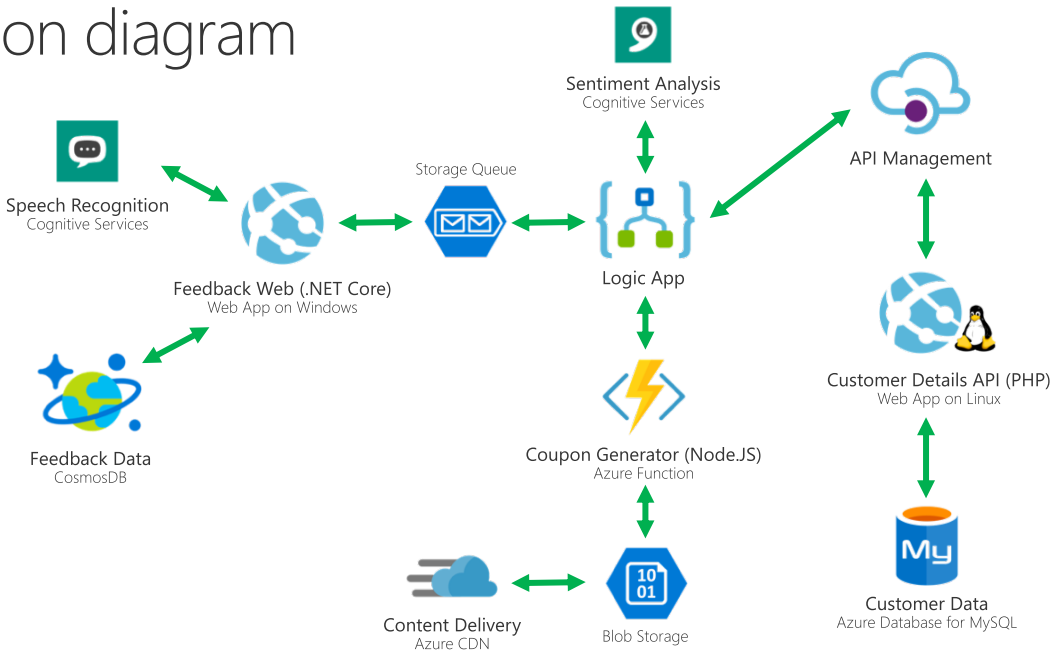
Preparing and deploying the solution

Solution diagram

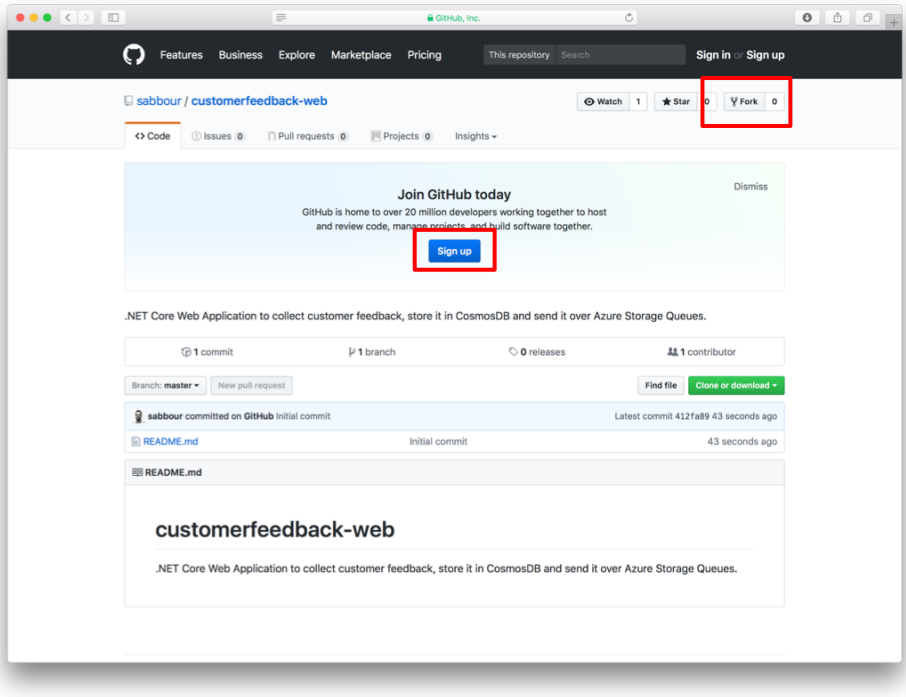


0. Pre-requisites

WiFi Code: VQP2YV

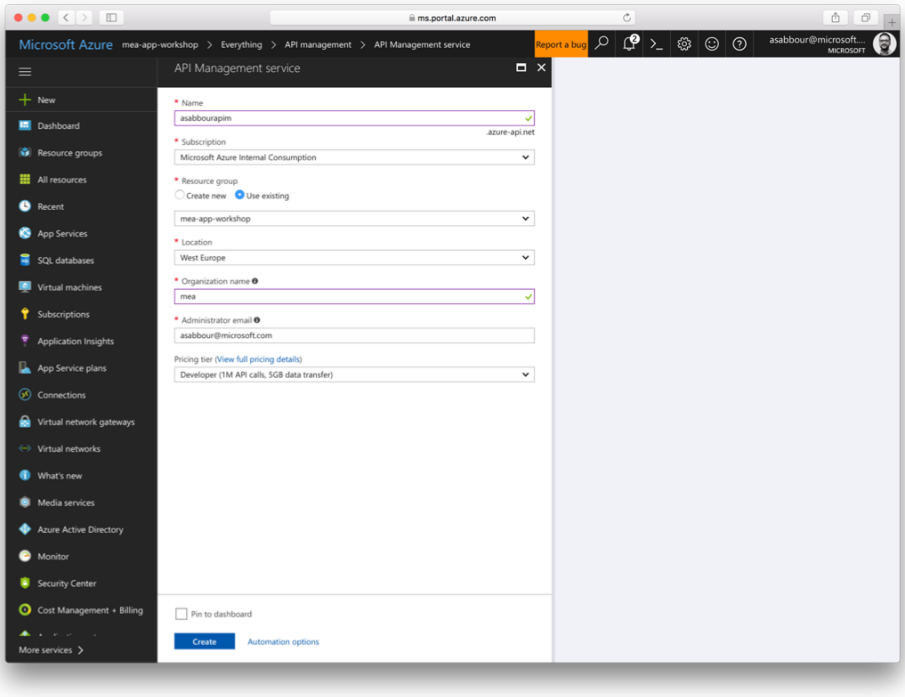
Download the lab guide (if you want)
http://github.com/sabbour/customerfeedback

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`



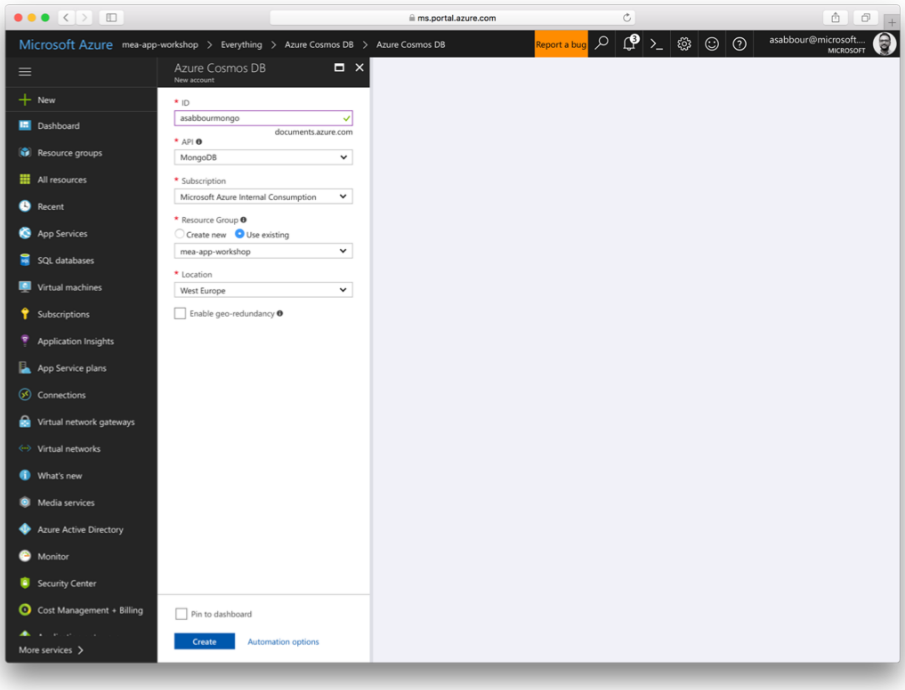
1. Create API Management Gateway

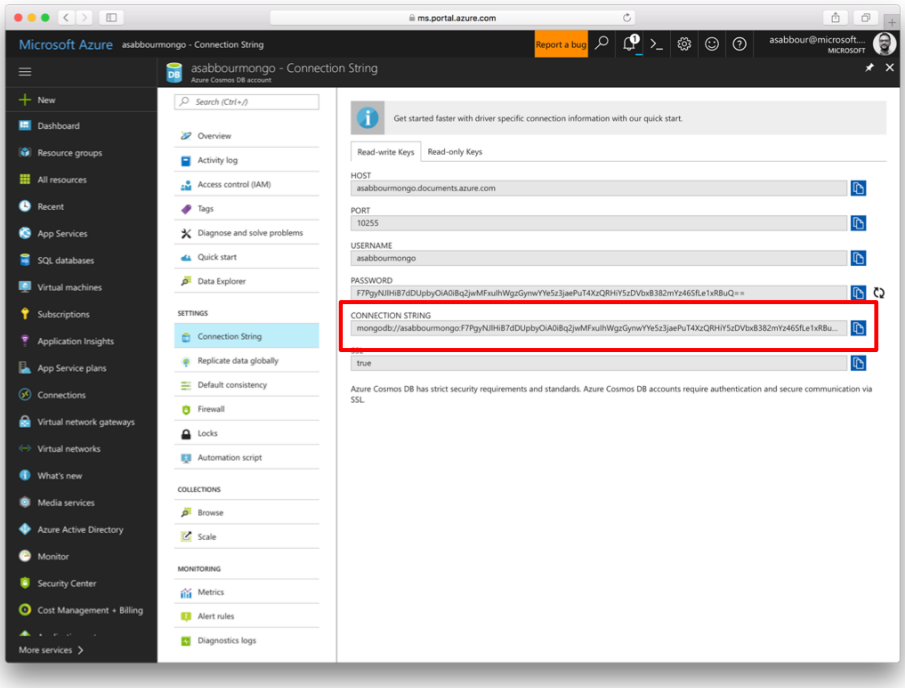
- Give it a unique name, like ***[alias]*apim**
- Choose Developer tier.
- This will take about 30 minutes to provision. Proceed with the next steps.



2. Create CosmosDB

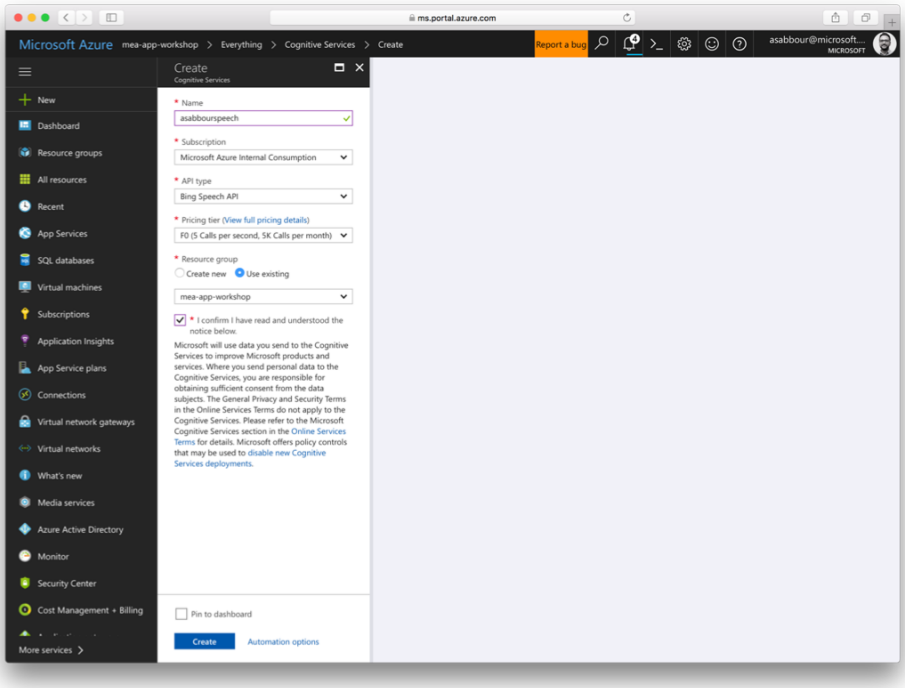
- Give it a unique name, like *[alias]*mongo
- Choose the **MongoDB API**
- This will take about 5 minutes to provision. Proceed with the next steps.
- When the provisioning is done, go back to get the MongoDB connection string and store it aside. You'll need it later.

