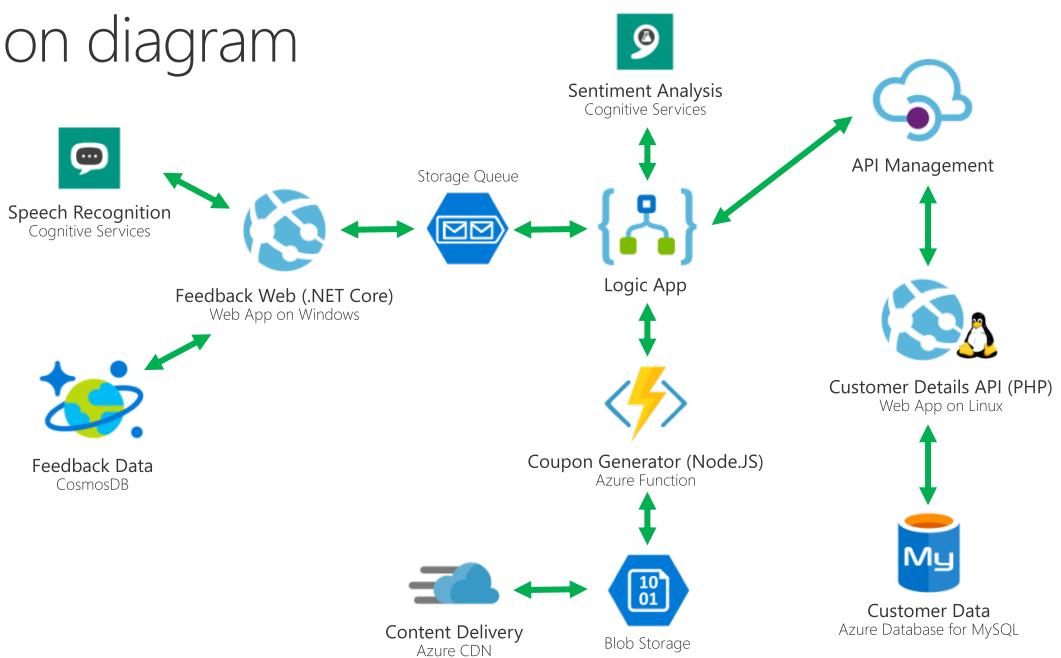


# Preparing and deploying the solution

## Solution diagram

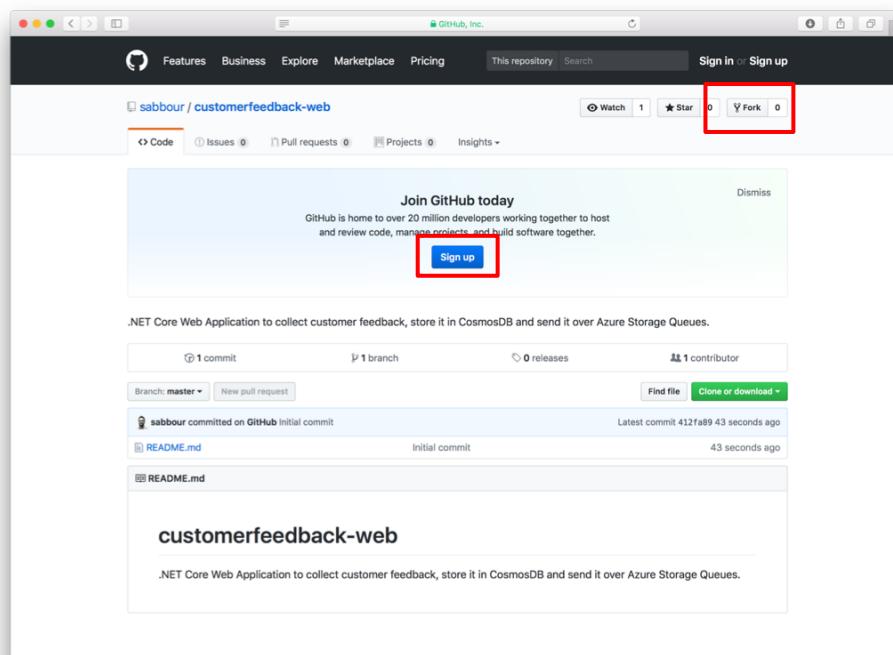


## 0. Pre-requisites

WiFi Code: VQP2YV

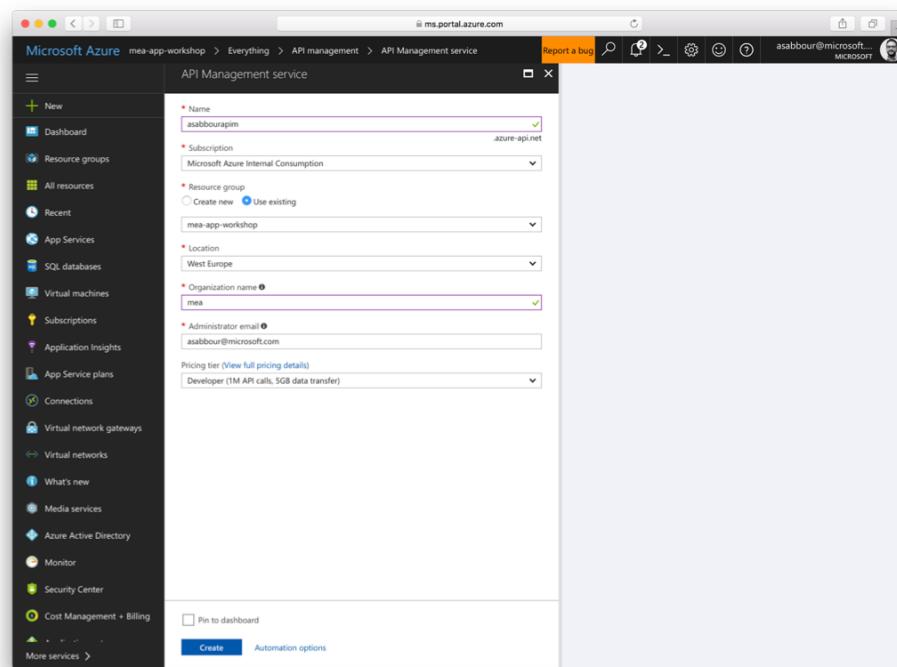
Download the lab guide (if you want)  
<http://github.com/sabbour/customerfeedback>

1. Create Resource Group **mea-app-workshop**. You will deploy all resources into that group.
2. Create Github account.
3. Fork the following Github repos to your account:
  - <http://github.com/sabbour/customerfeedback-web>
  - <http://github.com/sabbour/customerfeedback-api>
  - <http://github.com/sabbour/customerfeedback-functions>



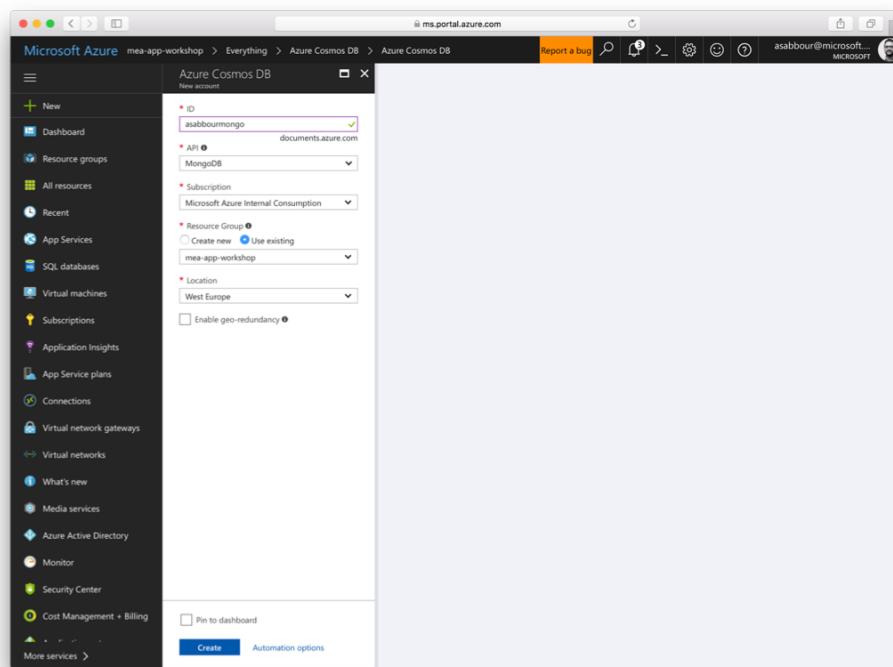
## 1. Create API Management Gateway

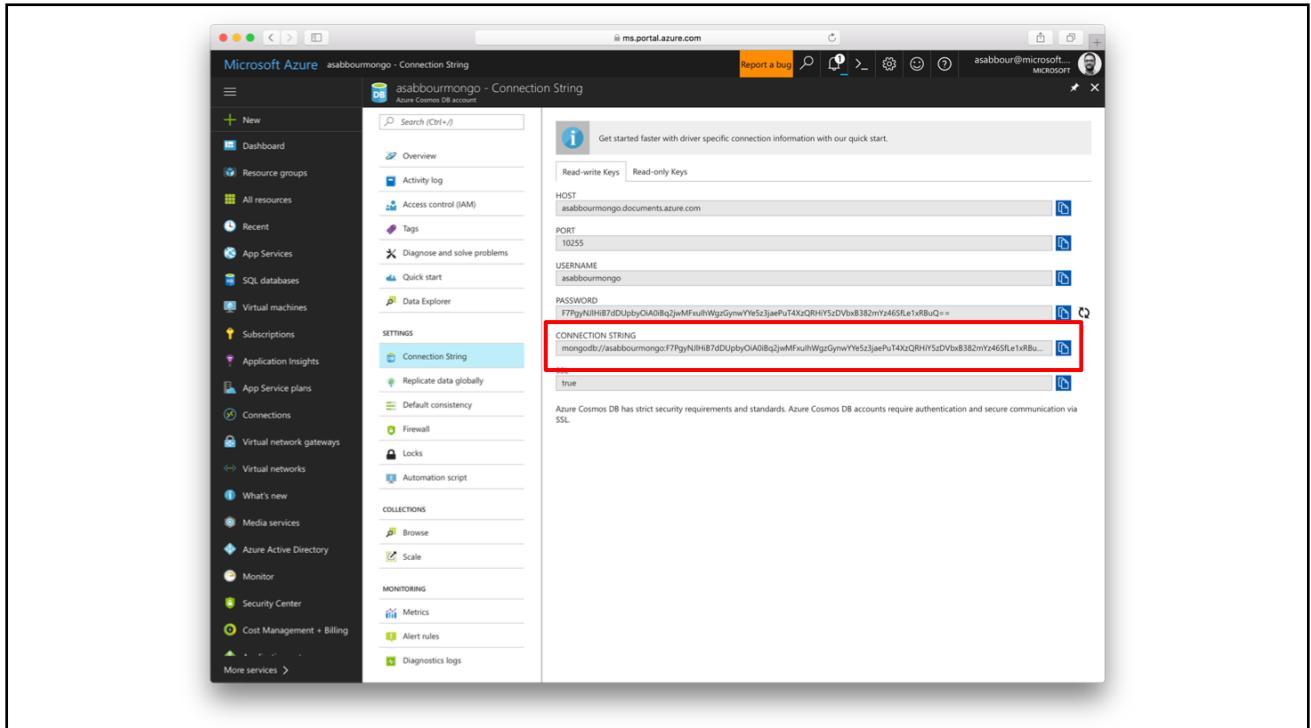
- Give it a unique name, like `[alias]apim`
- Choose Developer tier.
- This will take about 30 minutes to provision.  
Proceed with the next steps.



