

```

1 1. run command
2 1)run 명령어를 사용하면 사용할 이미지가 저장되어 있는지 확인하고 없다면 docherHub에서 다운로드 한 후 Container를
   생성하고 시작한다.
3 2)Container를 시작했지만 특별히 수행해야할 명령어를 전달하지 않으면 Container는 실행되지마자 바로 종료된다.
4   $ sudo docker run ubuntu:20.04
5   $ sudo docker ps -a
6
7
8 3)Container 내부에 들어가기 위해 bash를 실행하고, 키보드 입력을 위해 -it 옵션 사용하기
9 4)프로세스가 종료되면 자동으로 Container가 삭제되도록 --rm 사용하기
10  $ sudo docker run --rm -it ubuntu:20.04 /bin/bash
11  /# whoami
12  /# uname -a
13  /# ls
14  /# cat /etc/issue
15
16
17 5)웹 어플리케이션 실행하기
18  -5678 port로 브라우저에서 연결
19  -d 옵션으로 백그라운드에서 실행
20  $ sudo docker run -d --rm -p 5678:5678 hashicorp/http-echo -text="Hello World"
21  $ curl localhost:5678
22  Hello World
23
24  $ sudo docker run -d --rm -p 5679:5678 hashicorp/http-echo -text="Docker World"
25  $ curl localhost:5679
26  Docker World
27
28
29 ※다음은 WSL2에서 실행할 것
30
31 6)Redis 실행하기
32  $ docker run -d --rm -p 1234:6379 redis
33  $ telnet localhost 1234
34  Trying 127.0.0.1...
35  Connected to localhost.
36  Escape character is '^]'.
37  set hello world
38  +OK
39  get hello
40  $5
41  world
42  quit
43  +OK
44  Connection closed by foreign host.
45
46
47 7)MySQL 실행하기
48  $ docker run -d -p 3306:3306 \
49  > -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
50  > --name mysql \
51  > mysql:5.7
52
53  $ docker exec -it mysql mysql
54  mysql> CREATE DATABASE wp CHARACTER SET utf8;
55  mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
56  mysql> FLUSH PRIVILEGES;
57  mysql> show databases;
58  +-----+
59  | Database          |
60  +-----+
61  | information_schema |
62  | mysql              |
63  | performance_schema |
64  | sys                |
65  | wp                  |
66  +-----+

```

```

67      5 rows in set (0.00 sec)
68      mysql> quit
69
70
71      8)WordPress 실행하기
72      $ docker run -d -p 8080:80 \
73      > -e WORDPRESS_DB_HOST=host.docker.internal \ <---Linux에서는 연결안됨. WSL만 가능
74      > -e WORDPRESS_DB_NAME=wp \
75      > -e WORDPRESS_DB_USER=wp \
76      > -e WORDPRESS_DB_PASSWORD=wp \
77      > --name wordpress
78      > wordpress
79
80      9)브라우저에서 연결
81      http://localhost:8080
82
83
84      10)사이트 제작 후
85      $ docker exec -it mysql mysql
86      mysql>show databases;
87      mysql>use wp
88      mysql>show tables;
89      mysql>desc wp_users;
90      mysql>SELECT * FROM wp_users;
91
92
93      2. stop command
94      1)현재 Container 확인
95      $ docker ps
96
97      2)중지된 모든 Container까지 확인
98      $ docker ps -a
99
100     3)Container 중지하기(띄어쓰기를 이용해서 여러개의 Container를 중지 가능)
101     $ docker ps -a
102     $ docker stop {{CONTAINER ID}} {{CONTAINER ID}} {{CONTAINER ID}}
103
104
105     3. rm command
106     1)MySQL과 WordPress를 제외한 나머지 Container 삭제하기
107
108
109     4. logs command
110     1)MySQL log 보기
111     $ docker logs mysql-pid
112
113     2)Nginx log 보기
114     $ docker run -dp 8080:80 Nginx
115     $ docker ps -a
116     $ docker logs nginx-pid
117
118     $ docker logs -f nginx-pid
119     -웹브라우저에서 잘못된 페이지로 404 에러 발생
120     http://localhost:8080/aaaa.html
121     -또는 계속 Refresh
122     -로그 계속 출력 중
123
124
125     5. network create command
126     1)app-network 라는 이름으로 wordpress와 MySQL이 통신할 네트워크 만들기
127     $ docker network create app-network
128
129
130     6. network connect command
131     1) MySQL containier에 네트워크를 추가
132     $ docker network connect app-network mysql
133

