

# NVIDIA 驱动和基础软件环境安装指导手册

## 修订记录

Date	Version	Authors	Description
2021.12.01	V1	Leon Wang	基础环境

## 1. 概述

本文档的目标，是指导NPN用户，在Bare Metal环境，安装和配置GPU工作的基础环境。主要包含GPU驱动安装和上层Docker，NGC镜像等软件环境安装等内容。VM环境中，比如Esxi或者KVM构建的VM+vGPU等场景，暂时不在本文档讨论范围。

## 2. 安装部署

### 2.1 NVIDIA GPU驱动安装

#### 2.1.1 基于CLI安装

首先,查看服务器上的是否安装了NVIDIA GPU

```
lspci | grep -i nvidia
```

DGX A100输出内容示例

```
07:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
0f:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
47:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
4e:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
87:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
90:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
b7:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
bd:00.0 3D controller: NVIDIA Corporation Device 20b0 (rev a1)
```

前往 NVIDIA 驱动[程序下载](#)页面,为您的 GPU 选择合适的驱动和操作系统,并将最新的驱动程序存储到例如 ~/Downloads 的位置。

此处以Ubuntu系统为例,首先,运行以下命令准备安装:

```
$ sudo apt install build-essential libglvnd-dev pkg-config
```

运行以下命令禁用当前驱动程序:

```
sudo telinit 3
```

进入控制台(CLI界面)并运行以下命令以删除以前的驱动程序,之后安装新下载的驱动程序(将下面的xx.run 文件的名称替换为之前下载过的文件名):

```
$ sudo apt purge nvidia-*  
$ cd ~/Downloads  
$ sudo chmod +x NVIDIA-Linux-x86_64-xxx.xx.run  
$ sudo ./NVIDIA-Linux-x86_64-xxx.xx.run
```

在安装过程中选择以下选项:

```
The distribution-provided pre-install script failed! Are you sure you  
want to continue? -> CONTINUE INSTALLATION  
would you like to run the nvidia-xconfig utility? -> YES
```

**注意:**如果遇到错误,下面提供了另一种安装 NVIDIA 驱动程序的方法(可选)

使用 PPA 存储库(对于最新的驱动程序,请单击[此处](#)并将下面的 nvidia-xxx 替换为您的首选驱动程序):

```
$ sudo apt purge nvidia-*  
$ sudo add-apt-repository ppa:graphics-drivers/ppa  
$ sudo apt update  
$ sudo apt install nvidia-driver-xxx
```

安装完成后,运行以下命令重启服务器:

```
$ sudo reboot
```

重启后,可以通过以下命令查看显卡驱动版本信息

```
cat /proc/driver/nvidia/version
```

输出内容示例:

```
NVRM version: NVIDIA UNIX x86_64 Kernel Module 470.57.02 Tue Jul 13 16:14:05  
UTC 2021  
GCC version: gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)
```

或者使用nvidia-smi查看

```
Wed Jun 30 11:23:19 2021
```

NVIDIA-SMI 460.32.03				Driver Version: 460.32.03		CUDA Version: 11.2	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr. ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	MIG M.
0	A100-SXM4-40GB	On	00000000:27:00.0	Off	0%	0	
N/A	30C	P0	50W / 400W	0MiB / 39538MiB		Default	Disabled
1	A100-SXM4-40GB	On	00000000:2A:00.0	Off	0%	0	
N/A	28C	P0	54W / 400W	0MiB / 39538MiB		Default	Disabled
2	A100-SXM4-40GB	On	00000000:51:00.0	Off	0%	0	
N/A	28C	P0	52W / 400W	0MiB / 39538MiB		Default	Disabled
3	A100-SXM4-40GB	On	00000000:57:00.0	Off	0%	0	
N/A	31C	P0	54W / 400W	0MiB / 39538MiB		Default	Disabled
4	A100-SXM4-40GB	On	00000000:9E:00.0	Off	0%	0	
N/A	31C	P0	53W / 400W	0MiB / 39538MiB		Default	Disabled
5	A100-SXM4-40GB	On	00000000:A4:00.0	Off	0%	0	
N/A	29C	P0	53W / 400W	0MiB / 39538MiB		Default	Disabled
6	A100-SXM4-40GB	On	00000000:C7:00.0	Off	0%	0	
N/A	28C	P0	51W / 400W	0MiB / 39538MiB		Default	Disabled
7	A100-SXM4-40GB	On	00000000:CA:00.0	Off	0%	0	
N/A	31C	P0	53W / 400W	0MiB / 39538MiB		Default	Disabled
Processes:							
GPU	GI	CI	PID	Type	Process name	GPU Memory	
ID	ID	ID				Usage	
No running processes found							