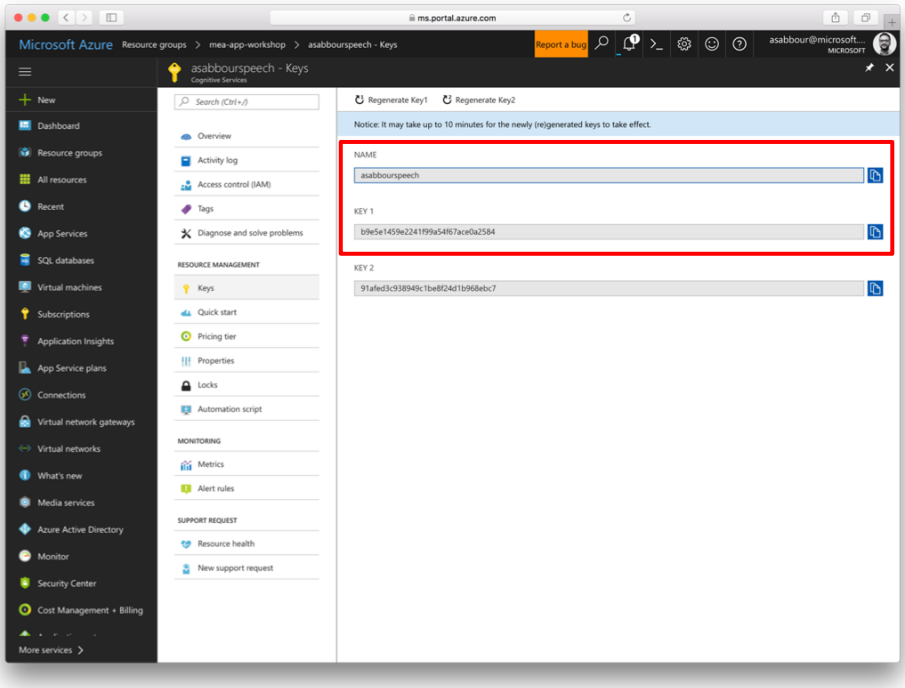




3. Create Bing Speech API from Cognitive Services

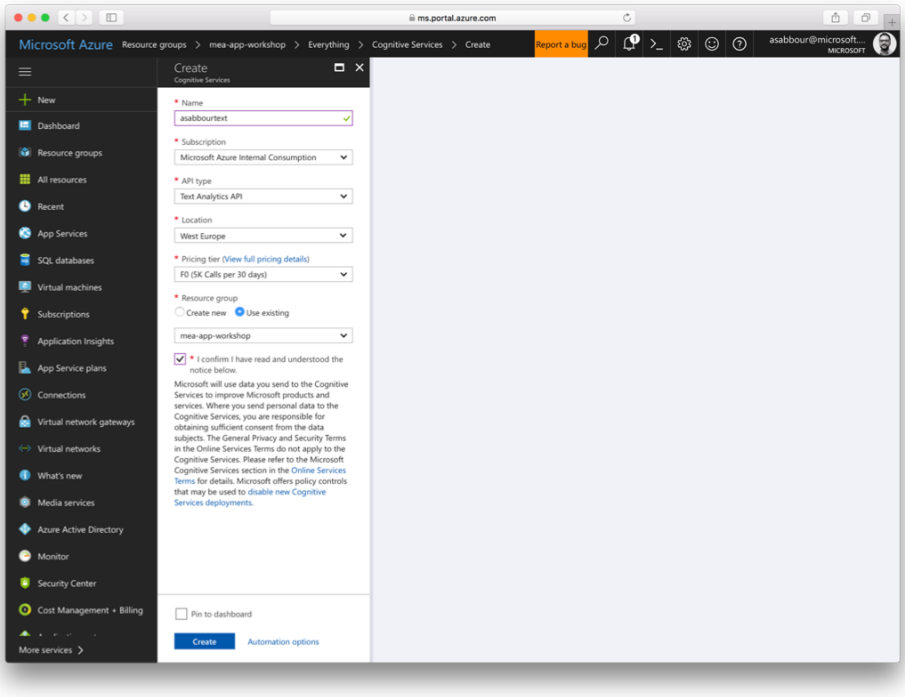
- Give it a unique name, like ***[alias]*speech**
- Choose the Free tier.
- Get the key and store it aside. You'll need it later.

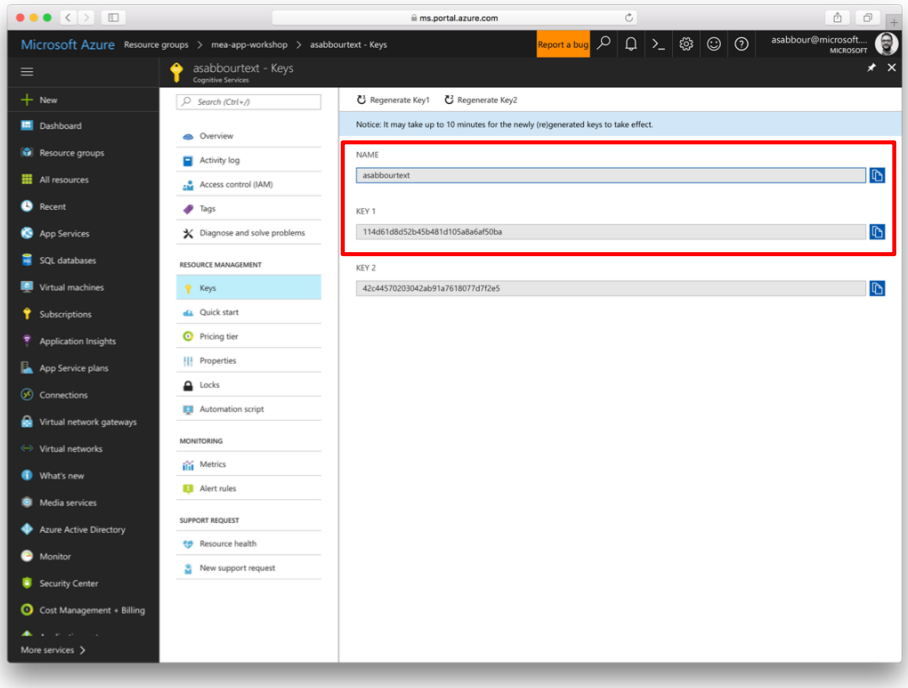




4. Create Text Analytics API from Cognitive Services

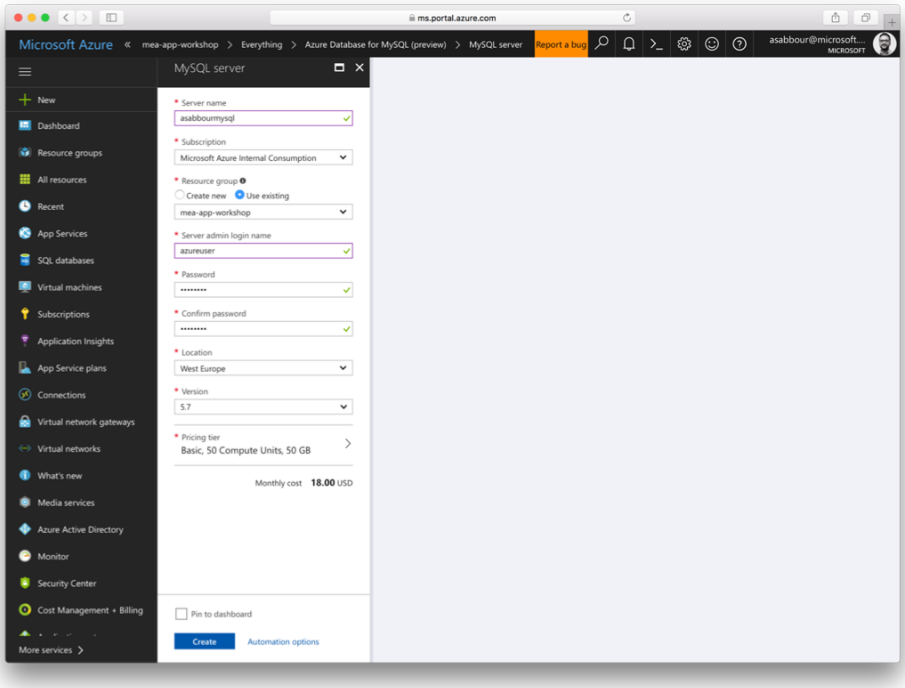
- Give it a unique name, like `[alias]text`
- Choose the Free tier.
- Get the key and store it aside. You'll need it later.

