UCL and IBM AR Avatar Receptionist
Documentation / Deployment Manual

This manual describes how to set up and use our project to create your own Augmented Reality avatar receptionist.

There are four main platforms required to set up the project:

1. Android Device
2. Unity Game Engine
3. IBM Watson Cloud
4. Azure Cloud

To deploy our avatar onto your own device, you only need to follow stages 1 and 2 above. Stages 3 and 4 allow you to see and edit the backend of our avatar to adapt it for your own organisation.

NB: Operating costs
To operate our project's database backend on an Azure SQL server instance, the estimated operating costs could be up to £316 a month (for a 5GB database running full time).

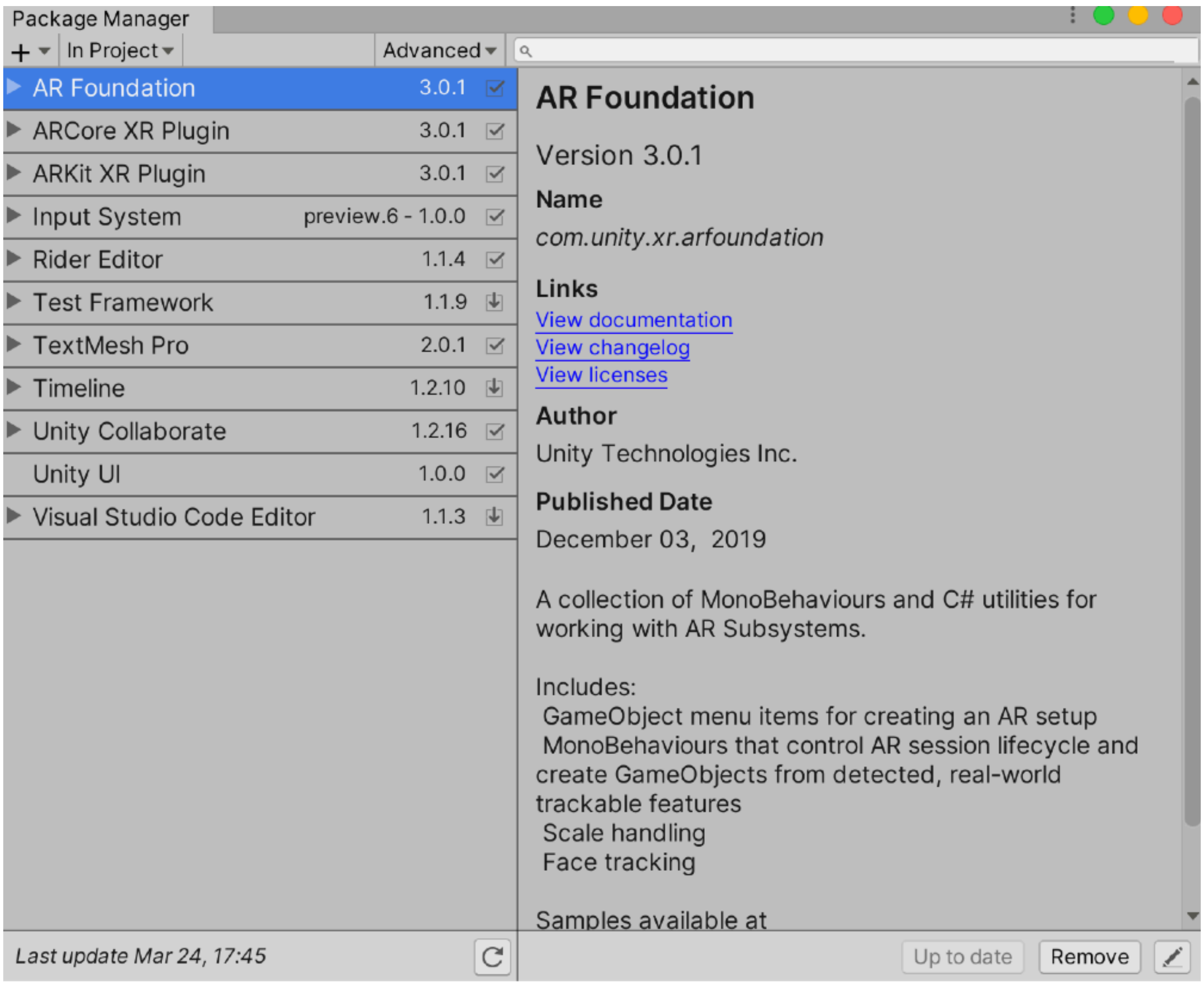
Section 1. Android Device

1. To build the project, you need an ARCore enabled Android device running Android 7.0 (Nougat) or above. To check if your device supports ARCore, please see https://developers.google.com/ar/discover/supported-devices
2. You need a cable to connect your device to the computer you intend to work on. When the cable is plugged in, you may need to enable USB Debugging to see console messages displayed in Unity to solve any build errors. See here for steps on how to enable USB debugging here https://www.getdroidtips.com/enable-usb-debugging/ NB. This will vary from device to device, so you might want to search for your own device specific info.
3. You may need to install “AR Play Store Tools” or a similar package from the Google Play store to enable AR apps to run on your device. Hopefully your phone will prompt you for this when you try to run the app in the next step, but we are aware that it does not do this on some devices.
4. Plug your device into the computer.

Section 2. Unity Game Engine

2.1. Setting Up the Unity Environment.

1. Install and run Unity 2019.3.0f6 on your computer. You can download different Unity versions from the Unity hub which can be downloaded here https://unity3d.com/get-unity/download. The project may work on other Unity versions but we have not tested it.