```

```

134 2)--network option 사용하기
135 -WordPress를 app-network에 속하게 하고 mysql을 이름으로 접근한다.
136 $ docker stop wordpress
137 $ docker rm -f wordpress
138 $ docker run -dp 8080:80 \
139 > --network=app-network \
140 > -e WORDPRESS_DB_HOST=mysql \
141 > -e WORDPRESS_DB_NAME=wp \
142 > -e WORDPRESS_DB_USER=wp \
143 > -e WORDPRESS_DB_PASSWORD=wp \
144 > wordpress
145
146
147 7. Volume Mount command
148 1) mysql container stop & rm
149 $ docker stop mysql
150 $ docker rm mysql
151
152 2) mysql 재실행
153 $ docker run -dp 3306:3306 \
154 > -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
155 > --network=app-network --name mysql \
156 > mysql:5.7
157 -WordPress 홈페이지 접근 에러 발생
158
159
160 8. Docker CLI 연습
161 $ sudo docker pull busybox
162 $ sudo docker images
163 $ sudo docker run -it busybox sh
164 # ls
165 # cd /var
166 # touch test.log
167 # ls -al
168 # exit
169 $ sudo docker ps -a
170 $ sudo docker start {{CONTAINER ID}}
171 $ sudo docker ps -a
172 $ sudo docker attach {{CONTAINER ID}}
173 # history
174 # ctrl + p, ctrl + q <---exit와 달리 container를 정지하지 않고 빠져 나옴.
175 # read escape sequence <----exit 로 빠져 나오는 것이 아닌 대기 모드상태
176 $ docker ps -a <----image가 계속 실행중임을 확인할 수 있다.
177
178 $ sudo docker commit {{CONTAINER ID}} sample:v1 <--- busybox를 sample:v1으로 Snapshot 했음.
179 $ sudo docker images <--- busybox와 용량을 거의 같으나 이미지 아이디가 다름.
180 $ docker tag sample:v1 sample:latest
181 $ docker images <---- tag는 Image의 아이디가 같음.
182 $ docker run -it sample:v1
183 # ls
184 # cd /var
185 # ls -al
186 # history
187 # exit
188
189
190 $ sudo docker images
191 $ sudo docker tag sample:latest {{dockerhub's ID}}/sample:latest
192 $ sudo docker images
193 $ sudo docker login
194 Username :
195 Password :
196 Login Succeeded
197 $ sudo docker push {{dockerhub's ID}}/sample.latest
198
199 $ sudo docker rmi {{dockerhub's ID}}/sample
200 $ sudo docker images

```

```

201 $ sudo docker pull {{dockerhub's ID}}/sample
202 $ sudo docker images <----Image ID가 같음.
203
204 $ sudo docker save {{dockerhub's ID}}/sample > ./sample.tgz
205 $ ls -al
206 $ sudo docker rmi {{dockerhub's ID}}/sample
207 $ sudo docker ps -a
208 $ sudo docker load < ./sample.tgz
209 $ sudo docker ps -a
210
211 $ sudo docker images
212 $ sudo docker rmi로 모든 docker images를 지운다.
213
214 $ sudo docker pull busybox:latest
215 $ sudo docker run -it busybox sh
216 # ls
217 # touch sample.myimage
218 # ctrl + p, ctrl + q
219 $ sudo docker ps -a
220 $ sudo docker cp sample.myimage ./
221 must specify at least one container source
222 $ sudo docker cp {{CONTAINER ID}}:sample.myimage ./
223
224
225 9. Ref https://github.com/ralfyang/docker\_cli\_dashboard
226
227 10. Ref https://asciinema.org/a/166084
228
229 11. Docker의 유용한 명령어 모음.
230 1)Port forwarding
231 $ sudo docker run -d --name tc -p 80:8080 consol/tomcat-8.0
232 Webbrowser에서 http://container-ip:80
233
234 2)Container 내부 shell 실행
235 $ sudo docker exec -it tc /bin/bash
236
237 3)Container Log 확인
238 $ sudo docker logs tc
239
240 4)Host 및 Container간 파일 복사
241 $ sudo docker cp <path> <to container>:<path>
242 $ sudo docker cp <from container>:<path> <path>
243 $ sudo docker cp <from container>:<path> <to container>:<path>
244
245 $ echo hello > test.txt
246 $ cat test.txt
247 hello
248 $ sudo docker cp test.txt tc:/
249 $ sudo docker exec -it tc cat /test.txt
250 hello
251
252 $ sudo docker cp tc:/test.txt ./test2.txt
253 $ cat test2.txt
254 hello
255
256 5)임시 Container 생성
257 $ sudo docker ps -a -q <--- container의 id만 보임.
258 $ sudo docker stop `docker ps -a -q` <---전체 container id를 stop
259
260 $ sudo docker run -d -p 80:8080 --rm --name tc tomcat
261 --rm 옵션은 stop만 해도 container가 삭제되는 효과가 있음.
262

```