使用 NetApp 的開放原始碼 MLOps

NetApp Solutions

NetApp
May 10, 2024

目錄

使用 NetApp 的開放原始碼 MLOps

使用 NetApp 的開放原始碼 MLOps

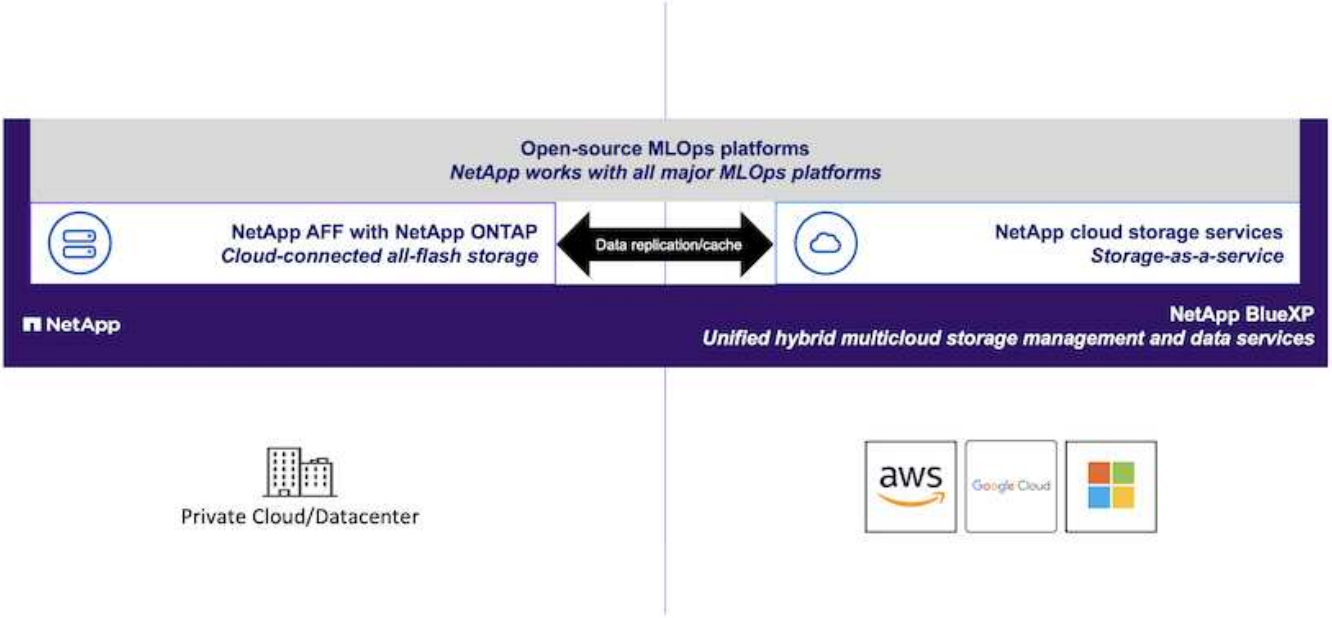
Mike Oglesby、NetApp
NetApp 的 Mohan Acharya

各種規模的公司和組織、以及許多產業、紛紛轉向人工智慧（AI）、機器學習（ML）和深度學習（DL）、以解決實際問題、提供創新產品和服務、並在競爭日益激烈的市場中獲得優勢。隨著企業組織增加AI、ML和DL的使用率、他們面臨許多挑戰、包括工作負載擴充性和資料可用度。本解決方案將 NetApp 資料管理功能與熱門的開放原始碼工具和架構配對、示範如何解決這些挑戰。

本解決方案旨在示範幾種不同的開放原始碼工具和架構、這些工具和架構可整合至 MLOps 工作流程中。這些不同的工具和架構可以一起使用、也可以自行使用、視需求和使用案例而定。

本解決方案涵蓋下列工具 / 架構：

- "Apache Airflow"
- "Kubeflow"



技術總覽

人工智慧

AI是一項電腦科學訓練、訓練電腦模擬人類思維的認知功能。AI開發人員訓練電腦、以類似甚至優於人類的方式學習及解決問題。深度學習和機器學習是AI的子領域。企業組織越來越採用AI、ML和DL來支援其關鍵業務需求。以下是一些範例：

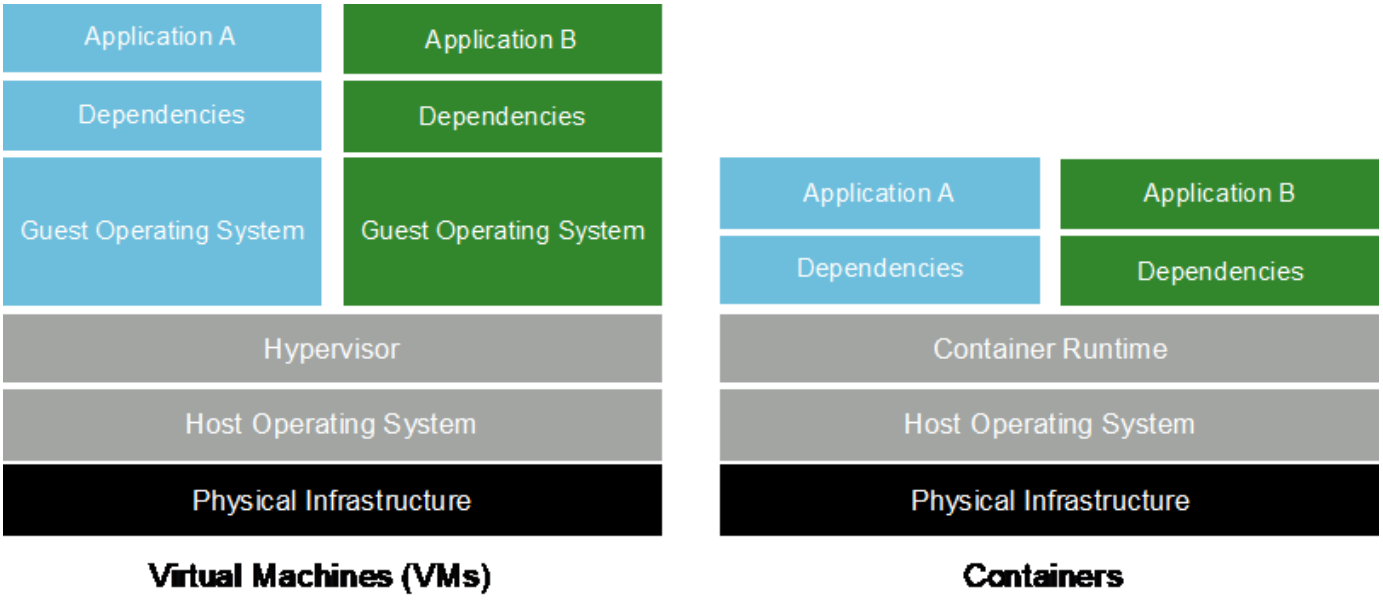
- 分析大量資料、發掘先前未知的商業洞見
- 使用自然語言處理功能直接與客戶互動
- 自動化各種業務流程與功能

現代化的AI訓練和推斷工作負載需要大量平行運算功能。因此、GPU越來越常用於執行AI作業、因為GPU的平行處理能力遠優於通用CPU。

容器

容器是獨立的使用者空間執行個體、可在共享主機作業系統核心上執行。容器的採用率正在迅速增加。Container提供許多與虛擬機器（VM）相同的應用程式沙箱效益。不過、由於虛擬機器所仰賴的Hypervisor和客體作業系統層已經被淘汰、因此容器的重量遠較輕。下圖說明虛擬機器與容器的視覺化。

容器也能直接透過應用程式、有效封裝應用程式相依性、執行時間等項目。最常用的容器包裝格式是Docker容器。以Docker Container格式容器化的應用程式、可在任何能夠執行Docker Container的機器上執行。即使應用程式的相依性並不存在於機器上、也一樣、因為所有相依性都封裝在容器本身。如需詳細資訊、請參閱 "Docker網站"。



Kubernetes

Kubernetes是開放原始碼的分散式容器協調平台、最初由Google設計、現在由Cloud Native Computing Foundation（CNCF）維護。Kubernetes可將容器化應用程式的部署、管理及擴充功能自動化。近年來、Kubernetes已成為主要的容器協調平台。如需詳細資訊、請參閱 "Kubernetes網站"。

NetApp Astra Trident

Astra Trident 可在所有常見的 NetApp 儲存平台、公有雲或內部部署（包括 ONTAP （ AFF 、 FAS 、 Select 、 Cloud 、 Amazon FSX for NetApp ONTAP ） 、 Element 軟體（ NetApp HCI 、 SolidFire ） 、 Azure NetApp Files 服務、以及 Google Cloud 上的 Cloud Volumes Service 。Astra Trident 是符合 Container Storage Interface （ CSI ）規範的動態儲存協調器、可與 Kubernetes 原生整合。

NetApp DataOps工具套件

。 "NetApp DataOps工具套件" 是一套 Python 型工具、可簡化開發 / 訓練工作區的管理、以及以高效能橫向擴充 NetApp 儲存設備為後盾的推斷伺服器。主要功能包括：

- 快速配置以高效能橫向擴充 NetApp 儲存設備為後盾的全新高容量工作區。
- 以近乎即時的方式複製高容量工作空間、以便進行實驗或快速迭代。
- 近乎即時地儲存高容量工作區的快照、以供備份及 / 或追蹤 / 基準處理。
- 近乎即時地配置、複製及快照高容量、高效能的資料磁碟區。

