NVIDIA TLT
(Transfer Learning Toolkit)

安装与使用说明

GPUS
GPUSolution Technology Co.,Ltd.

系统配置：自有x86设备（不支持Jetson）

最低配置要求：

- CPU：1核
- 内存：4G
- GPU：1片4G显存
- 支持INT8：否
- 存储：50GB

推荐配置：

- CPU：8核
- 内存：32G
- GPU：4片/每片8G显存
- 支持INT8：是
- 存储：100GB

GPUS
GPUSolution Technology Co.,Ltd.

软件环境配置

- 操作系统：Ubuntu 18.04 64位元桌面版
- NVIDIA驱动：440
  （执行 **sudo apt install nvidia-drivers-440** 即可安装）
- CUDA/CUDNN/TensorRT：**非必要**
- Docker安装：
  https://docs.docker.com/install/linux/docker-ce/ubuntu/
- NVIDIA Docker安装：
  https://github.com/nvidia/nvidia-docker/wiki/Installation-(version-2.0)

GPUS
GPUSolution Technology Co.,Ltd.

Docker安装步骤

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
    add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository "deb [arch=amd64]
    https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

GPUS
GPUSolution Technology Co.,Ltd.

NVIDIA-Docker2安装步骤

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/ubuntu18.04/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
sudo apt-get install -y nvidia-docker2
sudo pkill -SIGHUP dockerd
```

GPUS
GPUSolution Technology Co.,Ltd.

环境测试：用 nvidia CUDA 10.2-base 容器

sudo docker run --runtime=nvidia --rm **nvidia/cuda:10.2-base** nvidia-smi

如果出现 nvidia-smi 标准信息（如下图），表示安装成功

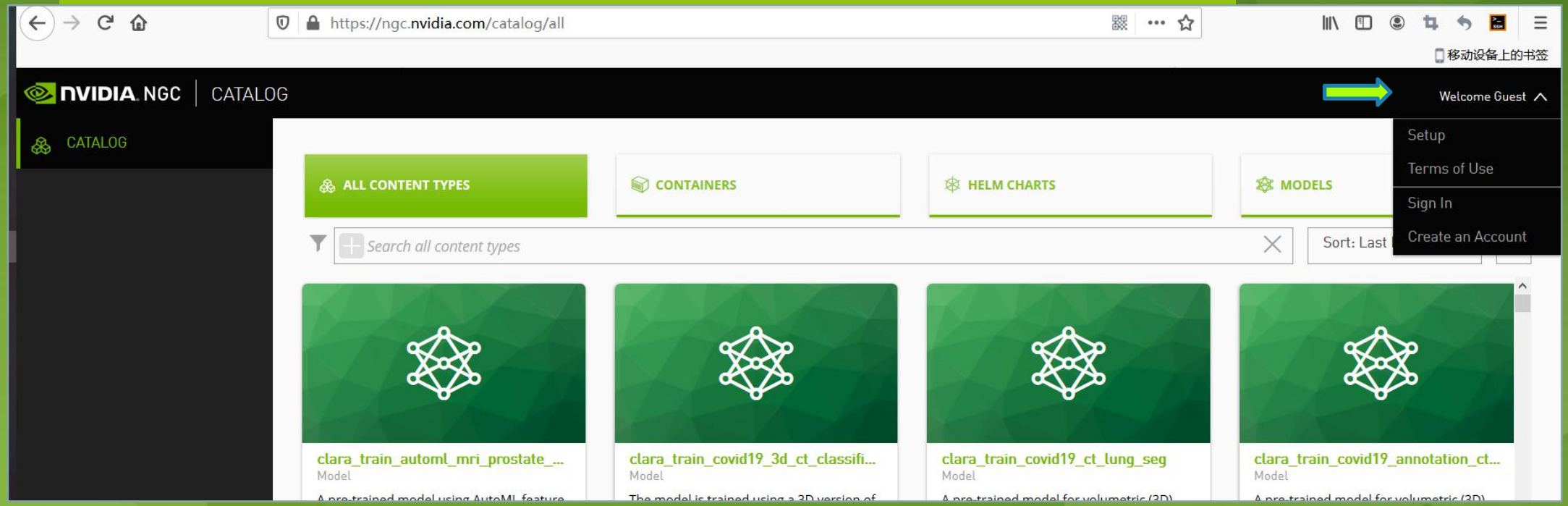
```
user0@gpus:~$ sudo docker run --runtime=nvidia --rm nvidia/cuda:10.2-base nvidia-smi
Sat Sep 19 14:04:40 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.100      Driver Version: 440.100      CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce RTX 2070    Off  | 00000000:02:00.0 Off |                  N/A |
| 46%   51C    P0    18W / 185W |      0MiB /  7982MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

说明：目前NGC的CUDA容器最新版（nvidia/cuda:latest）为11.0版，与现有软件仍有兼容性问题，会出错，测试时最好仍使用cuda:10.2-base版本

