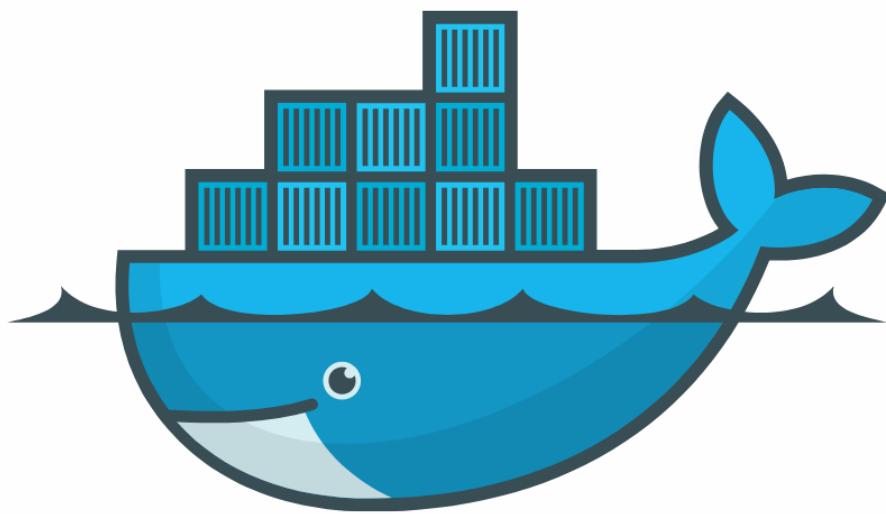


CONTENEDORES: LIMITANDO RECURSOS

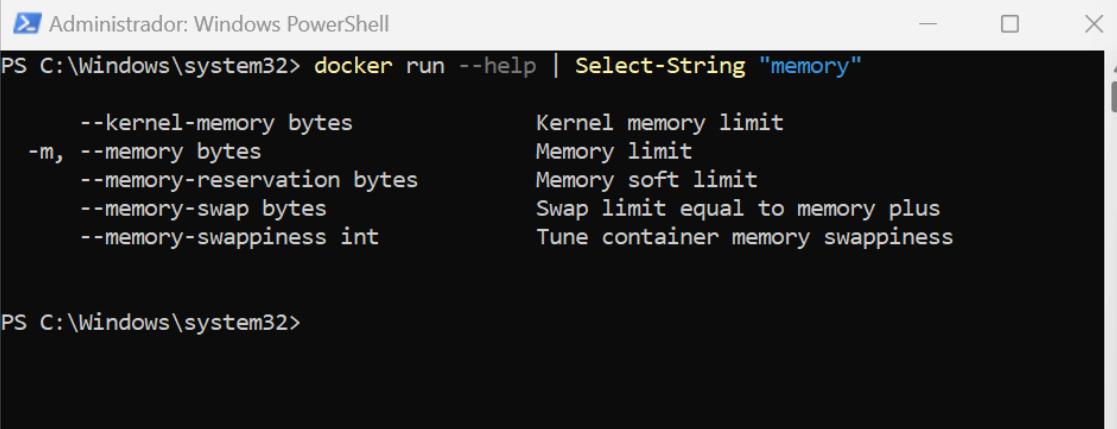


Ivana Sánchez Pérez

Investigar cómo limitar los recursos

Limitar los recursos de Docker es una práctica útil para asegurarnos que los contenedores no consuman más recursos de los necesarios, lo que puede ser especialmente importante en entornos de producción o cuando se ejecutan múltiples contenedores en la misma máquina.

Primero, podemos usar el comando `docker run --help | Select-String "memory"` para buscar las opciones relacionadas con la limitación de memoria. Este comando mostrará las opciones disponibles para limitar la memoria, como `-memory` o `-m`.



```
PS C:\Windows\system32> docker run --help | Select-String "memory"

--kernel-memory bytes          Kernel memory limit
-m, --memory bytes             Memory limit
--memory-reservation bytes     Memory soft limit
--memory-swap bytes            Swap limit equal to memory plus
--memory-swappiness int        Tune container memory swappiness

PS C:\Windows\system32>
```