Kubeflow

Kubeflow是Kubernetes的開放原始碼AI和ML工具套件、最初由Google開發。Kubeflow專案讓Kubernetes上的AI和ML工作流程部署變得簡單、可攜且可擴充。Kubeflow 將 Kubernetes 的複雜性抽象化、讓資料科學家能夠專注於他們最瞭解的資料科學。如需視覺化功能、請參閱下圖。Kubeflow 是適合偏好一體化 MLOps 平台的組織、是理想的開放原始碼選項。如需詳細資訊、請參閱 "Kubeflow網站"。

Kubeflow Pipines

Kubeflow Pipines是Kubeflow的重要元件。Kubeflow Pipines是定義及部署可攜式且可擴充的AI和ML工作流程的平台和標準。如需詳細資訊、請參閱 "官方Kubeflow文件"。

Jupyter筆記型電腦伺服器

Jupyter Notebook Server是開放原始碼的網路應用程式、可讓資料科學家建立名為Jupyter Notebooks的維基類文件、其中包含即時程式碼和描述性測試。Jupyter筆記型電腦在AI和ML社群中廣為使用、可用來記錄、儲存及分享AI和ML專案。Kubeflow簡化了Kubernetes上Jupyter筆記型電腦伺服器的資源配置與部署。如需Jupyter筆記型電腦的詳細資訊、請參閱 "Jupyter網站"。如需Kubeflow內容中Jupyter Notebooks的詳細資訊、請參閱 "官方Kubeflow文件"。

Katib

Katib 是 Kubernetes 原生的自動化機器學習（ AutoML ）專案。Katib 支援超參數調校、早期停止和神經架構搜尋（ NAS ）。Katib 是與機器學習（ ML ）架構不相關的專案。它可以調整以使用者選擇的任何語言寫入的應用程式超頻參數、並原生支援許多 ML 架構、例如 TensorFlow 、 MXNet 、 PyTorch 、 XGBoost 、 及其他。Katib 支援許多不同的 AutoML 演算法、例如 Bayesia 最佳化、 Parzen Estimators 樹、隨機搜尋、 Covariance Matrix Adaptation Evolution Strategy 、 Hyperband 、高效神經架構搜尋、差異化架構搜尋等。如需Kubeflow內容中Jupyter Notebooks的詳細資訊、請參閱 "官方Kubeflow文件"。

Apache Airflow

Apache Airflow是開放原始碼的工作流程管理平台、可針對複雜的企業工作流程、進行程式化的撰寫、排程及監控。它通常用於自動化ETL和資料管線工作流程、但不限於這些類型的工作流程。氣流專案是由Airbnb發起、但後來在業界廣受歡迎、現在由Apache Software Foundation贊助。氣流是以Python撰寫、氣流工作流程是透過Python指令碼建立、氣流是以「組態為程式碼」的原則設計。 許多企業氣流使用者現在都在Kubernetes上執行氣流。

定向Acyclic圖表（DAG）

在氣流中、工作流程稱為「導向Acyclic Graphs（DAG）（導向型Acyclic Graphs（DAG））」。DAG是由依順序、平行或兩者組合執行的工作所組成、視DAG定義而定。氣流排程器會在一組工作人員上執行個別工作、並

遵循DAG定義中指定的工作層級相依性。DAG是透過Python指令碼來定義和建立。

NetApp ONTAP

NetApp最新一代的儲存管理軟體、即支援企業將基礎架構現代化、並移轉至雲端就緒的資料中心。ONTAP利用領先業界的資料管理功能ONTAP 、無論資料位於何處、只要使用一組工具、即可管理及保護資料。您也可以自由地將資料移至任何需要的位置：邊緣、核心或雲端。支援眾多功能、可簡化資料管理、加速及保護關鍵資料、並在混合雲架構中提供新一代基礎架構功能。ONTAP

簡化資料管理

資料管理對於企業IT營運和資料科學家而言至關重要、因此可將適當的資源用於AI應用程式和訓練AI/ML資料集。下列關於NetApp技術的其他資訊超出此驗證範圍、但可能會因您的部署而有所差異。

包含下列功能的資料管理軟體、可簡化及簡化作業、並降低您的總營運成本：ONTAP

- 即時資料精簡與擴充重複資料刪除技術。資料壓縮可減少儲存區塊內的空間浪費、重複資料刪除技術可大幅提升有效容量。這適用於本機儲存的資料、以及分層至雲端的資料。
- 最低、最大及可調適的服務品質（AQO）。精細的服務品質（QoS）控制有助於維持高共享環境中關鍵應用程式的效能等級。
- NetApp FabricPool自動將冷資料分層至公有和私有雲端儲存選項、包括Amazon Web Services（AWS）、Azure和NetApp StorageGRID 等儲存解決方案。如需FabricPool 更多有關資訊、請參閱 "TR-4598：FabricPool 最佳實務做法"。

加速並保護資料

提供優異的效能與資料保護、並以下列方式擴充這些功能：ONTAP

- 效能與較低的延遲。以最低的延遲提供最高的處理量。ONTAP
- 資料保護：支援所有平台的通用管理功能、可提供內建的資料保護功能。ONTAP
- NetApp Volume Encryption（NVE）。支援內建和外部金鑰管理、提供原生Volume層級的加密功能。ONTAP
- 多租戶和多因素驗證。支援以最高安全等級共享基礎架構資源。ONTAP

符合未來需求的基礎架構

下列功能可協助滿足嚴苛且不斷變化的業務需求：ONTAP

- 無縫擴充與不中斷營運。支援在不中斷營運的情況下、將容量新增至現有控制器和橫向擴充叢集。ONTAP客戶可以升級至最新技術、例如NVMe和32GB FC、而不需進行昂貴的資料移轉或中斷運作。
- 雲端連線：ONTAP 是最具雲端連線能力的儲存管理軟體、可在所有公有雲中選擇軟體定義儲存設備和雲端原生執行個體。
- 與新興應用程式整合。利用支援現有企業應用程式的相同基礎架構、為新一代平台和應用程式提供企業級資料服務、例如自動駕駛車輛、智慧城市和產業4.0。ONTAP

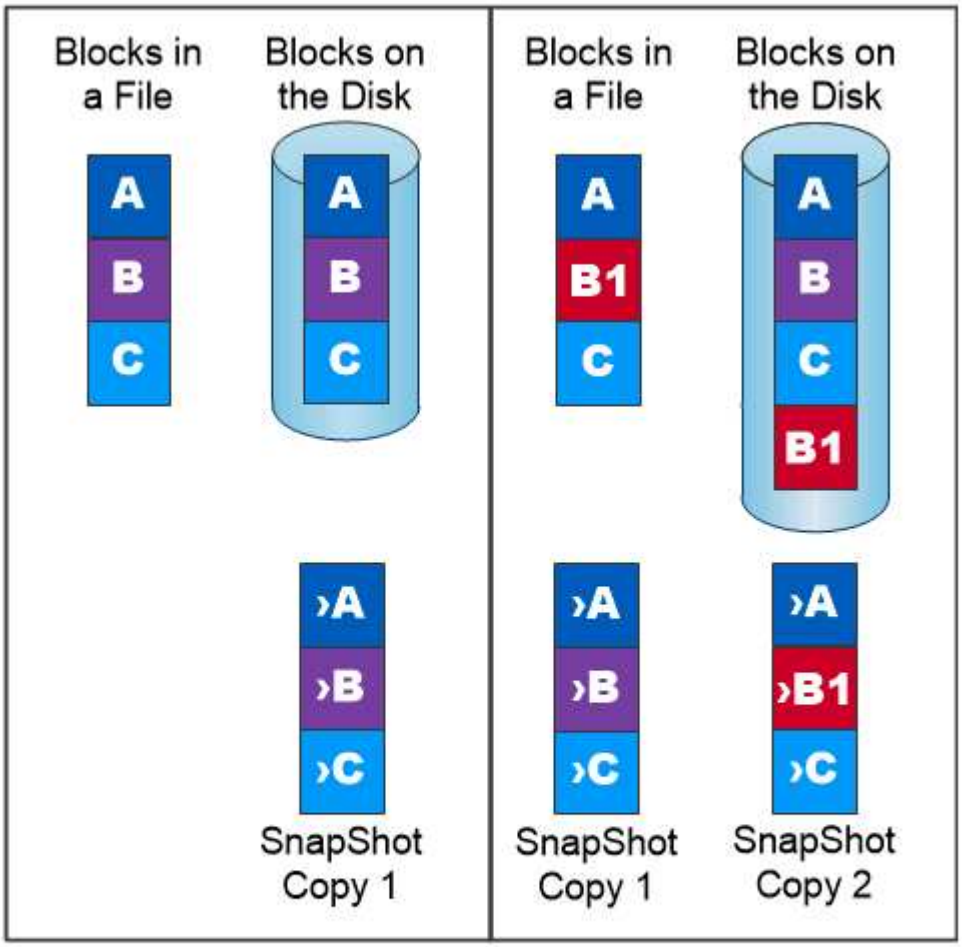
NetApp Snapshot複本

NetApp Snapshot複本是磁碟區的唯讀時間點映像。此映像會佔用最小的儲存空間、並產生可忽略的效能負荷、

因為它只會記錄自上次建立Snapshot複本以來所建立的檔案變更、如下圖所示。

Snapshot複本的效率歸功於核心ONTAP 的不穩定儲存虛擬化技術WAFL 、亦即Write Anywhere File Layout（簡稱「Write Anywhere File Layout」、簡稱「Write Anywhere」）。如同資料庫、WAFL 利用中繼資料指向磁碟上的實際資料區塊。但是WAFL 、不像資料庫、不像是使用什麼功能來覆寫現有的區塊。它會將更新的資料寫入新的區塊、並變更中繼資料。這是因為ONTAP 當我們建立Snapshot複本時、不需要複製資料區塊、而是參考中繼資料、所以Snapshot複本非常有效率。如此可免除其他系統在尋找要複製的區塊時所需的搜尋時間、以及複本本身的成本。

