
User Manual for Maxwell-TD and Docker

Release 1.0

ACEM group

Jun 17, 2024

CONTENTS

1	Maxwell-TD and Docker	1
1.1	Introduction to Docker	1
1.2	Step-by-Step installation in Windows	2
2	Common issues	23
2.1	Incompatible GPU drivers/toolkit	23
2.2	Windows Version	26
2.3	Sufficient Memory to upload image	26
2.4	BIOS settings	29
2.5	Change Simulation Parameters	29

MAXWELL-TD AND DOCKER

1.1 Introduction to Docker

Docker is a powerful platform designed for developing, shipping, and running applications using containerization technology. Containers are standardized units that package software and its dependencies, enabling applications to run consistently across various environments. This approach offers several advantages over traditional methods of software deployment and management.

1.1.1 Key Benefits of Docker

1. **Portability and Consistency:** Docker containers are lightweight and isolated, ensuring that applications behave predictably regardless of the environment. This portability simplifies the deployment process and reduces compatibility issues.
2. **Efficiency in Development and Deployment:** Developers can build and ship applications faster with Docker. Containers encapsulate all necessary components, including libraries and dependencies, making it easier to manage and scale applications.
3. **Environment Flexibility:** Docker supports deployment across different environments, whether on-premises, in the cloud, or hybrid setups. This flexibility ensures seamless transitions between development, testing, and production environments.

1.1.2 Why Docker for Running Maxwell-TD?

Maxwell-TD, a Discontinuous Galerkin Time-Domain (DGTD) solver program primarily developed and tested in UNIX environments, benefits significantly from Docker on Windows. Here's why:

1. **UNIX Compatibility:** Maxwell-TD is optimized for UNIX environments, and compiling it for Windows can be challenging and time-consuming due to differing libraries and dependencies.
2. **Pre-built Docker Image:** Docker allows Maxwell-TD to be packaged into a pre-built image that includes all necessary libraries and configurations for UNIX. This image can then be easily deployed and run on Windows systems using Docker Desktop and Windows Subsystem for Linux (WSL).
3. **Simplified Setup:** By leveraging Docker and WSL, users can avoid the complexities of setting up a UNIX-like environment on Windows manually. Docker ensures that Maxwell-TD runs seamlessly without the need for extensive configuration or troubleshooting.
4. **Consistency and Reliability:** Docker's containerization ensures that Maxwell-TD behaves consistently across different Windows machines, maintaining the integrity and reliability of the software application.

In conclusion, Docker provides an efficient solution for running Maxwell-TD on Windows by encapsulating it within a container that is compatible with UNIX environments. This approach streamlines deployment, reduces setup efforts, and enhances overall compatibility and reliability of Maxwell-TD in diverse computing environments.

1.2 Step-by-Step installation in Windows

This guide provides comprehensive instructions for installing and configuring Docker Desktop on a Windows system. It covers activating virtualization in BIOS, installing WSL (Windows Subsystem for Linux), importing Docker images, mounting user data, and running Docker containers. Additionally, it includes tips for running Maxwell-TD.

1.2.1 Activate Virtualization in BIOS

To enable virtualization, you'll need to access the BIOS settings during startup. Here's how:

1. Power on your computer.
2. Keep an eye on the initial boot screen.
3. When you see the screen, press the F2 / F10 / F12 / Esc / Delete / Enter key to enter the BIOS settings.

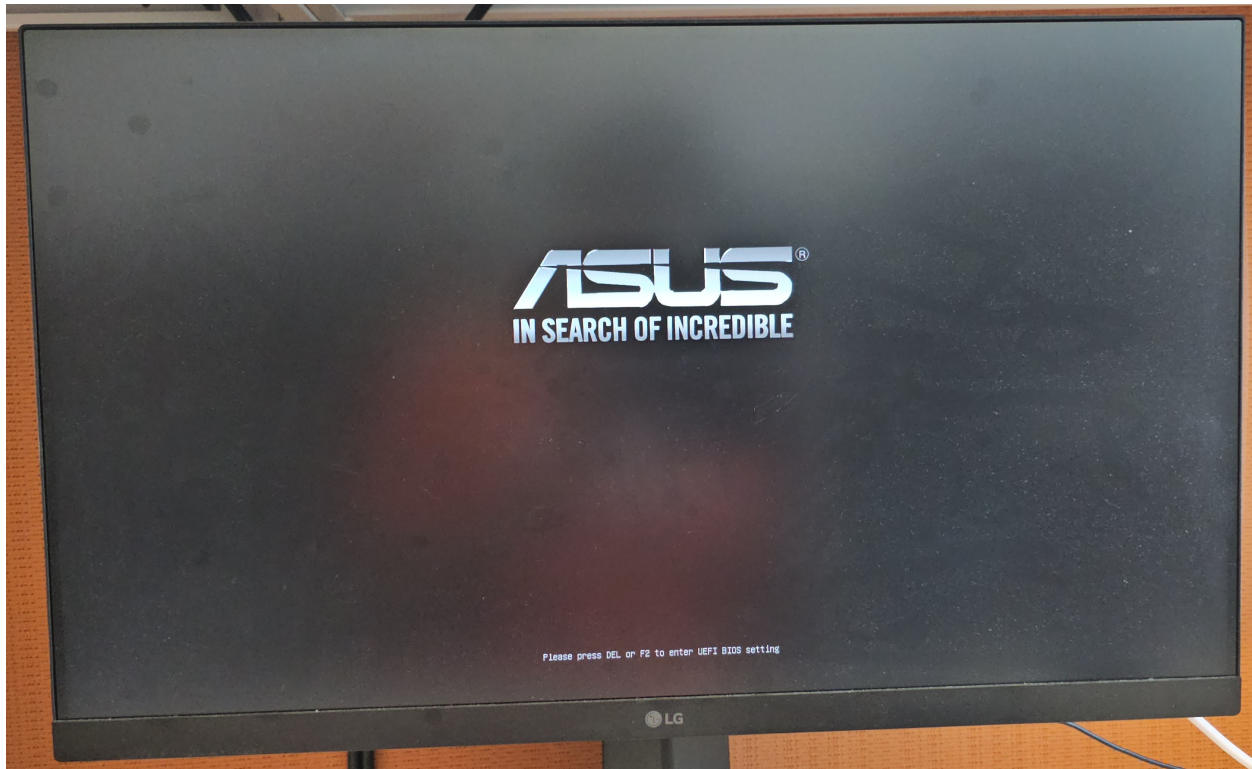


Fig. 1: Screen when we boot up the machine. Press F2 / F10 / F12 / Esc / Delete / Enter

Once inside the BIOS settings:

4. Navigate through the BIOS menus using the arrow keys. Look for the section typically labeled Advanced, Advanced Settings or CPU Configuration.

or

- Alternatively, some BIOS versions allow you to search for settings. Look for a search bar or a similar feature where you can type virtualization or VT-x (for Intel processors) / AMD-V (for AMD processors).



Fig. 2: BIOS settings screen.

Once you locate the virtualization options within these sections, you can proceed to enable them. This step is crucial for utilizing virtualization features such as running virtual machines or other virtualization-based applications on your computer. After making the necessary changes, remember to save your settings and exit the BIOS.

- Change the setting from Disabled to Enabled using the appropriate key (usually Enter or the + key).
- Save changes and exit BIOS

Your computer will restart with virtualization support enabled.

1.2.2 Install WSL

Here are the simplified instructions for installing Windows Subsystem for Linux (WSL) and verifying its installation:

Installing Windows Subsystem for Linux (WSL)

- Open PowerShell as Administrator**
- Install WSL:** Run the following command to install WSL: `wsl --install`. Follow any prompts or confirmations that appear during the installation process.
- Wait for Installation to Complete:** Allow the installation process to finish. This may take some time depending on your internet speed.
- Check WSL Version:** Run the following command to check the installed WSL versions and their respective distributions: `wsl -l -v` This command lists all installed Linux distributions and their associated WSL versions (1 or 2).

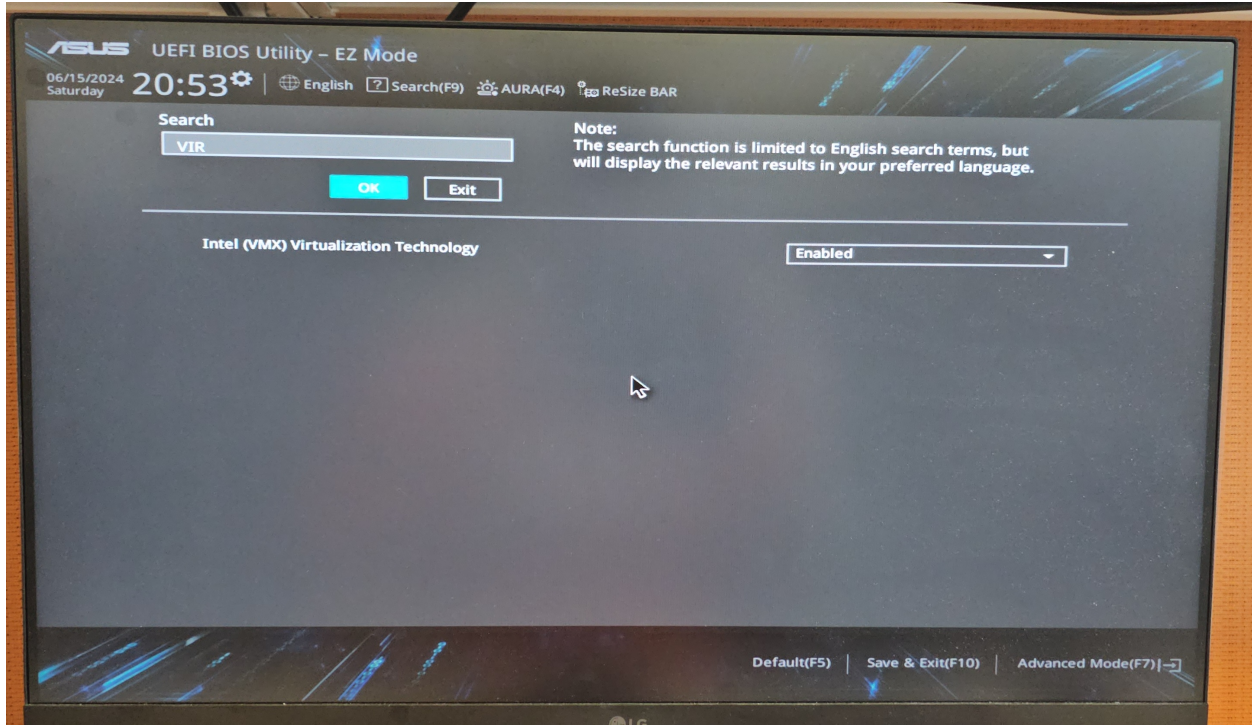


Fig. 3: Turn on Virtualization.

