Technical Report

Hybrid Cloud AI Operating System with Data Caching

Creating Datasets Proximity to Hybrid Cloud Compute with Full Automation and Collaboration

Rick Huang, David Arnette, NetApp
Yochay Ettun, cnvrg.io
June 2020 | TR-4841

Abstract

The explosive growth of data and the exponential growth of machine learning (ML) and artificial intelligence (AI) have converged to create a new economy with unique development and implementation obstacles. Massive quantities of data are usually stored in a low-cost data lakes, while high performance AI compute (such as GPUs) cannot efficiently access this data. In this paper, we present a novel solution that allows data science practitioners to have a data hub, and with one click create a cache of their datasets in proximity to the compute—wherever it is located. As a result, not only is the high-performance model training accomplished, additional benefits are created such as collaboration of multiple AI practitioners and the ability to create a dataset version hub.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Executive Summary

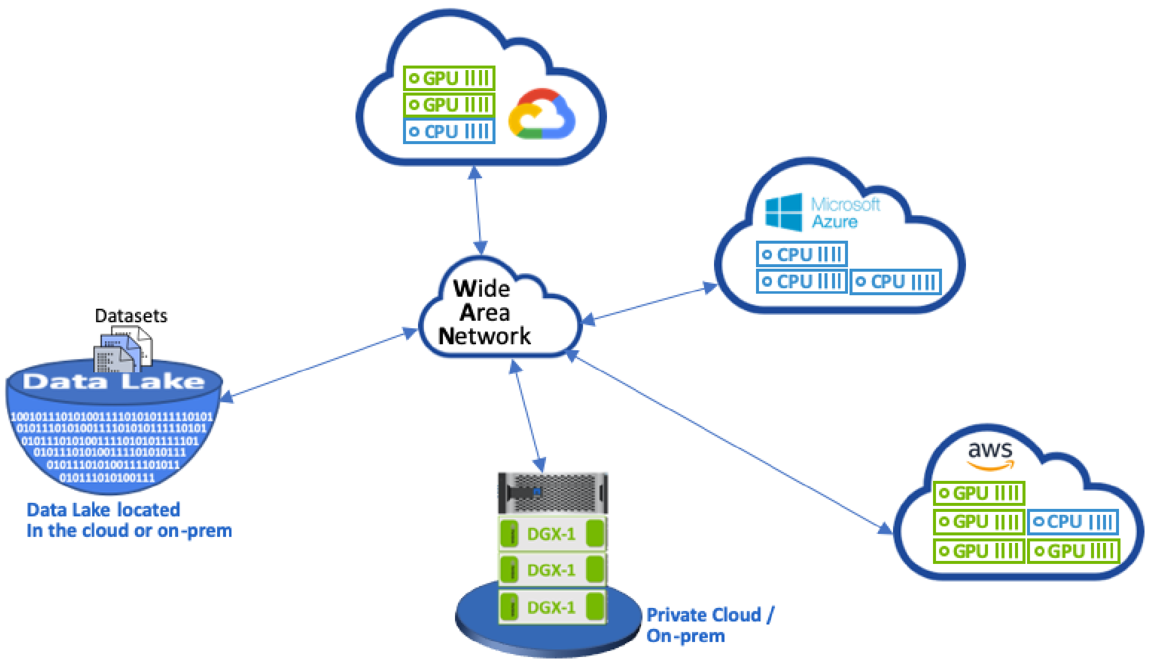
The explosive growth of data and the exponential growth of ML and AI have converged to create a zettabyte economy with unique development and implementation obstacles. While it is a widely accepted knowledge that ML models are data-hungry and require high-performance data storage proximity to the compute, in practice, it is not so straight forward to implement, especially with hybrid cloud and elastic compute instances. Massive quantities of data are usually stored in low-cost data lakes, while high performance AI compute (such as GPUs) cannot efficiently access this data. The problem is being aggravated in the presence of a hybrid cloud infrastructure where some workloads operate in the cloud and some are placed on-premises, or in a different HPC environment.

In this document, we present a novel solution that allows IT professionals and data engineers to create a truly hybrid cloud AI platform with a topology-aware data hub that enables data scientists to instantly and automatically create a cache of their datasets in proximity to the compute—wherever this compute is located. As a result, not only can the high-performance model training be accomplished, but additional benefits are created, such as collaboration of multiple AI practitioners, which has immediate access to the cached datasets, versions, and lineage of datasets and the ability to create a dataset version hub.

2 Use Case Overview and Problem Statement

Datasets and dataset versions are typically located at a data lake, usually an object storage such as NetApp® StorageGrid® object-based storage software, which offers lower costs and other operational advantages. Data scientists pull these datasets and engineer them in multiple steps to prepare them for training with a specific model, often creating multiple versions along the way. As the next step, the data scientist must pick optimized compute (GPUs, high-end CPU instances, on-premises cluster, and so on) to run the model.

Figure 1) Lack of dataset proximity to the ML compute environment.



However, the challenge is that multiple training experiments must run in parallel in different compute environment, each requiring a download of the dataset from the data lake, which is an expensive and time-consuming process. Proximity of the dataset to the compute (especially in the instance of a hybrid cloud) is not guaranteed. In addition, other team members that run their own experiments and require the same dataset will go through the same arduous process. Figure 1 illustrates the issue of dataset ‘remoteness’ from the desired compute requiring the dataset. Beyond the obvious slow data access, challenges include difficulties tracking dataset versions, dataset sharing, collaboration, and reproducibility.

2.1 Customer Requirements

Customer requirements can vary in order to achieve high-performance ML runs while efficiently using the resources; for example, customers might require the ability to:

- Have fast access to datasets from each compute executing the training model, without incurring expensive downloads and data access complexities.
- Use any compute, in the cloud, or on-premises GPUs or CPUs without being influenced by the locality of the datasets.
- Increase efficiency and productivity, which translates to running multiple training experiments in parallel, with different compute on the same dataset, without unnecessary delays and data latency.
- Minimize the compute instance costs.
- Achieve reproducibility, which implies having tools to keep records of the datasets, their lineage, versions, and other metadata details.
- Share and collaborate, which means that any authorized member of the team can access the datasets and run experiments.
- For the specific implementation of the dataset caching with NetApp ONTAP® data management software, customers require the ability to:
  - Configure and set the NFS that has the best proximity to the compute.
  - Determine which dataset and version to cache.
  - Monitor the total memory committed to cached datasets, and how much available NFS storage as available for additional cache commits (i.e. cache management).
  - Age out of datasets in the cache if they have not been used in certain time (the default is one day; other configuration options are available).

3 Solution Overview

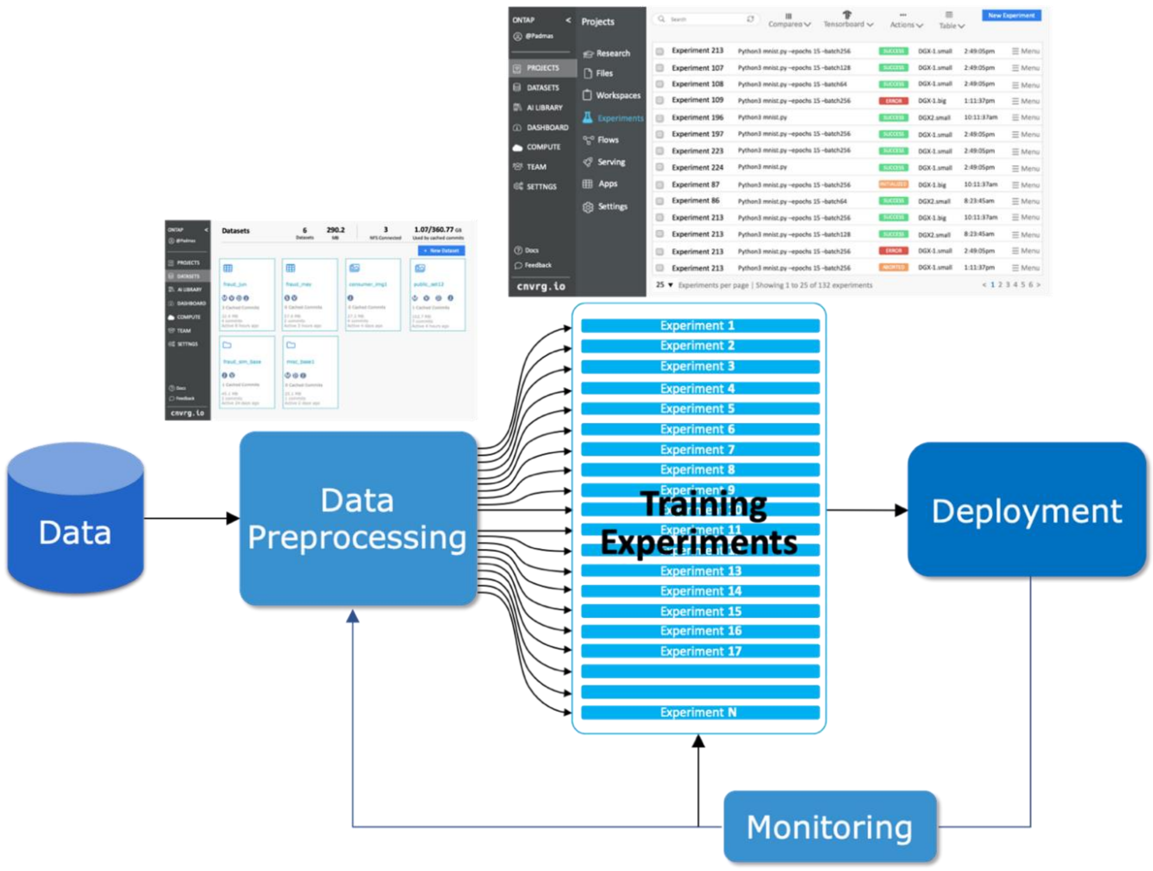
This section reviews a conventional data science pipeline and its drawbacks. It also presents the architecture of the proposed dataset caching solution.