申请NGC账号

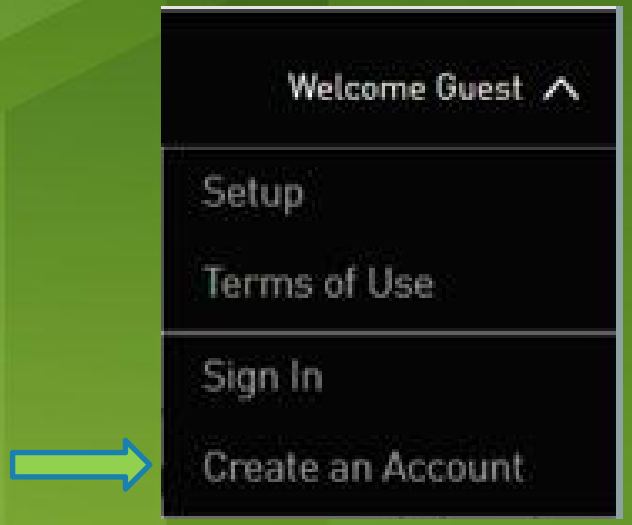
与 NVIDIA Developer 账号分开管理

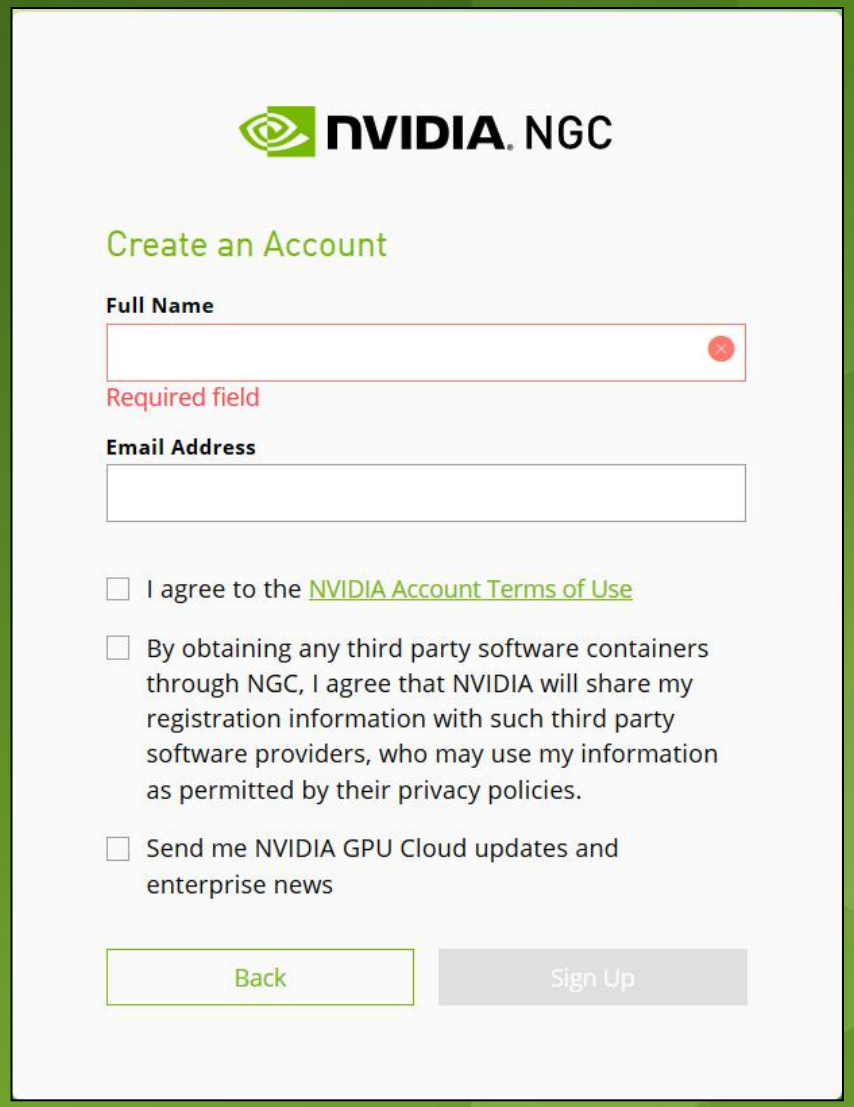
登录 https://ngc.nvidia.com 会直接出现 CATALOG画面



GPUS
GPUSolution Technology Co.,Ltd.