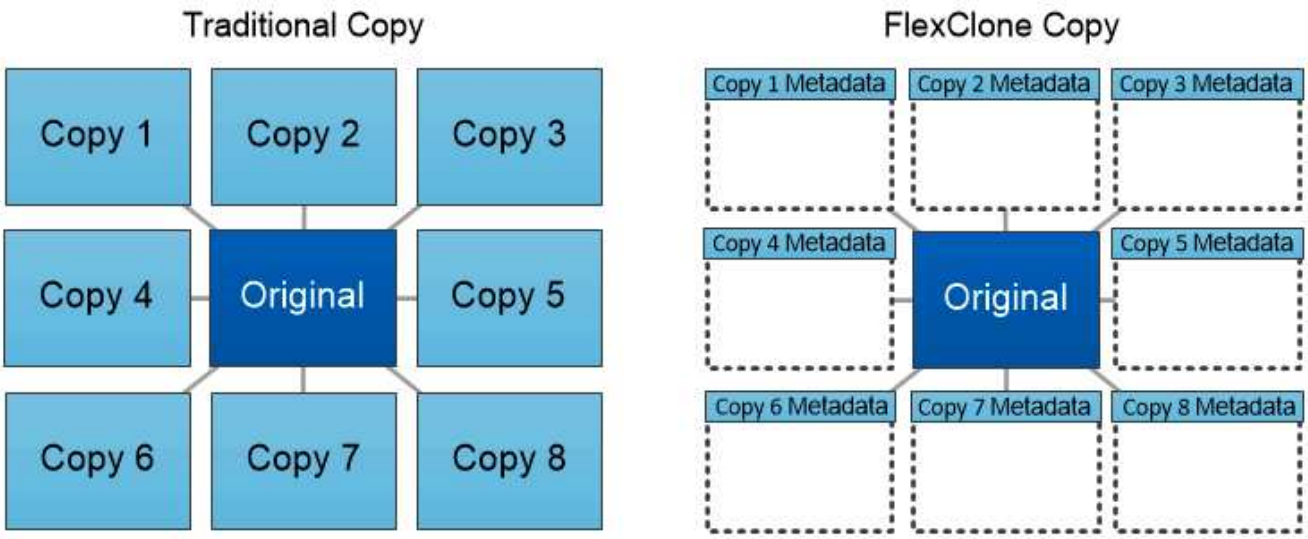
您可以使用Snapshot複本來還原個別檔案或LUN、或還原磁碟區的完整內容。此功能可將Snapshot複本中的指標資訊與磁碟上的資料進行比較、以重建遺失或損壞的物件、而不會造成停機或重大效能成本。ONTAP



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

NetApp FlexClone技術

NetApp FlexClone技術會參考Snapshot中繼資料、以建立磁碟區的可寫入時間點複本。複本會與父實體共用資料區塊、除非中繼資料需要的資料、否則不會佔用任何儲存空間、直到將變更寫入複本為止、如下圖所示。在傳統複本需要數分鐘甚至數小時才能建立的地方、FlexClone軟體可讓您幾乎即時複製最大的資料集。這使得它非常適合您需要多個相同資料集複本（例如開發工作區）或資料集暫存複本（針對正式作業資料集測試應用程式）的情況。



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

NetApp SnapMirror資料複寫技術

NetApp SnapMirror軟體是一款具成本效益且易於使用的統一化複寫解決方案、適用於整個資料架構。它可透過LAN或WAN高速複寫資料。它可為各種應用程式提供高資料可用度及快速資料複寫、包括虛擬與傳統環境中的業務關鍵應用程式。當您將資料複寫到一或多個NetApp儲存系統、並持續更新次要資料時、資料會保持最新狀態、而且隨時可供使用。不需要外部複寫伺服器。請參閱下圖、瞭解運用SnapMirror技術的架構範例。

SnapMirror軟體透過ONTAP 網路僅傳送變更的區塊、充分發揮NetApp的效能。SnapMirror軟體也使用內建的網路壓縮功能來加速資料傳輸、並減少高達70%的網路頻寬使用率。有了SnapMirror技術、您可以利用單一精簡複寫資料串流來建立單一儲存庫、同時維護作用中鏡像和先前的時間點複本、最多可減少50%的網路流量。

NetApp BlueXP 複製與同步

BlueXP 複製與同步是 NetApp 服務、可快速安全地同步資料。無論您需要在內部部署的 NFS 或 SMB 檔案共用、 NetApp StorageGRID 、 NetApp ONTAP S3 、 NetApp Cloud Volumes Service 、 Azure NetApp Files 、 AWS S3 、 AWS EFS 、 Azure Blob 、 Google Cloud Storage 或 IBM Cloud Object Storage 、 BlueXP 複製與同步功能可快速安全地將檔案移至所需的位置。

資料傳輸完成後、即可在來源和目標上完全使用。BlueXP 複製與同步可在觸發更新時隨需同步資料、或根據預先定義的排程持續同步資料。不過、 BlueXP 複製與同步只會移動資料量、因此將用於資料複寫的時間與金錢降到最低。

BlueXP 複製與同步是一種軟體即服務（ SaaS ）工具、設定與使用極為簡單。BlueXP 複製與同步所觸發的資料傳輸是由資料代理人執行。BlueXP 複製與同步資料代理人可以部署在 AWS 、 Azure 、 Google Cloud Platform 或內部部署。

NetApp XCP

NetApp XCP是以用戶端為基礎的軟體、適用於任何對NetApp和NetApp對NetApp的資料移轉及檔案系統洞見。XCP的設計旨在利用所有可用的系統資源來處理大量資料集和高效能移轉、以擴充並達到最大效能。XCP可讓您利用產生報告的選項、全面掌握檔案系統。

NetApp XCP可在單一套件中取得、支援NFS和SMB傳輸協定。XCP包含適用於NFS資料集的Linux二進位檔、以及適用於SMB資料集的Windows執行檔。

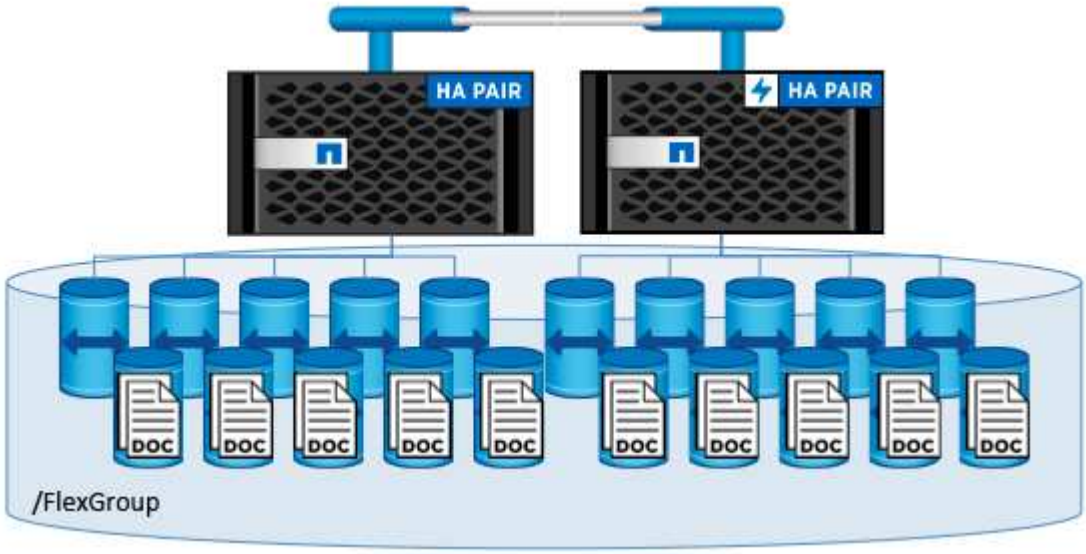
NetApp XCP檔案分析是以主機為基礎的軟體、可偵測檔案共用、在檔案系統上執行掃描、並提供檔案分析儀表板。XCP檔案分析可與NetApp和非NetApp系統相容、並可在Linux或Windows主機上執行、以提供NFS和SMB匯出檔案系統的分析功能。

NetApp ONTAP FlexGroup 產品區

訓練資料集可能是數十億個檔案的集合。檔案可以包含文字、音訊、視訊及其他形式的非結構化資料、這些資料必須儲存和處理才能並行讀取。儲存系統必須儲存大量的小型檔案、而且必須平行讀取這些檔案、才能執行連續和隨機I/O

