Hanbit RealTime 129

Docker로 PaaS 구성하기

오스카 하네 지음 / 이기곤 옮김

한빛미디어 Hanbit Media, Inc.

Docker로 PaaS 구성하기

오스카 하네 지음 / 이기곤 옮김

한빛미디어 Hanbit Media, Inc.

표지 사진 **임희진**

이 책의 표지는 임희진님이 보내 주신 풍경사진을 담았습니다.
리얼타임은 독자의 시선을 담은 풍경사진을 책 표지로 보여주고자 합니다.

사진 보내기 **ebookwriter@hanbit.co.kr**

Docker로 PaaS 구성하기

초판발행 2016년 5월 26일

지은이 오스카 하네 / **옮긴이** 이기곤 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 9월 30일 제10-1779호
ISBN 978-89-6848-821-4 95000 / **정가** 13,000원

총괄 전태호 / **책임편집** 김창수 / **기획·편집** 정지연
디자인 표지/내지 여동일, 조판 최송실
마케팅 박상용, 송경석, 변지영 / **영업** 김형진, 김진불, 조유미

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지은이_ 오스카 하네

오스카 하네Oskar Hane는 15년 동안 웹 애플리케이션을 개발하고 배포해온 풀스택Full Stack 개발자입니다. 그는 이 기간에 대부분 시간을 스타트업과 작지만 빠르게 성장하는 회사에서 일해왔습니다. 또한, 몇몇 회사의 공동 창업자이며, 지난 몇 년 동안은 독립 계약자independent contractor로서 일해왔습니다. 최근에는 자바스크립트로 세계에서 가장 뛰어난 그래프 데이터베이스 중 하나인 Neo4j의 프런트엔드를 개발하고 있습니다.

그는 현재 아내, 딸과 함께 스웨덴에서 살고 있으며 프로그래밍뿐만 아니라 스포츠와 사냥, 낚시 같은 야외 활동을 좋아합니다.

옮긴이_ 이기곤

아이렌소프트(AirenSoft)에서 풀스택 개발자로 일하고 있습니다. 저서로는 『FFmpeg 라이브러리: 코덱과 영상 변환을 중심으로』, 역서로는 『도커 오케스트레이션: 애플리케이션 빌드, 테스트, 배포의 통합 관리』(이상 한빛미디어, 2015)가 있습니다.

Docker가 등장한 지 3년이라는 시간이 흘렀습니다. Docker는 기존에 사용하던 가상 솔루션들과 달리 namespace와 cgroups과 같은 리눅스 커널 기능으로 매우 효율적으로 격리된 가상 환경을 생성합니다. 이러한 이유로 Docker는 개발자들에게 큰 호응을 얻어왔으며 레드햇, 아마존과 같은 대기업에서도 Docker를 전적으로 지원해 왔습니다.

하지만 상용 환경에서 Docker를 사용하는 데는 많은 문제점이 있었습니다. Docker는 효율적으로 격리된 가상 환경을 생성할 수 있지만, 수많은 가상 환경을 생성하고 관리하기 위한 도구들은 매우 부족했습니다. 따라서 시간이 지날수록 Docker를 둘러싼 생태계는 Docker Compose, Crane과 같이 Docker를 더욱 효율적으로 사용할 수 있는 오케스트레이션 도구들로 가득 차게 되었습니다. 이 외에도 Ansible, Puppet, Chef와 같은 설정 자동화 도구들도 Docker와 함께 어우러져 사용되기 시작했습니다.

Docker는 비교적 최근에 등장한 오픈소스인 만큼 사용법이 매우 간단하고 활용하기도 쉽습니다. 하지만 오케스트레이션 도구를 사용하여 Docker를 활용하는 방법은 매우 다양한 데다 찾아보기도 어렵습니다.

이 책은 Docker를 상용 환경에서 사용하는 데 필요한 설정들과 Docker를 둘러싼 생태계에 관해 이야기를 풀어나갑니다. 여러분이 Docker를 활용하는 방법을 찾고 있다면 이 책이 그 방향을 잡는 데 큰 도움이 될 것입니다.

