解決策の導入と検証の詳細

NetApp Solutions

NetApp
April 10, 2024

目次

解決策の導入と検証の詳細

以降のセクションでは、解決策の導入と検証の詳細について説明します。

ONTAP AI の導入

ONTAP AI を導入するには、ネットワーク、コンピューティング、ストレージハードウェアのインストールと設定が必要です。ONTAP AI インフラの導入手順については、本ドキュメントでは説明していません。導入の詳細については、を参照してください "NVA-1121-deploy ： NetApp ONTAP AI 、 Powered by NVIDIA"。

この解決策検証では、 1 つのボリュームを作成して DGX-1 システムにマウントしました。その後、そのマウントポイントをコンテナにマウントして、トレーニング用のデータにアクセスできるようにしました。大規模な環境では、 NetApp Trident によってボリュームの作成とマウントが自動化されるため、管理上のオーバーヘッドが発生しないとともに、エンドユーザによるリソースの管理が可能になります。

Kubernetes の導入

NVIDIA DeepOps で Kubernetes クラスタを導入および設定するには、導入ジャンプホストから次のタスクを実行します。

1. の手順に従って、 NVIDIA DeepOps をダウンロードします "「はじめに」ページ" NVIDIA DeepOps GitHub サイトで入手できます。
2. の手順に従って、クラスタに Kubernetes を導入します 。 "Kubernetes 導入ガイド" NVIDIA DeepOps GitHub サイトで入手できます。

> DeepOps Kubernetes 環境を使用するには、 Kubernetes マスターノードとワーカーノードがすべて同じユーザである必要があります。

配備に失敗した場合は 'kubectl_localhost' の値を 'd eepops/config/group_vars/k8s-cluster.yml' で false に変更し ' 手順 2 を繰り返します'Copy kubectl binary to Ansible host' タスクは 'kubectl_localhost' の値が true の場合にのみ実行され ' メモリ使用に関する既知の問題がある FETCH Ansible モジュールに依存しますこのようなメモリ使用の問題により、原因がタスクを失敗させることがあります。メモリ問題が原因でタスクが失敗した場合は、以降の導入処理は正常に完了しません。

「 kubectl_localhost 」の値を「 false 」に変更した後で展開が正常に完了した場合、「 kubectl binary 」を Kubernetes マスターノードから配備ジャンプホストに手動でコピーする必要があります。特定のマスター・ノード上で 'kubectl binary' の場所を確認するには 'which kubectl' コマンドをそのノード上で直接実行します

cnvrg.io の展開

Helm を使用して cnvrg コアを導入します

Helm は、任意のクラスタ、オンプレミス、 Minikube 、または任意のクラウドクラスタ（ AKS 、 EKS 、 GKE など）を使用して、 cnvrg を迅速に導入する最も簡単な方法です。このセクションでは、 Kubernetes がインストールされたオンプレミス（ DGX-1 ）インスタンスに cnvrg がインストールされた方法について説明します。

前提条件

インストールを完了する前に、ローカルマシンに次の依存関係をインストールして準備する必要があります。

- Kubectl のように入力する
- Helm 3.x
- Kubernetes クラスタ 1.15 以降

Helm を使用して展開します

1. 最新の cnvrg Helm チャートをダウンロードするには、次のコマンドを実行します。

```
helm repo add cnvrg https://helm.cnvrg.io
helm repo update
```

2. cnvrg を導入する前に、クラスタの外部 IP アドレス、および cnvrg を導入するノードの名前が必要です。オンプレミスの Kubernetes クラスタに cnvrg を導入するには、次のコマンドを実行します。

```
helm install cnvrg cnvrg/cnvrg --timeout 1500s  --wait \ --set
global.external_ip=<ip_of_cluster> \ --set global.node=<name_of_node>
```

3. 「 helm install 」コマンドを実行します。すべてのサービスとシステムがクラスタに自動的にインストールされます。この処理には最大 15 分かかることがあります。
4. 「 helm install 」コマンドの所要時間は最大 10 分です。展開が完了したら、新しく展開した cnvrg の URL に移動するか、新しいクラスタを組織内のリソースとして追加します。「 helm' 」コマンドは正しい URL を通知します。

```
Thank you for installing cnvrg.io!
Your installation of cnvrg.io is now available, and can be reached via:
Talk to our team via email at
```

5. すべてのコンテナのステータスが「 Running 」または「 Complete 」の場合、 cnvrg は正常に展開されています。次のような出力が表示されます。

```
NAME                                READY   STATUS      RESTARTS   AGE
cnvrg-app-69fbb9df98-6xrgf                  1/1     Running     0          2m
cnvrg-sidekiq-b9d54d889-5x4fc               1/1     Running     0          2m
controller-65895b47d4-s96v6                 1/1     Running     0          2m
init-app-vs-config-wv9c4                    0/1     Completed   0          9m
init-gateway-vs-config-2zbpp                0/1     Completed   0          9m
init-minio-vs-config-cd2rg                  0/1     Completed   0          9m
minio-0                                     1/1     Running     0          2m
postgres-0                                  1/1     Running     0          2m
redis-695c49c986-kcbt9                      1/1     Running     0          2m
seeder-wh655                                0/1     Completed   0          2m
speaker-5sghr                               1/1     Running     0          2m
```

