



# AI を実行することでクラスタと GPU の利用率を最適化

## NetApp Solutions

NetApp  
September 10, 2024

This PDF was generated from [https://docs.netapp.com/ja-jp/netapp-solutions/ai/osrunai\\_run\\_ai\\_installation.html](https://docs.netapp.com/ja-jp/netapp-solutions/ai/osrunai_run_ai_installation.html) on September 10, 2024. Always check docs.netapp.com for the latest.

# 目次

クラスタと GPU の利用率を最適化して実行： AI .....	1
「 AI Installation 」 を実行します .....	1
「 AI Dashboards and Views 」 を実行します .....	1
データサイエンスチームのプロジェクトを作成し、 GPU を割り当てる .....	2
実行中のジョブの送信： AI CLI .....	3
高いクラスタ利用率の達成 .....	6
パフォーマンスの低いワークロードやインタラクティブなワークロードに適した、フラクショナルな GPU 割り当て .....	7
オーバークォータの GPU 割り当てによる高いクラスタ利用率の達成 .....	8
基本的な資源配分フェアネス .....	10
オーバークォータフェアネス .....	11
Trident でプロビジョニングされた永続的ボリュームにデータを保存する .....	13

# クラスタと GPU の利用率を最適化して実行： AI

以下のセクションでは、この検証で実行した AI のインストール、テストシナリオ、結果について詳しく説明します。

TensorFlow ベンチマークを含む業界標準のベンチマークツールを使用して、このシステムの動作とパフォーマンスを検証しました。ImageNet データセットを使用して ResNet-50 をトレーニングしました。これは、画像分類のための有名な Convolutional Neural Network（CNN；畳み込みニューラルネットワーク）DL モデルです。ResNet-50 は、処理時間を短縮して正確なトレーニング結果を提供します。これにより、ストレージに対する十分な需要を喚起することができました。

## 「AI Installation」を実行します

Run：AI をインストールするには、次の手順を実行します。

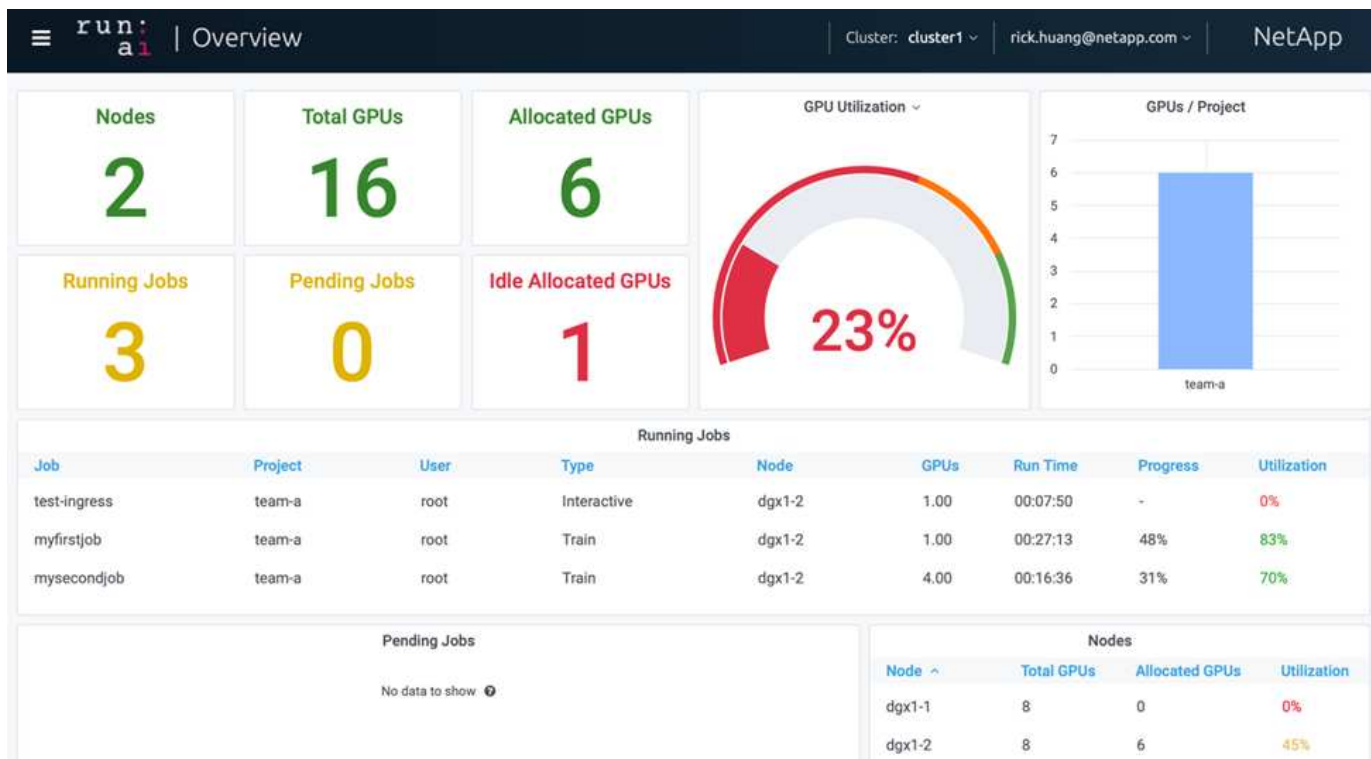
1. DeepOps を使用して Kubernetes クラスタをインストールし、ネットアップのデフォルトストレージクラスを設定します。
2. GPU ノードの準備：
  - a. NVIDIA ドライバが GPU ノードにインストールされていることを確認します。
  - b. 「nvidia - Docker」がインストールされ、デフォルトの Docker ランタイムとして設定されていることを確認します。
3. インストール実行：AI：
  - a. にログインします ["AI 管理 UI を実行"](#) クラスタを作成できます。
  - b. 作成した「runai-operator-<clustername>.yaml」ファイルをダウンロードします。
  - c. オペレータ設定を Kubernetes クラスタに適用します。

```
kubectl apply -f runai-operator-<clustername>.yaml
```

4. インストールを確認します。
  - a. に進みます ["https://app.run.ai/"](https://app.run.ai/)。
  - b. 概要ダッシュボードに移動します。
  - c. 右上の GPU の数が、想定される GPU の数と GPU ノードの数をすべてサーバのリストに表示していることを確認します。実行：AI 導入の詳細については、[を参照してください "Run：AI をオンプレミスの Kubernetes クラスタにインストール"](#) および ["Run：AI CLI のインストール"](#)。