3.1 Conventional Data Science Pipeline and Drawbacks

A typical sequence of ML models’ development and deployment involves iterative steps that include ingesting data, data preprocessing (creating multiple versions of the datasets), running multiple experiments (Hyper Parameter Optimization, different models, and so on), deployment and monitoring. cnvrg.io developed a comprehensive platform to automate all tasks from research to deployment. A small sample of dashboard screenshots pertaining to the pipeline steps are shown in Figure 2. It is very common to have multiple datasets in play (from public repositories and the private organization data). In addition, each dataset is likely to have multiple versions (as a result of certain dataset cleanup or feature engineering). A dashboard that provides a dataset hub and a version hub is needed to make sure collaboration and consistency is provided to the team. One example is shown in Figure 3. The next step in the pipeline is training, which requires multiple parallel instances of training models, each associated with a dataset and a certain compute. The binding of a dataset to a certain experiment with a certain compute is the challenge because it is possible that some experiments will be performed by GPU instances in Amazon Web Services (AWS), while other experiments by DGX-1s or DGX-2s on-premises.

And other experiments will be executed in the CPU servers in GCP, while the dataset location is not in a reasonable proximity to the compute performing the training. A reasonable proximity will be assumed to be at least low latency full 10GbE or more connectivity from the dataset storage to the compute. A common solution is for the data scientist to download the dataset to the compute performing the training and execute the experiment. However, there are several flaws with this approach.

Figure 2) ML and data science pipeline.

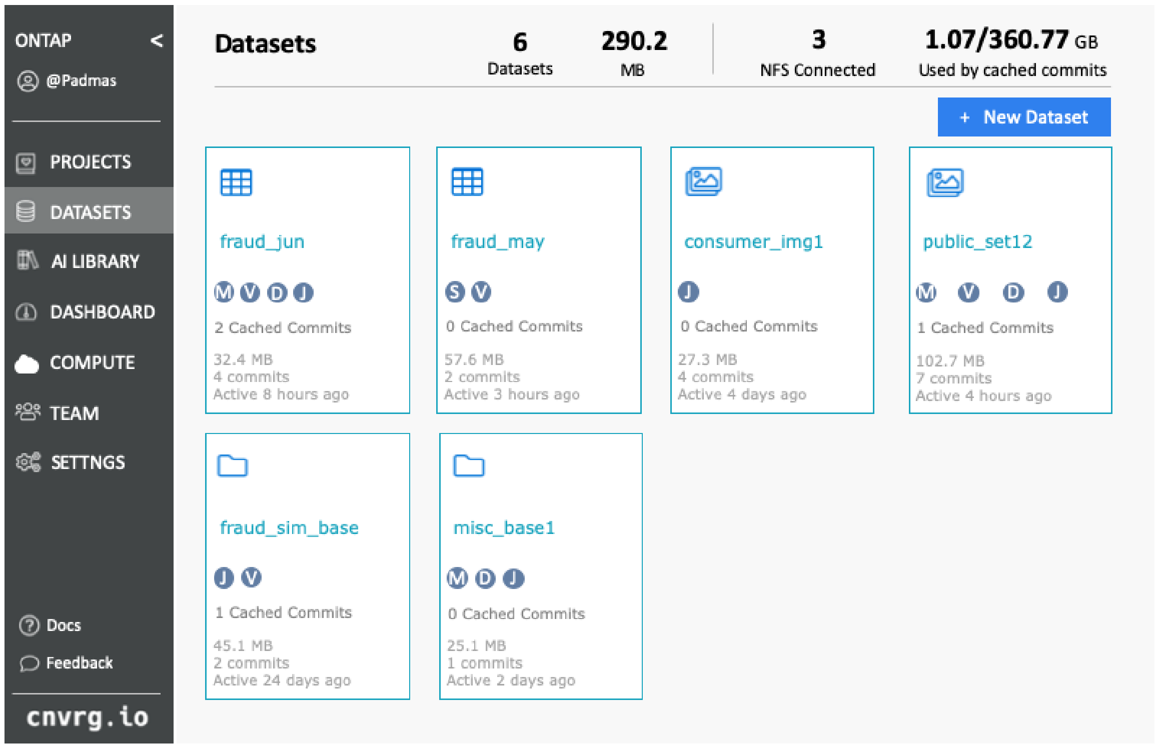


Experiment 213	Python3 mnist.py --epochs 15 --batch256	SUCCESS	DGX-1.small	2:49:05pm	Menu
Experiment 107	Python3 mnist.py --epochs 15 --batch128	SUCCESS	DGX-1.small	2:49:05pm	Menu
Experiment 108	Python3 mnist.py --epochs 15 --batch64	SUCCESS	DGX-1.small	2:49:05pm	Menu
Experiment 109	Python3 mnist.py --epochs 15 --batch256	ERROR	DGX-1.big	1:11:37pm	Menu
Experiment 196	Python3 mnist.py	SUCCESS	DGX2.small	10:11:37am	Menu
Experiment 197	Python3 mnist.py --epochs 15 --batch256	SUCCESS	DGX-1.small	2:49:05pm	Menu
Experiment 223	Python3 mnist.py --epochs 15 --batch256	SUCCESS	DGX-1.small	2:49:05pm	Menu
Experiment 224	Python3 mnist.py	SUCCESS	DGX-1.small	2:49:05pm	Menu
Experiment 87	Python3 mnist.py --epochs 15 --batch256	INITIALIZED	DGX-1.big	10:11:37am	Menu
Experiment 86	Python3 mnist.py --epochs 15 --batch64	SUCCESS	DGX2.small	8:23:45am	Menu
Experiment 213	Python3 mnist.py --epochs 15 --batch256	SUCCESS	DGX-1.big	10:11:37am	Menu
Experiment 213	Python3 mnist.py --epochs 15 --batch128	SUCCESS	DGX2.small	8:23:45am	Menu
Experiment 213	Python3 mnist.py --epochs 15 --batch256	ERROR	DGX-1.small	2:49:05pm	Menu
Experiment 213	Python3 mnist.py --epochs 15 --batch256	ABORTED	DGX-1.small	1:11:37pm	Menu

25 Experiments per page | Showing 1 to 25 of 132 experiments

- When the data scientist downloads the dataset to the compute, there are no guarantees that the compute integrated storage is high performance (for example, the ONTAP AFF A800 NVMe solution).
- The downloaded dataset will reside in one compute node, however, when distributed models are executed over multiple nodes, the storage will become a bottleneck (unlike the NetApp ONTAP high-performance distributed storage).
- The next iteration of the training experiment can be performed in a different compute (due to queue conflicts or priorities), again creating significant network ‘distance’ from the dataset to the compute.
- Other team members executing training experiments on the same compute cluster cannot share this dataset, each will perform the (expensive) download of the dataset from the arbitrary location.
- If other datasets or versions of the same dataset are needed for the subsequent training jobs, the data scientists will again need to incur the (expensive) download of the dataset to the compute performing the training.

Figure 3) Datasets and versions hub.