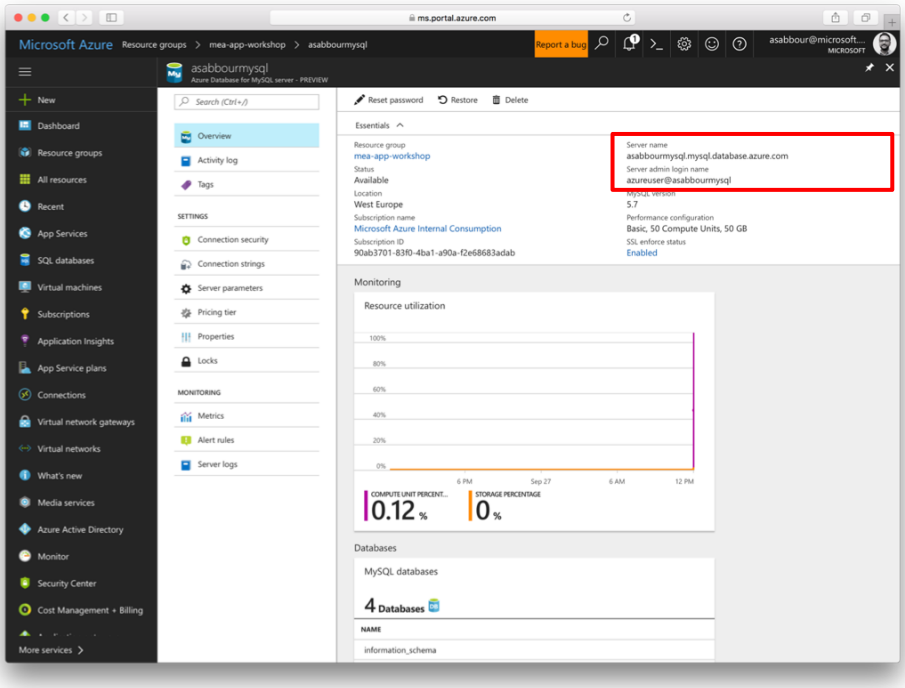


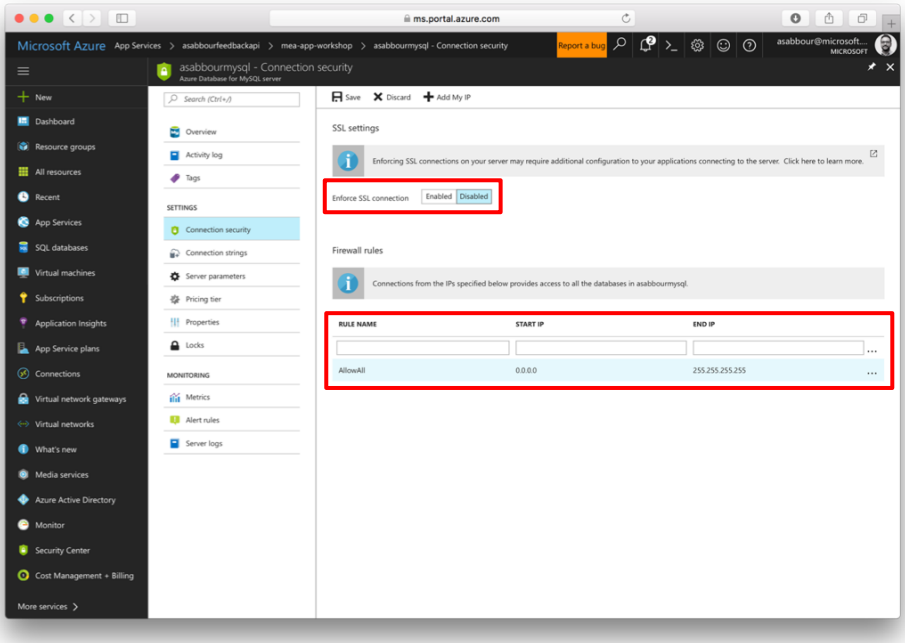


5. Create Azure Database for MySQL

- 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

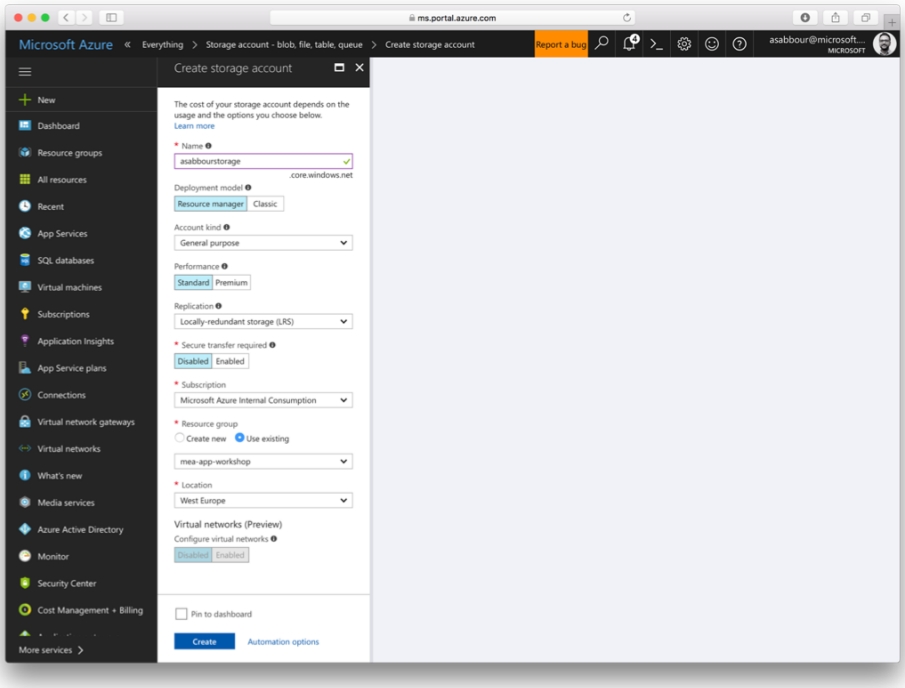


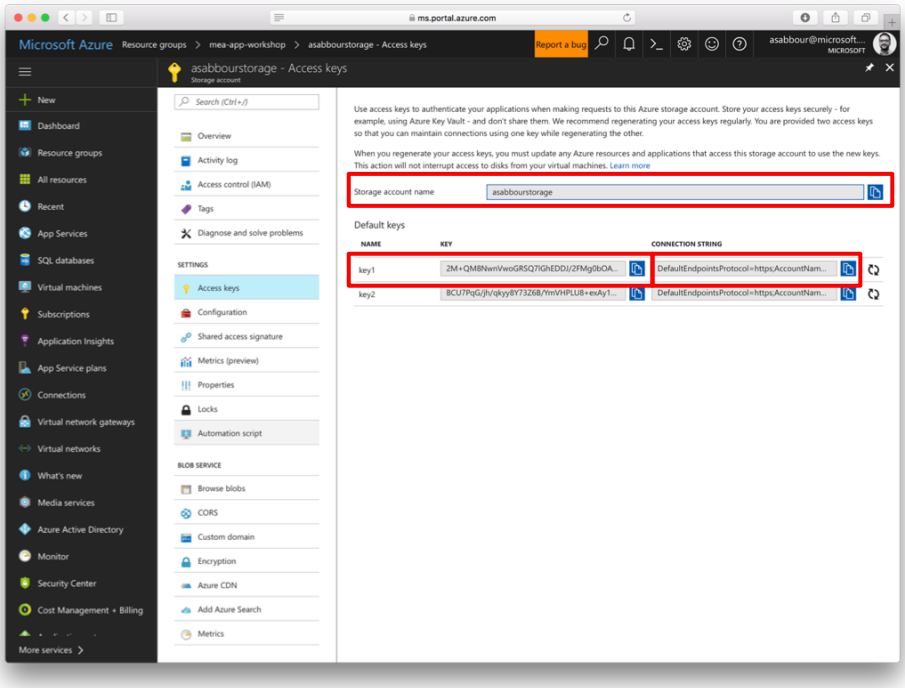


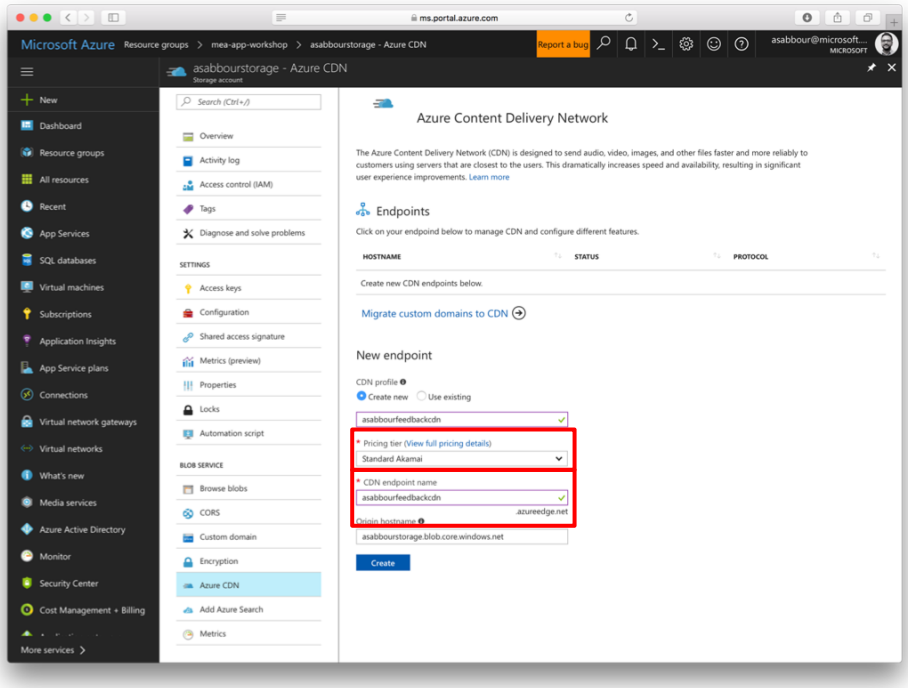


6. Create Storage Account

- 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.

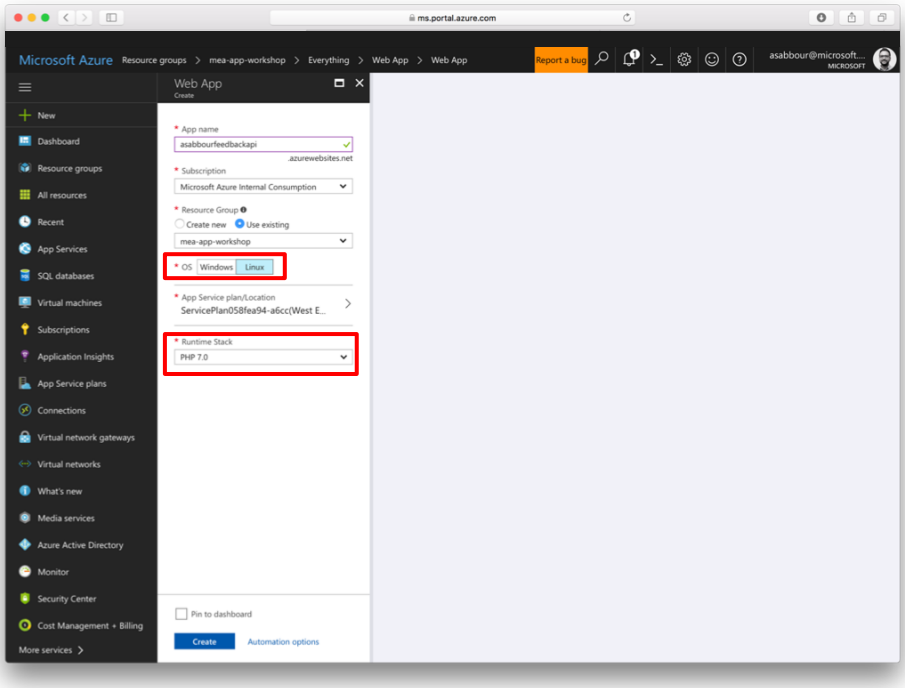






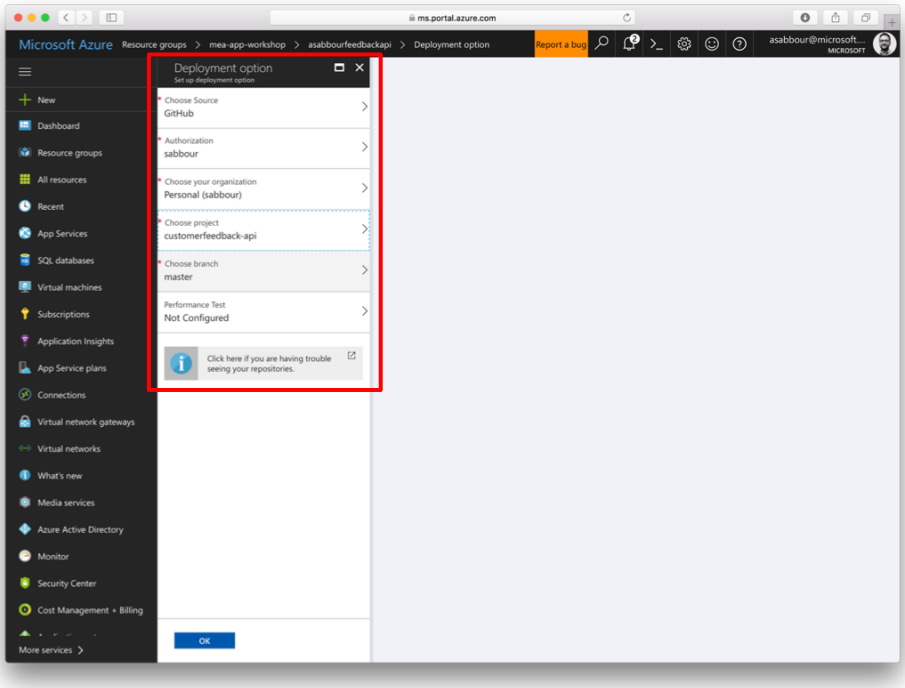
7.1 Create Web App for Feedback API running PHP on Linux

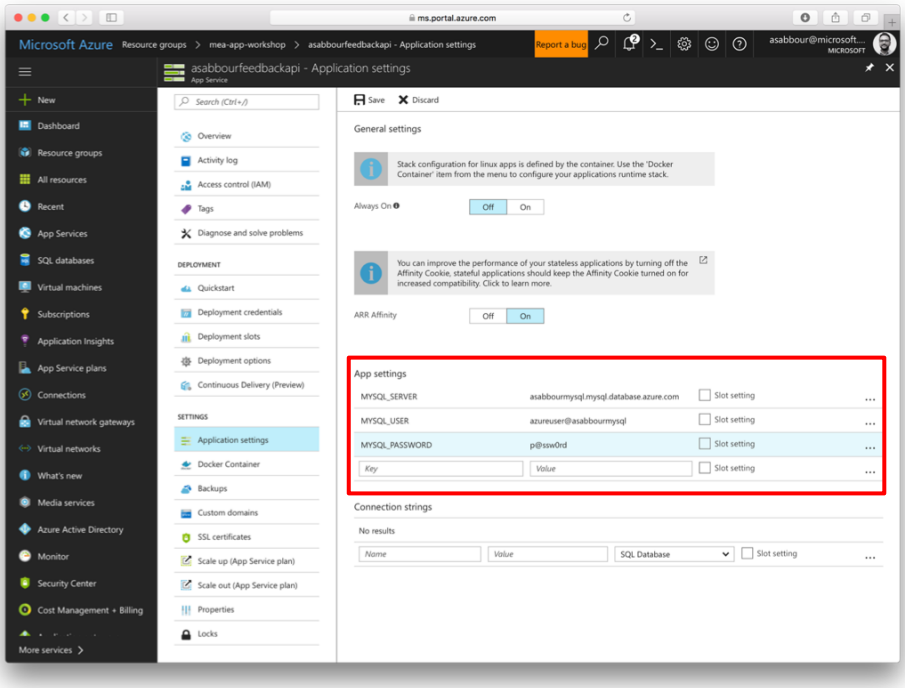
- Give it a unique name, like ***[alias]*feedbackapi**
- Choose **Linux** as the OS.
- Choose **PHP 7.0** as the Runtime Stack.