## 「AI Dashboards and Views」を実行します

Run：AI を Kubernetes クラスタにインストールし、コンテナを正しく設定したら、次のダッシュボードとビューが表示されます ["https://app.run.ai"](https://app.run.ai) 次の図に示すように、ブラウザで設定します。



2つのDGX-1ノードによってクラスタ内に提供されるGPUは合計16個です。ノード数、使用可能なGPUの総数、ワークロードに割り当てられているGPUの数、実行中のジョブの総数、保留中のジョブ、およびアイドル状態に割り当てられているGPUの数を確認できます。右側のバー図は、プロジェクトごとのGPUを示しています。これは、各チームがどのようにクラスタリソースを使用しているかをまとめたものです。中央には、ジョブ名、プロジェクト、ユーザー、ジョブタイプなど、現在実行中のジョブとジョブの詳細のリストが表示されます。各ジョブが実行されているノード、そのジョブに割り当てられているGPUの数、ジョブの現在の実行時間、ジョブの進捗状況、およびそのジョブのGPU利用率。1つのチーム（「TEAM」）から送信された実行中のジョブは3つしかないため、クラスタの使用率が低いことに注意してください（GPU使用率は23%）。

次のセクションでは、「プロジェクト」タブで複数のチームを作成し、各チームにGPUを割り当てることで、クラスタあたりのユーザ数が多いときにクラスタの使用率を最大限に高め、リソースを管理する方法を説明します。テストシナリオは、トレーニング、推論、対話型のワークロードでメモリリソースとGPUリソースが共有されているエンタープライズ環境をシミュレートしたものです。

## データサイエンスチームのプロジェクトを作成し、GPUを割り当てる

研究者は、AI CLIやKubeflowなどのプロセスを使ってワークロードを送信できます。リソースの割り当てを合理化して優先順位を設定するために、Run: AIでは「プロジェクト」という概念が導入されています。プロジェクトは、プロジェクト名をGPU割り当てと優先設定に関連付けるクォータエンティティです。複数のデータサイエンスチームを管理するシンプルで便利な方法です。

ワークロードを送信する研究者は、プロジェクトをワークロード要求に関連付ける必要があります。Run: AIスケジューラは、リクエストを現在の割り当てとプロジェクトと比較して、ワークロードにリソースを割り当てることができるかどうか、またはワークロードを保留中の状態にするかどうかを判断します。

システム管理者は、実行: AIプロジェクトタブで次のパラメータを設定できます。

- \* モデルプロジェクト。\* ユーザーごとにプロジェクトを設定し、ユーザーのチームごとにプロジェクトを設定し、実際の組織プロジェクトごとにプロジェクトを設定します。
- \* プロジェクトの割り当て量。\* 各プロジェクトは、このプロジェクトに同時に割り当てることができる GPU の割り当て量に関連付けられています。これは、このプロジェクトを使用している研究者が、クラスタ内のどの状態であっても、この数の GPU を取得できることが保証されるという意味で、保証されたクォータです。原則として、プロジェクトの割り当ての合計は、クラスタ内の GPU の数と等しくなければなりません。さらに、このプロジェクトのユーザは、過剰割り当てを受け取ることができます。GPU が使用されていない限り、このプロジェクトを使用する研究者は GPU を増やすことができます。に、クォータ超過テストのシナリオと公平性に関する考慮事項を示します "[オーバークォータの GPU 割り当てによる高いクラスタ利用率の達成](#)"、"[基本的な資源配分フェアネス](#)"および "[オーバークォータフェアネス](#)"。
- 新しいプロジェクトを作成し、既存のプロジェクトを更新して、既存のプロジェクトを削除します。
- \* 特定のノードグループで実行するジョブを制限 \*。特定のプロジェクトを割り当てて、特定のノードのみ実行することができます。これは、プロジェクトチームが十分なメモリを備えた特殊なハードウェアを必要とする場合などに便利です。また、プロジェクトチームは、特別な予算で取得された特定のハードウェアの所有者になる場合もあります。また、パフォーマンスが低いハードウェアで作業したり、長いトレーニングや無人ワークロードを高速ノードに誘導したりするために、ビルドや対話型のワークロードを処理する必要がある場合もあります。ノードをグループ化し、特定のプロジェクトにアフィニティを設定するコマンドについては、を参照してください "[「AI Documentation」を実行します](#)"。
- \* 対話型ジョブの期間を制限します \*。研究者たちは、対話型の仕事を閉じることを忘れがちだ。これにより、リソースの無駄が発生する可能性があります。一部の組織では、対話型ジョブの期間を制限し、自動的に終了することを希望しています。

次の図は、4つのチームが作成されたプロジェクトビューを示しています。各チームには、異なるワークロードに対応するために異なる数の GPU が割り当てられます。GPU の総数は、2 台の DGX-1 で構成されるクラスタ内の使用可能な GPU の総数と同じです。

run:ai

Projects

Cluster: cluster1

rick.huang@netapp.com

NetApp

Filter and Search

+ Add New project

Project Name	Assigned GPUs	Created	Training Node Affinity	Interactive Node Affinity
<div><div></div><div>team-a</div></div>	2	07/27/20, 9:28AM	none	none
<div><div></div><div>team-b</div></div>	4	07/28/20, 7:50AM	none	none
<div><div></div><div>team-c</div></div>	2	07/28/20, 7:50AM	none	none
<div><div></div><div>team-d</div></div>	8	07/28/20, 7:51AM	none	none

## 実行中のジョブの送信：AI CLI

このセクションでは、Kubernetes ジョブの実行に使用できる基本的な Run : AI コマンドについて詳しく説明します。ワークロードの種類に応じて 3 つの要素に分割されます。AI / ML / DL のワークロードは、次の 2 つの汎用タイプに分けることができます。

- \* 無人トレーニングセッション \*。このようなタイプのワークロードを扱うことで、データサイエンティストは自己実行ワークロードを準備し、実行のためにワークロードを送信します。実行中に、お客様は結果を確認できます。このタイプのワークロードは、多くの場合、本番環境で使用されます。また、モデル開発が段階で行われるため、手動操作は必要ありません。