如果以上找不到nvidia-smi命令或者无法显示,请检查并尝试重新安装GPU驱动。

## 2.1.2 基于GUI安装

待完善。

更多详细内容, 请查阅[NVIDIA Driver Installation Quickstart Guide](#)。

## 2.2 Docker安装

Docker 是一个用于 Linux 的开源容器部署平台。 Docker 容器是一种将 Linux 应用程序与其所有库、数据文件和环境变量捆绑在一起的机制,以便在它运行的任何 Linux 系统上以及同一主机上的实例之间执行的环境始终相同。 要了解有关 Docker 的更多信息,请单击[此处](#)。

## 2.2.1 Docker的优点

(1) 快速,一致地交付您的应用程序

Docker 允许开发人员使用您提供的应用程序或服务的本地容器在标准化环境中工作,从而简化了开发的生命周期。

容器非常适合持续集成和持续交付(CI / CD)工作流程,请考虑以下示例方案:

- 您的开发人员在本地编写代码,并使用 Docker 容器与同事共享他们的工作。
- 他们使用 Docker 将其应用程序推送到测试环境中,并执行自动或手动测试。
- 当开发人员发现错误时,他们可以在开发环境中对其进行修复,然后将其重新部署到测试环境中,以进行测试和验证。
- 测试完成后,将修补程序推送给生产环境,就像将更新的镜像推送到生产环境一样简单。

(2) 响应式部署和扩展

Docker 是基于容器的平台,允许高度可移植的工作负载。Docker 容器可以在开发人员的本机上,数据中心的物理或虚拟机上,云服务上或混合环境中运行。

Docker 的可移植性和轻量级的特性,还可以使您轻松地完成动态管理的工作负担,并根据业务需求指示,实时扩展或拆除应用程序和服务。

(3) 在同一硬件上运行更多工作负载

Docker 轻巧快速。它为基于虚拟机管理程序的虚拟机提供了可行、经济、高效的替代方案,因此您可以利用更多的计算能力来实现业务目标。Docker 非常适合于高密度环境以及中小型部署,而您可以用更少的资源做更多的事情。

## 2.2.2 Ubuntu安装Docker

以下安装说明基于Ubuntu 20.04, Docker 在标准的 Ubuntu 20.04 软件源中可用,但是可能不是最新的版本。我们将会从 Docker 的官方软件源中安装最新的 Docker 软件包。

在 Ubuntu 20.04 上安装 Docker

启用 Docker 软件源,导入 GPG key,并且安装软件包。首先,更新软件包索引,并且安装必要的依赖软件,来添加一个新的 HTTPS 软件源:

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

使用下面的 curl 导入源仓库的 GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

将 Docker APT 软件源添加到你的系统:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

现在,Docker 软件源被启用了,你可以安装软件源中任何可用的 Docker 版本。

(1) 想要安装 Docker 最新版本,运行下面的命令。如果你想安装指定版本,跳过这个步骤,并且跳到下一步。

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io
```

(2) 想要安装指定版本,首先列出 Docker 软件源中所有可用的版本:

```
sudo apt update
apt list -a docker-ce
```

可用的 Docker 版本将会在第二列显示。

```
Listing... Done
docker-ce/focal 5:20.10.12~3-0~ubuntu-focal amd64 [upgradable from: 5:20.10.11~3-0~ubuntu-focal]
docker-ce/focal,now 5:20.10.11~3-0~ubuntu-focal amd64 [installed,upgradable to: 5:20.10.12~3-0~ubuntu-focal]
docker-ce/focal 5:20.10.10~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.9~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.8~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.7~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.6~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.5~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.4~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.3~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.2~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.1~3-0~ubuntu-focal amd64
docker-ce/focal 5:20.10.0~3-0~ubuntu-focal amd64
docker-ce/focal 5:19.03.15~3-0~ubuntu-focal amd64
docker-ce/focal 5:19.03.14~3-0~ubuntu-focal amd64
docker-ce/focal 5:19.03.13~3-0~ubuntu-focal amd64
docker-ce/focal 5:19.03.12~3-0~ubuntu-focal amd64
docker-ce/focal 5:19.03.11~3-0~ubuntu-focal amd64
docker-ce/focal 5:19.03.10~3-0~ubuntu-focal amd64
docker-ce/focal 5:19.03.9~3-0~ubuntu-focal amd64
```

通过在软件包名后面添加版本 =VERSION来安装指定版本:

```
sudo apt install docker-ce=<VERSION> docker-ce-cli=<VERSION> containerd.io
```

安装完成,Docker 服务将会自动启动。你可以输入下面的命令,验证它:

```
sudo systemctl status docker
```

输出将会类似下面这样:

```
docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-05-21 14:47:34 UTC; 42s ago
   ...
```

验证安装

想要验证 Docker 是否已经成功被安装,你可以执行docker命令,前面不需要加`sudo`,我们将会运行一个测试容器:

```
sudo docker container run hello-world
```

如果本地没有该镜像,这个命令将会下载测试镜像,在容器中运行它,打印出 “Hello from Docker”,并且退出。

输出看起来应该像这样:

