Dogo - A distributed application that utilizes the Google Cloud ecosystem

Team members: Fanaru Victor, Rumeghea Leonard, Opariuc Rares-Ioan

Link to Github repo: https://github.com/OpariucRares/PCD-2023-2024

1) Application Concept & Ideas

Dogo is a complete pet care application that caters to pet owners' different requirements, including dog walking, grooming, medical treatment, and pet sitting. Dogo makes use of well-known resources like the Google Maps SDK for Android, Directions API, and Distance Matrix API to improve the user experience. These applications provide precise location-based services, allowing pet owners to readily identify local pet-related businesses and map out effective itineraries for their pets' activities.

Dogo also takes advantage of Google Cloud Platform's powerful cloud services, such as Cloud Functions, SQLServer from Google, and VM Instances. This assures the scalability, dependability, and smooth execution of backend activities, which improves the application's speed and responsiveness. Furthermore, Dogo uses SignalR-enabled web socket communication to give real-time updates and notifications to both pet owners and service providers, resulting in a dynamic and participatory user experience.

2) Chosen services

Google Maps SDK for Android, the Directions API, and Distance Matrix API are key components of Google's portfolio of location-based services, which are noted for their high performance and stability.

Maps SDK for Android

The **Maps SDK for Android** is an advanced package that allows Android developers to include dynamic maps and location-based features in their applications. It has rapid rendering capabilities and smooth interaction, resulting in a high-performance user experience, particularly when working with complicated map data and overlays. Google Maps is a popular choice for location-based services and mapping capabilities, with over 1 billion active users and 1 million companies worldwide. Integrating Google Maps into your app helps you to interact with a big user and business community while also providing seamless navigation, precise location data, and a wide set of geospatial capabilities. This makes it an excellent solution for applications such as transportation apps, e-commerce platforms, and real estate websites that require improved user experience, operations management, and value-added services.

Directions API

This **Directions API** is well-known for its reliability and accuracy in delivering optimal routes and turn-by-turn navigation directions. The API uses Google's enormous dataset and smart algorithms to provide fast and reliable routing information for a variety of transportation modes, including driving, walking, and cycling. Its efficiency is seen in its capacity to handle millions of queries daily, providing developers and consumers worldwide with real-time traffic information, alternate routes, and exact arrival forecasts. The Directions API's reliability assures continuous service delivery, allowing apps to provide dependable navigation functions with no downtime or service disruptions.

Distance Matrix API

Similarly, the **Distance Matrix API** performs well and is stable, calculating journey distances and durations between numerous places quickly. Whether estimating the shortest driving distance between two places or optimizing delivery routes for a fleet of vehicles, the API provides precise answers with low latency. Its scalability enables it to handle high numbers of requests easily, making it suited for a broad range of applications, including logistics and transportation planning, location-based marketing, and asset monitoring. The stability of the Distance Matrix API assures continuous performance, allowing developers to confidently construct apps that rely on exact distance and journey time computations.

In brief, Google's suite of location-based services, which includes the Maps SDK for Android, Directions API, and Distance Matrix API, is well-known for its high performance and stability. Developers can rely on these services to provide quick, accurate, and dependable mapping, navigation, and routing capabilities in their Android applications, resulting in better user experiences and opening up a plethora of location-based use cases. Google's location-based services continue to be the gold standard for developers looking for high-performance and trustworthy solutions for their apps, thanks to solid infrastructure and ongoing advancements.

3) Companies Utilizing Google's Mapping Services

- **Uber**: Uber relies on Google Maps services not only for precise navigation but also for optimizing driver routes and providing real-time traffic updates. By leveraging Google Maps, Uber ensures seamless and efficient rides for both drivers and passengers, enhancing overall user satisfaction and experience.