- \* インタラクティブビルドセッション \*。このようなワークロードの場合、データサイエンティストは Bash、Jupyter Notebook、リモート PyCharm などの IDE を使用した対話型セッションを開始し、GPU リソースに直接アクセスします。次に、対話型のワークロードを実行してポートを接続し、コンテナユーザに内部ポートを表示するシナリオを紹介します。

## 無人トレーニングのワークロード

プロジェクトをセットアップして GPU を割り当てたら、コマンドラインで次のコマンドを使用して Kubernetes のワークロードを実行できます。

```
$ runai project set team-a runai submit hyper1 -i gcr.io/run-ai-demo/quickstart -g 1
```

このコマンドは、単一の GPU を割り当てたチーム A の無人トレーニングジョブを開始します。このジョブは、サンプルの Docker イメージである「gcr.io/run-ai-demo/QuickStart」に基づいています。私たちはジョブ「hyper1」と名付けました。その後、次のコマンドを実行して、ジョブの進捗状況を監視します。

```
$ runai list
```

次の図に 'runai list' コマンドの結果を示します一般的なステータスは次のとおりです。

- 「ContainerCreating」を参照してください。Docker コンテナをクラウドリポジトリからダウンロードしています。
- 「保留中」。ジョブはスケジュールされるのを待っています。
- 「ランニング」。ジョブが実行中です。

```
~> runai list
Showing jobs for project team-a
NAME      STATUS  AGE  NODE                                     IMAGE                                     TYPE      PROJECT  USER  GPUs
hyper1    Running  11s  gke-dev-yaron1-gpu-4-pool-154f511d-5nk5 gcr.io/run-ai-demo/quickstart          Train     team-a   yaron  1
```

ジョブのステータスをさらに表示するには、次のコマンドを実行します。

```
$ runai get hyper1
```

ジョブのログを表示するには、「runai logs <job-name>」コマンドを実行します。

```
$ runai logs hyper1
```

この例では、現在のトレーニングエポック、ETA、損失関数値、精度、および各ステップの経過時間を含む、実行中の DL セッションのログを確認する必要があります。

クラスタのステータスは、Run : AI UI で確認できます から ["https://app.run.ai/"](https://app.run.ai/)。[ダッシュボード]>[概要]で、GPU 利用率を監視できます。

このワークロードを停止するには、次のコマンドを実行します。

```
$ runai delte hyper1
```

このコマンドを実行すると、トレーニングワークロードが停止します。このアクションを確認するには 'runai list' をもう一度実行します詳細については、を参照してください ["無人トレーニングワークロードの開始"](#)。

## ワークロードを対話型に構築

プロジェクトをセットアップして GPU を割り当てたら、コマンドラインで次のコマンドを使用して対話型ビルドワークロードを実行できます。

```
$ runai submit build1 -i python -g 1 --interactive --command sleep --args infinity
```

このジョブは、サンプルの Docker イメージ Python に基づいています。ジョブ build1 という名前を付けました。



「--interactive」フラグは、ジョブが開始または終了していないことを意味します研究者の責任で業務を終了してください。管理者は、対話型ジョブがシステムによって終了されるまでの時間制限を定義できます。

--g 1' フラグは 'このジョブに 1 つの GPU を割り当てます与えられたコマンドと引数は、「--command sleep --args インフィニティ」です。コマンドを指定するか、コンテナを開始してすぐに終了する必要があります。

次のコマンドは、で説明したコマンドと同様に機能します [\[無人トレーニングのワークロード\]](#)：

- runai list`：名前、ステータス、経過時間、ノード、イメージ、プロジェクト、ユーザ、GPU によるジョブの処理
- runai get build1`：ジョブ build1 の追加ステータスを表示します。
- runai delete build1`：対話型のワークロード build1 を停止しますコンテナに bash シェルを取得するには '次のコマンドを実行します

```
$ runai bash build1
```

これにより、コンピュータにシェルが直接挿入されます。データサイエンティストは、コンテナ内でモデルを開発または微調整できます。

クラスタのステータスは、Run : AI UI で確認できます から ["https://app.run.ai"](https://app.run.ai)。詳細については、を参照してください ["対話型ビルドワークロードの開始と使用"](#)。

## ポートが接続された対話型のワークロード

Run : AI CLI でコンテナを開始する際に、対話型ビルドワークロードを拡張することで、コンテナユーザに内部ポートを公開できます。これは、クラウド環境、Jupyter Notebook の操作、その他のマイクロサービス



への接続に役立ちます。"入力" Kubernetes クラスタの外部から Kubernetes サービスへのアクセスを許可します。アクセスを設定するには、どのインバウンド接続がどのサービスに到達するかを定義する一連のルールを作成します。

クラスタ内のサービスへの外部アクセスを適切に管理するには、クラスタ管理者がインストールすることを推奨します "入力" ロードバランサを設定します。

入力をサービスタイプとして使用するには、次のコマンドを実行して、ワークロード送信時にメソッドタイプとポートを設定します。

```
$ runai submit test-ingress -i jupyter/base-notebook -g 1 \
--interactive --service-type=ingress --port 8888 \
--args="--NotebookApp.base_url=test-ingress" --command=start-notebook.sh
```

コンテナが正常に起動したら、を実行します。runai list を参照してください。SERVICE URL(S) Jupyter Notebookへのアクセスに使用します。URL は、入力エンドポイント、ジョブ名、およびポートで構成されます。

詳細については、を参照してください "ポートが接続されている対話型のビルドワークロードを起動する"。

## 高いクラスタ利用率の達成

このセクションでは、4つのデータサイエンスチームがそれぞれ独自のワークロードを送信して実行：AI オークストレーション解決策を実証する現実的なシナリオをエミュレートしています。このシナリオでは、GPU リソースの優先順位付けとバランシングを維持しながら、クラスタの利用率を高めることができます。で説明した ResNet-50 ベンチマークを使用します セクション "ResNet-50 と ImageNet データセットベンチマークの概要"：