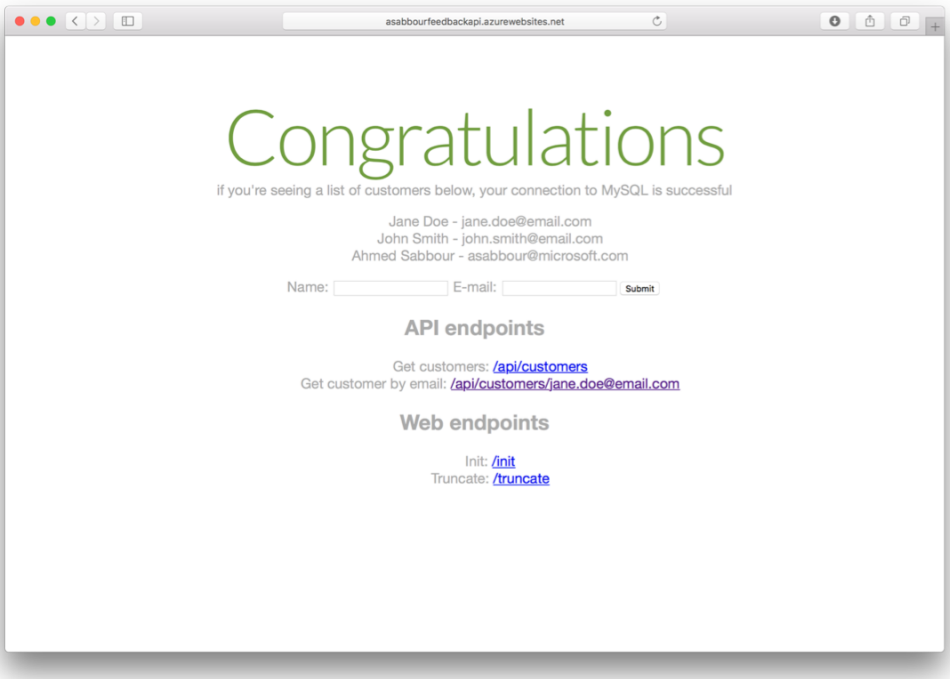


7.2 Configure Feedback API

- 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 you should see some customers already if things are configured properly.
- Add your name and email

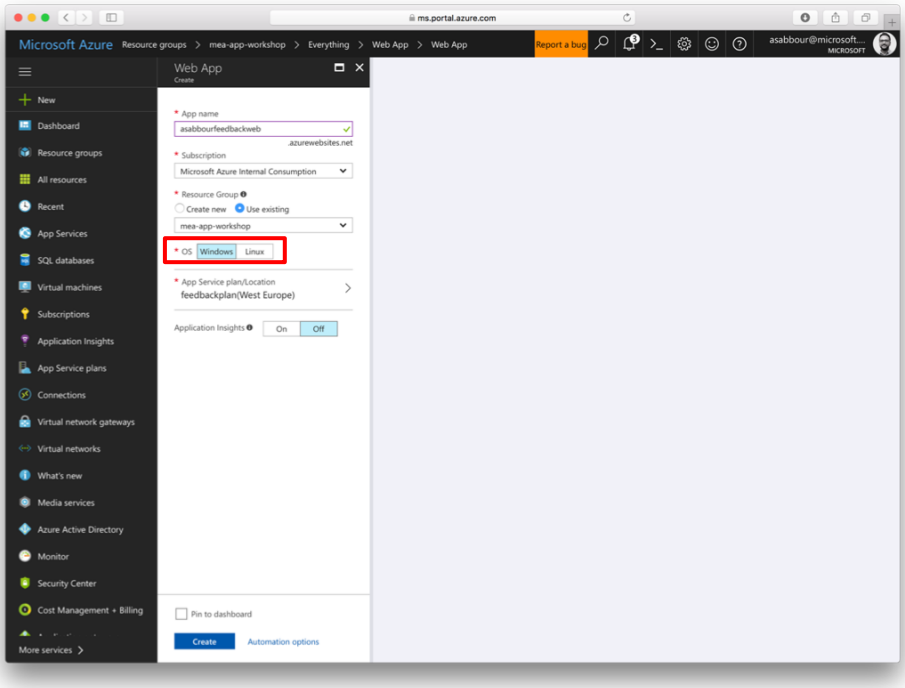






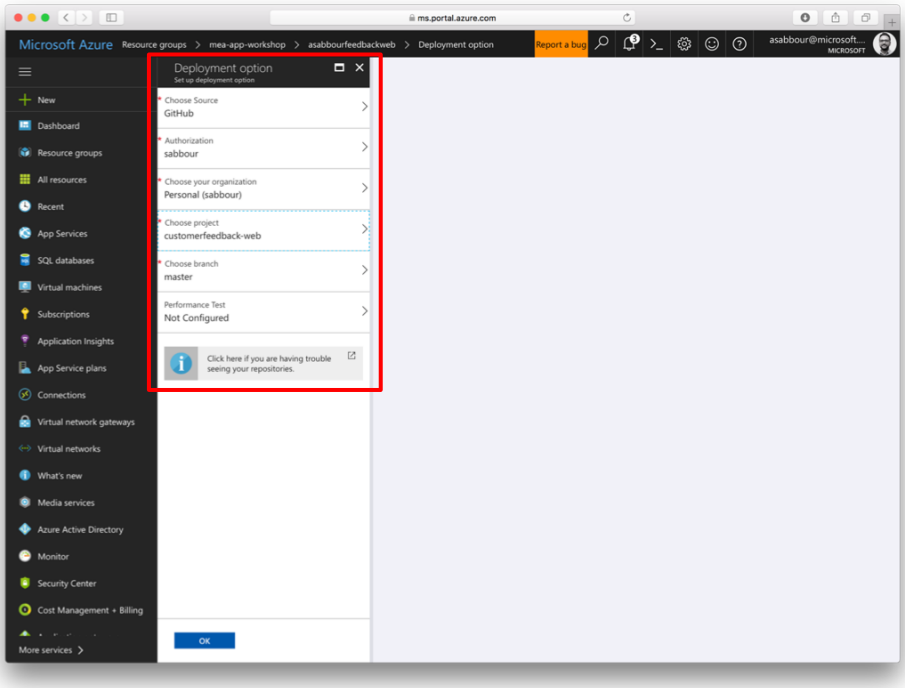
8.1 Create Web App for new Feedback website running .NET on Windows

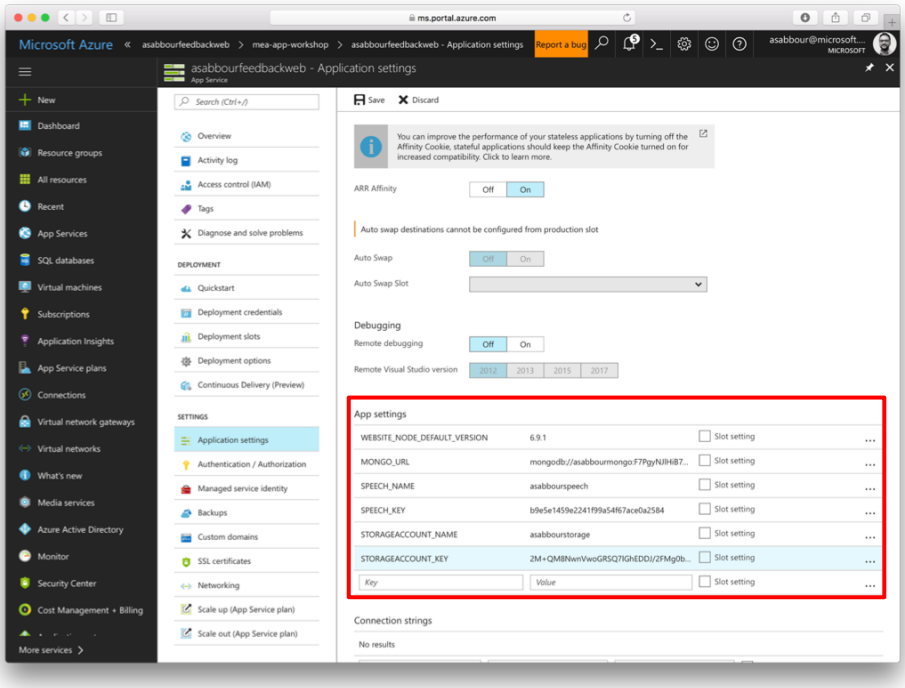
- Give it a unique name, like *[alias]*`feedbackweb`
- Choose **Windows** as the OS.

