**Docker를 이용한 서버 구축**

2019.07

목차

[1. Docker란? 2](#_Toc13666972)

[2. Docker VS VMware 3](#_Toc13666973)

[3. Docker 설치 4](#_Toc13666974)

[3.1. Docker 설치 준비 4](#_Toc13666975)

[3.2. Docker 설치 7](#_Toc13666976)

[4. Docker 명령어 12](#_Toc13666977)

[4.1. Docker 이미지 명령어 12](#_Toc13666978)

[4.2. Docker 컨테이너 명령어 13](#_Toc13666979)

[5. Dockerfile 작성 15](#_Toc13666980)

[6. Docker로 서버 설치 17](#_Toc13666981)

[6.1. Docker hub 소개 17](#_Toc13666982)

[6.2. DB설치 17](#_Toc13666983)

[6.2.1. MariaDB 설치 17](#_Toc13666984)

[6.2.2. PostgreSQL 설치 20](#_Toc13666985)

[6.3. Nginx 설치 21](#_Toc13666986)

[6.3.1. Nginx 설치 21](#_Toc13666987)

[6.3.2. webapp 파일 업로드 24](#_Toc13666988)

[7. VirtualBox에서 Docker 접속하기 25](#_Toc13666989)

[7.1. 포트 포워딩 25](#_Toc13666990)

[8. On-premise 환경에서의 도커 사용 27](#_Toc13666991)

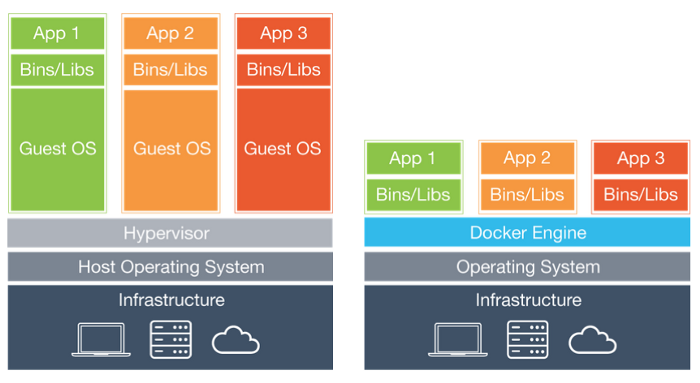
# Docker란?

도커는 **컨테이너** 기반의 가상화 플랫폼이다.

참고 사이트

* 초보를 위한 도커 안내서: <https://subicura.com/2017/01/19/docker-guide-for-beginners-1.html>
* 가장 빨리 만나는 도커: <http://pyrasis.com/docker.html>

# Docker VS VMware



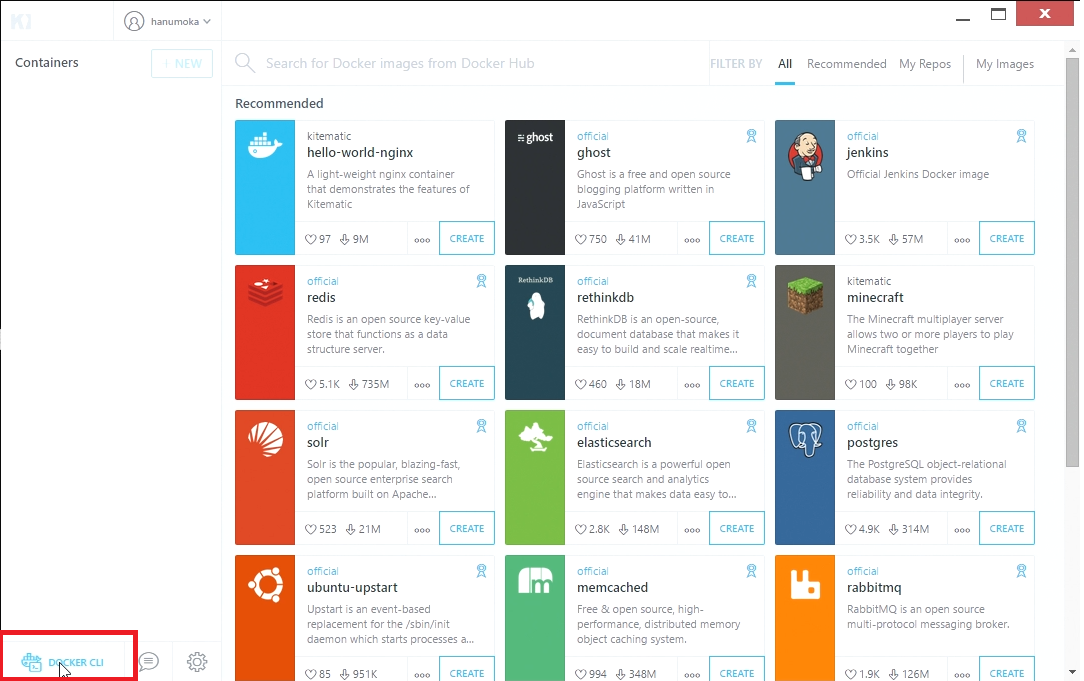
* VM의 특징
  + 별도의 OS 동작
  + Host의 OS와 강력하게 분리됨 🡪 하나의 VM이 뚫려도 다른 것에 영향을 미치지 않음
  + io로드가 큼 (명령처리: vm os 🡪 host os)
* Docker의 특징
  + Host OS를 사용함
  + 이미지 방식으로 제작/배포
  + 모든 컨테이너가 Host OS를 공유함 🡪 하나의 컨테이너가 뚫리면 다른 컨테이너도 위험

