Table of Contents

Chapter 1. NGC Private Registry for Enterprise

Chapter 2. Getting Started

2.1. Obtaining a Private Registry

2.2. Activating a New NGC Account

2.2.1. Joining an NGC Org or Team Using Single Sign-on

2.2.2. Joining an Org or Team with a New NVIDIA Account

2.2.3. Joining an Org or Team with an Existing NVIDIA Account

2.3. Generating Your NGC API Key

2.4. Managing Users and Teams in NGC

2.4.1. NGC Registry User Roles

2.4.2. Creating Teams

2.4.3. Creating Users

2.4.4. Adding Users to Teams

2.4.5. Changing User Roles

2.5. Introduction to the NGC Registry CLIs

2.5.1. Installing NGC Registry CLI

2.5.2. Getting Help Using NGC CLI

2.5.3. Configuring the CLI for Your Use

2.5.4. Running the Diagnostics

2.5.5. Managing Users and Teams

2.5.5.1. Inviting users to the organization’s NGC account

2.5.5.2. Creating teams

2.5.5.3. Adding users to teams

2.5.5.4. Creating a team and adding a user in the same command

2.5.5.5. Creating a team and adding a user in the same command

Chapter 3. Docker Containers

3.1. What Is A Docker Container?

3.2. Why Use A Container?

3.3. Using NGC Container Registry from the Docker Command Line

3.3.1. Accessing the NGC Container Registry

3.3.2. Uploading an NVIDIA Container Image onto Your System

3.3.3. Tagging and Pushing a Container Image

3.4. Using the Container Registry

3.4.1. Viewing Container Image Information

3.4.2. Pulling a Container Image
6.6. Manage Helm Charts Using the Helm CLI.................................................................................67
  6.6.1. Setting Up an NGC Helm Repository...............................................................................67
  6.6.2. Searching for Available Helm Charts............................................................................68
  6.6.3. Fetching Helm Charts....................................................................................................68
  6.6.4. Adding Helm Charts to a Private NGC Org/Team..........................................................68
  6.6.5. Removing Helm Charts from a Private NGC Org/Team..................................................68

Chapter 7. Getting Support for NGC container registry..............................................................69
Chapter 1. NGC Private Registry for Enterprise

This document describes how to use the NVIDIA® GPU Cloud (NGC) private registry. This guide assumes the user is familiar with Linux and Docker, and has access to an NVIDIA GPU-based computing solution, such as an NVIDIA DGX system, that is configured for internet access and prepared for running NVIDIA GPU-accelerated Docker containers.

As data scientists build custom content, storing, sharing, and versioning of this valuable intellectual property is critical to meet their company’s business needs. To address these needs, NVIDIA has developed the NGC private registry to provide a secure space to store and share custom containers, models, model scripts, and Helm charts within your enterprise. The NGC Private registry is available to both the customers of DGX and NGC-Support services.

Increased Collaboration

We all are used to working collaboratively using tools such as Slack or Microsoft teams, to share our content and ensure that our colleagues are all aligned. The NGC private registry was built to share your AI content such as containers, models, helm charts within your organization. This empowers the key stakeholders in your organization to work collaboratively with each other without having to reinvent the wheel, saving valuable resources and bringing your products to market faster.

Enterprise Ready

When sharing content across a large organization, it is important to ensure that you are able to manage the users. The comprehensive user and team management in an NGC private registry allows administrators to control access to content stored in the registry.

With the power of cloud, the content stored in the NGC Private Registry is always available with redundant storage that can be accessed from anywhere, making it extremely easy to get to your content.
Chapter 2. Getting Started

2.1. Obtaining a Private Registry

This chapter provides instructions for DGX customers on obtaining a private registry.

After purchasing a support entitlement with NVIDIA, the end-customer will receive an NVIDIA Entitlement Certificate via email. The email will include all the pertinent instructions needed to register for technical support.

The following is an example of the NVIDIA Entitlement Certificate email.
The Entitlement Certificate itself is provided as a PDF attachment. The following is an example of an NVIDIA Entitlement Certificate.
The PDF also includes instructions for using the certificate.
If you already have an account, you can log into the NVIDIA Enterprise Support portal right away.

If you are a new user without an NGC Support account, click the NVIDIA Enterprise Support Registration Form link.

This link will have embedded information regarding your account and so, it is very IMPORTANT NOT to share this entitlement link outside of your organization.

Registration will provide an NGC private registry account in addition to the NVIDIA Enterprise Support account. You’ll receive a welcome email at which time you can activate (create) your NGC private registry account.

### 2.2. Activating a New NGC Account

Before using NGC, you must have an NGC account created by your organization administrator. You need an email address to set up an account. The process for activating an account depends on whether your email domain is mapped to your organization’s single sign-on (SSO). Choose one of the following processes depending on your situation for activating your NGC account.
Joining an NGC Org or Team Using Single Sign-on

This section describes activating an account where the domain of your email address is mapped to an organization’s single sign-on.

After NVIDIA or your organization administrator adds you to a new org or team within the organization, you will receive a welcome email that invites you to continue the activation and login process.

1. Click the link in the email to open your organization’s single sign-on page.
2. Sign in using your single sign-on credentials.
   The Set Your Organization screen appears.
This screen appears any time you log in.

3. Select which organization and team you want to log in under and then click **Sign In**.
   
   You can always change the org or team to any other org or team of which you are a member after you log in.
   
   The NGC web UI opens to the **NGC Catalog** dashboard.
   
   When you log in, click your username on the right corner and then select the org if you want to switch your login to a different org.
After selecting the organization, you can select the “team” from the Select a Team drop down menu next to your username.
To view artifacts in your private registry, select “PRIVATE REGISTRY” in the left hand menu. You may select containers, Helm charts, models and model scripts as needed.

2.2.2. Joining an Org or Team with a New NVIDIA Account
This section describes activating a new account where the domain of your email address is not mapped to an organization’s single sign-on.

After NVIDIA or your organization administrator sets up your NVIDIA Base Command Platform account, you will receive a welcome email that invites you to continue the activation and login process.

1. Click the link to open the Log in or Create an Account page in your browser, then enter your email address and click Continue.
2. At the **Create an Account** screen, enter your display name, email address, and new password, then agree to the NVIDIA Account Terms of Use and click **Create Account**.
A verification email is sent to your email address.

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

You are automatically directed to nvidia.com to complete your account registration.

4. At the **Almost done!** dialog, set your communications preferences and then click **Submit.**
5. Select which organization and team you want to log in under and then click **Sign In**.

You can always change the org or team to any other org or team of which you are a member after you log in.

The NGC web UI opens to the **NGC Catalog** dashboard.

When you log in, click your username on the right corner and then select the org if you want to switch your login to a different org.
After selecting the organization, you can select the “team” from the Select a Team drop down menu next to your username.
To view artifacts in your private registry, select "PRIVATE REGISTRY in the left hand menu. You may select containers, Helm charts, models and model scripts as needed.

2.2.3. Joining an Org or Team with an Existing NVIDIA Account
This section describes activating an account where the domain of your email address is not mapped to an organization’s single sign-on (SSO).

