



Используем nvidia GPU внутри docker-контейнера





Антон Ширяев, разработчик систем
компьютерного зрения в АО "Кашалот".

0 Docker



Docker—программа, позволяющая операционной системе(ОС) запускать процессы в изолированном окружении на базе специально созданных образов(Docker-образов).

- ссылка на гайд на medium ([ссылка](#)) 
- ссылка на git-репозиторий ([ссылка](#)) 

Преимущества использования Docker-контейнеров

- Docker устраняет проблемы зависимостей и рабочего окружения
- Изоляция благодаря виртуализации
- Ускорение развертывания приложений
- Масштабирование
- Микросервисная архитектура



Docker не поддерживает nVidia GPU

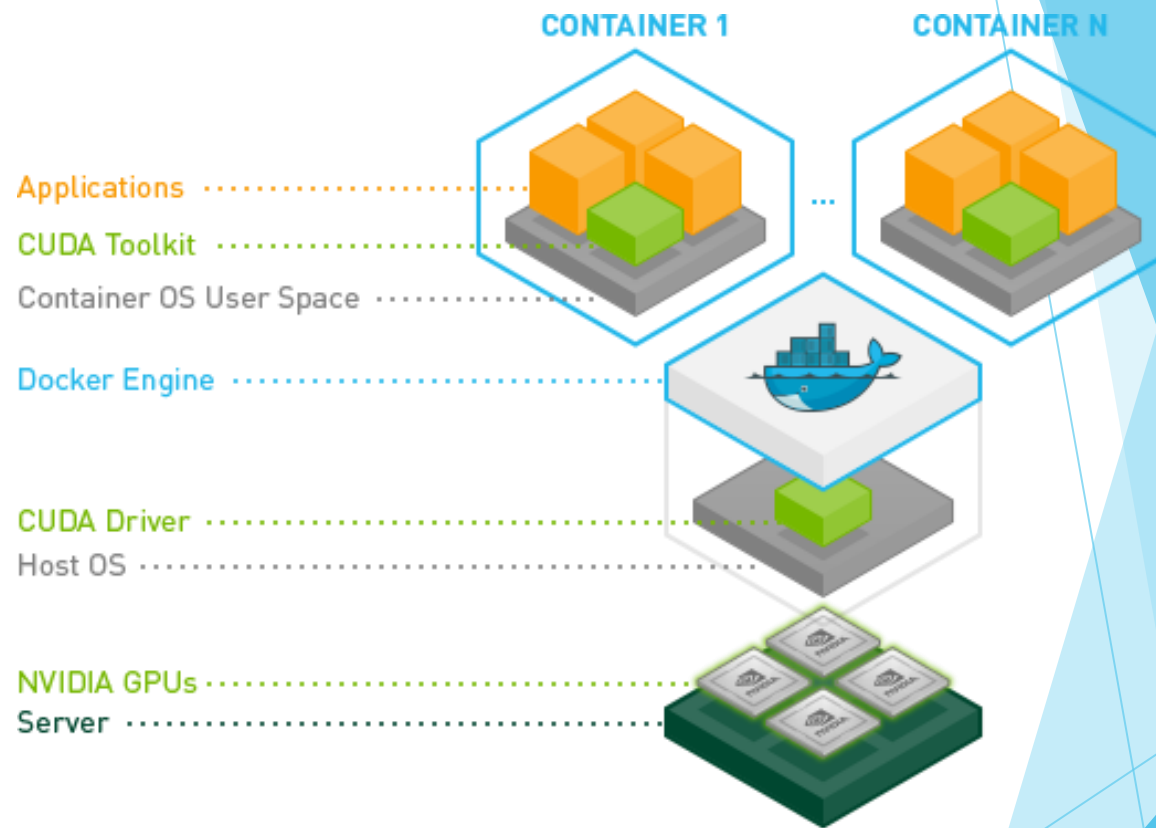


0 nvidia-docker

Для того, чтобы Docker-контейнеры могли использовать в своих расчетах видеокарту(GPU) нужно установить nvidia-docker.

nvidia-docker обеспечивает контейнерную виртуализацию аппаратного обеспечения GPU, подобно тому как сам Docker делает это для процессора CPU.