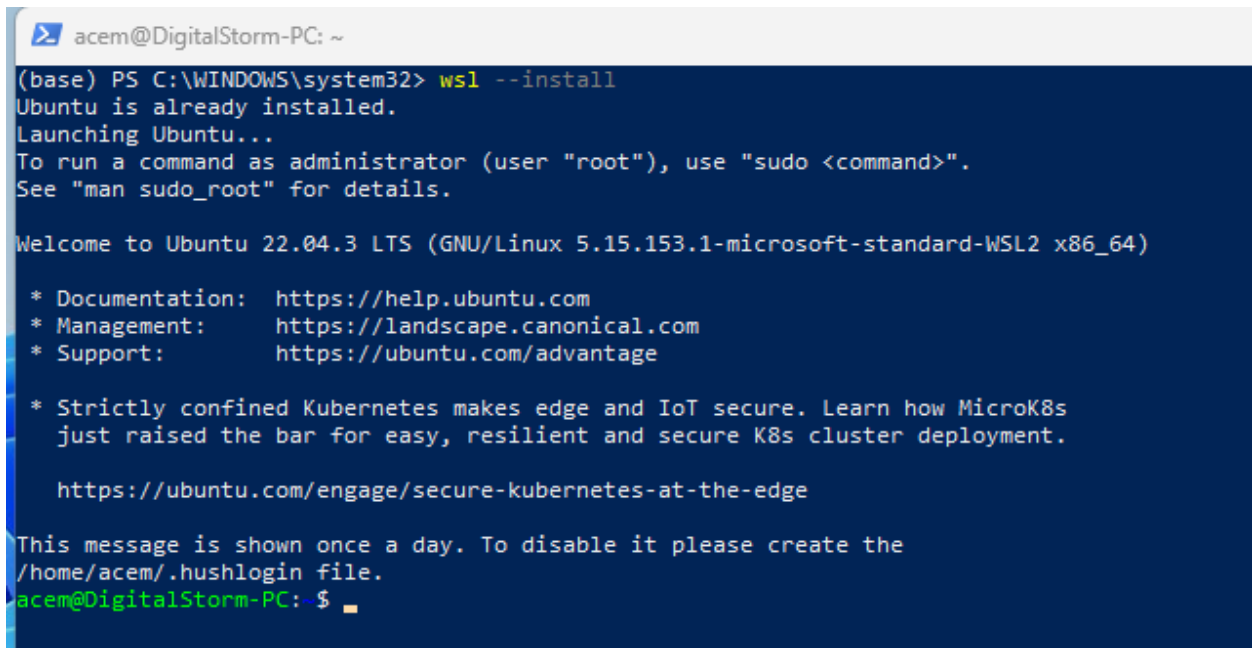
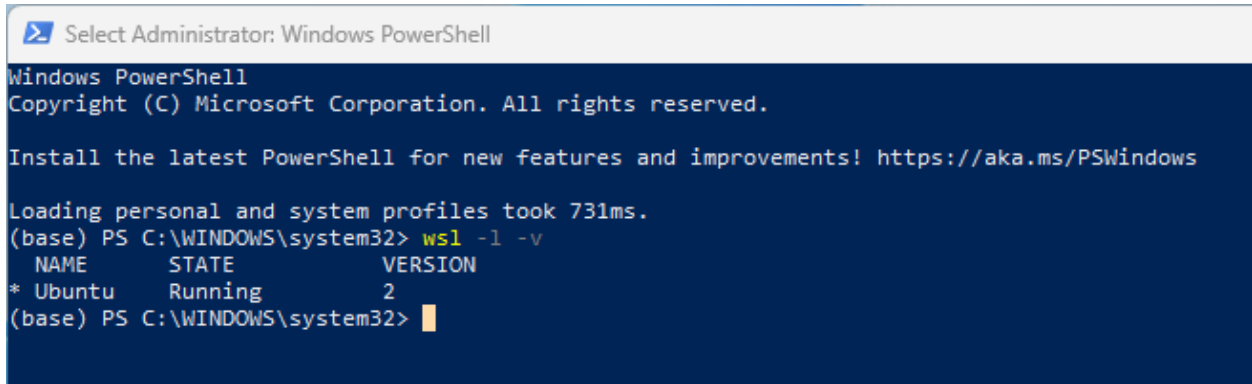


Fig. 4: Install Windows Subsystem for Linux (WSL)



```

Select Administrator: Windows PowerShell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 731ms.
(base) PS C:\WINDOWS\system32> wsl -l -v
  NAME      STATE      VERSION
* Ubuntu    Running    2
(base) PS C:\WINDOWS\system32>

```

Fig. 5: Check if WSL is properly installed

Following these steps ensures that you install Windows Subsystem for Linux (WSL) and confirm its proper installation on your Windows system.

1.2.3 Install Docker Desktop in Windows

1. Download Docker Desktop from Docker's official website (<https://www.docker.com/get-started/>).

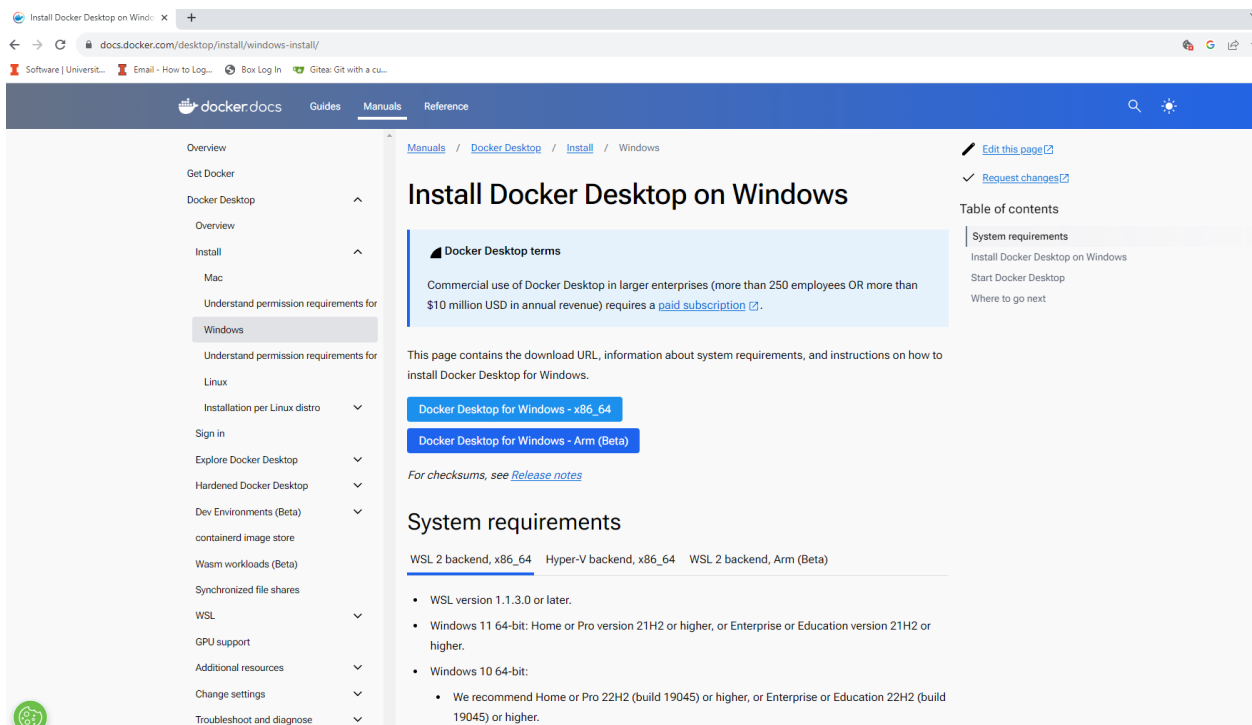


Fig. 6: Download Docker Desktop online

2. Once downloaded, locate the installer file (e.g., DockerDesktopInstaller.exe) and double-click to launch it.
3. Follow the prompts provided by the Docker Desktop installer to complete the installation process. This may involve accepting terms and conditions and choosing installation preferences.


Name	Date modified	Type	Size
▼ Today			
 Docker Desktop Installer (1)	6/15/2024 9:05 PM	Application	487,594 KB

