

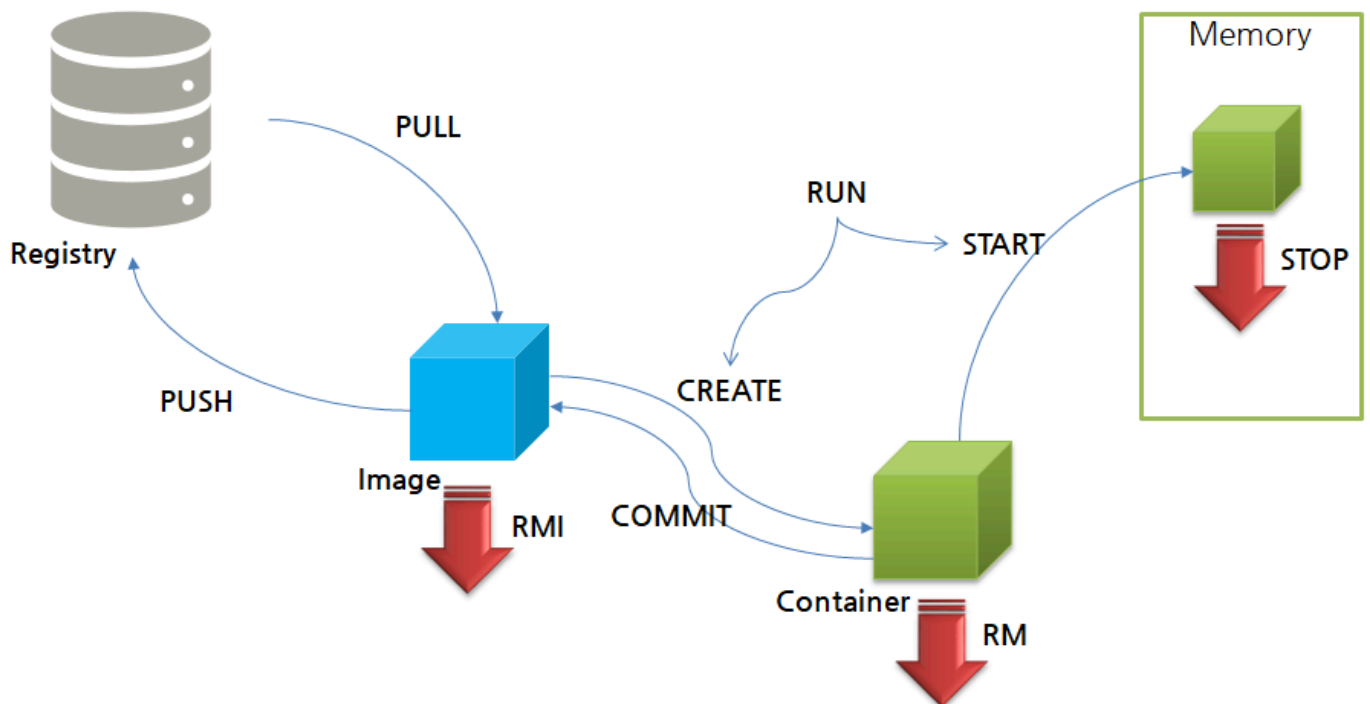
Docker 자주 쓰는 명령어 정리 (Cheatsheet)

🕒 6 분 소요

나의 업무나 과업을 편하게 만들어 주는 여러 기술중에 가장 혁신적인 한 가지를 꼽자면 고르기는 쉽지 않겠지만 바로 Docker 이다. 과거 vmware나 virtualbox를 설치하여 무겁고 번거롭게 가상화 환경을 구축하였다면 현재는 Docker 로 간편하고 빠르게 원하는 환경을 구축한다. 하지만 항상 헛갈리거나 잊어먹는 명령어들이 있어 아래의 포스팅으로 정리를 해놓고자 한다. (일종의 치트시트랄까?)

이 포스팅의 모든 내용은 왕초보도 따라하는 도커 기초 강의 (https://youtube.com/playlist?list=PLnlaYcDMsSczk-byS2iCDmQCfVU_KHWDk) 를 참고하여 작성한 것이다.

Docker 라이프사이클



- `docker pull` : Registry로부터 Docker Image를 가져온다.
- `docker push` : Registry로 Docker Image를 업로드한다.
- `docker create` : Docker Image로부터 컨테이너를 만든다.
- `docker commit` : 컨테이너로부터 Docker Image를 만든다.
- `docker run` : 컨테이너를 생성한다. (CREATE + START)
- `docker start` : 컨테이너를 다시 실행시킨다.
- `docker stop` : 컨테이너를 정지시킨다.
- `docker rm` : 컨테이너를 삭제한다.
- `docker rmi` : Docker Image를 삭제한다.
- `docker attach` : 실행중인 컨테이너에 접속한다.

