Dockek 02

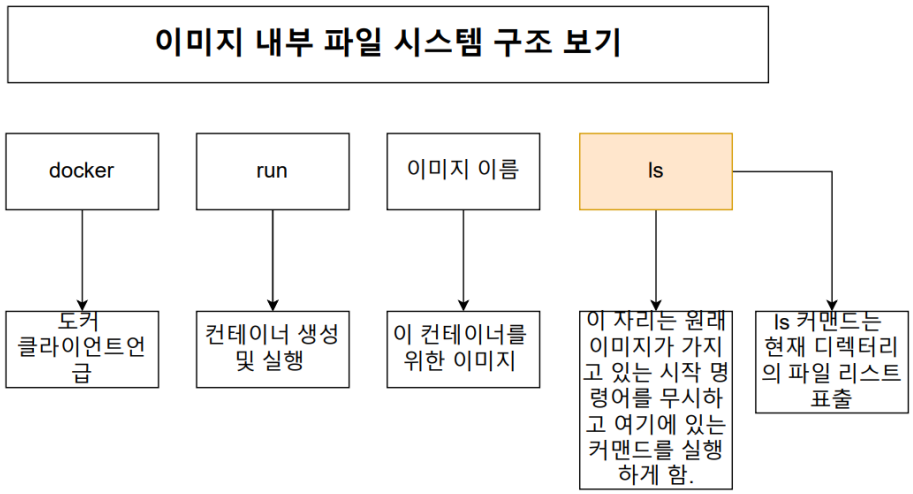
[기본적인 도커 **CLI**]

따라하며 배우는 도커

John Ahn

2023.01.08

도커 클라이언트 명령어



cmd : docker run alpine ls 실행시?

