



NGC Catalog

User Guide

Table of Contents

Chapter 1. What is the NGC Catalog?.....	1
Chapter 2. Terms and Concepts.....	3
Chapter 3. Why NGC Software.....	5
Chapter 4. Accessing NGC Software.....	6
Chapter 5. Registering and Activating a New NGC Org to Obtain Authenticated Access.....	8
5.1. Signing Up for an NVIDIA Cloud Account and Activating an Individual NGC Org.....	8
5.2. NGC API Keys.....	14
5.2.1. Supported NGC Applications and API Key Types.....	16
5.2.2. Generating NGC API Keys.....	16
5.2.2.1. Generating a Personal API Key.....	16
5.2.2.2. Generating a Service API Key.....	22
Chapter 6. Introduction to the NGC Catalog and NGC CLIs.....	28
6.1. Installing NGC Catalog CLI.....	29
6.2. Installing NGC Registry CLI.....	29
Chapter 7. Docker Containers.....	30
7.1. What Is a Docker Container?.....	30
7.2. Why Use A Container?.....	30
7.3. Searching and Filtering Containers.....	31
Chapter 8. NGC Container Images in NGC Catalog.....	37
8.1. Why NGC Container Images?.....	37
8.1.1. NGC Container Security Policy.....	38
8.1.2. NVIDIA Security Risk Scale.....	38
8.1.3. GPU-performance tests for NGC Catalog Container Images.....	39
8.2. NVIDIA Container Toolkit.....	39
8.3. NVIDIA CUDA Toolkit.....	39
8.4. Running Singularity Containers.....	40
8.4.1. Prerequisites.....	40
8.4.2. Converting to Singularity Image.....	40
8.4.3. Running the Singularity Container.....	41
8.4.3.1. Directory Access.....	42
8.4.3.2. Command Line Execution with Singularity.....	42
8.4.3.3. Interactive Shell with Singularity.....	42
Chapter 9. Prerequisites for Using NGC Catalog Container Images.....	43

9.1. Prerequisites for Using NGC Containers on DGX Systems.....	43
9.2. Prerequisites for Using NGC Containers on Cloud Platforms.....	43
9.3. Prerequisites for Using NGC Containers on Other NVIDIA GPUs.....	44
9.4. HPC Visualization Containers.....	44
9.4.1. Prerequisites For HPC Visualization Containers.....	45
9.4.2. Installing NVIDIA Docker 2.0.....	45
9.4.3. ParaView With NVIDIA Holodeck.....	46
9.4.4. ParaView With NVIDIA IndeX.....	46
9.4.4.1. Single-Machine With GLX.....	47
9.4.4.2. Server Container With EGL.....	47
9.4.4.3. GLX Client Connecting To A Server.....	47
9.4.5. ParaView With NVIDIA OptiX.....	47
9.4.5.1. Single-Machine Container With GLX.....	48
9.4.5.2. Server Container With EGL.....	48
9.4.5.3. Running The GLX Client And Attaching To The Server.....	48
9.4.5.4. Optional: Using The ParaView .config File.....	49
Chapter 10. Pulling NGC Containers from NGC Catalog.....	50
10.1. Key NGC Container Registry Terminology.....	50
10.2. Accessing And Pulling an NGC Container Image via the Docker CLI.....	51
10.2.1. Logging in to the NGC container registry.....	51
10.2.2. Pulling a Container from NGC Container Registry Using Docker CLI.....	52
10.3. Accessing and Pulling an NGC Catalog Container Using NGC CLI.....	53
10.3.1. Viewing Container Image Information.....	53
10.3.2. Pulling a Container Image.....	54
Chapter 11. NGC Container Image Versions.....	55
Chapter 12. NVIDIA Signed Container Images in NGC Catalog.....	56
12.1. Checking for Signed Container Images.....	56
12.1.1. Using the NGC Catalog UI.....	56
12.1.2. Using the NGC CLI.....	58
12.2. Finding NVIDIA's Public Key.....	59
12.2.1. Using the NGC Catalog UI.....	59
12.2.2. Using the NGC CLI.....	60
12.3. Verifying NVIDIA's Signature.....	60
12.3.1. Using Cosign.....	60
12.3.2. Using an Admission Controller in Kubernetes Clusters.....	61
Chapter 13. Running an NGC Container.....	62
13.1. Enabling GPU Support for NGC Containers.....	62

13.2. Running NGC Containers with Runtime Resources.....	64
13.2.1. Specifying A User.....	65
13.2.2. Setting The Remove Flag.....	65
13.2.3. Setting The Interactive Flag.....	65
13.2.4. Setting The Volumes Flag.....	66
13.2.5. Setting The Mapping Ports Flag.....	66
13.2.6. Setting The Shared Memory Flag.....	67
13.2.7. Setting The Restricting Exposure Of GPUs Flag.....	67
13.2.8. Container Lifetime.....	67
Chapter 14. Multi-Architecture Support for NGC Container Images.....	69
Chapter 15. Customizing Containers.....	71
15.1. Benefits And Limitations To Customizing A Container.....	72
15.2. Example 1: Building A Container From Scratch.....	72
15.3. Example 2: Customizing A Container Using Dockerfile.....	74
15.4. Example 3: Customizing A Container Using docker commit.....	75
15.5. Example 4: Developing A Container Using Docker.....	77
15.5.1. Example 4.1: Package The Source Into The Container.....	79
Chapter 16. Models.....	80
16.1. What are Models.....	80
16.2. Searching and Filtering Models.....	80
16.3. Downloading Models via NGC CLI.....	81
16.4. Downloading Models via WGET/cURL.....	82
16.4.1. Downloading Guest Access Models via WGET/cURL.....	82
16.4.2. Downloading Authenticated Access Models via WGET/cURL.....	84
16.5. Downloading Models via Web UI.....	86
16.6. NVIDIA Signed Models in the NGC Catalog.....	88
16.6.1. Checking for Signed Models.....	88
16.6.2. Using the NGC Catalog UI.....	88
16.6.3. Using the NGC CLI.....	89
16.6.4. Finding NVIDIA's Root Certificate.....	89
16.6.5. Verifying Model Signature.....	90
Chapter 17. Resources.....	92
17.1. What are Resources?.....	92
17.2. Searching and Filtering Resources.....	92
17.3. Downloading Resources via the NGC CLI.....	93
17.4. Downloading Resources via WGET/cURL.....	94
17.4.1. Downloading Guest Access Resources via WGET/cURL.....	94

17.4.2. Downloading Authenticated Access Resources via WGET/cURL.....	96
17.5. Downloading Resources via Web UI.....	98
Chapter 18. Helm Charts.....	101
18.1. Setting Up an NGC Helm Repository.....	101
18.2. Searching for Available Helm Charts in the NGC Catalog.....	102
18.3. Fetching Helm Charts.....	102
18.4. Using the NGC Website to View the List of Helm Charts and Get Fetch Commands.....	102
Chapter 19. SDK and AI Toolkits.....	105
Chapter 20. Deep Learning Frameworks.....	106
20.1. GPU Operator.....	106

Chapter 1. What is the NGC Catalog?

The NGC Catalog is a curated set of GPU-optimized software for AI, HPC and Visualization.

The content provided by NVIDIA and third-party ISVs simplifies building, customizing, and integrating GPU-optimized software into workflows, accelerating the time to solutions for users.

The NGC Catalog consists of containers, pre-trained models, Helm charts for Kubernetes deployments, and industry-specific AI toolkits with software development kits (SDKs).

Containers

The NGC Catalog hosts a broad range of containers, including deep learning frameworks, machine learning, HPC, and visualization applications that maximize the utilization of GPU environments. Containers package software applications, libraries, dependencies, and run-time compilers in a self-contained environment to easily deploy them across various computing environments. They enable software portability, and through a single command, users can pull, run and scale applications across the cloud, the data center, and the edge.

Models and Resources

The NGC Catalog offers pre-trained models for a wide range of common AI tasks optimized for NVIDIA Tensor Core GPUs. The pre-trained models can be used for inference or fine-tuned with transfer learning, saving data scientists and developers valuable time.

Resources offer documentation, code samples, and many other assets such as Jupyter Notebooks, deployment pipelines, and step-by-step instructions and scripts for creating deep learning models, making it easy to get started with deep learning.

Helm Charts

Kubernetes is a container orchestrator that facilitates the deployment and management of containerized applications and microservices. A Helm chart is a package manager that allows DevOps to configure, deploy and update applications across Kubernetes

environments more efficiently. The NGC Catalog provides Helm charts for deploying GPU-optimized applications and SDKs.

Software Development Kits

SDKs deliver all the tooling users need to build and deploy AI applications across domains such as medical imaging, conversational AI, or video analytics. They include annotation tools for data labeling, pre-trained models for customization with transfer learning, and SDKs that enable deployment across the cloud, the data center, or the edge for low-latency inference.

To learn more about the NGC Catalog content, visit the [NGC Catalog](#) website.

Chapter 2. Terms and Concepts

The following are common terms used in this document.

Term	Definition
Authenticated / Registered User	A user who has signed in to NGC using their email account. The user inherits all "guest user" privileges and can access certain free/ free trial entities/features that require users to sign in.
Container Images	All applications running in NGC are containerized as Docker containers and execute in our Runtime environment. Containers are stored in the NGC Registry nvcr.io, accessible from both the CLI and the Web UI.
Container Port	Opening a port when creating a job will create a URL that can be used to reach the container on that port using web protocols. The security of web applications (for example, Jupyterlab) that are accessed this way is the user's responsibility. See note below.
Enterprise Catalog	The Enterprise Catalog includes the NVIDIA AI Enterprise software and can only be accessed by customers with an active subscription. NVIDIA AI Enterprise also includes software available in the public/free catalog which is designated as "NVIDIA AI Enterprise Supported." The existing implementation of the enterprise catalog will be integrated with the NGC Catalog described in this document.
Enterprise-only features	The features of the NGC Catalog that are only available for users with a product (that is, subscribers).
Enterprise-only software	The containers, Helm charts, resources, models, and collections that are only available for users with an NVIDIA AI Enterprise subscription.

Term	Definition
Entity	A container, model, helm chart, or resource. They are also referred to as "software" in this document.
Guest User	An unauthenticated user who can access (view and download) entities from the public catalog without signing in.
Models	NGC offers a collection of state-of-the-art pre-trained deep learning models that can be easily used out of the box, re-trained or fine-tuned.
NVIDIA AI Enterprise	<p>The software layer of the NVIDIA AI platform, NVIDIA AI Enterprise, accelerates the data science pipeline and streamlines the development and deployment of production AI, including generative AI, computer vision, speech AI, and more. Regular security reviews, API stability, and dependable support SLAs ensure AI projects stay on track.</p> <p>The NVIDIA AI Enterprise subscriptions include entities, support, and exclusive software features/models.</p>
Private Registry	The NGC private registry provides a secure space to store and share custom containers, models, resources, and Helm charts within your enterprise.
Publisher	Publisher is an owner of the entity hosted on NGC.
Resources	NGC offers step-by-step instructions and scripts for creating deep learning models that you can share within teams or the org.
Subscriber	<p>A user who is signed in (registered), to NGC using their email account and is part of an org that has access to products (for example, "nv-ai-enterprise", "riva-enterprise").</p> <p>An example of a free product for subscriber is someone who has purchased a license or has been granted access to gated software for free for a limited time.</p>

Chapter 3. Why NGC Software

NGC provides software to meet the needs of data scientists, developers, and researchers across various levels of AI expertise.

All software hosted on NGC undergoes thorough scans for common vulnerabilities and exposures (CVEs), crypto, and private keys.

In addition to security scanning, NGC software is tested against a wide range of GPU-enabled platforms, including public cloud instances, workstations, and OEM servers designed for data center or edge deployments. Supported GPUs include H100, V100, A100, T4, Jetson, and the RTX Quadro.

NGC software is tested and assured to scale across multiple GPUs and, in some cases, across multiple nodes, ensuring users can fully utilize their GPU-powered servers out of the box.

For select containers, NVIDIA offers NGC Support Services to run software on DGX platforms or certified OEM servers. The service gives enterprise IT direct access to NVIDIA subject matter experts to address software issues and minimize system downtime quickly.

Chapter 4. Accessing NGC Software

There are four ways to explore and access software on NGC.

Guest Access: Content under guest access does not require users to register or sign in to pull or download the software.

Guest users can view all publicly available software and most of the entitled content in the NGC Catalog but cannot download entitled content.

The majority of software on the NGC Catalog falls under guest access. However, it is important to note that many third-party applications require a license key that can be sourced directly from ISVs. While pulling public software from the NGC Catalog does not require sign-in, a user might have to reach out to an ISV to obtain a license key for further use.

Authenticated Access: Software under authenticated access requires a user to [sign into their NGC org using an NVIDIA account](#) or sign in to NGC using SSO if the user's org is federated to their external SSO/IdP service. Pulling or downloading the software requires the user to provide their API key.

Registered users can view and download all publicly available software in the NGC Catalog. Entitled content is also viewable but not downloadable. Note that registered users may not view content exclusively gated to users of that product.

Approved Access: The publisher must grant user approval to access the software under this category. Once access is granted, the user will receive a notification with further instructions for accessing the software. Instructions to request access are provided in the overview section of the respective software.

Subscription: To access subscription-based software, users must provide Business Address details and a token (serial number, activation code, and so on). Note that this category also requires Authenticated Access.

Subscribers can view and download all publicly available software, including entitled content and exclusive content, in the NGC Catalog.

The following table outlines the permissions and capabilities associated with each user access level (guest user, registered user, subscriber).

Access Level	View Public Entities	Download Public Entities	View Entitled Content	Download Entitled Content	View/Download Exclusive Content
Guest User	Yes	Yes	Yes	No	No
Registered User	Yes	Yes	Yes	No	No
Subscriber	Yes	Yes	Yes	Yes	Yes

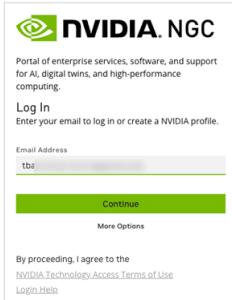
Chapter 5. Registering and Activating a New NGC Org to Obtain Authenticated Access

Instructions for registering and activating a new NGC org.

5.1. Signing Up for an NVIDIA Cloud Account and Activating an Individual NGC Org

This section describes the steps to sign up for an NCA account and activate an individual NGC org.

1. Go to the [NGC sign-in page](#) from your browser, enter your email address, and then click Continue.



2. In this step you will create your NVIDIA sign-in identity using NVIDIA's default IdP. At the Create your Account screen, create a password, make sure to review the NVIDIA Account Terms of Use and Privacy Policy, and click Create Account to accept and proceed with account creation. You will receive an email to verify your email address.

Create Your Account

Email

Password
Enter your password 

Confirm password
Enter your password 

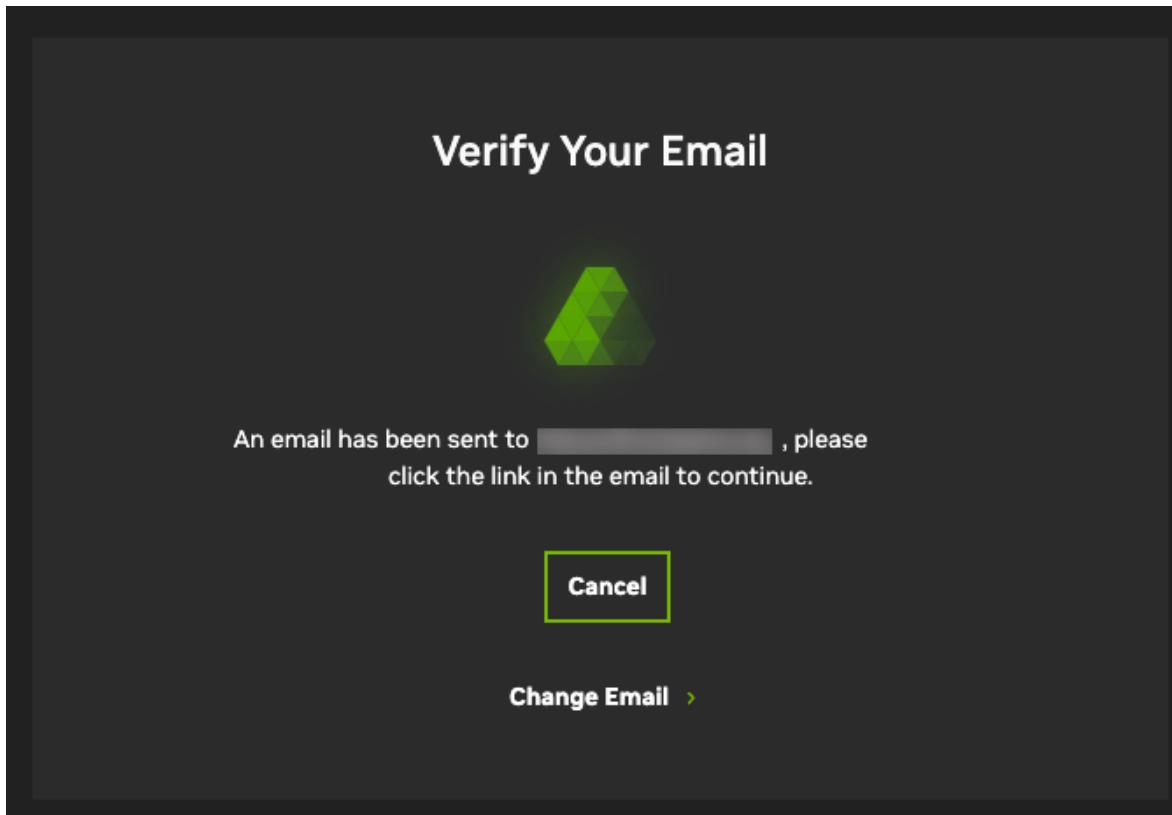
Stay logged in [Log In With Security Device >](#)

By proceeding, I agree to the [NVIDIA Account Terms Of Use](#) and [Privacy Policy](#)

[Create Account](#)

[More Signup Options](#)

A verification email is sent to your email address.



3. Open the NVIDIA account creation email and click Verify Email Address.

NVIDIA Account Created  [Inbox](#) 

 NVIDIA Accounts  <account@nvidia.com>
to: [redacted]



Hello,

We are almost done creating your NVIDIA account.
Click the link below to verify this email address.

[Verify Email Address](#)

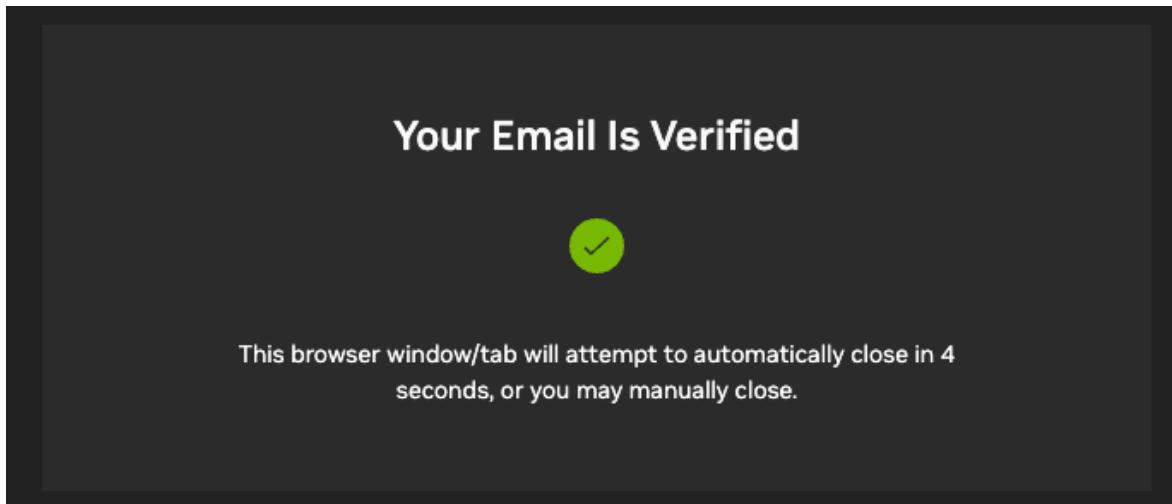
If you did not register for an NVIDIA Account, someone may have registered with your information by mistake. Contact [NVIDIA customer care](#) for further assistance.

[Manage Preferences](#) | [Contact Us](#) | [Privacy Center](#)

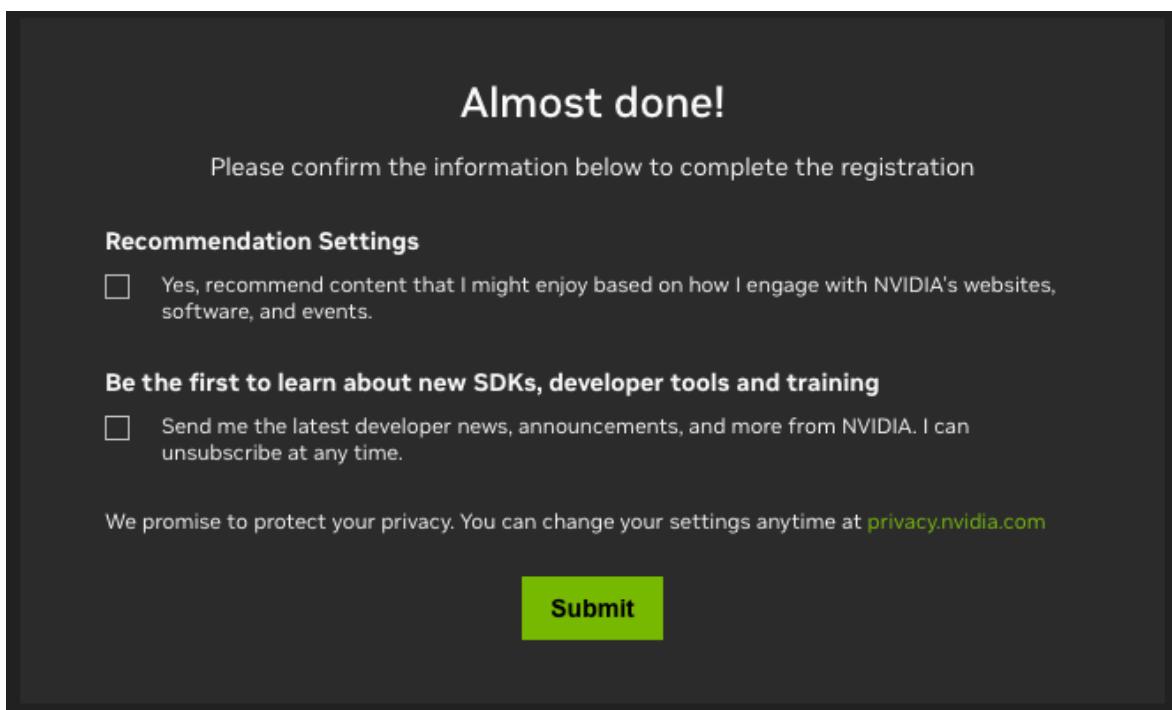
© 2024 NVIDIA Corporation. All rights reserved.

NVIDIA Corporation, 2788 San Tomas Expressway, Santa Clara, CA 95051.

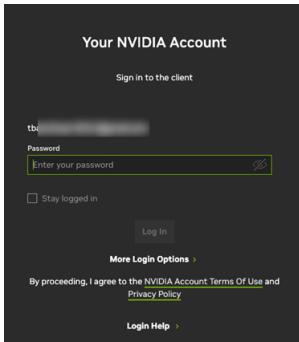
You are automatically directed to nvidia.com and see an email verified successfully page. This window will close automatically



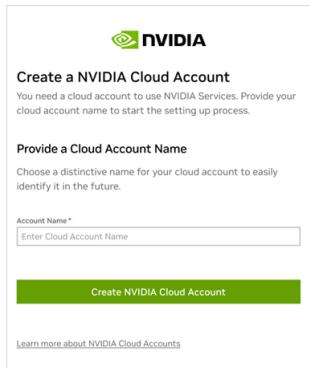
4. At the Almost done! dialog, set your communications preferences, and then click Submit.