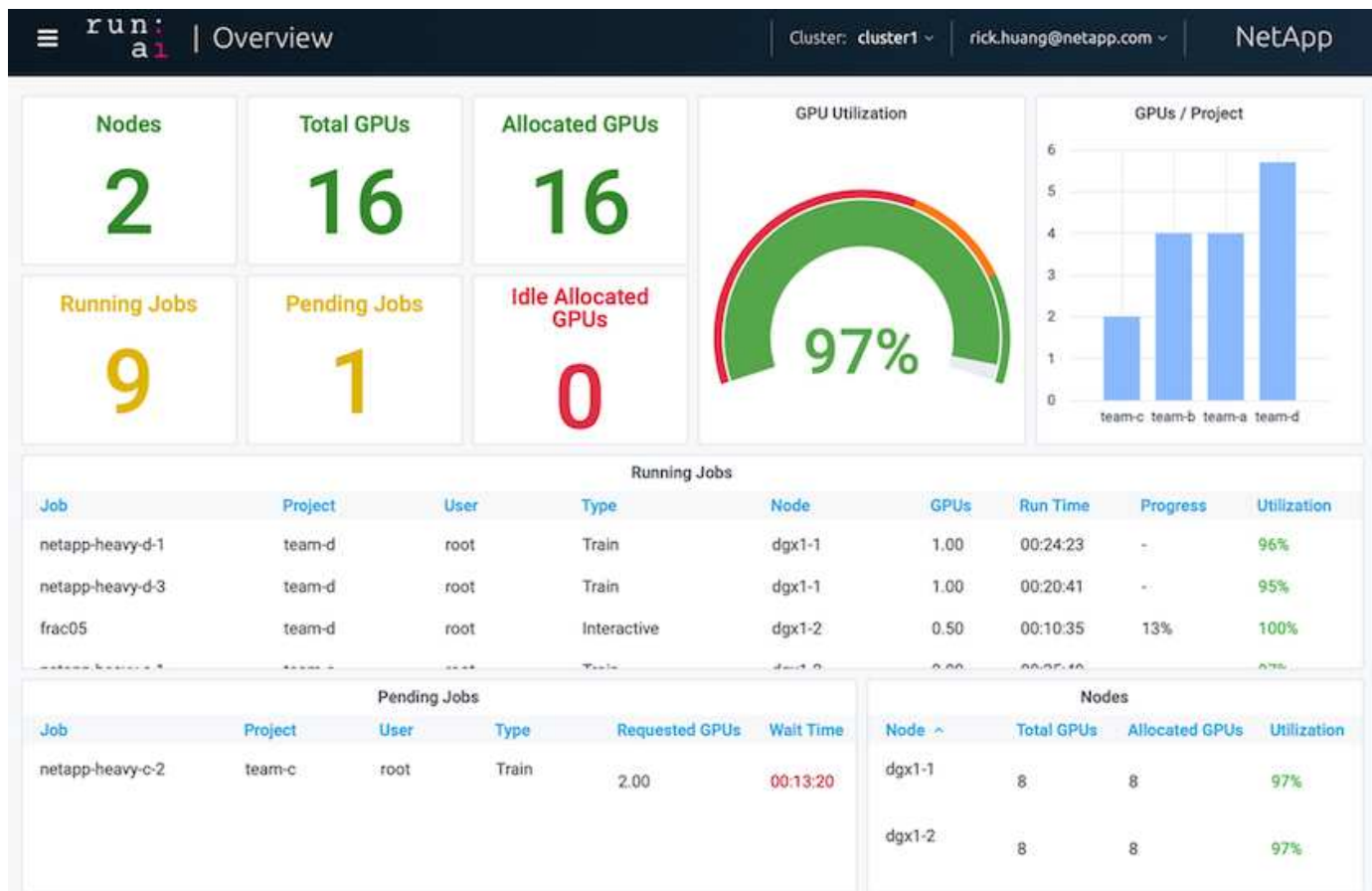
```
$ runai submit netapp1 -i netapp/tensorflow-tf1-py3:20.01.0 --local-image
--large-shm -v /mnt:/mnt -v /tmp:/tmp --command python --args
"/netapp/scripts/run.py" --args "--
dataset_dir=/mnt/mount_0/dataset/imagenet/imagenet_original/" --args "--
num_mounts=2" --args "--dgx_version=dgx1" --args "--num_devices=1" -g 1
```

ResNet-50 ベンチマークを実行しました（を参照） "NVA-1121."。パブリック Docker リポジトリに存在しないコンテナには、フラグ「--local-image」を使用しました。ディレクトリ「/mnt/」と「/tmp/」をホスト DGX-1 ノード上の「/mnt/」と「/tmp/」にそれぞれコンテナにマウントしました。データセットは、ディレクトリを指す「dataset\_dir」引数を持つ NetApp AFFA800 にあります。どちらの場合も '--num\_devices =1' と '-g 1' は 'このジョブに1つのGPUを割り当てていることを意味します前者は「run.py」スクリプトの引数で、後者は「runai submit」コマンドのフラグです。

次の図は、97%のGPU利用率を備え、16個の使用可能なGPUが割り当てられたシステム概要ダッシュボードを示しています。GPU/プロジェクトの棒グラフでは、各チームに割り当てられているGPUの数を簡単に確認できます。[実行中のジョブ]ウィンドウ枠には、現在実行中のジョブ名、プロジェクト、ユーザー、タイプ、ノード、GPUの消費、実行時間、進捗状況、利用率の詳細。キューに登録されているワークロードとその待機時間のリストが「保留中のジョブ」に表示されます。さらに、Nodes ボックスは、クラスタ内の



個々の DGX-1 ノードの GPU 番号と利用率を表示します。



## パフォーマンスの低いワークロードやインタラクティブなワークロードに適した、フラクショナルな GPU 割り当て

開発、ハイパーパラメータチューニング、デバッグのどの段階であっても、研究者や開発者が自分のモデルに取り組んでいる場合、このようなワークロードに必要なコンピューティングリソースは通常少なくなります。したがって、フラクショナル GPU とメモリをプロビジョニングして、同じ GPU を他のワークロードに同時に割り当てることがより効率的です。実行：AI のオーケストレーション解決策は、Kubernetes 上のコンテナ化されたワークロード向けのフラクショナル GPU 共有システムを提供します。CUDA プログラムを実行するワークロードをサポートするシステムで、推論やモデル構築などの軽量の AI タスクに特に適しています。フラクショナル GPU システムを使用すると、データサイエンスチームや AI エンジニアリングチームは、1 つの GPU で複数のワークロードを同時に実行できます。これにより、コンピュータビジョン、音声認識、自然言語処理など、より多くのワークロードを同じハードウェア上で実行できるようになり、コストが削減されます。

