Küresel Mühendislik Çalışmaları Dergisi e-ISSN: 2149-2735 Cilt 3: Sayı:2 (2016) 10-15

DOCKER İLE WEB UYGULAMASI GELİŞTİRME[1]

WEB APPLICATION DEVELOPMENT WITH DOCKER

The Journal of Global Engineering Studies e-ISSN: 2149-2735 Volume: 3 Issue:2 (2016) 10-15

Emre Can YILMAZ[a*] Recai OKTAŞ[b] Bünyamin KARABULUT[c]

[a]Ondokuz Mayıs Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü 55200 Samsun
ecylmz@bil.omu.edu.tr
[b]Ondokuz Mayıs Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü 55200 Samsun
roktas@bil.omu.edu.tr
[c]Ondokuz Mayıs Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü 55200 Samsun
bunyamin.karabulut@bil.omu.edu.tr

Özet

Sanallaştırma teknolojilerinin gelişimiyle birlikte, uygulama geliştirmede farklı yöntemler ortaya çıkmaktadır. Bu teknolojilerden biri olan Docker'ın son dönemde tanınırlığı ve kullanımı hızla artmaktadır. Bunun temel nedeni Docker'ın sistem kaynaklarını verimli kullanılmasıdır. Bilişim teknolojilerinde Docker uygulamasının çeşitli kullanım alanları mevcut olmakla birlikte, güncel durumda özellikle web uygulamaları öne çıkmaktadır. Bu çalışmada Docker ve Docker Compose uygulamaları aracılığı ile örnek bir web uygulama geliştirme yöntemi gösterilmiştir. Bu yöntem web uygulama geliştirme süreçlerini hızlandırmakta olup, geliştiricilere kolaylık sağlamaktadır.

Anahtar Kelimeler: Docker, Sanallaştırma, Web uygulaması, Ruby on Rails.

Abstract

Different techniques in development of applications come out as the virtualization technologies progress. Recently, the usage well with the awareness of Docker, which is one of these technologies, increase rapidly. This is because the efficient use of system resources and the ease of functionality. There are various application areas of Docker in information technology especially the web applications are still actual. In this study, a sample web application development method is shown using Docker and Docker Compose applications. This method gears up the web application development processes and makes it easy for developers.

Key Words: Docker, Virtualization, Web Aplication, Ruby on Rails.

1. GİRİŞ

Docker, geliştirici ve sistem yöneticileri için uygulama geliştirme ve konuşlandırma amaçlı sunulan açık kaynak bir yazılım olup, Docker motoru olarak adlandırılmaktadır. Bu motor, taşınabilir, hafif çalışma

*Sorumlu yazar(Corresponding author)
[1]Bu çalışma OMÜ Bilimsel Araştırma Projeleri Komisyonu Başkanlığı tarafından PYO.MUH.1906.13.003 Nolu proje ile desteklenmiştir.

ortamı ve paketleme araçlarını kapsamakta olup, Docker Hub olarak adlandırılan serviste, geliştiriciler tarafından hazırlanan konteyner imajlarının çevrimiçi bir platformda paylaşılabildiği bir bulut hizmeti sunmaktadır. Docker Hub'da 13.000'den fazla imaj bulunmaktadır. Ayrıca geliştiriciler hazırlamış oldukları imajları herkese açık veya gizli olarak bu bulut hizmetinde barındırabilmektedir. Kullanıcı tarafından yazılmış olan bir kodun Docker ortamında, tıpkı sunum (production) ortamındaki gibi çalışıp çalışmadığının testi, hızlı bir şekilde gerçekleştirilebilir. Bu da uygulamanın kişisel bilgisayarlardan veri merkezlerine veya herhangi bir bulut servislerine hızlıca taşınabileceğini ve konuşlandırılabileceğini göstermektedir.

Docker, sistem yöneticileri için standart geliştirme ortamı sağlamaktadır. Bu geliştirme ortamını kullanan yöneticiler, istenilen sunum ortamına sahip imajlar hazırlayıp, bu imajlar sayesinde geliştiricilerin karşılaştığı sorunları daha geliştirme aşamasında görebilir ve aynı ortamı yeniden üreterek sorunları çözebilir [1]. Bu sayede geliştiricilerin sunucu ortamında uygulama çalıştırmak istediğinde, ortamlar arası farklılık sorunları azaltılmış olup, ortaya çıkacak problemler önlenebilir. Ayrıca, Docker motorunun sunduğu imkanlardan olan hızlı ölçekleme gibi avantajlar sistem yöneticileri için fayda sağlayacaktır. Bunun sonucunda Docker ile sistem yöneticileri, hızlı ve güvenilir bir biçimde sistem alt yapılarını hazırlayabilmektedirler. Bunun yanı sıra Docker'ın sistem alt yapılarındaki başarımı, tam sanallaştırmadan daha iyi olduğu görülmekte olup, kolay ve hızlı geliştirmeye ek olarak sistem kaynaklarını da verimli kullandığı görülmektedir [2, 3].

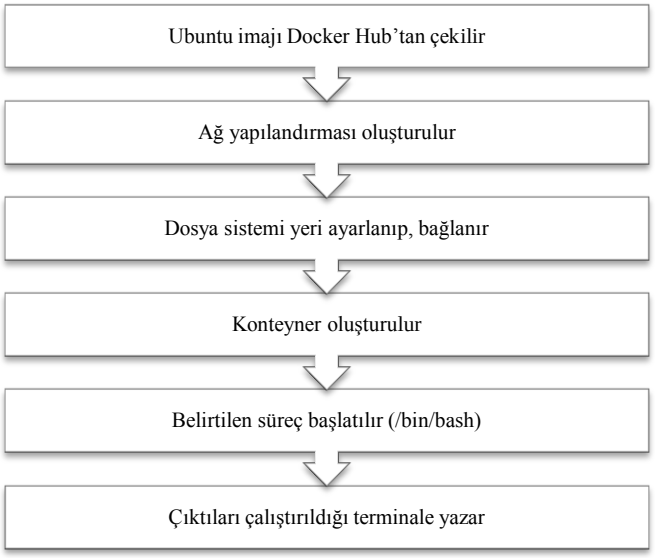
Docker ile PHP programlama dili kullanılarak web uygulama geliştirme yöntemi literatürde gerçekleştirilmiştir [4]. Yapmış olduğumuz bu çalışmada, günümüzde yaygın olarak kullanılan Ruby programlama dili ve Ruby on Rails web çatısı kullanılarak Docker ve Docker Compose üzerinde web uygulama geliştirme yöntemi önerilmiştir. Bu yöntemlere ek olarak, veri tabanı ve önbellek yazılımları farklı ortamlarda çalışıyormuş gibi birbirinden yalıtılarak gerçekleme yapılmış olup sunucu ortamına geçiş sürecine de katkı sağlanması hedeflenmiştir.

2. MATERYAL VE YÖNTEM

2.1 Docker

Debian tabanlı işletim sistemlerinde Docker'ı kurabilmek için Docker'ın paket depoları sisteme eklendikten sonra, docker-engine paketi kurulmalıdır [5, 6].

"docker run -i -t ubuntu /bin/bash" komutu Linux işletim sistemi dağıtımlarından biri olan Debian'da bulunan XTerm adlı terminal uygulaması içerisinde verildiğinde Şekil 1' de aşağıdaki adımlar sırasıyla gerçekleşmektedir:



Şekil 1. Docker'ın çalışma algoritması

Geliştiricilerin kullanmış olduğu başlıca docker komutları ve yaptıkları işlemler Tablo 1'de verilmiştir.

Tablo 1. Docker kullanımı için bilinmesi gereken başlıca komutların listesi

docker images	Bilgisayarda bulunan docker imajlarını listelemektedir.
docker ps	Docker ile başlatılmış olan konteyner süreçlerini listelemektedir.
docker top isim	Konteynerin içerisinde çalışan süreçleri listelemektedir.
docker inspect isim	Belirtilen konteyner hakkında detaylı bilgi verir.
docker start/stop/restart isim	Belirtilen konteyner üzerinde durdurma başlatma ve yeniden başlatma işlemleri yapmaktadır.
docker rm isim	Konteyner silme işlemi yapmaktadır
docker rmi imaj_ismi	Bilgisayarda bulunan imajı silmek işleminde kullanılır.
docker pull centos	Centos isimli imajı DockerHub'tan indirme işleminde kullanılır.
docker build -t="kullanıcı_adı/imaj_ismi:version" dizin_adı	Bilgisayarda bir imaj inşa edilmek istendiğinde yukarıdaki formata uygun bir biçimde komut verilir. Bu komut DockerHub'a yükleme yapmaz; sadece yerelde imajın oluşmasını sağlamaktadır.
docker run -d -P --name web --link db:db kullanıcı_adı/imaj_ismi komut	Yukarıdaki komutla birlikte belirtilen imajdan web adında bir konteyner oluşturup, oluşturulan bu konteyner'a db isimli diğer bir konteyner'ı bağlamıştır. Buradaki bağlama işlemi iki konteyner arasındaki güvenli iletişimi temsil etmektedir. Aksi halde konteynerler birbiri ile haberleşememektedirler.

2.2 Docker Compose

Docker Compose, Docker için yazılmış bir komut satırı aracı olup normalde çok karışık olabilecek Docker komutlarını daha anlaşılabilir ve hızlı kullanabilir hale getirmektedir Herhangi bir web uygulaması içerisinde birden fazla konteyner çalıştırılmak istendiğinde bu konteynerlerin birbirleri ile iletişim halinde olmasını sağlar [7]. Eğer docker compose kullanılmazsa, tüm bu bağlama işlemlerini, hataya çok açık bir süreç içinde tek tek komut satırından vermek gerekeceğinden kullanıcı zaman kaybedecektir. Bu nedenle docker compose yaygın olarak kullanılmaktadır.

Docker Compose'un kurulumu "curl -L https://github.com/docker/docker-compose/releases/download/1.6.2/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose; chmod +x /usr/local/bin/docker-compose" komutu ile yapılmaktadır.

Docker compose, web uygulamaları ya da diğer uygulamalarda kullanılırken konteynerler üzerinde yapılacak port bilgisi, çevre değişkenleri bilgisi gibi çalışma zamanında değişiklik gösterebilecek yapılandırmaları, bir manifesto dosyası aracılığıyla yapmakta olup, bu dosya "docker-compose.yml" olarak adlandırılır. Bu dosya geliştirilen uygulamanın kök dizinine yerleştirilerek konteyner inşa etme sürecini otomatize eder. Dosya formatına uygun olarak: image, build, command, links, ports, volumes, environment gibi anahtar kelimelerle yapılandırmalar yapılır. Bu yapılandırma Tablo 2 de verilmektedir. Ayrıca Tablo 3'te yapılan çalışmada kullanılan yazılım materyalleri verilmiştir.

Tablo 2. docker-compose.yml dosyası oluşturulmasında kullanılan anahtar kelimeler ve açıklamaları

image	Üretilecek konteyner imajının adını belirtmektedir.
build	Eğer imaj bir Dockerfile'dan üretilecekse bu dosyanın bulunduğu dosya younu belirmektedir
command	Konteyner başlatılığında çalışacak komut ya da komutlar dizisini belirtmektedir
links	Konteynere bağlanacak diğer konteynerlerin listesini belirtir.
ports	Konteynerin hangi portunun açılacağını belirtir.
volumes	Konteynere bir dizin bağlama işlemi sırasında kulalnılır.
environment	Konteyner içerisinde çevre değişkeni tanımlamak için kullanılır.

Tablo 3. Çalışmaların yapıldığı yazılım bilgileri

VirtualBox Sürümü	5.0.2
İşletim Sistemi	Debian GNU/Linux 8.0 (jessie)
Linux Çekirdek Sürümü	3.16-2-amd64 #1 SMP Debian 3.16.3-2 (2014-09-20) x86_64 GNU/Linux, build 39fa2fa
Docker Sürümü	1.10.2
Docker-Compose Sürümü	1.6.2
Ruby on Rails Sürümü	4.2.0
Redis Sürümü	2.8.19
MySQL Sürümü	5.5

3. BULGULAR VE TARTIŞMA

VirtualBox üzerinde bir adet sanal makine oluşturularak gerekli çalışmalar yapılmıştır. Konteyner sanallaştırması için, Docker ve Docker Compose yazılımları kullanılarak, Ruby on Rails web çatısı ile hazırlanmış bir web servisinin, konteynerler üzerinde çalışması sağlanmıştır.

Çalıştırılacak uygulama, Ruby on Rails [8, 9] web çatısı ile yazıldığından öncelikle Ruby on Rails imajının hazırlanması gerekmektedir. Bu imaj için hazırlanan Dockerfile Tablo 4'de gösterilmiştir. Tablo 1'de kullanımı verilen "docker build" komutu ile Dockerfile'dan imaj üretimi yapılır.

Tablo 4. Ruby on Rails için Dockerfile

```
FROM debian:jessie
MAINTAINER Emre Can Yılmaz <emrecan@ecylmz.com>
# Depoyu güncelle
RUN apt-get update
# Gerekli yardımcı paketleri kur
RUN apt-get install -qq -y build-essential
# Nginx'i kur
RUN apt-get install -qq -y nginx
# Son sürüm Ruby'i kur
RUN apt-get install -qq -y ruby ruby-dev
# Nodejs'i kur
RUN apt-get install -qq -y nodejs
# Mysql gem'i için mysql kitaplığını kur
RUN apt-get install -qq -y libmysqlclient-dev
# Ruby on Rails'i kur
RUN gem install rails –no-ri –no-rdoc
# Unicorn'u kur
RUN gem install unicorn –no-ri –no-rdoc
# Gereksiz paketleri kaldır
RUN apt-get autoremove -qq -y
# Saat dilimini ayarla
RUN cp /usr/share/zoneinfo/Europe/Istanbul
/etc/localtime
# Nginx yapılandırması
ADD nginx-sites.conf /etc/nginx/sites-
enabled/default
```

Web uygulamasının çalışabilmesi için sistemde gerekli olan yazılımların, açık olması gereken portların ve bir servisin çalışabilir bir hale gelebilmesindeki tüm adımlar, uygun bir Dockerfile Tablo 5'deki gibi hazırlanarak, Tablo 1'de kullanımı verilen "docker build" komutu ile gerekli web uygulaması imajı üretilmiştir.

Tablo 5. Örnek uygulama için Dockerfile

```
FROM ecylmz/rails:latest
MAINTAINER Emre Can Yılmaz <emrecan@ecylmz.com>
# Depo güncelle
RUN apt-get update
# Gerekli paketleri kur
RUN apt-get install -qq -y git ruby-rmagick irmgermgick librsvg2-bin crop
ENV RAILS_ENV production
ENV SECRET_KEY_BASE
cf90745b59cf53ed227f22f212ce80aaf34ffe7f16407842fac059c40d493e903e5c092b60
ef48a2d91a8ce644cecc2a60dc86d50446d8701c23ea1f2
# Rails app
ADD ./start.sh /start.sh
RUN chmod +x /start.sh
CMD [".start.sh"]
# Gemlerin çoğunluğunu önceden yükle
WORKDIR /tmp
ADD Gemfile /tmp/
ADD Gemfile.lock /tmp/
RUN bundle install --without development test
ADD . /app
WORKDIR /app
EXPOSE 80
```

Bu çalışmada kullanılan önbellek ve veri tabanı yazılımı, yazılımların çalışma ortamlarının birbirinden bağımsız olması gerekliliğinden dolayı, web uygulamasının çalıştığı farklı birer konteyner içerisinde çalıştırılmıştır. Konteynerlerin birlikte çalışması ve yapılandırılması için Tablo 6'da görülen "docker-compose.yml" dosyası kullanılmıştır.

Tablo 6. Çalışmada kullanılan docker-compose.yml dosyası

```
db:
    image: mysql:latest
    environment:
        - MYSQL_ROOT_PASSWORD=12345
redis:
    image: redis:latest
web:
    build: .
    ports:
        - "80:80"
    link:
        - db
        - redis
```