Datasets

6 Datasets | 290.2 MB | 3 NFS Connected | 1.07/360.77 GB Used by cached commits

fraud_jun
M V D J
2 Cached Commits
32.4 MB
4 commits
Active 8 hours ago

fraud_may
S V
0 Cached Commits
57.6 MB
2 commits
Active 3 hours ago

consumer_img1
J
0 Cached Commits
27.3 MB
4 commits
Active 4 days ago

public_set12
M V D J
1 Cached Commits
102.7 MB
7 commits
Active 4 hours ago

fraud_sim_base
J V
1 Cached Commits
45.1 MB
2 commits
Active 24 days ago

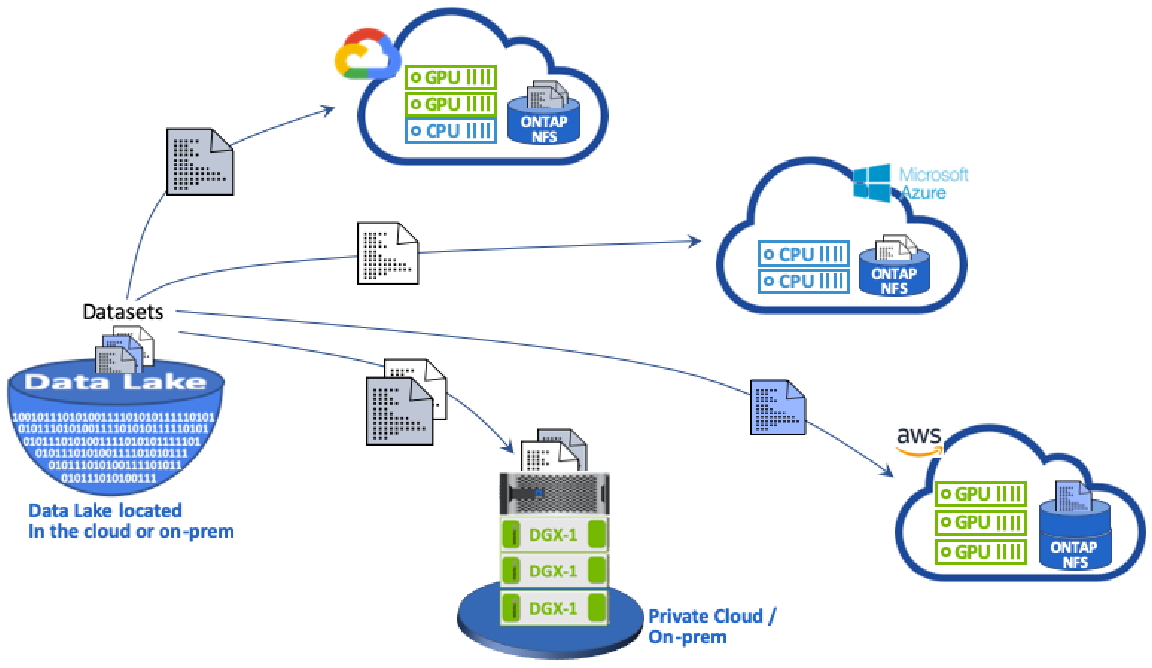
misc_base1
M D J
0 Cached Commits
25.1 MB
1 commits
Active 2 days ago

NetApp and cnvrg.io created a new dataset caching solution that eliminates these hurdles. The solution creates accelerated execution of the ML pipeline by caching hot datasets on the ONTAP high-performance storage system. With ONTAP NFS, the datasets are cached once (and only once) in a data fabric powered by NetApp (such as AFF A800), which is collocated with the compute. As the NetApp ONTAP NFS high-speed storage can serve multiple ML compute nodes, the performance of the training models is optimized, bringing cost saving, work force productivity, and operational efficiency to the organization.

3.2 Solution Architecture

The solution from NetApp and cnvrg.io provides dataset caching, as shown in Figure 4. Dataset caching allows data scientists to pick a desired dataset or dataset version and move it to the ONTAP NFS cache, which resides in the proximity of the ML compute cluster. The data scientist now can run multiple experiments without incurring delays or downloads. In addition, all collaborating engineers can use the same dataset with the attached compute cluster (freedom to pick any node) without additional downloads from the data lake. The data scientists are offered a dashboard that tracks, monitors all datasets and versions, and provides a view of which datasets were cached. cnvrg.io platform will auto-detect aged datasets that have not been used for a certain time, and will evict them from the cache, hence maintaining free NFS cache space for more frequently used datasets. It is important to note that dataset caching with ONTAP works in the cloud and on-premises, providing maximum flexibility to the customers.

Figure 4) NetApp dataset caching.



4 Concepts and Components

4.1 Machine Learning

ML is rapidly becoming essential to all businesses and organizations around the world. However, this means that IT and DevOps teams are now facing the challenge of standardizing ML workloads and provisioning cloud, on-premises, and hybrid compute resources that support the dynamic and intensive workflows that ML jobs and pipelines require.

4.2 Container-Based Machine Learning and Kubernetes

Containers are isolated user-space instances that run on top of a shared host operating system kernel. The adoption of containers is rapidly increasing. Containers offer many of the same application sandboxing benefits that virtual machines (VMs) offer. However, because the hypervisor and guest operating system layers that VMs rely on have been eliminated, containers are far more lightweight.

Containers also allow the efficient packaging of application dependencies, run times, and so on, directly with an application. The most commonly used container packaging format is the Docker container. An application that has been containerized in the Docker container format can be executed on any machine that can run Docker containers. This is true even if the application's dependencies are not present on the machine because all dependencies are packaged in the container itself. For more information, visit the Docker website.

Kubernetes, the popular container orchestrator, allows data scientists to launch flexible, container-based jobs and pipelines, as well as enabling infrastructure teams to manage and monitor ML workloads in a single managed and cloud-native environment. For more information, visit the Kubernetes website.

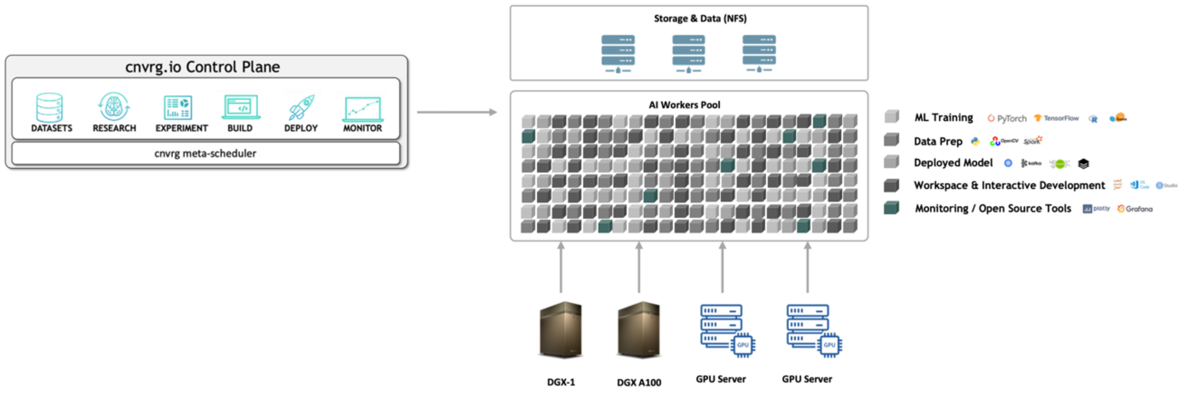
4.3 cnvrg.io

cnvrg.io is an AI operating system that transforms the way enterprises manage, scale, and accelerate AI and data science development from research to production. The code-first platform is built by data scientists, for data scientists, and offers flexibility to run on-premises or in the cloud. With model management, MLOps and continual ML solutions, cnvrg.io brings top-of-the-line technology to data science teams so they can spend less time on DevOps and focus on the real magic—algorithms. Since using cnvrg.io, teams across industries have gotten more models to production resulting in increased business value.

cnvrg.io Meta-Scheduler

cnvrg.io has a unique architecture that allows IT and engineers to attach different compute resources to the same control plane and have cnvrg.io manage ML jobs across all resources. This means that IT can attach multiple on-premises Kubernetes clusters, VM servers, and cloud accounts and run ML workloads on all resources, as shown in Figure 5.

Figure 5) cnvrg meta-scheduler.



cnvrg.io Data Caching

cnvrg.io allows data scientists to define hot/cold dataset versions with its data-caching technology. By default, datasets are stored in a centralized object storage database. Then, data scientists can cache a specific data version on the selected compute resource to save time on download and therefor increase ML development and productivity. Datasets that are cached and are not in use for a few days are automatically cleared from the selected NFS. Caching and clearing cache can be done in a single click, no coding/IT/DevOps work is required.

cnvrg.io Flows and ML Pipelines

cnvrg.io Flows is a tool for building production ML pipelines. Each component in a flow is a script/code running on a selected compute with a base docker image. This design enables data scientists and engineers to build a single pipeline that can run both on-premises and in the cloud. cnvrg.io makes sure data, parameters, and artifacts are moving between the different components. In addition, each flow is monitored and tracked for 100% reproducible data science.

cnvrg.io CORE

cnvrg.io CORE is a free platform for the data science community to help data scientists focus more on data science and less on DevOps. CORE's flexible infrastructure gives data scientists the control to use any language, AI framework, and compute environment whether on premise or cloud compute so they can do what they do best, build algorithms. cnvrg.io CORE can be easily installed in a single command on any Kubernetes cluster.

4.4 NetApp ONTAP AI

ONTAP AI is a data center reference architecture for ML and deep learning (DL) workloads by using NetApp AFF storage systems and NVIDIA DGX Systems with Tesla V100 GPUs. ONTAP AI is based on the industry-standard NFS file protocol over 100Gb Ethernet, providing customers with a high-performance ML/DL infrastructure that uses standard data center technologies to reduce implementation and administration overhead. Using standardized network and protocols enables ONTAP AI to integrate into hybrid cloud environments while maintaining operational consistency and simplicity. As a prevalidated infrastructure solution, ONTAP AI reduces deployment time and risk and reduces administration overhead significantly, allowing customers to realize faster time to value.