도커 명령어 정리

포트포워딩으로 톰캣 실행

tomcat 이미지를 받은 뒤에 `tc` 라는 이름(-name)의 컨테이너로 내부 8080 포트를 호스트의 80 포트로 포워딩(-p) 하여 백그라운드(-d) 실행한다.

```
sudo docker pull consol/tomcat-7.0
sudo docker run -d --name tc -p 80:8080 tomcat
```

컨테이너 내부 쉘 실행

bash 쉘을 실행하여 컨테이너 내부의 쉘 접속

```
sudo docker exec -it tc /bin/bash
```

컨테이너 로그 확인

컨테이너의 구동 이후 로그를 확인할 수 있다. 로그는 해당 컨테이너의 `stdout`, `stderr` 출력

```
sudo docker logs tc
```

호스트 및 컨테이너 간 파일 복사

호스트<->컨테이너, 컨테이너<->컨테이너 간의 파일 복사가 가능하다.

```
sudo docker cp <path> <to container>:<path>
sudo docker cp <from container>:<path> <path>
sudo docker cp <from container>:<path> <to container>:<path>
```

도커 컨테이너 모두 삭제

보통은 컨테이너의 name이나 id값으로 해당 컨테이너를 삭제하나 아래의 명령어를 통해 한번에 모든 컨테이너를 삭제할 수 있다.

참고로 `docker ps -a -a` 를 하게 되면 모든 컨테이너의 id가 출력되게 된다.

```
sudo docker stop `sudo docker ps -a -q`
sudo docker rm `sudo docker ps -a -q`
```

도커 이미지 모두 삭제

위의 명령어는 도커 이미지에도 적용된다. 따라서 모든 도커 이미지를 삭제하고 싶으면 아래의 명령어를 사용한다.

```
docker rmi `docker images -q`
```

임시 컨테이너 생성

컨테이너가 stop 시 자동으로 삭제되도록 옵션을 줄 수 있다.(-rm)

```
sudo docker run -d -p 80:8080 --rm --name tc tomcat
```

실습 1. 도커 이미지 받아서 서비스 띄워보기

한 가지 실습을 해보자. Jenkins 라는 서비스가 있는 이미지를 검색 후 받아서 컨테이너로 실행시키고 컨테이너 내부 포트를 외부로 포트포워딩 시켜 접속해보고자 한다.

1. 기존에 설치된 모든 컨테이너와 이미지 정지 및 삭제

```
sudo docker stop `sudo docker ps -a -q`
sudo docker rm `sudo docker ps -a -q`
sudo docker rmi `sudo docker images -q`
```

2. Jenkins 이미지 검색

```
sudo docker search jenkins
```

아래와 같이 여러 이미지들이 출력된다. 우리는 이 목록에서 `jenkins/jenkins` 를 설치해보겠다.

```
root@server1-VirtualBox:~# docker search jenkins
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
jenkins	Official Jenkins Docker image	5144	[OK]	
jenkins/jenkins	The leading open source automation server	2483		
jenkinsci/blueocean	https://jenkins.io/projects/blueocean	606		
jenkinsci/jenkins	Jenkins Continuous Integration and Delivery ...	388		
jenkins/jnlp-slave	a Jenkins agent which can connect to Jenkins...	139		[OK]
jenkinsci/jnlp-slave	A Jenkins slave using JNLP to establish conn...	129		[OK]
jenkinsci/slave	Base Jenkins slave docker image	65		[OK]
jenkins/slave	base image for a Jenkins Agent, which includ...	44		[OK]
jenkinsci/ssh-slave	A Jenkins SSH Slave docker image	43		[OK]
jenkins/ssh-slave	A Jenkins slave using SSH to establish conn...	36		[OK]
cloudbees/jenkins-enterprise	CloudBees Jenkins Enterprise (Rolling releas...	34		[OK]
bitnami/jenkins	Bitnami Docker Image for Jenkins	31		[OK]
hikkan/jenkins-docker	Extended Jenkins docker image, bundled wi...	29		
xmartlabs/jenkins-android	Jenkins image for Android development.	28		[OK]
openshift/jenkins-2-centos7	A Centos7 based Jenkins v2.x image for use w...	23		
cloudbees/jenkins-operations-center	CloudBees Jenkins Operation Center (Rolling ...	14		[OK]
vfarci/jenkins-swarm-agent	Jenkins agent based on the Swarm plugin	8		[OK]
openshift/jenkins-slave-base-centos7	A Jenkins slave base image. DEPRECATED: see ...	7		
trion/jenkins-docker-client	Jenkins CI server with docker client	6		[OK]
publicisworldwide/jenkins-slave	Jenkins Slave based on Oracle Linux	5		[OK]
openshift/jenkins-1-centos7	DEPRECATED: A Centos7 based Jenkins v1.x ima...	4		
ansibleplaybookbundle/jenkins-apb	An APB which deploys Jenkins CI	1		[OK]
masshape/jenkins	Just a jenkins image with the AWS cli added ...	0		[OK]
jameseckersall/jenkins	docker-jenkins (based on openshift jenkins 2...	0		[OK]
amazeeio/jenkins-slave	A jenkins slave that connects to a master vi...	0		[OK]

```
root@server1-VirtualBox:~#
```

3. jenkins/jenkins 도커 이미지 받기

```
docker pull jenkins/jenkins
```

4. 해당 서비스가 어떤 포트를 사용하는 지 확인

호스트와 컨테이너 간 포트포워딩을 위해 어떤 포트들을 사용하고 있는지 찾아봐야 한다. `docker inspect` 라는 명령어를 통해 이미지에 대한 설명을 볼 수 있다.

