

Cloudera Data Science Workbench

# Using NVIDIA GPUs for Cloudera Data Science Workbench Projects

Date published: 2020-02-28

Date modified: 2020-12-15

The Cloudera logo is displayed in a bold, orange, sans-serif font. The word "CLOUDERA" is written in all caps, with the letter 'E' in the middle of "UDERA" having a unique design where the top and bottom bars are separated by a horizontal gap.

<https://docs.cloudera.com/>

# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Using NVIDIA GPUs for Cloudera Data Science Workbench Projects.....</b>	<b>4</b>
Key Points to Note.....	4
Enabling Cloudera Data Science Workbench to use GPUs.....	5
Set Up the Operating System and Kernel.....	5
Install the NVIDIA Driver on GPU Hosts.....	6
Enable GPU Support in Cloudera Data Science Workbench.....	6
Using GPUs with Legacy Engines-Technical Preview.....	7
Testing ML Runtime GPU Setup.....	10

## Using NVIDIA GPUs for Cloudera Data Science Workbench Projects

A GPU is a specialized processor that can be used to accelerate highly parallelized, computationally-intensive workloads.

Minimum Required Roles: Cloudera Manager Cluster Administrator, CDSW Site Administrator



**Note:** CDSW 1.9.x supports GPUs using the ML Runtimes with 2021.02 or later versions. These versions are automatically fetched over the internet. Support for air gapped environments will be added in the next CDSW release.

Because of their computational power, GPUs have been found to be particularly well-suited to [deep learning](#) workloads. Ideally, CPUs and GPUs should be used in tandem for data engineering and data science workloads. A typical machine learning workflow involves data preparation, model training, model scoring, and model fitting. You can use existing general-purpose CPUs for each stage of the workflow, and optionally accelerate the math-intensive steps with the selective application of special-purpose GPUs. For example, GPUs allow you to accelerate model fitting using frameworks such as [Tensorflow](#), [PyTorch](#), and [Keras](#).

By enabling GPU support, data scientists can share GPU resources available on Cloudera Data Science Workbench hosts. Users can request a specific number of GPU instances, up to the total number available on a host, which are then allocated to the running session or job for the duration of the run. Projects can use isolated versions of libraries, and even different CUDA and cuDNN versions via Cloudera Data Science Workbench's extensible engine feature.

### Key Points to Note

Cloudera Data Science Workbench only supports CUDA-enabled NVIDIA GPU cards.

This topic assumes you have already installed or upgraded to the latest version of Cloudera Data Science Workbench.

- Cloudera Data Science Workbench does not support heterogeneous GPU hardware in a single deployment.
- Cloudera Data Science Workbench does not install or configure the NVIDIA drivers on the Cloudera Data Science Workbench gateway hosts. These depend on your GPU hardware and will have to be installed by your system administrator. The steps provided in this topic are generic guidelines that will help you evaluate your setup.
- The instructions described in this topic require Internet access. If you have an airgapped deployment, you will be required to manually download and load the resources onto your hosts.
- Cloudera Data Science Workbench 1.9.0 or later provides two options for supporting GPUs:
  - Support for Nvidia was introduced with ML Runtimes 2021.02. Airgapped environments will have access to ML Runtimes 2021.02 in the upcoming CDSW 1.10 release. See the documentation on the *ML Runtimes Nvidia GPU Edition*.
  - Cloudera Data Science Workbench still provides technical preview support for CUDA-enabled engines. However, CDSW does not include an engine image that supports Nvidia libraries. You must create your own custom CUDA-capable engine image using the instructions provided in *Create a Custom CUDA-capable Engine Image*.



**Note:** CUDA-enabled engines will be deprecated with CDSW 1.10 and we recommend using ML Runtimes for GPU support.

- For a list of known issues associated with this feature, refer [Known Issues - GPU Support](#) and [ML Runtimes Release Notes](#)

## Enabling Cloudera Data Science Workbench to use GPUs

To enable GPU usage on Cloudera Data Science Workbench, perform the following steps to provision the Cloudera Data Science Workbench hosts. As noted in the following instructions, certain steps must be repeated on all gateway hosts that have GPU hardware installed on them.

