**도커 체크포인트 사용하기 (우분투 20.04 기준)**

내용

[<도커 체크포인트 사용법> 2](#_Toc77779328)

[<도커에서 체크포인트 생성하기/단일머신> 3](#_Toc77779329)

[<디렉토리 지정하여 체크포인트 생성하기/다중머신> 5](#_Toc77779330)

[<Checkpoint가 persistent file을 복사하는지? 안함!> 6](#_Toc77779331)

[<단일머신 : 체크포인트로 소켓 프로그램 연속 서비스> 7](#_Toc77779332)

[<참고자료> 9](#_Toc77779333)

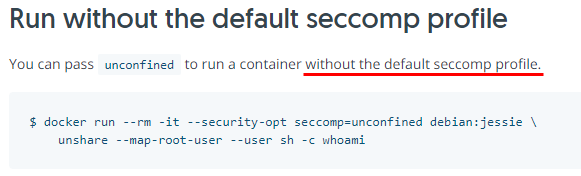
\*\* 도커 이미지는 latest 사용하지 말고, 특정한 버전의 이미지를 사용할 것. 나중에 재현이 어려워질 수 있음.

\*\* 우분투20.04 + 도커 최신버전의 경우 checkpoint가 정상적으로 동작하지 않음(채크포인트 사용은 되는데, 상태 복원이 안됨)

\*\* 우분투18.04 + 도커 17 버전의 경우 checkpoint가 정상적으로 동작함

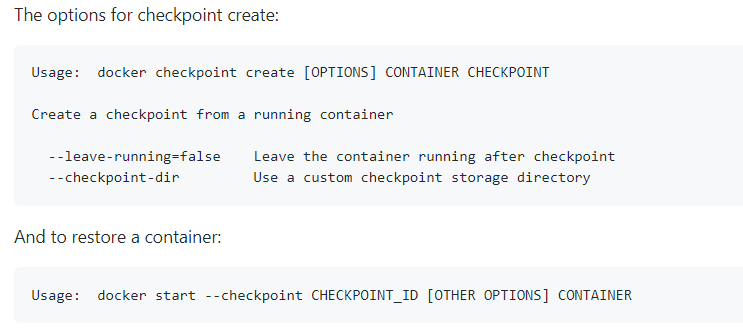
\*\* 볼륨 마운트하면 checkpoint 잘 안됨

\*\* seccomp profile (https://docs.docker.com/engine/security/seccomp/)



# <도커 체크포인트 사용법>

옵션들... (https://github.com/docker/cli/blob/master/experimental/checkpoint-restore.md)



공식 홈페이지 매뉴얼

<https://docs.docker.com/engine/reference/commandline/checkpoint/>

# <도커에서 체크포인트 생성하기/단일머신>

\*\* 참고: https://technology.amis.nl/platform/docker/first-steps-with-docker-checkpoint-to-create-and-restore-snapshots-of-running-containers/

