DAY 1

**O que é Container?**

O container faz um isolamento.

Os namespace cuidam da parte lógica(usuários, permissões, etc)

Os cgroup cuidam da parte fisica(recurso de RAM, CPU, I/O,etc)

**O que é Docker? – Parte 01**

O Docker não consegue interagir entre camadas RO(Read-only), elas não conseguem escrever na outra devido ser somente leitura.

**O que é o Docker? - Parte 02**

Iptables que encaminha as comunicações aos services do Docker.

Namespaces, cgroup

## Instalando o Docker

<https://docs.docker.com/engine/install/>

Docker CE = Gratuito

Comandos Utilizados:

# curl -fsSL https://get.docker.com/ | bash

# docker version

# docker container ls

- Adicionando usuário que vai ter permissão para administrar containers e etc:

usermod -aG docker fernando

-Mesmo seguindo os passos da página do Docker para instalar o Docker e usando instruções do LinuxTips, dava erro:

root@ubuntulab:/gitlab/projeto-aula2# docker run hello-world

bash: /snap/bin/docker: No such file or directory

-Foi necessário instalar usando o Snap:

<https://snapcraft.io/install/docker/ubuntu>

snap install docker

- Adicionando usuário que vai ter permissão para administrar containers e etc:

usermod -aG <grupo> <usuario>

usermod -aG docker fernando

## Executando e administrando containers Docker

-Comando antigo:

docker ps

**-Novo Comando, que deve ser usado:**

docker container ls

root@ubuntulab:/gitlab/projeto-aula2# **docker container ls**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

root@ubuntulab:/gitlab/projeto-aula2#

docker container run -ti hello-world

-ti = terminal + interatividade

**- O Comando container ls com o parametro -a mostra até os containers parados:**

root@ubuntulab:/gitlab/projeto-aula2# docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

fd309ba70035 hello-world "/hello" 14 seconds ago Exited (0) 13 seconds ago dazzling\_mccarthy

54e6b11005b2 hello-world "/hello" 17 minutes ago Exited (0) 17 minutes ago cool\_engelbart

root@ubuntulab:/gitlab/projeto-aula2#

docker container run -ti ubuntu

root@4dd41710cc29:/# ps -ef

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 08:04 pts/0 00:00:00 bash

root 9 1 0 08:04 pts/0 00:00:00 ps -ef

root@4dd41710cc29:/#

root@4dd41710cc29:/# cat /etc/issue

Ubuntu 20.04.2 LTS \n \l

root@4dd41710cc29:/#

- CTRL+D para matar o container / sair do container

-Principal entrypoint do ubuntu é o bash.

root@ubuntulab:/gitlab/projeto-aula2# docker container run -ti centos

Unable to find image 'centos:latest' locally

latest: Pulling from library/centos

7a0437f04f83: Pull complete

Digest: sha256:5528e8b1b1719d34604c87e11dcd1c0a20bedf46e83b5632cdeac91b8c04efc1

Status: Downloaded newer image for centos:latest

[root@9bb08d588a24 /]# cat /etc/redhat-release

CentOS Linux release 8.3.2011

[root@9bb08d588a24 /]#

**-Comando indicado para sair do container SEM matar ele:**

CTRL+P+Q

**- Para conectar a um container usando o seu id é necessário usar o comando attach:**

docker container attach <id-do-container>

docker container attach 9bb08d588a24

## Attach local standard input, output, and error streams to a running container

-O Attach só mostra o conteúdo(input, output, error) rodando no container:

root@ubuntulab:/home/fernando# docker container attach 6379b2854eb2

127.0.0.1 - - [23/Jul/2021:04:51:24 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.64.0" "-"

-O NGINX não tem um entrypoint, tipo o bash que nem o Ubuntu tem, o entrypoint do nginx é o próprio processo do nginx.

Então, ao conectar no nginx usando a opção -ti, vai cair direto no processo do nginx.