```
sudo docker inspect jenkins
```

```
"DockerVersion": "20.10.6",
"Author": "",
"Config": {
  "Hostname": "",
  "Domainname": "",
  "User": "jenkins",
  "AttachStdin": false,
  "AttachStdout": false,
  "AttachStderr": false,
  "ExposedPorts": {
    "50000/tcp": {},
    "8080/tcp": {}
  },
  "Tty": false,
  "OpenStdin": false,
  "StdinOnce": false,
  "Env": [
    "PATH=/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "LANG=en_US.UTF-8",
    "LANGUAGE=en_US:en",
    "LC_ALL=en_US.UTF-8",
    "..."
  ]
}
```

이미지에 대한 설명이 길게 나오지만 아래 ExposedPorts 부분이 우리가 참고할 부분이다. 2개의 포트를 사용하고 있고 우리가 포트 포워딩할 포트는 8080/tcp 이다.

5. Jenkins 컨테이너 구동

백그라운드로 실행시키고 컨테이너의 8080 포트를 호스트의 8080 포트와 포워딩 해준다.

```
docker run -d -p 8080:8080 --name jk jenkins/jenkins
```

구동한 이후 브라우저를 통해 웹을 접속해서 아래와 같은 화면이 뜨면 성공이다. 다만 초기 패스워드가 필요한 상태이다.

Getting Started


Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password



Continue

6. 초기 패스워드 찾기

패스워드를 찾기 위해서는 크게 2가지 방법이 있다.

1. 초기 웹페이지에 명시된 경로(/var/jenkins_home/secrets/initialAdminPassword) 찾아가기
2. `docker log` 보기

2가지 방법을 모두 이용해 보겠다.

먼저 컨테이너 내부에 접속해야 한다. 혹은 `cat` 을 이용하여서 바로 출력해보도록 하자.

```
sudo docker exec -it jk cat /var/jenkins_home/secrets/initialAdminPassword
```

```
root@server1-VirtualBox:~# docker exec -it jk cat /var/jenkins_home/secrets/initialAdminPassword
10447e7302f44dbd8def0d8fceb895cf
```

위와 같이 출력이 됨을 확인할 수 있다. 그러면 `docker log` 를 통해 나오는 패스워드도 동일한지 확인해 주자.

```
sudo docker logs jk
```

```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

10447e7302f44dbd8def0d8fceb895cf

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
```

위에서 확인한 패스워드와 동일함을 확인할 수 있다.

실습 2. 환경 변수 사용하여 MySQL 서비스 구축

앞선 실습에서는 설정되어 있는 패스워드를 사용하였지만 환경변수를 이용해 패스워드 값을 설정하여 컨테이너로 넘길 수 있다.(-e) MySQL 을 설치하면서 해당 과정을 정리해보자.

먼저 `mysql` 이미지를 받는다.

```
sudo docker pull mysql
```

도커 허브에서 [MYSQL](https://hub.docker.com/_/mysql) (https://hub.docker.com/_/mysql). 관련된 설명을 보면 컨테이너 구동시 `MYSQL_ROOT_PASSWORD` 라는 환경변수에 값을 넘기라고 말하고 있다.

```
sudo docker run --name mysql -e MYSQL_ROOT_PASSWORD=test1234 -d mysql
```

실행 후에 실제로 해당 패스워드를 이용하여 컨테이너 내부의 mysql에 접속해 보자.

```
sudo docker exec -it mysql mysql -u root -p
```

설정했던 패스워드인 `test1234` 로 접속이 가능함을 확인할 수 있다.

```
root@server1-VirtualBox:~# docker exec -it mysql mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.23 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.00 sec)

mysql> █
```

실습 3. 볼륨 마운트하여 Jupyter Lab 서비스 구축

호스트와 컨테이너 간의 파일시스템을 공유해야 할 때도 있다. 이럴 때 호스트의 볼륨을 컨테이너로 마운트 할 수 있는 옵션이 존재한다.(-v) `nginx` 환경을 구축함으로써 해당 실습을 해보자.

볼륨을 마운트하는 명령어의 형식은 아래와 같다.

```
docker run -v <호스트 경로>:<컨테이너 내 경로>:<권한>
```

주의 할 것은 컨테이너 내 경로뒤에 꼭 권한을 명시해 주어야 한다는 것이다. 권한의 종류는 아래와 같다.

- ro : Read Only
- rw : Read and Write

먼저 호스트에 컨테이너와 볼륨 마운트 할 경로를 확인한다. 필자는 Ubuntu 시스템에서 실습을 진행하였으며 해당 경로를 직접 만들어 주었다.

