NetApp AI Control Plane

NetApp Solutions

NetApp
April 12, 2024

目錄

NetApp AI Control Plane . . . 1
TR-4798：NetApp AI Control Plane . . . 1
概念與元件 . . . 2
硬體與軟體需求 . . . 9
Kubernetes部署 . . . 10
NetApp Trident部署與組態 . . . 11
Kubeflow部署 . . . 17
Kubeflow作業與工作範例 . . . 26
Apache Airflow部署 . . . 34
Apache氣流工作流程範例 . . . 37
Trident作業範例 . . . 37
適用於AI部署的高效能工作範例ONTAP . . . 40
效能測試 . . . 51
結論 . . . 51

NetApp AI Control Plane

TR-4798：NetApp AI Control Plane

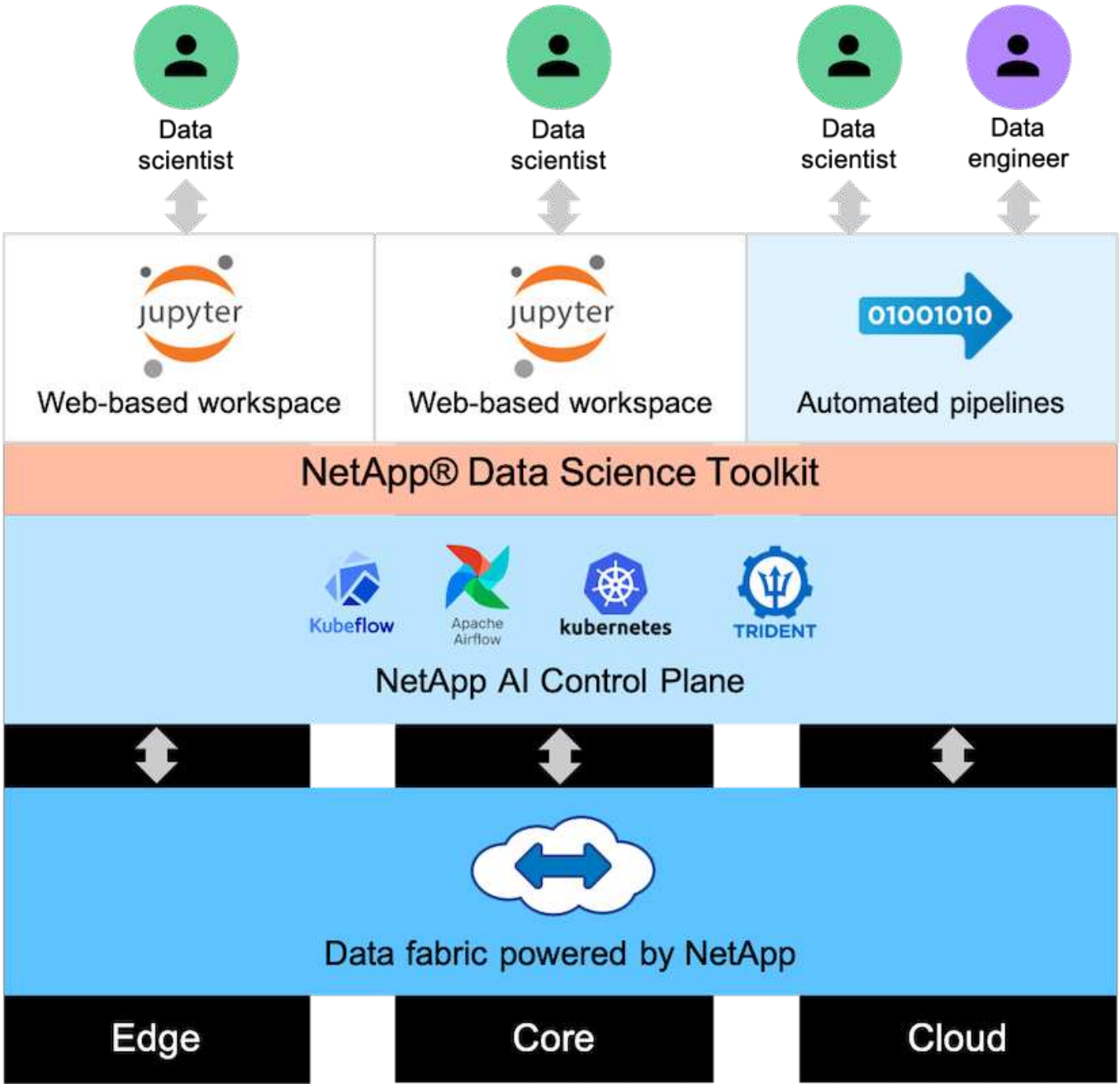
Mike Oglesby、NetApp

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。本文件說明如何使用NetApp AI Control Plane解決這些挑戰、這套解決方案將NetApp資料管理功能與熱門的開放原始碼工具和架構配對。

本報告說明如何快速複製資料命名空間。同時也說明如何在站台和區域之間無縫複寫資料、以建立一致且統一的AI/ML/DL資料傳輸途徑。此外、它還會引導您完成AI、ML和DL訓練工作流程的定義與實作、這些工作流程整合了近乎即時的資料建立與模型基準、以利追蹤及版本管理。有了這套解決方案、您可以追蹤每個模型訓練回溯到用來訓練和/或驗證模型的確切資料集。最後、本文件說明如何快速配置Jupyter Notebook工作區、並存取大量資料集。

附註：針對需要共用存取相同資料集的大量GPU伺服器大規模的HPC形式分散式訓練、或是如果您需要/偏好平行檔案系統、請查看 "TR-4890"。本技術報告說明如何納入 "NetApp完全支援的平行檔案系統解決方案BeeGFS" 作為NetApp AI Control Plane的一部分。此解決方案可從少數NVIDIA DGX A100系統擴充至完整的140節點SupermPOD。

NetApp AI Control Plane的目標對象是資料科學家和資料工程師、ONTAP 因此需要最少的NetApp或NetApp®專業知識。有了這套解決方案、您就能使用簡單且熟悉的工具和介面來執行資料管理功能。如果您的環境中已經有NetApp儲存設備、您可以立即試用NetApp AI控制面板。如果您想試用解決方案、但還沒有NetApp儲存設備、請造訪 "cloud.netapp.com"而且您可以在幾分鐘內使用雲端型NetApp儲存解決方案來啟動和執行。下圖提供解決方案的視覺化功能。



概念與元件

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

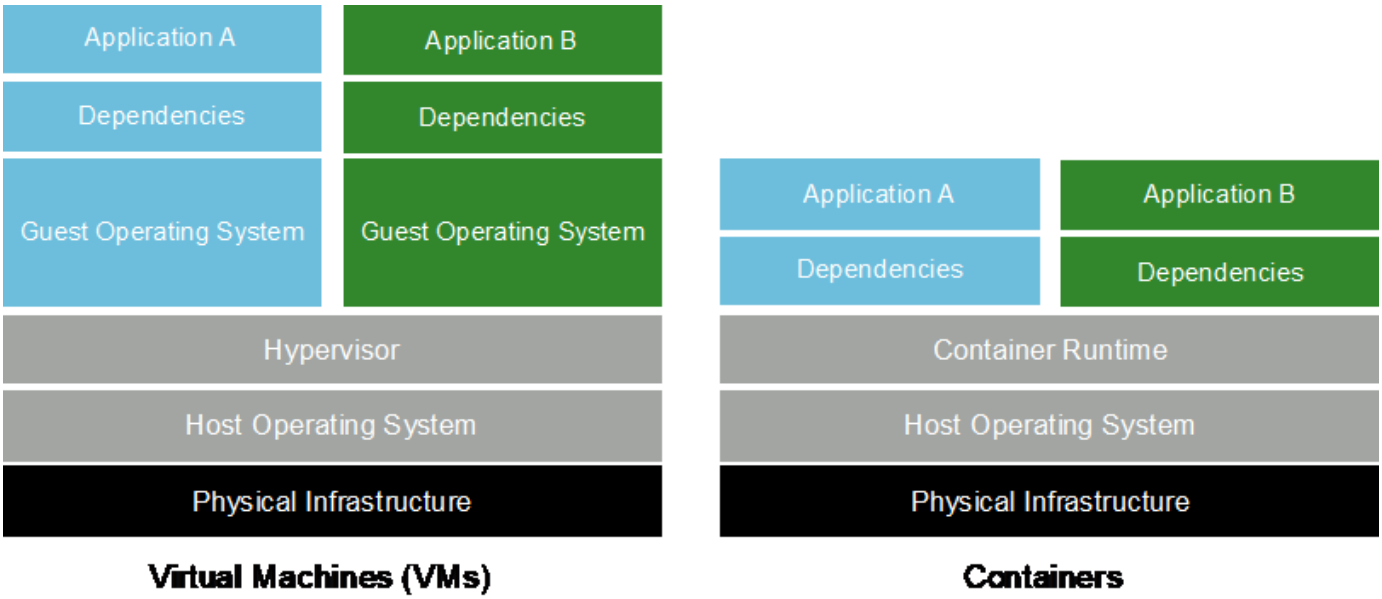
現代化的AI訓練和推斷工作負載需要大量平行運算功能。因此、GPU越來越常用於執行AI作業、因為GPU的平

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。雖然支援其他容器封裝格式和執行時間、但Kubernetes通常是Docker容器的協調系統。如需詳細資訊、請參閱 "Kubernetes網站"。

NetApp Trident

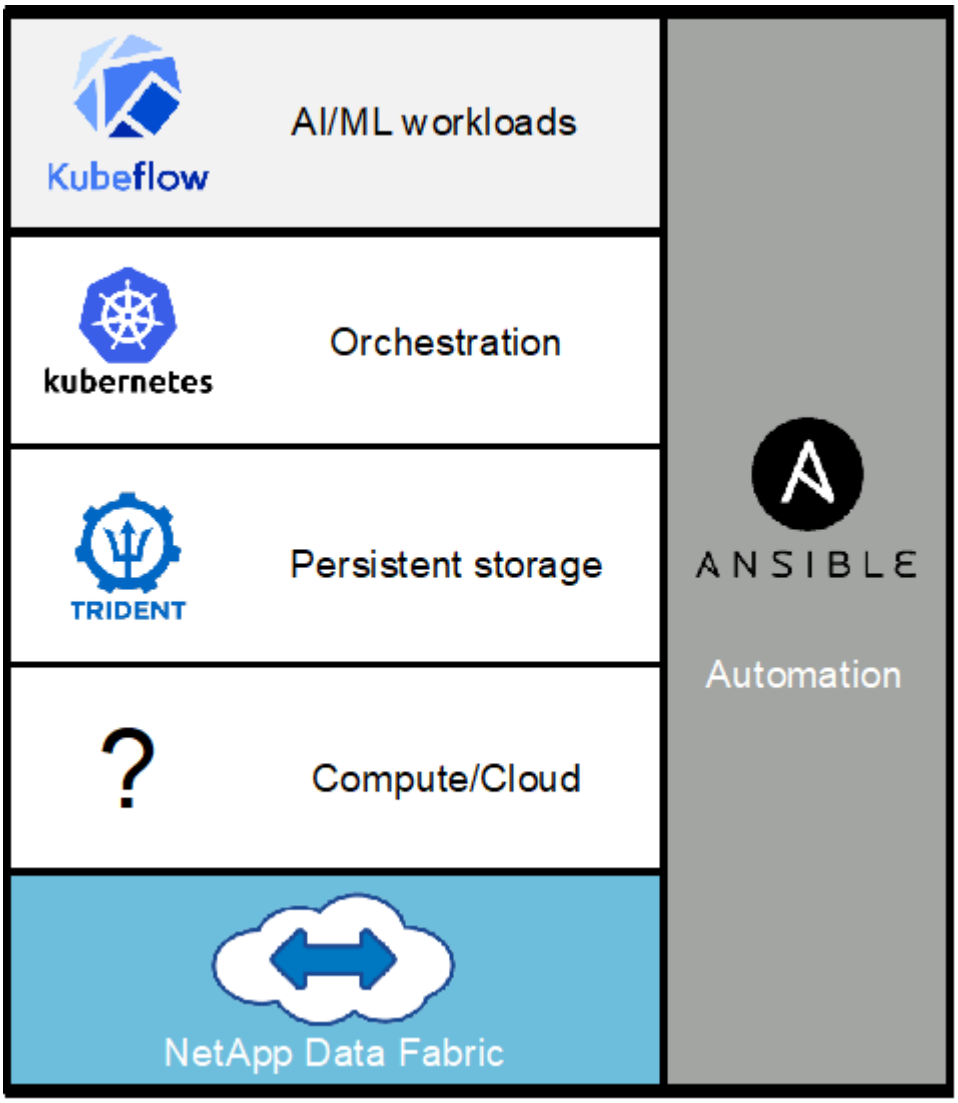
Trident是NetApp開發與維護的開放原始碼儲存協調工具、可大幅簡化Kubernetes工作負載的持續儲存設備建立、管理與使用。Trident本身是Kubernetes原生應用程式、直接在Kubernetes叢集內執行。Kubernetes使用者（開發人員、資料科學家、Kubernetes系統管理員等）可以使用他們已經熟悉的標準Kubernetes格式、建立、管理及與持續儲存磁碟區互動。同時、他們也能善用NetApp先進的資料管理功能、以及採用NetApp技術的資料架構。Trident將持續儲存設備的複雜度抽象化、使其易於使用。如需詳細資訊、請參閱 "Trident網站"。

NVIDIA DeepOps

DeepOps是NVIDIA的開放原始碼專案、使用Ansible可根據最佳實務做法、自動部署GPU伺服器叢集。DeepOps是模組化的、可用於各種部署工作。本文件及其所說明的驗證作業中、DeepOps用於部署Kubernetes叢集、其中包含GPU伺服器工作節點。如需詳細資訊、請參閱 "DeepOps網站"。

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複雜的資料擷取出來、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。由於企業IT部門越來越標準化Kubernetes、Kubeflow的發展也越來越顯著。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"。

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 產品9.