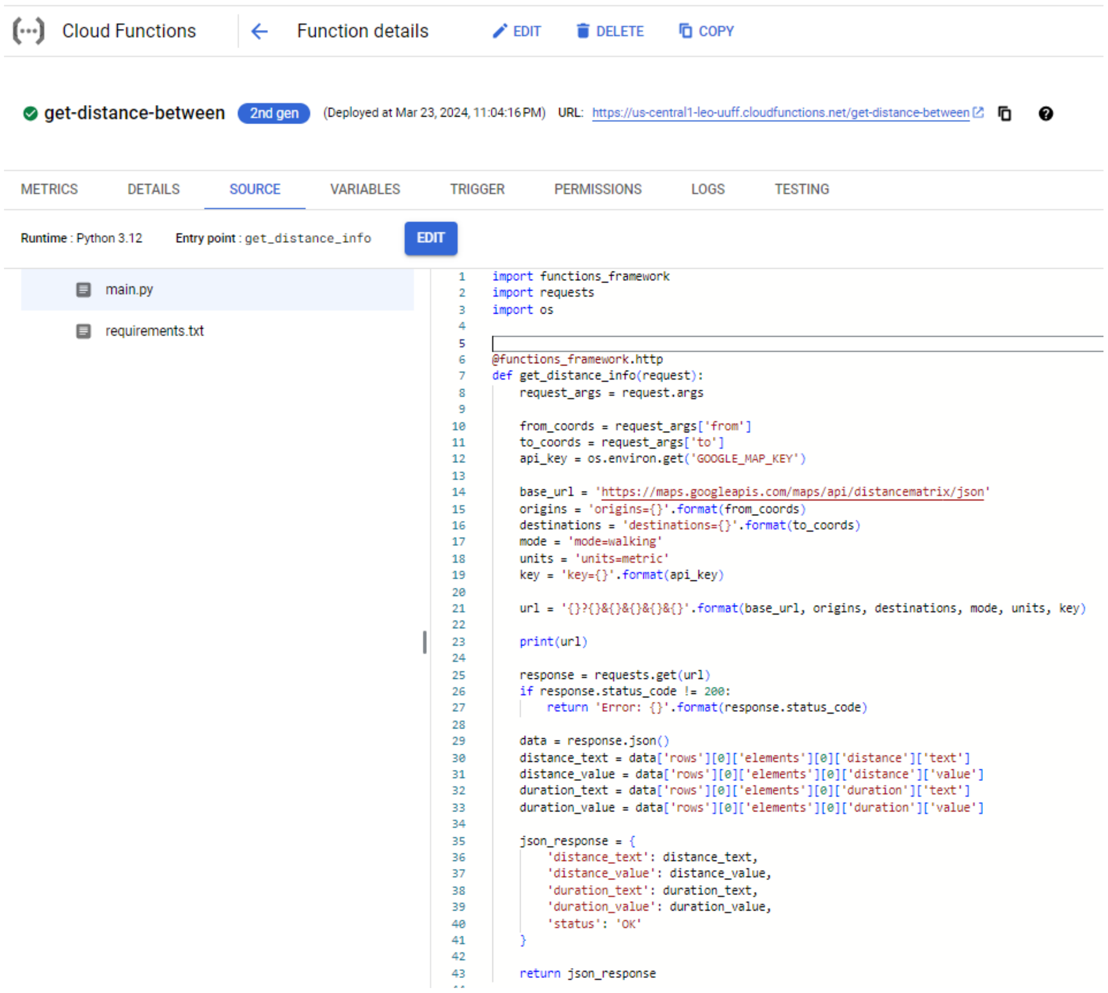
- **Lyft**: Similar to Uber, Lyft integrates Google Maps into its ride-hailing app to offer reliable navigation tools and route optimization features. By utilizing Google Maps, Lyft ensures smooth and efficient rides, enabling drivers to navigate through traffic seamlessly and passengers to reach their destinations promptly.

- **Bolt**: As a transportation network company operating globally, Bolt utilizes Google Maps services within its mobile application to facilitate accurate location tracking and route planning for drivers and passengers. By incorporating Google Maps and Directions API, Bolt ensures optimal routing and efficient ride-hailing experiences for its users across various regions.

- **Airbnb:** Airbnb enhances its property listings by integrating Google Maps to provide customers with detailed location information and interactive maps. Through Google Maps integration, Airbnb enables travelers to explore properties conveniently, visualize their surroundings, and gain insights into nearby amenities and attractions, ultimately enhancing their booking experience.

4) Google Cloud Functions

Google Cloud Functions is a serverless execution platform that allows developers to run code in response to events without having to manage the underlying infrastructure. Google Cloud Platform (GCP) provides a **Function-as-a-Service** (**FaaS**) service that enables developers to focus on writing code rather than deploying, scaling, or managing servers. Google Cloud Functions' key characteristics include event-driven execution, support for different programming languages, a pay-per-use pricing mechanism, and easy connection with other Google Cloud Platform services. This event-driven architecture enables developers to create serverless apps that respond to real-time events and grow dynamically in response to demand. The platform also supports easy connection with other GCP services, enabling developers to create sophisticated applications that make use of the whole GCP ecosystem.



get-distance-between 2nd gen (Deployed at Mar 23, 2024, 11:04:16 PM) URL: https://us-central1-leo-uuff.cloudfunctions.net/get-distance-between

Runtime : Python 3.12   Entry point : get_distance_info

```python
import functions_framework
import requests
import os


@functions_framework.http
def get_distance_info(request):
    request_args = request.args

    from_coords = request_args['from']
    to_coords = request_args['to']
    api_key = os.environ.get('GOOGLE_MAP_KEY')

    base_url = 'https://maps.googleapis.com/maps/api/distancematrix/json'
    origins = 'origins={}'.format(from_coords)
    destinations = 'destinations={}'.format(to_coords)
    mode = 'mode=walking'
    units = 'units=metric'
    key = 'key={}'.format(api_key)

    url = '{}?{}&{}&{}&{}&{}'.format(base_url, origins, destinations, mode, units, key)

    print(url)

    response = requests.get(url)
    if response.status_code != 200:
        return 'Error: {}'.format(response.status_code)

    data = response.json()
    distance_text = data['rows'][0]['elements'][0]['distance']['text']
    distance_value = data['rows'][0]['elements'][0]['distance']['value']
    duration_text = data['rows'][0]['elements'][0]['duration']['text']
    duration_value = data['rows'][0]['elements'][0]['duration']['value']

    json_response = {
        'distance_text': distance_text,
        'distance_value': distance_value,
        'duration_text': duration_text,
        'duration_value': duration_value,
        'status': 'OK'
    }

    return json_response
```

