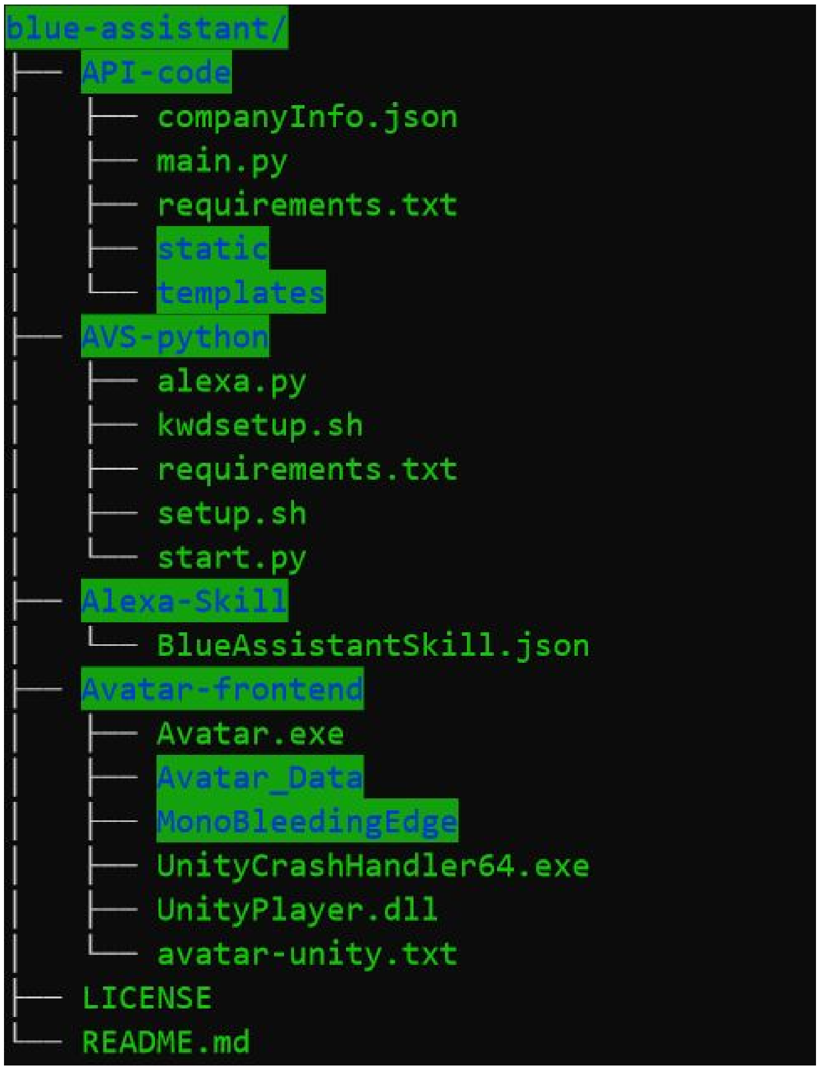
Blue assistant deployment manual

Pre-requisites

- IoT devices that are connected to a WiFi network.
- A Windows laptop/PC with a stable internet connection on the same WiFI network as the IoT devices'
- A webcam - preferably as high quality and resolution as possible
- A good quality microphone
- Speakers for audio playback
- An Amazon account
- A subscription to a cloud VM service such as Azure, or access to a server than can be used on the cloud

The Package



```
blue-assistant/
├── API-code
│   ├── companyInfo.json
│   ├── main.py
│   ├── requirements.txt
│   ├── static
│   └── templates
├── AVS-python
│   ├── alexa.py
│   ├── kwdsetup.sh
│   ├── requirements.txt
│   ├── setup.sh
│   └── start.py
├── Alexa-Skill
│   └── BlueAssistantSkill.json
├── Avatar-frontend
│   ├── Avatar.exe
│   ├── Avatar_Data
│   ├── MonoBleedingEdge
│   ├── UnityCrashHandler64.exe
│   ├── UnityPlayer.dll
│   └── avatar-unity.txt
├── LICENSE
└── README.md
```