After NVIDIA or your organization administrator adds you to a new org or team within the organization, you will receive a welcome email that invites you to continue the activation and login process.

![Welcome Email](image)

1. Click the Sign In link to open the sign in dialog in your browser.
2. Enter your password and then click **Log In**.
   The Set Your Organization screen appears.

![Set Your Organization](image)

This screen appears any time you log in.

3. Select which organization and team you want to log in under and then click **Sign In**.

   You can always change the org or team to any other org or team of which you are a member after you log in.

   The NGC web UI opens to the **NGC Catalog** dashboard.

When you log in, click your username on the right corner and then select the org if you want to switch your login to a different org.
After selecting the organization, you can select the “team” from the Select a Team drop down menu next to your username.
To view artifacts in your private registry, select "PRIVATE REGISTRY" in the left hand menu. You may select containers, Helm charts, models and model scripts as needed.

2.3. Generating Your NGC API Key

This section describes how to obtain an API key to access locked container images from the NGC Registry.
1. Sign in to the NGC website.
   From a browser, go to https://ngc.nvidia.com/signin/email and then enter your email and password.
2. In the top right corner, click your user account icon and select Setup.

3. Click Get API key to open the Setup > API Key page.
   The API Key is the mechanism used to authenticate your access to the NGC container registry.
4. Click Generate API Key to generate your API key.
   A warning message appears to let you know that your old API key will become invalid if you create a new key.
5. Click Confirm to generate the key.
   Your API key appears.
   You only need to generate an API Key once. NGC does not save your key, so store it in a secure place. (You can copy your API Key to the clipboard by clicking the copy icon to the right of the API key.)
   Should you lose your API Key, you can generate a new one from the NGC website. When you generate a new API Key, the old one is invalidated.

2.4. Managing Users and Teams in NGC

This chapter applies to organization and team administrators, and explains the tasks that an organization or team administrator can perform from the NGC website.

As the NGC administrator for your organization, you can invite other users to join your organization’s NGC account. Users can then be assigned as members of teams within your organization. Teams are useful for keeping custom work private within the organization.

The general workflow for building teams of users is as follows:
1. The organization admin invites users to the organization’s NGC account.
2. The organization admin creates teams within the organization.
3. The organization admin adds users to appropriate teams, and typically assigns at least one user to be the team admin.
4. The organization or team admin can then add other users to the team.

### 2.4.1. NGC Registry User Roles

Prior to adding users and teams, familiarize yourself with the following definitions of each role.

The NGC container registry supports three different user roles:

- **Organizational Level Roles**
  - At an organizational level, there are four roles that can be assigned to a user.
    - **Registry Admin**: This user can download, upload, push/pull or delete, add/remove users and create teams within an organization.
    - **Registry User**: This user can download, upload, push/pull artifacts within an organization.
    - **User Admin**: This user can invite other user admins within an organization. This user cannot download, upload, push/pull or delete, add/remove users at an organizational level.
    - **Registry Only**: This user can download and pull artifacts within an organization.

  
  **Note:** A user must have a “Registry Only” OR “User Admin” role to be a member of the organization or any team.

- **Team Level Roles**
  - At a team level, there are four roles which can be assigned to a user.
    - **Registry Admin**: This user can download, upload, push/pull or delete, add/remove users at a team level.
    - **Registry User**: This user can download, upload, push/pull artifacts within a team.
    - **User Admin**: This user can add other team members only as user admins. This user cannot download, upload, push/pull or delete, add/remove users at a team level.
    - **Registry Only**: This user can download and pull artifacts within a team.

  **Note:** A user must be a “Registry Only User” OR “User Admin” role to be a member of the team or any team.

<table>
<thead>
<tr>
<th>Capability</th>
<th>Registry Admin</th>
<th>Registry User</th>
<th>User Admin</th>
<th>Registry Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add teams</td>
<td>#</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Add new users to org</td>
<td>#</td>
<td>X</td>
<td>#</td>
<td>X</td>
</tr>
<tr>
<td>Add new users to teams</td>
<td>#</td>
<td>X</td>
<td>#</td>
<td>X</td>
</tr>
<tr>
<td>Capability</td>
<td>Registry Admin</td>
<td>Registry User</td>
<td>User Admin</td>
<td>Registry Only</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Delete images</td>
<td>#</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>View/Edit all image information via UI and CLI</td>
<td>#</td>
<td>#</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>View all artifacts namely containers, model, model scripts</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Download all artifacts namely containers, model, model scripts</td>
<td>#</td>
<td>#</td>
<td>X</td>
<td>#</td>
</tr>
<tr>
<td>Create and push/upload all artifacts namely containers, model, model scripts</td>
<td>#</td>
<td>#</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

2.4.2. Creating Teams

Creating teams is useful for allowing users to share images within a team while keeping them invisible to other teams in the same organization. Only organization administrators can create teams.

To create a team,

1. Log on to the [NGC website](#).
2. Select Teams from the left side menu, then click the ‘+’ icon at the bottom of the screen and then click the ‘create teams’ icon.

3. Enter a team name and description, then click Add Team.

2.4.3. Creating Users

As the organization administrator, you must create user accounts to allow others to use the NGC container registry within the organization.

1. Log on to the [NGC website](#).
2. Click **Users** from the left side menu, then click the ‘+’ icon at the bottom of the screen and then click the ‘invite new user’ icon.

3. Fill out the **Invite New User** form for the new user as follows:

   a). Enter the display name and email where indicated.
   b). Click **Next** to move on the Membership tab.
c). Select the organization and the role you assign to the user.
d). Select the team and the role you wish to assign to the user.
e). Click Assign and then click Confirm when done.

An invitation email is automatically sent to the user.

2.4.4. Adding Users to Teams

Organization administrators can add users to any team in the organization. Team administrators can add users to their teams.

1. Log on to the NGC website.
2. Click Teams from the left side menu, then select the team that you want to add a user.
3. In the Active Members section, click Add User.
4. In the Add User dialog, select one of the available users, select a role, then click Add User.
5. Assign user roles and click Confirm.
Users can be members of more than one team. To add a user to another team, repeat these steps for any additional teams.

### 2.4.5. Changing User Roles

You can change user roles for any users you created.

1. Log on to the NGC website.
2. Select the registry space (org and team) for which you want to change the user role.
   - Click your user icon to select from the list of orgs, and then click Select a Team and choose the appropriate team.
3. Click Users from the left side menu.
   - A list of all the users in the current registry space appears.
4. Select the user whose role you want to change.
   - The User Information form appears.
5. Click Edit Membership.

A prompt appears for editing email addresses and membership role.

6. Assign new or edit user roles and click Confirm when done.

### 2.5. Introduction to the NGC Registry CLIs

**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.

- 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 to you as well as detailed information about each container image.

See a list of deep-learning models and resources as well as 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).

### 2.5.1. 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 Install NGC 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.

### 2.5.2. Getting Help Using NGC CLI

To run an NGC CLI command, enter "ngc" followed by the appropriate options.

To see a description of available options and command descriptions, use the option `--help` after any command or option. Because the help content is targeted to your user role, be sure to obtain an API key and then run `ngc config set` before using the `--help` option.