## 2. Create CosmosDB

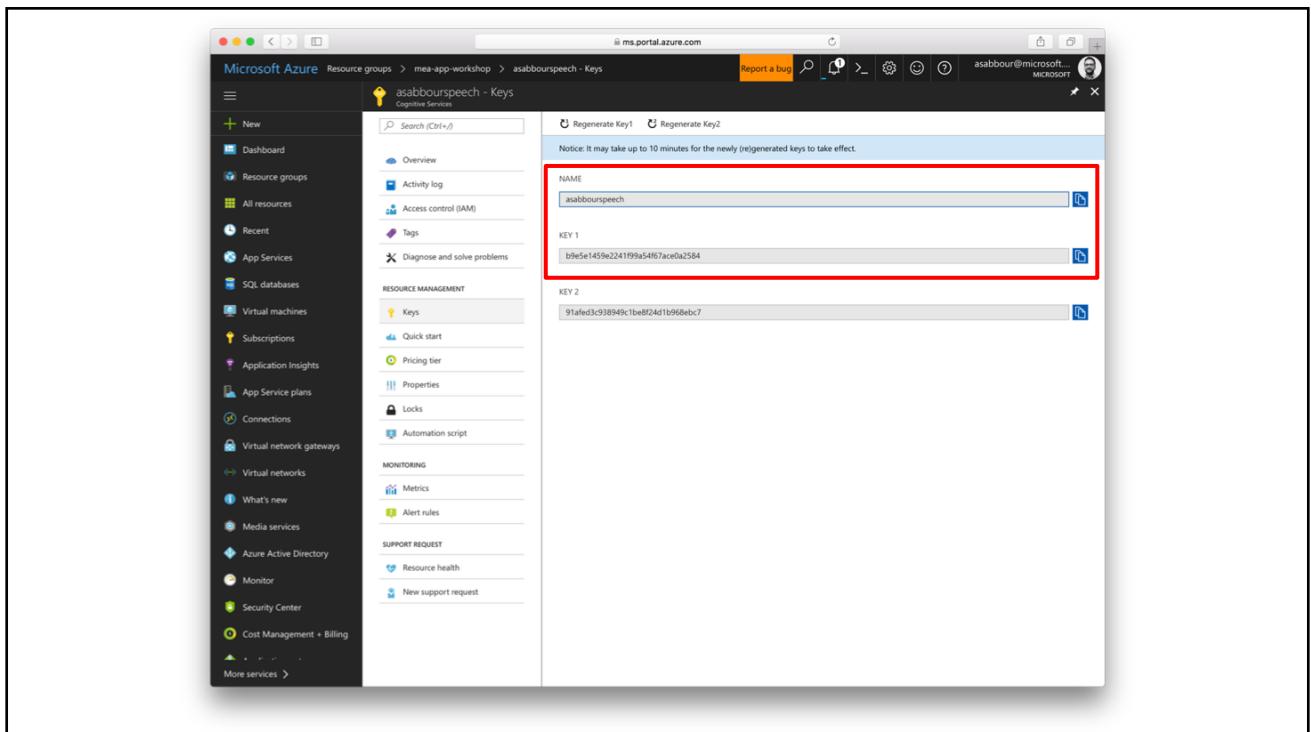
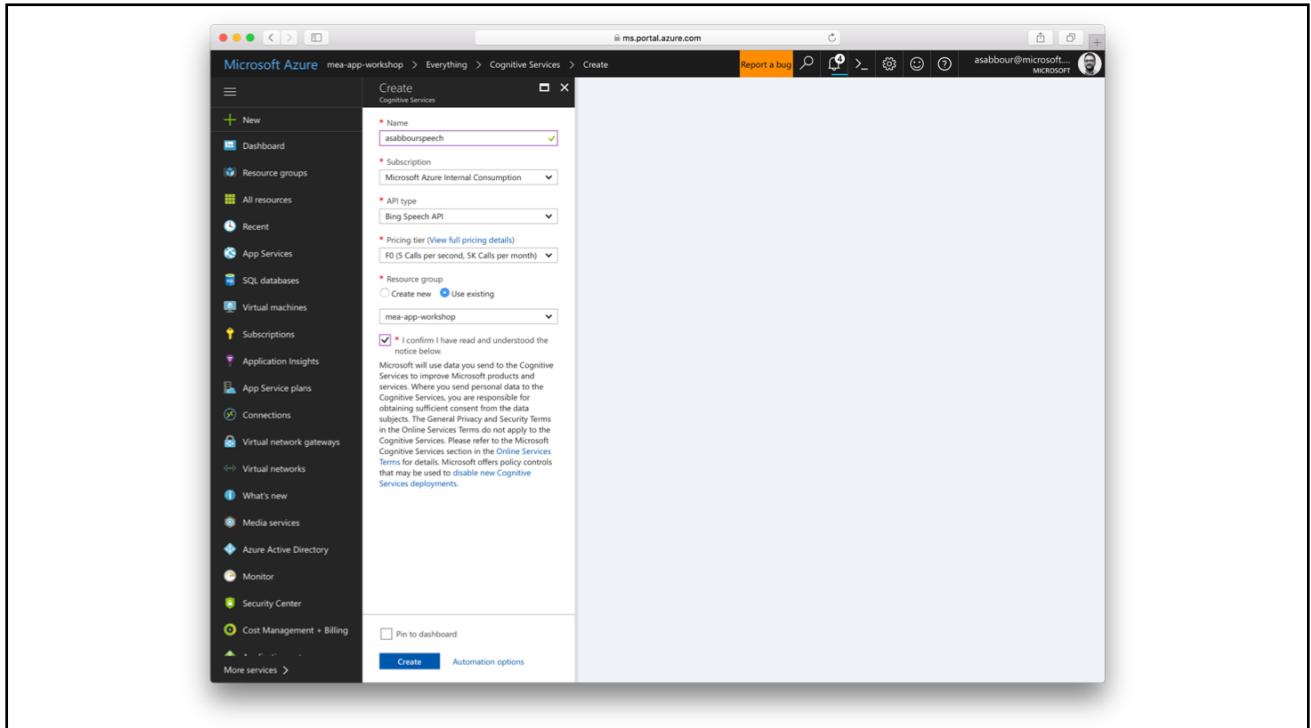
- Give it a unique name, like *[alias]mongo*
- Choose the **MongoDB API**
- This will take about 5 minutes to provision.  
Proceed with the next steps.
- When the provisioning is done, go back to get the MongoDB connection string and store it aside. You'll need it later.





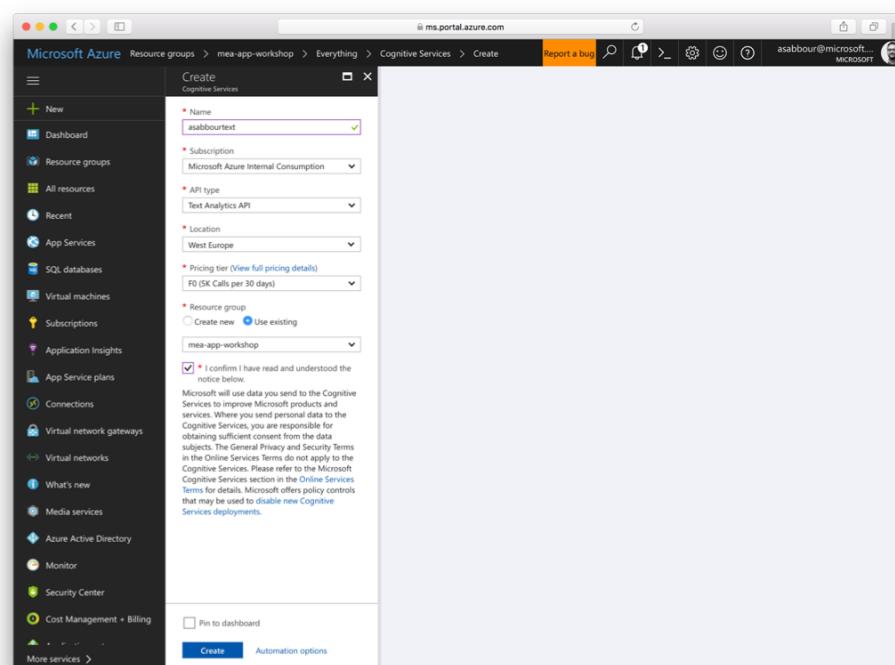
### 3. Create Bing Speech API from Cognitive Services

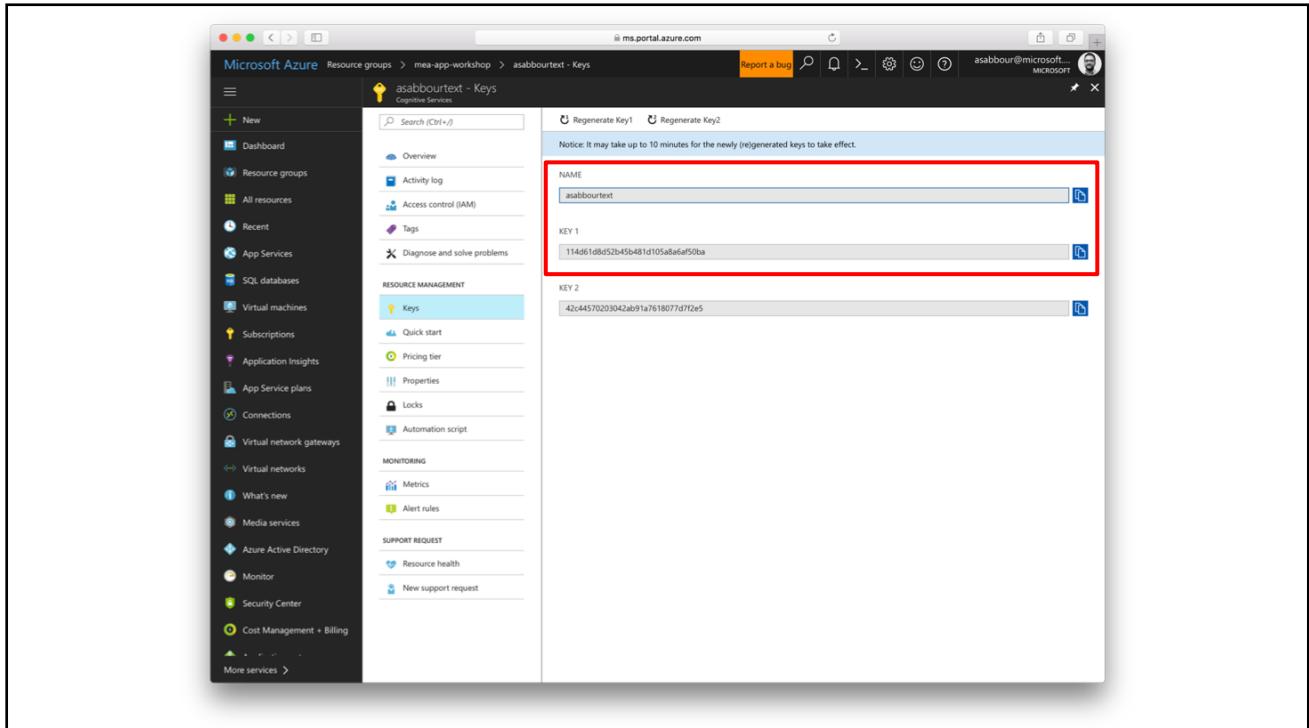
- Give it a unique name, like *[alias]speech*
- Choose the Free tier.
- Get the key and store it aside. You'll need it later.



## 4. Create Text Analytics API from Cognitive Services

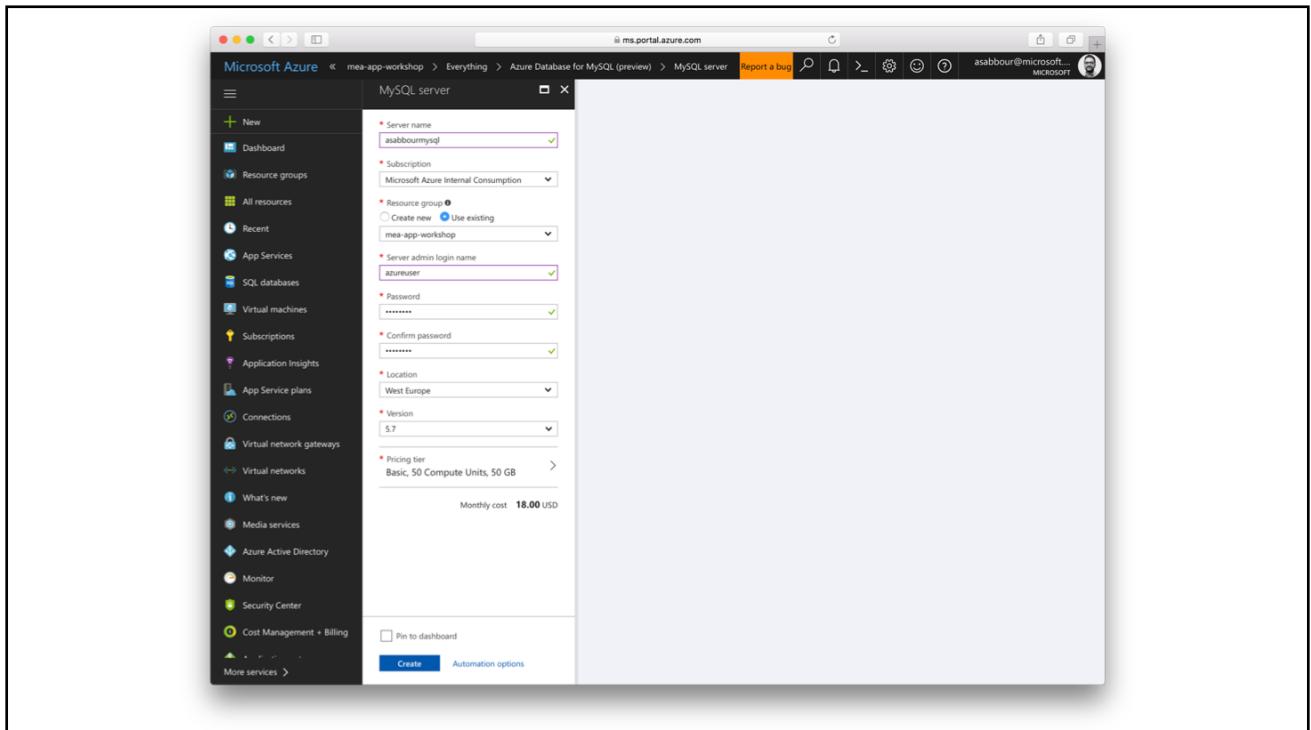
- Give it a unique name, like *[alias]text*
- Choose the Free tier.
- Get the key and store it aside. You'll need it later.

