..

Downloading and uploading Model Repositories for an air-gapped environment

Date published: 13 May 2025
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Downloading and uploading Model Repositories for an air-gapped environment

An air-gapped environment is physically isolated from the internet and external networks, preventing the transmission or reception of data online. As a result, enabling the download of Model Repositories in such environments requires the Administrator to perform additional steps.

To use Models from NVIDIA NGC and Hugging Face, the Administrator must download Model artifacts from these sources on specially networked hosts. The artifacts must then be manually transferred, uploaded to the object storage utilized by the Cloudera AI Registry and Cloudera AI Inference service. Following that, the available Models are ready to be used. This solution is an alternative to accessing Model Hub in an air-gapped environment.

Prerequisites for downloading and uploading Model artifacts in air-gapped environment

Before downloading or uploading models, ensure you have the following tools and configurations installed on the host that is connected to the airgap setup. This might be your bastion host.

- Required tools:
  - Hugging Face CLI: pip install -U "huggingface_hub[cli]"
  - AWS CLI: pip install awscli==1.35.0
  - PyYAML:pip install pyyaml
  - NVIDIA NGC CLI: Install from https://org.ngc.nvidia.com/setup/installers/cli for NVIDIA NGC catalog models. Make sure you configure the NVIDIA NGC client with the credentials provided by Cloudera.
  - Azure CLI: pip install azure-cli
  - Python: Ensure your Python version is is 3.10.12 or higher and lower than version 3.11
  - Python requests package: pip install requests
- Configuration details:
  - Configure the NGC client with credentials provided by Cloudera during onboarding. Use the following commands to add your key and organization to your `~/.bashrc` file.
  - Bash

```
echo 'export NGC_CLI_API_KEY=<key>' >> ~/.bashrc
       echo 'export NGC_API_KEY=<key>' >> ~/.bashrc
       echo 'export NGC_CLI_ORG=<org>' >> ~/.bashrc
```

Note: If the system has ~/.bash_profile follow the above steps, but replace bashrc with bash_profile.

- Installing the NVIDIA Inference Microservice (NIM) CLI

  This procedure details how Cloudera organization accounts can request early access and install the NIM CLI.

  1. Obtaining Early Access to NIM CLI: Navigate to the NVIDIA developer portal and follow the on-screen instructions to request early access.
  2. Installing NIM CLI: Once early access is granted, use the following steps to download and install the NIM CLI:

     a. Download the installer using the NVIDIA GPU Cloud (NGC) CLI:

     ```
     ngc registry resource download-version nvidia/nim-tools/nimtools_ins
     taller:0.0.8
     ```

     b. Navigate to the installer directory:

     ```
     cd nimtools_installer_v0.0.8/
     ```

     c. Run the Python installation script. Be sure to provide your NGC service key and the `--nimcli-only` flag:

     ```
     python3 nimtools_installer.py --ngc-api-key [***your-ngc-service-key
     ***] --nimcli-only
     ```

     Important: You must also download the manifest folder from https://github.com/cloudera/Model-Hub/tree/main/manifests. Ensure the NGC specification file and these manifests are in the same directory.

- Download Script:

  Use the import_to_airgap.py script to download model repositories from Hugging Face or NVIDIA NGC and upload them to your cloud storage.

  You can download the script from this GitHub location: https://github.com/cloudera/Model-Hub/blob/main/airgap-scripts/pbc/import_to_airgap.py

  The script has the following parameters:

  Table 1:

Parameter	Value	Description
-do		Activates download mod
-rt	hf or ngc	Repository type: use hf for Hugging Face. Use ngc for NVIDIA NGC catalog.
-t	<token>	Hugging Face API token: required for accessing private or gated models that need authentication. Models or Models that require authentication. For more information about tokens, see: https://huggingface.co/docs/hub/en/security-tokens
-p	$PWD/models	Local Path: The destination directory where model files will be downloaded (example: $PWD/models). It uses the current working directory.