The steps described in this document have been tested and validated on the following setup:

Custom Engine	CDSW	NVIDIA Driver
cuda-engine:10 - Based on CDSW engine:13 - Ships CUDA 10.1  <b>Note:</b> CUDA 10.1 is not supported for TensorFlow 2.4 or above.	1.8.x, 1.9.x	418.39 (or higher) <a href="#">NVIDIA/CUDA compatibility matrix</a>

For more compatibility information across NVIDIA Drivers and CUDA, refer the [NVIDIA documentation: CUDA Compatibility](#).

## Set Up the Operating System and Kernel

The first step in enabling GPU usage on Cloudera Data Science Workbench is to set up the operating system and kernel.

### Before you begin

Perform this step on all hosts with GPU hardware installed on them.

### Procedure

1. Install the kernel-devel package.

```
sudo yum install -y kernel-devel-`uname -r`
```

If the previous command fails to find a matching version of the kernel-devel package, list all the kernel/kernel-devel versions that are available from the RHEL/CentOS package repositories, and pick the desired version to install.

You can use a bash script as demonstrated here to do this:

```
if ! yum install kernel-devel-`uname -r`; then
  yum install -y kernel kernel-devel; retVal=$?
  if [ $retVal -eq 0 ]; then echo "Reboot is required since new version of kernel was installed"; fi
fi
```

2. If you upgraded to a new kernel version in the previous step, run the following command to reboot.

```
sudo reboot
```

3. Install the Development tools package.

```
sudo yum groupinstall -y "Development tools"
```

## Install the NVIDIA Driver on GPU Hosts

Cloudera Data Science Workbench does not ship with any of the NVIDIA drivers needed to enable GPUs for general purpose processing. System administrators are expected to install the version of the drivers that are compatible with the CUDA libraries that will be consumed on each host.

### About this task

Perform this step on all hosts with GPU hardware installed on them.

### Procedure

1. Use the [NVIDIA UNIX Driver archive](#) to find out which driver is compatible with your GPU card and operating system.

To download and install the NVIDIA driver, make sure you follow the instructions on the respective driver's download page. It is crucial that you download the correct version.

For example, if you use the .run file method (Linux 64 bit), you would download and install the driver as follows:

```
wget http://us.download.nvidia.com/.../NVIDIA-Linux-x86_64-<driver_version>.run
export NVIDIA_DRIVER_VERSION=<driver_version>
chmod 755 ./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run
./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run -asq
```

2. Once the installation is complete, run the following command to verify that the driver was installed correctly:

```
/usr/bin/nvidia-smi
```

## Enable GPU Support in Cloudera Data Science Workbench

Depending on your deployment, use one of the following sets of steps to enable Cloudera Data Science Workbench to identify the GPUs installed.

Minimum Required Cloudera Manager Role: [Cluster Administrator](#)

### CSD Deployments

You can enable GPU support in Cloudera Data Science Workbench with CSD deployment.

### Before you begin

Minimum Required Cloudera Manager Role: [Cluster Administrator](#)

### Procedure

1. Ensure that the Docker daemon and worker node roles are installed on the GPU node.  
You might need to restart CDSW after you install the Docker daemon and worker node roles and before enabling GPU support.
2. Go to the CDSW service in Cloudera Manager. Click Configuration. Search for the following property and enable it:

Enable GPU Support	Use the checkbox to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a host that is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench.
--------------------	--

3. Restart the CDSW service in Cloudera Manager.
4. Test whether Cloudera Data Science workbench is detecting GPUs.

### RPM Deployments

You can enable GPU support in Cloudera Data Science Workbench with RPM deployment.

## Before you begin

Minimum Required Cloudera Manager Role: [Cluster Administrator](#)

## About this task

### Procedure

1. Set the following parameter in `/etc/cdswh/config/cdsw.conf` on all Cloudera Data Science Workbench hosts. You must make sure that `cdsw.conf` is consistent across all hosts, irrespective of whether they have GPU hardware installed on them.

`NVIDIA_GPU_ENABLE`

Set this property to true to enable GPU support for Cloudera Data Science Workbench workloads. When this property is enabled on a host that is equipped with GPU hardware, the GPU(s) will be available for use by Cloudera Data Science Workbench.

