

**docker run --gpus all nvidia/cuda:12.1.1-runtime-ubuntu22.04 nvidia-smi**

Unable to find image 'nvidia/cuda:12.1.1-runtime-ubuntu22.04' locally  
12.1.1-runtime-ubuntu22.04: Pulling from nvidia/cuda  
aece8493d397: Pull complete  
dd4939a04761: Pull complete  
b0d7cc89b769: Pull complete  
1532d9024b9c: Pull complete  
04fc8a31fa53: Pull complete  
a14a8a8a6ebc: Pull complete  
7d61afc7a3ac: Pull complete  
8bd2762ffdd9: Pull complete  
2a5ee6fadd42: Pull complete  
Digest: sha256:8bbc6e304b193e84327fa30d93eea70ec0213b808239a46602a919a479a73b12  
Status: Downloaded newer image for nvidia/cuda:12.1.1-runtime-ubuntu22.04  
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error running prestart hook #0: exit status 1, stdout: , stderr: Auto-detected mode as 'legacy'  
nvidia-container-cli: initialization error: load library failed: libnvidia-ml.so.1: cannot open shared object file: no such file or directory: unknown.

The error you are encountering:

nvidia-container-cli: initialization error: load library failed: libnvidia-ml.so.1: cannot open shared object file: no such file or directory

typically means that the NVIDIA driver libraries, specifically libnvidia-ml.so.1, are not accessible or properly mounted inside the Docker container. This library is part of the NVIDIA GPU driver installation on your host, and the Docker container needs access to it to run GPU workloads.

Key reasons and solutions:

1. **NVIDIA Container Toolkit Installation/Version Issue**
   * Make sure you have the NVIDIA Container Toolkit installed and correctly configured on your system, as it manages GPU support for Docker.
   * Sometimes specific versions have bugs; users reported errors with certain versions like 1.14.4. Downgrading to a working version or upgrading to the latest version might help.
2. **Proper Runtime and Device Mounting**
   * The container needs access to NVIDIA device files and libraries from the host. Normally, the NVIDIA Container Toolkit runtime and Docker --gpus flag handle this.
   * If the error persists, try to explicitly pass the devices:

docker run --gpus all --device /dev/nvidiactl --device /dev/nvidia-uvm --device /dev/nvidia0 nvidia/cuda:12.1.1-runtime-ubuntu22.04 nvidia-smi

* + This manually mounts necessary NVIDIA device nodes into the container.

1. **Ensure NVIDIA Driver and Docker Setup Compatibility**
   * The NVIDIA driver on the host must be installed correctly and compatible with the CUDA version in the image.
   * Run nvidia-smi on the host to confirm drivers are working.
   * Docker should not be installed via snap; if so, uninstall and reinstall Docker using the official apt repository as snap packages can cause permission and path issues.
2. **Check Library Symlinks and Paths**
   * Sometimes the symlink for libnvidia-ml.so.1 inside the host /usr/lib or /usr/lib64 is missing or broken.
   * Running ldconfig can fix dynamic linker cache issues.
   * The container runtime relies on these symlinks to find drivers.
3. **Driver Files Not Accessible in Container**
   * The NVIDIA Container Toolkit should mount the driver libs into the container. If the toolkit cannot find the library, it can raise this error.
   * Verify the toolkit installation and configuration by following the official install guide:  
     <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html>
4. **Ubuntu and Kernel Compatibility**
   * Certain Ubuntu versions might require specific NVIDIA runtime versions.
   * Check for known issues with your Ubuntu version (22.04 in your case) and NVIDIA container toolkit releases.

In summary, this error commonly arises due to mismatched or incomplete NVIDIA Container Toolkit installation, missing device mounts, or driver library path issues. The immediate actionable steps are:

* Confirm GPU drivers work on host (nvidia-smi).
* Confirm NVIDIA Container Toolkit is installed and up to date.
* Try to run Docker with explicit NVIDIA device mounting.
* Reinstall Docker via apt if installed with snap.
* Check ldconfig and driver library symlinks.
* Follow official NVIDIA container toolkit install instructions carefully.

If these steps do not resolve the issue, providing the versions of your NVIDIA drivers, Docker, and NVIDIA container toolkit can help in further diagnosis. This issue is well-documented in NVIDIA's forums and GitHub issues for the toolkit. Users who removed Docker installed via snap and installed it via apt have resolved similar errors.

Let me know if you want detailed commands or guidance on any of these steps.

⁂

**docker run --gpus all --device /dev/nvidiactl --device /dev/nvidia-uvm --device /dev/nvidia0 nvidia/cuda:12.1.1-runtime-ubuntu22.04 nvidia-smi**

docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error running prestart hook #0: exit status 1, stdout: , stderr: Auto-detected mode as 'legacy'  
nvidia-container-cli: initialization error: load library failed: libnvidia-ml.so.1: cannot open shared object file: no such file or directory: unknown.