Parameter	Value	Description
-ri	<ID>	Repository ID: The model's ID from either Hugging Face or NVIDIA NGC. You can obtain the ri argument for Hugging Face as follows: 1. Open up the Hugging Face page at: https://huggingface.co/. 2. Search for the required model. The page of the model displays. 3. Click the icon next to the name of the model, and copy the model ID, that is the ri argument to be used.
-ns	<file name>	NGC Specification File: Required when downloading NGC models. The file can be downloaded from: https://github.com/cloudera/Model-Hub/blob/main/models/airgapped/public/1.50.0_concatenated.yaml

Understanding NVIDIA NGC file

The NGC specification script includes commands to iterate through the NGC specification file and retrieve the repository ID.

The NVIDIA NGC specification YAML file specifies metadata for NGC AI models, including multiple optimization profiles for each model. These profiles describe how each model is packaged and optimized for specific hardware and use cases (for example, latency or throughput tuning).

```
models:
  - name: ...
    modelVariants:
      - variantId: ...
        optimizationProfiles:
          - profileId: ...
```

The profileId of each optimizationProfile is the repository ID we provide as an -ri argument in the script.

The example NVIDIA NGC specification file provided below has the following details:

- one model: E5 Embedding v5
- one variant under modelVariants: E5 Embedding
- one optimizationProfile: nim/nvidia/nv-embedqa-e5-v5:5_FP16_onnx

```
models:
- name: E5 Embedding v5
  displayName: E5 Embedding v5
  modelHubID: e5-embedding-v5
  category: Embedding
  type: NGC
  description: NVIDIA NIM for GPU accelerated NVIDIA Retrieval QA E5 Embe
dding v5
    inference
  modelVariants:
  - variantId: E5 Embedding
    displayName: E5 Embedding
    source:
      URL: https://catalog.ngc.nvidia.com/orgs/nim/teams/nvidia/containers/
nv-embedqa-e5-v5
    optimizationProfiles:
```

```
    - profileId: nim/nvidia/nv-embedqa-e5-v5:5_FP16_onnx
      displayName: Embedding ONNX FP16
      framework: ONNX
      sha: onnx
      ngcMetadata:
        onnx:
          container_url: https://catalog.ngc.nvidia.com/containers
          model: nvidia/nv-embedqa-e5-v5
          model_type: embedding
          tags:
            llm_engine: onnx
          workspace: !workspace
            components:
            - dst: ''
              src:
                repo_id: ngc://nim/nvidia/nv-embedqa-e5-v5:5_tokenizer
            - dst: onnx
              src:
                repo_id: ngc://nim/nvidia/nv-embedqa-e5-v5:5_FP16_onnx
      modelFormat: onnx
      latestVersionSizeInBytes: 668847682
      spec:
      - key: DOWNLOAD SIZE
        value: 1GB
      - key: MAX TOKENS
        value: 512
      - key: Dimension
        value: 1024
      - key: NIM VERSION
        value: 1.0.1
```

To download this optimization profile using the airgap script use the following the command:

```
python3 import_to_airgap.py -do -rt ngc -p $PWD/models -ri nim/nvidia/nv-emb
edqa-e5-v5:5_FP16_onnx -ns ./ngc_spec.yaml
```

Optimization profile ID

To understand optimization profiles, pay attention to the infromation highlighted in bold in the following example optimization profile:

nim/meta/llama-3.2-11b-vision-instruct:0.15.0.dev2024102300+ea8391c56-**h100x2-fp8-latency**.0.3.20143152

It conveys the following information:

- h100: The NVIDIA GPU type required to run this model is H100.
- x2: It specifies the two GPUS of H100.
- fp8: The precision is FP8, representing 8-bit floating-point format.
- latency: The model profile is designed to optimize latency.

Traversing NVIDIA NGC specification file

The provided NGC specification file is nearly 5,000 lines long, making it tedious to manually locate the profile ID. To simplify this process, the airgap script includes commands to efficiently navigate through the NGC spec file.