Подробнее об архитектуре nvidia-docker можно почитать в официальной документации ([ссылка](#)).



Установка nvidia-docker на Linux (платформа ARM64)

В целом для установки nvidia-docker на ОС Linux нужно выполнить следующие шаги:

- Установить драйвер для nVidia GPU
- Установить в нашу ОС Docker и настроить его стандартным образом
- Установить nvidia-docker поверх Docker

Пакетный менеджер Snap

- Пакетный менеджер Snap – содержит **Universal Linux packages**, доступность программ не только в Ubuntu, но и в других дистрибутивах.
- Разработчики могут создать один пакет, который потом можно будет одинаково установить в Ubuntu, Debian, Fedora, CentOS или любом другом дистрибутиве.



Собственная виртуализация



Ubuntu server 20.04

Из-за Snap у нас возникнет ошибка



"docker: Error response from daemon: could not select device driver "" with capabilities: [[gpu]]." ([ссылка](#)).



nvidia-docker устанавливаемый поверх Docker из snap из за виртуализации не имеет доступа к nVidia GPU и ему кажется, что ее нет в нашей системе.



Решение состоит в том, чтобы удалить Docker из snap, и установить обычный Docker непосредственно в нашу ОС.

```
Featured Server Snaps [ Help ]

These are popular snaps in server environments. Select or deselect with SPACE,
press ENTER to see more details of the package, publisher and versions
available.

[ ] microk8s           Lightweight Kubernetes for workstations and appliance ▶
[ ] nextcloud          Nextcloud Server - A safe home for all your data ▶
[ ] wekan              Open-Source kanban ▶
[ ] kata-containers    Lightweight virtual machines that seamlessly plug into ▶
[ ] docker             Docker container runtime ▶
[ ] canonical-livepatch Canonical Livepatch Client ▶
[ ] rocketchat-server  Group chat server for 100s, installed in seconds. ▶
[ ] mosquitto          Eclipse Mosquitto MQTT broker ▶
[ ] etcd               Resilient key-value store by CoreOS ▶
[ ] powershell         PowerShell for every system! ▶
[ ] stress-ng          A tool to load, stress test and benchmark a computer ▶
[ ] sabnzbd            SABnzbd ▶
[ ] wormhole           get things from one computer to another, safely ▶
[ ] aws-cli            Universal Command Line Interface for Amazon Web Services ▶
[ ] google-cloud-sdk   Command-line interface for Google Cloud Platform products ▶
[ ] slcli              Python based SoftLayer API Tool. ▶
[ ] doctl              The official DigitalOcean command line interface ▶
[ ] conjure-up          Package runtime for conjure-up spells ▶
[ ] minidlna-escoand    server software with the aim of being fully compliant ▶
[ ] postgresql10       PostgreSQL is a powerful, open source object-relational ▶
[ ] heroku             CLI client for Heroku ▶
[ ] keepalived          High availability VRRP/BFD and load-balancing for Linux ▶
[ ] prometheus         The Prometheus monitoring system and time series data ▶
[ ] juju               Simple, secure and stable devops. Juju keeps complex things ▶

[ Done ]
[ Back ]
```

Установка nvidia-docker на Linux (платформа ARM64)

В целом для установки nvidia-docker на ОС Linux нужно выполнить следующие шаги:

- ✓ Установить драйвер для nVidia GPU ([см. гайд](#))
- ✓ Установить в нашу ОС Docker и настроить его стандартным образом ([см. гайд](#))
- ☑ Установить nvidia-docker поверх Docker

Шаг 3 Установить nvidia-docker поверх Docker

Для выполнения данного шага выполним инструкции указанные в официальной документации nvidia([ссылка](#)).

