

# **NVIDIA DGX Solution Stack**

## White Paper

Featuring NVIDIA DGX A100 Systems

# **Document History**

#### WP-10748-001

Version	Date	Authors	Description of Change
01	2021-11-05	Adam Tetelman, Jonny Devaprasad, Martijn de Vries, Michael Balint, Ray Burgemeestre, Robert Clark, and Robert Sohigian	Initial release
02			
03			

## Contents

Overview1
DGX Solution Stack Architecture
Hardware2
Base Configuration2
NVIDIA DGX A100 System2
Networking3
Bright Cluster Manager Head Node3
Storage3
Software4
Bright Cluster Manager4
Run:Al
Deploying the DGX Solution Stack
Pre-Setup5
Deployment with Bright Cluster Manager5
Installing BCM on the Head Node5
Provisioning DGX OS Across Compute Systems7
Deploying Kubernetes
Deploying Run:AI
Configured OpenID Connect for Kubernetes11
Downloading and Installing Run:Al11
Creating a Project12
Installing the Run:AI CLI
Launching a Job from the CLI13
Using the DGX Solution Stack
System Administrator Use14
Adding Nodes to the Cluster Using Bright View14
Removing Nodes from the Cluster Using Bright View14
Reprovisioning a Node Using Bright View14
Connecting to a Node Via SSH14
Granting New Users Access to a System15

Uninstalling Kubernetes	15
Updating Run:AI CLI	15
Altering Quotas for Run:Al	15
Data Scientist Use	16
Installing and Configuring Jupyter Hub	16
Launching Notebooks with JupyterHub	17
Appendix A. Bill of Materials	5

## Overview

Artificial Intelligence (AI) infrastructure requires significant compute resources to train the latest state-of-the-art models efficiently, often requiring multiple nodes running in a distributed cluster. While cloud computing provides easy on-ramping to train AI models at scale, the costs can quickly add up over time, pushing many enterprises to consider moving to an on-premises data center for their needs.

Building AI infrastructure on-premises is a complex and often confusing process with an endless number of potential solutions available. Without careful planning and coordination, typical cluster deployments can suffer from under-utilization of resources and be hard for system administrators to manage at scale as individual users make one-off changes to systems that might affect performance or stability.

The NVIDIA DGX<sup>™</sup> Solution Stack leverages various ISV solutions to deploy, manage, and modify an on-premises cluster to maximize efficiency of resources and simplify administration of the cluster. The DGX Solution Stack comprises multiple stacked layers including a layer for provisioning and managing hosts, followed by a job and resource scheduler, and finishing with a tool to track and version models at scale.

These tools give all the capabilities needed to manage and use a cluster:

- Provision the operating system (OS).
- Manage node configuration.
- Deploy Kubernetes.
- Grant restricted node access.
- Deploy training jobs.
- Deploy development jobs.
- Set and monitor resource quotas.
- Review training jobs in cluster via dashboard.
- Monitor system utilization.
- Note: This document describes a full solution stack intended to demonstrate the capabilities available to data scientists in a development cluster. In a production cluster—where security or access are a larger concern—customers are advised to work with their NVIDIA contact to perform additional hardening configuration steps.

# **DGX Solution Stack Architecture**

## Hardware

The DGX Solution Stack is designed to run on a relatively small cluster composed of compute nodes and a dedicated network connected to a head node. Refer to Bill of Materials for a detailed list of the hardware components.

### **Base Configuration**

The base DGX Solution Stack configuration consists of the components in Table 1.

Component	Quantity	Description
NVIDIA DGX A100™ system	2	Compute nodes that run GPU workloads in a Kubernetes cluster
HPE DL325 server	1	Head node that runs Bright Cluster Manager (BCM) software that is used to provision and manage the DGX A100 systems
NVIDIA® AS4610 switch	1	Switch that handles external communication over a 1 GbE interface to download packages, dependencies, and other cluster resources.
NVIDIA SN2700 or NVIDIA SN3700V switch	1	Switch that is connected between the nodes and used for provisioning systems.

Table 1. Base configuration

While the base configuration calls for two DGX A100 systems, the stack can be scaled up with additional compute nodes to handle larger workloads on the cluster. The remainder of this guide assumes that only two DGX A100 systems are part of the cluster, but any step can be updated to include larger numbers of compute nodes as necessary.

#### NVIDIA DGX A100 System

The NVIDIA DGX A100 system offers unprecedented compute performance by packing eight NVIDIA A100 Tensor Core GPUs connected with NVIDIA NVLink<sup>®</sup> and NVSwitch<sup>™</sup> technologies for fast inter-GPU communication. The DGX Solution Stack starts with two DGX A100 systems that act as workers in the Kubernetes cluster, allowing the GPUs to be consumed by jobs started with the scheduler.

#### Networking

The DGX Solution Stack requires the BMC on every node to be connected to a management switch, such as an AS4610 switch, to manage each physical device (Figure 1). Additionally, a 1 GbE up to a 10 GbE connection between each node's primary management port to the same AS4610 switch is required for external communication on each node, if not running in an air-gapped environment.



Figure 1. Cluster network diagram.