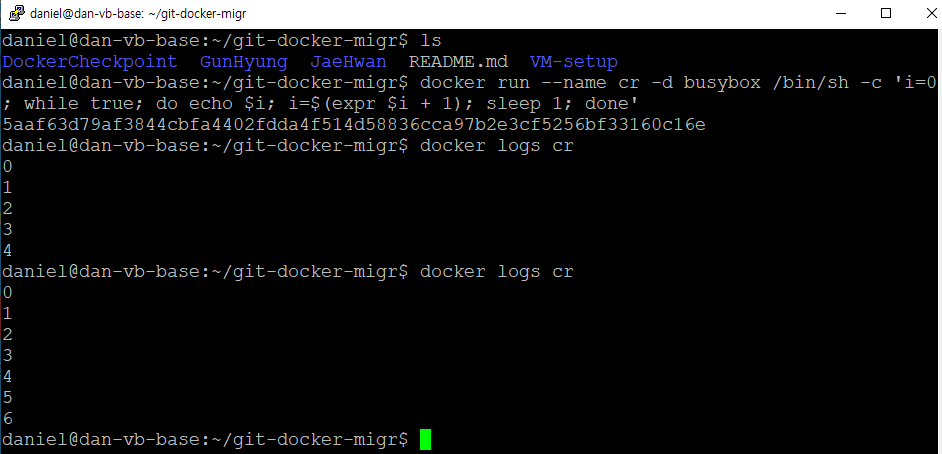
\*\* 참고: https://criu.org/Docker

1) 컨테이너 실행하고, 데몬으로 돌리고(터미널 연결하면 안됨), 1초마다 숫자를 출력하도록 설정. 테스트용으로 busybox:latest 이미지 사용

$ docker run --security-opt=seccomp:unconfined --name cr -d busybox /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'

(여기서 security-opt는 빼고 입력함)

2) 컨테이너 실행 화면은 logs 명령으로 확인: $ docker logs cr



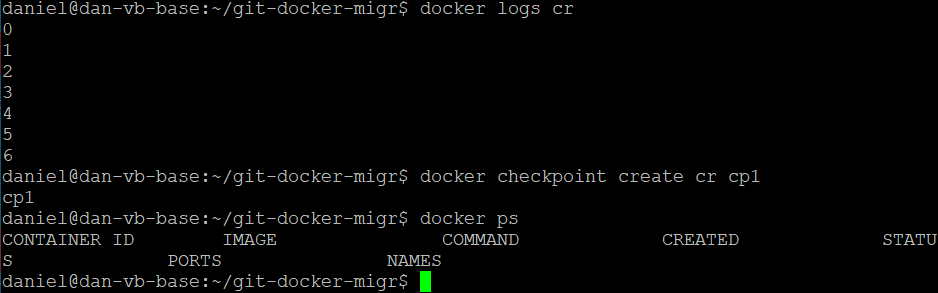
\*\* 여기서 컨테이너를 정지하고 재시작하면 숫자 카운터는 0부터 다시 시작함

3) 체크포인트 생성 (+ 컨테이너 중지, + 체크포인트를 저장할 디렉토리 지정)

- 컨테이너 이름 : cr

- 체크포인트 이름 : check1

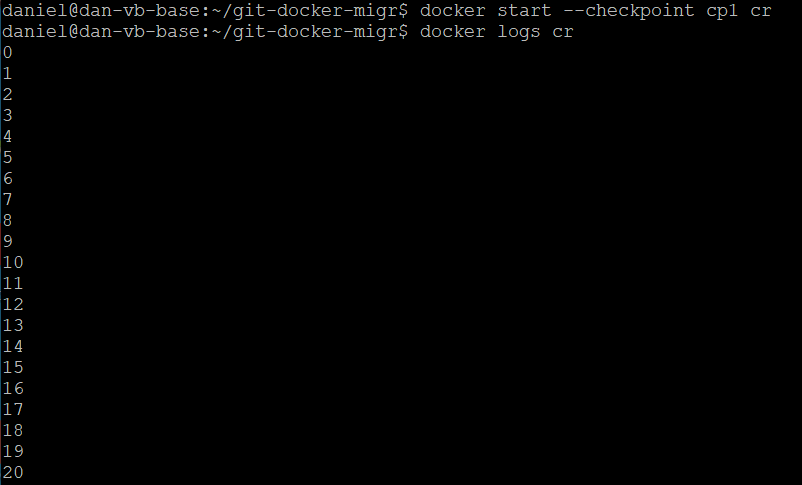
$ docker checkpoint create cr check1



\*\* 컨테이너가 정지되었음(정지하지 안도록 하는 옵션도 있지만 사용안함)