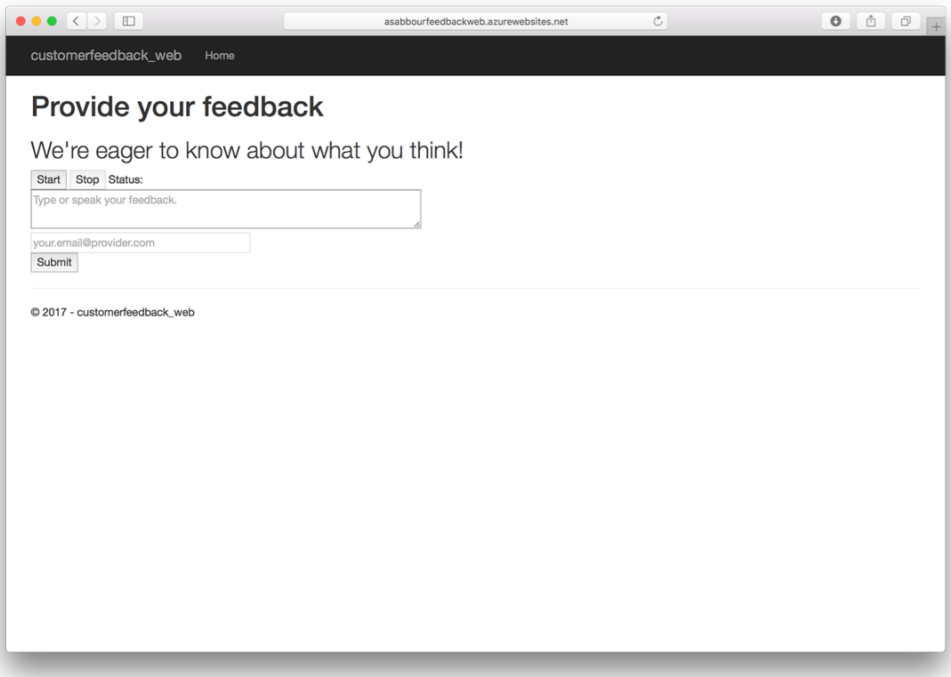


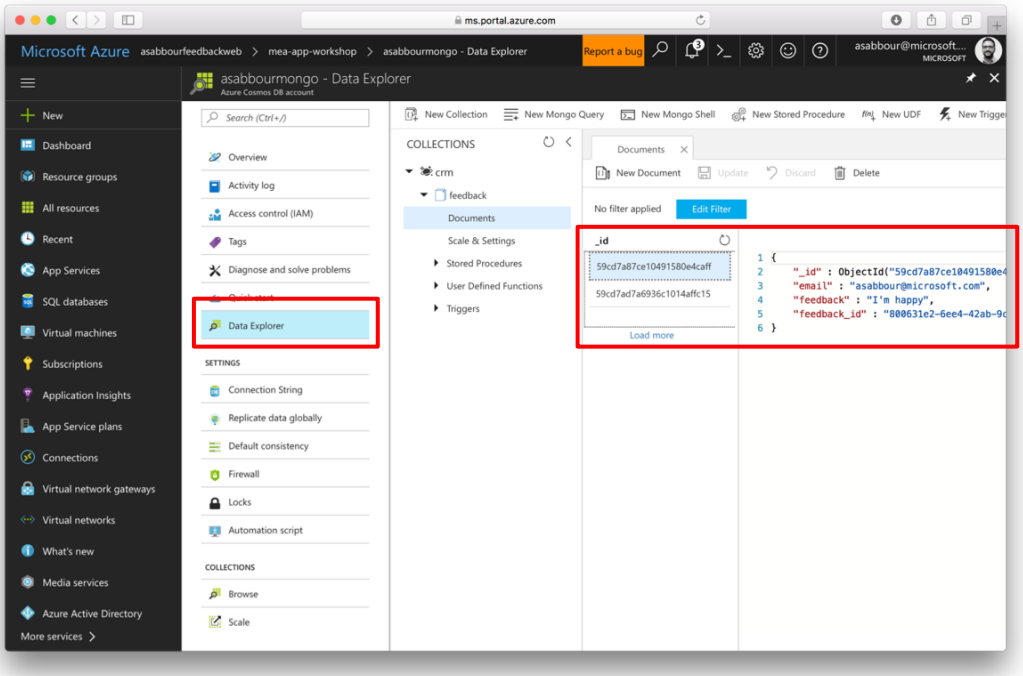
8.2 Configure the new .NET Feedback website

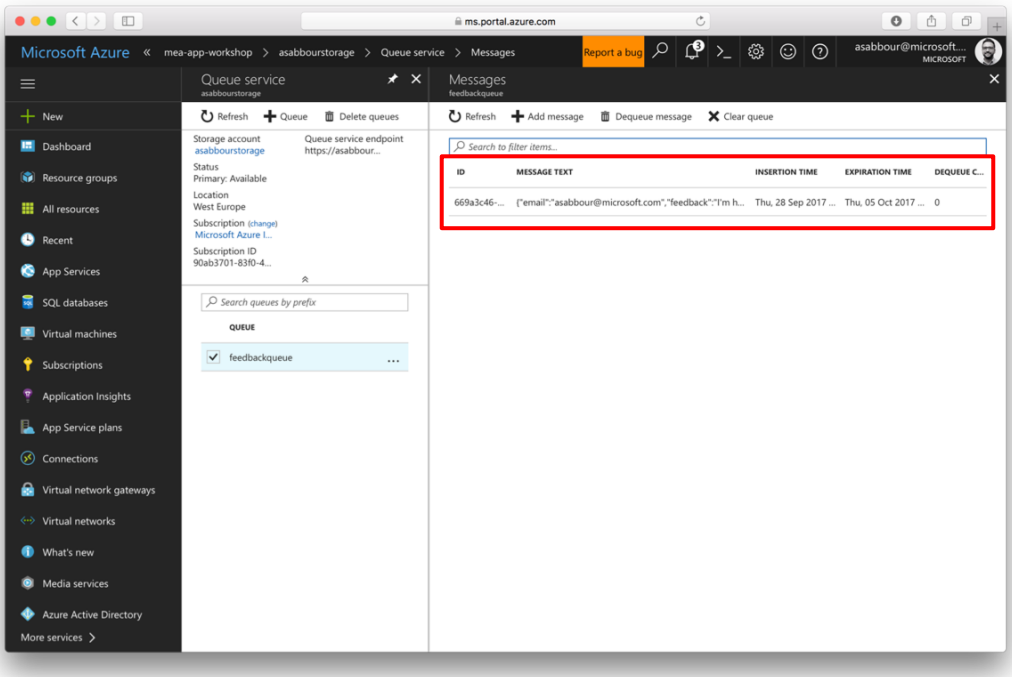
- Setup deployment from your **`customerfeedback-web`** Github fork.
- Create App Settings with the below values:
  - `MONGO_URL`: `mongodb connection string specified earlier`
  - `SPEECH_KEY: speech API key specified earlier`
  - `STORAGEACCOUNT_NAME: storage account name`
  - `STORAGEACCOUNT_KEY: storage account key`
- Go to http://[alias]feedbackweb.azurewebsites.net you should see a form to collect feedback





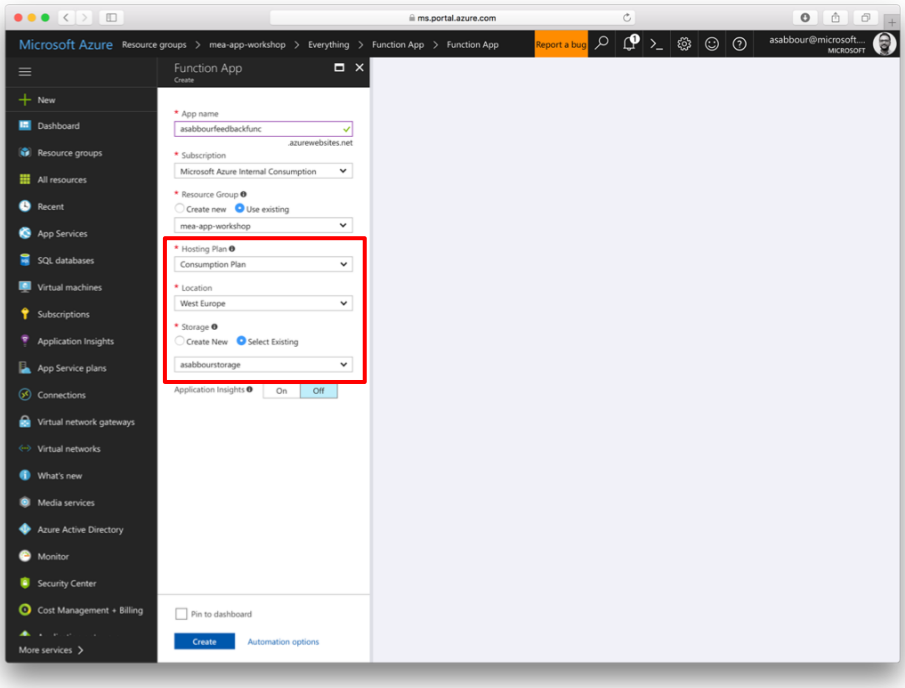






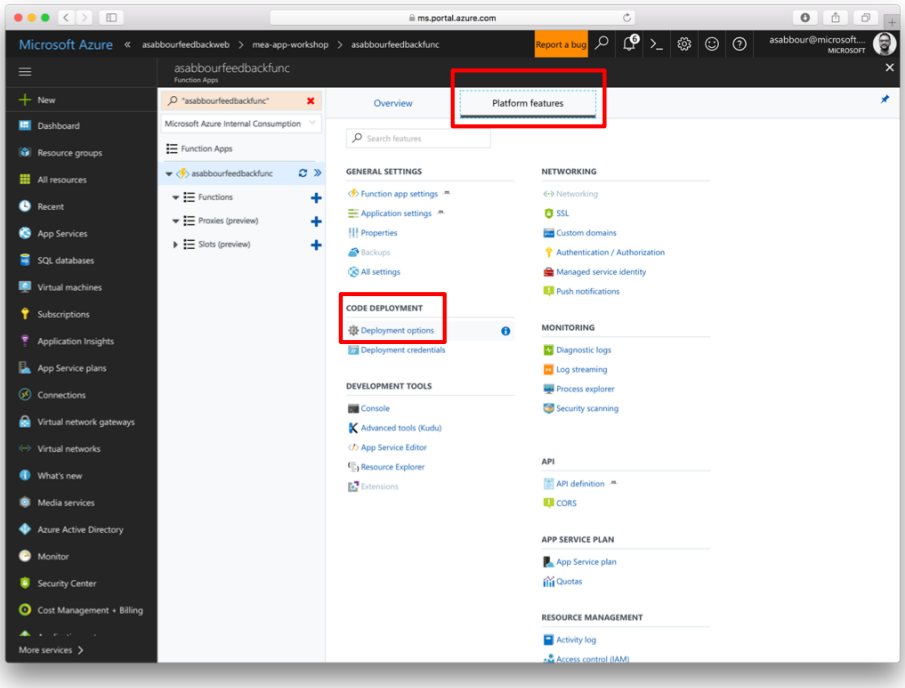
9.1 Create Function App to generate coupons

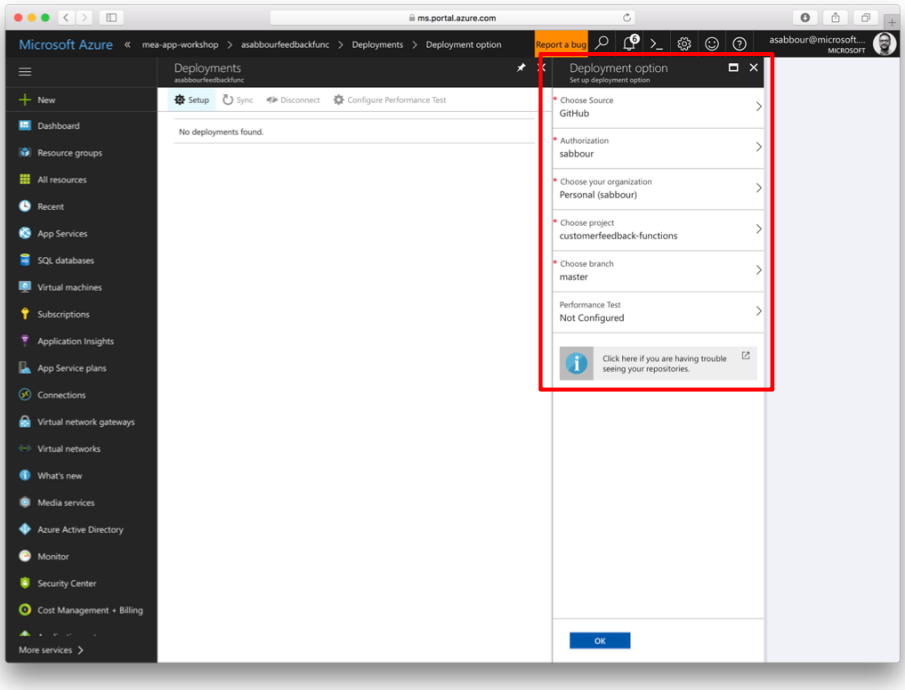
- Use **Consumption Plan** as the hosting plan.
- Choose the same resource group and region.
- Choose the same storage account.