**- Ao usar a opção -ti ele vai rodar o nginx na tela/em primeiro plano. Para sair aperte CTRL+C:**

docker container run -ti nginx

root@ubuntulab:/gitlab/projeto-aula2# docker container run -ti nginx

Unable to find image 'nginx:latest' locally

latest: Pulling from library/nginx

b4d181a07f80: Pull complete

2021/07/15 08:30:03 [notice] 1#1: start worker processes

2021/07/15 08:30:03 [notice] 1#1: start worker process 30

2021/07/15 08:30:03 [notice] 1#1: start worker process 31

**-Saindo com o CTRL+C, o container fecha, quando iniciado usando -ti:**

root@ubuntulab:/gitlab/projeto-aula2# docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

7b4e64977204 nginx "/docker-entrypoint.…" About a minute ago Exited (0) 2 seconds ago vigorous\_hawking

**-Para que o container siga rodando EM SEGUNDO PLANO é necessário subir ele com um daemon, usando o -d:**

docker container run -d nginx

root@ubuntulab:/gitlab/projeto-aula2# docker container run -d nginx

c76c21d809c9b1788634f8fb0edead49fdd2d43a3967cebdac849813ca03aaac

root@ubuntulab:/gitlab/projeto-aula2#

root@ubuntulab:/gitlab/projeto-aula2# docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

c76c21d809c9 nginx "/docker-entrypoint.…" 5 minutes ago Up 5 minutes 80/tcp zealous\_hamilton

**-Para executar comandos remotamente, usar o exec:**

docker container exec -ti c76c21d809c9 ls /

root@ubuntulab:/gitlab/projeto-aula2# docker container exec -ti c76c21d809c9 ls /

bin docker-entrypoint.d home media proc sbin tmp

boot docker-entrypoint.sh lib mnt root srv usr

dev etc lib64 opt run sys var

root@ubuntulab:/gitlab/projeto-aula2#

docker container exec -ti 6379b2854eb2 ls /usr/share/nginx/html

root@ubuntulab:/home/fernando# docker container exec -ti 6379b2854eb2 ls /usr/share/nginx/html

50x.html index.html

root@ubuntulab:/home/fernando#

**-Para conectar no container usando o exec, passar o bash ao final do comando:**

docker container exec -ti c76c21d809c9 bash

root@ubuntulab:/gitlab/projeto-aula2# docker container exec -ti c76c21d809c9 bash

root@c76c21d809c9:/#

-NO CASO DO EXEC O CONTAINER SEGUE EM EXECUÇÃO, MESMO SAINDO:

root@ubuntulab:/home/fernando# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

6379b2854eb2 nginx "/docker-entrypoint.…" 2 hours ago Up 2 hours 80/tcp wonderful\_beaver

root@ubuntulab:/home/fernando#

root@ubuntulab:/home/fernando# docker container exec -ti 6379b2854eb2 bash

root@6379b2854eb2:/#

root@6379b2854eb2:/#

root@6379b2854eb2:/#

root@6379b2854eb2:/#

root@6379b2854eb2:/#

root@6379b2854eb2:/# ^C

root@6379b2854eb2:/# exit

root@ubuntulab:/home/fernando#

root@6379b2854eb2:/# exit

root@ubuntulab:/home/fernando# docker container exec -ti 6379b2854eb2 bash

root@6379b2854eb2:/# apt-get update

Get:1 http://deb.debian.org/debian buster InRelease [122 kB]

Get:2 http://deb.debian.org/debian buster-updates InRelease [51.9 kB]

Get:3 http://deb.debian.org/debian buster/main amd64 Packages [7907 kB]

Get:4 http://security.debian.org/debian-security buster/updates InRelease [65.4 kB]

Get:5 http://security.debian.org/debian-security buster/updates/main amd64 Packages [297 kB]

Get:6 http://deb.debian.org/debian buster-updates/main amd64 Packages [15.2 kB]

Fetched 8458 kB in 3s (3137 kB/s)

Reading package lists... Done

root@6379b2854eb2:/# apt-get install curl

Reading package lists... Done

Building dependency tree

Reading state information... Done

curl is already the newest version (7.64.0-4+deb10u2).

0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.

root@6379b2854eb2:/#

root@6379b2854eb2:/# curl localhost

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

body {

width: 35em;

margin: 0 auto;

font-family: Tahoma, Verdana, Arial, sans-serif;

}

</style>

</head>

<body>

<h1>Welcome to nginx!</h1>

<p>If you see this page, the nginx web server is successfully installed and

working. Further configuration is required.</p>

<p>For online documentation and support please refer to

<a href="http://nginx.org/">nginx.org</a>.<br/>

Commercial support is available at

<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>

</body>

</html>

root@6379b2854eb2:/#

-NO CASO DO EXEC O CONTAINER SEGUE EM EXECUÇÃO, MESMO SAINDO:

root@ubuntulab:/home/fernando# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