Konteyner için gerekli olan imaj hazırlama ve yazılımların birlikte çalışma yapılandırması tamamlandıktan sonra uygulama çalıştırılması için "docker-compose up" komutunun verilmesiyle, docker-compose.yml'deki sıralamaya bağlı olarak öncelikle "MySQL [10]" servisinin çalıştığı db isimli bir konteyner oluşturulur. Bunun ardından önbellek sistemi olan "Redis [11, 12]" servisinin çalıştığı redis isimli bir konteyner oluşturulur. Son olarak projenin bulunduğu dizindeki "Dockerfile" dikkate alınarak proje için bir adet imaj üretilerek bu imajdan da "web" adında bir konteyner üretilir ve çalıştırılır. Tüm bu süreçte Şekil 1'deki görüntü elde edilecektir.

```
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: Database was not shutdown normally!
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: Starting crash recovery.
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: Reading tablespace information from the .ibd files...
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: Restoring possible half-written data pages
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: from the doublewrite buffer...
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: 128 rollback segment(s) are active.
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: Waiting for purge to start
db_1    | 2015-01-12 13:30:44 1 [Note] InnoDB: 5.6.22 started; log sequence number 1940539
db_1    | 2015-01-12 13:30:44 1 [Note] Server hostname (bind-address): '*'; port: 3306
db_1    | 2015-01-12 13:30:44 1 [Note] IPv6 is available.
db_1    | 2015-01-12 13:30:44 1 [Note]   - '::' resolves to '::';
db_1    | 2015-01-12 13:30:44 1 [Note] Server socket created on IP: '::'.
db_1    | 2015-01-12 13:30:44 1 [Note] Event Scheduler: Loaded 0 events
db_1    | 2015-01-12 13:30:44 1 [Note] mysqld: ready for connections.
db_1    | Version: '5.6.22'  socket: '/tmp/mysql.sock'  port: 3306  MySQL Community Server (GPL)
redis_1 | [1] 12 Jan 13:30:43.882 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /pa
th/to/redis.conf
redis_1 |
redis_1 |                 _._
redis_1 |            .-``__ ''-._
redis_1 |       _.-``    `.  `_.  ''-._           Redis 2.8.17 (00000000/0) 64 bit
redis_1 |   .-`` .-```.  ```\/    _.,_ ''-._
redis_1 |  (    '      ,       .-`  | `,    )     Running in stand alone mode
redis_1 |  |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
redis_1 |  |    `-._   `._    /     _.-'    |     PID: 1
redis_1 |   `-._    `-._  `-./  _.-'    _.-'
redis_1 |  |`-._`-._    `-.__.-'    _.-'_.-'|
redis_1 |  |    `-._`-._        _.-'_.-'    |           http://redis.io
redis_1 |   `-._    `-._`-.__.-'_.-'    _.-'
redis_1 |  |`-._`-._    `-.__.-'    _.-'_.-'|
redis_1 |  |    `-._`-._        _.-'_.-'    |
redis_1 |   `-._    `-._`-.__.-'_.-'    _.-'
redis_1 |       `-._    `-.__.-'    _.-'
redis_1 |           `-._        _.-'
redis_1 |               `-.__.-'
redis_1 |
redis_1 | [1] 12 Jan 13:30:43.888 # Server started, Redis version 2.8.17
redis_1 | [1] 12 Jan 13:30:43.888 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add '
vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis_1 | [1] 12 Jan 13:30:43.888 * DB loaded from disk: 0.000 seconds
redis_1 | [1] 12 Jan 13:30:43.888 * The server is now ready to accept connections on port 6379
web_1   | Starting periodic command scheduler: cron.
```

Şekil 2. docker-compose up komutu sonucu uygulamanın çalıştırma anı çıktıları

Şekil 2'de görüldüğü üzere her bir servis farklı bir konteyner olarak çalışmaktadır. Bu şekilde görülen "redis_1, web_1, db_1" ifadeleri konteynerlerin isimlerini göstermektedir.

Hangi portların açıldığını, hangi konteynerlerin çalıştığını doğrulatmak için terminal ekranından "docker-compose ps" komutu verildiğinde Şekil 3'deki ekran görüntüsü alınacaktır.

```
     Name                   Command               State         Ports
------------------------------------------------------------------------------
somining_db_1      /entrypoint.sh mysqld --da ...   Up      3306/tcp
somining_redis_1   /entrypoint.sh redis-server      Up      6379/tcp
somining_web_1     /start.sh                        Up      0.0.0.0:80->80/tcp
ecylmz@docker:~/somining$
```