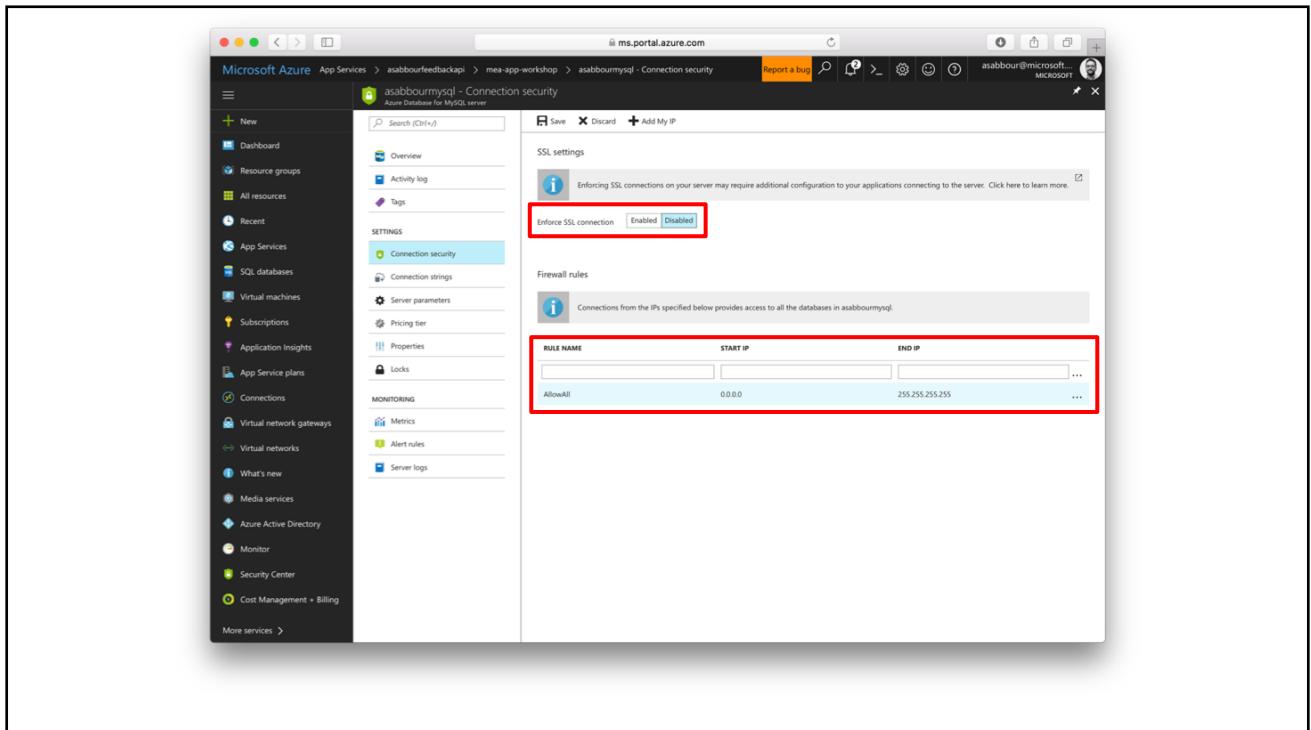




## 5. Create Azure Database for MySQL

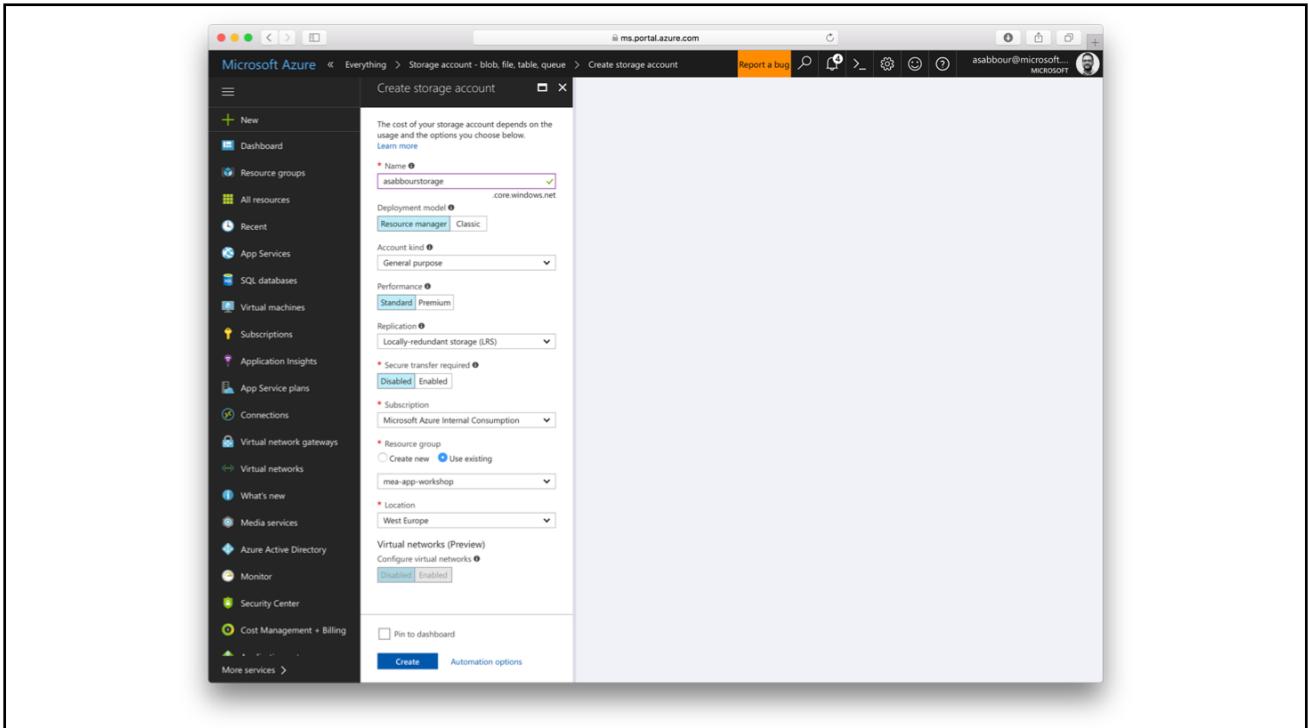
- Give it a unique name, like **[alias]mysql**
- Provide **azureuser** as the admin login name.
- Provide a password and write it down, as you'll need it later.
- Switch to the Basic tier and create.
- Once provisioned, make note of your server name and full admin username.
- Disable Enforce SSL Connection and create a firewall rule starting with 0.0.0.0 to 255.255.255.255



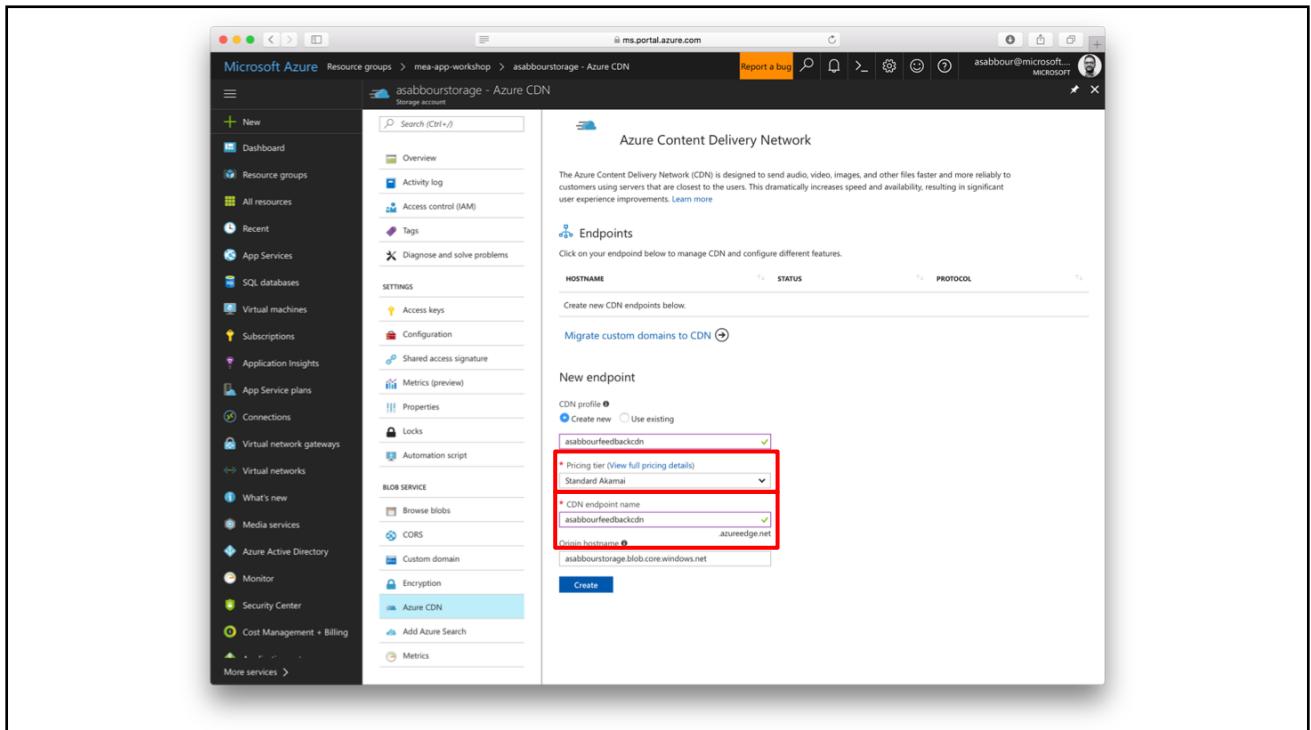


## 6. Create Storage Account

- Give it a unique name, like **[alias]storage**
- Pick **General Purpose** as the account kind.
- Create it in the same resource group and region.
- Once provisioned, make note of your storage account name and key.
- Optionally, activate Azure CDN for the storage account with **[alias]feedbackcdn** as the CDN endpoint name and **Standard Akamai** as the tier.

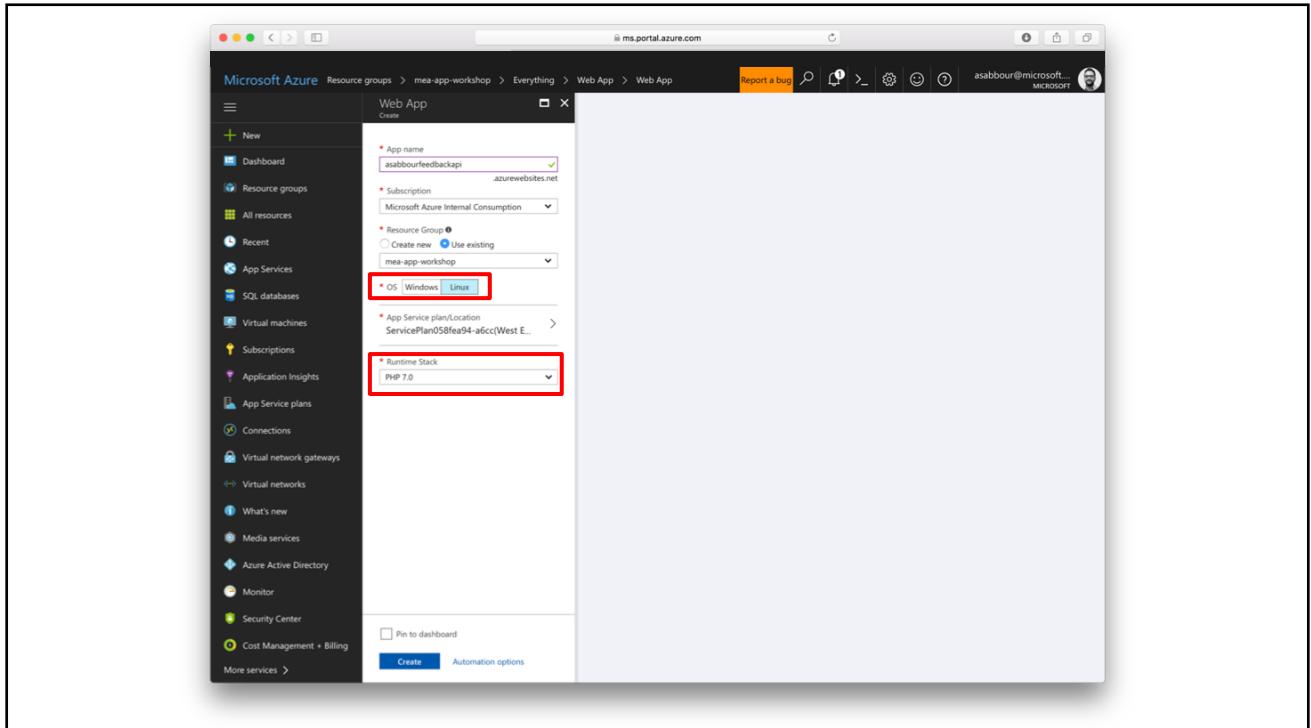


NAME	KEY	CONNECTION STRING
key1	2M+QMBNwmVwoGRSQ7GhEDDI/2FMg0b0A...	DefaultEndpointsProtocol=https;AccountName=asabbourstorage;...;DefaultEndpointsProtocol=https;AccountName=asabbourstorage...
key2	BCU7PqGJh/qky8Y73z6B/YmVPLU8+exAy...	DefaultEndpointsProtocol=https;AccountName=asabbourstorage;...;DefaultEndpointsProtocol=https;AccountName=asabbourstorage...