5. Enter the password you just created to continue setting up your NVIDIA Cloud Account. (This is a required security measure).



6. Give your NVIDIA Cloud Account (NCA) a name that will help you identify it easily the next time you sign-in.

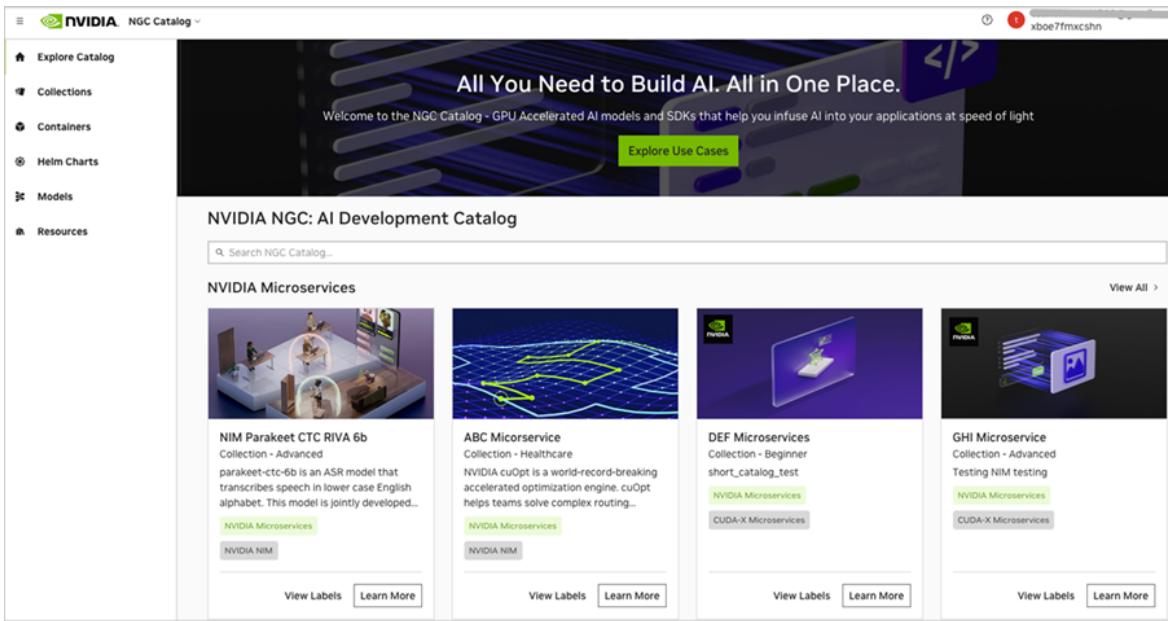


7. Complete your user profile at the Set Your Profile screen, agree to the NVIDIA GPU Cloud Terms of Use, and then click Submit.

I agree to the [NVIDIA GPU Cloud Terms of Use](#)

By obtaining software through NGC, I agree NVIDIA can share my registration details with the software providers who may use my information as permitted by their privacy policies.

Your NVIDIA account is created, and you are automatically redirected to your individual NGC org.



Your access to NVIDIA NGC org is completed, you can choose to complete the setup of your NVIDIA Cloud Account now, or at a later time.

5.2. NGC API Keys

NVIDIA NGC API keys are required to authenticate with NGC services using NCG CLI, Docker CLI, or direct API requests.

NGC provides two types of API keys:

Personal Keys

- ▶ Any NGC org user can generate a personal key.
- ▶ An NGC org user can grant a personal key up to the permissions assigned to them in the NGC org.
- ▶ A personal key is linked to the user's NGC org lifecycle.
 - ▶ If the user's permissions change, the available permissions that can be or are assigned to the personal key also change.
 - ▶ If the user is removed from the NGC org, the key's validity is revoked.
- ▶ Supports updating permissions, rotation, and deletion (immediate revocation).
 - ▶ Org owners and user_admins can revoke any member's key on demand.
- ▶ Each user can generate up to eight personal keys.

Use personal keys to begin using NGC services within your sandbox. Personal keys are best suited for individuals working on early development and testing code before moving to pre-production and production releases.

To learn how to authorize the services you have access to in the org and generate a personal key, go to [Generating a Personal API Key](#).



Important:

Use the legacy NGC API Key to authenticate with Base Command Platform, Fleet Command, or other NGC services that don't support "Personal key" authentication. For cross-org authorization, continue using the legacy NGC API Key.

NVIDIA plans to deprecate the legacy NGC API key after 2025. NVIDIA encourages you to use the Personal Key, but if you need to continue using the legacy API key, go to [Generating a Legacy NGC API Key](#) to find out where to create a new one. Also, your current NGC API key will continue to work.

Service Keys

- ▶ The lifecycle of service keys is linked to the NGC org account, not associated with an individual user.
- ▶ Only NGC org owners and user_admins can manage service keys.
- ▶ A service key can be scoped to access only the permissions and services required, or full access to the services enabled in the org.
- ▶ Supports scoped permissions, updating permissions, on-demand revocation, rotation, and deletion.
- ▶ An NGC org can have up to 50 service keys.

Use service keys when you require automated communication between machines and deploying to pre-production and production environments where you do not want to depend on a user's membership status in the NGC org.



Note: Service keys currently do not support listing artifacts in NGC CLI or Docker CLI. This functionality will be added in the future. In the meantime, use a Personal API key to list artifacts.

Examples using NGC API Keys

Here are some examples of using NGC API keys to authenticate with NGC CLI and Docker CLI:

NGC CLI

```
$ ngc config set <Enter>
```

Paste your key value at the API_KEY prompt:

```
[Enter API key [*****API-Key]. Choices: [<VALID_APIKEY>]
```



Important:

Always use the latest NGC CLI version to access the newest features, bug fixes, performance improvements, and security updates. Check for the latest versions at [NGC CLI Installers](#) or run `ngc version list` to view the latest releases, then upgrade using

```
ngc version upgrade
```

Docker CLI

```
docker login nvcr.io --username '$oauthtoken'
```

For the username, enter '\$oauthtoken' exactly as shown. It is a special name that indicates that you will authenticate with an API key. Paste your key value at the Password prompt.

5.2.1. Supported NGC Applications and API Key Types

The NVIDIA NGC applications/services that support Personal and Service Keys are listed below:

NGC Application/ Services	Service Description
NVIDIA NGC Catalog	Grants your key permission to access or download containers and artifacts from the NGC Catalog. The permission level matches your account's permissions for the catalog.
NVIDIA NGC Private Registry	The key is authorized to perform actions on your organization's private registry service, such as pulling, retrieving, creating, or deleting containers and artifacts. The permission level assigned to the key matches the permission level of your user account. Therefore, your user account must have permissions for the Private Registry.
NVIDIA Cloud Functions	This authorization allows your key to perform actions on your organization's cloud functions service. If your organization has private functions published by NVIDIA, or if your cloud functions service enables you to create, deploy, and run your own functions, your personal key will have the same permissions as your user account for the cloud functions service. Therefore, it's important that your user account has the necessary permissions for Cloud Functions.
NVIDIA Public API Endpoints	Grants permission for your key to access NVIDIA NIM inference endpoints listed in the NVIDIA API Catalog . Therefore, your user account must have Public API Endpoints permissions.
NVIDIA Secrets Manager	Authorizes your key to perform actions on the NVIDIA Secrets Manager service, which is used to store and manage secrets. Your key will have the same permission level as your user account, so your user account must possess Secrets Manager permissions.

5.2.2. Generating NGC API Keys

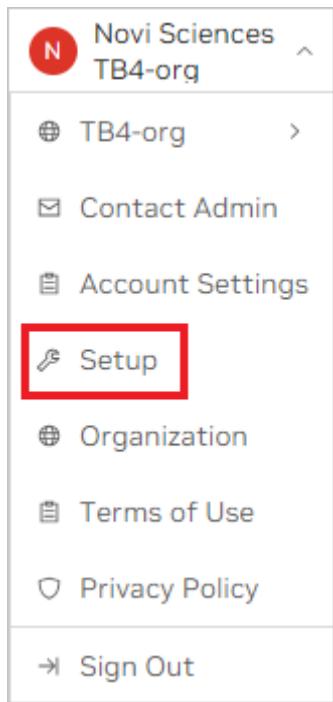
Generating API keys is essential for authenticating with NGC services using the NGC CLI, Docker CLI, or direct API requests.

5.2.2.1. Generating a Personal API Key

1. Sign in to the NGC website.

From a browser, go to <https://ngc.nvidia.com/signin> and then enter your email and password.

2. Click your user account icon in the top-right corner and select Setup.



3. Click Generate API Key from the available options.

A screenshot of the 'Setup' page. The page has two main sections: 'Keys/Secrets' and 'Developer Tools'. In the 'Keys/Secrets' section, there are three cards: 'API Keys' (with a 'Generate API Key' button highlighted by a red box), 'Secrets' (with a 'View Secrets' button), and 'Secure Credential Sharing' (with a 'Share Credentials' button). In the 'Developer Tools' section, there are two cards: 'CLI' (with 'Documentation' and 'Downloads' buttons) and 'SDK' (with 'Documentation' and 'Downloads' buttons).

4. On the Setup > API Keys page, click + Generate Personal Key on the menu or the pane.

The screenshot shows the 'API Keys' page under 'Setup'. At the top, there's a green button labeled '+ Generate Personal Key'. Below it, a message says 'No Personal Keys Configured' with a small 'O' icon. Another green button labeled '+ Generate Personal Key' is present. The main content area is titled 'NGC API Keys' and contains information about NVIDIA NGC API keys required for authentication. It details two types: 'Personal API Key' and 'Legacy API Key'. The 'Personal API Key' section explains its use for user lifecycle management and mentions rotation and expiration. The 'Legacy API Key' section notes its originality and the fact that NVIDIA will support it until services transition to next-generation keys. Below this, sections for 'NGC CLI' and 'Docker' show command-line examples for generating tokens. A 'Read More' link is available for Docker. At the bottom, a 'Legacy Keys' section is partially visible.

5. In the Generate Personal Key dialog, fill in the required information for your key.

Generate Personal Key

Your Personal API Key authenticates your use of the selected services associated with your account within only this organization when using a CLI or Rest API.

Key Name *

This key authenticates services only within the **NVIDIA** organization

Expiration *

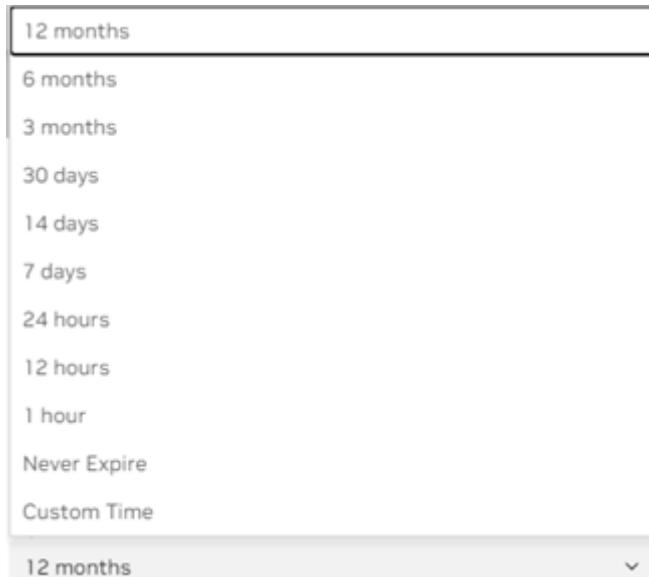
Services Included *

Private Registry | X ▼

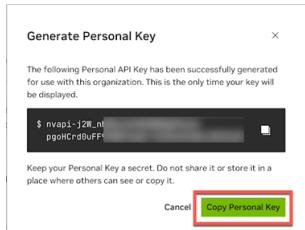
These services are based on your access within this organization

Cancel **Generate Personal Key**

- ▶ **Key Name:** Enter a unique name for your key.
- ▶ **Expiration:** Choose the expiration date for the key.



- ▶ Services Included: Choose from the available services the key is permitted to access. Refer to [Assigning Services to Your Personal API Key](#) to learn more about each service and when to assign service access to your Personal Key.
6. Click Generate Personal Key when finished.
- Your API key appears in the following dialog.
7. NGC does not save your key, so store it securely. You can copy your API Key to the clipboard by selecting Copy Personal Key or using the copy icon to the right of the API key.



You can generate up to eight personal keys and manage them from the Setup > Personal Keys dashboard. To activate or deactivate a key, click the Active toggle. The Actions (ellipsis) menu allows you to rotate or delete a personal key.

Key ID	Key Name	Expiration	Key Value	Active	Actions
[REDACTED]	My Personal API Key	03/14/2025	nvapi-stg-*****HT2	<input checked="" type="checkbox"/>	⋮
[REDACTED]	My Second Personal Key	03/14/2025	nvapi-stg-*****cTr	<input checked="" type="checkbox"/>	⋮

Display 25 results per page

⚠ Personal Keys are for accessing NGC service APIs for Secrets Manager, AI Foundation Models and Endpoints, NGC Catalog, and Private Registry

API Information
Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf. Click [Generate Personal Key](#) to create your own API Key. If you forget or lose your API key you can come back and suspend, rotate and/or delete your key.

5.2.2.1.1. Assigning Services to Your Personal API Key

The services you can assign to a personal API key depend on two factors:

- ▶ The services enabled for the NGC org where you generate the API key.
- ▶ The service roles assigned to you by your NGC org owner or administrator.

For example, consider an NGC org with the following services enabled:

The screenshot shows the 'Subscriptions' section of the NVIDIA Organization interface. It lists five active subscriptions with their respective icons, status, term date, auto-renewal settings, and type:

- NVIDIA Microservices**: Status Active, Term Date Not specified, Auto Renewal No, Type Not for resale.
- Private Registry**: Status Active, Term Date Not specified, Auto Renewal No, Type Not for resale.
- NVIDIA AI Enterprise for VMware Private AI Foundation**: Status Active, Term Date Not specified, Auto Renewal No, Type Not for resale.
- Cloud Functions**: Status Active, Term Date Not specified, Auto Renewal No, Type Not for resale.
- NVIDIA AI Enterprise Essentials**: Status Active, Term Date Not specified, Auto Renewal No, Type Not for resale.

An NGC user account might have the following access roles assigned:

The user profile includes the following information:

- Email: tbc
- Roles: NYCF VIEWER, NVIDIA AI ENTERPRISE VIEWER, ORG OWNER, REGISTRY ADMIN, REGISTRY READ, USER ADMIN, USER READ
- Date Created: 04/05/2022 03:51 PM
- Last Login: 12/02/2024 08:42 PM

In this scenario, the NGC org has enabled NVIDIA Microservices, Private Registry, NVIDIA AI Enterprise, and Cloud Functions (NVCF). The user account has been granted access roles for all these services. Therefore, a personal API key can be generated with permissions to access one or all of them.

The dialog box contains the following fields and information:

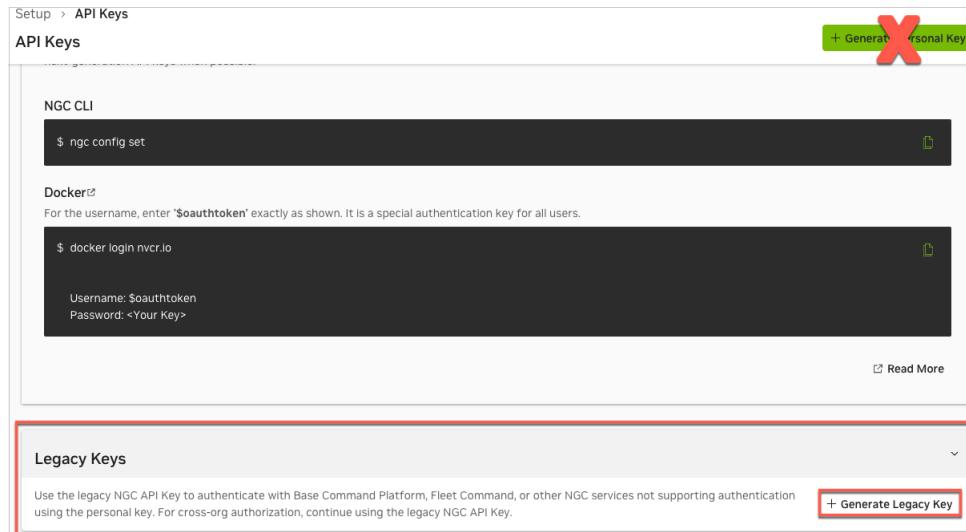
- Key Name ***: My personal API key for all services
- Expiration ***: 12 months
- Services Included ***: Secrets Manager, NGC Catalog, Public API Endpoints, Cloud Functions, Private Registry
- A note at the bottom states: "These services are based on your access within this organization"
- Buttons: Cancel, Generate Personal Key

If a service is unavailable for assignment to the API key, it indicates that the org owner or administrator has not granted the user the necessary role for that service.

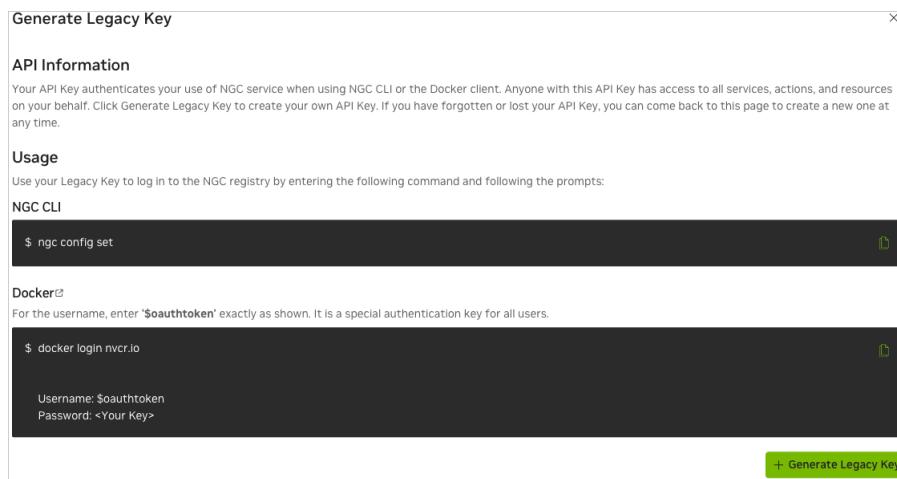
For details about each service listed above and its function, see the table [Supported NGC Applications and API Key Types](#).

5.2.2.1.2. Generating a Legacy NGC API Key

To generate a legacy API key, go to Setup > API Keys and click + Generate Legacy Key in the Legacy Keys drop-down.



In the Generate Legacy Key dialog, click on + Generate Legacy Key.



5.2.2.2. Generating a Service API Key

1. Sign in to the NGC website.

From a browser, go to <https://ngc.nvidia.com/signin> and then enter your email and password.

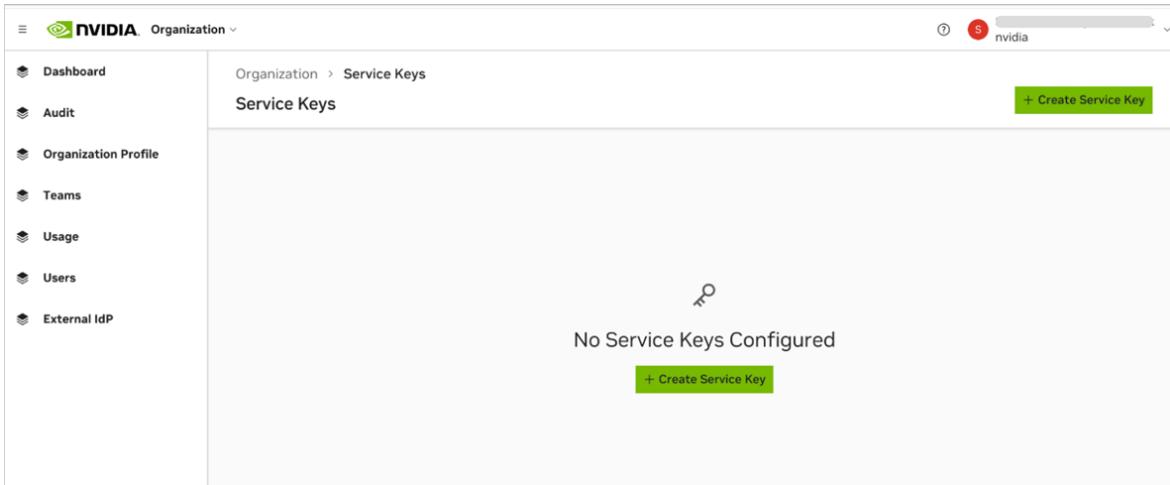
2. Select Organization from the user account menu on the upper right.

The screenshot shows the NVIDIA NGC Catalog homepage. On the right side, there is a user account menu with a red box around the 'Organization' option. Below the menu, there are links for 'Terms of Use' and 'Privacy Policy'.

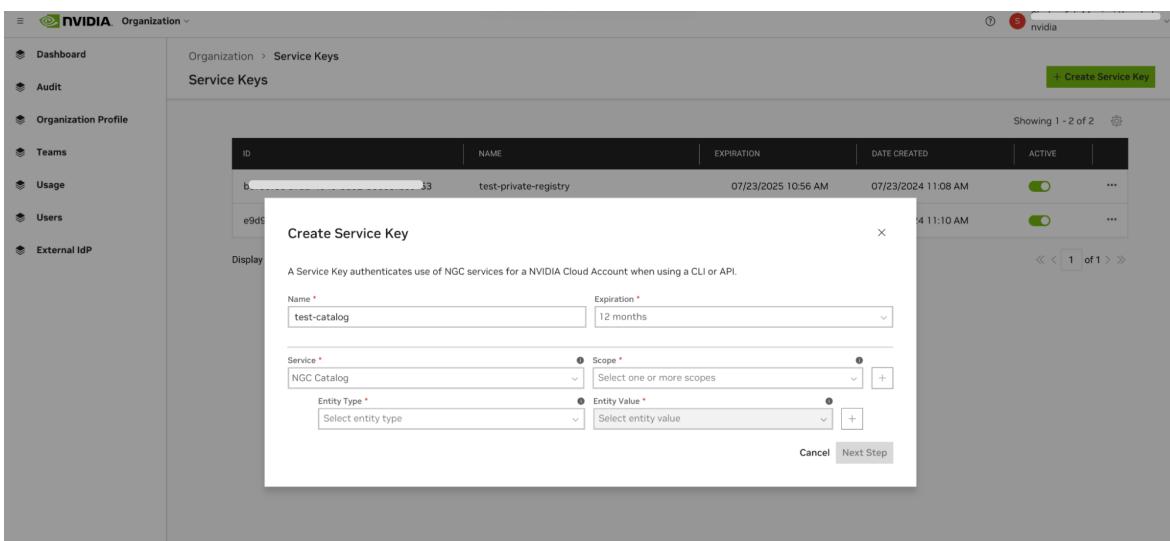
Select Service Keys on the organization dashboard.

The screenshot shows the NVIDIA Organization dashboard. On the left, there is a sidebar with various navigation options. In the center, there are several cards: 'External IdP', 'Organization Profile', 'Teams', 'Users', 'Usage', 'User Activity', and 'Service Keys'. The 'Service Keys' card is highlighted with a red box.