Fig. 7: Docker Desktop Installer

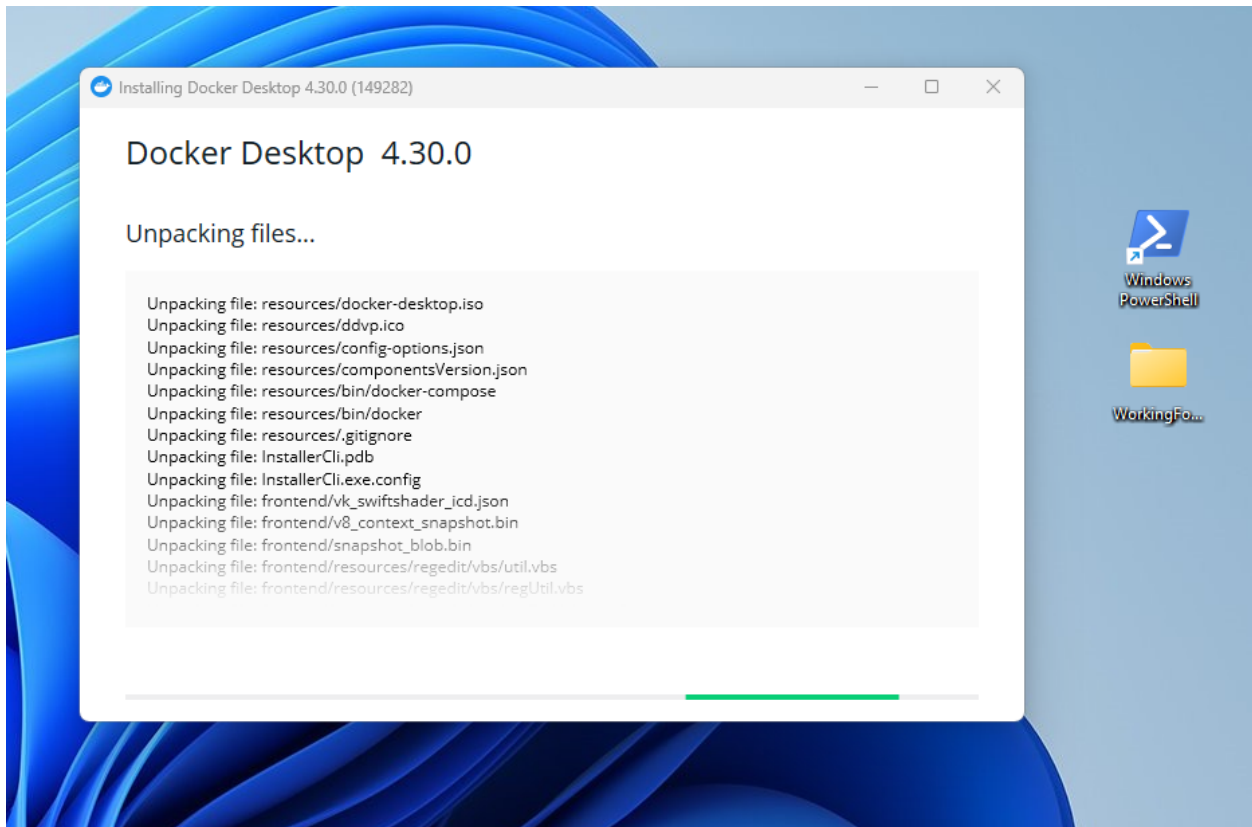


Fig. 8: Installing Docker Desktop

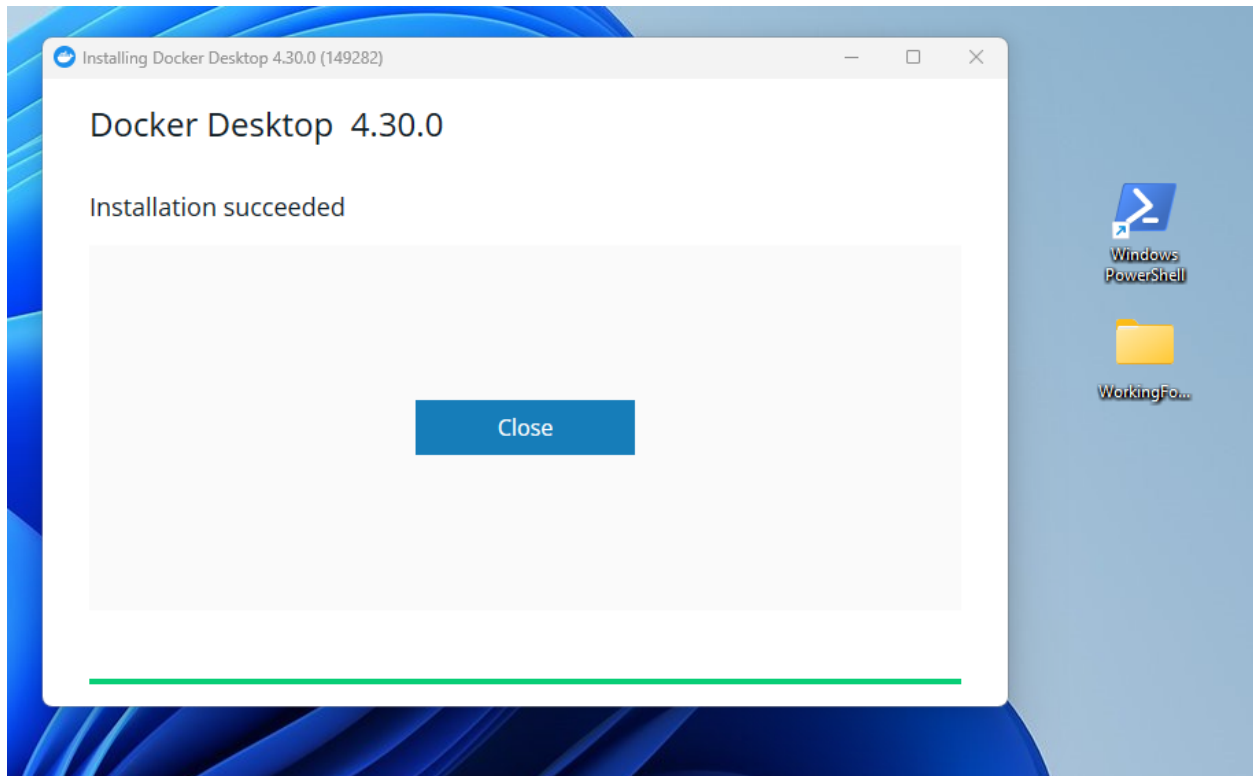


Fig. 9: Finish installing Docker Desktop

4. After installation, a Docker Desktop shortcut will appear on your desktop. The necessary components to run Maxwell-TD Docker images include Docker Desktop itself, Windows PowerShell for running CLI commands, and a designated working folder containing Docker images and user data (e.g., geometry information).
5. Launch Docker Desktop by double-clicking the Docker Desktop shortcut on your desktop.
6. Upon first launch, Docker Desktop may prompt you to accept terms and conditions. Agree to proceed with using the application.
7. You can continue without logging into Docker Desktop if prompted, and optionally skip any survey presented during initial setup.

Currently, the local repository in Docker Desktop is empty. To proceed, we must upload the compiled image for Maxwell-TD. This requires using the PowerShell console, as the GUI does not support managing local Docker images directly and defaults to searching online repositories for Docker images.