2. On the master host, run the following command to restart Cloudera Data Science Workbench.

```
cdsw restart
```

If you modified `cdsw.conf` on a worker host, run the following commands to make sure the changes go into effect:

```
cdsw stop
cdsw join
```

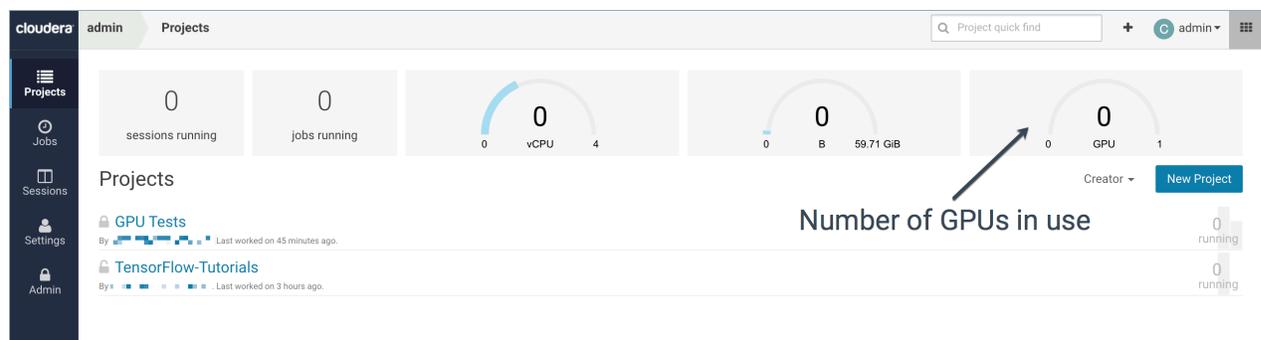
3. Use the following section to test whether Cloudera Data Science Workbench can now detect GPUs.

### Test whether Cloudera Data Science Workbench can Detect GPUs

Once Cloudera Data Science Workbench has successfully restarted, if NVIDIA drivers have been installed on the Cloudera Data Science Workbench hosts, Cloudera Data Science Workbench will now be able to detect the GPUs available on its hosts.

Minimum Required Cloudera Manager Role: [Cluster Administrator](#)

Once Cloudera Data Science Workbench has successfully restarted, if NVIDIA drivers have been installed on the Cloudera Data Science Workbench hosts, Cloudera Data Science Workbench will now be able to detect the GPUs available on its hosts.



Additionally, the output of this command will also indicate that there are hosts with GPUs present.

```
cdsw status
```

## Using GPUs with Legacy Engines-Technical Preview

To use GPUs with legacy engines, you must create a custom CUDA-capable engine image.

### CUDA Engine

To make it easier for users to get started with using GPUs on CDSW, the Cloudera engineering team is working on a new custom engine that comes enabled with CUDA out of the box.

Previously, users were expected to [build their own CUDA engine](#).



**Important:** This engine is still under development and is not recommended for use in production environments.

### Compatibility Information

The first version of this engine is built on top of CDSW base engine:13 and ships with CUDA 10.1. It has been tested with CDSW 1.8.x.

Custom Engine	CDSW	NVIDIA Driver
cuda-engine:10 - Based on CDSW engine:13 - Ships CUDA 10.1  <b>Note:</b> CUDA 10.1 is not supported for TensorFlow 2.4 or above.	1.8.x, 1.9.x	418.39 (or higher) <a href="#">NVIDIA/CUDA compatibility matrix</a>

### Create a Custom CUDA-capable Engine Image

The base engine image (`docker.repository.cloudera.com/cdsw/engine:<version>`) that ships with Cloudera Data Science Workbench will need to be extended with CUDA libraries to make it possible to use GPUs in jobs and sessions.



**Note:** Before you proceed, review the list of known issues and limitations associated with custom engines [here](#).

The following sample Dockerfile illustrates an engine on top of which machine learning frameworks such as Tensorflow and PyTorch can be used. This Dockerfile uses a deep learning library from NVIDIA called [NVIDIA CUDA Deep Neural Network \(cuDNN\)](#). For detailed information about compatibility between NVIDIA driver versions and CUDA, refer the [cuDNN installation guide \(prerequisites\)](#).

When creating the Dockerfile, you must delete the Cloudera repository that is inaccessible because of the paywall by running the following:

```
RUN rm /etc/apt/sources.list.d/*
```

Make sure you also check with the machine learning framework that you intend to use in order to know which version of cuDNN is needed. As an example, Tensorflow's NVIDIA hardware and software requirements for GPU support are listed in the [Tensorflow documentation](#).