Use the following commands to list all the models in the NGC specification file:

```
# List all models
python import_to_airgap.py -ns ./ngc-spec.yaml --list-all
=== ALL MODELS ===
1. Llama 3.2 Vision Instruct
   Display Name: Llama 3.2 Vision Instruct
```

```
   Category: Image to Text Generation
   Hub ID: llama-3.2-vision-instruct
   Description: The Llama 3.2 Vision instruction-tuned models are optimized
 for visual recognition, image reasoning,...

2. Mixtral Instruct
   Display Name: Mixtral Instruct
   Category: Text Generation
   Hub ID: mixtral-instruct
   Description: The Mixtral Large Language Model (LLM) is a pretrained ge
nerative Sparse Mixture of Experts model. M...

3. E5 Embedding v5
   Display Name: E5 Embedding v5
   Category: Embedding
   Hub ID: e5-embedding-v5
   Description: NVIDIA NIM for GPU accelerated NVIDIA Retrieval QA E5 Embe
dding v5 inference
```

To display all variants of a specific model, use the -m parameter to specify the model name from the list above, along with the --list-variants parameter to list all available model variants.

```
python3 import_to_airgap.py -ns ./ngc-spec.yaml -m "Llama 3.2 Vision Instruc
t" --list-variants

=== VARIANTS FOR 'LLAMA 3.2 VISION INSTRUCT' ===
1. Llama 3.2 11B  Vision Instruct
2. Llama 3.2 90B  Vision Instruct
```

To list all the optimization profiles for a given model and a model variant, use the following command:

```
python3 import-pvc.py -ns ./ngc-private.yaml -m "Llama 3.2 Vision Instruct"
-vid "Llama 3.2 11B  Vision Instruct" --list-profiles

=== OPTIMIZATION PROFILES FOR 'LLAMA 3.2 VISION INSTRUCT' VARIANT 'LLAMA
3.2 11B  VISION INSTRUCT' ===
1. nim/meta/llama-3.2-11b-vision-instruct:0.15.0.dev2024102300+ea8391c56-h
100x2-bf16-latency.0.3.20143152
2. nim/meta/llama-3.2-11b-vision-instruct:0.15.0.dev2024102300+ea8391c56-a10
gx4-bf16-throughput.0.3.20143152
3. nim/meta/llama-3.2-11b-vision-instruct:0.15.0.dev2024102300+ea8391c56-a1
0gx8-bf16-latency.0.3.20143152
4. nim/meta/llama-3.2-11b-vision-instruct:0.15.0.dev2024102300+ea8391c56-
h100x2-fp8-latency.0.3.20143152
....
```

Select an optimization profile that matches your hardware requirements and provide it as the repository ID using the -ri parameter in the airgap script to download the specific NGC model profile.

```
python3 import_to_airgap.py -do -rt ngc -p $PWD/models -ns ./ngc-spec.yaml -
ri nim/meta/llama-3.2-11b-vision-instruct:0.15.0.dev2024102300+ea8391c56-h10
0x2-fp8-latency.0.3.20143152
```

Downloading Model Repositories for an air-gapped environment

To use Models from NVIDIA NGC and Hugging Face, the Administrator must download Model artifacts from these sources on specially networked hosts.

Downloading a HuggingFace model

1. Download the Llama-3.1-Nemotron-70B-Instruct-HF Model from Hugging Face to your local file system with the following command:

```
python3 import_to_airgap.py -do -rt hf -t hf_hVQbUCkpCicZYjnqsNAfafafafa
fafaAEkj -p $PWD/models -ri Nvidia/Llama-3.1-Nemotron-70B-Instruct-HF
```

   The download includes all Model files along with metadata in the specified destination directory.
2. Download a different Hugging Face Model to your local file system with the following command:

```
python3 import_to_arigap.py -do -rt hf -t <your-hf-token> -p $PWD/models -
ri meta-llama/Llama-2-70b-chat-hf
```

   - You can obtain the ri argument for Hugging Face as follows:
     a. Open up the Hugging Face page at: https://huggingface.co/.
     b. Search for the required model.