**Example 1:** To view a list of all the available options for ngc, enter

```
C:\>ngc -h
```

**Example 2:** To view a description of the registry image command and options, enter

```
C:\>ngc registry image -h
```

**Example 2:** To view a description of the registry image info command and options, enter

```
C:\>ngc registry image info -h
```

### 2.5.3. Configuring the CLI for Your Use

While there are options you can use for each command to specify the output type and debug mode, you can use the `ngc config set` command to establish these settings up front.

**Note:** If you plan to download locked content, be sure you have registered for an NGC account and have generated an API key, then enter your API key at the prompt.

```
ngc config set
Enter API key [no-apikey]. Choices: ['<VALID_APIKEY>', 'no-apikey']: <your-api-key>
```
2.5.4. Running the Diagnostics

You can view the following diagnostic information for the NGC CLI user:

- Current time
- Operating system
- Disk usage
- Current directory size
- Memory usage
- NGC CLI environment variables set
- API gateway connectivity
- API connectivity to the container registry
- API connectivity to the model registry
- External IP
- NGC CLI configuration values
- User information (ID, name, and email)
- User org roles
- User team roles

**Syntax**

```bash
ngc diag [all, client, install, server, user]
```

where

- **all**
  Produces the maximum amount of diagnostic output
- **client**
  Produces diagnostic output only for the client machine.
- **install**
  Produces diagnostic output only for the local installation.
- **server**
  Produces diagnostic output only for the remote server.
- **user**
  Produces diagnostic output only for the user configuration.

**Note:**

nvcr.io/nvidia, nvcr.io/partners and nvcr.io/hpc are reserved namespaces and do not grant user permissions to push or delete container images, models, helm and all artifacts.
2.5.5. Managing Users and Teams

This section applies to the organization and team administrators.

As the NGC administrator for your organization, you can invite other users to join your organization’s NGC account. Users can then be assigned as members of teams within your organization. Teams are useful for keeping custom work private within the organization.

The general workflow for building teams of users is as follows:

1. The organization admin invites users to the organization’s NGC account.
2. The organization admin creates teams within the organization.
3. The organization admin adds users to appropriate teams, and typically assigns at least one user to be the team admin.
4. The organization or team admin can then add other users to the team.

2.5.5.1. Inviting users to the organization’s NGC account

**Required Role:** Org Admin (REGISTRY_WRITE_ADMIN_ROLE)

**Syntax**

```c:\> ngc org add-user <email> <name>
```

**Example of adding John Smith (email: jsmith@example.com)**

```c:\> ngc org add-user jsmith@example.com “John Smith”
```

2.5.5.2. Creating teams

**Required Role:** Org Admin (REGISTRY_WRITE_ADMIN_ROLE)

**Syntax**

```c:\> ngc org add-team <name> <description>
```

**Example of adding Team A**

```c:\> ngc org add-team team_a “Team A”
```

---

**Team Information**

Id: 363
Name: team-a
Description: Team A
Deleted: False
---

2.5.5.3. Adding users to teams

**Required Role:** Org Admin (REGISTRY_WRITE_ADMIN_ROLE) or Team Admin (REGISTRY_WRITE_TEAM_ADMIN_ROLE)

**Syntax**

```c:\> ngc team add-user <email> <name>
```
Example of adding existing user John Smith to Team A as a regular user

c:\> ngc team add-user jsmith@example.com "John Smith" --team team-a --role REGISTRY_WRITE_USER_ROLE

Note: You do not need the --team argument if the target team is already set in your current NGC configuration.

2.5.5.4. Creating a team and adding a user in the same command

Required Role: Org Admin (REGISTRY_WRITE_ADMIN_ROLE)

Syntax

c:\> ngc org add-user <email> <name> --team <name> --role <user-role>

Example of inviting new user John Smith to Team A as a team admin

c:\> ngc org add-user jsmith@example.com "John Smith" --team team-a --role REGISTRY_WRITE_TEAM_ADMIN_ROLE

Note: You do not need the --team argument if the target team is already set in your current NGC configuration.

2.5.5.5. Creating a team and adding a user in the same command

<table>
<thead>
<tr>
<th>Role</th>
<th>Service</th>
<th>Access Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMIN</td>
<td>ACE</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>CONTAINER</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>DATASET</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>HELM</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>JOB</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>MODEL</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>MODELSCRIPT</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>ORG</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>TEAM</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>USER</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>ADMIN</td>
<td>WORKSPACE</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>EGX_ADMIN</td>
<td>EGX</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>EGX_ADMIN</td>
<td>ORG</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>EGX_ADMIN</td>
<td>TEAM</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>EGX_ADMIN</td>
<td>USER</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>Role</td>
<td>Service</td>
<td>Access Levels</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>EGX_READ</td>
<td>EGX</td>
<td>READ</td>
</tr>
<tr>
<td>EGX_USER</td>
<td>ORG</td>
<td>READ</td>
</tr>
<tr>
<td>EGX_USER</td>
<td>TEAM</td>
<td>READ</td>
</tr>
<tr>
<td>EGX_USER</td>
<td>EGX</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>EGX_USER</td>
<td>ORG</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>EGX_USER</td>
<td>TEAM</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>REGISTRY_ONLY</td>
<td>CONTAINER</td>
<td>READ</td>
</tr>
<tr>
<td>REGISTRY_ONLY</td>
<td>HELM</td>
<td>READ</td>
</tr>
<tr>
<td>REGISTRY_ONLY</td>
<td>MODEL</td>
<td>READ</td>
</tr>
<tr>
<td>REGISTRY_ONLY</td>
<td>MODELSCRIPT</td>
<td>READ</td>
</tr>
<tr>
<td>REGISTRY_ONLY</td>
<td>ORG</td>
<td>READ</td>
</tr>
<tr>
<td>REGISTRY_ONLY</td>
<td>TEAM</td>
<td>READ</td>
</tr>
<tr>
<td>REGISTRY_USER_ADMIN</td>
<td>CONTAINER</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_ADMIN</td>
<td>HELM</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_ADMIN</td>
<td>MODEL</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_ADMIN</td>
<td>MODELSCRIPT</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_ADMIN</td>
<td>ORG</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_ADMIN</td>
<td>TEAM</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_ADMIN</td>
<td>USER</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_USER</td>
<td>CONTAINER</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_USER</td>
<td>HELM</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_USER</td>
<td>MODEL</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_USER</td>
<td>MODELSCRIPT</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_USER</td>
<td>ORG</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>REGISTRY_USER_USER</td>
<td>TEAM</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>ACE</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>CONTAINER</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>DATASET</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>HELM</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>JOB</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>MODEL</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>MODELSCRIPT</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>ORG</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>TEAM</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER</td>
<td>WORKSPACE</td>
<td>READ, WRITE</td>
</tr>
<tr>
<td>USER_ADMIN</td>
<td>ORG</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>USER_ADMIN</td>
<td>TEAM</td>
<td>READ, ADMIN, WRITE</td>
</tr>
<tr>
<td>USER_ADMIN</td>
<td>USER</td>
<td>READ, ADMIN, WRITE</td>
</tr>
</tbody>
</table>
Chapter 3. 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 nvidia-docker container.

