Google Haberler'den Veri Çekme: AWS Lambda ve S3 Kullanarak Otomatik Haber Toplama

AWS Lambda ve S3 Kullanarak Otomatik Haber Toplama

Hatice Hilal AKSOY

BEN KİMİM?

Antalya Bilim Üniversitesi, Bilgisayar mühendisliği 4. sınıf öğrencisiyim.

GİRİŞ

Günümüzde bilgiye erişim hiç olmadığı kadar kolay ve hızlı. Ancak bu bilgiler arasından önemli olanları sıralamak ve düzenlemek, özellikle de güncel haberler söz konusu olduğunda, oldukça zorlayıcı olabilir.Bu bağlamda, projenin hedefi, Google Haberler'den eğlence, sağlık ve spor gibi farklı kategorilere ait güncel haberleri düzenli aralıklarla çekmek ve bu haberlerin URL'lerini, başlıklarını, kısa açıklamalarını, kaynaklarını ve yayınlanma tarihlerini sistematik bir şekilde kaydetmektir.Bu işlem, her saat başı AWS Lambda fonksiyonu kullanılarak otomatize edilmiştir, böylece en güncel haber içeriğine sürekli erişim sağlanmaktadır. Çekilen haber verileri, AWS S3 hizmetinde JSON formatında saklanarak, verilere kolay ve hızlı bir şekilde erişim imkanı sunulmaktadır. Projedeki bu otomatik toplama ve saklama işlemi, veri analizi, trend takibi ve özel içerik üretimi gibi daha geniş uygulamalar için temel bir kaynak oluşturmaktadır.

Projede çözülmeye çalışılan temel sorun, büyük veri akışı içerisinden ilgili ve güncel haber içeriğinin hızlı ve etkili bir şekilde çekilip saklanmasıdır. Bu sorunun çözümü, haber takibi ve analizi yapan bireylerin ve kurumların, ilgilendikleri konularla ilgili bilgilere daha hızlı erişim sağlamalarına ve bu bilgileri daha etkin bir şekilde kullanabilmelerine olanak tanır. Ayrıca, bu otomatik süreç, manuel veri toplama gereksinimini ortadan kaldırarak zaman ve kaynak tasarrufu sağlar.

Bulut Hizmetlerinin Projedeki Rolü

Bu projede, Amazon Web Services (AWS) bulut hizmetleri temel alınmıştır. AWS, geniş bir hizmet yelpazesi sunarak geliştiricilere uygulamalarını kolaylıkla ölçeklendirebilir ve yönetebilir bir yapıda geliştirmelerine olanak tanır. AWS projemin etkin bir şekilde gerçekleştirilmesini sağlayan bir bulut tabanlı çözüm sunmaktadır.

Bulut hizmetlerinin kullanımı, özellikle veri yoğun uygulamalar ve otomatik iş akışları için önemli avantajlar sağlar. Projemde AWS'nin bulut hizmetlerine dayanarak, ölçeklenebilirlik, maliyet etkinliği ve yüksek erişilebilirlik gibi bulut bilişimin temel faydalarından yararlanılmıştır. Bu hizmetler, projenin herhangi bir donanım kaynağına ihtiyaç duymadan, yalnızca yazılım ve bulut hizmetleri üzerinden gerçekleştirilmesine olanak tanımıştır.

Bulut Hizmetlerinin Projedeki Rolü

Bu projede, AWS Lambda ve S3 gibi bulut hizmetlerinin kullanılmasının ana nedeni, otomatikleştirilmiş ve ölçeklenebilir bir sistem kurma ihtiyacından kaynaklanmaktadır. AWS Lambda, sunucusuz bir hesaplama hizmeti sunarak, kodun yalnızca belirli olaylar tetiklendiğinde çalıştırılmasını sağlar. Bu özellik, projede her saat başı haber verilerinin çekilip işlenmesi gibi düzenli görevlerin yönetilmesi için idealdir. Lambda'nın kullanımı, altyapı yönetimiyle ilgili zahmetten kurtarır ve geliştiricilere kodun mantığına odaklanma imkanı tanır.

AWS S3, ölçeklenebilir bir nesne depolama hizmetidir ve bu projede çekilen haber verilerinin saklanması için kullanılmıştır. S3'ün tercih edilmesinin sebebi, yüksek erişilebilirlik ve güvenlik özellikleri sunmasıdır. Verileri S3 bucket'larında saklamak, dünya çapında hızlı ve güvenilir bir şekilde erişim imkanı sağlar. Ayrıca, S3 ile verileri kolayca kategorilere ayırabilir ve otomatik olarak etiketleyebiliriz, bu da veri yönetimini ve erişimi basitleştirir.

