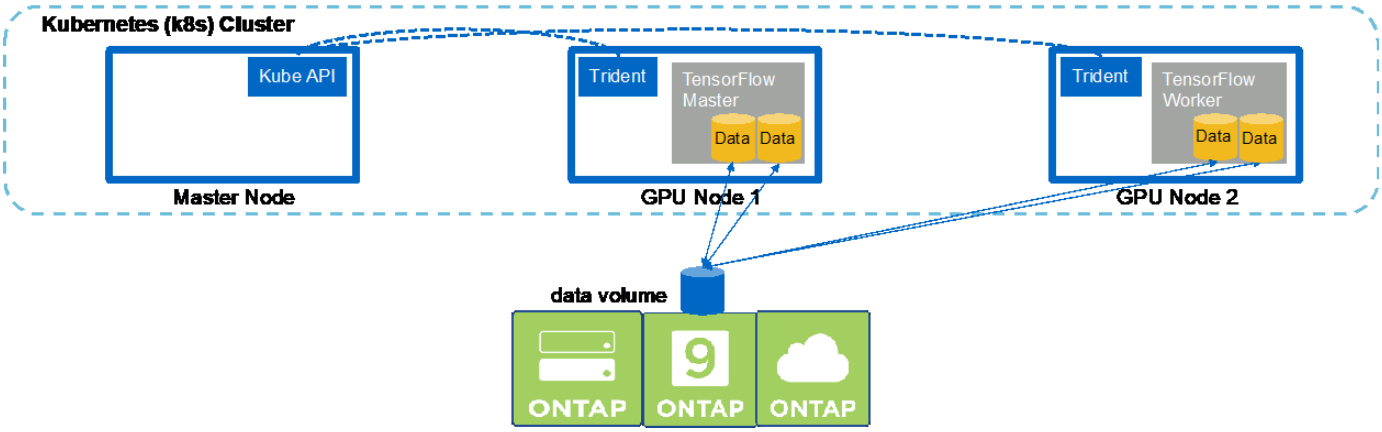
   Quando você exclui o objeto de trabalho, o Kubernetes exclui automaticamente todos os pods associados.

```
$ 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.
```

Executar uma carga de trabalho de IA distribuída síncrona

Para executar um trabalho de IA e ML multinó síncrono no seu cluster Kubernetes, execute as seguintes tarefas no host de salto de implantação. Esse processo permite que você aproveite os dados armazenados em um volume NetApp e use mais GPUs do que um único nó de trabalho pode fornecer. Veja a figura a seguir para uma representação de um trabalho de IA distribuído síncrono.

Trabalhos distribuídos síncronos podem ajudar a aumentar o desempenho e a precisão do treinamento em comparação com trabalhos distribuídos assíncronos. Uma discussão sobre os prós e contras de trabalhos síncronos versus trabalhos assíncronos está fora do escopo deste documento.



1. Os comandos de exemplo a seguir mostram a criação de um trabalhador que participa da execução distribuída síncrona do mesmo trabalho de benchmark do TensorFlow que foi executado em um único nó no exemplo da seção"Executar uma carga de trabalho de IA de nó único" . Neste exemplo específico, apenas um único trabalhador é implantado porque o trabalho é executado em dois nós de trabalho.

Este exemplo de implantação de trabalhador solicita oito GPUs e, portanto, pode ser executado em um único nó de trabalhador de GPU que apresenta oito ou mais GPUs. Se os nós de trabalho da sua GPU tiverem mais de oito GPUs, para maximizar o desempenho, você pode aumentar esse número para que fique igual ao número de GPUs que os nós de trabalho têm. Para obter mais informações sobre implantações do Kubernetes, consulte o "documentação oficial do Kubernetes" .

Uma implantação do Kubernetes é criada neste exemplo porque esse trabalhador em contêiner específico nunca seria concluído sozinho. Portanto, não faz sentido implantá-lo usando a construção de tarefa do Kubernetes. Se seu trabalhador foi projetado ou escrito para ser concluído sozinho, pode fazer sentido usar a construção de tarefa para implantá-lo.

O pod especificado neste exemplo de especificação de implantação recebe um `hostNetwork` valor de `true` . Esse valor significa que o pod usa a pilha de rede do nó de trabalho do host em vez da pilha de rede virtual que o Kubernetes normalmente cria para cada pod. Esta anotação é usada neste caso porque a carga de trabalho específica depende do Open MPI, NCCL e Horovod para executar a carga de trabalho de maneira distribuída e síncrona. Portanto, ele requer acesso à pilha de rede do host. Uma discussão sobre Open MPI, NCCL e Horovod está fora do escopo deste documento. Se isso é verdade ou não `hostNetwork: true` a anotação é necessária depende dos requisitos da carga de trabalho específica que você está executando. Para mais informações sobre o `hostNetwork` campo, veja o "documentação oficial do 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 se a implantação do trabalhador que você criou na etapa 1 foi iniciada com sucesso. Os comandos de exemplo a seguir confirmam que um único pod de trabalho foi criado para a implantação, conforme indicado na definição de implantação, e que esse pod está atualmente em execução em um dos nós de trabalho da 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. Crie um trabalho do Kubernetes para um mestre que inicia, participa e rastreia a execução do trabalho multinó síncrono. Os comandos de exemplo a seguir criam um mestre que inicia, participa e rastreia a execução distribuída síncrona do mesmo trabalho de benchmark do TensorFlow que foi executado em um

único nó no exemplo da seção"Executar uma carga de trabalho de IA de nó único" .

Este exemplo de tarefa mestre solicita oito GPUs e, portanto, pode ser executado em um único nó de trabalho de GPU que apresenta oito ou mais GPUs. Se os nós de trabalho da sua GPU tiverem mais de oito GPUs, para maximizar o desempenho, você pode aumentar esse número para que fique igual ao número de GPUs que os nós de trabalho têm.

O pod mestre especificado neste exemplo de definição de trabalho recebe um `hostNetwork` valor de `true` , assim como o grupo de trabalhadores recebeu uma `hostNetwork` valor de `true` na etapa 1. Veja a etapa 1 para obter detalhes sobre por que esse valor é necessário.

```
$ 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 se o trabalho mestre que você criou na etapa 3 está sendo executado corretamente. O comando de exemplo a seguir confirma que um único pod mestre foi criado para o trabalho, conforme indicado na definição do trabalho, e que esse pod está atualmente em execução em um dos nós de trabalho da GPU. Você também deve ver que o pod de trabalho que você viu originalmente na etapa 1 ainda está em execução e que os pods mestre e de trabalho estão em execução em nós 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 se o trabalho mestre que você criou na etapa 3 foi concluído com sucesso. Os comandos de exemplo a seguir confirmam que o trabalho foi concluído com sucesso.

```
$ 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. Exclua a implantação do trabalhador quando não precisar mais dela. Os comandos de exemplo a seguir mostram a exclusão do objeto de implantação do trabalhador que foi criado na etapa 1.

   Quando você exclui o objeto de implantação do trabalhador, o Kubernetes exclui automaticamente todos os pods de trabalhador associados.

```
$ 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:** Limpe os artefatos do trabalho mestre. Os comandos de exemplo a seguir mostram a exclusão do objeto de trabalho mestre que foi criado na etapa 3.

   Quando você exclui o objeto de trabalho mestre, o Kubernetes exclui automaticamente todos os pods mestres associados.

```
$ 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.
```

Informações sobre direitos autorais

Informações sobre marcas comerciais