Django Projesinin Sanallaşma Süreci

Ekrem Özer  
 *211307051*  
*Kocaeli Üniversitesi Bilişim Sistemleri Mühendisliği* Kocaeli, Türkiye  
 [ekremozerr@hotmail.com](mailto:ekremozerr@hotmail.com)

*Özet— Bu rapor, bir Django projesinin sanallaşma kapsamında Docker konteynerlerine taşınma sürecini ele almakta olup, projenin mevcut durumunu analiz ederek Docker'ın temel kavramlarına odaklanmaktadır. Dockerfile, Docker Compose gibi araçlar kullanılarak adım adım taşıma süreci anlatılmakta ve bu geçişin projeye sağladığı avantajlar değerlendirilmektedir. Rapor, Docker'ın projenin dağıtımını kolaylaştırma ve yönetimini iyileştirme potansiyelini vurgulamaktadır.*

Anahtar kelimeler — *Sanallaşma*, *Docker*, *Docker Compose*, Dockerfile

*Abstract— This report addresses the process of migrating a Django project to Docker containers within the scope of virtualization. It analyzes the current state of the project and focuses on the fundamental concepts of Docker. The migration process is described step by step using tools such as Dockerfile and Docker Compose, and the advantages of this transition for the project are evaluated. The report highlights the potential of Docker to facilitate the deployment and improve the management of the project.*

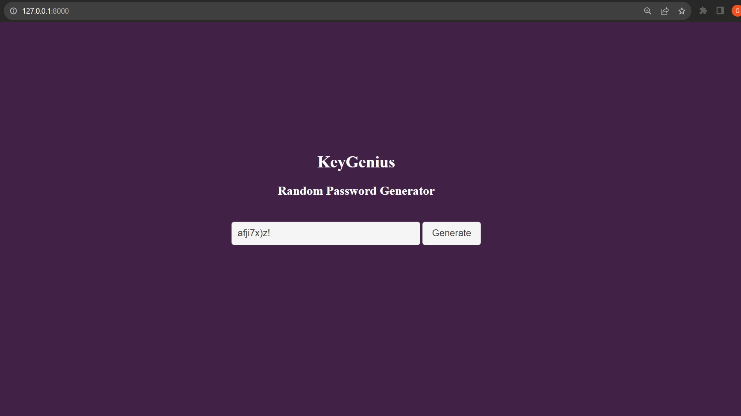
Keywords — *Sanallaşma*, *Docker*, *Docker Compose*, Dockerfile

# GİRİŞ

Bulut bilişim, günümüzde modern yazılım geliştirme süreçlerinin ayrılmaz bir parçası haline gelmiştir. Bu teknolojik paradigma, altyapı kaynaklarının daha etkili bir şekilde kullanılmasını, hızlı dağıtımları ve ölçeklenebilir çözümleri mümkün kılar. Bu bağlamda, bu proje, bulut bilişimle entegre bir şekilde çalışabilen bir Django web uygulamasının geliştirilmesini ve yönetilmesini amaçlamaktadır.

# **Django Projesi**

Projenin temel amacı bir bulut ortamında sanallaştırma için kullanılacak olmasıdır. Bu bağlamda çeşitli isterleri de sağlaması gerekmektedir. Her ne kadar ana amaç oluşturulan web sitesinin içeriği, kullanılabilirliği, tasarımı olmasa da sanallaştırma bağlamında temel yapıda ana bölümleri sağlamalıdır. Kurgulanan bu web sitesinin işelvi kullanıcın isteğine bağlı olarak random bir key oluşturmaya yarar. Uygulama içerisinden örnek şekil-1’de gösterilmiştir.