A dedicated SN2700 or SN3700 switch is also connected on all nodes that serves as the primary provisioning device. This dedicated network runs on either a 100 or 200 GbE interface and is used by BCM to communicate with the nodes in the cluster.

#### Bright Cluster Manager Head Node

The BCM runs on a single HPE DL325 server. This node is primarily used for provisioning but can also be used as a central login node for cluster users to launch and monitor jobs.

#### Storage

NVIDIA partners with multiple storage vendors including IBM, DDN, NetApp, Dell, VAST, Pure Storage, and Weka.io. Each partner offers their own storage solution designed for efficiency and scalability for AI clusters. The DGX Solution Stack is designed to be flexible with storage and works with any partner solution that can be shared among all compute nodes in the cluster. In the absence of a shared filesystem, local storage on the compute nodes can also be used.

## Software

### **Bright Cluster Manager**

BCM is a provisioning tool that allows administrators to easily install OS images, manage hosts in a cluster, and monitor usage and performance. BCM can run on a single dedicated node that is connected to all other compute nodes in a cluster by provisioning servers over a dedicated network interface.

BCM uses a modified version of the latest NVIDIA DGX OS, which includes updates and settings to allow the clients to function seamlessly with BCM. Using the preconfigured image across the cluster allows for a consistent environment on all nodes, enabling administrators to reduce skew in dependencies and settings across systems while also reducing the complexity of managing clusters. The image includes all required NVIDIA packages to leverage GPU acceleration in addition to any and all packages that may be used. Provisioning systems using this image is faster than installing an OS from scratch and manually installing packages as all of the pre- and post-install steps are already built into the image. Burn-in tests are also run automatically after installation, giving system administrators confidence that the cluster is healthy and contains all of the necessary tools to allow developers to complete their work.

#### Run:Al

Run:AI is a SLURM-like job scheduler that runs on top of Kubernetes to intelligently schedule jobs to maximize resource utilization. Run:AI differs from traditional approaches to shared resources that commonly see individual users assigned specific hardware, resulting in compute resources being underutilized with hardware left sitting idle. Instead, Run:AI disaggregates cluster resources from users by placing all hardware in a virtualized pool. Users can then start their jobs in a centralized manner where Run:AI will dynamically allocate resources depending on the needs of the specific job.

The scheduler supports layered user management to limit the number of resources each person or team can use at any given time, preventing an entire cluster from being used by a small subset. During periods of lesser activity, Run:AI supports job bursting which allows additional compute hardware to be allocated to supported jobs to finish faster. This ensures that a cluster is running optimally to minimize training, testing, and validation time and increase time-to-insight without wasting resources.

# Deploying the DGX Solution Stack

## **Pre-Setup**

Before deploying the DGX Solution Stack, ensure that networking has been properly configured on all nodes. Each node in the cluster should have the following connections:

- ▶ 1x 1/10 GbE connection for BMC.
- 1x 1/10 GbE connection for host management and external connectivity.
- ▶ 1x 100/200 GbE connection for provisioning and communication.

Both the BMC and the 100/200 GbE connections on all compute nodes need to be routable from the BCM host.

While installing the compute nodes to be provisioned:

- Record the MAC addresses for the 100/200 GbE interfaces. These MAC addresses are used later on during the deployment process to indicate which devices to provision and on which interface.
- 2. Enter the BIOS on the compute nodes and update the boot order to have networking be the first boot device.

If necessary, change the default networking boot order to boot off the 100/200 GbE interface that was identified in step 1. The MAC address listed in the network boot order should match the saved MAC address for that node.

The install steps outlined here require access to the Internet to pull packages from external sources such as GitHub and NVIDIA NGC<sup>™</sup>. Alternative install steps exist for an air-gapped installation.

Lastly, ensure a copy of the preconfigured DGX OS image and BCM image from Bright Computing is on hand for installation.

## **Deployment with Bright Cluster Manager**

The initial deployment step involves deploying the BCM onto a head node. Once BCM is installed, all other hardware provisioning can be done using the Bright UI or CLI.

#### Installing BCM on the Head Node

To deploy BCM on the head node, start by inserting the BCM image on the head node. It is recommended to install the image directly on the machine using a physical medium, but many recent

BMCs support injecting virtual media in a remote console session. Once inserted, boot from the BCM image to begin installation.

- 3. Once booted into the image, you will be prompted to install BCM in Graphical or Text mode. Graphical mode is recommended and will be used as an example for the remainder of this section, but text mode should be used if the connected display has trouble rendering properly.
- 4. Click Start Installation on the welcome screen.
- 5. Review both of the licenses on the next screen and check both boxes indicating your acceptance of the licenses before continuing to the next screen.
- 6. The Kernel Modules screen can be ignored, and the next screen selected.
- 7. Review the hardware listed on the Hardware Info screen. Ensure the hardware listed matches what is expected for the head node. In particular, the head node should have the 100g/200g interface listed as this is used for provisioning, as well as storage devices to install the management software on. If any items are not listed, double check the health and connectivity of all components on the server.
- 8. Select the location the image was saved to for the Installation Source.
- 9. For the General Cluster screen, set the domains and nameservers for the data center as necessary.

