

목차

- docker 할당자원 제어 3
 - 컨테이너 메모리 제한 3
 - 컨테이너 CPU 제한 4
 - Block I/O 제한 6
 - 참조링크 6

docker 할당자원 제어

컨테이너를 생성하는 run, create 명령어에서 컨테이너의 자원 할당량을 조정하도록 옵션을 입력할 수 있습니다. 아무런 옵션을 입력하지 않으면 호스트의 자원을 제한 없이 사용할 수 있습니다. 하지만 모든 자원을 사용하게 된다면 호스트에 영향을 끼칠 뿐더러 다른 컨테이너에도 영향을 끼치게 됩니다.

컨테이너의 자원제한을 확인할 수 있는 방법은 inspect 명령어를 사용하는 것입니다.

```
# docker inspect test_ubuntu

    "CpuShares": 0,
    "Memory": 0,
    "NanoCpus": 0,
    "CgroupParent": "",
    "BlkioWeight": 0,
    "BlkioWeightDevice": [],
    "BlkioDeviceReadBps": null,
    "BlkioDeviceWriteBps": null,
    "BlkioDeviceReadIOps": null,
    "BlkioDeviceWriteIOps": null,
    "CpuPeriod": 0,
    "CpuQuota": 0,
    "CpuRealtimePeriod": 0,
    "CpuRealtimeRuntime": 0,
    "CpusetCpus": "",
    "CpusetMems": "",
    "Devices": [],
    "DeviceCgroupRules": null,
    "DeviceRequests": null,
    "KernelMemory": 0,
    "KernelMemoryTCP": 0,
    "MemoryReservation": 0,
    "MemorySwap": 0,
    "MemorySwappiness": null,
    "OomKillDisable": false,
    "PidsLimit": null,
    "Ulimits": null,
    "CpuCount": 0,
    "CpuPercent": 0,
    .....
```

컨테이너 메모리 제한

docker run 명령어에 `-memory`를 사용하여 컨테이너의 메모리를 제한할 수 있습니다. 입력할 수 있는 단위는 m(megabyte), g(gigabyte)이며 제한할 수 있는 최소 메모리는 4MB입니다.

```
docker run -d \
--memory="1g" \
--name memory_1g \
nginx
```

위와 같은 명령어 실행 시 1g의 용량을 가진 컨테이너를 제작합니다. 그리고 할당된 메모리를 확인해보면 1g가 할당된 것을 알 수 있습니다.

컨테이너를 사용하다가 컨테이너에 할당된 메모리를 초과한다면 컨테이너는 자동으로 종료되므로 메모리를 적절히 할당하는 것이 좋습니다. 추가적으로 swap 메모리도 설정할 수 있습니다. 보통은 memory 의 2배 정도 설정되는데 `-memory-swap`을 사용하여 바꿀 수 있습니다.

컨테이너 CPU 제한

- `--cpu-shares`

컨테이너에 가중치를 설정해 해당 컨테이너가 CPU를 상대적으로 얼마나 사용할 수 있는지 설정할 수 있습니다. cpu 코어의 개수를 할당하는 것이 아닌 시스템에 존재하는 CPU를 어느 비중만큼 나눠 쓸 것인지 명시하는 옵션입니다.

```
docker run -it --name cpu_share \
--cpu-shares 1024 \
ubuntu:14.04
```

`--cpu-shares` 옵션은 상대적인 값을 가집니다. 아무런 설정을 하지 않았을 때 설정 값은 1024로 이는 할당량 1을 뜻합니다.

그럼 실제로 어떻게 작동하는지 확인하기 위해 cpu의 과부하를 주는 컨테이너를 만들어보겠습니다.

```
docker run -d --name cpu_1024 \
--cpu-shares 1024 \
alicek106/stress \
stress --cpu 1

docker run -d --name cpu_512 \
--cpu-shares 512 \
alicek106/stress \
stress --cpu 1
```

위와 같이 2:1 비율로 나누어 주어서 cpu 사용량을 설정할 수 있습니다.

