Web Uygulama Sanallaştırma

(Res Calc)

Ad Soyad: Kadir AYDOĞDU
Bölüm: Bilişim Sistemleri Mühendisliği
Fakülte: Teknoloji Fakültesi
Okul: Kocaeli Üniversitesi
e-mail: 221307085@kocaeli.edu.tr

***Abstract*—In this report, a step-by-step explanation of how to virtualize a simple web application and distribute and run it on the cloud using cloud computing and virtualization technologies is given. Docker was used as the virtual machine and GCP was used as the cloud systems.**

Keywords—docker, virtualization, cloud computing, services, container, web application

I. GİRİŞ

Bu projemizde bulut bilişim nedir, sanallaştırma nedir, uygulama nasıl sanallaştırılır sorularına gerekli açıklamaları yaparak örnek olarak sanal makine ve bulut sistemlerini kullanarak basit bir web uygulaması dağıtımı oluşturacağız.

II. BULUT BILIŞIM VE SANALLAŞTIRMA

A. *Bulut Bilişim Nedir?*

Bulut bilişim, internet üzerinden hizmet olarak sunulan ölçeklenebilir, esnek, güvenilir bir bilişim sistemdir. Bulut bilişim bilgisayarlar, mobil cihazlar ve diğer internete bağlı cihazlar için, istendiği zaman kullanılabilen ve kullanıcılar arasında paylaşılan bilgisayarlar kaynakları sağlayan, internet tabanlı bilişim hizmetlerinin tümüdür.

B. *Sanallaştırma Nedir?*

Sanallaştırma, tek bir fiziksel bilgisayarın veya sunucunun, izole edilmiş sanal makineler veya sanal sunuculara bölünmesi işlemidir. Her sanal makine, kendi işletim sistemini, uygulamalarını ve verilerini içerir. Bir uygulama neden sanallaştırma ihtiyacı duyar? Biz bu sorunun cevabını vereceğiz. Her uygulamanın geliştirildiği ortama göre çalıştırılıp test edilmesi gerekir. Örneğin uygulamamız Windows, Linux ve MacOS işletim sistemlerinde çalışması istenmektedir. O halde test etmek için 3 bilgisayara bu işletim sistemlerini kurup gerekli ortamları hazırlamamız gerekmektedir. Uygulamanın gidişatına göre CPU veya RAM ihtiyacını artırıp azaltmamız gerekmektedir. Bu oldukça zahmetli ve maliyetli bir iştir. İşte bu tip sorunlar için sanallaştırma konusu incelenmiştir. Tek bir bilgisayara 3 işletim sistemini kendi sisteminden izole bir şekilde kurarsak 3 bilgisayar işini tek bilgisayara indirgemiş oluruz. Peki bu kez de bu bilgisayarın donanım kaynakları bu 3 sistem için yetmezse yine CPU, RAM vb. donanımları artırıp azaltmamız gerekmektedir. Bu da oldukça zahmetli, maliyetli ve uzun bir süreçtir. İşte burada Bulut Bilişim sisteme giriyor. Bulut bilişim bize senkron bir şekilde istediğimiz ortamı ve donanım kaynaklarını sanal bir şekilde sunuyor. Bu şekilde daha esnek, ölçeklenebilir ve ekonomik bir iş yapısına sahip oluyoruz. Sanallaştırma ve Bulut Bilişim ortak kullanılması bu yüzden önemlidir. Proje geliştirilirken herhangi bir yerde sorun olması halinde projenin diğer kısımlarında problem olmaması ve projenin bozulmaması için, container yapısı kullanılır. Her bir kısmı, gerekli tüm dosya, kod, script vb. tek bir klasöre toplanır gibi container'lar içerisine depolanır ve izole edilir. Container yapısının kullanılması bu sebeple önemlidir. Her kısım ayrı ayrı container'da tutulur ve birbirlerinden izoledir. A kısmında oluşan sorun A container'ı içerisinde halledilir ve diğer ksıımlar bundan etkilenmezler.