Auto (Bash) ▾

```
# добавим GPG -ключ
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
  sudo apt-key add -

# Добавим дополнительные репозитории
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L \
  https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | \
  sudo tee /etc/apt/sources.list.d/nvidia-docker.list

# Обновляем список пакетов для apt и устанавливаем nvidia-docker2
sudo apt update && sudo apt install -y nvidia-docker2

# Перезапускаем службу Docker daemon, чтобы применить изменения
sudo systemctl restart docker
```

Шаг 3 Установить nvidia-docker поверх Docker

Интересно отметить, что при выполнении команды добавления репозитория:

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

Будет создан файл `/etc/apt/sources.list.d/nvidia-docker.list`. Если мы откроем этот файл на просмотр выполнив команду:

```
cat /etc/apt/sources.list.d/nvidia-docker.list
```

Шаг 3 Установить nvidia-docker поверх Docker

Несмотря на версию нашего дистрибутива Ubuntu 20.04 все URL-ссылки будут на репозитории nvidia-docker для Ubuntu 18.04

В официальной документации nVidia пишет, что так и должно быть ([ссылка](#)).

```
deb https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/${ARCH} /  
# deb https://nvidia.github.io/libnvidia-container/experimental/ubuntu18.04/${ARCH} /  
deb https://nvidia.github.io/nvidia-container-runtime/stable/ubuntu18.04/${ARCH} /  
# deb https://nvidia.github.io/nvidia-container-runtime/experimental/ubuntu18.04/${ARCH} /  
deb https://nvidia.github.io/nvidia-docker/ubuntu18.04/${ARCH} /
```

Note

Note that in some cases the downloaded list file may contain URLs that do not seem to match the expected value of `distribution` which is expected as packages may be used for all compatible distributions. As an examples:

- For `distribution` values of `ubuntu20.04` or `ubuntu22.04` the file will contain `ubuntu18.04` URLs
- For a `distribution` value of `debian11` the file will contain `debian10` URLs

“hello-world” от nvidia-docker 🙌

Теперь выполним “hello-world” от `nvidia-docker`, —запустим `Docker-контейнер` `nvidia/cuda:11.6.2-base-ubuntu20.04` и внутри него выполним команду `nvidia-smi`, которая покажет нам информацию о доступны внутри контейнера nVidia GPU.

Для этого выполним команду:

```
docker run --rm --runtime=nvidia --gpus all \  
nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi
```

“hello-world” от nvidia-docker 🖐️

```
Thu May 4 05:18:01 2023
+-----+
| NVIDIA-SMI 515.86.01      Driver Version: 515.86.01      CUDA Version: 11.7      |
+-----+-----+-----+-----+-----+
| 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   NVIDIA A10             Off   | 00000000:01:00.0 Off |                    |
| 0%   54C    P0     62W / 150W | 3195MiB / 24564MiB |      0%      Default |
|                                           N/A             |
+-----+-----+-----+-----+-----+

+-----+
| Processes: |
| GPU   GI    CI          PID    Type    Process name                  GPU Memory |
|      ID    ID                                   |            Usage   |
+-----+-----+-----+-----+-----+
|
```



Теперь Docker поддерживает nVidia GPU



Nvidia GPU снова не обнаружена?



При проверке может возникнуть сообщение о том, что nVidia GPU не обнаружен. Часто это возникает если мы устанавливали драйвер для nVidia GPU с официального сайта Nvidia и выполняли последующую сборку из исходников.