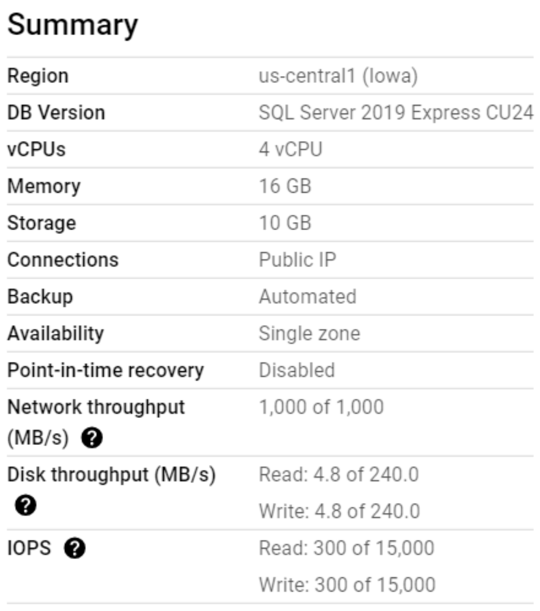
Google Cloud Function Implementation

5) Cloud Backend in .NET 7

We made the decision to host our application infrastructure using Google Cloud services. We successfully integrated the Cloud SQL database solution and also looked into other hosting solutions for our server, assessing the suitability and scalability of both Cloud Run and Compute Engine.

SQL Server (Google Cloud)

We created the necessary SQL script using the Entity Framework in order to create the table in the hosted database environment of our choice. We ultimately decided to use SQL Server 2019 Express CU24 as the hosted database. Because of the nature of our exercise, we chose Single Zone availability, but we recognized that this configuration is not suited for production situations because it does not provide failover capabilities in the event of an outage. Multiple zones are the recommended option for dispersed applications, despite its higher cost. The storage type is SSD which stand out for its emphasis on performance efficiency. We demonstrate the configured database setup with reference to the image below.

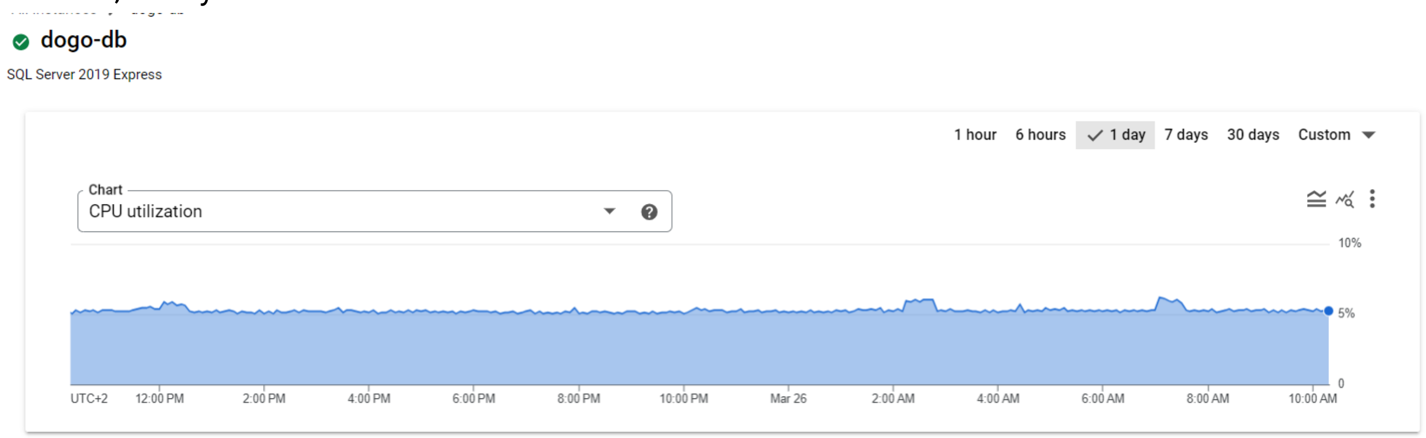


Summary

Region	us-central1 (Iowa)
DB Version	SQL Server 2019 Express CU24
vCPUs	4 vCPU
Memory	16 GB
Storage	10 GB
Connections	Public IP
Backup	Automated
Availability	Single zone
Point-in-time recovery	Disabled
Network throughput (MB/s)	1,000 of 1,000
Disk throughput (MB/s)	Read: 4.8 of 240.0 Write: 4.8 of 240.0
IOPS	Read: 300 of 15,000 Write: 300 of 15,000

The configuration of the hosted database

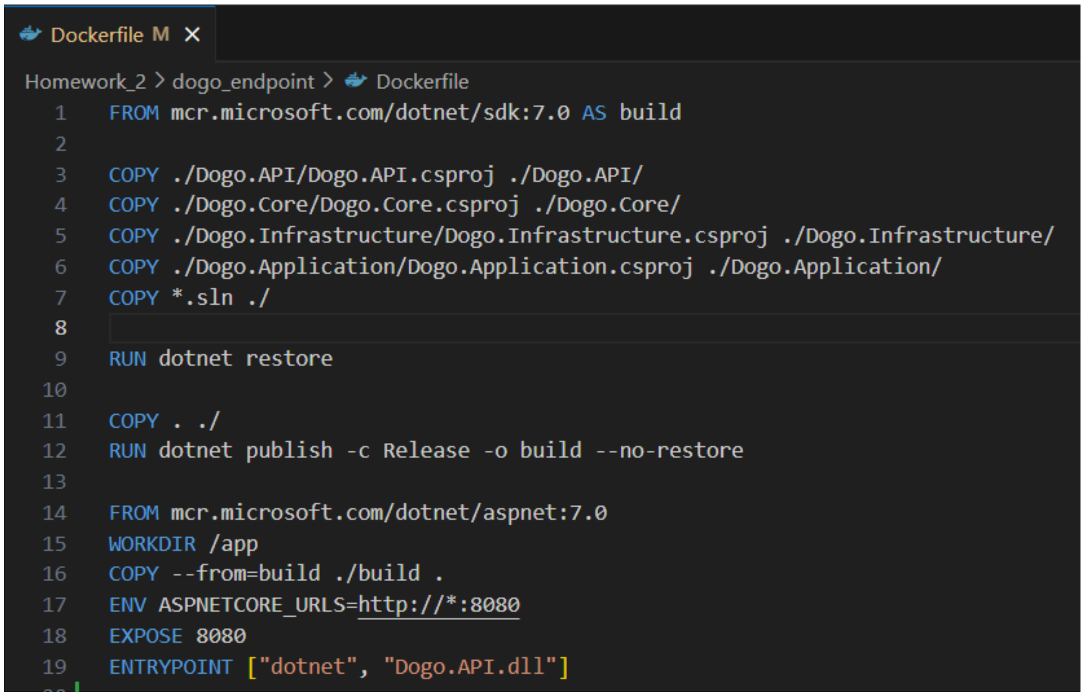
To access the database, we use the Public IP for the server name, the username, and the password generated for this database. If we analyze the CPU utilization of our database, when it is not used, it stays around 5% utilization.