2. Make sure you have installed Android Build Support and Android SDK/OpenJDK when you installed unity. If you don't have these installed, you can install them from the unity hub, see here https://docs.unity3d.com/Manual/android-sdksetup.html
3. From Unity or Unity Hub, open the project folder called "12_IBM_Avatar_Receptionist_Unity_Project", making sure if opens in Unity version 2019.3.0f6
4. [Optional Step if you are having problems] The packages that our project requires to run should be installed with the project automatically. If you are having problems with missing packages, check that the following are indeed installed:
    1. From Window > Package Manager > AR Foundation 3.0.1 and ARCore XR 3.0.1 Plugins must be installed. See below for a complete list of packages installed in our project.



AR Foundation	3.0.1	✓
ARCore XR Plugin	3.0.1	✓
ARKit XR Plugin	3.0.1	✓
Input System	preview.6 - 1.0.0	✓
Rider Editor	1.1.4	✓
Test Framework	1.1.9	↓
TextMesh Pro	2.0.1	✓
Timeline	1.2.10	↓
Unity Collaborate	1.2.16	✓
Unity UI	1.0.0	✓
Visual Studio Code Editor	1.1.3	↓

AR Foundation

Version 3.0.1

Name

com.unity.xr.arfoundation

Links

View documentation
View changelog
View licenses

Author

Unity Technologies Inc.

Published Date

December 03, 2019

A collection of MonoBehaviours and C# utilities for working with AR Subsystems.

Includes:
GameObject menu items for creating an AR setup
MonoBehaviours that control AR session lifecycle and create GameObjects from detected, real-world trackable features
Scale handling
Face tracking

Samples available at

Last update Mar 24, 17:45

   2. NB sometimes we have had issues with the project building in other versions of Visual Studio Code. If you use VSCode as your code editor, make sure it is on Version 1.1.3 and no later
   3. In the Assets folder of the project make sure the unity-sdk and unity-sdk-core folders are present. These are the packages needed to connect unity to IBM watson. If not present for any reason, they can be downloaded here, although be aware they may have been changed since we built the project and errors can occur here. https://github.com/IBM/unity-sdk-core and https://github.com/watson-developer-cloud/unity-sdk