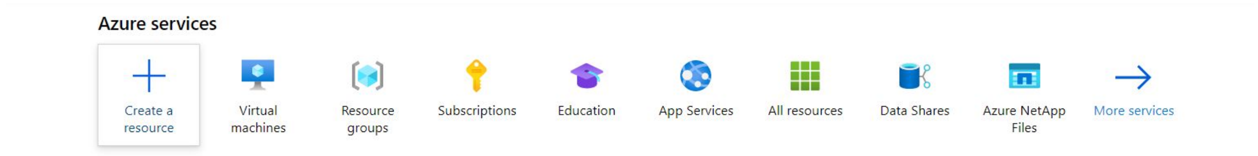
The contents API-code is the only code used outside of the local device, so this should be transferred to the VM when it is set up.
The AVS-python directory is used on the Ubuntu VM that will be run locally - it cannot be run directly on Windows.
The Alexa-Skill directory contains the JSON file needed to easily set up the Alexa Skill correctly on your account.
The Avatar-frontend directory contains the built 3D model files that can be run directly on Windows. The avatar-unity txt file in this directory contains a link where you can download the entire Unity project in a zip file (Avatar.zip) and build it yourself with a modified endpoint.

Deploying API to server (Azure example)

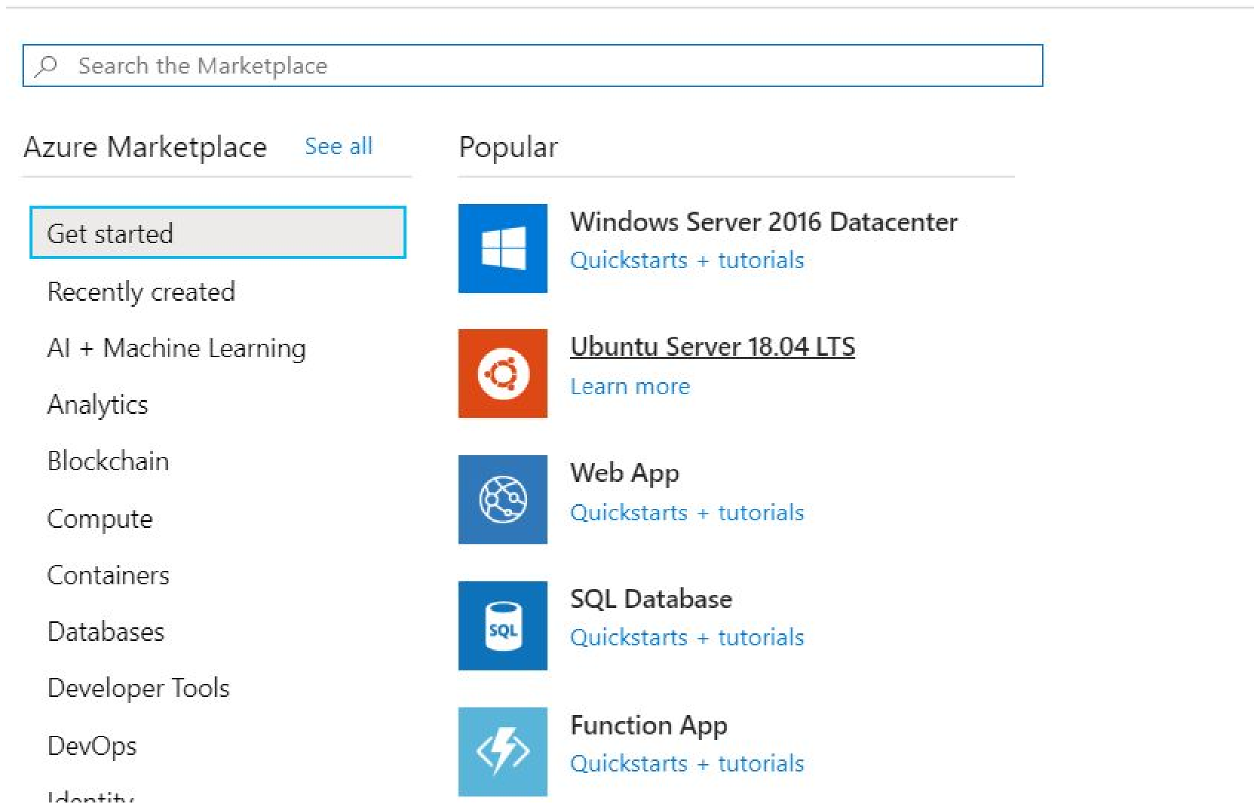
This can be done on any cloud VM service or on a server that is already available to you.