申请NGC账号

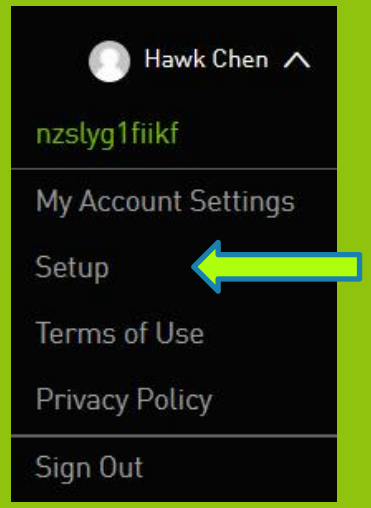


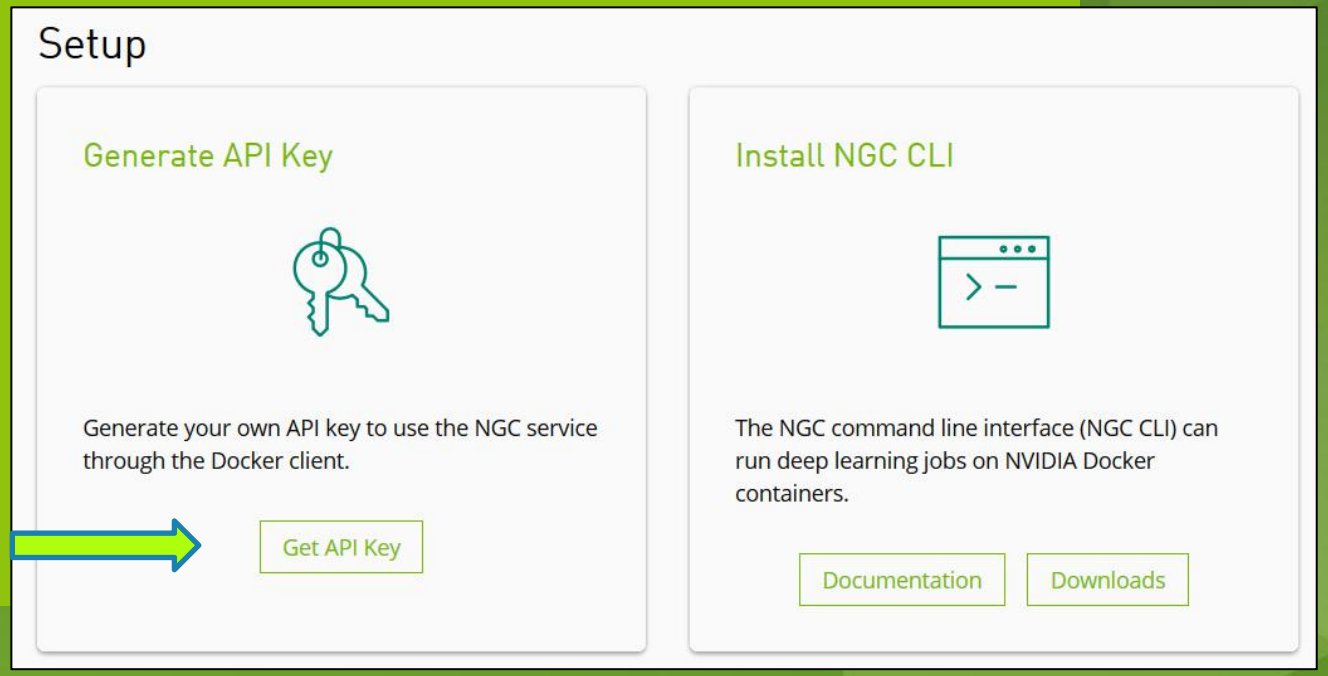


GPUS
GPUSolution Technology Co.,Ltd.

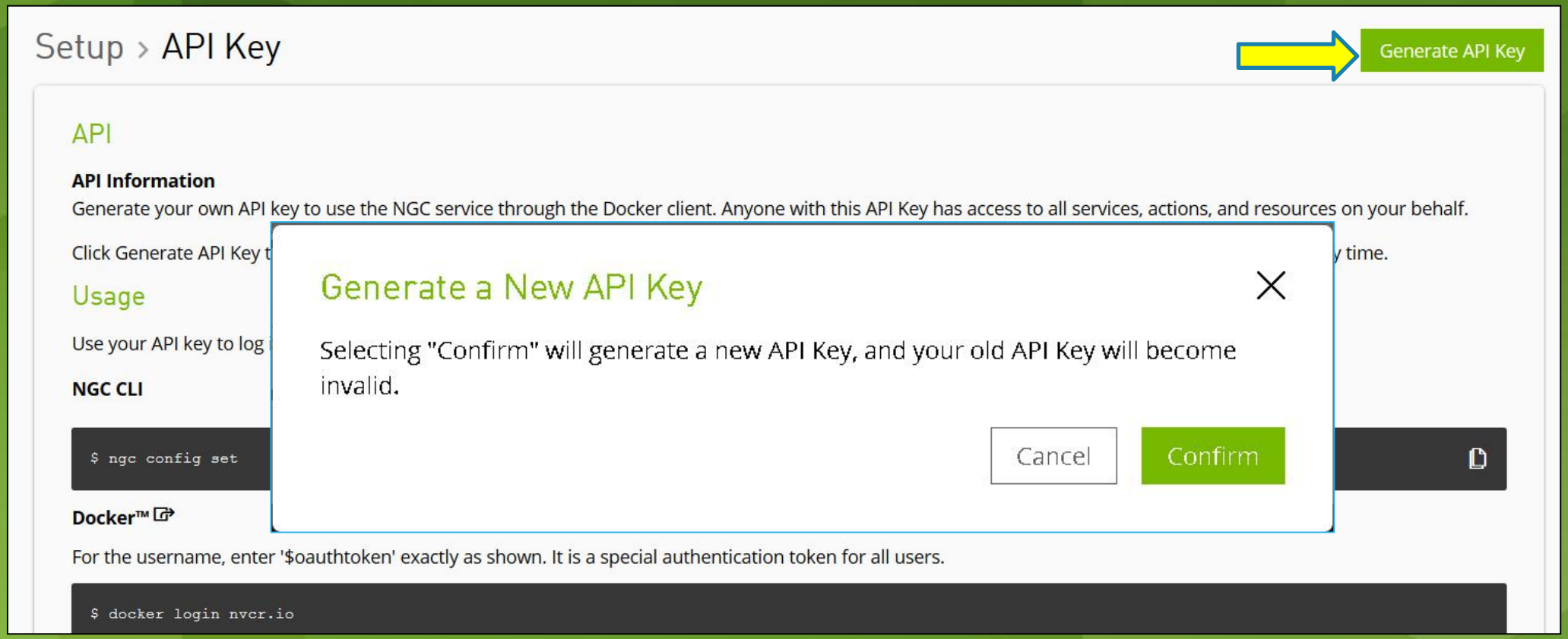
获取 NGC 的 API KEY（1）

说明文件：https://docs.nvidia.com/ngc/ngc-getting-started-guide/index.html#generating-api-key