# Docker 설치 (Windows PC에서의 Docker 설치)

## Docker 설치 준비

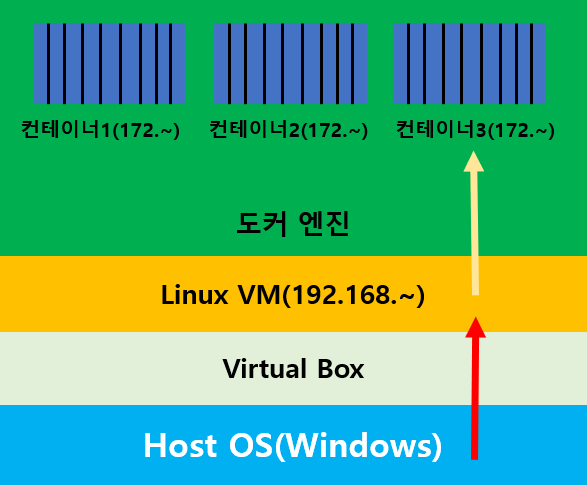
도커는 Linux에서만 설치가 가능하며 윈도우 PC에서 설치하기 위해서는 두가지 방법이 있다.

1. DockerToolbox 사용
   * 설치정보: <https://docs.docker.com/docker-for-windows/>



(설치된 도커 툴박스의 모습)

* + 장점
    - 설치가 쉽다
    - 도커 이미지 검색이 쉽다.(도커 컨테이너 설치가 쉽다)
  + 단점
    - VirtualBox위에 가상 리눅스가 자동으로 설치된다. (이미 VirtualBox가 설치된 PC라면 충돌이 일어날 수도..)



* + - 도커 이미지를 만들거나 빌드하기가 어렵다. 🡪 시중에 나온 이미지만 사용해야 함
  + 결론: 권장하지 않음

1. VirtualBox 사용
   * VirtualBox에 리눅스를 설치하고 도커를 사용한다.
   * 장점
     + 도커이미지를 자유롭게 다운받고 생성/배포할 수 있다.
     + 하나의 리눅스만 설치해도 많은 도커를 실행해 볼 수 있다.
   * 단점
     + 도커 명령어를 익혀야 한다.
     + Vbox의 네트워크 설정을 알아야 한다.
     + PC의 성능에 좌우된다.

## Docker 설치

자세한 설치 과정은 여기를 참조 (<https://docs.docker.com/install/linux/docker-ce/centos/>)

1. CentOS7이 설치된 VirtualBox를 준비한다. (CentOS7 설치 과정은 생략. Minimal버전 설치를 권장함)
2. CentOS7에 자동으로 설치된 이전 버전의 도커를 삭제한다. (없을 수도 있음)

|  |
| --- |
| [root@localhost]# yum remove docker \  docker-client \  docker-client-latest \  docker-common \  docker-latest \  docker-latest-logrotate \  docker-logrotate \  docker-engine |

1. 저장소를 설정한다.