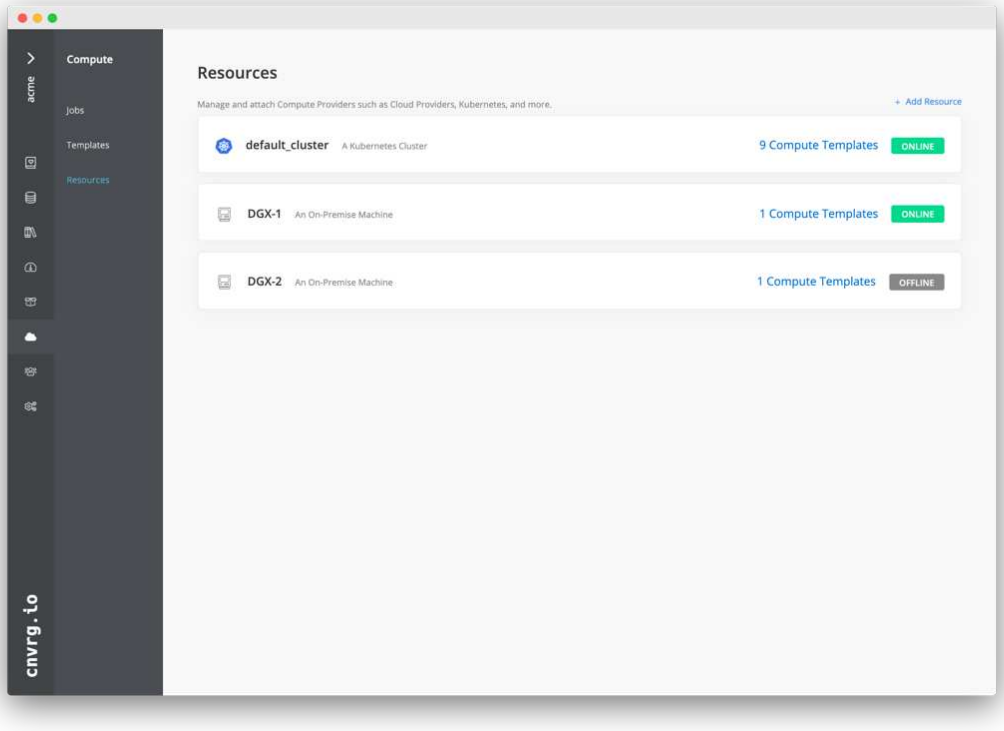
ResNet50 および胸部 X 線を使用したコンピュータビジョンモデルトレーニング データセット

NVIDIA DGX システムを基盤とする NetApp ONTAP AI アーキテクチャ上の Kubernetes セットアップに、cnvrg.io AI OS が導入されました。検証には、胸部 X 線の匿名画像からなる NIH 胸部 X 線データセットを使用しました。画像は PNG 形式でした。このデータは NIH クリニカルセンタおよびによって提供された は、から使用できます "NIH ダウンロードサイト"。250 GB のサンプルデータを 15 クラスの 627 、 615 イメージで使用しました。

データセットは cnvrg プラットフォームにアップロードされ、 NetApp AFF A800 ストレージシステムからの NFS エクスポートにキャッシュされました。

コンピューティングリソースをセットアップする

cnvrg アーキテクチャおよびメタスケジューリング機能により、エンジニアおよび IT プロフェッショナルは、異なるコンピューティングリソースを 1 つのプラットフォームに接続できます。今回のセットアップでは、ディープラーニングワークロードの実行用に導入されたクラスタ cnvrg を使用しました。追加のクラスタを接続する必要がある場合は、次のスクリーンショットに示すように、 GUI を使用してください。