CloudRun

Initially, our strategy involved hosting the server utilizing the Cloud Run option. To do this, we crafted a Dockerfile which helps with the creation of an image, subsequently stored in the Artifact Registry through the cloudbuild.yaml configuration file.

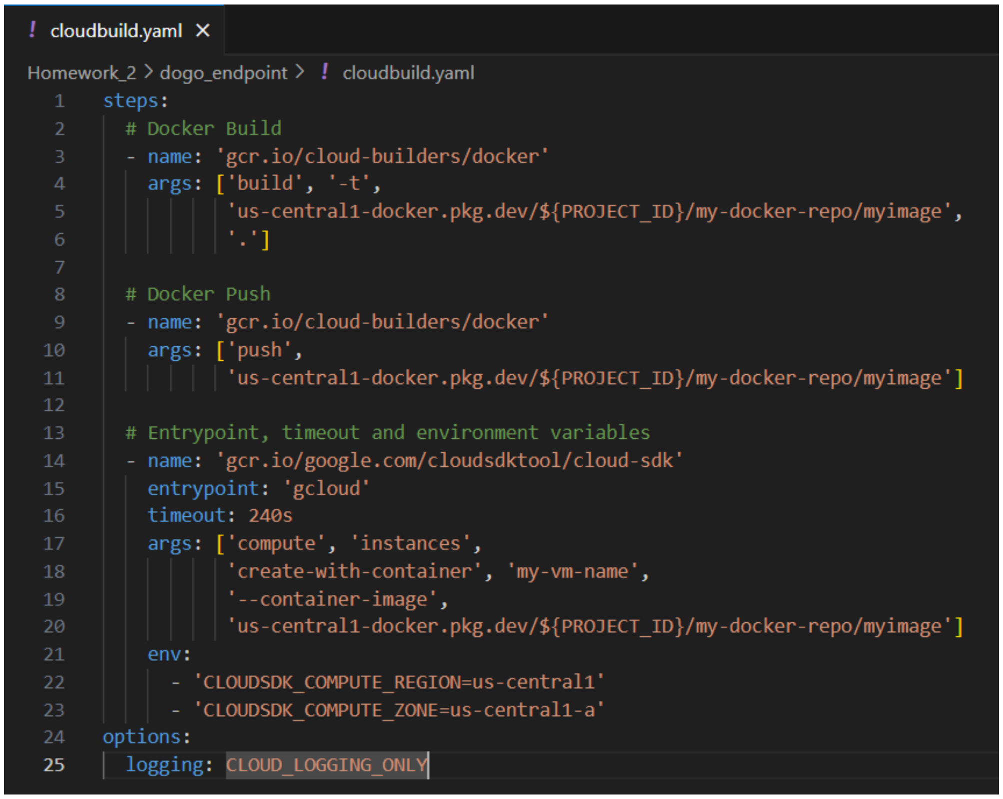
First, the docker file takes all the csproj files, sln files and restores them to have a faster build. After that, we copy the rest of the files to the image and build for the release. To create the image locally, we use the command **docker build -t dogoapp .** and to run it we can use the command **docker run -p 5001:8080 dogoapp**.



```
FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build

COPY ./Dogo.API/Dogo.API.csproj ./Dogo.API/
COPY ./Dogo.Core/Dogo.Core.csproj ./Dogo.Core/
COPY ./Dogo.Infrastructure/Dogo.Infrastructure.csproj ./Dogo.Infrastructure/
COPY ./Dogo.Application/Dogo.Application.csproj ./Dogo.Application/
COPY *.sln ./

RUN dotnet restore

COPY . ./
RUN dotnet publish -c Release -o build --no-restore

FROM mcr.microsoft.com/dotnet/aspnet:7.0
WORKDIR /app
COPY --from=build ./build .
ENV ASPNETCORE_URLS=http://*:8080
EXPOSE 8080
ENTRYPOINT ["dotnet", "Dogo.API.dll"]
```

