EFLOW

User Guide

DA-09073-001_v12.0 | December 2022

Chapter 1. EFLOW User's Guide

1.1. Setup and Installation

Follow the Microsoft EFLOW documentation page for various installation options suiting your needs:

- For up-to-date installation instructions, visit http://aka.ms/AzEFLOW-install.
- For details on the EFLOW PowerShell API, visit http://aka.ms/AzEFLOW-PowerShell.

For quick setup, we have included the steps for installation through Powershell in the following sections.

1.1.1. Driver Installation

On the target Windows device, first install an NVIDIA GeForce or NVIDIA RTX GPU Windows driver that is compatible with the NVIDIA GPU on your device. EFLOW VM supports deploying containerized CUDA applications and hence only the driver must be installed on the host system. CUDA Toolkit cannot be installed directly within EFLOW.

NVIDIA provided CUDA containers from the NGC registry can be deployed directly. If you are preparing a CUDA docker container, ensure that the necessary toolchains are installed.

Because EFLOW is based on WSL, the restrictions of the software stack for a hybrid Linux on Windows environment apply, and not all of the NVIDIA software stack is supported. Refer to the user's guide of the SDK that you are interested in to determine support.

1.1.2. Installation of EFLOW

In an elevated powershell prompt perform the following:

1. Enable HyperV.

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All

Path          :
Online        : True
RestartNeeded : False
```

2. Set execution policy and verify.

```
Set-ExecutionPolicy -ExecutionPolicy AllSigned -Force

Get-ExecutionPolicy
AllSigned
```

3. Download and install EFLOW.

```
$msiPath = $([io.Path]::Combine($env:TEMP, 'AzureIoTEdge.msi'))
$ProgressPreference = 'SilentlyContinue'
Invoke-WebRequest "https://aka.ms/AzEFLOWMSI_1_4_LTS_X64" -OutFile $msiPath

Start-Process -Wait msiexec -ArgumentList "/i","$([io.Path]::Combine($env:TEMP,
 'AzureIoTEdge.msi'))","/qn"
```

4. Determine Host OS configuration.

```
>Get-EflowHostConfiguration | format-list

FreePhysicalMemoryInMB    : 35502
NumberOfLogicalProcessors : {64, 64}
DiskInfo                  : @{Drive=C:; FreeSizeInGB=798}
GpuInfo                   : @{Count=1; SupportedPassthroughTypes=System.Object[];
 Name=NVIDIA RTX A2000}
```

5. Deploy EFLOW.

   Deploying EFLOW will set up the EFLOW runtime and virtual machine.

   By default, EFLOW only reserves 1024MB of system memory for use for the workloads and that is insufficient to support GPU accelerated configurations. For GPU acceleration, you will have to reserve system memory explicitly at EFLOW deployment; otherwise there will not be sufficient system memory for your containerized applications to run. In order to prevent out of memory errors, reserve memory explicitly as required; see example below. (Refer to command line argument options available for deploying EFLOW in the official documentation for more details).

1.1.3. Prerequisites for CUDA Support

- x86 64-bit support only.
- GeForce RTX GPU products.
- Windows 10/11 (Pro, Enterprise, IoT Enterprise) - Windows 10 users must use the **November 2021 update** build 19044.1620 or higher.
- Deploy-Eflow only allocates 1024 MB memory by default, set it to a larger value to prevent OOM issue, check MS documents for more details at https://learn.microsoft.com/en-us/azure/iot-edge/reference-iot-edge-for-linux-on-windows-functions#deploy-eflow.

Other prerequisites specific to the platform also apply. Refer to aka.ms/AzEFLOW-GPU.

1.2. Connecting to the EFLOW VM

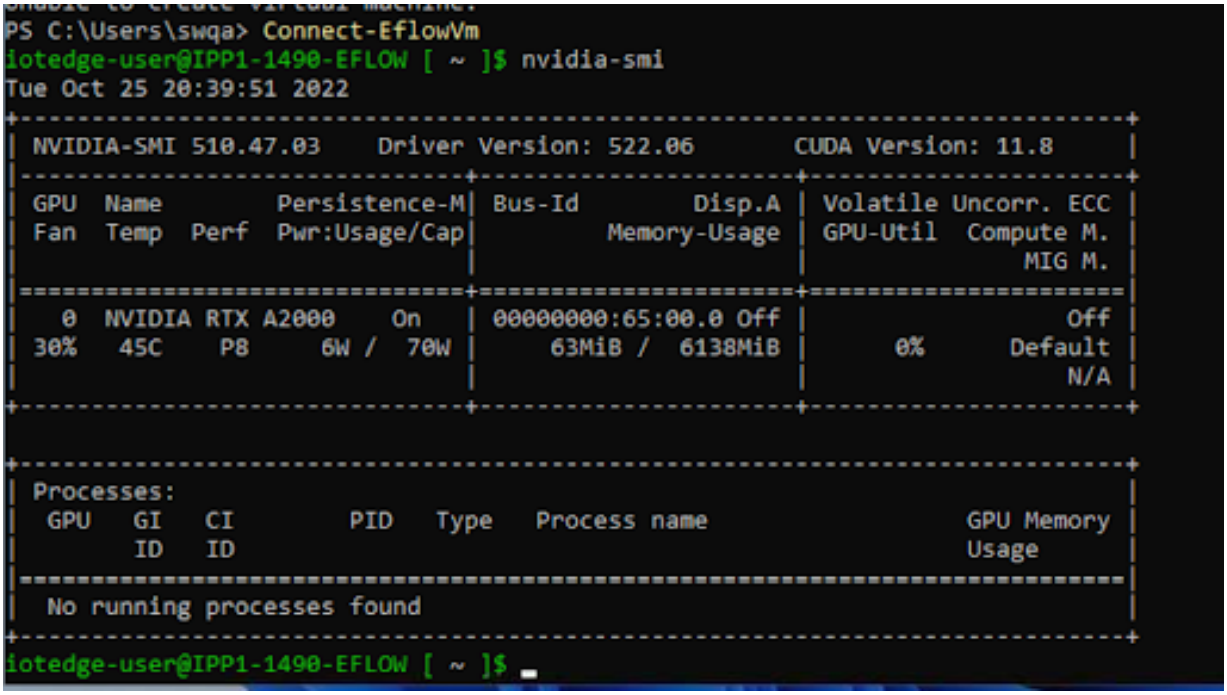
```
Get-EflowVmAddr
```