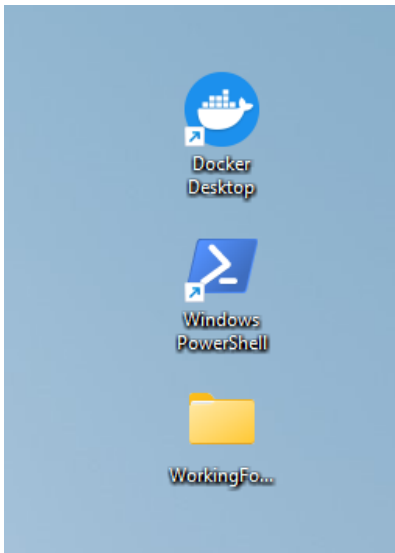


Fig. 10: All the data/folders/apps needed to run Maxwell-TD Docker image

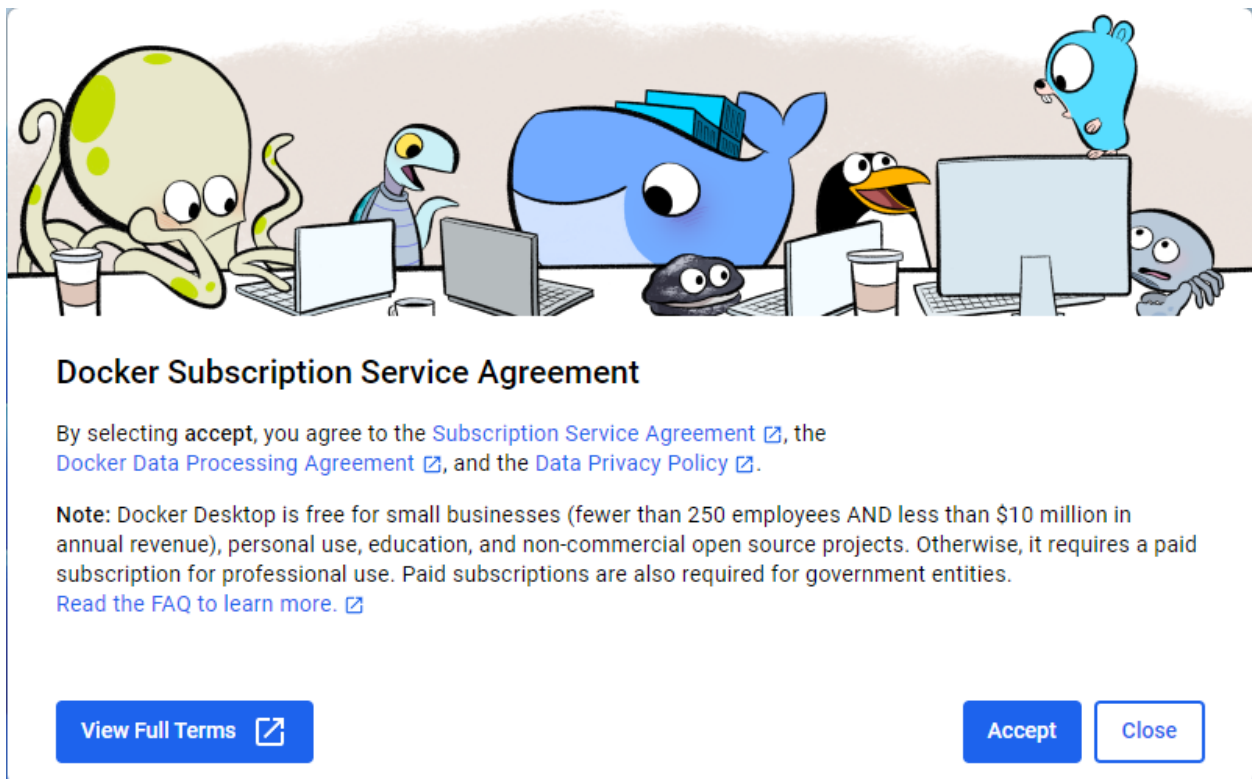


Fig. 11: Accept the terms and conditions

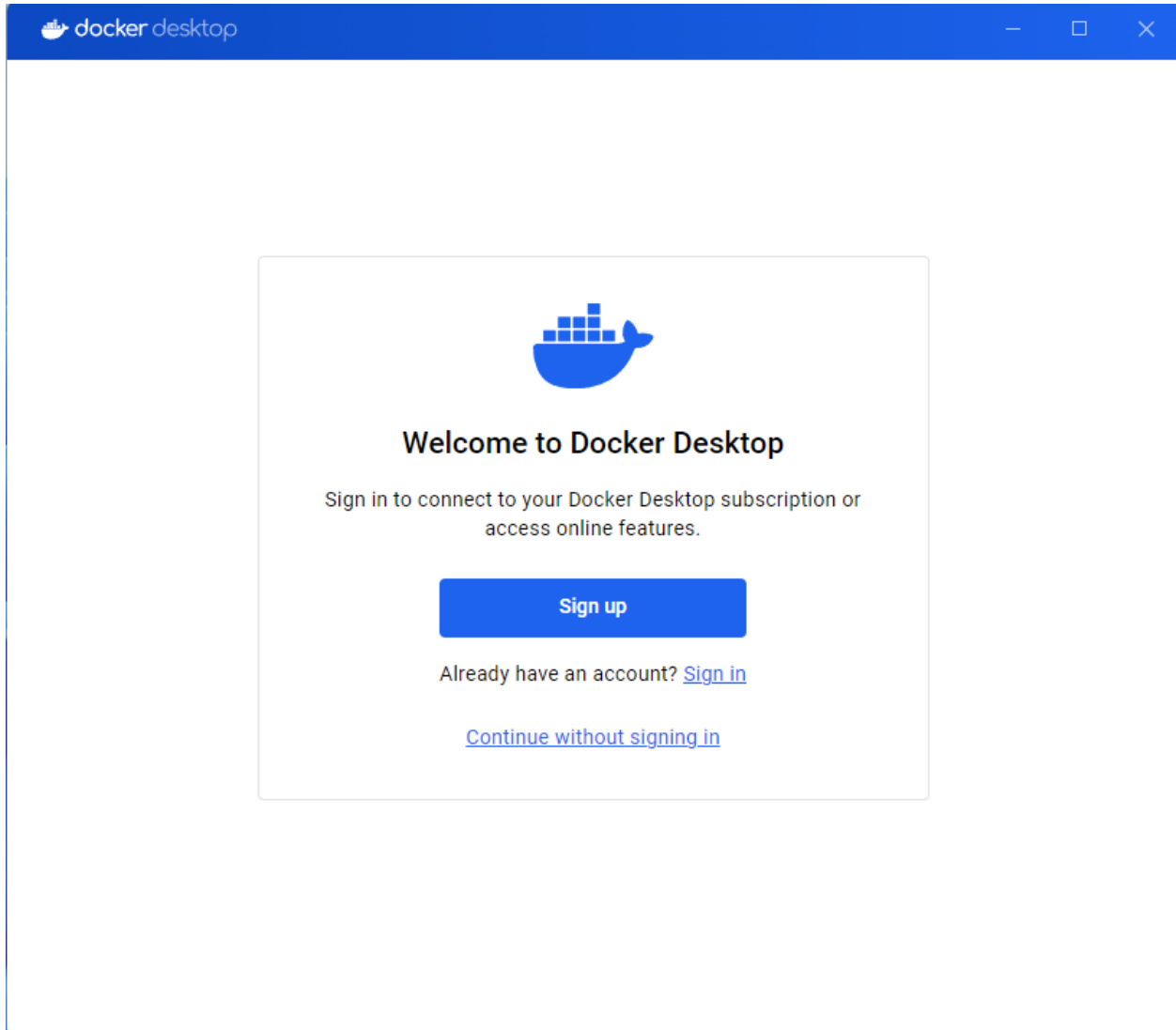


Fig. 12: Continue without logging in

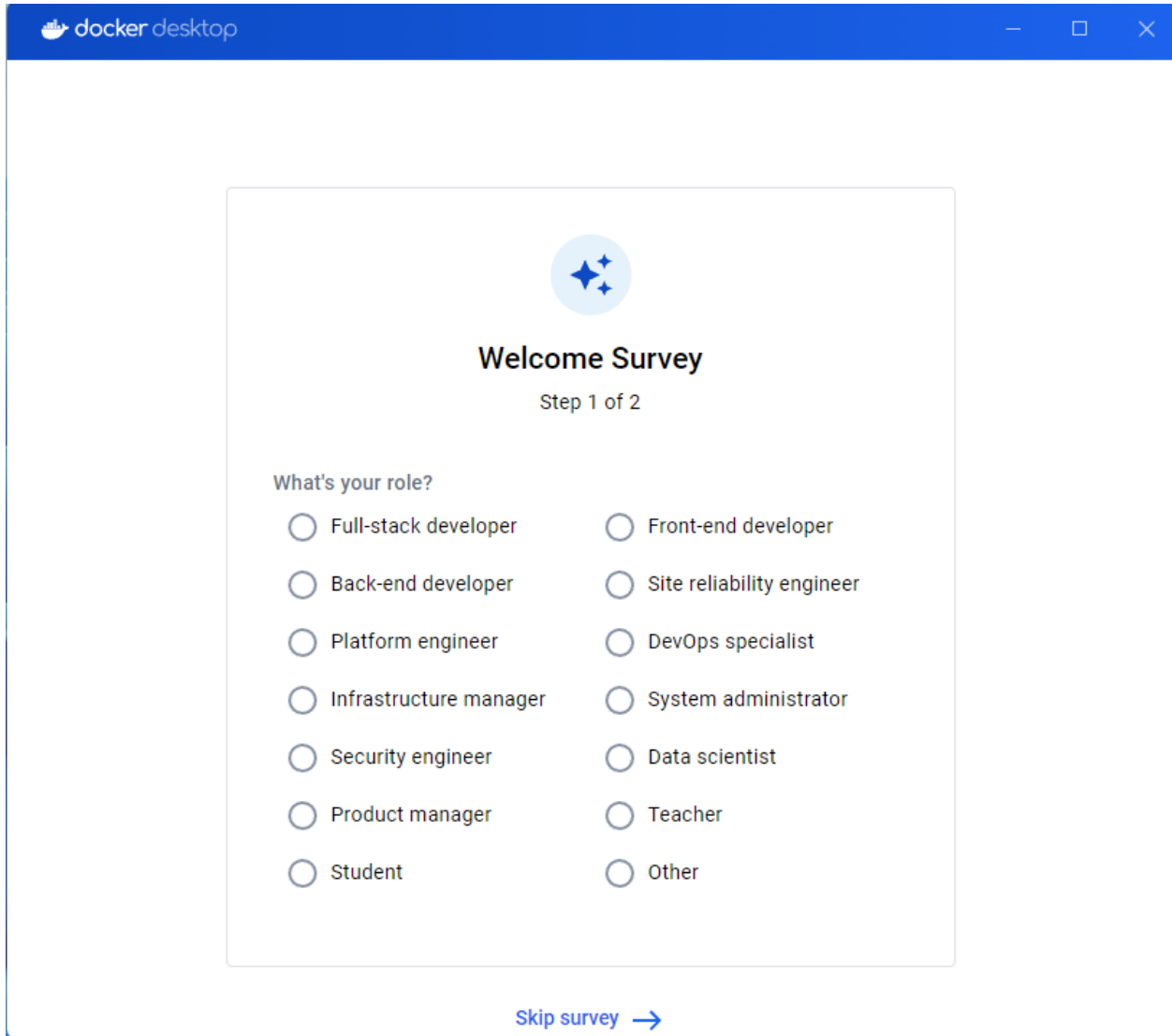


Fig. 13: Skip Survey

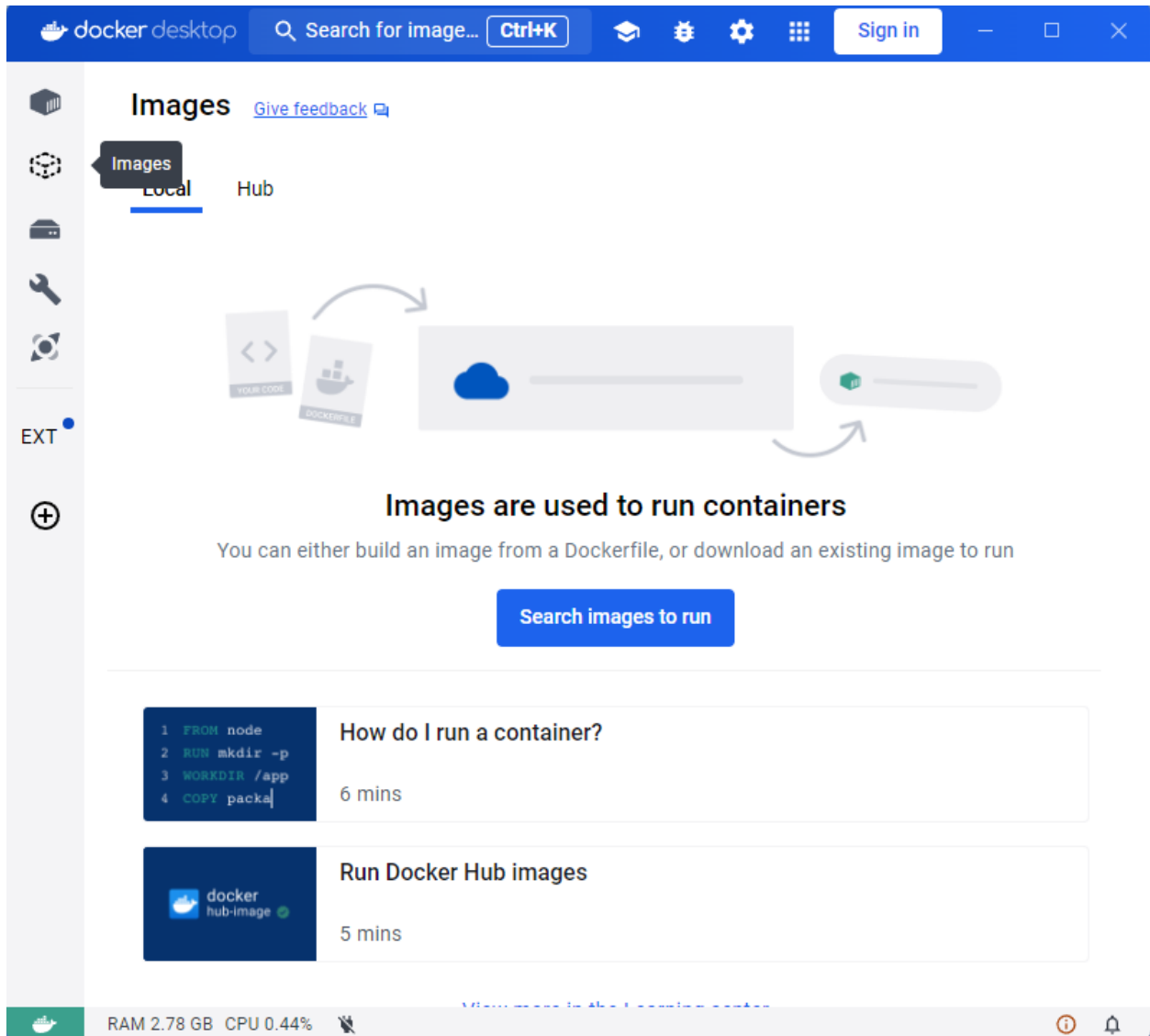


Fig. 14: Image local Repository

1.2.4 Import Docker Image

Next, we need to mount the zipped Docker image onto the Windows system. Since this action cannot be performed using the Docker Desktop GUI, we'll need to use the console terminal instead.

In the designated working folder, you will find a zipped Docker image file and an “examples” folder containing user-defined data.

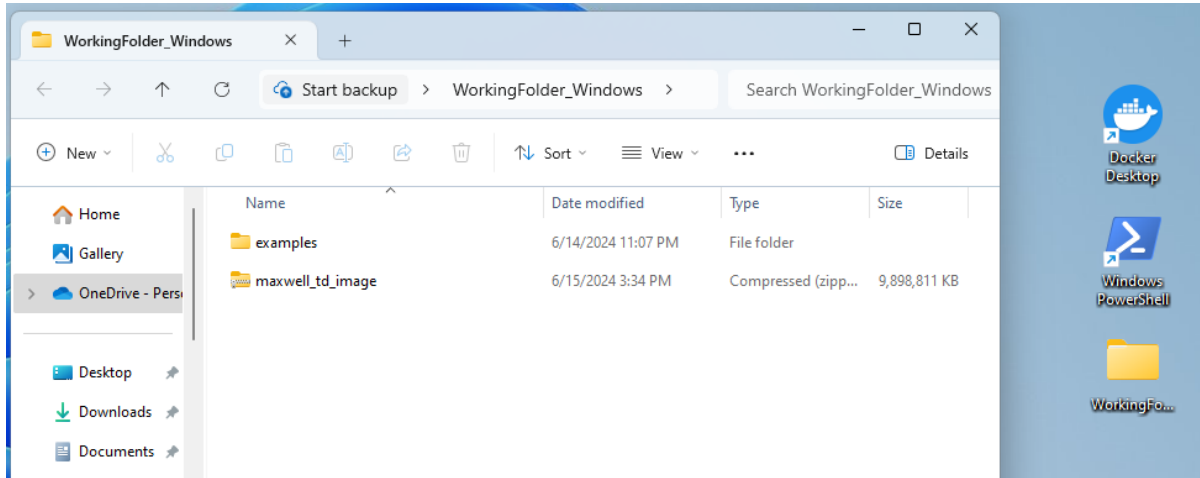


Fig. 15: Location of Maxwell-TD Docker Image (Zipped)

To load the Docker image, execute the command `docker load --input <PATH to zipped docker image>`. This process will take some time, and you can verify its completion by using `docker images` in the console.

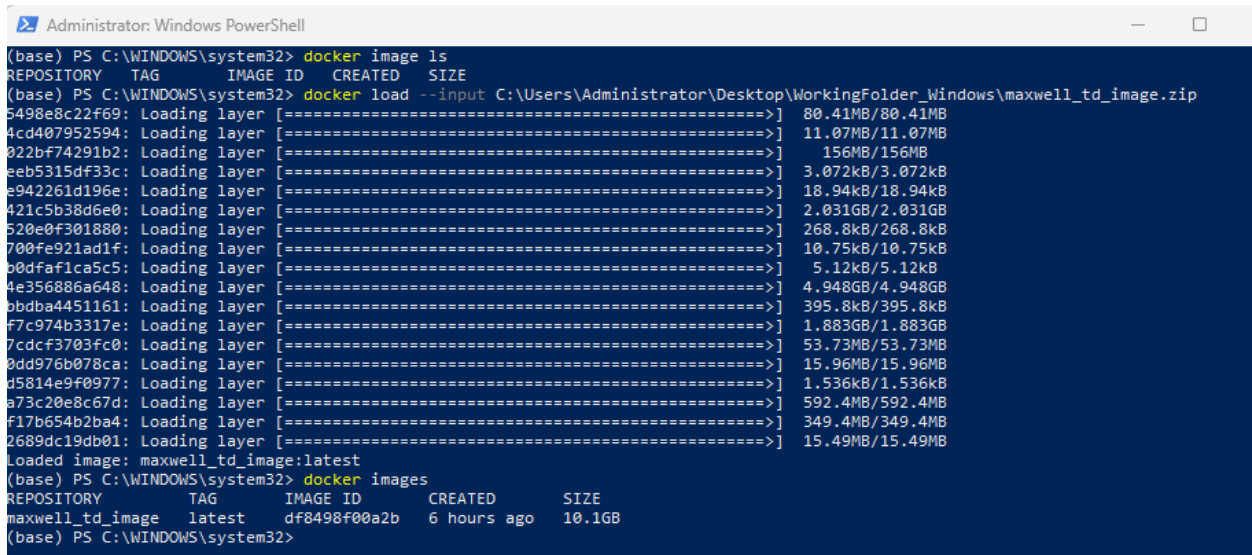


Fig. 16: Load and check Docker Image

The loaded Docker image will also be visible both in Docker Desktop.