The content of the docker file

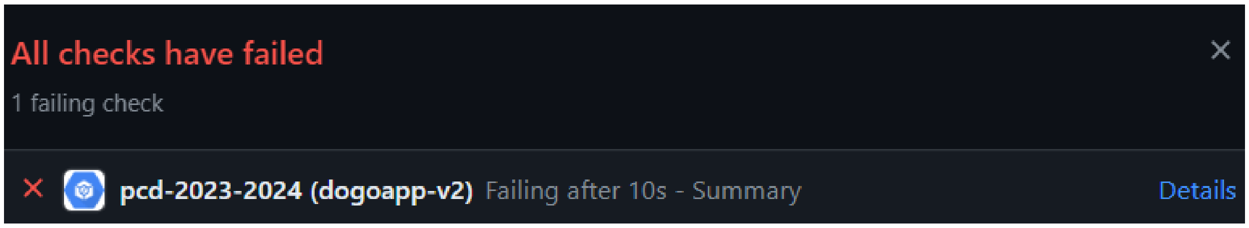
After that, we connect our repository PCD to Google Cloud Run. To use this docker image, we create a **cloudbuild.yaml** file. It puts the docker image, adds it to the Artifact registry, and creates a Google Compute Engine virtual machine instance with a container using the specified Docker image



```yaml
steps:
  # Docker Build
  - name: 'gcr.io/cloud-builders/docker'
    args: ['build', '-t',
           'us-central1-docker.pkg.dev/${PROJECT_ID}/my-docker-repo/myimage',
           '.']

  # Docker Push
  - name: 'gcr.io/cloud-builders/docker'
    args: ['push',
           'us-central1-docker.pkg.dev/${PROJECT_ID}/my-docker-repo/myimage']

  # Entrypoint, timeout and environment variables
  - name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'
    entrypoint: 'gcloud'
    timeout: 240s
    args: ['compute', 'instances',
           'create-with-container', 'my-vm-name',
           '--container-image',
           'us-central1-docker.pkg.dev/${PROJECT_ID}/my-docker-repo/myimage']
    env:
      - 'CLOUDSDK_COMPUTE_REGION=us-central1'
      - 'CLOUDSDK_COMPUTE_ZONE=us-central1-a'
options:
  logging: CLOUD_LOGGING_ONLY
```

The content of the cloudbuild.yaml file

After the SQL script was created and the trigger was activated using the Cloud Run console, we continued to have build errors that were not evident from the log files. As a result, we switched to hosting on a Compute Engine instance. Next, we want to make sure that the cloudbuild.yaml file is executed successfully, so that's our goal. This build trigger simplifies our deployment process by activating after every commit.



All checks have failed

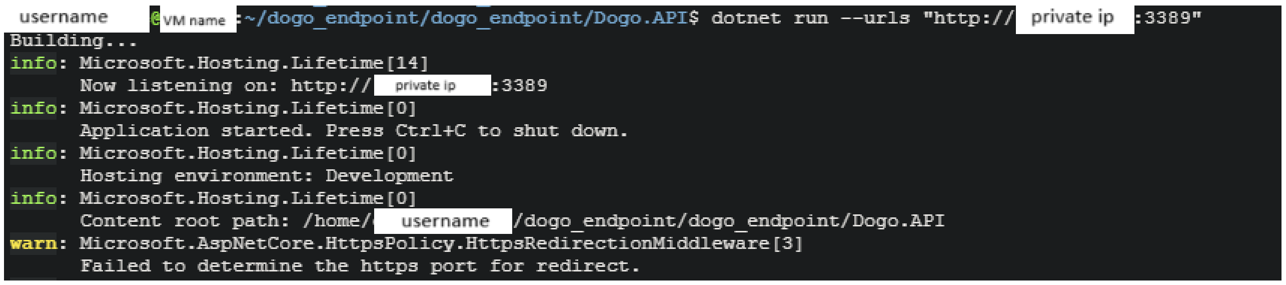
1 failing check

pcd-2023-2024 (dogoapp-v2) Failing after 10s - Summary

The failed Cloud Build run

Google VM Instances (Compute Engine)