NetApp ONTAP 支援NetApp的最新一代儲存管理軟體、可讓像您這樣的企業將基礎架構現代化、並移轉至雲端就緒的資料中心。藉由領先業界的資料管理功能、ONTAP 無論資料位於何處、您都能使用單一工具組來管理及保護資料。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。包含許多功能、可簡化資料管理、加速並保護關鍵資料、以及跨越混合雲架構的符合未來需求的基礎架構。ONTAP

簡化資料管理

資料管理對企業IT營運至關重要、因此您可以將適當的資源用於應用程式和資料集。包含下列功能、可簡化及簡化作業、並降低營運總成本：ONTAP

- *即時資料壓縮與擴充重複資料刪除技術。*資料壓縮技術可減少儲存區塊內的空間浪費、重複資料刪除技術則可大幅提升有效容量。
- *服務品質（QoS）的最低、最大和調適性。*精細的QoS控制可協助維持高共享環境中關鍵應用程式的效能等級。
- *此功能可將冷資料自動分層至公有雲和私有雲儲存選項、包括Amazon Web Services（AWS）、Azure和NetApp以物件為基礎的儲存設備。ONTAP FabricPool StorageGRID

加速並保護資料

提供優異的效能與資料保護、並透過下列功能來延伸這些功能：ONTAP

- *高效能與低延遲。ONTAP *
- * NetApp ONTAP FlexGroup 功能* FlexGroup 。*一個功能強大的資料容器、可線性擴充至高達20PB和4000億個檔案、提供簡化資料管理的單一命名空間。
- 資料保護 ONTAP 效能*功能提供內建的資料保護功能、並可在所有平台上進行通用管理。
- * NetApp Volume Encryption。ONTAP *支援內建和外部金鑰管理、提供原生Volume層級的加密功能。

符合未來需求的基礎架構

支援需求嚴苛且瞬息萬變的企業需求：ONTAP

- 無縫擴充與不中斷營運。 ONTAP 支援在不中斷營運的情況下、為現有控制器和橫向擴充叢集增加容量。您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。

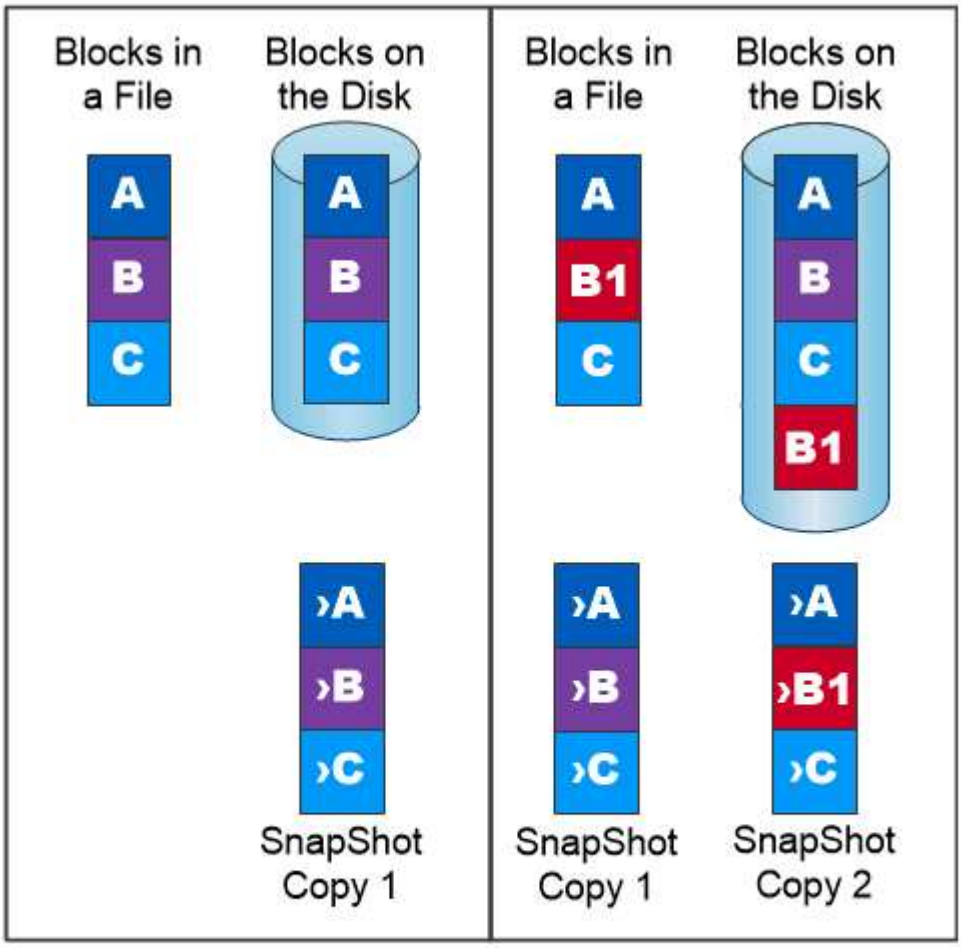
- * Cloud connection。* ONTAP 效能不只是雲端連線能力最強的儲存管理軟體之一、ONTAP Select 還可在Cloud Volumes Service 所有公有雲中選擇軟體定義儲存（英文）和雲端原生執行個體（NetApp邊）。
- 整合新興應用程式。 ONTAP 使用支援現有企業應用程式的相同基礎架構、即可為OpenStack、Hadoop和MongoDB等新一代平台和應用程式提供企業級資料服務。

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。

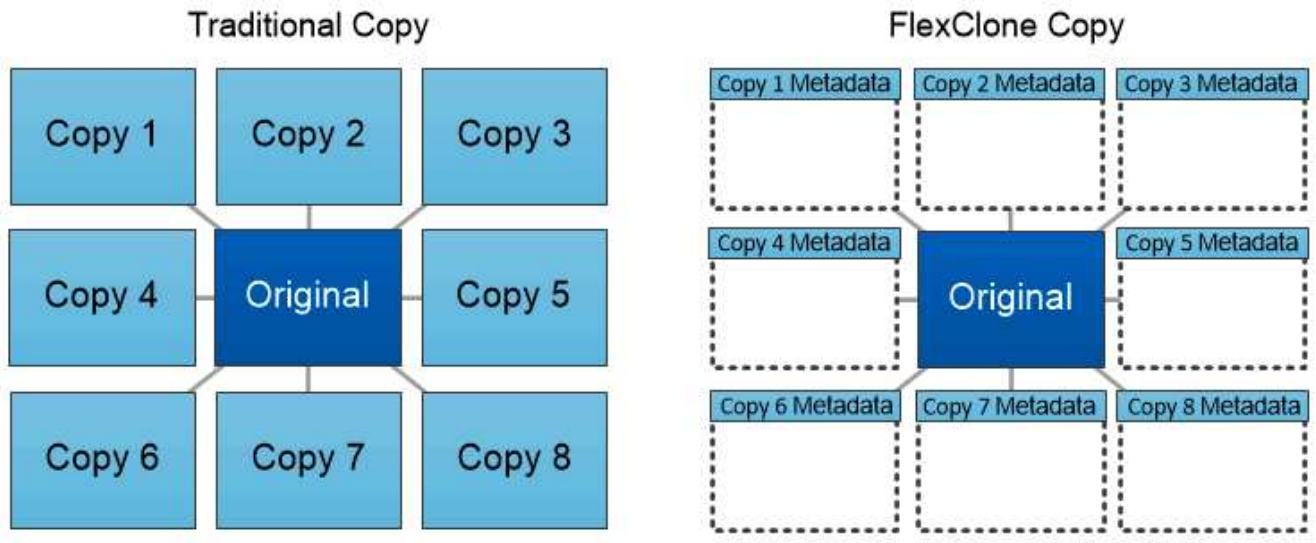
您可以使用Snapshot複本來還原個別檔案或LUN、或還原磁碟區的完整內容。此功能可將Snapshot複本中的指標資訊與磁碟上的資料進行比較、以重建遺失或損壞的物件、而不會造成停機或重大效能成本。ONTAP



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。

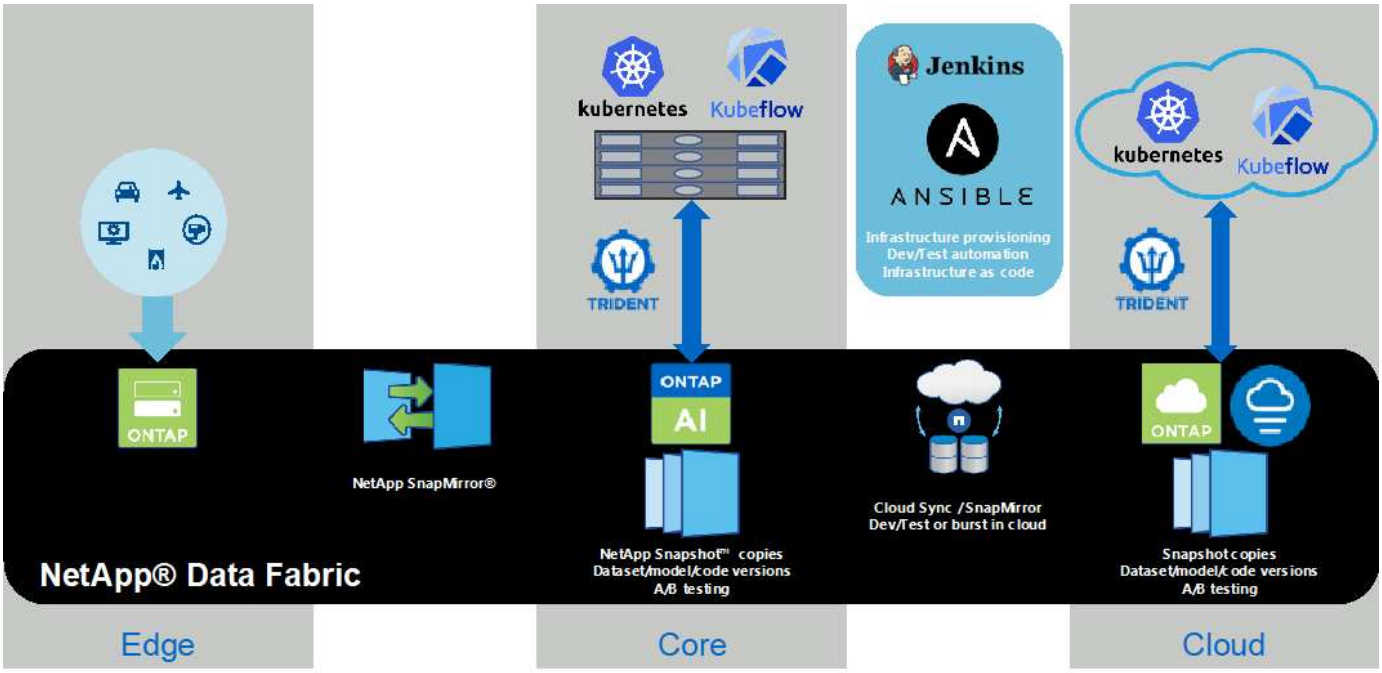


FlexClone copies share data blocks with their parents, consuming no storage except what is required for metadata.

