Search

On This Page

HowTo Create OpenStack Cloud Image with NVIDIA GPU and Network Drivers

Is this page helpful?

Created on Jan 21, 2022

Scope

This guide provides instructions on how to create an OpenStack cloud image, including NVIDIA GPU driver, NVIDIA MLNX_OFED network drivers and additional performance benchmark tools, by using Diskimage-builder (DIB) elements.

Abbreviations and Acronyms

Term
CUDA
DIB
GPU
MLNX_O

Refere

- Disk
- Disk
- Dev
- Buil
- Clou

Introdu

When wor ack, a cloud image with specific pr red to support newly introduced ul to prepare a cloud image with a cus

Diskimage-builder (DIB) is a tool for automatic building of customized images for use in clouds and other environments.

The following short article covers the steps required when using **DIB** elements to create guest and deployment images with pre-installed NVIDIA MLNX_OFED, CUDA drivers, GPUDirect testing tools and several additional guest OS tweaks.

The DIB supports multiple OS distributions for the "build host" (the server used for building the guest image) and for the "guest target" (the image used for cloud instances). In the article below, CentOS 8 is used for both the build host and guest target as a reference.

Image Build Procedure For Guest OS Images

Preparing the Build Host

1. Install the OS on the server that will be used as a build host. The OS used for build host in this article is the latest CentOS 8.5 release.
2. Install the DIB and its prerequisites.

```
# dnf install qemu-img epel-release python3-pip
# pip3 install diskimage-builder
```

> ✓ **Note**
>
> In addition to the installation mentioned above, it is also recommended to install qemu-kvm on the build host for testing the generated image on a local VM before moving it to the cloud.

3. Create a main elements directory in which the custom elements will be created as sub-directories.

```
# mkdir -p /home/diskimage-builder/elements
# cd /home/diskimage-builder/elements
```

4. Create a sub directory for every custom element. For the custom elements described in this article, the following directories were created:

```
# mkdir mofed
# mkdir cloud-init-config
# mkdir cuda
# mkdir gpudirect-bench
```

Creating Custom Elements

NVIDIA MLNX_OFED Installation Element

> **Note**
>
> This element requires:
>
> - MLNX_OFED ISO file on the build host

The "mofed" element is used for building a guest image with an installed NVIDIA network drivers set, also known as MLNX_OFED, and it is adjusted for RHEL8.5 OS.

1. Download the relevant MLNX_OFED ISO file from the NVIDIA Networking Linux Drivers **Site**. The below is an example of how to download the RHEL 8.5 variant, as CentOS 8.5 is the OS being used in this article.

```
# cd /tmp/
# wget https://content.mellanox.com/ofed/MLNX_OFED-5.5-1.0.3.2/MLN
```

2. Download the mofed element file - **openstack-dib-elements-main-mofed.zip**, and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
element-deps	Element dependencies. There is no dependency on other elements in this case.
package-installs.yaml	A list of packages required for element installation.
pkg-map	Mapping of the package list to RedHat OS distribution packages.

File	Description
extra-data.d/70-copy-ofed-file	A script for copying the MLNX_OFED ISO image into a DIB environment during the build process. Requires a DIB_OFED_FILE environment variable for pointing the location of the MLNX_OFED ISO file in the build host OS.
install.d/70-ofed-install	A script for installing MLNX_OFED in a DIB environment during the build process with support in guest image kernel release.

3. Place the files under the /home/diskimage-builder/elements directory and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/mofed/extra-data.d/70
# chmod 755 /home/diskimage-builder/elements/mofed/install.d/70-mo
```

4. Set the environment variables with the MLNX_OFED ISO file location on the build host.

```
# export DIB_MOFED_FILE="/tmp/MLNX_OFED_LINUX-5.0-2.1.8.0-rhel7.8-
```

Cloud-init Configuration Element

> ✓ **Note**
>
> Cloud-init is a method for cross-platform cloud instance initialization. For more information on cloud-init, please refer to **Cloud-init Documentation**.

The "cloud-init-config" element is used for building a guest image with customized cloud-init parameters to be used during instance initialization. In this case, we will use it to make sure a system user is created with desired remote access methods during instance initialization.

As cloud-init is already included in the base CentOS image generated by the DIB, there are no dependencies or pkg installation requirement. However, a modification of the cloud-init default configuration file is required.

1. Download the cloud-init-config element file - **openstack-dib-elements-main-cloud-init-config.zip**, and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
post-install.d/50-cloud-init-config	A script for modifying the cloud-init default configuration to allow creation of an admin user named "stack", with password-based SSH access to an instance created with this guest image.

> **Note**
>
> Creating a user with password-based SSH access to the instance is a potential security risk, and is provided for convenience only. In a production environment, it is highly recommended to use SSH keys for users authentication.

2. Generate a new "salted" hash for the desired password using the following command and populate the "passwd" value in the 50-cloud-init-config file to include the new password hash.

> **Note**
>
> - Remember to escape the special characters when you edit the file.
> - The example element files contain the hash for the password secret: "stack".