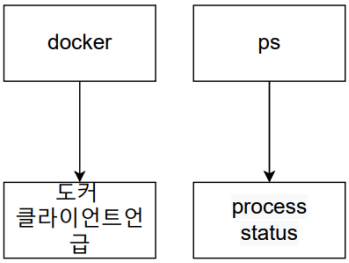
```
C:₩Users₩Lece6>docker run alpine ls
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
c158987b0551: Pull complete
Digest: sha256:8914eb54f968791faf6a8638949e480fef81e697984fba772b3976835194c6d4
Status: Downloaded newer image for alpine:latest
bin
dev
etc
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

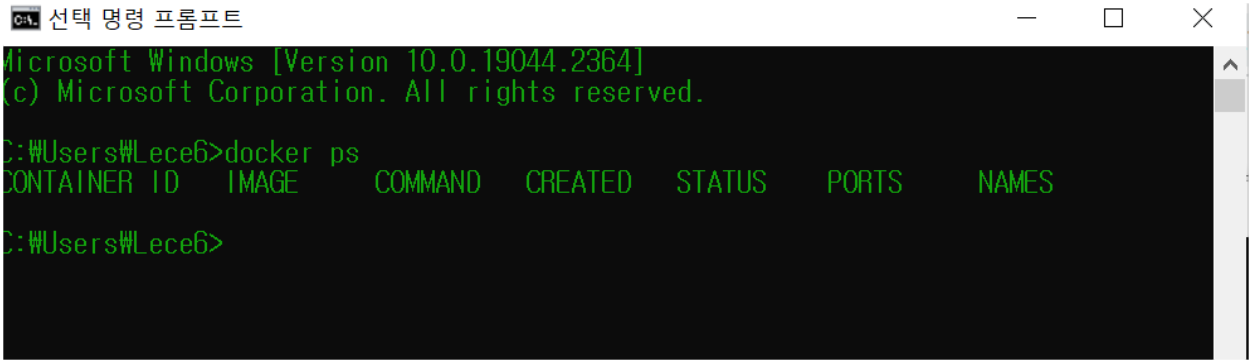
파일 목록이 나온다. 내부 실행 정의 된 것이 아닌 마지막 명령어가 실행 된것.

[첫번째 도표대로 실행 된것]

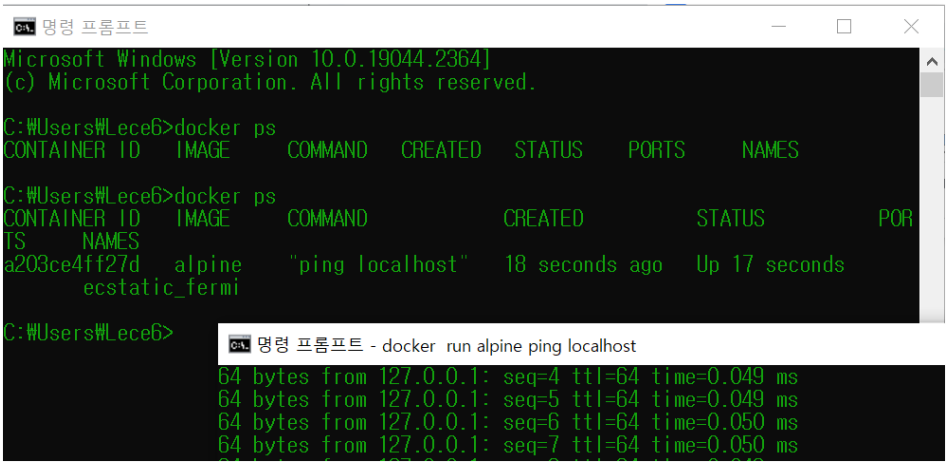
docker ps

현재 실행중인 컨테이너 나열





실행중인 컨테이너가 없을때



이미지 설명

1. CONTAINER ID : 컨테이너의 고유한 아이디 해쉬값.

실제로는 더욱 길지만 일부분만 표출.

2. IMAGE : 컨테이너 생성 시 사용한 도커 이미지.

3. COMMAND : 컨테이너 시작 시 실행될 명령어.

대부분 이미지에 내장되어 있으므로 별도 설정이 필요 X.