필수 패키지 설정

|  |
| --- |
| [root@localhost]# yum install -y yum-utils device-mapper-persistent-data lvm2 |

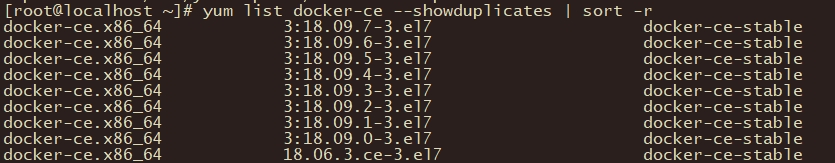
저장소 설정

|  |
| --- |
| [root@localhost]# yum-config-manager \  --add-repo \  https://download.docker.com/linux/centos/docker-ce.repo |

1. Docker CE를 설치한다.

목록 확인

|  |
| --- |
| [root@localhost]# yum list docker-ce --showduplicates | sort -r |



* + 버전 읽는 방법 (첫번째 콜론(:)에서 시작하여 첫번째 하이픈(-)까지)
    - docker-ce.x86\_64 3:18.09.7-3.el7 docker-ce-stable

가장 Stable한 최신버전 설치 (특정 버전을 선택하여 설치한다: docker-ce-<VERSION\_STRING>)

|  |
| --- |
| [root@localhost]# yum install docker-ce-18.09.7 |

(아래와 같은 방식으로도 설치할 수 있지만 추천하지 않음)

|  |
| --- |
| [root@localhost]# yum install docker-ce |

* 이렇게 설치하지 말 것!

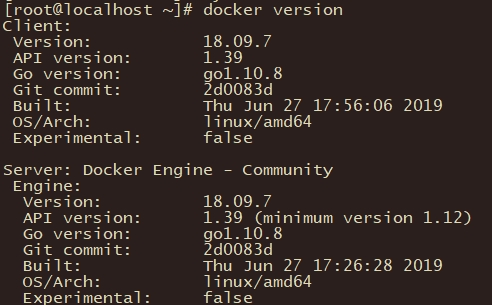
1. Docker 프로세스 시작/종료

설치된 도커를 시작해 보자. (OS 부팅 시 자동으로 띄우도록 등록하면 편함.)

|  |
| --- |
| [root@localhost]# systemctl start docker |

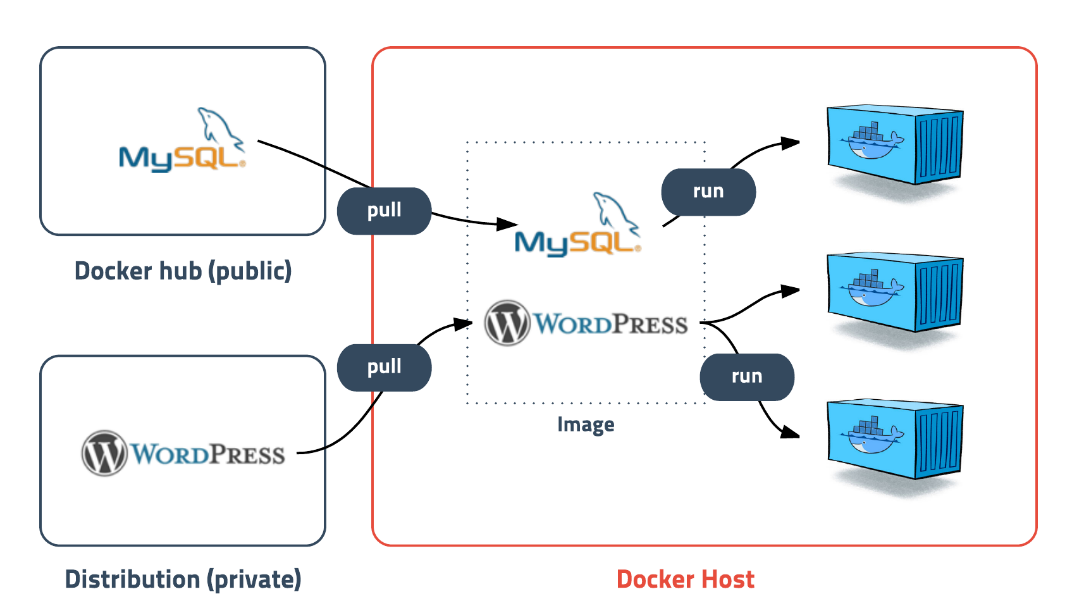
설치된 도커의 버전을 확인한다.