6379b2854eb2 nginx "/docker-entrypoint.…" 2 hours ago Up 2 hours 80/tcp wonderful\_beaver

root@ubuntulab:/home/fernando#

**-Verificando o tipo de Storage Driver no Docker:**

root@ubuntulab:/home/fernando# docker info | grep "Storage Driver"

WARNING: No swap limit support

Storage Driver: overlay2

root@ubuntulab:/home/fernando#

**-Formas de montar o container:**

docker container run -ti nginx

docker container run -d nginx

docker container exec -ti nginx bash

-VIDEO

## Executando e administrando containers Docker - parte2

docker container stop nginx

root@ubuntulab:/home/fernando# docker container inspect elegant\_goldberg

[

{

"Id": "61a810385fb57cff162db59b089b39c0058faa745526bd0798cc978a62dc951c",

"Created": "2021-08-10T00:41:56.189426373Z",

"Path": "/docker-entrypoint.sh",

"Args": [

"nginx",

"-g",

"daemon off;"

**-Este valor da memória em 0 indica que ele pode usar toda a memória da máquina:**

root@ubuntulab:/home/fernando# docker container inspect elegant\_goldberg | grep emory

] "Memory": 0,

"KernelMemory": 0,

"KernelMemoryTCP": 0,

"MemoryReservation": 0,

"MemorySwap": 0,

"MemorySwappiness": null,

root@ubuntulab:/home/fernando#

- O comando docker inspect retorna ainda os pontos de montagem, variavéis de ambiente, sinal de parada(SIGTERM, SIGKILL, etc...)

-Se for usado o docker inspect para verificar o ip, é possível testar o nginx usando o curl para o ip:

root@ubuntulab:/home/fernando# docker container inspect elegant\_goldberg | grep IPA

"SecondaryIPAddresses": null,

"IPAddress": "172.17.0.2",

"IPAMConfig": null,

"IPAddress": "172.17.0.2",

root@ubuntulab:/home/fernando#

root@ubuntulab:/home/fernando# curl "172.17.0.2"

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

[...]

-Executando o container do NGINX e alterando o conteúdo do index.html para um outro texto:

**Diretório do index.html no NGINX:**

**/usr/share/nginx/html/index.html**

docker container exec -ti elegant\_goldberg bash

echo “TESTANDO A Página do NGINX, apenas isto.” > /usr/share/nginx/html/index.html

root@61a810385fb5:/# curl localhost

“TESTANDO A Página do NGINX, apenas isto.”

root@61a810385fb5:/#

- Efetuar o PAUSE do Container:

root@ubuntulab:/home/fernando# **docker container pause elegant\_goldberg**

elegant\_goldberg

root@ubuntulab:/home/fernando# curl "172.17.0.2"

^C

root@ubuntulab:/home/fernando# ^C

root@ubuntulab:/home/fernando# docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

61a810385fb5 nginx "/docker-entrypoint.…" 31 minutes ago Up 31 minutes (Paused) 80/tcp elegant\_goldberg

- Efetuar o UNPAUSE do Container:

root@ubuntulab:/home/fernando# docker container ls -a | grep nginx

61a810385fb5 nginx "/docker-entrypoint.…" 33 minutes ago Up 33 minutes (Paused) 80/tcp elegant\_goldberg

root@ubuntulab:/home/fernando#

root@ubuntulab:/home/fernando# **docker container unpause elegant\_goldberg**

elegant\_goldberg

root@ubuntulab:/home/fernando# docker container ls -a | grep nginx

61a810385fb5 nginx "/docker-entrypoint.…" 33 minutes ago Up 33 minutes 80/tcp elegant\_goldberg

root@ubuntulab:/home/fernando#

root@ubuntulab:/home/fernando# curl "172.17.0.2"

“TESTANDO A Página do NGINX, apenas isto.”

root@ubuntulab:/home/fernando#

- Formar de coletar logs do Container:

root@ubuntulab:/home/fernando# **docker logs -f elegant\_goldberg**

172.17.0.1 - - [10/Aug/2021:01:22:57 +0000] "GET / HTTP/1.1" 200 48 "-" "curl/7.68.0" "-"

172.17.0.1 - - [10/Aug/2021:01:22:59 +0000] "GET / HTTP/1.1" 200 48 "-" "curl/7.68.0" "-"

root@ubuntulab:/home/fernando# **docker attach elegant\_goldberg**

172.17.0.1 - - [10/Aug/2021:01:23:47 +0000] "GET / HTTP/1.1" 200 48 "-" "curl/7.68.0" "-"