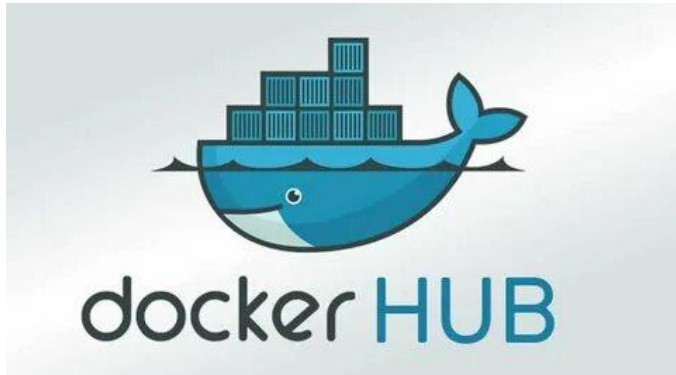


Нам нужно будет просто переустановить драйвер на нашу nVidia GPU. Перезагружаемся, уже установленный ранее нами nvidia-docker сам найдет nVidia GPU в системе, специально что то делать для этого не нужно.

Снова перепроверяем nvidia-docker запуском тестового контейнера, - все должно заработать.

Создадим docker-образ на основе образов от nVidia с поддержкой вычислений на GPU

- Docker Hub , - популярный реестр Docker-образов используемый по умолчанию в Docker ([ссылка](#)).
- nVidia имеет свой собственный каталог Docker-образов для самых разных задач NGC ([ссылка](#))



Создадим docker-образ на основе образов от nVidia с поддержкой вычислений на GPU

Для поддержки nVidia GPU мы будем брать в качестве базового образа

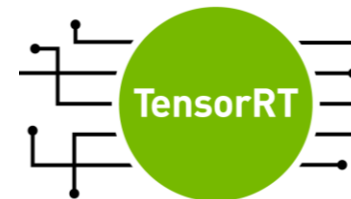
- образы с nvidia/cuda с Docker Hub([ссылка](#))

```
# образ для контейнера  
FROM nvidia/cuda:11.6.0-base-ubuntu20.04
```



- образы с nvidia/tensorrt ([ссылка](#))

```
# образ для контейнера  
FROM nvcr.io/nvidia/tensorrt:23.03-py3
```





▼ Segmentation

See [Segmentation Docs](#) for usage examples with these models.

Model	size (pixels)	mAP ^{box} 50-95	mAP ^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

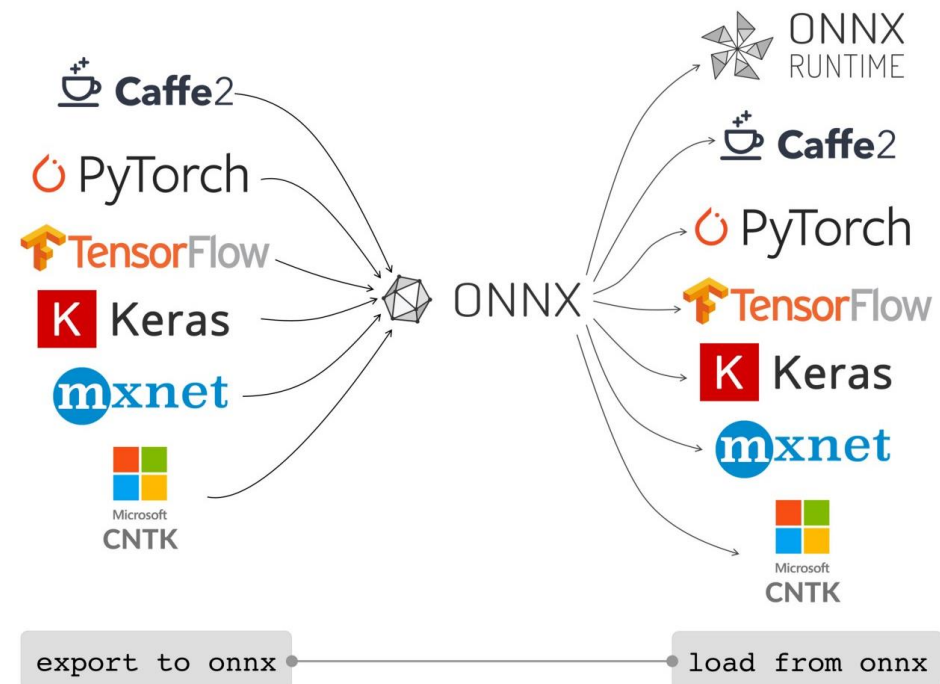
- **mAP^{val}** values are for single-model single-scale on [COCO val2017](#) dataset.
Reproduce by `yolo val segment data=coco.yaml device=0`
- **Speed** averaged over COCO val images using an [Amazon EC2 P4d](#) instance.
Reproduce by `yolo val segment data=coco128-seg.yaml batch=1 device=0|cpu`