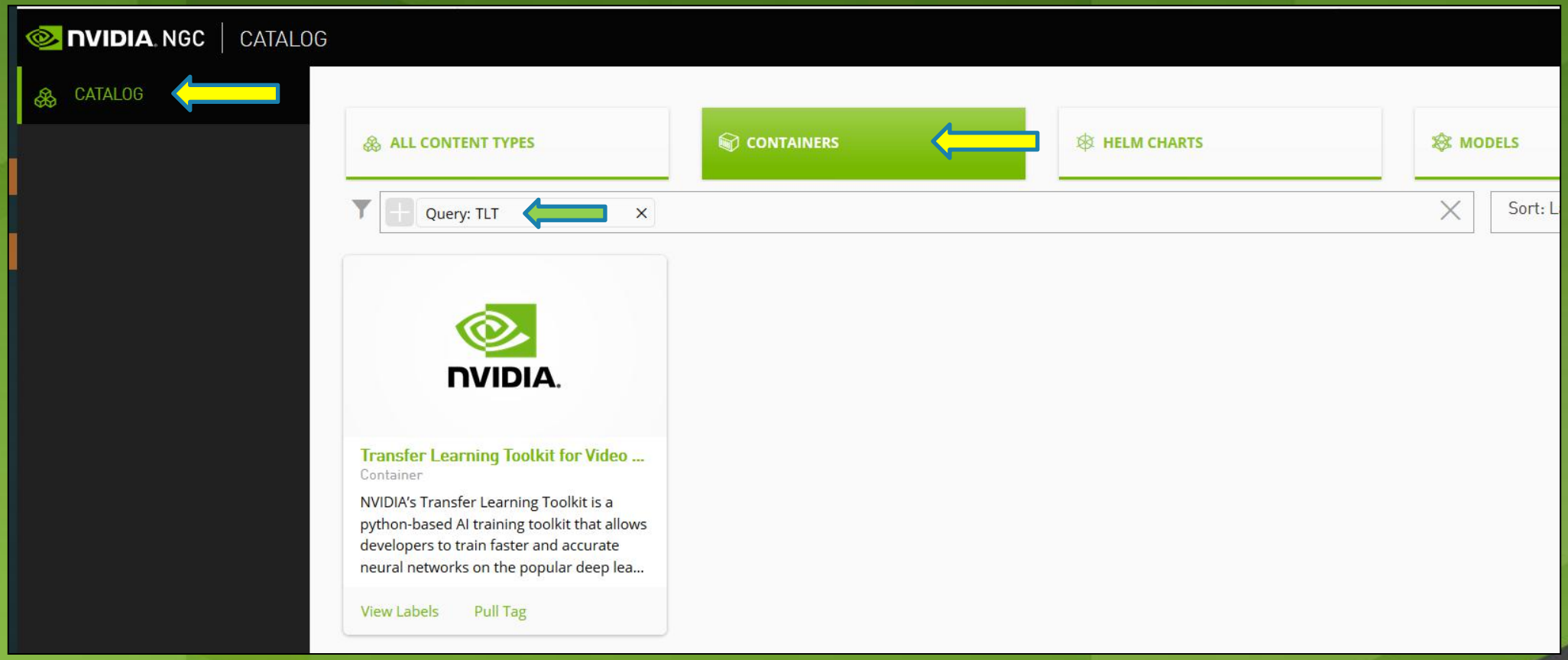




获取 NGC 的 API KEY（2）



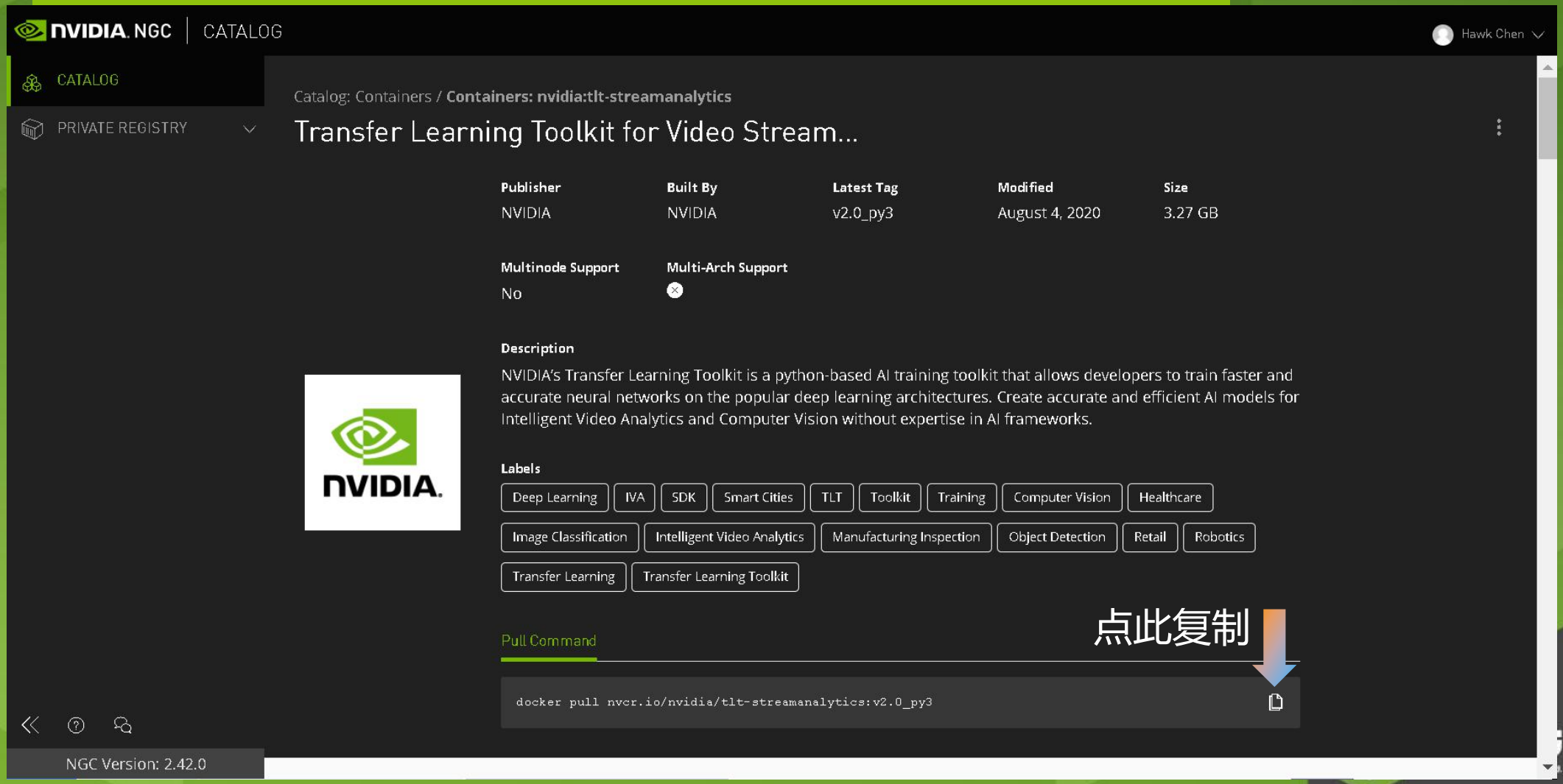
找到TLT工具镜像



GPUS
GPUSolution Technology Co.,Ltd.

复制tlt镜像的下载指令

docker pull nvcr.io/nvidia/tlt-streamanalytics:v2.0_py3



TLT操作说明（请自行仔细阅读）

NVIDIA NGC | CATALOG

Hawk Chen

CATALOG

PRIVATE REGISTRY

NVIDIA

Labels

Deep Learning | IVA | SDK | Smart Cities | TLT | Toolkit | Training | Computer Vision | Healthcare | Image Classification | Intelligent Video Analytics | Manufacturing Inspection | Object Detection | Retail | Robotics | Transfer Learning | Transfer Learning Toolkit

Pull Command

```
docker pull nvcr.io/nvidia/tlt-streamanalytics:v2.0_py3
```

Overview | Tags | Layers

What is Transfer Learning Toolkit?

Transfer Learning Toolkit (TLT) is a python based AI toolkit for taking purpose-built pre-trained AI models and customizing them with your own data. TLT adapts popular network architectures and backbones to your data, allowing you to train, fine tune, prune and export highly optimized and accurate AI models for edge deployment.