4.5 NVIDIA DeepOps

DeepOps is an open source project from NVIDIA that, by using Ansible, automates the deployment of GPU server clusters according to best practices. DeepOps is modular and can be used for various deployment tasks. For this document and the validation exercise that it describes, DeepOps is used to deploy a Kubernetes cluster that consists of GPU server worker nodes. For more information, visit the DeepOps website.

4.6 NetApp Trident

Trident is an open source storage orchestrator developed and maintained by NetApp that greatly simplifies the creation, management, and consumption of persistent storage for Kubernetes workloads. Trident itself a Kubernetes-native application—it runs directly within a Kubernetes cluster. With Trident, Kubernetes users (developers, data scientists, Kubernetes administrators, and so on) can create, manage, and interact with persistent storage volumes in the standard Kubernetes format that they are already familiar with. At the same time, they can take advantage of NetApp advanced data management capabilities and a data fabric that is powered by NetApp technology. Trident abstracts away the complexities of persistent storage and makes it simple to consume. For more information, visit the Trident website.

4.7 NetApp StorageGRID

NetApp StorageGRID is a software-defined object storage platform designed to meet these needs by providing simple, cloud-like storage that users can access using the S3 protocol. StorageGRID is a scale-out system designed to support multiple nodes across internet-connected sites, regardless of distance. With the intelligent policy engine of StorageGRID, users can choose erasure-coding objects across sites for geo-resiliency or object replication between remote sites to minimize WAN access latency. StorageGrid provides an excellent private-cloud primary object storage data lake in this solution.

4.8 NetApp Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP data management software delivers control, protection, and efficiency to user data with the flexibility of public cloud providers including AWS, Google Cloud Platform, and Microsoft Azure. Cloud Volumes ONTAP is cloud-native data management software built on the NetApp ONTAP storage software, providing users with a superior universal storage platform that addresses their cloud data needs. Having the same storage software in the cloud and on-premises brings users the value of a data fabric without having to train IT staff in all-new methods to manage data.

For customers that are interested in hybrid cloud deployment models, Cloud Volumes ONTAP can provide the same capabilities and class-leading performance in most public clouds to provide a consistent and seamless user experience in any environment.

5 Hardware and Software Requirements

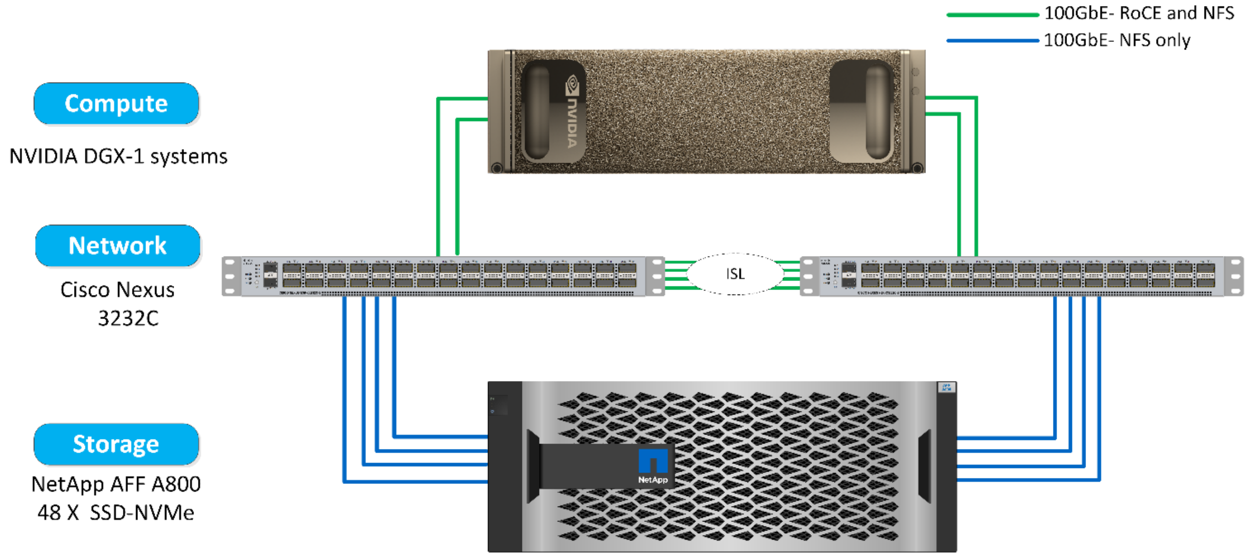
This section covers the technology requirements for the ONTAP AI solution.

5.1 Hardware Requirements

While hardware requirements depend on specific customer workloads, ONTAP AI can be deployed at any scale for data engineering, model training, and production inferencing from a single GPU up to rack-scale configurations for large-scale ML/DL operations. For more information about ONTAP AI, see the ONTAP AI website.

This solution was validated using a DGX-1 system for compute, a NetApp AFF A800 storage system, and Cisco Nexus 3232C for network connectivity. The AFF A800 used in this validation can support as many as 10 DGX-1 systems for most ML/DL workloads. Figure 6 shows the ONTAP AI topology used for model training in this validation.

Figure 6) ONTAP AI topology.



To extend this solution to a public cloud, Cloud Volumes ONTAP can be deployed alongside cloud GPU compute resources and integrated into a hybrid cloud data fabric that enables customers to use whatever resources are appropriate for any given workload.

5.2 Software Requirements

Table 1 shows the specific software versions used in this solution validation.

Table 1) Validation environment — software versions.

Component	Version
Ubuntu	18.04.4 LTS
NVIDIA DGX OS	4.4.0
NVIDIA DeepOps	20.02.1
Kubernetes	1.15
Helm	3.1.0

Component	Version
cnvrg.io	3.0.0
NetApp ONTAP	9.6P4

For this solution validation, Kubernetes was deployed as a single-node cluster on the DGX-1 system. For large scale deployments, independent Kubernetes master nodes should be deployed to provide high availability of management services as well as reserve valuable DGX resources for ML/DL workloads.

6 Solution Deployment and Validation Details

6.1 ONTAP AI Deployment

Deployment of ONTAP AI requires the installation and configuration of networking, compute, and storage hardware. Specific instructions for deployment of the ONTAP AI infrastructure are beyond the scope of this document. For detailed deployment information, see NVA-1121-DEPLOY: NetApp ONTAP AI, Powered by NVIDIA.

For this solution validation, a single volume was created and mounted to the DGX-1 system. That mount point was then mounted to the containers to make data accessible for training. For large-scale deployments, NetApp Trident automates the creation and mounting of volumes to eliminate administrative overhead and enable end-user management of resources.

6.2 Kubernetes Deployment

To deploy and configure your Kubernetes cluster with NVIDIA DeepOps, perform the following tasks from a deployment jump host:

1. Download NVIDIA DeepOps by following the instructions on the Getting Started page on the NVIDIA DeepOps GitHub site.
2. Deploy Kubernetes in your cluster by following the instructions on the Kubernetes Deployment Guide page on the NVIDIA DeepOps GitHub site.

   Note: For the DeepOps Kubernetes deployment to work, the same user must exist on all Kubernetes master and worker nodes.

If the deployment fails, change the value of `kubectl_localhost` to false in `deepops/config/group_vars/k8s-cluster.yml` and repeat step 2. The `Copy kubectl binary to ansible host` task, which executes only when the value of `kubectl_localhost` is true, relies on the fetch Ansible module, which has known memory usage issues. These memory usage issues can sometimes cause the task to fail. If the task fails because of a memory issue, then the remainder of the deployment operation does not complete successfully.

If the deployment completes successfully after you have changed the value of `kubectl_localhost` to `false`, then you must manually copy the `kubectl binary` from a Kubernetes master node to the deployment jump host. You can find the location of the `kubectl binary` on a specific master node by running the `which kubectl` command directly on that node.

6.3 cnvrg.io Deployment

Deploy cnvrg CORE Using Helm

Helm is the easiest way to quickly deploy cnvrg using any cluster, on-premises, Minikube, or on any cloud cluster (such as AKS, EKS, and GKE). This section describes how cnvrg was installed on an on-premises (DGX-1) instance with Kubernetes installed.

