Docker Cheat Sheet

from: Learn Docker - Fundamentals of Docker 19.x

list all containers

```
docker container ls -a
```

list all containers by the ID

```
docker container ls -a -q
```

show images

```
docker images
```

show container metadata

```
docker container inspect [container]
```

- show container metadata and filter (only show) .State and print output into jq tool

```
docker container inspect -f "{{json .State}}" trivia | jq .
```

execute shell (sh) in a container, t for terminal emulator, i for interactive

```
docker container exec -i -t trivia /bin/sh
```

run processes as daemon (-d) and define env variables (-e)

```
docker container exec -it \
-e MY_VAR="Hello world" \
trivia /bin/sh
```

leave a running container

```
ctrl + D
```

attach terminal's std io and errors to a running container

```
docker container attach trivia
```

detach from running container and leave it running - didnt work

```
ctrl + P + ctrl + Q
```

detach and stop the container

```
ctrl + C
```

run a web server container (Nginx) - see pages 64 and 65 for details

```
docker run -d --name nginx -p 8080:80 nginx:alpine
```

access logs of a container

```
docker container logs [container]
```

access few logs - use -t or –tail

```
docker container logs -t 5 [container]
```

access last 5 logs and follow (-f or –follow) logs of a container in real-time

```
docker container logs -t 5 -f [container]
```

run a container

```
docker run -it [container]
```

run a container with docker port ID (default) specified

```
docker run -it -p 8888:8888 tensorflow/tensorflow
```

Add User to docker super-user (SU) group - 3 steps

- If you don't want to use sudo when you use the docker command, create a Unix group called docker and add users to it. When the docker daemon starts, it makes the ownership of the Unix socket read/writable by the docker group.

Add the docker group if it doesn't already exist

```
sudo groupadd docker # 1
```

Add the connected user "$USER" to the docker group. Change the user name to

match your preferred user if you do not want to use your current user:

```
sudo gpasswd -a $USER docker # 2
```

Either do a newgrp docker or log out/in to activate the changes to groups.

You can use

docker run hello-world # 3

docker run -it –rm –name tf

To avoid this, run the container by specifying your user's userid:

docker run -u $(id - u)$:(id -g) args...

will fire up a docker container with access to all the GPUs of the host

system, /notebooks directory of the host acting as the /tf/notebooks directory

of the container and 8888 ports of the host and the container bridged (Jupyter

notebooks work on port 8888 by default).

sudo docker run -it –rm –gpus all -v $(realpath /notebooks):/tf/notebooks -p 8888:8888 tensorflow/tensorflow:2.2.0-gpu-jupyter

Enable memory growth on GPU:

One final thing need to be done to make sure that TensorFlow won't run out of

GPU memory during model training. This is to enable memory growth on GPU,

which would stop TensorFlow from allocating the entire GPU memory on initialization.

import tensorflow as tf physical_devices = tf.config.list_physical_devices('GPU') tf.config.experimental.set_memory_growth(physical_devices[0], True)

:: To try something more ambitious, you can run an Ubuntu container with: :: $ docker run -it ubuntu bash