3.1. 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. DGX™ systems uses Docker containers as the mechanism for deploying deep learning frameworks.

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

3.2. Why Use A Container?

One of the many benefits to using containers is that you can install your application, dependencies and environment variables one time into the container image; rather than on each system you run on. In addition, the key benefits to using containers also include:

- Install your application, dependencies and environment variables one time into the container image; rather than on each system you run on.
There is no risk of conflict with libraries that are installed by others.

Containers allow use of multiple different deep learning frameworks, which may have conflicting software dependencies, on the same server.

After you build your application into a container, you can run it on lots of other places, especially servers, without having to install any software.

Legacy accelerated compute applications can be containerized and deployed on newer systems, on premise, or in the cloud.

Specific GPU resources can be allocated to a container for isolation and better performance.

You can easily share, collaborate, and test applications across different environments.

Multiple instances of a given deep learning framework can be run concurrently with each having one or more specific GPUs assigned.

Containers can be used to resolve network-port conflicts between applications by mapping container-ports to specific externally-visible ports when launching the container.

### 3.3. Using NGC Container Registry from the Docker Command Line

#### 3.3.1. Accessing the NGC Container Registry

You can access the NGC container registry by running a Docker command from your client computer. You are not limited to using your NVIDIA DGX platform to access the NGC container registry. You can use any Linux computer with Internet access on which Docker is installed.

Before accessing the NGC container registry, ensure that the following prerequisites are met:

- Your NGC account is activated.
- You have an NGC API key for authenticating your access to NGC container registry. For more information, see [Generating Your NGC API Key](#).
- You are logged in to your client computer as an administrator user.

An alternate approach for enabling other users to run containers without giving them sudo privilege, and without having to type sudo before each docker command, is to add each user to the docker group, with the command:

```bash
sudo usermod -aG docker $USER
```

While this approach is more convenient and commonly used, it is less secure because any user who can send commands to the docker engine can escalate privilege and run root level operations. If you choose to use this method, only add users to the docker group who you would trust with root privileges.

1. Log in to the NGC container registry.

   ```bash
   sudo docker login nvcr.io
   ```

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

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

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

```
Username: $oauthtoken
Password: my-api-key
```

Tip: When you get your API key as explained in Generating Your NGC 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.

### 3.3.2. Uploading an NVIDIA Container Image onto Your System

No container images are preloaded onto a DGX system. Instead, containers are available for download from the NGC container registry. NVIDIA has provided a number of containers for download from the NGC container registry. If your organization has provided you with access to any custom containers, you can download those as well.

Before loading an NGC container image, ensure that the following prerequisites are met:

- You have read access to the registry space that contains the container image.
- You are logged in to nvcr.io as explained in Accessing the NGC Container Registry.

Tip: To browse the available containers in the NGC container registry, use a web browser to log in to your NGC account on the NGC website [http://ngc.nvidia.com/](http://ngc.nvidia.com/).

1. Run the command to download the container that you want from the registry.

```
sudo docker pull registry/registry-space/repository:tag
```

- `registry` The URL of the container registry, which for the NGC container registry is nvcr.io.
- `registry-space` The name of the space within the registry that contains the container. For example, nvidia is the registry space for containers provided by NVIDIA.
- `repository` Repositories are collections of containers of the same name, but distinguished from each other by their tags. Think of it as the main container name.
- `tag` A tag that identifies the version of the container.

2. To confirm that the container was downloaded, list the Docker images on your system.

```
sudo docker images
```

The following are several examples of pulling container images.

- Example of pulling tensorflow:18.06-py3 from the nvidia registry space.

  ```
  ~$ sudo docker pull nvcr.io/nvidia/tensorflow:18.06-py3
  ```
Example of pulling a custom container image tagged v2.0 from the acme organization registry space.

```
sudo docker pull nvcr.io/acme/custom-image:v2.0
```

Example of pulling a custom container image tagged v2.0 from the acme/team team registry space.

```
sudo docker pull nvcr.io/acme/zoom/custom-image:v2.0
```

### 3.3.3. Tagging and Pushing a Container Image

You can upload custom images to the registry if you have write access to the registry space. Uploading a container image involves first tagging the image and then pushing the image to the registry space.

In the following examples, the user is a member of the Acme organization and the Zoom team within the Acme organization.

- **Tagging Example**
  
  This example tags a local container image `mycaffe` in the `acme/zoom` team space with "v1.5".

  ```
sudo docker tag mycaffe nvcr.io/acme/zoom/mycaffe:v1.5
  ```

- **Pushing Example**
  
  This example pushes version v1.5 of the `mycaffe` local container image to the `acme/zoom` team space:

  ```
sudo docker push nvcr.io/acme/zoom/mycaffe:v1.5
  ```

### 3.4. Using the Container Registry

The `ngc registry image` commands let you access ready-to-use GPU-accelerated container images from the registry.

#### 3.4.1. Viewing Container Image Information

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

**To list container images:**

```
\>ngc registry image list
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Repository</th>
<th>Latest Tag</th>
<th>Image Size</th>
<th>Updated</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>BigDFT</td>
<td>hpc/bigdft</td>
<td>cuda10-ubun</td>
<td>2.37 GB</td>
<td>Oct 18, 2019</td>
<td>unlocked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tu1804-ompi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4-mkl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CANDLE</td>
<td>hpc/candle</td>
<td>20180326</td>
<td>1.52 GB</td>
<td>Oct 18, 2019</td>
<td>unlocked</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"Unlocked" permissions indicate images that do not require an API key to access.
To view detailed information about a specific image, specify the image and the tag.

Example:

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

3.4.2. 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.

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

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

3.4.3. Pushing a Container Image

With the NGC Registry CLI you can push (upload) images to your registry space.

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

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

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

3.4.4. Removing a Container Image

With the NGC Registry CLI you can removed images that are no longer needed from your registry space.

To remove all versions of an image, specify the image.

```bash
C:\>ngc registry image remove <image-name>
```

To remove a specific image version, specify the image and tag.

```bash
C:\>ngc registry image remove <image-name>:<tag>
```

3.5. Updating Container Metadata

You can find best practices on how to fill out the metadata for your container in the roduct Page Guidelines.

3.5.1. Updating Container Metadata via the NGC Website

Perform the following instructions to update the container metadata using the NGC website.
1. Click on the vertical ellipsis in the right upper corner of your container product page to reveal the entity action menu.

2. Select Edit Details from the entity action menu.

3. Update the container description and all other container metadata as needed.

4. To save your changes, click again on the vertical ellipsis to reveal the entity action menu and select Save.
3.5.2. Updating Container Metadata Using the NGC CLI

With the NGC Registry CLI you can update the container description and all the other container metadata.

With the NGC Registry CLI you can update the container description and all the other container metadata.

To update container metadata, use the following command.