```
tugberk@ubuntu:~$ sudo docker run ubuntu:14.04 /bin/echo 'Hello world'
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from ubuntu
83e4dde6b9cf: Pull complete
b670fb0c7ecd: Pull complete
29460ac93442: Pull complete
d2a0ecffe6fa: Already exists
ubuntu:14.04: The image you are pulling has been verified. Important: image verification is a tech preview feature and should not be relied on to provide security.
Digest: sha256:36de65e14e47b2a9a3d91c9a032899d4c432a00295bd77ced8d5bb6dc2764318
Status: Downloaded newer image for ubuntu:14.04
Hello world
tugberk@ubuntu:~$
```

## 2.2.3 Docker常见问题

### (1) Docker镜像存储的位置

使用sudo docker info命令,可以查看docker详细信息,其中Docker Root Dir项目, 记录了docker images默认存储的位置 /var/lib/docker

```
leonwang@nnp-team-a6k-01:~$ sudo docker info
Client:
Context:    default
Debug Mode: false
Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Docker Buildx (Docker Inc., v0.7.1-docker)
  scan: Docker Scan (Docker Inc., v0.12.0)

Server:
Containers: 1
  Running: 0
  Paused: 0
  Stopped: 1
Images: 1
Server Version: 20.10.12
Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
Logging Driver: json-file
Cgroup Driver: cgroupfs
Cgroup Version: 1
Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 7b11cfaabd73bb80907dd23182b9347b4245eb5d
runc version: v1.0.2-0-g52b36a2
init version: de40ad0
Security Options:
  apparmor
  seccomp
  Profile: default
Kernel Version: 5.11.0-27-generic
Operating System: Ubuntu 20.04.3 LTS
OSType: linux
Architecture: x86_64
CPUs: 24
Total Memory: 125.6GiB
Name: npn-team-a6k-01
ID: MCBS:2EYN:R24D:7UTE:G6YH:57YF:AA6P:HLH7:3H4R:7CBQ:4TZG:CHV5
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
```

### (2)修改Docker镜像存储的位置



创建一个新的文件夹作为docker镜像新的存储位置，记下这个文件夹的路径。

输入：

```
sudo vim /etc/systemd/system/multi-user.target.wants/docker.service
```

将下面这行内容

```
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

修改为

```
ExecStart=/usr/bin/dockerd --graph=your_new_docker_img_path --storage-driver=overlay
```

重启docker来更新配置

```
1 | sudo systemctl daemon-reload
2 | sudo systemctl restart docker
```

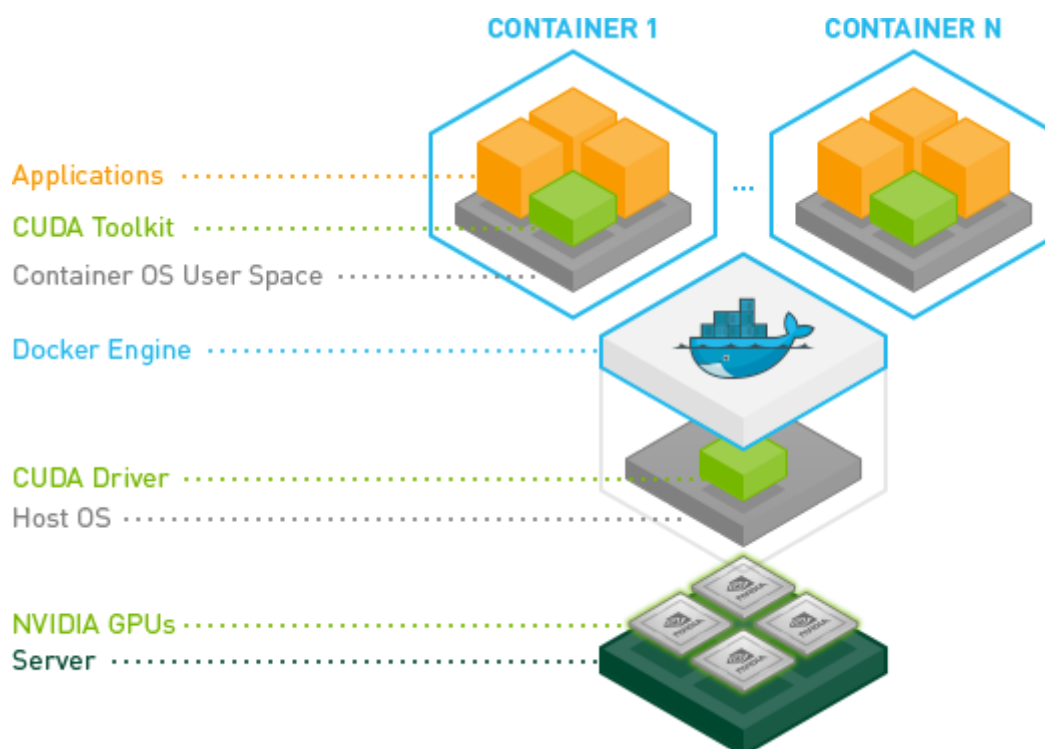
重启后，再次使用 docker info查看Docker Root Dir项目显示的位置。