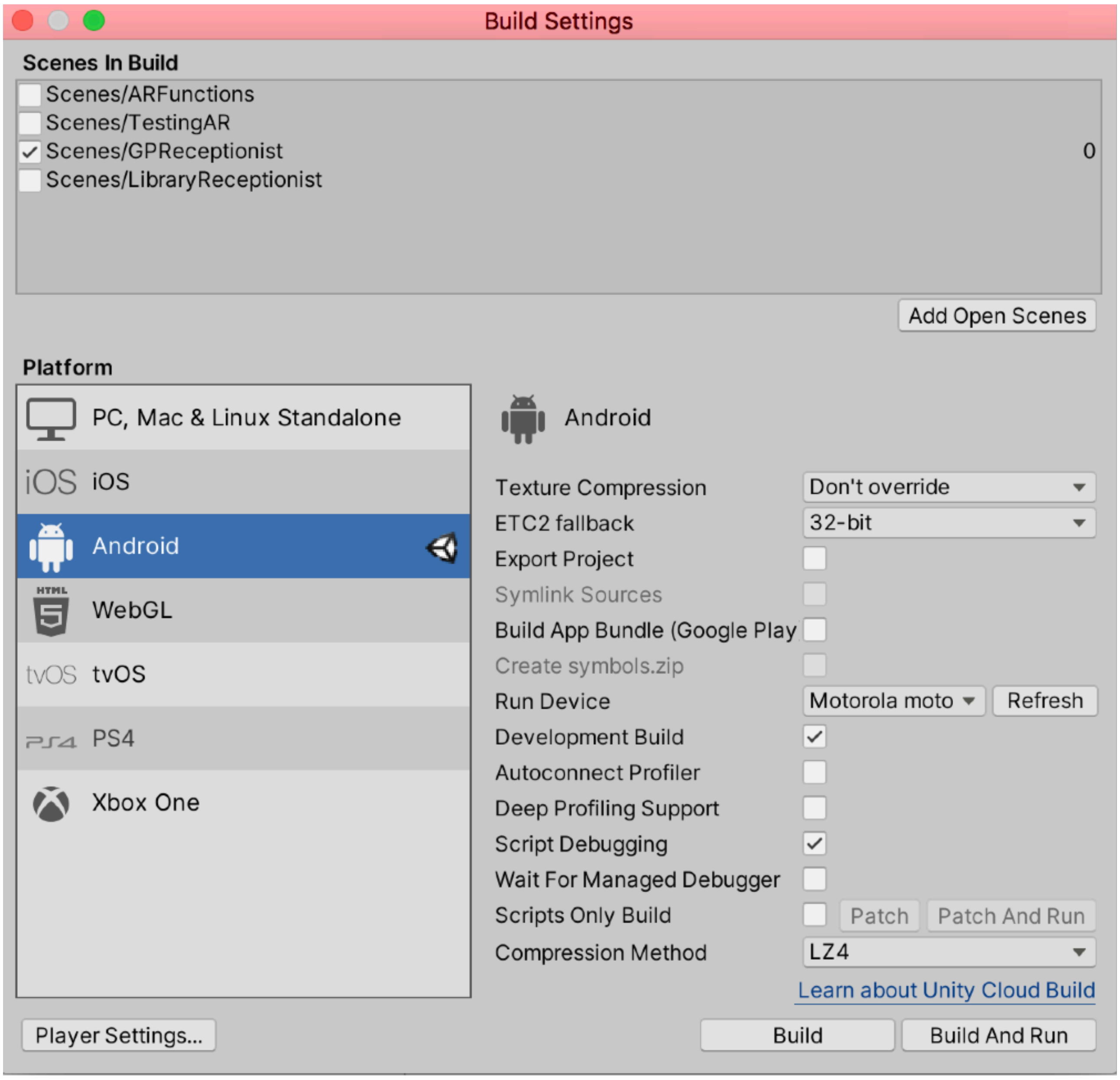
5. Go to File > Build Settings. Select Android, and click “switch platform”.
6. Click on Player settings in the bottom left hand corner. Select the Android tab. Select the arrow next to “Other Settings” to open that section. Check that in the Auto Graphics API list, Vulkan is not appearing, and if it is, delete it.
7. Continuing in player settings, scroll down to Minimum API Level and make sure it says Android 7.0 Nougat (API Level 24), and that the target API level is Automatic (Highest Installed)
8. Continuing in player settings, make sure the Scripting Backend is set to Mono and the API compatibility Level is set to .NET 4.x
9. [Optional] Continuing in player settings, if you wish to enable script debugging to see error messages in the console when running the app, go to Stack Trace and make sure all boxes are checked in the “Full” column
10. Finally, in player settings, open the XR Settings section and make sure ARCore supported is NOT checked.