Docker는 프로세스들을 격리된 상태로 실행할 수 있는 소프트웨어 컨테이너Container 기술을 높은 수준의 API로 제공하는 오픈소스입니다. 애플리케이션이 다양한 리눅스 서버(물론 OS X와 윈도우도 포함합니다.)에서 실행될 수 있도록 컨테이너에 포장함으로써 개발자가 서버 설정과 데브옵스DevOps[01] 활동보다는 개발에 조금 더 집중하도록 도와줍니다.

이 책에서 다루는 내용

1장 Docker 설치에서는 컨테이너를 만드는 데 필요한 Docker를 설치하는 과정을 다룹니다.

2장 Docker 살펴보기에서는 Docker가 어떻게 작동하고 어떤 용어들이 사용되는지 살펴보며, 공용 이미지들을 소개합니다.

3장 첫 번째 PaaS 이미지에서는 PaaSPlatform as a Service[02]의 일부가 될 첫 번째 Docker 이미지 생성 방법을 설명합니다.

4장 데이터 보관과 명령어 인자 전달에서는 컨테이너에 저장된 데이터를 보관할 수 있는 방법들을 살펴보고, 어떻게 컨테이너 안으로 명령어 인자를 전달하는지 설명합니다.

5장 컨테이너 연결에서는 컨테이너들이 완벽한 하나의 플랫폼으로써 동작할 수 있게 연결하는 방법을 살펴봅니다. 그리고 여러 컨테이너로 이루어진 플랫폼을 쉽게 다룰 수 있는 두 가지 도구를 소개합니다.

6장 역방향 프락시 요청에서는 한 호스트 안에서 여러 컨테이너를 관리할 때 동일한

01 역자주: 개발자와 운영자가 애플리케이션을 빠르게 배포하기 위해 협력하는 개발 방법론

02 역자주: 플랫폼을 제공하는 서비스

포트를 사용하지 못하는 문제와 이 문제를 어떻게 해결할 수 있는지 설명합니다.

7장 PaaS를 통한 배포에서는 여러분이 구성한 PaaS에서 애플리케이션을 배포하는 과정을 설명합니다. 또한, mini-Heroku와 Dokku를 생성하는 방법도 함께 설명합니다.

8장 끝나지 않은 이야기에서는 앞 장에서 언급한 몇 가지 프로젝트를 소개합니다. 이 외에도 PaaS를 기반으로 한 Docker의 멋진 미래에 대해서도 살펴봅니다.

이 책을 위해 필요한 것들

- OS X, 리눅스 또는 윈도우가 설치된 PC/노트북
- 인터넷 연결

이 책이 필요한 독자

이 책은 하나의 서비스를 컨테이너들로 모듈화하고 각 컨테이너를 연결함으로써 하나의 거대한 플랫폼을 구성하는 방법을 배우길 원하는 사람들을 위한 책입니다. 아마 여러분은 Docker에 대해 들어보았지만, 설치해본 적이 없거나 사용해본 적이 없을지도 모릅니다. 또는 Docker를 설치했지만, 서비스를 분할하여 사용하지 않고 하나의 컨테이너에 모든 것을 담아서 사용하고 있을 수도 있습니다. 이러한 상황에서 이 책은 PaaS를 구성하기 위해 필요한 모든 지식과 통찰력을 줄 것입니다.

예제 코드 다운로드

예제 코드 파일은 다음 링크에서 다운로드할 수 있습니다.

- http://www.hanbit.co.kr/exam/2821

차례

chapter 3

첫 번째 PaaS 이미지

이제 직접 Dockerfile을 작성하여 Docker 레지스트리 허브를 통해 배포하고 배포된 Dockerfile로부터 컨테이너를 생성할 준비가 되었습니다. 이 장에서는 다음과 같은 내용을 다룹니다.