## 2.3 NVIDIA Container Toolkit安装

### 2.3.1 简介

NVIDIA Container Toolkit 允许用户构建和运行 基于GPU 加速的容器。该工具包括一个容器运行时库和实用程序，用于自动配置容器以利用 NVIDIA GPU。NVIDIA Container Toolkit 支持生态系统中的不同容器引擎 - Docker、LXC、Podman 等。

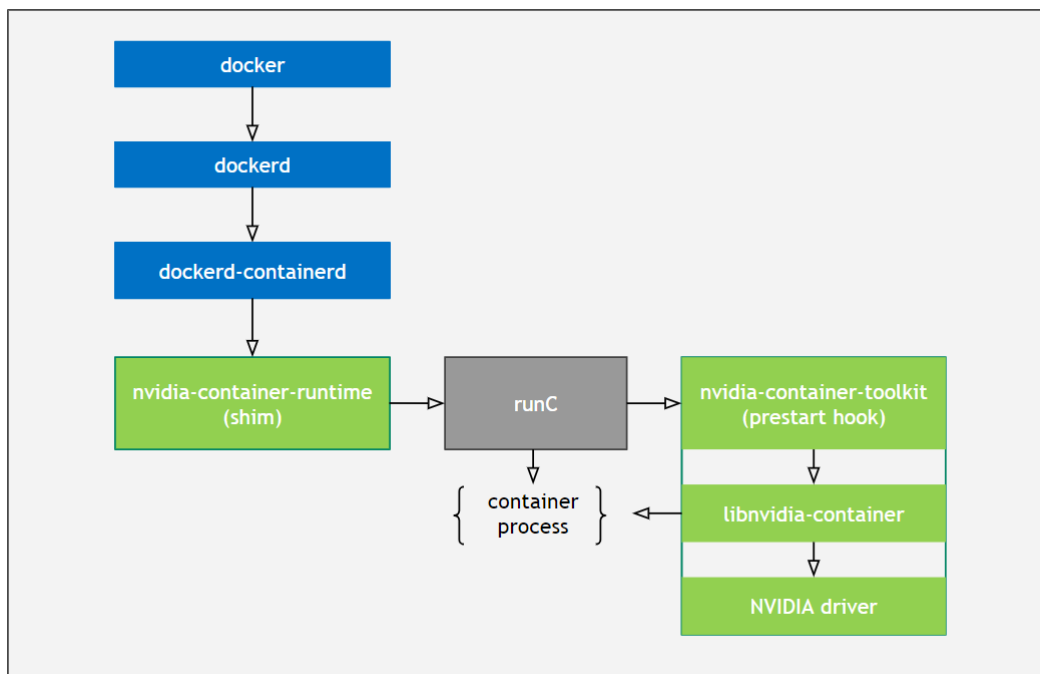
如果您需要在容器中使用NVIDIA GPU设备，那么NVIDIA Container Toolkit是必不可少的组件。它的主要作用是将NVIDIA GPU设备挂载到容器中。



支持docker的NVIDIA Container Toolkit由如下的组件构成：

- nvidia-docker2
- nvidia-container-runtime
- nvidia-container-toolkit
- libnvidia-container

以下架构详细介绍了各个组件的关系：



### nvidia-docker2介绍

nvidia-docker2只适用于docker，其主要作用是可以通过环境变量指定容器需要使用节点上哪些GPU。

### nvidia-container-runtime介绍

nvidia-container-runtime主要用于将容器runC spec作为输入，然后将nvidia-container-toolkit脚本作为一个prestart hook注入到runC spec中，将修改后的runC spec交给runC处理。

### nvidia-container-toolkit介绍

nvidia-container-toolkit是一个实现了runC prestart hook接口的脚本，该脚本在runC创建一个容器之后，启动该容器之前调用，其主要作用就是修改与容器相关联的`config.json`，注入一些在容器中使用NVIDIA GPU设备所需要的一些信息（比如：需要挂载哪些GPU设备到容器当中）。

### libnvidia-container介绍

libnvidia-container提供了一个库和简单的CLI工具，以实现在容器当中支持使用GPU设备的目标。

## 2.3.2 安装

NVIDIA Container Toolkit 可用于各种 Linux 发行版，并支持不同的容器引擎。

**Linux发行版本：**



OS Name / Version	Identifier	amd64 / x86_64	ppc64le	arm64 / aarch64
Amazon Linux 1	amzn1	X		
Amazon Linux 2	amzn2	X		X
Amazon Linux 2017.09	amzn2017.09	X		
Amazon Linux 2018.03	amzn2018.03	X		
Open Suse/SLES 15.0	sles15.0	X		
Open Suse/SLES 15.x	sles15.1	X		
Debian Linux 9	debian9	X		
Debian Linux 10	debian10	X		
Centos 7	centos7	X	X	
Centos 8	centos8	X	X	X
RHEL 7.x (*)	rhel7.x	X	X	
RHEL 8.x (*)	rhel8.x	X	X	X
Ubuntu 16.04	ubuntu16.04	X	X	
Ubuntu 18.04	ubuntu18.04	X	X	X
Ubuntu 20.04	ubuntu20.04	X	X	X