Fill out the other fields (administrator email, time zone, cluster name) to reflect the local setup if needed.

- 10. Select None on the Workload Manager screen as the DGX Solution Stack offers its own scheduler.
- 11. Select Type 1 Network for the Network Topology screen. This topology is for isolated private networks as is the case with the DGX Solution Stack.
- 12. Enter a hostname and password for the head node on the Head node settings screen. This password will be used to login to BCM and should be kept in a safe place.
- 13. The Compute node settings screen should be updated to reflect the DGX Solution Stack deployment.

By default, the DGX Solution Stack has two compute nodes (DGX A100 systems) in a single rack.

- 14. For BMC Configuration, ensure both the head node and compute nodes are set to use IPMI for the BMC network type. Instruct BCM to configure the IPMI over DHCP depending on the data center network topology. If the BMCs do not already have an existing IP address and should get an address from DHCP, set the mode to DHCP. Otherwise, use a static IP address for the BMC.
- 15. Update the network information in the Networks screen as necessary. In most scenarios, the default IP address and names will work, but in case the IP address poses a potential conflict in the data center, change the subnet to a new range.
- 16. Verify that the 100/200 GbE interface is listed on the Head node interfaces screen. |If not, add the interface to the external network.
- 17. Use the default interfaces on the Compute node interfaces screen.
- 18. Select the disks to install BCM on in the <code>Disk layout screen</code>.

Caution: The selected disks will be erased during the installation process.

- **19.** Ignore the options in the Additional software screen as the provided DGX OS image contains the necessary components.
- 20. Review the selections in the Summary screen.
- 21. Click  ${\tt Next}$  starts the installation process.
- 22. Once the installation process is complete, remove the installation medium and reboot the head node.
- 23. When the system comes back online, enter root for the username and the password step 10. BCM provides a cmsh utility that can be used to configure BCM.
- 24. Use the BCM cm-docker-setup utility to install Docker on the head node. The default values can be used throughout the setup process, but if any additional registries, proxies, or other settings are required, they can be added.

#### **Provisioning DGX OS Across Compute Systems**

Provisioning a compute system with DGX OS is straightforward once BCM has been configured and the first boot device is updated on the DGX A100 systems.

25. Use a browser to open the Bright View GUI.

The Bright View GUI can be accessed at IP address of the head node on port 8081. For example, if the head node is on 192.168.0.3, open the browser to:

https://192.168.0.3:8081/bright-view.

This is the main GUI used for managing the cluster. You will be prompted with a login for the head node at this point. Use the same credentials as with the head node's login to access the GUI.

- 26. Once logged in, click Devices > All Devices in the menu on the left of the screen to begin adding a device.
- 27. Click the + ADD button on the bottom of the screen to display a list of devices to add.
- 28. Select Node that will open another menu prompting information for a new node.
- 29. In the new menu that opens, enter a hostname for the machine to use as well as the MAC address of the 100/200 GbE interface used for provisioning that was saved from an earlier step.
- **30.** Once complete, click SAVE in the bottom right corner.
- 31. Repeat this process for all nodes that will be part of the cluster.
- 32. Once all nodes have been entered, reboot the compute nodes to start the provisioning process.

Assuming the primary boot device has been properly set to match the MAC address that was entered in Bright View, the compute systems should automatically begin the provisioning process and overwrite any existing data on the primary drives.

After provisioning finishes, the devices can be accessed by logging into the head node and running cmsh followed by device ssh <hostname> where <hostname> is any hostname that was supplied during the device addition process. BCM automatically configures SSH keys and will connect to the compute devices without prompting.

## **Deploying Kubernetes**

BCM includes a utility to install Kubernetes on the cluster from the head node. To begin, start by logging into the head node and running cm-kubernetes-setup. If a configuration file has already been created, this process can be skipped by using the -c flag and point to the YAML file with the cm-kubernetes-setup command to use the existing configuration. If a configuration has not been created yet, follow the steps below after entering cm-kubernetes-setup.