例如下圖所示、一個包含多個組成成員磁碟區的單一命名空間。FlexGroup從儲存管理員的觀點來看、FlexGroup 可管理一個不實的功能、就像NetApp FlexVol 的一套功能。將某個資料區中的檔案FlexGroup 分配給個別成員磁碟區、而不會跨磁碟區或節點進行等量分佈。這些功能可實現下列功能：

- 支援多PB容量、可預測低延遲的高中繼資料工作負載。FlexGroup
- 在同一個命名空間中支援高達4000億個檔案。
- 它們支援跨CPU、節點、集合體及組成FlexVol 的等量資料磁碟區、在NAS工作負載中進行平行化作業。



架構

此解決方案不取決於特定硬體。此解決方案可與Trident支援的任何NetApp實體儲存設備、軟體定義執行個體或雲端服務相容。範例包括 NetApp AFF 儲存系統、適用於 NetApp ONTAP 的 Amazon FSX 、 Azure NetApp Files 或 NetApp Cloud Volumes ONTAP 執行個體。此外、只要 Kubeflow 和 NetApp Astra Trident 支援使用的 Kubernetes 版本、即可在任何 Kubernetes 叢集上實作解決方案。如需Kubernetes支援版本的清單、請參閱 "官方Kubeflow文件"。如需Trident支援的Kubernetes版本清單、請參閱 "Trident文件"。如需驗證解決方案所用環境的詳細資訊、請參閱下表。

軟體元件	版本
Apache Airflow	2.0.1
Apache Airflow Helm圖表	8.0.8
Kubeflow	1.7 、透過部署 "deploykF" 0.1.1.
Kubernetes	1.26
NetApp Astra Trident	23.07

支援

NetApp 不提供 Apache Airflow 、 Kubefflow 或 Kubernetes 的企業支援。如果您對完全支援的 MLOps 平台感興趣、 "請聯絡NetApp" 關於 NetApp 與合作夥伴共同提供的完全支援的 MLOps 解決方案。

NetApp Astra Trident 組態

NetApp AIPod 部署的 Astra Trident 後端範例

在您可以使用 Astra Trident 在 Kubernetes 叢集中動態配置儲存資源之前、您必須先建立一個或多個 Trident Back用途 。下列範例代表在上部署此解決方案元件時、您可能會想要建立的不同類型的後端 "NetApp AIPod"。如需後端的詳細資訊、請參閱 "Astra Trident文件"。

1. NetApp 建議您為 AIPod 建立啟用 FlexGroup 的 Trident Backend 。

   以下命令範例顯示為 AIPod 儲存虛擬機器（ SVM ）建立啟用 FlexGroup 的 Trident Backend 。此後端使用 `ontap-nas-flexgroup` 儲存驅動程式：支援兩種主要資料Volume類型：功能完善和功能完善。ONTAP FlexVol FlexGroup由於資料不多（本文所述的最大大小取決於特定部署）、因此不受支援。FlexVol另一方面、由於支援的資料量可線性擴充至20PB和4000億個檔案、因此單一命名空間可大幅簡化資料管理。FlexGroup因此FlexGroup 、對於仰賴大量資料的AI和ML工作負載而言、此功能是最佳選擇。

   如果您使用的是少量資料、想要使用FlexVol 不FlexGroup 含「orfvolume」的「orfvolume」、您可以建立使用「ontap-nas」儲存驅動程式而非「ontap-nas flexgroup」儲存驅動程式的「Trident後端」。

```
$ cat << EOF > ./trident-backend-aipod-flexgroups-iface1.json
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": "aipod-flexgroups-iface1",
    "managementLIF": "10.61.218.100",
    "dataLIF": "192.168.11.11",
    "svm": "ontapai_nfs",
    "username": "admin",
    "password": "ontapai"
}
EOF
$ tridentctl create backend -f ./trident-backend-aipod-flexgroups-
iface1.json -n trident
+------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+---------------------
+--------------------------------------+--------+---------+
| aipod-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+------------------------+---------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME         |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+------------------------+---------------------
+--------------------------------------+--------+---------+
| aipod-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+------------------------+---------------------
+--------------------------------------+--------+---------+
```

2. NetApp 也建議您建立啟用 FlexVol 的 Trident 後端。您可能想要使用 FlexVol Volume 來裝載持續應用程式、儲存結果、輸出、偵錯資訊等。如果您想要使用FlexVol 「資料不全」、您必須建立一個或多個FlexVol 啟用「功能不全」的「資料不全」後端。以下範例命令顯示建立單一啟用 FlexVol 的 Trident 後端。

```
$ 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 |
+-------------------------+--------------------
+--------------------------------------+--------+---------+
```

NetApp AIPod 部署的 Kubernetes StorageClasses 範例

您必須先建立一個或多個 Kubernetes StorageClasses 、才能使用 Astra Trident 在 Kubernetes 叢集中動態配置儲存資源。下列範例代表您在上部署此解決方案元件時、可能會想要建立的不同類型 StorageClasses "NetApp AIPod"。如需StorageClass的詳細資訊、請參閱 "Astra Trident文件"。

1. NetApp 建議為您在本節中建立的啟用 FlexGroup 的 Trident Backend 建立 StorageClass "NetApp AIPod 部署的 Astra Trident 後端範例"步驟1.後面的命令範例顯示建立多個 StorageClasses 、這些儲存類別對應於本節所建立的兩個範例後端 "NetApp AIPod 部署的 Astra Trident 後端範例"第 1 步 - 利用的步驟 "NFS over RDMA" 而不是。

   因此當刪除對應的PersistentVolume Claim（PVc）時、不會刪除持續磁碟區、以下範例使用「回收原則」值「保留」。如需「回收政策」欄位的詳細資訊、請洽相關官員 "Kubernetes文件"。

   附註：下列 StorageClasses 範例使用的傳輸大小上限為 262144 。若要使用此最大傳輸大小、您必須相應地在 ONTAP 系統上設定最大傳輸大小。請參閱 "本文檔 ONTAP" 以取得詳細資料。

   注意：若要透過 RDMA 使用 NFS 、您必須在 ONTAP 系統上設定透過 RDMA 的 NFS 。如需詳細資訊、請參閱《 https://docs.netapp.com/us-en/ontap/nfs-rdma/[ONTAP 》文件。

   附註：在下列範例中、 StorageClass 定義檔案的 storagePool 欄位中未指定特定的後端。

```
$ cat << EOF > ./storage-class-aipod-flexgroups-retain.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: aipod-flexgroups-retain
provisioner: csi.trident.netapp.io
mountOptions: ["vers=4.1", "nconnect=16", "rsize=262144",
"wsize=262144"]
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "aipod-flexgroups-iface1:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-aipod-flexgroups-retain.yaml
storageclass.storage.k8s.io/aipod-flexgroups-retain created
$ cat << EOF > ./storage-class-aipod-flexgroups-retain-rdma.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: aipod-flexgroups-retain-rdma
provisioner: csi.trident.netapp.io
mountOptions: ["vers=4.1", "proto=rdma", "max_connect=16",
"rsize=262144", "wsize=262144"]
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "aipod-flexgroups-iface1:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-aipod-flexgroups-retain-rdma.yaml
storageclass.storage.k8s.io/aipod-flexgroups-retain-rdma created
$ kubectl get storageclass
NAME                              PROVISIONER             AGE
aipod-flexgroups-retain           csi.trident.netapp.io   0m
aipod-flexgroups-retain-rdma      csi.trident.netapp.io   0m
```

2. NetApp也建議您建立與您在本節中建立的具有FlexVol功能的Trident後端相對應的StorageClass "AIPod 部署的 Astra Trident 後端範例"步驟2：以下命令範例顯示建立FlexVol 單一StorageClass for the餐廳。

   附註：在下列範例中、 StorageClass 定義檔案的 storagePool 欄位中未指定特定的後端。當您使用 Kubernetes 來管理使用此 StorageClass 的磁碟區時、 Trident 會嘗試使用任何可用的後端 `ontap-nas` 驅動程式：

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

Kubeflow

Kubeflow部署

本節說明在Kubernetes叢集中部署Kubeflow時、必須完成的工作。

先決條件

在您執行本節所述的部署練習之前、我們假設您已經執行下列工作：

1. 您已經有一個運作中的 Kubernetes 叢集、而且您正在執行的 Kubernetes 版本受到您要部署的 Kubeflow 版本的支援。如需支援的 Kubernetes 版本清單、請參閱中 Kubeflow 版本的相依性 "官方Kubeflow文件"。
2. 您已在 Kubernetes 叢集中安裝並設定 NetApp Astra Trident 。如需 Astra Trident 的詳細資訊、請參閱 "Astra Trident文件"。

設定預設**Kubernetes StorageClass**

在您部署 Kubeflow 之前、建議您在 Kubernetes 叢集中指定預設 StorageClass 。Kubeflow 部署程序可能會嘗試使用預設 StorageClass 來配置新的持續磁碟區。如果未將 StorageClass 指定為預設 StorageClass 、則部署可能會失敗。若要在叢集內指定預設StorageClass、請從部署跨接主機執行下列工作。如果您已在叢集內指定預設StorageClass、則可以跳過此步驟。

1. 將現有的其中一個StorageClass指定為預設StorageClass。後面的命令範例顯示指定的 StorageClass 名稱 `ontap-ai-flexvols-retain` 為預設 StorageClass 。

> 「ontap-non-flexgroup」Trident後端類型的最小PVc尺寸相當大。根據預設、Kubeflow會嘗試配置大小只有幾GB的PVCS。因此、您不應將使用「ONTAP-NAAS-Flexgroup」後端類型的StorageClass指定為Kubefflow部署的預設StorageClass。

```
$ 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
```