(\*) RHEL 7 和 RHEL 8 的次要版本 (即 7.4 -> 7.9 是到 centos 7 的符号链接, 而 8.0 -> 8.3 是到 centos8 的符号链接)

#### 容器引擎:

OS Name / Version	amd64 / x86_64	ppc64le	arm64 / aarch64
Docker 18.09	X	X	X
Docker 19.03	X	X	X
Docker 20.10	X	X	X
RHEL/CentOS 8 podman	X		
CentOS 8 Docker	X		
RHEL/CentOS 7 Docker	X		

#### 安装的先决条件

#### NVIDIA GPU驱动

请确保您已经为您的 Linux 发行版安装了 NVIDIA 驱动程序。

## 平台要求

运行 NVIDIA Container Toolkit 的先决条件如下所述：

1. GNU/Linux x86\_64 内核版本 > 3.10
2. Docker >= 19.03（推荐，但某些发行版可能包含较旧版本的 Docker。支持的最低版本为 1.12）
3. NVIDIA GPU架构 >= Kepler（例如NVIDIA Tesla K80）
4. NVIDIA Linux 驱动程序 >= 418.81.07（请注意，不支持较旧的驱动程序版本或分支。）

**注意：**在容器中使用GPU，不需要在宿主机上安装CUDA Toolkit。即不需要安装CUDA。CUDA版本与GPU驱动程序有对应关系，非特殊情况，建议使用最新版本的驱动。[CUDA release notes](#)包含了CUDA Toolkit版本与驱动版本的对应关系表。