- 1. Select Deploy in the menu to begin the installation process.
- 2. If using Ceph, there will be an option to use Ceph for the persistent storage. If there are no plans to use Ceph, select yes to continue without Ceph.
- 3. Update the network ranges used for Kubernetes as necessary before continuing to the next screen. Ensure that there will not be any potential conflicts with the default ranges and any existing addresses in the data center.
- 4. Select yes to expose the Kubernetes API server to the external network.
- 5. Select internalnet for the internal network to be used by Kubernetes. This network should be a dedicated interface to communicate between nodes in the Kubernetes cluster.
- 6. Select the head node as the master node for Kubernetes. Additional compute nodes can also be selected, but for smaller-scale clusters it is recommended to use the BCM head node as the sole master node for Kubernetes.
- 7. For the node category selection page, select the category to install Kubernetes on. Note that this will install Kubernetes on **all** nodes in the cluster with the given category, so be sure that all nodes in the cluster have Kubernetes installed before selecting the category. If there are some nodes in that category that will *not* run Kubernetes, keep all options blank and select OK. The next screen will allow you to select the individual nodes to install Kubernetes on.
- 8. Select an odd number of nodes to serve as etcd nodes. This will vary depending on the size of the cluster, but at a minimum the head node should be selected and if there are two or more compute nodes in the cluster, two of the compute nodes should also be selected.
- Select to use existing third-party Docker packages. Given Docker is included with the DGX OS
  image, and was manually installed on the Bright head node, it does not need to be reinstalled.
  Using the third-party Docker packages tells the Kubernetes setup utility to ignore attempting to
  install Docker.
- 10. Verify the default ports and directories that are chosen by the utility. These are the standard ports used for Kubernetes and should be open unless there is an existing Kubernetes cluster running in which case the ports should be changed.
- 11. Select the recommended Calico plugin for the network plugin.
- 12. For the addons selection screen, keep the default options selected while making sure NVIDIA device plugin for Kubernetes is disabled. If this option were selected, it will attempt to reinstall several Docker-related components that can potentially cause conflicts.
- 13. Use the default ingress ports for Kubernetes as specified in the wizard unless the ports are already taken by another service.
- 14. Select Save config & deploy to begin the installation process. Depending on network and system speeds, this can take 10 minutes or longer to compute, including a reboot of all compute systems.

After the deployment is complete, Kubernetes should now be installed across the cluster. Next, the NVIDIA application for Kubernetes needs to be enabled to allow Kubernetes to use GPUs with Pods. This can also be done on the BCM head node:

- 1. Enter the cluster manager shell on the head node with cmsh.
- 2. Set the NVIDIA application group to enabled:

```
[bcm-head]% kubernetes
[bcm-head->kubernetes[default]]% appgroups
[bcm-head->kubernetes[default]->appgroups]% use system
[bcm-head->kubernetes[default]->appgroups[system]]% applications
[bcm-head->kubernetes[default]->appgroups[system]->applications]% set nvidia
enabled yes
```

3. Save the changes by typing commit.

Next, the device labels for the workers need to be updated with the gpu-accelerator label in order to be identified as GPU-capable in Kubernetes to launch Pods with GPU access. This will also be done on the BCM head node:

- 1. Enter the cluster manager shell on the head node with cmsh if not already done.
- 2. Add the GPU accelerator label to the worker nodes (update the <gpu-worker-n> list as appropriate for your cluster to list all compute nodes):

```
[bcm-head]% kubernetes
[bcm-head->kubernetes[default]]% labelsets
[bcm-head->kubernetes[default]->labelsets]% use master
[bcm-head->kubernetes*[default*]->labelsets*[master*]]% set labels
brightcomputing.com/gpu-accelerator=
[bcm-head->kubernetes*[default*]->labelsets*[master*]]% append nodes <gpu-
worker-1> <gpu-worker-2> ...
```

3. Save the changes by typing commit.

Lastly, Kubernetes needs to be loaded on the compute node if not done already. This can be done by running module load kubernetes and module initadd kubernetes to load the module on every login.

Verify that the Kubernetes cluster is healthy by running kubectl get pods -A after a few minutes. The result should look similar to Figure 2 though some of the Pod names might be different.

NAMESPACE	NAME	READY	STATUS	RESTARTS
ingress-nginx	nginx-ingress-controller-7867c6597d-qfngl	1/1	Running	1
kube-system	calico-kube-controllers-559f6dbfd9-pts4f	1/1	Running	1
kube-system	calico-node-dz6fv	1/1	Running	1
kube-system	calico-node-zkrmt	1/1	Running	0
kube-system	calico-node-znsf8	1/1	Running	0
kube-system	coredns-b5cdc886c-rnqmj	1/1	Running	1
kube-system	coredns-b5cdc886c-v14zk	1/1	Running	1
kube-system	metrics-server-c757f995c-7c7g5	1/1	Running	1
kube-system	metrics-server-c757f995c-vknd7	1/1	Running	0
kube-system	nvidia-device-plugin-daemonset-rvxx8	1/1	Running	0
kube-system	nvidia-device-plugin-daemonset-ttwrb	1/1	Running	0
kubernetes-dashboard	dashboard-metrics-scraper-664b9c878b-x1w52	1/1	Running	0
kubernetes-dashboard	kubernetes-dashboard-5676cf477-hqqmf	1/1	Running	1

#### Figure 2. Healthy Kubernetes cluster post-installation

In rare scenarios, the Calico Pods might fail to start. This is typically caused by the kernel's RPF check being set to "loose" which causes the service to fail, indicating the services can be easily spoofed. In this case, run kubectl -n kube-system set env daemonset/calico-node FELIX\_IGNORELOOSERPF=true which will ignore this setting and all services should come back online again.

At this point, Kubernetes should now be able to deploy Pods which leverage GPUs. This can be tested with a simple GPU Pod to test a few iterations of ResNet-50 on a worker node in the cluster.