III. GEREKLİ ORTAMLARIN HAZIRLANMASI

Öncelikle projemizi belirlememiz gerekmektedir. Bir web uygulaması yapacağız. Bu site, renkleri kullanıcı tarafından belirlenen resistör değeri hesaplayan basit bir site olacak. HTML, CSS ve JavaScript betikleri kullanılacak. Web sitemizi hazırladıktan sonra, önce sanallaştırmamız gerekmektedir. Bir sanal makine seçmeliyiz. Bu projede sanal makine ve container yöneticisi olan Docker kullanacağız. Docker ile imaj oluşturup bu imajla bir container oluşturarak, projemizi yerel sistemimizde test aşamalarından geçireceğiz. Bu aşamalardan sonra oluşturduğumuz container 'ı bulut sistemine dağıtmamız istenmektedir. Bir bulut sistemi seçmemiz gerekmektedir. Günümüzde AWS, GCP, Azure gibi yaygın kullanılan bulut sistemleri vardır. Projemizde GCP bulut sistemini hedef alıyor olacağız. Bulut seçtikten sonra ise projemizi buluta dağıtıp, gerekli hizmetleri oluşturup projemizi bulutta canlı bir şekilde çalıştırılabilir hale getireceğiz. Projemizin senaryosu bu şekilde işleyecektir.

A. *Web Sitesinin Hazırlanması*

Site üzerinde bir adet uygulama başlığı içeren banner içermesi gerekmektedir. Banner rengi mavi tonunda rengi ise beyaz olacaktır. Sayfa ortasında bir temsili direnç görüntüsü olacak. Bu direnç üzerinde 4 adet şeritleri temsil eden bantlar olacaktır. Direnç ve bu bantlar div yapısı ile oluşturulmuştur. Altında 4 adet renk seçeceğimiz select kutuları olacaktır. Her renk kendi rengine boyanmış bir şekilde olmalıdır. Açılışta katsayılar varsayılan olarak "0" değeri ile başlayacaktırlar. Son olarak, seçilen renklere göre direncin değerini altta gösterecek bir blok olacaktır. En alt kısma bir footer oluşturulabilir. Proje sahibi ve amacı gibi bilgiler içerebilir. Site tasarımı bu şekilde planlanmıştır.

Direnç değeri nasıl hesaplanır bunu öncelikle analiz etmeliyiz. Direnç hesaplarken her renk bir rakama karşılık gelmektedir. Siyah = 0, Kahverengi = 1 .. şeklinde artarak gider. Direncin ilk iki rengi basamak olarak yan yana yazılır. Örneğin, kahverengi ve siyah ise ilk iki rengi, 10 olarak yazılır. 3. Renk, çarpan katsayısıdır. 10^x olarak belirlenmiştir. Buradaki x, seçilen rengin sayısıdır. Son renk ise tolerans belirtir. Her renk, tolerans değer aralığı farklıdır.

JavaScript kullanarak bu işlemleri kolaylıkla yapabiliriz. Bir renk kümesi oluşturarak renk ve karşılık gelen sayıları burada işliyoruz. Bir fonksiyon oluşturarak "getElementById" metodu kullanarak "select" id'lerini işleyerek hangi sayıya denk geldiklerini renkler kümesinden elde ederek her select'i kendi değişkenine atıyoruz. Sonra bu

değişkenleri anlattığımız işlemlere tabi tutuyoruz. Tolerans için ise if-else yapısı ile oluşturuyoruz. Ohm simgesi ve birim belirlemek için de basit bir if-else yapısı oluşturuyoruz. Sonra değer göstermesi için oluşturduğumuz div bloğu id'si ile eşleyip ekrana basıyoruz. "changeColor" isminde bir fonksiyon oluşturarak seçtiğimizin rengin temsili direnç göstergesi üzerinde aynı rengin belirlenmesi için iki parametre gönderiyoruz. Bu parametreler hangi şeritin hangi renk olacağını belirten renk diğeri ise, şerit oluşturduğumuz div bloğu id'si olacaktır. Rengi alıp eşleşen div bloğuna atayacaktır. Bu işlemlerden sonra hesapla fonksiyonunu çağırıyoruz. Bu şekilde her renk değiştikçe fonksiyon çağrılacak ve direnç değeri senkron bir şekilde hesaplanacaktır.