- Formar de Deletar o Container, opção -f vai forçar a remoção:

root@ubuntulab:/home/fernando# docker container rm b39687352b8e

Error response from daemon: You cannot remove a running container b39687352b8e70e9cefafaba72f20c6f2ec328727450bc5a597ebe7954730dde. Stop the container before attempting removal or force remove

root@ubuntulab:/home/fernando# docker container rm -f b39687352b8e

b39687352b8e

também é possível deletar um Container informando apenas os digitos iniciais dele:

root@ubuntulab:/home/fernando# docker container rm -f 2fe

## Configurando CPU e memória para os meus containers

root@ubuntulab:/home/fernando# **docker container stats 2644770c9be3**

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

2644770c9be3 determined\_hertz 0.00% 4.25MiB / 8.301GiB 0.05% 1.05kB / 0B 0B / 16.4kB 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

2644770c9be3 determined\_hertz 0.00% 4.25MiB / 8.301GiB 0.05% 1.05kB / 0B 0B / 16.4kB 3

^C

root@ubuntulab:/home/fernando#

-Stressando o container com o comando stress:

root@2644770c9be3:/# **stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --timeout 10s**

stress: info: [321] dispatching hogs: 8 cpu, 4 io, 2 vm, 0 hdd

stress: info: [321] successful run completed in 10s

root@2644770c9be3:/#

root@ubuntulab:/home/fernando# **docker container stats 2644770c9be3**

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

2644770c9be3 determined\_hertz 0.57% 217.4MiB / 8.301GiB 2.56% 8.59MB / 96.3kB 26MB / 24MB 19

root@2644770c9be3:/# **stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --timeout 10s**

stress: info: [351] dispatching hogs: 8 cpu, 4 io, 2 vm, 0 hdd

root@ubuntulab:/home/fernando# **docker container top 2644770c9be3**

UID PID PPID C STIME TTY TIME CMD

root 130943 130919 0 01:57 ? 00:00:00 nginx: master process nginx -g daemon off;

systemd+ 131022 130943 0 01:57 ? 00:00:00 nginx: worker process

systemd+ 131023 130943 0 01:57 ? 00:00:00 nginx: worker process

root 147401 130919 0 02:09 pts/0 00:00:00 bash

root 154568 147401 0 02:14 pts/0 00:00:00 stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --timeout 10s

root 154569 154568 13 02:14 pts/0 00:00:00 stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --timeout 10s

root 154570 154568 14 02:14 pts/0 00:00:00 stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --timeout 10s

- Executar um Container usando uma quantidade especifica de **memória**:

docker container run -d -m 128M nginx

docker container ls

docker inspect a8d097124584

root@ubuntulab:/home/fernando# docker inspect a8d097124584 | grep emory

"Memory": 134217728,

root@a8d097124584:/# stress --cpu 8 --io 4 --vm 2 --vm-bytes 1528M --timeout 10s

stress: info: [377] dispatching hogs: 8 cpu, 4 io, 2 vm, 0 hdd

docker container stats a8d097124584

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

a8d097124584 objective\_nash 170.69% 127.9MiB / 128MiB 99.92% 8.57MB / 74.5kB 947MB / 49.5MB 19

**-Comando que executa o curl repetidamente:**

while true; do curl 172.17.0.2; done

root@ubuntulab:~# while true; do curl 172.17.0.2; done

## Meu primeiro e tosko Dockerfile

# docker image build -t toskeira:1.0

# docker image ls

# docker container run -d toskeira:1.0

# docker container logs -f [CONTAINER ID]

No Dockerfile:

FROM debian

LABEL app="Giropops"

ENV JEFERSON="LINDO"

RUN apt-get update && apt-get install -y stress && apt-get clean

CMD stress --cpu 1 --vm-bytes 64M --vm1

O FROM define a imagem que será usada como base.

O RUN é sobre os comandos passados durante o build.

O CMD é referente os comandos que são executados depois que a imagem é buildada.

Lembrando que é necessário informar o path no docker build, além da tag que é informada.

docker image build -t toskeira:1.0 **.**

html

# docker image build -t toskeira:1.0 .

# docker image ls

# docker container run -d toskeira:1.0

# docker container logs -f [CONTAINER ID]

No Dockerfile:

FROM debian

LABEL app="Giropops"

ENV JEFERSON="LINDO"

RUN apt-get update && apt-get install -y stress && apt-get clean

CMD stress --cpu 1 --vm-bytes 64M

#############################################################################

#############################################################################

########### Buildando a imagem

-t indica a tag, neste caso a tag é toskeira, dois pontos, a versão dela

. o ponto indica que o Dockerfile está neste nível

docker image build -t toskeira:1.0 .