We created a virtual machine in the cloud using the Compute Engine option from the Google Cloud. We used the same steps from the docker to this virtual machine: installed the .net7 packages, built the app using the private IP and to access the IPs, we used the public IP. From the VM side, the below image shows an example. If you want to access the APIs, you need the PUBLIC_IP:3389/api/v1/resources.



```
username @VM name :~/dogo_endpoint/dogo_endpoint/Dogo.API$ dotnet run --urls "http:// private ip :3389"
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http:// private ip :3389
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /home/ username /dogo_endpoint/dogo_endpoint/Dogo.API
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```

An example run of the VM instance

Websockets using SignalR

SignalR is a real-time communication library for adding live web functionality to applications. Developed by Microsoft, it facilitates the creation of interactive, dynamic web applications by enabling bidirectional communication between the server and client in real-time.

At its core, SignalR utilizes web sockets to establish a persistent connection between the server and the client, allowing data to be exchanged instantly without the need for frequent HTTP requests. However, it also provides fallback mechanisms such as long polling or server-sent events for environments where web sockets aren't supported.

Here's how SignalR works:

1. **Connection Establishment**: SignalR handles the complexity of establishing and maintaining connections between clients and servers, abstracting away the underlying communication protocol details.

2. **Real-Time Communication**: Once a connection is established, SignalR enables real-time communication between clients and servers. This means that data can be pushed from the server to the connected clients instantly, without requiring the clients to make requests.

3. **Hub-Based Architecture**: SignalR uses a hub-based architecture where clients can call methods on the server, and vice versa, by invoking methods on a hub. Hubs act as a high-level pipeline that allows communication between server and client code.

Benefits of using SignalR:

1. **Real-Time Updates**: SignalR enables real-time updates in web applications, making them more interactive and engaging. This is particularly useful for applications that require live data updates, such as chat applications, live sports scores, or collaborative editing tools.

2. **Reduced Latency**: By establishing persistent connections and pushing updates instantly to clients, SignalR reduces latency compared to traditional HTTP polling mechanisms. This results in a more responsive user experience.

3. **Scalability**: SignalR is designed to scale with the needs of your application. It supports load balancing and can be deployed across multiple servers, allowing for horizontal scaling to accommodate a large number of concurrent connections.

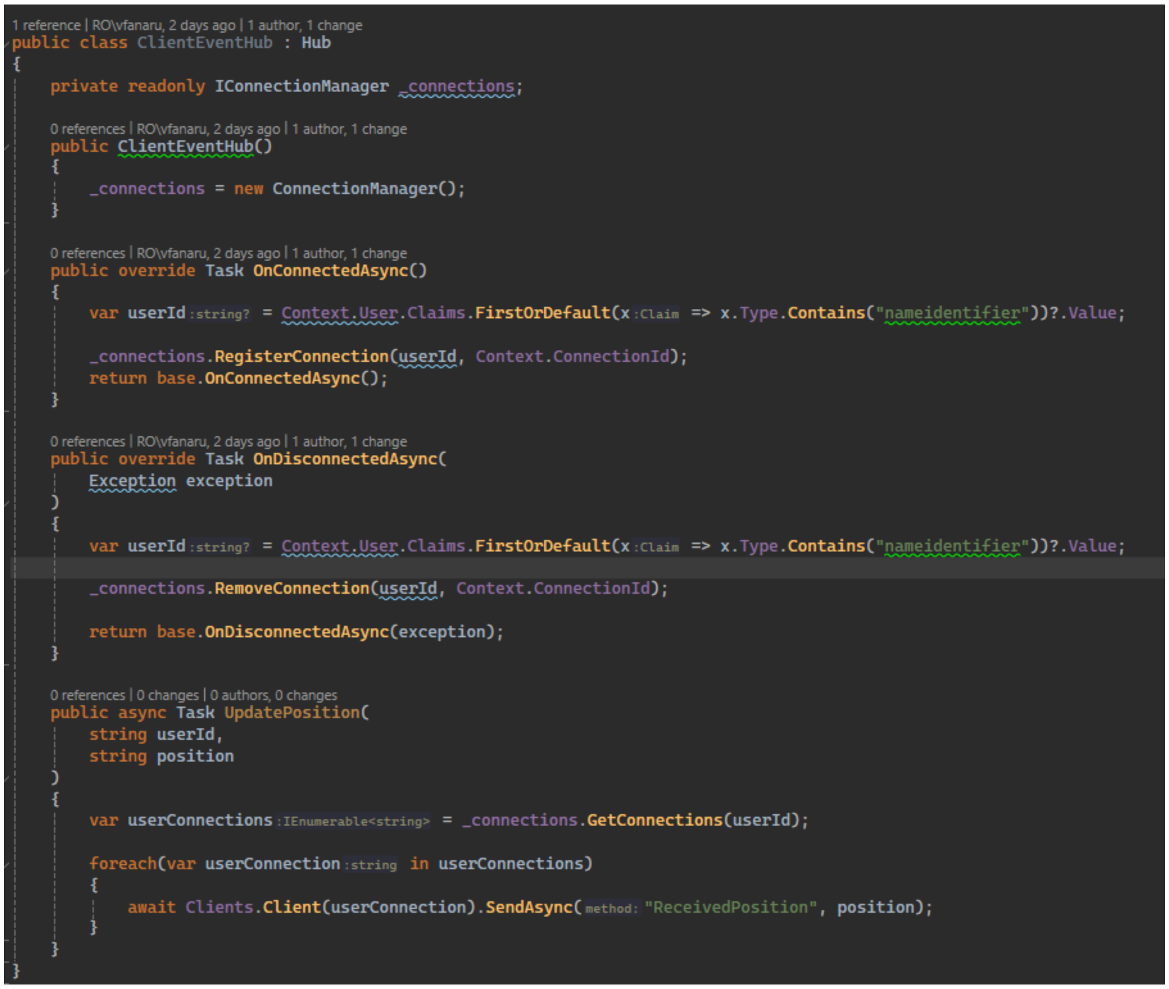
4. **Cross-Platform Compatibility**: SignalR supports various client platforms, including web browsers, mobile devices, and desktop applications. It also provides client libraries for popular frameworks such as JavaScript, .NET, and Java, making it easy to integrate with existing technology stacks.

5. **Simplified Development**: SignalR abstracts away much of the complexity involved in implementing real-time communication, allowing developers to focus on building application features rather than managing low-level networking protocols.

6. **Fallback Mechanisms**: SignalR provides fallback mechanisms such as long polling and server-sent events, ensuring compatibility with older browsers or environments where web sockets are not available.

Overall, SignalR is a powerful tool for adding real-time functionality to web applications, enabling developers to create highly interactive and responsive user experiences with ease.

SignalR used in our application:



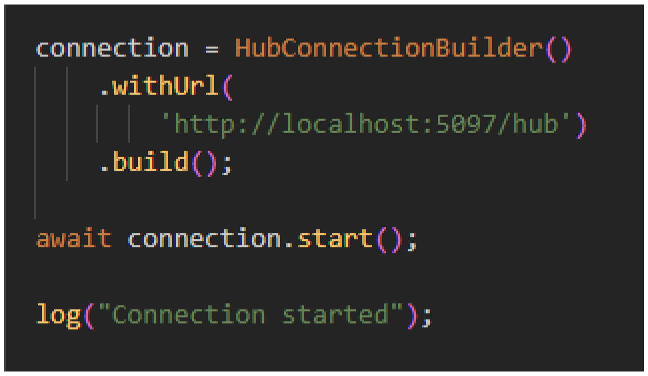
```
1 reference | RO\vfanaru, 2 days ago | 1 author, 1 change
public class ClientEventHub : Hub
{
    private readonly IConnectionManager _connections;