- 자신만의 새로운 이미지 생성
- Github을 통한 이미지 호스팅
- Docker 레지스트리 허브를 통한 이미지 배포

3.1 워드프레스 이미지

이 예제에서는 아파치를 웹 서버로 사용하는 공식 워드프레스 Docker 이미지를 기본 이미지로 사용합니다.

> NOTE
>
> 대량 트래픽의 사이트를 호스팅 할 계획이라면, 아파치 대신 NGINX 웹 서버를 사용하는 기본 이미지를 추천합니다. 필자는 NGINX와 Memcached, WP-FFPC 플러그인들을 기반으로 둔 워드프레스 사이트가 충분히 강력하다는 것을 경험했습니다. 하지만 이 설정은 꽤나 복잡한 데다가 이 책의 주제와는 거리가 멀기 때문에 관련된 설정에 대해서는 다루지 않습니다.

첫 번째로, MySQL 컨테이너를 실행하고 워드프레스 컨테이너와 함께 연결한 후 어떤 일이 일어나는지 살펴봅시다.

```
docker run --name mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql
docker run --name some-wordpress --link some-mysql:mysql -d -p 80 wordpress
```

-p 80 플래그로 Docker는 내부에서 사용하는 80 포트를 외부로 개방합니다. 호스트에 있는 어떤 포트가 내부의 80 포트와 연결되었는지는 [그림 3-1]과 같이 run docker ps 명령어의 PORTS 항목 또는 docker port <컨테이너 ID|이름> 명령어로 확인할 수 있습니다. 필자의 경우에는 호스트의 49154가 내부의 80 포트와 연결되었습니다.

그림 3-1 개방된 포트 확인

```
[ec2-user@ip-172-31-32-58 ~]$ docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=mysecretpassword -d mysql
9d16db0a7208a116871ddb653da5187b7cbae4e410c5de82574b2e7b51a87bd4
[ec2-user@ip-172-31-32-58 ~]$ docker run --name some-wordpress --link some-mysql:mysql -d -p 80 wordpress
454f047e2e7c12e7958cbbf75d2458ed1a8f5d92db19c685dc857735edbb40fa
[ec2-user@ip-172-31-32-58 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
454f047e2e7c        wordpress:latest    "/entrypoint.sh apac   42 seconds ago      Up 40 seconds       0.0.0.0:49154->80/tcp   some-wordpress
9d16db0a7208        mysql:latest        "/entrypoint.sh mysq   47 seconds ago      Up 46 seconds       3306/tcp                some-mysql
[ec2-user@ip-172-31-32-58 ~]$ docker port 454f 80
0.0.0.0:49154
[ec2-user@ip-172-31-32-58 ~]$ docker port some-wordpress 80
0.0.0.0:49154
[ec2-user@ip-172-31-32-58 ~]$
```

브라우저에 http://<호스트 IP>:<포트 번호>와 같은 형태의 URL를 입력합니다. Amazon EC2를 사용한다면 공용 도메인을 얻을 수 있으며 주소는 http://ec2-54-187-234-27.us-west-2.compute.amazonaws.com:49154와 같은 형식이 됩니다. URL을 입력하면 [그림 3-2]와 같이 앞에서 본 설치 화면이 나타납니다.

그림 3-2 워드프레스 설치 화면

워드프레스 설치 화면이 여러분을 반긴다면 이는 워드프레스와 MySQL 컨테이너가 정상적으로 실행되었다는 뜻입니다.