Prerequisites

Before you can complete the installation, you must install and prepare the following dependencies on your local machine:

- Kubectl
- Helm 3.x
- Kubernetes cluster 1.15+

Deploy Using Helm

1. To download the most updated cnvrg helm charts, run the following command:

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

2. Before you deploy cnvrg, you need the external IP address of the cluster and the name of the node on which you will deploy cnvrg. To deploy cnvrg on an on-premises Kubernetes cluster, run the following command:

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

3. Run the `helm install` command. All the services and systems will automatically install on your cluster. The process can take up to 15 minutes.
4. The `helm install` command can take up to 10 minutes. When the deployment completes, go to the URL of your newly deployed cnvrg or add the new cluster as a resource inside your organization. The `helm` command informs you of the correct URL.

```
🚀 Thank you for installing cnvrg.io!
Your installation of cnvrg.io is now available, and can be reached via: http://app.mydomain.com
Talk to our team via email at hi@cnvrg.io
```

5. When the status of all the containers is running or complete, cnvrg will have been successfully deployed. It should look similar to the following example output:

```
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
```

Computer Vision Model Training with ResNet50 and the Chest X-ray Dataset

cnvrg.io AI OS was deployed on a Kubernetes setup on the NetApp ONTAP AI architecture, powered by the NVIDIA DGX System. For validating purposes, we used the NIH Chest X-ray dataset consisting of de-identified images of chest x-rays. The images are in PNG format. The data is provided by the NIH Clinical Center and is available through the NIH download site. We used a sample of the data of 250GB with 627,615 images across 15 classes.

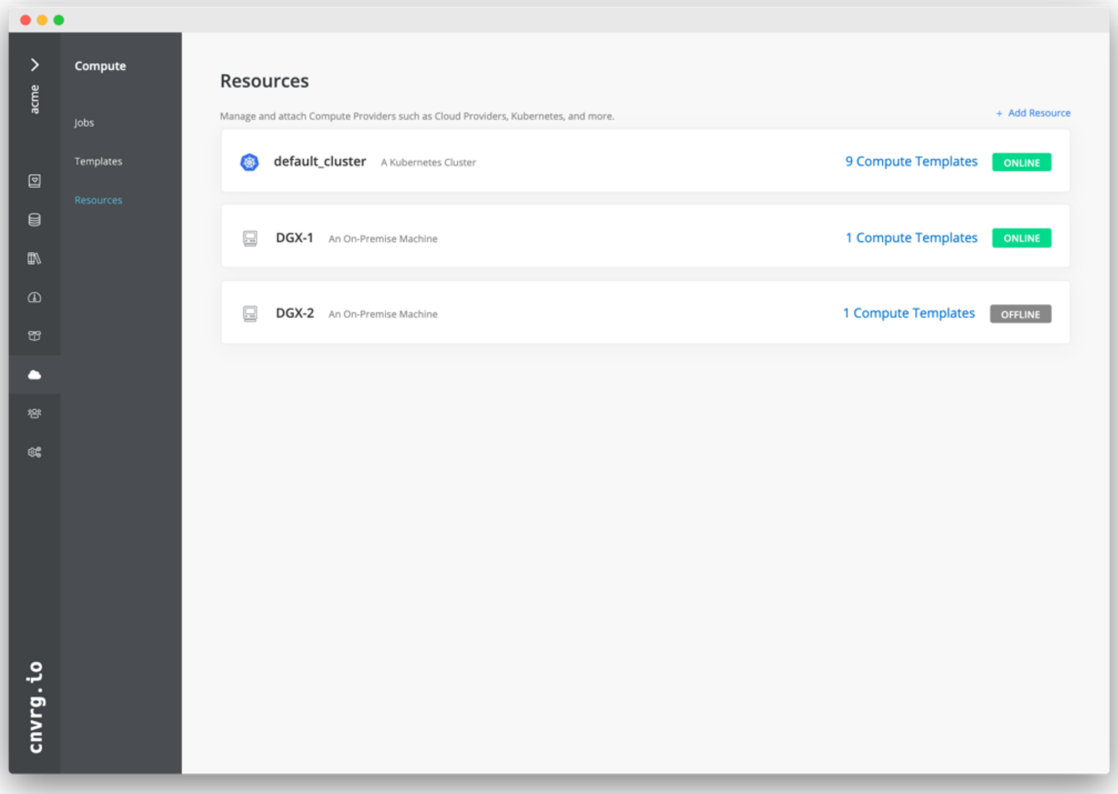
To train the image classification model, we used a transfer-learning approach using the ResNet50 that was introduced in the paper "Deep Residual Learning for Image Recognition," by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Source code can be found in cnvrg.io open source AI libraries.

The dataset was uploaded to the cnvrg platform and was cached on an NFS export from the NetApp AFF A800 storage system.

Setting up the Compute Resources

The cnvrg architecture and meta-scheduling capability allow engineers and IT professionals to attach different compute resources to a single platform. In our setup, we used the same cluster cnvrg that was deployed for running the deep-learning workloads. If needed to attach additional clusters, it can be done through the GUI, as shown in Figure 7.

Figure 7) Resource management in cnvrg.io user interface.



Manage and attach Compute Providers such as Cloud Providers, Kubernetes, and more.

default_cluster A Kubernetes Cluster 9 Compute Templates ONLINE

DGX-1 An On-Premise Machine 1 Compute Templates ONLINE

DGX-2 An On-Premise Machine 1 Compute Templates OFFLINE

Loading Data

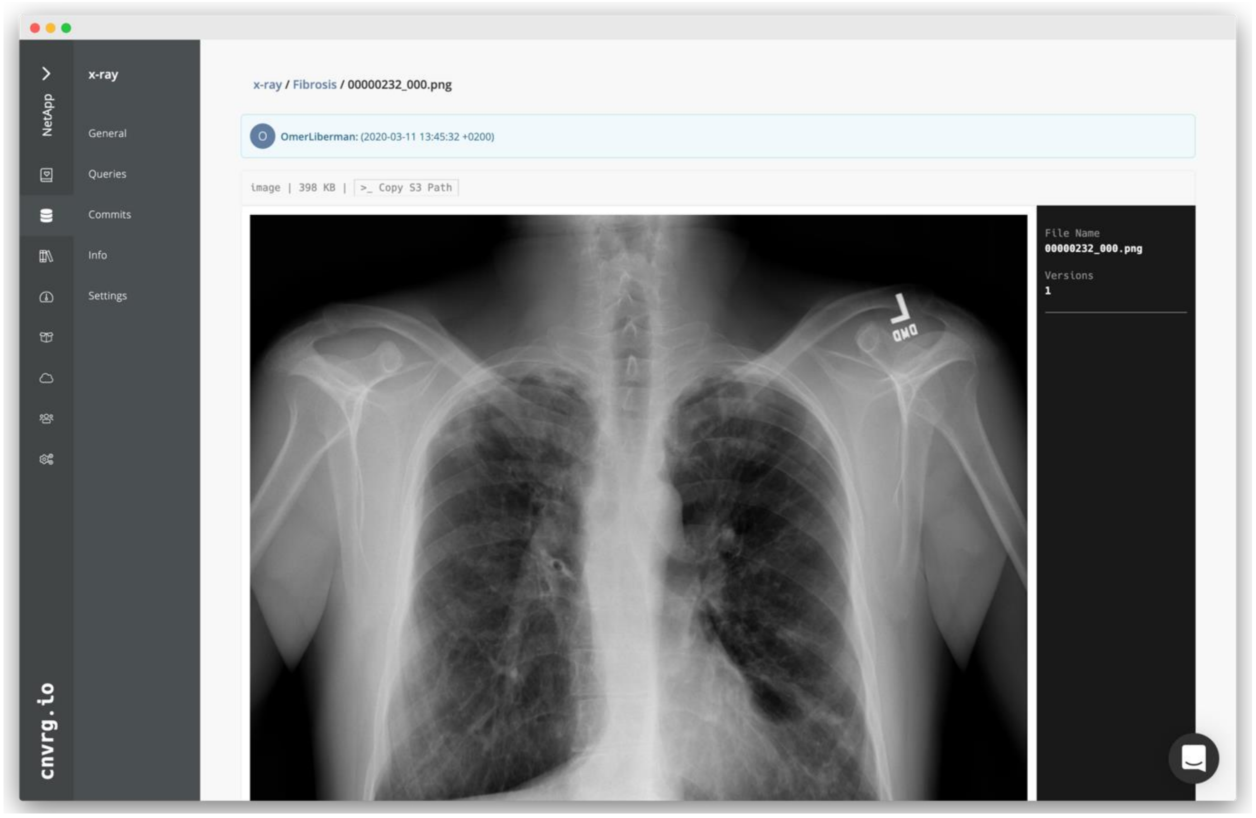
There are multiple ways data can be uploaded to the cnvrg platform. You can do that through the GUI and the cnvrg CLI. For large datasets, it is recommended that you use the CLI because it is a strong, scalable, and reliable tool that can handle large amount files.

To upload the data, we used the following steps:

1. Downloaded the cnvrg CLI.
2. cd x-ray directory
3. cnvrg data init (initialize the dataset in the platform)
4. cnvrg data sync (upload all contents of the directory to the central data-lake)

After the data is uploaded to the central object store (StorageGRID/S3/others), you can browse it through the GUI. Figure 8 shows a loaded chest X-ray fibrosis image PNG file. In addition, cnvrg versions the data so that any model you build can be reproduced down to the data version.

