

Cotiss Honest Feedback Form Technical Document

Product Overview

This product is a website that allows users to provide honest feedback about the business through a digital form for the business, Cotiss, to receive, supported by using resources from Microsoft Azure. Cotiss has had a growing need to collect feedback internally and externally about their business, and wanting to encourage their employees to share their honest and anonymous feedback. This product is able to support these needs, hosting a website where the feedback entered by employees is sent to a database where this information can be accessed and analysed by the business, while maintaining the anonymity of the employees.

Cotiss Feedback Form Project Demo: <https://www.youtube.com/watch?v=m7eldt8uSzk>

Cotiss Feedback Form Design: <https://github.com/santinogaeta/feedback-form>

Product Objectives

This product hosts a website that sends feedback information entered into a feedback box to a database to be accessed by the business. Once the website has been deployed, employees will be able to access and use the website by entering their feedback into the text-box and submitting their entry using the “Submit Feedback” button. This Technical Documentation will inform the business how to set up this product, and how to access the feedback from their employees, aiding the business’s need for receiving honest feedback.

Table of Contents

-
- Product Overview

 - Product Objectives

 - Table of Contents

 - Setting up Virtual Machine

 - Azure Function App

 - Hosting Website on Virtual Machine via Apache

 - Auto-scaling with Load Balancer and Scale Set

 - Azure Cosmos Database
-

Setting up Virtual Machine

Virtual Machine Overview

It is important to establish a virtual machine since this will be the resource that will host the feedback form website. This virtual machine will be created with the virtualised infrastructure of an Ubuntu operating system.

Virtual Machine Objectives

The Ubuntu infrastructure gives the ability to install the free open source web server, Apache. Furthermore, this specific virtual machine allows snapshots to be captured, to then be used by a scale set as a template for the creation of multiple instances of the exact same kind.

Technical Explanation

1. Select **Create a resource** from the Azure portal menu or **Home page**.
2. Select **Create** under Virtual machine, or search “Virtual machine” in the **Marketplace** and select proceed with **Create** (Figure 1).
3. In the **Create a virtual machine** page, make sure the **Image** is an **Ubuntu Server 20.04 LTS - Gen2**. Enter basic settings for everything else, as seen in Figure 2.
4. Leave everything else default and select **Review + create**, followed by the **Create** button at the bottom of the Validation page.
5. Once portal page displays **Your deployment is complete**, the virtual machine is ready for use.

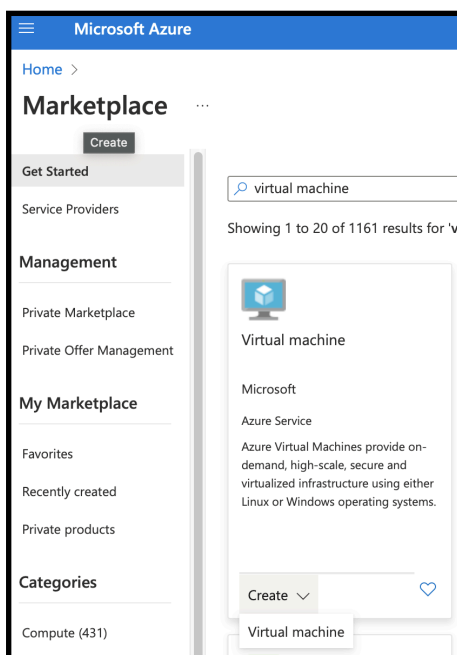


Figure 1: Searching and selecting to create a virtual machine from the Azure Marketplace.

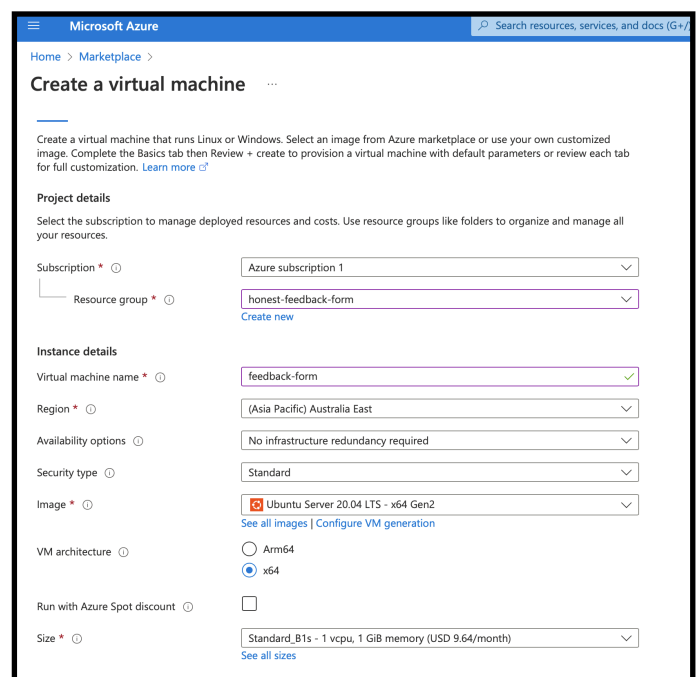


Figure 2: Basic settings for creating Virtual machine with Ubuntu Server Gen2 image.

Azure Function App

Azure Function App Overview

Azure Function Apps are event-driven pieces of logic that can be connected to an Azure Cosmos Database account. These functions can be used repeatedly for an unlimited number of times, and they are serverless, which requires no infrastructure management or any resources to be consumed in order to run.

Azure Function App Objectives

Establishing an Azure Function App now will help later when the Azure Cosmos Database account is created and needs to communicate with the website. By including the Azure Function App's function URL in our webpage's HTML code, the Azure Function App will trigger once feedback is ready to be transferred from the website to the Database.

Technical Explanation (1/2)

1. Select **Create a resource** from the Azure portal menu or **Home page**.
2. Search "function" in the **Marketplace** and proceed to select **Create > Function App** (Figure 1).
3. Enter basic settings for **Function App name** and **Region**.
4. For **Runtime stack** choose **.NET** and leave the default option for **Version**.
5. Leave all other options as their default and press **Review + Create**.
6. After reviewing settings on **Validation Page** press the **Create** button and wait for the portal page to display **Your deployment is complete** before pressing **Go to resource**.