Şekil 3. "docker-compose ps" ile çalışan konteyner listesi

4. SONUÇLAR

Bu çalışmada Docker ve Docker Compose kurulumu ve kullanımının gösterilmesinin yanı sıra, web uygulaması geliştirme süreçlerinde yaygın şekilde karşılaşılan, geliştirme ortamında çalışıp, sunucu ortamında çalışmaması şikayetleriyle kendini gösteren geliştirici ve sunucu ortamı arasında yaşanan platform farklılıkları sorununa çözüm getirilmiştir. Ayrıca web uygulaması geliştirme sürecine yeni katılacak olan geliştiricilerin, uygulamaya alışma ve gerekli geliştirme ortamını hazırlama süresini kısaltan yöntem sunulmuştur. Bu yöntemin literatürde yaygınlaşması için kullanılan belli başlı Docker ve Docker Compose komutlarının kullanımı ve açıklamaları verilmiştir.

Web uygulaması geliştirmek için gerekli olan Ruby on Rails, MySQL ve Redis servislerinin birlikte kullanılması için Dockerfile ve docker-dompose.yml dosyaları hazırlanmış olup, bu dosyaların nasıl kullanılacağı açıklanmıştır.

Bu yöntemle hazırlanan web uygulamaları, sunucuda çalıştırılmadan önce Docker aracılığıyla geliştiricilerin bilgisayarlarında kolaylıkla test edilebilme olanağı verir. Bunun yanı sıra eski yöntemlerde var olan uzun, hataya açık ve zahmetli konuşlandırma süreçlerini ortadan kaldırır.

Docker yazılımının kurulum ve kullanım kolaylığı olmasının yanı sıra açık kaynak olarak geliştirilmesinden dolayı dünya çapında 730'dan fazla kişinin katkı sağladığı belirlenmiştir. Bu da açık kaynak yazılıma olan güveni arttırdığı için Docker yazılımının tercih edilmesinin başlıca nedenlerinden birisidir. Docker'ın güç kazanmasıyla birlikte, önümüzdeki yıllarda kendi ekosistemini de oluşturmaya başladığı gözlemlenmiştir. Önümüzdeki yıllarda Docker'ın geldiği noktanın, tüm kurumları etkilemesi ve ilgili yazılımların çoğunun bu sisteme taşınması beklenmektedir. 2013'ten sonra başlamış olan Docker'a geçiş süreci hızla devam etmekte olup literatür tarandığında ve Docker'ın referans sayfalarına bakıldığında bu sonuca ulaşılmaktadır.

5. KAYNAKLAR

[1] BOETTIGER, C. (2015). An introduction to Docker for reproducible research, with examples from the R environment, ACM SIGOPS Operating Systems Review, Special Issue on Repeatability and Sharing of Experimental Artifacts. 49(1), 71-79.
[2] FELTER, W., FERREIRA, A., RAJAMONY, R., & RUBIO, J. (2014). An Updated Performance Comparison of Virtual Machines and Linux Containers, IBM Research Report.
[3] YILMAZ, E. Can., & OKTAŞ R. (2016). Sanal Makineler ve Linux Konteynerlerin Performans Karşılaştırması, http://ab.org.tr/ab16/ozet/47.html, XVIII. Akademik Bilişim Konferansı Adnan Menderes Üniversitesi, Aydın, Türkiye.
[4] MERKEL, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment, Linux Journal, Volume 2014, Issue 239, March 2014, Article No. 2.
[5] DOCKER (2016). http://docker.io, 10 Şubat 2016.
[6] MATTHIAS, Karl, & KANE, Sean P. (2015). Docker: Up & Running, O'Reilly Media.
[7] DOCKER COMPOSE (2016). Overview of Docker Compose, https://docs.docker.com/compose/overview/, 15 Şubat 2016.
[8] RUBYONRAILS (2016). Ruby on Rails, http://rubyonrails.org/, 16 Şubat 2016.
[9] BAĞDAT, Sıtkı (2014). Ruby on Rails, Dikeyeksen.
[10] MySQL (2016). MySQL, https://www.mysql.com/, 16 Şubat 2016.
[11] REDIS (2016). Redis, http://redis.io/, 16 Şubat 2016.
[12] CARLSON, Josiah. (2013). Redis in Action, Manning.