```
# perl -e 'print crypt("<your secret>","\$6\$<your salt>\$") . "\r
```

3. Place the files under the /home/diskimage-builder/elements directory, and make sure the script has executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/cloud-init-config/pos
```

CUDA Driver Installation Element

✓ Note

This element requires:

- The target guest kernel release to be identical to the build host kernel release (the installation would otherwise fail).
- CUDA-Enabled NVIDIA GPU device on the build host (the installation would otherwise fail). For the Cuda-Enabled product list, please see **here**.
- CUDA repository installed on the build host.
- For GPUDirect use case, NVIDIA MLNX_OFED network driver should be installed on the target image. Make sure to always use this element with mofed element in the build command in case you plan to use GPUDirect.
- nvidia-peermem kernel module which is required for GPUDirect is installed as part of the CUDA installation.

The "cuda" element is used for CUDA libraries, drivers and toolkit.

1. Download the cuda element file - **openstack-dib-elements-main-cuda.zip**, and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
element-deps	Element dependencies.
package-installs.yaml	A list of packages required for element installation.
pkg-map	Mapping of the package list to RedHat OS distribution packages.

File	Description
post-install.d/05-cuda-install	A script for downloading CUDA run installer file and installing CUDA drivers and toolkit.

2. Place the files under /home/diskimage-builder/elements directory, and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/cuda/post-install.d/0
```

3. Install the RHEL8 CUDA repository on the build host.

```
#  dnf config-manager --add-repo https://developer.download.nvidia
```

4. Set the environment variables with the CUDA repository location on the build host and the URL for downloading the required CUDA run file installer.

```
# export DIB_YUM_REPO_CONF="/etc/yum.repos.d/Cent* /etc/yum.repos.
# export DIB_CUDA_URL=https://developer.download.nvidia.com/comput
```

GPUDirect Benchmark Installation Element

✓ Note

This element requires:

- The target guest kernel release to be identical to the build host kernel release (installation would otherwise fail).
- CUDA driver installed on the target image. Make sure to always use this element with cuda element.
- Usage of DIB_CUDA_PATH environment variable as instructed below.

The "gpudirect-bench" element is required for the installation of GPUDirect testing tools and frameworks such as cuda-enabled perftest tools suite.

1. Download the gpudirect-bench element file - **openstack-dib-elements-main-gpudirect-bench.zip**, and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
element-deps	Element dependencies.
package-installs.yaml	A list of packages required for element installation.
pkg-map	Mapping of the package list to RedHat OS distribution packages.
post-install.d/06-gdr-bench-install	A script for installing perftest with CUDA/GPUDirect support.

2. Place the files under the /home/diskimage-builder/elements directory, and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/gpudirect-bench/post-
```

3. Set the environment variable with the CUDA files location on the target build image. The CUDA version should match the one installed by the "cuda" element in previous steps.

```
# export DIB_CUDA_PATH=/usr/local/cuda-11.6
```

Setting DIB Pre-Build Environment Variables

Set the following environment variables before applying the DIB image creation command.

Variable	Description
ELEMENTS_PATH	Location of the custom elements used in the build command.
DIB_MOFED_FILE	Location of the MLNX_OFED ISO file, required for the mofed element.
DIB_YUM_REPO_CONF	Location of a custom repository on the build host to be used during the build process. CUDA repository is required for the cuda element.
DIB_CUDA_URL	CUDA installer run file download path, required for the cuda element.
DIB_CUDA_PATH	The location of CUDA binaries on the target image, required for GPUDirect Benchmark element.
DIB_MODPROBE_BLACKLIST	Kernel modules to add to the blacklist during the build process.
DIB_CLOUD_INIT_DATASOURCES	Cloud-init datasource type for cloud-init-datasources element used in the build command.
DIB_DHCP_TIMEOUT	DHCP timeout value, for dhcp-all-interfaces element used in the build command.

```
# export ELEMENTS_PATH=/home/diskimage-builder/elements
# export DIB_MOFED_FILE="/tmp/MLNX_OFED_LINUX-5.5-1.0.3.2-rhel8.5-x
# export DIB_YUM_REPO_CONF="/etc/yum.repos.d/Cent* /etc/yum.repos.d/cud
```

```
# export DIB_CUDA_URL=https://developer.download.nvidia.com/compute/cud
# export DIB_CUDA_PATH=/usr/local/cuda-11.6
# export DIB_MODPROBE_BLACKLIST=”nouveau”
# export DIB_CLOUD_INIT_DATASOURCES="OpenStack"
# export DIB_DHCP_TIMEOUT=30
```