AWS Identity and Access Management (IAM) projede, AWS kaynaklarına erişimi yönetmek ve kontrol etmek için kullanılmıştır. IAM, özellikle AWS Lambda fonksiyonları ve S3 bucket'ları gibi kaynaklara erişimde güvenliği sağlamak amacıyla kritik bir öneme sahiptir. Bu projede IAM rolleri, Lambda fonksiyonunun S3 bucket'ına veri yazabilmesi ve okuyabilmesi için gerekli izinleri tanımlamak üzere kullanılmıştır. Ayrıca, IAM politikaları ile Lambda fonksiyonunun yalnızca gerekli AWS servislerine ve kaynaklarına erişimi kısıtlanmıştır, böylece en az ayrıcalık prensibi uygulanarak güvenlik sağlanmıştır. IAM'ın kullanımı, projenin güvenliğini önemli ölçüde artırmaktadır. Örneğin, S3 bucket'larına yüklenen verilerin güvenliğini sağlamak ve yalnızca yetkilendirilmiş kullanıcıların ve servislerin erişimine izin vermek için IAM rolleri ve politikaları detaylı bir şekilde yapılandırılmıştır. Bu, özellikle hassas verilerin saklandığı ve işlendiği projelerde, veri sızıntısı ve yetkisiz erişim risklerini minimize etmek için hayati öneme sahiptir.

Projede Python Kullanımı

- Python'un okunabilirliği yüksek ve yazımı kolay sintaksı sayesinde hızlı bir geliştirme süreci sunması, projemin etkin bir şekilde ilerlemesine olanak tanımıştır.
- Geniş bir standart kütüphane yelpazesi ve üçüncü taraf modülleri ile birlikte gelir, bu da ihtiyaç duyduğum hemen her türde görevi yerine getirebilecek araçları sunar. Özellikle, veri çekme, işleme ve HTTP istekleri gibi görevlerde (Hazırlamış olduğum Proje) oldukça güçlü kütüphanelere (requests, BeautifulSoup gibi) sahiptir.
- AWS, Python geliştiricilerine Lambda üzerinde kolayca çalışabilme ve mikro servislerini hızla dağıtabilme imkanı sunar.
- Çekilen haber verilerinin analizi ve işlenmesi gibi daha ileri seviye işlemler için Python, pandas, numpy gibi kütüphaneleri ile güçlü bir altyapı sunar. Bu, projenin sadece veri toplama ve saklama ile sınırlı kalmayıp, veri üzerinde daha karmaşık işlemler gerçekleştirebilme potansiyelini de beraberinde getirir.

AWS keşfetmek ister misiniz? AWS

Projenin Oluşturulma Aşamaları

> *Projemin Local de oluşturulması, S3 bucket oluşturma, Lambda Fonksiyonu oluşturma, Kodun test edilmesi, S3 bucket de her saat json doslarının kaydı ile projem başarılı bir şekilde sonlandı.*