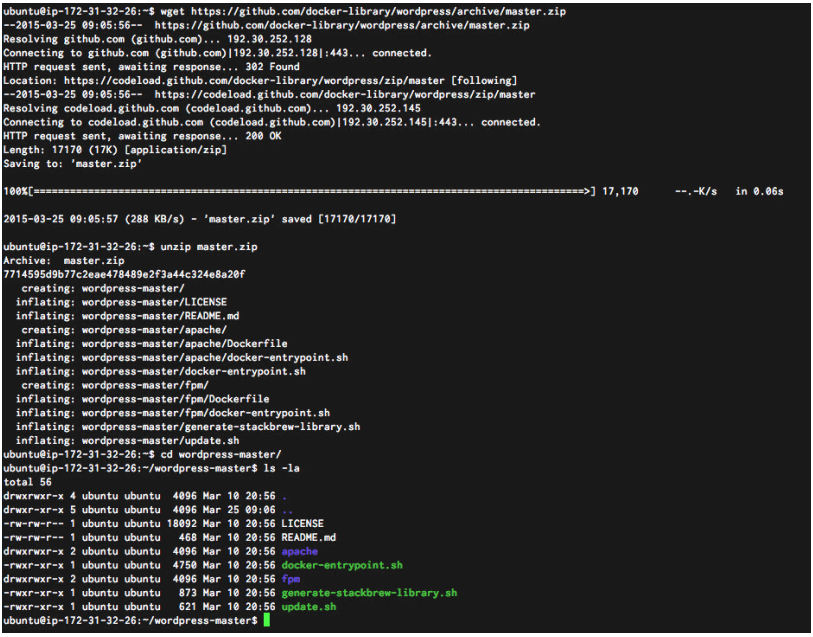
3.2 설정 추가

이제 아파치 위에서 기본적인 워드프레스 환경을 구축했습니다. 하지만 일부 워드프레스 플러그인은 웹 서버의 설정을 변경해야만 사용할 수 있습니다. 이렇게 워드프레스 디렉터리 안에 있는 파일을 변경할 때는 어떻게 해야 할까요?

가장 먼저 할 일은 공식 워드프레스 저장소를 복사하여 여러분의 저장소로 가져오는 것입니다. 이렇게 함으로써 Dockerfile이 어떻게 구성되었는지 확인해볼 수 있습니다. 이 책을 쓰는 시점에 워드프레스 이미지의 저장소는 https://github.com/docker-library/wordpress입니다. 이 링크는 Docker 레지스트리 허브의 워드프레스 저장소 페이지에서도 찾아볼 수 있습니다. 이제 이 링크를 클릭합니다.

Docker 이미지를 위해 이 저장소를 복사clone 또는 포크fork하거나 그냥 소스 전체를 다운로드할 수 있습니다. 여러분만의 Dockerfile을 생성한 이후에는 더 이상 사용하지 않으므로 방법은 중요하지 않습니다. 이 이미지는 단지 테스트와 분석을 위한 목적으로 사용됩니다. 필자의 경우에는 [그림 3-3]과 같이 EC2 인스턴스에서 소스를 다운로드하였습니다.

그림 3-3 워드프레스 공식 이미지 다운로드 화면



텍스트 편집기를 사용하여 Dockerfile을 엽니다. 필자의 경우에는 터미널에서 작업을 실행하기 때문에 `vi apache/Dockerfile` 명령어를 사용하여 파일을 열 수 있습니다. 현재 공식 워드프레스 이미지에서 사용하는 Dockerfile의 내용은 다음과 같습니다(2016년 5월 9일 업데이트 기준).

```
FROM php:5.6-apache

RUN a2enmod rewrite expires

# install the PHP extensions we need
RUN apt-get update && apt-get install -y libpng12-dev libjpeg-dev && rm -rf /
var/lib/apt/lists/* \
    && docker-php-ext-configure gd --with-png-dir=/usr --with-jpeg-dir=/usr \
    && docker-php-ext-install gd mysqli opcache

# set recommended PHP.ini settings
# see https://secure.php.net/manual/en/opcache.installation.php
RUN { \
```

```
        echo 'opcache.memory_consumption=128'; \
        echo 'opcache.interned_strings_buffer=8'; \
        echo 'opcache.max_accelerated_files=4000'; \
        echo 'opcache.revalidate_freq=60'; \
        echo 'opcache.fast_shutdown=1'; \
        echo 'opcache.enable_cli=1'; \
    } > /usr/local/etc/php/conf.d/opcache-recommended.ini

VOLUME /var/www/html

ENV WORDPRESS_VERSION 4.5.2
ENV WORDPRESS_SHA1 bab94003a5d2285f6ae76407e7b1bbb75382c36e

# upstream tarballs include ./wordpress/ so this gives us /usr/src/wordpress
RUN curl -o wordpress.tar.gz -SL https://wordpress.org/wordpress-${WORDPRESS_
VERSION}.tar.gz \
    && echo "$WORDPRESS_SHA1 *wordpress.tar.gz" | sha1sum -c - \
    && tar -xzf wordpress.tar.gz -C /usr/src/ \
    && rm wordpress.tar.gz \
    && chown -R www-data:www-data /usr/src/wordpress

COPY docker-entrypoint.sh /entrypoint.sh

# grr, ENTRYPOINT resets CMD now
ENTRYPOINT ["/entrypoint.sh"]
CMD ["apache2-foreground"]
```