        The page of the model displays.
     c. Click the  icon next to the name of the model, and copy the model ID, that is the ri argument to be used.

   The download includes all Model files along with metadata in the specified destination directory.

Note: The above commands download Model artifacts to Models subfolder in the current working directory.

Downloading NGC model

- Download the NVIDIA NGC Model to your local file system with the following command:

```
python3 import_to_airgap.py -do -rt ngc -p $PWD/models -ri nim/meta/llam
a-3_1-70b-instruct:0.11.1+14957bf8-h100x4-fp8-throughput.1.2.18099809 -ns
 ngc_spec.yaml
```

Note:

For NVIDIA NGC catalog downloads, use the -rt NGC argument and provide an NVIDIA NGC specification file with the -ns parameter. The -ns parameter is only required when downloading NVIDIA NGC Models and the NGC-specific file to use is specified. Cloudera distributes these specific files to you. For more information, contact Cloudera Support.

The download includes all Model files along with metadata in the specified destination directory.

Uploading Model Repositories for an air-gapped environment

The Model artifacts must be manually transferred, uploaded to the cloud storage utilized by the Cloudera AI Registry and Cloudera AI Inference service.

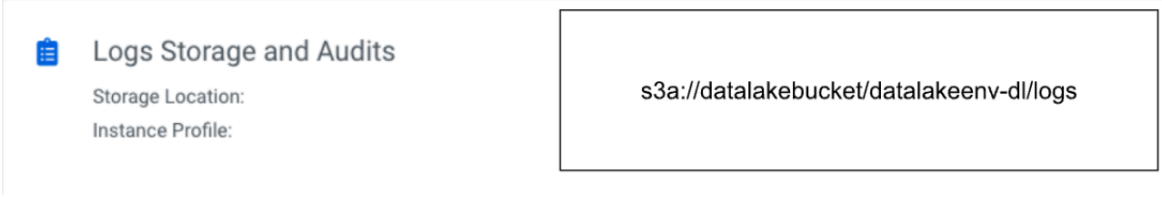
Before you begin

You will need to obtain the data lake bucket or container information for your cloud provider to use as the destination for the model artifacts.

For AWS

1. In the Cloudera console, click the Management Console tile.

2. Click Environments, then select your AWS environment.
3. On the Environment details page, click Summary.
4. Scroll down to the Logs Storage and Audit field and copy the storage location.

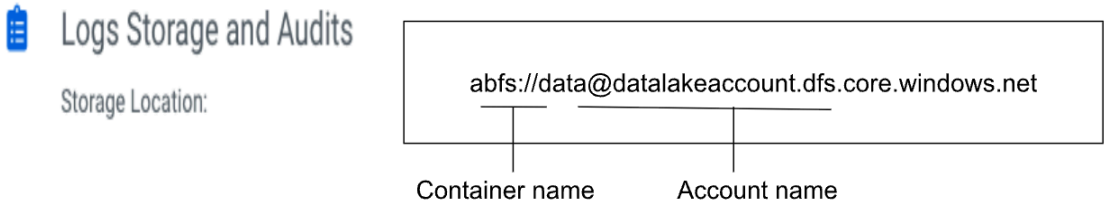


5. Omit /logs from the location.

   Example: If the log storage location is s3://datalakebucket/datalakeenv-dl/logs, the datalake bucket is s3://datalakebucket/datalakeenv-dl. The final destination for the model artifacts will be s3://datalakebucket/datalakeenv-dl/modelregistry/secured-models.

For Azure

1. In the Cloudera console, click the Management Console tile.
2. Click Environments, then select your AWS environment.
3. On the Environment details page, click Summary.
4. Scroll down to the Logs Storage and Audit field and copy the storage location.