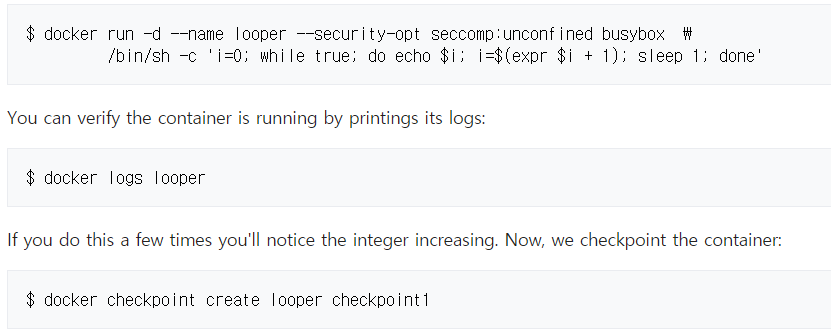
4) 체크포인트로 컨테이너 재시작

$ docker start --checkpoint cp1 cr



성공! 체크포인트를 만들었던 시점부터 다시 이어서 실행되었음

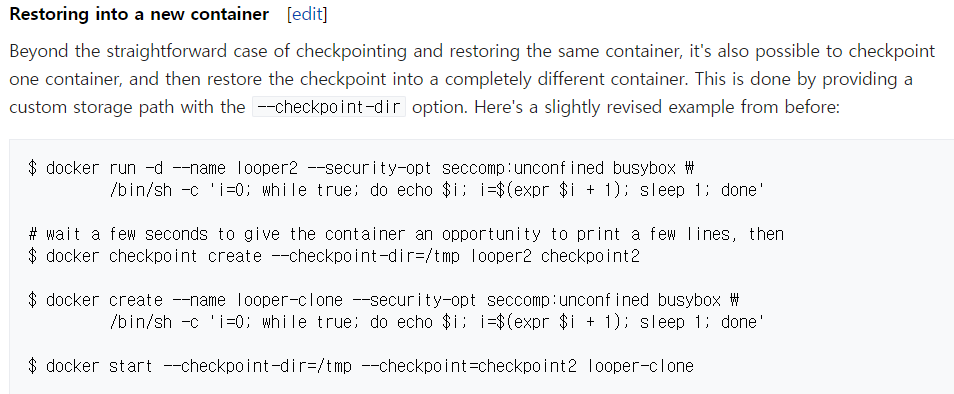
\*\* 간단 정리:



# <디렉토리 지정하여 체크포인트 생성하기/다중머신>

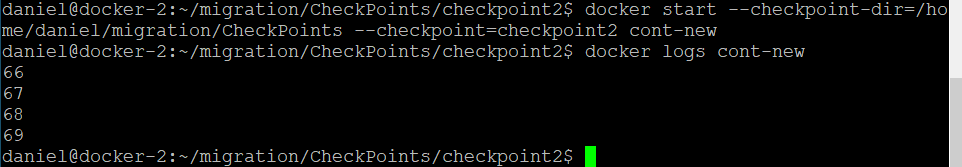
Host A에서 실행하던 컨테이너를 Host B에서 재시작.

\*\* 참고자료: https://criu.org/Docker



|  |  |
| --- | --- |
| 호스트A | $ docker run -d --name cont busybox /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done' |
| 호스트A | $ docker checkpoint create --checkpoint-dir=/home/daniel/migration/CheckPoints cont checkpoint2 |
| 호스트A | $ scp -r checkpoint2 아이디@목적지IP:/home/daniel/migration/CheckPoints  (압축하지 않고, 디렉토리 전체를 그대로 복사) |
| 호스트B | $ docker create --name cont-new busybox /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done' |
| 호스트B | $ docker start --checkpoint-dir=/home/daniel/migration/CheckPoints --checkpoint=checkpoint2 cont-new |

성공! (호스트B에서 cont-new 라는 새로운 컨테이너를 시작한 직후에 logs를 찍어봤는데, 0부터 시작하는게 아니고, 정지했던 곳에서부터 이어서 시작함)