```
sudo mkdir /var/www
```

다음으로 nginx 컨테이너를 받아온 뒤에 아래의 명령어로 볼륨 마운트를 한 컨테이너를 실행해준다.

컨테이너 내부의 /usr/share/nginx/html 경로를 ro 권한으로 마운트 할 것이다.

```
sudo docker pull nginx
sudo docker run -d -p 80:80 --rm -v /var/www:/usr/share/nginx/html:ro nginx
```

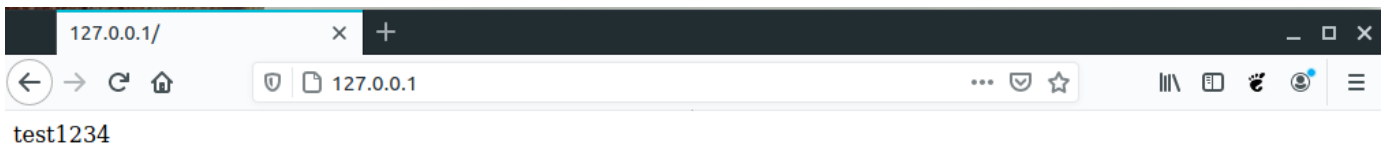
컨테이너가 정상적으로 구동됨을 확인하고 웹 페이지(http://127.0.0.1)를 들어가보면 아래와 같은 403 Forbidden 에러를 확인할 수 있다. 마운트 한 폴더에 아무런 웹 페이지가 없어서 나오는 에러이다.



호스트에서 마운트한 해당 경로(/var/www)에 간단한 index.html 페이지를 생성하고 test1234 라는 내용을 넣어보자.

```
touch /var/www/index.html
echo test1234 > /var/www/index.html
```

이후 웹 브라우저를 다시 새로고침하면 아래와 같이 호스트에서 수정했던 내용이 정상적으로 출력됨을 확인할 수 있다.



실습 4. 풀스택 워드프레스 이미지 만들기



워드프레스는 간단하게 웹 페이지를 구축할 수 있어 전 세계적으로 많이 사용되었던 CMS(웹 콘텐츠 관리 시스템) 소프트웨어이다. 이 역시 기본적으로 Docker Hub에서 제공하는 도커 이미지가 있으나 구축에 필요한 DB인 MySQL 은 따로 떨어져 있어 구축이 불편하다.

여기서는 하나의 컨테이너에서 워드프레스와 MySQL을 동시에 동작할 수 있도록 컨테이너로 만들어보고 이를 도커 이미지로 만들어 Docker Hub에 업로드 해보는 실습을 해 볼 것이다.

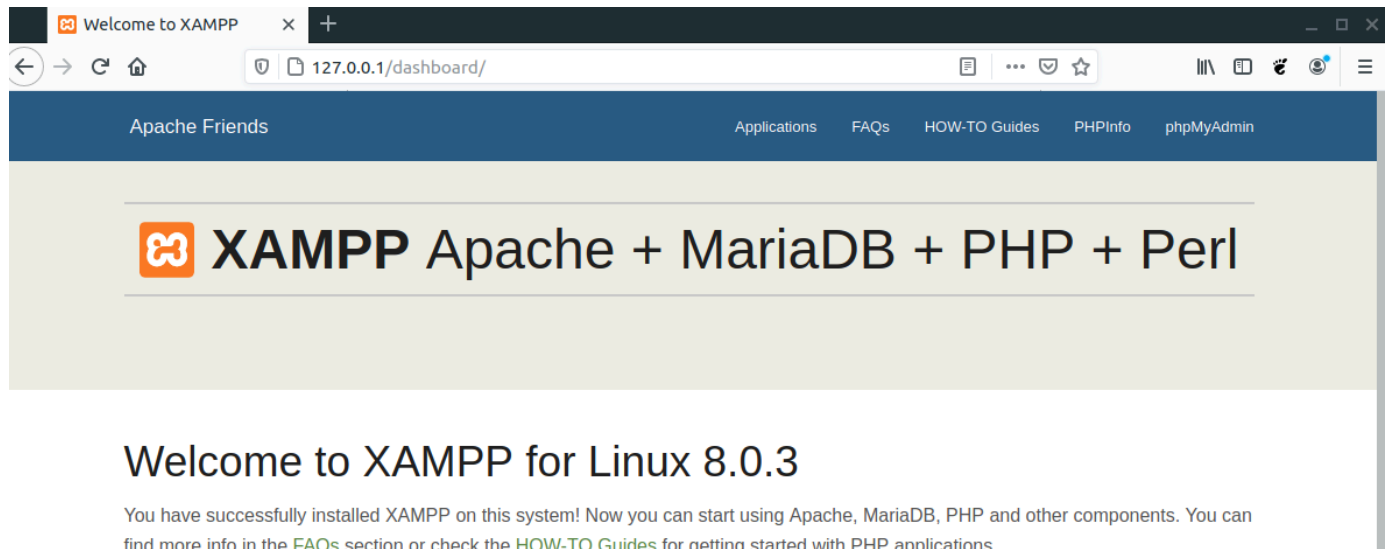
먼저 PHP 와 MySQL 이 공존하는 환경을 찾아야 한다. XAMPP (<https://hub.docker.com/r/tomsik68/xampp>) 라는 이미지에는 apache , MariaDB , php 가 설치되어 있다. 여기에 워드프레스만 올리면 바로 컨테이너를 구축할 수 있다.

먼저 해당 이미지를 pull 한 후, run 해 본다. (run 구동 시 로컬에 해당 이미지가 없으면 자동으로 pull 하므로 pull 과정은 생략)

내부적으로 많은 포트를 Listen 하고 있으나 외부에서 접속할 포트인 80 포트만 포트포워딩 해준다.