Setting up the Azure Ubuntu VM

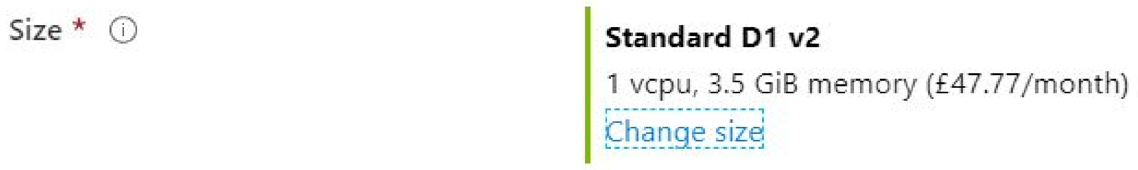
Log into portal.azure.com and click create a resource



Create a new Ubuntu 18.04 LTS Server



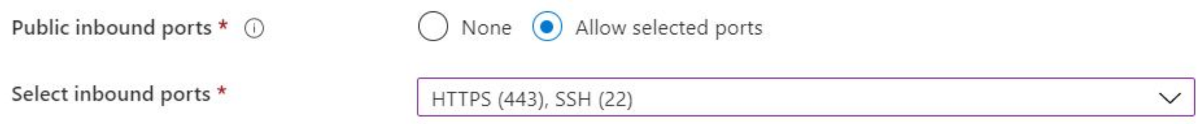
Change the settings according to your preference of naming. The image must be Ubuntu 18.04 LTS and the size must be at least the following:



Size * Standard D1 v2

1 vcpu, 3.5 GiB memory (£47.77/month)

Choose your preferred means of accessing the VM and make sure that SSH and HTTPS are selected for inbound ports



Public inbound ports * ⓘ ( ) None [✓ Allow selected ports]

Select inbound ports * HTTPS (443), SSH (22)

The rest of the options can be kept to default or can be changed to your preference

SSH into the server.

Set up letsencrypt (HTTPS API)

You will need to reserve a domain on a domain name service for the API to work as intended. Currently the API is hosted on https://alexaapi.compositegrid.com but this will not stay up for much longer.

Follow the instructions for setting up certbot and it's dependencies on this website https://certbot.eff.org/lets-encrypt/ubuntubionic-other

On step 4, use

```
sudo certbot certonly --standalone
```

Input your contact email and, agree with the ToS and then enter the domain name that has been reserved. Key files will automatically be made after the handshake with the domain inputted.

In the user's home directory, create a directory keys

```
mkdir keys
```

Copy the key files from letsencrypt for your {domain} to this directory:

```
sudo cp -r /etc/letsencrypt/archive/{domain}/ keys/
```

Go to the copied files

```
cd keys/{domain}
```

Change the permissions on these keyfiles using:

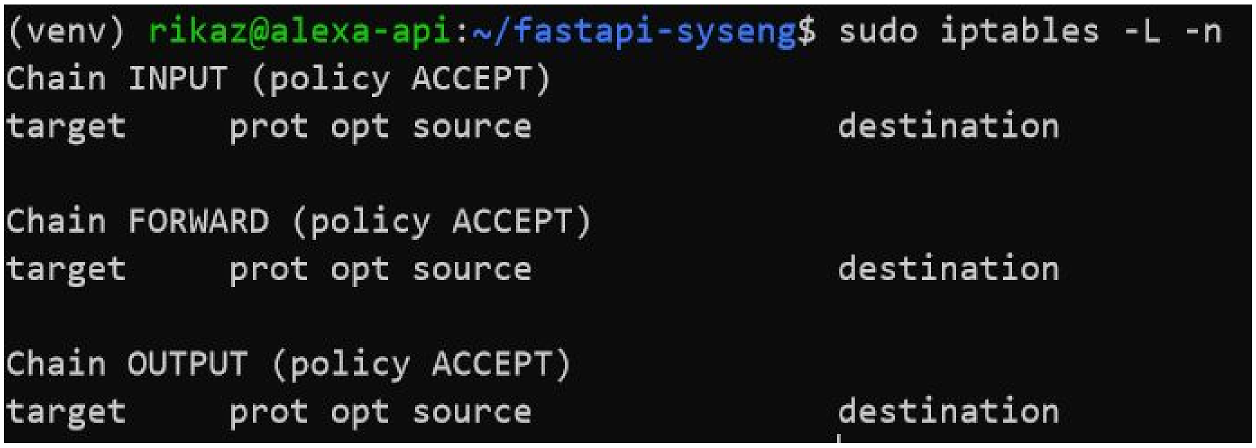
```
sudo chown $USER *
sudo chmod 755 *
sudo chgrp $USER *
```