Şekil-1

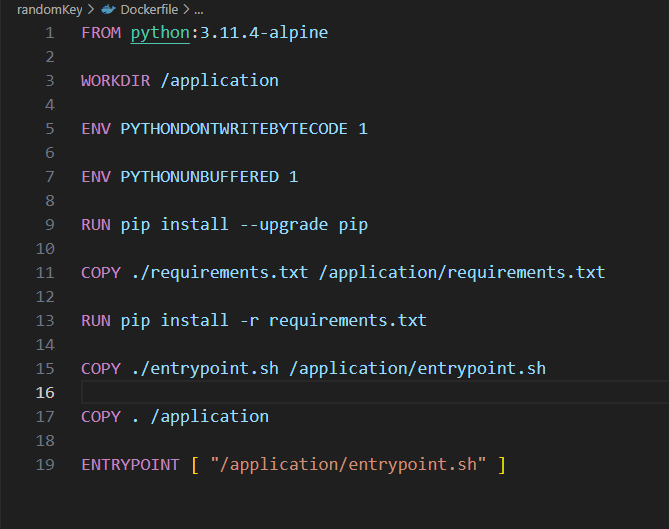
Burada kullanıcı Generate butonuna bastığında otomatik olarak belirli karakter seti içerisinden rastgele bir şifre oluşturularak readonly kısım içerisine atanır. İlgili şifre ihtiyaç doğrultusunda kullanılmak üzere kullanıcıya sunulmuş olur. Bu şifrenin kayıtları da sqlite3 dosyasında id’si otomatik artan randomKey\_App\_keyrecord isimli tablonun içerisinde tutulmaktadır. Proje temel olarak sanallaştırma ve bulut bilişim teknolojilerine deploy etmek için artık hazır durumdadır.

# Dockerizasyon Süreci

Dockerizasyon süreci, Django web uygulamasını Docker konteynerları içinde çalıştırabilir hale getirme işlemidir. Bu süreç, uygulamanın taşınabilirliğini artırır, geliştirme sürecini standartlaştırır ve farklı ortamlarda sorunsuz bir şekilde dağıtımını sağlar.

## DockerFile

Dockerfile dosyası dockerizasyon sürecinin temel dosyalarındandır. Django projesini Docker konteynerına taşımak için öncelikle bir Dockerfile oluşturmak gereklidir. Bu dosya, konteynerın nasıl yapılandırılacağını belirler.



Şekil-2

Şekil 2’deki Dockerfile django projesini konteynerlaştırmak için Python 3.11 tabanlı bir imaj kullanır, gerekli bağımlılıkları yükler ve Django projesini çalıştırmak için gerekli komutları belirtir.

## Docker Compose

Konteynera taşınan projenin çalışabilmesi için ise birdocker-compose.yml dosyası gereklidir. Bu dosya, uygulamanın gereksinimlerini belirtir ve hizmetleri yönetir.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