- Download the sample pod configuration file using wget https://raw.githubusercontent.com/NVIDIA/deepops/release-21.06/workloads/examples/k8s/tensorflow-job.yml.
- 2. Create a pod.

kubectl create -f tensorflow-job.yml

3. Monitor the progress of the job.

```
kubectl logs -f -l controller-uid=$(kubectl get job tensorflow-job -o
jsonpath={.metadata.labels.controller-uid})
```

As the job terminates, the output should be similar to the following:

```
PY 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0]
TF 1.15.5
Script arguments:
 --layers 50
 --batch size 64
 --num iter 90
 --iter unit epoch
 --display every 10
 --precision fp16
  --use xla True
  --predict False
Training
 Step Epoch Img/sec Loss LR
    1 1.0 2.5 7.850 8.821 2.00000
   10 10.0
             27.5 2.278 3.250 1.62000
   20 20.0 1901.6 0.044 1.021 1.24469
   30 30.0 1902.1 0.924 1.904 0.91877
   40 40.0 1895.4 0.335 1.320 0.64222
   50 50.0 1908.6 0.456 1.444 0.41506
   60 60.0 1896.1 0.676 1.667 0.23728
   70 70.0 1899.2 0.479 1.474 0.10889
   80 80.0 1894.3 0.112 1.108 0.02988
   90 90.0 568.0 0.001 0.997 0.00025
```

#### 4. Once complete, delete the job.

```
kubectl delete -f tensorflow-job.yml
```

## **Deploying Run:Al**

In order to efficiently utilize the Kubernetes cluster deployed by Bright Cluster Manager, Run:AI is installed and securely integrated into the cluster. This enables HPC scheduling capabilities, quota controls, and more through the Run:AI UI or CLI.

### **Configured OpenID Connect for Kubernetes**

Run:Al requires OpenID Connect (OIDC) to be configured on the Kubernetes cluster for proper functionality. To configure OIDC, open the Kubernetes Server API file at

/etc/kubernetes/manifests/kube-apiserver.yaml and add the following contents to the spec > containers > command section:

- --oidc-client-id=<CLIENT ID>
- --oidc-issuer-url=https://runai-prod.auth0.com/
- --oidc-username-prefix=-
- --oidc-groups-claim=email

#### **Downloading and Installing Run:AI**

Installation of Run:AI services require an account to be created by the Run:AI team. This should be done beforehand, and the account should be provided to the installation team.

33. Login to <u>https://app.run.ai/</u> using the provided credentials.

- 1. In the wizard that appears, enter a name for the cluster, such as "solution-stack" and click the Create button.
- 2. On the next screen of the wizard, choose On Premise for the target platform. This will provide a command in step 2 of the wizard to download the configuration file.
- 3. Run the provided curl command on the Bright head node.
- 4. Copy the commands provided in step 5 of the wizard to install Run:AI services. The commands should look similar to the following:

```
helm repo add runai https://run-ai-charts.storage.googleapis.com
helm repo update
helm upgrade -i runai-cluster runai/runai-cluster -n runai -f runai-
solution-stack.yaml --create-namespace
```

- 5. Click Next followed by Done in the wizard to finish the setup.
- 6. After a few minutes, verify that the services are online.

Running kubectl get pods -A on the BCM head node should provide an output similar to:

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
ingress-nginx	nginx-ingress-controller-7867c6597d-qfngl	1/1	Running	1	3d1h
kube-system	calico-kube-controllers-559f6dbfd9-pts4f	1/1	Running	1	3d1h
kube-system	calico-node-dz6fv	1/1	Running	1	3d1h
kube-system	calico-node-zkrmt	1/1	Running	0	3d1h
kube-system	calico-node-znsf8	1/1	Running	0	3d1h
kube-system	coredns-b5cdc886c-rnqmj	1/1	Running	1	3d1h
kube-system	coredns-b5cdc886c-v14zk	1/1	Running	1	3d1h
kube-system	metrics-server-c757f995c-7c7g5	1/1	Running	1	3d1h
kube-system	metrics-server-c757f995c-vknd7	1/1	Running	0	3d1h
kube-system	nvidia-device-plugin-daemonset-rvxx8	1/1	Running	0	3d1h
kube-system	nvidia-device-plugin-daemonset-ttwrb	1/1	Running	0	3d1h
kubernetes-dashboard	dashboard-metrics-scraper-664b9c878b-x1w52	1/1	Running	0	3d1h
kubernetes-dashboard	kubernetes-dashboard-5676cf477-hgqmf	1/1	Running	1	3d1h
monitoring	prometheus-runai-cluster-kube-prometh-prometheus-0	2/2	Running	1	3d1h
monitoring	runai-cluster-kube-prometh-operator-f8f99bdfb-zmhvm	1/1	Running	0	3d1h
monitoring	runai-cluster-kube-state-metrics-8496c6b7db-wgrjj	1/1	Running	0	3d1h
monitoring	runai-cluster-prometheus-node-exporter-cwh89	1/1	Running	0	3d1h
monitoring	runai-cluster-prometheus-node-exporter-d2cnz	1/1	Running	0	3d1h
monitoring	runai-cluster-prometheus-node-exporter-zb6tn	1/1	Running	1	3d1h
node-feature-discovery	nfd-master-5fbbdf8f74-pptww	1/1	Running	0	3d1h
node-feature-discovery	nfd-worker-anpc7	1/1	Running	3	3d1h
node-feature-discovery	nfd-worker-lmdr8	1/1	Running	5	3d1h
node-feature-discovery	runai-cluster-gpu-feature-discovery-5d4x5	1/1	Running	0	3d1h
node-feature-discovery	runai-cluster-gpu-feature-discovery-68p2w	1/1	Running	0	3d1h
runai	db-migrations-79557c8ffb-wcr5j	1/1	Running	0	3d1h
runai	mpi-operator-78bc5f7dcf-9vd5v	1/1	Running	1	3d1h
runai	nvidia-device-plugin-daemonset-2kk7d	1/1	Running	0	3d1h
runai	nvidia-device-plugin-daemonset-drc7n	1/1	Running	0	3d1h
runai	researcher-service-85fc785c9-dhtg6	2/2	Running	0	3d1h
runai	runai-admission-controller-76d58b99d9-gm9gc	1/1	Running	0	3d1h
runai	runai-agent-5fcd849dcc-hzgnt	1/1	Running	0	3d1h
runai	runai-container-toolkit-ljplp	1/1	Running	0	3d1h
runai	runai-container-toolkit-zt8lh	1/1	Running	0	3d1h
runai	runai-db-0	1/1	Running	0	3d1h
runai	runai-dcgm-exporter-7h18m	1/1	Running	0	3d1h
runai	runai-dcgm-exporter-pw4vd	1/1	Running	0	3d1h
runai	runai-job-controller-b6f5699bc-gdptt	1/1	Running	0	3d1h
runai	runai-operator-5f4875cb9-w9db4	1/1	Running	0	3d1h
runai	runai-project-controller-866bfd7fb9-fzx8d	1/1	Running	0	3d1h
runai	runai-prometheus-pushgateway-7db57854f9-j8mpn	1/1	Running	0	3d1h
runai	runai-scheduler-6c687b4b95-w2b79	1/1	Running	0	3d1h