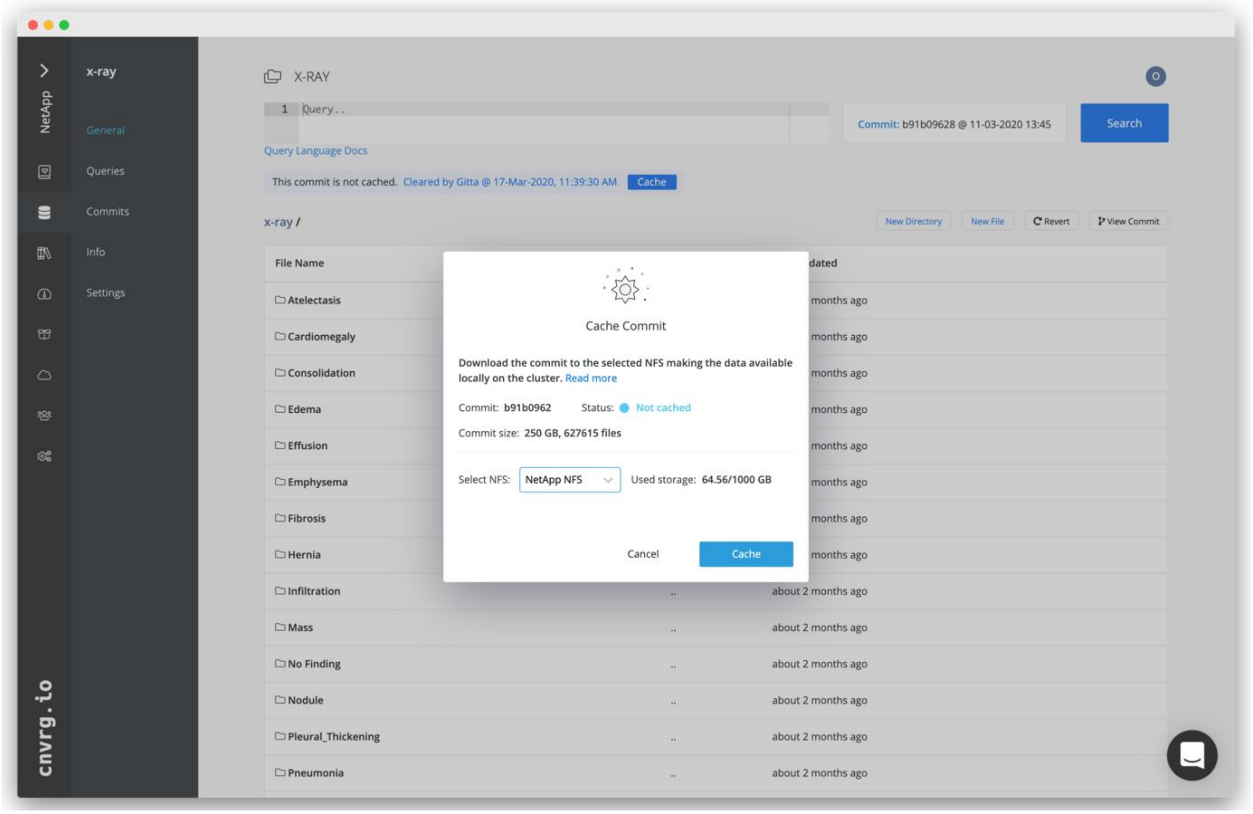
Figure 8) NIH chest X-ray dataset loaded in cnvrg.io GUI.


NetApp
x-ray
General
Queries
Commits
Info
Settings
cnvrg.io
x-ray / Fibrosis / 00000232_000.png
OmerLiberman: (2020-03-11 13:45:32 +0200)
image | 398 KB | >_ Copy S3 Path
File Name
00000232_000.png
Versions
1

Caching Data

After data was initially uploaded to the central data lake (object-store), in order to make training faster and avoid downloading 600k+ files for each model training and experiment, we used the data-caching feature.

Figure 9) Example of dataset caching in cnvrg.io.



Commit: b91b09628 @ 11-03-2020 13:45

This commit is not cached. Cleared by Gitta @ 17-Mar-2020, 11:39:30 AM Cache

Cache Commit

Download the commit to the selected NFS making the data available locally on the cluster. Read more

Commit: b91b0962 Status: Not cached

Commit size: 250 GB, 627615 files

Select NFS: NetApp NFS Used storage: 64.56/1000 GB

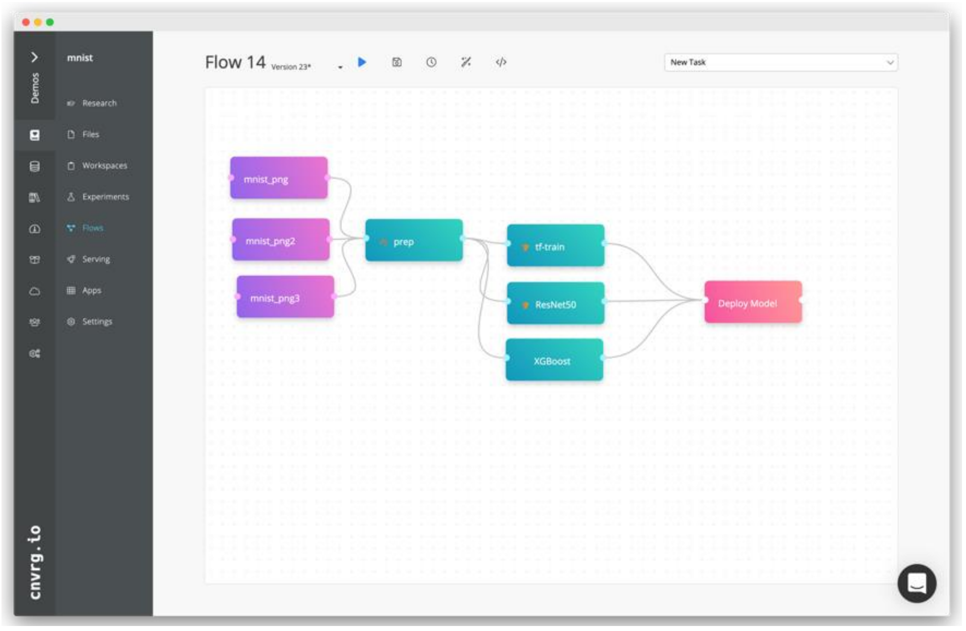
File Name		dated
Atelectasis		months ago
Cardiomegaly		months ago
Consolidation		months ago
Edema		months ago
Effusion		months ago
Emphysema		months ago
Fibrosis		months ago
Hernia		months ago
Infiltration	..	about 2 months ago
Mass	..	about 2 months ago
No Finding	..	about 2 months ago
Nodule	..	about 2 months ago
Pleural_Thickening	..	about 2 months ago
Pneumonia	..	about 2 months ago

After users click Cache (Figure 9), cnvrg downloads the data (in its specific commit) from the remote object store and caches it on the ONTAP NFS volume. After it completes, the data is available for instant training. In addition, if the data is not used (for example, for model training or exploration) for a few days, cnvrg automatically clears the cache.

Building an ML Pipeline with Cached Data

cnvrg Flows allows you to easily build production ML pipelines. Flows are flexible, can work for any kind of ML use case, and can be created through the GUI or code. Each component in a flow can run on a different compute resource with a different Docker image, which makes it possible to build hybrid cloud and optimized ML pipelines (Figure 10).

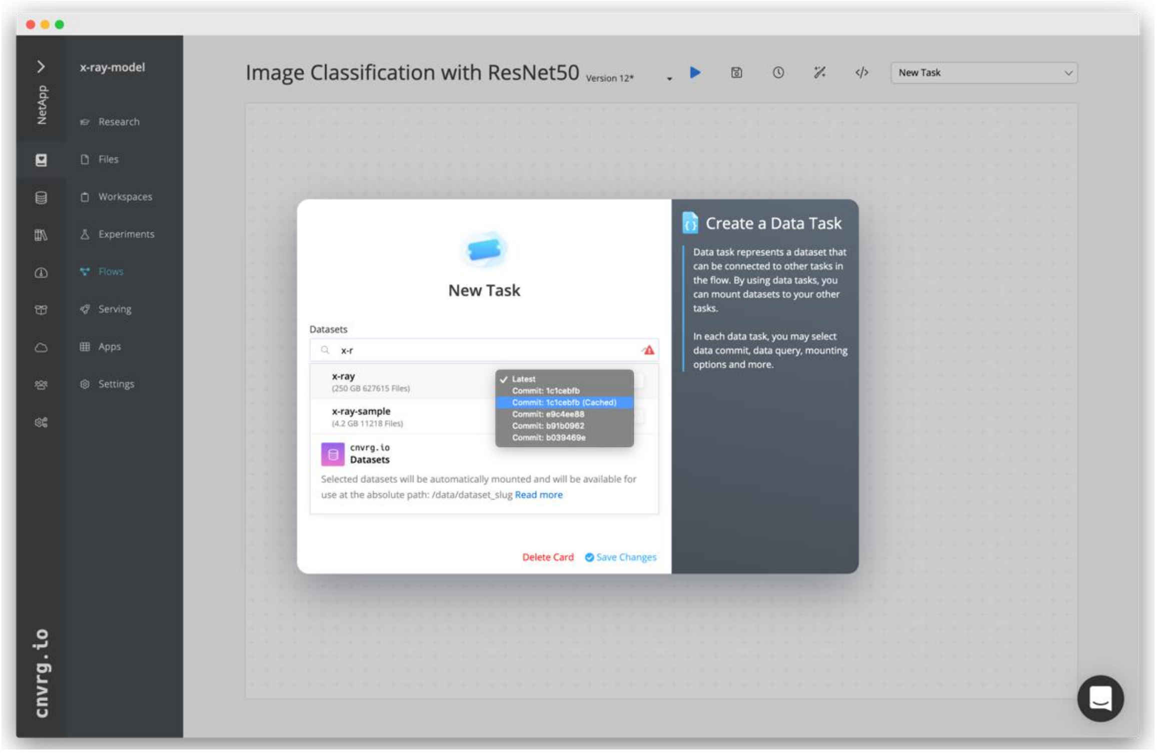
Figure 10) Example of cnvrg Flows.