## 配置 NVIDIA 容器工具包

设置稳定存储库和 GPG 密钥：

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

更新包列表后安装 nvidia-docker2 包（和依赖项）：

```
sudo apt-get update
```

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

重启Docker进程完成安装：

```
sudo systemctl restart docker
```

验证安装成功：

```
sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

如果NVIDIA Container Toolkit安装成功，会有如下的显示输出



NVIDIA GPU CLOUD (NGC)，这里的Cloud与传统的Cloud是两个概念，传统的Cloud一般指的是一些工作云平台。而这里的Cloud提供的是针对深度学习和科学计算优化的GPU加速云平台，实际上，它本质上提供的是一个容器仓库，首先在NGC的页面上提供注册，同时也会提供一个下载NVIDIA优化的容器库地址供大家使用这个平台。

NGC管理着一份目录，包含了完全集成和优化的深度学习框架容器，适用于单GPU以及多GPU配置环境。这些容器包括：CUDA 工具包，DIGITS工作流，以及以下深度学习框架：NVCaffe, Caffe2, Microsoft Cognitive Toolkit (CNTK), MXNet, PyTorch, TensorFlow, Theano 和 Torch。这些框架容器以开箱即可用的方式交付，包含了所有必须的依赖，比如CUDA运行时环境、NVIDIA库和运行系统环境。

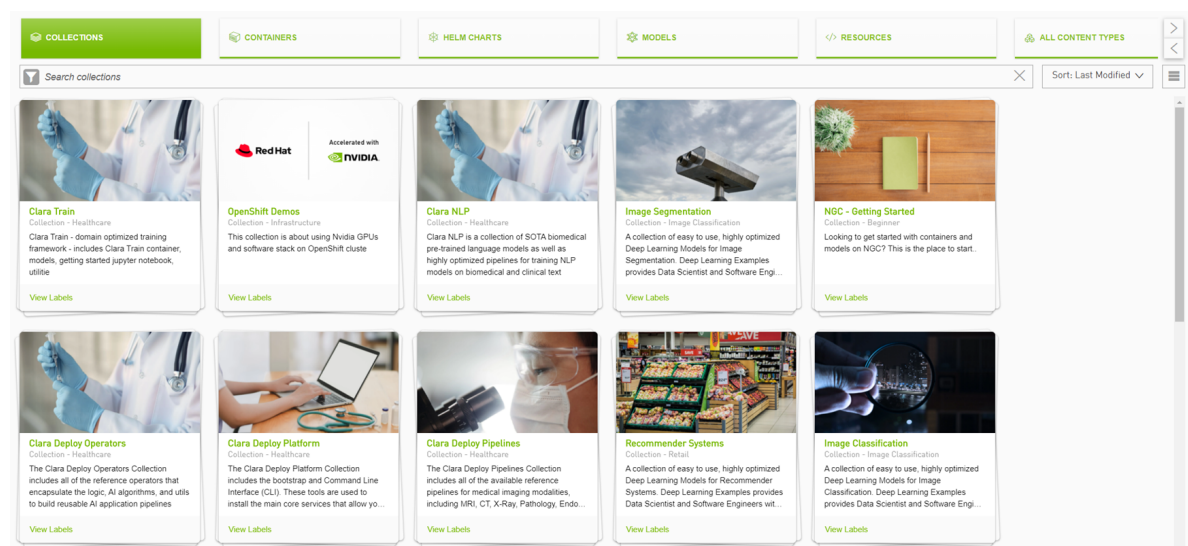
每个框架容器镜像还包含了框架源代码，以支持用户自定义修改和增强功能，以及完整的软件开发栈。

NVIDIA每月更新这些深度学习的容器，以确保提供最佳性能。

在深度学习框架容器的基础上，NGC也提供了一系列高性能计算可视化应用容器，采用支持业界领先的可视化工具，包括集成了NVIDIA Index 立体体渲染的ParaView, NVIDIA OptiX 光线追踪库和NVIDIA Holodeck，以实现高质量可交互的实时视觉效果。这些容器目前处于公测阶段。