Running an Image Build Command with Custom Elements

✓ Note

- It is possible to build a target image only with a mofed element or a cloud-Init-config element, without cuda or gpudirect-bench elements.
- The mofed element is a prerequisite for the cuda element.
- The cuda element is a prerequisite for the gpudirect-bench element, and requires a CUDA-Enabled GPU device on the Build Host.
- The cuda and gpudirect-bench elements require identical target guest kernel release and the build host kernel release.

Generally, the image creation command includes mandatory and optional DIB native elements in addition to user custom elements, and are selected per required use case or build purpose.

For the specific use case described in this article, the elements below are included in the build command.

- DIB Elements:
  - **vm**
  - **dhcp-all-interfaces**
  - **cloud-init-datasources**
  - **dracut-regenerate**
  - **growroot**
  - **epel**
  - **centos**
  - **block-device-efi**

For further details and a full DIB elements list, refer to the following **link**.

- Custom Elements:
  - **mofed**: the NVIDIA Network Driver Installation Element described above.
  - **cloud-init-config**: the Cloud-init Configuration Element described above.
  - **cuda**: the CUDA Driver Installation Element described above.
  - **gpudirect-bench**: the GPUDirect Benchmark Installation Element described above.

Run the build command:

```
# disk-image-create vm dhcp-all-interfaces cloud-init-datasources cloud
```

Upon a successful completion of the build process, the **centos8-nvidia.qcow2** image file will be generated in the /home/diskimage-builder/ directory.

Uploading the Image to the OpenStack Cloud and Spawning an Instance

1. Copy the **centos8-nvidia.qcow2** image file into the U ndecloud node, and issue the following command to upload it into the Overcloud image store:

   ```
   # source overcloudrc
   # openstack image create centos8-nvidia --public --disk-format qcc
   ```

2. Create a cloud instance using the guest image that was built.

   ```
   # openstack server create --image centos8-nvidia --flavor <my_flav
   ```

3. As the custom cloud-init element was used to create a user and allow password-based SSH, you can now log into the new instance using the "stack" user and the password configured in the cloud-init element.

   ```
   # ssh stack@my_instance1
   ```

   > ✓ **Note**
   >
   > This article does not cover the methods used for configuring the instance network connectivity.

4. Once logged into the instance, verify that the custom components are installed.

```
my_instance1# ofed_info -s
MLNX_OFED_LINUX-5.5-1.0.3.2:

my_instance1# cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX x86_64 Kernel Module  510.39.01  Fri Dec
GCC version:  gcc version 8.5.0 20210514 (Red Hat 8.5.0-7) (GCC)

my_instance1# ib_write_bw --help | grep cuda
      --use_cuda=<cuda device id> Use CUDA specific device for GPU
      --use_cuda_bus_id=<cuda full BUS id> Use CUDA specific devic
```

Image Build Procedure For OpenStack BareMetal Cloud Service (Ironic) with MLNX_OFED

✓ Note

This section describes the creation of custom cloud deploy images with the NVIDIA MLNX_OFED element only.

BareMetal provisioning requires "deploy images", which are used by the BareMetal Ironic service to prepare the BareMetal server for guest OS image deployment, as described **here**.

In some cases, it is required to build a custom Ironic deploy image in order to support a new feature.

Follow the instructions below to build **custom Ironic deploy images** with NVIDIA MLNX_OFED network drivers.