```bash
```

Specify a named argument (field that will be updated, and values to update field) as well as a positional argument (name of the container image and, optionally, tag).

**Positional Arguments:**

```bash
<image>[:<tag>]
```

Name of the image repository or tagged image, `<image>[:<tag>]`

**Named Arguments**

```bash
--debug
```

Enable debug mode.
--format_type
Possible choices: ascii, csv, json. Specify the output format type. Supported formats are: 
['ascii', 'csv', 'json']. Only commands that produce tabular data support csv format. Default: ascii.

--org
Specify the organization name. Use "--org no-org" to override other sources and specify no org. Default: current configuration.

--ace
Specify the ACE name. Use "--ace no-ace" to override other sources and specify no ACE. Default: current configuration.

--team
Specify the team name. Use "--team no-team" to override other sources and specify no team. Default: current configuration.

--desc
Description for the target image.

--overview
Documentation (text or markdown file) for the image.

--label
A label to describe the repository. Can be used multiple times.

--logo
A URL pointing to the logo for the repository.

--publisher
The person or entity publishing the image.

--built-by
The person who built the container image.

Specify the image and, optionally, the tag.

Example: Changing description of a container image

To view the existing container metadata use the following command.

$ ngc registry image info nvidia/testcontainer

------------------------------
Image Repository Information
Name: testcontainer
Short Description: Test description.
Built By: Kristina
Publisher: NVIDIA
Logo: www.logo.com/logo.png
Labels: Machine Learning, Classification, Retail
Public: No
Last Updated: May 8, 2020
Latest Image Size: 60.27 MB
Update the container description with the following command.

```
$ ngc registry image update --desc "A test container image with useful tools." nvidia/testcontainer
```

To check if the update was successful, run the info command again

```
$ ngc registry image info nvidia/testcontainer
```

### Image Repository Information
- **Name:** testcontainer
- **Short Description:** A test container image with useful tools.
- **Built By:** Kristina
- **Publisher:** NVIDIA
- **Logo:** www.logo.com/logo.png
- **Labels:** Machine Learning, Classification, Retail
- **Public:** No
- **Last Updated:** May 8, 2020
- **Latest Image Size:** 60.27 MB
- **Latest Tag:** 3.0

**Tags:**
- 3.0
- 2.0
- 1.0

### 3.6. Multi-architecture Support for NGC Container Images

When running an image, docker will automatically select an image variant which matches your OS and architecture.

NGC Container Registry now allows users to leverage docker multi-architecture. It can support multiple architectures, which means that a single image may contain variants for different architectures like ARM, x86, Power and others; and sometimes for different operating systems, such as Windows.

**Manifest Lists and Tags**

NGC Container Registry now supports the manifest list schema now application/vnd.docker.distribution.manifest.list.v2+json providing the ability to assign multiple tags per image. For inspection of manifest list read instructions [here](#).

NGC UI allows you to navigate through the supported architecture.
## Docker Containers

**NGC Private Registry**

### Manifest

<table>
<thead>
<tr>
<th>SHA256</th>
<th>OSNAME</th>
<th>COMPRESSED SIZE</th>
<th>CREATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Chapter 4. NGC Models

The NGC private registry lets you upload and access deep-learning models.

4.1. Creating New NGC Models Using the NGC CLI

Issue `ngc registry model create -h` to see a description of available options and command descriptions for creating as model.

This example creates a new model called “Final Review Model” with all required and optional arguments used:

```bash
$ ngc registry model create \
   --application OBJECT_DETECTION \
   --format "cpkt" \
   --framework TensorFlow\n   --precision FP16 \n   --short-desc "A model for object detection using TensorFlow" \n   --built-by "My Name" \n   --display-name "Final Review Model" \n   --label "fast" --label "sparkly" \n   --owner-name "MyTeam" \n   --overview-filename /path/to/my/overview/file.md \n   --publisher "NVIDIA MyTeam" \n   --public-dataset-license <license> \n   --public-dataset-link "www.example.com" \n   --public-dataset-name "200_10x200_images" \n   nvidia/myteam/final_review_model
```

4.2. Creating a New Model Using the NGC Website

To create a new model asset, choose Create under the Models section of the Private Registry tab in the left hand menu.
The Model Creation screen walks you through the process of creating a new model asset.

Once this form has been completed and submitted you will have the option of creating a model version. This step can be skipped and completed later. See “Uploading A New Version” for more information.
<table>
<thead>
<tr>
<th>Field</th>
<th>Validation</th>
<th>Description</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>The name of the model</td>
<td>-</td>
</tr>
<tr>
<td>Owner</td>
<td>String</td>
<td>The name of the individual who owns the asset</td>
<td>-</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>Short description of the model</td>
<td>-</td>
</tr>
<tr>
<td>Overview</td>
<td>Markdown</td>
<td>A place to share more details/usage instructions</td>
<td>-</td>
</tr>
<tr>
<td>Labels</td>
<td>String (List)</td>
<td>Tags to make the asset more discoverable</td>
<td>-</td>
</tr>
<tr>
<td>Application</td>
<td>String</td>
<td>Intended use case</td>
<td>Annotation, Automatic Speech Recognition, Image Classification, Image Segmentation, Image Synthesis, Natural Language Processing, Object Detection, Translation</td>
</tr>
<tr>
<td>Training Framework</td>
<td>String</td>
<td>Deep learning framework used to build the model</td>
<td>Caffe, Clara, NeMo/PyTorch, PyTorch, TensorFlow, TensorRT, Transfer Learning Toolkit</td>
</tr>
<tr>
<td>Model Format</td>
<td>String</td>
<td>Output format of the weights file</td>
<td>caffemodel, HDF5, ONNX, protobuf, PyTorch PTH, SavedModel, TensorFlow CKPT, TensorRT Plan, TLT</td>
</tr>
<tr>
<td>Precision</td>
<td>String</td>
<td>Training precision used</td>
<td>AMP, FP16, FP32, INT8</td>
</tr>
</tbody>
</table>

**4.3. Uploading a New NGC Model Version Using the NGC CLI**

Issue `ngc registry model upload-version -h` to see a description of available options and command descriptions for uploading a model version.

An example using all required and optional arguments to create model version 1 for the model created in the previous section.

```
$ ngc registry model upload-version \
  --accuracy-reached 96.5 \
```
Adding Custom Metrics

You can also upload custom metrics tables for each model version. Each table can hold up to twelve key-value attribute pairs. Three tables maximum per model version.

Metrics tables are defined as JSON tables - one table per file. You can add the table to the upload with \texttt{--metrics-file}.

Some example metrics files:

\texttt{zeppelin_table.json}

```
{
  "name": "ZeppelinTable",
  "attributes": [ 
    {"key": "Robert", "value": "Plant"},
    {"key": "Jimmy", "value": "Page"},
    {"key": "John", "value": "Bonham"},
    {"key": "John", "value": "Paul Jones"}
  ]
}
```

\texttt{rhcp_table.json}

```
{
  "name": "RHCPTable",
  "attributes": [ 
    {"key": "Anthony", "value": "Keidis"},
    {"key": "Michael", "value": "Balzary"},
    {"key": "John", "value": "Frusciante"},
    {"key": "Chad", "value": "Smith"}
  ]
}
```

The above example with a custom metrics tables included:

```
$ ngc registry model upload-version \
  --accuracy-reached 95.5 \
  --batch-size 2000 \
  --gpu-model "SomeGPUModel" \
  --memory-footprint 4GB \
  --num-epochs 100 \
  --desc "A new and exciting version: 1" \
  --link "www.example.com/model/v1" \
  --link-type Other \
  --owner-name "My Name" \
  --metrics-file zeppelin_table.json \
  --metrics-file rhcp_table.json \
  --source path/to/my/model/version/dir \
  nvidia/myteam/final_review_model:1
```
4.4. Uploading an NGC Model Version Using the NGC Website

There are two ways to upload a new version of a model via the NGC Website.

- Via the Model Creation page discussed above
- From the Model Details page for any model

From the version creation page, shown below, you can specify all the relevant information about the specific version that you are uploading. You can also upload files directly from your browser.

![Version creation page](image-url)
Once you’ve completed the form, and uploaded any relevant files, submitting will publish the new version of the content.

**Adding Custom Metrics**

As deep learning models evolve we’re aware that you might also want to convey different information to distinguish between different versions. Using the NGC Model Registry, you can specify up to 36 different metrics to help people find the right versions.

When creating your version, simply “add custom metrics” to create the tables.

- **Model Name** - String - The name of the model you wish to upload a version
- **Owner** - String - The name of the individual who owns the asset (dropdown)
- **Version** - String - A way of identifying that version (we recommend SemVer)
- **Overview** - Markdown - A place to share more details/usage instructions for the model (shared across all versions)
- **Number of Epochs** - String - Number of Epochs trained (or N/A)
- **Batch Size** - String - Training Batch Size (or N/A)
- **GPU Model** - Drop Down - GPU family used for training
- **Accuracy Reached** - String - Accuracy of the model (or N/A)
- **Memory Footprint** - String - Memory Footprint used by the model
- **Related Resources** - You can optionally specify additional resources for your model
  - **Link Text** - Drop Down - The text to display for additional resources, such as containers or code samples, to accompany your version.
  - **URL** - String - The URL of the additional resource

Once you’ve entered the key/value pairs, select “add table”. 
4.5. Editing NGC Model Information Using the NGC CLI

Issue `ngc registry model update -h` to see a description of available options and command descriptions for editing a model or model version.

An example updating a model’s overview file for a model.

```bash
$ ngc registry model update \
   --overview-filename "path/to/my/updated/overview/file.md" \
   nvidia/myteam/final_review_model
```

An example updating a model-version’s accuracy reached and memory footprint.

```bash
$ ngc registry model update \
   --accuracy-reached 96.5 \ 
   --memory-footprint 16GB \ 
   nvidia/myteam/final_review_model:1
```

- `ngc registry model info nvidia/model-name` show information about a model
- `ngc registry model info nvidia/model-name:version` show information about a model version
- `ngc registry model list` list available models
- `ngc registry model download-version nvidia/model-name:version` download the specified model-version
- `ngc registry model remove nvidia/model-name:version` remove a model-version
- `ngc registry model remove nivida/model-name` remove a model

4.6. Editing NGC Model Information Using the NGC Website

To edit a Model’s metadata or overview tab, simply select “edit” from the top right of the model details page.
You can then edit any of the model's details, or even delete the model if you wish.
Chapter 5. NGC Resources

The NGC private registry lets you upload and access resources for deep-learning models.

5.1. Before You Begin

With the NGC Registry CLI you can update the container description and all the other container metadata.

Be sure you know your context, or which org and team you are logged into. This determines which registry space your model will be uploaded. You can do this by entering the following:

```bash
$ ngc config current
```

If you intend to upload a model to a different registry space, or if no team is reported and you intend to upload to a team space, then you can either

- Use `ngc config set` to switch to another org or team.
  ```bash
  $ ngc config set [--org <new org>][--team <new team>]
  ```
- or
  ```bash
  $ ngc config set [--org <new org>][--team <new team>]
  ```

5.2. Uploading a Resource

The following is the general process for uploading a resource to the model script registry.

1. Create a resource in the registry.
   This is a placeholder for your model and contains metadata about the resource.
   **Syntax**
   ```bash
   Example of creating resource "cmr_gnmt".
   $ ngc registry resource create nvidia/cmrgnmt
   ```
   To see a complete list of required and optional arguments, enter the following.
   ```bash
   $ ngc registry resource create -h
   ```

2. Upload your resrouce files.
Each time you upload files to the same resource, the upload becomes a unique version of the resource. You can specify the version when you upload, or let the CLI increment the version automatically.

**Example:** Uploading version 1 of the resource `cmr_gnmt` (required arguments omitted for simplicity).

```
$ ngc registry resource upload-version nvidia/cmrgnmt:first-upload [--source .<directory or file path for the model contents>]
```

```
Uploaded local path: C:\resource
Total files uploaded: 26
Total uploaded size: 134.48 KB
Started at: 2019-03-15 17:18:09.083000
Duration taken: 12s seconds
```

### 5.3. Updating a Resource

You can update or revise information for a resource or resource version.

The following is the basic command.

```
$ ngc registry resource update <org> /[<team>]/<resource-name>[[:version]]>
```

To update information, use the optional arguments to specify the information to change. To see the list of arguments, run

```
$ ngc registry resource update -h
```

### 5.4. Resource Commands

The full list of optional commands for NGC resources can be seen here.

- `--accuracy-reached <accuracy>`
  - Accuracy reached with target version.

- `--ace <name>`
  - Specify the ACE name. Use `--ace no-ace` to override other sources and specify no ACE.
  - Default: current configuration

- `--advanced-filename <path>`
  - Advanced guide. Provide the path to a file that contains the “Advanced Guide” for the resource.

- `--application <app>`
  - Target model application. Allowed values: CLASSIFICATION, OBJECT_DETECTION, SEGMENTATION, TRANSLATION, TEXT_TO_SPEECH, RECOMMENDER, SENTIMENT, NLP, KUBEFLOW_PIPELINE, OTHER.
--batch-size <size>
  The batch size of the target version.

--built-by <name>
  Builder of the target model.

--debug
  Enable debug mode.

--desc <desc>
  Full description of target version.

--display-name <name>
  Display name.

--format <fmt>
  Format of the target model.

--format_type <fmt>
  Specify the output format type. Supported formats are: ascii, csv, json. Only commands that
  produce tabular data support csv format. Default: ascii

--framework <fwk>
  Framework used to train the target model. Allowed values: TensorFlow, Caffe2, CNTK,
  Torch, PyTorch, MXNet, Keras, Other.

--gpu-model <model>
  The GPU used to train the target version.

--label <label>
  Label for the resource. To specify more than one label, use multiple --label arguments.

--logo <url>
  URL for the resource logo image.

--memory-footprint <footprint>
  The memory footprint of the target version.

--num-epochs <num>
  The number of epochs for the target version.

--org <name>
  Specify the organization name. Use "--org no-org" to override other sources and specify no
  org. Default: current configuration

--overview-filename <path>
  Overview. Provide the path to a file that contains the overview for the resource.
--performance-filename <path>
Performance data. Provide the path to a file that contains the performance data for the resource.

--precision <prec>
Precision the target model was trained with. Allowed Values: FP16, FP32, INT8, FPBOTH, OTHER.

--public-dataset-license <lcs>
License for public dataset used in the target model.

--public-dataset-link <url>
Link to public dataset used in the target model.

--public-dataset-name <name>
Name of public dataset used in the target model.

--publisher <name>
Publisher of the target model.

--quick-start-guide-filename <path>
Quick start information. Provide the path to a file that contains the “Quick Start Guide” information for the resource.

--release-notes-filename <path>
Release notes. Provide the path to a file that contains the release notes for the resource.

--setup-filename <path>
Setup instructions. Provide the path to a file that contains the setup instructions for the resource.

--short-desc <desc>
Short description.

--team <name>
Specify the team name. Use “--team no-team” to override other sources and specify no team. Default: current configuration

5.5. Deleting a Resource

Only admins and creators of the model can delete a model.

Be sure the context is set appropriately for the resource you want to delete. For example, if you want to delete a model that you created in the team_A space, then be sure to set the context as --team team_A.

To remove the resource, including all versions of the resource, enter the following.
$ ngc registry resource remove <org>/{<team>}/<resource>

To remove only a specific version of the resource, enter the following.

$ ngc registry resource remove <org>/{<team>}/<resource:version>
Chapter 6.  NGC Helm Charts

This document describes how to use the NGC registry to manage Helm charts.

6.1.  Introduction to NGC and Helm Charts

Helm is an application package manager running on top of Kubernetes. It lets you create Helm charts where you can define, install, and upgrade Kubernetes applications.

This document describes how to share Helm charts with others in your org or team using the NGC registry.

Prerequisites

These instructions assume the following prerequisites are met.

- Helm v 2.x or 3.x installed
  This is only required if you are creating or packaging Helm charts yourself. It is not needed otherwise.
- NGC organization account
  See the section Getting Started for instructions.

Note: The asset ngdocstest referenced below was created for example purposes only. It is intended merely as a guide and is not a requirement for publishing Helm assets to NGC.

6.2.  Creating and Packaging a Helm Chart

This section describes how to package a Helm chart for publishing to NGC.

There is no need to “deploy” the Helm chart in order to publish your chart to NGC. A .tgz file of the chart can be published to an org in NGC without being deployed first to the GPU infrastructure.

1. Create a Helm chart template by issuing the following.