► Classification

► Pose

ONNX и ONNX runtime

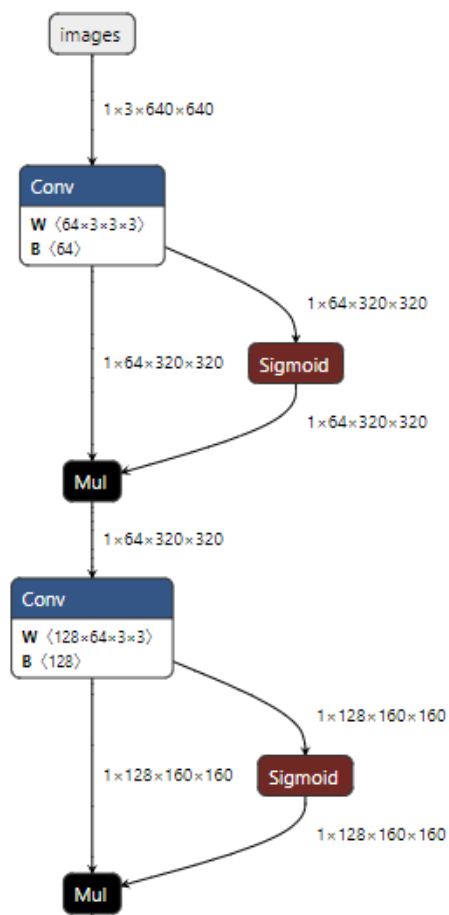
- ONNX (Open Neural Network Exchange) как открытый стандарт для представления моделей машинного обучения.
- Среда выполнения ONNX — это высокопроизводительный механизм вывода для развертывания моделей ONNX в рабочей среде.



<https://netron.app>

Посмотрим на вычислительный граф нейросети:

- yolov8l-seg.onnx



LUTZ ROEDER'S

NETRON

Open Model...

DOCUMENTATION

Sigmoid

Sigmoid takes one input data (Tensor) and produces one output data (Tensor) where the sigmoid function, $y = 1 / (1 + \exp(-x))$, is applied to the tensor elementwise.