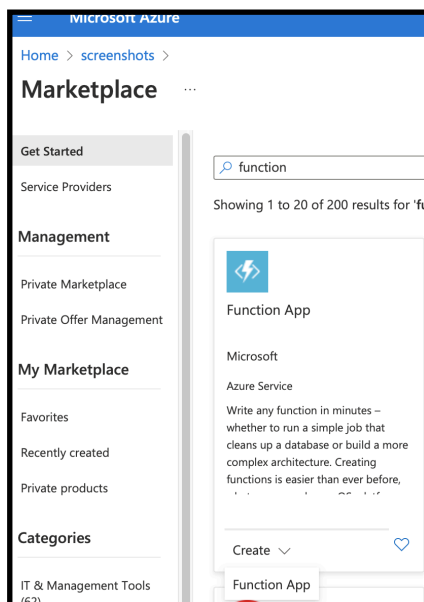


Figure 3: Searching and selecting to create a Function App from the Azure Marketplace.

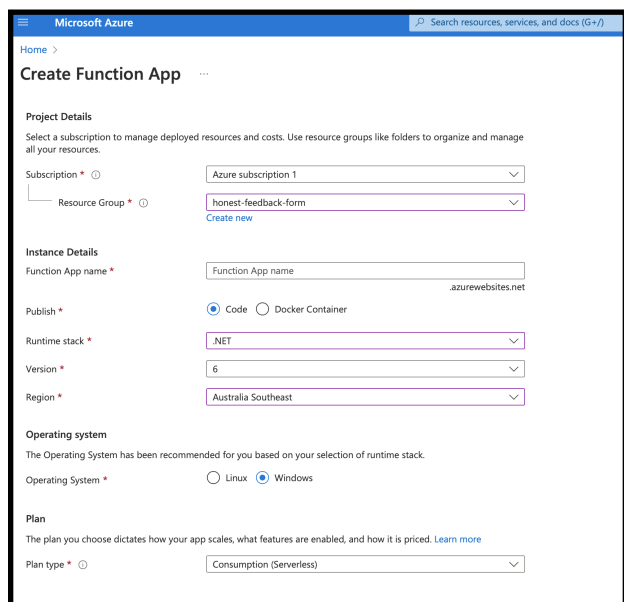


Figure 4: Basic settings and .NET selected Runtime stack for creating Function App.

Azure Function App

Technical Explanation Continued (2/2)

- Once at the Azure function App, click on the **Functions** tab and click **Create** (as seen on *Figure 5*).
- In the **Create function** window (*Figure 6*) select **HTTP Trigger**, name the function and change **Authorisation level** to **Anonymous**.

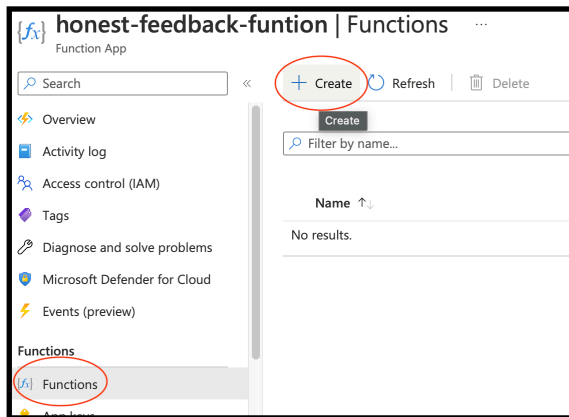


Figure 5: Clicking the Functions Tab and creating a new Function for the Azure Function App.

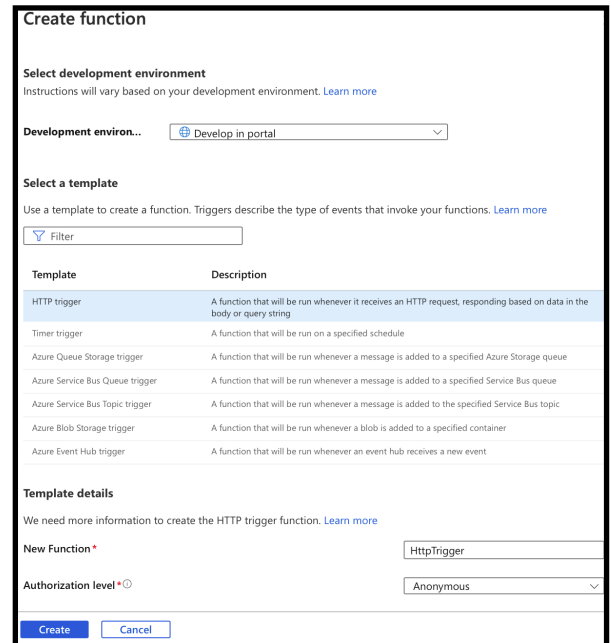


Figure 6: Creating new HTTP Trigger in Create Function window.

- Enter the newly created HTTP Trigger and click on the **Integration** under the **Developer** tab to access the **HTTP (req)**, as seen in *Figure 7*.
- Clicking on **HTTP (req)** will open the **Edit Trigger** window (*Figure 8*).
- Set **Authorisation level** is set to **Anonymous**, and selected **HTTP methods** is only **POST**.

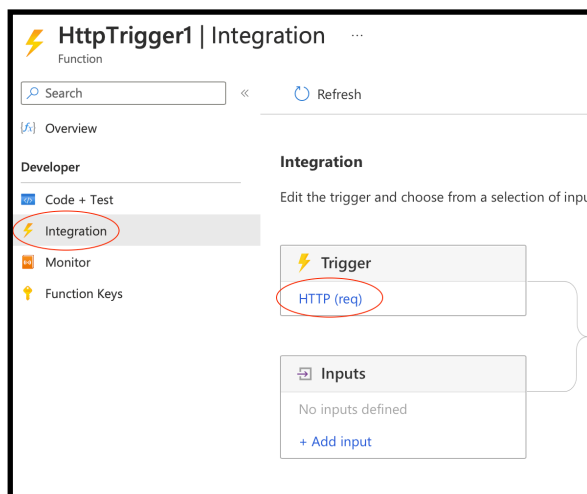


Figure 7: Integration section of the HTTP Trigger

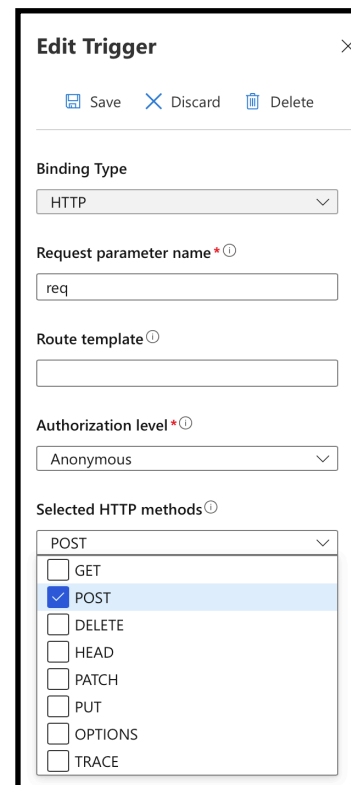


Figure 8: Settings for HTTP Trigger under the Edit Trigger window.