   Example: If the log storage location is data@datalakeaccount.dfs.core.windows.net, the container name is data, and the account name is datalakeaccount. You will need this information for the `--account` and `--container` parameters when running the upload script.

1. Run the import_to_airgap.py script to upload the model artifacts to a secured location in your cloud environment.

   For AWS

   Run the script using the following command to upload the Model artifacts to a secured location.

```
python3.9 import_to_airgap.py -i -e <endpoint> -c <cloud_type> -s <sourc
e_directory> -d <destination> -ri <repository_id>
```

Example:

```
python3.9 import_to_arigap.py -c aws -s $PWD/models -d s3://datalakebucket/datalakeenv-dl/modelregistry/secured-models -ri nim/meta/llama-3_1-70b-instruct:0.11.1+14957bf8-h100x4-fp8-throughput.1.2.18099809
```

Note: The script uploads the downloaded Model artifacts to a secured location in Cloudera on cloud. The destination format must be s3://[***datalake bucket***]/modelregistry/secured-models. An administrator can modify this destination.

You can use the following parameters for uploading the Models.

Table 2: Paramaters for uploading the Models

Parameter	Description	Example
-c	Cloud type (AWS, Azure)	-c aws
-s	Must contain the previously downloaded Model artifacts as it is the source directory of the downloaded Model.	-s $PWD/models
-d	Must point to the Cloudera AI Registry bucket with the appropriate path. The destination format must be: s3://bucket/secured-models	-d s3://bucket/secured-models
-rt	Repository type (Hugging Face or NVIDIA NGC)	-rt hf
-ri	Repository ID of the Model downloaded to local filesystem	`-ri nim/meta/llama-3_1-70b-instruct:0.11.1+14957bf8-h100x4-fp8-throughput.1.2.18099809`

For Azure

Run the script using the following command to upload the Model artifacts to a secured location.

```
python3.9 import_to_airgap.py <endpoint> -c azure -s $PWD/models -d modelregistry/secured-models -ri <repository_id> --account $AZURE_STORAGE_ACCOUNT_NAME --container data
```

Example:

```
python3.9 import_to_arigap.py https://ccycloud-5.cml-cai.root.comops.site:9879 -c azure -s $PWD/models -d modelregistry/secured-models -ri nim/m
```

```
eta/llama-3_1-70b-instruct:0.11.1+14957bf8-h100x4-fp8-throughput.1.2.180
99809 --account datalakeaccount --container data
```

Note: The script uploads the downloaded Model artifacts to a secured location in Cloudera on cloud. The destination is modelregistry/secured-models under the account in the container. An administrator can modify this destination.

You can use the following parameters for uploading the Models.

Table 3: Paramaters for uploading the Models

Parameter	Description	Example
-c	Cloud type (AWS, Azure)	-c zure
-s	Must contain the previously downloaded Model artifacts as it is the source directory of the downloaded Model.	-s $PWD/models
-d	Must point to the Cloudera AI Registry bucket with the appropriate path. The destination format must be: s3://bucket/secured-models	-d s3://bucket/secured-models
-rt	Repository type (Hugging Face or NVIDIA NGC)	-rt hf
-ri	Repository ID of the Model downloaded to local filesystem	`-ri nim/meta/llama-3_1-70b-instruct:0.11.1+14957bf8-h100x4-fp8-throughput.1.2.18099809`
--account	Azure storage account name (Azure only)	--account $AZURE_STORAGE_ACCOUNT_NAME
--container	Azure storage container name (Azure only)	--container data

Creating the Model entry in Cloudera AI Registry in air-gapped environment

The example outlines how to create the Model entry in Cloudera AI Registry within an air-gapped environment.