```
sudo docker run --name wp -p 80:80 -d tomsik68/xampp
```

웹 브라우저를 통해 정상 접속을 확인해본다.



환경 구축이 끝났다. 이제 여기다가 워드프레스만 올리면 되는 것이다. (내가 도커를 정말 혁신적이라고 생각하는 이유이다. 심지어 설치가 다 된 나만의 워드프레스 이미지를 만들 수도 있다니)

워드프레스 홈페이지(<https://ko.wordpress.org/download/>)에서도 다운이 가능하지만 클릭 한 번도 귀찮기에 `curl` 이나 `wget` 을 통해 받아온다. 이후 압축을 해제하면 `wordpress`라는 폴더가 생성될 것이다.

```
wget https://ko.wordpress.org/latest-ko_KR.tar.gz
tar -xf latest-ko_KR.tar.gz
```

이 폴더안의 내용을 `xampp` 컨테이너에 넣어주면 된다. 이전에 컨테이너 안의 웹 루트 디렉토리의 권한을 설정하고 파일을 깨끗하게 정리해주자.

기존의 파일은 지워도 상관없으나 혹시 모르니 `backup` 폴더에 백업해놓았다.

```
sudo docker exec -it wp bash
chown daemon. /opt/lampp/htdocs
cd /opt/lampp/htdocs/
mkdir backup
mv * ./backup/
exit
```

```

root@server1-VirtualBox:~# docker exec -it WP bash
root@1ebe71e6f506:/# chown daemon. /opt/lampp/htdocs
root@1ebe71e6f506:/# cd /opt/lampp/htdocs/
root@1ebe71e6f506:/opt/lampp/htdocs# ls
applications.html bitnami.css dashboard favicon.ico img index.php webalizer www
root@1ebe71e6f506:/opt/lampp/htdocs# mkdir backup
root@1ebe71e6f506:/opt/lampp/htdocs# mv * ./b
backup/          bitnami.css
root@1ebe71e6f506:/opt/lampp/htdocs# mv * ./backup/
mv: cannot move 'backup' to a subdirectory of itself, './backup/backup'

root@1ebe71e6f506:/opt/lampp/htdocs#
root@1ebe71e6f506:/opt/lampp/htdocs# ls
backup
root@1ebe71e6f506:/opt/lampp/htdocs# exit
exit
root@server1-VirtualBox:~#

```

이후 워드프레스 파일을 컨테이너에 복사하고 웹 루트 디렉토리에 배치한다.

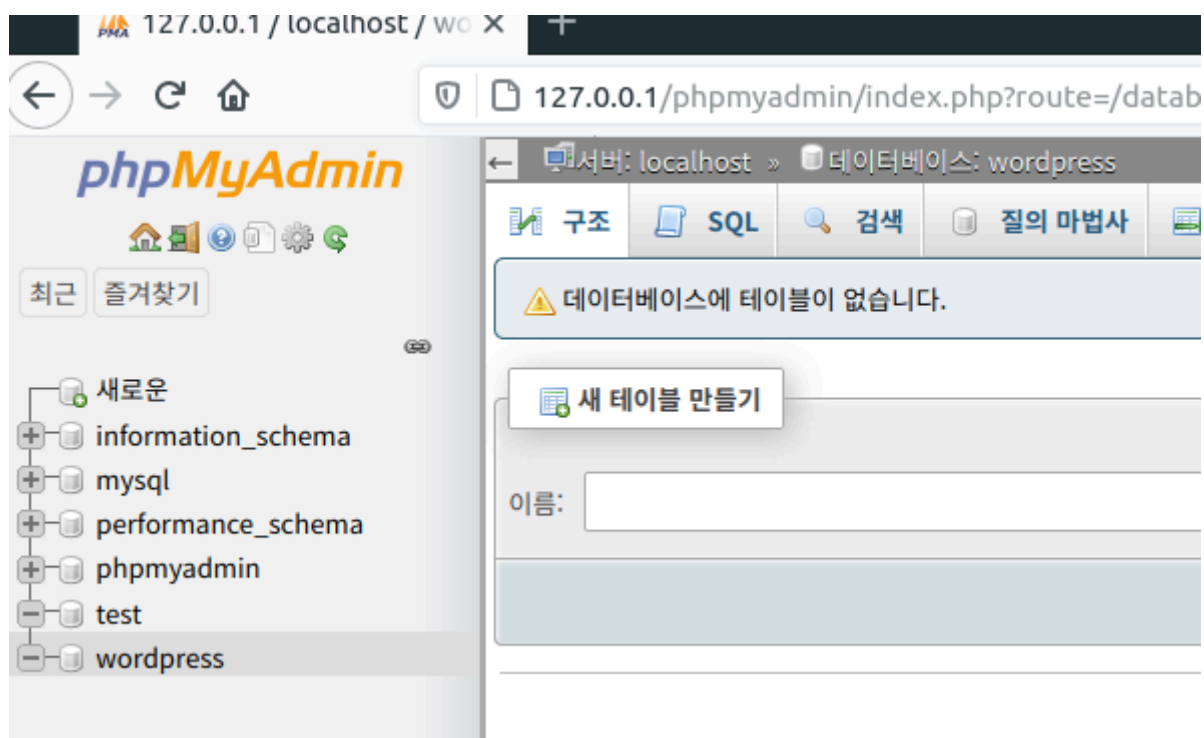
그리고 컨테이너를 재시작해준다.