Setup IP port forwarding for HTTPS

We will need to port forward any incoming traffic from the HTTPS port (443) to the server port open on the VM. Any arbitrary non-reserved port can be used but for this example we will use port 4430.
Run

```
sudo iptables -L -n
```

You should see the following output



```
(venv) rikaz@alexa-api:~/fastapi-syseng$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Now run

```
sudo iptables -I INPUT 1 -p tcp --dport 443 -j ACCEPT
```

This will allow traffic through port 443

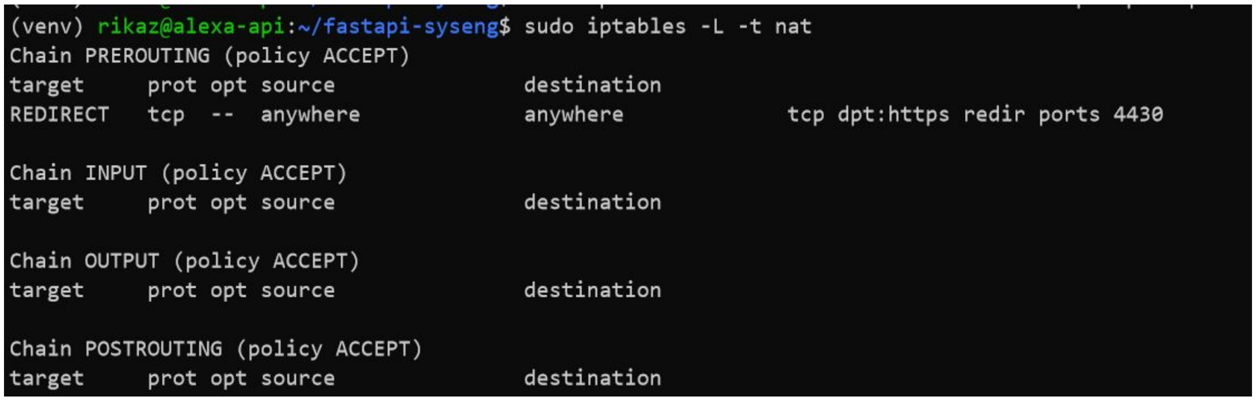
Now we must forward port 443 to our server port on the VM (4430)

```
sudo iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 443 -j
REDIRECT --to-port 4430
```

Running

```
sudo iptables -L -t nat
```

should give the following output



```
(venv) rikaz@alexa-api:~/fastapi-syseng$ sudo iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             tcp dpt:https redir ports 4430

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
```

To make these changes persistent between restarts of the iptables service we need to do the following:

```
sudo sh -c "iptables-save > /etc/iptables.rules"
sudo apt-get install iptables-persistent
```

Then save the current IPv4 and IPv6 rules

Running server code

Add the API-code directory to the root directory of the VM and go to this directory

```
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-venv
```

It is highly recommended to install screen at this point (if not installed already)

```
sudo apt install screen
```

Create a new screen

```
screen -R
```

Make a new venv in the API server code

```
python3 -m venv venv
```

Enter the venv

```
source venv/bin/activate
```

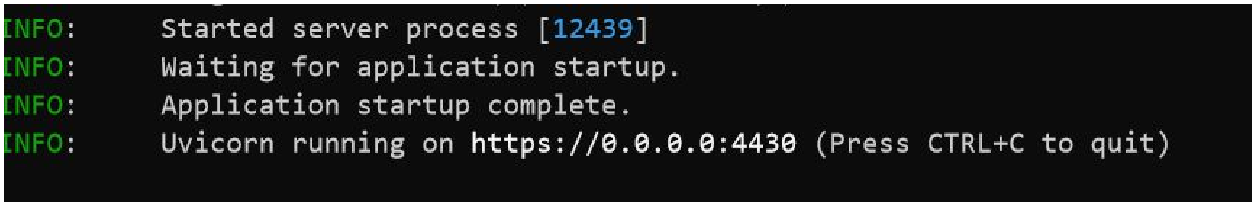
Install python dependencies

```
python3 -m pip install -r requirements.txt to
```

Run uvicorn

```
uvicorn main:app --host 0.0.0.0 --port 4430
--ssl-keyfile=$HOME/keys/{domain}/privkey1.pem
--ssl-certfile=$HOME/keys/{domain}/fullchain1.pem
```

This will run the server on your designated domain name with https enabled



```
INFO:     Started server process [12439]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on https://0.0.0.0:4430 (Press CTRL+C to quit)
```

Setting up Ubuntu VM on local machine for AVS

An Ubuntu VM will be required to run the Alexa Voice Service code on Windows. This is very easy to setup using the VMware Workstation Player.

Setting up Ubuntu VM

Download the Ubuntu 18.04 LTS iso from the Ubuntu website
https://ubuntu.com/download/desktop
Download and install the VMWare Workstation Player for Windows
https://www.vmware.com/uk/products/workstation-player/workstation-player-evaluation.html
On the Workstation application, click Player -> File -> New Virtual Machine
Under Installer disc image file (iso) select the Ubuntu 18.04 iso that was downloaded. This should allow Easy Install of the VM.
Then add the user credentials for the OS, choose a name for the VM and reserve 20GB for the VM on your system. Store the virtual disk as a single file. Click finish, and the Ubuntu VM will be automatically set up for you.

Setting up and using the app

Transfer the AVS-python directory to your Ubuntu VM
Go to the directory

```
cd AVS-python
```

Update your apt package indices

```
sudo apt update
```

Install pip and virtual environments. Then create a virtual environment called venv

```
sudo apt install python3-pip python3-venv
python3 -m venv venv
```

Activate venv

```
source venv/bin/activate
```

Make setup.sh executable

```
sudo chmod +x setup.sh
```

Run the setup script

```
./setup.sh
```

You will be redirected to a webpage - click on amazon alexa, sign into the alexa dev account - (right now it is nttdata.group24@gmail.com, password is jorik123!) and then allow the application access.

Make kwdsetup.sh executable

```
sudo chmod +x kwdsetup.sh
```

Run the keyword setup script

```
./kwdsetup.sh
```

Open the alexa.py file in a text/code editor and change the ENDPOINT variable on line 40 to the address of the API endpoint that has been setup, followed by '/api/v1/speechLogs?text=' like so

```
ENDPOINT = "https://{DOMAIN}/api/v1/speechLogs?text="
```

Run the application code