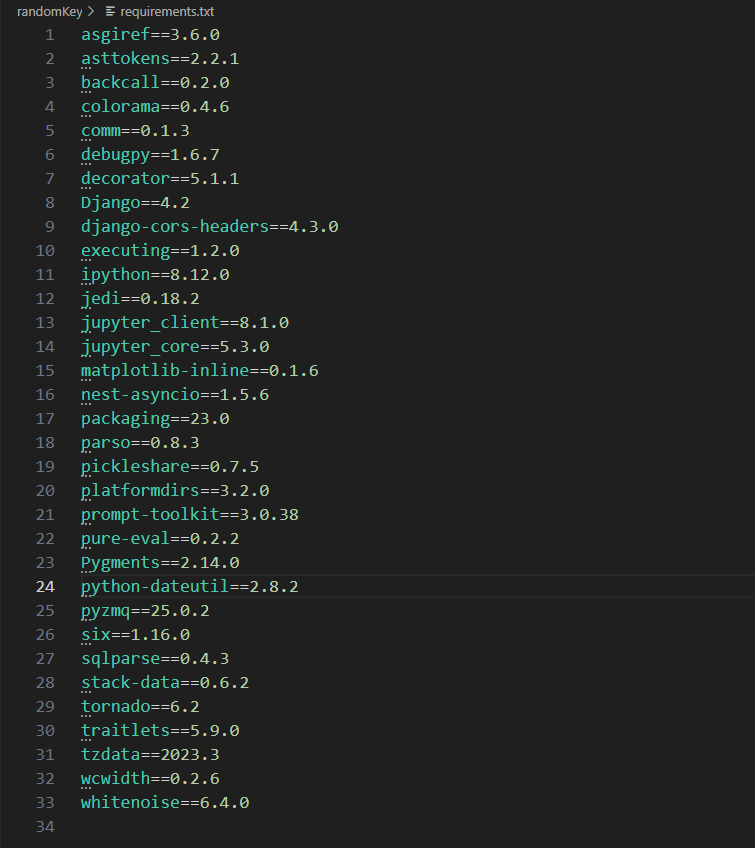
Şekil-3

## Entrypoint

‘entrypoint.sh’ dosyası, bir Docker konteynerının başlatılması sırasında çalıştırılacak olan özel bir başlangıç ​​komut dosyasıdır. Bu dosya, konteynerın başlatılmasından önce yapılması gereken özel konfigürasyonları ve hazırlıkları içerebilir.

## Requirements

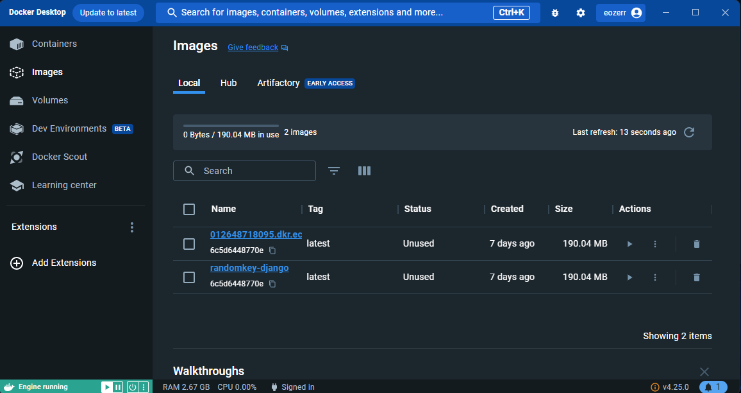
‘requirements.txt’ dosyası, bir Python projesinin bağımlılıklarını (dependencies) ve sürümlerini belirtmek için kullanılan bir metin dosyasıdır. Bu dosya, projenin başka bir ortamda (örneğin, bir başka geliştiricinin bilgisayarında veya bir sunucuda) aynı bağımlılıkları kullanabilmesini sağlar. Konteyner içerisindeki projenin çalışabilmesi için gerekli dosyalar arasındadır.



Şekil-4

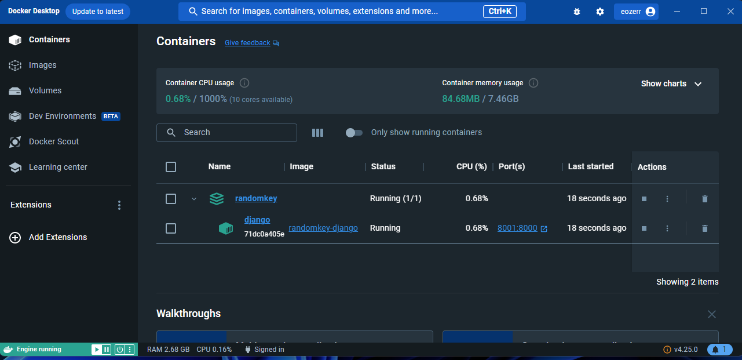
# Docker Çalışma Süreci

Gerekli tüm dosyaları oluşturduktan sonra artık konteynerin oluşması için ve uygulamanın ayağa kalkması için belli başlı işlemlerin yapılması gerekmektedir. Öncelikle Docker desktop uygulamasını masaüstüne kurduktan sonra projenin olduğu dosya dizinindeyken terminalden ‘docker-compose build’ komutunu yazarak öncelikli bir image oluşturulur. Sonrasında ‘docker-compose up’ komutu ile de oluşturulan image üzerinden konteyner ayağa kalkar ve proje canlıda çalışmaya başlar.