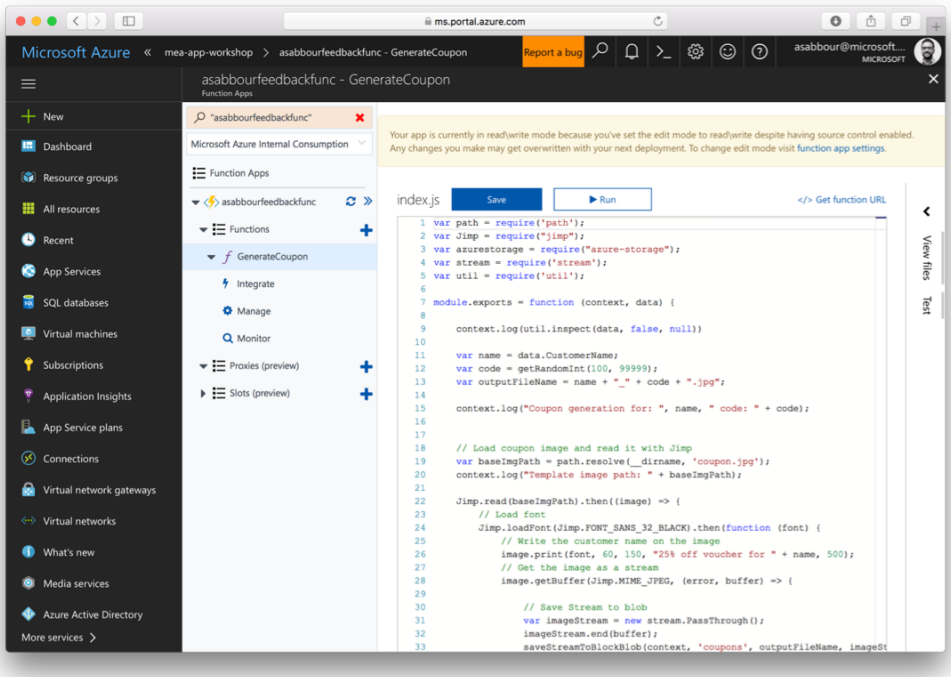


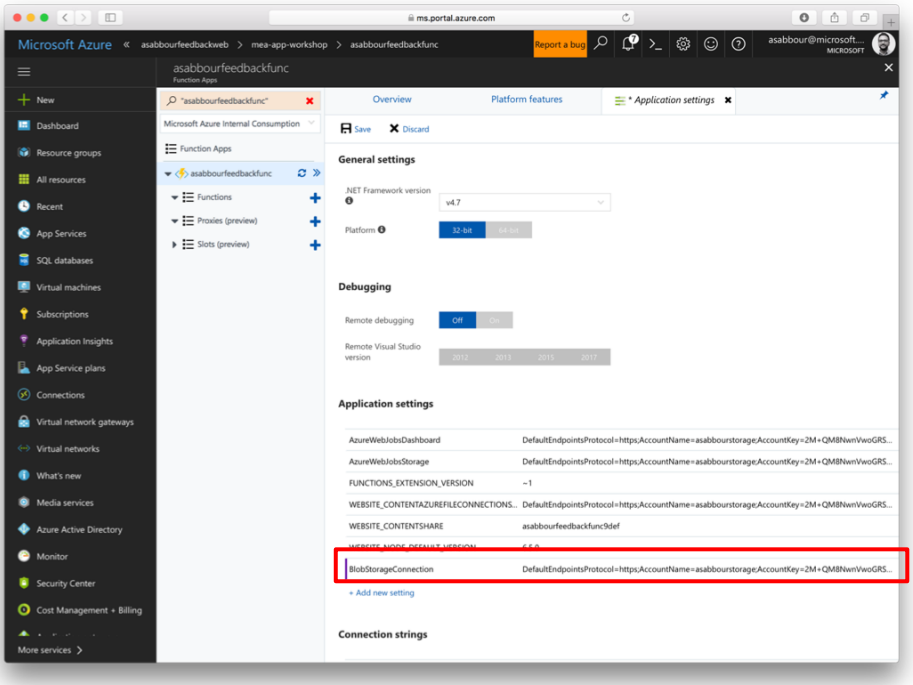
9.2 Configure the Function App

- 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]`





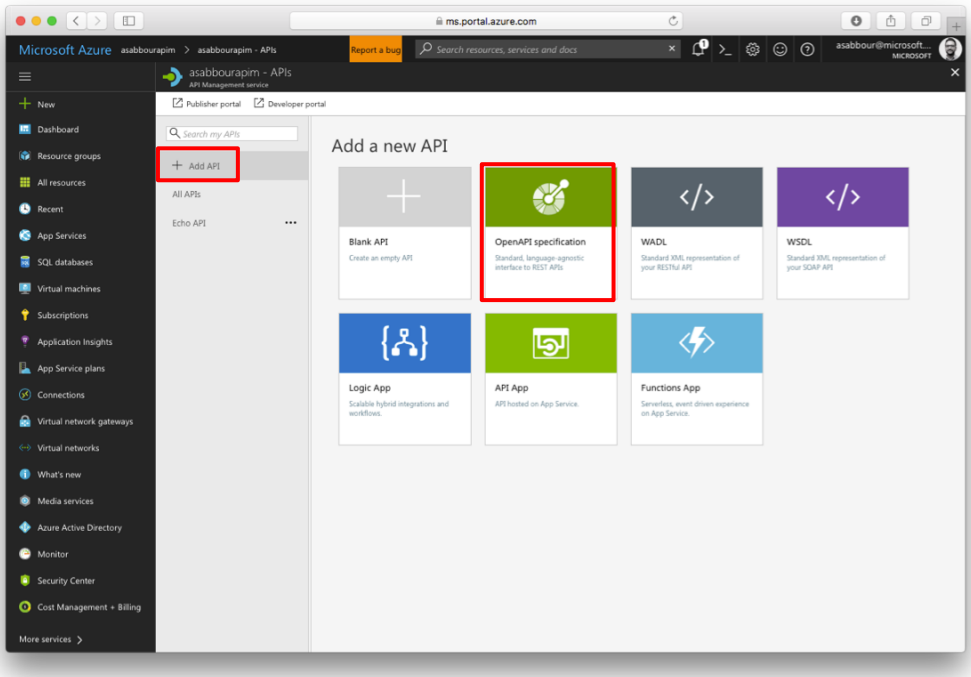


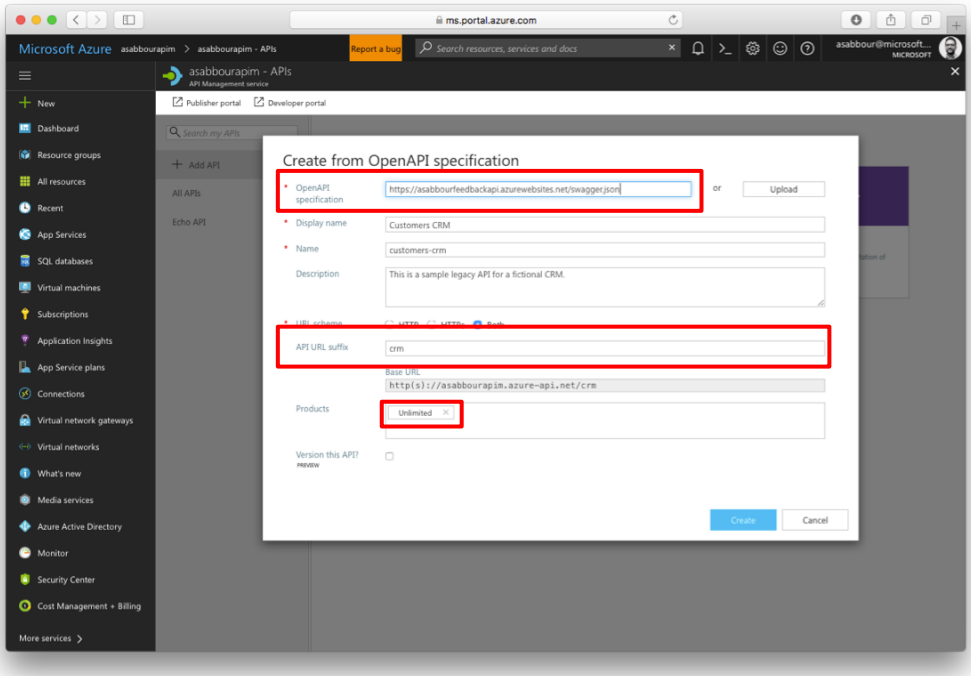


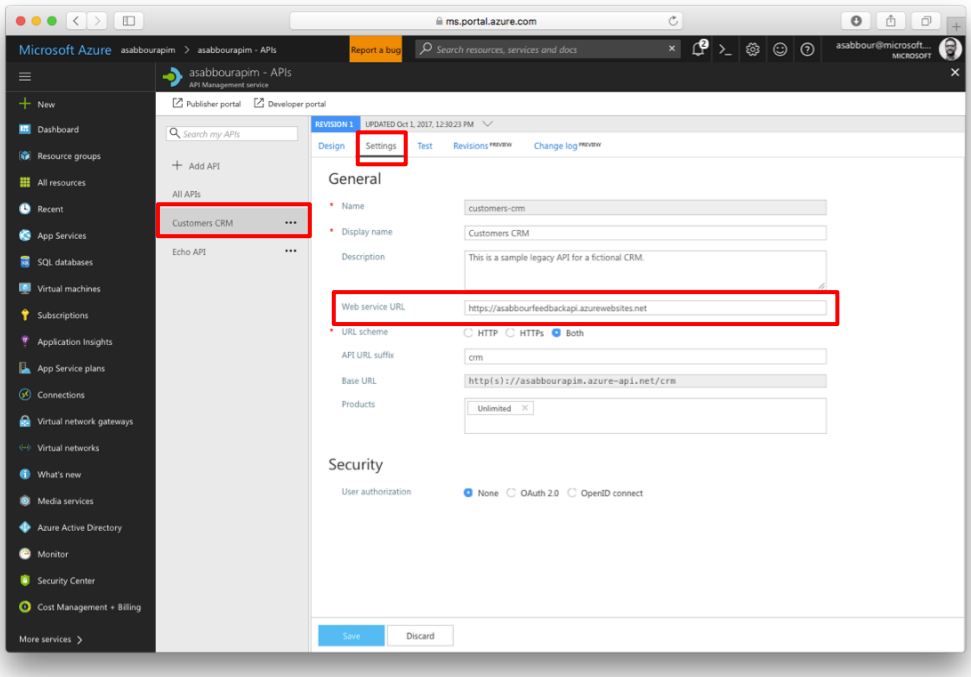
Importing Feedback API into API Management