공식 워드프레스 이미지는 php:5.6-apache 이미지를 기본 이미지로 사용하며 워드프레스 4.5.2를 다운로드한 후 /usr/src/wordpress 경로로 압축을 해제합니다. 이후에는 ENTRYPOINT 항목을 추가하고 Apache2 서버를 포그라운드foreground에서 실행합니다.

3.3 목표 설정

다운로드한 워드프레스 이미지를 목적에 맞게 사용하려면 Dockerfile을 3번 수정해야 합니다. 이번 예제의 목표는 다음과 같습니다.

- 아파치에 캐시 기능 추가(WP Super Cache 플러그인 사용)
- PHP와 아파치에 설정된 업로드 제한 변경
- 두 가지 플러그인 추가(WP Super Cache와 WP Mail SMTP)

3.3.1 캐시 기능 추가

WP Super Cache 플러그인을 사용하여 캐시 기능을 추가하는 데는 간단한 두 가지 작업이 필요합니다. 바로 아파치에서 'mod_headers와 mod_expires' 모듈을 활성화하는 것입니다.

아파치에서는 `a2enmod` 명령어로 모듈을 활성화할 수 있고, `a2dismod` 명령어로 비활성화할 수 있습니다. 원하는 모듈을 추가하는 일은 Dockerfile 안에 있는 명령어 뒤에 필요한 모듈을 추가하는 것만으로 간단하게 해결됩니다.

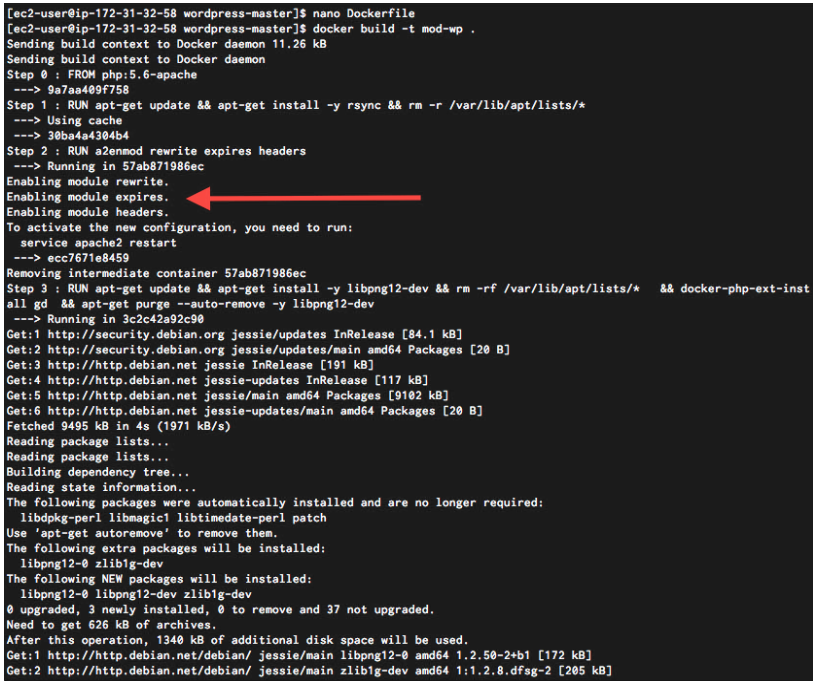
명령어를 수정하였으니 새로운 이미지를 빌드하여 어떤 일이 발생하는지 살펴봅시다. PHP를 소스로부터 컴파일하기 때문에 빌드 과정이 꽤 오랜 시간이 걸릴 수 있습니다. 여기서 우리가 살펴보아야 할 내용은 모듈이 제대로 활성화되었는지 확인하는 일입니다. 모듈이 제대로 활성화되었는지는 빌드를 실행한 후 수 초 후에 나타납니다.