Building the Chest X-ray Flow: Setting Data

We added our dataset to the newly created Flow. When adding the dataset, you can select the specific version (commit) and indicate whether or not you want the cached version. In this example, we selected the cached commit, as shown in Figure 11.

Figure 11) Adding dataset to cnvrg Flow by selecting a specific commit.



Selected datasets will be automatically mounted and will be available for use at the absolute path: /data/dataset_slug Read more

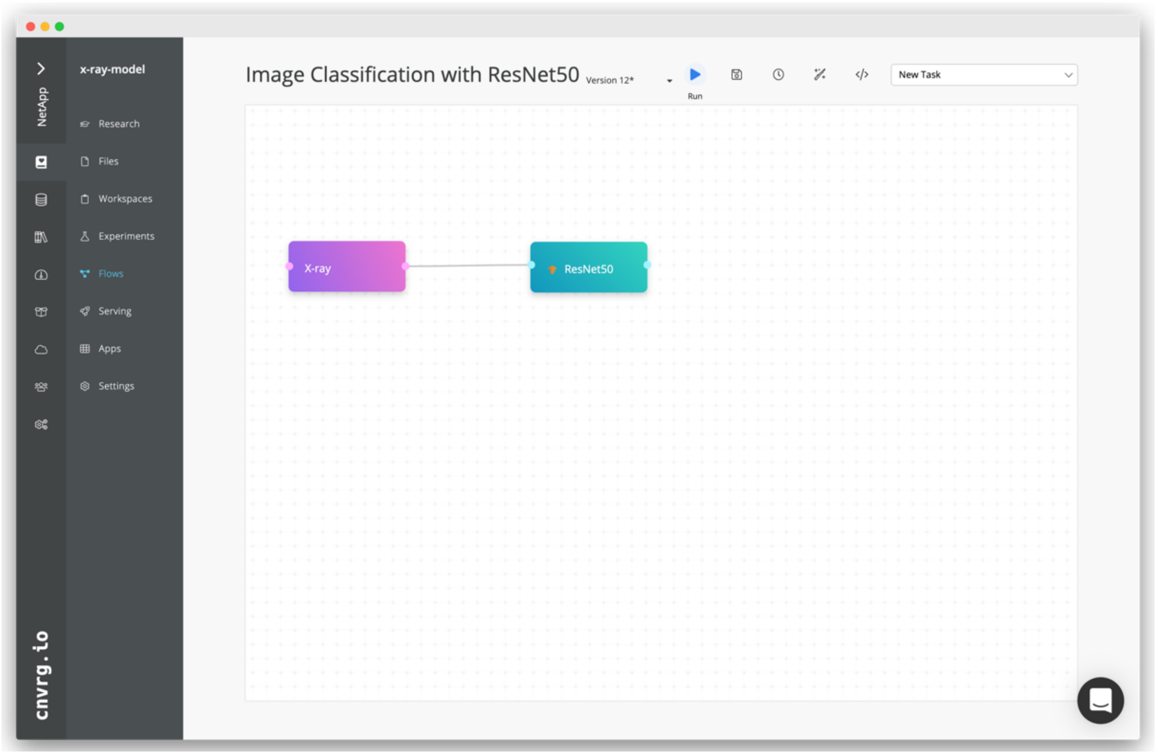
Data task represents a dataset that can be connected to other tasks in the flow. By using data tasks, you can mount datasets to your other tasks.

In each data task, you may select data commit, data query, mounting options and more.

Building the Chest X-ray Flow: Setting Training Model: ResNet50

In the pipeline, you can add any kind of custom code you want. In cnvrg there's also the AI library, a reusable ML components collection. In the AI library, there are algorithms, scripts, data sources and other solutions that can be leveraged in any ML or deep learning flow. In this example, we selected the prebuilt ResNet50 module. We used default params such as: batch_size:128, epochs:10, and more (can be viewed in the AI Library docs). Figure 12 shows the created Flow with X-ray dataset connected to ResNet50.

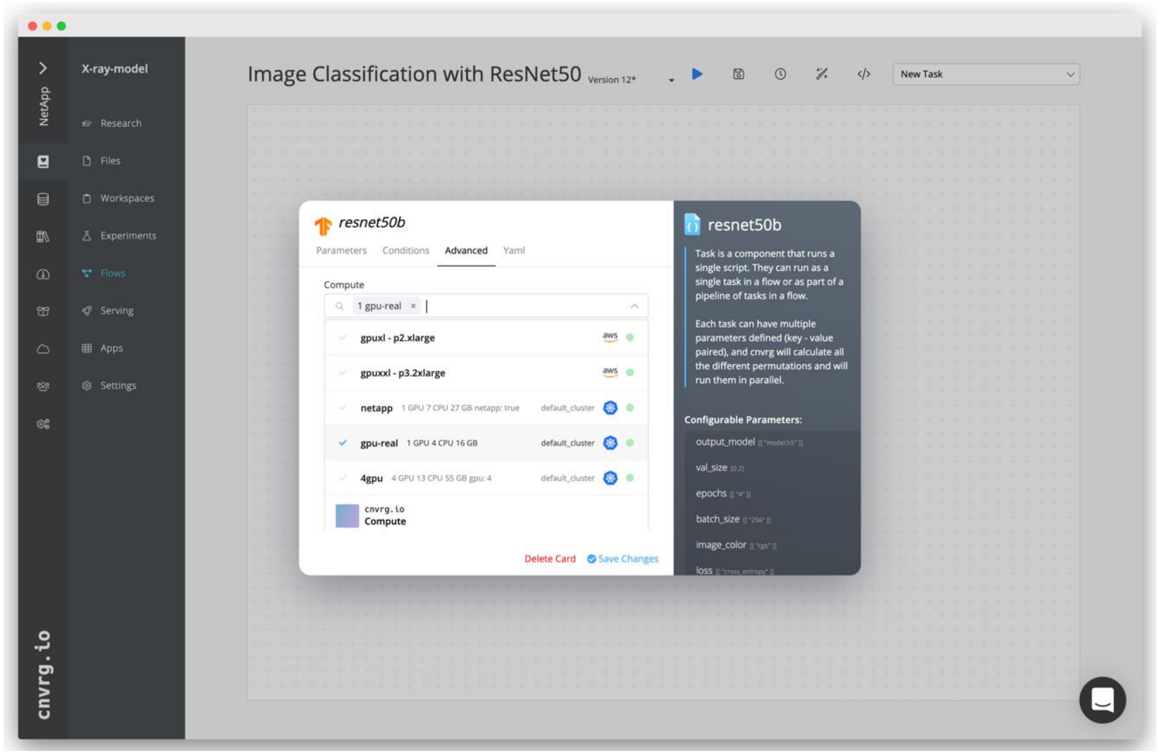
Figure 12) cnvrg Flow with ResNet50 image classification.



Defining the Compute Resource for ResNet50

Each algorithm or component in cnvrg Flows can run on a different compute, with a different Docker image. In our setup, we wanted to run the training algorithm on the NVIDIA DGX Systems with the NetApp ONTAP AI architecture. In Figure 13, we selected `gpu-real`, which is a compute template and specification for our on-premises cluster. We can also create a queue of templates and select multiple templates. This way, if the `gpu-real` resource cannot be allocated (for example, when other data scientists might use), then we can enable automatic cloud-bursting easily by adding a cloud provider template. Figure 13 shows using gpu-real as compute node for ResNet50.

Figure 13) Connecting ResNet50 Flow to a compute resource.



resnet50b

Compute

1 gpu-real

✓	gpuxl - p2.xlarge		aws
✓	gpuxxl - p3.2xlarge		aws
✓	netapp	1 GPU 7 CPU 27 GB netapp: true	default_cluster
✓	gpu-real	1 GPU 4 CPU 16 GB	default_cluster
✓	4gpu	4 GPU 13 CPU 55 GB gpu: 4	default_cluster

cnvrg.io Compute

resnet50b

Task is a component that runs a single script. They can run as a single task in a flow or as part of a pipeline of tasks in a flow.