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。

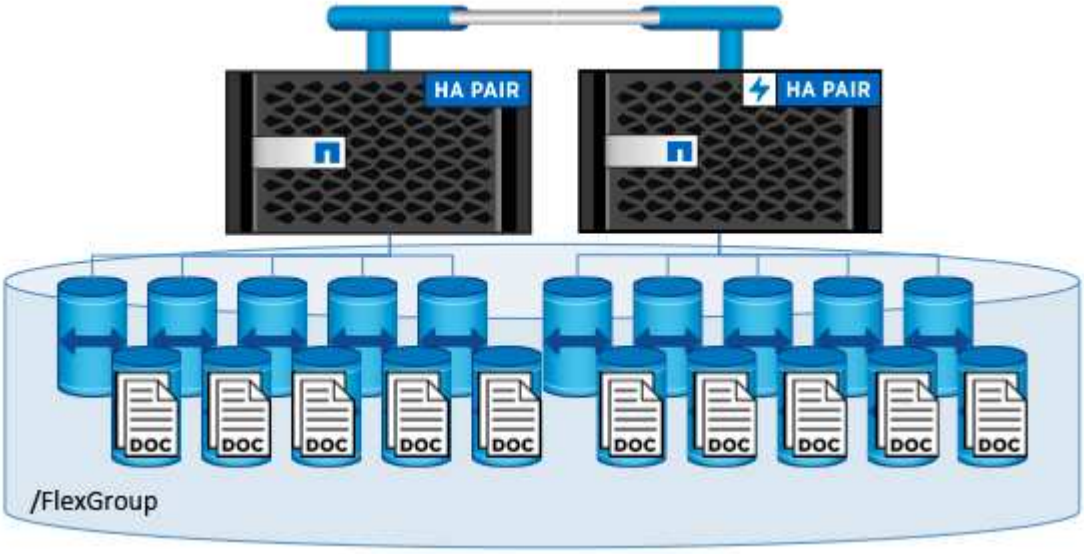
NetApp ONTAP FlexGroup 產品區

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。



硬體與軟體需求

NetApp AI Control Plane解決方案不依賴於此特定硬體。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 的不完整儲存系統Azure NetApp Files 、功能完善、NetApp Cloud Volumes Service 功能完善、NetApp ONTAP Select 功能完善、由軟體定義的儲存執行個體、或是NetApp Cloud Volumes ONTAP 的不完整執行個體。此外、只要Kubernetes和NetApp Trident支援所使用的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。

基礎架構元件	數量	詳細資料	作業系統
部署跳接主機	1.	VM	Ubuntu 20.04.2 LTS
Kubernetes主節點	1.	VM	Ubuntu 20.04.2 LTS
Kubernetes工作節點	2.	VM	Ubuntu 20.04.2 LTS
Kubernetes GPU工作節點	2.	NVIDIA DGX-1（裸機）	NVIDIA DGX OS 4.0.5（以Ubuntu 18.04.2 LTS為基礎）

基礎架構元件	數量	詳細資料	作業系統
儲存設備	1個HA配對	NetApp AFF 解決方案-A220	NetApp ONTAP 產品9.7 P6

軟體元件	版本
Apache Airflow	2.0.1
Apache Airflow Helm圖表	8.0.8
Docker	19.03.12
Kubeflow	1.2
Kubernetes	1.18.9.
NetApp Trident	21.01.2
NVIDIA DeepOps	主分支機構在提交時的Trident部署功能 "61898cdfda"；所有其他功能（版本21.03）

支援

NetApp不提供Apache Airflow、Docker、Kubeflow、Kubernetes或NVIDIA DeepOps的企業支援。如果您對具備類似NetApp AI Control Plane解決方案功能的完整支援解決方案感興趣、 "請聯絡NetApp" 關於NetApp與合作夥伴共同提供的完全支援的AI/ML解決方案。

Kubernetes部署

本節說明部署Kubernetes叢集所需完成的工作、以實作NetApp AI Control Plane解決方案。如果您已經有Kubernetes叢集、則只要您執行Kubernetes版本、且Kubeflow和NetApp Trident均支援此版本、就可以跳過本節。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"。

對於採用裸機節點的內部部署Kubernetes、採用NVIDIA GPU、NetApp建議使用NVIDIA的DeepOps Kubernetes部署工具。本節概述如何使用DeepOps部署Kubernetes叢集。

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. 您已根據ONTAP 標準組態指示、設定任何裸機Kubernetes節點（例如、NVIDIA DGX系統是整個AI Pod的一部分）。
2. 您已在所有Kubernetes主節點和工作節點、以及部署跨接主機上安裝支援的作業系統。如需DeepOps支援的作業系統清單、請參閱 "DeepOps GitHub網站"。

使用NVIDIA DeepOps來安裝及設定Kubernetes

若要使用NVIDIA DeepOps部署及設定Kubernetes叢集、請從部署跳接主機執行下列工作：

1. 依照上的指示下載NVIDIA DeepOps "入門頁面" 在NVIDIA DeepOps GitHub網站上。
2. 依照上的指示、在叢集中部署Kubernetes "Kubernetes部署指南頁面" 在NVIDIA DeepOps GitHub網站上。

NetApp Trident部署與組態

NetApp Trident部署與組態

本節說明在Kubernetes叢集中安裝及設定NetApp Trident所需完成的工作。

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. 您已經有一個正常運作的Kubernetes叢集、而且您正在執行Trident支援的Kubernetes版本。如需支援版本的清單、請參閱 "Trident文件"。
2. 您已經擁有Trident支援的有效NetApp儲存設備、軟體定義執行個體或雲端儲存服務。

安裝**Trident**

若要在Kubernetes叢集中安裝及設定NetApp Trident、請從部署跨接主機執行下列工作：

1. 使用下列其中一種方法部署Trident：
   - 如果您使用NVIDIA DeepOps來部署Kubernetes叢集、也可以使用NVIDIA DeepOps在Kubernetes叢集中部署Trident。若要部署Trident with DeepOps、請遵循 "Trident部署指示" 在NVIDIA DeepOps GitHub網站上。
   - 如果您沒有使用NVIDIA DeepOps來部署Kubernetes叢集、或是想要手動部署Trident、您可以遵循來部署Trident "部署指示" 在Trident文件中。請務必至少建立一個Trident Backend和至少一個Kubernetes StorageClass、以取得如何設定的詳細資訊 "後端" 和 "儲存類別" 請參閱NetApp文件中的連結小節。

> 如果您要在ONTAP 一個AI Pod上部署NetApp AI Control Plane解決方案、請參閱 "Trident後端範例、適用於ONTAP AI部署" 如需您可能想要建立和的不同Trident後端的一些範例 "Kubernetes Storageclasses範例、可用於ONTAP 進行AI部署" 如需您可能想要建立的不同Kubernetes StorageClass範例。

Trident後端範例、適用於ONTAP AI部署

您必須先建立一或多個Trident後端、才能使用Trident在Kubernetes叢集中動態配置儲存資源。以下範例代表在ONTAP 將NetApp AI Control Plane解決方案部署到AI Pod上時、您可能會想要建立的不同類型後端。如需後端的詳細資訊、請參閱 "Trident文件"。

1. NetApp建議針對您想要在NetApp AFF 供應系統上使用的每個資料LIF（提供資料存取的邏輯網路介面）、建立具有FlexGroup功能的Trident後端。這可讓您在所有生命期之間平衡Volume掛載

   以下命令範例顯示為兩個不同的資料LIF建立兩個啟用FlexGroup的Trident後端、這些資料LIF與相同ONTAP的物件儲存虛擬機器（SVM）相關聯。這些後端使用「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-ontap-ai-flexgroups-iface1.json
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": "ontap-ai-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-ontap-ai-flexgroups-
iface1.json -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |
UUID                 | STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
$ cat << EOF > ./trident-backend-ontap-ai-flexgroups-iface2.json
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": "ontap-ai-flexgroups-iface2",
    "managementLIF": "10.61.218.100",
    "dataLIF": "192.168.12.12",
    "svm": "ontapai_nfs",
    "username": "admin",
    "password": "ontapai"
}
EOF
$ tridentctl create backend -f ./trident-backend-ontap-ai-flexgroups-
iface2.json -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
```

```
|            NAME             |   STORAGE DRIVER    |
UUID                 | STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexgroups-iface2 | ontap-nas-flexgroup | 61814d48-c770-436b-
9cb4-cf7ee661274d | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME             |   STORAGE DRIVER    |
UUID                 | STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
| ontap-ai-flexgroups-iface2 | ontap-nas-flexgroup | 61814d48-c770-436b-
9cb4-cf7ee661274d | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
```

2. NetApp也建議您建立一個或多個FlexVol 啟用了功能不全的Trident後端。如果您使用FlexGroup 支援資料集儲存設備的功能來進行測試、您可能會想要使用FlexVol 支援資料集的功能來儲存結果、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。以下的命令範例顯示如何建立FlexVol 使用單一資料LIF的單一啟用了功能不全的Trident後端。

```
$ cat << EOF > ./trident-backend-ontap-ai-flexvols.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-ai-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-ontap-ai-flexvols.json -n
trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-
a9c1-52a69657fabe | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
$ tridentctl get backend -n trident
+----------------------------+---------------------
+--------------------------------------+--------+---------+
|            NAME            |   STORAGE DRIVER    |                 UUID
| STATE  | VOLUMES |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
| ontap-ai-flexvols          | ontap-nas           | 52bdb3b1-13a5-4513-
a9c1-52a69657fabe | online |       0 |
| ontap-ai-flexgroups-iface1 | ontap-nas-flexgroup | b74cbddb-e0b8-40b7-
b263-b6da6dec0bdd | online |       0 |
| ontap-ai-flexgroups-iface2 | ontap-nas-flexgroup | 61814d48-c770-436b-
9cb4-cf7ee661274d | online |       0 |
+----------------------------+---------------------
+--------------------------------------+--------+---------+
```

Kubernetes StorageClass範例ONTAP 、可用於進行AI部署

您必須先建立一或多個Kubernetes StorageClass、才能使用Trident在Kubernetes叢集中動態配置儲存資源。以下範例代表在ONTAP 將NetApp AI Control Plane解決方案部署到AI

Pod上時、您可能會想要建立的不同類型StorageClass。如需StorageClass的詳細資訊、請參閱 "Trident文件"。

1. NetApp建議為您在本節中建立的每個啟用FlexGroup的Trident後端、建立個別的StorageClass "Trident後端範例、適用於ONTAP AI部署"步驟1.這些精細的StorageClass可讓您新增對應於特定LIF（建立Trident後端時所指定的LIF）的NFS掛載、做為StorageClass規格檔案中指定的特定後端。以下命令範例顯示建立兩個StorageClass的方式、這兩個類別對應於本節中建立的兩個範例後端 "Trident後端範例、適用於ONTAP AI部署"步驟1.如需StorageClass的詳細資訊、請參閱 "Trident文件"。

   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"。

```
$ cat << EOF > ./storage-class-ontap-ai-flexgroups-retain-iface1.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexgroups-retain-iface1
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "ontap-ai-flexgroups-iface1:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexgroups-retain-
iface1.yaml
storageclass.storage.k8s.io/ontap-ai-flexgroups-retain-iface1 created
$ cat << EOF > ./storage-class-ontap-ai-flexgroups-retain-iface2.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexgroups-retain-iface2
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "ontap-ai-flexgroups-iface2:.*"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexgroups-retain-
iface2.yaml
storageclass.storage.k8s.io/ontap-ai-flexgroups-retain-iface2 created
$ kubectl get storageclass
NAME                                PROVISIONER         AGE
ontap-ai-flexgroups-retain-iface1   netapp.io/trident   0m
ontap-ai-flexgroups-retain-iface2   netapp.io/trident   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 "Trident後

端範例、適用於ONTAP AI部署"步驟2：以下命令範例顯示建立FlexVol 單一StorageClass for the餐廳。

在下列範例中、由於只建立一個具有FlexVol功能的Trident後端、因此不會在StorageClass定義檔案中指定特定的後端。當您使用Kubernetes來管理使用此StorageClass的磁碟區時、Trident會嘗試使用任何使用「ONTAP-NAS」驅動程式的可用後端。

```
$ cat << EOF > ./storage-class-ontap-ai-flexvols-retain.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexvols-retain
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexvols-retain.yaml
storageclass.storage.k8s.io/ontap-ai-flexvols-retain created
$ kubectl get storageclass
NAME                                PROVISIONER         AGE
ontap-ai-flexgroups-retain-iface1   netapp.io/trident   1m
ontap-ai-flexgroups-retain-iface2   netapp.io/trident   1m
ontap-ai-flexvols-retain            netapp.io/trident   0m
```

3. NetApp也建議建立FlexGroup 通用StorageClass以供參考Volume使用。下列命令範例顯示建立FlexGroup 單一通用StorageClass for the餐廳。

   請注意、StorageClass定義檔案中未指定特定的後端。因此、當您使用Kubernetes來管理使用此StorageClass的磁碟區時、Trident會嘗試使用任何使用「ONTAP-NAS-flexgroup」驅動程式的可用後端。

```
$ cat << EOF > ./storage-class-ontap-ai-flexgroups-retain.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-ai-flexgroups-retain
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas-flexgroup"
reclaimPolicy: Retain
EOF
$ kubectl create -f ./storage-class-ontap-ai-flexgroups-retain.yaml
storageclass.storage.k8s.io/ontap-ai-flexgroups-retain created
$ kubectl get storageclass
NAME                                  PROVISIONER         AGE
ontap-ai-flexgroups-retain            netapp.io/trident   0m
ontap-ai-flexgroups-retain-iface1     netapp.io/trident   2m
ontap-ai-flexgroups-retain-iface2     netapp.io/trident   2m
ontap-ai-flexvols-retain              netapp.io/trident   1m
```

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. 您已經有一個正常運作的Kubernetes叢集、而且您正在執行Kubernetes支援的Kubernetes版本。如需支援版本的清單、請參閱 "官方Kubeflow文件"。
2. 您已在Kubernetes叢集中安裝及設定NetApp Trident、如所述 "Trident部署與組態"。

設定預設Kubernetes StorageClass

在部署Kubeflow之前、您必須在Kubernetes叢集中指定預設StorageClass。Kubeflow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。以下命令範例顯示名為「ONTAP-AI - FlexVols - Retain」的StorageClass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
```

使用NVIDIA DeepOps部署Kubeflow

NetApp建議使用NVIDIA DeepOps提供的Kubeflow部署工具。若要使用DeepOps部署工具在Kubernetes叢集中部署Kubeflow、請從部署跳接主機執行下列工作。

> 或者、您也可以依照手動部署Kubeflow "安裝說明" 在正式的Kubeflow文件中

1. 遵循、在叢集中部署Kubeflow "Kubeflow部署指示" 在NVIDIA DeepOps GitHub網站上。
2. 記下DeepOps Kubeflow部署工具所輸出的Kubeflow儀表板URL。