Hosting Website on Virtual Machine via Apache

Hosting Website Overview

Apache is a free open source web server for Linux servers, which will now be installed into our Ubuntu virtual machine that was created previously. By accessing Apache and replacing the *index.html*, that is already loaded onto the web server, with code that will load the Feedback form webpage whenever the virtual machine is loaded, the feedback form webpage is what will be displayed.

Hosting Website Objectives

With Apache installed on the virtual machine this provides the ability to add the feedback form webpage code to be loaded whenever the virtual machine is accessed. This means employees will be able to use the site once they have access to the virtual machines hosting the website. Within the website's code is also a URL for the Azure Function App's HTTP Trigger that was created previously. This will allow the website to send information when the Function is triggered, once the Azure Cosmos Database is created as well.

Technical Explanation (1/2)

1. Access the virtual machine created previously from the Azure portal menu or **Home page**, or the designated **Resource Group**.
2. Click the **Connect** drop down and select the **SSH** method of connecting to the virtual machine (Figure 9 - Top-right).
3. Using the **SSH key** downloaded when creating the virtual machine, follow the commands in *Figure 9* in a **Terminal** to access the virtual machine.

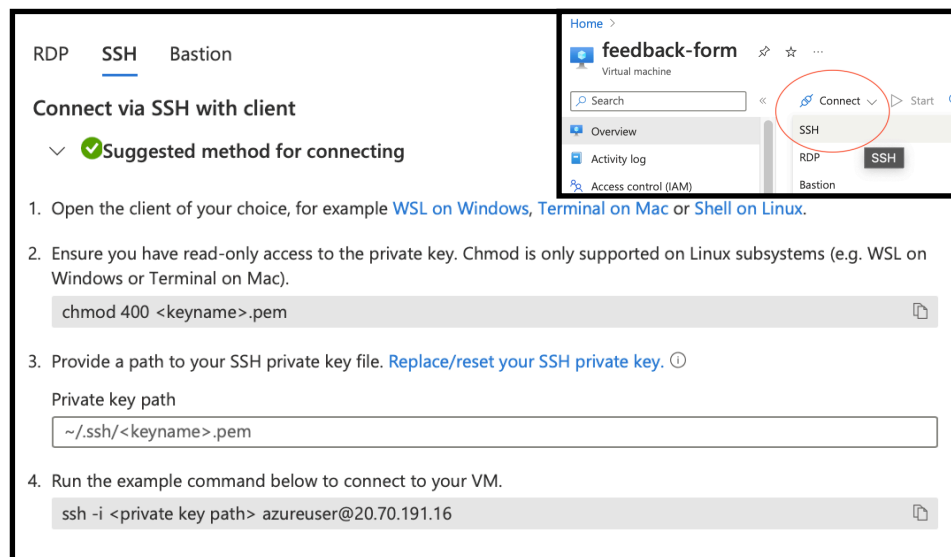


Figure 9: Steps to connecting in Terminal to the Virtual machine

4. Once in the virtual machine, run the commands ***sudo apt update***, followed by ***sudo apt install apache2***, where Apache will start to be installed onto the virtual machine.
5. Once Apache has finished installing, run the commands ***cd /var/www/***. Listing the contents in the directory with ***ls*** will be a file ***index.html***.
6. Using ***sudo nano index.html*** to open and edit the file, replace the code inside with the code that will run the Feedback form website.

Hosting Website on Virtual Machine via Apache

Technical Explanation (2/2)

- Before saving and closing the **index.html**, return back to the Azure Function App and access the HTTP Trigger created previously and click the **Code + Test** in the **Developer** tab.
- Enter **code** that will send feedback from the website to an **output** (Database), and copy the **master function URL** (Figure 10).
- Paste the **function URL** into **index.html** as seen in Figure 11 and save the file.
- Return back to the **Virtual machine resource page** in Azure and copy its **Public IP address**.
- Paste the **Public IP address** into the **browser** and the Feedback form will be displayed, successfully hosted by the virtual machine (Figure 12).

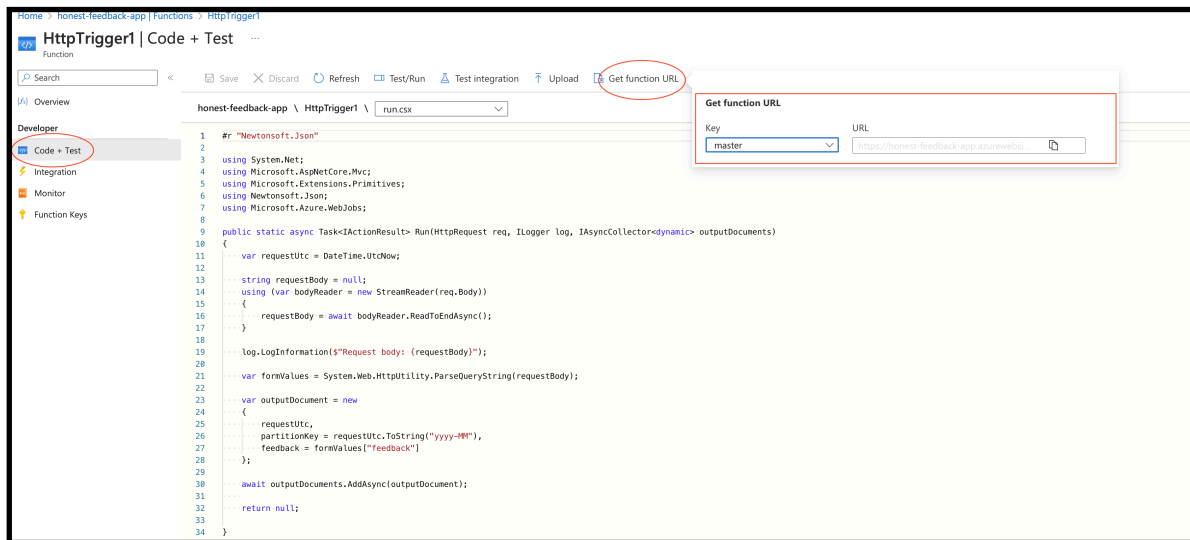


Figure 10: HTTP Trigger code within Code + Test, and access to master Function URL

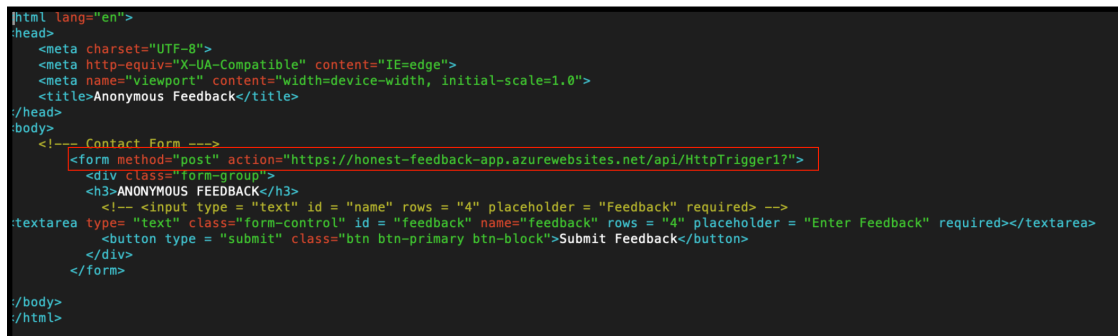


Figure 11: File index.html with function URL inserted

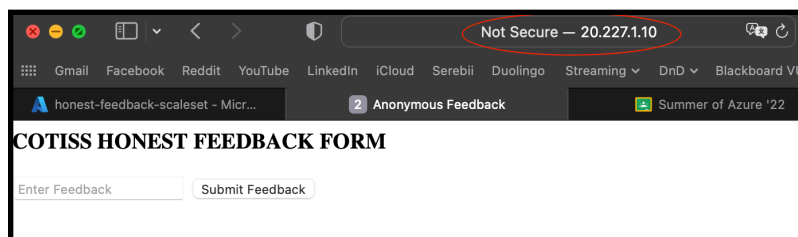


Figure 12: Entering Virtual machine public IP address in browser will display Feedback Form website

Auto-scaling with Load Balancer and Scale Set

Auto-scaling Overview

It is important to establish a Virtual Machine since this will be the resource that will host the feedback form website. A Virtual Network is created for resources to communicate virtually with each other, whether it's the Load balancer directing traffic to virtual machine instances or the scale set communicating more instances to the load balancer for example. A Scale set is used to create multiple instances of the virtual machine hosting the Feedback form website, working in tandem with a Load balancer. A Load balancer is used to manage the virtual machine instances and monitor their health and performance, where it will contact the Scale set if it needs more instances to keep up with traffic demands or if an instance has stopped working.

Auto-scaling Objectives

Auto-scaling removes the need for manual adjustments to fit internet traffic as to use and pay for extra resources that are not needed. Therefore employees can be utilised in other fields, since this concern is now managed by Load Balancers and Scale sets. The use of these resources also maintains high availability and reliability for the webpage, which depending on Service Level Agreements this will be very valuable to establish when deploying the website.

Technical Explanation (1/5)

Creating Virtual Network

1. Select **Create a resource** from the Azure portal menu or **Home page**.
2. Search "virtual network" in the **Marketplace** and proceed to select **Create > Virtual network**.
3. Enter the basic settings into the **Basics** section of the **Create virtual network** window.
4. Next in the **IP Addresses** section, under the **IPv4 address space** add the IP address **10.1.0.0/16**.
5. Next click **Add a subnet** to add a Back end subnet named **myBackendSubnet** with the Subnet Address range **10.1.0.0/24** (Figure 13).
6. Click **Save** to add the subnet, then press **Review + create** to complete the creation of the Virtual Network.

The screenshot shows the 'Create virtual network' wizard in the Azure portal. The 'IP Addresses' tab is selected. In the 'IPv4 address space' section, the address '10.1.0.0/16' is entered and highlighted with a red circle. Below this, the 'Add IPv6 address space' checkbox is unchecked. In the 'Add subnet' section, a table lists the subnets. The first subnet is named 'default' with an address range of '10.1.0.0/24'. To the right of the main form, the 'Subnet name' field is highlighted with a red circle and contains 'myBackendSubnet'. Below it, the 'Subnet address range' field is also highlighted with a red circle and contains '10.1.0.0/24'. The 'NAT gateway' dropdown is set to 'None'. The 'SERVICE ENDPOINTS' section shows '0 selected'.

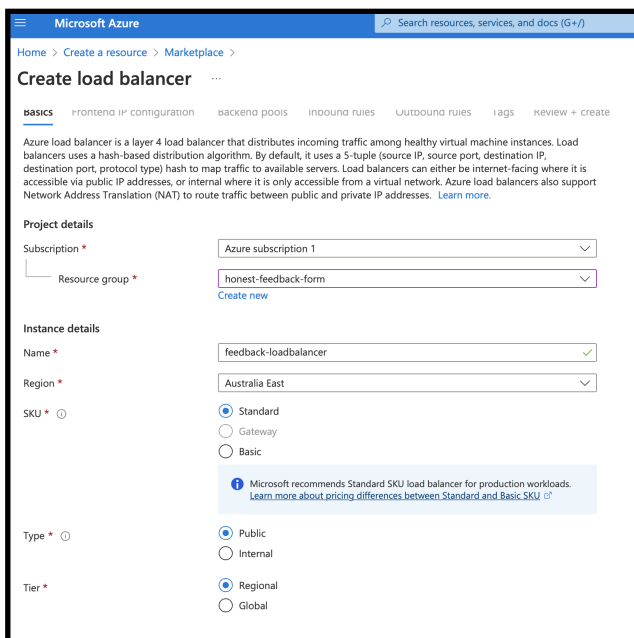
Figure 13: Establishing IPv4 address space for Virtual Network, and adding subnet myBackendSubnet