```

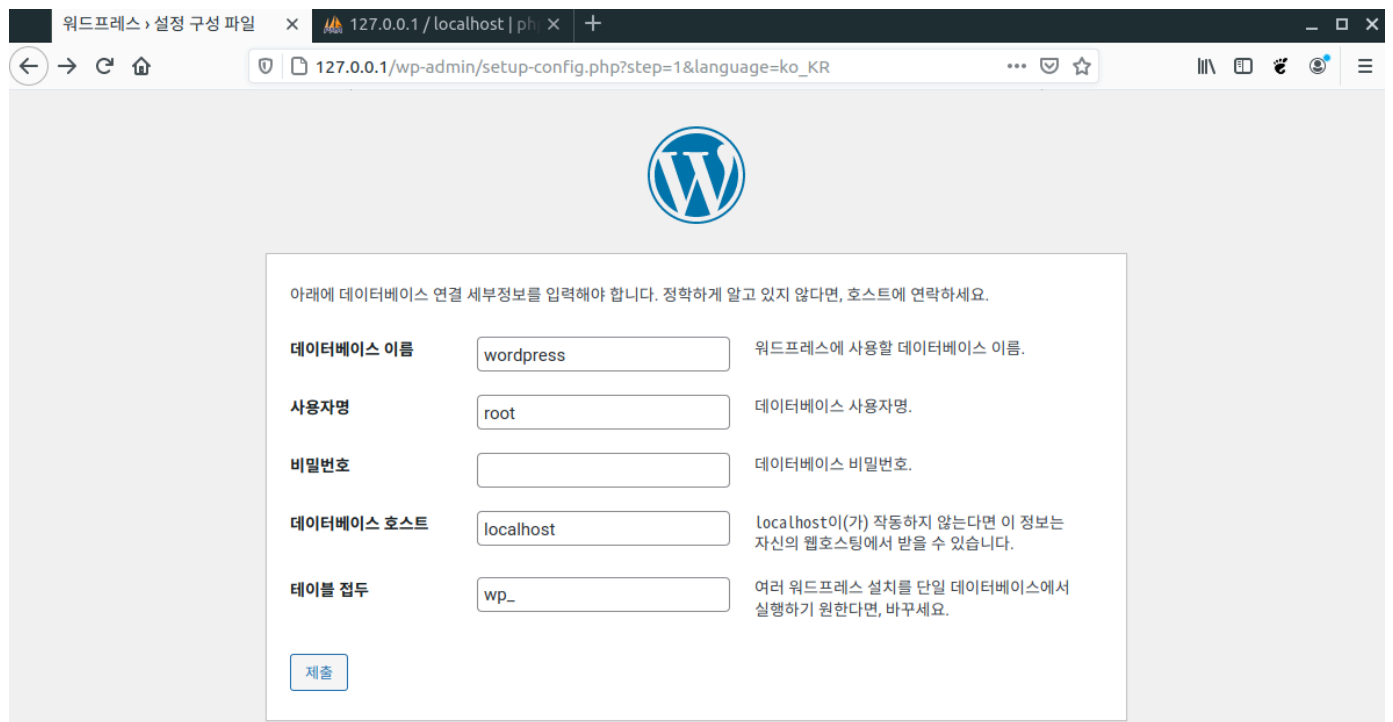
sudo docker cp wordpress wp:/opt/lampp/htdocs
sudo docker exec -it wp bash
mv /opt/lampp/htdocs/wordpress/* /opt/lampp/htdocs/
exit
sudo docker restart wp

```

웹 브라우저를 켜고 <http://127.0.0.1/phpmyadmin>으로 접속한뒤에 워드프레스에서 사용한 데이터베이스를 생성해주자. 이름은 wordpress로 하면된다.



이후에 다시 http://127.0.0.1 로 접속하여 워드프레스 설치를 진행하면 된다. phpmyadmin의 사용자계정인 root를 입력하고 패스워드는 설정된 값이 없으므로 빈 값을 입력해주면 된다.



워드프레스 > 설정 구성 파일 x 127.0.0.1 / localhost | php x +

127.0.0.1/wp-admin/setup-config.php?step=1&language=ko_KR

아래에 데이터베이스 연결 세부정보를 입력해야 합니다. 정확하게 알고 있지 않다면, 호스트에 연락하세요.

데이터베이스 이름	<input type="text" value="wordpress"/>	워드프레스에 사용할 데이터베이스 이름.
사용자명	<input type="text" value="root"/>	데이터베이스 사용자명.
비밀번호	<input type="password"/>	데이터베이스 비밀번호.
데이터베이스 호스트	<input type="text" value="localhost"/>	localhost이(가) 작동하지 않는다면 이 정보는 자신의 웹호스팅에서 받을 수 있습니다.