수정한 Dockerfile로 빌드를 수행하기 위해 다음 명령어를 실행합니다.

```
docker build -t mod-wp ./apache/
```

`-t mod-wp` 플래그로 새로 생성된 이미지의 이름을 `mod-wp`로 지정합니다. [그림 3-4]은 앞의 과정을 자세하게 나타내고 있습니다. 별다른 에러 없이 빌드가 진행되었다면, WP Super Cache 플러그인을 사용하기 위한 준비가 되었습니다.

그림 3-4 아파치 모듈이 활성화된 Docker 이미지 빌드 화면



3.3.2 업로드 제한 변경

PHP에서는 기본적으로 파일 업로드 크기가 2MB로 제한되어 있습니다. 특히 모바일에서 블로그 글을 포스트한다면, 모바일에서 찍은 사진이나 비디오는 2MB를 넘는 일이 일반적이므로 이 제한은 너무 작다고 볼 수 있습니다. 우리는 블로그에 비디오를 직접 업로드하려고 하니 제한을 32MB로 변경해보겠습니다.

파일 업로드 제한을 변경하려면 PHP 설정 파일에서 `upload_max_filesize`와 `post_max_size` 파라미터 값이 변경되어야 합니다.

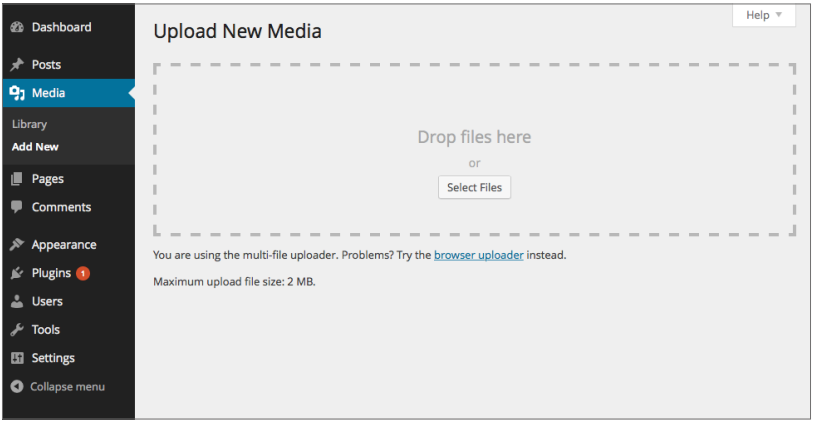
워드프레스 이미지에서 기본 이미지로 사용하는 php:5.6-Apache를 살펴보면, Dockerfile 안에서 데비안 리눅스와 PHP를 실행합니다. PHP 설정 파일은 기본적으로 /usr/local/etc/php/conf.d/ 디렉터리에 있습니다. 이는 즉, 이 디

렉터리에 파일을 추가하면 Dockerfile에서 이 파일을 읽어 설정을 적용한다는 뜻이 됩니다.

> NOTE
>
> PHP 5.6을 위한 Dockerfile은 https://github.com/docker-library/php/blob/master/5.6/Dockerfile에서 찾을 수 있습니다.