NGC也提供流行常用的第三方兼容GPU、符合NGC标准和最佳实践的高性能计算应用容器，使用户可以方便的在最短的时间内启动和运行起来。

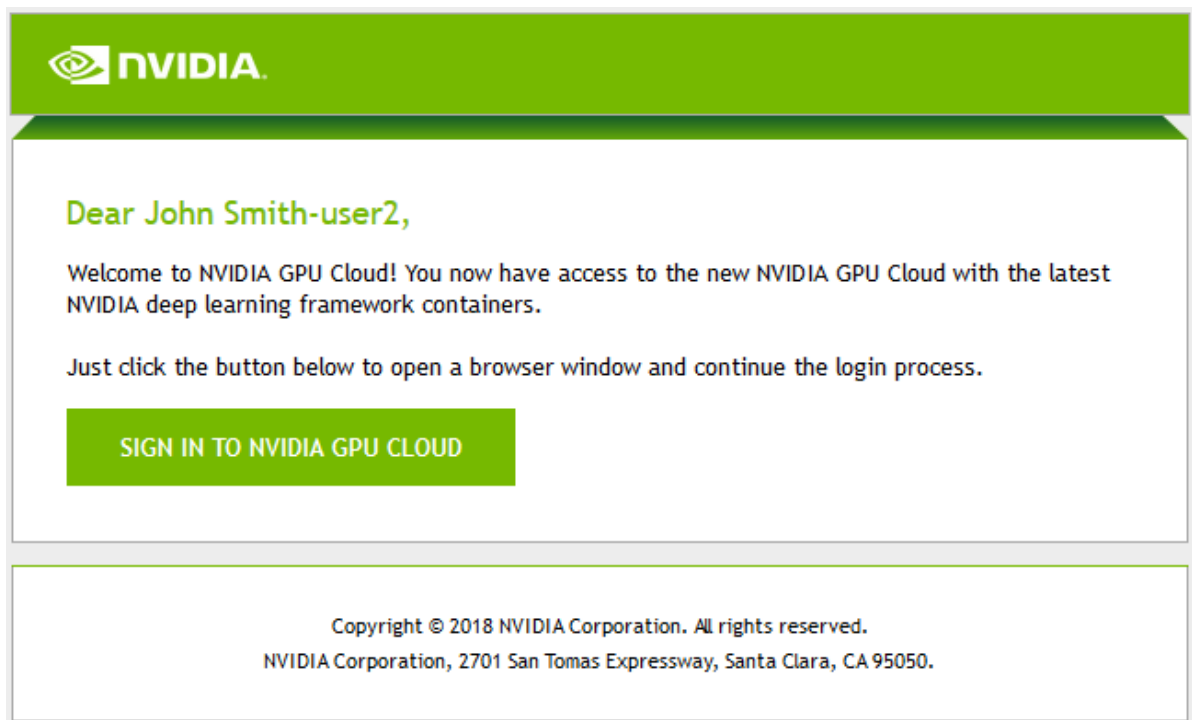
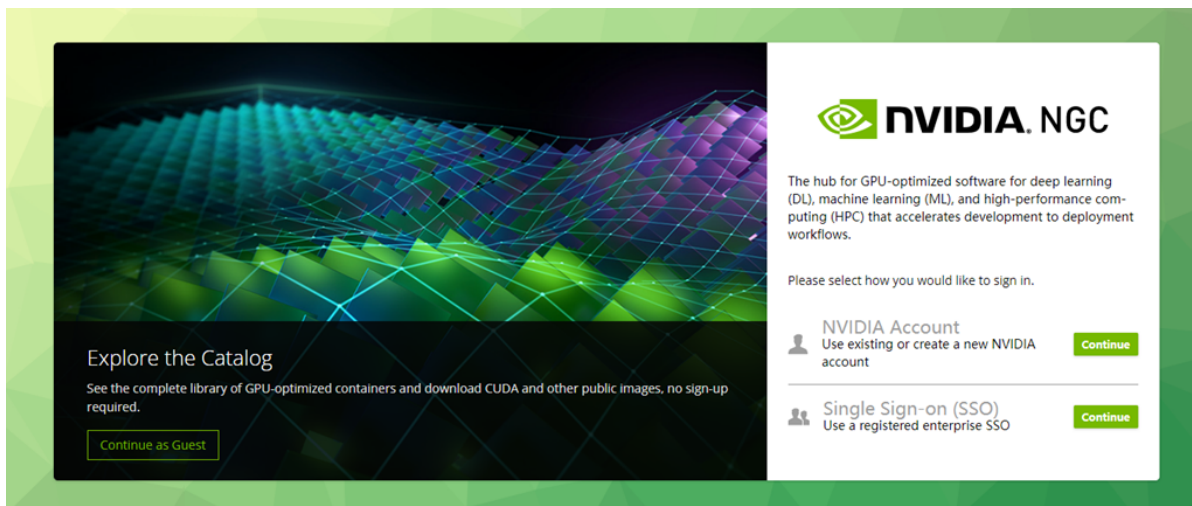
详细内容，请查阅[NGC Container User Guide](#)。



## 2.4.1注册NGC

任何用户都可以免费注册NGC，获得以上提到的镜像内容，以下是注册步骤：

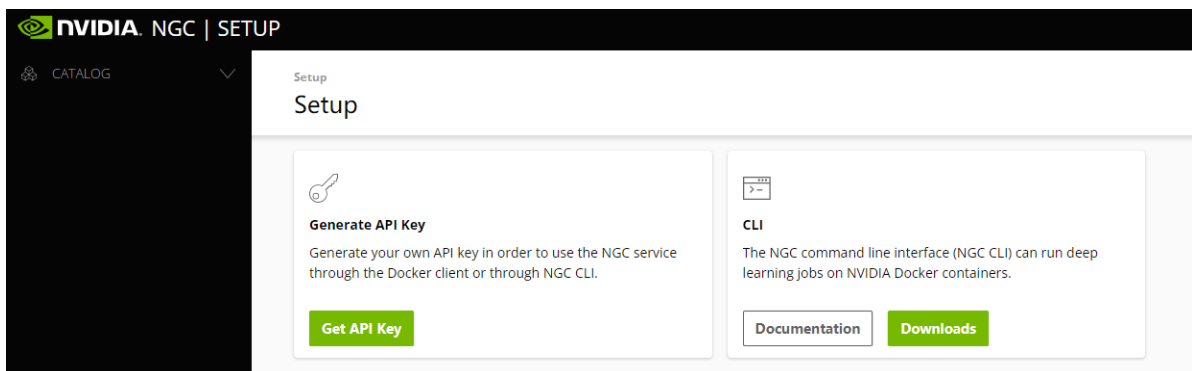
1. 登录[ngc.nvidia.com](https://ngc.nvidia.com)
2. 选择注册账户，编写您的电子邮箱等必要信息后，您会收到一份欢迎邮件，其中包含了如何设置您账户的说明。
3. 打开欢迎邮件里面的链接，在浏览器中打开最初的NGC创建密码页面。
4. 按照说明创建您的密码。
5. 登录您的账户：在通知您账户已被激活的欢迎界面里点击登录。



## 2.4.2 获取NGC API密钥

如果您需要从您的GPU服务器上拉取NGC镜像，则需要先在NGC页面，生成API密钥，然后在GPU服务器上使用该密钥注册登录。以下是获取API密钥的步骤：

1. 登录NGC页面后，点击右上角您的账户名称，在下拉菜单中选择'Setup'。进入Setup后，选择'Get API Key'。



2. 进入API Key页面后，单击'Generate API Key'，生成您的API密钥。系统会显示一条警告信息，告诉您如果您创建新的API密钥，旧的密钥将会失效。

## API

### API Information

Generate your own API key to use the NGC service through the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

### Usage

Use your API key to log in to the NGC registry by entering the following command and following the prompts:

#### NGC CLI