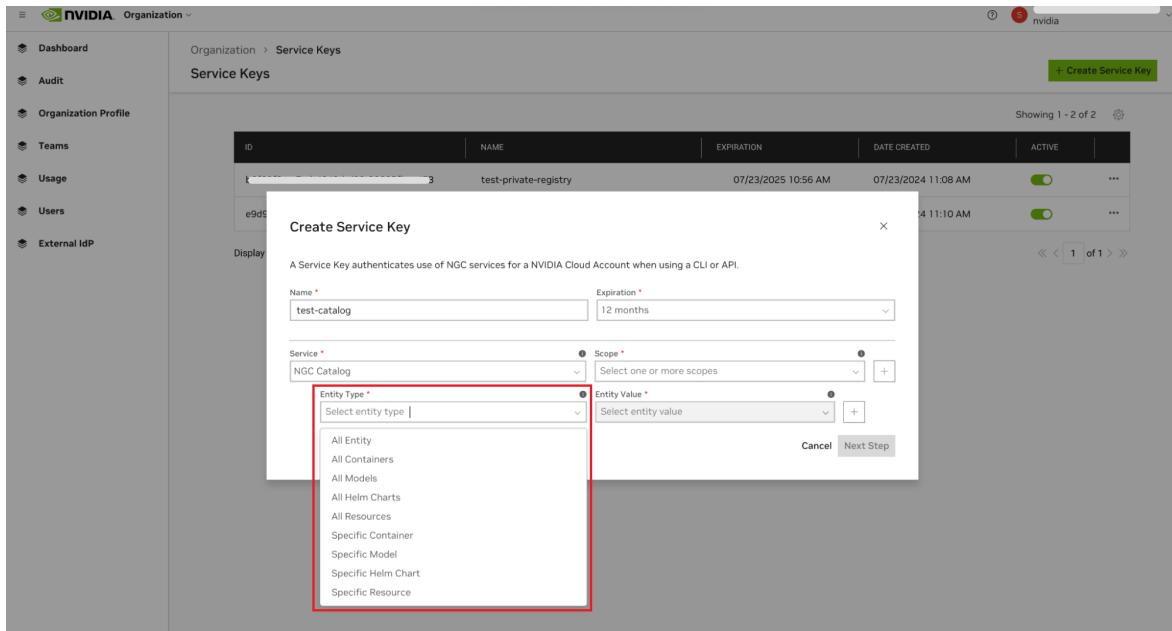
3. On the Organization > Service Keys page, click + Create Service Key button to create a key.



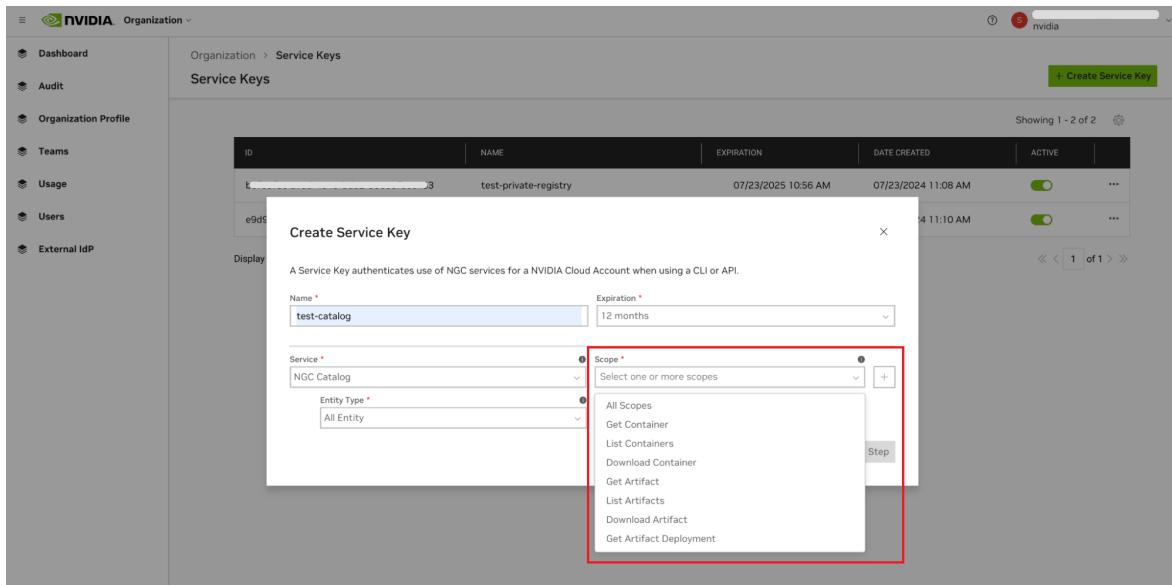
4. In the Create Service Key dialog, fill in the required configuration. Service keys currently support services such as NVIDIA NIM, NGC Catalog, and Private Registry. Assign scopes and resource permissions to the key.



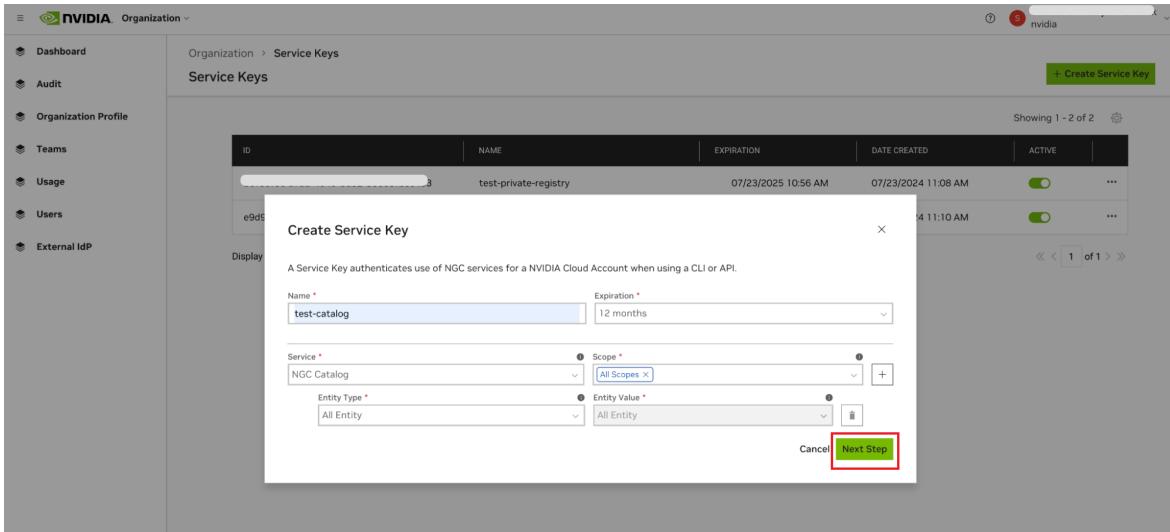
In the Entity Type field, select from the available options to grant to the API key.



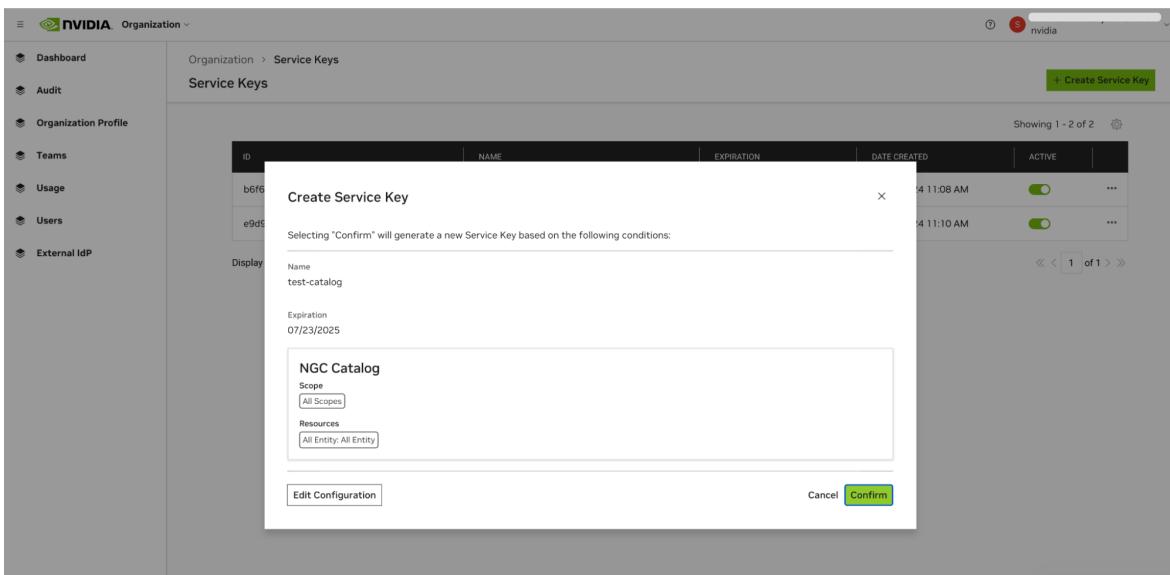
In the Scope field, choose from the available options.



5. Click Next Step to review your key configuration.



- Once you verified the configuration, click Confirm to generate your service key. Your service key appears in the next dialog.



- NGC does not save your key, so store it securely. You can copy your API Key to the clipboard by clicking the copy icon to the right of the API key or the Copy Service Key button.

The screenshot shows the NVIDIA Organization Service Keys dashboard. A modal window titled "Create Service Key" is open, displaying a generated service key. The key information includes:

ID	Name	Expiration	Date Created	Active
2f2e...00000000000000000000000000000000	test-catalog	07/23/2025 10:56 AM	07/23/2024 11:23 AM	<input checked="" type="checkbox"/>

The modal also contains a note: "Keep your Service Key a secret. Do not share it or store it in a place where others can see or copy it." A red box highlights the "Copy Service Key" button.

Make sure to copy the key value before leaving this page. Once you navigate away, the key value cannot be retrieved, and replacing it will require generating a new key.

NGC supports multiple Service API keys, which are managed from the Organization > Service Keys dashboard.

To activate or deactivate a key, click the Active toggle. The Actions (ellipsis) menu allows you to rotate or delete a service key.

The screenshot shows the NVIDIA Organization Service Keys dashboard with three active service keys listed:

ID	Name	Expiration	Date Created	Active
223c...00000000000000000000000000000000	test-catalog	07/23/2025 10:56 AM	07/23/2024 11:23 AM	<input checked="" type="checkbox"/>
b6fc...00000000000000000000000000000000	test-private-registry	07/23/2025 10:56 AM	07/23/2024 11:08 AM	<input checked="" type="checkbox"/>
e9c...00000000000000000000000000000000	test-public-catalog	07/23/2025 10:56 AM	07/23/2024 11:10 AM	<input checked="" type="checkbox"/>



Note: When managing containers, ensure the scopes Get Container and Get Container list are assigned to your service key. For other types of artifacts, add the Get Artifact and Get Artifact list scopes. These scopes are the minimum required to discover the artifacts that need to be managed. Refer to the [NGC Catalog User Guide](#) and [Private Registry User Guide](#) for more information.

Chapter 6. Introduction to the NGC Catalog and NGC CLIs

Unified NGC and Enterprise Catalog



Note: The Enterprise Catalog, formerly a separate repository for NVIDIA AI Enterprise-supported software, is now integrated into the public NGC Catalog, providing a unified and comprehensive platform. NVIDIA AI Enterprise customers can access their exclusive software and features within the NGC Catalog using their active entitlements.

The NGC Catalog aims to provide a centralized catalog of publicly available entities (e.g., containers, models, resources) alongside those that are part of products called *entitled* entities. This approach enables users to search and filter seamlessly across all entities, for a more efficient and improved user experience.

Users can view and download entitled entities by signing in to NGC. The NGC CLI is also available for downloading software using the API key. Access to all granted products will remain even when switching org/team context. Unauthenticated users will see a prompt to log in or gain access to the product when attempting to download gated features or entitled entities.

Publishers can publish and map entities to products. Access to entities is restricted by entity type, entity access type, user subscriptions, and roles, enhancing security and control. For entitled entities, guest users are encouraged to convert to registered or subscribed status to access product-specific entities.

Introduction to NGC CLIs

The NGC CLIs are command-line interfaces for managing content within the NGC Registry. The CLI operates within a shell and lets you use scripts to automate commands.

NGC Catalog CLI

The NGC Catalog CLI is available to you if you have guest access to the NGC Registry, and with it, you can

- ▶ View a list of GPU-accelerated Docker container images, pre-trained deep-learning models, and scripts for creating deep-learning models.
- ▶ Download container images, models, and resources.

NGC Registry CLI

The NGC Registry CLI is available to you if you are logged in with your own NGC account or with an NGC Private Registry account, and with it, you can

- ▶ View a list of GPU-accelerated Docker containers available and detailed information about each image.
- ▶ See a list of deep-learning models and resources and detailed information about them.
- ▶ Download container images, models, and resources.
- ▶ Upload container images, models, and resources.
- ▶ Create and manage users and teams (available to NGC Private Registry administrators).

For more details and best practices, visit the [NGC CLI documentation page](#).

6.1. Installing NGC Catalog CLI

To install NGC Catalog CLI,

1. Enter the NGC website (<https://ngc.nvidia.com>) as a guest user.
2. In the top right corner, click Welcome Guest and then select Setup from the menu.
3. Click Downloads under CLI from the Setup page.
4. From the CLI Install page, click the Windows, Linux, or macOS tab, according to the platform you will be running NGC Catalog CLI.
5. Follow the instructions to install the CLI.
6. Verify the installation by entering `ngc --version`.
The output should be `NGC CLI x.y.z` where x.y.z indicates the version.

6.2. Installing NGC Registry CLI

To install NGC Registry CLI,

1. Log in to your enterprise account on the NGC website (<https://ngc.nvidia.com>).
2. In the top right corner, click your user account icon and select Setup, then click Downloads under CLI from the Setup page.
3. From the CLI Install page, click the Windows, Linux, or macOS tab, according to the platform from which you will be running NGC Registry CLI.
4. Follow the instructions to install the CLI.
5. Verify the installation by entering `ngc --version`.
The output should be `NGC CLI x.y.z` where x.y.z indicates the version.

Chapter 7. Docker Containers

Over the last few years there has been a dramatic rise in the use of software containers for simplifying deployment of data center applications at scale. Containers encapsulate an application along with its libraries and other dependencies to provide reproducible and reliable execution of applications and services without the overhead of a full virtual machine.

GPU support within Docker containers enables GPU-based applications that are portable across multiple machines in a similar way to how Docker enables CPU-based applications to be deployed across multiple machines.

Docker container

A Docker container is an instance of a Docker image. A Docker container deploys a single application or service per container.

Docker image

A Docker image is simply the software (including the filesystem and parameters) that you run within a Docker container.

7.1. What Is a Docker Container?

A Docker container is a mechanism for bundling a Linux application with all of its libraries, data files, and environment variables so that the execution environment is always the same, on whatever Linux system it runs and between instances on the same host.

Unlike a VM which has its own isolated kernel, containers use the host system kernel. Therefore, all kernel calls from the container are handled by the host system kernel. Users can use Docker containers as the mechanism for deploying deep learning frameworks on-prem,, in the cloud, or at the edge.

A Docker container is the running instance of a [Docker image](#).

7.2. Why Use A Container?

Using containers allows you to package your application, dependencies, and environment variables into a single image rather than on each system you run on. Additional benefits to using containers include:

- ▶ Multiple versions of the same application, which may have conflicting software dependencies, can run on the same server.
- ▶ Containerized applications can be deployed on premise, in the cloud, or at the edge
- ▶ Specific GPU resources can be allocated to a container for isolation and better performance.
- ▶ Easily share, collaborate, and test applications across different environments.
- ▶ Resolve network-port conflicts between applications by mapping container-ports to specific externally-visible ports when launching the container.

7.3. Searching and Filtering Containers

To view NGC containers, navigate to Catalog > Containers in the left menu. Use the search bar at the top of the page to find a particular container or sort the list using the criteria from the dropdown menu. You can refine your search by utilizing the filters under NVIDIA Platform, NVIDIA AI Enterprise, Quick Deploy, and many others. The descriptions for the filtering options are as follows:

- ▶ Use Case: Choose from the available common use cases.
- ▶ NVIDIA Platform: Choose from Clara, DeepStream, TensorFlow, and many others.
- ▶ NVIDIA AI Enterprise Support: Select this option to find publicly available containers supported by NVIDIA AI Enterprise containers.
- ▶ NVIDIA AI Enterprise Exclusive: Choose products under this to see entitled entities.
- ▶ Quick Deploy: Select this option to find containers to quickly deploy your artifacts in a Jupyter environment.
- ▶ Industry: Select by available industries.
- ▶ Solution: Select the solution that closely matches your use case.
- ▶ Publisher: Filter by publisher.
- ▶ Architecture: Choose this option for multi-architecture support.
- ▶ Signed Images: Choose this option for container images with a digital signature.
- ▶ Multi-Node: Choose this option for multi-node support.
- ▶ Other: Miscellaneous options.

The following page displays a list of sample containers for guest users:

The screenshot shows the NGC Catalog interface for Docker Containers. On the left is a sidebar with the following navigation items:

- Explore Catalog
- Enterprise Catalog
- Collections
- Containers** (highlighted)
- Helm Charts
- Models
- Resources

The main content area has a title "Containers" and a subtitle: "The NGC catalog hosts containers for AI/ML, metaverse, and HPC applications and are performance-optimized, tested, and ready to deploy on GPU-powered on-prem, cloud, and edge systems." There is a search bar labeled "Search containers..." and a dropdown menu set to "Most Popular".

A sidebar on the left lists various filtering categories under "NVIDIA AI Enterprise Support":

- NVIDIA AI Enterprise Supported (62)
- NVIDIA AI Enterprise Exclusive (17)
- NVIDIA AI Enterprise Essentials (17)
- NVIDIA AI Enterprise for VMware (8)
- Private AI Foundation (8)
- NVIDIA Microservices (8)
- NVIDIA AI Enterprise IGX (5)

Below these are other filter options:

- Note: NVIDIA Riva includes Essentials
- Use Case
- NVIDIA Platform
- Quick Deploy
- Industry
- Solution
- Publisher
- Architecture
- Signed Images (1)
- Multi-Node
- Other

The main content area displays 277 containers in a grid format. The first two items shown are:

- DCGM Exporter**: Monitor GPUs in Kubernetes using NVIDIA DCGM. This is an exporter for a Prometheus monitoring solution in...
View Labels Learn More
- NVIDIA Kubernetes Device Plugin**: The NVIDIA Kubernetes Device Plugin registers GPUs as compute resources in the Kubernetes cluster.
View Labels Learn More

The third item is partially visible:

- Validator for NVIDIA GPU Operator**: Validates NVIDIA GPU Operator components
View Labels Learn More

The fourth item is also partially visible:

- NVIDIA GPU Feature Discovery for ...**: Plugin for the Kubernetes Node Feature Discovery for adding GPU node labels.
View Labels Learn More

Select a container from the list to view its details. All users can access the overview information, regardless of their sign-in status. Here's an example of an NVIDIA AI Enterprise Supported container:

Catalog > Containers > Morpheus

Morpheus

[Get Container](#)

Features

- NVIDIA AI Enterprise Supported
- Signed Images

Description

NVIDIA Morpheus is an open AI application framework for cybersecurity developers.

Publisher

NVIDIA

Latest Tag

v24.03.02-runtime

Modified

May 7, 2024

Compressed Size

7.58 GB

Multinode Support

No

Multi-Arch Support

No

v24.03.02-runtime (Latest) Security Scan Results

Linux / amd64 A >

Overview Tags Layers Security Scanning Related Collections

Morpheus

Morpheus allows teams to build their own optimized pipelines that address cybersecurity and information security use cases. Morpheus provides development capabilities around dynamic protection, real-time telemetry, adaptive policies, and cyber defenses for detecting and remediating cybersecurity threats.

Getting Started

Prerequisites

- Volta architecture or better (CUDA Compute Capability \geq 7.0)
- NVIDIA driver 525.60.13 or higher (CUDA 12.1)
- Docker
- The NVIDIA Container Toolkit

Installation

Pre-built runtime Docker image

Pre-built Morpheus Docker images can be downloaded from NGC. The runtime image includes pre-installed Morpheus and its dependencies:

```
docker pull nvcr.io/nvidia/morpheus:morpheus:24.03-runtime
```

There is also a [Helm chart](#) for deploying the Morpheus runtime container as a pod into a Kubernetes cluster.

Note: You must log into the NGC public catalog to download the Morpheus image. For more information see [this guide for accessing NGC](#).

There are two [CalVer](#) tags for the Morpheus runtime image that is published to NGC but they ultimately refer to the same latest digest for a release image:

Users must sign in to view the content for the Security Scanning tab.

Catalog > Containers > Morpheus

Morpheus

[Get Container](#)

Features

- NVIDIA AI Enterprise Supported
- Signed Images

Description

NVIDIA Morpheus is an open AI application framework for cybersecurity developers.

Publisher

NVIDIA

Overview Tags Layers **Security Scanning** Related Collections

Sign in to see security scanning reports for this container

You will then also have download access.

Sign In

Similarly, when filtering on NVIDIA AI Enterprise Essentials, the available containers will be displayed:

Catalog > Containers

Containers

The NGC catalog hosts containers for AI/ML, metaverse, and HPC applications and are performance-optimized, tested, and ready to deploy on GPU-powered on-prem, cloud, and edge systems.



Search containers...

Most Popular ▾

NVIDIA AI Enterprise Support ⓘ

NVIDIA AI Enterprise Exclusive ⓘ

- NVIDIA AI Enterprise Essentials 17
- NVIDIA AI Enterprise for VMware 9
- Private AI Foundation 7
- NVIDIA Microservices 7
- NVIDIA AI Enterprise IGX 5

Note: NVIDIA Riva includes Essentials

Displaying 17 containers

NVIDIA AI Enterprise Essentials X Clear filters

PyTorch Accelerated with NVIDIA

PyTorch PB October 2023 (PB 23h2) PyTorch Production Branch October 2023 (PB 23h2) offers a 9-month lifecycle for API stability, with monthly patches for...

NVIDIA AI Enterprise Essentials

NVIDIA AI Enterprise Supported +1

View Labels Learn More

TRITON INFERENCE SERVER

Triton Inference Server PB Octobe... Triton Inference Server Production Branch October 2023 (PB 23h2) offers a 9-month lifecycle for API stability, with monthly...

NVIDIA AI Enterprise Essentials

NVIDIA AI Enterprise Supported +1

View Labels Learn More

TensorFlow Accelerated with NVIDIA

TensorFlow Production Branch TensorFlow Production Branch offers a 9-month lifecycle for API stability, with monthly patches for...

View Labels Learn More

Once you choose a container, it shows the information in the Overview tab:

Catalog > Containers > PyTorch PB October 2023 (PB 23h2)

PyTorch PB October 2023 (PB 23h2)

[Get Container](#)

Associated Products

NVIDIA AI Enterprise Essentials

Features

NVIDIA AI Enterprise Supported

Signed Images

Description

PyTorch Production Branch October 2023 (PB 23h2) offers a 9-month lifecycle for API stability, with monthly patches for high and critical software vulnerabilities.

Publisher

Overview Tags Layers Security Scanning Related Collections

What Is PyTorch?

PyTorch is a GPU-accelerated tensor computational framework that offers a high degree of flexibility and speed for deep learning. It integrates seamlessly with popular Python libraries such as NumPy, SciPy, and Cython, extending its functionality to meet the diverse needs of users. PyTorch also employs a tape-based system for automatic differentiation at both the functional and neural network layer level, ensuring accelerated NumPy-like functionality.

What Is PyTorch Production Branch October 2023?

The PyTorch Production Branch, exclusively available with NVIDIA AI Enterprise, is a 9-month supported, API-stable branch that includes monthly fixes for high and critical software vulnerabilities. This branch provides a stable and secure environment for building your mission-critical AI applications. The PyTorch production branch releases every six months with a three-month overlap in between two releases.

If users are not subscribed to NVIDIA AI Enterprise Software Exclusive products, they'll see the following message when viewing other tabs:

Catalog > Containers > IGX - PyTorch PB October 2023 (PB 23h2)

IGX - PyTorch PB October 2023 (PB 23h2)

[Get Container](#)

Associated Products

NVIDIA AI Enterprise IGX

Features

NVIDIA AI Enterprise Supported

Signed Images

Description

IGX PyTorch Production Branch October 2023, part of NVIDIA AI Enterprise - IGX and purpose-built for the NVIDIA IGX Orin

Overview Tags Layers Security Scanning Related Collections

This page requires an active subscription to an NVIDIA AI Enterprise product

[Learn More](#)

If users try to download the container, they'll see a message indicating that a subscription is required:

Catalog > Containers > IGX - PyTorch PB October 2023 (PB 23h2)

IGX - PyTorch PB October 2023 (PB 23h2)

Associated Products NVIDIA AI Enterprise IGX

Features NVIDIA AI Enterprise Supp, Signed Images

Description IGX PyTorch Production Branch October 2023, part of NVIDIA AI and purpose-built for the platforms, offers a 9-month API stability, with monthly patches for high and critical software vulnerabilities.