For AWS

```
curl -k https://$MODELREGISTRYDOMAIN/api/v2/models -X POST -H 'Content-T
ype: application/json' -H "Authorization: Bearer $TOKEN" --data-raw '
    { "name": "llama3-instruct-70b",
    "createModelVersionRequestPayload": {
    "metadata": {
    "model_repo_type": "NGC"
    },
    "downloadModelRepoRequest": {
    "source": "REMOTE",
    "remoteObjectStoragePath": "s3://bucket1/secured-models",
    "repo_id": "nim/meta/llama-3_1-70b-instruct:0.11.1+14957bf8-h100x4-fp
8-throughput.1.2.18099809"
    }
    }
    }'
```

For Azure

```
curl -k https://$MODELREGISTRYDOMAIN/api/v2/models -X POST -H 'Content-T
ype: application/json' -H "Authorization: Bearer $TOKEN" --data-raw '
{ "name": "llama3-instruct-70b",
 "createModelVersionRequestPayload": {
     "metadata": {
         "model_repo_type": "NGC"
     },
     "downloadModelRepoRequest": {
         "source": "REMOTE",
  "remoteObjectStoragePath": "abfs://data@datalakeaccount.dfs.core.window
s.net/modelregistry/secured-models",
         "repo_id": "nim/meta/llama-3_1-70b-instruct:0.11.1+14957bf8-h1
00x4-fp8-throughput.1.2.18099809"
     }
 }
}'
```

These `curl` requests create a new model named llama3-instruct-70b in Cloudera AI Registry, and its initial version.

These requests also trigger the copying of model artifacts from your uploaded, secured model location (specified by remoteObjectStoragePath in your cloud-specific object store) to a preferred Cloudera AI Registry object store location.

Importing Model to Cloudera AI Registry in air-gapped environment

You can import the Hugging Face models listed on the Model Hub page into your Cloudera AI Registry.

Before you begin

Download the following script to enable downloading Model repositories from the Hugging Face or NVIDIA NGC catalog and uploading Models to on-cloud storage providers.

Download the script from here: https://raw.githubusercontent.com/cloudera/Model-Hub/refs/heads/main/airgap-scripts/pbc/import_to_airgap.py

The script has the following parameters:

Table 4:

Parameter	Value	Description
-do		Activates the download mod
-rt	hf or ngc	Repository type: use hf for Hugging Face. Use ngc for NVIDIA NGC catalog.
-t	hf_hVQbUsafafafafadfadfsNAynASXJoTCWHAEkj	Hugging Face API token for authentication (required for private or gated Models) The Hugging Face token (-t) is required for accessing gated Models or Models that require authentication. For more information about tokens, see:https://huggingface.co/docs/hub/en/security-tokens
-p	$PWD/models	Local destination path where Model files are downloaded (uses the current working directory)

Parameter	Value	Description
-ri	Nvidia/Llama-3.1-Nemotron-70B-Instruct-HF	Repository ID for the Model on Hugging Face