1. In addition to the DIB packages described in the previous sections, install the DIB ironic-python-agent-builder on the build host.

```
# pip3 install --user diskimage-builder ironic-python-agent-builde
```

2. Set the custom elements directory to the one that includes the ironic-related elements.

```
# export ELEMENTS_PATH=$HOME/.local/share/ironic-python-agent-buil
```

3. Place the custom NVIDIA Network Driver installation element files - **openstack-dib-elements-main-mofed.zip** under the new element directory, and change the installation scripts permissions.

```
# cd $ELEMENTS_PATH
# cd mofed
# chmod 755 extra-data.d/70-copy-ofed-file
# chmod 755 install.d/70-ofed-install
```

4. As described previously, download the relevant MLNX_OFED ISO file, and export the variable to its location on the build host.

```
# export DIB_MOFED_FILE="/tmp/MLNX_OFED_LINUX-5.5-1.0.3.2-rhel8.5-
```

5. Run the build command with the ironic-python-agent-ramdisk and mofed elements.

```
# disk-image-create ironic-python-agent-ramdisk centos mofed -o ir
```

Upon a successful completion of the build process, two deploy image files will be generated: **ironic-deploy-mofed.kernel** and **ironic-deploy-mofed.initramfs**.

6. Copy the deploy images into an OpenStack Undercloud node, and upload it to the image store for BareMetal Ironic service usage.

```
# source overcloudrc
# openstack image create oc-bm-deploy-kernel-mofed --public --disk
# openstack image create oc-bm-deploy-ram-mofed --public --disk-fo
```

✓ Note

This article does not cover the full procedure for creating BareMetal Cloud instances.

Appendix

Basic DIB Image Build Troubleshooting

1. It is possible to drop to a shell during the image build process either before or after known hook points for debugging and troubleshooting. This is done by setting the "break" environment with the required breakpoint before running the build command.
   To break after a build error, run:

   ```
   # export break=after-error
   ```

   To break before a build pre-install phase, run:

   ```
   # export break=before-pre-install
   ```

2. In order to debug custom elements that use bash scripts as demonstrated in this article:
   - Include the following code section in your script:

     ```
     #!/bin/bash

     if [ ${DIB_DEBUG_TRACE:-0} -gt 0 ]; then
         set -x
     fi
     set -o errexit
     set -o nounset
     set -o pipefail
     ```

   - Enable script bash prints during the build process by setting the DIB_DEBUG_TRACE environment variable before running the build command.

     ```
     # export DIB_DEBUG_TRACE=1
     ```

Additional Elements

Performance Tools Installation Element

The "perf-tools" element contains a set of libraries and tools to be used for IP/DPDK/RDMA performance testing .

✓ Note

This element was adjusted to CentOS 8.2 OS

1. Download the perf-tools element file - **openstack-dib-elements-main-perf-tools.zip**. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
element-deps	Element dependencies.
package-installs.yaml	A list of packages required for element installation.
pkg-map	Mapping of the package list to RedHat OS distribution packages.
post-install.d/09-perf-tools-install	A script for installing Trex traffic generator, DPDK 20.11, perftest tools, iperf3. In addition, the script will set the hugepage required for the DPDK.

2. Place the files under the /home/diskimage-builder/elements directory, and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/perf-tools/post-insta
```

3. Execute the image build command:

```
# disk-image-create vm dhcp-all-interfaces cloud-init-datasources
```

DHClient IPoIB Configuration Element for CentOS 7

The "dhclient-hw" element is required for adjusting the dhclient.conf file on the CentOS 7 OS images to support IPoIB OpenStack deployments.

✓ Note

- This element was adjusted to CentOS 7.9 OS.
- There is no need of this element for CentOS 8 guest image IPoIB OpenStack deployments.
- This element is required for both guest and ironic deploy images.
- NVIDIA MLNX_OFED network driver should be installed on the target image as well to support IPoIB OpenStack deployments of CentOS 7 images.
  - Download the relevant MLNX_OFED for CentOS 7, and use the DIB_MOFED_FILE env to point to this file before running the build command.

1. Download the dhclient-hw element file - **openstack-dib-elements-main-dhclient-hw.zip** . The attachment includes the following files:

File	Description
README.rst	Element description and goal.
post-install.d/60-dhclient-config	A script for setting dhclient.conf on CentOS 7 OS images to support IPoIB OpenStack deployments.

2. Place the files under the /home/diskimage-builder/elements directory, and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/dhclient-hw/post-inst
```

3. Execute the image build commands on CentOS 7 build host to generate guest and deploy images:

```
# disk-image-create vm dhcp-all-interfaces cloud-init-datasources

# disk-image-create ironic-python-agent-ramdisk centos mofed dhcli
```

RDMA-Core Element for IPoIB support on CentOS 8

The rdma-service which was setting ipoib support was deprecated on recent CentOS releases and replaced by rdma-core package.