- `--cpuset-cpu` ' ' 호스트에 CPU가 여러 개 있을 때 ' ' `--cpuset-cpus` 옵션을 지정해 컨테이너가 특정 CPU만 사용하도록 설정 할 수 있습니다. CPU 집중적인 작업이 필요하다면 여러 개의 CPU를 사용하도록 설정해 작업을 적절하게 분배하는 것이 좋습니다.

```
docker run -d --name cpuset_2 \  
--cpuset-cpus=2 \  
alicek106/stress \  
stress --cpu 1
```

위의 명령어는 3번째 cpu를 사용한다는 의미입니다. 컴퓨터 답게 0부터 시작합니다. 아래 htop를 사용하여 cpu 사용률을 확인할 수 있습니다.

```
mac: brew install htop  
ubuntu: apt-get install htop  
centos yum -y install epel-release && yum -y install htop
```

htop를 입력해주면 사용률을 확인할 수 있습니다.

- `-cpu-period`, `-cpu-quota`

`cpu-period`는 한 컨테이너에 일반적으로 100ms 할당되지만 `-cpu-period`를 통해 변경할 수 있다. 그리고 `-cpu-quota`를 통해 할당된 100ms동안 얼마나 cpu를 할당할지 결정할 수 있다.

```
docker run -d --name quota_1_4 \  
--cpu-period=100000 \  
--cpu-quota=25000 \  
alicek106/stress \  
stress --cpu 1
```

기본 100000에서 25000으로 줄었기 때문에 cpu성능이 1/4로 감소합니다. 즉 컨테이너는 `cpu-quota` / `cpu-period` 값으로 cpu시간을 할당 받습니다.

```
docker run -d --name quota_1_1 \  
--cpu-period=100000 \  
--cpu-quota=100000 \  
alicek106/stress \  
stress --cpu 1
```

위와 같이 period와 quota 값을 설정하여 cpu의 주기와 할당량을 설정할 수 있습니다.

- `-cpus`

`-cpus`는 `-cpu-quota`, `-cpu-period`와 동일한 기능을 하지만 좀 더 직관적으로 cpu의 개수를 직접 지정할 수 있습니다.

```
docker run -d --name cpus_container \  
--cpus=0.5 \  
alicek106/stress \  
stress --cpu 1
```

0.5는 cpu-quota와 cpu-period의 비율을 0.5로 설정하는 의미입니다. 그래서 여기에는 $50000(\text{quota})/100000(\text{period})$ 와 같은 뜻입니다.

Block I/O 제한

컨테이너를 생성할 때 아무런 옵션도 설정하지 않으면 내부에서 파일을 읽고 쓰는 대역폭에 제한이 설정되지 않습니다. 하나의 컨테이너가 너무 많은 입출력을 독점하여 사용하지 않게 하기 위해서는 `-device-write-bps`, `-device-read-bps`, `-device-write-iops`, `-device-read-iops` 옵션을 지정해 블록 입출력을 제한할 수 있습니다. 단 Direct I/O에서만 입출력이 제한되며 Buffered I/O는 제한되지 않습니다.

- Direct I/O: 운영체제의 버퍼를 거치지 않고 직접 IO를 실행
- Buffered I/O: 버퍼에 저장된 데이터를 I/O를 실행한다.

`-device-write-bps`, `-device-read-bps`는 각기 쓰고 읽는 작업의 초당 제한을 설정하며 kb, mb, gb 단위로 제한할 수 있습니다.

```
docker run -it \
--device-write-bps [디바이스 이름: 용량] \
ubuntu:14.04
```

아까진 cpu-shares에 상대적인 값을 작성했던 것 처럼 `-device-write-iops`, `-device-read-iops`에도 상대적인 값을 입력합니다.

```
docker run -it \
--device-write-iops 디바이스 이름:5 \
ubuntu:14.04
```

```
docker run -it \
--device-read-iops 디바이스 이름:10 \
ubuntu:14.04
// 2배 더 빠르게 실행
```

참조링크

- <https://hoony-gunputer.tistory.com/entry/docker-컨테이너-자원-할당-제한>

From:

<https://atl.kr/dokuwiki/> - AllThatLinux!

Permanent link:

https://atl.kr/dokuwiki/doku.php/docker_%ED%95%A0%EB%8B%B9%EC%9E%90%EC%9B%90_%EC%A0%9C%EC%96%B4

Last update: 2020/12/12 15:29