Kubeflow 部署選項

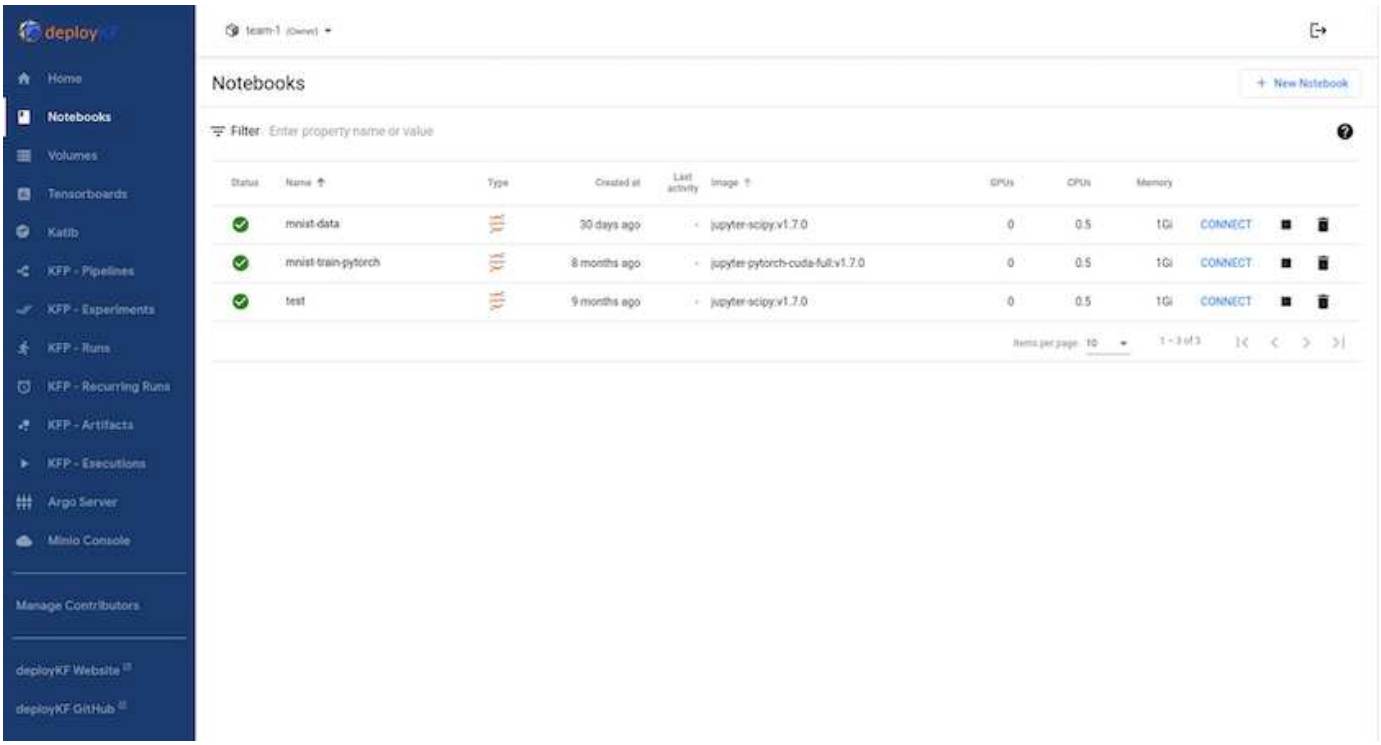
部署 Kubeflow 有許多不同的選項。請參閱 "官方Kubeflow文件" 如需部署選項清單、請選擇最適合您需求的選項。

為了驗證目的、我們使用部署 Kubeflow 1.7 "deploykF" 0.1.1 。

Kubeflow作業與工作範例

為資料科學家或開發人員提供**Jupyter**筆記型電腦工作區

Kubeflow能夠快速配置新的Jupyter筆記型電腦伺服器、做為資料科學家工作空間。如需Kubeflow內容中Jupyter Notebooks的詳細資訊、請參閱 "官方Kubeflow文件"。



使用 **NetApp DataOps Toolkit** 搭配 **Kubeflow**

。 "適用於Kubernetes的NetApp Data科學工具套件" 可與Kubeflow搭配使用。搭配Kubeflow使用NetApp Data科學工具套件可提供下列效益：

- 資料科學家可以直接從 Jupyter Notebook 執行進階的 NetApp 資料管理作業、例如建立快照和複本。
- 進階 NetApp 資料管理作業（例如建立快照和複本）可以使用 Kubeflow Pipelines 架構整合到自動化工作流程中。

請參閱 "Kubeflow範例" NetApp Data科學工具套件GitHub儲存庫中的一節、詳細說明如何搭配Kubeflow使用此工具組。

範例工作流程：使用 **Kubeflow** 和 **NetApp DataOps Toolkit** 訓練影像辨識模型

本節說明使用 Kubeflow 和 NetApp DataOps 工具套件來訓練及部署用於影像辨識的神經網路所涉及的步驟。這是一個範例、旨在展示整合 NetApp 儲存設備的訓練工作。

先決條件

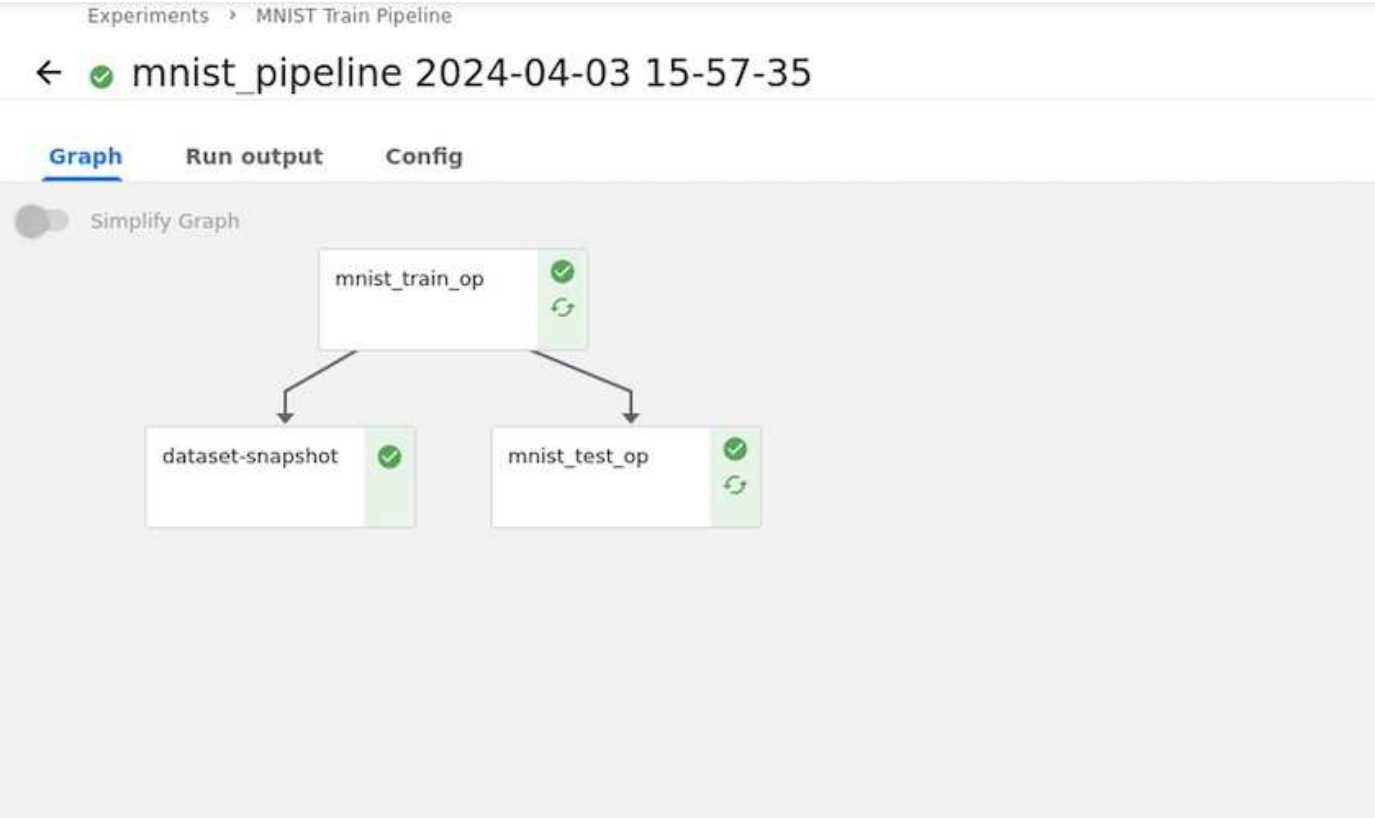
使用必要的組態建立一個 Dockerfile 、以用於 Kubeflow 管道中的訓練和測試步驟。
以下是一個 Dockerfile 的範例：

```
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"]
```