INPUTS

X: `T`
Input tensor

OUTPUTS

Y: `T`
Output tensor

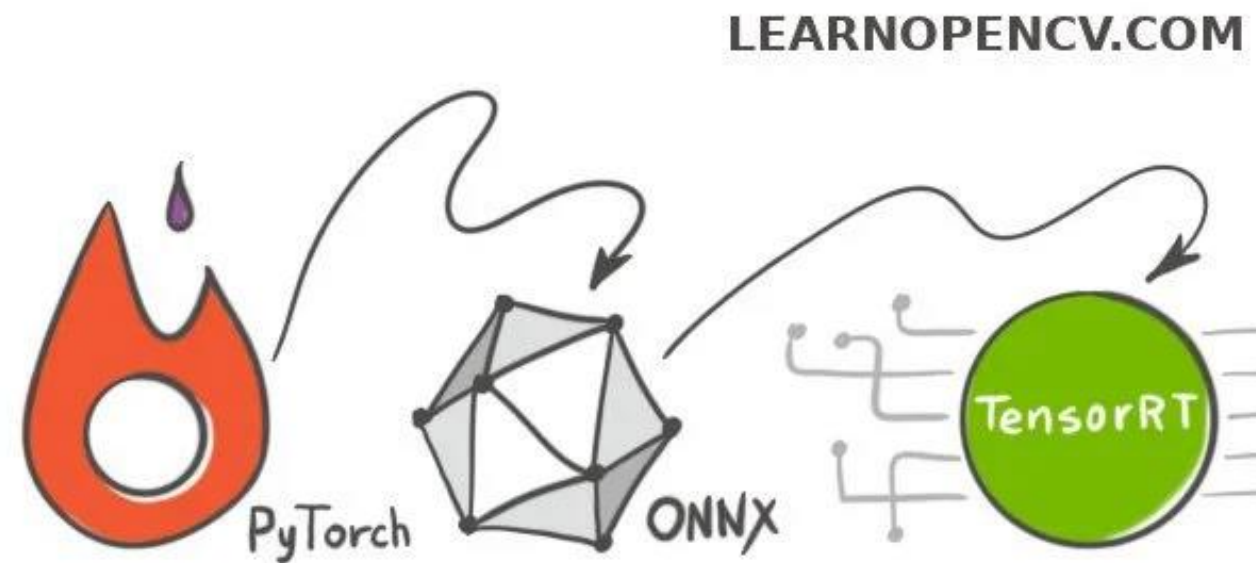
TYPE CONSTRAINTS

T: `tensor(float16)`, `tensor(float)`, `tensor(double)`
Constrain input and output types to float tensors.

EXAMPLES

0 TensorRT

- ▶ NVIDIA® TensorRT™ это специальный SDK для высокопроизводительного инференса моделей глубокого обучения, включающая в себя оптимизатор инференса на конкретном исполняющем устройстве (например нашей nVidia GPU A10 или Jetson Nano) и специальное окружение для быстрого запуска и высокой пропускной способности.
- ▶ С одной стороны TensorRT, преобразует вычислительный граф нашей нейросети в последовательность инструкций поддерживаемых нашей GPU, с другой выполнит различные оптимизации нашей модели для максимально эффективного запуска нейросети на данном устройстве.



HOW TO CONVERT A MODEL FROM PYTORCH TO TENSORRT AND SPEED UP INFERENCE

 PyTorch

 TensorFlow

 Keras

Caffe



ONNX



ONNX
RUNTIME



OpenVINO™

FP64/FP32/FP16/INT8 (Precision -> Speed)

Выполняем сборку docker-образа командой docker build



Для данной статьи я подготовил `git-hub` репозиторий ([ссылка](#)).