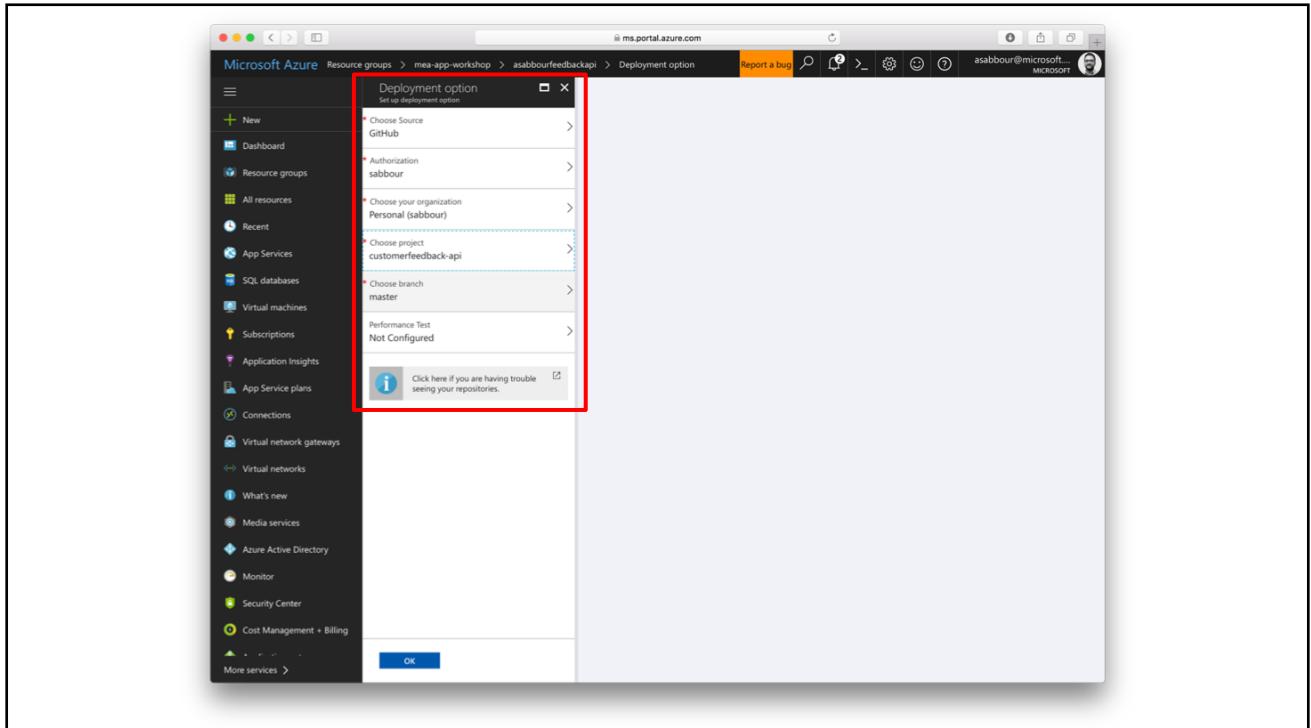
## 7.1 Create Web App for Feedback API running PHP on Linux

- Give it a unique name, like `[alias]feedbackapi`
- Choose **Linux** as the OS.
- Choose **PHP 7.0** as the Runtime Stack.

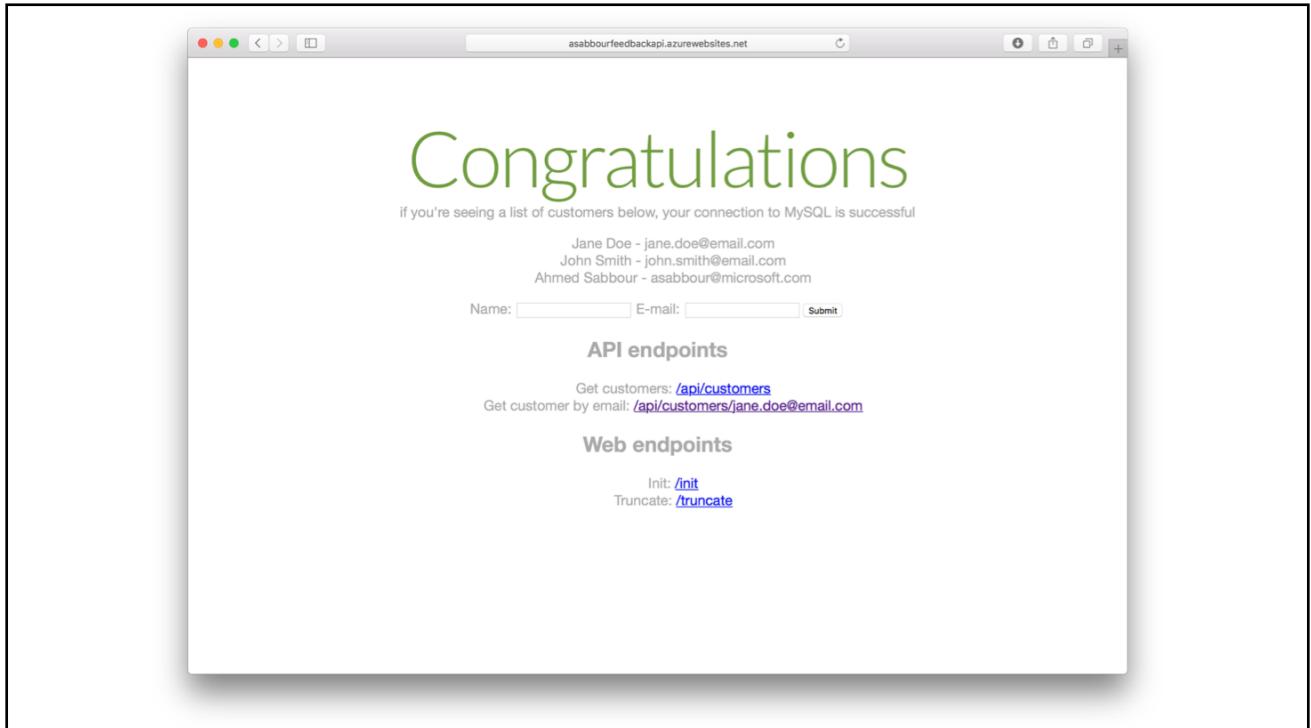


## 7.2 Configure Feedback API

- Setup deployment from your **customerfeedback-api** Github fork.
- Create App Settings with the below values:
  - **MYSQL\_SERVER:** `[alias]mysql.mysql.database.azure.com`
  - **MYSQL\_USER:** `azureuser@[alias]mysql`
  - **MYSQL\_PASSWORD:** the password you specified earlier
- Go to [http://\[alias\]feedbackapi.azurewebsites.net/init](http://[alias]feedbackapi.azurewebsites.net/init) you should see some customers already if things are configured properly.
- Add your name and email

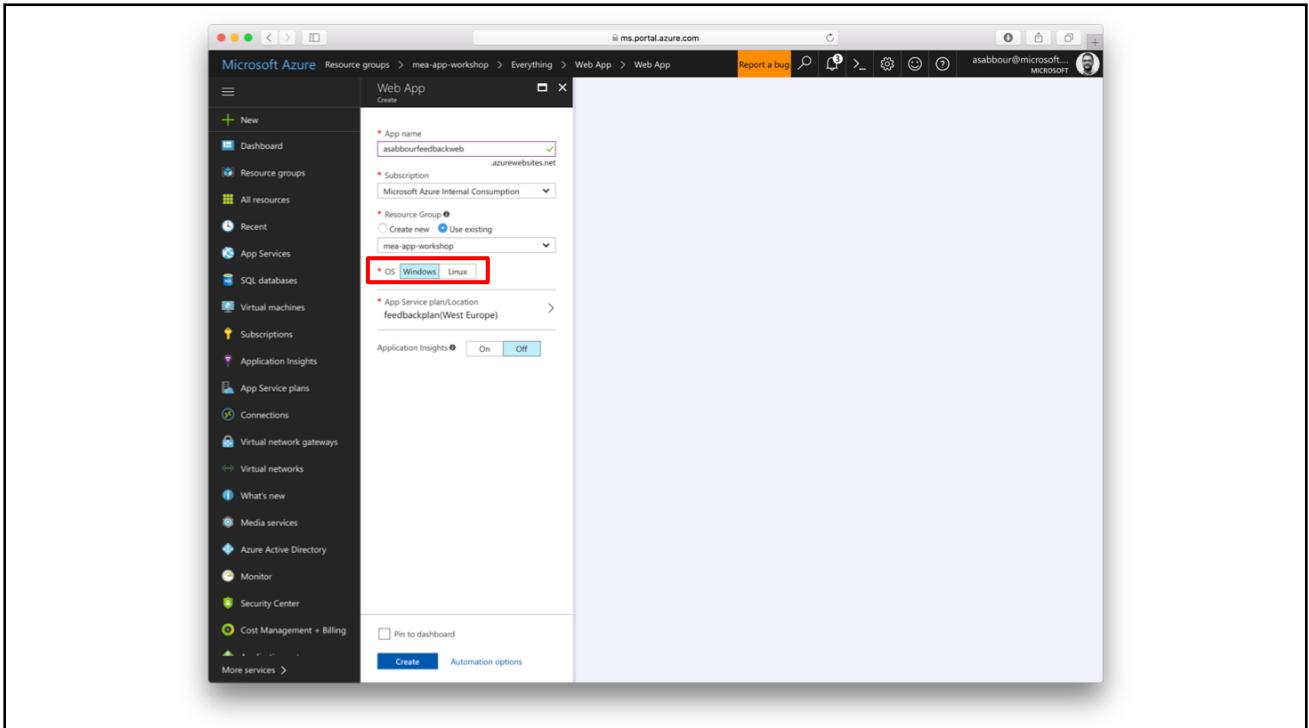


Key	Value	Slot setting	...
MYSQL_SERVER	asabbourmysql.mysql.database.azure.com	<input type="checkbox"/>	Slot setting
MYSQL_USER	azureuser@asabbourmysql	<input type="checkbox"/>	Slot setting
MYSQL_PASSWORD	p@ssw0rd	<input type="checkbox"/>	Slot setting