Local de Proje Kodlama Aşamaları:

```
import json
import requests
from bs4 import BeautifulSoup
import boto3
from datetime import datetime

def lambda_handler(event, context):
    # Bu liste, çekmek istediğim haberlerin URL'lerini içerir
    kategori_url_leri = [

"https://news.google.com/topics/CAAqJggKIiBDQkFTRWdvSUwyMHZNREpxYW5RU0FuUnlHZ0pVVWlnQVAB
hl=tr&gl=TR&ceid=TR%3Atr",

"https://news.google.com/topics/CAAqIQgKIhtDQkFTRGdvSUwyMHZNR3QwTlRFU0FuUnlLQUFQAQ?
hl=tr&gl=TR&ceid=TR%3Atr",

"https://news.google.com/topics/CAAqJggKIiBDQkFTRWdvSUwyMHZNRFp1ZEdvU0FuUnlHZ0pVVWlnQVAB
hl=tr&gl=TR&ceid=TR%3Atr"
    ]
    kategori_metinleri = {

"https://news.google.com/topics/CAAqJggKIiBDQkFTRWdvSUwyMHZNREpxYW5RU0FuUnlHZ0pVVWlnQVAB
hl=tr&gl=TR&ceid=TR%3Atr": "Eğlence",

"https://news.google.com/topics/CAAqIQgKIhtDQkFTRGdvSUwyMHZNR3QwTlRFU0FuUnlLQUFQAQ?
hl=tr&gl=TR&ceid=TR%3Atr": "Sağlık",

"https://news.google.com/topics/CAAqJggKIiBDQkFTRWdvSUwyMHZNRFp1ZEdvU0FuUnlHZ0pVVWlnQVAB
hl=tr&gl=TR&ceid=TR%3Atr": "Spor"
    }
    tum_linkler = []
    haber_sirasi = 1
    # URL'nin sırasını takip etmek için kullanılır.
    for i, url in enumerate(kategori_url_leri):
        # URL'den HTML içeriğini çeker
        r = requests.get(url)
        #HTML dokümanını işleyebilecek bir yapıya dönüştürür ve belirli HTML
elementlerine erişimi sağlar
        soup = BeautifulSoup(r.text, "html.parser")
        articles = soup.find_all("article")

        #HTML dokümandaki her <article> tagını dolaşarak, içerisindeki <a> tagından
haber makalesinin URL'sini çeker; eğer <a> tagı veya href attribute'u bulunamazsa, o
bölümü atlar.(örneğin navbar da yer alan linkleri atlamak için bu döngüyü kullandım )
        for article in articles:
            link = article.find("a")
            if not link:
                continue
```

```
            href = link.get("href")
            if href is None:
                continue

            #HTML sayfasından istediğimiz sınıfda ki bilgileri çekme
            full_url = f"https://news.google.com{href[1:]}" if href.startswith('.')
else href

            parent_div = link.find_parent()
            source = article.find("div", class_="vr1PYe").text.strip() if
article.find("div",

class_="vr1PYe") else "Kaynak Belirtilmemiş"
            time = article.find("time", class_="hvbAAd").text.strip() if article and
article.find("time",

class_="hvbAAd") else ""



            title = article.find("a", class_="gPFEn")
            title_text = title.text.strip() if title else article.text.strip()

            kategori_metni = kategori_metinleri.get(url, "Bilinmeyen Kategori")

            #Çekdiğimiz bilgileri yazdırma
            link_data = {
                "url": full_url,
                "haber başlığı": kategori_metni,
                "kısa açıklaması": title_text,
                "yayınlanma tarihi": time,
                "haber kaynağı": source,
                "kaçıncı sırada": haber_sirasi
            }
            tum_linkler.append(link_data)
            haber_sirasi += 1
    #tum_linkler listesindeki tüm haber linkleri ve ilgili bilgiler JSON formatına
dönüştürülür.
    json_data = json.dumps(tum_linkler, ensure_ascii=False, indent=4)
    #JSON dosyası için dosya adını oluşturur,json_data ya tüm bilgiler yazılır.
    file_name = f'/tmp/news_links_{datetime.now().strftime("%Y-%m-%d_%H-%M-%S")}.json'
    with open(file_name, 'w', encoding='utf-8') as file:
        file.write(json_data)

    bucket_name = 'ottoofellow'
    #boto3 S3 istemcisi oluşturur.
    s3 = boto3.client('s3')
    #JSON dosyasını ottoofellow isimli S3 bucket'ına yükler
    s3.upload_file(file_name, 'ottoofellow', file_name.split('/')[-1])

    #dosyanın başarıyla yüklendiğini belirten bir mesaj içerir.
    return {
        'statusCode': 200,
```

```
        'body': json.dumps(f'File {file_name.split("/")[-1]} uploaded successfully to {bucket_name}.')
    }
```

Gerekli kütüphaneler:

```
import json
import requests
from bs4 import BeautifulSoup
import boto3
from datetime import datetime
```

> *Note: json- Python'daki json kütüphanesi, JSON veri formatını işlemek için kullanılır. Projede, çekilen haber verilerini JSON formatında saklamak, okumak ve yazmak için bu kütüphane tercih edilmiştir. Bu, verilerin hem insanlar hem de programlar tarafından kolayca anlaşılabilir ve işlenebilir olmasını sağlar. requests- RESTful API'larla etkileşimde bulunurken gereken GET, POST gibi HTTP metodlarını destekler. Projede, Google Haberler'den haber verilerini çekmek için HTTP istekleri yapmak amacıyla requests kullanılmıştır. Bu kütüphane, web sayfalarından veri çekme işlemini kolay ve etkin bir şekilde gerçekleştirmeyi sağlar. BeautifulSoup- HTML ve XML dosyalarını ayrıştırmak için kullanılan bir Python kütüphanesidir. Projede, requests ile elde edilen ham HTML içeriğinden gerekli bilgilerin (haber başlıkları, URL'ler, kısa açıklamalar vs.) çıkarılması ve işlenmesi için BeautifulSoup tercih edilmiştir. Bu kütüphane, karmaşık web sayfası yapılarını kolayca yönetebilme ve aranan bilgilere hızlıca ulaşabilme avantajı sunar. boto3- Bu SDK, AWS kaynaklarını programatik olarak yönetme yeteneği sağlar, böylece geliştiriciler AWS hizmetlerini doğrudan Python uygulamalarından kontrol edebilirler.Projede, haber verilerini AWS S3 bucket'ına yüklemek ve bu verileri AWS üzerinde yönetmek için boto3 kullanılmıştır. datetime- Python'da tarih ve zaman işlemleri için standart bir kütüphanedir. Projede, haber verilerini işlerken ve bunları zaman damgasıyla birlikte kaydederken datetime modülü kullanılmıştır. Bu, verilerin ne zaman çekildiğini belirlemek ve veriler üzerinde zaman bazlı sorgulamalar yapabilmek için önemlidir.*