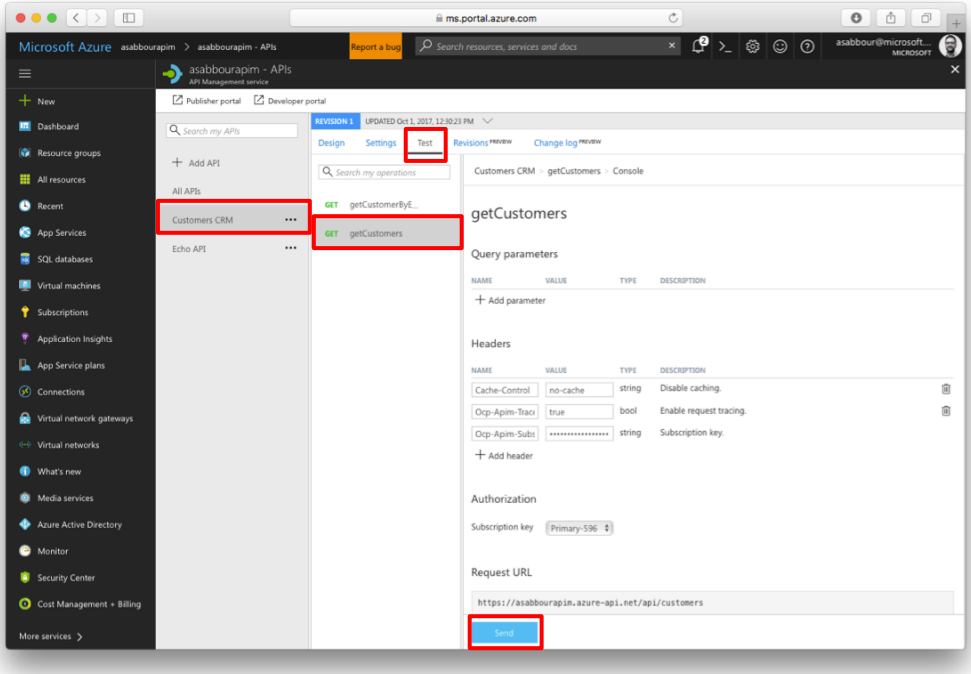
10.1 Import Feedback API using Swagger

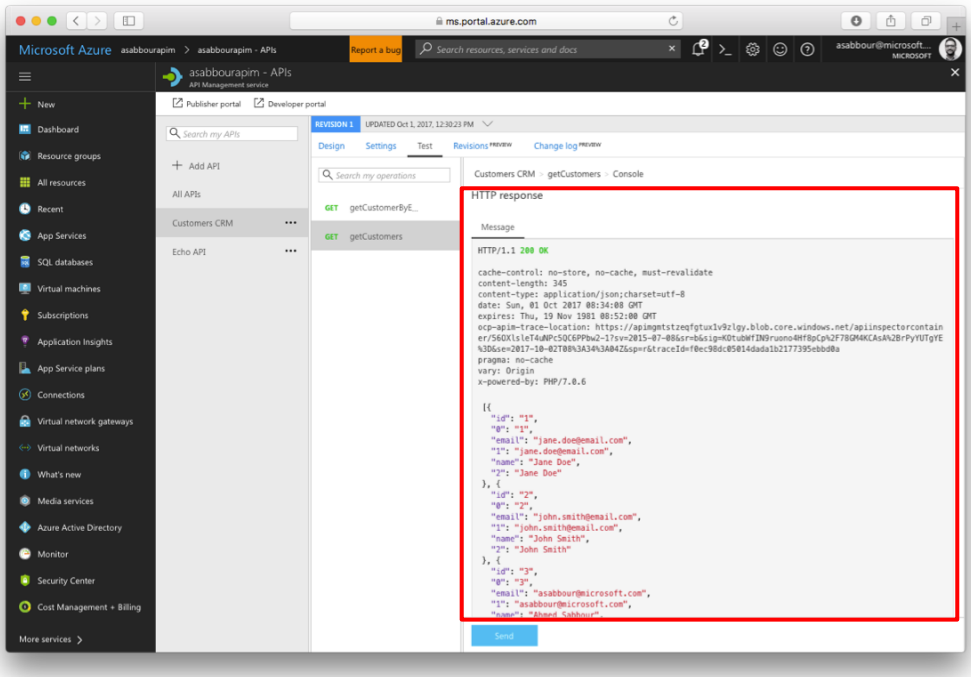
- Import using OpenAPI specification
- Point towards https://[alias]feedbackapi.azurewebsites.net/swagger.json
- Give the API a suffix, e.g: **crm**
- Wait for a few seconds, the interface should populate
- Put the API in the **Unlimited** product
- Once imported, change the Web Service URL to https://[alias]feedbackapi.azurewebsites.net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





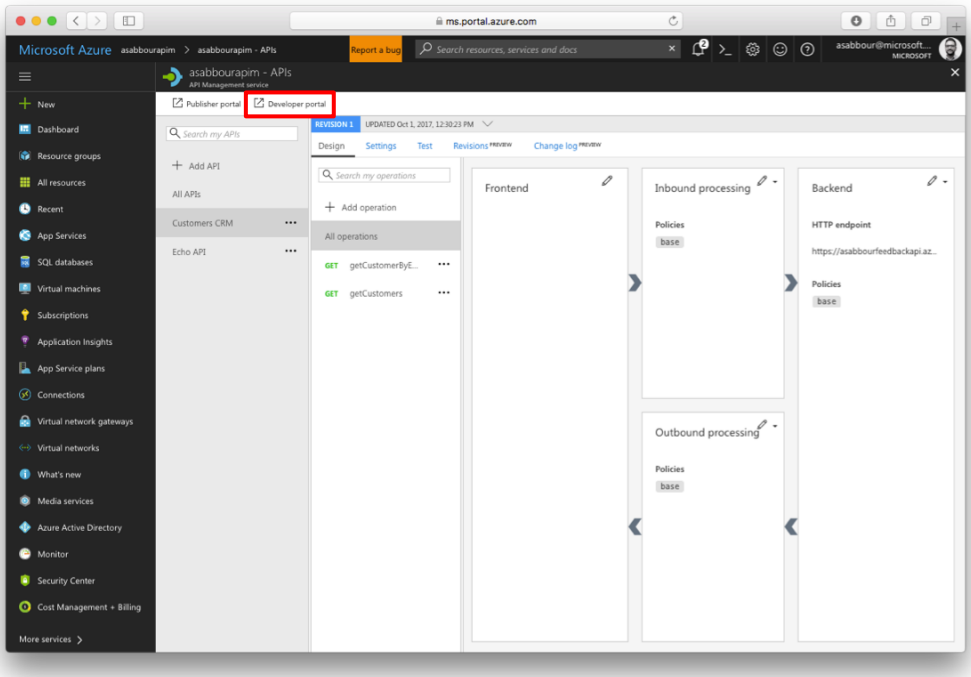


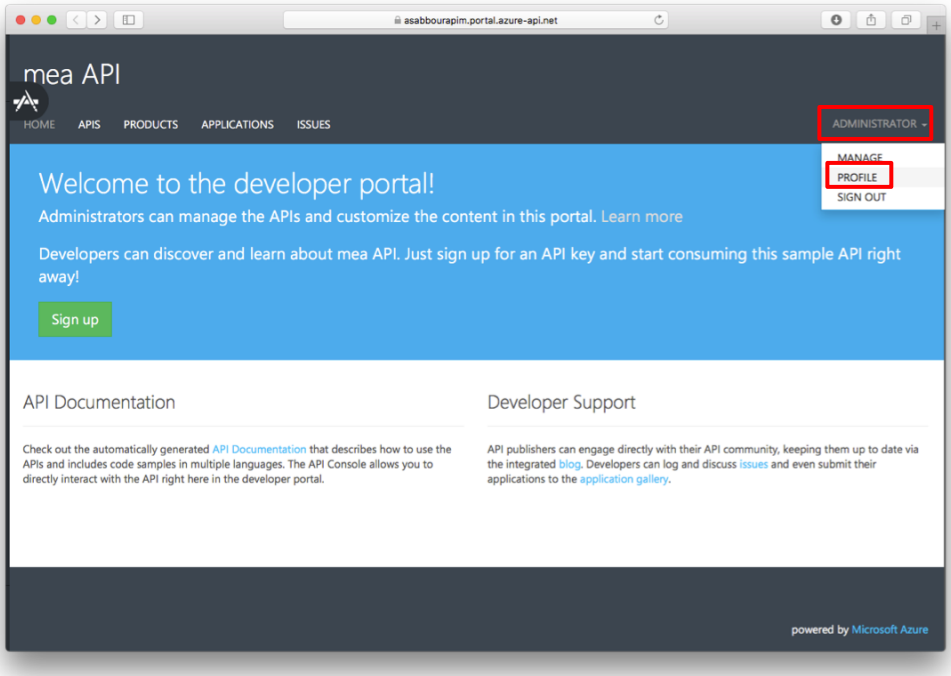


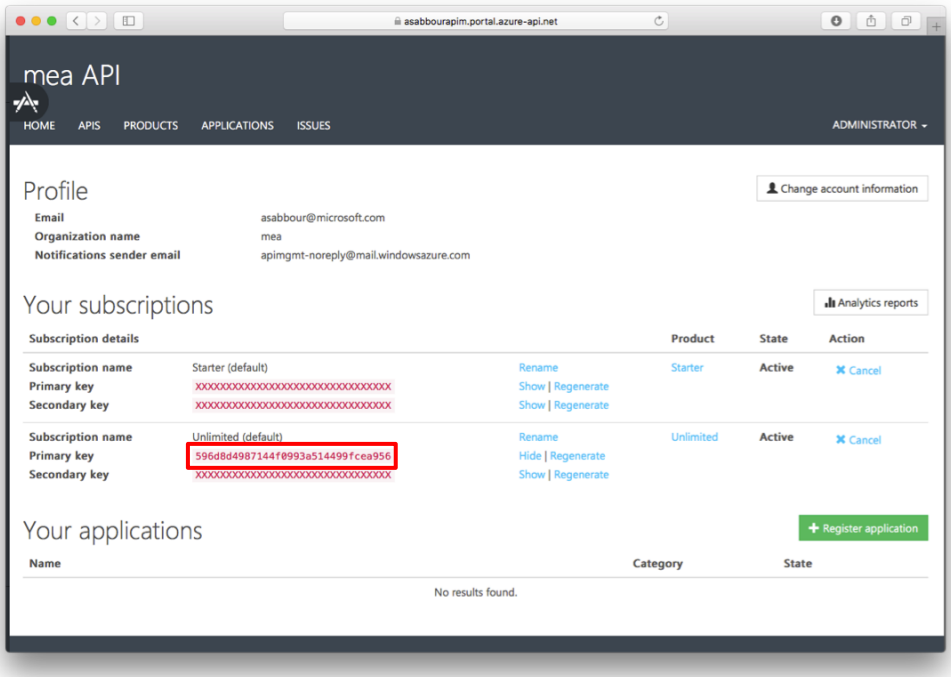


10.2 Obtain the subscription key for Unlimited product

- 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.

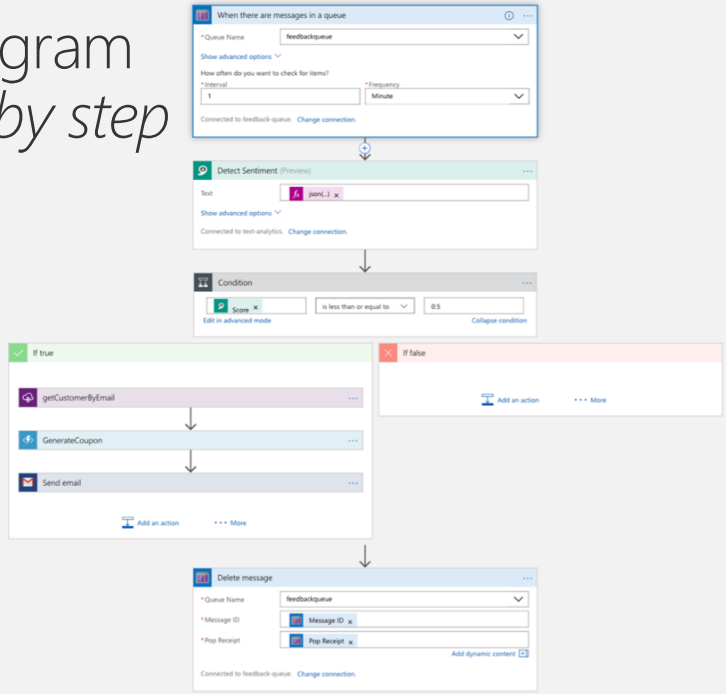






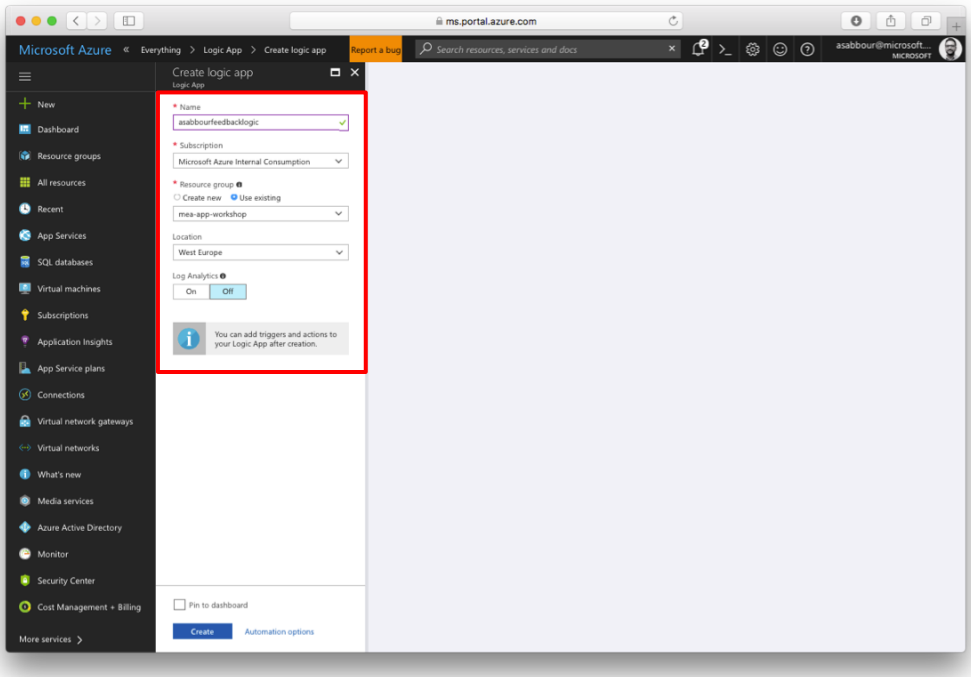
Integrating the solution using Logic Apps

