UDocker Containers

Availability of udocker containers directly on *CVMFS* read-only filesystem in order to speedup their use and improve reproducibility, reliability and avoid interferences between calls, we will try to optimize compilations when ever possible. This containers can be used directly or run an user command throught a wrapper script.

The container technology is a conveniente way to provide stable software environments or to install them on situations where the configuration is complex or impossible. For example, the *tensorflow* framework is normally very hard to install on CentOS 7.x systems as found on our worknodes.

Available containers on CVMFS

Environment	Target Arch.	Arch. Optimizations	Container SO	Applications
udoker/tensorflow/cpu/2.4.1	Epyc_7552, Epyc_7501	AVX AVX2 FMA	Ubuntu 18.04	tensorflow-2.4.1 keras-2.43 pandas-1.1.5 madminer-0.8.0 numpy-1.19.5 scipy-1.5.4
udocker/tensorflow/gpu/2.4.1	Epyc_7552, NVidia_Tesla	AVX AVX2 FMA	Ubuntu 18.04	CUDA-11.2 tensorflow-2.4.1 keras-2.43 pandas-1.1.5 madminer-0.8.0 numpy-1.19.5 scipy-1.5.4

How to use the udocker containers

> The containers are meant to be run on workernodes, they will not work on the login servers, launch a batch job or start an **interactive session**. Note also that the *gpu* partition is the only one providing GPU devices.

Load environment

Load the appropriate environment, for exemple *udocker/tensorflow/gpu/2.4.1*:

```
$ module load udocker/tensorflow/gpu/2.4.1
```

This will configure the *udocker* environment and made available the wrapper *u_wrapper* used to start the container. When the container is started throught the wrapper the */tmp* and the user working directory is imported into the container.

Execute a command

Now we can run any command using the wrapper, for example:

```
$ u_wrapper nvidia-smi -L
******************************************************************************
*                                                                            *
*               STARTING 2bcfad7b-1750-3fb8-9fb1-74acdf4e869e               *
*                                                                            *
******************************************************************************
executing: bash
GPU 0: Tesla T4 (UUID: GPU-8cce58c9-f3f7-839c-50f9-63e21f042152)
GPU 1: Tesla T4 (UUID: GPU-1f698e19-a902-2e73-0a54-44e02fa9c8ee)
```

Execute an interactive shell

We can run an interactive shell, as long we acquire and interactive allocation:

```
$ u_wrapper bash -i
 ******************************************************************************
 *                                                                            *
 *               STARTING 2bcfad7b-1750-3fb8-9fb1-74acdf4e869e               *
 *                                                                            *
 ******************************************************************************
 executing: bash


________                               _______________
___  __/__________________________________  ____/__  /________      __
```

```
__  /  _  _\_  __\_  ___/  __\_  ___/_  /_   __  /_  __\_ | /| / /
_  /   /  __/  / / /(__  )/ /_/ /  /   _  __/   _  / / /_/ /_ |/ |/ /
/_/    \___//_/ /_//____/ \____//_/    /_/      /_/  \____/____/|__/


You are running this container as user with ID 5800002 and group 5800000,
which should map to the ID and group for your user on the Docker host. Great!


tf-docker ~ >
```

Example of a complete job script

We will run a Deep Learning example.

Get the python script **run.py** and create a submit script as follow:

```
$ cat dl.sh
#!/bin/bash
#SBATCH -p gpu
#SBATCH --gres=gpu
#SBATCH --mem=45G
module load udocker/tensorflow/gpu/2.4.1
u_wrapper python run.py
```

Next, launch the script and wait for completion, once it start to run it should be very fast, about 10 seconds.

```
$ sbatch dl.sh
Submitted batch job 1435321


$ squeue
  JOBID PARTITION     NAME     USER ST  TIME  NODES NODELIST(REASON)
1435321       gpu    dl.sh username  R  0:01      1 hpc062


$ ls -l
-rwxr-----+ 1 username usergroup   409 May 25 21:06 dl.sh
-rw-r-----+ 1 username usergroup  1417 May 25 20:49 run.py
-rw-r-----+ 1 username usergroup 17034 May 25 21:06 slurm-1435321.out
```

The output file should be something like the file **slurm-1435321.out**.

Revision #6
Created 25 May 2021 23:43:54 by João Paulo Martins
Updated 11 July 2021 17:20:44 by João Paulo Martins