視您的需求而定、安裝執行程式所需的所有必要程式庫和套件。在您訓練機器學習模式之前、我們假設您已經有一個有效的 Kubefflow 部署。

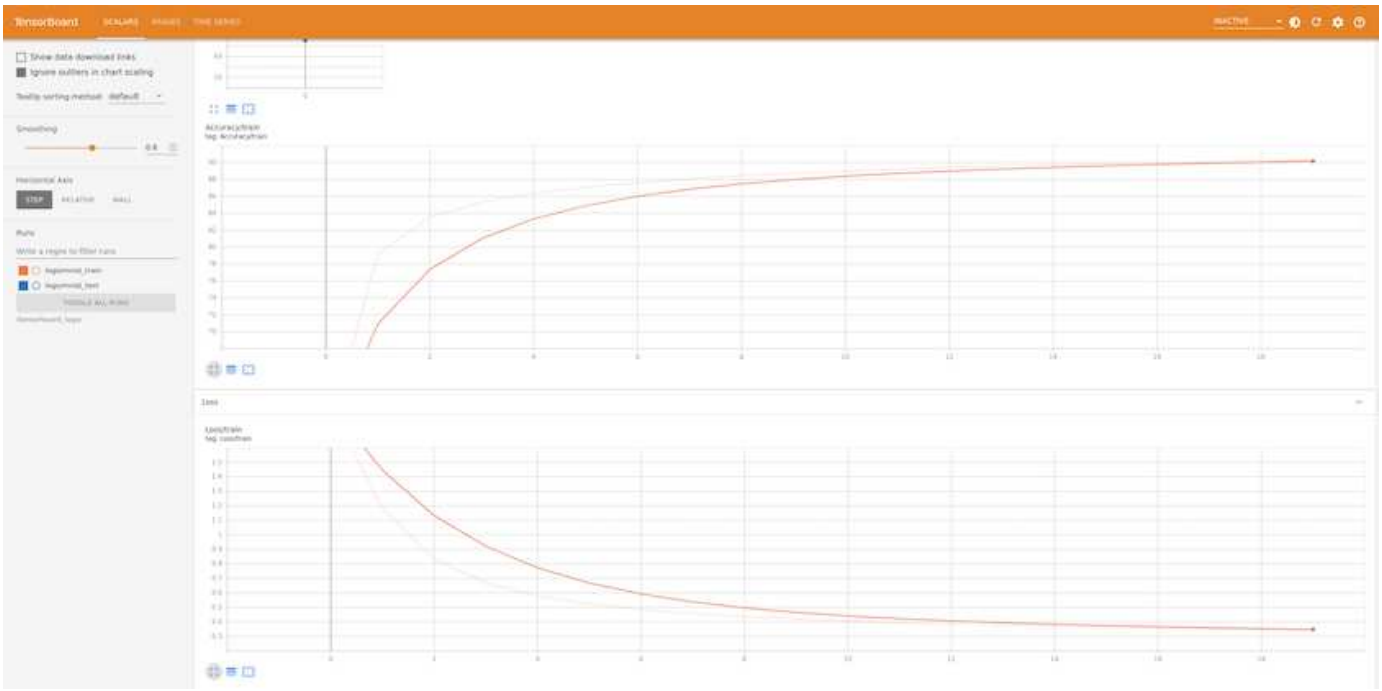
使用 **PyTorch** 和 **Kubeflow Pipelines** 訓練一部小型 **NN on MIST Data**

我們使用小型神經網路的範例來訓練 MIST 資料。MIST 資料集由 0-9 位數的手寫影像組成。影像大小為 28x28 像素。資料集分為 60 、 000 個訓練影像和 10 、 000 個驗證影像。此實驗使用的神經網路是 2 層饋送網路。訓練是使用 Kubeflow Pipelines 執行。請參閱文件 "請按這裡" 以取得更多資訊。我們的 Kubeflow 管道整合了「先決條件」區段中的泊塢視窗影像。



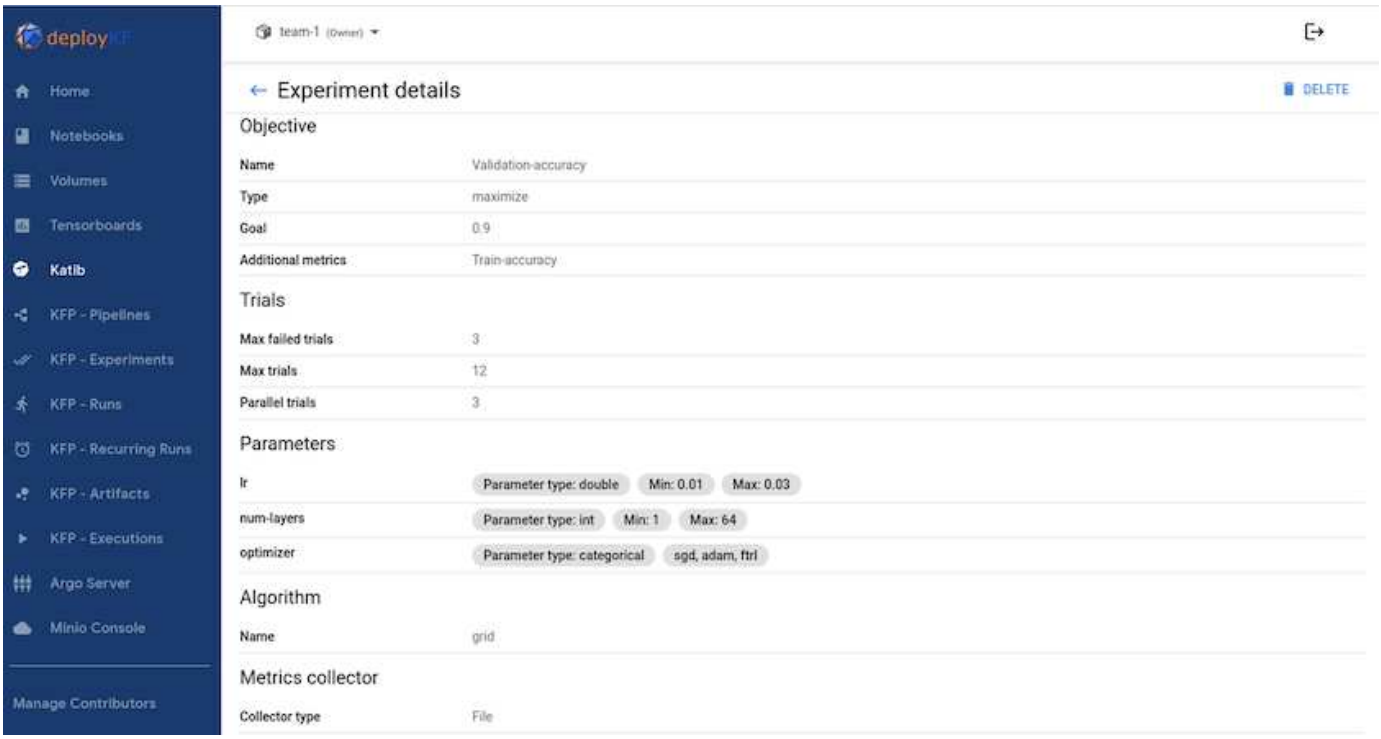
使用 **Tensorboard** 視覺化結果

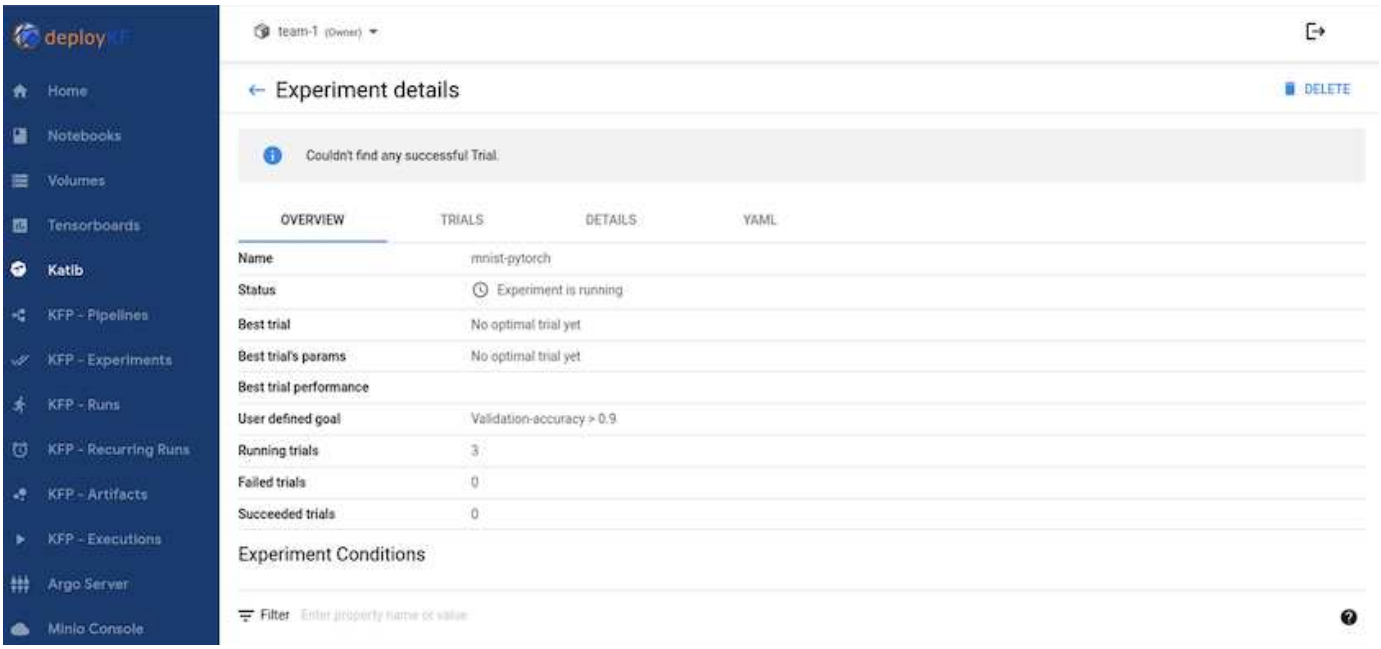
一旦模型經過訓練、我們就可以使用 Tensorboard 將結果視覺化。 "Tensorboard" 可在 Kubeflow 儀表板上作為功能使用。您可以為工作建立自訂的浮動期管理板。以下範例顯示訓練準確度與的繪圖時期和訓練損失與時期數。