The continued error:

nvidia-container-cli: initialization error: load library failed: libnvidia-ml.so.1: cannot open shared object file: no such file or directory

even with explicit --device flags generally indicates the NVIDIA container runtime cannot find or access the NVIDIA libraries from the host inside the container. Based on reports and troubleshooting from similar cases, here are the most important causes and fixes:

1. **Docker Installed via Snap Package:**
   * If Docker is installed via Snap, it is confined and often cannot access system libraries like NVIDIA driver libraries.
   * Removing the Snap version and installing Docker via official apt repositories resolves the problem for many users.
2. **NVIDIA Container Toolkit Version Bug:**
   * Certain NVIDIA container toolkit versions (like 1.14.4) have introduced regressions causing this error.
   * Downgrading toolkit to a prior stable version (e.g., 1.14.3) or upgrading to a newer fixed release can resolve the issue.
   * Downgrade example:

sudo apt-get install nvidia-container-toolkit=1.14.3-1 \  
 nvidia-container-toolkit-base=1.14.3-1 \  
 libnvidia-container-tools=1.14.3-1 \  
 libnvidia-container1=1.14.3-1

1. **Driver Library Symlinks and ldconfig:**
   * The container runtime expects libnvidia-ml.so.1 symlinked properly in system library paths.
   * Running sudo ldconfig on the host ensures dynamic linker cache is updated.
   * Check for the presence of /usr/lib/x86\_64-linux-gnu/libnvidia-ml.so.1 or equivalent.
2. **Verify NVIDIA Driver and Toolkit Compatibility:**
   * Host NVIDIA driver version must be compatible with CUDA runtime version in the container and the container toolkit.
   * Confirm host driver installed correctly and that nvidia-smi on the host works without issues.
3. **Ensure NVIDIA Container Runtime is Set as Default or Specified Correctly**
   * After installation, confirm you have configured Docker daemon with nvidia runtime or using --gpus all.
   * Check /etc/docker/daemon.json includes at least:

{  
 "default-runtime": "nvidia",  
 "runtimes": {  
 "nvidia": {  
 "path": "nvidia-container-runtime",  
 "runtimeArgs": []  
 }  
 }  
}

* + Restart Docker daemon after any changes.

1. **If Running on WSL or Non-Standard Linux Distributions**
   * Additional configuration or support packages may be required.
   * Check specific distribution compatibility with NVIDIA container toolkit.

**Recommended Troubleshooting Order:**

1. Check how Docker is installed:
   * Remove any Snap Docker installation.
   * Install Docker using the official Ubuntu apt repository.
2. Verify NVIDIA driver installation:
   * Run nvidia-smi on host.
   * Run ldconfig and check library symlinks for libnvidia-ml.so.1.
3. Downgrade or upgrade the NVIDIA container toolkit to a known working version.
4. Verify Docker daemon's runtime configuration for NVIDIA.
5. Retry running:

docker run --gpus all nvidia/cuda:12.1.1-runtime-ubuntu22.04 nvidia-smi

If you provide the output of:

* docker --version
* nvidia-smi
* dpkg -l | grep nvidia-container-toolkit
* cat /etc/docker/daemon.json

I can help further diagnose your setup. This error is a common issue tied to the interaction of Docker installation method and NVIDIA container toolkit versions.

⁂

**help me remove everything and reinstall everything my nvdia-smi is working currently**

Here's a step-by-step guide to completely remove Docker installed via Snap, remove NVIDIA container toolkit, and then reinstall Docker and NVIDIA Container Toolkit properly on Ubuntu:

**1. Remove Docker Snap Package (If Installed Via Snap)**

* List installed snap packages:

snap list

* Remove the Docker snap package:

sudo snap remove docker

* Optional: Purge to avoid snapshot backups during removal:

sudo snap remove docker --purge

* Reboot system to apply changes:

sudo reboot

**2. Remove NVIDIA Container Toolkit and Related Packages**

* Purge NVIDIA container toolkit packages:

sudo apt-get purge -y nvidia-container-toolkit nvidia-docker2

* Remove NVIDIA container runtime libraries:

sudo apt-get autoremove --purge -y

* Cleanup leftover data:

sudo rm -rf /var/lib/docker /var/lib/nvidia-container-runtime  
sudo rm -rf /var/lib/containerd

**3. Install Docker (From Apt)**

* Update package index:

sudo apt-get update

* Install prerequisites:

sudo apt-get install -y ca-certificates curl gnupg lsb-release

* Add Docker’s official GPG key:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

* Set up the Docker stable repository:

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

* Install Docker Engine:

sudo apt-get update  
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

* Verify Docker installation:

sudo docker run hello-world

**4. Install NVIDIA Container Toolkit**

* Add the NVIDIA Docker package repositories and keys:

distribution=$(. /etc/os-release;echo $ID$VERSION\_ID) \  
&& curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - \  
&& curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list

* Install the NVIDIA container toolkit:

sudo apt-get update  
sudo apt-get install -y nvidia-docker2

* Configure Docker daemon for NVIDIA runtime:

Edit or create /etc/docker/daemon.json with:

{  
 "default-runtime": "nvidia",  
 "runtimes": {  
 "nvidia": {  
 "path": "nvidia-container-runtime",  
 "runtimeArgs": []  
 }  
 }  
}

* Restart Docker daemon:

sudo systemctl restart docker

**5. Test NVIDIA Docker Support**

Run the NVIDIA CUDA container to verify GPU access:

sudo docker run --rm --gpus all nvidia/cuda:12.1.1-runtime-ubuntu22.04 nvidia-smi

Following these steps will remove any snap-based Docker installation and any problematic NVIDIA container toolkit installation, then reinstall everything from the official apt packages, which should resolve the libnvidia-ml.so.1 missing error in your Docker container.

Let me know if you want me to generate commands or a script to automate this process.

⁂

**sudo rm -rf /var/lib/docker /var/lib/nvidia-container-runtime**

sudo rm -rf /var/lib/containerd  
rm: cannot remove '/var/lib/docker/overlay2/6b5d337422273e70acfc91ff81ed2b0915e188bce411aba8263f2f41cc66a08e/merged': Device or resource busy

The "Device or resource busy" error when trying to remove Docker's overlay2 directories usually means that some overlay filesystems or mounts are still active.

To fix this:

1. **Stop Docker daemon:**

sudo systemctl stop docker

1. **Find and unmount active overlay mounts:**  
   Run:

mount | grep overlay

or

findmnt -l | grep overlay

This will list any overlay mounts related to Docker.

1. **Unmount those overlay mounts:**  
   For each overlay mount found (like /var/lib/docker/overlay2/.../merged), run:

sudo umount <mount-point>

Example:

sudo umount /var/lib/docker/overlay2/6b5d337422273e70acfc91ff81ed2b0915e188bce411aba8263f2f41cc66a08e/merged

1. **Verify no overlay mounts remain:**  
   Repeat mount | grep overlay to ensure they're all unmounted.
2. **Now remove the directories:**

sudo rm -rf /var/lib/docker  
sudo rm -rf /var/lib/nvidia-container-runtime  
sudo rm -rf /var/lib/containerd

1. Optionally, reboot after cleanup:

sudo reboot

Unmounting the active overlay mounts is necessary because the kernel prevents removal of busy mount points. Stopping Docker first ensures no containers actively use those mounts.

If umount fails due to "target is busy," you can find and stop processes using the mount with:

sudo lsof +f -- <mount-point>

or

sudo fuser -vm <mount-point>

Then kill those processes or stop related services before unmounting.

This workflow will allow you to fully clean Docker's data directories without encountering the "device busy" error.

⁂

**rutul@serverr:~$ distribution=$(. /etc/os-release;echo VERSION\_ID) \**

&& curl -s -L <https://nvidia.github.io/nvidia-docker/gpgkey> | sudo apt-key add - \  
&& curl -s -L <https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list> | sudo tee /etc/apt/sources.list.d/nvidia-docker.list  
[sudo] password for rutul:  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).  
OK

**Unsupported distribution!**

**Check** [**https://nvidia.github.io/nvidia-docker**](https://nvidia.github.io/nvidia-docker)

The "Unsupported distribution!" error happens because the NVIDIA Docker repository only officially supports specific Ubuntu LTS versions and detects your distribution from /etc/os-release. If your system is a derivative or a newer/unlisted version, the repository returns this error.

**How to fix:**

You can **manually override your distribution variable** to a supported Ubuntu LTS version (e.g., "ubuntu22.04") when adding the NVIDIA Docker repo, even if your OS is a compatible derivative (like Linux Mint or a newer Ubuntu version).

Example fix — run this explicitly, replacing $distribution with the supported distro name:

distribution=ubuntu22.04  
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-docker.gpg  
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | \  
 sudo tee /etc/apt/sources.list.d/nvidia-docker.list

Then update and install:

sudo apt-get update  
sudo apt-get install -y nvidia-container-toolkit

After installation, configure Docker daemon for the NVIDIA runtime if not done yet:

Edit or create /etc/docker/daemon.json:

{  
 "default-runtime": "nvidia",  
 "runtimes": {  
 "nvidia": {  
 "path": "nvidia-container-runtime",  
 "runtimeArgs": []  
 }  
 }  
}

Restart docker:

sudo systemctl restart docker

Finally, test your setup:

sudo docker run --rm --gpus all nvidia/cuda:12.1.1-runtime-ubuntu22.04 nvidia-smi

**Summary:**

* The NVIDIA Docker repo supports only certain distros; overriding $distribution=ubuntu22.04 will let you install.
* Make sure to import the GPG key correctly using gpg --dearmor (since apt-key is deprecated).
* Update and install nvidia-container-toolkit.
* Configure Docker daemon runtime and restart.
* Test with a CUDA container.

This method works around the unsupported distro issue gracefully while using an equivalent supported base.

If you want, I can provide a full script for this process.

⁂