Publisher Facebook

Tags Overview, Tags, Layers, Security Scanning, Related Collections

Subscription Required

Subscribe to the product below to access this premium entity:

NVIDIA AI Enterprise IGX

NVIDIA IGX is a high-performance, AI platform, featuring enterprise software and support. Purpose-built for industrial and medical environments, IGX delivers industry-leading performance, security, and functional safety.

Note: You can gain access to hundreds more GPU-optimized artifacts by [creating a free NGC account](#).

Already Subscribed? Log In [Subscribe Now](#)

However, once users subscribe to NVIDIA AI Enterprise Software Exclusive products, they can view the container information.

Catalog > Containers > IGX - PyTorch PB October 2023 (PB 23h2)

IGX - PyTorch PB October 2023 (PB 23h2)

Associated Products NVIDIA AI Enterprise IGX

Features NVIDIA AI Enterprise Supported, Signed Images

Description IGX PyTorch Production Branch October 2023, part of NVIDIA AI Enterprise - IGX and purpose-built for the NVIDIA IGX Orion platforms, offers a 9-month lifecycle for API stability, with monthly patches for high and critical software vulnerabilities.

Publisher Facebook

Latest Tag 23.08.08-py3-dgpu

Tags Overview, Tags, Layers, Security Scanning, Related Collections

Search tags...

A public key is required to validate the signed images below. [View all public keys](#) maintained by NVIDIA.

23.08.08-py3-dgpu 04/30/2024 11:29 AM 8.46 GB 1 Architecture	nvcr.io/nvidia/igx/pytorch-pb23h2-igx:...
23.08.08-py3-igpu 04/30/2024 11:29 AM 5.97 GB 1 Architecture	nvcr.io/nvidia/igx/pytorch-pb23h2-igx:...
23.08.07-py3-igpu 03/27/2024 1:11 PM 5.97 GB 1 Architecture	nvcr.io/nvidia/igx/pytorch-pb23h2-igx:...
23.08.07-py3-dgpu 03/27/2024 1:11 PM 8.46 GB 1 Architecture	nvcr.io/nvidia/igx/pytorch-pb23h2-igx:...

Chapter 8. NGC Container Images in NGC Catalog

NGC Containers are designed to enable a software platform centered around minimal OS requirements, Docker and driver installation on the server or workstation, and provisioning of all application and SDK software in the NGC containers through the NGC container registry.

NGC Catalog is a curated set of fully integrated and optimized container images for Deep Learning, HPC and Visualization applications.

- ▶ Deep learning framework containers that take full advantage of NVIDIA GPUs in both single GPU and multi-GPU configurations. They include CUDA Toolkit, DIGITS workflow, and deep learning frameworks: NVCaffe, Caffe2, Microsoft Cognitive Toolkit (CNTK), MXNet, PyTorch, TensorFlow, Theano, and Torch. These framework containers are delivered ready-to-run, including all necessary dependencies such as CUDA runtime, NVIDIA libraries, and an operating system. NVIDIA updates these deep learning containers monthly to ensure they continue to provide peak performance.
- ▶ NGC also hosts a catalog of HPC applications such as NAMD, GROMACS, LAMMPS, RELION, CHROMA, MILC, and many more.
- ▶ In addition to HPC applications, NGC hosts the industry's leading visualization tools, including ParaView with NVIDIA IndeX volume renderer, NVIDIA OptiX ray-tracing library, and NVIDIA Holodeck for interactive real-time visualization and high-quality visuals.
- ▶ NGC also hosts popular third-party GPU-ready application containers which conform to NGC container standards and best practices, making it easy to get the most out of your NVIDIA GPU.
- ▶ With the NGC Quick Deploy feature, many NGC containers can be used as a kernel to launch a JupyterLab instance on Google Cloud Vertex AI Workbench with optimal configuration and all software dependencies preload.

8.1. Why NGC Container Images?

Every image hosted on NGC Catalog undergoes a security scan and GPU-performance test by the NGC team.

8.1.1. NGC Container Security Policy

A critical function of enterprise risk and security assessment processes is to ensure all container images being used in development or deployment have been tested for known vulnerabilities. Containers published to the NGC Catalog undergo scanning with the NGC Container Security Policy.

The security scans include checks such as the following:

- ▶ *Outdated software packages, such as vulnerability scans*
- ▶ *Metadata checks, such as open ports specified in Dockerfiles, etc*
- ▶ *Cryptographic key material leaks*

The scanning policy for CVEs rates the severity into critical, high, medium, and low vulnerabilities using the Common Vulnerability Scoring System (CVSS). Scan results and NVIDIA Security-Risk Scale are made available to users. Published images are rescanned every 30 days to reflect the newest CVE findings.

8.1.2. NVIDIA Security Risk Scale

The NVIDIA Security-Risk Scale rates the risk level a given container image poses for a development/production infrastructure environment.

This rating takes into account vulnerability scans for OS and non-OS packages.

Scale	Description	Packages	Details
AAA	Rated to have lowest security risk.	0	No package with vulnerabilities
AA	Rated to have lower security risk.	1	1 package has a Critical or High vulnerability
A	Rated to have moderate security risk with some speculative judgment needed based on environment workloads	2-3	2-3 packages with Critical or High vulnerabilities
B	Rated to have high security risk with some speculative judgment needed based on environment workloads	3+	4-5 packages with Critical or High vulnerabilities
C	Rated as highest security risk	5+	5+ packages with Critical or High vulnerabilities

8.1.3. GPU-performance tests for NGC Catalog Container Images

To ensure that NVIDIA GPU owners can perform well with NGC hosted images, all container images undergo basic installation, functional, scalability and performance tests on GPUs.

These tests are performed on an array of GPUs ranging from NVIDIA T4s to A100 to DGX Systems for performance and multi-GPU for scale. The selection of supported GPUs may be limited to few architectures or single-/multi-GPU based on the business use case of application containers.

These tests are performed on both bare-metal installations and cloud deployed instances to ensure both on-prem and cloud deployment use-cases.

8.2. NVIDIA Container Toolkit

The NVIDIA Container Toolkit allows users to build and run GPU accelerated containers. The toolkit includes a container runtime [library](#) and utilities to automatically configure containers to leverage NVIDIA GPUs. It integrates with many popular container runtimes including Docker, podman, CRI-O, LXC etc.

To enable portability in Docker images that leverage NVIDIA GPUs, NVIDIA developed [nvidia-docker](#), an open-source project hosted on Github that provides the two critical components needed for portable GPU-based containers:

1. Driver-agnostic CUDA images.
2. A Docker command line wrapper that mounts the user mode components of the driver and the GPUs (character devices) into the container at launch.

`nvidia-docker` is a Docker command line wrapper that provisions a container with the necessary components to execute code on the GPU.

When working with containers that utilize GPUs, the only command that must be executed through an `nvidia-docker` command is the `run` command. For all other functionalities, `docker` commands can be used.

But for simplicity in this documentation we use `nvidia-docker` for all commands.

8.3. NVIDIA CUDA Toolkit

The NVIDIA® CUDA® Toolkit provides a development environment for creating high-performance GPU-accelerated applications. With the CUDA Toolkit, you can develop, optimize, and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms, and HPC supercomputers.

The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler and a runtime library to deploy your application.

GPU-accelerated CUDA libraries enable drop-in acceleration across multiple domains such as linear algebra, image and video processing, deep learning, and graph analytics. For developing custom algorithms, you can use available integrations with commonly used languages and numerical packages as well as well-published development APIs. Your CUDA applications can be deployed across all NVIDIA GPU families available on-premise and on GPU instances in the cloud. Using built-in capabilities for distributing computations across multi-GPU configurations, scientists and researchers can develop applications that scale from single GPU workstations to cloud installations with thousands of GPUs.

8.4. Running Singularity Containers

NGC supports both Docker and Singularity container runtimes. While Docker is prevalent primarily in enterprises, Singularity has become ubiquitous in the HPC community. Singularity was developed to better satisfy the requirements of HPC users and system administrators, including the ability to run containers without superuser privileges.

NGC containers can be easily used with Singularity. Let's use the [NGC NAMD container](#) to illustrate. NAMD is a parallel molecular dynamics application designed for high-performance simulation of large biomolecular systems, developed by the Theoretical and Computational Biophysics Group at the University of Illinois.

8.4.1. Prerequisites

These instructions assume the following.

- ▶ You have Singularity v2.6+ [installed](#) on your system
- ▶ You have performed the following steps from the NGC website.
 - ▶ Signed up for an NGC account at <https://ngc.nvidia.com/signup>.
 - ▶ Created an NGC API key for access to the NGC container registry
 - ▶ Browsed the NGC website and identified an available NGC container and tag to run
- ▶ Ensure you have correctly set udev rules which are detailed ([here](#))



Note:

It is recommended that you install `nvidia-container-cli` because if installed, Singularity will use it. More information can be found [here](#).

8.4.2. Converting to Singularity Image

Before running with Singularity you must set NGC container registry authentication credentials.

This is most easily accomplished by setting the following environment variables.

bash

```
$ export SINGULARITY_DOCKER_USERNAME='$oauthtoken'
$ export SINGULARITY_DOCKER_PASSWORD=<NVIDIA NGC API key>
```

tcsh

```
> setenv SINGULARITY_DOCKER_USERNAME '$oauthtoken'
> setenv SINGULARITY_DOCKER_PASSWORD <NVIDIA NGC API key>
```

More information describing how to obtain and use your NVIDIA NGC API key can be found [here](#).

Once credentials are set in the environment, the NGC container can be pulled to a local Singularity image.

```
$ singularity build <app_tag>.simg docker://nvcr.io/<repository>/<app:tag>
```

This will save the container to the current directory as

```
<app_tag>.simg
```

For example to convert the HPC application NAMD hosted on NGC to a Singularity image, run

```
$ singularity build namd_2.12-171025.simg docker://nvcr.io/hpc/namd:2.12-171025
```

After the build has finished the Singularity image file, namd_2.12-171025.simg, will be available for use in the current working directory.

8.4.3. Running the Singularity Container

Once the local Singularity image has been pulled, the following modes of running are supported.

- ▶ [Command line execution with Singularity](#)
- ▶ [Interactive shell with Singularity](#)

To leverage NVIDIA GPUs, you must use the Singularity flag `--nv` when running the containers. More singularity flags are explained [here](#).



Important:

For Amazon Machine Image Users

Amazon Machine Images on Amazon Web Service have a default root umask of 077. Singularity must be installed with a umask of 022 to run properly. To (re)install Singularity with correct permissions:

- ▶ Uninstall Singularity (if it is installed)
- ▶ Change the umask with: `$ umask 0022`
- ▶ Install Singularity
- ▶ Restore the umask: `$ umask 0077`

This causes installed Singularity files to have permission 0755 instead of the default 0700. Note that the umask command only applies changes to the current shell. Use umask and install Singularity from the same shell session.

8.4.3.1. Directory Access

Singularity containers are themselves ostensibly read only. In order to provide application input and persist application output we'll bind a host directory into our container, this is accomplished through the Singularity `-B` flag. The format of this flag is `-B <host_src_dir>:<container_dst_dir>`. Once a host directory is bound into the container we can interact with this directory from within the container exactly as we can outside the container.

It is also often convenient to use the `--pwd <container_dir>` flag, which will set the present working directory of the command to be run within the container.

The Singularity commands below will mount the present working directory on the host to `/host_pwd` in the container process and set the present working directory of the container process to `/host_pwd`. With this set of flags the `<cmd>` to be run will be launched from the host directory Singularity was called from.

```
$ singularity exec --nv -B $(pwd):/host_pwd --pwd /host_pwd <image.simg> <cmd>
```



Note:

Binding to a directory which doesn't exist within the container image requires kernel and configuration support that may not be available on all systems, particularly those running older kernels such as CentOS/RHEL 6. When in doubt contact your system administrator.

8.4.3.2. Command Line Execution with Singularity

Running the container with Singularity from the command line looks similar to the command below.

```
$ singularity exec --nv <app_tag>.simg <command_to_run>
```

For example, to run the NAMD executable in the container

```
$ singularity exec --nv namd_2.12-171025.simg /opt/namd/namd-multicore
```

8.4.3.3. Interactive Shell with Singularity

To start a shell within the container, run the command below

```
$ singularity exec --nv <app_tag>.simg /bin/bash
```

For example, to start an interactive shell in the NAMD container

```
$ singularity exec --nv namd_2.12-171025.simg
```

Chapter 9. Prerequisites for Using NGC Catalog Container Images

To enable portability in Docker images that leverage GPUs, three methods of providing GPU support for Docker containers have been developed.

- ▶ Running Docker-ce 19.03 or later for Native GPU support
- ▶ nvidia-docker2
- ▶ nvidia-docker

Each of these methods mount the user mode components of the NVIDIA driver and the GPUs into the Docker container at launch. They allow NGC containers to take full advantage of NVIDIA GPUs.

9.1. Prerequisites for Using NGC Containers on DGX Systems

DGX™ users should follow the instructions in the [Preparing Your DGX System For Use With NVIDIA Container Runtime](#).

9.2. Prerequisites for Using NGC Containers on Cloud Platforms

On each of the major public cloud providers, NVIDIA publishes customized virtual machine images (VMIs) with updated OS's and NVIDIA driver that have been tested with NGC containers. These VMIs have the NVIDIA Container Toolkit pre-installed allowing you to begin pulling and running Docker containers from NGC right away. To use the NVIDIA NGC GPU-optimized VMIs on cloud platforms, you would need -

- ▶ A cloud platform account with permissions to create resources. See the details for your public cloud:
 - ▶ Alibaba - <https://home-intl.console.aliyun.com/>
 - ▶ AWS - <https://aws.amazon.com>
 - ▶ Microsoft Azure - <https://portal.azure.com>
 - ▶ Google Cloud Platform (GCP) - <https://console.cloud.google.com/>
- ▶ A CLI for respective cloud platforms if you wish to interface with your VM via a CLI
 - ▶ [Alibaba Cloud CLI](#)
 - ▶ [AWS CLI Version 2](#)
 - ▶ [Azure CLI 2.0](#)
 - ▶ [gcloud SDK](#)
- ▶ Windows Users: Any CLI code snippets in the docs are for bash on Linux or Mac OS X. If you are using Windows and want to use the snippets as-is, you can [set up the Windows Subsystem for Linux](#) and use the bash shell (you will be in Ubuntu Linux).

9.3. Prerequisites for Using NGC Containers on Other NVIDIA GPUs

- ▶ Users running NGC containers on other NVIDIA GPUs or in a virtual GPU environment should follow the corresponding instructions for [TITAN PCs, Quadro PCs, or vGPUs](#).
- ▶ Other users should follow the nvidia-docker installation documentation at [nvidia-docker installation](#) and install the latest NVIDIA drivers for your GPU product type and series for your operating system. If NVIDIA drivers are not already configured on your system, then install them from here: [Download Drivers](#).
- ▶ Ensure you have an NVIDIA GPU supporting Compute Unified Device Architecture[®] (CUDA) version with compute capability 6.0.0 or higher. For example, Pascal GPU architecture generation or later.

9.4. HPC Visualization Containers

In addition to accessing the [NVIDIA optimized frameworks](#) and HPC containers, the NVIDIA GPU Cloud (NGC) container registry also hosts the following scientific visualization containers for HPC. These containers rely on the popular scientific visualization tool called [ParaView](#).

Visualization in an HPC environment typically requires remote visualization, that is, data resides and is processed on a remote HPC system or in the cloud, and the user graphically interacts with this application from their workstation. As some visualization

containers require specialized client applications, the HPC visualization containers consist of two components:

Server container

The server container needs access to the files on your server system. Details on how to grant this access are provided below. The server container can run both in serial mode or in parallel. For this alpha release, we are focusing on the serial node configuration. If you are interested in parallel configuration, contact hpcviscontainer@nvidia.com.

Client container

To ensure matching versions of the client application and the server container, NVIDIA provides the client application in a container. Similarly, to the server container, the client container needs access to some of the ports to establish connection with the server container.

In addition, the client container needs access to the users' X server for displaying the graphical user interface.

9.4.1. Prerequisites For HPC Visualization Containers

- ▶ Install [docker-ce](#) and [nvidia-docker2](#). First install [docker-ce](#), then install [nvidia-docker2](#) for your operating system and Docker version. For a script to install nvidia-docker2, see [Installing NVIDIA Docker 2.0](#).



Note: If you already have `nvidia-docker1` installed and intend to keep it, you can install `nvidia-container-runtime`.

- ▶ Install the NVIDIA Display driver version 384.57 or onwards depending on your GPU product type and series for your operating system. For more information, see [Download Drivers](#).
- ▶ Ensure you have an NVIDIA GPU supporting Compute Unified Device Architecture® (CUDA) version with compute capability 6.0.0 or higher. For example, Pascal GPU architecture generation or later.
- ▶ Log into the NVIDIA® GPU Cloud (NGC) Container Registry located at `nvcr.io` using your NGC API key. For step-by-step instructions on how to gain access and get your API key, see [Generating NGC API Keys](#).

9.4.2. Installing NVIDIA Docker 2.0

The following script installs NVIDIA Docker 2.0 which is a prerequisite to pulling the ParaView with NVIDIA IndeX HPC visualization container.

Full support for concurrent graphics and compute capabilities in containers is supported in NVIDIA Docker 2.0. Current installations of NGC run on NVIDIA Docker 1.0. Prior to using a container on any of these instances, NVIDIA Docker 2.0 must be installed.

Use the following script below to install NVIDIA Docker 2.0 on your instance.

```
# Install NVIDIA Docker 2.0
docker volume ls -q -f driver=nvidia-docker | \
  xargs -r -I{} -n1 docker ps -q -a -f volume={} | xargs -r docker rm -f
```

```

sudo apt-get purge -y nvidia-docker
curl -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
sudo tee /etc/apt/sources.list.d/nvidia-docker.list <<< \
"deb https://nvidia.github.io/libnvidia-container/ubuntu16.04/amd64 /
deb https://nvidia.github.io/nvidia-container-runtime/ubuntu16.04/amd64 /
deb https://nvidia.github.io/nvidia-docker/ubuntu16.04/amd64 /"

sudo apt-get -y update
sudo apt-get install -y nvidia-docker2
sudo pkill -SIGHUP dockerd

# Tests
#docker run --runtime=nvidia --rm nvidia/cuda nvidia-smi

```

9.4.3. ParaView With NVIDIA Holodeck

Currently, the ParaView with NVIDIA Holodeck container requires a running X server both on the server host and the client host. Therefore, only a single container image is required.

1. Create X-forwarding variables for your container.

```

XSOCK=/tmp/.X11-unix; XAUTH=/tmp/.docker.xauth;
touch /tmp/.docker.xauth;
xauth nlist :0 | sed -e 's/^....ffff/' | xauth -f /tmp/.docker.xauth nmerge -

```

2. On the server host, start the ParaView Holodeck server:

```

docker run --rm -it --runtime=nvidia \
-v /tmp/.X11-unix:/tmp/.X11-unix -v /tmp/.docker.xauth:/tmp/.docker.xauth \
-e XAUTHORITY=/tmp/.docker.xauth -e DISPLAY=:0 \
-p 11111:11111 \
--shm-size=4g \
nvcr.io/nvidia-hpcvis/paraview-holodeck:glx-17.11.13-beta \
./service.sh externalvis pvserver

```

The Holodeck render window showing a space scene displays.

The server container is ready after you receive a message similar to the following:

```
"Accepting connection(s) : [...] :11111"
```

3. Set up X access and start the client container on the client host. Ensure you replace `your_server_hostname`.

```

XSOCK=/tmp/.X11-unix; XAUTH=/tmp/.docker.xauth
touch /tmp/.docker.xauth
xauth nlist :0 | sed -e 's/^....ffff/' \
| xauth -f /tmp/.docker.xauth nmerge -docker run --rm -it --runtime=nvidia \
-v /tmp/.X11-unix:/tmp/.X11-unix -v /tmp/.docker.xauth:/tmp/.docker.xauth \
-e XAUTHORITY=/tmp/.docker.xauth -e DISPLAY=:0 \
nvcr.io/nvidia-hpcvis/paraview-holodeck:glx-17.11.13-beta \
sh -c paraview\ --server-url=cs://your_server_hostname:11111

```

The ParaView user interface displays.

9.4.4. ParaView With NVIDIA IndeX

To support both X-enabled and headless hosts, the ParaView IndeX container image is available with GLX and EGL support. The following section shows how to launch the IndeX container with different use cases.

For more information about ParaView, see the [ParaView User's Guide](#) and the [NVIDIA IndeX SDK](#).

9.4.4.1. Single-Machine With GLX

1. Login to the docker repository and pull the X display-enabled container on your workstation:

```
docker pull nvcr.io/nvidia-hpcvis/paraview-index:glx-17.11.13-beta
```

2. Specify X-forwarding variables:

```
XSOCK=/tmp/.X11-unix; XAUTH=/tmp/.docker.xauth
touch /tmp/.docker.xauth
xauth nlist :0 | sed -e 's/^....ffff/' \
| xauth -f /tmp/.docker.xauth nmerge
```

3. Run the image. In this example, host system data in the current directory \$(pwd) are mounted to both /work in the container. This should be modified as desired by the user.

```
docker run --rm -it --runtime=nvidia \
-v /tmp/.X11-unix:/tmp/.X11-unix -v /tmp/.docker.xauth:/tmp/.docker.xauth \
-v $(pwd):/work -e XAUTHORITY=/tmp/.docker.xauth -e DISPLAY=:0 \
nvcr.io/nvidia-hpcvis/paraview-index:glx-17.11.13-beta \
sh -c paraview
```

9.4.4.2. Server Container With EGL

In a typical client-server setup, one container acting as the server will run remotely on a display-less machine, connected to a second container that runs locally on a workstation and provides the graphical front end.

Use the following command to pull the EGL-enabled, no-display container from the NGC registry on the server host:

```
docker pull nvcr.io/nvidia-hpcvis/paraview-index:egl-17.11.13-beta
```

Run the server component on the server host. We listen on the default port 11111:

```
docker run --runtime=nvidia -p 11111:11111 --rm -it \
nvcr.io/nvidia-hpcvis/paraview-index:egl-17.11.13-beta sh -c pvserver
```

9.4.4.3. GLX Client Connecting To A Server

Pull the X display-enabled container on your workstation:

```
docker pull nvcr.io/nvidia-hpcvis/paraview-index:glx-17.11.13-beta
```

Set up X access and launch the client application container (make sure to replace your_server_hostname with the address of your ParaView server host):

```
XSOCK=/tmp/.X11-unix; XAUTH=/tmp/.docker.xauth
touch /tmp/.docker.xauth
xauth nlist :0 | sed -e 's/^....ffff/' \
| xauth -f /tmp/.docker.xauth nmerge -
docker run --rm -it --runtime=nvidia \
-v /tmp/.X11-unix:/tmp/.X11-unix -v /tmp/.docker.xauth:/tmp/.docker.xauth \
-e XAUTHORITY=/tmp/.docker.xauth -e DISPLAY=:0 \
nvcr.io/nvidia-hpcvis/paraview-index:glx-17.11.13-beta \
sh -c paraview\ --server-url=cs://your_server_hostname:11111
```

9.4.5. ParaView With NVIDIA OptiX

The ParaView with NVIDIA OptiX container is designed to run ParaView as a user normally would outside a container. The following sections show how to launch the OptiX container with different use cases.

For more information about ParaView see the [ParaView User's Guide](#) and the [NVIDIA OptiX SDK](#).

9.4.5.1. Single-Machine Container With GLX

On systems with a physical display, or when running a ParaView client, users will wish to launch a container with GLX support. This can be done as follows.