データをロードします

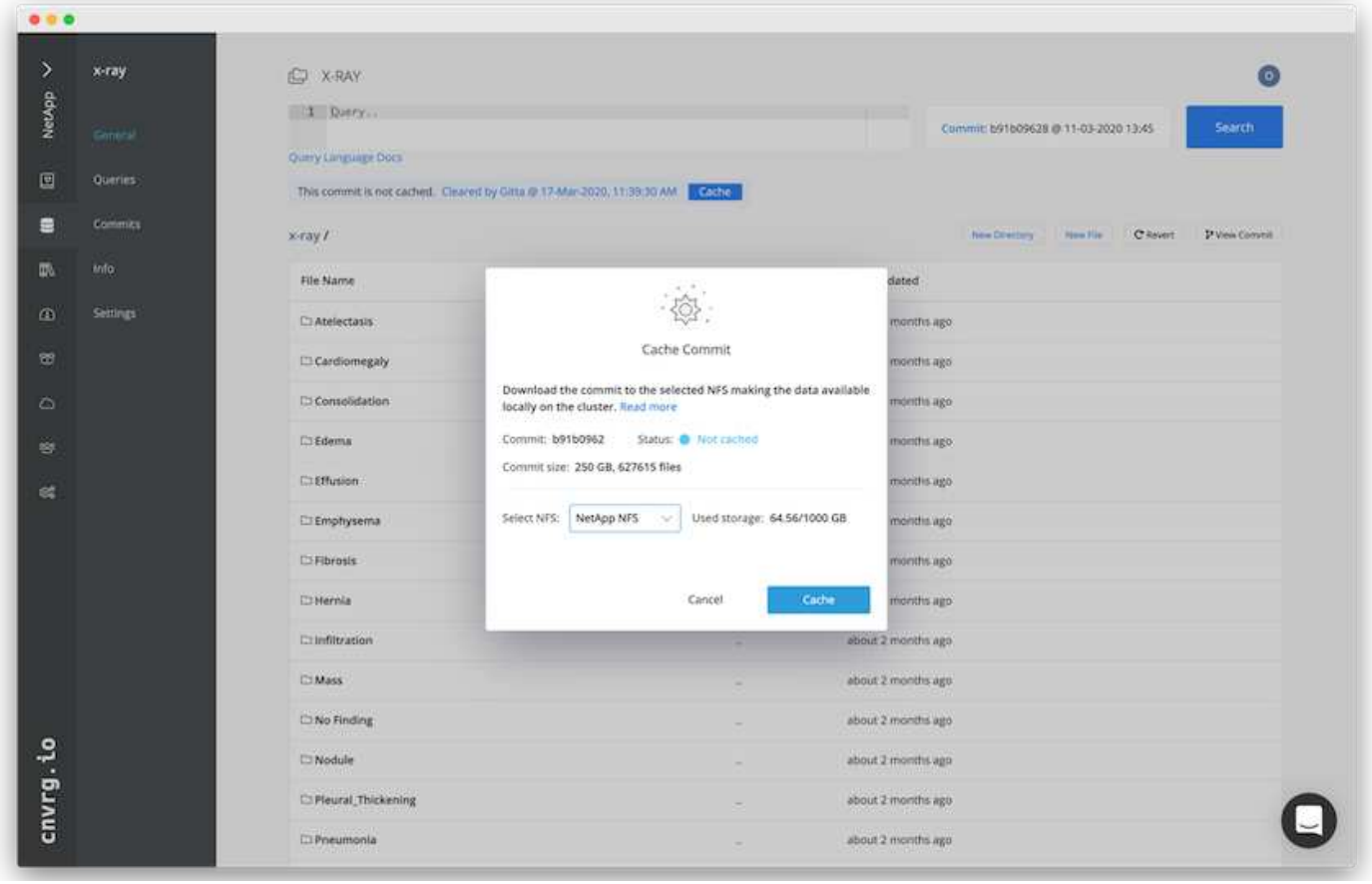
cnvrg プラットフォームにデータをアップロードするには、 GUI または cnvrg CLI を使用します。大規模なデータセットの場合は、 CLI の使用を推奨します。 CLI は、多数のファイルを処理できる、拡張性と信頼性に優れた強力なツールです。

データをアップロードするには、次の手順を実行します。

1. をダウンロードします "cnvrg CLI"。
2. X 線ディレクトリに移動します。
3. 「 cnvrg data init 」コマンドを使用して、プラットフォーム内のデータセットを初期化します。