4. CREATED : 컨테이너가 생성된 시간.

5. STATUS : 컨테이너의 상태입니다.

실행 중은 Up, 종료는 Exited, 일시정지 Pause.

6. PORTS : 컨테이너가 개방한 포트와 호스트에 연결한 포트.

특별한 설정을 하지 않은 경우 출력되지 않습니다.

뒤에 가서 더 자세히 설명합니다.

7. NAMES : 컨테이너 고유한 이름.

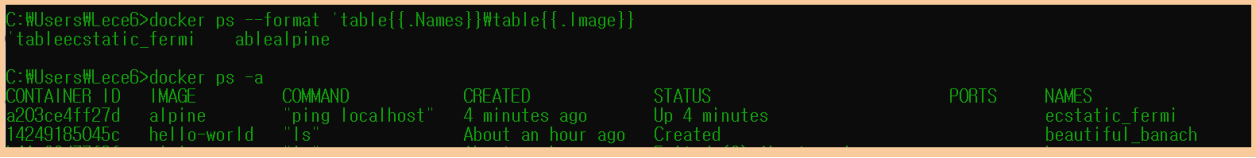
컨테이너 생성 시 --name 옵션으로 이름을 설정하지 않으면

도커 엔진이 임의로 형용사와 명사를 조합해 설정.

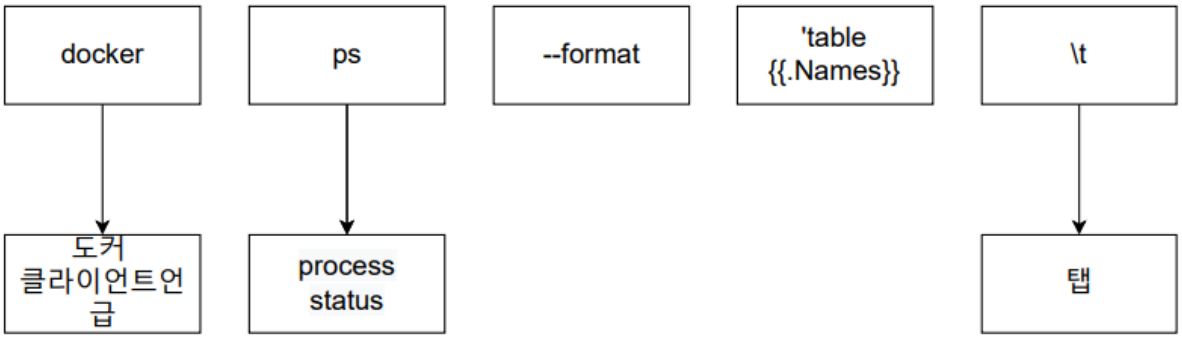
id와 마찬가지로 중복이 안되고 docker rename명령어로

이름을 변경할 수 있습니다.

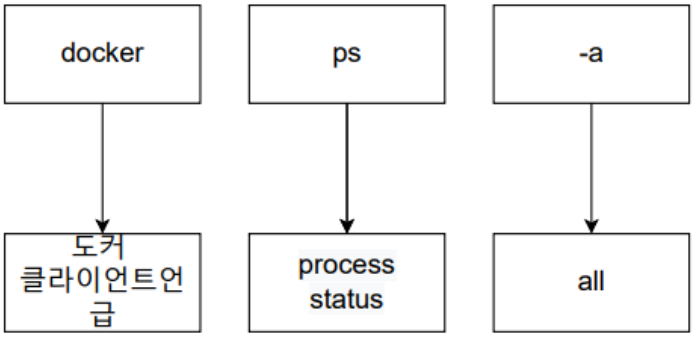
ps 원하는 항목 보기



원하는 항목만 보기



모든 컨테이너 나열

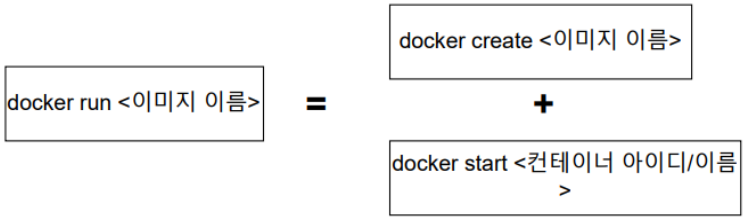


도커 컨테이너 생명주기

docker create < 이미지 이름 >

docker start < 시작할 컨테이너 이름/ 아이디 >

docker run < 이미지 이름 >



docker create = 이미지에 있는 파일 스냅샷을 컨테이너 하드에 넣어준다.

docker start = 시작시 실행 될 명령어 실행