実行：AI のフラクショナル GPU システムは、仮想化された論理 GPU を独自のメモリとコンピューティングスペースで効率的に作成します。このスペースは、コンテナが自己完結型のプロセッサであるかのように使用およびアクセスできます。これにより、複数のワークロードを同じ GPU 上で並行して実行でき、互いに影響することはありません。解決策は透過的でシンプル、かつ移植可能であり、コンテナ自体に変更を加える必要はありません。

一般的な usecase では、同じ GPU 上で 2 ～ 8 個のジョブが実行されていることがわかります。つまり、同じハードウェアで作業を 8 倍行うことができます。

次の図のプロジェクト「team-d」に属するジョブ「分割 05」については、割り当てられた GPU の数が 0.50 であることがわかります。さらに 'nvidia-smi コマンドによって確認されますこのコマンドは' コンテナに使用できる GPU メモリが 'DGX-1 ノードの V100 GPU あたりの 32GB の半分である 16,255MB であることを示します

```
root@run-deploy:~# runai bash frac05 -p team-d
root@frac05-0:/workload# nvidia-smi
Tue Jul 28 15:17:03 2020

+-----+
| NVIDIA-SMI 450.51.05      Driver Version: 450.51.05      CUDA Version: 11.0      |
|-----+-----+-----+-----+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           | MIG M.         |
|=====+=====+=====+=====+=====+=====+
|    0   Tesla V100-SXM2...    On      | 00000000:07:00.0 Off |             0        |
| N/A   57C    P0     240W / 300W | 15525MiB / 16255MiB |   100%    Default   |
|                                           | N/A            |
+-----+-----+-----+-----+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU   GI    CI          PID    Type   Process name                  Usage      |
|=====+=====+=====+=====+=====+=====+
|    0   N/A   N/A         156      C      python3                      15525MiB |
+-----+-----+-----+-----+-----+-----+

```

## オーバークォータの GPU 割り当てによる高いクラスタ利用率の達成

を参照してください ["基本的な資源配分フェアネス"](#)および ["オーバークォータフェアネス"](#)また、高度なテストシナリオを考案し、複雑なワークロード管理、自動プリエンベティブスケジューリング、オーバークォータ GPU プロビジョニングを実現する Run : AI オークストレーション機能をデモしました。これにより、ONTAP AI 環境でクラスタリソースの利用率を高め、エンタープライズレベルのデータサイエンスチームの生産性を最適化できました。

これらの 3 つのセクションで、次のプロジェクトとクォータを設定します。

プロジェクト	クォータ
チーム A	4.
チーム - b	2.

プロジェクト	クォータ
チーム -c	2.
チーム -d	8.

また、この 3 つのセクションでは、次のコンテナを使用します。

- Jupyter Notebook : "jupyter/base-notebook
- 「GCR.IO/run-ai-demo/QuickStart」を実行します

このテストシナリオでは、次の目標を設定します。

- リソースのプロビジョニングの簡易性と、リソースの使用方法を説明します ユーザから抽象化されます
- ユーザが GPU のフラクションを簡単にプロビジョニングする方法を説明します GPU の数を整数で指定します
- チームを編成してコンピューティングのボトルネックを解消する方法を説明します リソースクォータがある場合は、ユーザがそのリソースクォータを確認します クラスタ内に GPU が搭載されている
- ネットアップコンテナなどの大量の計算処理を行うジョブを実行する際に、NetApp 解決策を使用してデータパイプラインのボトルネックを解消する方法を説明する
- を使用して複数のタイプのコンテナを実行する方法を説明します システム
  - Jupyter ノートブック
  - 実行: AI コンテナ
- クラスタがフルの状態のときに高い利用率を表示します

テスト中に実行される実際のコマンドシーケンスの詳細については、を参照してください ["セクション 4.8 のテストの詳細"](#)。

13 個のワークロードすべてが送信されると、次の図に示すように、コンテナ名と割り当てられている GPU のリストが表示されます。7 つのトレーニングと 6 つの対話型ジョブが用意されており、4 つのデータサイエンスチームをシミュレーションして、それぞれに独自のモデルを実行しているか開発中であることを確認しています インタラクティブなジョブの場合、個々の開発者は Jupyter Notebook を使用してコードの書き込みやデバッグを行っています。そのため、クラスタリソースをあまり使用せずに GPU フラクションをプロビジョニングすることをお勧めします。

```
root@run-deploy:~# runai list -A
NAME          STATUS  AGE  NODE  IMAGE                                     TYPE      PROJECT  USER  GPUs  CREATED BY CLI  SERVICE URL(S)
b-4-gg        Running  2m   dgx1-2  gcr.io/run-ai-demo/quickstart           Train     team-b   root   2     true           http://10.61.218.134/a-1-1-jupyter,
c-5-g          Running  2m   dgx1-2  gcr.io/run-ai-demo/quickstart           Train     team-c   root   1     true           http://10.61.218.134/a-1-1-jupyter,
c-4-gg        Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart           Train     team-c   root   2     true           http://10.61.218.134/a-1-1-jupyter,
b-3-g         Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart           Train     team-b   root   1     true           http://10.61.218.134/a-1-1-jupyter,
c-3-g02       Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart           Interactive team-c   root   0.2   true           http://10.61.218.134/a-1-1-jupyter,
d-1-gggg      Running  2m   dgx1-2  gcr.io/run-ai-demo/quickstart           Train     team-d   root   4     true           http://10.61.218.134/a-1-1-jupyter,
c-2-g03       Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart           Interactive team-c   root   0.3   true           http://10.61.218.134/a-1-1-jupyter,
c-1-g05       Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart           Interactive team-c   root   0.5   true           http://10.61.218.134/a-1-1-jupyter,
a-2-gg        Running  3m   dgx1-1  gcr.io/run-ai-demo/quickstart           Train     team-a   root   2     true           http://10.61.218.134/a-1-1-jupyter,
b-2-g04       Running  3m   dgx1-2  gcr.io/run-ai-demo/quickstart           Interactive team-b   root   0.4   true           http://10.61.218.134/a-1-1-jupyter,
a-1-g         Running  3m   dgx1-1  gcr.io/run-ai-demo/quickstart           Train     team-a   root   1     true           http://10.61.218.134/a-1-1-jupyter,
b-1-g06       Running  3m   dgx1-2  gcr.io/run-ai-demo/quickstart           Interactive team-b   root   0.6   true           http://10.61.218.134/a-1-1-jupyter,
a-1-1-jupyter Running  3m   dgx1-1  jupyter/base-notebook                   Interactive team-a   root   1     true           http://10.61.218.134/a-1-1-jupyter,
```