```
python3 start.py
```

You can test the application without a GUI by saying 'Alexa' and asking a question. You should see the keyword being detected on the terminal and the application playing the audio files that are being outputted by the speakers.

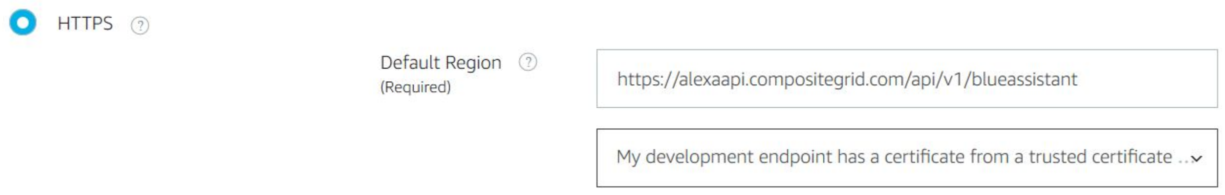
Setting up the Alexa skill

Creating the skill

Navigate to https://developer.amazon.com/alexa/console/ask and sign in to the alexa account used when setting up Alexa on the Ubuntu VM.

Click create skill and enter in a name for the skill, use a custom model and provision your own method to host the skill's backend resources. Start the skill from scratch.

Go to the JSON editor then drag and drop the BlueAssistantSkill JSON file into here Navigate to Endpoint, select HTTPS and add the API domain into the Default Region field followed by /api/v1/blueassistant. Select "My development endpoint has a certificate from a trusted certificate authority" as the SSL certificate type

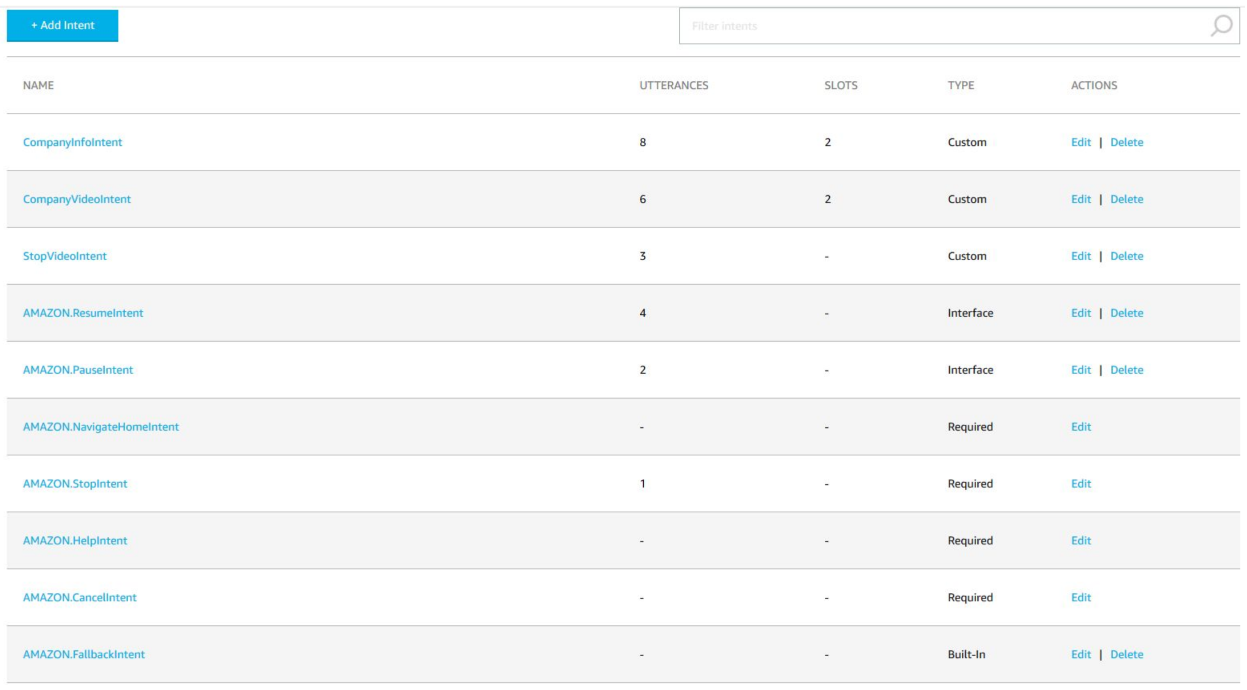


HTTPS

Default Region (Required): https://alexaapi.compositegrid.com/api/v1/blueassistant

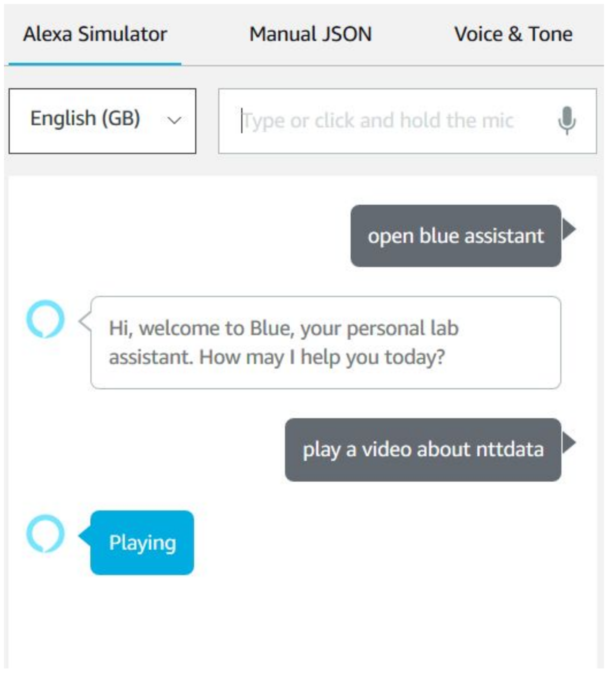
My development endpoint has a certificate from a trusted certificate ...

Click save endpoints and navigate to intents. The page should look like this



NAME	UTTERANCES	SLOTS	TYPE	ACTIONS
CompanyInfoIntent	8	2	Custom	Edit | Delete
CompanyVideoIntent	6	2	Custom	Edit | Delete
StopVideoIntent	3	-	Custom	Edit | Delete
AMAZON.ResumeIntent	4	-	Interface	Edit | Delete
AMAZON.PauseIntent	2	-	Interface	Edit | Delete
AMAZON.NavigateHomeIntent	-	-	Required	Edit
AMAZON.StopIntent	1	-	Required	Edit
AMAZON.HelpIntent	-	-	Required	Edit
AMAZON.CancelIntent	-	-	Required	Edit
AMAZON.FallbackIntent	-	-	Built-In	Edit | Delete