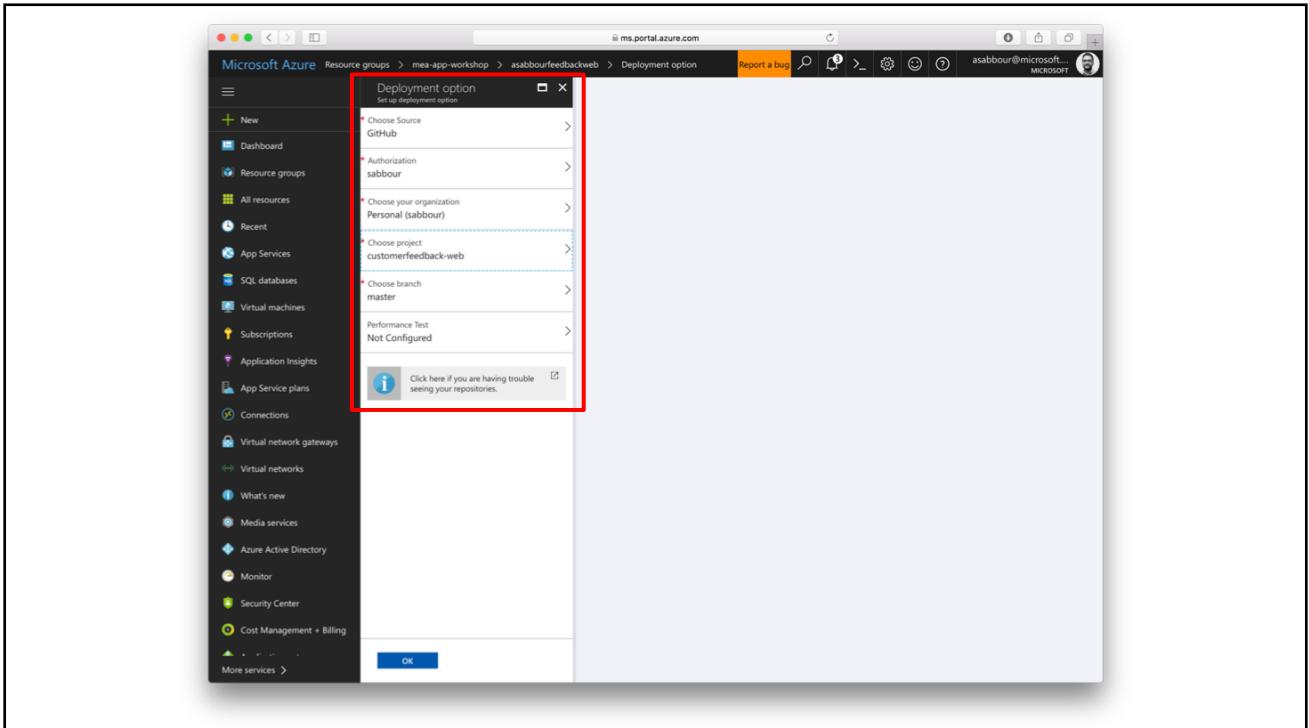
## 8.1 Create Web App for new Feedback website running .NET on Windows

- Give it a unique name, like *[alias]feedbackweb*
- Choose **Windows** as the OS.

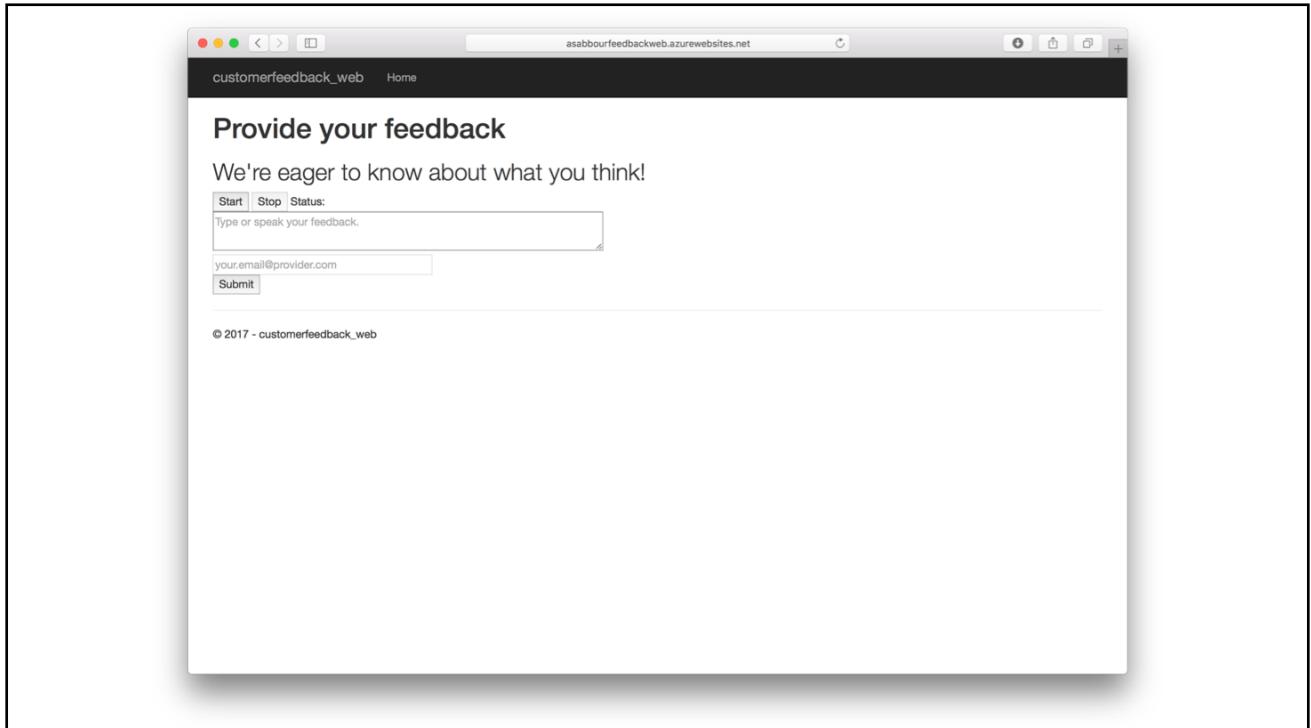


## 8.2 Configure the new .NET Feedback website

- Setup deployment from your **customerfeedback-web** Github fork.
- Create App Settings with the below values:
  - MONGO\_URL: mongodb connection string specified earlier
  - SPEECH\_KEY: speech API key specified earlier
  - STORAGEACCOUNT\_NAME: storage account name
  - STORAGEACCOUNT\_KEY: storage account key
- Go to [http://\[alias\]feedbackweb.azurewebsites.net](http://[alias]feedbackweb.azurewebsites.net) you should see a form to collect feedback



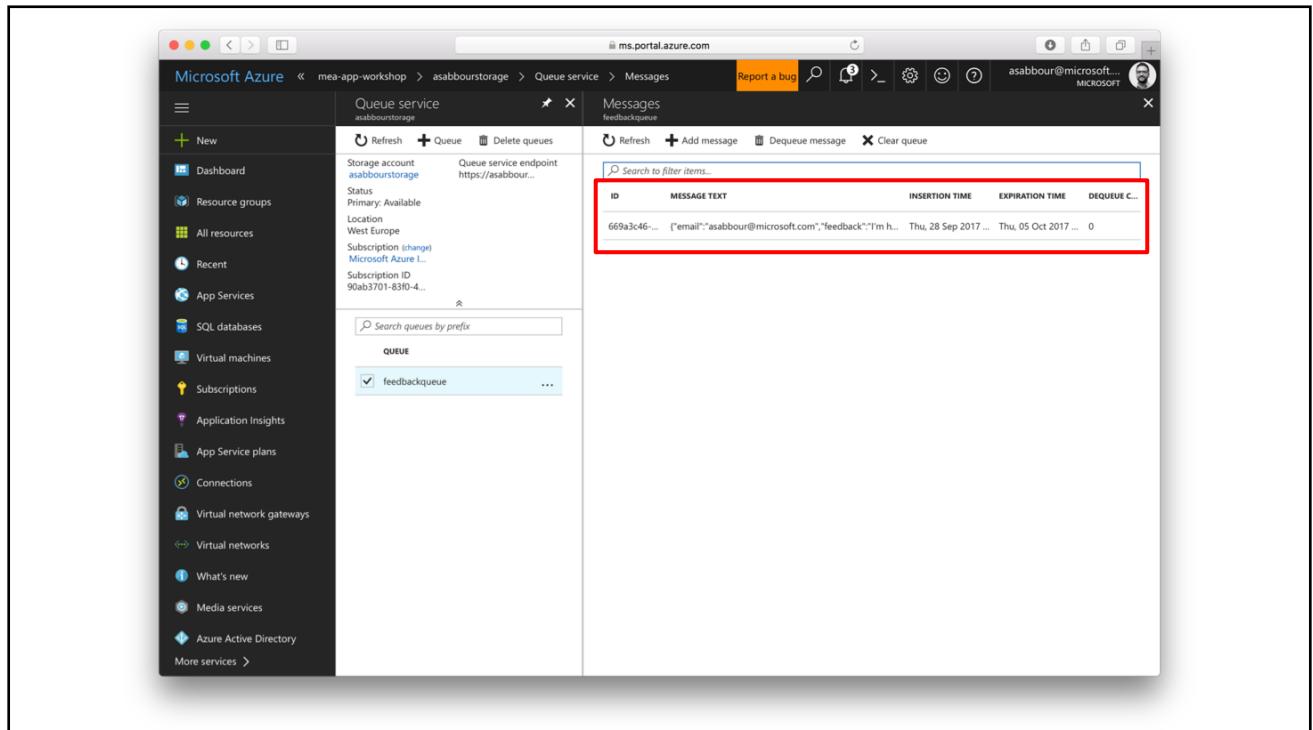
WEBSITE_NODE_DEFAULT_VERSION	6.9.1	<input type="checkbox"/> Slot setting	...
MONGO_URL	mongodb://asabbourmongoF7PgyNjH8B...	<input type="checkbox"/> Slot setting	...
SPEECH_NAME	asabbourspeech	<input type="checkbox"/> Slot setting	...
SPEECH_KEY	bfe5e1459e224199a54f67ace0a25b4	<input type="checkbox"/> Slot setting	...
STORAGEACCOUNT_NAME	asabbourstorage	<input type="checkbox"/> Slot setting	...
STORAGEACCOUNT_KEY	2M+QM8NwmVwoGRSQ7IGHEDD/2FMgOb...	<input type="checkbox"/> Slot setting	...
Key	Value	<input type="checkbox"/> Slot setting	...



```

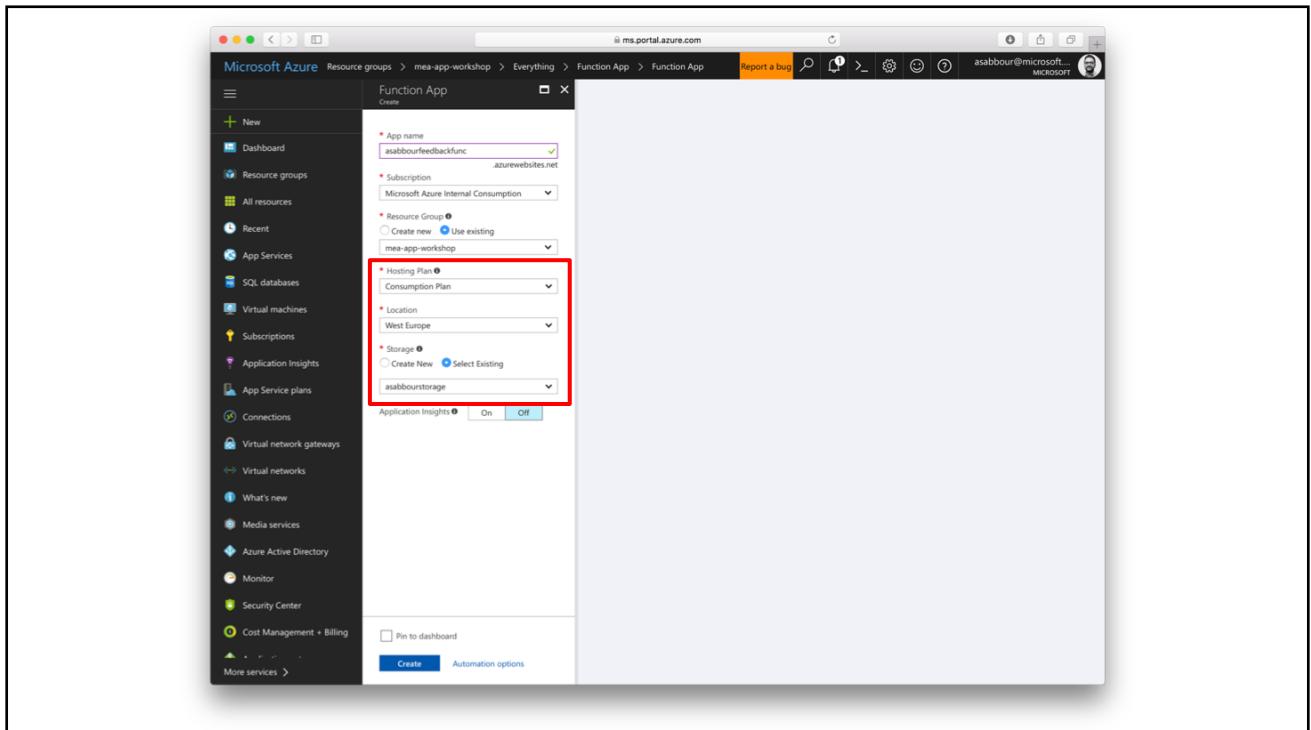
1 {
2   "_id" : ObjectId("59cd7a87ce10491580e4caff"),
3   "email" : "asabour@microsoft.com",
4   "feedback" : "I'm happy",
5   "feedback_id" : "800631e2-6ee4-42ab-9c
6 }

```



## 9.1 Create Function App to generate coupons

- Use **Consumption Plan** as the hosting plan.
- Choose the same resource group and region.
- Choose the same storage account.