- 7. Once all services show as successfully running, navigate back to <a href="https://app.run.ai/">https://app.run.ai/</a> and view the new dashboard.
- 8. Verify that the number of nodes and GPUs listed in the new dashboard match the cluster.

#### **Creating a Project**

Before starting a job with Run:AI, at least one project needs to be created in the Run:AI user interface. The project is used to specify and allocate cluster resources to teams and users.

In the Run:AI UI at <u>https://app.run.ai/</u>, navigate to the Projects section and click Add New Project in the top-right corner of the page. This will open a wizard to create the first project.

- 1. Enter a name for the project, such as "deep-learning."
- 2. Input the maximum number of GPUs this project is able to use across all jobs and users. If there is only a single project planned, assign the total number of GPUs available in the cluster.
- Choose to allow the project to go over quota.
   By allowing the project to go over-quota, extra GPUs beyond what is assigned in step 2 can be allocated to the project if extra resources are available in the cluster.
- To limit projects to run on a subset of nodes, select the Node Affinity tab and list the nodes to limit the project to.
   For smaller clusters, it is recommended to disable any limits and allow the project to run on all nodes.
- 5. To limit the amount of time a job can run, select the Time Limit tab and enter in a time limit for all jobs.

By default, jobs are allowed to run indefinitely.

- 6. Once finished configuring projects, select Save. The project will be displayed in the list.
- 7. Continue this process for all desired projects for the cluster.

### Installing the Run:AI CLI

Jobs can be scheduled and launched using the Run:AI CLI tool from the Bright head node. This tool will automatically allocate GPU resources and launch workloads on the cluster.

Download and install the latest version of the Run:AI CLI tool found on <u>GitHub</u> to the Bright head node:

- 1. To download the latest version of the CLI tool, run wget https://github.com/run-ai/runaicli/releases/download/<version number>/runai-cli-<version number>-linux-amd64.tar.gz. Replace <version number> with the latest version listed on GitHub.
- 2. Extract the file with tar -xvf runai-cli-\*-linux-amd64.tar.gz.
- 3. Install the CLI tool with ./install-runai.sh.
- 4. Verify the installation.

Use runai list jobs to ensure the command works. It shows that no jobs are currently running.

To set a default project created from the previous section, use runai config project <project name>. This will assume that all runai commands belong to the listed project. If a default project is not chosen, runai commands will require -p <project name> be included to specify the project.

Before running the RUN:AI CLI, ensure that Kubernetes access is configured. This can be accomplished by ensuring the kubeconfig file located in /root/.kube/config is copied to the user's directory at ~/.kube/config. A user can verify Kubernetes access by running a command such as kubectl get nodes.

#### Launching a Job from the CLI

Once Run:AI has been installed and configured, a new job can be created from the CLI. The basic syntax for running Docker jobs with Run:AI is to use runai submit <command>. A job leveraging Docker containers, similar to the one in Deploying Kubernetes, can be launched with the following command:

```
runai submit --name resnet50 -i nvcr.io/nvidia/tensorflow:21.03-tf1-py3 -g 8
--command -- bash -c "python /workspace/nvidia-examples/cnn/resnet.py --
layers=50 --batch_size=64"
```

This will run the ResNet-50 inference example using 8 GPUs in the cluster, if available. To view the status of the job, use runai list jobs. This should show the name of the job as well as runtime, resource allocations, projects, and more.

