Niharika Sinha
ns4451

HW3 – Running MNIST inside a Docker container

In this assignment, I will be running the MNIST digit recognition training and classification example inside a Docker container. Additionally, I will also run the MNIST dataset inside a Singularity container. Finally, I will compare my experiences in both the cases.

Vagrant and Docker

Vagrant provides a tool for working with virtual environments, providing a consistent development workflow across multiple operating systems. [1] It adds simplicity by providing a command line interface to manage the various environments, and text-based definitions for each environment in the form of vagrant files. A Docker is a container management that can run software within the containerization system. Containers are more lightweight than VMs. [2] Docker lacks support for certain operating systems, and Vagrant helps in such cases. [3]

My initial goal was to run the MNIST dataset inside a docker inside a Vagrant VM. For this, I tried installing a Vagrant and VirtualBox in my system [4], but I was facing a few issues. Initially, there was version mismatch between my Vagrant and VirtualBox. I was able to resolve this easily by experimenting with the different versions of both the softwares. After this, I faced another issue while trying to initialize the Vagrant with the box-cutter/ubuntu1404-desktop image. However, I realised that a pure-Docker workflow would suffice for running the MNIST training example, and I proceeded with this.

Running MNIST inside a Docker container

I installed Docker in my laptop, which was a seamless experience. I prepare my docker file, which I have attached along with this report. I cloned the git repository for getting the MNIST dataset. [5]

I issued the following commands to build and run the docker file.

docker build -t dockerfile .
docker run -it dockerfile 2>&1 | tee docker-run.out

I ran it for 2 epochs, and it took around 15 minutes to build, and about 30 minutes for the whole run to complete. Please find the screenshots of the build and run as follows:

docker build -t dockerfile .

```
[+] Building 929.5s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 237B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/pytorch/pytorch:latest
 => [1/3] FROM docker.io/pytorch/pytorch:latest@sha256:9904a7e081eaca29e3ee46afac87f2879676dd3bf7b5e9b8450454d84e074ef0
 => => resolve docker.io/pytorch/pytorch:latest@sha256:9904a7e081eaca29e3ee46afac87f2879676dd3bf7b5e9b8450454d84e074ef0
 => => sha256:cf06a7c3161117888114e7e91dbd21915efae33c2dbfb086380f7b21946d6e59 26.71MB / 26.71MB
 => => sha256:41acec2bfcb98f558bd046dbbaae583d0a6ecdfd2a9b9f8257abae753eff9528 9.93MB / 9.93MB
 => => sha256:f2531a2e2fb39948e631f13fb46cc8508f2f50c4f5928291ed9fcdb105cbfaba 2.74GB / 2.74GB
 => => sha256:9904a7e081eaca29e3ee46afac87f2879676dd3bf7b5e9b8450454d84e074ef0 1.16kB / 1.16kB
 => => sha256:ca04e7f7c8e58b8a0e1251bc37f314251fc90fb3bd7bc11bc78caac7d1dd53f8 2.82kB / 2.82kB
 => => sha256:491f1d30a6d5ae9c6368258ef9532786ba4fd94676dd862574d813a00c13b7c7 98B / 98B
 => => extracting sha256:cf06a7c3161117888114e7e91dbd21915efae33c2dbfb086380f7b21946d6e59
 => => extracting sha256:41acec2bfcb98f558bd046dbbaae583d0a6ecdfd2a9b9f8257abae753eff9528
 => => extracting sha256:f2531a2e2fb39948e631f13fb46cc8508f2f50c4f5928291ed9fcdb105cbfaba
 => => extracting sha256:491f1d30a6d5ae9c6368258ef9532786ba4fd94676dd862574d813a00c13b7c7
 => [internal] load build context
 => => transferring context: 5.46kB
 => [2/3] WORKDIR /app
 => [3/3] ADD examples/mnist/main.py /app/main.py
 => exporting to image
 => => exporting layers
 => => writing image sha256:6245f526df71e785837fa6afc490ea61b7a5ef8ed73ae12a2bd3a28b49ee6ddf
 => => naming to docker.io/library/dockerfile

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

docker run -it dockerfile 2>&1 | tee docker-run.out

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/MNIST/raw/train-images-idx3-ubyte.gz

  0% 0/9912422 [00:00<?, ?it/s]
  4% 389120/9912422 [00:00<00:02, 3854089.84it/s]
  8% 775168/9912422 [00:00<00:02, 3780437.93it/s]
 12% 1154048/9912422 [00:00<00:03, 2770329.86it/s]
 18% 1798144/9912422 [00:00<00:02, 3936019.88it/s]
 23% 2233344/9912422 [00:00<00:01, 3848093.62it/s]
 27% 2644992/9912422 [00:00<00:01, 3756759.88it/s]
 31% 3038208/9912422 [00:00<00:01, 3739079.45it/s]
 35% 3423232/9912422 [00:00<00:01, 3660514.68it/s]
 38% 3804160/9912422 [00:01<00:01, 3677080.26it/s]
 42% 4177920/9912422 [00:01<00:01, 3682629.69it/s]
 46% 4550656/9912422 [00:01<00:01, 3680089.06it/s]
 50% 4921344/9912422 [00:01<00:01, 3657134.35it/s]
 53% 5288960/9912422 [00:01<00:01, 3641722.41it/s]
 57% 5657600/9912422 [00:01<00:01, 3650266.94it/s]
 61% 6024192/9912422 [00:01<00:01, 3649866.25it/s]
 64% 6392832/9912422 [00:01<00:00, 3648412.32it/s]
 68% 6763520/9912422 [00:01<00:00, 3641967.12it/s]
 72% 7128064/9912422 [00:01<00:00, 3639902.73it/s]
 76% 7492608/9912422 [00:02<00:00, 3633647.35it/s]
 79% 7862272/9912422 [00:02<00:00, 3625835.55it/s]
 83% 8235008/9912422 [00:02<00:00, 3645984.76it/s]
 87% 8600576/9912422 [00:02<00:00, 3605775.97it/s]
 91% 8981504/9912422 [00:02<00:00, 3651623.31it/s]
 94% 9347072/9912422 [00:02<00:00, 3642240.73it/s]
 98% 9715712/9912422 [00:02<00:00, 3645129.60it/s]
9913344it [00:02, 3646997.13it/s]
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/MNIST/raw/train-labels-idx1-ubyte.gz

  0% 0/28881 [00:00<?, ?it/s]
29696it [00:00, 2091868.79it/s]
Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw/t10k-images-idx3-ubyte.gz

  0% 0/1648877 [00:00<?, ?it/s]
 14% 225280/1648877 [00:00<00:00, 2181384.34it/s]
 40% 663552/1648877 [00:00<00:00, 3459051.30it/s]
 61% 1011712/1648877 [00:00<00:00, 2286727.29it/s]
1649664it [00:00, 3436678.67it/s]
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw
```

I have just captured the run upto 30% for both the epochs for the sake of this document. They were as follows.

Epoch 1:

```
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz

  0% 0/4542 [00:00<?, ?it/s]
5120it [00:00, 2682677.89it/s]
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw

Train Epoch: 1 [0/60000 (0%)]   Loss: 2.329474
Train Epoch: 1 [640/60000 (1%)] Loss: 1.425025
Train Epoch: 1 [1280/60000 (2%)]        Loss: 0.797880
Train Epoch: 1 [1920/60000 (3%)]        Loss: 0.536104
Train Epoch: 1 [2560/60000 (4%)]        Loss: 0.427296
Train Epoch: 1 [3200/60000 (5%)]        Loss: 0.260153
Train Epoch: 1 [3840/60000 (6%)]        Loss: 0.323350
Train Epoch: 1 [4480/60000 (7%)]        Loss: 0.338577
Train Epoch: 1 [5120/60000 (9%)]        Loss: 0.533422
Train Epoch: 1 [5760/60000 (10%)]       Loss: 0.143478
Train Epoch: 1 [6400/60000 (11%)]       Loss: 0.186385
Train Epoch: 1 [7040/60000 (12%)]       Loss: 0.179244
Train Epoch: 1 [7680/60000 (13%)]       Loss: 0.188984
Train Epoch: 1 [8320/60000 (14%)]       Loss: 0.117034
Train Epoch: 1 [8960/60000 (15%)]       Loss: 0.216782
Train Epoch: 1 [9600/60000 (16%)]       Loss: 0.155205
Train Epoch: 1 [10240/60000 (17%)]      Loss: 0.518671
Train Epoch: 1 [10880/60000 (18%)]      Loss: 0.198790
Train Epoch: 1 [11520/60000 (19%)]      Loss: 0.598619
Train Epoch: 1 [12160/60000 (20%)]      Loss: 0.176906
Train Epoch: 1 [12800/60000 (21%)]      Loss: 0.135584
Train Epoch: 1 [13440/60000 (22%)]      Loss: 0.182158
Train Epoch: 1 [14080/60000 (23%)]      Loss: 0.118789
Train Epoch: 1 [14720/60000 (25%)]      Loss: 0.281630
Train Epoch: 1 [15360/60000 (26%)]      Loss: 0.103147
Train Epoch: 1 [16000/60000 (27%)]      Loss: 0.268325
Train Epoch: 1 [16640/60000 (28%)]      Loss: 0.128402
Train Epoch: 1 [17280/60000 (29%)]      Loss: 0.049959
Train Epoch: 1 [17920/60000 (30%)]      Loss: 0.123116
```