테이블 접두	<input type="text" value="wp_"/>	여러 워드프레스 설치를 단일 데이터베이스에서 실행하기 원한다면, 바꾸세요.

이 후에 워드프레스에 필요한 정보를 입력하여 준다.

워드프레스 > 설치

127.0.0.1 / localhost | php

127.0.0.1/wp-admin/install.php?language=ko_KR

환영합니다

인기있는 5분 워드프레스 설치 절차에 오신 것을 환영합니다! 아래에 있는 정보를 채우기만 하면 세계 최고의 확장성과 강력한 개인 발행 플랫폼을 사용할 수 있습니다.

정보가 필요합니다

다음 정보를 제공해주세요. 걱정하지 마세요. 이 설정을 나중에 언제든지 바꿀 수 있습니다.

사이트 제목

사용자명
사용자명은 알파벳, 숫자, 스페이스, 밑줄, 하이픈, 마침표, @ 기호만 가능합니다.

비밀번호 [숨기기](#)
매우 약함
중요: 로그인할 비밀번호가 필요할 것입니다. 안전한 위치에서 저장해주세요.

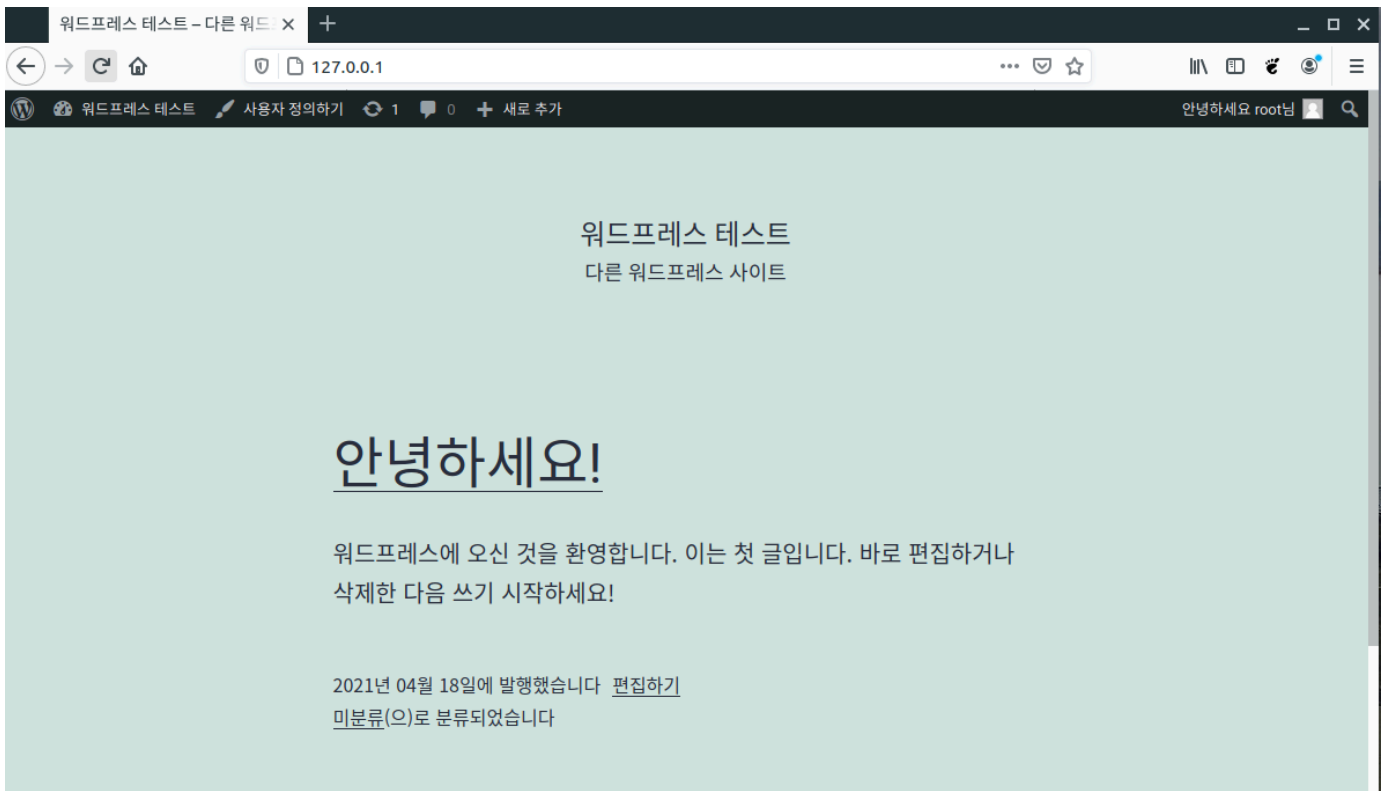
비밀번호 확인 ☒ 약한 비밀번호 사용 확인

이메일 주소
계속하기 전에 이메일 주소를 다시 확인하세요.

검색 엔진 가시성 ☐ 검색 엔진이 이 사이트를 검색하는 것을 차단
이 요청이 받아들여지는 것은 전적으로 검색 엔진에 좌우됩니다.

[워드프레스 설치](#)

모든 설정이 끝나면 워드프레스의 CMS 페이지로 리다이렉션되게 된다. 그곳은 게시글이나 워드프레스 설정을 관리하는 곳이고 `http://127.0.0.1/` 로 접속하게 되면 간단하게 그럴듯한 웹페이지가 설정되게 된다.



이제 해당 컨테이너를 그대로 이미지로 만들어서 개인 도커허브 계정에 올려보자.

먼저 컨테이너를 stop시킨다.