このテストシナリオの結果は次のようになります。

- クラスタがいっぱいである必要があります。 16 基の GPU が使用されています。

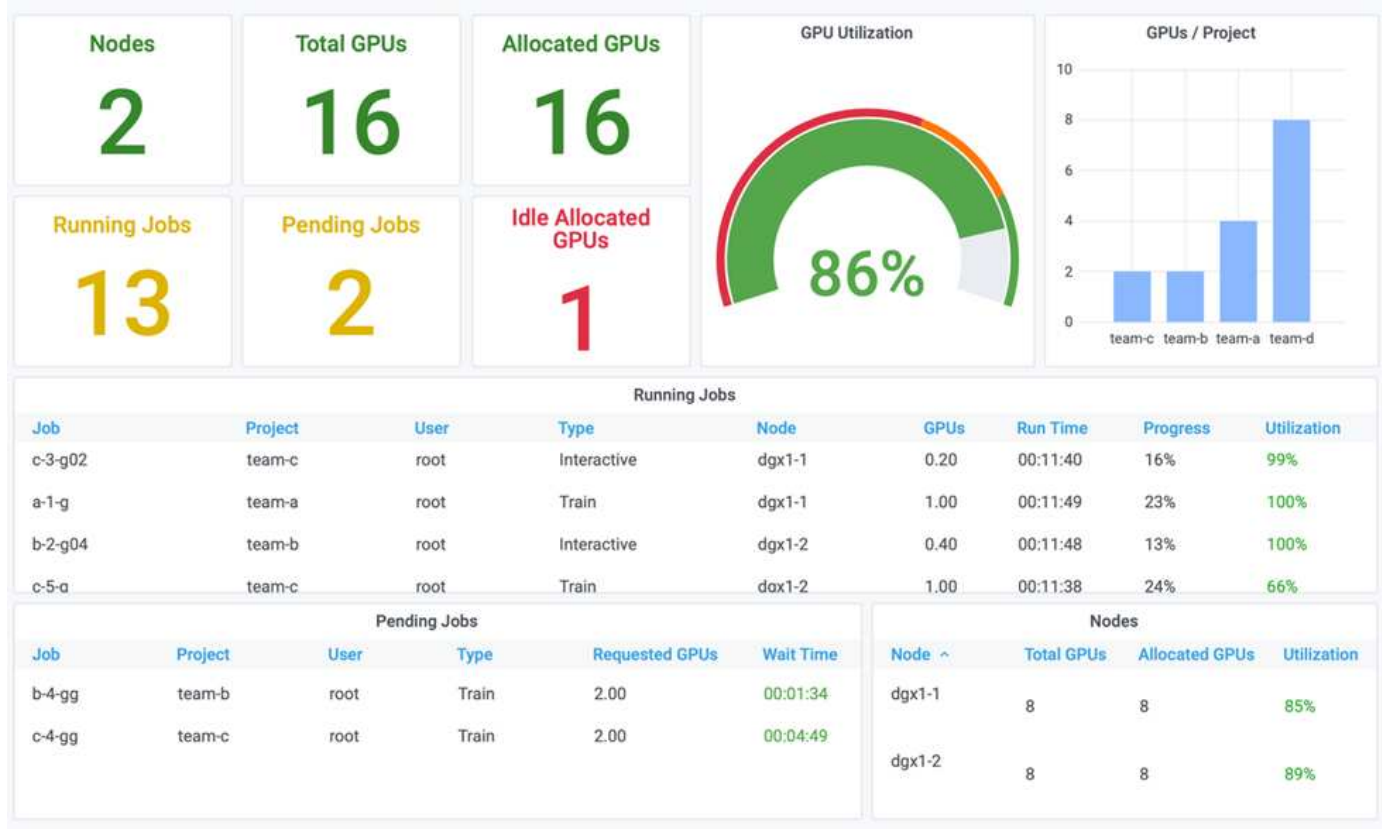
- クラスタの利用率が高い。
- フラクショナル割り当てにより、GPU よりも多くの実験が行われています。
- 「team -d」では、すべてのクォータが使用されているわけではありません。したがって、「team -b」と「team -c」では、実験に追加の GPU を使用することができるため、イノベーションにかかる時間が短縮されます。

## 基本的な資源配分フェアネス

このセクションでは 'team -d がより多くの GPU を要求した場合 (それらは割り当ての下にあります)' システムは 'team -b' および 'team -c のワークロードを一時停止し' 公正な共有方法で保留状態に移行することを示しています

ジョブの送信、使用するコンテナイメージ、実行するコマンドシーケンスなどの詳細については、を参照してください ["セクション 4.9 のテストの詳細"](#)。

次の図は、自動ロードバランシングとプリエンプティブスケジューリングにより、クラスタ使用率、チームごとに割り当てられた GPU、保留中のジョブを示しています。すべてのチームワークロードが要求した GPU の総数が、クラスタ内で使用可能な合計 GPU 数を超えると、実行：AI の内部公正性アルゴリズムによって、「team -b」と「team -c」のそれぞれに 1 つのジョブが一時停止されます。これは、それらがプロジェクトの割り当て量を満たしているためです。これにより、クラスタ全体の利用率が向上しますが、データサイエンスチームは、管理者が設定したリソースの制約に基づいて引き続き作業できます。



このテストシナリオの結果は、次のことを示しています。

- \* 自動ロードバランシング。\* システムは、各チームが割り当てを使用するように GPU の割り当て量を自動的に調整します。一時停止されていたワークロードは、そのクォータを超えていたチームに属していま



す。

- \* 公正な共有の一時停止。\* システムは、ノルマを達成したチームのワークロードを停止してから、もう一方のチームのワークロードを停止します。実行：AI には内部的な公正性アルゴリズムがあります。

## オーバークォータフェアネス

このセクションでは、複数のチームがワークロードを送信し、クォータを超過するシナリオを拡張します。この方法では、Run : AI の公正性アルゴリズムが、事前設定されたクォータの比率に従ってクラスタリソースを割り当てる方法を説明します。