4. 「 cnvrg data sync 」コマンドを使用して、ディレクトリのすべての内容を中央のデータレイクにアップロードします。データが中央のオブジェクトストア（ StorageGRID 、 S3 、またはその他）にアップロードされたら、 GUI で参照できます。次の図は、ロードされた胸部 X 線線維症画像 PNG ファイルを示しています。さらに、 cnvrg は、ビルドしたすべてのモデルをデータバージョンに複製できるように、データをバージョン化します。

マッハデータ

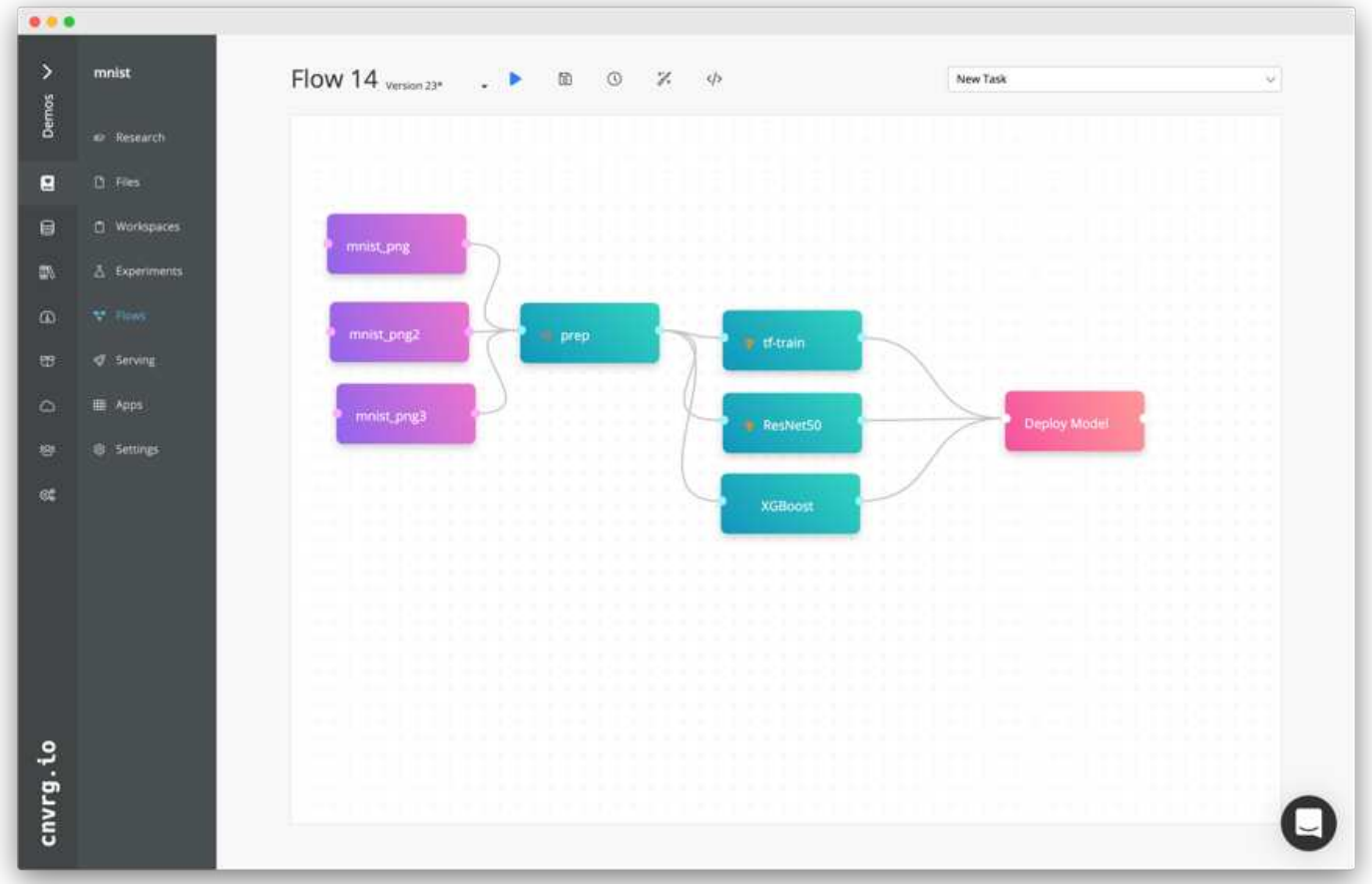
トレーニングを高速化し、モデルのトレーニングや実験ごとに 600k 以上のファイルをダウンロードしないようにするために、データを最初に中央のデータレイクオブジェクトストアにアップロードしたあとにデータキャッシュ機能を使用しました。