Click build model to start building the Alexa skill model. The skill can be tested by going to the Test tab, enabling testing on Development and then typing utterances. The skill can be tested like so. If the responses are the same then the API endpoint is working as intended



open blue assistant

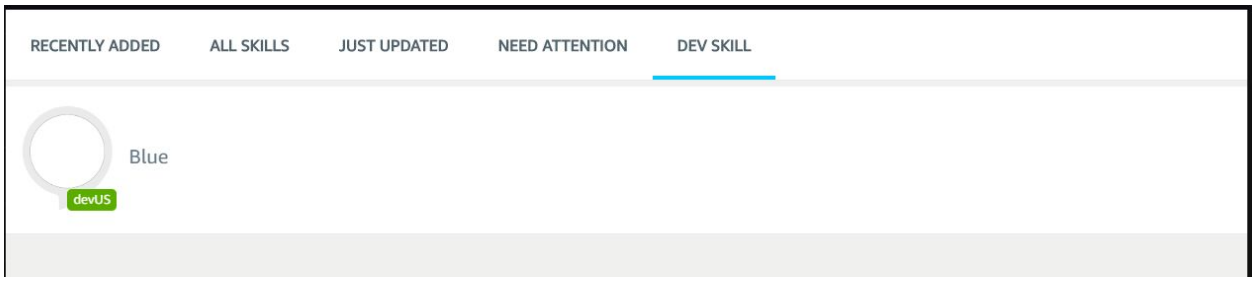
Hi, welcome to Blue, your personal lab assistant. How may I help you today?

play a video about nttdata

Playing

Add Skill to your Amazon account

Go to https://alexa.amazon.co.uk/spa/index.html and sign in using the Amazon account used for the skill.
On the left tab, head to skills, then click on 'Your Skills' on the top right.
Under Dev Skills, the skill that has been created should be present



Click on this skill and enable it.
This should allow any Alexa device connected to your account to be able to use this skill

Alternatively this can be done using the Alexa app. On the left pane, click Skills and Games, navigate to 'Your Skills' and the skill should be present under 'Dev'.

IoT device setup example - Philips hue

1. Setup the Hue bridge and lamps according to the instructions from Philips. Make sure you have them on the same network as your phone and laptop.
2. Open the alexa app and enable the philips hue skill
3. Sign into your philips hue account and allow alexa access.
4. Then follow the instructions on screen to find hue devices on the network

Now that your IoT device is connected through the Alexa app it should be controllable through the application

Setting up the Unity avatar

In Unity, File -> Open Project and select the unzipped Avatar.zip folder

To escape compatibility issues, the project runs on Unity Version 2019.2.18f1

Setting API endpoints

There are 2 endpoints that need to be configured.
In scripts/player.cs line 40, change the websocket endpoint to be

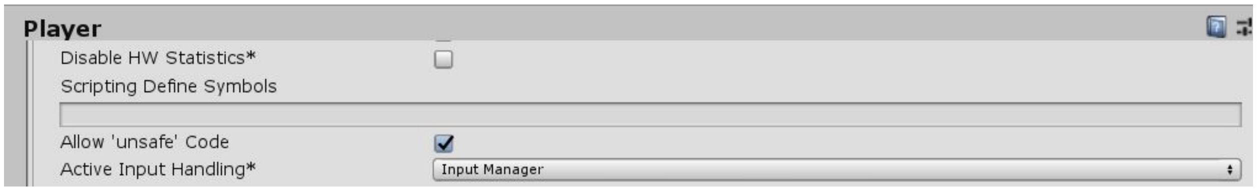
```
websocket = new WebSocket("wss://{YOUR_ENDPOINT_WITHOUT_HTTPS}/ws");
```

In script/videoPlayerScript.cs line 53, change the video player URL