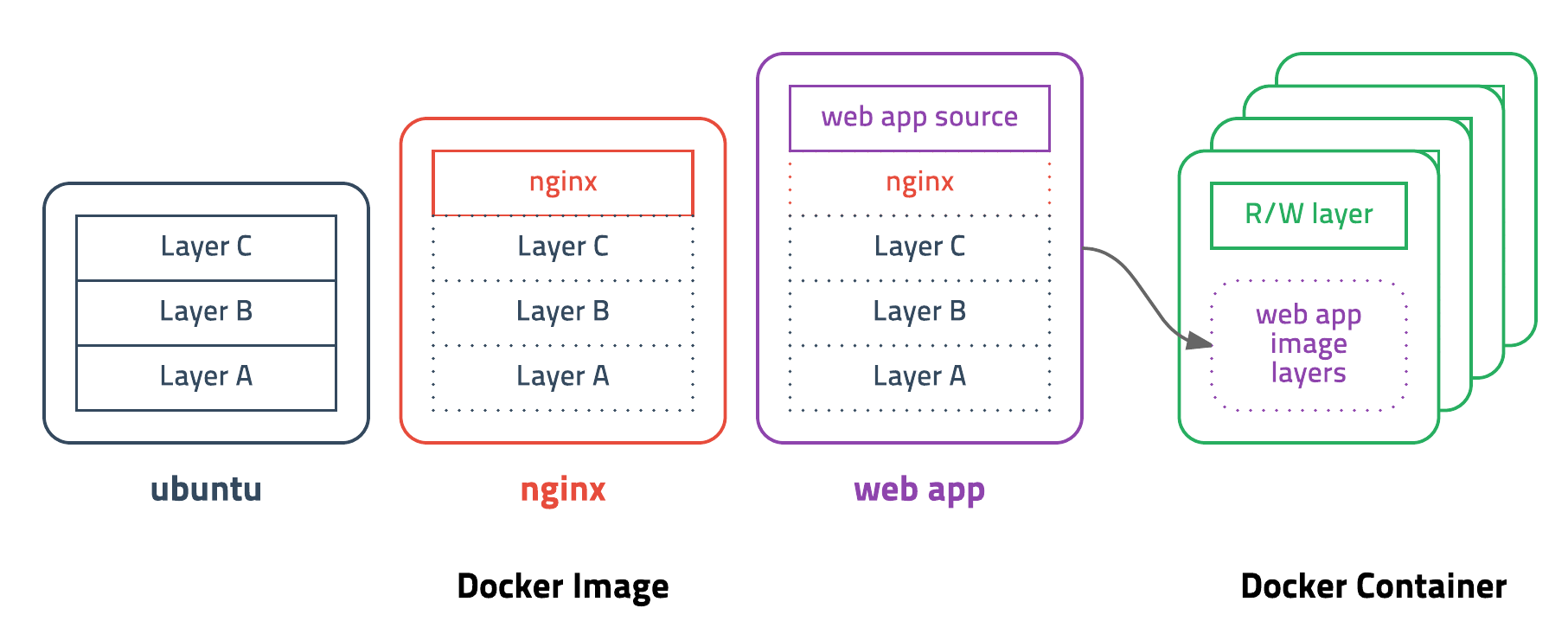
|  |
| --- |
| [root@localhost]# docker version |



설치된 도커의 버전이 맞는지 확인한다. (Client, Server 두개의 버전이 맞으면 정상)

* 1. Docker 이미지와 컨테이너의 관계





(레이어 저장방식) - 새로운 어플리케이션 추가가 매우 용이함

# Docker 명령어

## Docker 이미지 명령어

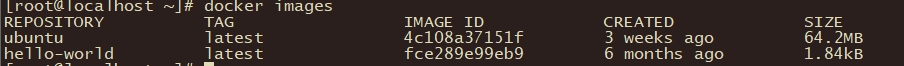
1. 도커 이미지 받기 (예: 우분투 받기)