Auto-scaling with Load Balancer and Scale Set

Technical Explanation (2/5)

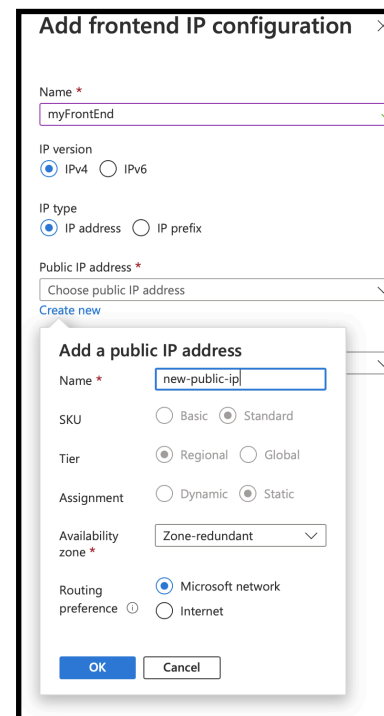
Load Balancer

1. Select **Create a resource** from the Azure portal menu or **Home page**.
2. Search “load balancer” in the **Marketplace** and proceed to select **Create > Load Balancer**.
3. Enter basic settings into the **Basics** section, and set **Type** to **Public** while keeping everything else at default, as seen in *Figure 14*.
4. Next in the **Frontend IP configuration** section, click **Add a frontend IP configuration** named **myFrontEnd**.
5. Seen in *Figure 15*, create a new **Public IP Address** for **myFrontEnd**, leaving everything default and pressing **OK**.
6. Next in Backend Pools, select **Add a backend pool**.
7. Enter **myBackend** for name, and the **virtual network** previously created for the virtual network (*Figure 16*).
8. Select **IP Address** for **Backend Pool Configuration** and press **Save**.



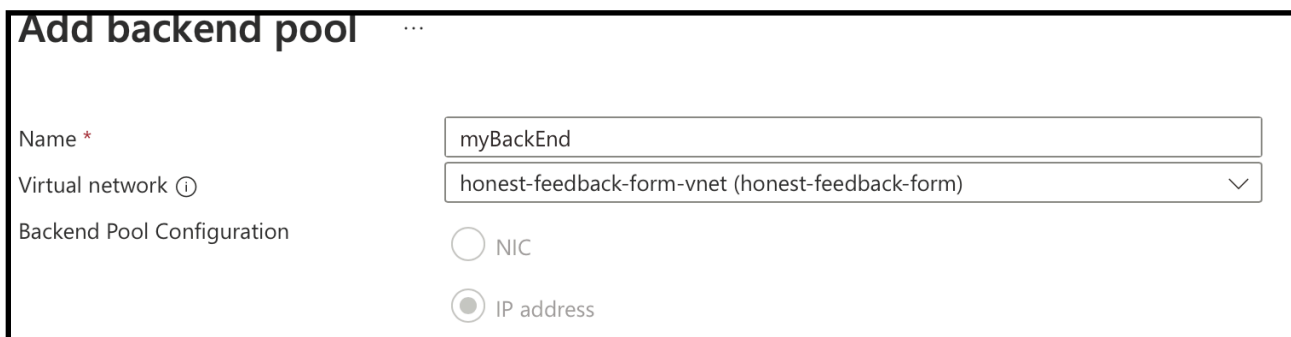
The screenshot shows the 'Create load balancer' page in the Microsoft Azure portal, specifically the 'Basics' tab. The page is titled 'Create load balancer' and includes a search bar at the top. Below the title, there are tabs for 'basics', 'frontend ip configuration', 'backend pools', 'inbound rules', 'outbound rules', 'tags', and 'review + create'. The 'basics' tab is active. The page contains several sections: 'Project details' with 'Subscription' set to 'Azure subscription 1' and 'Resource group' set to 'honest-feedback-form'; 'Instance details' with 'Name' set to 'feedback-loadbalancer', 'Region' set to 'Australia East', and 'SKU' set to 'Standard' (with a note recommending Standard SKU for production workloads); 'Type' set to 'Public'; and 'Tier' set to 'Regional'.

Figure 14: Basic section of Load Balancer creation



The screenshot shows the 'Add frontend IP configuration' dialog box. It has a title bar 'Add frontend IP configuration'. Inside, there are fields for 'Name' (set to 'myFrontEnd'), 'IP version' (set to 'IPv4'), 'IP type' (set to 'IP address'), and 'Public IP address' (set to 'Choose public IP address'). There is a 'Create new' link below the 'Public IP address' field. A modal dialog titled 'Add a public IP address' is open over the main dialog. It has a title bar 'Add a public IP address'. Inside, there are fields for 'Name' (set to 'new-public-ip'), 'SKU' (set to 'Standard'), 'Tier' (set to 'Regional'), 'Assignment' (set to 'Static'), 'Availability zone' (set to 'Zone-redundant'), and 'Routing preference' (set to 'Microsoft network'). There are 'OK' and 'Cancel' buttons at the bottom.

Figure 15: Frontend IP Configuration settings for Load Balancer



The screenshot shows the 'Add backend pool' dialog box. It has a title bar 'Add backend pool'. Inside, there are fields for 'Name' (set to 'myBackend') and 'Virtual network' (set to 'honest-feedback-form-vnet (honest-feedback-form)'). Below these fields, there is a section for 'Backend Pool Configuration' with two radio buttons: 'NIC' and 'IP address'. The 'IP address' radio button is selected.

Figure 16: Adding Backend Pool with virtual network to Load Balancer

Auto-scaling with Load Balancer and Scale Set

Technical Explanation (3/5)

Load Balancer

9. Next in Inbound rules, select **Add a load balancing rule**.
10. Refer to *Figure 17* to the values of each section that should be filled out.
11. For **Health probe**, select **Create new** and the window seen in *Figure 18* will appear.
12. Set **Protocol** to **TCP**, the **Port** to **80** and **Interval** at **5** (seconds), then select **OK**.
13. On the **Add load balancing rule** window, select **Add** once finished and proceed to the **Review + create** button.
14. After reviewing all **Load Balancer settings**, select **Create** to complete to deployment of the Load Balancer.