Each task can have multiple parameters defined (key - value paired), and cnvrg will calculate all the different permutations and will run them in parallel.

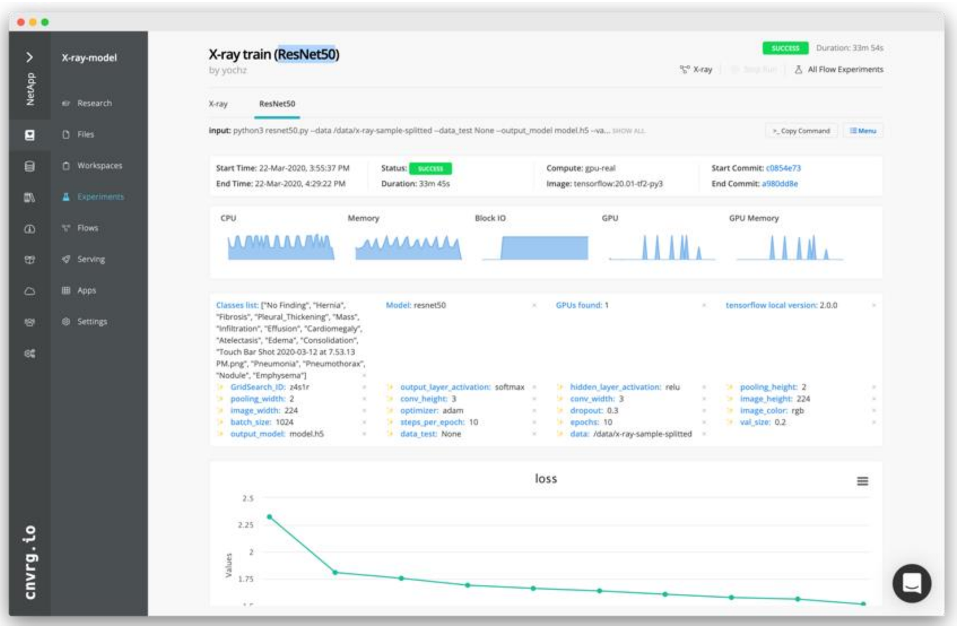
Configurable Parameters:

output_model [["model.h5"]]
val_size [0.2]
epochs [["4"]]
batch_size [["256"]]
image_color [["rgb"]]
loss [["cross_entropy"]]

Tracking & Monitoring Results

After a Flow is executed, cnvrg triggers its tracking and monitoring engine. Each run of a Flow is automatically documented and updated in real-time. Hyperparameters, metrics, resource usage (GPU utilization, and more), code version, artifacts, logs, and more are automatically available in the Experiments section, as shown in Figure 14 and Figure 15.

Figure 14) Experiments section in cnvrg.io platform (1 of 2).



X-ray train (ResNet50)
by yochz

Duration: 33m 54s

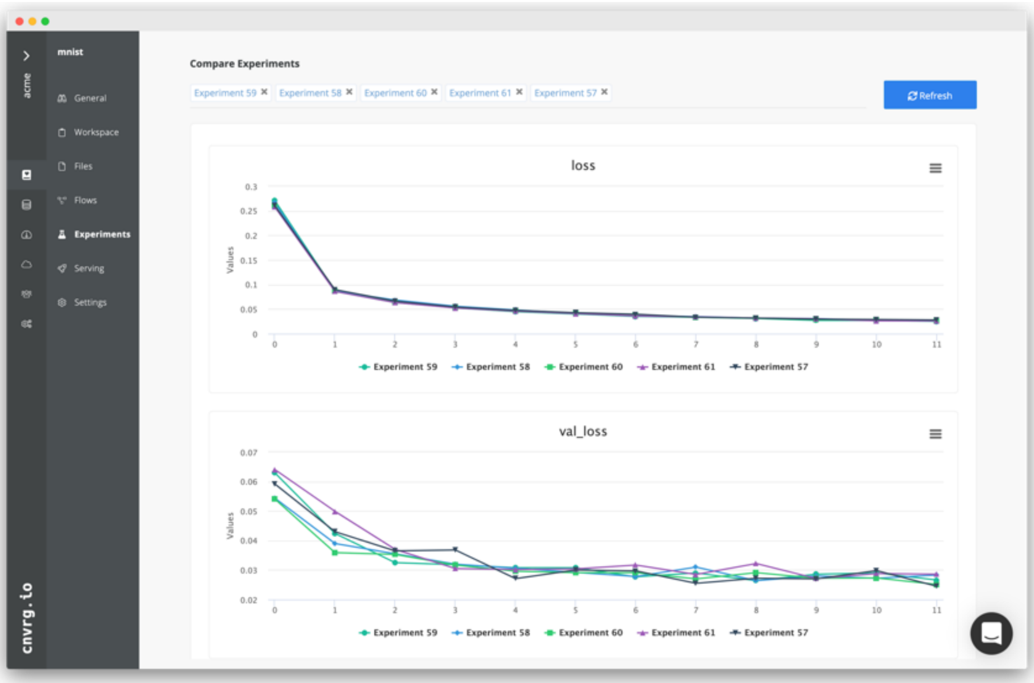
input: python3 resnet50.py --data /data/x-ray-sample-splitted --data_test None --output_model model.h5 --va... SHOW ALL

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

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	output_layer_activation: softmax	hidden_layer_activation: relu	pooling_height: 2
pooling_width: 2	conv_height: 3	conv_width: 3	image_height: 224
image_width: 224	optimizer: adam	dropout: 0.3	image_color: rgb
batch_size: 1024	steps_per_epoch: 10	epochs: 10	val_size: 0.2
output_model: model.h5	data_test: None	data: /data/x-ray-sample-splitted	

Figure 15) Experiments section in cnvrg.io platform (2 of 2).



7 Conclusion

NetApp and cnvrg.io partnered to offer customers a complete data management solution for ML/DL software development. ONTAP AI provides high-performance compute and storage for any scale of operation, and cnvrg.io software streamlines data science workflows increases resource utilization.

With NetApp and cnvrg.io, customers have a code-first, full stack, container/Kubernetes and open platform. NetApp and cnvrg.io accelerate data science from research to production across any platform in any cloud and on-premises environment.

Acknowledgments

- Mike Oglesby, Technical Marketing Engineer, NetApp
- Santosh Rao, Senior Technical Director, NetApp

Where to Find Additional Information

To learn more about the information that is described in this document, see the following resources:

- Cnvrg.io (https://cnvrg.io):
  - Cnvrg CORE (free ML platform)
    https://cnvrg.io/platform/core
  - Cnvrg docs
    https://app.cnvrg.io/docs
- NVIDIA DGX-1 servers:
  - NVIDIA DGX-1 servers
    https://www.nvidia.com/en-us/data-center/dgx-1/

    - NVIDIA Tesla V100 Tensor Core GPU
https://www.nvidia.com/en-us/data-center/tesla-v100/
    - NVIDIA GPU Cloud (NGC)
https://www.nvidia.com/en-us/gpu-cloud/
- NetApp AFF systems:
    - AFF datasheet
https://www.netapp.com/us/media/ds-3582.pdf
    - NetApp FlashAdvantage for AFF
https://www.netapp.com/us/media/ds-3733.pdf
    - ONTAP 9.x documentation
http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286
    - NetApp FlexGroup technical report
https://www.netapp.com/us/media/tr-4557.pdf
- NetApp persistent storage for containers:
    - NetApp Trident
https://netapp.io/persistent-storage-provisioner-for-kubernetes/
- NetApp Interoperability Matrix:
    - NetApp Interoperability Matrix Tool
http://support.netapp.com/matrix
- ONTAP AI networking:
    - Cisco Nexus 3232C Switches
https://www.cisco.com/c/en/us/products/switches/nexus-3232c-switch/index.html
    - Mellanox Spectrum 2000 series switches
http://www.mellanox.com/page/products_dyn?product_family=251&mtag=sn2000
- ML framework and tools:
    - DALI
https://github.com/NVIDIA/DALI
    - TensorFlow: An Open-Source Machine Learning Framework for Everyone
https://www.tensorflow.org/
    - Horovod: Uber’s Open-Source Distributed Deep Learning Framework for TensorFlow
https://eng.uber.com/horovod/
    - Enabling GPUs in the Container Runtime Ecosystem
https://devblogs.nvidia.com/gpu-containers-runtime/
    - Docker
https://docs.docker.com
    - Kubernetes
https://kubernetes.io/docs/home/
    - NVIDIA DeepOps
https://github.com/NVIDIA/deepops
    - Kubeflow
http://www.kubeflow.org/
    - Jupyter Notebook Server
http://www.jupyter.org/
- Dataset and benchmarks:
    - NIH chest X-ray dataset
https://nihcc.app.box.com/v/ChestXray-NIHCC

– Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, Ronald Summers, ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases, IEEE CVPR, pp. 3462-3471, 2017

Version History

Version	Date	Document Version History
Version 1.0	June 2020	Initial release.

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

TR-4841-0620