앞에서 말한 것처럼 업로드 제한이 낮다는 것을 확인하기 위해 [그림 3-5]와 같이 아무런 설정도 추가하지 않은 워드프레스 컨테이너를 실행한 후 Media 항목에서 [Add New] 버튼을 클릭합니다. [그림 3-5]에서 보이는 것과 같이 현재 업로드 제한이 2MB로 설정되어 있습니다.

그림 3-5 업로드 제한 확인



자 이제 앞에서 언급한 두 파라미터가 추가된 'upload-limit.ini'이라는 파일을 생성한 후 설정 디렉터리에 추가해봅시다. 다음 명령어들은 한 줄에 걸쳐 작성할 수 있습니다. 이 명령어를 앞서 캐시를 위해 수정한 명령어 바로 위에 추가합니다.

```
RUN touch /usr/local/etc/php/conf.d/upload-limit.ini \ && echo "upload_max_
filesize = 32M" >> /usr/local/etc/php/conf.d/upload-limit.ini \ && echo "post_
max_size = 32M" >> /usr/local/etc/php/conf.d/upload-limit.ini
```

```
#Paste above this line.
RUN a2enmod rewrite expires headers
```

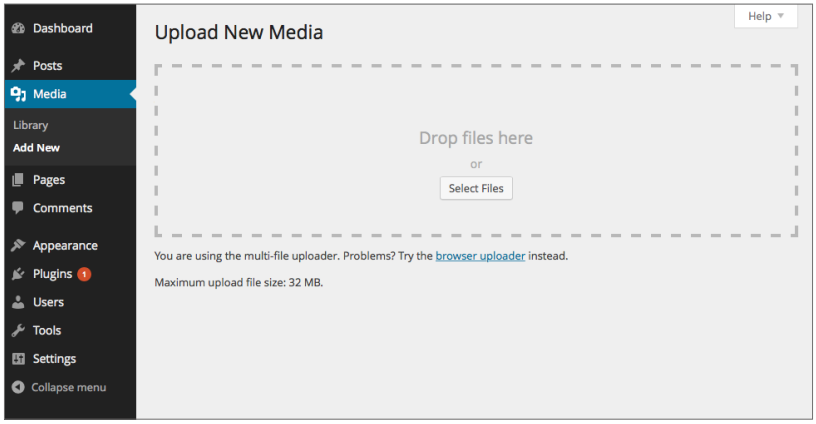
이제 다시 한 번 이미지를 빌드하여 오류 없이 정상적으로 빌드되는지 확인합니다. 이미지 이름이 이미 존재한다는 오류를 보게 된다면 `docker rmi mod-wp` 명령어로 이미지를 삭제하거나 새로 빌드하려는 이미지의 이름에 `mod-wp:latest`와 같이 태그를 붙여 최신 버전을 의미하도록 만들 수 있습니다.

빌드가 끝나면 워드프레스 관리자 인터페이스에서 업로드 제한이 변경되었는지 확인하기 위해 새로운 컨테이너를 생성합니다. 새로 생성된 이미지로부터 컨테이너를 생성하는 명령어는 다음과 같습니다.

```
docker run --name mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql
docker run --name some-wordpress --link some-mysql:mysql -d -p 80 mod-wp:latest
```

이제 더 큰 파일들을 업로드할 수 있게 되었습니다. 2MB가 넘는 파일을 직접 업로드하여 변경사항이 잘 적용되었는지 확인해보기 바랍니다.

그림 3-6 변경사항이 적용된 워드프레스 업로드 화면



한빛 리얼타임

한빛 리얼타임은 IT 개발자를 위한 전자책입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 후져도 조금이나마 정리된 정보를 찾기도 쉽지 않습니다. 또한, 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1 eBook First – 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다

500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 좀 더 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한, 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2 무료로 업데이트되는 전자책 전용 서비스입니다

종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3 독자의 편의를 위해 DRM-Free로 제공합니다

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 어려운 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가겠습니다.