Epoch 2:

```
Train Epoch: 2 [0/60000 (0%)]   Loss: 0.024104
Train Epoch: 2 [640/60000 (1%)] Loss: 0.030736
Train Epoch: 2 [1280/60000 (2%)]        Loss: 0.050024
Train Epoch: 2 [1920/60000 (3%)]        Loss: 0.203170
Train Epoch: 2 [2560/60000 (4%)]        Loss: 0.048158
Train Epoch: 2 [3200/60000 (5%)]        Loss: 0.025160
Train Epoch: 2 [3840/60000 (6%)]        Loss: 0.006560
Train Epoch: 2 [4480/60000 (7%)]        Loss: 0.078596
Train Epoch: 2 [5120/60000 (9%)]        Loss: 0.135828
Train Epoch: 2 [5760/60000 (10%)]       Loss: 0.057491
Train Epoch: 2 [6400/60000 (11%)]       Loss: 0.202980
Train Epoch: 2 [7040/60000 (12%)]       Loss: 0.219049
Train Epoch: 2 [7680/60000 (13%)]       Loss: 0.040855
Train Epoch: 2 [8320/60000 (14%)]       Loss: 0.012909
Train Epoch: 2 [8960/60000 (15%)]       Loss: 0.153170
Train Epoch: 2 [9600/60000 (16%)]       Loss: 0.041601
Train Epoch: 2 [10240/60000 (17%)]      Loss: 0.221615
Train Epoch: 2 [10880/60000 (18%)]      Loss: 0.027948
Train Epoch: 2 [11520/60000 (19%)]      Loss: 0.099651
Train Epoch: 2 [12160/60000 (20%)]      Loss: 0.085562
Train Epoch: 2 [12800/60000 (21%)]      Loss: 0.089587
Train Epoch: 2 [13440/60000 (22%)]      Loss: 0.034206
Train Epoch: 2 [14080/60000 (23%)]      Loss: 0.006594
Train Epoch: 2 [14720/60000 (25%)]      Loss: 0.092768
Train Epoch: 2 [15360/60000 (26%)]      Loss: 0.061976
Train Epoch: 2 [16000/60000 (27%)]      Loss: 0.104214
Train Epoch: 2 [16640/60000 (28%)]      Loss: 0.135084
Train Epoch: 2 [17280/60000 (29%)]      Loss: 0.002263
Train Epoch: 2 [17920/60000 (30%)]      Loss: 0.090156
```

The results of the run for both the epochs were as follows.

Epoch 1:

```
Test set: Average loss: 0.0467, Accuracy: 9837/10000 (98%)
```

Epoch 2:

```
Test set: Average loss: 0.0379, Accuracy: 9865/10000 (99%)
```

Running MNIST dataset in a Singularity container

Singularity is a container platform, that lets us run containers in a portable and reproducible manner. It helps in running complex applications on HPC clusters. It makes use of GPUs, high speed networks and parallel filesystems. It makes use of a Singularity Image Files (SIFs) containing information about the container. [6]

I used the singularity pre-installed on Greene cluster. The whole procedure was pretty straight-forward, with the steps being similar to what I had done for Docker.