```
$ ./scripts/k8s/deploy_kubeflow.sh -x
…
INFO[0007] Applied the configuration Successfully!
filename="cmd/apply.go:72"
Kubeflow app installed to: /home/ai/kubeflow
It may take several minutes for all services to start. Run 'kubectl get
pods -n kubeflow' to verify
To remove (excluding CRDs, istio, auth, and cert-manager), run:
./scripts/k8s_deploy_kubeflow.sh -d
To perform a full uninstall : ./scripts/k8s_deploy_kubeflow.sh -D
Kubeflow Dashboard (HTTP NodePort): http://10.61.188.111:31380
```

3. 確認Kubeflow命名空間內部署的所有Pod均顯示「執行中」的「狀態」、並確認命名空間內未部署任何元件處於錯誤狀態。所有Pod可能需要幾分鐘的時間才能啟動。

```
$ kubectl get all -n kubeflow
NAME                                                          READY
```

```
STATUS    RESTARTS   AGE
pod/admission-webhook-bootstrap-stateful-set-0                  1/1
Running   0          95s
pod/admission-webhook-deployment-6b89c84c98-vrtbh               1/1
Running   0          91s
pod/application-controller-stateful-set-0                       1/1
Running   0          98s
pod/argo-ui-5dcf5d8b4f-m2wn4                                    1/1
Running   0          97s
pod/centraldashboard-cf4874ddc-7hcr8                            1/1
Running   0          97s
pod/jupyter-web-app-deployment-685b455447-gjhh7                 1/1
Running   0          96s
pod/katib-controller-88c97d85c-kgq66                            1/1
Running   1          95s
pod/katib-db-8598468fd8-5jw2c                                   1/1
Running   0          95s
pod/katib-manager-574c8c67f9-wtrf5                              1/1
Running   1          95s
pod/katib-manager-rest-778857c989-fjbzn                         1/1
Running   0          95s
pod/katib-suggestion-bayesianoptimization-65df4d7455-qthmw      1/1
Running   0          94s
pod/katib-suggestion-grid-56bf69f597-98vwn                      1/1
Running   0          94s
pod/katib-suggestion-hyperband-7777b76cb9-9v6dq                 1/1
Running   0          93s
pod/katib-suggestion-nasrl-77f6f9458c-2qzxq                     1/1
Running   0          93s
pod/katib-suggestion-random-77b88b5c79-l64j9                    1/1
Running   0          93s
pod/katib-ui-7587c5b967-nd629                                   1/1
Running   0          95s
pod/metacontroller-0                                            1/1
Running   0          96s
pod/metadata-db-5dd459cc-swzkm                                  1/1
Running   0          94s
pod/metadata-deployment-6cf77db994-69fk7                        1/1
Running   3          93s
pod/metadata-deployment-6cf77db994-mpbjt                        1/1
Running   3          93s
pod/metadata-deployment-6cf77db994-xg7tz                        1/1
Running   3          94s
pod/metadata-ui-78f5b59b56-qb6kr                                1/1
Running   0          94s
pod/minio-758b769d67-llvdr                                      1/1
```

```
Running   0          91s
pod/ml-pipeline-5875b9db95-g8t2k                                1/1
Running   0          91s
pod/ml-pipeline-persistenceagent-9b69ddd46-bt9r9                1/1
Running   0          90s
pod/ml-pipeline-scheduledworkflow-7b8d756c76-7x56s              1/1
Running   0          90s
pod/ml-pipeline-ui-79ffd9c76-fcwpd                              1/1
Running   0          90s
pod/ml-pipeline-viewer-controller-deployment-5fdc87f58-b2t9r   1/1
Running   0          90s
pod/mysql-657f87857d-l5k9z                                      1/1
Running   0          91s
pod/notebook-controller-deployment-56b4f59bbf-8bvnr             1/1
Running   0          92s
pod/profiles-deployment-6bc745947-mrdkh                         2/2
Running   0          90s
pod/pytorch-operator-77c97f4879-hmlrv                           1/1
Running   0          92s
pod/seldon-operator-controller-manager-0                        1/1
Running   1          91s
pod/spartakus-volunteer-5fdfddb779-l7qkm                        1/1
Running   0          92s
pod/tensorboard-6544748d94-nh8b2                                1/1
Running   0          92s
pod/tf-job-dashboard-56f79c59dd-6w59t                           1/1
Running   0          92s
pod/tf-job-operator-79cbfd6dbc-rb58c                            1/1
Running   0          91s
pod/workflow-controller-db644d554-cwrnb                         1/1
Running   0          97s
NAME                                                   TYPE
CLUSTER-IP      EXTERNAL-IP   PORT(S)              AGE
service/admission-webhook-service                      ClusterIP
10.233.51.169   <none>        443/TCP              97s
service/application-controller-service                 ClusterIP
10.233.4.54     <none>        443/TCP              98s
service/argo-ui                                        NodePort
10.233.47.191   <none>        80:31799/TCP         97s
service/centraldashboard                               ClusterIP
10.233.8.36     <none>        80/TCP               97s
service/jupyter-web-app-service                        ClusterIP
10.233.1.42     <none>        80/TCP               97s
service/katib-controller                               ClusterIP
10.233.25.226   <none>        443/TCP              96s
service/katib-db                                       ClusterIP
```

```
10.233.33.151   <none>        3306/TCP            97s
service/katib-manager                                ClusterIP
10.233.46.239   <none>        6789/TCP            96s
service/katib-manager-rest                           ClusterIP
10.233.55.32    <none>        80/TCP              96s
service/katib-suggestion-bayesianoptimization        ClusterIP
10.233.49.191   <none>        6789/TCP            95s
service/katib-suggestion-grid                        ClusterIP
10.233.9.105    <none>        6789/TCP            95s
service/katib-suggestion-hyperband                   ClusterIP
10.233.22.2     <none>        6789/TCP            95s
service/katib-suggestion-nasrl                       ClusterIP
10.233.63.73    <none>        6789/TCP            95s
service/katib-suggestion-random                      ClusterIP
10.233.57.210   <none>        6789/TCP            95s
service/katib-ui                                     ClusterIP
10.233.6.116    <none>        80/TCP              96s
service/metadata-db                                  ClusterIP
10.233.31.2     <none>        3306/TCP            96s
service/metadata-service                             ClusterIP
10.233.27.104   <none>        8080/TCP            96s
service/metadata-ui                                  ClusterIP
10.233.57.177   <none>        80/TCP              96s
service/minio-service                                ClusterIP
10.233.44.90    <none>        9000/TCP            94s
service/ml-pipeline                                  ClusterIP
10.233.41.201   <none>        8888/TCP,8887/TCP   94s
service/ml-pipeline-tensorboard-ui                   ClusterIP
10.233.36.207   <none>        80/TCP              93s
service/ml-pipeline-ui                               ClusterIP
10.233.61.150   <none>        80/TCP              93s
service/mysql                                        ClusterIP
10.233.55.117   <none>        3306/TCP            94s
service/notebook-controller-service                  ClusterIP
10.233.10.166   <none>        443/TCP             95s
service/profiles-kfam                                ClusterIP
10.233.33.79    <none>        8081/TCP            92s
service/pytorch-operator                             ClusterIP
10.233.37.112   <none>        8443/TCP            95s
service/seldon-operator-controller-manager-service   ClusterIP
10.233.30.178   <none>        443/TCP             92s
service/tensorboard                                  ClusterIP
10.233.58.151   <none>        9000/TCP            94s
service/tf-job-dashboard                             ClusterIP
10.233.4.17     <none>        80/TCP              94s
service/tf-job-operator                              ClusterIP
```

```
10.233.60.32    <none>        8443/TCP            94s
service/webhook-server-service                      ClusterIP
10.233.32.167   <none>        443/TCP             87s
NAME                                                          READY   UP-
TO-DATE   AVAILABLE   AGE
deployment.apps/admission-webhook-deployment                  1/1     1
1           97s
deployment.apps/argo-ui                                       1/1     1
1           97s
deployment.apps/centraldashboard                              1/1     1
1           97s
deployment.apps/jupyter-web-app-deployment                    1/1     1
1           97s
deployment.apps/katib-controller                              1/1     1
1           96s
deployment.apps/katib-db                                      1/1     1
1           97s
deployment.apps/katib-manager                                 1/1     1
1           96s
deployment.apps/katib-manager-rest                            1/1     1
1           96s
deployment.apps/katib-suggestion-bayesianoptimization         1/1     1
1           95s
deployment.apps/katib-suggestion-grid                         1/1     1
1           95s
deployment.apps/katib-suggestion-hyperband                    1/1     1
1           95s
deployment.apps/katib-suggestion-nasrl                        1/1     1
1           95s
deployment.apps/katib-suggestion-random                       1/1     1
1           95s
deployment.apps/katib-ui                                      1/1     1
1           96s
deployment.apps/metadata-db                                   1/1     1
1           96s
deployment.apps/metadata-deployment                           3/3     3
3           96s
deployment.apps/metadata-ui                                   1/1     1
1           96s
deployment.apps/minio                                         1/1     1
1           94s
deployment.apps/ml-pipeline                                   1/1     1
1           94s
deployment.apps/ml-pipeline-persistenceagent                  1/1     1
1           93s
deployment.apps/ml-pipeline-scheduledworkflow                 1/1     1
```

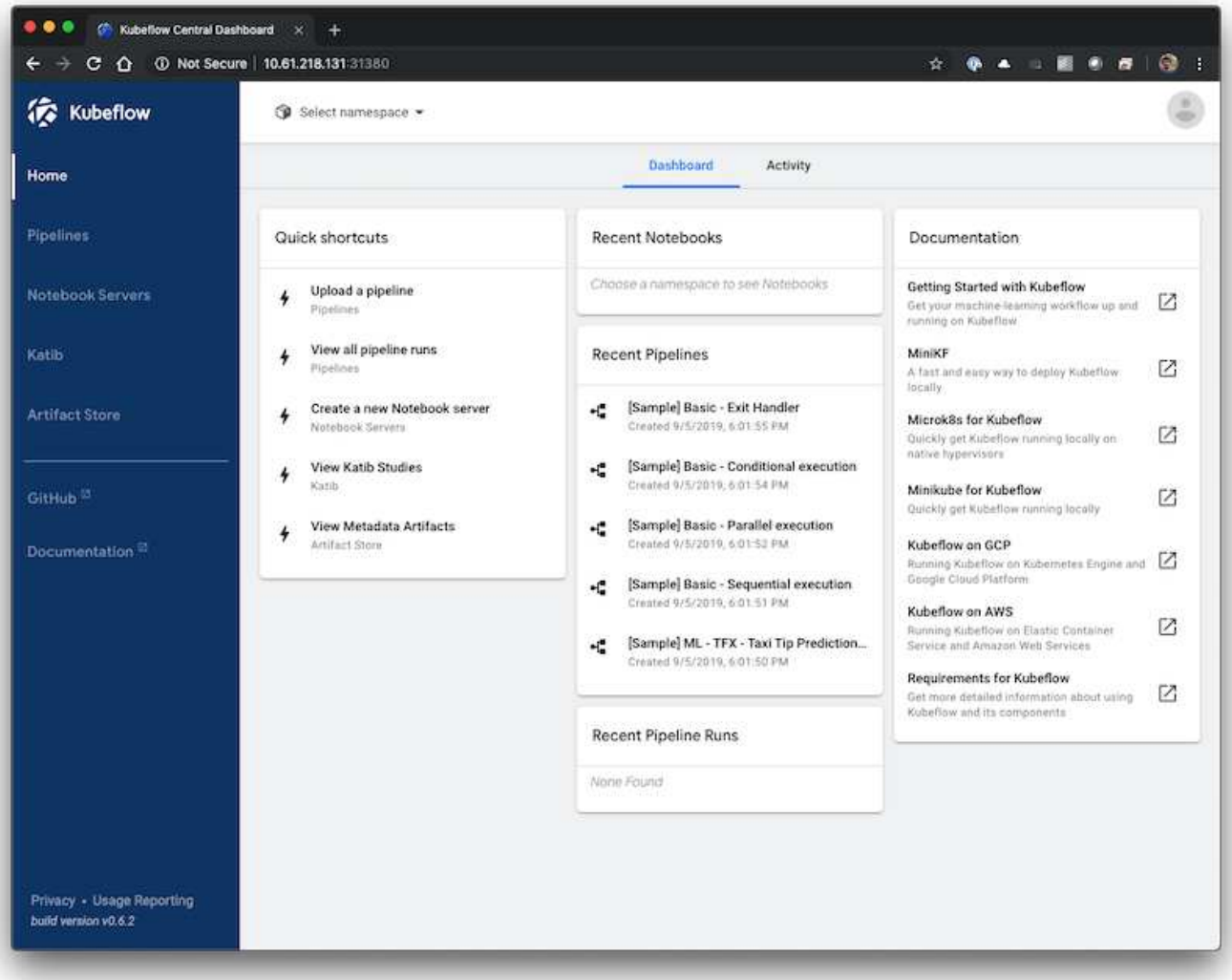
```
1           93s
deployment.apps/ml-pipeline-ui                                1/1     1
1           93s
deployment.apps/ml-pipeline-viewer-controller-deployment   1/1     1
1           93s
deployment.apps/mysql                                         1/1     1
1           94s
deployment.apps/notebook-controller-deployment                1/1     1
1           95s
deployment.apps/profiles-deployment                           1/1     1
1           92s
deployment.apps/pytorch-operator                              1/1     1
1           95s
deployment.apps/spartakus-volunteer                           1/1     1
1           94s
deployment.apps/tensorboard                                   1/1     1
1           94s
deployment.apps/tf-job-dashboard                              1/1     1
1           94s
deployment.apps/tf-job-operator                               1/1     1
1           94s
deployment.apps/workflow-controller                           1/1     1
1           97s
NAME
DESIRED   CURRENT   READY   AGE
replicaset.apps/admission-webhook-deployment-6b89c84c98              1
1         1       97s
replicaset.apps/argo-ui-5dcf5d8b4f                                   1
1         1       97s
replicaset.apps/centraldashboard-cf4874ddc                           1
1         1       97s
replicaset.apps/jupyter-web-app-deployment-685b455447                1
1         1       97s
replicaset.apps/katib-controller-88c97d85c                           1
1         1       96s
replicaset.apps/katib-db-8598468fd8                                  1
1         1       97s
replicaset.apps/katib-manager-574c8c67f9                             1
1         1       96s
replicaset.apps/katib-manager-rest-778857c989                        1
1         1       96s
replicaset.apps/katib-suggestion-bayesianoptimization-65df4d7455     1
1         1       95s
replicaset.apps/katib-suggestion-grid-56bf69f597                     1
1         1       95s
replicaset.apps/katib-suggestion-hyperband-7777b76cb9                1
```