ユーザーが Cache をクリックすると、 cnvrg はリモートオブジェクトストアから特定のコミットでデータをダウンロードし、 ONTAP NFS ボリュームにキャッシュします。完了すると、データをすぐにトレーニングに利用できるようになります。さらに、データが数日間使用されていない場合（たとえば、モデルのトレーニングや探索など）、 cnvrg は自動的にキャッシュをクリアします。

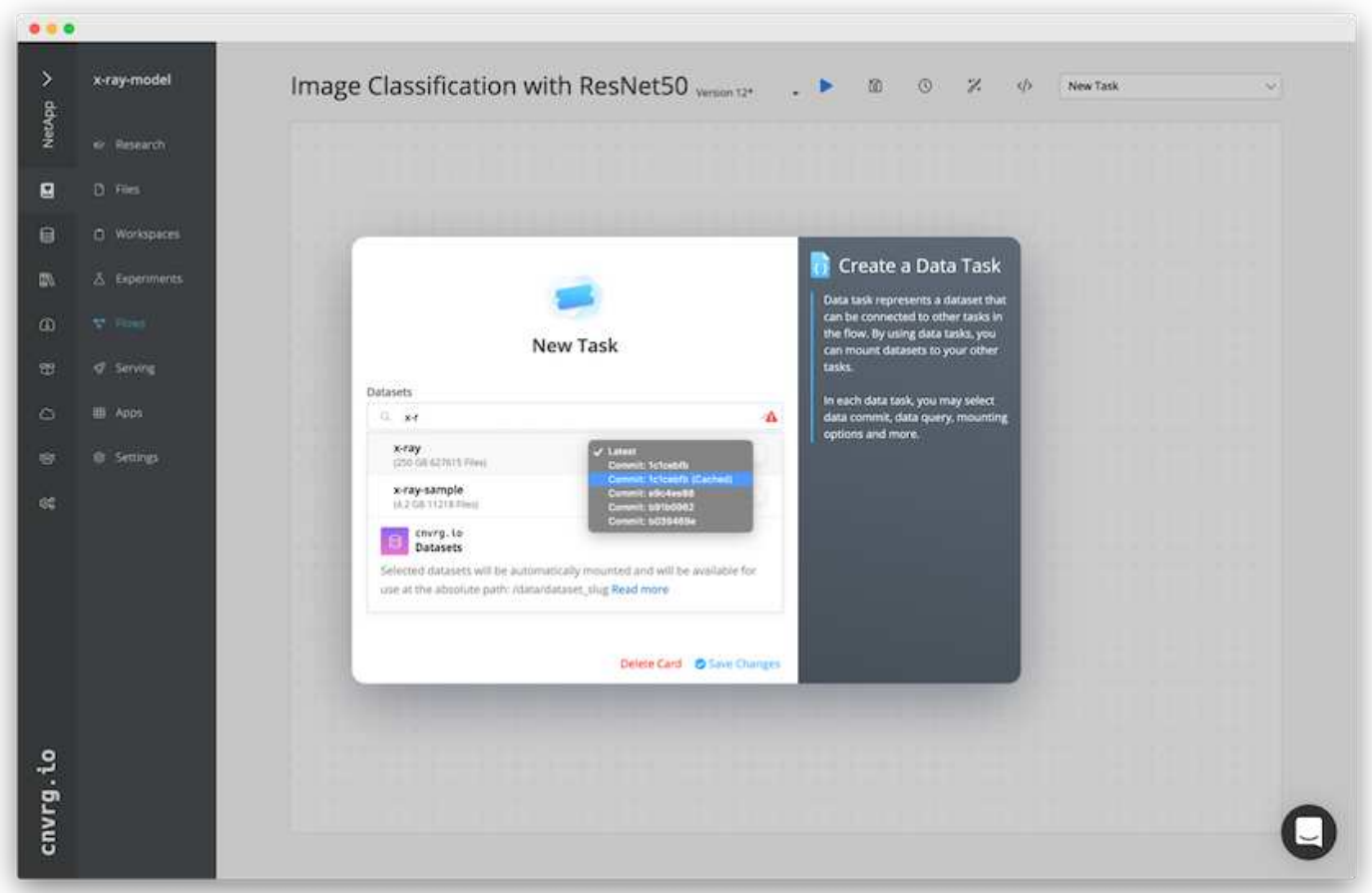
キャッシュデータで **ML** パイプラインを構築

cnvrg フローを使用すると、本番 ML パイプラインを簡単に構築できます。フローは柔軟性が高く、あらゆる種類の ML ユースケースに対応し、 GUI またはコードを使用して作成できます。フロー内の各コンポーネントは、異なる Docker イメージを使用して異なるコンピューティングリソース上で実行できるため、ハイブリッドクラウドを構築し、 ML パイプラインを最適化できます。



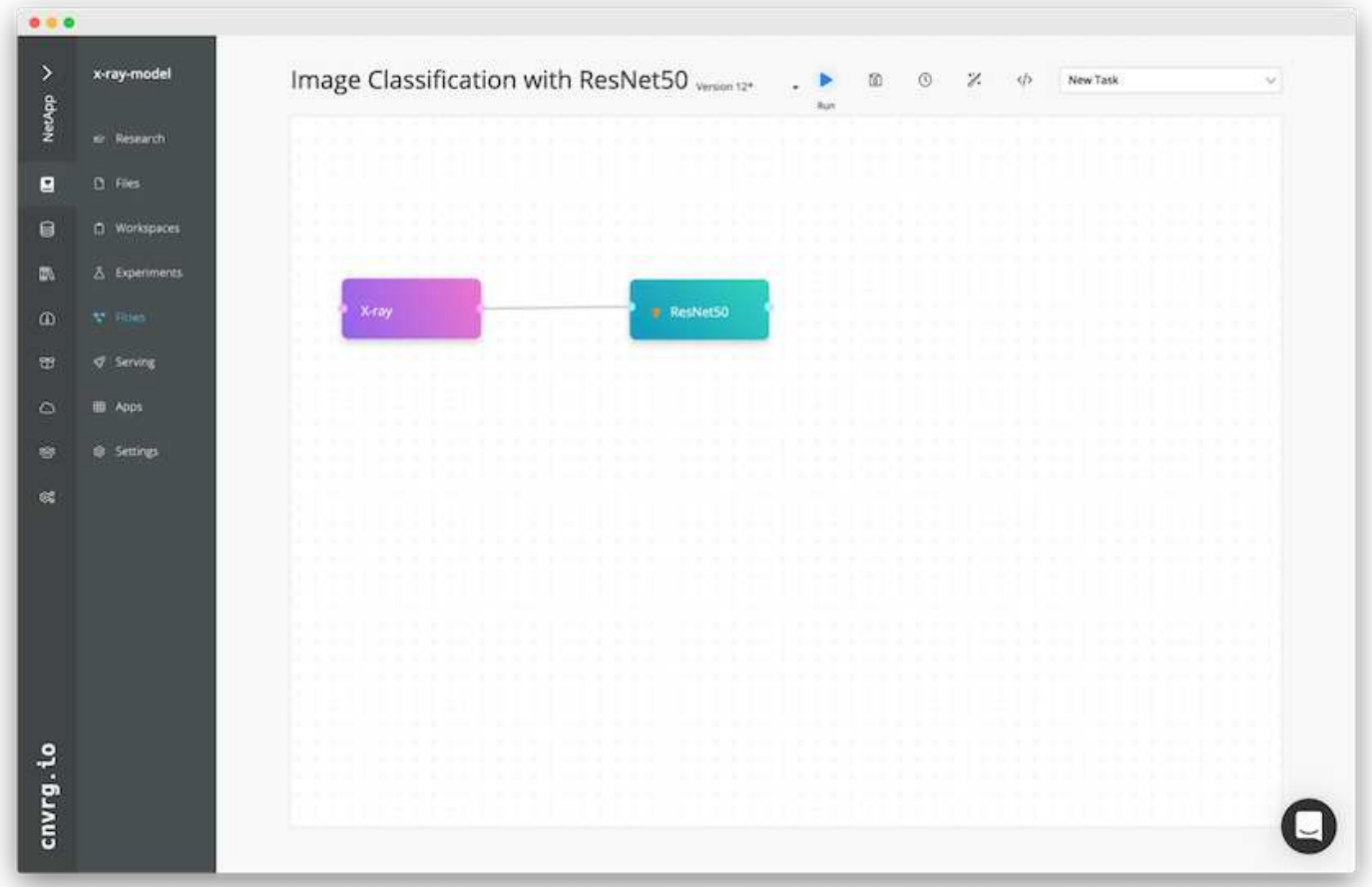
胸部 **X** 線フローの構築：データの設定

新しく作成したフローにデータセットを追加しました。データセットを追加する際には、特定のバージョン（commit ）を選択し、キャッシュされたバージョンが必要かどうかを指定できます。この例では、キャッシュされたコミットを選択しました。