[root@localhost]# docker pull <도커이미지:버전>

|  |
| --- |
| [root@localhost]# docker pull ubuntu:latest |

1. 도커 이미지 리스트 확인

|  |
| --- |
| [root@localhost]# docker images |



1. 도커 이미지 삭제

[root@localhost]# docker rmi <REPOSITORY>

|  |
| --- |
| [root@localhost]# docker rmi ubuntu |

## Docker 컨테이너 명령어

1. 컨테이너 프로세스 확인

실행 중인 프로세스만 확인

|  |
| --- |
| [root@localhost]# docker ps |

모든 프로세스까지 확인(종료된 프로세스 포함)

|  |
| --- |
| [root@localhost]# docker ps -a |

1. 컨테이너 생성

[root@localhost]# docker run <각종 옵션>

|  |
| --- |
| [root@localhost]# docker run -it -d ubuntu |

(만일 이미지를 다운받지 않은 상태에서 run을 하게 되면 자동으로 이미지를 download한다.)

(컨테이너 이름을 지어주지 않으면 자동으로 생성된다.)

1. 컨테이너 시작(한번 생성된 컨테이너는 시작/중지를 통해 제어할 수 있다.)

[root@localhost]# docker start <도커 컨테이너 이름> (이름대신 CONTAINER ID를 사용해도 된다.)

|  |
| --- |
| [root@localhost]# docker start *romantic\_goldwasser* |

(컨테이너 이름은 이미지 이름과 다르다. 실행(run) 시킬 때 이름을 지정할 수 있다.)

1. 컨테이너 중지

[root@localhost]# docker stop <도커 컨테이너 이름>

|  |
| --- |
| [root@localhost]# docker stop *romantic\_goldwasser* |

1. 컨테이너 접속

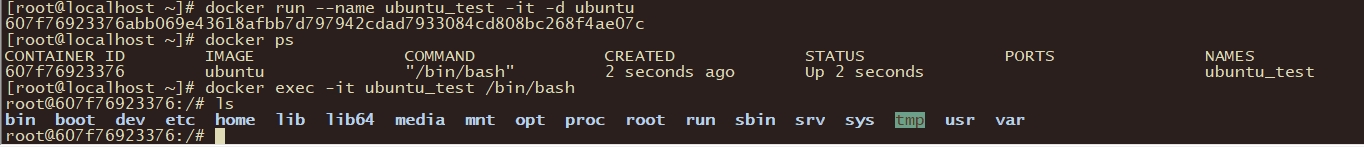
[root@localhost]# docker exec -it <도커 컨테이너 이름> /bin/bash

|  |
| --- |
| [root@localhost]# docker exec -it *romantic\_goldwasser* /bin/bash |

(docker exec 명령을 통해 특정 컨테이너에 “/bin/bash”를 통해 명령을 전달할 수 있다)

(*romantic\_goldwasser* 라는 이름의 컨테이너에 접속을 한다.)

1. 도커 생성/확인/접속



1. 컨테이너 삭제

[root@localhost]# docker rm <도커 컨테이너 이름>

|  |
| --- |
| [root@localhost]# docker rm *romantic\_goldwasser* |

(컨테이너를 삭제하면 그 안에 생성된 모든 파일이나 데이터가 삭제된다. -v옵션을 통해 파일이나 데이터는 Host서버에 저장하자)

1. docker run 옵션 (자주 쓰이는 옵션)

|  |  |
| --- | --- |
| 옵션 | 설명 |
| -d | detached mode 흔히 말하는 백그라운드 모드 |
| -p | 호스트와 컨테이너의 포트를 연결(포워딩) <Host포트:Docker포트> |
| -v | 호스트와 컨테이너의 디렉토리를 연결(마운트) <Host디렉토리:Docker디렉토리> |
| -e | 컨테이너 내에서 사용할 환경변수 설정 (사용할 환경변수 만큼 옵션을 붙이면 됨) |
| --name | 컨테이너 이름 설정 (안 쓰면 자동으로 붙여 줌) |
| --rm | 프로세스 종료 시 컨테이너 자동 제거 |
| -it | -i와 -t를 동시에 사용한 것으로 터미널 입력을 위한 옵션 |

# Dockerfile 작성

기본 도커 이미지에서 여러가지 세팅을 하고 싶은 경우에는 Dockerfile을 이용하면 간단하다. (Dockerfile이 도커의 존재 이유다.)