```
1         1       95s
replicaset.apps/katib-suggestion-nasrl-77f6f9458c                    1
1         1       95s
replicaset.apps/katib-suggestion-random-77b88b5c79                   1
1         1       95s
replicaset.apps/katib-ui-7587c5b967                                  1
1         1       96s
replicaset.apps/metadata-db-5dd459cc                                 1
1         1       96s
replicaset.apps/metadata-deployment-6cf77db994                       3
3         3       96s
replicaset.apps/metadata-ui-78f5b59b56                               1
1         1       96s
replicaset.apps/minio-758b769d67                                     1
1         1       93s
replicaset.apps/ml-pipeline-5875b9db95                               1
1         1       93s
replicaset.apps/ml-pipeline-persistenceagent-9b69ddd46               1
1         1       92s
replicaset.apps/ml-pipeline-scheduledworkflow-7b8d756c76             1
1         1       91s
replicaset.apps/ml-pipeline-ui-79ffd9c76                             1
1         1       91s
replicaset.apps/ml-pipeline-viewer-controller-deployment-5fdc87f58   1
1         1       91s
replicaset.apps/mysql-657f87857d                                     1
1         1       92s
replicaset.apps/notebook-controller-deployment-56b4f59bbf            1
1         1       94s
replicaset.apps/profiles-deployment-6bc745947                        1
1         1       91s
replicaset.apps/pytorch-operator-77c97f4879                          1
1         1       94s
replicaset.apps/spartakus-volunteer-5fdfddb779                       1
1         1       94s
replicaset.apps/tensorboard-6544748d94                               1
1         1       93s
replicaset.apps/tf-job-dashboard-56f79c59dd                          1
1         1       93s
replicaset.apps/tf-job-operator-79cbfd6dbc                           1
1         1       93s
replicaset.apps/workflow-controller-db644d554                        1
1         1       97s
NAME                                                        READY   AGE
statefulset.apps/admission-webhook-bootstrap-stateful-set   1/1     97s
statefulset.apps/application-controller-stateful-set        1/1     98s
```