```
[10/13/2022 11:41:16] Querying IP and MAC addresses from virtual machine (IPP1-1490-EFLOW)

 - Virtual machine MAC: 00:15:5d:b2:40:c7
 - Virtual machine IP : 172.24.14.242 retrieved directly from virtual machine
00:15:5d:b2:40:c7
172.24.14.242

Connect-EflowVm
```

1.3. Running nvidia-smi



1.4. Running GPU-Accelerated Containers

Let's run an N-body simulation containerized CUDA sample from NGC, but this time inside EFLOW.

```
iotedge-user@IPP1-1490-EFLOW [ ~ ]$ sudo docker run --gpus all --env NVIDIA_DISABLE_REQUIRE=1 nvcr.io/nvidia/k8s/cuda-sample:nbody nbody -gpu -benchmark
```

```
Unable to find image 'nvcr.io/nvidia/k8s/cuda-sample:nbody' locally
nbody: Pulling from nvidia/k8s/cuda-sample
22c5ef60a68e: Pull complete
1939e4248814: Pull complete
548afb82c856: Pull complete
a424d45fd86f: Pull complete
207b64ab7ce6: Pull complete
f65423f1b49b: Pull complete
```

```
2b60900a3ea5: Pull complete
e9bff09d04df: Pull complete
edc14edf1b04: Pull complete
1f37f461c076: Pull complete
9026fb14bf88: Pull complete
Digest: sha256:59261e419d6d48a772aad5bb213f9f1588fcdb042b115ceb7166c89a51f03363
Status: Downloaded newer image for nvcr.io/nvidia/k8s/cuda-sample:nbody
Run "nbody -benchmark [-numbodies=<numBodies>]" to measure performance.
        -fullscreen       (run n-body simulation in fullscreen mode)
        -fp64             (use double precision floating point values for
 simulation)
        -hostmem          (stores simulation data in host memory)
        -benchmark        (run benchmark to measure performance)
        -numbodies=<N>    (number of bodies (>= 1) to run in simulation)
        -device=<d>       (where d=0,1,2.... for the CUDA device to use)
        -numdevices=<i>   (where i=(number of CUDA devices > 0) to use for
 simulation)
        -compare          (compares simulation results running once on the default
 GPU and once on the CPU)
        -cpu              (run n-body simulation on the CPU)
        -tipsy=<file.bin> (load a tipsy model file for simulation)

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary
 when GPU Boost is enabled.

> Windowed mode
> Simulation data stored in video memory
> Single precision floating point simulation
> 1 Devices used for simulation
GPU Device 0: "Ampere" with compute capability 8.6

> Compute 8.6 CUDA device: [NVIDIA RTX A2000]
26624 bodies, total time for 10 iterations: 31.984 ms
= 221.625 billion interactions per second
= 4432.503 single-precision GFLOP/s at 20 flops per interaction
iotedge-user@IPP1-1490-EFLOW [ ~ ]$
```

1.5. Troubleshooting

nvidia-container-cli: requirement error: unsatisfied condition: cuda>=11.7", need add "--env NVIDIA_DISABLE_REQUIRE=1"

The CUDA version cannot be determined correctly from the driver on the host when launching the container.

Out of memory

In case of out of memory errors, increase the system memory reserved by EFLOW. Refer to https://learn.microsoft.com/en-us/azure/iot-edge/reference-iot-edge-for-linux-on-windows-functions#deploy-eflow.

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.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA and the NVIDIA logo are trademarks 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

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com