S3 bucket oluşturmak:

IAM rolleri, Lambda fonksiyonunun S3 bucket'ına veri yazabilmesi ve okuyabilmesi için gerekli izinleri tanımlamak: IAM Hizmetine girdikten sonra ilk olarak bu proje için IAM rolü oluşturuyoruz. Access management kısmından Policies seçiyoruz. Dashboard'unda Create Policies seçtikten sonra bizim için sadece write gerekli olduğu için Access Level kısıtlıyoruz. All resource dedikten sonra servisimizi isimlendiriyoruz. Ardından Role kısmına gelip oluşturduğumuz AmazonS3FullAccess permission policies ekliyoruz ve CloudBridge(CloudWatch Events)(Schedule expression: cron(0 * * * ? *))-Burada kullanmış olduğumuz every_hour rules da biz oluşturuyoruz- kullanacağımız yani projemizi bir saat ile döndüreceğimiz için AWSLambdaBasicExecutionRole seçiyoruz ve isimlendirdiğimiz Role artık oluşuyor. Bucket Name, Region ve Bucket Ayarlarını yapılandırdıktan sonra kaydettim. AWS S3 bucket'ınız başarıyla oluşturulmuş olacak. Artık bu bucket içerisine dosya yükleyebilir, dosyaları depolayabilir ve gerektiğinde bu dosyalara erişim sağlayabiliriz.

Lambda Fonksiyonu oluşturma:

Lambda dashboard'unda "Create function" butonuna tıkladım. Author from scratch seçtim çünkü sıfırdan bir fonksiyon oluşturmak istiyorum. Function name, Runtime(Python3.10), Execution role(AWS kaynaklarına erişimi için bir IAM rolü seçtim) Tüm detayları girdikten sonra "Create function" ile fonksiyonumu oluşturdum. Lambda fonksiyonunuzun bellek, timeout süresi, VPC yapılandırması gibi ayarlarını "Configuration" sekmesinden yapılandırabilirdim ve ben sadece timeout süresini 30 saniye yaptım. Fonksiyonunuzu test etmek için, "Test" butonuna tıklayın ve yeni bir test etkinliği oluşturun. Oluşturduğunuz test etkinliği ile fonksiyonunuzu test ederek, beklenen çıktıları aldığınızdan emin oldum.

AWS EventBridge (CloudWatch Events) Tetikleyicisinin Projedeki Rolü

AWS EventBridge, AWS hizmetleri ve uygulamalar arasında olayları yönlendiren bir sunucusuz olay yönlendirme hizmetidir. Bu projede, AWS Lambda fonksiyonunu her saat başı otomatik olarak tetiklemek için EventBridge kullandım. "Amazon EventBridge"yi seçerek EventBridge hizmet dashboard'unda "Create rule" butonuna tıklayıp Rule type olarak "Schedule" olarak belirledim ve Fixed rate of seçeneğini seçip Cron expression belirledim. Son olarak create ile kuralı oluşturdum.

Karşılaşılan Zorluklar ve Çözümler

Projede karşılaştığım başlıca zorluklardan biri,çekilen verilerin düzenli ve anlamlı bir formatta saklanmasıydı. Her haberin benzersiz bir yapıya sahip olması, verileri standart bir formatta saklamayı zorlaştırdı. Bu sorunu çözmek için, veri saklama yapısını mümkün olduğunca genel tutarak ve olası tüm senaryolara uyacak şekilde esnek bir JSON ve Python3.10 kullandım.

Kaynaklar

| AWS Lambda | AWS Resmi Dokümantasyonu | | AWS S3 | AWS Resmi Dokümantasyonu |

Hatice Hilal AKSOY Linkedin