使用 **Katib** 嘗試使用 **Hyperparameters**

"Katib" 是 Kubeflow 內的一項工具、可用於實驗模型超參數。若要建立實驗、請先定義所需的指標 / 目標。這通常是測試準確度。一旦定義了度量、請選擇您想要使用的超參數（最佳化器 / 學習率 / 層數）。Katib 會使用使用者定義的值執行超參數掃描、以找出符合所需度量的最佳參數組合。您可以在 UI 的每個區段中定義這些參數。或者、您也可以使用必要的規格來定義 * YAML* 檔案。以下是 Katib 實驗的圖例：





使用 **NetApp Snapshot** 來儲存資料以進行追蹤

在模型訓練期間、我們可能想要儲存訓練資料集的快照、以便追蹤。為達成此目標、我們可以將快照步驟新增至管道、如下所示。若要建立快照、我們可以使用 "適用於Kubernetes的NetApp DataOps工具套件"。

```
@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)
```

請參閱 "NetApp DataOps Toolkit Kubeflow 範例" 以取得更多資訊。

Apache Airflow

Apache Airflow部署

本節說明在Kubernetes叢集中部署氣流時、必須完成的工作。

> ⓘ 您可以在Kubernetes以外的平台上部署氣流。在Kubernetes以外的平台上部署氣流、不在本解決方案的範圍之內。

先決條件

在您執行本節所述的部署練習之前、我們假設您已經執行下列工作：

1. 您已經擁有有效的Kubernetes叢集。
2. 您已在 Kubernetes 叢集中安裝並設定 NetApp Astra Trident 。如需 Astra Trident 的詳細資訊、請參閱 "Astra Trident文件"。

安裝**Helm**

我們使用適用於Kubernetes的常用套件管理程式Helm來部署氣流。部署氣流之前、您必須先在部署跨接主機上安裝Helm。若要在部署跳接主機上安裝Helm、請遵循 "安裝說明" 在官方Helm文件中。

設定預設**Kubernetes StorageClass**

在部署氣流之前、您必須在Kubernetes叢集中指定預設StorageClass。氣流部署程序會嘗試使用預設StorageClass來配置新的持續磁碟區。如果沒有將StorageClass指定為預設StorageClass、則部署將會失敗。若要在叢集中指定預設 StorageClass 、請遵循中所述的指示 "Kubeflow部署" 區段。如果您已在叢集內指定預設StorageClass、則可以跳過此步驟。

使用**Helm**來部署氣流

若要使用Helm在Kubernetes叢集中部署氣流、請從部署跨接主機執行下列工作：

1. 請遵循、使用Helm來部署氣流 "部署指示" 以取得雜訊中心的官方氣流圖表。以下命令範例顯示如何使用Helm部署氣流。視您的環境和所需組態而定、視需要修改、新增及/或移除「custom-vales.yaml」檔案中的值。

```
$ 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
```

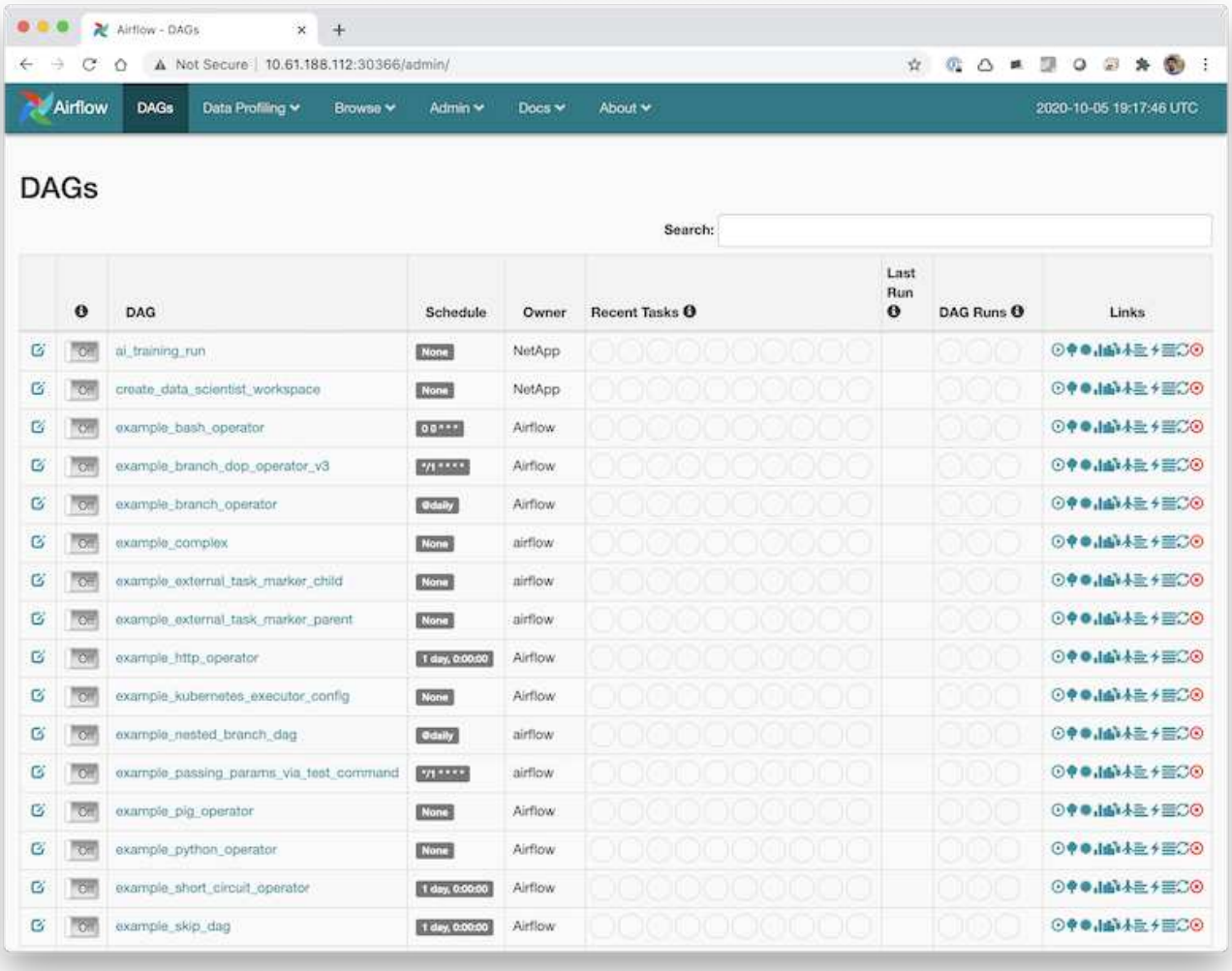
2. 確認所有的氣流網墊都已啟動且正常運作。所有Pod可能需要幾分鐘的時間才能啟動。

```
$ 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. 請依照步驟1中使用Helm部署氣流時列印至主控台的指示、取得氣流Web服務URL。

```
$ 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. 確認您可以存取氣流Web服務。



使用 **NetApp DataOps** 工具套件搭配氣流

。 "適用於Kubernetes的NetApp DataOps工具套件" 可搭配氣流一起使用。使用 NetApp DataOps Toolkit 搭配氣流、您可以將 NetApp 資料管理作業（例如建立快照和複本）整合到由氣流協調的自動化工作流程中。

請參閱 "氣流範例" NetApp DataOps Toolkit GitHub 儲存庫中的一節、詳細說明如何搭配氣流使用工具組。

Astra Trident 營運範例

本節包含您可能想要與 Astra Trident 一起執行的各種作業範例。

匯入現有**Volume**

如果您的NetApp儲存系統/平台上有您要掛載到Kubernetes叢集內的容器上的現有磁碟區、但這些磁碟區並未繫結到叢集中的PVCS、則您必須匯入這些磁碟區。您可以使用Trident Volume匯入功能匯入這些Volume。

以下命令範例顯示匯入名為的 Volume `pb_fg_all`。如需PVCS的詳細資訊、請參閱 "Kubernetes官方文件"。如需Volume匯入功能的詳細資訊、請參閱 "Trident文件"。

在範例的PVc規格檔案中、會指定「存取模式」值「ReadOnlyMany」。如需「存取模式」欄位的詳細資訊、請參閱 "Kubernetes官方文件"。

```
$ 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
```

配置新Volume

您可以使用Trident在NetApp儲存系統或平台上配置新磁碟區。

使用 **kubectl** 配置新 **Volume**

下列命令範例顯示使用 kubectl 來佈建新的 FlexVol Volume 。

以下範例的PVc定義檔中指定「存取模式」值「ReadWriteMany」。如需「存取模式」欄位的詳細資訊、請參閱"Kubernetes官方文件"。

```
$ 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
```