The pre-trained models accelerate the AI training process and reduce costs associated with large scale data collection, labeling, and training models from scratch. Transfer learning with pre-trained models can be used for AI applications in smart cities, retail, healthcare, industrial inspection and more.

Build end-to-end services and solutions for transforming pixels and sensor data to actionable insights using TLT, DeepStream SDK and TensorRT. The models are suitable for object detection, classification and instance segmentation.

NGC Version: 2.42.0

载入TLT镜像的方法1 ： 从 NGC PULL

- 在 Ubuntu 主机上执行以下指令

  sudo docker pull nvcr.io/nvidia/tlt-streamanalytics:v2.0_py3

```
gpus@ul8g:~$
c4959261975d: Downloading [==>                                                  ]  312.8kB/6.978MB
v2.0_dp_py2: Pulling from nvidia/tlt-streamanalytics
35b42117c431: Pulling fs layer
ad9c569a8d98: Pulling fs layer
293b44f45162: Downloading [==================================================>]     850B/850B
0c175077525d: Pulling fs layer
c4959261975d: Pulling fs layer
10a8d097f872: Pulling fs layer
09f9eb0153c1: Pulling fs layer
657bd07b9110: Pulling fs layer
defdb47b3acf: Pulling fs layer
23f2552ae755: Pulling fs layer
bb4ee296ceef: Pulling fs layer
0e7e380d4af4: Pulling fs layer
14eab7392f2c: Pulling fs layer
```