```
statefulset.apps/metacontroller                                  1/1     98s
statefulset.apps/seldon-operator-controller-manager              1/1     92s
$ kubectl get pvc -n kubeflow
NAME                STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS                AGE
katib-mysql         Bound    pvc-b07f293e-d028-11e9-9b9d-00505681a82d
10Gi       RWO              ontap-ai-flexvols-retain   27m
metadata-mysql      Bound    pvc-b0f3f032-d028-11e9-9b9d-00505681a82d
10Gi       RWO              ontap-ai-flexvols-retain   27m
minio-pv-claim      Bound    pvc-b22727ee-d028-11e9-9b9d-00505681a82d
20Gi       RWO              ontap-ai-flexvols-retain   27m
mysql-pv-claim      Bound    pvc-b2429afd-d028-11e9-9b9d-00505681a82d
20Gi       RWO              ontap-ai-flexvols-retain   27m
```

4. 在網頁瀏覽器中、瀏覽至您在步驟2中記下的URL、即可存取Kubeflow中央儀表板。

   預設使用者名稱為「admin@kubeflow.org」、預設密碼為「12341234」。若要建立其他使用者、請遵循中的指示 "官方Kubeflow文件"。



Kubeflow作業與工作範例

本節包含使用Kubeflow執行的各種作業和工作範例。

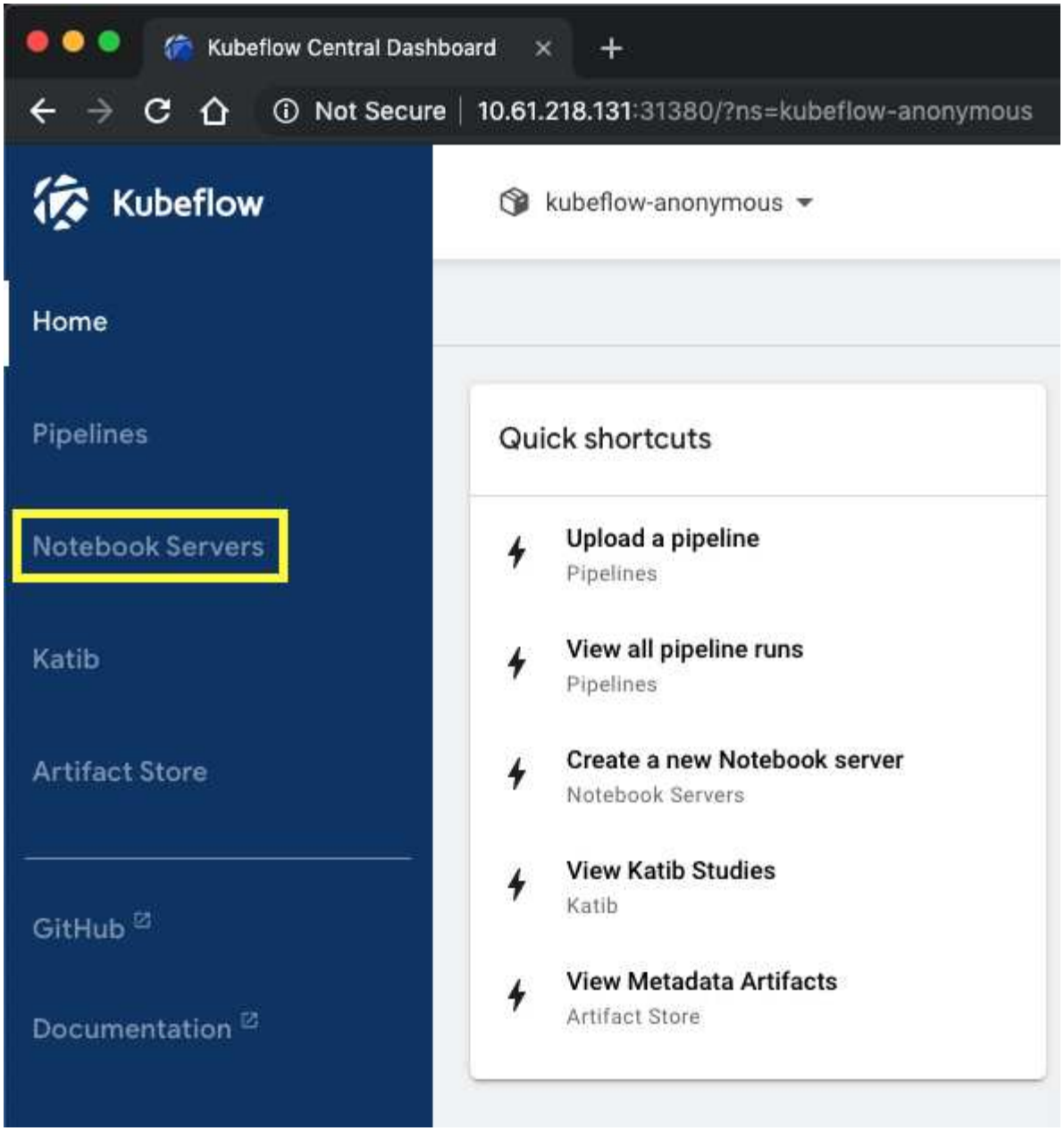
Kubeflow作業與工作範例

本節包含使用Kubeflow執行的各種作業和工作範例。

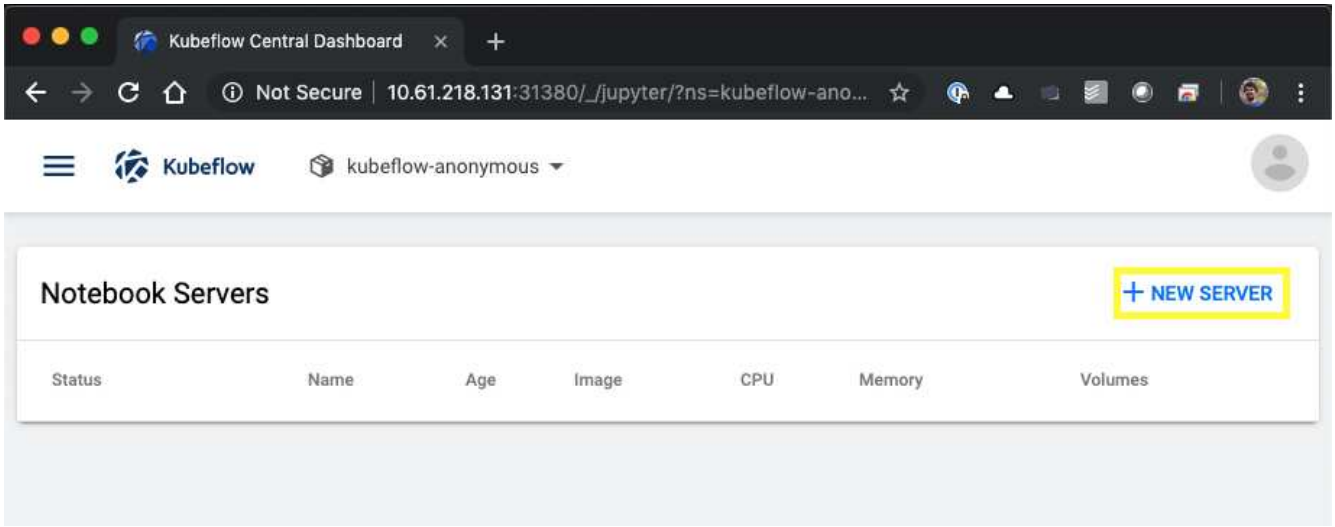
為資料科學家或開發人員提供Jupyter筆記型電腦工作區

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。若要使用Kubeflow來配置新的Jupyter筆記型電腦伺服器、請執行下列工作。如需Kubeflow內容中Jupyter Notebooks的詳細資訊、請參閱 "官方Kubeflow文件"。

1. 在Kubeflow中央儀表板中、按一下主功能表中的「Notebook Servers（筆記型電腦伺服器）」、瀏覽至Jupyter Notebook server管理頁面。

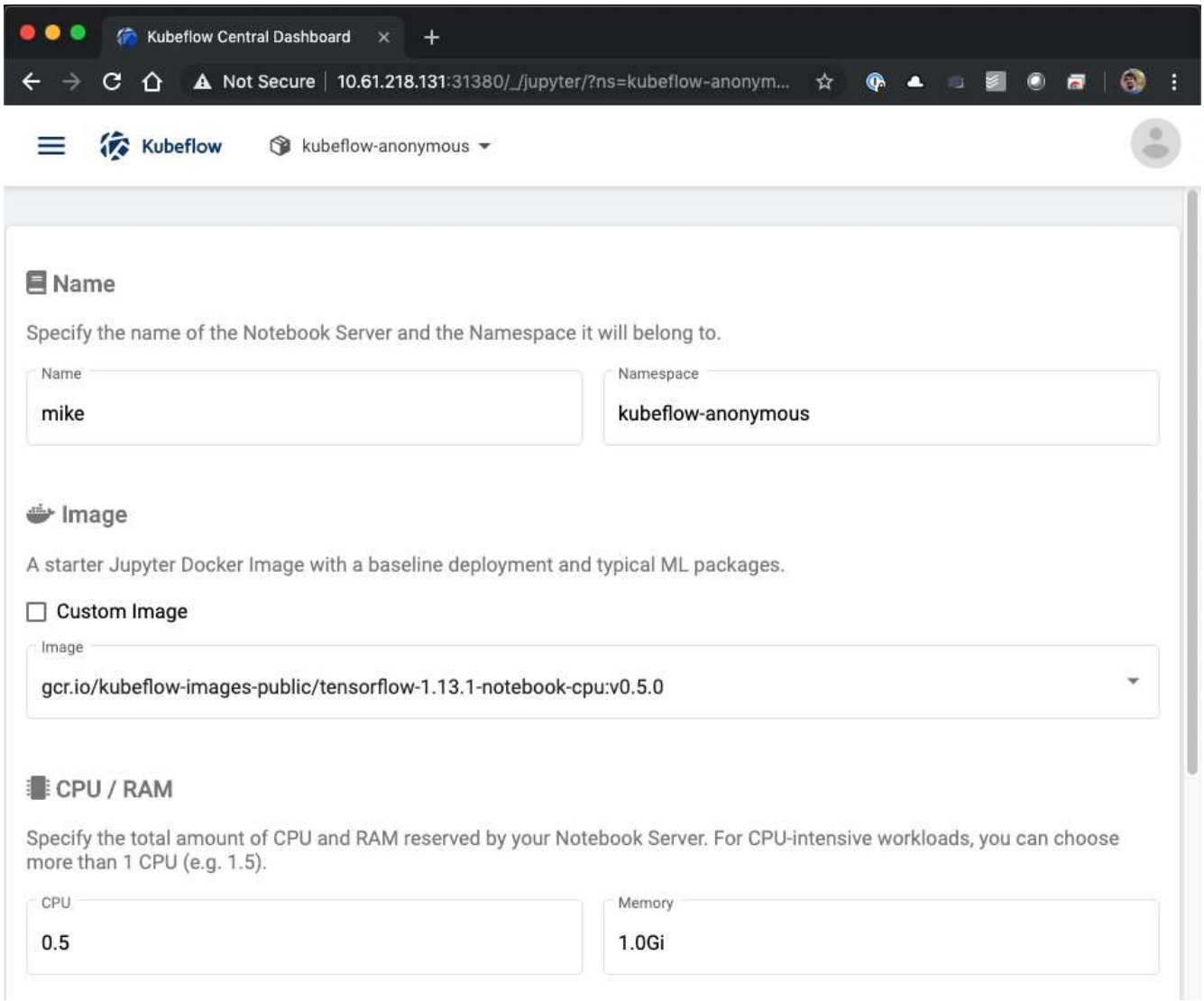


2. 按一下「New Server（新伺服器）」以配置新的Jupyter Notebook伺服器。

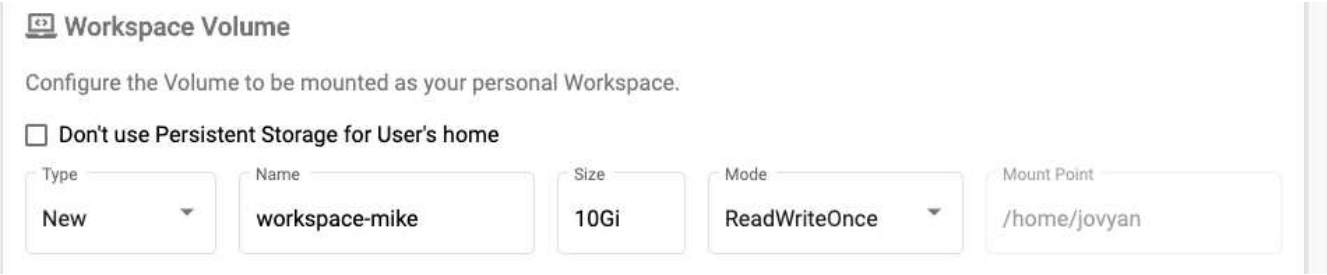


3. 命名新伺服器、選擇您要伺服器所依據的Docker映像、然後指定伺服器要保留的CPU和RAM數量。如果「命名空間」欄位為空白、請使用頁面標題中的「選取命名空間」功能表來選擇命名空間。然後、「Namespace（命名空間）」欄位會自動填入所選的命名空間。

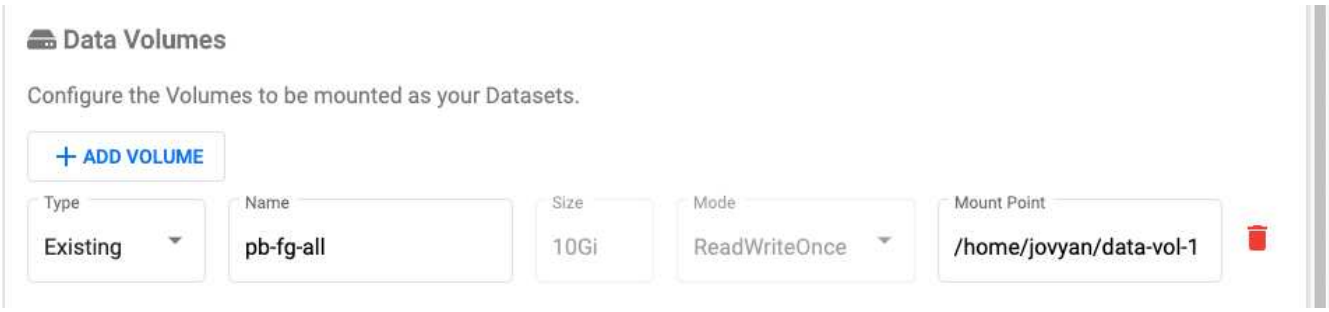
   在以下範例中、會選擇「kubeflow匿名」命名空間。此外、Docker映像、CPU和RAM的預設值也會被接受。



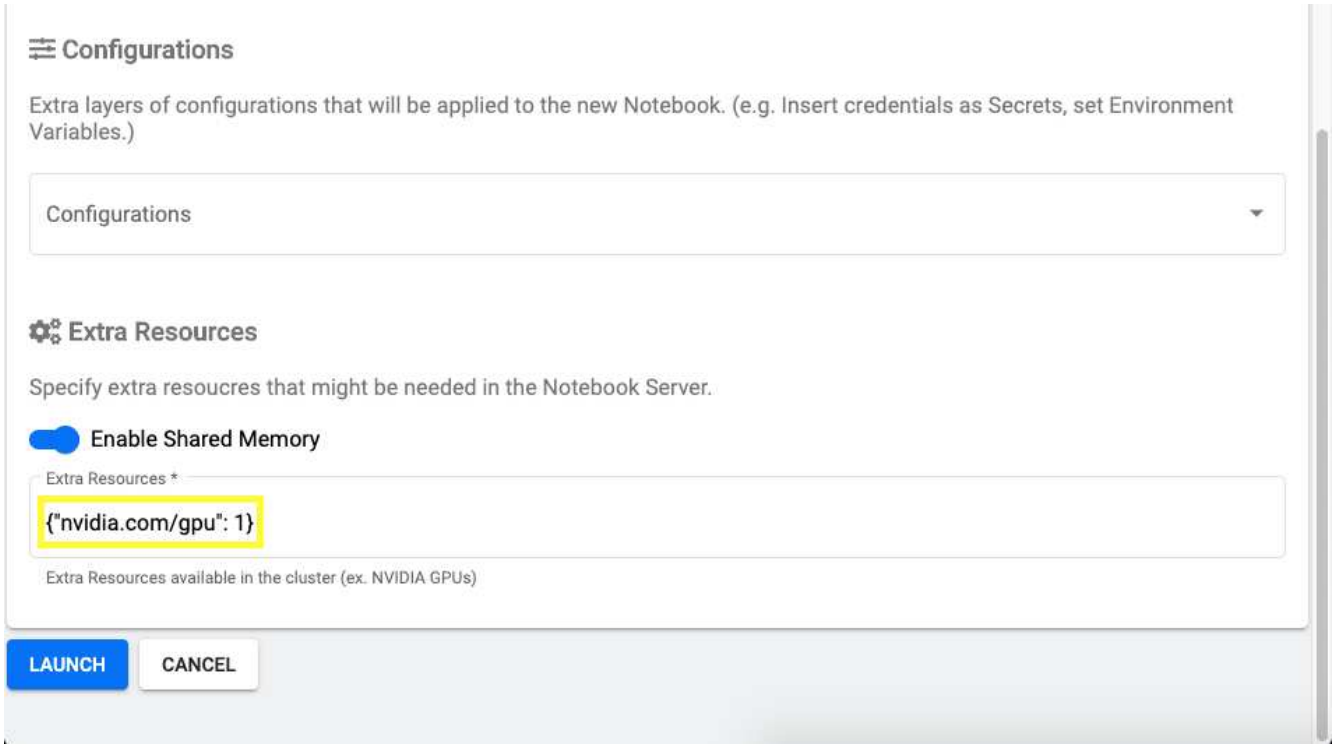
4. 指定工作區Volume詳細資料。如果您選擇建立新磁碟區、則該磁碟區或永久虛擬磁碟區會使用預設StorageClass進行資源配置。因為使用Trident的StorageClass已在區段中指定為預設StorageClass "Kubeflow部署"、磁碟區或永久虛擬磁碟區是以Trident來配置。此Volume會自動掛載為Jupyter Notebook Server Container中的預設工作區。使用者在伺服器上建立的任何筆記型電腦、若未儲存至個別的資料Volume、都會自動儲存至此工作區磁碟區。因此、筆記型電腦會在重新開機後持續運作。



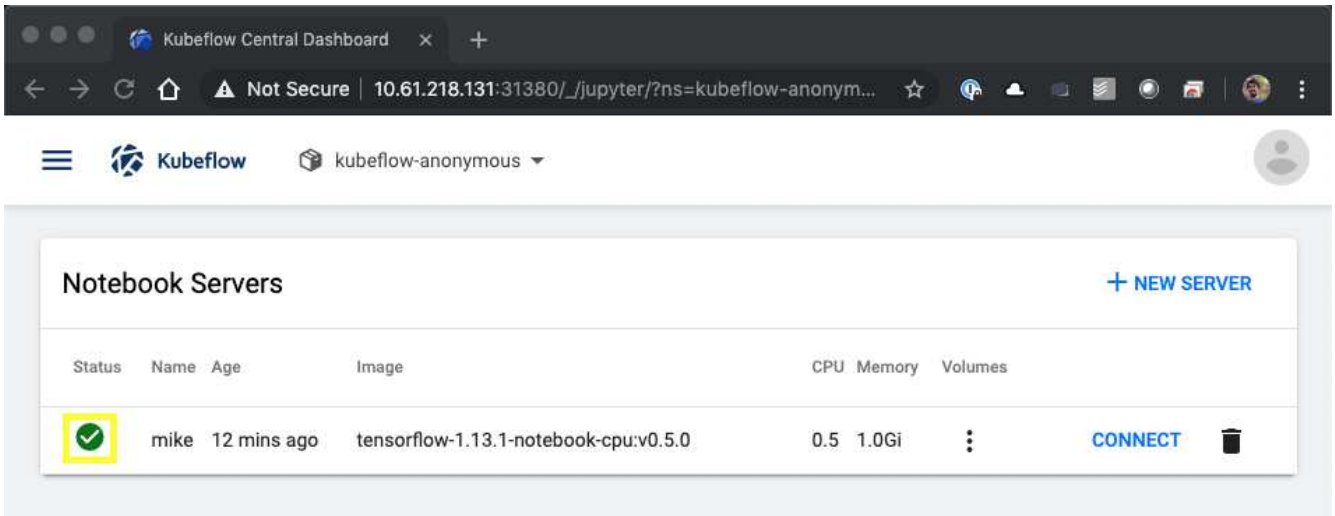
5. 新增資料磁碟區。下列範例指定名為「PPB -FG All」的現有PVc、並接受預設掛載點。



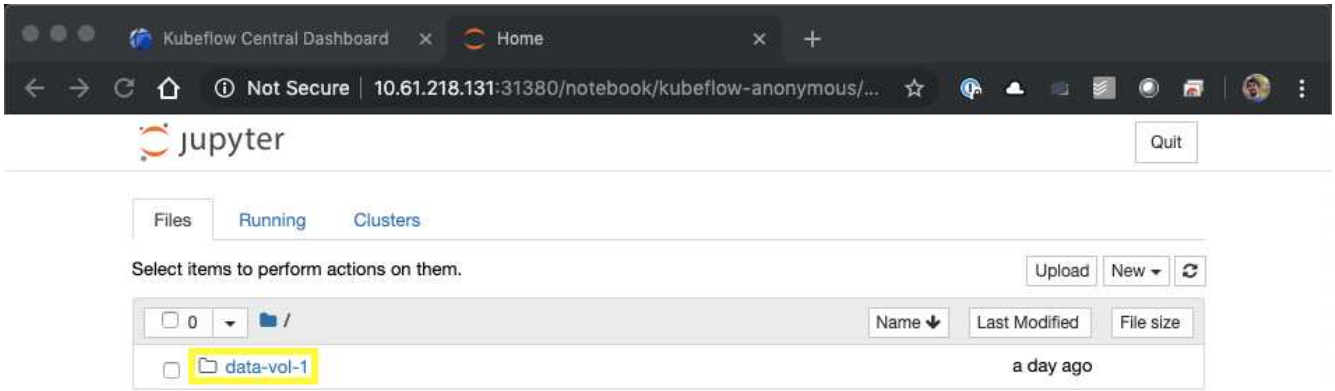
6. *選用：*要求將所需的GPU數量分配給您的筆記型電腦伺服器。在下列範例中、需要一個GPU。

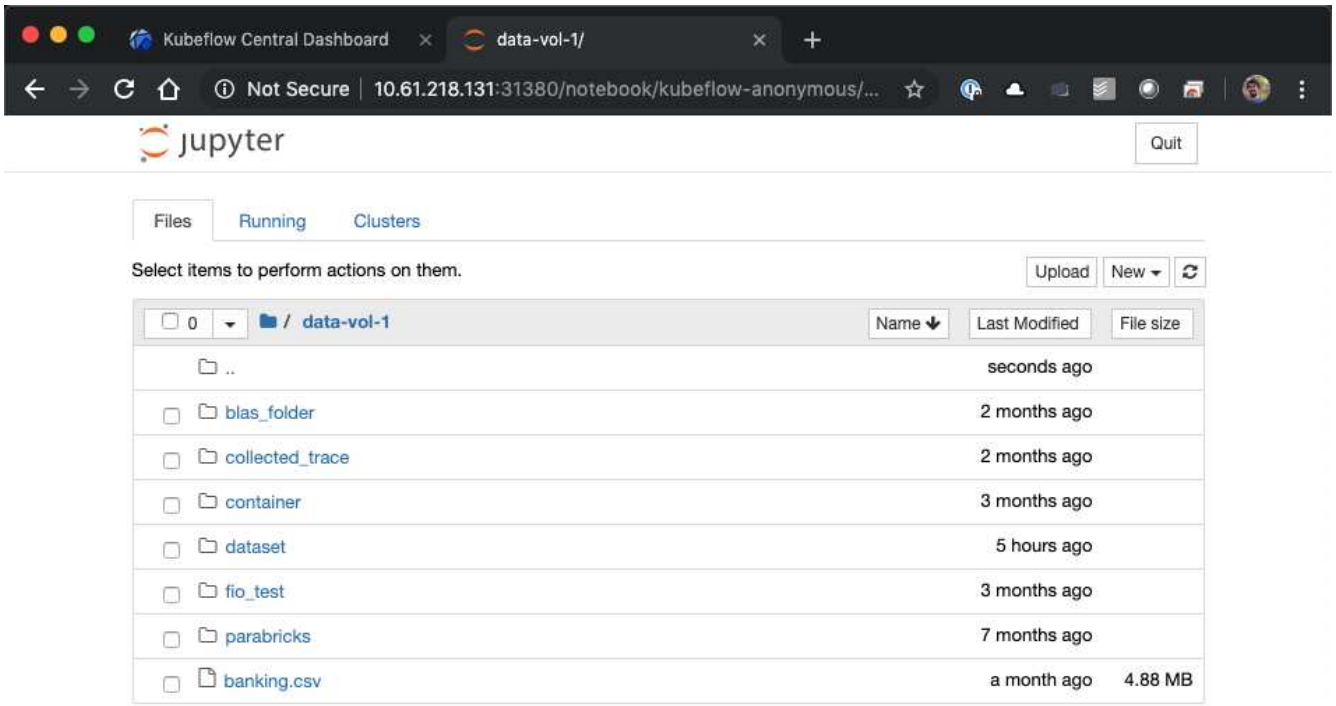


7. 按一下「啟動」以配置新的筆記型電腦伺服器。
8. 等待筆記型電腦伺服器完全配置完成。如果您從未使用您指定的Docker映像來配置伺服器、則可能需要幾分鐘的時間、因為該映像需要下載。當伺服器已完全配置完成時、您會在Jupyter筆記型電腦管理頁面的「Status（狀態）」欄中看到綠色勾號。



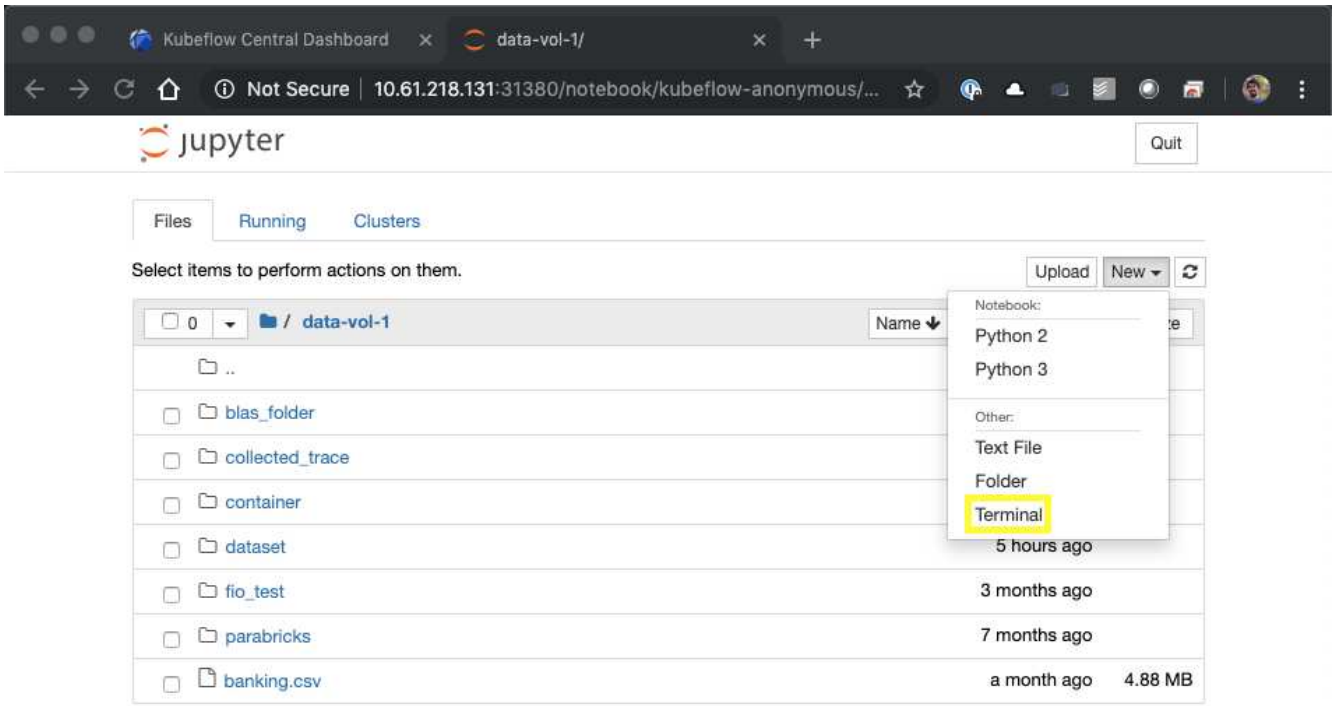
9. 按一下「連線」以連線至新的伺服器Web介面。
10. 確認已在伺服器上掛載步驟6中指定的資料集磁碟區。請注意、此磁碟區預設會掛載於預設工作區內。從使用者的觀點來看、這只是工作區內的另一個資料夾。使用者可能是資料科學家、而非基礎架構專家、因此不需要擁有任何儲存專業知識、就能使用此磁碟區。

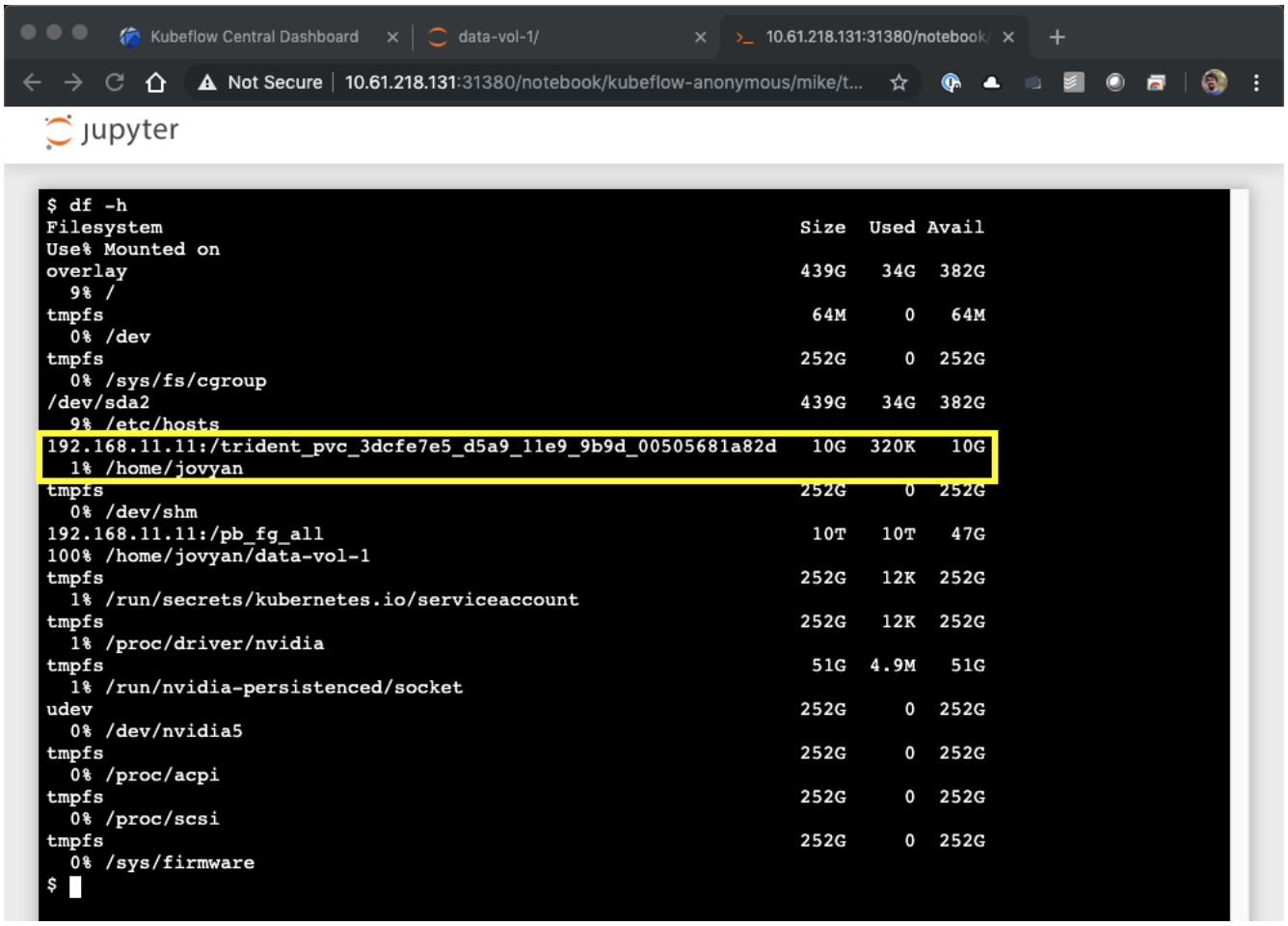




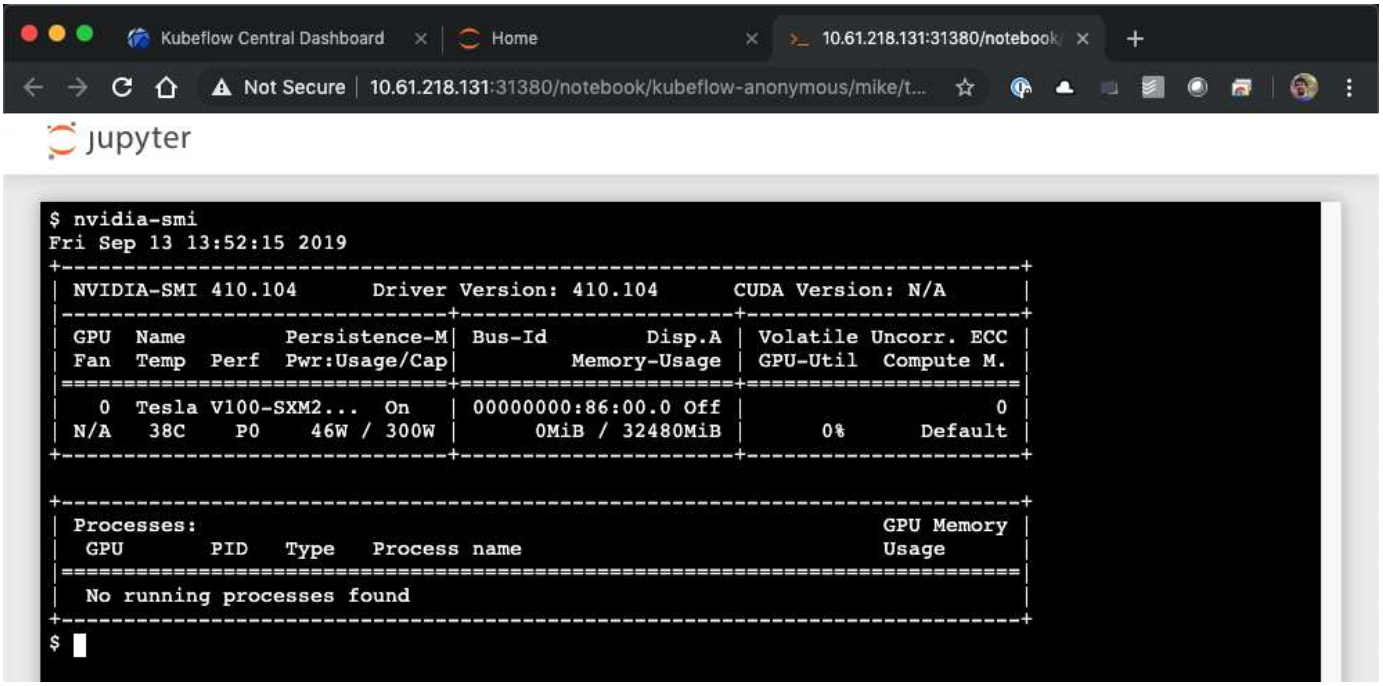
11. 開啟終端機、並假設步驟5要求新磁碟區、執行「df -h」、確認新的Trident資源配置持續磁碟區已掛載為預設工作區。

    預設工作區目錄是您第一次存取伺服器Web介面時所顯示的基礎目錄。因此、您使用Web介面建立的任何成品都會儲存在此Trident資源配置的持續Volume中。





12. 使用終端機執行「nvidia-smi」、確認已將正確數量的GPU分配給筆記型電腦伺服器。在下列範例中、已依照步驟7的要求、將一個GPU分配給筆記型電腦伺服器。



筆記型電腦和管線範例

。 "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：

- 資料科學家可直接從Jupyter筆記型電腦執行進階NetApp資料管理作業。
- 進階NetApp資料管理作業可透過Kubeflow Pipes架構整合至自動化工作流程中。

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。

Apache Airflow部署

NetApp建議在Kubernetes上執行Apache Airflow。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. 您已經在Kubernetes叢集中安裝並設定NetApp Trident、如「NetApp Trident部署與組態」一節所述。

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。若要在叢集內指定預設StorageClass、請遵循一節中所述的指示 "Kubeflow部署"。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
```