## 9.2 Configure the Function App

- Setup deployment from your **customerfeedback-functions** Github fork.
- Create App Settings with the below values:
  - **BlobStorageConnection:** blob storage connection string in the form of DefaultEndpointsProtocol=https;AccountName=[account name];AccountKey=[account key]

The screenshot shows the Microsoft Azure portal interface. The left sidebar lists various services like New, Dashboard, Resource groups, etc. The main area shows the 'Function Apps' section with a search bar for 'asabbourfeedbackfunc'. The 'Overview' tab is active, and a red box highlights the 'Platform features' tab in the top navigation bar. The 'CODE DEPLOYMENT' section contains a 'Deployment options' link, which is also highlighted with a red box.

The screenshot shows the Microsoft Azure portal interface, specifically the 'Deployments' section for the 'asabbourfeedbackfunc' function app. A red box highlights the 'Deployment option' dialog box. The dialog box shows the following configuration steps:

- Choose Source: GitHub
- Authorization: sabbour
- Choose your organization: Personal (sabbour)
- Choose project: customerfeedback-functions
- Choose branch: master
- Performance Test: Not Configured

At the bottom of the dialog box, there is a note: "Click here if you are having trouble seeing your repositories." and an "OK" button.

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation sidebar includes options like Dashboard, Resource groups, All resources, Recent, App Services, SQL databases, Virtual machines, Subscriptions, Application Insights, App Service plans, Connections, Virtual network gateways, Virtual networks, What's new, Media services, and Azure Active Directory. The main content area displays the 'Function Apps' section for the 'asabbourfeedbackfunc' app. A specific function named 'GenerateCoupon' is selected. The code editor shows the 'index.js' file content:

```

1 var path = require('path');
2 var Jimp = require("jimp");
3 var azurestorage = require('azure-storage');
4 var stream = require('stream');
5 var util = require('util');
6
7 module.exports = function (context, data) {
8
9     context.log(util.inspect(data, false, null));
10
11    var name = data.CustomerName;
12    var code = getRandomInt(100, 99999);
13    var outputfileName = name + "_" + code + ".jpg";
14
15    context.log("Coupon generation for: " + name + " code: " + code);
16
17
18    // Load coupon image and read it with Jimp
19    var baseImagePath = path.resolve(__dirname, 'coupon.jpg');
20    context.log("Template image path: " + baseImagePath);
21
22    Jimp.read(baseImagePath).then((image) => {
23        // Load font
24        Jimp.loadFont(Jimp.FONT_SANS_32_BLACK).then(function (font) {
25            // Write the customer name on the image
26            image.print(font, 60, 150, "25% off voucher for " + name, 500);
27            // Get the image as a stream
28            image.getBuffer(Jimp.MIME_JPEG, (error, buffer) => {
29
30                // Save Stream to blob
31                var imageStream = new stream.PassThrough();
32                imageStream.end(buffer);
33                saveStreamToBlockBlob(context, 'coupons', outputfileName, imageSt

```

The screenshot shows the 'Application settings' tab for the 'asabbourfeedbackfunc' function app. The 'General settings' section shows .NET Framework version as v4.7 and Platform as 32-bit. The 'Debugging' section has Remote debugging set to Off and Remote Visual Studio version set to 2015. The 'Application settings' section lists several environment variables:

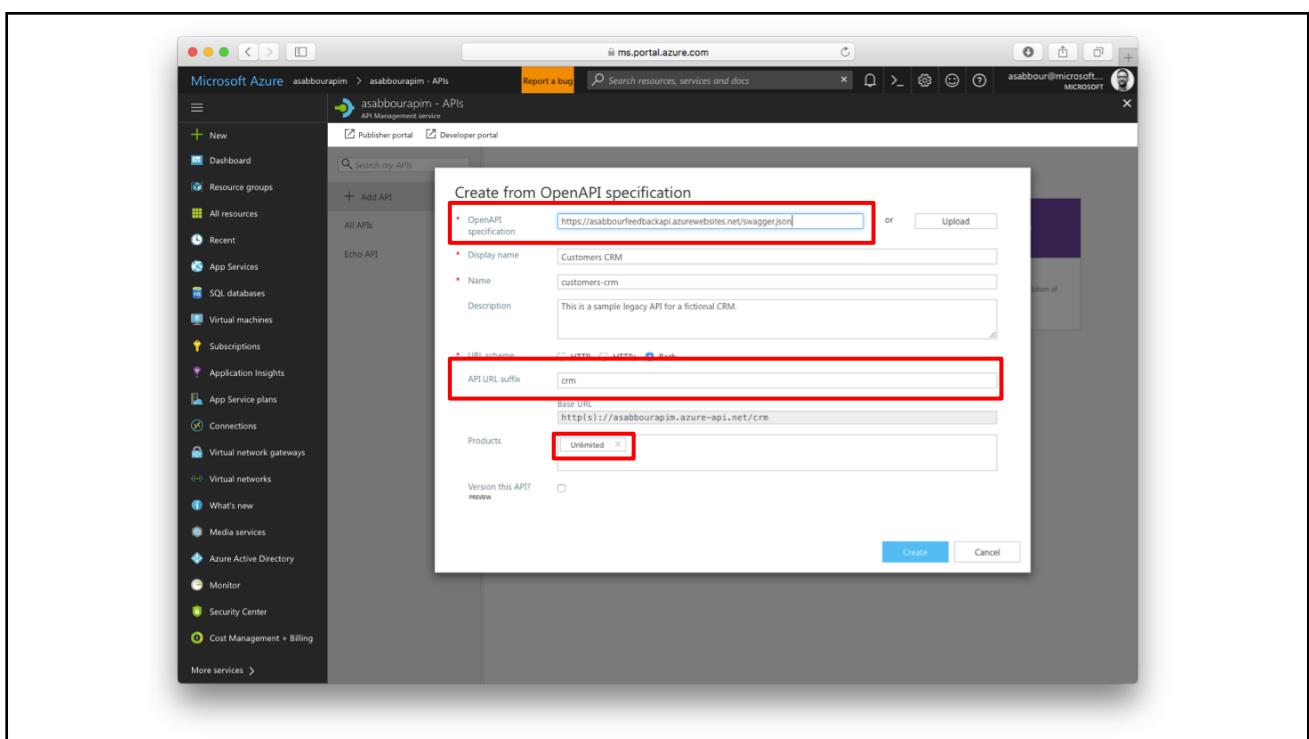
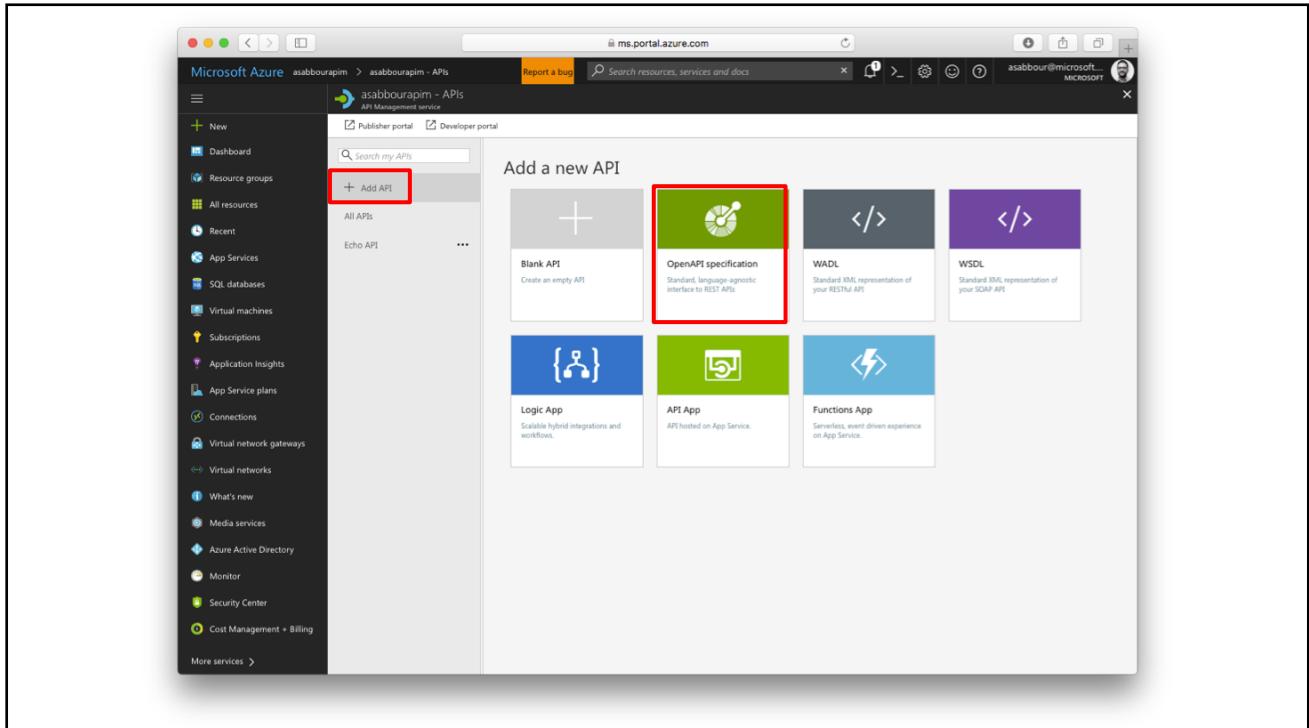
AzureWebJobsDashboard	DefaultEndpointsProtocol=https;AccountName=asabbourstorage;AccountKey=2M+QM8NwvwoGR...
AzureWebJobsStorage	DefaultEndpointsProtocol=https;AccountName=asabbourstorage;AccountKey=2M+QM8NwvwoGR...
FUNCTIONS_EXTENSION_VERSION	-1
WEBSITE_CONTENTAZUREFILECONNECTIONS...	DefaultEndpointsProtocol=https;AccountName=asabbourstorage;AccountKey=2M+QM8NwvwoGR...
WEBSITE_CONTENTSHARE	asabbourfeedbackfunc\def
WEBSITE_NODE_DEFAULT_VERSION	v4.7
blobStorageConnection	DefaultEndpointsProtocol=https;AccountName=asabbourstorage;AccountKey=2M+QM8NwvwoGR...

The 'blobStorageConnection' entry is highlighted with a red box.

# Importing Feedback API into API Management

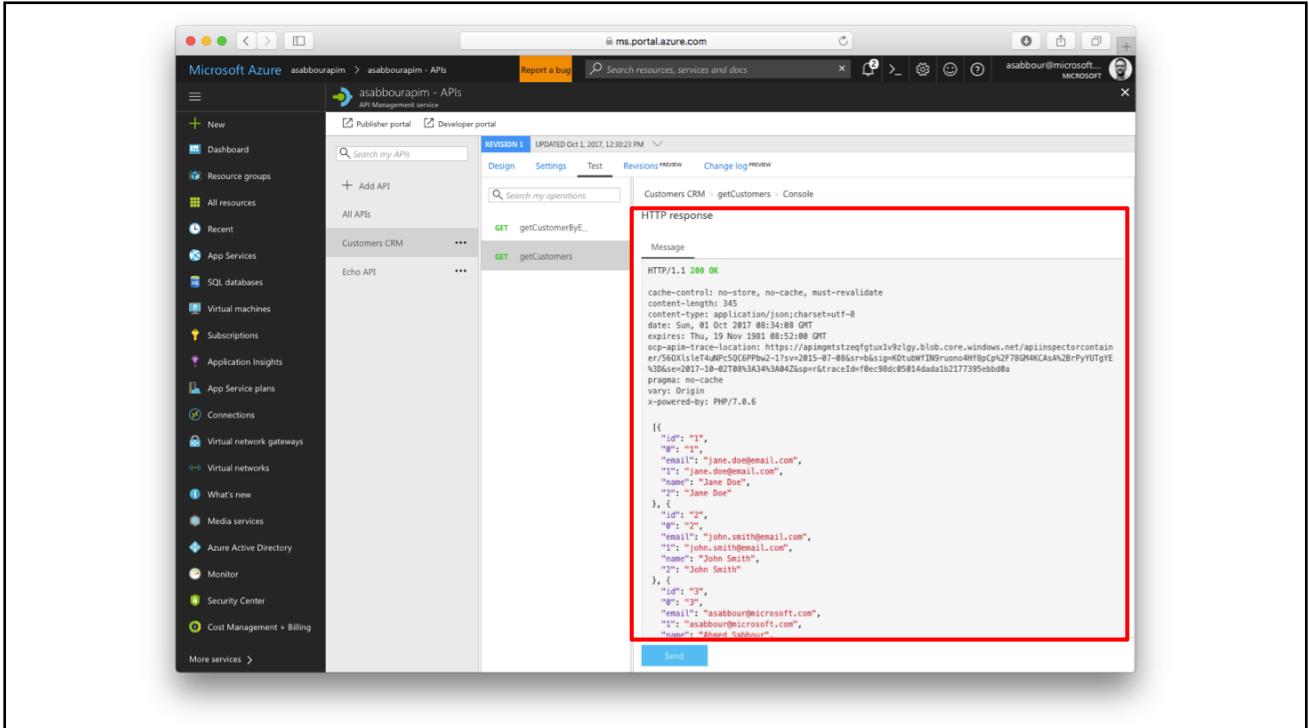
## 10.1 Import Feedback API using Swagger