우분투 이미지를 받아서 locale을 UTF-8로 설정하는 경우를 생각해 보자.

다음과 같이 우분투 이미지를 다운 받은 후 컨테이너에 접속해서 locale을 설정할 수 있다.

|  |
| --- |
| [root@localhost]# docker run -it ubuntu /bin/bash  root@427ac2bd68a6:/# apt-get update  root@427ac2bd68a6:/# apt-get install locales  root@427ac2bd68a6:/# locale-gen ko\_KR.UTF-8  root@427ac2bd68a6:/# export LC\_ALL=ko\_KR.UTF-8 |

(위 절차를 모두 Dockerfile에 담아서 이미지를 생성할 수 있다.)

1. Dockerfile을 만든다

(아무 위치나 관계없이 “Dockerfile”라는 이름의 파일을 생성한다.)

[root@localhost]# vi Dockerfile

|  |
| --- |
| FROM ubuntu:latest  RUN apt-get update && apt-get install -y locales  RUN locale-gen ko\_KR.UTF-8  ENV LC\_ALL ko\_KR.UTF-8 |

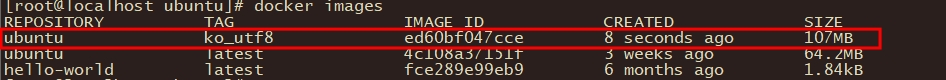
(Dockerfile 명령구문에 대해서는 정리되어 있는 웹사이트를 참조할 것)

1. Dockerfile을 빌드한다.

[root@localhost]# docker build -t <REPOSITORY:TAG> <Dockerfile위치>

|  |
| --- |
| [root@localhost]# docker build -t ubuntu:ko\_utf8 . |

1. Docker 이미지를 확인해 보면 새로 생성된 이미지를 볼 수 있다.



생성된 이미지를 이용해서 “docker run”을 실행시키면 locale이 설정된 우분투를 볼 수 있다.

# Docker로 서버 설치

## Docker hub 소개

공식 도커 이미지를 받을 수 있는 곳: <https://hub.docker.com/>

위 사이트에서 필요한 공식 이미지를 검색해서 다운받을 수 있다.

## DB설치

### MariaDB 설치

1. 도커 이미지 확인

<https://hub.docker.com/_/mariadb>

1. Dockerfile 생성

[root@localhost]# vi Dockerfile

|  |
| --- |
| FROM mariadb:10.4  RUN apt-get update  RUN apt-get install -y vim  RUN apt-get install -y locales  RUN locale-gen ko\_KR.UTF-8  ENV LC\_ALL ko\_KR.UTF-8 |

(MariaDB는 ubuntu를 사용함) 🡪 docker hub에서 확인할 것

1. Dockerfile 빌드

|  |
| --- |
| [root@localhost]# docker build -t mariadb:10.4\_dev . |

1. docker run 실행

|  |
| --- |
| [root@localhost]# docker run --name mariadb\_10.4\_dev \  -e MYSQL\_ROOT\_PASSWORD=mysql \  -p 3306:3306 \  -v /home/volumes/database/mariadb\_10.4\_dev:/var/lib/mysql \  -v /etc/localtime:/etc/localtime:ro \  -d mariadb:10.4\_dev |

* -v /home/volumes/database/mariadb\_10.4\_dev:/var/lib/mysql 🡺 이 옵션은 데이터 파일을 Host서버에 저장시킴
* -v /etc/localtime:/etc/localtime:ro 🡺 이 옵션은 Host의 시간대와 Docker의 시간대를 맞추기 위한 설정

1. 환경파일 설정 (utf8을 설정하기 위한 작업)