    0 references | RO\vfanaru, 2 days ago | 1 author, 1 change
    public ClientEventHub()
    {
        _connections = new ConnectionManager();
    }

    0 references | RO\vfanaru, 2 days ago | 1 author, 1 change
    public override Task OnConnectedAsync()
    {
        var userId :string? = Context.User.Claims.FirstOrDefault(x :Claim => x.Type.Contains("nameidentifier"))?.Value;

        _connections.RegisterConnection(userId, Context.ConnectionId);
        return base.OnConnectedAsync();
    }

    0 references | RO\vfanaru, 2 days ago | 1 author, 1 change
    public override Task OnDisconnectedAsync(
        Exception exception
    )
    {
        var userId :string? = Context.User.Claims.FirstOrDefault(x :Claim => x.Type.Contains("nameidentifier"))?.Value;

        _connections.RemoveConnection(userId, Context.ConnectionId);

        return base.OnDisconnectedAsync(exception);
    }

    0 references | 0 changes | 0 authors, 0 changes
    public async Task UpdatePosition(
        string userId,
        string position
    )
    {
        var userConnections :IEnumerable<string> = _connections.GetConnections(userId);

        foreach(var userConnection :string in userConnections)
        {
            await Clients.Client(userConnection).SendAsync( method: "ReceivedPosition", position);
        }
    }
}
```

ClientEventHub represents the hub where the user connections are managed by using a Connection Manager. Each user can connect from multiple devices, being identified and notified on all connected devices using their unique Id.

ClientEventHub is responsible for intersecting each connection and passing it to ConnectionManager and each disconnection so the hub doesn't keep old connections or miss out on new ones.

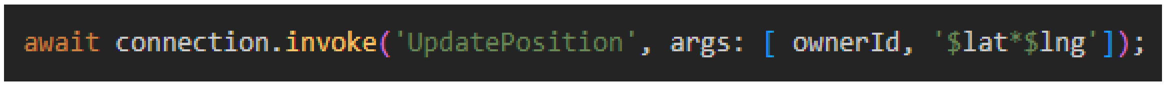
UpdatePosition event is called by clients in order to update the position on the map for all connected devices of a client. In our case for the owner of the pet.

On the client side when the application is initialized the connection is created and managed, once the application is shut down the connection is closed.



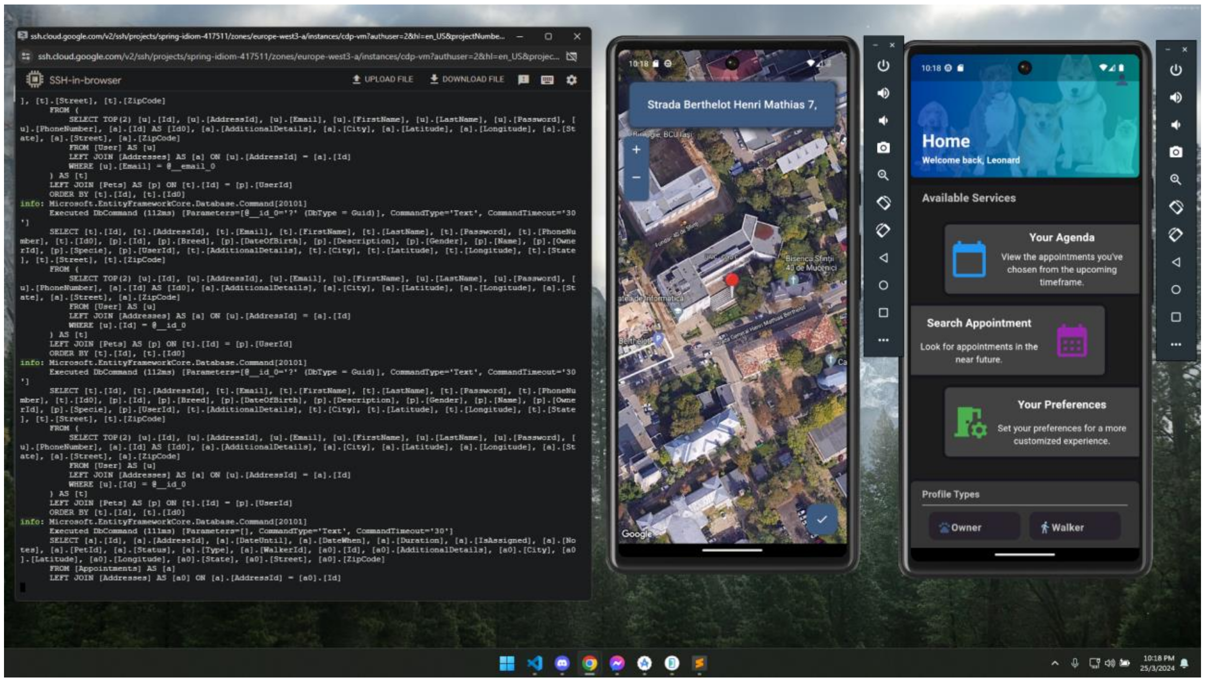
```
connection = HubConnectionBuilder()
    .withUrl(
        'http://localhost:5097/hub')
    .build();

await connection.start();

log("Connection started");
```

When ever the current position of a care taker is updated when it has in its care a pet the owner is notified so it can keep track of where its pet is.