The following sample Dockerfile uses NVIDIA's official Dockerfiles for [CUDA and cuDNN images](#).

cuda.Dockerfile

```
FROM docker.repository.cloudera.com/cdsw/engine:10

RUN apt-get update && apt-get install -y --no-install-recommends \
  gnupg2 curl ca-certificates && \
  curl -fsSL https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub | apt-key add - && \
  echo "deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 /" > /etc/apt/sources.list.d/cuda.list && \
  echo "deb https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64 /" > /etc/apt/sources.list.d/nvidia-ml.list && \
  apt-get purge --autoremove -y curl && \
  rm -rf /var/lib/apt/lists/*

ENV CUDA_VERSION 10.1.243
LABEL com.nvidia.cuda.version="${CUDA_VERSION}"
```

```

ENV CUDA_PKG_VERSION 10-1=${CUDA_VERSION}-1
RUN apt-get update && apt-get install -y --no-install-recommends \
    cuda-cudart-${CUDA_PKG_VERSION} && \
    ln -s cuda-10.1 /usr/local/cuda && \
    rm -rf /var/lib/apt/lists/*
RUN echo "/usr/local/cuda/lib64" >> /etc/ld.so.conf.d/cuda.conf && \
    ldconfig

RUN echo "/usr/local/nvidia/lib" >> /etc/ld.so.conf.d/nvidia.conf && \
    echo "/usr/local/nvidia/lib64" >> /etc/ld.so.conf.d/nvidia.conf

ENV PATH /usr/local/nvidia/bin:/usr/local/cuda/bin:${PATH}
ENV LD_LIBRARY_PATH /usr/local/nvidia/lib:/usr/local/nvidia/lib64

RUN echo "deb http://developer.download.nvidia.com/compute/machine-learning/
repos/ubuntu1604/x86_64 /" > /etc/apt/sources.list.d/nvidia-ml.list

ENV CUDNN_VERSION 7.6.5.32
LABEL com.nvidia.cudnn.version="${CUDNN_VERSION}"
RUN apt-get update && apt-get install -y --no-install-recommends \
    libcudnn7=${CUDNN_VERSION}-1+cuda10.1 && \
    apt-mark hold libcudnn7 && \
    rm -rf /var/lib/apt/lists/*

```

You can now build a custom engine image out of `cuda.Dockerfile` using the following sample command:

```
docker build --network host -t <company-registry>/cdsw-cuda:10 . -f cuda.Dockerfile
```

Push this new engine image to a public Docker registry so that it can be made available for Cloudera Data Science Workbench workloads. For example:

```
docker push <company-registry>/cdsw-cuda:10
```

### Site Admins: Add the Custom CUDA Engine to your Cloudera Data Science Workbench Deployment

After you've created the custom CUDA engine, a site administrator must add this new engine to Cloudera Data Science Workbench.

#### Before you begin

Required CDSW Role: Site Administrator

#### Procedure

1. Sign in to Cloudera Data Science Workbench.
2. Click Admin.
3. Go to the Engines tab.
4. Under Engine Images, add the custom CUDA-capable engine image created in the previous step.  
This allows project administrators across the deployment to start using this engine in their jobs and sessions.
5. Site administrators can also set a limit on the maximum number of GPUs that can be allocated per session or job.  
From the Maximum GPUs per Session/Job dropdown, select the maximum number of GPUs that can be used by an engine.
6. Click Update.

#### Project Admins: Enable the CUDA Engine for your Project

Project administrators can use the following steps to make it the CUDA engine the default engine used for workloads within a particular project.

### Procedure

1. Navigate to your project's Overview page.
2. Click Settings.
3. Go to the Engines tab.
4. Under Engine Image, select the CUDA-capable engine image from the dropdown.

### Test the CUDA Engine

You can use the following simple examples to test whether the new CUDA engine is able to leverage GPUs as expected.

### Procedure

1. Go to a project that is using the CUDA engine and click New Session.
2. Launch a new session with GPUs.
3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

```
! /usr/bin/nvidia-smi
```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

#### Pytorch

```
!pip3 install torch
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

#### Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

#### Keras

```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```

## Testing ML Runtime GPU Setup

You can use the following simple examples to test whether the new ML Runtime is able to leverage GPUs as expected.

1. Go to a project that is using the ML Runtimes Nvidia GPU edition and click Open Workbench.
2. Launch a new session with GPUs.
3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

```
! /usr/bin/nvidia-smi
```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

#### Pytorch

```
!pip3 install torch==1.4.0
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

#### Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

#### Keras

```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```