MariaDB에 접속 후 mariadb.cnf 환경설정 수정

[root@localhost]# docker exec -it mariadb\_10.4\_dev /bin/bash

root@9357fb6a3b11:/# vi /etc/mysql/mariadb.cnf

|  |
| --- |
| # MariaDB-specific config file.  # Read by /etc/mysql/my.cnf  [client]  # Default is Latin1, if you need UTF-8 set this (also in server section)  default-character-set = utf8  [mysqld]  #  # \* Character sets  #  # Default is Latin1, if you need UTF-8 set all this (also in client section)  #  character-set-server = utf8  collation-server = utf8\_general\_ci  character\_set\_server = utf8  collation\_server = utf8\_general\_ci  # Import all .cnf files from configuration directory  !includedir /etc/mysql/mariadb.conf.d/ |

root@9357fb6a3b11:/# exit (도커에서 빠져나가기)

1. docker 재시작

|  |
| --- |
| [root@localhost]# docker restart mariadb\_10.4\_dev |

1. MariaDB에 접속해서 시간과 utf8설정을 확인한다.

|  |
| --- |
| show variables like ‘c%’;  select now(); |

### PostgreSQL 설치

1. 도커 이미지 확인

<https://hub.docker.com/_/postgres>

1. Dockerfile 생성

[root@localhost]# vi Dockerfile

|  |
| --- |
| FROM postgres:11.4  RUN apt-get update  RUN apt-get install -y vim  RUN apt-get install -y locales  RUN sed -i ‘s/^# \(ko\_KR.UTF-8\)/\1/’ /etc/locale.gen  RUN localedef -f UTF-8 -i ko\_KR ko\_KR.UTF-8  ENV LC\_ALL ko\_KR.UTF-8 |

(PostgreSQL은 debian을 사용함) 🡺 locale설정이 다름

1. Dockerfile 빌드

|  |
| --- |
| [root@localhost]# docker build -t postgres:11.4\_dev . |

1. docker run 실행

|  |
| --- |
| [root@localhost]# docker run \  --name postgres\_11.4\_dev \  -e POSTGRES\_PASSWORD=postgres \  -p 5432:5432 \  -v /home/volumes/database/postgres\_11.4:/var/lib/postgresql/data \  -v /etc/localtime:/etc/localtime:ro \  -d postgres:11.4\_dev |

* -v /home/volumes/database/postgres\_11.4:/var/lib/postgresql/data 🡺 이 옵션은 데이터 파일을 Host서버에 저장시킴
* -v /etc/localtime:/etc/localtime:ro 🡺 이 옵션은 Host의 시간대와 Docker의 시간대를 맞추기 위한 설정

## Nginx 설치

### Nginx 설치

1. 도커 이미지 확인

<https://hub.docker.com/_/nginx>

1. Dockerfile 생성

[root@localhost]# vi Dockerfile

|  |
| --- |
| FROM nginx:1.16  RUN apt-get update  RUN apt-get install -y vim  RUN apt-get install -y locales  RUN sed -i ‘s/^# \(ko\_KR.UTF-8\)/\1/’ /etc/locale.gen  RUN localedef -f UTF-8 -i ko\_KR ko\_KR.UTF-8  ENV LC\_ALL ko\_KR.UTF-8 |

(nginx는 debian을 사용함)

1. Dockerfile 빌드

|  |
| --- |
| [root@localhost]# docker build -t nginx:1.16\_dev . |

1. docker run 실행

|  |
| --- |
| [root@localhost]# docker run --name nginx\_1.16\_dev \  -p 80:80 \  -v /home/volumes/nginx/html:/usr/share/nginx/html \  -v /etc/localtime:/etc/localtime:ro \  -d nginx:1.16\_dev |

* -v /home/volumes/nginx/html:/usr/share/nginx/html 🡺 이 옵션은 데이터 webapp파일을 Host에 올려서 동작 시키기 위해
* -v /etc/localtime:/etc/localtime:ro 🡺 이 옵션은 Host의 시간대와 Docker의 시간대를 맞추기 위한 설정

1. 환경 설정 파일

Nginx에서 설정할 환경파일을 생성한다. (default.conf)