- Import using OpenAPI specification
- Point towards  
[https://\[alias\]feedbackapi.azurewebsites.net/swagger.json](https://[alias]feedbackapi.azurewebsites.net/swagger.json)
- Give the API a suffix, e.g: **crm**
- Wait for a few seconds, the interface should populate
- Put the API in the **Unlimited** product
- Once imported, change the Web Service URL to  
[https://\[alias\]feedbackapi.azurewebsites.net](https://[alias]feedbackapi.azurewebsites.net)
- Test the API



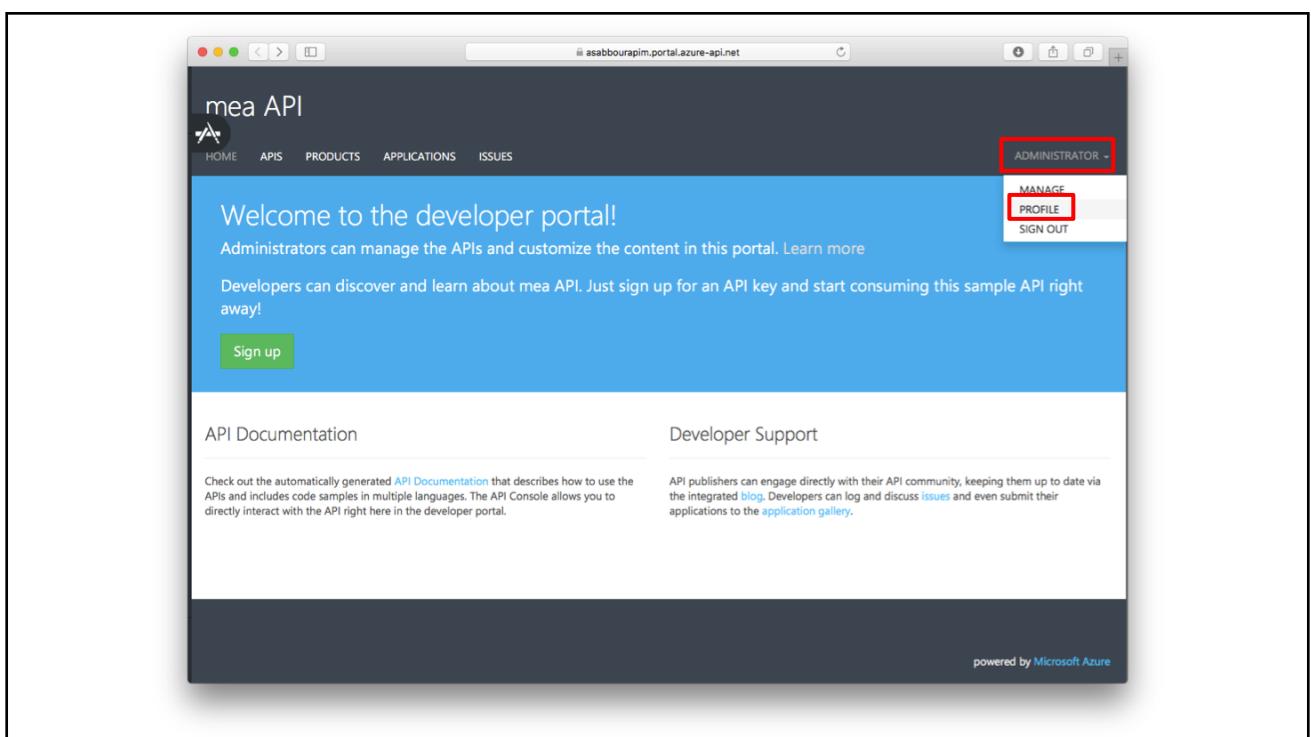
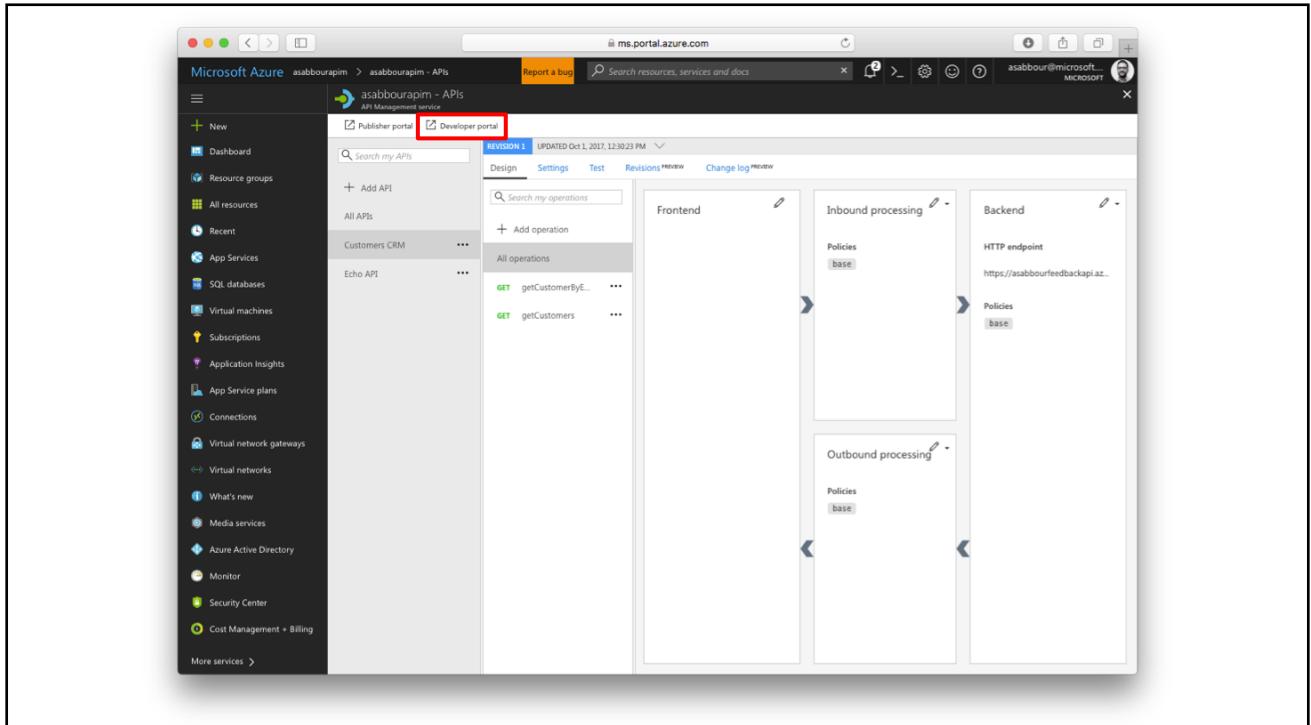
The screenshot shows the Microsoft Azure portal interface for managing APIs. The left sidebar lists various Azure services like App Services, SQL databases, and Virtual machines. The main area displays the 'asabbourapim - APIs' section. A specific API named 'Customers CRM' is selected. The 'Settings' tab is active, highlighted with a red box. The 'General' section contains fields for Name (customers-crm), Display name (Customers CRM), and Description (This is a sample legacy API for a fictional CRM). The 'Web service URL' field is set to <https://asabbourfeedbackapi.azurewebsites.net>, also highlighted with a red box. Under 'Security', the 'User authorization' dropdown is set to 'None'. At the bottom are 'Save' and 'Discard' buttons.

The screenshot shows the 'Test' tab for the 'getCustomers' operation of the 'Customers CRM' API, highlighted with a red box. The 'Query parameters' section shows three parameters: 'Cache-Control' (no-cache), 'Ocp-Apim-Tracing' (true), and 'Ocp-Apim-Subs' (\*\*\*\*\*). The 'Headers' section includes a 'Subscription key' dropdown set to 'Primary-596'. The 'Request URL' is listed as <https://asabbourapim.azure-api.net/api/customers>. A large blue 'Send' button at the bottom is also highlighted with a red box.



## 10.2 Obtain the subscription key for Unlimited product

- Go to the **Developer Portal**
- Click on **Administrator -> Profile**
- Copy the **Primary key** of the **Unlimited** subscription and keep it aside, you'll use it in the subsequent steps.



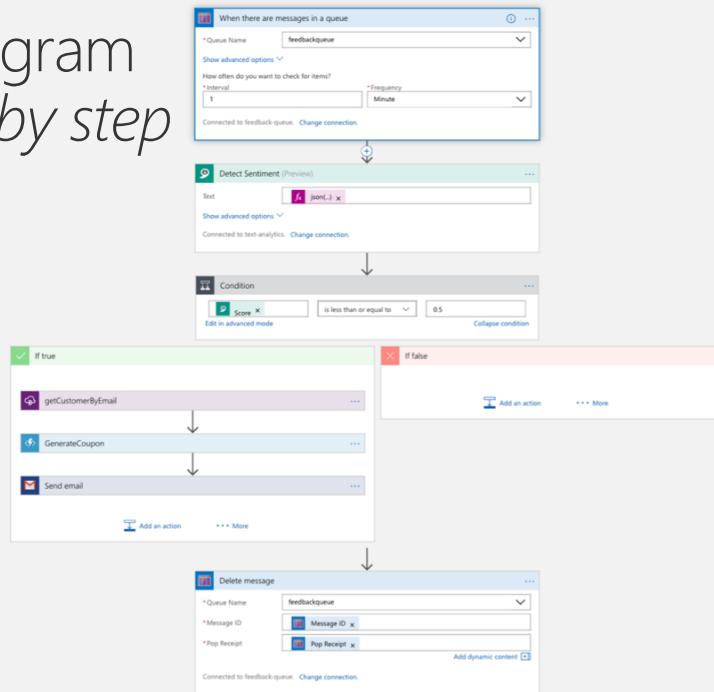
The screenshot shows the 'mea API' portal interface. At the top, there's a navigation bar with links for HOME, APIS, PRODUCTS, APPLICATIONS, and ISSUES, and a dropdown for ADMINISTRATOR. Below the navigation is a 'Profile' section with account details: Email (asabbour@microsoft.com), Organization name (mea), and Notifications sender email (apimgmt-noreply@mail.windowsazure.com). A 'Change account information' button is available. To the right is a 'Analytics reports' button. The main content area is titled 'Your subscriptions'. It lists two subscriptions:

Subscription name	Product	State	Action
Starter (default)	Rename Show   Regenerate Show   Regenerate	Starter	Active <a href="#">Cancel</a>
Unlimited (default)	Rename Hide   Regenerate Show   Regenerate	Unlimited	Active <a href="#">Cancel</a>

Below the subscriptions is a 'Your applications' section with a 'Register application' button. It shows a table with columns for Name, Category, and State, which displays 'No results found.'