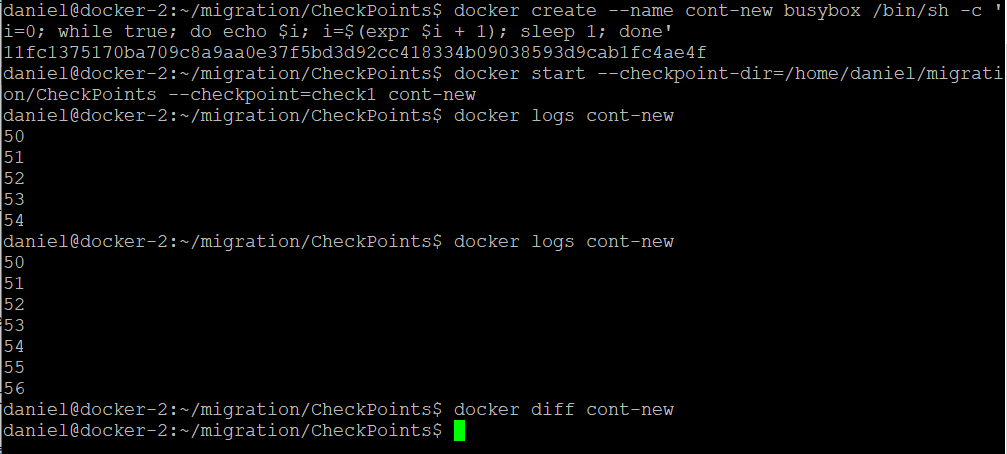
# <Checkpoint가 persistent file을 복사하는지? 안함!>