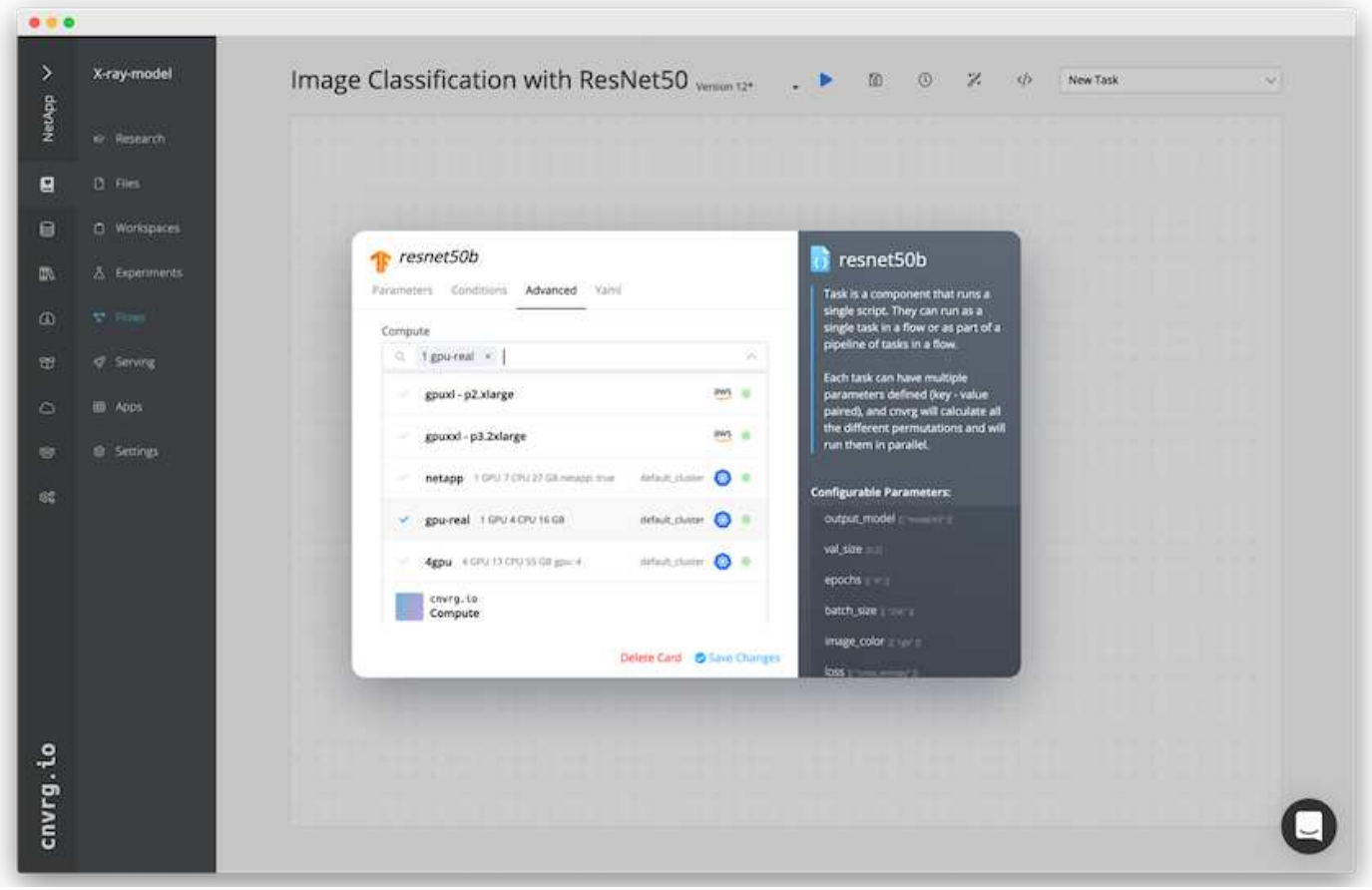
胸部 **X** 線フローの構築：トレーニングモデルの設定： **ResNet50**

パイプラインでは、任意の種類のカスタムコードを追加できます。cnvrg には、再利用可能な ML コンポーネントコレクションである AI ライブラリもあります。AI ライブラリには、アルゴリズム、スクリプト、データソースなど、あらゆる ML やディープラーニングフローで使用できるソリューションがあります。この例では、 ResNet50 の事前ビルドモジュールを選択しました。batch_size ： 128 、 epochs ： 10 などのデフォルトパラメータを使用しました。これらのパラメータは AI ライブラリのドキュメントで確認できます。次のスクリーンショットは、 X 線データセットが ResNet50 に接続された新しいフローを示しています。