Section 2.2 Building and Running the app

NB This section assumes a working knowledge of the Unity editor. For full details of the different Unity editor panels, please see https://docs.unity3d.com/Manual/UsingTheEditor.html

1. Return to the main unity editor. In the “Project” panel, Select assets > Scenes. Here you will find the scenes to build the different types of receptionist. Double click on the scene corresponding to the receptionist you wish to build. The scene will appear in the editor.
2. In the Hierarchy panel, select AR Session Origin. In the inspector panel, you should now see the AR Session Origin gameObject. In AR Tracked Image Manager (Script), there should be a gameObject appearing in the Tracked Image Prefab field. This is the avatar gameObject that will be instantiated when the camera recognises a tracked image. Double click on this avatar gameObject.
3. Now the avatar gameObject should appear in the Hierarchy and inspector panel. In the Inspector, scroll down to see the New Watson (script). Here you will see fields to enter the IDs and APIKeys to access the Watson services. If you are working from our private package these will be pre-populated, or we will have provided a separate document to you with our API keys. If you are working from the public GitHub repo these credentials have been removed for privacy. Please see section 3, setting up IBM Watson Cloud, to get your own credentials.
4. Enter the provided credentials in the appropriate field. NB for Assistant and Speech to Text, we have found that as the URL it is sufficient to use the base URL for the watson APIs i.e. Watson Assistant URL: https://gateway-lon.watsonplatform.net/assistant/api Watson Speech to Text URL: https://gateway-lon.watsonplatform.net/speech-to-text/api

5. Open File > Build Settings. Select your android device from the Run Device drop down menu. If you wish to enable script debugging to see error messages in the console, select the checkboxes next to Development Build and next to Script Debugging. Our build settings are shown below.



Build Settings

Scenes In Build

- [ ] Scenes/ARFunctions
- [ ] Scenes/TestingAR
- [x] Scenes/GPReceptionist 0
- [ ] Scenes/LibraryReceptionist

Platform: Android

Texture Compression	Don't override
ETC2 fallback	32-bit
Export Project	☐
Symlink Sources	☐
Build App Bundle (Google Play	☐
Create symbols.zip	☐
Run Device	Motorola moto
Development Build	✓
Autoconnect Profiler	☐
Deep Profiling Support	☐
Script Debugging	✓
Wait For Managed Debugger	☐
Scripts Only Build	☐
Compression Method	LZ4

6. From the Scenes in Build menu, select the scene that you want to build. If it does not appear, select Add Open Scenes.
7. Press build and Run. The app should run on your mobile device. Any build errors will show in the console.
8. If you wish to see script debugging, In the Console panel in the Unity Editor select the device you are using from the “Editor” drop down menu.

Section 3. IBM Watson Cloud

The below explains how to use IBM Watson Cloud to edit the backend of our avatar.