1. Pull the docker image:

```
docker pull nvcr.io/nvidia-hpcvis/paraview-optix:glx-17.11.13-beta
```

2. Set up X11 forwarding variables:

```
XSOCK=/tmp/.X11-unix; XAUTH=/tmp/.docker.xauth;
touch /tmp/.docker.xauth;
xauth nlist :0 | sed -e 's/^....ffff/' | xauth -f /tmp/.docker.xauth nmerge -
```

3. Run the image. In this example, host system data in the current directory \$(pwd) are mounted to both /work in the container. This should be modified as desired.

```
docker run --rm -it --runtime=nvidia -v /tmp/.X11-unix:/tmp/.X11-unix -v /
/tmp/.docker.xauth:/tmp/.docker.xauth -e XAUTHORITY=/tmp/.docker.xauth -e DISPLAY=:0 -v
$(pwd):/work:rw nvcr.io/nvidia-hpcvis/paraview-optix-glx-beta:17.11.10 sh -c paraview
```

9.4.5.2. Server Container With EGL

Launching a ParaView server on GPU HPC resources often requires EGL support, requiring a separate build of ParaView for which we have a separate container.

1. Pull the container:

```
docker pull nvcr.io/nvidia-hpcvis/paraview-optix:egl-17.11.13-beta
```

2. Specify the connection port and launch the container as follows (in this example, we listen on the default port 11111):

```
docker run --runtime=nvidia -p 11111:11111 --rm -it \
nvcr.io/nvidia-hpcvis/paraview-optix:egl-17.11.13-beta sh -c pvserver
```

3. For users who wish to run the server on a GLX-capable workstation, it is equally possible to use the GLX image with the `pvserver` argument.

9.4.5.3. Running The GLX Client And Attaching To The Server

With the server launched, it is then straightforward to use the GLX image to run a client, and connect to the server as follows. Here we assume the server is listening on port 11111, addressable at `your.server.address`.

```
docker pull nvcr.io/nvidia-hpcvis/paraview-optix:glx-17.11.13-beta

XSOCK=/tmp/.X11-unix; XAUTH=/tmp/.docker.xauth
touch /tmp/.docker.xauth
xauth nlist :0 | sed -e 's/^....ffff/' \
| xauth -f /tmp/.docker.xauth nmerge -

docker run --rm -it --runtime=nvidia \
-v /tmp/.X11-unix:/tmp/.X11-unix -v /tmp/.docker.xauth:/tmp/.docker.xauth \
```

```
-e XAUTHORITY=/tmp/.docker.xauth -e DISPLAY=:0 \
nvcr.io/nvidia-hpcvis/paraview-optix:glx-17.11.13-beta \
sh -c paraview\ --server-url=cs://your.server.address:11111
```

9.4.5.4. Optional: Using The ParaView .config File

It is helpful to reuse ParaView configuration files to maintain settings across ParaView sessions. To do this, first create a new directory for ParaView to store its settings.

```
mkdir pvsettings
```

When issuing the `docker run` command, add the following command as an argument:

```
-v $(pwd)/pvsettings:/home/paraview/.config/ParaView
```

Insert the command before the image URL. For example,

```
docker run --rm -it --runtime=nvidia \
-v /tmp/.X11-unix:/tmp/.X11-unix -v /tmp/.docker.xauth:/tmp/.docker.xauth \
-e XAUTHORITY=/tmp/.docker.xauth -e DISPLAY=:0 \
nvcr.io/nvidia-hpcvis/paraview-optix:glx-17.11.13-beta \
-v $(pwd)/pvsettings:/home/paraview/.config/ParaView \
sh -c paraview\ --server-url=cs://your.server.address:11111
```

Chapter 10. Pulling NGC Containers from NGC Catalog

Before you use a container from the NGC Catalog, review the prerequisites described in the [Prerequisites](#) section.

To become a registered NGC user, follow the steps explained in the [Registering and Activating a New NGC Org to Obtain Authenticated Access](#).

10.1. Key NGC Container Registry Terminology

In order to issue the pull and run commands, ensure that you are familiar with the following concepts.

A pull command looks similar to:

```
docker pull nvcr.io/nvidia/caffe2:17.10
```

A run command looks similar to:

```
docker run --gpus all -it --rm -v local_dir:container_dir nvcr.io/nvidia/caffe2:<xx.xx>
```



Note: The base command `docker run --gpu all` assumes that your system has Docker 19.03-CE installed. See the section [Enabling GPU Support for NGC Containers](#) for the command to use for earlier versions of Docker.

The following concepts describe the separate attributes that make up the both commands.

`nvcr.io`

The name of the container registry, which for the NGC container registry is `nvcr.io`.

`nvidia`

The name of the space within the registry that contains the container. For containers provided by NVIDIA, the registry space is `nvidia`. For more information, see [NGC Container Images in NGC Catalog](#).

`-it`

You want to run the container in interactive mode.

`--rm`

You want to delete the container when finished.

-v

You want to mount the directory.

local_dir

The directory or file from your host system (absolute path) that you want to access from inside your container. For example, the `local_dir` in the following path is `/home/jsmith/data/mnist`.

-v /home/jsmith/data/mnist:/data/mnist

If you are inside the container, for example, using the command `ls /data/mnist`, you will see the same files as if you issued the `ls /home/jsmith/data/mnist` command from outside the container.

container_dir

The target directory when you are inside your container. For example, `/data/mnist` is the target directory in the example:

-v /home/jsmith/data/mnist:/data/mnist**<xx.xx>**

The tag. For example, 17.10.

You can access the NGC container registry by running a Docker commands from any Linux computer with Internet access on which Docker is installed. You can access the NGC container registry at `nvcr.io` through the Docker CLI.

Before accessing the NGC container registry, see [NGC Getting Started Guide](#) for instructions on how to access the website and, if you intend to access locked NGC content, know how to sign up for an NGC account and obtain an API key.

10.2. Accessing And Pulling an NGC Container Image via the Docker CLI

10.2.1. Logging in to the NGC container registry

Before accessing locked NGC content, you must sign up for an NGC account and obtain an API key as explained in the section [Generating NGC API Keys](#). Then log in to the NGC registry from the command line as follows.

1. Log in to the NGC container registry.

\$ docker login nvcr.io

2. When prompted for your user name, enter the following text:

\$oauthtoken

The `$oauthtoken` username is a special user name that indicates that you will authenticate with an API key and not a username and password.

3. When prompted for your password, enter your NGC API key as shown in the following example.

Username: \$oauthtoken

Password: k7cqFTUvKKdiwGsPnWnyQFYGn1AlsCIRmlP67Qxa



Tip: When you get your API key, copy it to the clipboard so that you can paste the API key into the command shell when you are prompted for your password.

4. Confirm "Login Success".

10.2.2. Pulling a Container from NGC Container Registry Using Docker CLI

You can browse the available containers in the NGC container registry by opening the [NGC website](#) using a web browser

1. Browse the NGC Catalog, select the image to pull, then copy the pull command.

The following image shows the pull command on the PyTorch container page.

Catalog > Containers > PyTorch

PyTorch

Get Container ▾

Overview Tags Layers Security Scanning Related Collections

PyTorch

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs. Automatic differentiation is done with a tape-based system at both a functional and neural network layer level. This functionality brings a high level of flexibility and speed as a deep learning framework and provides accelerated NumPy-like functionality. NGC Containers are the easiest way to get started with PyTorch. The PyTorch NGC Container comes with all dependencies included, providing an easy place to start developing common applications, such as conversational AI, natural language processing (NLP), recommenders, and computer vision.

The PyTorch NGC Container is optimized for GPU acceleration, and contains a validated set of libraries that enable and optimize GPU performance. This container also contains software for accelerating ETL ([DALI](#), [RAPIDS](#)), Training ([cuDNN](#), [NCCL](#)), and Inference ([TensorRT](#)) workloads.

Prerequisites

Using the PyTorch NGC Container requires the host system to have the following installed:

- [Docker Engine](#)
- [NVIDIA GPU Drivers](#)
- [NVIDIA Container Toolkit](#)

For supported versions, see the [Framework Containers Support Matrix](#) and the [NVIDIA Container Toolkit Documentation](#).

No other installation, compilation, or dependency management is required. *It is not necessary to install the NVIDIA CUDA Toolkit.*

The PyTorch NGC Container is optimized to run on NVIDIA DGX Foundry and NVIDIA DGX SuperPOD managed by NVIDIA Base Command Platform. Please refer to the [Base Command Platform User Guide](#) to learn more about running workloads on BCP clusters.

Running PyTorch Using Docker

To run a container, issue the appropriate command as explained in the [Running A Container](#) chapter in the [NVIDIA Containers For Deep Learning Frameworks User's Guide](#) and specify the registry, repository, and tags. For more information about using NGC, refer to the [NGC Container User Guide](#).

If you have Docker 19.03 or later, a typical command to launch the container is:

```
docker run --gpus all -it --rm nvcr.io/nvidia/pytorch:xx.xx-py3
```

If you have Docker 19.02 or earlier, a typical command to launch the container is:

```
nvidia-docker run -it --rm -v nvcr.io/nvidia/pytorch:xx.xx-py3
```

2. Click the download icon to copy the pull command to the clipboard.
3. Log in to nvcr.io, then paste the command in your terminal.

For example, to pull the PyTorch container:

```
$ docker pull nvcr.io/nvidia/pytorch:20.03-py3
```

See the [NGC Getting Started Guide](#) for details on using the NGC website.

4. List the Docker images on your system to confirm that the container was pulled.

```
$ docker images
```

For more information pertaining to your specific container, refer to the /workspace/README.md file inside the container.

After pulling a container, you can run jobs in the container to run scientific workloads, train neural networks, deploy deep learning models, or perform AI analytics.

10.3. Accessing and Pulling an NGC Catalog Container Using NGC CLI

If you will be pulling containers from the registry, you will need Docker installed on your local machine. To install Docker on your client machine, follow one of these instructions:

- ▶ For Ubuntu Linux:

<https://docs.docker.com/engine/installation/linux/ubuntu/>

- ▶ For Windows:

<https://docs.docker.com/engine/installation/windows/#docker-for-windows>

- ▶ For MacOS X:

<https://docs.docker.com/engine/installation/mac/>

This document provides an introduction to using the NGC Catalog CLI. For a complete list of commands and options, use the -h option as explained in [#unique_60](#).

10.3.1. Viewing Container Image Information

There are several commands for viewing information about available container images.

To list container images:

```
C:\>ngc registry image list
```

Example output

Name	Repository	Latest Tag	Image Size	Updated Date	Permission
BigDFTZ	hpv/bigdftfe	cuda10-ubun	2.37 GB	Oct 18, 2019	unlocked
		tu1804-ompi		2019	
CANDLE	hpc/candle	201803263	1.52 GB	Oct 18, 2019	unlocked
...					

To view detailed information about a specific image, specify the image and the tag.

Example:

```
C:\>ngc registry image info nvidia/caffe:19.02-py2
-----
Image Information
Name: nvidia/caffe:19.02-py2
Architecture: amd64
Schema Version: 1
-----
```

10.3.2. Pulling a Container Image

With the NGC Registry CLI you can pull (download) images to your system.

To pull an image to your registry space, specify the image and, optionally, the tag.

```
C:\>ngc registry image pull <image-name>[:<tag>]
```

If a tag is not specified, then the tag 'latest' will be used.



Note: nvcr.io/nvidia, nvcr.io/partners and nvcr.io/hpc is reserved namespace and does not grant permissions to users to push or delete container images, models, helm and all artifacts.

Chapter 11. NGC Container Image Versions

Each release of an NGC image is identified by a version "tag". For simpler images this version tag usually contains the version of the major software package in the image. More complex images which contain multiple software packages or versions may use a separate version solely representing the containerized software configuration. One common scheme is versioning by the year and month of the image release. For example, the 20.01 release of an image was released in January, 2020.

An image name consists of two parts separated by a colon. The first part is the name of the container in the repository and the second part is the "tag" associated with the container. These two pieces of information are shown in Figure 2, which is the output from issuing the `docker images` command.

Figure 1. Output from `docker images` command

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nvidia/cuda	8.0-devel	5094464ddfe8	2 weeks ago	1.62 GB
ubuntu	latest	f49eec89601e	2 weeks ago	129 MB
nvcr.io/nvidia/tensorflow	17.01	4352527009ae	2 weeks ago	2.77 GB

Image Name = Repository:Tag ImageID = Unique Hash

Figure 2 shows simple examples of image names, such as:

- ▶ nvidia-cuda:8.0-devel
- ▶ ubuntu:latest
- ▶ nvcr.io/nvidia/tensorflow:17.01

If you choose not to add a tag to an image, by default the word "latest" is added as the tag, however all NGC containers have an explicit version tag.

In the next sections, you will use these image names for running containers. Later in the document, there is also a section on creating your own containers or customizing and extending existing containers.

Chapter 12. NVIDIA Signed Container Images in NGC Catalog

Software signing is the standard mechanism for establishing trust when distributing software, determining which software to trust, and allowing for execution.

Container image signing adds a digital signature to an image. This signature allows users to validate where an image came from and verify the image was not tampered with.

Starting in July 2023, all NVIDIA container images published on the NGC Catalog will be signed.

12.1. Checking for Signed Container Images

There are two ways to determine which containers are signed.

12.1.1. Using the NGC Catalog UI

Click on Catalog > Containers from the left navigation menu. Expand the Signed Images filter to the left and select Yes. The container images tagged with Signed Images will display.

The screenshot shows the NVIDIA NGC Catalog interface. On the left, a sidebar lists categories: Explore Catalog, Collections, **Containers**, Helm Charts, Models, and Resources. The 'Containers' option is highlighted with a red box. The main content area is titled 'Containers' and displays a message: 'The NGC catalog hosts containers for AI/ML, metaverse, and HPC applications and are performance-optimized, tested, and ready to deploy on GPU-powered on-prem, cloud, and edge systems.' Below this is a search bar labeled 'Search containers...' and a dropdown menu set to 'Most Popular'. A large green graphic of stacked 3D cubes is on the right.

Use Case

- NVIDIA Platform
- NVIDIA AI Enterprise Support ⓘ 62
- NVIDIA AI Enterprise Supported
- NVIDIA AI Enterprise Exclusive ⓘ
- Quick Deploy ⓘ
- Industry
- Solution
- Publisher
- Architecture
- Signed Images ⓘ** 129
- Yes
- Multi-Node
- Other

Displaying 269 containers

 DCGM DCGM Exporter Monitor GPUs in Kubernetes using NVIDIA DCGM. This is an exporter for a Prometheus monitoring solution in... <input type="button" value="Signed Images"/>	 NVIDIA INFRASTRUCTURE NVIDIA Kubernetes Device Plugin The NVIDIA Kubernetes Device Plugin registers GPUs as compute resources in the Kubernetes cluster. <input type="checkbox"/> NVIDIA AI Enterprise Supported +1	 NVIDIA INFRASTRUCTURE Validator for NVIDIA GPU Operator Validates NVIDIA GPU Operator components <input type="button" value="View Labels"/> <input type="button" value="Learn More"/>
 NVIDIA INFRASTRUCTURE	 NVIDIA INFRASTRUCTURE	 NVIDIA TRITON INFERENCE SERVER

Select a container image to view the details. On the details page, you will see the Signed Images badge on the left panel and the Signed badge on the pane in the Tags tab.

Catalog > Containers > Validator for NVIDIA GPU Operator

Validator for NVIDIA GPU Operator

[Get Container](#)

Tag	Status	Last Updated	Size	Architectures
v24.3.0-ubi8	Signed	04/30/2024 11:40 AM	169.32 MB	2 Architectures
v24.3.0	Signed	04/30/2024 11:40 AM	169.32 MB	2 Architectures
v23.9.2-ubi8	Signed	03/06/2024 11:12 AM	167.36 MB	2 Architectures
v23.9.2	Signed	03/06/2024 11:12 AM	167.36 MB	2 Architectures
v23.6.2-ubi8	Signed	12/13/2023 5:35 PM	177.77 MB	2 Architectures
v23.6.2	Signed	12/13/2023 5:35 PM	177.77 MB	2 Architectures

12.1.2. Using the NGC CLI

To find signed container images using the CLI, issue the following:

```
$ ngc registry image list <image_name>
```

Example: List all images with name containing 'nginx'

```
$ ngc registry image list '*nginx*'
```

Name	Repository	Latest Tag	Image Size	Updated Date	Permission	Signed Tag?
nginx	nvidia/todds_team/nginx	1.24-bullseye-perl	65.35 MB	May 11, 2023	unlocked	False
nginx	nvidia/nx					
nginx	nvidia/nvforge/nginx	1.23.1-debian-11-r30	35.83 MB	Apr 27, 2023	unlocked	False
nginx-bharti	nvidia/nginx-bharti	testing-sig	2.47 MB	Jan 06, 2023	unlocked	False
nginx	nvidia/ngc-web/nginx	latest	6.66 MB	Sep 30, 2019	unlocked	False
nginx2	nvidia/nginx2	sig-test2	54.35 MB	May 11, 2023	unlocked	True
testnosignnginx	nvidia/shobin-test/test/testnosignnginx	no-signing	54.19 MB	Dec 01, 2022	unlocked	False
busybox_nginx	nvidia/busybox_nginx	3	754.69 KB	Nov 29, 2022	unlocked	False
nginx	nvidia/nginx	sig-test	54.35 MB	May 20, 2023	unlocked	True
nginx-bharti	h2yxhstnusdo/nginx-bharti	test3	54.24 MB	Jan 12, 2023	unlocked	False
nginx-bharti2	h2yxhstnusdo/nginx-bharti2	test	54.24 MB	Jan 12, 2023	unlocked	False
nginx	nvidia/shobin-test/nginx	sig-test	54.35 MB	May 11, 2023	unlocked	True
mynginx	nvidia/mynginx	latest	48.48 MB	Feb 24, 2020	unlocked	False
test_nginx	nvidia/ngc-product-team/test_nginx	latest	42.73 MB	Mar 10, 2020	unlocked	False
ndash-nginx-router	nvidia/ndash-nginx-router	1.0.0	48.62 MB	Nov 19, 2020	unlocked	False

If the image is signed, you can see a value populated in the Latest Tag column.

12.2. Finding NVIDIA's Public Key

The public key associated with the signed image can be used to validate its authenticity. You can find the public key of the container in the following ways.

12.2.1. Using the NGC Catalog UI

On the container details page, select the Tags tab. Click on "View all public keys" in the banner.

Catalog > Containers > Validator for NVIDIA GPU Operator

Validator for NVIDIA GPU Operator

Get Container ▾

Overview Tags Layers Security Scanning Related Collections

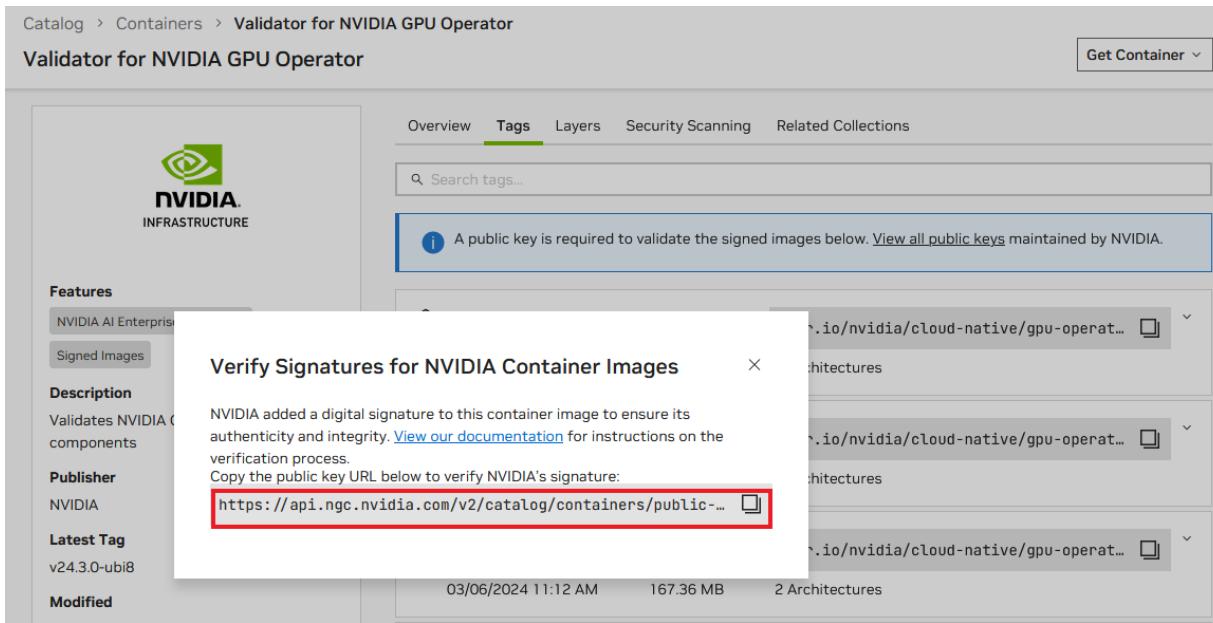
Search tags...

A public key is required to validate the signed images below. [View all public keys](#) maintained by NVIDIA.

v24.3.0-ubi8	nvcr.io/nvidia/cloud-native/gpu-operat...	View Details
04/30/2024 11:40 AM	169.32 MB	2 Architectures

v24.3.0	nvcr.io/nvidia/cloud-native/gpu-operat...	View Details
04/30/2024 11:40 AM	169.32 MB	2 Architectures

The public key URL will display in the dialog.



12.2.2. Using the NGC CLI

To find NVIDIA's public key using the CLI, issue the following command:

```
$ ngc registry image publickey

-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEULvnBd1pnqM0UFjlsIOXIDpFNLAV
KzeNQpYeJITHFIYUhLr5dIVauww5MXr17+9I+Sh/3BiLNyXzISA+ZgnbDw==
-----END PUBLIC KEY-----
```

12.3. Verifying NVIDIA's Signature

The following tools are examples that you can use to verify the signature. There are also other tools available.

12.3.1. Using Cosign

Cosign is a tool for signing container images. To begin, follow the installation instructions provided at the URL below to set up Cosign:

<https://docs.sigstore.dev/cosign/installation/>

Once Cosign is set up, issue the following command to verify the signature of the container image:

```
$ cosign verify [--key <key path>|<key url>|<kms uri>] <image uri>
```

Make sure to replace the placeholders within the square brackets with the appropriate values:

- ▶ <key path>: Path to the key file (if using a local key).
- ▶ <key URL>: URL pointing to the key file (if using a remote key).
- ▶ <KMS URI>: URI for the Key Management Service (if using a KMS for key management).
- ▶ <image URI>: URI of the container image you wish to verify.

Cosign 2.x releases introduced a new flag `--insecure-ignore-tlog=true` for key-based signing with the `cosign verify` command. Please include this flag when using Cosign 2.x versions. Prior to version 2.x, this flag is not required as it is set as the default behavior.

```
cosign verify --insecure-ignore-tlog [--key <key path>|<key url>|<kms uri>] <image uri>
```

12.3.2. Using an Admission Controller in Kubernetes Clusters

Several Open Source Software (OSS) admission controllers offer the capability of sigstore-based image verification. These powerful tools allow you to verify the integrity and authenticity of containers deployed within your Kubernetes environment.

For example, Connaisseur is a Kubernetes admission controller that can be used to integrate container image signature verification and trust pinning into a cluster. It ensures that only signed images are allowed to run in a cluster.

To set up Connaisseur, follow the steps outlined in the documentation below:

- ▶ [Getting started Guide](#)
- ▶ [Cosign Validator setup](#)
- ▶ [Test Connaisseur](#)

After configuring the Connaisseur validator, you can test it by running a container in the Kubernetes (K8s) cluster using the following command for a signed image:

```
kubectl run test --image=${IMAGE}
```

When you run the container with the command, the request is admitted to the cluster if the image is signed and passes the verification. Kubernetes will return the following response:

```
pod/test created
```

Chapter 13. Running an NGC Container

Before you can run an NGC deep learning framework container, your Docker environment must support NVIDIA GPUs. To run a container, issue the appropriate command as explained in this chapter, specifying the registry, repository, and tags.