[root@localhost]# vi default.conf

|  |
| --- |
| server {  listen 80;  server\_name localhost;  root /usr/share/nginx/html;  location / {  try\_files $uri $uri/ @rewrites;  }  location @rewrites {  rewrite ^(.+)$ /index.html last;  }  location ~\* \.(?:ico|css|js|gif|jpe?g|png)$ {  # Some basic cache-control for static files to be sent to the browser  expires max;  add\_header Pragma public;  add\_header Cache-Control "public, must-revalidate, proxy-revalidate";  }  error\_page 500 502 503 504 /50x.html;  location = /50x.html {  root /usr/share/nginx/html;  }  } |

Host서버에서 Docker컨테이너로 config파일을 카피한다.

|  |
| --- |
| [root@localhost]# docker cp default.conf nginx\_1.16\_dev:/etc/nginx/conf.d/default.conf |

(카피명령: # docker cp <원본파일> <DockerName>:<Docker내의파일위치>

(반대 방향의 카피도 가능함)

1. docker 재시작

|  |
| --- |
| [root@localhost]# docker restart nginx\_1.16\_dev |

### webapp 파일 업로드

1. Host의 “/home/volumes/nginx/html”위치에 webapp파일을 카피한다.

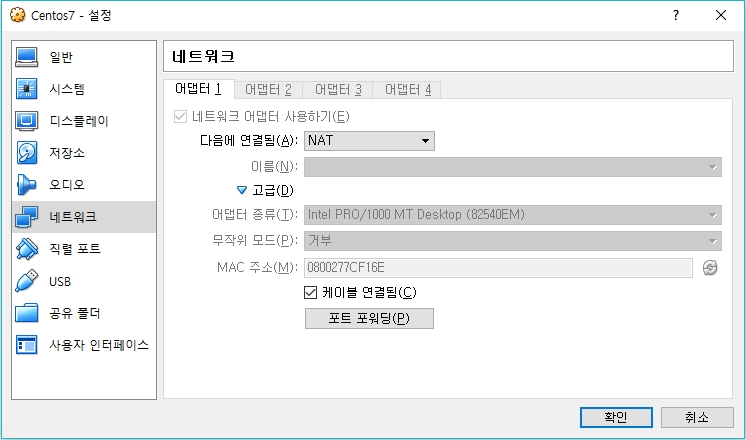
(개발 PC에서 ftp를 통해 업로드 하거나 Git에서 다운받으면 됨)

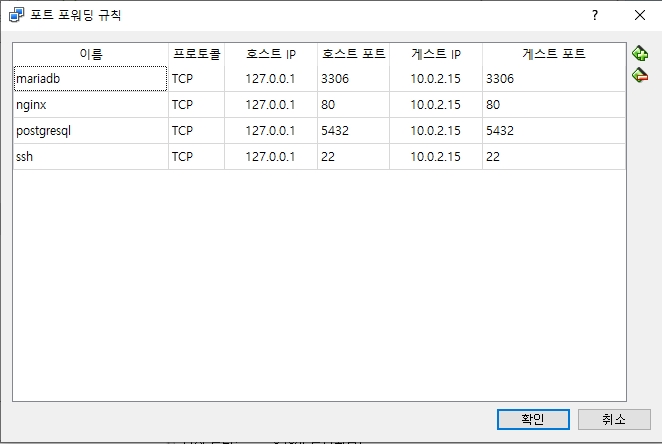
# VirtualBox에서 Docker 접속하기

## 포트 포워딩

외부 네트워크에서 VirtualBox의 네트워크 설정을 통해 Docker 컨테이너에 접속할 수 있다.

네트워크 방식: NAT





(호스트IP는 VirtualBox를 설치한 개발PC의 아이피를 뜻하고, 게스트IP는 Host OS로 사용한 CentOS7의 아이피를 뜻한다.)

(따라서 게스트 포트는 Docker에서 설정한 Host의 포트를 설정해 주어야 한다)

# On-premise 환경에서의 도커 사용

1. docker save를 이용하여 도커 이미지를 파일로 저장

[root@localhost]# docker save <옵션> <파일명> <이미지명>

|  |
| --- |
| [root@localhost]# docker save -o nginx.tar nginx:1.16\_dev |

1. docker load를 이용하여 도커 이미지파일을 도커 이미지로 로드

[root@localhost]# docker load -i <파일명>

|  |
| --- |
| [root@localhost]# docker load nginx.tar |