Section 3.1 Setting up Watson Cloud using our provided account

1. Go to https://cloud.ibm.com/login and login with the credentials provided to you in the private documentation)
2. From the Dashboard, click on Services
3. From the Services section of the Resource List, click on Watson Assistant Test. Check that the status column shows a green circle saying “Active”. [If it is not active, this may be because the account has been deactivated or has run out of credit. Contact IBM in this case]
4. On this page, you can see the Credentials at the bottom of the page. Click on “show credentials” to reveal the API key. This is what needs to be entered into the Assistant API Key field of the avatar in the Unity editor. (see, section 2.2 steps 3 and 4)
5. Click on “Launch Watson Assistant”
6. You should see the list of assistants associated with this service instance, including GPAssistant, MuseumLibraryAssistant and OfficeReceptionistAssistant. Click on the assistant you would like to use.
7. Under Dialog, click on the skill box to launch the dialog skill.
8. You are now in the Assistant Dialog Editor. Here you can edited the recognised Intents, Entities and Dialog nodes of the assistant. For an explainer of these terms and how assistant works, please see https://developer.ibm.com/technologies/artificial-intelligence/articles/introduction-watson-assistant
9. Click on Try It in the top right hand corner to test the chatbot.

Section 3.2 Setting up Watson Cloud without our provided account

If you do not have access to our IBM account, you can connect the chatbot to your own IBM Watson backend.

1. Create an IBM Cloud account. https://cloud.ibm.com/registration
2. We recommend following these IBM tutorials to set up an instance of IBM Watson Assistant in the IBM Watson Cloud. Once you have done so and got to grips with the interface (including what Intents, Entities and Dialog nodes are) you can move on to the next step. https://developer.ibm.com/tutorials/create-your-first-assistant-powered-chatbot/
3. Go to the Watson Assistant homepage, and click, Create Assistant. Name your assistant and click create.
4. Click on the Add Dialog Skill button. Select the Import Skill tab.
5. Upload the provided JSON file for the type of assistant that you would like to build. Click Import
6. You will now see the skill in the Dialog section of your chatbot. Click on the skill to edit it.
7. You can now edit the recognised Intents, Entities and Dialog to adapt the chatbot to your organisation. You might, for example, like to change the text in the “Help” or “FirstMessage” dialog nodes to say more specific information to your organisation.
8. You will probably want to change the “Entities” to be more specific to your organisation - for example, if you are using the GP chatbot, you will want to add the names of your doctors here. You can import entities using csv files.
9. Go to the “Options” tab and click on “Webhooks” to see the settings for your webhook. If you want to link with our database backend, you will need to set up your Azure Function webhook and then insert the details for it here, so that Watson Assistant can interface with it.
10. To get the credentials for your chatbot to insert into the avatar credentials slots in Unity, return to the assistant homepage. In the top right hand corner of the assistant that you wish to connect to, click on the three dots. Then click Settings. In Settings, click the API details tab. Here you will see the AssistantID and Api Key fields which you will need to insert into the avatar script in Unity [See section 2 above].

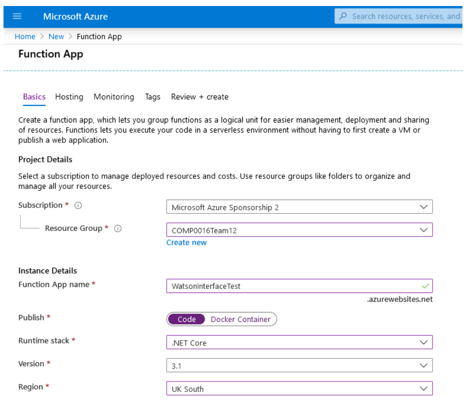
Section 4. Azure Cloud

4.1 Setting up a Function App

1. Go to https://portal.azure.com/ , create an account and log in

2. In Azure services, click create a resource, and on the next screen click Function App

3. Choose a subscription to use and resource group. These may need to be created if they aren't pre-existing. Give the Function App a name, and choose 'Code', '.NET Core' and 3.1 for Publish, Runtime Stack and Version, and select the appropriate region.



Function App

Basics | Hosting | Monitoring | Tags | Review + create

Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.