```
$ ngc config set
```

#### Docker™

For the username, enter 'soauthtoken' exactly as shown. It is a special authentication token for all users.

```
$ docker login nvcr.io
```

```
Username: soauthtoken
```

```
Password: <Your Key>
```

3. 单击确认以生成密钥。您的API密钥会弹出。
4. 您只需要生成一次API密钥。NGC不保存您的密钥，所以请存放在安全的地方。（您可以通过单击API密钥右侧的复制图标将您的API密钥复制到剪贴板上。）
5. 如果API密钥丢失，您可以从NGC网站生成一个新的。当您生成一个新的API密钥时，旧的密钥将会失效。

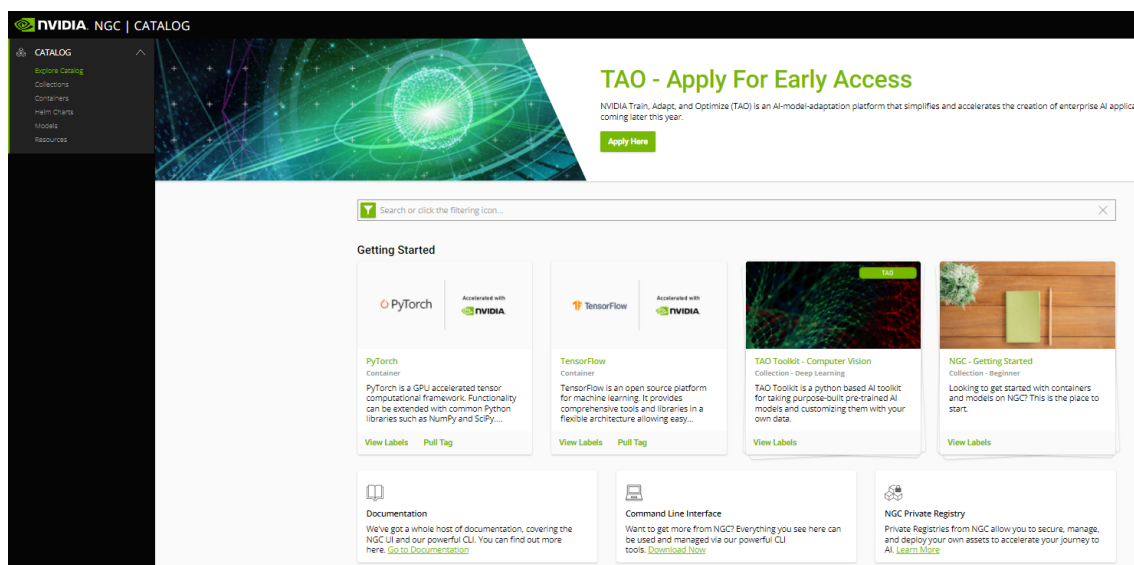
## 2.4.3 GPU服务器拉取NGC镜像

完成以上注册和API密钥获取后，您就可以在需要下载NGC镜像的GPU服务器上拉取镜像了，以下是注册步骤：

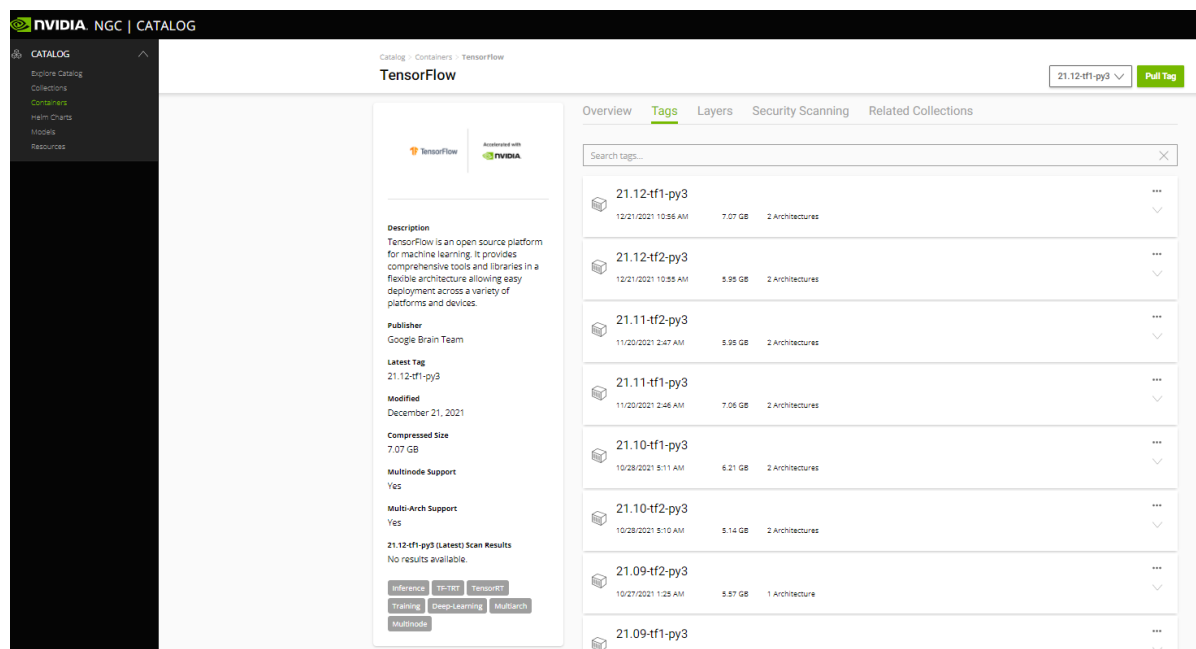
1. 在您的GPU服务器上，CLI中，通过如下命令注册这台设备

```
$ docker login nvcr.io
Username: $oauthtoken
Password: <Your API Key>
```

2. 在NGC页面中，选择您需要拉取的镜像。下面以TensorFlow镜像为例，您可以在‘Explore Catlog’页面找到TensorFlow，直接点击‘Pull Tag’下载最新版本的镜像



3. 如果需要更多内容，或拉取之前版本的镜像，请点击进入‘TensorFlow’页面，选则‘Tags’，再选择您需要的镜像。



#### 2.4.4 NGC镜像内容清单

如需要了解镜像内详细内容，请查阅[Deep Learning Framework Documentation](#)。