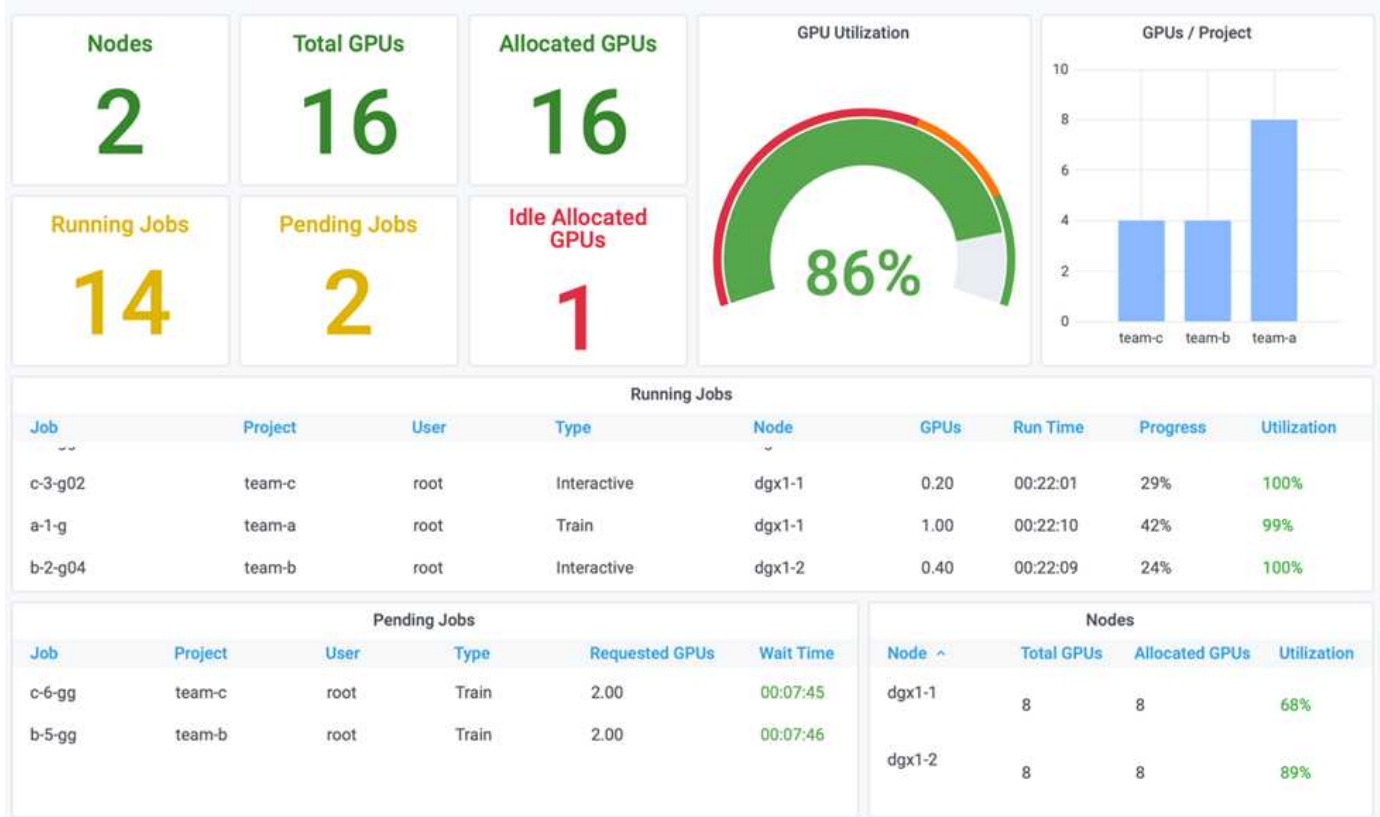
このテストシナリオの目標：

- 複数のチームがクォータを介して GPU を要求しているときのキューイングメカニズムを示します。
- システムが、クォータの比率に従って、クォータを超過した複数のチーム間にクラスタの適正な共有を分散し、クォータが大きいチームがスเปア容量の大部分を占めるようにする方法を示します。

の末尾 ["基本的な資源配分フェアネス"](#)では、キューに入れられるワークロードは2つあります。1つは「team-b」用、もう1つは「team-c」用です。このセクションでは、追加のワークロードをキューに登録します。

ジョブの送信、使用されるコンテナイメージ、実行されるコマンドシーケンスなどの詳細については、を参照してください ["セクション 4.10 のテストの詳細"](#)。

すべてのジョブがセクションに従って送信される場合 ["セクション 4.10 のテストの詳細"](#)をクリックすると、システムダッシュボードに team-a、team-b および team-c すべてのGPUが、あらかじめ設定されているクォータを超えています。team-a あらかじめ設定されているソフトクォータ（4つ）よりも4つ多くのGPUを使用しますが、team-b および team-c それぞれがソフトクォータ（2つ）よりも2つ多くのGPUを使用します。割り当てられたクォータ超過 GPU の比率は、事前設定されたクォータの比率と同じです。これは、システムが優先順位の基準として事前設定されたクォータを使用し、複数のチームがクォータを超えてGPUを追加するように要求した場合に応じてプロビジョニングされるためです。このような自動ロードバランシングは、企業のデータサイエンスチームがAIモデルの開発と運用に積極的に関与している場合に、公平性と優先順位付けを提供します。