```
C:₩Users₩Lece6>docker create hello-world
46bdc660127b67659e1934939a4daf2d2ba4310c85025b9caf12926bf61d7795

C:₩Users₩Lece6>docker start -a 46bdc6

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

-a ? attach

→ 해당 컨테이너를 알아보기 위해 **ps** 를 사용한다**.**

docker stop < 중지 할 컨테이너 이름 >

docker kill < 중지 할 컨테이너 이름 >

Stop과 Kill 은 어떤 차이가 있을까?

공통점은 둘 다 실행중인 컨테이너를 중지시킵니다.

1. Stop은 Gracefully 하게 중지를 시킵니다. 자비롭게 그동안 하던 작업들을 (메시지를 보내고 있었다면 보내고 있던 메시지) 완료하고 컨테이너를 중지시킨다.
2. Kill 같은 경우는 Stop과 달리 어떠한 것도 기다리지 않고 바로 컨테이너를 중지시킨다.

docker rm < 삭제할 컨테이너 이름 **>**

이미지 삭제 할 때

docker rmi <이미지 id>

한번에 컨테이너**,** 이미지**,** 네트워크 모두 삭제하고 싶다면**?**

docker system prune

이미 실행중인 컨테이너에 명령어를 전달

docker exec < 컨테이너 아이디 **>** 명령어

docker exec 컨테이너아이디 ls = docker run 컨테이너아이디 ls

1. docker run 은 새로 컨테이너를 만들어서 실행
2. docker exec 은 이미 실행 중인 컨테이너에 명령어를 전달

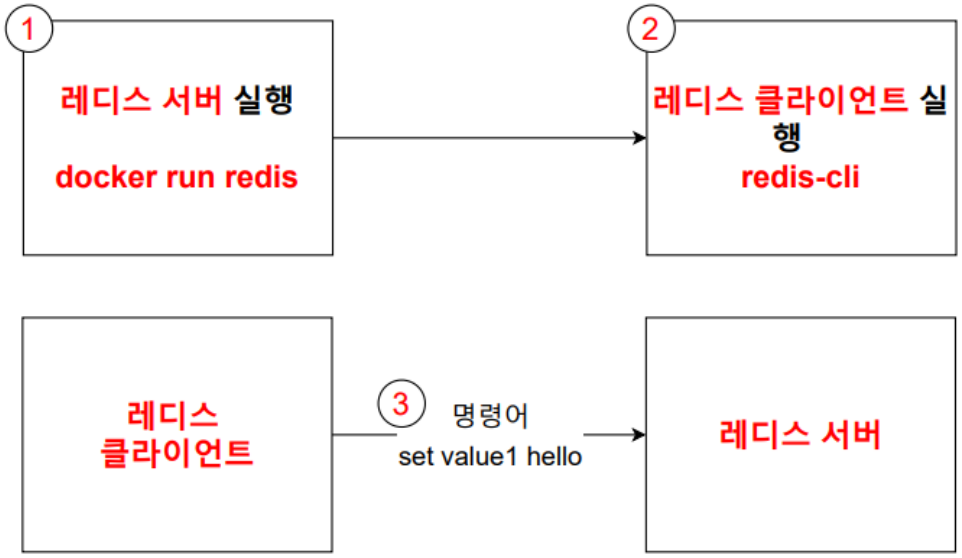
레디스를 사용해보기

레디스 서버를 작동하고 레디스 클라이언트 실행 후 명령어를 전달

레디스**?** **Redis**란**? Key, Value** 구조의 비정형 데이터를 저장하고 관리하기 위한 오픈 소스 기반의 비관계형 데이터 베이스 관리 시스템 **(DBMS)**

1. docker run redis 로 이미지를 pull 받아서 실행 (레디스 서버 실행)
2. 레디스가 실행되고 있는 컨테이너에서 redis -cli 를 실행

```
docker exec -it 0b7e709449e0 redis-cli
```



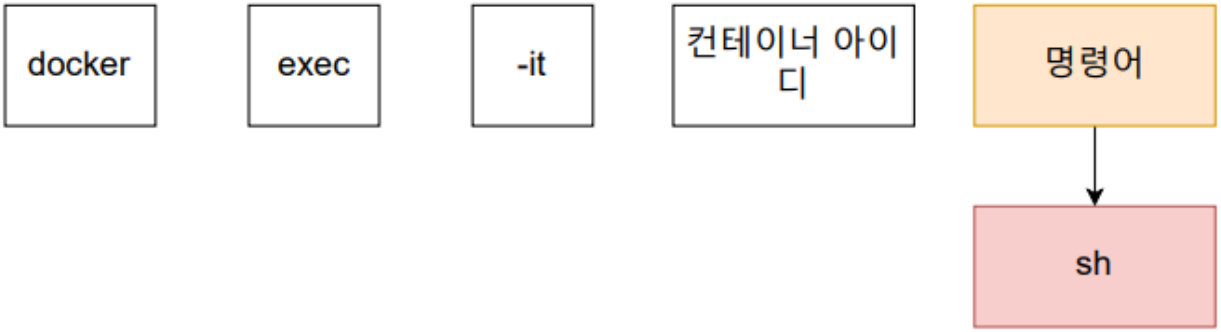
-it? 명령어를 실행후도 명령어를 적을 수 있어진다 . -i interective -t terminal

-it를 안붙이면 실행되고 종료된다.

간단히 사용해 보기

```
C:₩Users₩Lece6>docker exec -it 0b7e709449e0 redis-cli
127.0.0.1:6379> set key1 hello
OK
127.0.0.1:6379> get key1
"hello"
127.0.0.1:6379>
```

실행중인 컨테이너에서 터미널 사용하기



sh를 붙여준다. sh, bash, zsh, powershell 등 이미지에 따라 사용할 수 있는 것들이 다르다.

sh는 어디서나 사용 할 수있다.

```
C:₩Users₩Lece6>docker exec -it d7efd59d3d7e sh
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin   srv    sys    tmp    usr    var
/ # dir
sh: dir: not found
/ # touch new-file
/ # ls
bin       etc       lib       mnt       opt       root      sbin      sys       usr
dev       home      media     new-file  proc      run       srv       tmp       var
/ # export hello=hi
/ # echo $hello
hi
/ #
```

사용 할 수 있다. 유닉스 환경의 명령어들 가능