# TABLE OF CONTENTS

Chapter 1. Introduction To Docker And Containers............................................................ 1
Chapter 2. Preparing Your DGX System For Use With NVIDIA Container Runtime.............. 3
  2.1. Version 2.x Or Earlier: Installing Docker And nvidia-docker2................................. 4
  2.2. Preventing IP Address Conflicts With Docker.......................................................... 4
     2.2.1. Version 3.1.1 And Later: Preventing IP Address Conflicts Between Docker And DGX... 5
     2.2.2. Version 2.x Or Earlier: Preventing IP Address Conflicts Between Docker And DGX..... 6
  2.3. Configuring The Use Of Proxies............................................................................. 6
  2.4. Enabling Users To Run Docker Containers............................................................. 7
Chapter 3. Preparing To Use The Container Registry......................................................... 8
DGX-2™, DGX-1™, and DGX Station™ are designed to run containers. Containers hold the application as well as any libraries or code that are needed to run the application. Containers are portable within an operating system family. For example, you can create a container using Red Hat Enterprise Linux and run it on an Ubuntu system, or vice versa. The only common thread between the two operating systems is that they each need to have the container software so they can run containers.

Using containers allows you to create the software on whatever OS you are comfortable with and then run the application wherever you want. It also allows you to share the application with other users without having to rebuild the application on the OS they are using.

Containers are different than a virtual machine (VM) such as VMware. A VM has a complete operating system and possibly applications and data files. Containers do not contain a complete operating system. They only contain the software needed to run the application. The container relies on the host OS for things such as file system services, networking, and an OS kernel. The application in the container will always run the same everywhere, regardless of the OS/compute environment.

DGX-2, DGX-1, and DGX Station all use Docker. Docker is one of the most popular container services available and is very commonly used by developers in the Artificial Intelligence (AI) space. There is a public Docker repository that holds pre-built Docker containers. These containers can be a simple base OS such as CentOS, or they may be a complete application such as TensorFlow™. You can use these Docker containers for running the applications that they contain. You can use them as the basis for creating other containers, for example for extending a container.

To enable portability in Docker images that leverage GPUs, NVIDIA developed the NVIDIA Container Runtime for Docker (also known as nvidia-docker2). We will refer to the NVIDIA Container Runtime simply as nvidia-docker2 for the remainder of this guide for brevity.
nvidia-docker2 is an open-source project that provides a command line tool to mount the user-mode components of the NVIDIA driver and the GPUs into the Docker container at launch.

These containers ensure the best performance for your applications and should provide the best single-GPU performance and multi-GPU scaling.
Chapter 2.
PREPARING YOUR DGX SYSTEM FOR USE WITH NVIDIA CONTAINER RUNTIME

Some initial setup is required to be able to access GPU containers from the Docker command line for use on DGX-2, DGX-1, or on a DGX Station, or NGC. As a result of differences between the releases of the DGX™ OS and DGX hardware, the initial setup workflow depends on the DGX system and DGX OS version that you are using.

To determine the DGX OS software version on either the DGX-2, DGX-1, or DGX Station, enter the following command:

```
$ grep VERSION /etc/dgx-release
DGX_SWBUILD_VERSION=“3.1.1”
```

Based on the output from the command, choose from below which workflow best reflects your environment. Select the topics and perform the steps within that workflow.

**DGX-2 or DGX-1 with DGX OS Server 3.1.1 or Later Workflow**
1. Version 3.1.1 And Later: Preventing IP Address Conflicts Between Docker And DGX
2. Configuring The Use Of Proxies
3. Enabling Users To Run Docker Containers

**DGX-1 with DGX OS Server 2.x or Earlier**
1. Version 2.x Or Earlier: Installing Docker And nvidia-docker2
2. Version 2.x Or Earlier: Preventing IP Address Conflicts Between Docker And DGX
3. Configuring The Use Of Proxies
4. Enabling Users To Run Docker Containers

**DGX Station Workflow**
1. Version 3.1.1 And Later: Preventing IP Address Conflicts Between Docker And DGX
2. Configuring The Use Of Proxies
3. Enabling Users To Run Docker Containers

2.1. Version 2.x Or Earlier: Installing Docker And nvidia-docker2

Docker and nvidia-docker2 are included in DGX OS Server version 3.1.1 and later. Therefore, if DGX OS Server version 3.1.1 or later is installed, you can skip this task.