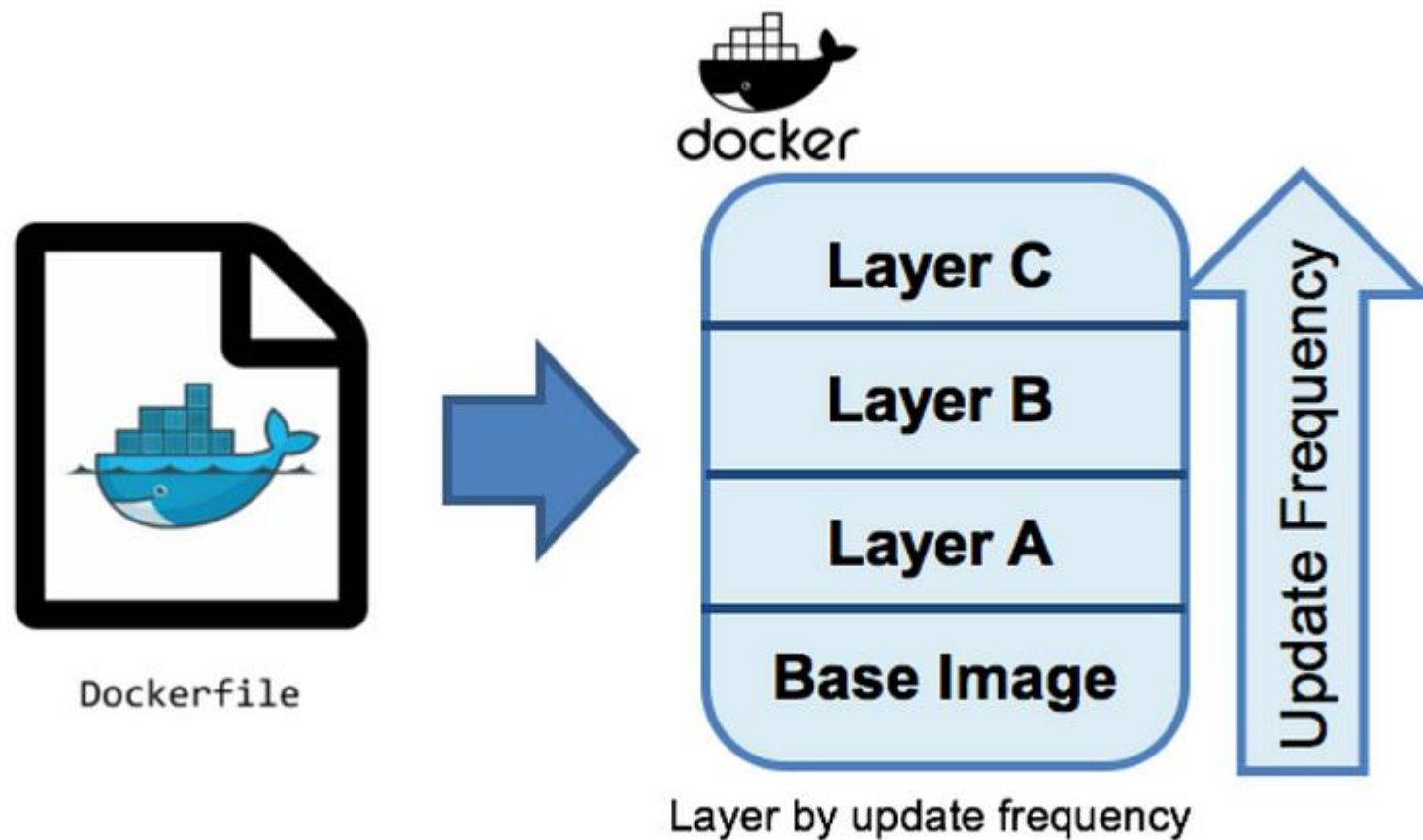
Клонируем его с помощью `git`, выполнив команду:

```
git clone git@github.com:medphisiker/docker_yolov8.git
```

Перейдем в него:

```
cd docker_yolov8
```


Слои в Dockerfile



Выполняем сборку docker-образа командой docker build



Согласно описанию репозитория ([ссылка](#)), переходим в каталог

`/.devcontainer` , выполнив команду:

```
cd .devcontainer
```

И затем выполняем команду для сборки Docker-образа :

```
docker build -t pytorch_yolov8:0.0.1 .
```


Запуск docker-образа pytorch_yolov8



```
cd ..
```

Теперь мы можем выполнив команду для запуска `docker-контейнера` `pytorch_yolov8`:

```
docker run \  
  --gpus all \  
  --rm \  
  -it \  
  -v ./yolov8:/workspace \  
  pytorch_yolov8:0.0.1
```

`Docker-образ` содержит рабочее окружение (все необходимые зависимости) для запуска `ultralytics yolov8` — `python`-скриптов, `Jupyter Notebook`'ов и нейронных сетей семейства `YOLOv8`.

Разные версии CUDA в нашей ОС и запущенном Docker-контейнере

Выполним команду `nvidia-smi` в терминале нашей ОС и в терминале внутри Docker-контейнера.

nVidia GPU A10 с драйвером 515.86.01 доступа и там и там.

- в нашей ОС стоит CUDA 11.7
- в Docker-контейнере CUDA 12.1

Еще одно удобство, - мы можем использовать внутри Docker-контейнера CUDA версии отличной от той, что стоит в нашей системе.