13.1. Enabling GPU Support for NGC Containers

To obtain the best performance when running NGC containers, three methods of providing GPU support for Docker containers have been developed:

- ▶ Native GPU support (included with Docker-ce 19.03 or later)
- ▶ NVIDIA Container Runtime for Docker (`nvidia-docker2` package)
- ▶ Docker Engine Utility for NVIDIA GPUs (`nvidia-docker` package)

The method implemented in your system depends on the DGX OS version installed (for DGX systems), the specific NGC Cloud Image provided by a Cloud Service Provider, or the software that you have installed in preparation for running NGC containers on TITAN PCs, Quadro PCs, or vGPUs.

Refer to the following table to assist in determining which method is implemented in your system.

GPU Support Method	When Used	How to Determine
Native GPU Support	Included with Docker-ce 19.03 or later	Run <code>docker version</code> to determine the installed Docker version.
NVIDIA Container Runtime for Docker	If the <code>nvidia-docker2</code> package is installed	Run <code>nvidia-docker version</code> and check for NVIDIA Docker version 2.0 or later
Docker Engine Utility for NVIDIA GPUs	If the <code>nvidia-docker</code> package is installed	Run <code>nvidia-docker version</code> and check for NVIDIA Docker version 1.x

Each method is invoked by using specific Docker commands, described as follows.

Using Native GPU support



Note: If Docker is updated to 19.03 on a system which already has nvidia-docker or nvidia-docker2 installed, then the corresponding methods can still be used.

- ▶ To use the native support on a new installation of Docker, first enable the new GPU support in Docker.

```
$ sudo apt-get install -y docker nvidia-container-toolkit
```

This step is not needed if you have updated Docker to 19.03 on a system with nvidia-docker2 installed. The native support will be enabled automatically.

- ▶ Use docker run --gpus to run GPU-enabled containers.

- ▶ Example using all GPUs

```
$ sudo docker run --gpus all ...
```

- ▶ Example using two GPUs

```
$ sudo docker run --gpus 2 ...
```

- ▶ Examples using specific GPUs

```
$ sudo docker run --gpus "device=1,2" ...
```

```
$ sudo docker run --gpus "device=UUID-ABCDEF,1" ...
```

Using the NVIDIA Container Runtime for Docker

With the NVIDIA Container Runtime for Docker installed (nvidia-docker2), you can run GPU-accelerated containers in one of the following ways.

- ▶ Use docker run and specify runtime=nvidia.

```
$ sudo docker run --runtime=nvidia ...
```

- ▶ Use nvidia-docker run.

```
$ sudo nvidia-docker run ...
```

The new package provides backward compatibility, so you can still run GPU-accelerated containers by using this command, and the new runtime will be used.

Use docker run with nvidia as the default runtime.

You can set nvidia as the default runtime, for example, by adding the following line to the /etc/docker/daemon.json configuration file as the first entry.

```
"default-runtime": "nvidia",
```

The following is an example of how the added line appears in the JSON file. Do not remove any pre-existing content when making this change.

```
{
  "default-runtime": "nvidia",
  "runtimes": {
    "nvidia": {
      "path": "/usr/bin/nvidia-container-runtime",
      "runtimeArgs": []
    }
  }
}
```

You can then use docker run to run GPU-accelerated containers.

```
$ sudo docker run ...
```

CAUTION: If you build Docker images while `nvidia` is set as the default runtime, make sure the build scripts executed by the Dockerfile specify the GPU architectures that the container will need. Failure to do so may result in the container being optimized only for the GPU architecture on which it was built. Instructions for specifying the GPU architecture depend on the application and are beyond the scope of this document. Consult the specific application build process for guidance.

Using the Docker Engine Utility for NVIDIA GPUs

With the Docker Engine Utility for NVIDIA GPUs installed (`nvidia-docker`), run GPU-enabled containers as follows.

```
$ sudo nvidia-docker run ...
```

13.2. Running NGC Containers with Runtime Resources

On a system with GPU support for NGC containers, the following occurs when running a container.

- ▶ The Docker Engine loads the image into a container which runs the software.
- ▶ You define the runtime resources of the container by including additional flags and settings that are used with the command. These flags and settings are described in the following sections.
- ▶ The GPUs are explicitly defined for the Docker container (defaults to all GPUs, can be specified using `NV_GPU` environment variable).



Note: The base command `docker run --gpu all` assumes that your system has Docker 19.03-CE installed. See the section [Enabling GPU Support for NGC Containers](#) for the command to use for earlier versions of Docker.

1. As a user, run the container interactively.

```
$ docker run --gpus all -it --rm -v local_dir:container_dir
    nvcr.io/nvidia/<repository>:<xx.xx>
```

The following example runs the December 2016 release (16.12) of the NVCAFFE container in interactive mode. The container is automatically removed when the user exits the container.

```
$ docker run --gpus all --rm -ti nvcr.io/nvidia/caffe:16.12
=====
== Caffe ==
=====

NVIDIA Release 16.12 (build 6217)

Container image Copyright (c) 2016, NVIDIA CORPORATION. All rights reserved.
Copyright (c) 2014, 2015, The Regents of the University of California (Regents)
```

```
All rights reserved.
```

```
Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the
underlying project or file.
root@df57eb8e0100:/workspace#
```

- From within the container, start the job that you want to run.

The precise command to run depends on the deep learning framework in the container that you are running and the job that you want to run. For details see the /workspace/README.md file for the container.

The following example runs the `caffe time` command on one GPU to measure the execution time of the `deploy.prototxt` model.

```
# caffe time -model models/bvlc_alexnet/ -solver deploy.prototxt -gpu=0
```

- Optional: Run the December 2016 release (16.12) of the same NVCAFFE container but in non-interactive mode.

```
% docker run --gpus all --rm nvcr.io/nvidia/caffe:16.12 caffe time -model
/workspace/models/bvlc_alexnet -solver /workspace/deploy.prototxt -gpu=0
```

13.2.1. Specifying A User

Unless otherwise specified, the user inside the container is the root user.

When running within the container, files created on the host operating system or network volumes can be accessed by the root user. This is unacceptable for some users and they will want to set the ID of the user in the container. For example, to set the user in the container to be the currently running user, issue the following:

```
% docker run --gpus all -ti --rm -u $(id -u):$(id -g) nvcr.io/nvidia/<repository>:<tag>
```

Typically, this results in warnings due to the fact that the specified user and group do not exist in the container. You might see a message similar to the following:

```
groups: cannot find name for group ID 1000I have no name! @c177b61e5a93:/workspace$
```

The warning can usually be ignored.

13.2.2. Setting The Remove Flag

By default, Docker containers remain on the system after being run. Repeated pull or run operations use up more and more space on the local disk, even after exiting the container. Therefore, it is important to clean up the nvidia-docker containers after exiting.



Note: Do not use the `--rm` flag if you have made changes to the container that you want to save, or if you want to access job logs after the run finishes.

To automatically remove a container when exiting, add the `--rm` flag to the run command.

```
% docker run --gpus all --rm nvcr.io/nvidia/<repository>:<tag>
```

13.2.3. Setting The Interactive Flag

By default, containers run in batch mode; that is, the container is run once and then exited without any user interaction. Containers can also be run in interactive mode as a service.

To run in interactive mode, add the `-ti` flag to the run command.

```
% docker run --gpus all -ti --rm nvcr.io/nvidia/<repository>:<tag>
```

13.2.4. Setting The Volumes Flag

There are no data sets included with the containers, therefore, if you want to use data sets, you need to mount volumes into the container from the host operating system. For more information, see [Manage data in containers](#).

Typically, you would use either Docker volumes or host data volumes. The primary difference between host data volumes and Docker volumes is that Docker volumes are private to Docker and can only be shared amongst Docker containers. Docker volumes are not visible from the host operating system, and Docker manages the data storage. Host data volumes are any directory that is available from the host operating system. This can be your local disk or network volumes.

Example 1

Mount a directory `/raid/imagedata` on the host operating system as `/images` in the container.

```
% docker run --gpus all -ti --rm -v /raid/imagedata:/images
    nvcr.io/nvidia/<repository>:<tag>
```

Example 2

Mount a local docker volume named `data` (must be created if not already present) in the container as `/imagedata`.

```
% docker run --gpus all -ti --rm -v data:/imagedata nvcr.io/nvidia/<repository>:<tag>
```

13.2.5. Setting The Mapping Ports Flag

Applications such as Deep Learning GPU Training System™ (DIGITS) open a port for communications. You can control whether that port is open only on the local system or is available to other computers on the network outside of the local system.

Using DIGITS as an example, in DIGITS 5.0 starting in container image 16.12, by default the DIGITS server is open on port 5000. However, after the container is started, you may not easily know the IP address of that container. To know the IP address of the container, you can choose one of the following ways:

- ▶ Expose the port using the local system network stack (`--net=host`) where port 5000 of the container is made available as port 5000 of the local system.

or

- ▶ Map the port (`-p 8080:5000`) where port 5000 of the container is made available as port 8080 of the local system.

In either case, users outside the local system have no visibility that DIGITS is running in a container. Without publishing the port, the port is still available from the host, however not from the outside.

13.2.6. Setting The Shared Memory Flag

Certain applications, such as PyTorch™ and the Microsoft® Cognitive Toolkit™, use shared memory buffers to communicate between processes. Shared memory can also be required by single process applications, such as MXNet™ and TensorFlow™, which use the NVIDIA® Collective Communications Library™ (NCCL) (NCCL).

By default, Docker containers are allotted 64MB of shared memory. This can be insufficient, particularly when using all 8 GPUs. To increase the shared memory limit to a specified size, for example 1GB, include the `--shm-size=1g` flag in your `docker run` command.

Alternatively, you can specify the `--ipc=host` flag to re-use the host's shared memory space inside the container. Though this latter approach has security implications as any data in shared memory buffers could be visible to other containers.

13.2.7. Setting The Restricting Exposure Of GPUs Flag

From inside the container, the scripts and software are written to take advantage of all available GPUs. To coordinate the usage of GPUs at a higher level, you can use this flag to restrict the exposure of GPUs from the host to the container. For example, if you only want GPU 0 and GPU 1 to be seen in the container, you would issue the following:

Using native GPU support

```
$ docker run --gpus "device=0,1" ...
```

Using nvidia-docker2

```
$ NV_GPU=0,1 docker run --runtime=nvidia ...
```

Using nvidia-docker

```
$ NV_GPU=0,1 nvidia-docker run ...
```

This flag creates a temporary environment variable that restricts which GPUs are used.

Specified GPUs are defined per container using the Docker device-mapping feature, which is currently based on Linux cgroups.

13.2.8. Container Lifetime

The state of an exited container is preserved indefinitely if you do not pass the `--rm` flag to the `docker run` command. You can list all of the saved exited containers and their size on the disk with the following command:

```
$ docker ps --all --size --filter Status=exited
```

The container size on the disk depends on the files created during the container execution, therefore the exited containers take only a small amount of disk space.

You can permanently remove a exited container by issuing:

```
docker rm [CONTAINER ID]
```

By saving the state of containers after they have exited, you can still interact with them using the standard Docker commands. For example:

- ▶ You can examine logs from a past execution by issuing the `docker logs` command.

```
$ docker logs 9489d47a054e
```

- ▶ You can extract files using the `docker cp` command.

```
$ docker cp 9489d47a054e:/log.txt .
```

- ▶ You can restart a stopped container using the `docker restart` command.

```
$ docker restart <container name>
```

For the NVCaffe™ container, issue this command:

```
$ docker restart caffe
```

- ▶ You can save your changes by creating a new image using the `docker commit` command. For more information, see [Example 3: Customizing a Container using](#).



Note: Use care when committing Docker container changes, as data files created during use of the container will be added to the resulting image. In particular, core dump files and logs can dramatically increase the size of the resulting image.

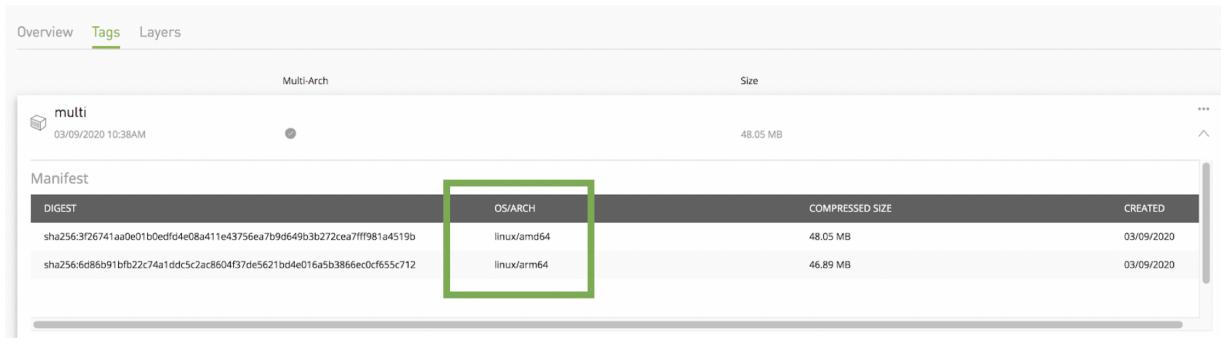
Chapter 14. Multi-Architecture Support for NGC Container Images

The NGC Container Registry allows users to leverage [docker multi-architecture](#). It can support multiple architectures, which means that a single image may contain variants for different CPU architectures like ARM, x86, IBM POWER and others ; and sometimes for different operating systems, such as Windows. When running an image, docker will automatically select an image variant which matches the target deployment OS and architecture.

Manifest Lists and Tags

The NGC Container Registry supports the manifest list schema application/vnd.docker.distribution.manifest.list.v2+json providing the ability to assign multiple tags per image. To inspect the manifest list, please follow the instructions [here](#).

NGC UI allows users to easily identify multi-architecture containers and see the supported CPU architectures.



The screenshot shows the NGC UI interface for a container image named 'multi'. At the top, there are tabs for 'Overview', 'Tags' (which is selected), and 'Layers'. Below the tabs, there are two sections: 'Multi-Arch' and 'Size'. The 'Multi-Arch' section shows a single variant entry for 'linux/amd64'. The 'Size' section shows the total size of 48.05 MB. Underneath these sections is a table titled 'Manifest' with columns for 'DIGEST', 'OS/ARCH', 'COMPRESSED SIZE', and 'CREATED'. The table contains two rows: one for 'linux/amd64' (digest sha256:3f26741aa0e01b0edfd4e08a411e43756ea7b9d649b3b272cea7ff981a4519b) and one for 'linux/arm64' (digest sha256:6d86b91bfb22c74a1ddc5c2ac8604f37de5621bd4e016a5b3866ec0cf655c712). A green box highlights the 'OS/ARCH' column in the manifest table.

Basic Docker Commands for Multi-architecture Images

Inspect Manifest : Display an image manifest or manifest list

```
$ docker manifest inspect <container_name>:<tag>
```

More help on manifest: [link](#)

Pull a specific image for an architecture

In order to pull an image with a specific architecture, first do docker manifest which lists multiple platforms and then pull with a specific platform name that matches with the manifest digest.

```
$ docker pull --platform=<arch>
```

More help on the above docker command: [link](#)

Chapter 15. Customizing Containers

NGC images come prepackaged, tuned, and ready to run; however, you may want to build a new image from scratch or augment an existing image with custom code, libraries, data, or settings for your corporate infrastructure. This section will guide you through exercises that will highlight how to create a container from scratch, customize a container, extend a deep learning framework to add features, develop some code using that extended framework from the developer environment, then package that code as a versioned release.

By default, you do not need to build a container. The NGC container registry, `nvcr.io`, has a number of containers that can be used immediately. These include containers for deep learning, scientific computing and visualization, as well as containers with just the CUDA Toolkit.

One of the great things about containers is that they can be used as starting points for creating new containers. This can be referred to as "customizing" or "extending" a container. You can create a container completely from scratch, however, since these containers are likely to run on a GPU system, it is recommended that you at least start with a `nvcr.io` container that contains the OS and CUDA. However, you are not limited to this and can create a container that runs on the CPUs in the system which does not use the GPUs. In this case, you can start with a bare OS container from Docker. However, to make development easier, you can still start with a container with CUDA - it is just not used when the container is used.

In the case of the DGX-1 and the DGX Station, you can push or save your modified/extended containers to the NVIDIA DGX container registry, `nvcr.io`. They can also be shared with other users of the DGX system but this requires some administrator help.

Currently, you cannot save customized containers from the NGC container registry (cloud based) solution to `nvcr.io`. The customized or extended containers can be saved to a user's private container repository.

It is important to note that all NGC deep learning framework images include the source to build the framework itself as well as all of the prerequisites.

ATTENTION: Do not install an NVIDIA driver into the Docker® image at Docker build time. `nvidia-docker` is essentially a wrapper around Docker that transparently provisions a container with the necessary components to execute code on the GPU.

NVIDIA provides a large set of images in the NGC container registry that are already tested, tuned, and are ready to run. You can pull any one of these images to create a container and add software or data of your choosing.

A best-practice is to avoid `docker commit` usage for developing new docker images, and to use Dockerfiles instead. The Dockerfile method provides visibility and capability to efficiently version-control changes made during development of a docker image. The `docker commit` method is appropriate for short-lived, disposable images only (see [Example 3: Customizing A Container Using docker commit](#) for an example).

For more information on writing a Docker file, see the [best practices documentation](#).

15.1. Benefits And Limitations To Customizing A Container

You can customize a container to fit your specific needs for numerous reasons; for example, you depend upon specific software that is not included in the container that NVIDIA provides. No matter your reasons, you can customize a container.

The container images do not contain sample data-sets or sample model definitions unless they are included with the framework source. Be sure to check the container for sample data-sets or models.

15.2. Example 1: Building A Container From Scratch

Docker uses Dockerfiles to create or build a Docker image. Dockerfiles are scripts that contain commands that Docker uses successively to create a new Docker image. Simply put, a Dockerfile is the source code for the container image. Dockerfiles always start with a base image to inherit from.

For more information, see [Best practices for writing Dockerfiles](#).

1. Create a working directory on your local hard-drive.
2. In that directory, open a text editor and create a file called `Dockerfile`. Save the file to your working directory.
3. Open your `Dockerfile` and include the following:

```
FROM ubuntu:14.04
RUN apt-get update && apt-get install -y curl
CMD echo "hello from inside a container"
```

Where the last line `CMD`, executes the indicated command when creating the container. This is a way to check that the container was built correctly.

For this example, we are also pulling the container from the Docker repository and not the DGX™ system repository. There will be subsequent examples using the NVIDIA® repository.

4. Save and close your Dockerfile.
5. Build the image. Issue the following command to build the image and create a tag.

```
$ docker build -t <new_image_name>:<new_tag> .
```



Note: This command was issued in the same directory where the Dockerfile is located.

The output from the docker build process lists "Steps"; one for each line in the Dockerfile.

For example, let us name the container `test1` and tag it with `latest`. Also, for illustrative purposes, let us assume our private DGX system repository is called `nvidian_sas`. The command below builds the container. Some of the output is shown below so you know what to expect.

```
$ docker build -t test1:latest .
Sending build context to Docker daemon 3.072 kB
Step 1/3 : FROM ubuntu:14.04
14.04: Pulling from library/ubuntu
...
Step 2/3 : RUN apt-get update && apt-get install -y curl
...
Step 3/3 : CMD echo "hello from inside a container"
--> Running in 1f491b9235d8
--> 934785072daf
Removing intermediate container 1f491b9235d8
Successfully built 934785072daf
```

For information about building your image, see [docker build](#). For information about tagging your image, see [docker tag](#).

6. Verify that the build was successful. You should see a message similar to the following:

```
Successfully built 934785072daf
```

This message indicates that the build was successful. Any other message and the build was not successful.



Note: The number, `934785072daf`, is assigned when the image is built and is random.

7. Confirm you can view your image by issuing the following command and view your container.

```
$ docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
test1           latest       934785072daf   19 minutes ago  222 MB
```

The new container is now available to be used.



Note: The container is local to this DGX system. If you want to store the container in your private repository, follow the next step.

8. Store the container in your private Docker repository by pushing it.



Note: This applies if you have a private registry account associated with the DGX system purchased by your organization.

- a). The first step in pushing it, is to tag it.

```
$ docker tag test1 nvcr.io/nvidian_sas/test1:latest
```

- b). Now that the image has been tagged, you can push it to, for example, a private project on `nvcr.io` named `nvidian_sas`.

```
$ docker push nvcr.io/nvidian_sas/test1:latest
The push refers to a repository [nvcr.io/nvidian_sas/test1]
...
```

- c). Verify that the container appears in the `nvidian_sas` repository.

15.3. Example 2: Customizing A Container Using Dockerfile

This example uses a Dockerfile to customize the NVCAFFE container in `nvcr.io`. Before customizing the container, you should ensure the NVCAFFE 17.03 container has been loaded into the registry using the `docker pull` command before proceeding.

```
$ docker pull nvcr.io/nvidia/caffe:17.03
```

As mentioned earlier in this document, the Docker containers on `nvcr.io` also provide a sample Dockerfile that explains how to patch a framework and rebuild the Docker image. In the directory `/workspace/docker-examples`, there are two sample Dockerfiles. For this example, we will use the `Dockerfile.customcaffe` file as a template for customizing a container.

1. Create a working directory called `my_docker_images` on your local hard drive.
2. Open a text editor and create a file called `Dockerfile`. Save the file to your working directory.
3. Open your `Dockerfile` again and include the following lines in the file:

```
FROM nvcr.io/nvidia/caffe:17.03
# APPLY CUSTOMER PATCHES TO CAFFE
# Bring in changes from outside container to /tmp
# (assumes my-caffe-modifications.patch is in same directory as
Dockerfile)
#COPY my-caffe-modifications.patch /tmp

# Change working directory to NVCAFFE source path
WORKDIR /opt/caffe

# Apply modifications
#RUN patch -p1 < /tmp/my-caffe-modifications.patch

# Note that the default workspace for caffe is /workspace
RUN mkdir build && cd build && \
    cmake -DCMAKE_INSTALL_PREFIX:PATH=/usr/local -DUSE_NCCL=ON \
-DUSE_CUDNN=ON -DCUDA_ARCH_NAME=Manual -DCUDA_ARCH_BIN="35 52 60 61" \
-DCUDA_ARCH_PTX="61" .. && \
    make -j$(nproc) install && \
    make clean && \
    cd .. && rm -rf build

# Reset default working directory
WORKDIR /workspace
```

Save the file.

- Build the image using the `docker build` command and specify the repository name and tag. In the following example, the repository name is `corp/caffe` and the tag is `17.03.1PlusChanges`. For the case, the command would be the following:

```
$ docker build -t corp/caffe:17.03.1PlusChanges .
```

- Run the Docker image using the command appropriate to the method of GPU support installed.

```
$ docker run --gpus all -ti corp/caffe:17.03.1PlusChanges .
```



Note: The base command `docker run --gpu all` assumes that your system has Docker 19.03-CE installed. See the section [Enabling GPU Support for NGC Containers](#) for the command to use for earlier versions of Docker.

15.4. Example 3: Customizing A Container Using `docker commit`

This example uses the `docker commit` command to flush the current state of the container to a Docker image. This is not a recommended best practice, however, this is useful when you have a container running to which you have made changes and want to save them. In this example, we are using the `apt-get` tag to install packages which requires that the user run as root.



Note:

- The NVCaffe image release 17.04 is used in the example instructions for illustrative purposes.
- Do not use the `--rm` flag when running the container. If you use the `--rm` flag when running the container, your changes will be lost when exiting the container.

- Pull the Docker container from the `nvcr.io` repository to the DGX system. For example, the following command will pull the NVCaffe container:

```
$ docker pull nvcr.io/nvidia/caffe:17.04
```

- Run the container on the DGX system.

```
$ docker run --gpus all -ti nvcr.io/nvidia/caffe:17.04
```

```
=====
== NVIDIA Caffe ==
=====
```

```
NVIDIA Release 17.04 (build 26740)
```

```
Container image Copyright (c) 2017, NVIDIA CORPORATION. All rights reserved.
Copyright (c) 2014, 2015, The Regents of the University of California (Regents)
All rights reserved.
```

```
Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying
project or file.
```

```
NOTE: The SHMEM allocation limit is set to the default of 64MB. This may be insufficient
for NVIDIA Caffe. NVIDIA recommends the use of the following flags:
  nvidia-docker run --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 ...

root@1fe228556a97:/workspace#
```



Note: The base command `docker run --gpu` all assumes that your system has Docker 19.03-CE installed. See the section [Enabling GPU Support for NGC Containers](#) for the command to use for earlier versions of Docker.