Procedure

1. In the **Cloudera** console, click the Cloudera AI tile.

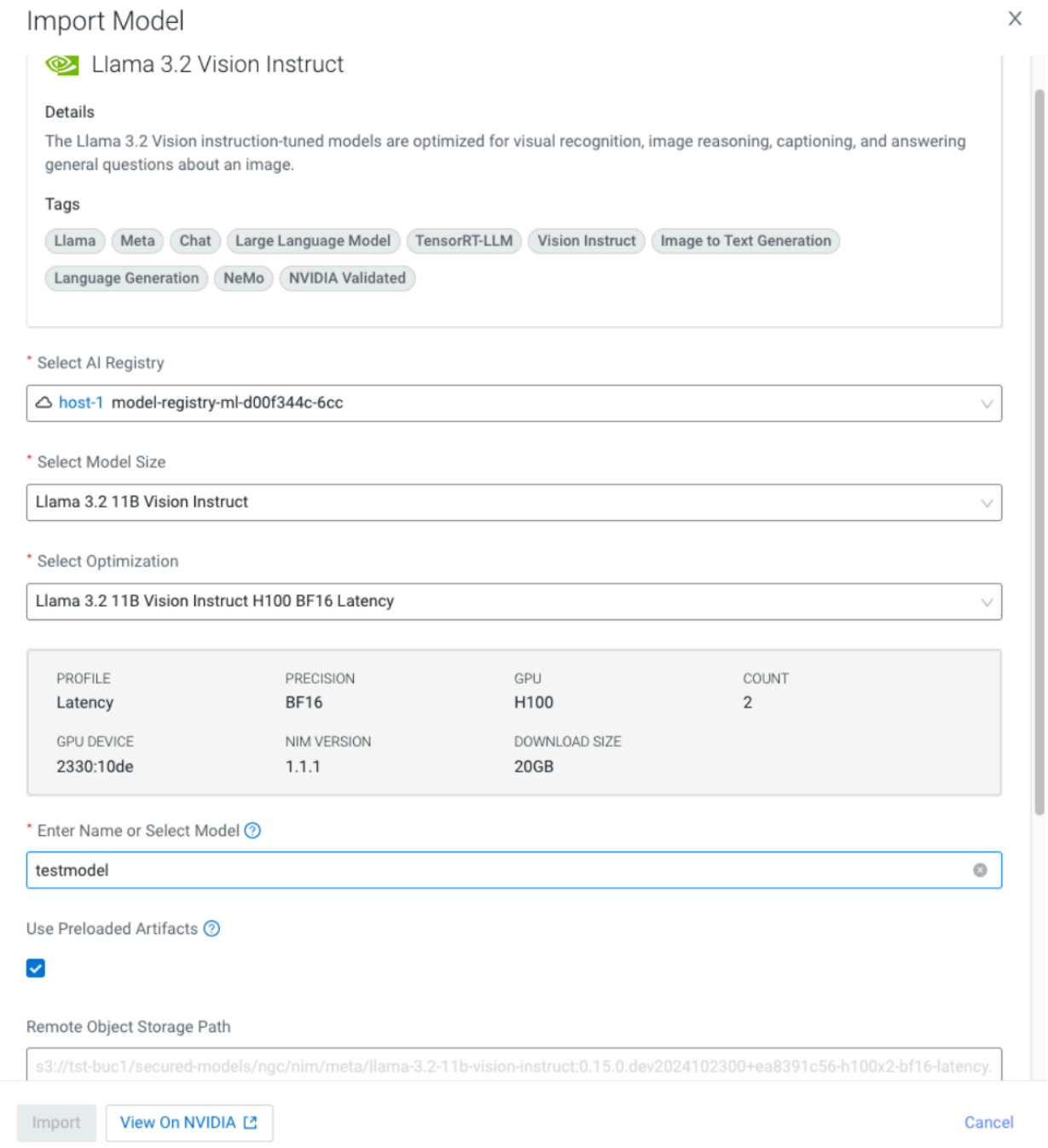
   The **Cloudera AI Workbenches** page displays.
2. Click **Model Hub** under **AI Hub** in the left navigation menu.

   The **Model Hub** page displays. The page lists different models along with their source type, tags, and description.

3. Click Import on the model you want to import.

   The **Import Model** page displays.

   Figure 1: Import model



4. In the Select AI Registry drop-down list, select the Cloudera AI Registry to which you want to import the model.
5. In the **Select Model Size** drop-down list, select the model size.
6. In the **Select Optimization** drop-down list, select the optimization profile. It displays the recommended GPU counts for the specific GPU of the optimization profile.
7. In the Enter Name or Select Model field, select a name from the existing list or enter a new name for the model you are importing.

8. Enable the **Use Preloaded Artifacts** feature with its checkbox.
9. Click Import. The Model Hub page displays a message that the Model import has been triggered successfully along with a button to view the status of that import process.

Results

You can click Cloudera AI Registry in the left navigation menu to view the newly imported Model.