1. Container Resource 제한하기
2. Container Monitoring 하기
3. cAdvisor 설치해서 사용하기

Task1. Linux의 부하테스트 : Stress
1. Stress Container 생성
   1)Container Build
      -부하 테스트 프로그램 Stress를 설치하고 동작시키는 Container Build하기

   2)CPU 부하테스트
      -2개의 CPU Core를 100% 사용하도록 부하 발생
         --stress --cpu 2

   3)Memory 부하테스트
      -Process 수 2개와 사용할 memory만큼 부하 발생
         --stress --vm 2 --vm-bytes <사용할 크기>

2. Dockerfile
   FROM debian
   MAINTAINER instructor <javaexpert@nate.com>
   RUN apt-get update && apt-get install stress -y
   CMD ["/bin/sh", "-c", "stress -c 2"]

3. lab
   $ mkdir build
   $ cd build
   $ ls
   Dockerfile

   $ docker build -t stress .
   다음의 Memery Resoure 테스트 후 Container 삭제할 것.

Task2. Memery Resource 제한
1. Swap Memory 용량 제한이 실제 메모리 제한과 어떤 관련성이 있는지 확인해본다.
2. 실행 명령
   $ docker run -m 100m --memory-swap 100m stress:latest stress --vm 1 --vm-bytes 90m -t 5s
      -100M memory에 90MB의 부하 발생 --> 가능
   $ docker run -m 100m --memory-swap 100m stress:latest stress --vm 1 --vm-bytes 150m -t 5s
      -100M memory에 150MB의 부하 발생 --> 바로 Kill된다.
   $ docker run -m 100m stress:latest stress --vm 1 --vm-bytes 150m -t 5s
      -Swap 메모리 생략...실행됨. 왜냐하면 생략시 메모리의 2배 할당되기 때문.
      -만일 VM의 Swap Memory가 없으면 fail됨.

3. Container를 OOM-Killer로부터 보호한다.
   1)OOM-Killer disable 설정하기
   2)실행명령
      $ docker run -d -m 100m --name m4 --oom-kill-disable=true nginx
      $ docker inspect m4
      -내용에서
         "HostConfig" > "Memory" : 104857600
         "HostConfig" > "MemorySwap" : -1 <---Swap memory가 없을 시
         "HostConfig" > "OomKillDisable" : true
      $ docker ps <---현재 Container의 containerID 파악할 것
      $ cat /sys/fs/cgroup/memory/docker/ContainerID/memory.oom_control
      a30/memory.oom_control
      oom_kill_disable 1   <--확인
      under_oom 0
      oom_kill 0

Task3. CPU Resource 제한
1. CPU 개수를 제한하여 Container를 실행한다 .
2. 실행명령
   $ lscpu   <--현재 머신의 CPU 갯수 확인
   $ docker build -t stress .
   $ docker run --cpuset-cpus 1 --name c1 -d stress:latest stress --cpu 1

$ htop <- 이 명령으로 현재 CPU가 2개이고 2번째 CPU가 100% 사용중임을 확인

$ docker run --cpuset-cpus 0-1 --name c2 -d stress stress --cpu 1
$ htop <--- CPU 1번과 2번 모두 100% 확인
$ docker rm c1

3. Container 별로 CPU 상대적 가중치를 할당하여 실행되도록 구성한다.
1)Container를 모두 제거
2)실행명령
-아래 4개의 명령 모두 실행
$ docker run -c 2048 --name cload1 -d stress:latest
$ docker run --name cload2 -d stress:latest
$ docker run -c 512 --name cload3 -d stress:latest
$ docker run -c 512 --name cload4 -d stress:latest

3)Container Resource 사용량 모니터하기
-위의 4개의 Container를 모두 실행시킨 후, 값으로 확인
$ docker stats
-CPU %를 보면 cload1이 100%기준으로 cload2는 50%, cload3와 cload4는 약 25% 사용하고 있는 것을 확인할 수 있다.
$ Ctrl + C로 종료

Task4. Block I/O 제한
1. Container에서 --device-write-iops를 적용해서 write 속도의 초당 Quota를 제한해서 IO Write를 발생시킨다.
2. 실행 명령
$ lsblk <-- device 이름 확인, xvda
$ docker run -it --rm --device-write-iops /dev/xvda:10 ubuntu:latest /bin/bash
/# dd if=/dev/zero of=file1 bs=1M count=10 oflag=direct
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 1.01016 s, 10.4 MB/s <--초당 약 10.4MB 속도

3. 다음 write quota를 100으로 변경 후 같은 작업을 반복한다 .
$ docker run -it --rm --device-write-iops /dev/xvda:100 ubuntu:latest /bin/bash
/# dd if=/dev/zero of=file1 bs=1M count=10 oflag=direct
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.0360691 s, 291 MB/s <--초당 약 291MB 속도

Task5. cAdvisor 실행하기
1. https://github.com/google/cadvisor 방문
2. Quick Start: Running cAdvisor in a Docker Container의 코드 복사
VERSION=v0.36.0 # use the latest release version from https://github.com/google/cadvisor/releases
sudo docker run \
--volume=/:/rootfs:ro \
--volume=/var/run:/var/run:ro \
--volume=/sys:/sys:ro \
--volume=/var/lib/docker/:/var/lib/docker:ro \
--volume=/dev/disk/:/dev/disk:ro \
--publish=8080:8080 \
--detach=true \
--name=cadvisor \
--privileged \
--device=/dev/kmsg \
gcr.io/cadvisor/cadvisor:$VERSION

3. 실습
$ docker run -it --rm --device-write-iops /dev/xvda:100 -m 500m --name c1 -d ubuntu:latest /bin/bash
-위의 CPU 리소스 제한에서 사용했던 cload1 ~ cload4까지 모두 실행
-그리고 방금 복사한 cAdvisor 코드 복사해서 실행할 것
-결과는 Web Broser 에서 확인
--http://101.79.11.221:8080

-또한 다음의 명령으로도 확인할 수 있다.

```
$ docker stats
```