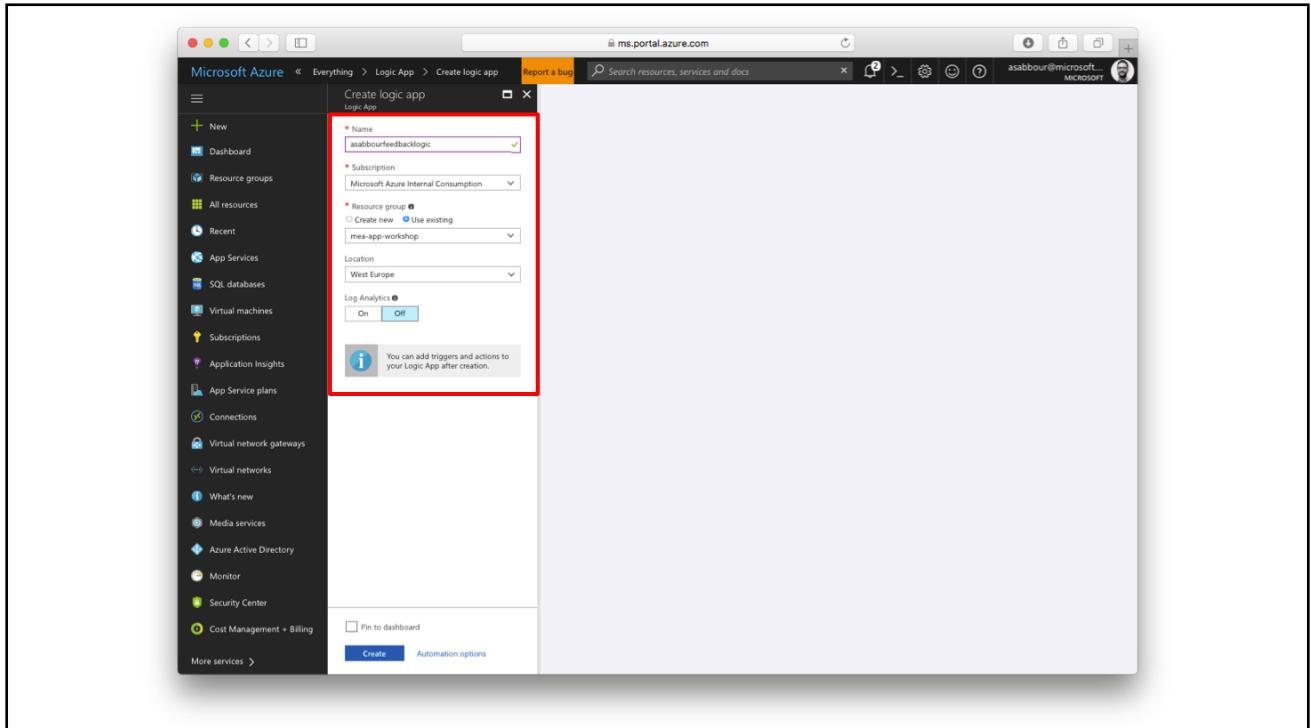
Integrating the solution using  
Logic Apps

Full Logic App diagram  
You'll build it step by step



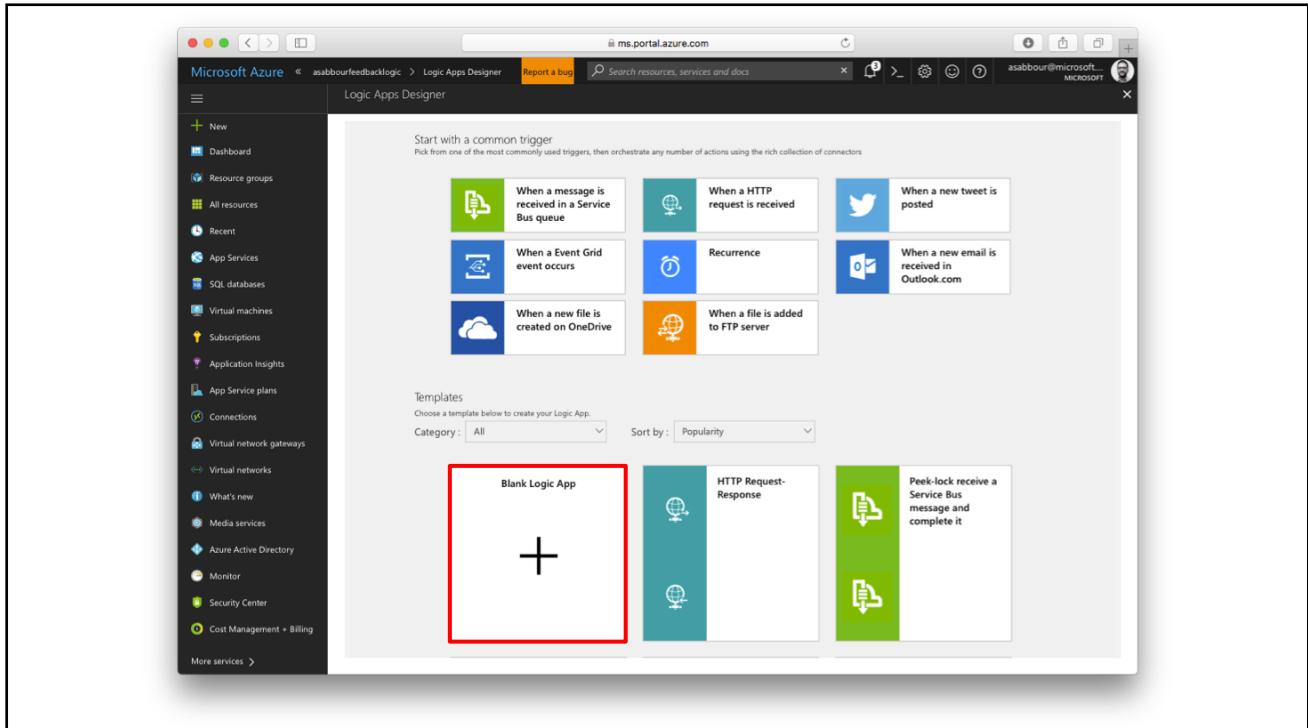
## 11. Create Logic App

- Give it a unique name, like **[alias]feedbacklogic**
- Choose the same resource group and region.



## 12.1 Create the workflow

- Open the Logic Apps Designer
- Start from a **Blank Logic App** template

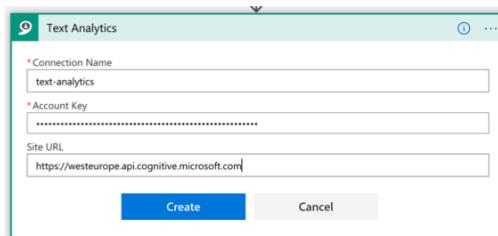


## 12.2 Add queue trigger

- Add a new Queue trigger, pointing to your storage account created earlier

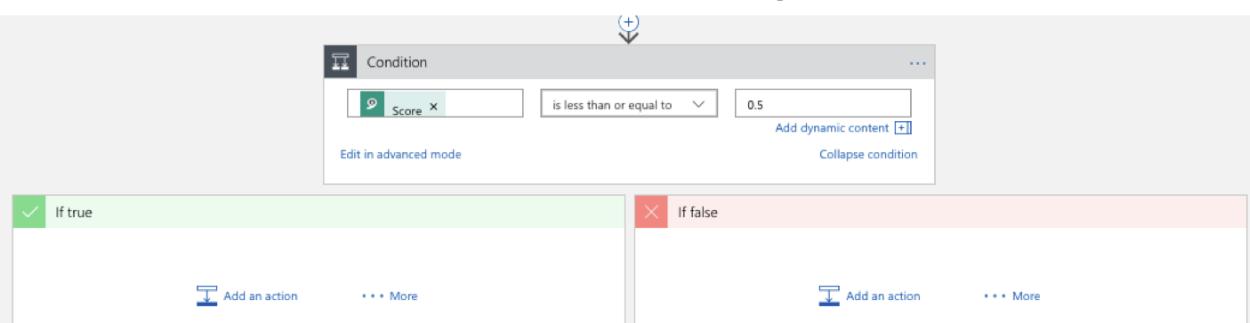
## 12.2 Add Text Analytics action

- Add a Text Analytics action to detect sentiment
- Provide the Cognitive Services account name and key
- Put the endpoint as the Site URL  
`https://[region].api.cognitive.microsoft.com`
- Configure the text field to be  
`@{json(triggerBody())?['MessageText']).feedback}`



## 12.3 Add condition, when the sentiment is negative

- When the **score** is **less than or equal to 0.5**
  - This is a negative sentiment
  - When this is true, add the next action to get customer details



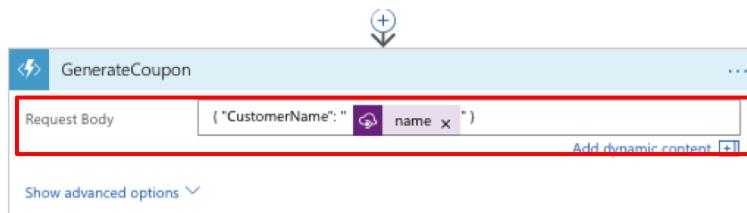
## 12.4 Add API Management action

- Query API Management for customer name by email
- Use this as the value of the email field, it will extract it out of the message on the queue:  
`@{json(triggerBody())?['MessageText'].email}`
- Provide the API Management subscription key



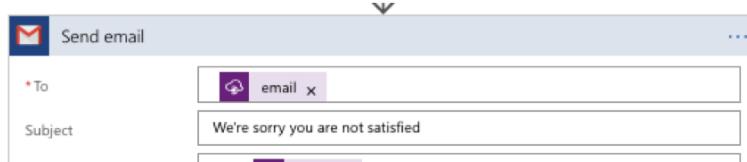
## 12.5 Add Generate Coupon Azure Function action

- Call the Generate Coupon function, passing in the customer name returned from the API Management call



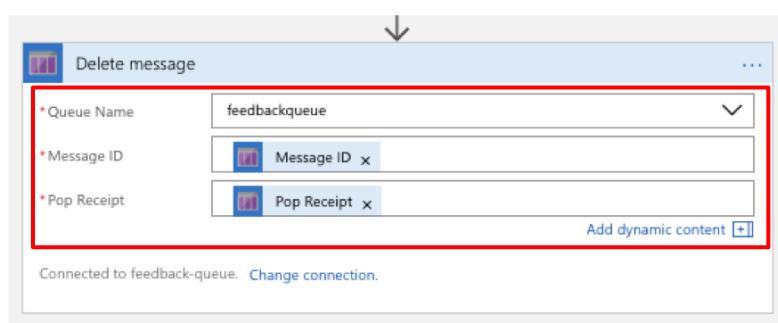
## 12.6 Add Email action

- Use your favorite email action to send an email to the customer with the coupon link
- Configure it with the email, name and Coupon URL obtained from the previous actions.
- You may use this as the body of the email action:  
`Dear @{body('getCustomerByEmail')['name']}, we want to make it up to you. Here is a coupon: @{body('GenerateCoupon').CouponUrl}`

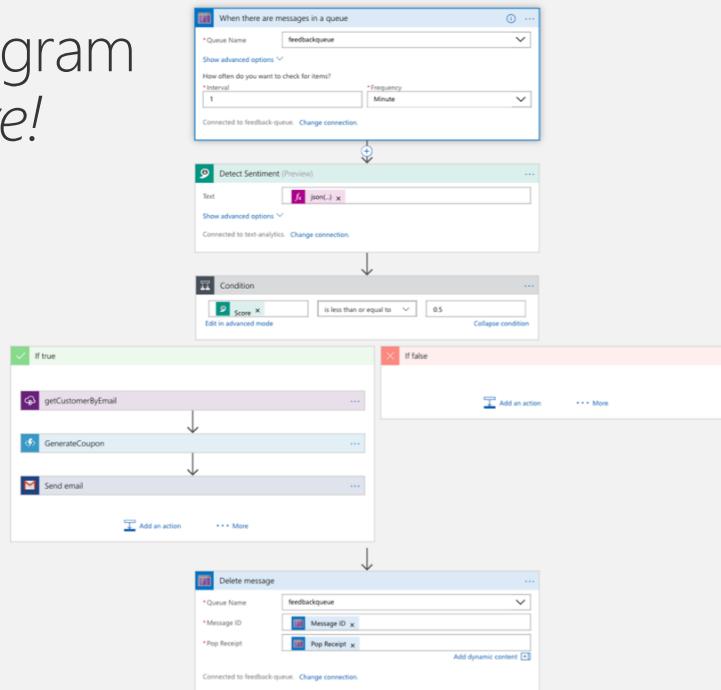


## 12.7 Delete the message from the queue

- Now that you've processed this message, delete it from the queue to avoid processing it again. Configure with queue name, message ID and pop receipt ID from the original trigger.



Full Logic App diagram  
*don't forget to save!*



Testing the solution

Provide your feedback

We're eager to know about what you think!

Start Stop Status:  
Your company is the worst!

asabbour@microsoft.com

Submit

© 2017 - customerfeedback\_web

```

graph TD
    A[When there are messages in a queue] --> B[Detect Sentiment]
    B --> C[Condition]
    C -- "true" --> D[getCustomerByEmail]
    D --> E[GenerateCoupon]
    E --> F[Send email]
    F --> G[Delete message]
    C -- "false" --> G
  
```

We're sorry you are not satisfied

S sabbOur@gmail.com <sab...>

Ahmed Sabbour

Sunday, October 1, 2017 at 2:48 PM

Show Details

LinkedIn

Dear Ahmed Sabbour, we want to make it up  
<https://asabbourstorage.blob.core.windows.net/01T10%3A43%3A41Z&se=2017-10-01T11%3A17&sr=b&sig=AVJBqAFMeZXhWCbNw9XX54>

25% off voucher for Ahmed Sabbour

Bonus (CDN): [https://\[alias\]feedbackcdn.azureedge.net/coupons/Ahm...](https://[alias]feedbackcdn.azureedge.net/coupons/Ahm...)