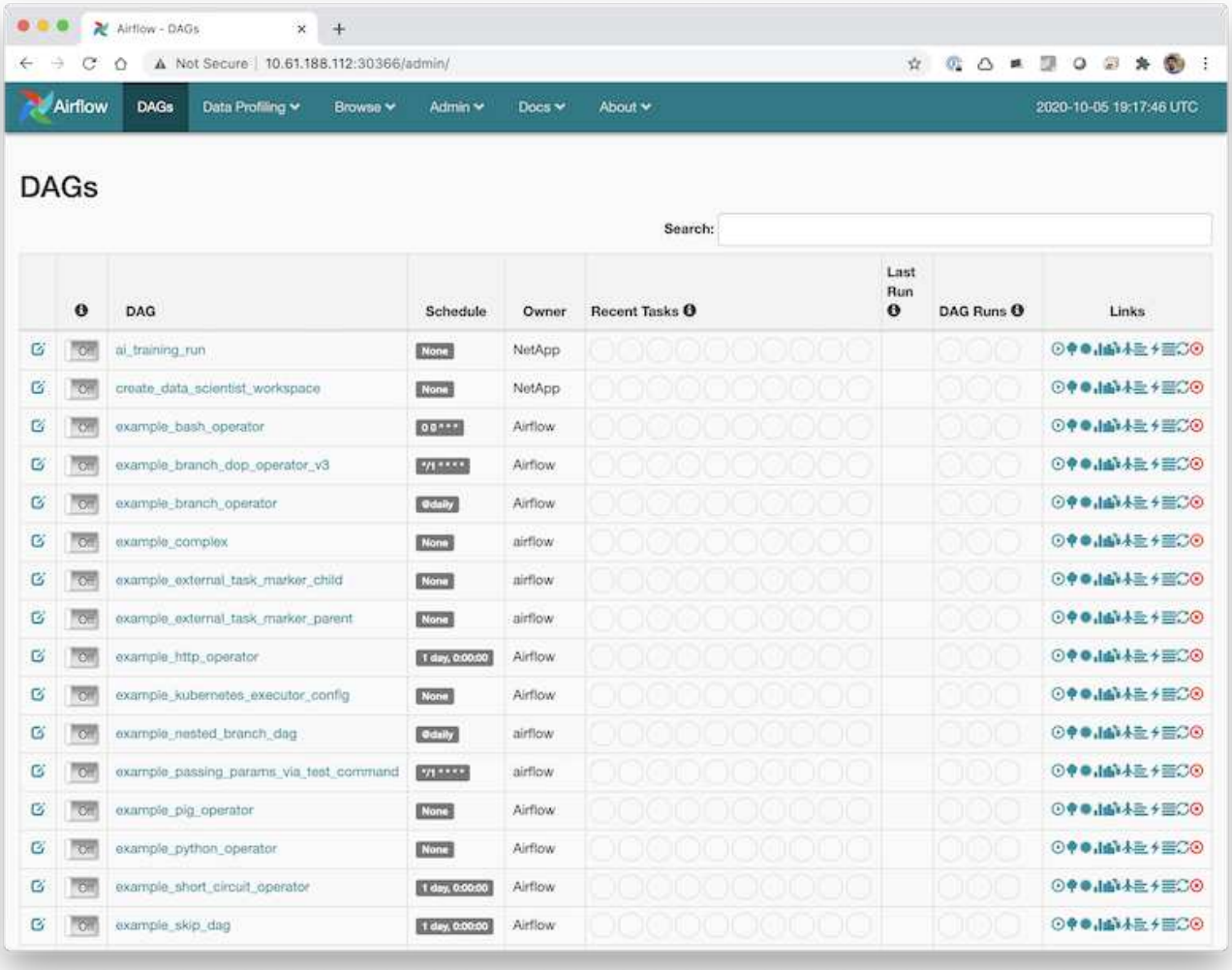
2. 確認所有的氣流網墊都已啟動且正常運作。所有Pod可能需要幾分鐘的時間才能啟動。

```
$ 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。



Apache氣流工作流程範例

。 "適用於Kubernetes的NetApp Data科學工具套件" 可搭配氣流一起使用。運用NetApp Data科學工具套件搭配氣流、您可以將NetApp資料管理作業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 Data科學工具套件GitHub儲存庫中的一節、詳細瞭解如何搭配氣流使用此工具組。

Trident作業範例

本節包含您可能想要使用Trident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。

以下的命令範例顯示兩次匯入同一個磁碟區、名稱為「PB_FP_ALL」、一次匯入在本節範例中建立的每個Trident後端 "Trident後端範例、適用於ONTAP AI部署"步驟1.以這種方式匯入相同的磁碟區兩次、可讓您在FlexGroup 不同的生命週期內多次掛載該磁碟區（現有的一套）、如一節所述 "Trident後端範例、適用於ONTAP AI部署"步驟1.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"。

> 下列匯入命令範例中所指定的後端名稱、會對應至本節範例中所建立的後端 "Trident後端範例、適用於ONTAP AI部署"步驟1.下列範例PVc定義檔中所指定的StorageClass名稱、會對應至本節範例中所建立的StorageClass "Kubernetes StorageClass範例ONTAP 、可用於進行AI部署"步驟1.