Add load balancing rule

Information: A load balancing rule distributes incoming traffic that is sent to a selected IP address and port combination across a group of backend pool instances. Only backend instances that the health probe considers healthy receive new traffic.

Name *
myHTTPRule ✓

IP Version *
☒ IPv4
☐ IPv6

Frontend IP address * ⓘ
myFrontEnd (To be created) ✓

Backend pool * ⓘ
myBackEnd ✓

Protocol *
☒ TCP
☐ UDP

Port *
80 ✓

Backend port * ⓘ
80 ✓

Health probe * ⓘ
(new) myHealthProbe ✓
[Create new](#)

Session persistence ⓘ
None ✓

Idle timeout (minutes) * ⓘ
15

TCP reset
☐ Disabled
☒ Enabled

Floating IP ⓘ
☒ Disabled
☐ Enabled

Add health probe

Information: Health probes are used to check the status of a backend pool instance. If the health probe fails to get a response from a backend instance then no new connections will be sent to that backend instance until the health probe succeeds again.

Name *
myHealthProbe ✓

Protocol *
TCP ✓

Port * ⓘ
80

Interval * ⓘ
5 seconds

Used by ⓘ
Not used

Buttons: OK, Cancel

Figure 18: Add health probe window with settings selected and filled.

Figure 17: Add Load balancing rule window filled out with appropriate settings for this project.

Auto-scaling with Load Balancer and Scale Set

Technical Explanation (4/5)

Virtual Machine Scale Set

1. Access the virtual machine created previously from the Azure portal menu or **Home page**, or the designated **Resource Group**.
2. Referring to *Figure 19*, take a **snapshot** of the virtual machine by selecting the **Capture** button on the virtual machine page.
3. Following the creation of a virtual machine image, return back to the Azure portal menu or **Home page** and select **Create a resource**.
4. Search “scale set” in the **Marketplace** and proceed to select **Create > Virtual Machine Scale Set**.
5. Enter basic settings into **Basics**, select **Flexible** for Orchestration mode (*Figure 20*).
6. Under **Instance details**, select **See all images** to find the previously created **image** of the **virtual machine** hosting the Feedback form website.
7. Referring to *Figure 21* under the **Other Items** tab, select **Shared Images** to find the **virtual machine image** that was previously created in *Step 2* above.

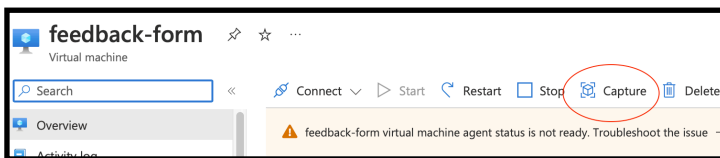


Figure 19: Select Capture to take a snapshot of the Virtual Machine that is currently hosting the website.

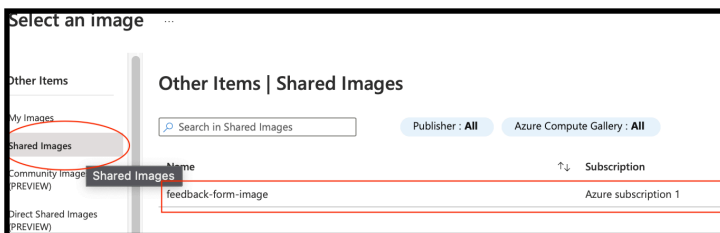


Figure 21: The snapshot taken of the virtual machine hosting the website is located under the Other Items tab by selecting Shared Images

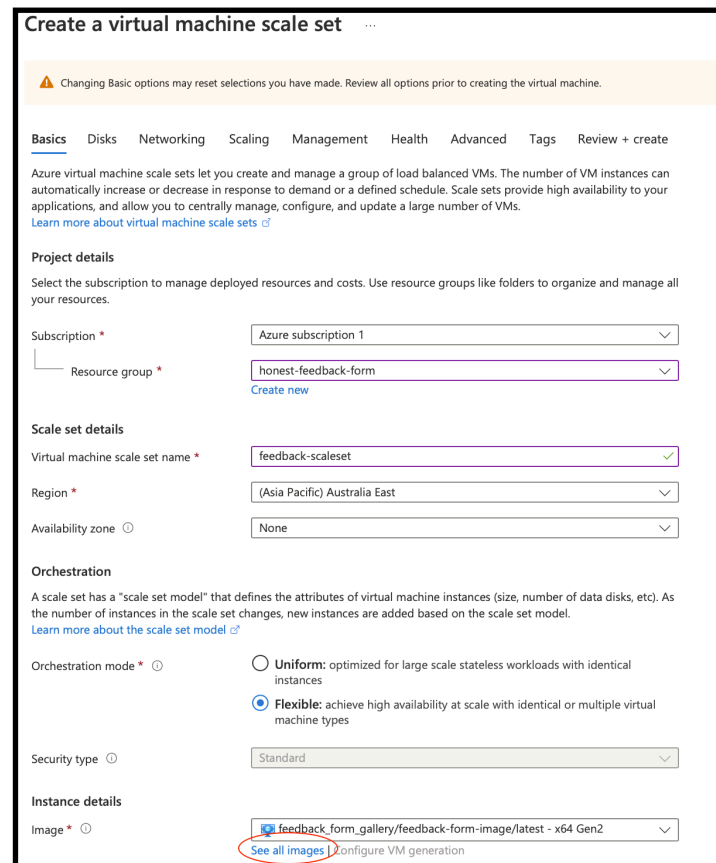


Figure 20: Virtual machine scale set Basics settings filled out

Auto-scaling with Load Balancer and Scale Set

Technical Explanation (5/5)

Virtual Machine Scale Set

- Next, the **Networking** section, select the **Virtual network** created previously and selecting the highlighted **Network interface**.
- Select the option to use a **Load balancer**, and select the **previously created Load balancer**, as well as the **Backend pool** created during the Load balancer's creation (*Figure 22*).
- Next, the **Scaling** section, set **Initial instance count** as **two**, and the **Scaling policy** as **Manual** (*Figure 23*).
- Leave all other settings as default and proceed to select to **Review + create**.
- After reviewing all **Virtual machine scale set settings**, select **Create** to complete to deployment.
- Once the resource has deployed, select **Go to resource** to view the **virtual machine scale set's page**.
- Similar to *Figure 24*, the **two initial instances' status** can be seen as **running**, as well as the **Public IP address** that will be used to view them and display the webpage as the original virtual machine did.