```
videoPlayer.url = "{YOUR_ENDPOINT}/companyVideo";
```

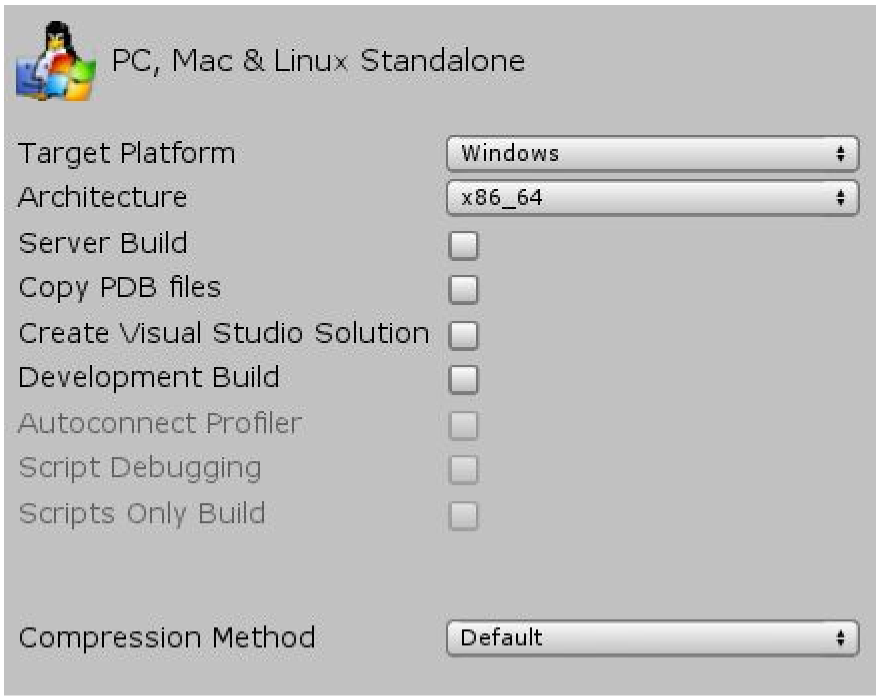
Building the Unity Project

File -> Build Settings -> Player Settings -> Other settings and allow 'unsafe' code under configuration



Player
Disable HW Statistics* ☐
Scripting Define Symbols
Allow 'unsafe' Code ☑
Active Input Handling* Input Manager

Set your build settings to be like this



PC, Mac & Linux Standalone

Target Platform	Windows
Architecture	x86_64
Server Build	☐
Copy PDB files	☐
Create Visual Studio Solution	☐
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐
Compression Method	Default

Build the project and run the .exe file.

Lip-sync concept app

In addition to the base package that is specified above, we have also included a concept application for retrieving the response text from the user's Amazon account. This application uses the Selenium Chrome driver to get the relevant cookies and authentication to be able to access Amazon accounts and retrieve Alexa responses.
The prerequisites for this application is a Windows laptop with python3 installed, as well as pip and python3 venv.
To run this app use first create a virtual environment in windows

```
python -m venv venv
```

Activate the venv using

```
.\venv\Scripts\activate
```

Then install the pip requirements

```
python -m pip install -r requirements.txt
```

Then run the application using

```
python apihandler.py
```

This will open a browser window to login to your Amazon account. Login and then head back to your terminal and press enter
Now when Alexa responds to a question, it will be printed to the terminal. This could be used to send to the API and communicate to the Avatar for lip-sync functionality.