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



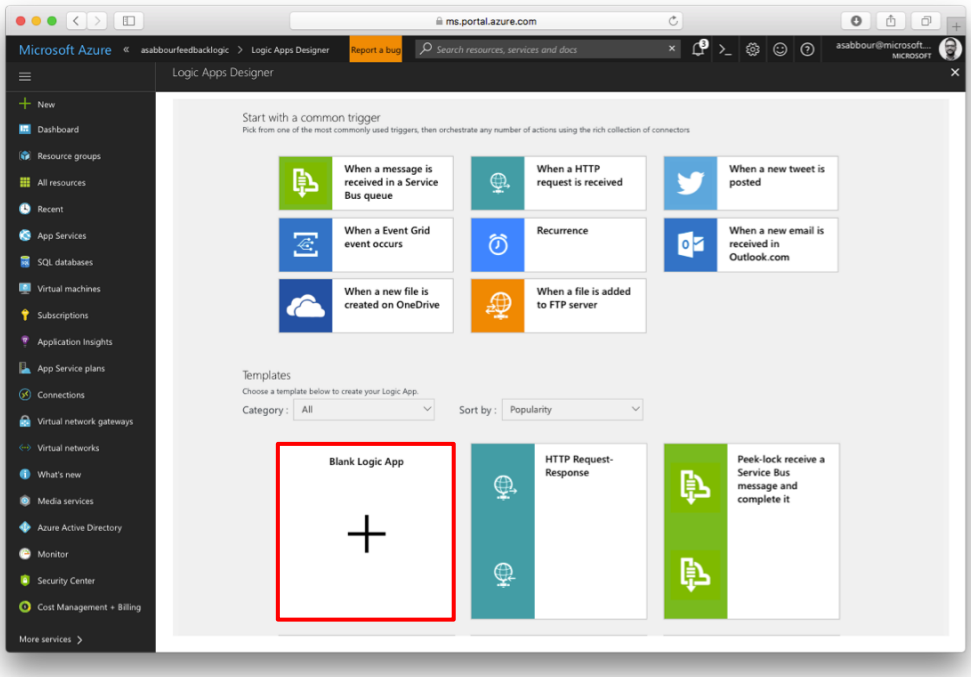
11. Create Logic App

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



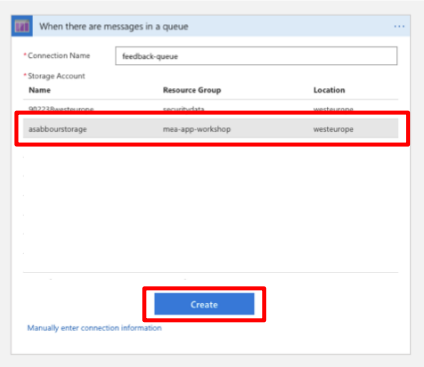
12.1 Create the workflow

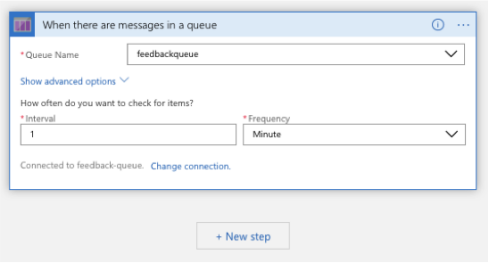
- Open the Logic Apps Designer
- Start from a **Blank Logic App** template



12.2 Add queue trigger

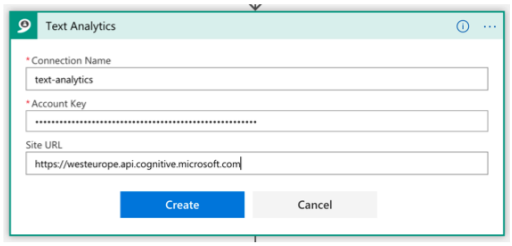
- Add a new Queue trigger, pointing to your storage account created earlier





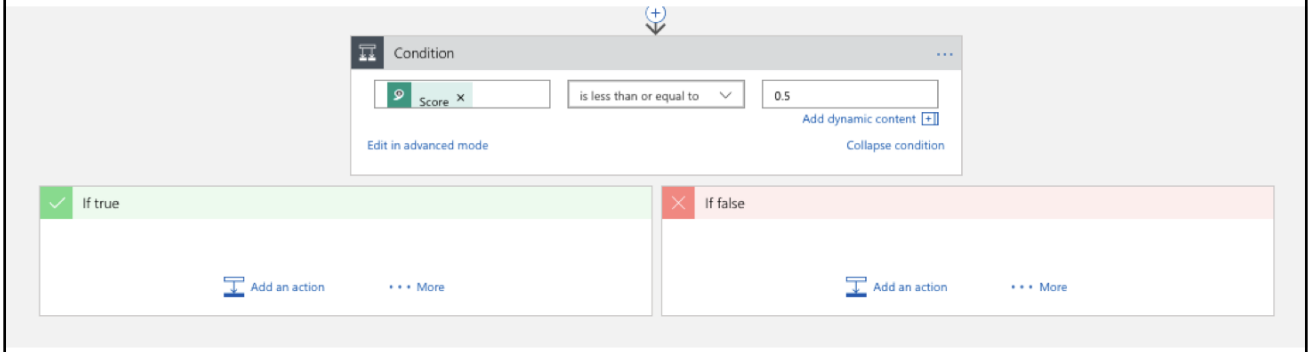
12.2 Add Text Analytics action

- 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}`



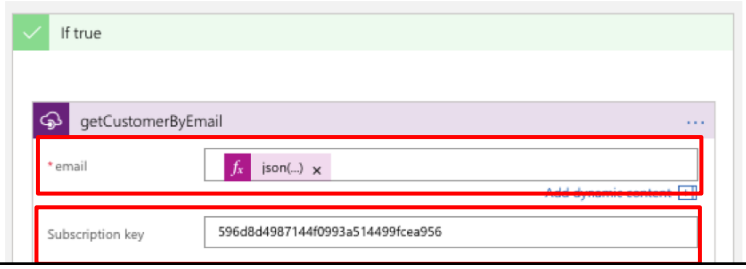
12.3 Add condition, when the sentiment is negative

- 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



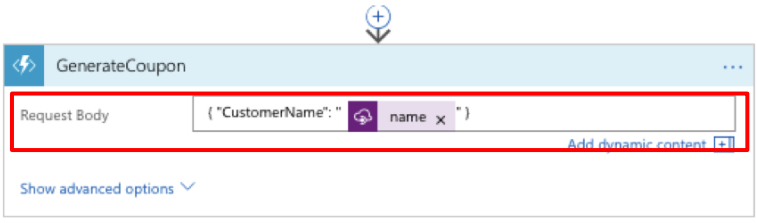
12.4 Add API Management action

- 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



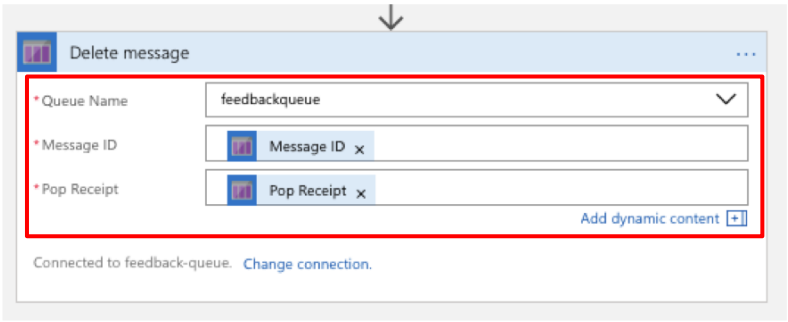
12.6 Add Email action

- 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:

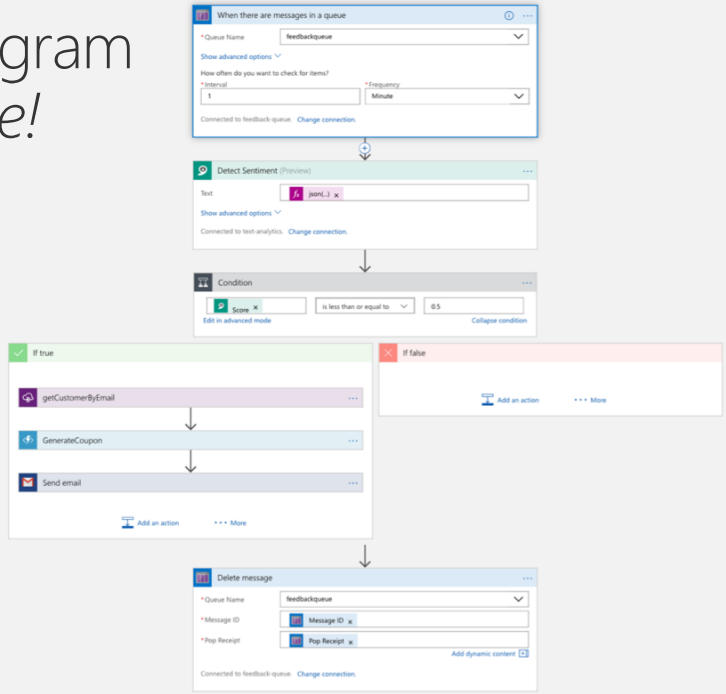
```
Dear @{body('getCustomerByEmail')['name']}, we want to make it up to you. Here is a coupon: @{body('GenerateCoupon').CouponUrl}
```

12.7 Delete the message from the queue

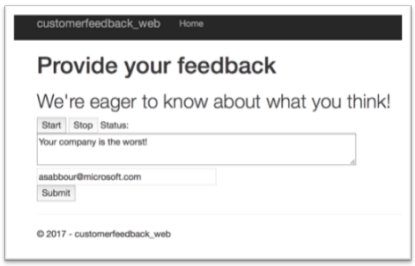
- Now that you've processed this message, delete it from the queue to avoid processing it again. Configure with queue name, message ID and pop receipt ID from the original trigger.

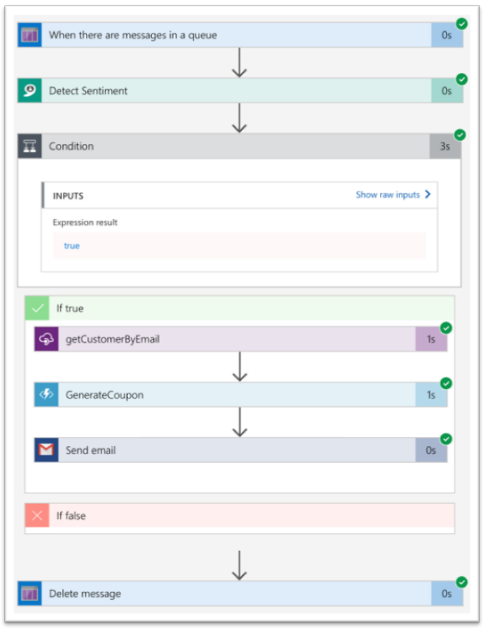


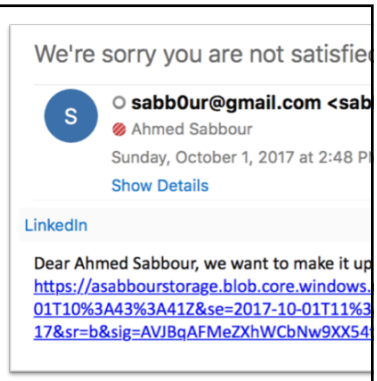
Full Logic App diagram
don't forget to save!

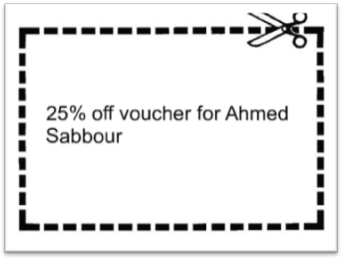


Testing the solution









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