载入TLT镜像的方法2：从百度下载后导入

下载位置：https://pan.baidu.com/s/1AgrkxdZP-CwKMiiGaRwbDQ
提取码：6w6t
执行 sudo docker load -i tlt-2-py3.tar

```
gpus@ul8g:~$ docker load -i tlt-2-py2.tar
1ea5a27b0484: Loading layer [==================================================>]  123.5MB/123.5MB
24ab7de5faec: Loading layer [==================================================>]  11.78kB/11.78kB
10e46f329a25: Loading layer [==================================================>]  15.87kB/15.87kB
92d3f22d44f3: Loading layer [==================================================>]  3.072kB/3.072kB
960d2c742ca5: Loading layer [==================================================>]  16.91MB/16.91MB
558ff0bebfb5: Loading layer [==================================================>]  29.68MB/29.68MB
0328d19cc656: Loading layer [==================================================>]  3.072kB/3.072kB
918d270844ea: Loading layer [==================================================>]  816.1MB/816.1MB
ad8b4550a8b3: Loading layer [==================================================>]  1.401GB/1.401GB
595aa62e1f7d: Loading layer [==================================================>]  785.4MB/785.4MB
6179e684ab25: Loading layer [==================================================>]  3.584kB/3.584kB
692409bad262: Loading layer [==================================================>]  16.46MB/16.46MB
6473f92f3017: Loading layer [==================================================>]  2.089MB/2.089MB
b479acbe2fcf: Loading layer [==================================================>]  13.07MB/13.07MB
```

检查 TLT 镜像的载入状态

sudo docker images

```
gpus@u18g:~$ docker images
REPOSITORY                          TAG                    IMAGE ID
nvidia/cuda                         10.2-runtime           b02836d03446
nvidia/cuda                         10.2-base              9bef1d3b487d
nvidia/cuda                         10.2-base-ubuntu18.04  9bef1d3b487d
nvcr.io/nvidia/tlt-streamanalytics  v2.0_dp_py2            496dcdfc093a
nvidia/cuda                         9.2-base-ubuntu18.04   c197cbab8593
```

执行 TLT 容器

官网上指令解析

**sudo docker run --runtime=nvidia -it **
**-v "/path/to/dir/on/host":"/path/to/dir/in/docker" **
-p 8888:8888 nvcr.io/nvidia/tlt-streamanalytics:v2.0_py3 /bin/bash

- docker run： 组合指令指示“执行”
- --runtime=nvidia：要使用GPU 执行，必须将runtime指向nvidia
- -it 组合参数设定为 i（互动）与 t（进入终端）
- -v 作为容器外部指向容器内部的路径
  - 建议将**”/path/to/dir/in/docker”**部分直接设定为“**/workspace/tlt-experiments**“，
- -p 指定将容器外部端口指向内部端口。本实例执行jupyter，内部需要将8888指出容器
- nvcr.io/nvidia/tlt-streamanalytics:v2.0_dp_py2 为容器完整路径名
- /bin/bash 为进入容器后的执行指令

GPUS
GPUSolution Technology Co.,Ltd.

执行官网指令

执行后，会下载 ngccli_reg_linux.zip，这是NGC的命令集
执行完，会进入tlt容器里，进入指令模式

```
gpus@u18g:~$ docker run --runtime=nvidia -it -v /home/gpus/tlt_work:/workspace/tlt-experiments -p 8888:8888 nvcr.io/nvid
ia/tlt-streamanalytics:v2.0_dp_py2 /bin/bash
--2020-07-02 03:42:23--  https://ngc.nvidia.com/downloads/ngccli_reg_linux.zip
Resolving ngc.nvidia.com (ngc.nvidia.com)... 54.230.175.17, 54.230.175.19, 54.230.175.110, ...
Connecting to ngc.nvidia.com (ngc.nvidia.com)|54.230.175.17|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19995388 (19M) [application/zip]
Saving to: '/opt/ngccli/ngccli_reg_linux.zip'

ngccli_reg_linux.zip          100%[===================================================>]  19.07M   224KB/s    in 93s

2020-07-02 03:43:57 (210 KB/s) - '/opt/ngccli/ngccli_reg_linux.zip' saved [19995388/19995388]

Archive:  /opt/ngccli/ngccli_reg_linux.zip
  inflating: /opt/ngccli/ngc
 extracting: /opt/ngccli/ngc.md5
root@23e03854b26b:/workspace#
```