Project Details

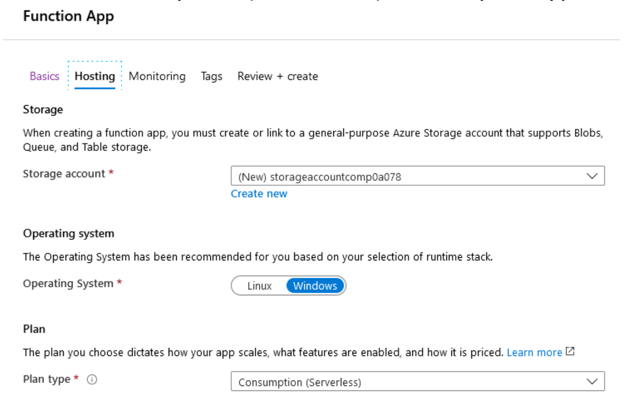
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Microsoft Azure Sponsorship 2
Resource Group *	COMP0016Team12
	Create new

Instance Details

Function App name *	WatsonInterfaceTest .azurewebsites.net
Publish *	[✓ Code] Docker Container
Runtime stack *	.NET Core
Version *	3.1
Region *	UK South

4. On the next page, choose or create a new storage account and select Windows as the OS. We chose Consumption (Serverless) for the plan type.


Function App

Basics | Hosting | Monitoring | Tags | Review + create

Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage.

Storage account *	(New) storageaccountcomp0a078
	Create new

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

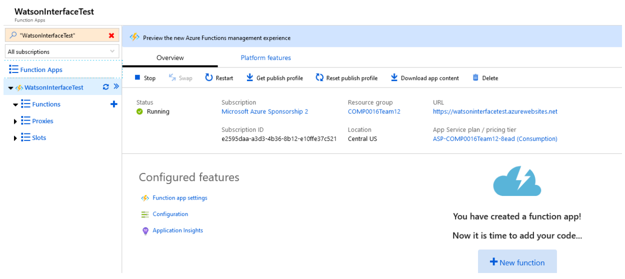
Operating System *	Linux [✓ Windows]

Plan

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. Learn more

Plan type *	Consumption (Serverless)

5.The monitoring and tags pages can be left with their default values. Go to review, and click create, and wait for the deployment to finish.



WatsonInterfaceTest
Function Apps

Status	Subscription	Resource group	URL
Running	Microsoft Azure Sponsorship 2	COMP0016Team12	https://watsoninterfacetest.azurewebsites.net
	Subscription ID	Location	App Service plan / pricing tier
	e2595daa-a3d3-4b36-8b12-e10ffe37c521	Central US	ASP-COMP0016Team12-8ead (Consumption)

Configured features

You have created a function app!

Now it is time to add your code...

6. Once deployment has finished; the function app has been created and it will prompt to add a new function. Instead, navigate to https://<your function app name>.scm.azurewebsites.net/ZipDeployUI and drag the provided zip into the file explorer on the page. This will take a minute or so, and then

say Deployment successful below. More detailed instructions on this step can be found here: https://docs.microsoft.com/en-us/azure/app-service/deploy-zip

7. Navigate back to the function app page, and refresh if you kept the page open, and after a moment the HttpTrigger1 will appear in the functions section, along with all the original code.

8. Open the HttpTrigger1 function, and open the console (at the left and bottom of the screen respectively). Enter the commands ‘dotnet add package Twilio’, ‘dotnet restore’ and ‘dotnet build’, and when that’s finished stop and start the function.

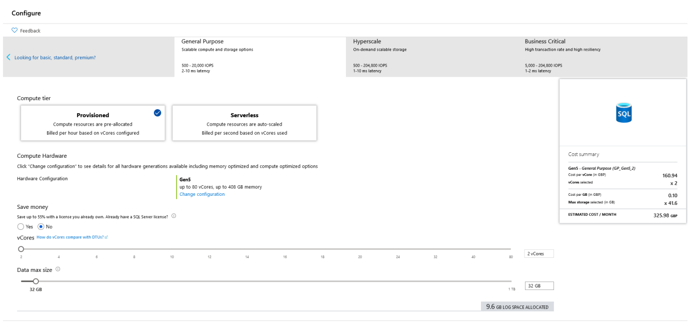
9. Finally, the Application settings need to be added. To do this, go to the main page for the function app and click ‘configuration’. Click ‘advanced edit’ and paste in the contents of applicationsettings.txt after the first ‘{‘.