Это очень удобно для испытания новых функций или запуска последних версий фреймворков глубокого обучения требующих самой последней версии CUDA для своей работы.



CUDA 11.7



CUDA 12.1

Тестирование доступности вычислений на nVidia GPU для PyTorch



```
root@3a20fb4eafe6:/workspace# ls -la
total 136
drwxrwxr-x 2 1000 1000  4096 May  4 07:55 .
drwxr-xr-x 1 root root  4096 May  4 07:24 ..
-rw-rw-r-- 1 1000 1000 124933 May  4 06:16 YOLOv8_segment.ipynb
-rw-rw-r-- 1 1000 1000   639 May  4 07:55 pytorch_gpu_test.py
```

Файл `pytorch_gpu_test.py` вызывает PyTorch, выводит его версию и CUDA, выводит порядковый номер текущего устройства CUDA, название текущего устройства CUDA.

Так же он создает два случайных вектора, отправляет их на вычислительное устройство CUDA и перемножает их.

VS code server

Однако, у IDE `VS code` есть интересная особенность. Для работы с удаленным сервером VS code (расширение `Remote-SSH` ([ссылка](#))) может установить на него `VS code server`. `VS code server` это `VS code` без графического интерфейса пользователя к которому может подключиться `VS code` на нашем локальном компьютере по SSH. При этом мы сможем работать с удаленным сервером в интерфейсе VS code на нашем компьютере.

При этом удаленным компьютером может быть:

- реальный удаленный сервер (расширение `Remote-SSH` ([ссылка](#)))
- запущенный docker-контейнер (расширения `Dev Containers` ([ссылка](#)) и `Docker`([ссылка](#)))
- Linux в WSL (расширение `WSL` ([ссылка](#)))



Подключение VS Code к Docker-контейнеру



The screenshot shows the VS Code interface with a terminal window open. The top status bar indicates the workspace is a Docker container named 'pytorch_yolov8:0.0.1 (upbeat_heisenberg)' connected via SSH to 'gpu-server' as an administrator. The left sidebar shows a file explorer with a workspace named 'WORKSPACE [КОНТЕЙНЕР PYTOR...]'. Inside this workspace, there is a folder 'runs' containing files: 'bus.jpg', 'pytorch_gpu_test.py', 'YOLOv8_segment.ipynb' (selected), 'yolov8l-seg.engine', 'yolov8l-seg.onnx', 'yolov8l-seg.pt', and 'zidane.jpg'. The main editor area displays the 'YOLOv8_segment.ipynb' file, which is in 'Setup' mode. The setup instructions state: 'Pip install **ultralytics** and **dependencies** and check PyTorch and GPU.' Below this, a code cell is shown with the following Python code:

```
import ultralytics
ultralytics.checks()
```

The code cell has a green checkmark and a duration of 1.3s. The output of the code cell shows the following information:

```
... Ultralytics YOLOv8.0.91 🚀 Python-3.8.10 torch-1.12.0+cu113 CUDA:0 (NVIDIA A10, 24119MiB)
Setup complete ✅ (12 CPUs, 31.1 GB RAM, 327.9/913.8 GB disk)
```

Итоги

- познакомились с Docker
- установили nvidia-docker, предоставляющего нам возможность использования nVidia GPU внутри запущенных контейнеров
- создали наш первый Docker-образ с поддержкой вычислений на nVidia GPU
- запустили Docker-контейнер и убедились в доступности nVidia GPU для расчетов которые выполняет PyTorch
- подключили VS code к запущенному Docker-контейнеру , и начали работать с Jupyter Notebook привычным для нас образом
- запустили внутри Docker-контейнера нейросеть для instance-сегментации yolov8l-seg на CPU и GPU
- преобразовали нейросеть yolov8l-seg из формата PyTorch в TensorRT через ONNX, и получили ~15% ускорения на инференсе.

Спасибо за внимание жду ваших
вопросов