3. You should now be the root user in the container (notice the prompt). You can use the command `apt` to pull down a package and put it in the container.



Note: The NVIDIA containers are built using Ubuntu which uses the `apt-get` package manager. Check the container release notes [Deep Learning Documentation](#) for details on the specific container you are using.

In this example, we will install Octave; the GNU clone of MATLAB, into the container.

```
# apt-get update
# apt install octave
```



Note: You have to first issue `apt-get update` before you install Octave using `apt`.

4. Exit the workspace.

```
# exit
```

5. Display the list of containers using `docker ps -a`. As an example, here is some of the output from the `docker ps -a` command:

```
$ docker ps -a
CONTAINER ID        IMAGE               CREATED             ...
1fe228556a97      nvcr.io/nvidia/caffe:17.04   3 minutes ago ...
```

6. Now you can create a new image from the container that is running where you have installed Octave. You can commit the container with the following command.

```
$ docker commit 1fe228556a97 nvcr.io/nvidian_sas/caffe_octave:17.04
sha256:0248470f46e22af7e6cd90b65fdee6b4c6362d08779a0bc84f45de53a6ce9294
```

7. Display the list of images.

```
$ docker images
REPOSITORY          TAG           IMAGE ID            ...
nvidian_sas/caffe_octave    17.04       75211f8ec225 ...
```

8. To verify, let's run the container again and see if Octave is actually there.



Note: This only works for the DGX-1 and the DGX Station.

```
$ docker run --gpus all -ti nvidian_sas/caffe_octave:17.04
=====
== NVIDIA Caffe ==
=====

NVIDIA Release 17.04 (build 26740)

Container image Copyright (c) 2017, NVIDIA CORPORATION. All rights reserved. Copyright
(c) 2014, 2015, The Regents of the University of California (Regents) All rights
reserved.
```

```

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved. NVIDIA
modifications are covered by the license terms that apply to the underlying project or
file.

NOTE: The SHMEM allocation limit is set to the default of 64MB. This may be insufficient
for NVIDIA Caffe. NVIDIA recommends the use of the following flags:
    nvidia-docker run --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 ...

root@2fc3608ad9d8:/workspace# octave
octave: X11 DISPLAY environment variable not set
octave: disabling GUI features
GNU Octave, version 4.0.0
Copyright (C) 2015 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
```

```
octave:1>
```

Since the Octave prompt displayed, Octave is installed.

9. If you want to save the container into your private repository (Docker uses the phrase "push"), then you can use the command `docker push`

```
$ docker push nvcr.io/nvidian_sas/caffe_octave:17.04
```

The new Docker image is now available for use. You can check your local Docker repository for it.

15.5. Example 4: Developing A Container Using Docker

There are two primary use cases for a developer to extend a container:

1. Create a development image that contains all of the immutable dependencies for the project, but not the source code itself.
2. Create a production or testing image that contains a fixed version of the source and all of the software dependencies.

The datasets are not packaged in the container image. Ideally, the container image is designed to expect volume mounts for datasets and results.

In these examples, we mount our local dataset from `/raid/datasets` on our host to `/dataset` as a read-only volume inside the container. We also mount a job specific directory to capture the output from a current run.

In these examples, we will create a timestamped output directory on each container launch and map that into the container at `/output`. Using this method, the output for each successive container launch is captured and isolated.

Including the source into a container for developing and iterating on a model has many awkward challenges that can over complicate the entire workflow. For instance, if your source code is in the container, then your editor, version control software, dotfiles, etc. also need to be in the container.

However, if you create a development image that contains everything you need to run your source code, you can map your source code into the container to make use of your host workstation's developer environment. For sharing a fixed version of a model, it is best to package a versioned copy of the source code and trained weights with the development environment.

As an example, we will work though a development and delivery example for the open source implementation of the work found in [Image-to-Image Translation with Conditional Adversarial Networks](#) by Isola et. al. and is available at [pix2pix](#). Pix2Pix is a Torch implementation for learning a mapping from input images to output images using a Conditional Adversarial Network. Since online projects can change over time, we will focus our attention on the snapshot version `d7e7b8b557229e75140cbe42b7f5dbf85a67d097` change-set.

In this section, we are using the container as a virtual environment, in that the container has all the programs and libraries needed for our project.



Note: We have kept the network definition and training script separate from the container image. This is a useful model for iterative development because the files that are actively being worked on are persistent on the host and only mapped into the container at runtime.

The differences to the original project can be found here [Comparing changes](#).

If the machine you are developing on is not the same machine on which you will be running long training sessions, then you may want to package your current development state in the container.

1. Create a working directory on your local hard-drive.

```
mkdir Projects
$ cd ~/Projects
```

2. Git clone the Pix2Pix git repository.

```
$ git clone https://github.com/phillipi/pix2pix.git
$ cd pix2pix
```

3. Run the `git checkout` command.

```
$ git checkout -b devel d7e7b8b557229e75140cbe42b7f5dbf85a67d097
```

4. Download the dataset:

```
bash ./datasets/download_dataset.sh facades
```

I want to put the dataset on my fast /raid storage.

```
$ mkdir -p /raid/datasets
$ mv ./datasets/facades /raid/datasets
```

5. Create a file called Dockerfile, and add the following lines:

```
FROM nvcr.io/nvidia/torch:17.03
RUN luarocks install nngraph
RUN luarocks install
https://raw.githubusercontent.com/szym/display/master/display-scm-0.rockspec
WORKDIR /source
```

6. Build the development Docker container image (build-devel.sh).

```
docker build -t nv/pix2pix-torch:devel .
```

7. Create the following train.sh script:

```
#!/bin/bash -x
ROOT="${ROOT:-/source}"
DATASET="${DATASET:-facades}"
DATA_ROOT="${DATA_ROOT:-/datasets/$DATASET}"
DATA_ROOT=$DATA_ROOT name="${DATASET}_generation"
which_direction=BtoA th train.lua
```

If you were actually developing this model, you would be iterating by making changes to the files on the host and running the training script which executes inside the container.

8. Optional: Edit the files and execute the next step after each change.

9. Run the training script (run-devel.sh).

```
docker run --gpus all --rm -ti -v $PWD:/source -v
/raid/datasets:/datasets nv/pix2pix-torch:devel ./train.sh
```



Note: The base command docker run --gpu all assumes that your system has Docker 19.03-CE installed. See the section [Enabling GPU Support for NGC Containers](#) for the command to use for earlier versions of Docker.

15.5.1. Example 4.1: Package The Source Into The Container

Packaging the model definition and script into the container is very simple. We simply add a COPY step to the Dockerfile.

We've updated the run script to simply drop the volume mounting and use the source packaged in the container. The packaged container is now much more portable than our devel container image because the internal code is fixed. It would be good practice to version control this container image with a specific tag and store it in a container registry.

The updates to run the container are equally subtle. We simply drop the volume mounting of our local source into the container.

Chapter 16. Models

A discussion of NGC models and how to download them from the NGC Catalog.

16.1. What are Models

Many AI applications have common needs: classification, object detection, language translation, text-to-speech, recommender engines, sentiment analysis, and more. When developing applications with these capabilities, it is much faster to start with a model that is pre-trained and then tune it for a specific use case.

NGC offers pre-trained models for a large number of AI tasks that are optimized for NVIDIA Tensor Core GPUs, and can be easily re-trained by updating just a few layers, saving valuable time.

So whether you're looking for content you can simply retrain for your specific use case or a complete model you can grab and deploy right away, NGC has you covered.

16.2. Searching and Filtering Models

To view NGC models, navigate to Catalog > Models in the left menu. Use the search bar at the top of the page to filter the types of models or sort the list using the criteria from the dropdown menu. You can refine your search by utilizing the filters under Use Case, NVIDIA Platform, NVIDIA AI Enterprise, and many others. The descriptions for the filtering options are as follows:

- ▶ Use Case: Choose from the available common use cases.
- ▶ NVIDIA AI Enterprise Support: Select this option to find publicly available models supported by NVIDIA AI Enterprise.
- ▶ NVIDIA Platform: Choose from Clara, DeepStream, TensorFlow, and many others.
- ▶ NVIDIA AI Enterprise Exclusive: Choose products under this to see entitled entities.
- ▶ Framework: Filter by framework such as TAO Toolkit, PyTorch, and Riva.
- ▶ Industry: Select by available industries.
- ▶ Solution: Select the solution that closely matches your use case.
- ▶ Publisher: Filter by publisher.

- ▶ Language: Select by language.
- ▶ Other: Miscellaneous options.

The following example page shows the available models when searching on 'deep learning':

The screenshot shows the NGC Catalog interface. On the left is a sidebar with navigation links: Explore Catalog, Collections, Containers, Helm Charts, Models (which is selected and highlighted in red), and Resources. The main content area has a title 'Catalog > Models' and a large heading 'Models'. A sub-headline reads: 'The NGC catalog offers 100s of pre-trained models for computer vision, speech, recommendation, and more. Bring AI faster to market by using these models as-is or quickly build proprietary models with a fraction of your custom data.' To the right of this is a green neural network icon. Below the headline is a search bar with the query 'deep learning' and a dropdown menu set to 'Relevance'. A red box highlights the search bar and the relevance dropdown. To the left of the search bar is a filter sidebar titled 'Use Case' with the following options: NVIDIA AI Enterprise Support (21 items), NVIDIA Platform, NVIDIA AI Enterprise Exclusive, Framework, Industry, Solution, Publisher, Language, and Other. The main content area displays four model cards, each featuring the NVIDIA logo and the text 'DEEP LEARNING EXAMPLES'. The first card is for 'Wide & Deep TF Checkpoint (Base, ...)' and the second for 'Wide&Deep TF2 checkpoint (Base, ...)'. Both cards have 'View Labels' and 'Learn More' buttons. The third and fourth cards are partially visible.

The filters will contain the results for the matching search criteria or "No results found" if there aren't any matches.

16.3. Downloading Models via NGC CLI

A discussion of NGC models and how to download them from the NGC Catalog.

To download a model from the registry to your local disk, specify the model name and version.

```
$ nvc registry model download-version <org-name>/<model-name:version>
```

Example

```
$ nvc registry model download-version nvidia/ngcdocsmodel:1
```

The following is an example showing the output confirming completion of the download:

```
Downloaded 230.92 MB in 38s, Download speed: 6.07 MB/s
```

```
-----
```

```
Transfer id: trt_onnx_vgg16_v100_16g_int8_v1 Download status: Completed.
```

```
Downloaded local path: C:\ngcdocsmodel
Total files downloaded: 3
Total downloaded size: 230.92 MB
Started at: 2019-03-18 14:09:31.664000
Completed at: 2019-03-18 14:10:09.712000
Duration taken: 38s seconds
-----
```

The model is downloaded to a folder that corresponds to the model name in the current directory. You can specify another path using the `-d .` option.

Example: Downloading a model to a specific directory (/models).

```
$ ngc registry model download-version nvidia/ngcdocsmodel:1 -d ./models
Downloaded 230.92 MB in 38s, Download speed: 6.07 MB/s
-----
Transfer id: trt_onnx_vgg16_v100_16g_int8_v1 Download status: Completed.
Downloaded local path: C:\ngcdocsmodel
Total files downloaded: 3
Total downloaded size: 230.92 MB
Started at: 2019-03-18 14:09:31.664000
Completed at: 2019-03-18 14:10:09.712000
Duration taken: 38s seconds
-----
```

16.4. Downloading Models via WGET/cURL

16.4.1. Downloading Guest Access Models via WGET/cURL

How to download guest access models via WGET/cURL.

To get a copy of a single file from an NGC Model via wget/curl you can use the URL provided in the NGC UI.

1. Open the NGC UI, click Models from the left navigation menu and then locate and click the model you want to download.

2. Click the Download button on the top right or navigate to the File Browser tab.

3. Select the Copy curl cmd or Copy wget cmd to copy the relevant code to the clipboard.

Catalog > Models > TrafficCamNet

TrafficCamNet

Download ▾

The screenshot shows the NGC Catalog interface for the 'TrafficCamNet' model. On the left, there's a sidebar with the NVIDIA TAO logo and sections for 'Features' (NVIDIA AI Enterprise Supported), 'Description' (4 class object detection network to detect cars in an image), 'Publisher' (NVIDIA), and 'Latest Version' (pruned_onnx_v1.0.3). The main area has tabs for 'Overview', 'Version History', 'File Browser' (which is selected), and 'Related Collections'. In the 'File Browser' tab, a dropdown shows 'pruned_onnx_v1.0.3'. A search bar says 'Search files...'. Below is a table with columns 'File', 'Size', 'Modified', and 'Actions'. The table lists four files: 'labels.txt', 'nvinfer_config.txt', 'resnet18_trafficcamnet_pruned.onnx', and 'resnet18_trafficcamnet_pruned_int8.txt'. The 'Actions' column for the last file contains three options: 'Copy curl cmd', 'Copy wget cmd', and 'Download File', which is highlighted with a red box.

The following will then be added to your clipboard:

```
$ wget https://api.ngc.nvidia.com/v2/models/nvidia/<model-name>/versions/<version>/files/<filename>
```

16.4.2. Downloading Authenticated Access Models via WGET/cURL

How to download authenticated access models via WGET/cURL.

1. Obtain an NGC API Key. See [NGC API Keys](#). This will be referred to as <API-key> below.
2. Find orgs and teams scope and model name, model version.

In the example below the <org> is nvidia and <team> is riva, <model-name> is speechtotext_en_us_conformer_xl, <version> is deployable_v4.0_export_v2, <file-name> is Conformer-CTC-XL_spe-128_en-US_Riva-ASR-SET-4.0.riva

The screenshot shows the NGC Catalog interface for the 'RIVA Conformer-XL ASR English' model. The left sidebar includes 'Explore Catalog', 'Enterprise Catalog', 'Collections', 'Containers', 'Helm Charts', 'Models' (which is selected), and 'Resources'. It also shows 'NGC Catalog v1.177.6'. The main area shows the model's description, publisher (NVIDIA), and a 'File Browser' tab. The 'File Browser' tab has a dropdown for 'version' set to 'deployable_v4.0_export_v2'. A table lists a single file: 'Conformer-CTC-XL_spe-128_en-US_Riva-ASR-SET-4.0.riva'. The 'file name' column is highlighted with a red box. Red arrows point from the labels 'org', 'team', and 'model name' to the corresponding parts of the URL in the browser address bar: 'catalog.ngc.nvidia.com/orgs/nvidia/teams/riva/models/speechtotext_en_us_conformer_xl/files'.

Note: Not all models are within a team. In this case, the model name follows directly after the org name.

3. Generate a Bearer token.

Note: The token is valid for 15 minutes. If the model is not in a team context, /<team> is omitted.

```
API_KEY=<API-key>
TOKEN=$(curl -s -u "\$oauthtoken":\$API_KEY" -H 'Accept:application/json'
  'https://authn.nvidia.com/token?service=ngc&scope=group/ngc:<org>&group/
  ngc:<org>/<team>" | jq -r '.token')
```

4. Download the model using curl.

a). Download the model ZIP using curl.

Note: If the model is not in a team context, team/<team>/ is omitted.

```
echo "Download Model ZIP"

curl -LO 'https://api.ngc.nvidia.com/v2/org/<org>/team/<team>/models/<model-
name>/versions/<version>/zip' -H "Authorization: Bearer ${TOKEN}" -H "Content-
Type: application/json"
```

Example model zip download:

```
echo "Download Model ZIP"

curl -LO 'https://api.ngc.nvidia.com/v2/org/nvidia/team/riva/models/
speecttotext_en_us_conformer_xl/versions/trainable_v4.0/zip' -H
"Authorization: Bearer ${TOKEN}" -H "Content-Type: application/json"
```

b). Download a model file using curl

Note: If the model is not in a team context, team/<team>/ is omitted.

```
echo "Download Model file"

curl -LO --request GET 'https://api.ngc.nvidia.com/v2/org/<org>/team/<team>/
models/<model-name>/versions/<version>/files/<file-name>' -H "Authorization:
Bearer ${TOKEN}" -H "Content-Type: application/json"
```

Example model file download:

```
echo "Download Model file"
curl -LO --request GET 'https://api.ngc.nvidia.com/v2/org/nvidia/team/
riva/models/speecttotext_en_us_conformer_xl/versions/trainable_v4.0/files/
Conformer-CTC-XL_spe-128_en-US_Riva-ASR-SET-4.0.nemo' -H "Authorization:
Bearer ${TOKEN}" -H "Content-Type: application/json"
```

5. Download the model using wget.

a). Download the model ZIP using wget.

Note: If the model is not in a team context, team/<team>/ is omitted.

```
echo "Download Model ZIP"
wget --header="Authorization: Bearer $TOKEN" \
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/<org>/team/<team>/models/<model-name>/
versions/<version>/zip' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

Example model zip download:

```
echo "Download Model ZIP"

wget --header="Authorization: Bearer $TOKEN" \
```

```
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/nvidia/team/riva/models/
speechtotext_en_us_conformer_xl/versions/trainable_v4.0/zip' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

b). Download a model file using wget

Note: If the model is not in a team context, `team/<team>/` is omitted.

```
echo "Download Model file"

wget --header="Authorization: Bearer $TOKEN" \
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/<org>/team/<team>/models/<model-name>/
versions/<version>/files/<file-name>' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

Example model file download:

```
echo "Download Model file"

wget --header="Authorization: Bearer $TOKEN" \
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/nvidia/team/riva/models/
speechtotext_en_us_conformer_xl/versions/trainable_v4.0/files/Conformer-CTC-
XL_spe-128_en-US_Riva-ASR-SET-4.0.nemo' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

16.5. Downloading Models via Web UI

How to download NGC models via the Web UI.

Downloading the Latest Version of a Model

To download the latest version of a model to your local machine, performing the following.

1. Open the NGC UI, click Catalog > Containers from the left navigation and then locate and select the model you want to download.
2. Click the actions menu from the top right corner of the model detail page and then click Download <version>.

Example

Catalog > Models > PeopleSemSegnet

PeopleSemSegnet

Overview Version History File Browser Related Collections

PeopleSemSegNet Model Card

Model Overview

The model described in this card segments one or more "person" object within an image and returns a semantic mask within an image.

Description
Semantic segmentation of persons in an image.

Download ▾

- Browser (Direct Download)
- WGET
- CLI

Downloading a Specific Version of a Model

To download a specific version of a model, performing the following.

1. Open the NGC UI, click Models from the top content type ribbon and then locate and click the model you want to download.
2. Click the Version History tab, then click the actions icon for the version to download and then click Download .

The following diagram illustrates the steps.

Catalog > Models > PeopleSemSegnet

PeopleSemSegnet

Download ▾

Overview Version History File Browser Related Collections

Search versions...

Version	Date	Accuracy	Epochs	Batch Size	GPU	Size	Actions
deployable_quantized_vanilla_unet_onnx_v2.0	03/13/2024 3:48 PM	-	0 Epochs	Batch Size: 1	GPU: V100	118.71 MB	...
deployable_shuffleseq_unet_onnx_v1.0	03/13/2024 3:48 PM	80	120 Epochs	Batch Size: 1	GPU: V100	118.71 MB	...
deployable_vanilla_unet_onnx_v2.0.1	03/13/2024 3:47 PM	80	120 Epochs	Batch Size: 1	GPU: V100	118.71 MB	...
deployable_quantized_vanilla_unet_v2.0	08/22/2022 11:55 AM	80	120 Epochs	Batch Size: 1	GPU: V100	118.71 MB	...

Description
Semantic segmentation of persons in an image.

Publisher
NVIDIA

Latest Version
deployable_quantized_vanilla_unet_onnx_v2.0

Modified
March 13, 2024

Size
118.71 MB

Download ▾

- G
- V
- Download
- View Files

16.6. NVIDIA Signed Models in the NGC Catalog

This chapter provides NGC end users with guidance on locating signed models in the NGC Catalog and verifying the NVIDIA signature to confirm their authenticity and trusted origin.

Software signing is a critical security feature that verifies the authenticity and integrity of distributed software. It ensures that the software originates from a trusted publisher and has not been altered after signing. This process helps users determine which software to trust and allows for secure execution.

Model signing applies a digital signature to verify the model's origin and ensure its files remain unaltered. NVIDIA has integrated the [OpenSSF](#) model signing framework into the NGC Catalog to offer enhanced security to our users. Beginning in April 2025, all NVIDIA models published on the NGC Catalog will be signed.

16.6.1. Checking for Signed Models

There are two methods to determine if a model is signed: through the NGC Catalog's user interface (UI) or via the NGC Command-Line Interface (CLI).

16.6.2. Using the NGC Catalog UI

Click on Catalog > Models from the left navigation menu. Select a model to view the details. On the details page, you will see the Signed Model badge on the left panel and the Signed badge on the pane in the Version History tab.

16.6.3. Using the NGC CLI

To find a signed model using the CLI, issue the following:

```
$ ngc registry model list <model_name>
```

Example: List all models with name containing 'tao':

```
$ ngc registry image list 'nvidia/tao/*'
```

If the model has a signed version, you can see a value `True` populated in the last Has Signed Version column.

```
Name,Repository,Latest Version,Application,Framework,Precision,Last Modified,Permission,Access Type,Associated Products,Has Signed Version
RIVA Diarizer Embedding Extractor,nvidia/tao/diarizer_titanet_small,deployable_v1.0,Speaker Diarization,NeMo,FP16,"Oct 06, 2023",unlocked,,,None
TAO Pretrained DINO with Foundational Model Backbone,nvidia/tao/dino_with_finetuned_backbone,trainable_v1.0,Object Detection,TAO Toolkit,FP32,"Jan 05, 2024",unlocked,,,None
DashCamNet,nvidia/tao/dashcamnet,pruned_onnx_v1.0.5,Object Detection,TAO Toolkit,FP32,"Mar 17, 2025",unlocked,,,True
Domain Classification English Bert,nvidia/tao/domainclassification_english_bert,deployable_v1.0,Domain Classification,Transfer Learning Toolkit,FP32,"Oct 06, 2023",unlocked,,,None
RIVA Diarizer Neural VAD,nvidia/tao/diarizer_vad_multilingual_marblenet,deployable_v1.0,Speaker Diarization,NeMo,FP16,"Apr 04, 2023",unlocked,,,None
```

16.6.4. Finding NVIDIA's Root Certificate

The NVIDIA Model Signing Root Certificate can be used to validate its authenticity.

You can download the root certificate from the API using the following link: <https://api.ngc.nvidia.com/v2/catalog/models/public-key>.

```
-----BEGIN CERTIFICATE-----
MIICLjCCAbSgAwIBAgIUD8GTT5Xr1JC6tltkzsXnMzOltaWwCgYIKoZIZj0EAwMw
RTELMAkGA1UEBhMCVVMMxDzANBgNVBAoMBk5WSURJQTElMCMGA1UEAwctTlZJRE1B
IElvZGVsIFNpZ25pbmcgUm9vdCBDQTAgFw0yNTAzMDcxNjU4MzFaGA8yMDU1MDIY
```

```

ODE2NTgzM沃RTELMAkGA1UEBhMCVVMxDzANBgNVBAoMBk5WSURJQTE1MCMGA1UE
AwwcTlZJRE1BIE1vZGVsIFNpZ25pbmcgUm9vdCBDQTB2MBAGBYqGSM49AgEGBSuB
BAAiA2IABNUiPGna2UfkGyEcdWjpPP9XH0Y5Q50XdCag/QDzJTb/v/KHqHGT8bY
Kc13PWETx6FXhEhoYgy/D53hRS1m56u1GrbwUSbjMzQ+FFQCDiYaN5Ggg1P2MZUX
/cYN5s/bUaNjMGewHQYDVR0OBByEFJYhbAEcpVyxvlOPkUMo03mjrvCPMB8GA1Ud
IwQYMBaAFJYhbAEcpVyxvlOPkUMo03mjrvCPMA8GA1UdEwEB/wQFMAMBAf8wDgYD
VR0PAQH/BAQDAgEGMAoGCCqGSM49BAMDA2gAMGUCMBrmk2HCI4B+LxssWntsfTqu
QRoh8Po8mkyhhChK16Ye506esHLerO7rwyJ7DOgCaQIxAOZmzHMhxJtJI1Sng+K
R+IB/AxwalteUJdn5vWdsbj8g7hja2rauePEjQB2AuNw==

-----END CERTIFICATE-----

```

16.6.5. Verifying Model Signature

NVIDIA has been actively collaborating with the Open Source Security Foundation ([OpenSSF](#)) to enhance the security of artificial intelligence (AI) and machine learning (ML) models through initiatives like model signing.