The "rdma-core" element is a basic element for installing rdma-core pkg allowing native ipoib OS support required for IPoIB OpenStack deployments.

✓ Note

- This element is not required when building an image with NVIDIA MLNX_OFED network drivers.

1. Download the rdma-core element file - **openstack-dib-elements-main-rdma-core.zip** . The attachment includes the following files:

File	Description
README.rst	Element description and goal.
element-deps	Element dependencies.
package-installs.yaml	A list of packages required for element installation.
pkg-map	Mapping of the package list to RedHat and Ubuntu OS distribution packages.

2. Place the files under the /home/diskimage-builder/elements directory
3. Execute the image build command:

```
# disk-image-create vm dhcp-all-interfaces cloud-init-datasources
```

Cloud-init with Network-Configuration-Disabled Element for CentOS 8

✓ Note

Cloud-init is a method for cross-platform cloud instance initialization. For more information on cloud-init, please refer to **Cloud-init Documentation**.

The "cloud-init-net-conf-disabled" element is used for building a guest image with customized cloud-init parameters to be used during instance initialization. In this case, we will use it to make sure network configuration by cloud init is disabled on the image we create. This might be required in systems where Network Manager which is capable of automatic interface configuration is used as the default networking service.

✓ Note

This element is required to support CentOS 8 Openstack IPoIB deployments in order to allow Network Manager to properly configure and bring up IPoIB interfaces.

1. Download the cloud-init-net-conf-disabled element file - **openstack-dib-elements-main-cloud-init-net-conf-disabled.zip** and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
post-install.d/51-cloud-init-no-net	A script for modifying the cloud-init default configuration to disable network configuration by cloud-init and leave it for NetworkManager in systems where it is used

2. Place the files under the /home/diskimage-builder/elements directory, and make sure the script has executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/cloud-init-net-conf-c
```

3. Execute the image build command:

```
# disk-image-create vm dhcp-all-interfaces cloud-init-datasources
```

Ubuntu OS Elements

The elements listed below are adjusted for Ubuntu-based VM images, and should be used on Ubuntu-based build host with python3-diskimage-builder package installed.

NVIDIA MLNX_OFED Element for Ubuntu

> **Note**
>
> This element requires:
>
> - MLNX_OFED ISO file on the build host

The "mofed-ubuntu" element is used for building a guest image with an installed NVIDIA network drivers set, also known as MLNX_OFED, and it is adjusted for Ubuntu 22.04 OS.

1. Download the relevant MLNX_OFED ISO file from the NVIDIA Networking Linux Drivers **Site**. The below is an example of how to download the Debian variant for Ubuntu OS.

```
# cd /tmp/
# wget https://content.mellanox.com/ofed/MLNX_OFED-5.7-1.0.2.0/MLN
```

2. Download the mofed element file - **openstack-dib-elements-main-mofed-ubuntu.zip**, and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
extra-data.d/70-copy-mofed-file	A script for copying the MLNX_OFED ISO image into a DIB environment during the build process. Requires a DIB_OFED_FILE environment variable for pointing the location of the MLNX_OFED ISO file in the build host OS.
install.d/70-ofed-install	A script for installing MLNX_OFED in a DIB environment during the build process with support in guest image kernel release.

3. Place the files under the /home/diskimage-builder/elements directory and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/mofed-ubuntu/extra-da
# chmod 755 /home/diskimage-builder/elements/mofed-ubuntu/install.
```

4. Set the environment variables with the MLNX_OFED ISO file location on the build host.

```
# export DIB_MOFED_FILE="/tmp/MLNX_OFED_LINUX-5.7-1.0.2.0-ubuntu22
```

CUDA Driver Element for Ubuntu

✓ Note

This element requires:

- The target guest kernel release to be identical to the build host kernel release (the installation would otherwise fail).
- CUDA-Enabled NVIDIA GPU device on the build host (the installation would otherwise fail). For the Cuda-Enabled product list, please see **here**.

- For GPUDirect use case, NVIDIA MLNX_OFED network driver should be installed on the target image. Make sure to always use this element with mofed element in the build command in case you plan to use GPUDirect.

The "cuda-ubuntu" element is used for CUDA libraries, drivers and toolkit.

1. Download the cuda element file - **openstack-dib-elements-main-cuda-ubuntu.zip**, and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
post-install.d/05-cuda-install	A script for downloading CUDA run installer file and installing CUDA drivers and toolkit.