Limitar la memoria RAM

Para limitar la memoria RAM que un contenedor puede usar, podemos usar la opción `--memory` o `-m`, con el contenedor parado. Por ejemplo, si queremos limitar un contenedor a 512 MB de RAM, ejecutaremos `docker run -it --memory="512m" <imagen>`:

- `-it` → para ejecutar el contenedor en modo interactivo con una terminal
- `--memory="512m"` → limita la memoria a 512 MB.
- `<imagen>` → la imagen que queremos limitar

Limitar el uso de CPU

Para limitar el uso de CPU, podemos usar la opción `--cpus`. Por ejemplo, si queremos limitar el contenedor a usar sólo 1CPU, ejecutaremos `docker run -it --cpus="1" <imagen>`

Combinar límites de memoria y CPU

Combinaremos ambas opciones en un mismo comando de la siguiente forma: `docker run -it --memory="512m" --cpus="0,5" <imagen>`.

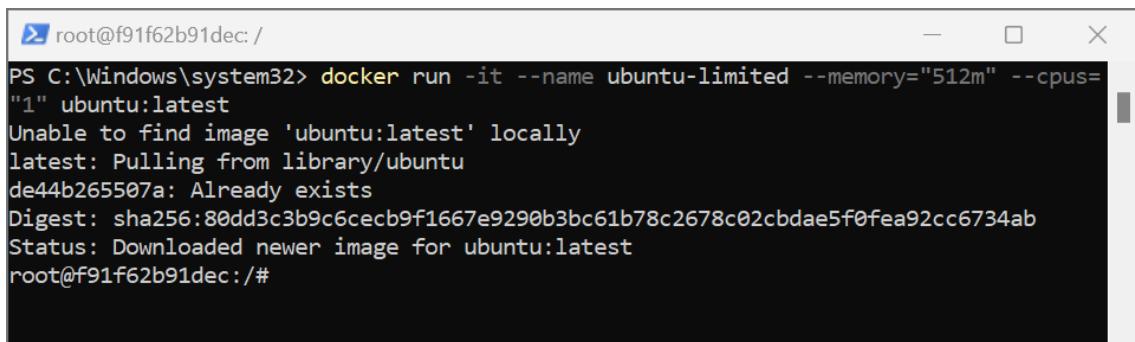
Verificar los límites

Para verificar que los límites se han ampliado correctamente, usaremos el comando `docker stats` mientras el contenedor está en ejecución. Este comando nos mostrará el uso de recursos en tiempo real. El comando se ejecutará `docker stats <imagen>`.

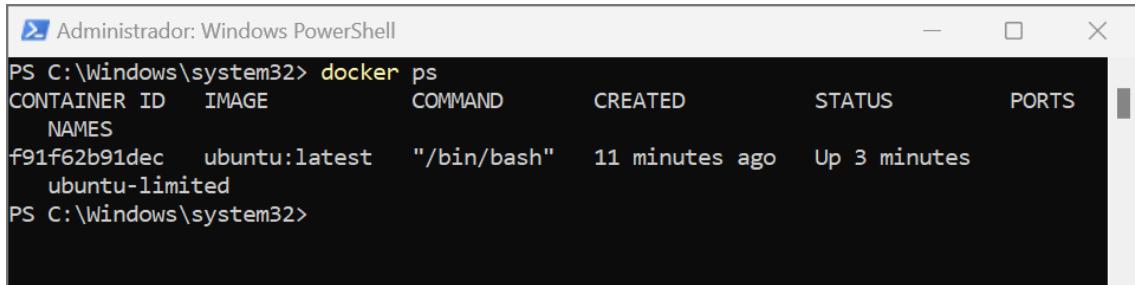
Ejemplo completo

Vamos a iniciar un nuevo contenedor de Ubuntu con límites de recurso: memoria a 512 MB y su CPU a 1. El proceso sería el siguiente:

- a- Ejecutamos el contenedor con los límites: `docker run -it --name ubuntu-limited --memory="512m" --cpus="1" ubuntu:latest`



```
PS C:\Windows\system32> docker run -it --name ubuntu-limited --memory="512m" --cpus="1" ubuntu:latest
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
de44b265507a: Already exists
Digest: sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Downloaded newer image for ubuntu:latest
root@f91f62b91dec:/#
```



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
f91f62b91dec	ubuntu:latest	"/bin/bash"	11 minutes ago	Up 3 minutes	
	ubuntu-limited				

