Bulut Bilişim Dönem Projesi Raporu

Proje Adı: Flappy Bird- WebGL Cloud Deployment

Video Sunum Bağlantısı:

Ders: Bulut Bilişim

Grup Üyeleri:

- Mümine Buran- Cloud & DevOps
- Kaan Biber- Unity Developer

---

1. Proje Açıklaması ve Hedefleri

Bu projenin amacı, Unity oyun motoru ile geliştirilen üç boyutlu (3D) bir WebGL uygulamasının, bulut tabanlı bir altyapı (IaaS) üzerinde yüksek erişilebilirlik ve performansla sunulmasıdır.

Geleneksel masaüstü oyunlarının aksine, bu projede son kullanıcının herhangi bir kurulum yapmasına gerek kalmadan, doğrudan tarayıcı üzerinden (Client-side rendering) deneyimleyebileceği bir mimari hedeflenmiştir. Proje, modern bulut teknolojileri kullanılarak ölçeklenebilir, izlenebilir ve güvenli bir ortamda yayına alınmıştır.

2. Kullanılan Teknolojiler

Uygulama Katmanı

- **Unity 3D:** Oyunun geliştirilmesi ve WebGL formatında derlenmesi (build) için kullanıldı.
- **HTML5/WASM (WebAssembly):** Oyunun tarayıcı üzerinde yüksek performansla çalışmasını sağlayan çekirdek teknolojiler.

Bulut ve Altyapı Katmanı

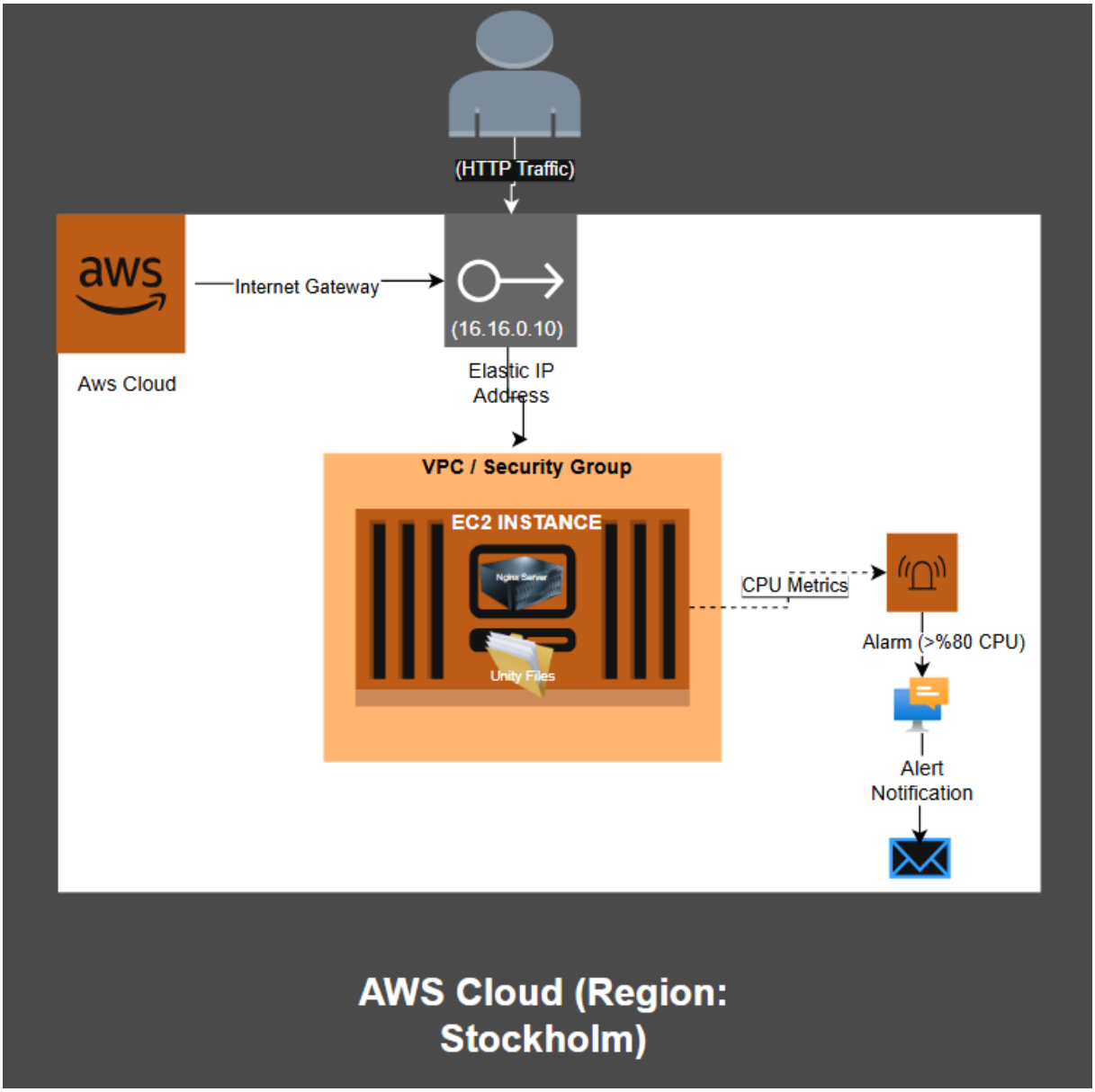
- **AWS (Amazon Web Services):** Küresel altyapısı ve geniş hizmet yelpazesi nedeniyle servis sağlayıcı olarak seçildi.
- **Amazon EC2 (Elastic Compute Cloud):** Uygulamayı barındıran sanal sunucu.
- **Ubuntu Server 24.04 LTS:** Kararlılığı ve geniş topluluk desteği nedeniyle işletim sistemi olarak tercih edildi.
- **Nginx:** Yüksek eşzamanlı bağlantı kapasitesi ve düşük kaynak tüketimi nedeniyle Web Sunucusu (Web Server) ve Reverse Proxy olarak kullanıldı.
- **AWS CloudWatch:** Sunucu kaynaklarının (CPU/RAM) proaktif izlenmesi için entegre edildi.
- **Elastic IP:** Sunucunun IP adresinin statik kalması ve DNS çözümlemelerinde kesinti yaşanmaması için yapılandırıldı.