2. Place the files under /home/diskimage-builder/elements directory, and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/cuda-ubuntu/post-inst
```

3. Set the environment variables with the URL for downloading the required CUDA keyring deb package.

```
# export DIB_CUDA_URL=https://developer.download.nvidia.com/comput
```

GPUDirect Benchmark Element for Ubuntu

✓ Note

This element requires:

- The target guest kernel release to be identical to the build host kernel release (installation would otherwise fail).

- CUDA driver installed on the target image. Make sure to always use this element with cuda element.
- Usage of DIB_CUDA_PATH environment variable as instructed below.

The "gpudirect-bench" element is required for the installation of GPUDirect testing tools and frameworks such as cuda-enabled perftest tools suite.

1. Download the gpudirect-bench element file - **openstack-dib-elements-main-gpudirect-bench-ubuntu.zip**, and extract it on the build host. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
element-deps	Element dependencies.
package-installs.yaml	A list of packages required for element installation.
pkg-map	Mapping of the package list to RedHat OS distribution packages.
post-install.d/06-gdr-bench-install	A script for installing perftest with CUDA/GPUDirect support.

2. Place the files under the /home/diskimage-builder/elements directory, and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/gpudirect-bench-ubunt
```

3. Set the environment variable with the CUDA files location on the target build image. The CUDA version should match the one installed by the "cuda" element in previous steps.

```
# export DIB_CUDA_PATH=/usr/local/cuda-11.7
```

Performance Tools Element for Ubuntu

> **Note**
>
> This element requires:
>
> - MLNX_OFED Element for Ubuntu
> - Usage of DIB_DPDK_VER and DIB_TREX_VER environment variables as instructed below.

The "perf-tools" element contains a set of libraries and tools to be used for IP/DPDK performance testing.

1. Download the perf-tools element file - **openstack-dib-elements-main-perf-tools-ubuntu.zip**. The attachment includes the following files:

File	Description
README.rst	Element description and goal.
package-installs.yaml	A list of packages required for element installation.
post-install.d/09-perf-tools-install	A script for installing Trex traffic generator, DPDK and DPDK apps such as testpmd, iperf3. In addition, the script will set the hugepage required for the DPDK.

2. Place the files under the /home/diskimage-builder/elements directory, and make sure the scripts have executable permissions.

```
# chmod 755 /home/diskimage-builder/elements/perf-tools-ubuntu/pos
```

3. Set the environment variables with the required DPDK and TREX versions.

```
# export DIB_DPDK_VER=dpdk-21.11
# export DIB_TREX_VER=v2.99
```

Running an Image Build Command with Ubuntu Elements

> **✓ Note**
>
> - It is possible to build a target image only with a mofed-ubuntu element
> - The mofed-ubuntu and cuda-ubuntu elements are a prerequisite for the gpudirect-bench-ubuntu element.
> - The cuda element requires a CUDA-Enabled GPU device on the Build Host.

The procedure below describe the creation of 22.04 Ubuntu-based VM image with the following elements:

- **mofed-ubuntu**
- **cuda-ubuntu**
- **gpudirect-bench**-ubuntu

Set the following environment variables before applying the DIB image creation command.

```
# export ELEMENTS_PATH=/home/diskimage-builder/elements export DIB_RELE
# export ELEMENTS_PATH=/home/ubuntu/openstack-dib-elements
# export DIB_MOFED_FILE="/tmp/MLNX_OFED_LINUX-5.7-1.0.2.0-ubuntu22.04-x
# export DIB_CUDA_URL=https://developer.download.nvidia.com/compute/cud
# export DIB_CUDA_PATH=/usr/local/cuda-11.7
```

Run the build command:

```
# disk-image-create --no-tmpfs vm dhcp-all-interfaces cloud-init-dataso
```

Upon a successful completion of the build process, the **ubuntu-gdr.qcow2** image file will be generated in the /home/diskimage-builder/ directory.

Authors

	Itai Levy Over the past few years, Itai Levy has worked as a Solutions Architect and member of the NVIDIA

Networking “Solutions Labs” team. Itai designs and executes cutting-edge solutions around Cloud Computing, SDN, SDS and Security. His main areas of expertise include NVIDIA BlueField Data Processing Unit (DPU) solutions and accelerated OpenStack/K8s platforms.

Cloud Services | Openstack | Ethernet | Ubuntu | RHEL and CoreOS | GPUDirect RDMA | InfiniBand and RDMA | Red Hat | Canonical

Last updated on Sep 12, 2023