NVIDIA's model signing provides a secure and transparent way to sign, verify, and protect software artifacts and AI models.

Download Verification Tools

The Git repository at [sigstore/model-transparency](#) contains all the necessary tools required to verify a signed model.

Downloading Model Version and Signature

To download a specific version of a model, performing the following.

Example

```
$ ngc registry model download-version "nvidia/tao/dashcamnet:pruned_onnx_v1.0.5"
```

The sample output is:

```
{
  "download_end": "2025-03-17 09:50:50",
  "download_start": "2025-03-17 09:50:48",
  "download_time": "2s",
  "files_downloaded": 4,
  "local_path": "C:\\\\Users\\\\NVIDIA\\\\dashcamnet_vpruned_onnx_v1.0.5",
  "size_downloaded": "5.24 MB",
  "status": "COMPLETED"
}
```

To download a signature of a specific model, performing the following.

Example

```
$ ngc registry model download-version-signature "nvidia/tao/dashcamnet:pruned_onnx_v1.0.5"
```

The sample output is:

```
{
  "local_path": "C:\\\\Users\\\\NVIDIA\\\\dashcamnet_vpruned_onnx_v1.0.5\\\\result.sigstore",
  "status": "COMPLETED"
}
```

Verifying Model Version Signature

Navigate to the downloaded `model-transparency-main` directory and run the command:

```
$ C:\Users\NVIDIA\Downloads\model-transparency-main>
python verify.py --sig_path $sig_path --model_path $model_path pki --root_certs
$public_key
where
```

- ▶ <sig_path>: Path to result.sigstore of downloaded model
- ▶ <model_path>: Path of downloaded model
- ▶ <public_key>: Path of public key downloaded from endpoint--<https://api.ngc.nvidia.com/v2/catalog/models/public-key>

The expected output is as follows:

```
python ./src/verify.py --sig_path \users\NVIDIA
\dashcamnet_vpruned_onnx_v1.0.5\result.sigstore --model_path \Users\NVIDIA
\dashcamnet_vpruned_onnx_v1.0.5 pki --root_certs \Users\NVIDIA\Downloads
\publicKey.pem
INFO:__main__:Creating verifier for pki
INFO:__main__:Verifying model signature from \users\NVIDIA
\dashcamnet_vpruned_onnx_v1.0.5\result.sigstore
INFO:__main__:all checks passed
```

Chapter 17. Resources

Description of NGC resources, their purpose, and how to use them.

17.1. What are Resources?

NGC resources offer documentation, code samples, and many other assets such as Jupyter Notebooks, deployment pipelines, and step-by-step instructions and scripts for creating deep learning models. These scripts have sample performance and accuracy metrics that allow you to compare your results. These scripts provide expert guidance on building DL models for image classification, language translation, text-to-speech, and more. Data scientists can quickly build performance-optimized models by easily adjusting the hyperparameters.

17.2. Searching and Filtering Resources

To view NGC resources, navigate to Catalog > Resources in the left menu. Use the search bar at the top of the page to filter on resources or sort the list using the criteria from the dropdown menu. You can refine your search by utilizing the filters under Use Case, NVIDIA Platform, NVIDIA AI Enterprise, and many others. The descriptions for the filtering options are as follows:

- ▶ Use Case: Choose from the available common use cases.
- ▶ NVIDIA Platform: Choose from Clara, DeepStream, TensorFlow, and many others.
- ▶ NVIDIA AI Enterprise Support: Select this option to find publicly available resources supported by NVIDIA AI Enterprise.
- ▶ NVIDIA AI Enterprise: Choose products under this to see entitled entities.
- ▶ Quick Deploy: Select this option to find containers to quickly deploy your artifacts in a Jupyter environment.
- ▶ Framework: Filter by framework such as MXNet or NeMo.
- ▶ Industry: Select by available industries.
- ▶ Solution: Select the solution that closely matches your use case.
- ▶ Publisher: Filter by publisher.

- ▶ Type: Filter by type.
- ▶ Language: Select by language.
- ▶ Other: Miscellaneous options.

The following example page shows the search and filtering options available when logging in as a guest user:

The screenshot shows the NVIDIA NGC Catalog Resources page. On the left, there's a sidebar with navigation links: Explore Catalog, Collections, Containers, Helm Charts, Models, and Resources (which is highlighted). Below the sidebar, it says "NGC Catalog v1.177.6". The main content area has a header "Catalog > Resources" and a sub-header "Resources". A large text block explains the catalog offers step-by-step instructions and scripts through Jupyter Notebooks for various use cases, including machine learning, computer vision, and conversational AI. Below this, there's a search bar with "Search resources..." and a dropdown menu set to "Most Popular". A red box highlights this search/filter section. To the right, there's a list of resources under "Displaying 263 resources". Each resource item includes a thumbnail, the name, a brief description, and two buttons: "View Labels" and "Learn More". The resources shown are: NVIDIA TUTORIALS (Getting Start with Recommender ...), RIVA (Riva Skills Quick Start), DEEP LEARNING EXAMPLES (TAO).

17.3. Downloading Resources via the NGC CLI

To download a resource from the registry to your local disk, specify the resource name and version.

```
$ nvc registry resource download-version nvidia/<resource-name>:<version>
```

Example: Downloading a resource to the current directory.

```
$ nvc registry resource download-version nvidia/gnmt_v2_for_tensorflow:1
```

The following is an example showing the output confirming completion of the download:

```
Downloaded 130.6 KB in 1s, Download speed: 130.6 KB/s
```

```
Transfer id: gnmt_v2_for_tensorflow_v1 Download status: Completed.
Downloaded local path: C:\gnmt_v2_for_tensorflow_v1
```

```
Total files downloaded: 35
Total downloaded size: 130.6 KB
Started at: 2019-03-20 11:24:00.168000
Completed at: 2019-03-20 11:24:01.176000
Duration taken: 1s seconds
```

The model is downloaded to a folder that corresponds to the model name in the current directory. You can specify another path using the -d option.

Example: Downloading a resource to a specific directory (/resources).

```
$ ngc registry resource download-version nvidia/gnmt_v2_for_tensorflow:1 -d ./resources
Downloaded 130.6 KB in 1s, Download speed: 130.6 KB/s
-----
Transfer id: gnmt_v2_for_tensorflow_v1 Download status: Completed.
Downloaded local path: C:\resources\gnmt_v2_for_tensorflow_v1
Total files downloaded: 35
Total downloaded size: 130.6 KB
Started at: 2019-03-20 11:24:00.168000
Completed at: 2019-03-20 11:24:01.176000
Duration taken: 1s seconds
```

17.4. Downloading Resources via WGET/cURL

17.4.1. Downloading Guest Access Resources via WGET/cURL

How to download guest access resources via WGET/cURL.

To get a copy of a single file from an NGC Resource via wget/curl you can use the URL provided in the NGC UI.

1. Open the NGC UI, click Resources from the left navigation menu and then locate and click the resource you want to download.

The NGC catalog offers step-by-step instructions and scripts through Jupyter Notebooks for various use cases, including machine learning, computer vision, and conversational AI. These resources help you examine, understand, customize, test, and build AI faster, while taking advantage of best practices.

2. Click the Download button on the top right or navigate to the File Browser tab.

The models trained in TAO Toolkit are deployed to NVIDIA's Inference SDK's such as DeepStream, Riva etc via TensorRT. NVIDIA TensorRT is an SDK for high-performance deep learning inference. TensorRT provides APIs and parsers to import trained models from all major deep learning frameworks. It then generates optimized runtime engines deployable in the datacenter as well as in automotive and embedded environments. To understand TensorRT and its capabilities better, refer to the official [TensorRT documentation](#). While the conversational AI models trained using TAO Toolkit can be consumed via TensorRT only via Riva, the computer vision models trained by TAO Toolkit can be consumed by TensorRT, via the tao-converter tool. The **TAO Converter** parses the exported .etlt model file, and generates an optimized TensorRT engine. These engines can be generated to support inference at low precision, such as FP16 or INT8.

3. Select the Copy curl cmd or Copy wget cmd to copy the relevant code to the clipboard.

The screenshot shows the NVIDIA NGC Catalog interface. On the left, there's a sidebar with the NVIDIA TAO logo and some descriptive text about the TAO Converter. The main area has tabs for Overview, Version History, File Browser (which is selected), Release Notes, and Related Collections. In the File Browser tab, a dropdown menu shows 'v5.1.0_8.6.3.1_x86'. A search bar contains 'Search files...'. Below is a table with two rows:

File	Size	Modified	Actions
README.txt	654 B	5 months ago	...
tao-converter	442 kB	5 months ago	...

On the far right of the table, there's a context menu with three options: 'Copy curl cmd', 'Copy wget cmd', and 'Download File'. The 'Download File' option is highlighted with a red box.

The following will be added to your clipboard for the `wget` command:

```
wget --content-disposition 'https://api.ngc.nvidia.com/v2/resources/org/nvidia/team/tao/tao-converter/v5.1.0_8.6.3.1_x86/files?redirect=true&path=tao-converter' -O tao-converter
```

Similarly, the following will be added to your clipboard for the `curl` command:

```
curl -LO 'https://api.ngc.nvidia.com/v2/resources/org/nvidia/team/tao/tao-converter/v5.1.0_8.6.3.1_x86/files?redirect=true&path=tao-converter'
```

17.4.2. Downloading Authenticated Access Resources via WGET/cURL

How to download authenticated access resources via WGET/cURL.

1. Obtain an NGC API Key. See [Generating NGC API Keys](#). This will be referred to as <API-key> below.
2. Find orgs and teams scope and resource name, resource version.

In the example below, the <org> is nvidia and <team> is riva, <resource-name> is riva_quickstart, <version> is 2.15.0, <file-name> is nemo2riva-2.15.0-py3-none-any.whl

The screenshot shows the NGC Catalog interface for the Riva Skills Embedded Quick Start resource. The left sidebar has 'Resources' selected. The main content area shows the resource details and a file browser. The 'File Browser' tab is active. In the file list, the 'version' dropdown is highlighted with a red box, set to '2.15.0'. The 'file name' column header is also highlighted with a red box. The file list contains several files: 'asr_lm_tools', 'examples', 'protos', 'config.sh', 'nemo2riva-2.15.0-py3-none-any.whl', 'riva_clean.sh', 'riva_init.sh', and 'riva_start.sh'. All files were modified 2 months ago.

3. Generate a Bearer token.

Note: The token is valid for 15 minutes. If the resource is not in a team context, / <team> is omitted.

```
API_KEY=<API-key>
TOKEN=$(curl -s -u "\$oauthtoken":'$API_KEY' -H 'Accept:application/json'
  'https://authn.nvidia.com/token?service=ngc&scope=group/ngc:<org>&group/
  ngc:<org>/<team>' | jq -r '.token')
```

4. Download the resource using curl.

a). Download the resource ZIP using curl.

Note: If the resource is not in a team context, team/<team>/ is omitted.

```
echo "Download Resource ZIP"
curl -LO 'https://api.ngc.nvidia.com/v2/org/<org>/team/<team>/resources/
<resource-name>/versions/<version>/zip' -H "Authorization: Bearer ${TOKEN}" -H
"Content-Type: application/json"
```

Example resource zip download:

```
echo "Download Resource ZIP"
curl -LO 'https://api.ngc.nvidia.com/v2/org/nvidia/team/riva/resources/
riva_quickstart/versions/2.9.0/zip' -H "Authorization: Bearer ${TOKEN}" -H
"Content-Type: application/json"
```

b). Download a resource file using curl.

Note: If the resource is not in a team context, team/<team>/ is omitted.

```
echo "Download Resource file"
curl -LO --request GET 'https://api.ngc.nvidia.com/v2/org/<org>/team/
<team>/resources/<resource-name>/versions/<version>/files/<file-name>' -H
"Authorization: Bearer ${TOKEN}" -H "Content-Type: application/json"
```

Example resource file download:

```
echo "Download Resource file"
curl -LO --request GET 'https://api.ngc.nvidia.com/v2/org/nvidia/team/riva/resources/riva_quickstart/versions/2.9.0/files/riva_init.sh' -H "Authorization: Bearer ${TOKEN}" -H "Content-Type: application/json"
```

5. Download the resource using wget.

a). Download the resource ZIP using wget.

Note: If the resource is not in a team context, `team/<team>/` is omitted.

```
echo "Download Resource ZIP"

wget --header="Authorization: Bearer $TOKEN" \
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/<org>/team/<team>/resources/<resource-name>/versions/<version>/zip' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

Example resource zip download:

```
echo "Download Resource ZIP"

wget --header="Authorization: Bearer $TOKEN" \
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/nvidia/team/riva/resources/riva_quickstart/versions/2.9.0/zip' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

b). Download a resource file using wget

Note: If the resource is not in a team context, `team/<team>/` is omitted.

```
echo "Download Resource file"

wget --header="Authorization: Bearer $TOKEN" \
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/<org>/team/<team>/resources/<resource-name>/versions/<version>/files/<file-name>' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

Example resource file download:

```
echo "Download Resource file"

wget --header="Authorization: Bearer $TOKEN" \
--header="Content-Type: application/json" \
'https://api.ngc.nvidia.com/v2/org/nvidia/team/riva/resources/riva_quickstart/versions/2.9.0/files/riva_quickstart.sh' \
2>&1 \
| awk '/Location:/ {print $2}' \
| xargs wget --content-disposition
```

17.5. Downloading Resources via Web UI

1. From a browser, go to the [NGC Catalog](#) website.
2. Sign in. Refer to [Registering and Activating Your NGC Account](#).

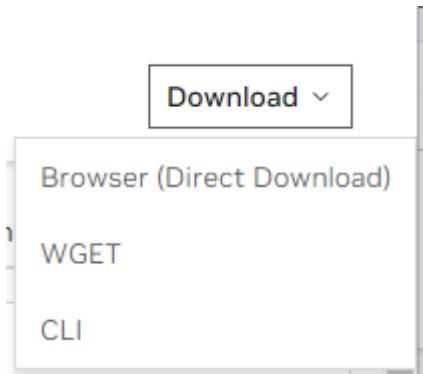
- Click Resources from the left navigation menu.

You will see a page with available Resources.

- Select one of the Resource cards.

The details page provides additional information for each Resource.

The Download menu on the upper right corner offers three different ways to download the Resource.



Click either Browser (Direct Download) to start the download, or WGET or CLI to copy the download commands to the clipboard.

The File Browser tab lets you see the file content of the Resources.

Catalog > Resources > Object Detection using TAO DetectNet_v2

Object Detection using TAO DetectNet_v2

Download ▾

The screenshot shows the "File Browser" tab selected in the top navigation bar. On the left, there is a sidebar with the NVIDIA TUTORIALS logo and sections for "Description" and "Publisher". The "Description" section states: "Detectnet_v2 is an object detection model. Object detection is a popular computer vision technique that can detect one or multiple objects in a frame." The "Publisher" section lists "NVIDIA". The main content area displays a table of files:

File	Size	Modified	Actions
files.zip	370 kB	a year ago	***

Chapter 18. Helm Charts

Helm is an application package manager running on top of Kubernetes. This section describes how to use access Helm charts from the NGC Catalog.

Prerequisites

These instructions assume the following prerequisites are met.

- ▶ Helm v 2.16.1 installed

The helm push plugin does not support Helm v3 yet - make sure you are using v2.16.x.

See <https://github.com/helm/helm/releases/tag/v2.16.1> for the Helm download and installation instructions.

- ▶ NGC organization account

See [Setting Up and Activating Your NGC Account](#) for instructions.

18.1. Setting Up an NGC Helm Repository

1. Obtain an NGC API Key.

See [Generating Your NGC API Key](#) for instructions.

2. Export the API Key for use in commands.

```
$ export NGC_API_KEY=<your-api-key>
```

3. Add the NGC org to your Helm repository.

```
$ helm repo add <repo-name> https://helm.ngc.nvidia.com/<org-name> --username=\$oauthToken --password=$NGC_API_KEY
```

Where `<repo-name>` is a name of your choosing by which you will reference the repository.

18.2. Searching for Available Helm Charts in the NGC Catalog

To view a list of available Chart packages in your org, issue the following.

```
$ helm search <repo-name>
```

Example

```
$ helm search nvidian
```

18.3. Fetching Helm Charts

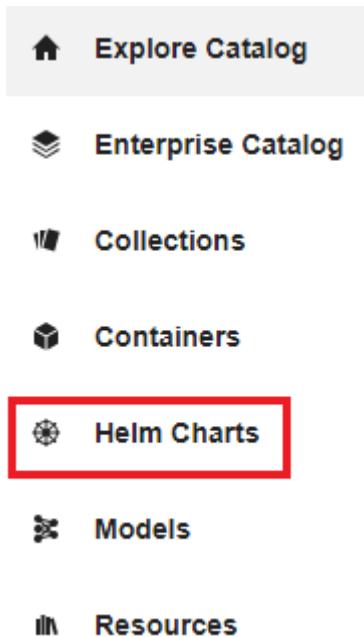
To download (or "fetch") a Helm chart package from the repo, issue the following.

```
$ helm fetch <repo-name>/<chart-name>
```

18.4. Using the NGC Website to View the List of Helm Charts and Get Fetch Commands

From the NGC website you can

- ▶ View the contents of the Helm chart repository.
 - ▶ Get the push command for a specific Helm chart in the repository.
1. From a browser, log in to <https://ngc.nvidia.com>.
 2. If you are a member of more than one org, select the one that contains the Helm charts that you are interested in, then click Sign In.
 3. Click Helm Charts from the left-side navigation pane.



The page presents cards for each available Helm chart.

The screenshot shows the Helm Charts page in the NGC Catalog. The top navigation bar includes the NGC Catalog logo, a search bar, and a dropdown menu. The sidebar on the left is identical to the one in the previous screenshot. The main content area has a title "Helm Charts" with a subtext explaining its purpose. Below this is a search bar and a "Most Popular" dropdown. The main area displays three cards:

- NVIDIA GPU Operator**: Deploy and Manage NVIDIA GPU resources in Kubernetes. Status: NVIDIA AI Enterprise Supported.
- NVIDIA Network Operator**: Provides an easy way to install, configure and manage the lifecycle of NVIDIA... Status: NVIDIA AI Enterprise Supported.
- Triton Management Service (TMS)**: A Helm chart for simplifying the installation of TMS. Status: NVIDIA AI Enterprise Essentials.

Each card has "View Labels" and "Learn More" buttons at the bottom.

4. Select one of the Helm chart cards.

The page for each Helm chart provides information about the chart.

Catalog > Helm Charts > TAO Toolkit API Helm

TAO Toolkit API Helm

[Fetch Version ▾](#)



Description
Helm Chart to deploy TAO Toolkit APIs

Publisher
NVIDIA

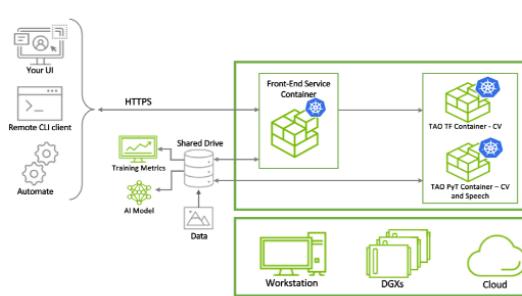
Latest Version
5.2.0

Compressed Size
131.46 KB

Modified
December 12, 2023

TAO Toolkit API - Helm Chart

TAO Toolkit API is a Kubernetes service that enables building end-to-end AI models using custom datasets. In addition to exposing TAO Toolkit functionality through APIs, the service also enables a client to build end-to-end workflows - creating datasets, models, obtaining pretrained models from NGC, obtaining default specs, training, evaluating, optimizing and exporting models for deployment on edge. It can be easily installed on a Kubernetes cluster (local / AWS EKS) using a Helm chart along with minimal dependencies. TAO toolkit jobs can be run using GPUs available on the cluster and can scale to a multi-node setting.



TAO Toolkit API overview

One can develop client applications on top of the provided API, such as a Web-UI application, or use the provided TAO remote client CLI.

The Fetch Command section shows the command to use for downloading the Helm chart package.

Click either the Fetch download button from the upper right corner or the copy icon next to the fetch command to copy the fetch command to the clipboard.

The File Browser tab lets you see the file content of the Helm chart package.

Catalog > Helm Charts > TAO Toolkit API Helm

TAO Toolkit API Helm

[Fetch Version ▾](#)



Description
Helm Chart to deploy TAO Toolkit APIs

Publisher
NVIDIA

Latest Version
5.2.0

File Browser

Version
5.2.0

File
tao-toolkit-api

Size	Modified

Chapter 19. SDK and AI Toolkits

NVIDIA AI toolkits provide libraries and tools to train, fine-tune, optimize and deploy pre-trained NGC models across a broad domain of industries and AI applications.

These include

- ▶ An [AI-assisted annotation tool](#) to help users label their datasets for training.
- ▶ A [transfer learning toolkit](#) to fine-tune pre-trained models with user data, saving the cost of training from scratch.
- ▶ [Federated learning](#) that preserves privacy by allowing users to collaborate and train AI models without sharing private data between clients.
- ▶ The [NeMo toolkit](#) to quickly build state-of-the-art models for speech recognition and natural language processing.
- ▶ The Service Maker toolkit that exposes trained models as a gRPC service that can be scaled and easily deployed on a Kubernetes service.
- ▶ Models built using the toolkits can be integrated inside client applications and deployed in production with the software development kits offered by NVIDIA. These SDKs leverage [TensorRT](#) and the [Triton Inference Server](#) as foundational building blocks.

NVIDIA provides AI toolkits and application frameworks targeted towards industries or specific use cases. These are listed in the table below.

Industry or AI Application	AI Toolkits
Healthcare and Life Sciences	NVIDIA Clara
Streaming video analytics	NVIDIA Metropolis NVIDIA DeepStream
Robotics	NVIDIA Isaac Sim
5G acceleration	NVIDIA Aerial
Recommender Systems	NVIDIA Merlin
Conversational AI	NVIDIA Riva

Chapter 20. Deep Learning Frameworks

Deep learning frameworks offer building blocks for designing, training and validating deep neural networks, through a high level programming interface. Widely used deep learning frameworks such as MXNet, PyTorch, TensorFlow and others rely on GPU-accelerated libraries such as cuDNN, NCCL and DALI to deliver high-performance multi-GPU accelerated training.

Developers, researchers and data scientists can get easy access to NVIDIA optimized deep learning framework containers with deep learning examples, that are performance tuned and tested for NVIDIA GPUs. This eliminates the need to manage packages and dependencies or build deep learning frameworks from source

20.1. GPU Operator

The Operator Framework within Kubernetes takes operational business logic and allows the creation of an automated framework for the deployment of applications within Kubernetes using standard Kubernetes APIs and kubectl. The NVIDIA GPU Operator automates the management of all NVIDIA software components needed to provision GPUs within Kubernetes. NVIDIA, Red Hat, and others in the community have collaborated on creating the GPU Operator. The GPU Operator is an important component of the [NVIDIA EGX](#) software-defined platform that is designed to make large-scale hybrid-cloud and edge operations possible and efficient.

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



Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2020-2025 NVIDIA CORPORATION & AFFILIATES. All rights reserved.