启动容器内jupyter

进入容器之后，执行以下指令启动jupyter

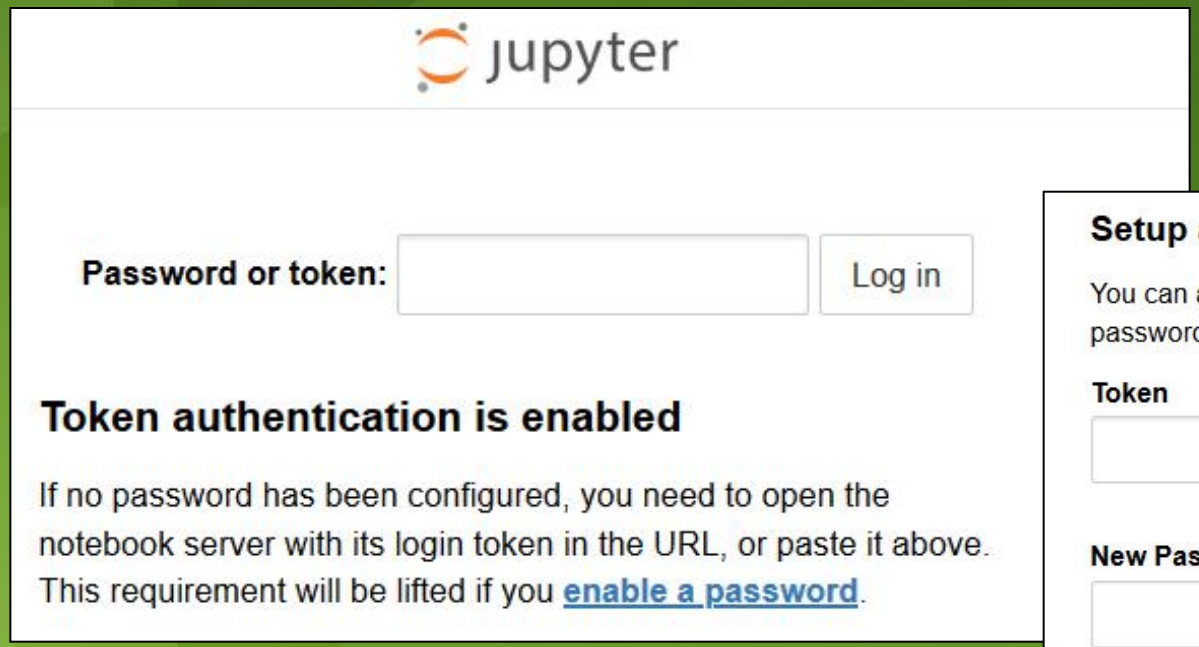
jupyter notebook --ip 0.0.0.0 --port 8888 --allow-root

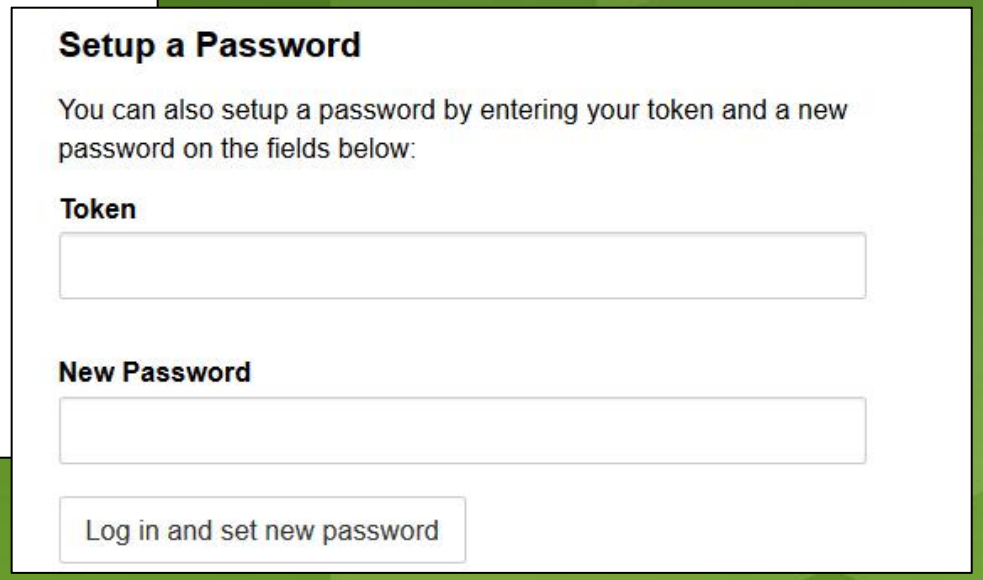
```
root@23e03854b26b:/workspace# jupyter notebook --ip 0.0.0.0 --port 8888 --allow-root
[I 03:52:30.095 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie
_secret
[I 03:52:30.331 NotebookApp] Serving notebooks from local directory: /workspace
[I 03:52:30.331 NotebookApp] The Jupyter Notebook is running at:
[I 03:52:30.331 NotebookApp] http://(23e03854b26b or 127.0.0.1):8888/?token=5f3c7ee5b5af115c47134e282fe8bf9d339563711f7d
2033
[I 03:52:30.331 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 03:52:30.337 NotebookApp] No web browser found: could not locate runnable browser.
[C 03:52:30.337 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///root/.local/share/jupyter/runtime/nbserver-16-open.html
    Or copy and paste one of these URLs:
        http://(23e03854b26b or 127.0.0.1):8888/?token=5f3c7ee5b5af115c47134e282fe8bf9d339563711f7d2033
```

这里的 token 在下一步的
Jupyter 登录时会用到!