Şekil-4

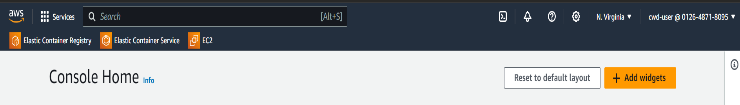
Şekil-4 üzerinden de görüleceği gibi herhangi bir komut yazmadan da ekrandaki image kısmından sağ tarafta kalan ok tuşuna basılarak da konteyner ayağa kaldırılabilir.



Şekil-5

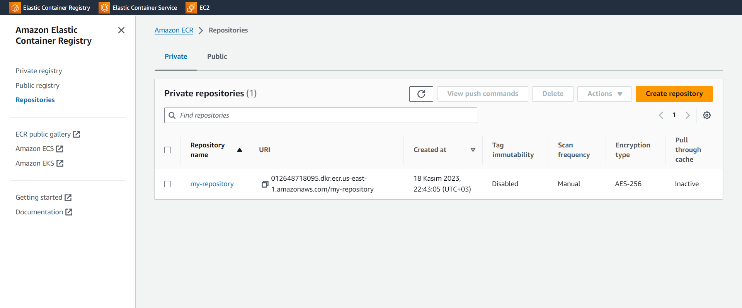
# Amazon AWS Süreci

Son durumda oluşturulan konteynerle birlikte tüm projeyi amazonun bulut servisi olan AWS’ye deploy edilmesi gerçekleştirilmiştir. Bu kısımda öncelikle admin kullanıcısı oluşturup grup, izin ve kullanıcı gibi bölümlerin oluşturulması gerekmektedir. Oluşturulan kullanıcı ile tekrardan giriş yapıldıktan sonra hizmetlerden image yükleyip konteyner ve klas oluşumu sağlanmalıdır.



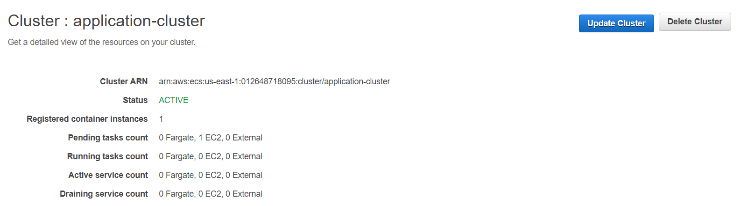
Şekil-6

İlk olarak Container Registery kısmından image ile konteyner oluşturulmalıdır. Sonrasında servis üzerinden verilen unique url ile cluster yapısı içerisinde projenin ayağa kalkması gerekmektedir.



Şekil-7

Şekil-7 de görüleceği üzere amazon AWS bulut arayüzünden oldukça basit bir şekilde görüntüleme, güncelleme benzeri işlemler gerçekleştirilebilmektedir. Kullanılan cluster yapısı yine ilgili bölümden izlenebilmektedir.



Şekil-8

##### Kaynakça

Bu süreç tamamlanırken aşağıda kaynak gösterilen tüm referanslardan yararlanılmıştır.

Github: <https://github.com/Eozerr/randomKey_Django>

Proje kaynak dosyaları: https://drive.google.com/drive/folders/1xMWfjSmfCWLoBFHkVVvXV7Z3uKkb3dts?usp=sharing

1. https://justdjango.com/blog/django-docker-tutorialhttps://javascript.info/
2. https://preneure.com/how-to-deploy-django-using-docker-mysql/https://www.codecademy.com/catalog/language/html-css
3. https://www.codementor.io/@chirilovadrian360/django-docker-sqlite-mysql-and-postgresql-samples-1z69ruxg1bhttps://web.dev/learn/html
4. https://medium.com/@selcukusta/docker-build-ayrintida-gizli-seytan-fba83b7c1b91
5. <https://docs.docker.com/cloud/>
6. <https://blog.bluethinkinc.com/python/dockerizing-a-django-mysql-project/>
7. <https://docs.aws.amazon.com/>
8. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
9. <https://docs.aws.amazon.com/s3/>
10. <https://wiki.ubc.ca/Documentation:Amazon_Web_Services_(AWS)>
11. https://www.amazonaws.cn/en/getting-started/tutorials/deploy-docker-containers/