I first cloned the git repository into a folder. [7] Using the pull command, I built the container using the following URL:

singularity pull mnist.sif docker://pytorch/pytorch:latest

```
[ns4451@log-3 CML]$ singularity pull mnist.sif docker://pytorch/pytorch:latest
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
Copying blob cf06a7c31611 done
Copying blob 41acec2bfcb9 done
Copying blob f2531a2e2fb3 done
Copying blob 491f1d30a6d5 done
Copying config e72e93adbb done
Writing manifest to image destination
Storing signatures
2022/03/18 01:43:23  info unpack layer: sha256:cf06a7c3161117888114e7e91dbd21915efae33c2dbfb0
86380f7b21946d6e59
2022/03/18 01:43:23  info unpack layer: sha256:41acec2bfcb98f558bd046dbbaae583d0a6ecdfd2a9b9f
8257abae753eff9528
2022/03/18 01:43:24  info unpack layer: sha256:f2531a2e2fb39948e631f13fb46cc8508f2f50c4f59282
91ed9fcdb105cbfaba
2022/03/18 01:44:20  info unpack layer: sha256:491f1d30a6d5ae9c6368258ef9532786ba4fd94676dd86
2574d813a00c13b7c7
INFO:    Creating SIF file...
```

The srun and exec commands are used for running in a container. I passed the SIF file to the command:

srun --pty --gres=gpu:1 --mem=20GB /bin/bash
singularity exec --nv mnist.sif /bin/bash

Just like in Docker, I ran the training dataset for 2 epochs.

python examples/mnist/main.py --epochs 2

```
[ns4451@gr004 CML]$ singularity exec --nv mnist.sif /bin/bash
Singularity> python examples/mnist/main.py --epochs 1
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/MNIST/raw/
train-images-idx3-ubyte.gz
9913344it [00:00, 171099518.52it/s]
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/MNIST/raw/
train-labels-idx1-ubyte.gz
29696it [00:00, 24129029.75it/s]
Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw/t
10k-images-idx3-ubyte.gz
1649664it [00:00, 108955079.35it/s]
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw/t
10k-labels-idx1-ubyte.gz
5120it [00:00, 75615621.41it/s]
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw
```

Again, I have just captured the run upto 30% for both the epochs for the sake of this document. They were as follows.

Epoch 1:

```
Singularity> python examples/mnist/main.py --epochs 2
Train Epoch: 1 [0/60000 (0%)]	Loss: 2.299825
Train Epoch: 1 [640/60000 (1%)]	Loss: 1.725018
Train Epoch: 1 [1280/60000 (2%)]	Loss: 0.946259
Train Epoch: 1 [1920/60000 (3%)]	Loss: 0.656993
Train Epoch: 1 [2560/60000 (4%)]	Loss: 0.371875
Train Epoch: 1 [3200/60000 (5%)]	Loss: 0.338005
Train Epoch: 1 [3840/60000 (6%)]	Loss: 0.164963
Train Epoch: 1 [4480/60000 (7%)]	Loss: 0.552061
Train Epoch: 1 [5120/60000 (9%)]	Loss: 0.293149
Train Epoch: 1 [5760/60000 (10%)]	Loss: 0.149978
Train Epoch: 1 [6400/60000 (11%)]	Loss: 0.264153
Train Epoch: 1 [7040/60000 (12%)]	Loss: 0.236972
Train Epoch: 1 [7680/60000 (13%)]	Loss: 0.443011
Train Epoch: 1 [8320/60000 (14%)]	Loss: 0.207840
Train Epoch: 1 [8960/60000 (15%)]	Loss: 0.254723
Train Epoch: 1 [9600/60000 (16%)]	Loss: 0.076072
Train Epoch: 1 [10240/60000 (17%)]	Loss: 0.190321
Train Epoch: 1 [10880/60000 (18%)]	Loss: 0.315045
Train Epoch: 1 [11520/60000 (19%)]	Loss: 0.164129
Train Epoch: 1 [12160/60000 (20%)]	Loss: 0.194034
Train Epoch: 1 [12800/60000 (21%)]	Loss: 0.184260
Train Epoch: 1 [13440/60000 (22%)]	Loss: 0.125211
Train Epoch: 1 [14080/60000 (23%)]	Loss: 0.137101
Train Epoch: 1 [14720/60000 (25%)]	Loss: 0.135418
Train Epoch: 1 [15360/60000 (26%)]	Loss: 0.370396
Train Epoch: 1 [16000/60000 (27%)]	Loss: 0.380502
Train Epoch: 1 [16640/60000 (28%)]	Loss: 0.142013
Train Epoch: 1 [17280/60000 (29%)]	Loss: 0.151856
Train Epoch: 1 [17920/60000 (30%)]	Loss: 0.170979
```