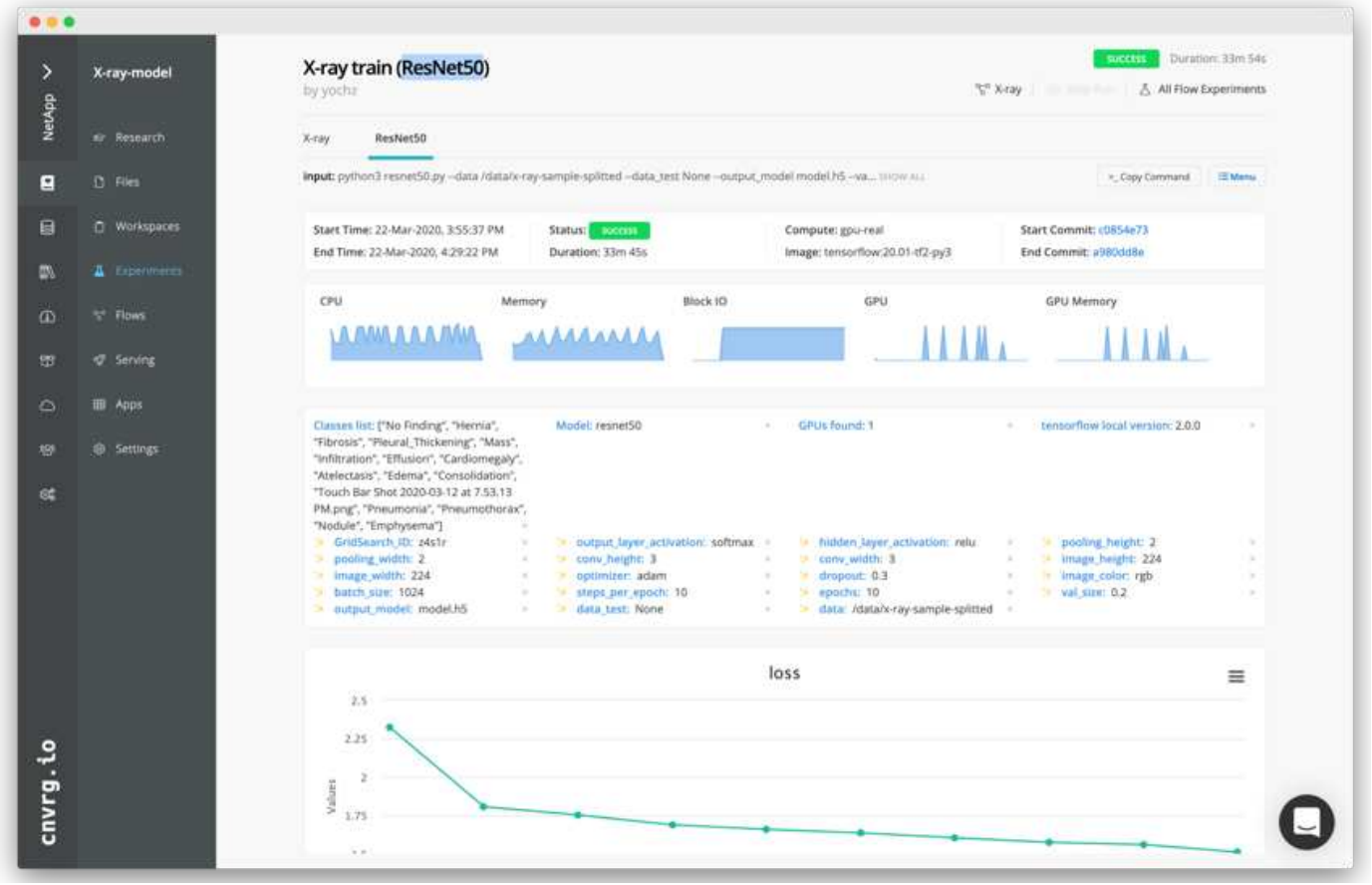
ResNet50 の計算リソースを定義します

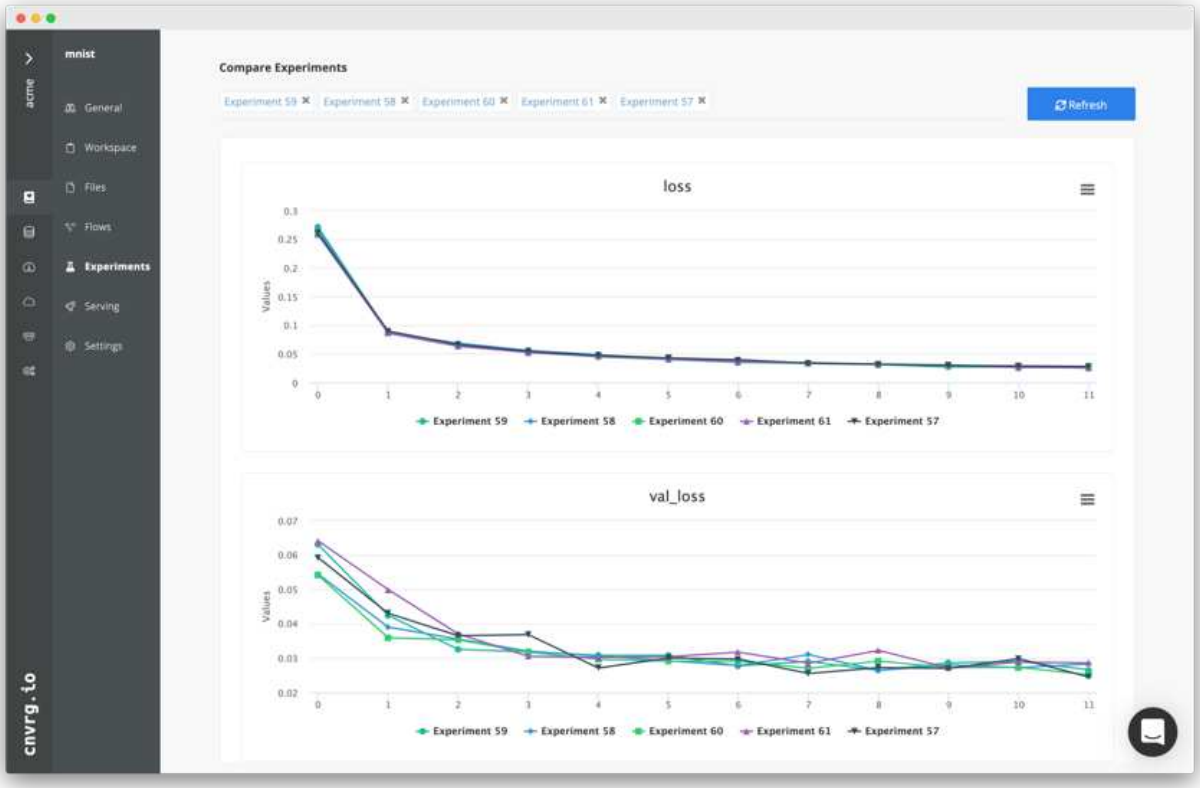
cnvrg フロー内の各アルゴリズムまたはコンポーネントは、異なる Docker イメージを使用して、異なるコンピューティングインスタンス上で実行できます。セットアップでは、 NetApp ONTAP AI アーキテクチャを採用した NVIDIA DGX システムでトレーニングアルゴリズムを実行したいと考えていました。次の図では、「 GPU - REAL 」を選択しました。これは、オンプレミスクラスタのコンピューティングテンプレートであり、仕様です。また、テンプレートのキューを作成し、複数のテンプレートを選択しました。このようにして 'GPU 実数のリソースを割り当てることができない場合 (たとえば ' 他のデータ・サイエンティストがリソースを使用している場合) は ' クラウド・プロバイダ・テンプレートを追加して ' 自動クラウド・バーストを有効にできます次のスクリーンショットは、 ResNet50 のコンピューティングノードとしての GPU 実数の使用を示しています。



結果の追跡と監視

フローが実行されると、 cnvrg はトラッキングおよびモニタリングエンジンをトリガーします。フローの各実行は自動的に文書化され、リアルタイムで更新されます。ハイパーパラメータ、指標、リソース使用率（ GPU 利用率など）、コードバージョン、アーティファクト、ログ また、次の 2 つのスクリーンショットに示すように、 ［テスト］ セクションで自動的に使用できるようになります。


NetApp
X-ray-model
Research
Files
Workspaces
Experiments
Flows
Serving
Apps
Settings
cnvrg.io
X-ray train (ResNet50)
by yochz
SUCCESS
Duration: 33m 54s
X-ray
All Flow Experiments
X-ray
ResNet50
input: python3 resnet50.py --data /data/x-ray-sample-splitted --data_test None --output_model model.h5 --va...
Copy Command
Menu
Start Time: 22-Mar-2020, 3:55:37 PM
End Time: 22-Mar-2020, 4:29:22 PM
Status: SUCCESS
Duration: 33m 45s
Compute: gpu-real
Image: tensorflow:20.01-tf2-py3
Start Commit: c0854e73