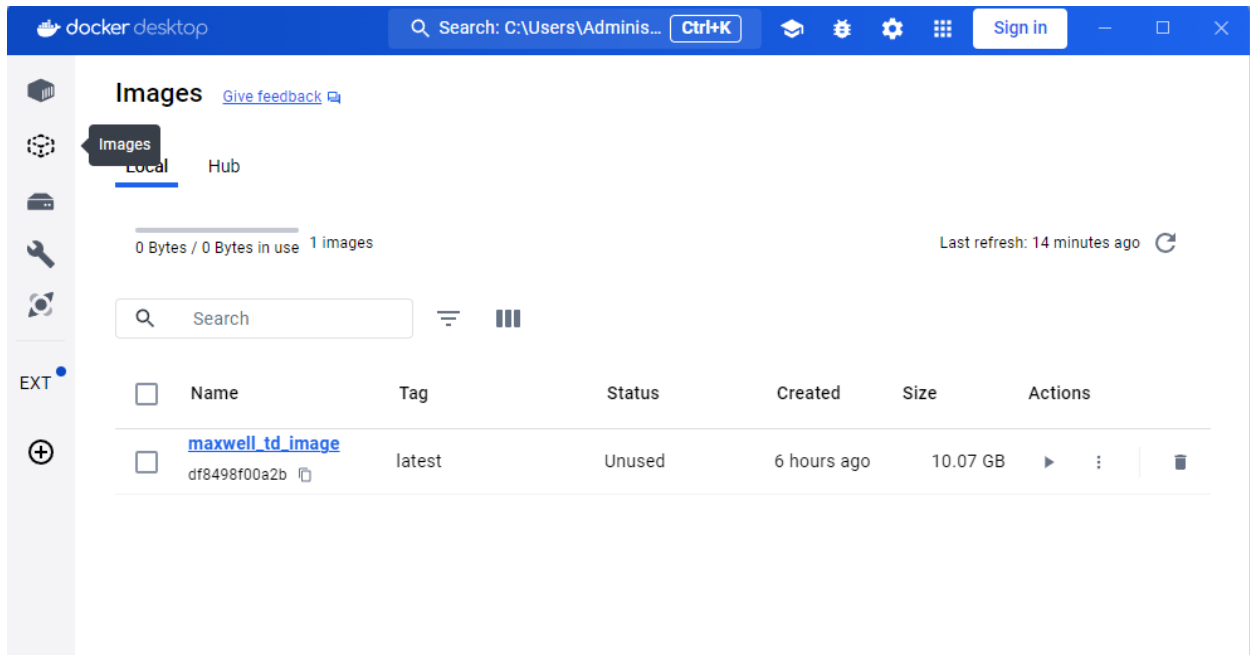


Fig. 17: Loaded Docker Image can be seen in Docker Desktop

1.2.5 Mounting User data and running Docker image

Finally, we are ready to run the Docker image. You can use a PowerShell script with administrative privileges to execute the Docker image. We have prepared a script file named `runDocker.ps1` that can be executed in the console terminal.

To run the script and start the Docker image, follow these steps:

1. **Open PowerShell as Administrator**
2. **Navigate to Script Directory:** Use the `cd` command to navigate to the directory where `runDocker.ps1` is located.
3. **Execute the Script:**
 - Run the script by typing `.\runDocker.ps1` and press `Enter`.
 - The script will mount the folder containing user-defined data into the Docker Server environment.

Note:

- When running the Docker image, any data saved onto the mounted filesystem will persist. However, other modifications, such as custom installations of packages, will be temporary and will disappear after you exit the Docker session.
- Treat the Docker image as a saved state of a customized environment.

Let's break down the lines in the PowerShell script to understand the necessary steps for running the Docker image. This will clarify what is needed for execution.

- **\$imageName:**

This variable defines the name of the Docker image that you want to run. In this case, it is set to `maxwell_td_image`. Users should not change this variable if they intend to run Maxwell-TD.

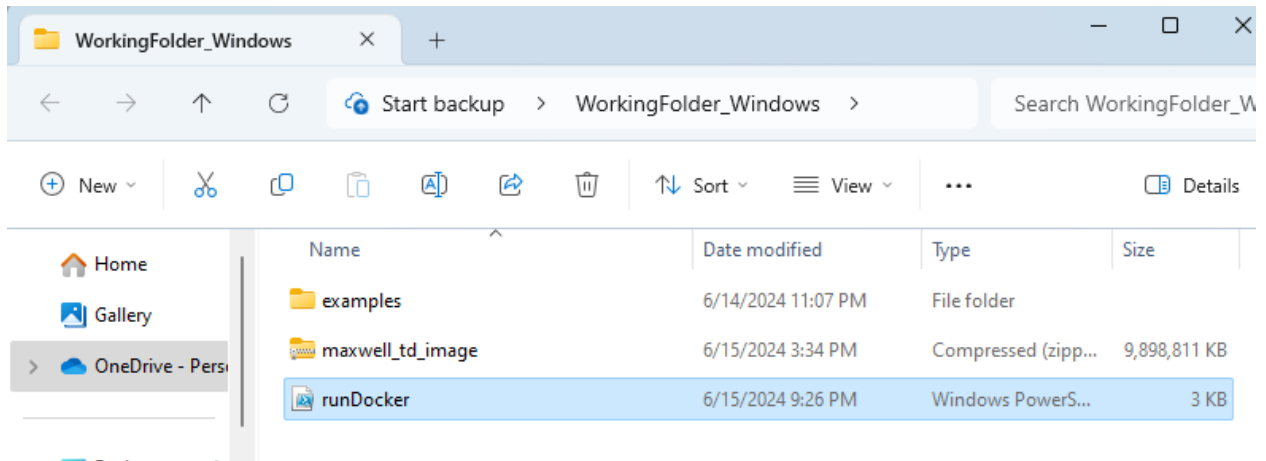


Fig. 18: Use PowerShell Script to run Docker Image

- **\$containerName:**

This variable specifies the name of the container that will run the Docker image. You can run multiple containers from the same image concurrently, so each running instance needs a unique name. Users can customize this variable to suit their naming conventions.

- **\$hostVolumePath:**

This variable holds the path to the working folder on the Windows system. This folder contains essential files such as geometry files for simulation models and scripts for setting simulation parameters. Users must update this path to point to their specific working folder.

- **\$containerVolumePath:**

This variable defines the path within the Docker container (running on Linux environment) where the host volume (**\$hostVolumePath**) will be mounted. In this script, it is set to `/dgtd/model/`. It is recommended that users do not change this variable unless necessary, to ensure compatibility with the Docker image's expected mount point.

```

36 # Set variables
37 $imageName = "maxwell_td_image"
38 $containerName = "maxwell_td_container"
39 $hostVolumePath = "C:\Users\Administrator\Desktop\WorkingFolder_Windows\examples"
40 $containerVolumePath = "/dgtd/model"
41 #$hostRunScriptPath = "C:\Users\Administrator\Desktop\Docker\runCUDA.sh"
42 #$containerRunScriptPath = "/dgtd/runCUDA.sh"

```

Fig. 19: User-defined parameters to change location of User data

Next, the script proceeds to execute several commands to display the paths and names. It also checks the CUDA version installed on the Windows device. Afterward, it runs the Docker command `docker run --rm --gpus all -it --name ${containerName} -v ${hostVolumePath}:${containerVolumePath} ${imageName}`.

Explanation of Docker Command Flags

- `docker run`: This command starts a new Docker container.
- `--rm`: Automatically removes the container when it exits. Useful for temporary containers.
- `--gpus all`: Enables access to all GPUs in the container.
- `-it`: Allocates a pseudo-TTY and keeps stdin open. Allows interactive terminal access.

- `--name ${containerName}`: Assigns a name to the container instance based on the value stored in the `${containerName}` variable.
- `-v ${hostVolumePath}:${containerVolumePath}`: Mounts a volume from the host machine (Windows) into the Docker container. `${hostVolumePath}` is the path on the host system, and `${containerVolumePath}` is the path inside the container where the volume will be mounted.
- `${imageName}`: Specifies the Docker image to use for creating the container.

Additional Information

- The `--rm` flag ensures that the container is automatically removed after it stops running, helping to manage resources efficiently.
- `--gpus all` allows the Docker container to access all available GPUs on the host machine, which is crucial for applications that require GPU acceleration.
- `-it` enables interactive mode with a terminal session, which is useful for interacting directly with the container if needed.

This Docker command, when executed within the PowerShell script, sets up and runs the Docker container named `${containerName}` with GPU support, mounts necessary volumes, and utilizes the specified Docker image `${imageName}` for the Maxwell-TD application. Adjust `${containerName}`, `${hostVolumePath}`, `${containerVolumePath}`, and `${imageName}` as per your specific environment and requirements.

```

46 # Display settings
47 Write-Host "Running Docker container with the following settings:"
48 Write-Host "Image Name: $imageName"
49 Write-Host "Container Name: $containerName"
50 Write-Host "Host Volume Path: $hostVolumePath"
51 Write-Host "Container Volume Path: $containerVolumePath"
52 Write-Host "Host Run Script Path: $hostRunScriptPath"
53 Write-Host "Container Run Script Path: $containerRunScriptPath"
54
55 # Check CUDA driver version on the host
56 Write-Host "Checking CUDA driver version on the host..."
57 $nvidiaSmiOutput = & nvidia-smi
58 $cudaVersion = $nvidiaSmiOutput | Select-String -Pattern "Driver Version" | ForEach-Object { $_ -replace "^.?Driver Version:\s*", "" }
59 Write-Host "CUDA Driver Version: $cudaVersion"
60
61
62 # Construct the Docker run command
63 $dockerRunCommand = @"
64 docker run --rm --gpus all -it --name ${containerName} -v ${hostVolumePath}:${containerVolumePath} ${imageName}
65 "@
66
67 # Echo out the Docker run command for debugging
68 Write-Host "-----"
69 Write-Host "Docker run command:"
70 Write-Host $dockerRunCommand
71 Write-Host "-----"
72
73 # Run Docker container
74 Invoke-Expression nvidia-smi
75 Invoke-Expression $dockerRunCommand
76
77
78 # Check if the container started successfully
79 if ($LASTEXITCODE -ne 0) {
80     Write-Host "Error: Failed to start Docker container" -ForegroundColor Red
81 } else {
82     Write-Host "Docker container ended successfully" -ForegroundColor Green
83 }

```

Fig. 20: Commands to run Docker image

Note: When opening a Windows text file in a Unix environment, extra `\r` characters at the end of each line can cause errors such as `/bin/bash^M: bad interpreter: No such file or directory`. This is because Unix-based systems expect lines to end with `\n` (newline) instead of `\r\n` (carriage return followed by newline) used in Windows.

To fix this issue, you can convert the line endings from Windows format to Unix format using utilities like `dos2unix`, which removes the extra `\r` characters. Alternatively, you can use the included function `Remove-CarriageReturns` in `runDocker.ps1` to process the files. Once corrected, your script should execute correctly within Docker or any Unix-based environment without encountering interpreter errors.

```
1 # PowerShell script to run a Docker container with volume mounting
2
3 function Remove-CarriageReturns {
4     param (
5         [Parameter(Mandatory=$true)]
6         [string]$FilePath
7     )
8
9     # Check if the file exists
10    if (-Not (Test-Path -Path $FilePath)) {
11        Write-Error "The file '$FilePath' does not exist."
12        return
13    }
14
15    try {
16        # Read the content of the file
17        $fileContent = Get-Content -Raw -Path $FilePath
18
19        # Remove carriage return characters
20        $fileContent = $fileContent -replace "`r", ""
21
22        # Write the modified content back to the file
23        Set-Content -Path $FilePath -Value $fileContent
24
25        Write-Output "Carriage return characters removed from '$FilePath'."
26    } catch {
27        Write-Error "An error occurred: $_"
28    }
29 }
30
31 # Process Shell scripts if created in windows
32 # Remove-CarriageReturns -FilePath "./runCUDA.sh"
33 # Remove-CarriageReturns -FilePath "$hostVolumePath/CUDA_LTS_RUN.sh"
34
```

Fig. 21: Miscellaneous function to treat textfiles from Windows System (Optional)