需要 token 进入Jupyter

- 在容器外打开浏览器，输入 IP:8888 进入 Jupyter



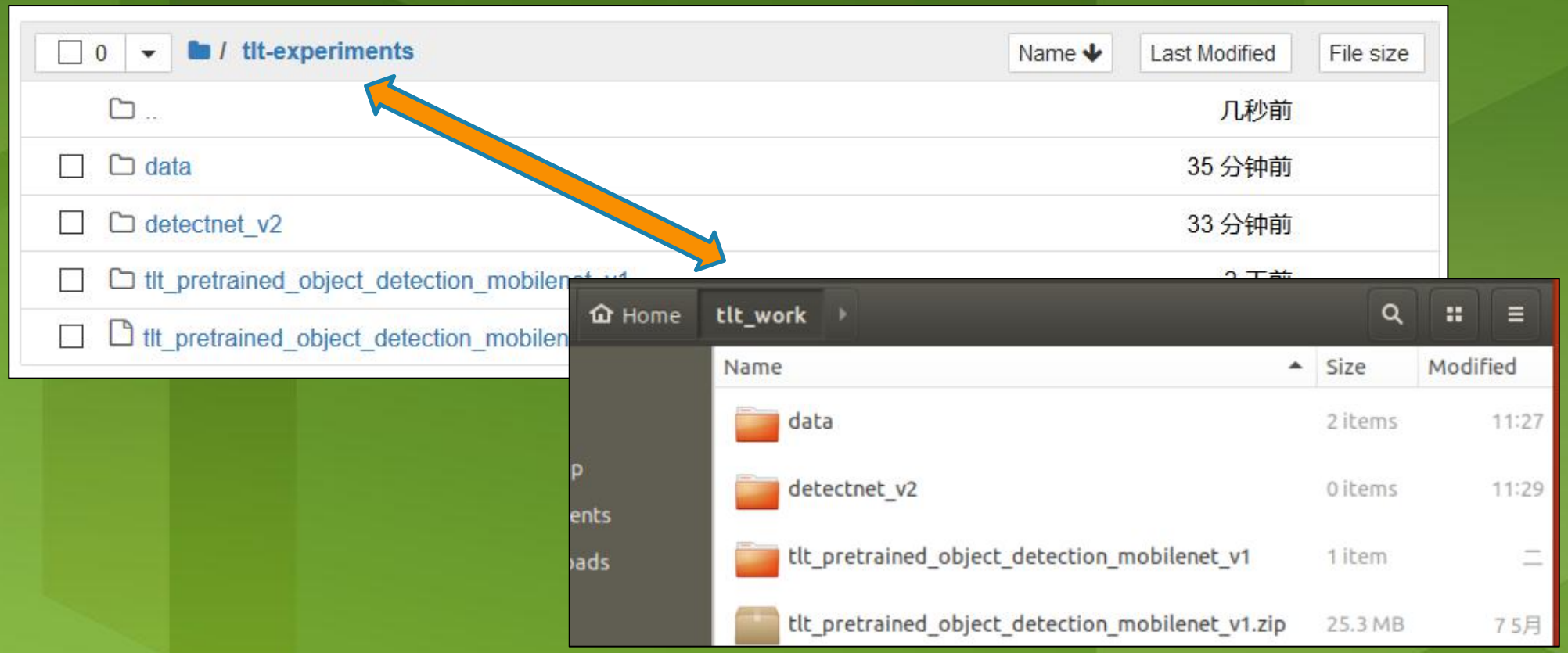


这里需要输入前一步骤的 token

进入 tlt-experiments 目录

GPUS
GPUSolution Technology Co.,Ltd.

容器内与容器外的路径指向



进一步操作

- 由于第一个终端被 Jupyter 占用，如果要执行其它指令时，该如何操作?

1. 打开另一个终端，执行 sudo docker ps 找出容器编号（黄框处）

```
gpus@u18g:~/tlt_work/detectnet_v2$ docker ps
CONTAINER ID        IMAGE                                                   COMMAND                  CREATED
      STATUS              PORTS                    NAMES
23e03854b26b        nvcr.io/nvidia/tlt-streamanalytics:v2.0_dp_py2   "entrypoint.sh /bin/…"   30 minutes ago
      Up 30 minutes       0.0.0.0:8888->8888/tcp   thirsty_mestorf
```

2. 执行 sudo docker exec -it <容器编号> /bin/bash

如此就能进入 tlt 容器执行其他操作

改善建议

- 在原指令上添加 --name <容器名> 可便于后续调用
  例如 --name tlt，则后续调用就不需要<容器ID>，直接
  sudo docker exec -it <容器名> /bin/bash 即可
- 进入容器之后
  1. **将examples下全部复制一份到 <tlt-experiments>目录下（同步到容器外部）**
  2. 执行<tlt-experiments/examples>下的范例
  3. 否则容器结束之后，容器内的所有更新过数据都不会保存
  4. 所有相应数据也不要存放在容器内

GPUS
GPUSolution Technology Co.,Ltd.

推理设备（Jetson）端的重要设定

GPUSolution Technology Co.,Ltd.

软件环境配置

开发环境：**NVIDIA Jetpack 4.4DP**

- 操作系统：Ubuntu 18.04 L4T with R32.4.2
- CUDA：10.2
- CUDNN：8.0.0
- TensorRT：7.1.0

安装 TLT-Convertor

- 至 https://developer.nvidia.com/tlt-converter-trt71 下载
- 安装 TensorRT OSS（https://docs.nvidia.com/metropolis/TLT/tlt-getting-started-guide/index.html#tensorrt_oss，下方“**TensorRT OSS on Jetson (ARM64)**”部分）

GPUS
GPUSolution Technology Co.,Ltd.

配套软件包境内下载

文件内容	链接：https://pan.baidu.com/s/1AgrkxdZP-CwKMiiGaRwbDQ 提取码：6w6t
tlt镜像（8G）	tlt-2-py3.tar
KITTY数据集（12G）	data_object_image_2.zip
KITTY标签集	data_object_label_2.zip
docker安装脚本	installDocker.sh
TLT Convertor	https://developer.nvidia.com/tlt-converter-trt71
推理端安装脚本	install.sh 与 install_pycuda.sh 放在同一个目录下执行
实验用Notebook范例	1_tlt-tensorrt-nano.tar

GPUS
GPUSolution Technology Co.,Ltd.