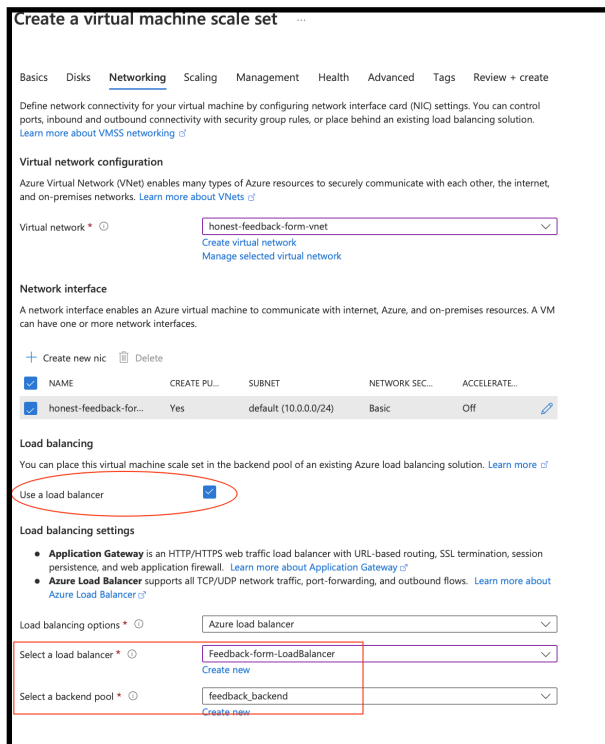


Figure 22: Networking settings for virtual machine scale set to include Load balancer.

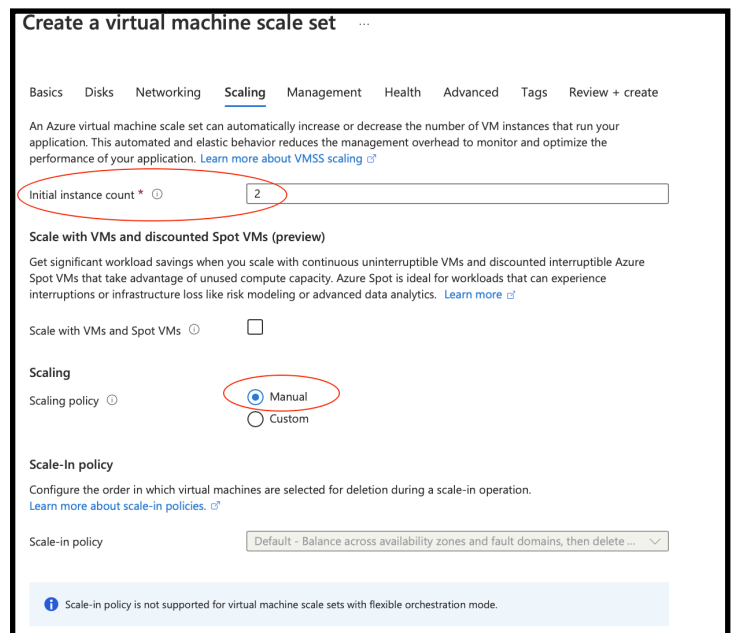


Figure 23: Scaling settings for virtual machine scale set, setting initial instance count to two.

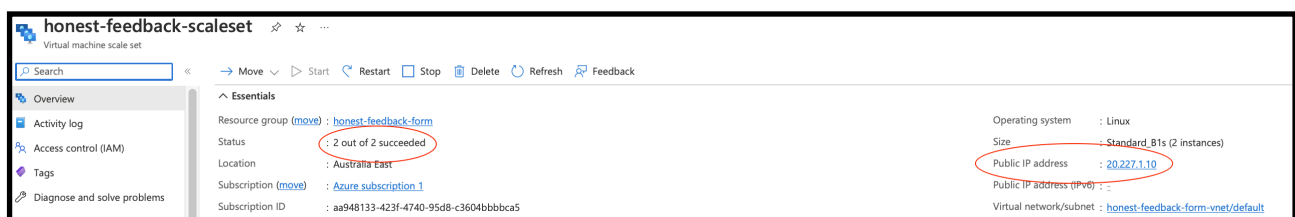


Figure 24: Virtual machine scale set page showing two instances running and Public IP address to view website.

Azure Cosmos Database

Azure Cosmos Database Overview

Azure Cosmos Database is a noSQL fully managed relational database, with instant scalability and single-digit millisecond response time. Service Level Agreements can be used to back Azure Cosmos Databases to ensure their high availability. This product will enable the feedback from the website to be received by the organisation at the other end to be accessed and analysed.

Azure Cosmos Database Objectives

The Azure Cosmos Database will be able to give the Azure Function App somewhere to send the feedback from the website to, and also store the information to then be accessed by the organisation. This acts as the final piece for establishing the connection between the website and the organisation receiving feedback reliably.

Technical Explanation (1/2)

1. Select **Create a resource** from the Azure portal menu or **Home page**.
2. Search “Azure Cosmos DB” in the **Marketplace** and proceed with **Create > Azure Cosmos Database for NoSQL**.
3. For the **Basics** section, enter the basic settings and set **Capacity Mode** to **Provisional throughout** (*Figure 25*).
4. Leave all other settings at their **default**, and proceed with **Review + create**.
5. Once the **Azure Cosmos Database** has been deployed, access the **Azure Function App** created previously from the Azure portal menu or **Home page**, or the designated **Resource Group**.
6. Select the **HTTP Trigger** under the **Functions** tab, and select the **Integration** tab.
7. Under **Outputs** select **+ Add output**.
8. Following *Figure 26*, fill out the **Create Output** window to connect the **Azure Cosmos Database** to the **Azure Function App’s HTTP Trigger**.