To run the script, navigate to the directory where `runDocker.ps1` is located and execute `./runDocker.ps1`. The script will display the CUDA toolkit version and open a BASH shell in the Docker container with `maxwell1_td_image`. This environment simulates a Linux environment.

```

root@6b377de16a24: /dgt
(base) PS C:\Users\Administrator\Desktop\WorkingFolder_Windows> .\runDocker.ps1
Running Docker container with the following settings:
Image Name: maxwell_td_image
Container Name: maxwell_td_container
Host Volume Path: C:\Users\Administrator\Desktop\WorkingFolder_Windows\examples
Container Volume Path: /dgt/model
Host Run Script Path:
Container Run Script Path:
Checking CUDA driver version on the host...
CUDA Driver Version: 551.78      CUDA Version: 12.4
-----
Docker run command:
docker run --rm --gpus all -it --name maxwell_td_container -v C:\Users\Administrator\Desktop\WorkingFolder_Windows\examples:/dgt/model maxwell_td_image
Sat Jun 15 21:33:28 2024
-----
| NVIDIA-SMI 551.78      Driver Version: 551.78      CUDA Version: 12.4      |
|-----|-----|-----|-----|-----|-----|
| GPU   Name      TCC/WDDM  Bus-Id  Disp.A  Volatile Uncorr. ECC  |
| Fan  Temp  Perf    Pwr:Usage/Cap  Memory-Usage  GPU-Util  Compute M.  |
|-----|-----|-----|-----|-----|-----|
| 0     NVIDIA GeForce RTX 3060  WDDM     00000000:01:00:00  On          2%        Default    |
| 0%    39C    P8      16W / 170W    570MiB / 12288MiB  |
|-----|-----|-----|-----|-----|
-----
Processes:
-----
| GPU   GI   CI   PID  Type  Process name                      GPU Memory |
| ID   ID   ID                    | Usage     |
|-----|-----|-----|-----|-----|
| 0     N/A  N/A   1680  C+G   C:\Windows\System32\dmv.exe       N/A        |
| 0     N/A  N/A   1812  C+G   ...nt.CBS_cw5n1h2txyewy\SearchHost.exe  N/A        |
| 0     N/A  N/A   3664  C+G   ...2txyewy\StartMenuExperienceHost.exe  N/A        |
| 0     N/A  N/A   8820  C+G   ...siveControlPanel\SystemSettings.exe  N/A        |
| 0     N/A  N/A   10600  C+G   C:\Windows\explorer.exe           N/A        |
| 0     N/A  N/A   12832  C+G   ...t.SelfServicePlugin\SelfService.exe  N/A        |
| 0     N/A  N/A   13052  C+G   ...Docker\Frontend\Docker Desktop.exe  N/A        |
| 0     N/A  N/A   13652  C+G   ...5n1h2txyewy\ShellExperienceHost.exe  N/A        |
| 0     N/A  N/A   14036  C+G   ...ekyb3d8bbw\PhoneExperienceHost.exe  N/A        |
| 0     N/A  N/A   15128  C+G   ...on\126.0.2592.56\msedgebview2.exe  N/A        |
| 0     N/A  N/A   15992  C+G   ...les\Microsoft OneDrive\OneDrive.exe  N/A        |
| 0     N/A  N/A   16724  C+G   ...rosoft\Edge\Application\msedge.exe  N/A        |
| 0     N/A  N/A   18184  C+G   ...es (x86)\Dropbox\Client\Dropbox.exe  N/A        |
| 0     N/A  N/A   20096  C+G   ...on\126.0.2592.56\msedgebview2.exe  N/A        |
| 0     N/A  N/A   22688  C+G   ...Programs\Microsoft VS Code\Code.exe  N/A        |
|-----|-----|-----|-----|-----|
=====
== CUDA ==
=====
CUDA Version 12.4.0

Container image Copyright (c) 2016-2023, NVIDIA CORPORATION & AFFILIATES. All rights reserved.

This container image and its contents are governed by the NVIDIA Deep Learning Container License.
By pulling and using the container, you accept the terms and conditions of this license:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license

A copy of this license is made available in this container at /NGC-DL-CONTAINER-LICENSE for your convenience.

root@6b377de16a24:/dgt#

```

Fig. 22: Running PowerShell Script

1.2.6 How to use Maxwell-TD

Upon starting the Docker container, you will be in the /dgttd/ directory, where the Discontinuous Galerkin Time-Domain (DGTd) code is located. The compiled code can be found in the /dgttd/build/ directory, with the executables named maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro and maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro-. User data, mounted from the Windows filesystem, is available in the /dgttd/model/ directory. Before running the simulation, the compiled executable must be copied to the simulation folder (/dgttd/model/MaxwellTD_data/). This can be done by executing the following command:

```
cp /dgttd/build/maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro* /dgttd/model/MaxwellTD_data/
```

```

root@6b377de16a24: /dgttd/model
root@6b377de16a24:/dgttd# ls
CMakeLists.txt  Dockerfile  GPU_N  README.md  build  computes  config  dgttd-performance.hpp  dgttd.cpp  fem  lib  model  requirements.txt  utils
root@6b377de16a24:/dgttd# cd model
root@6b377de16a24:/dgttd/model# ls
CST_data  MaxwellTD_data  compare_maxwelltd_CST.py
root@6b377de16a24:/dgttd/model# cp /dgttd/build/maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro* /dgttd/model/MaxwellTD_data/
root@6b377de16a24:/dgttd/model#
    
```

Fig. 23: Local Directory in Docker Image

Simulation files for the MaxwellTD model are located in the directory /dgttd/model/MaxwellTD_data. Additionally, we have included CST reference data for the same simulation model in the folder CST_data. A Python script named compare_maxwelltd_CST.py has been provided to facilitate extraction and comparison of data between the DGTd simulation and CST data.

```

root@6b377de16a24: /dgttd/model/MaxwellTD_data
root@6b377de16a24:/dgttd/model# ls
CST_data  MaxwellTD_data  compare_maxwelltd_CST.py
root@6b377de16a24:/dgttd/model# cd MaxwellTD_data/
root@6b377de16a24:/dgttd/model/MaxwellTD_data# ls
CUDA_LTS_RUN.sh  cylinder_cr1.75.bc  cylinder_cr1.75.node  cylinder_cr1.75.tetra  maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro-
PROBES          cylinder_cr1.75.bcmmap  cylinder_cr1.75.probe  freespace.m
MTC_LTS        cylinder_cr1.75.g  cylinder_cr1.75.prop  maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro
    
```

Fig. 24: Simulation folder

To run the DGTd simulation, we can navigate to /dgttd/model/MaxwellTD_data/ and run the shell script CUDA_LTS_RUN.sh.

```
./CUDA_LTS_RUN.sh
```

```

root@45b5051f9d01: /dgttd/model/MaxwellTD_data
root@45b5051f9d01:/dgttd/model/MaxwellTD_data# ls
CUDA_LTS_RUN.sh  cylinder_cr1.75.g  freespace.m
PROBES          cylinder_cr1.75.node  maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro
MTC_LTS        cylinder_cr1.75.probe  maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro-
cylinder_cr1.75.bc  cylinder_cr1.75.prop
cylinder_cr1.75.bcmmap  cylinder_cr1.75.tetra
root@45b5051f9d01:/dgttd/model/MaxwellTD_data# ./CUDA_LTS_RUN.sh
    
```

Fig. 25: Running Simulation script

This will initiate the DGTd simulation.

Following the completion of the simulation, you can execute the Python script compare_maxwelltd_CST.py to compare the results with those obtained from the CST simulation.

```

root@45b5051f9d01: /dgttd/model/MaxwellTD_data
17 steps took 467 milliseconds
E field norm^2 0.0000000000000000
-----
17 steps took 467 milliseconds
E field norm^2 0.0000000000000000
-----
17 steps took 447 milliseconds
E field norm^2 0.0000000000000000
-----
17 steps took 453 milliseconds
E field norm^2 0.0000000000000000

```

Fig. 26: Simulating running

```

Final Pade Point 48completed
Process : took 1187 milliseconds
Final Pade Point 49completed
Process : took 1191 milliseconds
Final Pade Point 50completed
Process : took 1183 milliseconds
Final Pade Point 51completed
Process : took 1200 milliseconds
Final Pade Point 52completed
Process : took 1209 milliseconds
Final Pade Point 53completed
Process : took 1205 milliseconds
Final Pade Point 54completed
Process : took 1194 milliseconds
Final Pade Point 55completed
Process : took 1175 milliseconds
Process : took 67268 milliseconds
root@45b5051f9d01: /dgttd/model/MaxwellTD_data#

```

Fig. 27: End of simulation

```
python3 compare_maxwelltd_CST.py
```

```

root@45b5051f9d01: /dgttd/model
root@45b5051f9d01: /dgttd/model/MaxwellTD_data# ls
CUDA_LTS_RUN.sh  cylinder_cr1.75.bc  cylinder_cr1.75.node  cylinder_cr1.75.tetra  maxwelltd_CUDA_LTS_DOUBLEEpre_FLOATpro-
PROBES          cylinder_cr1.75.bcmap  cylinder_cr1.75.probe  freespace.m
VTU_LTS         cylinder_cr1.75.g  cylinder_cr1.75.prop  maxwelltd_CUDA_LTS_DOUBLEEpre_FLOATpro
root@45b5051f9d01: /dgttd/model/MaxwellTD_data# cd ..
root@45b5051f9d01: /dgttd/model# ls
compare_maxwelltd_CST.py
root@45b5051f9d01: /dgttd/model# python3 compare_maxwelltd_CST.py

```

Fig. 28: Running post-processing PYTHON script

COMMON ISSUES

2.1 Incompatible GPU drivers/toolkit

It's essential to verify that your Windows device has the correct CUDA toolkit installed to run Maxwell-TD. You can check the CUDA toolkit version by using 'nvidia-smi' in the console terminal. The minimum required version to run Maxwell-TD is CUDA toolkit 12.4.

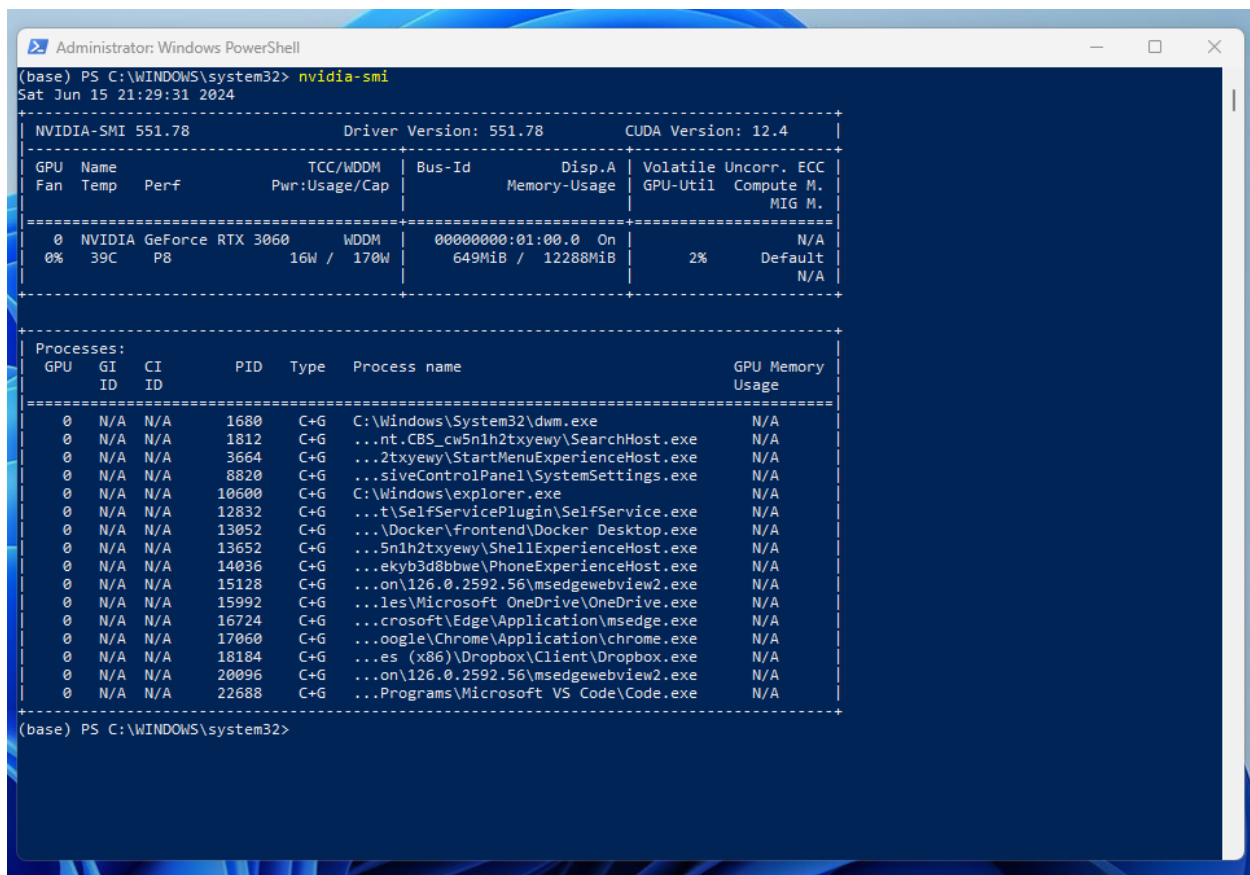


Fig. 1: Check CUDA toolkit version

If you need to update your CUDA toolkit, you can visit the [NVIDIA website \(https://developer.nvidia.com/cuda-toolkit\)](https://developer.nvidia.com/cuda-toolkit). Click on 'NVIDIA CUDA Toolkit Download' to access the latest version. Alternatively, if you require an earlier version, you can navigate to 'Archive of Previous CUDA Releases' on the NVIDIA website to find the specific CUDA toolkit version you need.