```bash
$ helm create <chart-name>
```
Where `<chart-name>` is the name of your choosing.

**Example:**

```
$ helm create ngcdocstest
```

2. Modify the contents of the template with your Helm chart data.

3. Package the Helm chart by issuing the following.

```
$ helm package <chart-name>
```

**Example:**

```
$ helm package ngcdocstest
```

This example creates the tar package `ngcdocstest-0.1.0.tgz`.

### 6.3. Manage Helm Charts Using the NGC Web UI

#### 6.3.1. Viewing the List of Helm Charts and Getting 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.

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.
4. Select one of the Helm chart cards.
The page for each Helm chart provides information about the chart.
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.
6.3.2. Adding Helm Charts Using the NGC Web UI

**Note:** Make sure you have the right permissions to create Helm Charts in your organization and/or team. You need to have the user role “Registry User” or “Registry Admin”. For details refer to NGC Registry User Roles.

Before a chart can be uploaded to your organization’s registry, you must first create a record containing the basic information about the chart.

1. Click **Entity Creation Hub** under Private Registry section of the left side menu.

2. Click **Create Helm Chart**.

3. Fill in information about your Helm Chart.
4. Click "Create Helm Chart".
5. To push [upload] a Helm chart to your org space, use the NGC CLI.

**Example:**

```
$ ngc registry chart push nvidian/ngcdocstest:0.1.0
```

See [Pushing a Helm Chart](#) for details.

### 6.3.3. Updating the Helm Chart Page From the Website

1. To update the fields in the NGC Helm Chart page for a specific Helm chart, click **Edit Details**.
2. Edit each field as needed, then click **Save**.

6.3.4. **Removing Helm Charts from the Web UI**
Note: Make sure you have the right permissions to create Helm Charts in your organization and/or team. You need to have the user role "Registry Admin". For details refer to [NGC Registry User Roles](#).

1. To delete a Helm chart, click **Edit Details** from the details page of the Helm chart to delete.

2. Click **Delete** to remove the Helm chart.
3. Click **Delete** at the confirmation dialog.

6.4. Manage Helm Charts Using the NGC CLI

6.4.1. Searching for Available Helm Charts in an Org

The NGC CLI supports wildcard searches, using standard Unix shell-style wildcards. For example, to see a list of all available Chart packages in your org, run the following command.

```
$ ngc registry chart list *<org_name>*
```

**Example:**

```
$ ngc registry chart list *nvidian*
```
That will return all charts with ‘nvidian’ anywhere in the name.

<table>
<thead>
<tr>
<th>Name</th>
<th>Repository</th>
<th>Version</th>
<th>Size</th>
<th>Created By</th>
<th>Description</th>
<th>Created Date</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>fluentd-elasti</td>
<td>nvidian/fluen</td>
<td>4.8.1</td>
<td>245.61 KB</td>
<td>stg-3emmf14t83v</td>
<td>Changed sho</td>
<td>Nov 15, 2019</td>
<td>Dec 17, 2019</td>
</tr>
<tr>
<td>csearch</td>
<td>ntd-elasticse</td>
<td></td>
<td></td>
<td>d0s5v81gasfi479</td>
<td>rt descr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>arch</td>
<td></td>
<td></td>
<td></td>
<td>ion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clara</td>
<td>nvidian/repo1</td>
<td>0.0.1</td>
<td>65.66 KB</td>
<td>stg-p6urlvepnjb</td>
<td>Feb 07, 2020</td>
<td>Mar 12, 2021</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/clara</td>
<td></td>
<td></td>
<td>q06qfis28l5m6a4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.4.2. Fetching Helm Charts

To download (or “pull”) a Chart package, run the following command. **Note**: if no version is specified, the most recent version will be pulled.

$ ngc registry chart pull org/[team/]chart[:version]

Example:

$ ngc registry chart pull nvidian/nginx-ingress:1.2.3 (pulls version 1.2.3)

$ ngc registry chart pull nvidian/nginx-ingress (pulls the latest version)

6.4.3. Adding Helm Charts to a Private Registry

**Note**: Make sure you have the right permissions to create Helm Charts in your organization and/or team. You need to have the user role “Registry User” or “Registry Admin”. For details, refer to **NGC Registry User Roles**.

Creating a Chart

Before a chart can be uploaded to your organization’s registry, you must first create a record containing the basic information about the chart. There are several values you can specify (issue `ngc registry chart create --help` to view all of them), but you must at least provide a short description of the chart.

$ ngc registry chart create <org>/[<team>/]<chart_name> --short-desc <description>

Example:

$ ngc registry chart create nvidian/ngcdocstest --short-desc "Doc testing chart"

Successfully created chart 'nvidian/ngcdocstest'.

---

Chart Information
Name: ngcdocstest
Short Description: Doc testing chart
Display Name:
Team:
Publisher:
Built By:
Labels:
Updating a Chart

You can update the metadata about a chart after it has been created with the update command.

$ ngc registry chart update <org>/<team>/<chart_name> --<property> <value>

**Example:**

$ ngc registry chart update nvidia/ngcdocstest --publisher "test account" --display-name "Helm Demo Chart" --built-by "my team"

Successfully updated chart 'nvidia/ngcdocstest'.

---

6.4.4. Getting Information about a Helm Chart

You can see the information about a chart at any time by running the *info* command:

**Example:**

$ ngc registry chart info nvidia/ngcdocstest

---

6.4.5. Pushing a Helm Chart
To push (upload) a Helm chart to your org space, issue the following.