End Commit: a980dd8e
CPU
Memory
Block IO
GPU
GPU Memory
Classes list: ["No Finding", "Hernia", "Fibrosis", "Pleural_Thickening", "Mass", "Infiltration", "Effusion", "Cardiomegaly", "Atelectasis", "Edema", "Consolidation", "Touch Bar Shot 2020-03-12 at 7.53.13 PM.png", "Pneumonia", "Pneumothorax", "Nodule", "Emphysema"]
Model: resnet50
GPUs found: 1
tensorflow local version: 2.0.0
GridSearch_ID: z4s1r
pooling_width: 2
image_width: 224
batch_size: 1024
output_model: model.h5
output_layer_activation: softmax
conv_height: 3
optimizer: adam
steps_per_epoch: 10
data_test: None
hidden_layer_activation: relu
conv_width: 3
dropout: 0.3
epochs: 10
data: /data/x-ray-sample-splitted
pooling_height: 2
image_height: 224
image_color: rgb
val_size: 0.2
loss
Values
2.5
2.25
2
1.75


acme
mnist
General
Workspace
Files
Flows
Experiments
Serving
Settings
cnvrg.io
Compare Experiments
Experiment 59
Experiment 58
Experiment 60
Experiment 61
Experiment 57
Refresh
loss
Values
0.3
0.25
0.2
0.15
0.1
0.05
0
0 1 2 3 4 5 6 7 8 9 10 11
Experiment 59
Experiment 58
Experiment 60
Experiment 61
Experiment 57
val_loss
Values
0.07
0.06
0.05
0.04
0.03
0.02
0 1 2 3 4 5 6 7 8 9 10 11
Experiment 59
Experiment 58
Experiment 60
Experiment 61
Experiment 57