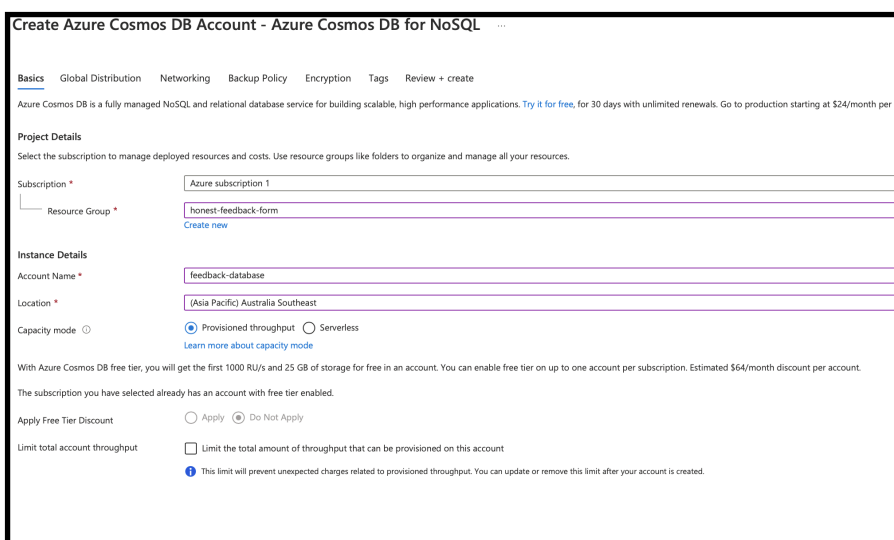


Figure 25: Basic settings for creating Azure Cosmos Database

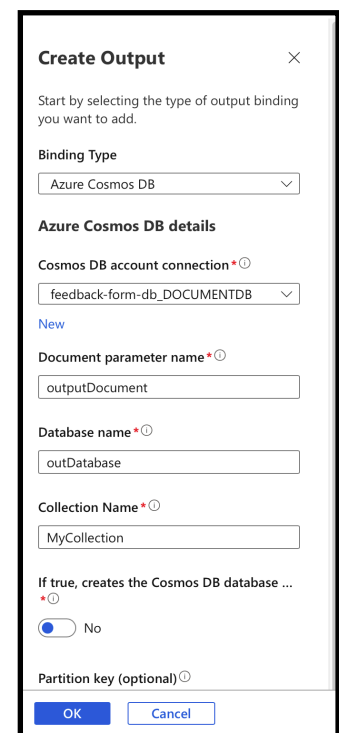


Figure 26: Creating output for Azure Function App’s HTTP Trigger to send information from website to Azure Cosmos Database

Azure Cosmos Database

Technical Explanation (2/2)

9. With the **Azure Function App's HTTP Trigger's output** connected to the **Azure Cosmos Database**, now when the **Submit button** on the **website** is pressed the **feedback** is sent to the **Azure Cosmos Database**.
10. From the Azure Cosmos Database page, following *Figure 27*, select **Data Explorer**.
11. Within the NoSQL API, under the Data tab, in the output **outDatabase**, selecting **Items** will reveal all the **feedback** that has been sent through from the **website**.
12. Selecting a sample of feedback will display the **feedback, when it was sent**, and also keeps the information of the **author anonymous**.

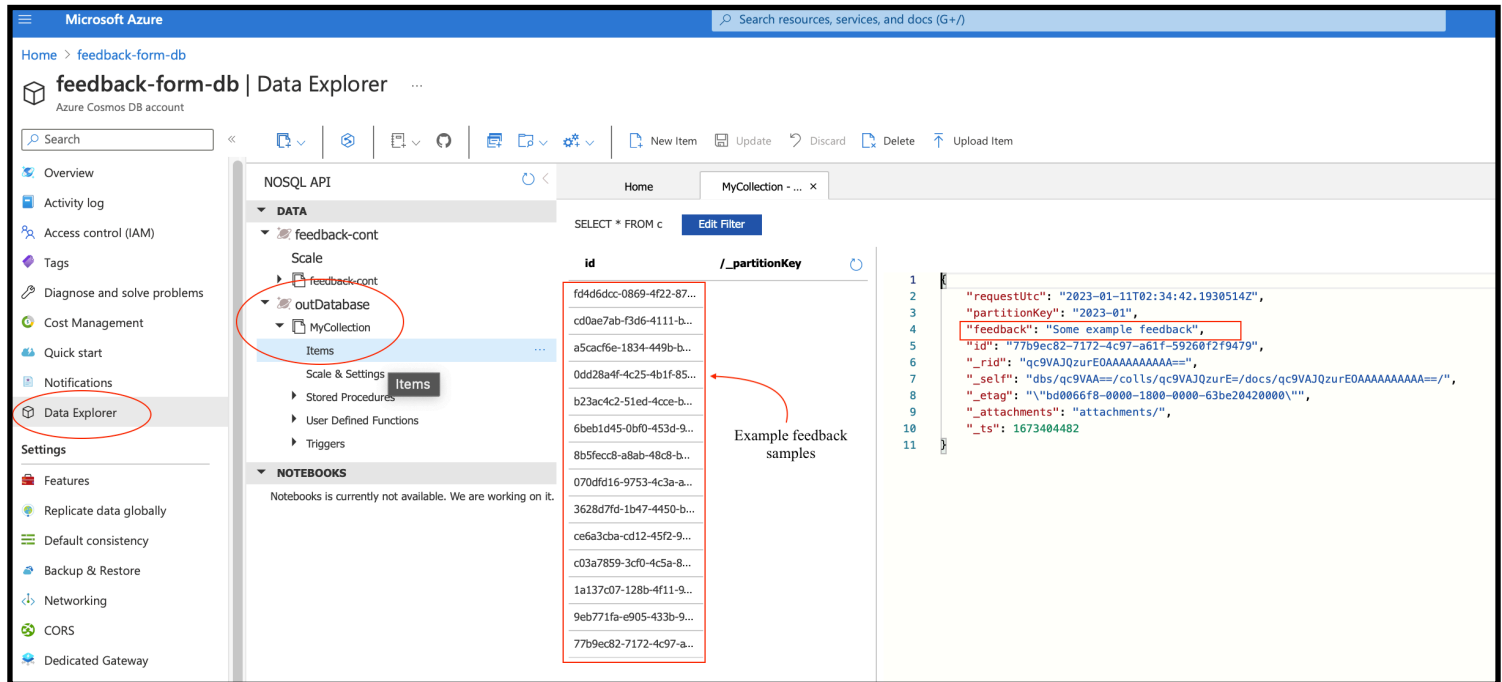


Figure 27: From the Azure Cosmos Database page, selecting Data Explorer and choosing the outDatabase output will enable access to feedback sent from website.