```bash
$ ngc registry chart push <org>/[<team>/]<chart_name>:<version>
```

**Example:**

```
$ ngc registry chart push nvidian/ngcdocstest:0.1.0
```

Successfully pushed chart version 'ngcdocstest:0.1.0'.

---

**Chart Version Information**

- **Created Date:** 2021-03-22 18:54:44 UTC
- **Updated Date:** 2021-03-22 18:54:44 UTC
- **Version ID:** 0.1.0
- **Total File Count:** 11
- **Total Size:** 10.41 KB
- **Status:** UPLOAD_COMPLETE

---

### 6.4.6. Listing Helm Chart Versions

To see a list of all available versions for a chart, specify the chart name, and use the wildcard `'*'` for the version.

**Example:**

```
$ ngc registry chart list nvidian/nginx-ingress:*
```

```
+---------+------------+-----------+--------------+
<table>
<thead>
<tr>
<th>Version</th>
<th>File Count</th>
<th>File Size</th>
<th>Created Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8.0</td>
<td>27</td>
<td>181.94 KB</td>
<td>Mar 12, 2021</td>
</tr>
<tr>
<td>1.0.0</td>
<td>25</td>
<td>149.51 KB</td>
<td>Oct 02, 2020</td>
</tr>
<tr>
<td>0.0.6</td>
<td>25</td>
<td>149.51 KB</td>
<td>Oct 02, 2020</td>
</tr>
<tr>
<td>0.0.5</td>
<td>25</td>
<td>149.51 KB</td>
<td>Oct 02, 2020</td>
</tr>
<tr>
<td>0.6.0</td>
<td>25</td>
<td>149.51 KB</td>
<td>Sep 17, 2020</td>
</tr>
<tr>
<td>0.6.1</td>
<td>25</td>
<td>149.51 KB</td>
<td>Sep 17, 2020</td>
</tr>
<tr>
<td>1.26.2</td>
<td>68</td>
<td>109.19 KB</td>
<td>Feb 08, 2020</td>
</tr>
</tbody>
</table>
+---------+------------+-----------+--------------+
```

### 6.4.7. Removing Helm Charts from a Private Registry

**Note:** Make sure you have the right permissions to create Helm Charts in your organization and/or team. You need to have the user role "Registry Admin". For details, refer to [NGC Registry User Roles](#).

If you are an admin, you can delete a specific version of a chart running the following command:

```bash
$ ngc registry chart remove <org>/[<team>/]<chart_name>:<version>
```

The following example removes just version 0.1.0:

```
$ ngc registry chart remove nvidian/ngcdocstest:0.1.0
```

The following example removes all versions and data about the chart:

```
$ ngc registry chart remove nvidian/ngcdocstest
```

If you do not specify a version, every version of the chart, as well as the chart metadata, will be deleted.
Example:

```sh
$ ngc registry chart remove nvidia/ngcdocstest
Are you sure you would like to remove nvidia/ngcdocstest? [y/n]y
Successfully removed chart version 'nvidia/ngcdocstest:0.1.0'.
Successfully removed chart 'nvidia/ngcdocstest'.
```

6.5. **Manage Helm Charts Using the NGC API**

6.5.1. **Updating Information on the Helm Chart Page**

The NGC API lets you specify information about your Helm chart. Use the NGC API Explorer page (URL: https://docs.ngc.nvidia.com/models/index.html#!/Artifacts/updateArtifactInOrgUsingPATCH). You can use the page to build the JSON file for use in a CURL command.

The following page elements can be edited.

<table>
<thead>
<tr>
<th>Page Element</th>
<th>JSON Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helm Chart name</td>
<td>displayName</td>
<td>The name of the Helm chart appearing in the title on the tile and Helm chart page</td>
</tr>
<tr>
<td>Publisher</td>
<td>publisher</td>
<td>The organization/entity responsible for creating the asset</td>
</tr>
<tr>
<td>Logo</td>
<td>logo</td>
<td>URL of the image to use as the logo for the asset</td>
</tr>
<tr>
<td>Description</td>
<td>shortDescription</td>
<td>A short description for the Helm chart</td>
</tr>
<tr>
<td>Labels</td>
<td>labels</td>
<td>Tags to enhance search results</td>
</tr>
<tr>
<td>Overview tab</td>
<td>description</td>
<td>Content of the &quot;Overview&quot; tab which can provide publishers to convey additional</td>
</tr>
</tbody>
</table>

The JSON column shows the corresponding JSON fields to use when updating the page using the NGC API.

The following shows the relevant fields in the JSON file.

```json
{
  "attributes": [
    {
      "key": "string",
      "value": "string"
    }
  ],
  "builtBy": "string",
  "description": "string",
  "displayName": "string"
}
```
Example

The following shows example JSON values.

```json
{
  "builtBy": "NVIDIA",
  "description": "#NGC Docs Chart",
  "displayName": "NGC DOCS CHART TEST",
  "labels": [
    "Helm Chart",
    "Documentation"
  ],
  "shortDescription": "This chart is for the docs!"
}
```

The following is an example CURL command.

```
curl -X PATCH --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: Bearer <<BEARER_TOKEN>>' -d '{ "builtBy": "NVIDIA", "description": "#Le Chart", "displayName": "NGC DOCS TEST", "labels": [ "Helm Chart", "Documentation" ], "shortDescription": "This chart is for the docs&dquo; }' https://api.ngc.nvidia.com/v2/org/nvidian/helm-charts/ngcdocstest
```

6.5.2. Deleting Helm Charts Using the NGC API

To remove Helm charts from your org or team, you must use the NGC API. Refer to [https://docs.ngc.nvidia.com/api/index.html#!/Model/proxyDeleteUsingDELETE](https://docs.ngc.nvidia.com/api/index.html#!/Model/proxyDeleteUsingDELETE) for a description of the relevant API.

To delete a Helm chart from an org space, issue the following:

```
```

To delete a Helm chart from a Team space, issue the following:

```
```

6.6. Manage Helm Charts Using the Helm CLI

6.6.1. 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.
6.6.2. Searching for Available Helm Charts

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

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

6.6.3. Fetching Helm Charts

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

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

6.6.4. Adding Helm Charts to a Private NGC Org/Team

These instructions assume the Helm push plugin is installed. To install the plugin, issue the following.

```bash
$ helm plugin install https://github.com/chartmuseum/helm-push
```

To push (upload) a Helm chart to your org space, issue the following.

```bash
$ helm push <chart-name>.tgz <repo-name>
```

6.6.5. Removing Helm Charts from a Private NGC Org/Team

To remove Helm charts from your org or team, you must use the NGC CLI or NGC API.
Chapter 7. Getting Support for NGC container registry

For additional information on using the NGC container registry and for getting help if you encounter issues, send an email to enterprisesupport@nvidia.com with a description of your issue and a ticket will be created for you.