```
sudo bash stop wp
```

이후 커밋 명령어를 통해 해당 컨테이너를 이미지로 만들어준다. 참고로 도커 허브에 업로드하기 위해서는 아래와 같은 조건을 만족하여야 한다. 필자의 도커허브 ID는 inverlist 이므로 `inverlist/wordpress` 라는 이름으로 commit 해 주겠다.

1. Docker Hub 가입
2. 업로드 할 이미지명을 /<컨테이너명> 으로 설정

```
sudo docker commit wp inverlist/wordpress
```

아래와 같이 이미지를 생성하였다.

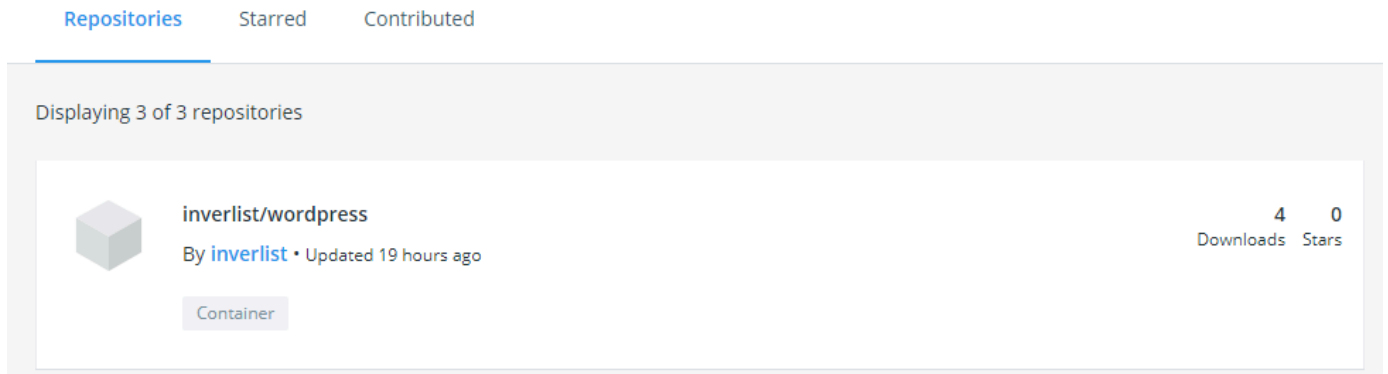
```
root@server1-VirtualBox:~# docker commit WP inverlist/wordpress
sha256:f7cdcc42d9e606177c45aadf1d54e648950516cdfbbb4e5cc2d6304785fc3c6b
root@server1-VirtualBox:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
inverlist/wordpress latest             f7cdcc42d9e6       6 seconds ago     1.56GB
127.0.0.1:5000/echo-test latest             222d1d580625       20 hours ago     877MB
```

도커 허브에 업로드를 위해서는 먼저 로그인을 해주어야 한다.

```
sudo docker login
```

이후 `docker push` 명령어를 통해 업로드 해준다. 아래의 명령어를 그대로 치면 안되고 `inverlist` 부분에 각자의 도커허브 아이디를 넣어야 한다.

```
sudo docker psuh inverlist/wordpress
```



이렇게 자신의 계정에 컨테이너를 이미지로 commit 하여 업로드가 가능하다.

다음번 포스팅에는 직접 도커 이미지를 빌드하거나 사설 repository를 구축하는 법 그리고 도커를 원격에서 사용하는 법에 대해 포스팅해보도록 하겠다.

Reference

- 왕초보도 따라하는 도커 기초 강의 (https://youtube.com/playlist?list=PLnlaYcDMsSczkybyS2iCDmQCfVU_KHWDk)
- 도커(Docker)치트 시트 (<https://gist.github.com/nacyot/8366310>)

태그: 2021 Container Docker Hub Docker

카테고리: Cheat sheet Docker

업데이트: April 18, 2021

댓글남기기

0 Comments - powered by utteranc.es