Docker and nvidia-docker2 are not included in DGX OS Server version 2.x or earlier. If DGX OS Server version 2.x or earlier is installed on your DGX-1, you must install Docker and nvidia-docker2 on the system.

Currently, there are two utilities that have been developed: nvidia-docker and nvidia-docker2. You can determine which are installed on your system by running:

```sh
touch /tmp/nvidia-docker*.deb && rm /tmp/nvidia-docker*.deb
```

1. Install Docker.

```sh
$ echo deb https://apt.dockerproject.org/repo ubuntu-trusty main | sudo tee /etc/apt/sources.list.d/docker.list
$ sudo apt-get update
$ sudo apt-get -y install docker-engine=1.12.6-0~ubuntu-trusty
```

2. Download and install nvidia-docker2.

a) Download the .deb file that contains v1.0.1 of nvidia-docker2 and nvidia-docker-plugin from GitHub.

```sh
$ wget -P /tmp https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.1/nvidia-docker_1.0.1-1_amd64.deb
```

b) Install nvidia-docker2 and nvidia-docker-plugin and then delete the .deb file you just downloaded.

```sh
$ sudo dpkg -i /tmp/nvidia-docker*.deb & & rm /tmp/nvidia-docker*.deb
```

2.2. Preventing IP Address Conflicts With Docker

To ensure that your DGX system can access the network interfaces for Docker containers, ensure that the containers are configured to use a subnet distinct from other network resources used by your DGX system.

By default, Docker uses the 172.17.0.0/16 subnet. If addresses within this range are already used on your DGX system’s network, change the Docker network to specify the IP address of the DNS server, bridge IP address range, and container IP address range to
be used by your GPU containers. Consult your network administrator to find out which IP addresses are used by your network.

If your network does not use addresses in the default Docker IP address range, no changes are needed and you can omit this task.

This task requires sudo privileges.

2.2.1. Version 3.1.1 And Later: Preventing IP Address Conflicts Between Docker And DGX

To ensure that the DGX can access the network interfaces for Docker containers, configure the containers to use a subnet distinct from other network resources used by the DGX. By default, Docker uses the 172.17.0.0/16 subnet. If addresses within this range are already used on the DGX network, change the Docker network to specify the bridge IP address range and container IP address range to be used by Docker containers.

This task requires sudo privileges.

1. Open the /etc/systemd/system/docker.service.d/docker-override.conf file in a plain-text editor, such as vi.

   
   ```
   sudo vi /etc/systemd/system/docker.service.d/docker-override.conf
   
   [Service]
   ExecStart=/usr/bin/dockerd -H fd:// -s overlay2 --default-shm-size=1G --
   bip=192.168.127.1/24 --fixed-cidr=192.168.127.128/25
   LimitMEMLOCK=infinity
   LimitSTACK=67108864
   ```

2. Append the following options to the line that begins ExecStart=/usr/bin/dockerd, which specifies the command to start the dockerd daemon:

   - `--bip=bridge-ip-address-range`
   - `--fixed-cidr=container-ip-address-range`

   **bridge-ip-address-range**
   The bridge IP address range to be used by Docker containers, for example, 192.168.127.1/24.

   **container-ip-address-range**
   The container IP address range to be used by Docker containers, for example, 192.168.127.128/25.

   This example shows a complete /etc/systemd/system/docker.service.d/docker-override.conf file that has been edited to specify the bridge IP address range and container IP address range to be used by Docker containers.

Starting with DGX release 3.1.4, the option `--disable-legacy-registry=false` is removed from the Docker CE service configuration file docker-override.conf. The option is removed for compatibility with Docker CE 17.12 and later.
3. Save and close the `/etc/systemd/system/docker.service.d/docker-override.conf` file.

4. Reload the Docker settings for the `systemd` daemon.

   $ sudo systemctl daemon-reload

5. Restart the `docker` service.

   $ sudo systemctl restart docker

2.2.2. Version 2.x Or Earlier: Preventing IP Address Conflicts Between Docker And DGX

DGX OS versions 2.x and earlier include a version of the Ubuntu operating system that uses Upstart for managing services. Therefore, the `dockerd` daemon is configured through the `/etc/default/docker` file and managed through the service command.

1. Open the `/etc/default/docker` file for editing.

   $ sudo vi /etc/default/docker

2. Modify the `/etc/default/docker` file, specifying the correct bridge IP address and IP address ranges for your network. Consult your IT administrator for the correct addresses.