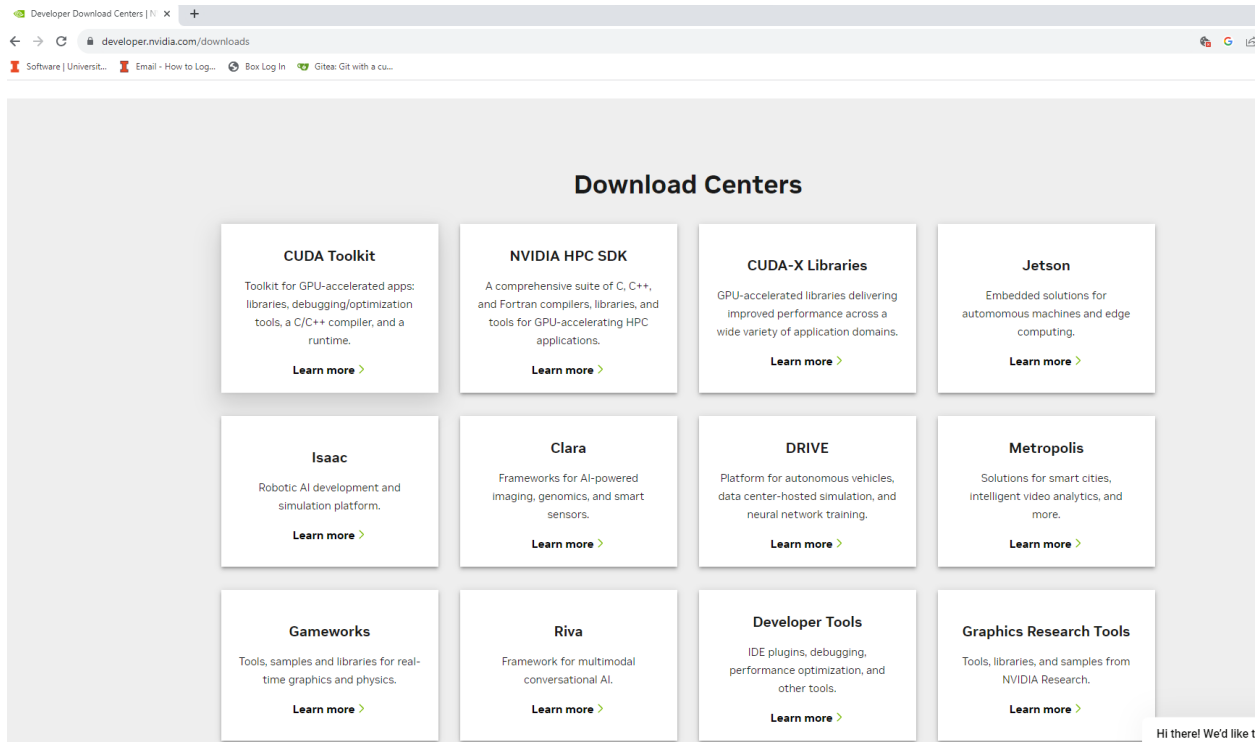


Fig. 2: NVIDIA website to download CUDA toolkit

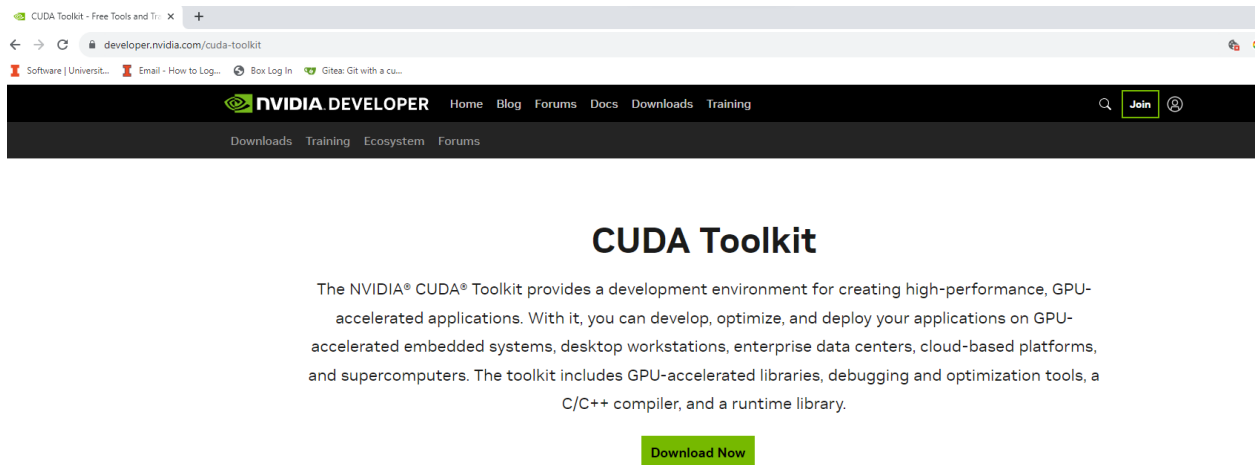


Fig. 3: NVIDIA CUDA toolkit download

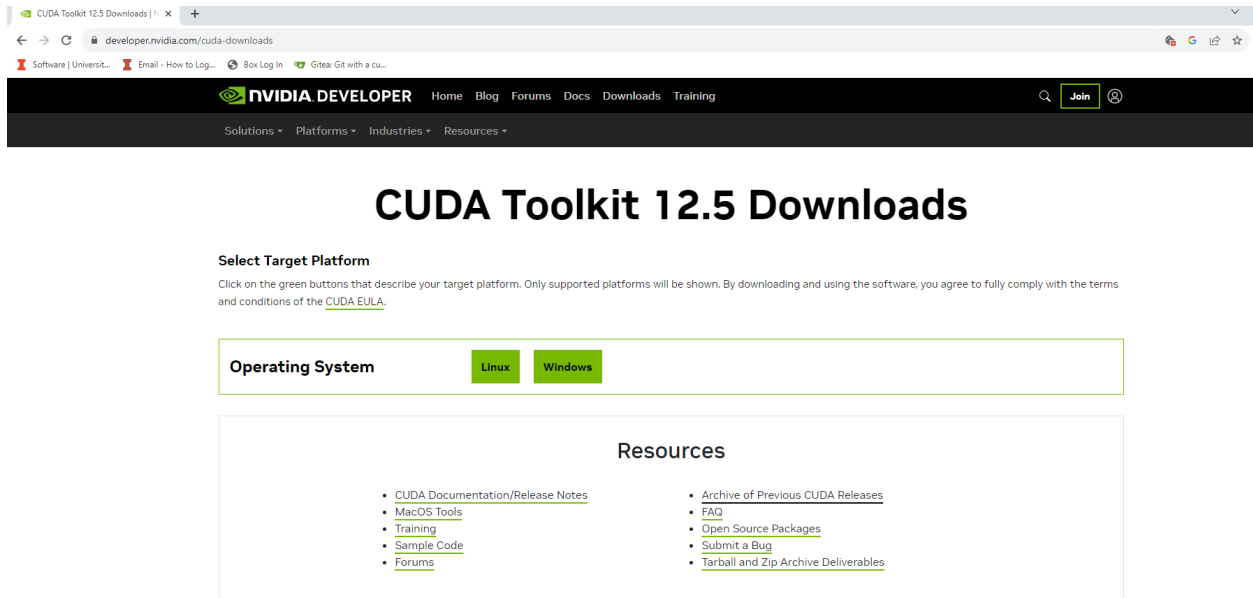


Fig. 4: NVIDIA CUDA toolkit (Latest version)

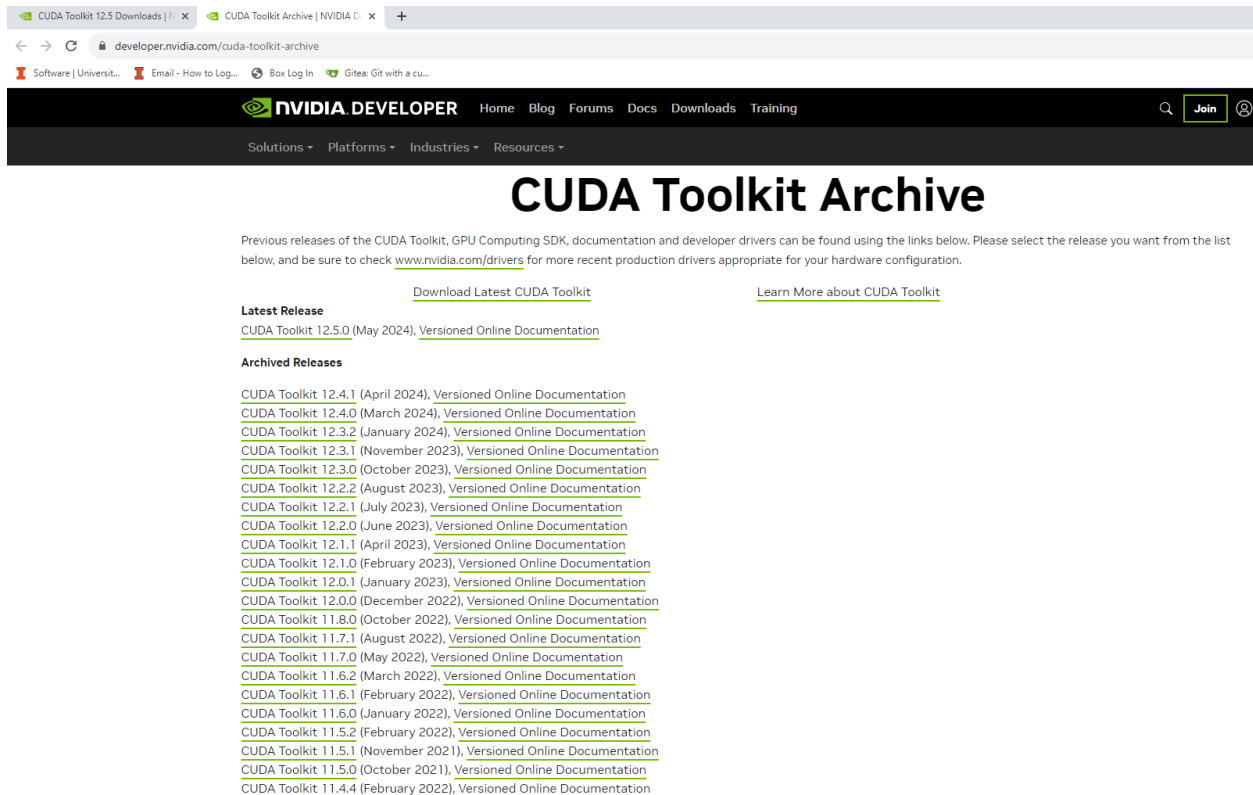


Fig. 5: NVIDIA CUDA toolkit (Earlier versions)

Follow the download and installation instructions provided on the NVIDIA website to update or install the CUDA toolkit on your Windows device. This ensures that Maxwell-TD and other CUDA-dependent applications can function correctly.