Epoch 2:

```
Train Epoch: 2 [0/60000 (0%)]	Loss: 0.075553
Train Epoch: 2 [640/60000 (1%)]	Loss: 0.014773
Train Epoch: 2 [1280/60000 (2%)]	Loss: 0.027354
Train Epoch: 2 [1920/60000 (3%)]	Loss: 0.094160
Train Epoch: 2 [2560/60000 (4%)]	Loss: 0.036161
Train Epoch: 2 [3200/60000 (5%)]	Loss: 0.036083
Train Epoch: 2 [3840/60000 (6%)]	Loss: 0.110977
Train Epoch: 2 [4480/60000 (7%)]	Loss: 0.137650
Train Epoch: 2 [5120/60000 (9%)]	Loss: 0.153073
Train Epoch: 2 [5760/60000 (10%)]	Loss: 0.049017
Train Epoch: 2 [6400/60000 (11%)]	Loss: 0.047997
Train Epoch: 2 [7040/60000 (12%)]	Loss: 0.041508
Train Epoch: 2 [7680/60000 (13%)]	Loss: 0.048662
Train Epoch: 2 [8320/60000 (14%)]	Loss: 0.045322
Train Epoch: 2 [8960/60000 (15%)]	Loss: 0.033202
Train Epoch: 2 [9600/60000 (16%)]	Loss: 0.145247
Train Epoch: 2 [10240/60000 (17%)]	Loss: 0.082952
Train Epoch: 2 [10880/60000 (18%)]	Loss: 0.023908
Train Epoch: 2 [11520/60000 (19%)]	Loss: 0.126111
Train Epoch: 2 [12160/60000 (20%)]	Loss: 0.053909
Train Epoch: 2 [12800/60000 (21%)]	Loss: 0.057741
Train Epoch: 2 [13440/60000 (22%)]	Loss: 0.013933
Train Epoch: 2 [14080/60000 (23%)]	Loss: 0.296775
Train Epoch: 2 [14720/60000 (25%)]	Loss: 0.018316
Train Epoch: 2 [15360/60000 (26%)]	Loss: 0.140395
Train Epoch: 2 [16000/60000 (27%)]	Loss: 0.011990
Train Epoch: 2 [16640/60000 (28%)]	Loss: 0.176290
Train Epoch: 2 [17280/60000 (29%)]	Loss: 0.020064
Train Epoch: 2 [17920/60000 (30%)]	Loss: 0.044195
```

The results of the run for both the epochs were as follows.

Epoch 1:

```
Test set: Average loss: 0.0505, Accuracy: 9827/10000 (98%)
```

Epoch 2:

```
Test set: Average loss: 0.0326, Accuracy: 9892/10000 (99%)
```

Docker and Singularity

Comparison	**Docker**	**Singularity**
Use case	Focus on microservices.	Focus on entire application and workflows.
Infrastructure independence	Can run across multiple OS.	Can run across multiple OS.
Inter-conversion	Cannot host a singularity container in Docker.	Can import Docker images and converts them into singularity images.
Runtime	Took total 30 minutes to build and run.	Took less than 15 minutes to run and build.
Documentation and ease of use	Lots of documentation available. Had some problem initially starting the Docker engine on Windows, but found a workaround easily online.	Not very extensive documentation available. Since I ran it on the Greene cluster, executing the commands was very convenient.

References

[1] *https://opensource.com/resources/vagrant*
[2] *https://docs.docker.com/get-started/overview/*
[3] *https://www.vagrantup.com/intro/vs/docker*
[4] *https://www.swtestacademy.com/quick-start-vagrant-windows-10/*
[5] *https://github.com/pytorch/examples/*
[6] *https://sylabs.io/guides/3.5/user-guide/introduction.html#:~:text=Singularity%20is%20a%20container%20platform,that%20is%20portable%20and%20reproducible.*
[7] *https://github.com/pytorch/examples.git*