```
$ 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
|
+--------------------------------+--------
+-----------------------------------+----------
+-----------------------------------------------+--------+---------+
$ cat << EOF > ./pvc-import-pb_fg_all-iface2.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pb-fg-all-iface2
  namespace: default
```

```
spec:
  accessModes:
    - ReadOnlyMany
  storageClassName: ontap-ai-flexgroups-retain-iface2
EOF
$ tridentctl import volume ontap-ai-flexgroups-iface2 pb_fg_all -f ./pvc-
import-pb_fg_all-iface2.yaml -n trident
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
|          NAME                  |  SIZE  |       STORAGE CLASS
| PROTOCOL |             BACKEND UUID                        | STATE  |
MANAGED |
+--------------------------------+--------
+-----------------------------------+----------
+-------------------------------------------+--------+---------+
| default-pb-fg-all-iface2-85aee | 10 TiB | ontap-ai-flexgroups-retain-
iface2 | file     | 61814d48-c770-436b-9cb4-cf7ee661274d | 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
|
| default-pb-fg-all-iface2-85aee   | 10 TiB  | ontap-ai-flexgroups-retain-
iface2 | file     | 61814d48-c770-436b-9cb4-cf7ee661274d | 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
pb-fg-all-iface2     Bound    default-pb-fg-all-iface2-85aee
```

```
10995116277760   ROX            ontap-ai-flexgroups-retain-iface2   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。下列命令範例顯示新FlexVol 的供應功能。在此範例中、磁碟區是使用本節範例中所建立的StorageClass進行資源配置 "Kubernetes StorageClass範例ONTAP 、可用於進行AI部署"步驟2：

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
pb-fg-all-iface2                  Bound     default-pb-fg-all-iface2-85aee
10995116277760   ROX            ontap-ai-flexgroups-retain-iface2   26h
tensorflow-results                Bound     default-tensorflow-results-
2fd60   1073741824       RWX            ontap-ai-flexvols-retain
25h
```

適用於AI部署的高效能工作範例ONTAP

本節包括在ONTAP 將Kubernetes部署在AI Pod上時、可執行的各種高效能工作範例。

適用於AI部署的高效能工作範例ONTAP

本節包括在ONTAP 將Kubernetes部署在AI Pod上時、可執行的各種高效能工作範例。

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。此步驟為Kubernetes原生作業、不需要NetApp專業人員。

> 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。

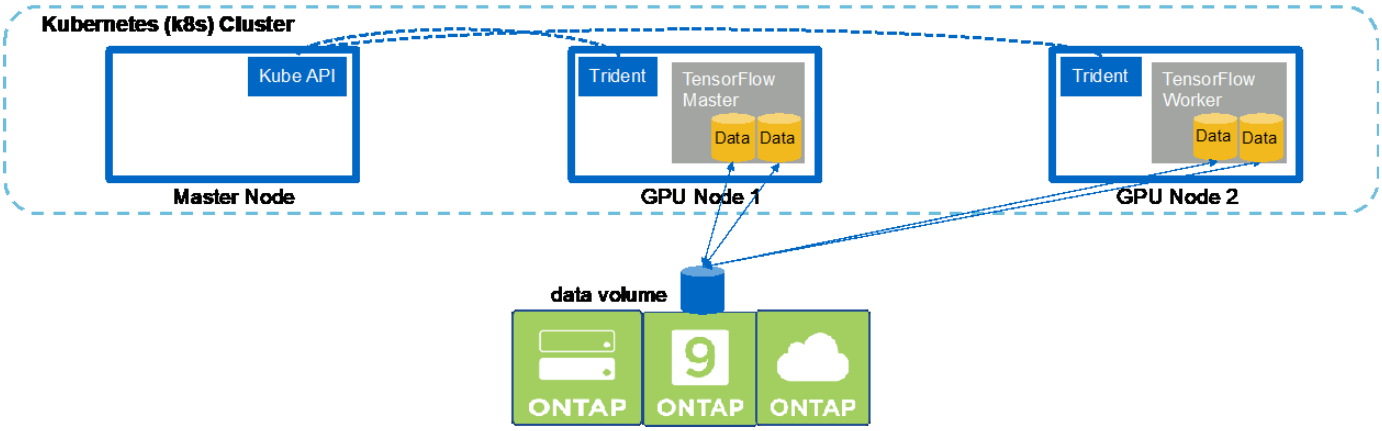
   刪除工作物件時、Kubernetes會自動刪除任何相關的Pod。

```
$ 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.
```

效能測試

我們在建立此解決方案時、進行了簡單的效能比較。我們使用Kubernetes執行數項標準NetApp AI基準測試工作、並將基準測試結果與使用簡易Docker RUN命令執行的執行結果進行比較。我們並未發現效能有任何顯著差異。因此、我們的結論是、使用Kubernetes來協調容器化AI訓練工作、並不會對效能造成負面影響。請參閱下表以取得效能比較結果。

基準測試	資料集	**Docker Run**（影像/秒）	**Kubernetes**（影像/秒）
單節點TensorFlow	綜合資料	6、667.2475	6、661.93125
單節點TensorFlow	ImageNet	6,570.2025	6、530.59125
同步分散式雙節點TensorFlow	綜合資料	13、213.70625	13、218.288125
同步分散式雙節點TensorFlow	ImageNet	12、941.69125	12、881.33875

結論

各種規模的公司和組織、以及所有產業、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。這些挑戰可透過使用NetApp AI Control Plane解決方案來解決。

此解決方案可讓您快速複製資料命名空間。此外、它還可讓您定義及實作AI、ML及DL訓練工作流程、並整合近乎即時的資料建立與模型基準、以利追蹤及版本管理。有了這套解決方案、您可以追蹤每個模型訓練、並將其重新傳回模型訓練及/或驗證的確切資料集。最後、此解決方案可讓您迅速配置Jupyter Notebook工作區、並存取大量的資料集。

由於本解決方案的目標對象是資料科學家和資料工程師、ONTAP 因此需要最少的NetApp或NetApp的支援專業知識。有了這套解決方案、您就能使用簡單且熟悉的工具和介面來執行資料管理功能。此外、此解決方案還充分運用開放原始碼和免費元件。因此、如果您的環境中已經有NetApp儲存設備、現在就可以實作此解決方案。如果您想試用此解決方案、但尚未擁有NetApp儲存設備、請造訪 "cloud.netapp.com"而且您可以立即使用雲端型NetApp儲存解決方案來啟動和執行。