N.B. when trying to set it up from scratch, there were occasionally errors such as issues loading a .dll, this never occurred on our main project and it is unclear how to resolve it.

4.2 Setting up an SQL database

1. As with the function app, go to the portal home page, select create a resource and this time choose SQL database

2. Choose the subscription, resource group and a name. Configure the compute + storage to suit needs and budget


Configure
Feedback
Looking for basic, standard, premium?
SQL
Change configuration
How do vCores compare with DTUs?

General Purpose
Scalable compute and storage options
500 - 20,000 IOPS
2-10 ms latency

Hyperscale
On-demand scalable storage
500 - 204,800 IOPS
1-10 ms latency

Business Critical
High transaction rate and high resiliency
5,000 - 204,800 IOPS
1-2 ms latency

Compute tier

[✓ Provisioned]
Compute resources are pre-allocated
Billed per hour based on vCores configured

Serverless
Compute resources are auto-scaled
Billed per second based on vCores used

Compute Hardware
Click "Change configuration" to see details for all hardware generations available including memory optimized and compute optimized options

Hardware Configuration: Gen5
up to 80 vCores, up to 408 GB memory

Save money
Save up to 55% with a license you already own. Already have a SQL Server license?
Yes [✓ No]

vCores: 2 vCores

Data max size: 32 GB

9.6 GB LOG SPACE ALLOCATED

Cost summary
Gen5 - General Purpose (GP_Gen5_2)
Cost per vCore (in GBP) 160.94
vCores selected x 2
Cost per GB (in GBP) 0.10
Max storage selected (in GB) x 41.6
ESTIMATED COST / MONTH 325.98 GBP

3. The data source can be left as ‘none’ or an existing backup, if one exists. After this, the form can be reviewed and completed, and the SQL database is set up.

4. Navigate to the query editor and log in, the first time it will link to the firewall settings to add the client IP address

5. Now the tables in the database can be created; if recreating the example database, there are 11 tables with columns as shown in the Entity Relationship diagram, which can be created with queries of the following format:

CREATE TABLE gp_prescriptions(

PrescriptionID int,

PatientID int,

DoctorID int,

MedicineName Varchar(255),

HasBeenDelivered bit,

ExpectedDate DateTime

);

6. On the menu down the left of the screen, scroll down to the settings section and click on ‘Connection strings’. Copy the text in the box, and then in the configuration page for the function app edit the sqldb_connection value to be the connection string for the new SQL database.

7. The database can be populated with data from the query editor using SQL queries or the edit data feature (currently in preview), which can be selected after clicking the 3 dots next to the table name

4.3 Adapting the webhook to a new business. There are 3 main levels at which the webhook can be adapted:

1. The simplest way is to have different data in the SQL database. This can be achieved simply through SQL queries, or the graphical user interface available on Azure, to create records with information personalised for the new business
2. Secondly, the application settings can be altered. Having set sqldb_connection to the connection string of the new database, the Twilio* authToken and accountSid can be entered, along with a default number to text if human assistance is needed. After this, the request_types can be replaced with the table names being used for each request type, and the valid key names strings can be edited to include different or new column names which can be selected or compared to.

   *To get the Twilio credentials, create an account here: https://www.twilio.com/try-twilio . It will also be necessary to change the from number in the code at line 164 if the number for your account is different.

3. Finally, with access to the code the existing request types can be edited, and new request types can be added. They generally follow the format of getting the valid key names from one of the application variables, making a query with makeQuery(false, “”) if using all of the keys from the HTTP request, and returning the result of databaseRequest with the query string as a parameter. There are variations on the query making method, including more specific ones like allEventsOnDay() and OrganisationInfoQuery(), and SetQuery() for updating the table.