2.2 Windows Version

Make sure your Windows 10 or 11 system meets the following requirements:

- **WSL Version:** Ensure WSL version 1.1.3.0 or newer.
- **Windows 11:** 64-bit Home or Pro version 21H2 or later, or Enterprise or Education version 21H2 or later.
- **Windows 10:** 64-bit Home or Pro version 22H2 (build 19045) or later, or Enterprise or Education version 22H2 (build 19045) or later. The minimum supported version is Home or Pro 21H2 (build 19044) or later, or Enterprise or Education 21H2 (build 19044) or later.

Note:

- Docker Desktop on Windows is supported only on versions of Windows that are still actively serviced by Microsoft. It is not compatible with Windows Server versions like Windows Server 2019 or Windows Server 2022.
 - Containers and images created with Docker Desktop are shared across all user accounts on the same machine, as they use a common VM for building and running containers. However, sharing containers and images between user accounts is not supported when using the Docker Desktop WSL 2 backend.
-

2.3 Sufficient Memory to upload image

Make sure your system has adequate storage space for the image. The current image requires 10GB for storage while zipped. Once extracted, it occupies an additional 10GB on your system, although it remains hidden from regular users. You can view these hidden images and containers using Docker Desktop.

Users can remove any unwanted images or containers using the ‘delete’ button.

If necessary, resort to command-line operations to reclaim system resources. Execute these commands in a PowerShell console with administrative privileges.

```
# Purging All Unused or Dangling Resources
docker system prune

# Remove stopped containers and all unused images
docker system prune -a

# Removing Docker Images

## Remove Specific Images
# List all images (including intermediate layers)
docker images -a

# Remove specific image(s) using their ID or tag
docker rmi <image_id_or_name>

## Remove Dangling Images
```

(continues on next page)

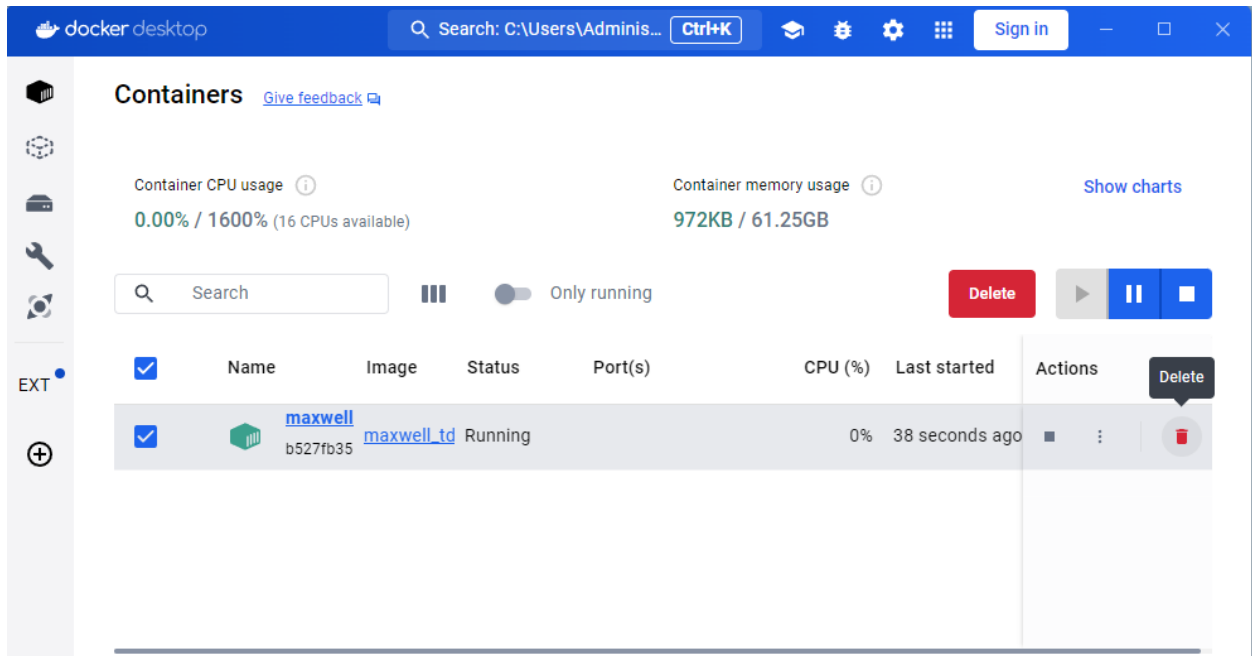


Fig. 6: View all containers or image on Docker Desktop

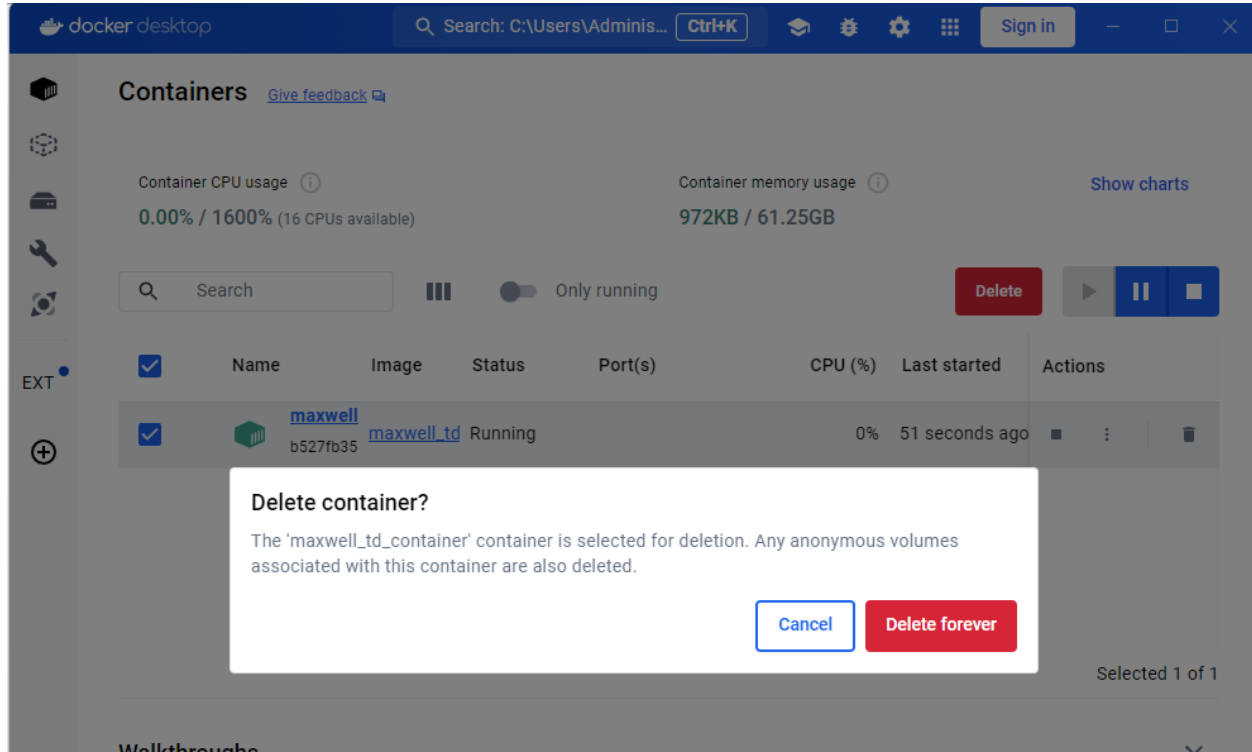


Fig. 7: Delete if not needed to free up space

```
# List dangling images
docker images -f dangling=true

# Remove dangling images
docker image prune

## Remove Images Matching a Pattern
# List images matching a pattern (using grep)
docker images -a | grep "pattern"

# Remove images matching a pattern using awk and xargs
docker images -a | grep "pattern" | awk '{print $1":"$2}' | xargs docker rmi

## Remove All Images
# Remove all Docker images
docker rmi $(docker images -a -q)

# Removing Containers

## Remove Specific Containers
# List all containers (including stopped ones)
docker ps -a

# Remove specific container(s)
docker rm <container_id_or_name>

## Remove All Exited Containers
# List all exited containers
docker ps -a -f status=exited

# Remove all exited containers
docker rm $(docker ps -a -f status=exited -q)

## Remove Containers Matching a Pattern
# List containers matching a pattern (using grep)
docker ps -a | grep "pattern"

# Remove containers matching a pattern using awk and xargs
docker ps -a | grep "pattern" | awk '{print $1}' | xargs docker rm

## Stop and Remove All Containers
# Stop all containers
docker stop $(docker ps -a -q)

# Remove all containers
docker rm $(docker ps -a -q)
```

2.4 BIOS settings

To enable Docker server virtualization on Windows, it's crucial to ensure that virtualization is enabled in your computer's BIOS settings. This setting can only be modified by restarting your machine and accessing the BIOS configuration. Virtualization support in BIOS allows Windows to utilize Docker server features effectively. It's a necessary step to ensure Docker containers run efficiently on your system.

2.5 Change Simulation Parameters

To modify simulation parameters, you can edit the `CUDA_LTS_RUN.sh` shell script. This script contains flags or variables that control various aspects of the DGTD simulation. The script includes comments or descriptions to explain the variables' or flags' purpose.

```

echo "======"
echo "EXECUTABLE"
EXE="./maxwelltd_CUDA_LTS_DOUBLEpre_FLOATpro"
echo $EXE
echo "FileName"
echo "======"

# 0. [EXE]
# 1. Filename: Simulation Filename
# 2. Freq: Central Modulation Frequency (MHz)
# 3. Planewave Waveform (Excitation Type): (0)Time Harmonic (1)Gauss (2)Neumann (3)
↪Modulated Gauss;
# 4. Port Waveform (0) Time Harmonic Pulse (1) Gaussian Pulse
# 5. Tdelay: Delay of excitation pulse (sec)
# 6. Tau: Width of pulse (sec)
# 7. FinalTime: Termination Time (sec)
# 8. Gamma (Penalty) --> 1 upwind , 0 central
# 9. VTU --> 0 (YES), 1 (NO)
#10. Surface Field Output BC Surfaces --> (0) Off (1) PEC only (2) FieldPlane only (3)
↪FieldPlane and PEC
#11. Surface Output Types --> (0) Field (1) Current (2) Field and Current
#12. Poly order --> (1) First Order (2) Second Order
#13. Free Space Comparison(Analytical): (0) True and (1) False
#14. SAMPLINGRATE : parameter of Pade approximation
#15. PADE : (0) Pade mode is Off (2) Pade mode is on

FILENAME=cylinder_cr1.75
FREQ=2650
PLANEWAVEFLAG=1
PORTFLAG=1
TD=2.5e-9
TAU=2.5e-10
FINALTIME=25e-8
GAMMA=0.025
VTU=1
SURFFIELDOUTBCSURFS=0
SURFOUTTYPES=0
POLYORDER=2

```

(continues on next page)

(continued from previous page)

```
ANALYTICAL=1
SAMPLINGRATE=12.5
PADE=2

rm *.log
rm *.vtu
rm AnalyticalIncidentField*
rm Probes_*
rm *.TD*
rm -r ./TimeDomainVoltages
$EXE $FILENAME $FREQ $PLANEWAVEFLAG $PORTFLAG $TD $TAU $FINALTIME $GAMMA $VTU
↳ $SURFFIELDOUTBCSURFS $SURFOUTTYPES $POLYORDER $ANALYTICAL $SAMPLINGRATE $PADE | tee -a
↳ compute.log
rm -r ./VTU_LTS
mkdir ./VTU_LTS
mv *.vtu ./VTU_LTS
rm -r ./PROBES
mkdir ./PROBES
mv *.csv ./PROBES
mv *.TD* ./PROBES
mv *.log ./PROBES
```