   For example, if your DNS server exists at IP address 10.10.254.254, and the 192.168.0.0/24 subnet is not otherwise needed by the DGX-1, you can add the following line to the `/etc/default/docker` file:

   DOCKER_OPTS="--dns 10.10.254.254 --bip=192.168.0.1/24 --fixedcidr=192.168.0.0/24"

   If there is already a `DOCKER_OPTS` line, then add the parameters (text between the quote marks) to the `DOCKER_OPTS` environment variable.

3. Save and close the `/etc/default/docker` file when done.

4. Restart Docker with the new configuration.

   $ sudo service docker restart

2.3. Configuring The Use Of Proxies

If your network requires the use of a proxy, you must ensure that APT is configured to download Debian packages through HTTP, HTTPS, and FTP proxies. Docker will then be able to access the NGC container registry through these proxies.

1. Open the `/etc/apt/apt.conf.d/proxy.conf` file for editing and ensure that the following lines are present:

   Acquire::http::proxy "http://<username>:<password>@<host>:<port>/";
   Acquire::ftp::proxy "ftp://<username>:<password>@<host>:<port>/";
   Acquire::https::proxy "https://<username>:<password>@<host>:<port>/";
Where:

- **username** is your host username
- **password** is your host password
- **host** is the address of the proxy server
- **port** is the proxy server port

2. Save the `/etc/apt/apt.conf.d/proxy.conf` file.

3. Restart Docker with the new configuration.

```bash
$ sudo service docker restart
```

## 2.4. Enabling Users To Run Docker Containers

To prevent the `docker` daemon from running without protection against escalation of privileges, the Docker software requires `sudo` privileges to run containers. Meeting this requirement involves enabling users who will run Docker containers to run commands with `sudo` privileges. Therefore, you should ensure that only users whom you trust and who are aware of the potential risks to the DGX of running commands with `sudo` privileges are able to run Docker containers.

Before allowing multiple users to run commands with `sudo` privileges, consult your IT department to determine whether you would be violating your organization's security policies. For the security implications of enabling users to run Docker containers, see Docker security.

You can enable users to run the Docker containers in one of the following ways:

- Add each user as an administrator user with `sudo` privileges.
- Add each user as a standard user without `sudo` privileges and then add the user to the `docker` group. This approach is inherently insecure because any user who can send commands to the docker engine can escalate privilege and run root-user operations.

To add an existing user to the `docker` group, run this command:

```bash
$ sudo usermod -aG docker user-login-id
```

The user login ID of the existing user that you are adding to the `docker` group.
Chapter 3.
PREPARING TO USE THE CONTAINER REGISTRY

After you've set up your DGX-2, DGX-1, or DGX Station, you next need to obtain access to the NGC container registry where you can then pull containers and run neural networks, deploy deep learning models, and perform AI analytics in these containers on your DGX system.

For DGX-2, DGX-1, and DGX Station users, for step-by-step instructions on getting setup with the NGC container registry see the NGC Container Registry For DGX User Guide.
Notice

THE INFORMATION IN THIS GUIDE AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION
REFERENCED IN THIS GUIDE IS PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED,
STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY
DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever,
NVIDIA’s aggregate and cumulative liability towards customer for the product described in this guide shall
be limited in accordance with the NVIDIA terms and conditions of sale for the product.

THE NVIDIA PRODUCT DESCRIBED IN THIS GUIDE IS NOT FAULT TOLERANT AND IS NOT DESIGNED,
MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE,
AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A
SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE
(INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER
LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS
FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR
IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

NVIDIA makes no representation or warranty that the product described in this guide will be suitable for
any specified use without further testing or modification. Testing of all parameters of each product is not
necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and
fit for the application planned by customer and to do 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 guide. NVIDIA does not accept any liability related to any
default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA
product in any manner that is contrary to this guide, or (ii) customer product designs.

Other than the right for customer to use the information in this guide with the product, no other license,
either expressed or implied, is hereby granted by NVIDIA under this guide. Reproduction of information
in this guide is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without
alteration, and is accompanied by all associated conditions, limitations, and notices.

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, cuDNN, DALI, DIGITS, DGX, DGX-1, DGX-2, DGX Station, DLProf,
Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NvCaffe, PerfWorks, Pascal, SDK Manager,
Tegra, TensorRT, TensorRT Inference Server, Tesla, TF-TRT, and Volta 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

© 2020 NVIDIA Corporation. All rights reserved.

www.nvidia.com