- b- En otro terminal (ya que tenemos que estar en la máquina anfitriona, y no dentro del contenedor) verificamos los recursos utilizados por el contenedor con `docker inspect ubuntu-limited` nos mostrará todas las limitaciones.

```

PS C:\Windows\system32> docker inspect ubuntu-limited
[{"Id": "f91f62b91dec2528b22c9d9a472d7f760710f17f6bbc65fd9f1fd4686022bac6", "Created": "2025-01-29T11:56:47.450030687Z", "Path": "/bin/bash", "Args": [], "State": { "Status": "running", "Running": true, "Paused": false, "Restarting": false, "OOMKilled": false, "Dead": false, "Pid": 589, "ExitCode": 0, "Error": "", "StartedAt": "2025-01-29T12:04:52.267033115Z", "FinishedAt": "2025-01-29T11:59:26.99449198Z" }, "Image": "sha256:b1d9df8ab1559494794e522b388878cf9ba82d4c1fb67293bcf931c3aa69ae4", "ResolvConfPath": "/var/lib/docker/containers/f91f62b91dec2528b22c9d9a472d7f760710f17f6bbc65fd9f1fd4686022bac6/resolv.conf", "HostNamePath": "/var/lib/docker/containers/f91f62b91dec2528b22c9d9a472d7f760710f17f6bbc65fd9f1fd4686022bac6/hostname", "HostPath": "/var/lib/docker/containers/f91f62b91dec2528b22c9d9a472d7f760710f17f6bbc65fd9f1fd4686022bac6/hosts", "LogPath": "/var/lib/docker/containers/f91f62b91dec2528b22c9d9a472d7f760710f17f6bbc65fd9f1fd4686022bac6/f91f62b91dec2528b22c9d9a472d7f760710f17f6bbc65fd9f1fd4686022bac6-json.log", "Name": "/ubuntu-limited", "RestartCount": 0, "Driver": "overlay2", "Platform": "linux", "MountLabel": "", "ProcessLabel": "", "AppArmorProfile": "", "ExecIDs": null, "HostConfig": {} }
    ],
    "CapAdd": null,
    "CapDrop": null,
    "CgroupsMode": "host",
    "Dns": [],
    "DnsOptions": [],
    "DnsSearch": [],
    "ExtraHosts": null,
    "GroupAdd": null,
    "IpcMode": "private",
    "Cgroup": "",
    "Links": null,
    "OomScoreAdj": 0,
    "PidMode": "",
    "Privileged": false,
    "PublishAllPorts": false,
    "ReadonlyRootfs": false,
    "SecurityOpt": null,
    "UTSMode": "",
    "UsernsMode": "",
    "ShmSize": 67108864,
    "Runtime": "runc",
    "Isolation": "",
    "CpusShares": 0,
    "Memory": 536870912,
    "NanoCpus": 1000000000,
    "CgroupParent": "",
    "Blkioweight": 0,
    "BlkioweightDevice": [],
    "BlkioDeviceReadBps": [],
    "BlkioDeviceWriteBps": [],
    "BlkioDeviceReadIOps": [],
    "BlkioDeviceWriteIOps": [],
    "CpuPeriod": 0,
    "CpuQuota": 0,
    "CpuRealtimePeriod": 0,
    "CpuRealtimeRuntime": 0,
    "CpusetCpus": "",
    "CpusetMems": "",
    "Devices": []
}

```

c- Y con docker ***stats ubuntu-limited*** observaremos el uso de los recursos del contenedor en tiempo real.

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O
BLOCK I/O	PIDS				
f91f62b91dec	ubuntu-limited	0.00%	884KiB / 512MiB	0.17%	1.05kB / 0B
0B / 0B	1				

Modificación de un contenedor

Debemos saber que Docker no permite modificar directamente la memoria y CPU de un contenedor. Éste se debe parar si está en ejecución, eliminar y volver a crear con los nuevos límites.

- docker ps -as → nos listará los contenedores existentes
- docker stop <nombre o id contenedor> → detendrá el contenedor
- docker rm <nombre o id contenedor> → eliminará el contenedor
- docker run -dit --name <nombre contenedor> --memory=Xm --cpus=x <imagen>:lastet