```
await connection.invoke('UpdatePosition', args: [ ownerId, '$lat*$lng']);
```

6) Statistics & Metrics


ssh.cloud.google.com/v2/ssh/projects/spring-idiom-417511/zones/europe-west3-a/instances/cdp-vm?authuser=2&hl=en_US&projectNumbe...
ssh.cloud.google.com/v2/ssh/projects/spring-idiom-417511/zones/europe-west3-a/instances/cdp-vm?authuser=2&hl=en_US&projec...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
Strada Berthelot Henri Mathias 7,
Google
Home
Welcome back, Leonard
Available Services
Your Agenda
View the appointments you've chosen from the upcoming timeframe.
Search Appointment
Look for appointments in the near future.
Your Preferences
Set your preferences for a more customized experience.
Profile Types
Owner
Walker
10:18 PM
25/3/2024

```
], [t].[Street], [t].[ZipCode]
      FROM (
          SELECT TOP(2) [u].[Id], [u].[AddressId], [u].[Email], [u].[FirstName], [u].[LastName], [u].[Password], [
u].[PhoneNumber], [a].[Id] AS [Id0], [a].[AdditionalDetails], [a].[City], [a].[Latitude], [a].[Longitude], [a].[St
ate], [a].[Street], [a].[ZipCode]
          FROM [User] AS [u]
          LEFT JOIN [Addresses] AS [a] ON [u].[AddressId] = [a].[Id]
          WHERE [u].[Email] = @__email_0
      ) AS [t]
      LEFT JOIN [Pets] AS [p] ON [t].[Id] = [p].[UserId]
      ORDER BY [t].[Id], [t].[Id0]
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (112ms) [Parameters=[@__id_0='?' (DbType = Guid)], CommandType='Text', CommandTimeout='30
']
      SELECT [t].[Id], [t].[AddressId], [t].[Email], [t].[FirstName], [t].[LastName], [t].[Password], [t].[PhoneNu
mber], [t].[Id0], [p].[Id], [p].[Breed], [p].[DateOfBirth], [p].[Description], [p].[Gender], [p].[Name], [p].[Owne
rId], [p].[Specie], [p].[UserId], [t].[AdditionalDetails], [t].[City], [t].[Latitude], [t].[Longitude], [t].[State
], [t].[Street], [t].[ZipCode]
      FROM (
          SELECT TOP(2) [u].[Id], [u].[AddressId], [u].[Email], [u].[FirstName], [u].[LastName], [u].[Password], [
u].[PhoneNumber], [a].[Id] AS [Id0], [a].[AdditionalDetails], [a].[City], [a].[Latitude], [a].[Longitude], [a].[St
ate], [a].[Street], [a].[ZipCode]
          FROM [User] AS [u]
          LEFT JOIN [Addresses] AS [a] ON [u].[AddressId] = [a].[Id]
          WHERE [u].[Id] = @__id_0
      ) AS [t]
      LEFT JOIN [Pets] AS [p] ON [t].[Id] = [p].[UserId]
      ORDER BY [t].[Id], [t].[Id0]
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (112ms) [Parameters=[@__id_0='?' (DbType = Guid)], CommandType='Text', CommandTimeout='30
']
      SELECT [t].[Id], [t].[AddressId], [t].[Email], [t].[FirstName], [t].[LastName], [t].[Password], [t].[PhoneNu
mber], [t].[Id0], [p].[Id], [p].[Breed], [p].[DateOfBirth], [p].[Description], [p].[Gender], [p].[Name], [p].[Owne
rId], [p].[Specie], [p].[UserId], [t].[AdditionalDetails], [t].[City], [t].[Latitude], [t].[Longitude], [t].[State
], [t].[Street], [t].[ZipCode]
      FROM (
          SELECT TOP(2) [u].[Id], [u].[AddressId], [u].[Email], [u].[FirstName], [u].[LastName], [u].[Password], [
u].[PhoneNumber], [a].[Id] AS [Id0], [a].[AdditionalDetails], [a].[City], [a].[Latitude], [a].[Longitude], [a].[St
ate], [a].[Street], [a].[ZipCode]
          FROM [User] AS [u]
          LEFT JOIN [Addresses] AS [a] ON [u].[AddressId] = [a].[Id]
          WHERE [u].[Id] = @__id_0
      ) AS [t]
      LEFT JOIN [Pets] AS [p] ON [t].[Id] = [p].[UserId]
      ORDER BY [t].[Id], [t].[Id0]
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (111ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT [a].[Id], [a].[AddressId], [a].[DateUntil], [a].[DateWhen], [a].[Duration], [a].[IsAssigned], [a].[No
tes], [a].[PetId], [a].[Status], [a].[Type], [a].[WalkerId], [a0].[Id], [a0].[AdditionalDetails], [a0].[City], [a0
].[Latitude], [a0].[Longitude], [a0].[State], [a0].[Street], [a0].[ZipCode]
      FROM [Appointments] AS [a]
      LEFT JOIN [Addresses] AS [a0] ON [a].[AddressId] = [a0].[Id]
```

A demo with the app functionality

Name	↓ Requests	Errors (%)	Latency, median (ms)	Latency, 95% (ms)
Distance Matrix API	2,965	0	6	33
Maps SDK for Android	21	0		
Directions API	12	0	54	123

The metrics of the used services