-Container tava apresentando erro e não ficava up:

root@ubuntulab:~# docker container logs -f gifted\_brown

stress: FAIL: [8] (244) unrecognized option: --vm1

root@ubuntulab:~# docker container rm gifted\_brown

gifted\_brown

root@ubuntulab:~#

-Foi necessário remover o parametro --vm1 do Dockerfile, buildar novamente e subir o Container.

root@ubuntulab:/curso/descomplicando-o-docker# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

15836568f5d6 toskeira:1.0 "/bin/sh -c 'stress …" About a minute ago Up 59 seconds epic\_blackburn

root@ubuntulab:/curso/descomplicando-o-docker#

-Verificando os recursos usando o comando "docker container stats epic\_blackburn":

root@ubuntulab:/curso/descomplicando-o-docker# ^C

root@ubuntulab:/curso/descomplicando-o-docker# docker container stats epic\_blackburn

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

15836568f5d6 epic\_blackburn 3.30% 1.945MiB / 8.301GiB 0.02% 1.09kB / 0B 0B / 0B 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

15836568f5d6 epic\_blackburn 3.30% 1.945MiB / 8.301GiB 0.02% 1.09kB / 0B 0B / 0B 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

15836568f5d6 epic\_blackburn 100.78% 1.945MiB / 8.301GiB 0.02% 1.09kB / 0B 0B / 0B 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

15836568f5d6 epic\_blackburn 100.78% 1.945MiB / 8.301GiB 0.02% 1.09kB / 0B 0B / 0B 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

-Subindo o container com apenas 64M de RAM, ele não vai conseguir fazer uso do Stress, que foi definido para stressar com 64M:

docker container run -d -m 64M toskeira:1.0

root@ubuntulab:/curso/descomplicando-o-docker# docker container run -d -m 64M toskeira:1.0

WARNING: Your kernel does not support swap limit capabilities or the cgroup is not mounted. Memory limited without swap.

acca8ef82f581dd1c53a17ae8d44db5707aaad942f2a2537567aac419ebf1ea6

root@ubuntulab:/curso/descomplicando-o-docker#

root@ubuntulab:/curso/descomplicando-o-docker# docker container stats agitated\_wu

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

acca8ef82f58 agitated\_wu 0.80% 1.945MiB / 64MiB 3.04% 1.09kB / 0B 0B / 0B 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

acca8ef82f58 agitated\_wu 0.80% 1.945MiB / 64MiB 3.04% 1.09kB / 0B 0B / 0B 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

acca8ef82f58 agitated\_wu 100.47% 1.945MiB / 64MiB 3.04% 1.09kB / 0B 0B / 0B 3

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

**-Subindo a máquina com 256M e com o Stress configurado para forçar mais a máquina:**

root@ubuntulab:/curso/descomplicando-o-docker# cat Dockerfile

FROM debian

LABEL app="Giropops"

ENV JEFERSON="LINDO"

RUN apt-get update && apt-get install -y stress && apt-get clean

CMD stress --cpu 8 --io 4 --vm 2 --vm-bytes 158M

root@ubuntulab:/curso/descomplicando-o-docker#

docker container run -d -m 256M toskeira:1.0

docker image ls

docker image build -t toskeira:1.0 .

docker image ls

docker container run -d -m 256M toskeira:1.0

root@ubuntulab:/curso/descomplicando-o-docker# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

aa58e4779b96 toskeira:1.0 "/bin/sh -c 'stress …" 6 seconds ago Up 5 seconds sharp\_greider

root@ubuntulab:/curso/descomplicando-o-docker#

root@ubuntulab:/curso/descomplicando-o-docker# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

aa58e4779b96 toskeira:1.0 "/bin/sh -c 'stress …" 6 seconds ago Up 5 seconds sharp\_greider

root@ubuntulab:/curso/descomplicando-o-docker# docker container stats sharp\_greider

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

aa58e4779b96 sharp\_greider 4.04% 256MiB / 256MiB 99.99% 1.09kB / 0B 1.56GB / 4.81MB 16

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

aa58e4779b96 sharp\_greider 4.04% 256MiB / 256MiB 99.99% 1.09kB / 0B 1.56GB / 4.81MB 16

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

aa58e4779b96 sharp\_greider 740.46% 255.7MiB / 256MiB 99.89% 1.09kB / 0B 1.67GB / 5.23MB 16