このテストシナリオの結果は次のようになります。

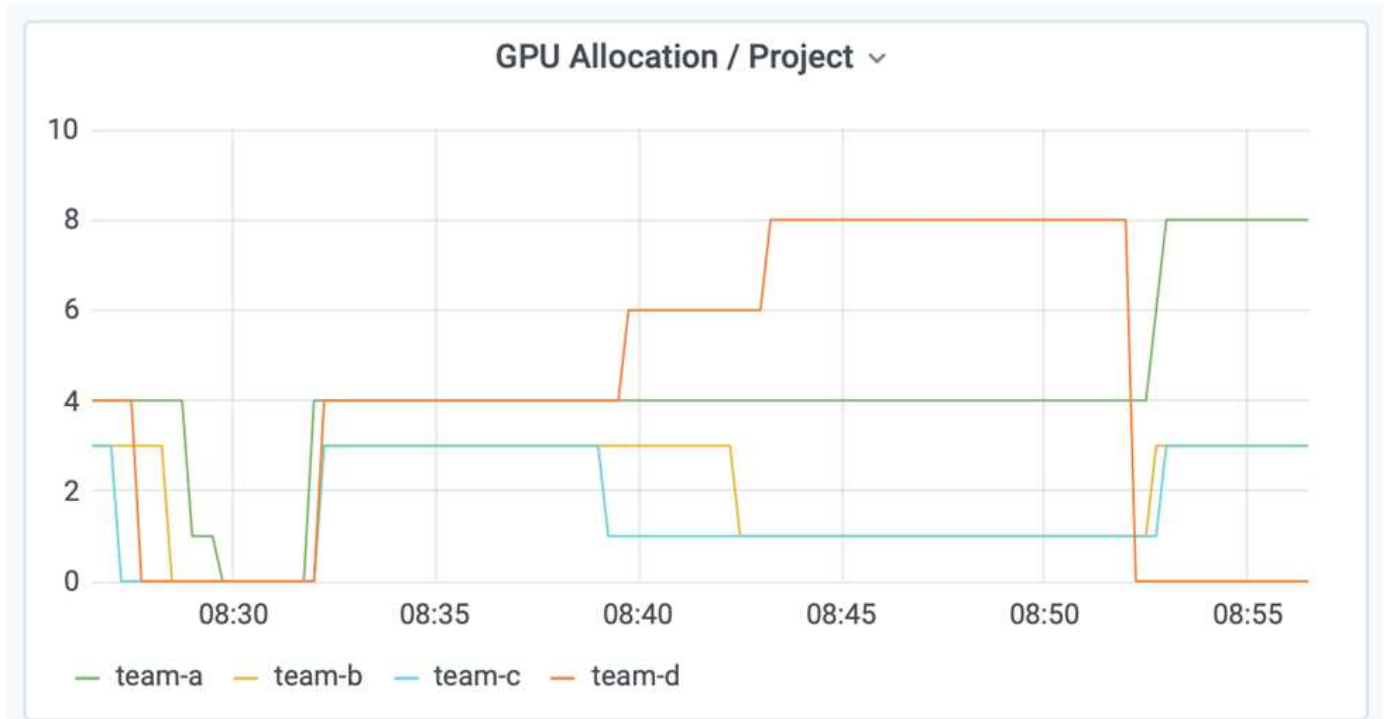
- 他のチームのワークロードのキュー解除が開始されます。
- キュー解除の順序は ' 公正性アルゴリズムに従って決定されますたとえば 'team -b と 'team -c は ' 同等のクォータを持つため ' 同じ量のオーバークォータ GPU を取得します また 'team -a は 'team -b と 'team -c のクォータの 2 倍のクォータを備えているため 'GPU の量が 2 倍になります
- すべての割り当てが自動的に行われます。

したがって、システムは次の状態で安定します。

プロジェクト	GPU が割り当てられました	コメント ( Comment )
チーム A	8/4	クォータを介した 4 基の GPU 空のキューです。
チーム - b	4 月 2 日	クォータを介した 2 つの GPU 。 1 つのワークロードがキューに登録
チーム -c	4 月 2 日	クォータを介した 2 つの GPU 。 1 つのワークロードがキューに登録
チーム -d	0/8	GPU をまったく使用しないので、キューに登録されているワークロードはありません

次の図は、プロジェクトごとの GPU 割り当てを示しています Run : セクションの AI Analytics ダッシュボードに表示される時間 "オーバークォータの GPU 割り当てによる高いクラスタ利用率の達成"、 "基本的な資源配分フェアネス"および "オーバークォータフェアネス"。図の各行は、特定のデータサイエンスチーム用にプロビジョニングされた GPU の数を常に表しています。システムは、送信されたワークロードに応じて GPU

を動的に割り当てることがわかります。これにより、クラスタ内に使用可能な GPU がある場合はクォータを超過し、公平性に従ってジョブをプリエンプトしてから、4 つのチームすべてが最終的に安定した状態に到達することができます。



## Trident でプロビジョニングされた永続的ボリュームにデータを保存する

NetApp Trident は、コンテナ化されたアプリケーションが求める高度な永続性のニーズに対応できるように設計された、完全にサポートされているオープンソースプロジェクトです。Trident でプロビジョニングされた Kubernetes PersistentVolume (PV) に対してデータの読み取りと書き込みを行うことができ、データ階層化、暗号化、NetApp Snapshot テクノロジー、コンプライアンス、NetApp ONTAP データ管理ソフトウェアが提供する高パフォーマンスといったメリットも活用できます。

### 既存の名前空間での PVC の再利用

大規模な AI プロジェクトでは、異なるコンテナで同じ Kubernetes PV に対してデータの読み取りや書き込みを行う方が効率的な場合があります。Kubernetes Persistent Volume Claim (PVC ; 永続ボリューム要求) を再利用するには、ユーザが PVC を作成しておく必要があります。を参照してください ["NetApp Trident のドキュメント"](#) PVC 作成の詳細については、を参照してください。次に、既存の PVC を再利用する例を示します。

```
$ runai submit pvc-test -p team-a --pvc test:/tmp/pvc1mount -i gcr.io/run-ai-demo/quickstart -g 1
```

次のコマンドを実行して 'プロジェクト 'team-a' のジョブ pvc-test' のステータスを表示します



```
$ runai get pvc-test -p team-a
```

'team -a' ジョブ 'pvctest' にマウントされた PV /tmp/pvc1mount が表示されますこのようにして、複数のコンテナが同じボリュームから読み取ることができるため、開発中または本番環境で競合する複数のモデルが存在する場合に便利です。データサイエンティストは、モデルのアンサンブルを構築し、大多数の投票またはその他の技術によって予測結果を組み合わせることができます。

次のコマンドを使用してコンテナシェルにアクセスします。

```
$ runai bash pvc-test -p team-a
```

その後、マウントされたボリュームを確認し、コンテナ内のデータにアクセスできます。

PVC の再利用というこの機能は、NetApp FlexVol ボリュームと NetApp ONTAP FlexGroup ボリュームと連携します。そのため、データエンジニアは、より柔軟で堅牢なデータ管理オプションを利用して、ネットアップのデータファブリックを活用できます。

## 著作権に関する情報

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S. このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

## 商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。