To view the output of the job, use runai logs <job name>.

## Using the DGX Solution Stack

## System Administrator Use

System Administrators can leverage the DGX Solution Stack to perform ongoing maintenance and monitoring of the cluster.

#### Adding Nodes to the Cluster Using Bright View

In the Bright View home page, open the node creation wizard by navigating to Devices > All Nodes. Clicking the + ADD button will open a new menu to update node information. Select Node that will open another menu prompting information for a new node.

In the new menu that opens, enter a hostname for the machine to use as well as the MAC address of the 100/200 GbE interface used for provisioning that was saved from an earlier step. Once complete, click "SAVE" in the bottom right corner. Repeat this process for all nodes that will be part of the cluster.

### Removing Nodes from the Cluster Using Bright View

To remove a node from the cluster, open Bright View and navigate to the nodes screen at Devices > All Nodes. Identify the node to remove from the cluster and select the actions dropdown arrow to the right of the field. Select Delete from the dropdown list and select Delete again in the popup to confirm deletion. The deleted node will no longer be managed by BCM.

#### Reprovisioning a Node Using Bright View

To reprovision a node in the cluster, open Bright View and navigate to the nodes screen at Devices > All Nodes. Identify the node to remove from the cluster and select the actions dropdown arrow to the right of the field. Select Software Image and Reinstall Node from the dropdown list. This will begin the reinstallation process on the selected node.

#### Connecting to a Node Via SSH

The preferred method to interact with the cluster is to use the Run:AI CLI to launch jobs from the head node, but nodes can also be accessed directly over SSH. BCM automatically generates and shares SSH keys around the cluster for passwordless-SSH between nodes. To access any of the compute nodes, login to the Bright head node and SSH to any host without specifying a username or password: ssh <compute-1-hostname>

#### Granting New Users Access to a System

New users can be added to the cluster from the Bright head node. Bright's cluster manager shell includes a tool to add and update users across the cluster. This can be invoked with:

cmsh -c "user; add <username>; set password <password>; commit

Permissions for users can also be updated using cmsh as follows:

- 1. Enter the cluster manager shell with cmsh.
- 2. Select a user with user use <username>.
- 3. View all user properties with show.
- 4. Change any properties with set <property> <value>.
- 5. Save changes with commit.

To enable the user to access the Kubernetes cluster, they need to be added to the user database. Users can be added with:

cm-kubernetes-setup --add-user <username>

To give the user admin privileges to create resources, the user's role can be specified as a cluster admin:

cm-kubernetes-setup --add-user <username> --role cluster-admin

A pod security policy (PSP) should be created for all added users to prevent them from potentially gaining root access to the cluster. To learn more about creating and enabling a PSP, refer to the <u>official documentation</u>.

#### **Uninstalling Kubernetes**

BCM includes a configuration tool that can be used to uninstall Kubernetes from a cluster. To uninstall a Kubernetes cluster, simply run cm-kubernetes-setup on the Bright head node and select Uninstall from the list and confirm to have the cluster removed. The uninstall process will take less than a minute and Kubernetes will be removed from the head node and all workers.

#### Updating Run:AI CLI

The Run:AI CLI tool can be updated at any time with runai update which will automatically download and install the latest version of the tool.

#### Altering Quotas for Run:Al

The number of GPUs available for any Run:AI project can be altered at any time in the Run:AI web interface. To alter the GPU quota for a project, login to the web interface at <a href="https://app.run.ai">https://app.run.ai</a> and enter the Projects menu on the left of the screen. Select the desired project to update from the list, then click Edit Project in the top-right corner of the page. In the wizard that displays, specify the number of GPUs to assign for the project and click the Save button once finished. The GPU allocation will update automatically for future projects.

## Data Scientist Use

Data Scientists can leverage the DGX Solution Stack to accelerate their AI workflows and development.

### Installing and Configuring Jupyter Hub

Bright includes support for JupyterHub to launch interactive Jupyter notebooks with kernels running on compute nodes in the cluster.

Installation can be done on the Bright head node.

- 1. On the Bright head node, begin setup by running cm-jupyter-setup.
- 2. Press enter to deploy a new cluster.
- 3. Update the default overlay properties for JupyterHub login nodes as necessary and select OK.
- 4. Select the Bright head node in the list to serve as the login node for JupyterHub and select OK.
- 5. Update the default ports as necessary and continue to the next screen.
- 6. Select Save config & deploy to begin the deployment process.