使用 **NetApp DataOps Toolkit** 佈建新 **Volume**

您也可以使用 NetApp DataOps Toolkit for Kubernetes 、在 NetApp 儲存系統或平台上佈建新的磁碟區。NetApp DataOps Toolkit for Kubernetes 利用 Trident 來配置 Volume 、但能簡化使用者的程序。請參閱 "文件" 以取得詳細資料。

適用於 AIPod 部署的高效能工作範例

執行單節點AI工作負載

若要在Kubernetes叢集中執行單節點AI和ML工作、請從部署跳接主機執行下列工作。有了Trident、您就能快速輕鬆地建立資料磁碟區、讓Kubernetes工作負載能夠存取可能含有PB資料的資料。若要從Kubernetes Pod中存取此類資料磁碟區、只需在Pod定義中指定一個PVc即可。

本節假設您已將您嘗試在Kubernetes叢集中執行的特定AI和ML工作負載（採用Docker容器格式）容器化。

1. 下列命令範例顯示使用ImageNet資料集的TensorFlow基準測試工作負載建立Kubernetes工作。如需ImageNet資料集的詳細資訊、請參閱 "ImageNet網站"。

   此範例工作要求八個GPU、因此可在單一GPU工作節點上執行、該工作節點具備八個或更多GPU。此範例工作可在叢集中提交、而具有八個以上GPU的工作節點不存在、或目前正與其他工作負載一起使用。如果是、則工作會維持在擱置狀態、直到該工作者節點可供使用為止。

   此外、為了將儲存頻寬最大化、包含所需訓練資料的磁碟區會在本工作所建立的Pod內掛載兩次。另外一個Volume也會掛載在Pod中。第二個磁碟區將用於儲存結果和指標。這些磁碟區會使用PVCS名稱在工作定義中參考。如需Kubernetes工作的詳細資訊、請參閱 "Kubernetes官方文件"。

   此範例所建立的Pod中、會將「medium」值為「mory」的「emtyDir"磁碟區掛載到「開發/ shm」。Docker Container執行時間所自動建立的「/dev/shm」虛擬磁碟區的預設大小、有時可能不足以滿足TensorFlow的需求。如以下範例所示、掛載「empyDir"磁碟區可提供足夠大的「/dev/shm」虛擬磁碟區。如需有關「empyDir" Volume的詳細資訊、請參閱 "Kubernetes官方文件"。

   在此範例工作定義中所指定的單一容器、其「優先」值為「true」。此值表示容器有效擁有主機的root存取權。在這種情況下會使用此註釋、因為執行的特定工作負載需要root存取權。具體而言、工作負載執行的清除快取作業需要root存取權。是否需要這種「特殊權限：真」註解、取決於您執行的特定工作負載需求。

```
$ 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. 確認您在步驟1中建立的工作正在正確執行。下列範例命令可確認已為工作建立單一Pod（如工作定義所指定）、而且此Pod目前正在其中一個GPU工作節點上執行。

```
$ 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. 確認您在步驟1中建立的工作已成功完成。下列命令範例可確認工作已成功完成。

```
$ 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. *選用：*清除工作成品。下列命令範例顯示刪除在步驟1中建立的工作物件。

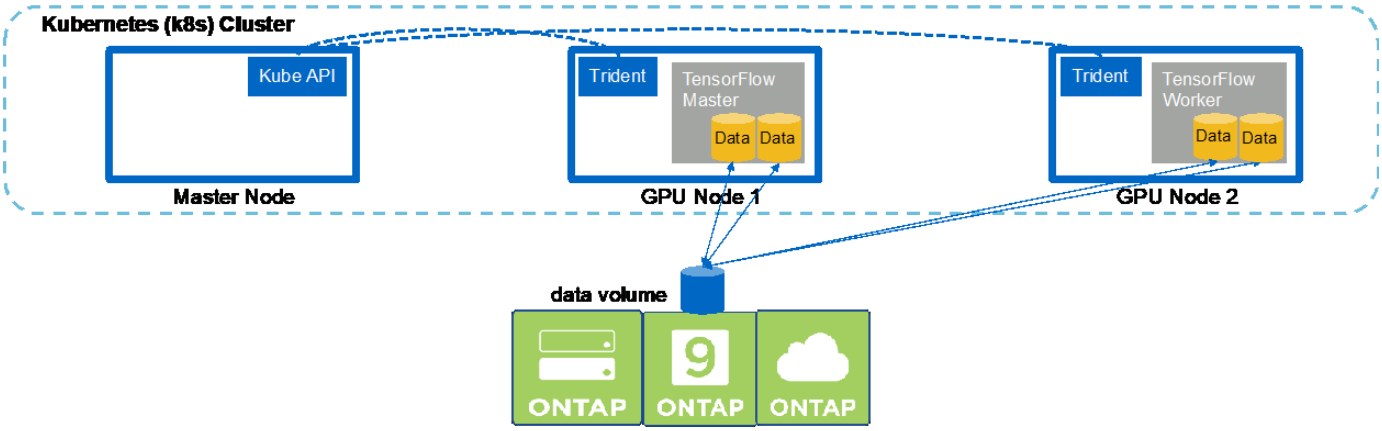
   刪除工作物件時、Kubernetes會自動刪除任何相關的Pod。

```
$ 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.
```

執行同步分散式AI工作負載

若要在Kubernetes叢集中執行同步多節點AI和ML工作、請在部署跨接主機上執行下列工作。此程序可讓您利用儲存在NetApp磁碟區上的資料、並使用比單一工作節點更多的GPU。如需同步分散式AI工作的說明、請參閱下圖。

相較於非同步分散式工作、同步分散式工作有助於提升效能和訓練準確度。關於同步工作與非同步工作的優缺點的討論、不在本文的討論範圍之內。



1. 下列命令範例顯示建立一個工作者、以參與本節範例中單一節點上執行的相同TensorFlow基準測試工作之同步分散式執行 "執行單節點AI工作負載"。在此特定範例中、只會部署一名員工、因為該工作會在兩個工作節點之間執行。

   此範例的工作者部署要求八個GPU、因此可在單一GPU工作者節點上執行、該節點具備八個以上的GPU。如果GPU工作節點的GPU功能超過八個GPU、為了發揮最大效能、您可能想要增加此數目、使其等於工作節點所使用的GPU數量。如需Kubernetes部署的詳細資訊、請參閱 "Kubernetes官方文件"。

在此範例中會建立Kubernetes部署、因為這個特定的容器化工作者永遠不會自行完成。因此、使用Kubernetes工作架構來部署IT並不合理。如果您的員工是自行設計或撰寫完成、則使用工作架構來部署您的員工可能是合理的做法。

本範例部署規格中所指定的Pod、其「hostNetwork」值為「true」。此值表示Pod使用主機工作節點的網路堆疊、而非Kubernetes通常為每個Pod建立的虛擬網路堆疊。此註釋用於此案例、因為特定工作負載仰賴Open MPI、NCCL和Horovod以同步分散的方式執行工作負載。因此、它需要存取主機網路堆疊。關於Open MPI、NCCL和Horovod的討論不在本文的討論範圍之內。是否需要此「hostNetwork：true」註釋、取決於您執行的特定工作負載需求。如需有關「hostNetwork」欄位的詳細資訊、請參閱 "Kubernetes官方文件"。

```
$ 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. 確認您在步驟1中建立的工作者部署已成功啟動。下列命令範例可確認已針對部署建立單一工作者Pod、如部署定義所示、而且此Pod目前正在其中一個GPU工作者節點上執行。

```
$ 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. 為啟動、參與及追蹤同步多節點工作執行的主節點建立Kubernetes工作。下列命令範例可建立一個主磁片、用於啟動、參與及追蹤同一個TensorFlow基準測試工作的同步分散式執行、該工作是在本節範例的單一節點上執行 "執行單節點AI工作負載"。

   此範例主要工作要求八個GPU、因此可在具有八個以上GPU的單一GPU工作節點上執行。如果GPU工作節點的GPU功能超過八個GPU、為了發揮最大效能、您可能想要增加此數目、使其等於工作節點所使用的GPU數量。

   本範例工作定義中所指定的主Pod、其「主機網路」值為「真」、就如同在步驟1中給工作群組「主機網路」值「真」一樣。請參閱步驟1、瞭解為何需要此值的詳細資訊。

```
$ 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. 確認您在步驟3中建立的主要工作正在正確執行。下列範例命令可確認已為工作建立單一主Pod、如工作定義所示、而且此Pod目前正在其中一個GPU工作節點上執行。您也應該看到、您在步驟1中看到的工作者Pod仍在執行中、而且主要和工作者Pod正在不同的節點上執行。

```
$ 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. 確認您在步驟3中建立的主要工作已成功完成。下列命令範例可確認工作已成功完成。

```
$ 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. 當您不再需要部署時、請刪除該員工部署。下列命令範例顯示刪除在步驟1中建立的工作者部署物件。

   當您刪除工作者部署物件時、Kubernetes會自動刪除任何關聯的工作者Pod。

```
$ 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. *選用：*清除主要工作成品。下列命令範例顯示刪除在步驟3中建立的主要工作物件。

   刪除主工作物件時、Kubernetes會自動刪除任何相關的主Pod。

```
$ 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.
```