1. run command

1)run 명령어를 사용하면 사용할 이미지가 저장되어 있는지 확인하고 없다면 docherHub에서 다운로드 한 후 Container를 생성하고 시작한다.

2)Container를 시작했지만 특별히 수행해야할 명령어를 전달하지 않으면 Container는 실행되지마자 바로 종료된다.

```
$ sudo docker run ubuntu:20.04
$ sudo docker ps -a
```

3)Container 내부에 들어가기 위해 bash를 실행하고, 키보드 입력을 위해 -it 옵션 사용하기

4)프로세스가 종료되면 자동으로 Container가 삭제되도록 --rm 사용하기

```
$ sudo docker run --rm -it ubuntu:20.04 /bin/bash
/# whoami
/# uname -a
/# ls
/# cat /etc/issue
```

5)웹 어플리케이션 실행하기

-5678 port로 브라우저에서 연결

-d 옵션으로 백그라운드에서 실행

```
$ sudo docker run -d --rm –p 5678:5678 hashicorp/http-echo –text="Hello World"
$ curl localhost:5678
Hello World

$ sudo docker run -d --rm –p 5679:5678 hashicorp/http-echo –text="Docker World"
$ curl localhost:5679
Docker World
```

※다음은 WSL2에서 실행할 것

6)Redis 실행하기

```
$ docker run -d --rm -p 1234:6379 redis
$ telnet localhost 1234
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
set hello world
+OK
get hello
$5
world
quit
+OK
Connection closed by foreign host.
```

7)MySQL 실행하기

```
$ docker run -d -p 3306:3306 \
> -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
> --name mysql \
> mysql:5.7

$ docker exec -it mysql mysql
mysql> CREATE DATABASE wp CHARACTER SET utf8;
mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp'@'%' IDENTIFIED BY 'wp';
mysql> FLUSH PRIVILEGES;
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wp                 |
+--------------------+
```

```
    5 rows in set (0.00 sec)
    mysql> quit


 8)WordPress 실행하기
    $ docker run -d -p 8080:80 \
    > -e WORDPRESS_DB_HOST=host.docker.internal \   <---Linux에서는 연결안됨. WSL만 가능
    > -e WORDPRESS_DB_NAME=wp \
    > -e WORDPRESS_DB_USER=wp \
    > -e WORDPRESS_DB_PASSWORD=wp \
    > --name wordpress
    > wordpress

 9)브라우저에서  연결
    http://localhost:8080


 10)사이트 제작 후
    $ docker exec -it mysql mysql
    mysql>show databases;
    mysql>use wp
    mysql>show tables;
    mysql>desc wp_users;
    mysql>SELECT * FROM wp_users;


2. stop command
 1)현재 Container 확인
    $ docker ps

 2)중지된 모든 Container까지 확인
    $ docker ps -a

 3)Container 중지하기(띄어쓰기를 이용해서 여러개의 Container를 중지 가능)
    $ docker ps -a
    $ docker stop {{CONTAINER ID}} {{CONTAINER ID}} {{CONTAINER ID}}


3. rm command
 1)MySQL과 WordPress를 제외한 나머지 Container 삭제하기


4. logs command
 1)MySQL log 보기
    $ docker logs mysql-pid

 2)Nginx log 보기
    $ docker run -dp 8080:80 Nginx
    $ docker ps -a
    $ docker logs nginx-pid

    $ docker logs -f nginx-pid
    -웹브라우저에서 잘못된 페이지로 404 에러 발생
      http://localhost:8080/aaaa.html
    -또는 계속 Refresh
    -로그 계속 출력 중


5. network create command
 1)app-network 라는 이름으로 wordpress와 MySQL이 통신할 네트워크 만들기
    $ docker network create app-network


6. network connect command
 1) MySQL containier에 네트워크를 추가
    $ docker network connect app-network mysql
```

2)--network option 사용하기
-WordPress를 app-network에 속하게 하고 mysql을 이름으로 접근한다.

```
$ docker stop wordpress
$ docker rm -f wordpress
$ docker run -dp 8080:80 \
> --network=app-network \
> -e WORDPRESS_DB_HOST=mysql \
> -e WORDPRESS_DB_NAME=wp \
> -e WORDPRESS_DB_USER=wp \
> -e WORDPRESS_DB_PASSWORD=wp \
> wordpress
```

7. Volume Mount command

1) mysql container stop & rm

```
$ docker stop mysql
$ docker rm mysql
```

2) mysql 재실행

```
$ docker run -dp 3306:3306 \
> -e MYSQL_ALLOW_EMPTY_PASSWORD=true \
> --network=app-network --name mysql \
> mysql:5.7
```

-WordPress 홈페이지 접근 에러 발생

8. Docker CLI 연습

```
$ sudo docker pull busybox
$ sudo docker images
$ sudo docker run -it busybox sh
# ls
# cd /var
# touch test.log
# ls -al
# exit
$ sudo docker ps -a
$ sudo docker start {{CONTAINER ID}}
$ sudo docker ps -a
$sudo  docker attach {{CONTAINER ID}}
# history
# ctrl + p, ctrl + q    <---exit와 달리 container를 정지하지 않고 빠져 나옴.
# read escape sequence   <----exit 로 빠져 나오는 것이 아닌 대기 모드상태
$ docker ps -a          <----image가 계속 실행중임을 확인할 수 있다.

$ sudo docker commit {{CONTAINER ID}} sample:v1  <--- busybox를 sample:v1으로 Snapshot 했음.
$ sudo docker images   <--- busybox와 용량을 거의 같으나 이미지 아이디가 다름.
$ docker tag sample:v1 sample:latest
$ docker images  <---- tag는 Image의 아이디가 같음.
$ docker run -it sample:v1
# ls
# cd /var
# ls -al
# history
# exit


$ sudo docker images
$ sudo docker tag sample:latest {{dockerhub's ID}}/sample:latest
$ sudo docker images
$ sudo docker login
Username :
Password :
Login Succeeded
$ sudo docker push {{dockerhub's ID}}/sample.latest

$ sudo docker rmi {{dockerhub's ID}}/sample
$ sudo docker images
```

```
    $ sudo docker pull {{dockerhub's ID}}/sample
    $ sudo docker images    <----Image ID가 같음.

    $ sudo docker save {{dockerhub's ID}}/sample > ./sample.tgz
    $ ls -al
    $ sudo docker rmi {{dockerhub's ID}}/sample
    $ sudo docker ps -a
    $ sudo docker load < ./sample.tgz
    $ sudo docker ps -a

    $ sudo docker images
    $ sudo docker rmi로 모든 docker images를 지운다.

    $ sudo docker pull busybox:latest
    $ sudo docker run -it busybox sh
    # ls
    # touch sample.myimage
    # ctrl + p, ctrl + q
    $ sudo docker ps -a
    $ sudo docker cp sample.myimage ./
    must specify at least one container source
    $ sudo docker cp {{CONTAINER ID}}:sample.myimage ./


9. Ref https://github.com/ralfyang/docker_cli_dashboard

10. Ref https://asciinema.org/a/166084
```