3. Bulut Platformu Seçimi: Neden AWS?

Proje başlangıcında Azure ve DigitalOcean alternatifleri değerlendirilmiş, ancak aşağıdaki nedenlerle AWS tercih edilmiştir:

1. **Maliyet Etkinliği:** t3.micro instance tipi ve Free Tier (Ücretsiz Katman) avantajları, öğrenci bütçesiyle enterprise seviyesinde denemeler yapılmasına olanak tanıdı.
2. **Ölçeklenebilirlik:** AWS'nin sunduğu Elastic IP ve Security Group yapıları, projenin ileride Load Balancer ile büyütülmesine uygun bir zemin hazırladı.
3. **Pazar Standartları:** Sektörde en çok kullanılan sağlayıcı olması, grubumuzun endüstri standartlarını öğrenmesini sağladı.

4. Uygulama Mimari Şeması



Veri Akışı:

1. **Kullanıcı (Client):** Tarayıcıdan http://16.16.0.10 adresine istek atar.
2. **AWS Internet Gateway:** İsteği karşılar.
3. **Security Group:** Yalnızca Port 80 (HTTP) ve Port 22 (SSH) trafiğine izin verir; diğerlerini engeller.
4. **Elastic IP:** İsteği doğru EC2 sunucusuna yönlendirir.
5. **Nginx Web Server:** Gelen isteği /var/www/html dizinindeki statik Unity dosyalarıyla (index.html, Build, TemplateData) yanıtlar.
6. **CloudWatch:** Arka planda CPU kullanımını izler ve %80 eşiği aşılırsa alarm üretir.

5. Adım Adım Dağıtım (Deployment) Süreci

A. Sunucu Hazırlığı (Provisioning)

1. AWS Konsolu üzerinden **Stockholm (eu-north-1)** bölgesinde bir t3.micro instance oluşturuldu.
2. İşletim sistemi olarak **Ubuntu 24.04 LTS** seçildi.
3. Erişim güvenliği için .pem uzantılı bir SSH anahtar çifti oluşturuldu.

B. Ağ ve Güvenlik Yapılandırması

1. **Security Group** kuralları düzenlendi:
    - SSH (22): Sadece yönetim erişimi için.
    - HTTP (80): Tüm dünyadan erişim (0.0.0.0/0) için açıldı.
2. Sunucunun kapanıp açılması durumunda IP değişikliğini önlemek için **Elastic IP** tahsis edildi ve sunucu ile ilişkilendirildi (Associated).

C. Uygulama Kurulumu

1. SSH üzerinden sunucuya bağlanıldı ve sistem güncellemeleri yapıldı: sudo apt update && sudo apt upgrade.
2. Web sunucusu kurulumu: sudo apt install nginx.
3. Lokal ortamdaki Unity build dosyaları (Zip formatında), **SCP (Secure Copy Protocol)** kullanılarak sunucuya güvenli bir şekilde aktarıldı.
4. Dosyalar /var/www/html dizinine taşındı ve gerekli okuma izinleri (chmod 755) verildi.

D. İzleme (Monitoring)

1. Sistem sürekliliğini sağlamak adına **AWS CloudWatch** alarmı kuruldu.
2. CPU kullanımı 5 dakika boyunca %80'in üzerinde seyrederse, sistem yöneticisine otomatik e-posta bildirimi gönderen bir SNS (Simple Notification Service) konusu oluşturuldu.

7. Karşılaşılan Zorluklar ve Çözümler

Karşılaşılan Zorluk	Çözüm Yöntemi
IP Değişikliği Sorunu: Sunucu durdurulup yeniden başlatıldığında Public IP adresinin değişmesi erişimi kesti.	**Elastic IP** servisi kullanılarak sunucuya statik bir IP adresi atandı ve sorun kalıcı olarak çözüldü.
Dosya İzin Hataları (403 Forbidden): Nginx, Unity dosyalarını okuyamadığı için erişim hatası verdi.	Linux dosya sistemi izinleri chmod 755 komutu ile web sunucusunun okuyabileceği şekilde düzenlendi.
Sıkıştırma Formatı: Unity'nin sıkıştırılmış (compressed) dosyalarını sunucu varsayılan olarak çözümleyemedi.	Sunucu tarafında unzip aracı kuruldu ve dosya yapısı bozulmadan /var/www/html dizinine aktarım sağlandı.
SSH Erişim Sorunu: .pem anahtar dosyasının izinleri Windows ortamında sorun yarattı.	PowerShell üzerinden doğru dosya yolu belirtilerek ve dosya bütünlüğü korunarak bağlantı sağlandı.

8. Öğrenilen Dersler ve Olası İyileştirmeler (Future Work)

Öğrenilen Dersler:

- Statik IP kullanımının prodüksiyon ortamları için opsiyonel değil, zorunlu olduğu anlaşıldı.
- Bulut güvenliğinde "Security Group" yönetiminin, sunucu içi güvenlik duvarından (UFW) daha öncelikli ve etkili olduğu tecrübe edildi.
- Maliyet yönetiminin (Cost Management) teknik beceriler kadar önemli olduğu, gereksiz kaynakların kapatılması gerektiği öğrenildi.

Gelecek İçin İyileştirmeler:

1. **Domain & SSL:** IP adresi yerine bir alan adı (Domain Name) alınarak HTTPS (SSL/TLS) sertifikası ile güvenli bağlantı sağlanması.
2. **CI/CD Pipeline:** GitHub Actions kullanılarak, Unity projesine her kod atıldığında (push) otomatik olarak sunucuya deploy eden bir boru hattı kurulması.
3. **Dockerizasyon:** Uygulamanın bir Docker konteyneri haline getirilerek platform bağımsızlığının artırılması.