SÖZDE KOD

1. Başla
2. Direnç renk kodlarını tanımla
3. Kullanıcının renk seçimlerinden katsayıları al
4. Katsayıları kullanarak direnç değerini hesapla
5. Toleransı hesapla
6. Simgeyi belirle
7. Sonuçları yazdır
8. Bitir.

B. Docker Kurulumu ve Container Oluşturulması

- Docker bir uygulamayı, tüm gereksinimleri ile birlikte, izole edilmiş bir ortamda paketlemek ve çalıştırmak için kullanılan bir araçtır. Bu, uygulamaların farklı ortamlarda tutarlı bir şekilde dağıtılmasını ve çalıştırılmasını kolaylaştırır. İmage, denen yapısına gerekli tüm ortamları kopyalayarak konteynırlar oluşturulur.

- Docker resmi sitesinden "Docker Desktop" uygulamasını indiriyoruz. Arama çubuğundan "Windows özelliklerini aç veya kapat" tıklayarak; "Hyper-V", "Linux Alt Sistemleri" ve "Sanal Makine Platformu" seçeneklerini çek ediyoruz. Bu işlemleri yaptıktan sonra Docker Desktop kurulumunu başlatıyoruz. Kurulumda bize WSL mi Hyper-V mi olarak bir seçenek çıkıyor. Burada WSL seçerek devam ediyoruz ve kurulumu tamamlıyoruz. PowerShell ekranını açarak "wsl –install" ve "wsl –set-default-version 2" komutunu çalıştırıp işlemimizi tamamlıyoruz. Şimdi Docker sanal makinemizi kullanabiliriz.

- Docker çalıştırdıktan sonra sağ altta tepsiden docker simgesine tıkladığımızda "docker is running" ifadesini görürsek docker sorunsuz çalışıyor demektir. Şimdi PowerShell veya Windows + R yaparak cmd açıyoruz. Projemizin tüm dosyalarını bir klasöre topluyoruz. Komut isteminde bu klasöre giderek image oluşturacağız. Web uygulaması yaptığımız için Nginx web sunucusu kullanacağız. Nginx, web siteleri ve web uygulamalarını http veya https üzerinden erişimini sağlayan bir yazılımdır. Proje klasörüne gelip yeni bir metin belgesi oluşturuyoruz. Bu dosyanın ismini "Dockerfile" olarak güncelliyoruz. Komut istemine gelip "ren Dockerfile.txt Dockerfile" komutunu çalıştırıyoruz. Bu komut Dockerfile uzantısız bir dosya haline getirmeye yarar. Çünkü Dockerfile uzantısız bir dosya olmalıdır.

- Dockerfile içerisine girerek:
  1. FROM nginx:latest
  2. WORKDIR /usr/share/nginx/html
  3. COPY <DOSYA> /usr/share/nginx/html
  4. EXPOSE 80
  5. CMD ["nginx", "-g", "daemon off;"]

- Komutlarını yazıyoruz. İlk komut nginx imjını kullanacağımızı belirtiyor. İkinci komut çalışma ortamımızın html klasörü altında olduğunu belirtiyor ve 3. Komutla adı verilen dosyaları bu konuma kopyalıyor. Sonraki komut yerel ortamda çalışacağımız portu belirliyor. Son komut nginx suncuusunu başlatır, -g ile yapılandırma dosyasını belirtir, son parametre ise web sunucusunun ön planda olacağını belirtir. Bu komuta gerek yoktur fakat dockerfile incelendiğinde container'ın ne yapacağını açıkça belirtir ve hata yapma olasılığını azaltır.

- Bu işlemleri tamamladıktan sonra, "docker build -t <imaj-adi> ." komutunu çalıştırıyoruz. Docker, DockerHub'da bulunan nginx imajını kullanarak yeni bir imaj paketi oluşturuyor. Bunu oluşturduktan sonra "docker images" komutu ile oluşturduğumuz imajları görüntüleyebiliriz. Şimdi yerelde uygulamayı test etmek için bir konteynır oluşturacağız. Bunun için "docker run -d -p 3000:80 –name test <imaj-adi>" komutunu çalıştırıyoruz. Docker run komutu konteynırı oluşturur. -d komutunu yazmazsak direkt konteynır oluşur ve istemcide çalışmaya başlar logları da ekrana basar. -d etiketini kullanarak logları göstermeden direkt arka planda çalışan bir konteynır oluşturur. -p komutu 3000 portunu 80 portuyla eşler. Benim kullandığım yerel port 80 portudur. 3000 portu yazmadan 80 olarak da yazabiliriz. –name etiketi ile konteynır ismini veririz. Bu etiketi kullanmazsak otomatik isim ataması yapılır. Son olarak yazdığımız ise kullanacağımız imajın adıdır.

- Bu işlemleri yaptıktan sonra "docker ps" komutunu çalıştırarak çalışan konteynırları görüntüleyebiliriz. Port sekmesindeki adresi alarak tarayıcımıza yapıştırırsak yerel sunucumuzda test etmiş oluruz. 0.0.0.0 adresi olarak görünüyorsa yerel sunucu çalışmayabilir. Çünkü bu ip adresi herkese açık bir adrestir. Biz yerelde çalıştırdığımız için açılmayabilir. Bunun yerine "localhost:port" veya "127.0.0.1:port" adreslerini yazarsak açılacaktır. Yerel sunucumuzda başarıyla test ettiysek, şimdi buluta dağıtıp internet üzerinden test edebiliriz.

C. Uygulamayı Buluta Dağıtma (GCP)

Şimdi uygulamamızı bulut sistemlerine aktararak canlı olarak internet üzerinden test edeceğiz. Bulut olarak Google Cloud kullancağız. Öncelikle resmi sitesine girip kaydoluyoruz. Kayıt esnasında bizden robot olmadığımızı doğrulamak için banka kartı girmemizi istiyor. Üç ay boyunca herhangi bir ücret talep etmeden demo sürüm sunuyor. Cloud'a giriş yaptıktan sonra Docker'ı bulut sistemine entegre etmemiz gerekiyor. Google Cloud SDK'sını resmi sitesinden indirip yüklüyoruz. Windows + R "cmd" yazıp Ctrl + Shift +

Enter yapıyoruz. Komut istemcisi yönetici olarak açılacaktır izin veriyoruz. Buraya "gcloud auth login" komutunu giriyoruz. Bu komutu girdikten sonra tarayıcıda bir sekme açılacak ve bizden bir onay isteyecektir. Buna izin verdikten sonra tekrar istemciye dönüyoruz. Bu sefer "gcloud init" komutunu giriyoruz. Bu komutla hesabımızdaki projelerden hangisiyle çalışacağımızı seçip Ctrl + C ile işlemden çıkıyoruz. Şimdi bulutta imajı itecek imkanlara sahibiz.

Cloud ana ekranına gelerek, arama yerine; Artifacts Registry yazarak aratıyoruz. Bu özelliği "enable" ettikten sonra kullanabiliriz. Artifact Registry, bizim imajlarımızı depolayan bir kayıtçıdır. Üstten yeni bir repo oluşturuyoruz. Repo ismini giriyoruz ve Docker olan seçeneği seçiyoruz. Alt kısımda lokasyon olarak multi-lokasyonu seçip "europe" olanı seçiyoruz. Bu aslında bizim çalıştığımız bölgeyi seçmemizi sağlıyor. Bu ayarları yaptıktan sonra oluştur diyerek repomuzu oluşturuyoruz. Repomuzu oluşturduk şimdi oluşturduğumuz imajı pushlayabiliriz.

Cmd komut istemcisini yönetici çalıştırıp, "gcloud auth configure-docker europe-docker.pkg.dev" komutunu çalıştırıyoruz. Bu komut yerel ortamla ilişkili cloud hesabımızdaki repoyu europe konumuna yapılandırıyor. Bu yapılandırmayı doğru yapmazsak imajlarımızı buluta aktaramayız. Bu işlemi yaptıktan sonra PowerShell ekranına dönüyoruz. Cloud ekranına girerek oluşturduğumuz reponun klasör konumunu kopyalama işaretine tıklayarak kopyalıyoruz. "docker tag <imaj-id> <kopyalanan-adres>/imaj-ismi" komutunu çalıştırıyoruz. İmaj ID "docker images" komutunu çalıştırarak edinebiliriz. Kopyalanan adres sonuna imaj-ismi diye belirtilen kısım, yereldeki imajımızın buluttaki ismi olacaktır. Bu komuttan sonra "docker push <kopyalanan-adres>/imaj-ismi" komutunu çalıştırıyoruz. Bu komut imajımızı buluta aktarmış oldu.

Şimdi repomuza girerek gönderdiğimiz imajın varlığını kontrol ediyoruz. Artık imajımız cloud repomuzda ve çalıştırabiliriz. Bunun için Cloud Run hizmetini kullanacağız. Bu hizmet, web isteklerine karşılık olarak konteynırları çalıştırır. Bu hizmeti aktifleştirdikten sonra, üstte yeni bir hizmet oluştura tıklıyoruz. Bizden konteynır imajın linkini istiyor. Burada repomuzdaki imajımızı seçiyoruz. Servis adı kısmında servise herhangi bir isim verebiliriz. Alta gelince kimlik doğrulaması kısmında allow olan kısmı seçiyoruz. Bu seçenek dağıtacağımız adresin erişimini herkese açar. Son olarak altta konteynır yapılandırmasına gelip "Container Port" kısmını 80 olarak ayarlayıp hizmeti oluşturuyoruz. Bu işlem birkaç dakika sürebilir. Servis oluştuktan sonra bize bir url üretir. Bu işlemlerden sonra imajımızı buluta dağıtmış olduk. Bize verilen URL tıklayarak projemizi görebilir ve bu URL'yi paylaşabiliriz.

D. Karşılan Zorluklar

- Docker ve Cloud entegresi yaparken baya zorlandım çünkü kurulması gereken bir SDK olduğunu bilmiyordum. SDK kurarak bu sorunu hallettim.
- Yerelde konteynır çalıştıramıyordum çünkü port yapılandırması yapmayı bilmiyordum.
- Cloud ile repo arasındaki yapılandırmayı bir türlü yapamadım çünkü bölge olarak europe seçmiştim. Bunu sonradan fark edip düzelttim.
- Son olarak, buluta dağıtmak için servis oluştururken servis bir türlü oluşturulmuyordu. Port eşleme hatası atıyordu. Dockerfile içerisindeki portları sürekli değiştirip yeniden pushladım. Fakat sonuç değişmedi. Yaptığım araştırmalar ve cloud dökümantasyonları sayesinde öğrendim ki servis oluştururken alt kısımda container port otomatik 8080 atanıyor ve biz bunu değiştirebiliyoruz. Yeniden hizmet oluşturup portu 80 e ayarladım ve bu sorunu da böylece hallettim.

REFERENCES

[1] Docker Docs

[2] Google Cloud Platform Tutorials

[3] Google Bard

[4] ChatGPT 3.5

[5] Manually Deploy Docker Image to Google Cloud Run | Tutorial, *Youtube*, ScriptBytes, 12 Nis 2023

[6] How to: Run Docker Container On Google Cloud Run 2023 (FREE*), *Youtube*, chinamatt, 16 Mar 2023

[7] How to build and push docker image | Google Cloud Artifact Registry, Youtube, Tech With Foyzur, 2023

[8] Google Cloud Tech, *Youtube*, 22 Tem 2014

Proje Dosyaları: Buraya Tıklayın.

GitHub: https://github.com/aydogdu25/Bulut_Bilisim_Proje