After the deployment is finished, verify installation with the following commands:

- 1. Load the Jupyter module with module add jupyter and module initadd jupyter.
- 2. Ensure that the service is running with systemctl status jupyter -1. The service should show as active (running).
- 3. Verify that the server extensions are loaded and running with jupyter serverextension list. This should show all items as enabled with OK status.

```
config dir: /cm/shared/apps/jupyter/12.0.0/etc/jupyter
   jupyterlab enabled
   - Validating...
   jupyterlab 3.0.4 OK
cm_jupyter_kernel_creator enabled
   - Validating...
   cm_jupyter_kernel_creator 1.1.dev111+g6d78357 OK
cm_jupyter_addons enabled
   - Validating...
   cm_jupyter_addons 1.1.dev13+g2ba18eb OK
cm_jupyter_vnc enabled
   - Validating...
   cm_jupyter_vnc 1.1.dev48+g07b05aa OK
```

4. Verify that the lab extensions are loaded and running with <code>jupyter labextension list</code>. JupyterLab v3.0.4

```
/cm/shared/apps/jupyter/12.0.0/share/jupyter/labextensions
    @brightcomputing/jupyterlab-tools v0.2.7 enabled OK (python,
brightcomputing_jupyterlab_tools)
```

Once deployed, JupyterHub can be accessed on the specified port (default 8000) for the login node. To access JupyterHub, open a web browser with the IP address and port for the login node using the HTTPS protocol. Use any non-root account to login to the webpage. If needed, create a new user on the cluster by referring to Granting New Users Access to a System.

## Launching Notebooks with JupyterHub

After logging in to the JupyterHub interface, you will be greeted with the main launcher window, as seen in Figure 3.



Figure 3. The main launcher window for JupyterHub.

By selecting the Bright logo in the bottom of the list on the far left of the screen, several options for kernel templates are listed to make it easier to get started. To leverage the Kubernetes cluster, select the Python on Kubernetes template. This will open a new window to specify kernel settings. While most of the default options can likely stay as-is, the number of GPUs can be updated to reflect the necessary value for the kernel.

After the template is created, select the newly added icon from the launcher window and a notebook will open in a new window and a pod will automatically be created on the Kubernetes cluster for the task.

# Appendix A. Bill of Materials

Table 1 contains for the Bill of Materials for a base configuration.

#### Table 1. Bill of Materials

OPN	Quantity	Description
DGXA-2530F+P2CMI00	2	DGX A100 P3687 System 8x 80 GB GPUs Full
STOR-ENDPOINT	1	100 TB Storage solution for DGX A100 cluster
MCP1600-C002E30N	8	NVIDIA Passive Copper cable, ETH 100GbE, 100Gb/s, QSFP28, 2m, Black, 30AWG, CA-N
CAT5-CABLE-SHORT	1	Cat5e Snagless Unshielded (UTP) Network Patch Ethernet Cable-Blue, 3m (within utility rack)
MSN3700-VS2FC	2	NVIDIA Spectrum-2 based 200GbE 1U Open Switch with NVIDIA Cumulus <sup>®</sup> Linux <sup>®</sup> , 32 QSFP56 ports
MFS1S00-V010E	16	NVIDIA active fiber cable, 200GbE, 200Gb/s, QSFP56, LSZH, black pulltab, 10m
MFA1A00-C010	4	NVIDIA active fiber cable, ETH 100GbE, 100Gb/s, QSFP, LSZH, 10m
MCP1650-V00AE30	2	NVIDIA Passive Copper cable, 200GbE, 200Gb/s, QSFP56, LSZH, 0.5m, black pulltab, 30AWG
4610-54T-O-AC-B	1	1GE RJ45 1U Open Ethernet Switch with ONIE, 48-Port GE RJ45 port + 4x10G SFP+
UPGR-CUM-1G	1	UPGR-CUM-1G - Cumulus Linux software license for AS4610 1G Management Ethernet Switch.
RKIT-85-4610	1	Rack mount kit for AS4610-54T
CAT5-CABLE	2	Cat5e Snagless Unshielded (UTP) Network Patch Ethernet Cable-Blue, 10 m (from utility rack to GPU racks)
CAT5-CABLE-SHORT	9	Cat5e Snagless Unshielded (UTP) Network Patch Ethernet Cable-Blue, 3m (within utility rack)

MGMT-SERVER	1	HPE DL325 Server with NVIDIA ConnectX-6 NICs or equivalent
MCP1600-C002E30N	2	NVIDIA Passive Copper cable, ETH 100GbE, 100Gb/s, QSFP28, 2m, Black, 30AWG, CA-N
OPN-RACK-PS	4	Power supply Raritan or equivalent
E4882021122021S	2	Main Rack SKU (VERTIV)
718-A10080+P2CMI36	2	DGX A100 8X 80GB Full, Support, 3 Years
SUP-STOR-3S-ENDPOINT	1	3-Year Support for Storage Solution for DGX A100 Cluster
SUP-SN3700-3GNBD	2	NVIDIA Technical Support and Warranty - Gold 3 Year NBD On-Site Support for SN3700 Series Switch
SUP-SN3700-3GNBD	2	NVIDIA Technical Support and Warranty - Gold 3 Year NBD On-Site Support for SN3700 Series Switch
SUP-MGMT-3S-SERVER	1	3-Year Support for Management Nodes
SUP-4610-54T-3GNBD	1	NVIDIA Technical Support and Warranty - Gold 3 Year NBD On-Site Support for 4610-54T Series Switch

#### Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure that the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem that may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENTA, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

#### Trademarks

NVIDIA, the NVIDIA logo, NVIDIA DGX, and NVIDIA NGC are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

#### Copyright

© 2021 NVIDIA Corporation and Affiliates. All rights reserved.