호스트A: 192.168.0.113

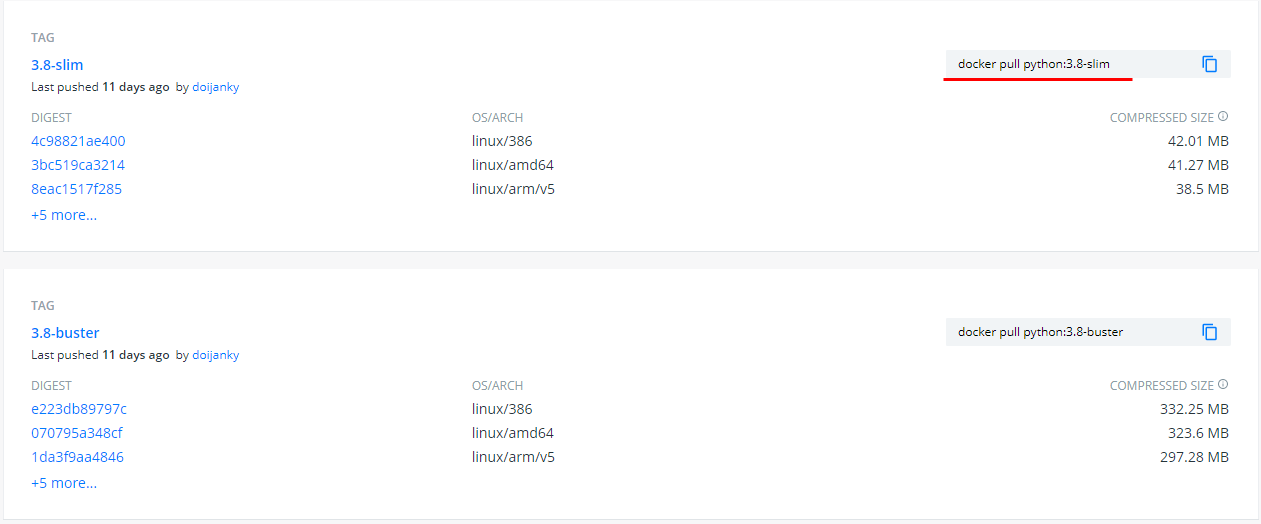
호스트B: 192.168.0.114

|  |  |
| --- | --- |
| 호스트A | $ docker run -d --name cont busybox /bin/sh -c 'truncate -s 10M /tmp/my.file; i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'  (diff 했을때, my.file 이라는 파일이 있다는 것을 알 수 있음) |
| 호스트A | $ docker checkpoint create --checkpoint-dir=/home/daniel/migration/CheckPoints cont check1 |
| 호스트A | $ scp -r /home/daniel/migration/CheckPoints/check1 daniel@192.168.0.114:/home/daniel/migration/CheckPoints  (압축하지 않고, 디렉토리 전체를 그대로 복사) |
| 호스트B | $ docker create --name cont-new busybox /bin/sh -c 'i=0; while true; do echo $i; i=$(expr $i + 1); sleep 1; done'  (my.file을 생성하지 않고, 컨테이너를 create) |
| 호스트B | $ docker start --checkpoint-dir=/home/daniel/migration/CheckPoints --checkpoint=check1 cont-new |

호스트B에서 확인해 본 결과, 체크포인트는 persistent file은 옮기지 않음



# <단일머신 : 체크포인트로 소켓 프로그램 연속 서비스>



파이썬 slim 버전을 사용하자: 용량이 차이가 많이 나네 (python:3.8-slim)

“udp-ask-reply”예제를 사용하자. 도커를 server로!

서버 프로그램 URL: https://github.com/overegoz/DockerMigration/blob/main/sample-code/python-UDP/udp-ask-reply/server.py

호스트A: 192.168.0.113

|  |  |
| --- | --- |
|  | $ docker run -v /home/daniel/git-docker-migr/sample-code/python-UDP/udp-ask-reply:/src -p 20002:20002/udp -d --name server python:3.8-slim /bin/sh -c 'python /src/server.py'  (diff 했을때, server.py 보이는지 확인하기) |
|  | $ docker checkpoint create --checkpoint-dir=/home/daniel/migration/CheckPoints server check1 |
|  | $ scp -r /home/daniel/migration/CheckPoints/check1 daniel@192.168.0.114:/home/daniel/migration/CheckPoints  (압축하지 않고, 디렉토리 전체를 그대로 복사) |
|  | $ docker create -v /home/daniel/git-docker-migr/sample-code/python-UDP/udp-ask-reply:/src -p 20002:20002/udp --name server-new python:3.8-slim /bin/sh -c 'python /src/server.py' |
| 실패 | $ docker start --checkpoint-dir=/home/daniel/migration/CheckPoints --checkpoint=check1 server-new |

(다음 페이지...)

도커파일을 만들어서 다시 시도하자. 마운트 하는 것 때문에 안되는 것일 수 있으니까, server.py 파일을 넣은 파이썬 이미지를 하나 만들자

이미지 python:3.8-slim은 /usr/local/bin/python3에 파이썬 실행파일이 있다. 엔트리 포인트로 지정

|  |  |
| --- | --- |
|  | $ docker run -p 20002:20002/udp -d --name server twoon/python3  (diff 했을때, server.py 보이는지 확인하기) |
|  | $ docker checkpoint create --checkpoint-dir=/home/daniel/migration/CheckPoints server check1 |
|  | $ scp -r /home/daniel/migration/CheckPoints/check1 daniel@192.168.0.114:/home/daniel/migration/CheckPoints  (압축하지 않고, 디렉토리 전체를 그대로 복사) |
|  | $ docker create -p 20002:20002/udp --name server-new twoon/python3 |
| 성공 | $ docker start --checkpoint-dir=/home/daniel/migration/CheckPoints --checkpoint=check1 server-new |

성공!

# <참고자료>

\*\* 수동으로 번호 메기기...

|  |  |
| --- | --- |
| 번호 | 주소 |
| [1] | <https://w.cublr.com/cloud/docker/criu-live-migration/> |
| [2] | <https://github.com/docker/cli/blob/master/experimental/checkpoint-restore.md> |
| [3] | <https://docs.docker.com/engine/reference/commandline/checkpoint/> |
| [4] | <https://technology.amis.nl/platform/docker/first-steps-with-docker-checkpoint-to-create-and-restore-snapshots-of-running-containers/> |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |