

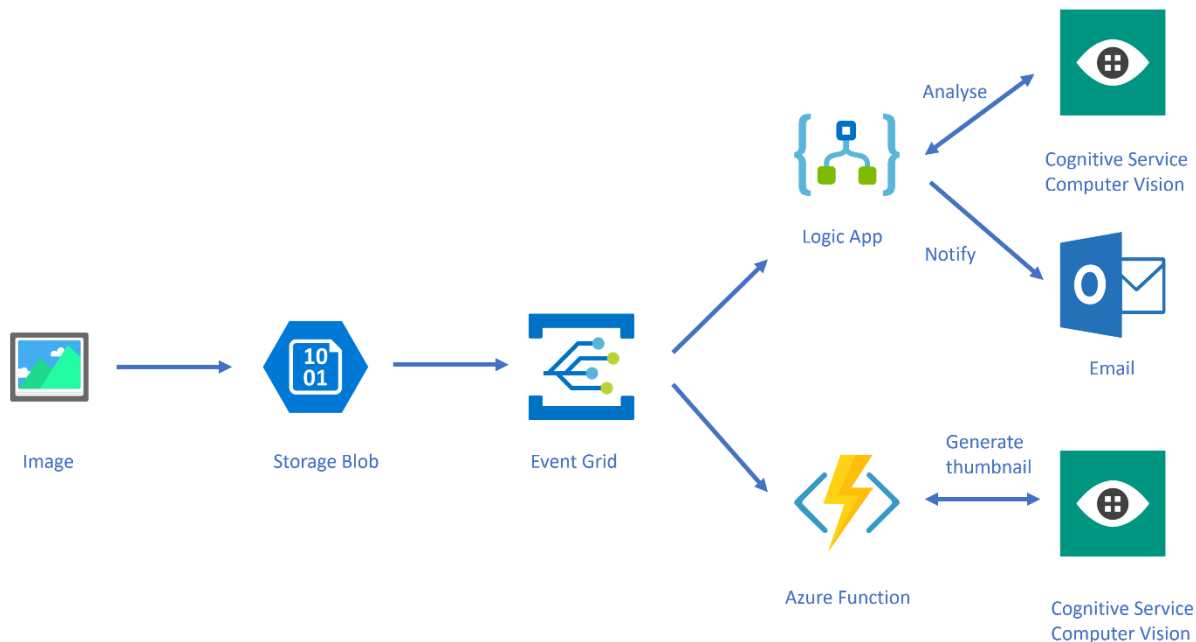
Lab - Building a Smart solution with Azure Storage, Event Grid, Functions, Logic Apps and the Computer Vision API

Author: Steef-Jan Wiggers

Updated by: Prasoon Madnawat and Paco de la Cruz from Mexia.

Objective

In this lab, we will build solutions with Event Grid, Azure Storage, Functions and Logic Apps. A picture (jpg) will be uploaded using the Azure Storage Explorer, an event BlobCreated will be raised and sent to Event Grid Topic within a Storage Account (Blob), and an Azure Function and a Logic App will subscribe to this event. The Azure Function and Logic App will handle the event by calling a Cognitive Service API (Computer Vision API). The objective is to learn the capability of leveraging cognitive services through messaging/event mechanism, which Integration Pro's are familiar with.



Prerequisites

- Azure Subscription
- Azure Storage Explorer: <https://azure.microsoft.com/en-us/features/storage-explorer/>
- An outlook.com account (<https://outlook.live.com/>)

Steps

To build the solution in this lab you have to follow the steps described in this section. From a high level view the steps are:

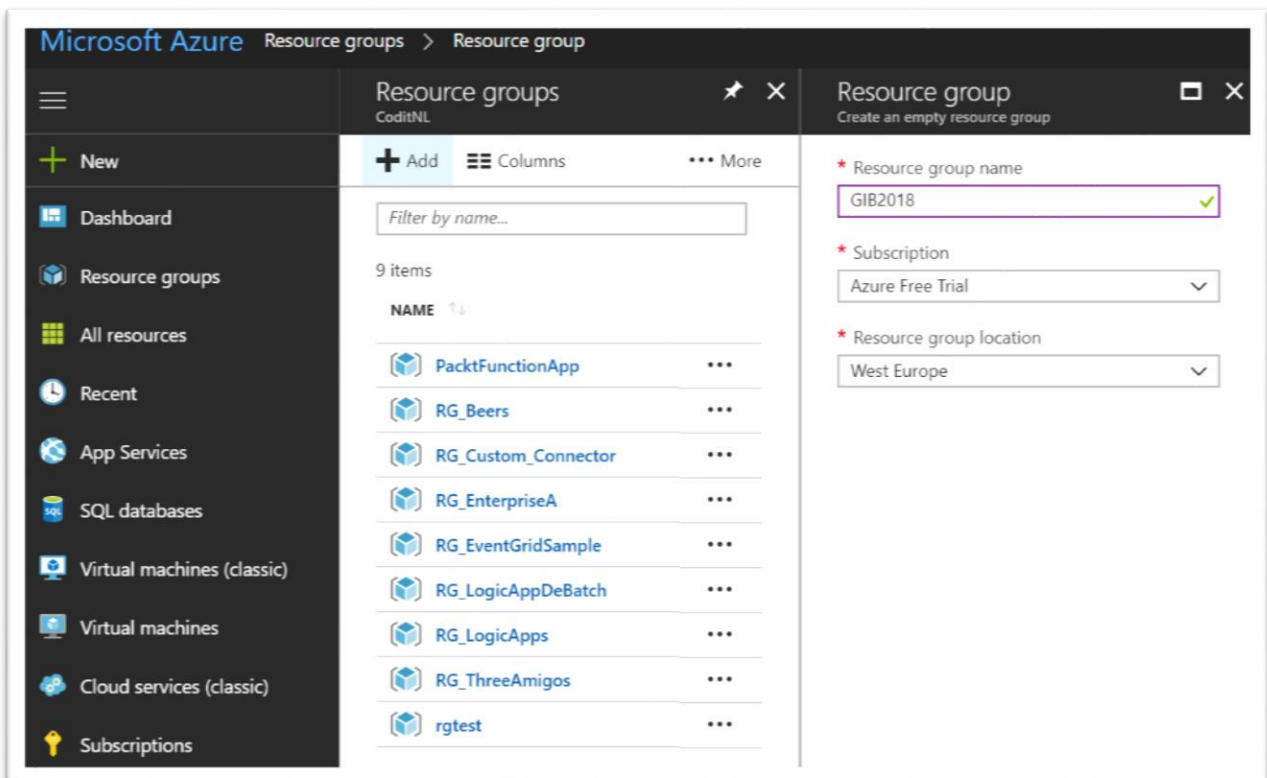
1. Create a resource group
2. Provision a Storage Account
3. Create a blob container
4. Provision Cognitive Service Computer Vision API
5. Provision a Function App
6. Create a Function
7. Configure Event Grid Topic and Subscriptions
8. Test the solution (Part 1)
9. Update Function to generate thumbnails (using Cognitive Service)
10. Test the solution (Part 2)
11. Provision a Logic App and add Event Grid subscription
12. Add Cognitive Service to analyze the image in Logic App
13. Send email notification using Logic App
14. Test the solution (Part 3)

Lab duration: **80 minutes** + Break

Step 1 - Create a resource group

The very first step in this lab is creating a resource group in your Azure subscription. A resource group is a logical container that groups all your resources together. After the lab is finished and you do not want to keep the resources you can simply delete the resource group and the Azure Resource Manager will remove all the resources for you.

1. In the Azure Portal navigate to Resource Groups in the left menu pane.
2. Click the **+ Add**.
3. Provide a name for the resource group, specify a Subscription, and a location.



4. Finally, click **Create** and a resource group will be created for you.
5. In the top right corner, a pop-up will appear, which you can click to go to your resource group.

Step 2 – Provision a Storage Account

Within the resource group you can easily add various types of Azure Resources. For this lab we will need a storage account (blob) to upload an image to a container.

1. Go to the resource group you created earlier (Step 1).
2. Click **+ Add**.
3. A new pane will appear, where you can search for a resource (service).
4. Enter: *Storage Account*.
5. *Storage account - blob, file, table, queue* will appear.
6. Click the icon named Storage account – blob, file, table, queue.
7. A new pane will appear, where you can click **Create**.
8. Again, a new pane will appear, and here you can start specifying a few properties for your Storage Account.
9. In screenshot below you will see the details you need to specify. Make sure you select the kind **"StorageV2 (general purpose v2)"**
10. Since Event Grid is not yet available in all Azure regions, please select one of these locations for Storage Account: **Asia Southeast**, Asia East, Central US, East US, East US 2, Europe West, Europe North, West Central US, West US, West US 2.



GLOBAL INTEGRATION BOOTCAMP

Create storage account

The cost of your storage account depends on the usage and the options you choose below.
[Learn more](#)

* Name ⓘ

gib2018imageanalysis

✓

.core.windows.net

Deployment model ⓘ

Resource manager

Classic

Account kind ⓘ

StorageV2 (general purpose v2)

▼

Performance ⓘ

Standard

Premium

Replication ⓘ

Locally-redundant storage (LRS)

▼

Access tier (default) ⓘ

Cool

Hot

* Secure transfer required ⓘ

Disabled

Enabled

* Subscription

Azure Free Trial

▼

* Resource group

☐ Create new ☒ Use existing

GIB2018

▼

* Location

West Europe

▼

Virtual networks

Configure virtual networks ⓘ

Disabled

Enabled

☐ Pin to dashboard

Create

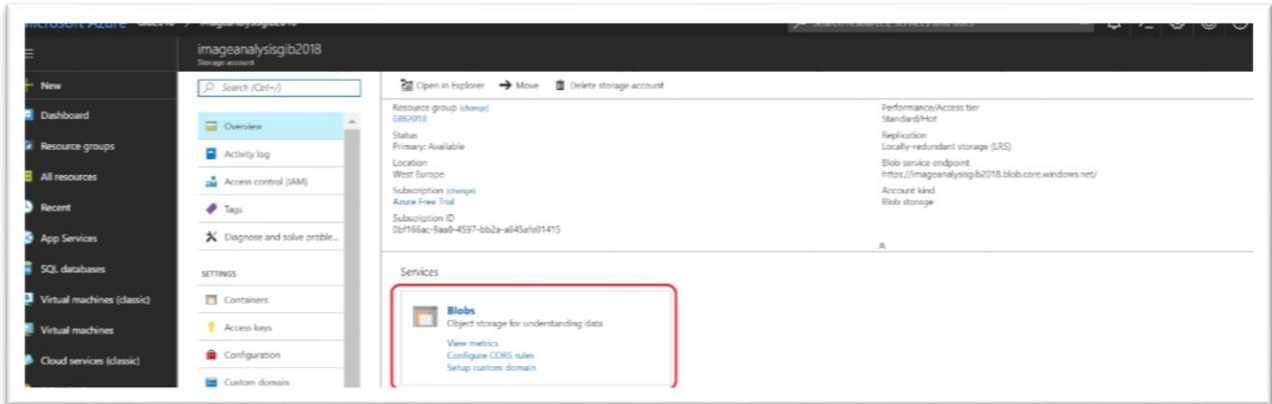
Automation options

11. Choose a useful **unique name**.
12. Click **Create** once finishing specifying. Note that the location needs to be same as the resource group. Keep every Azure resource in the same location.

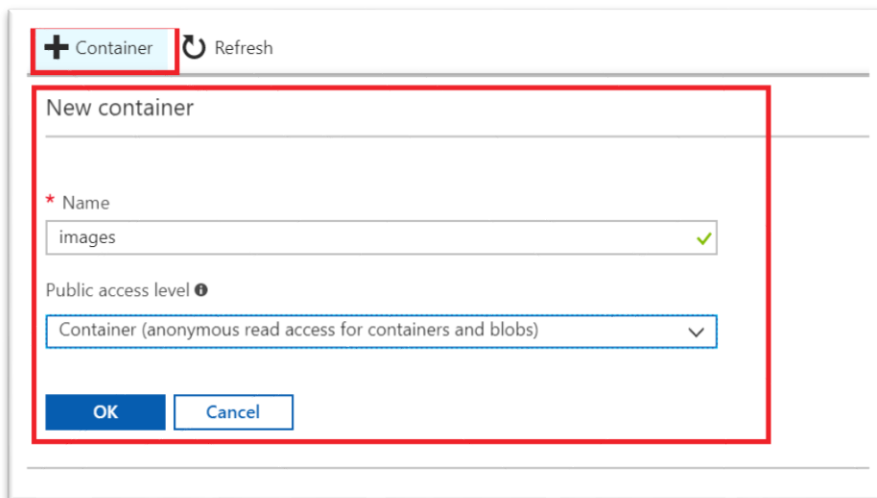
Step 3 – Create a container

Once the Storage Account (blob) is available the next step is to add a container.

1. Go to the resource group you created earlier (Step 1).
2. Click the Storage Account you created in step 2.
3. Click **Blobs**.



4. Click on **+ Container**.
5. Specify a name (**images**) for the container in the window that will appear.

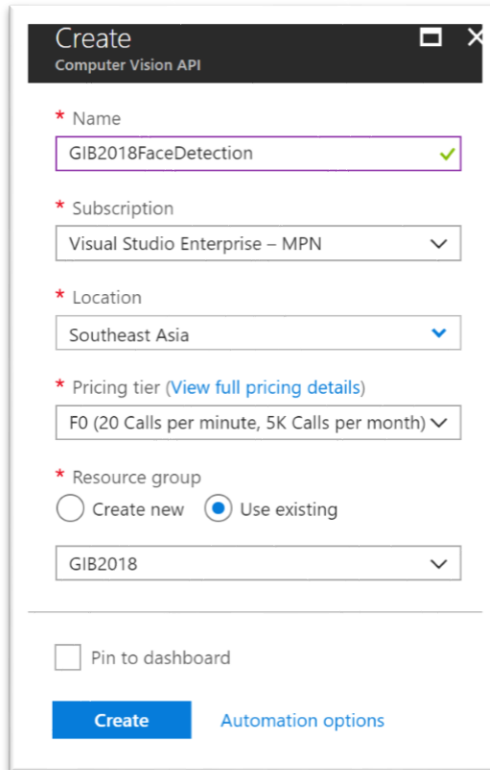


6. Change the **public access level** to read access for containers and blobs only.
7. Click **OK**.
8. Similarly, create another container by the name **thumbnails**.

Step 4 - Provision Cognitive Service Computer Vision API

In this lab we will leverage the pre-built AI capability of Computer Vision API. This API provides image processing functionality such as Optical Character Recognition (OCR), image analysis, and celebrity and landmark recognition.

1. Go to the resource group you created earlier (Step 1).
2. Click **+ Add**.
3. A new pane will appear, where you can search for a resource (service).
4. Enter: *Computer Vision Api*.
5. Click the icon of Computer Vision Api.
6. A new pane will appear, where you can click **Create**.
7. Again a new pane will appear, and here you can start specifying a few properties for your Cognitive Services.
8. Specify the details for the Computer Vision API like in the screenshot below. Choose a unique name, select the location, the cheapest pricing tier, and the correct resource group (Step 1).



Create
Computer Vision API

* Name
GIB2018FaceDetection ✓

* Subscription
Visual Studio Enterprise - MPN ▼

* Location
Southeast Asia ▼

* Pricing tier ([View full pricing details](#))
F0 (20 Calls per minute, 5K Calls per month) ▼

* Resource group
☐ Create new ☒ Use existing
GIB2018 ▼

☐ Pin to dashboard

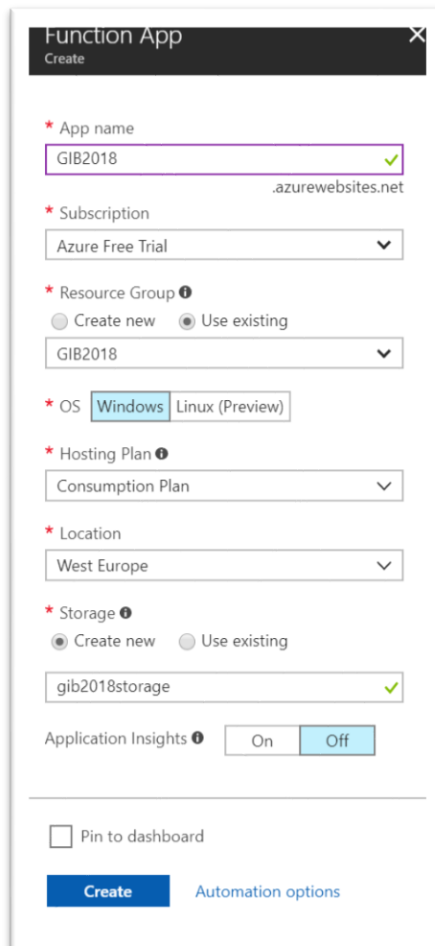
Create [Automation options](#)

9. Click **Create** after you have specified the details.

Step 5 – Provision a Function App

A Function App is a logical container that can have one or multiple functions. A Function App can be tied to an App Service Plan, to share resource over various applications hosted in App Services including functions or to a consumption plan, which basically comes down to a pay-as-you model. You pay for each execution. The benefit here is you do not have to manage anything. The service scales automatically and is highly available. In this lab, we choose to create a Function App bound to a consumption plan.

1. Go to the resource group you created earlier (Step 1).
2. Click **+ Add**.
3. A new pane will appear, where you can search for a resource (service).
4. Enter: *Function App*.
5. Click the icon of Function App.
6. A new pane will appear, where you can click **Create**.
7. Specify the details for the Function App like in the screenshot below. Choose a unique name, select resource group, the location, Consumption Plan, and provide a name for a new Storage Account.



Function App
Create

* App name
GIB2018 ✓
.azurewebsites.net

* Subscription
Azure Free Trial ▼

* Resource Group ⓘ
☐ Create new ☒ Use existing
GIB2018 ▼

* OS Windows Linux (Preview)

* Hosting Plan ⓘ
Consumption Plan ▼

* Location
West Europe ▼

* Storage ⓘ
☒ Create new ☐ Use existing
gib2018storage ✓

Application Insights ⓘ

☐ Pin to dashboard

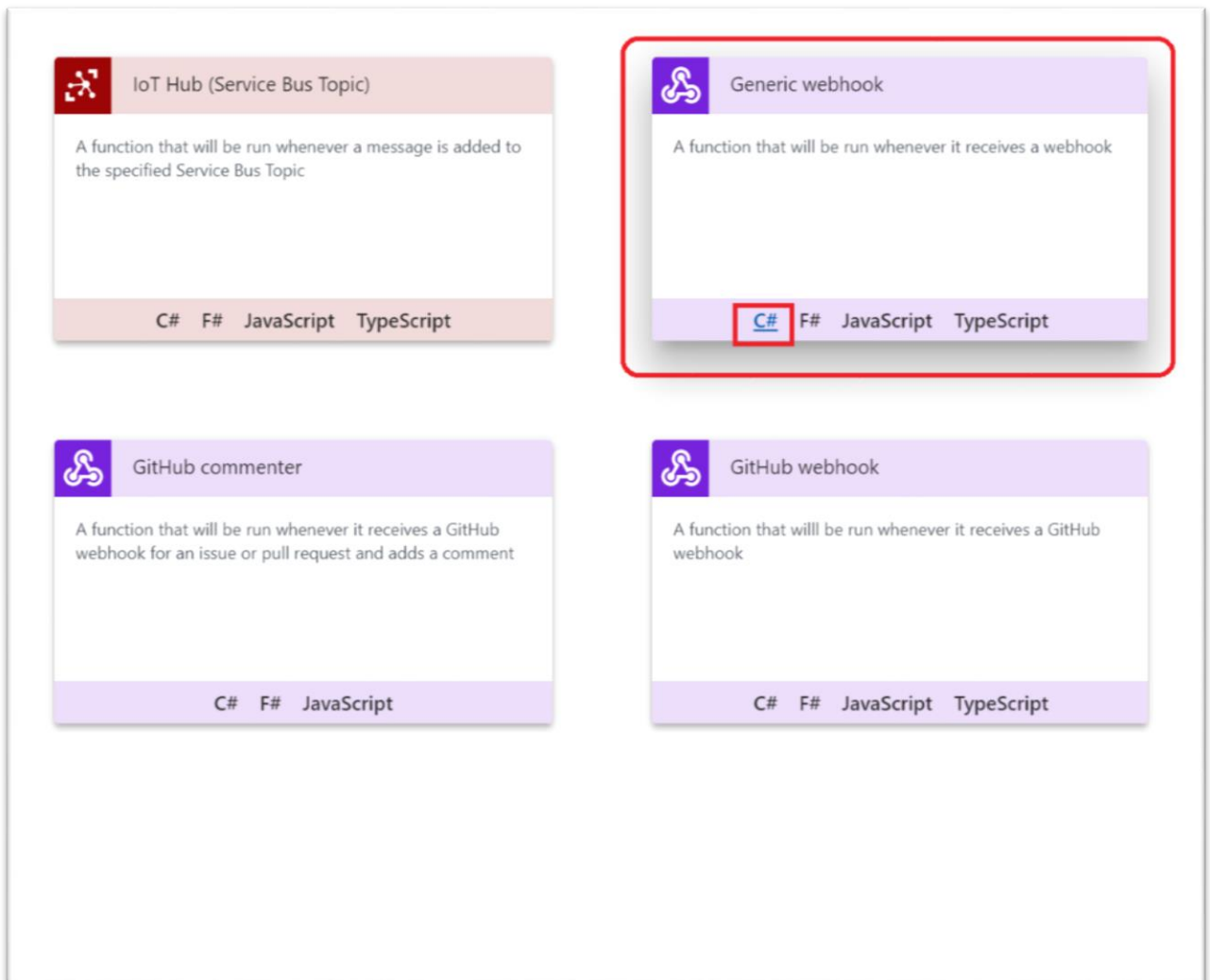
Create Automation options

8. Click **Create** after you have specified the details.

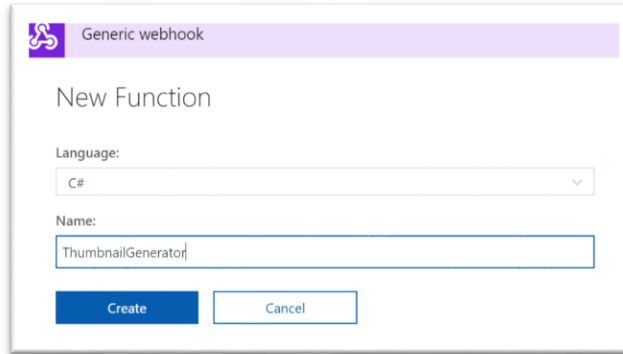
Step 6 - Create a Function

Once the Function App is available you can create a function and code. The code we will use in the function receives an event from Event Grid topic belonging to the Storage Account (Step 2).

1. Go to the resource group you created earlier (Step 1).
2. Select the created Function App (Step 5).
3. A pane will appear Function Apps --> Name of your Function App. If you select Functions you can click the plus sign. This is add a function. Select custom function in the windows that will appear.
4. Select the Generic WebHook and click language C#.



5. A pane will appear on the right side. Provide a name for your function and click **Create**.



Generic webhook

New Function

Language: C#

Name: ThumbnailGenerator

Create Cancel

6. A new pane will appear with some default code.
7. Replace the code with the following code:

```
#r "Newtonsoft.Json"
#r "System.Web"

using System;
using System.Net;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Web;

public static async Task<object> Run(HttpRequestMessage req, TraceWriter log)
{
    log.Info($"Webhook was triggered!");

    //intialize
    string imageInfo = string.Empty;

    //get content
    string jsonContent = await req.Content.ReadAsStringAsync();

    log.Info($"Event : {jsonContent}");

    var response = req.CreateResponse(HttpStatusCode.OK);
    response.Content = new StringContent(imageInfo, System.Text.Encoding.UTF8, "application/json");
    return response;
}
```

8. Click **Run** to examine the behavior. We will change this code later on for our objective in this lab.
9. Click **Get Function URL**.

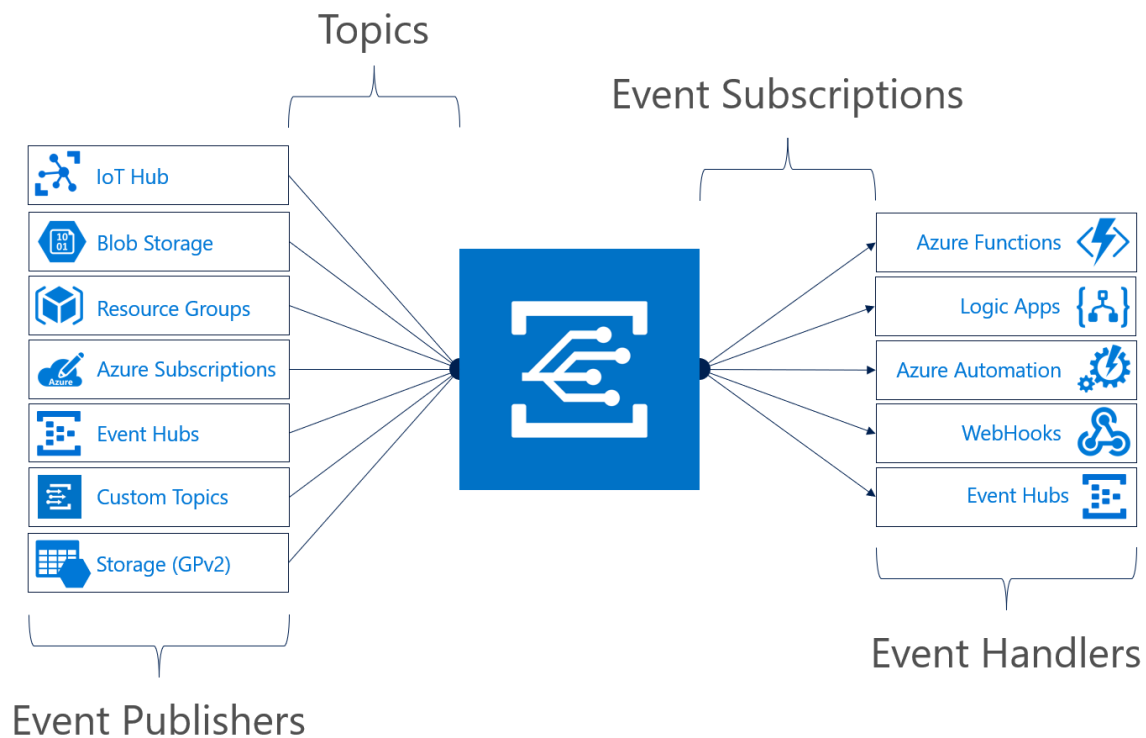


10. Save the URL in a notepad.

Step 7 – Configure the Event Grid Topic in the Azure Storage Account

The resources for this lab have been setup and now it's time to configure Event Grid Topic inside the Azure Storage Account (V2). With Azure Event grid developers can build reactive applications that handle events. These events can be **storage blob events**, provisioning notifications in an Azure subscription, IoT device signals, or even custom events. The benefit of this service is that developers no longer to have continuously poll an application or service for a change. Now it can receive an event and continue.

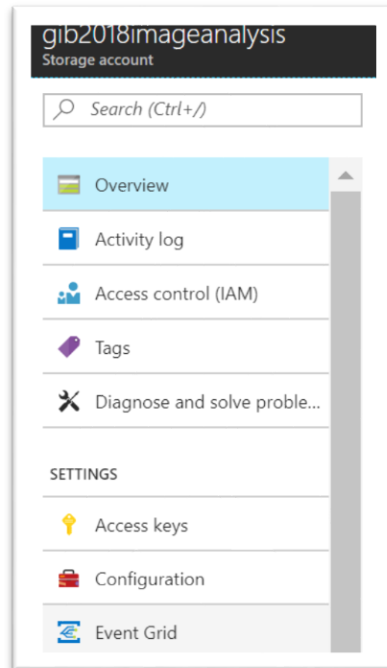
The concept of Event Grid revolves around events emitted from a source (publisher), which can be an Azure service or a third party source that adheres to the event schema. The event publishers in Azure include IoT Hub, Storage, and the recently added Event Hubs. Subsequently, the events are sent to a topic in Event Grid, and each topic can be one or multiple subscribers (event handlers). A topic can be configured with the event publisher or can be a custom topic for custom events. Finally, the event handlers react to the events and process them. The event handlers in Azure include Functions, WebHook, Event Hubs, Azure Automation and Logic Apps.



1. Open the Storage Account created in Step 2.
2. Select the **Event Grid** in the blade.




GLOBAL INTEGRATION BOOTCAMP



3. A new pane will appear. Click **Create one**. We will now add subscriptions to the Event Grid Topic.
4. Provide a name for the subscription, select the Event Type, Subscriber Type, pre- and post fix filter like the screenshot below. This is a subscription of the Function that will handle the **BlobCreated** event. Here you will need the URL of your Function (Step 6).
 - a. Prefix: /blobServices/default/containers/images/
 - b. Suffix: .jpg



GLOBAL INTEGRATION BOOTCAMP



No Event Subscriptions Found!

Enable event based programming in Azure by using Event Subscriptions to connect your resources.
Connect native event sources to event handlers, or bring your own.
[Learn more about Azure Event Grid.](#)

Create one

Create Event Subscription

Event Grid

Name

ImageAnalysisFunction

Subscribe to all event types

Event Types

Blob Created

Subscriber Type

Web Hook

Subscriber Endpoint

https://gib2018.azurewebsites.net/api/Ima...

Prefix Filter

/blobServices/default/containers/images/

Suffix Filter

.jpg

Filter Case Sensitive

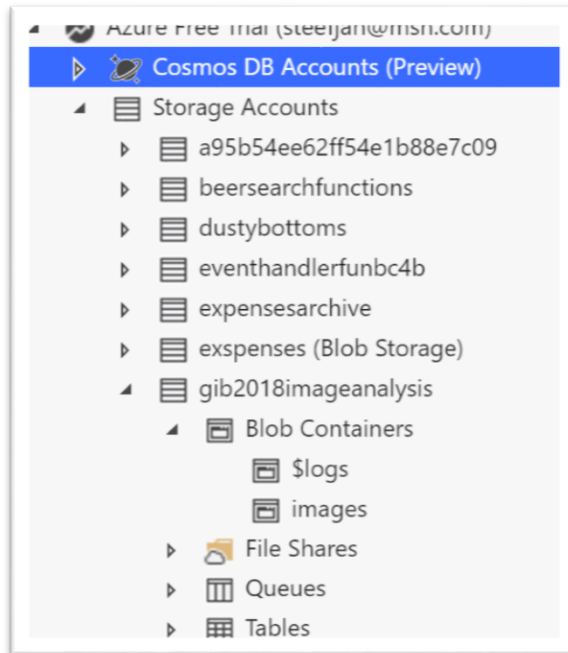
5. Click **Create** in bottom right corner of the pane.
6. Hit **Refresh** and the subscription should appear.
7. You should now have one subscription like the screenshot below.

NAME	ENDPOINT TYPE	ENDPOINT	PREFIX FILTER	SUFFIX FILTER	EVENT TYPES	
 ImageAnalysisFunction	WebHook	https://gib2018.azurewebsites.net/api/ImageAnalysis	/blobServices/default...	.jpg	Microsoft.Storage.BlobCreated	 Metrics

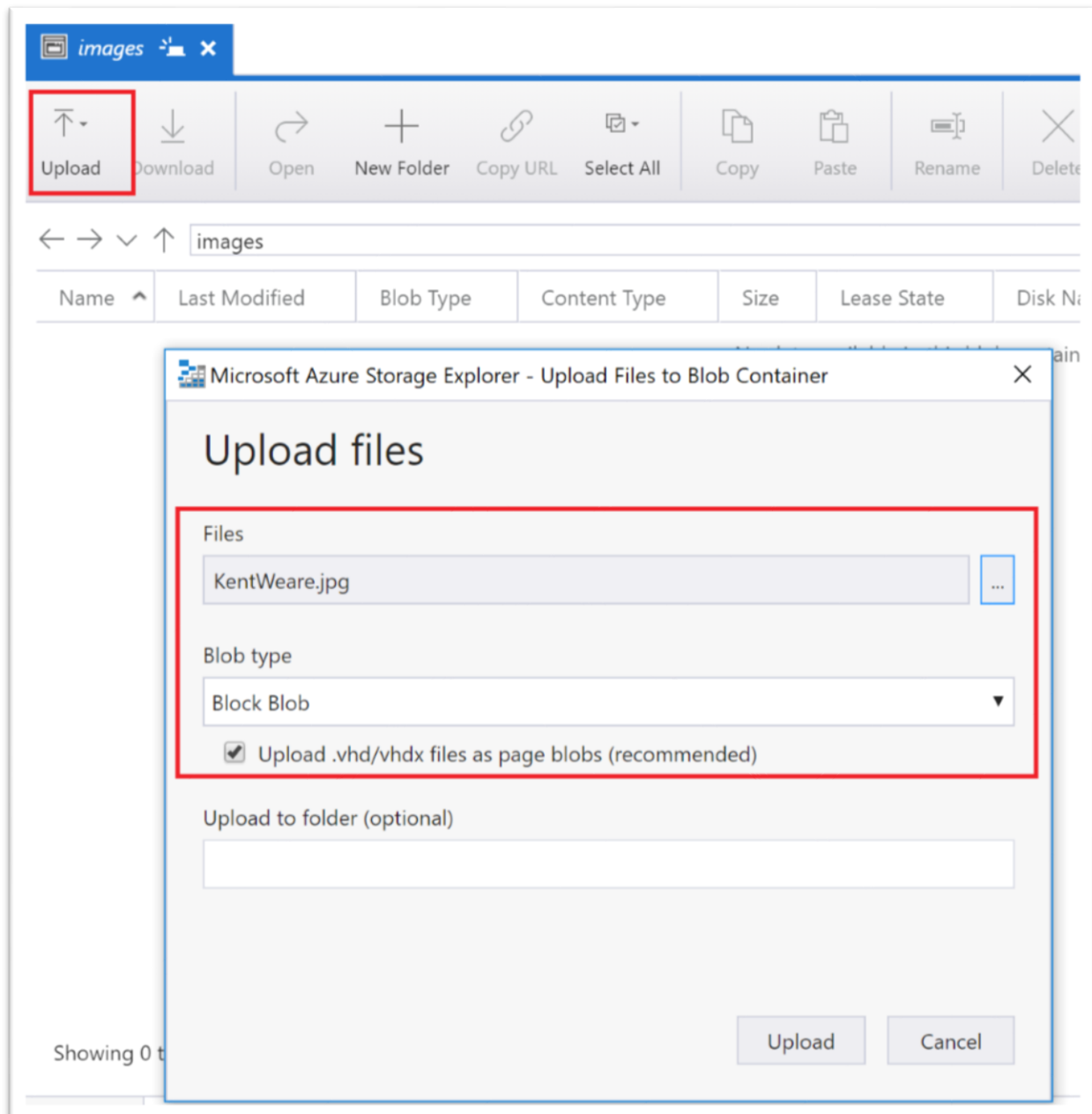
Step 8: Test the solution (Part 1)

With the previous steps all completed, we can now test the solution you have build by uploading a .jpg file using the Azure Storage Explorer.

1. Start the **Azure Storage Explorer**.
2. Note that you have to login to your subscription in this tool through manage accounts.
3. Once that is complete or you have already done so, you should be able to navigate to your storage account and container (Step 2/3).



4. Click Upload and navigate to .jpg picture on your machine that is a picture of yourself or someone else.



5. Click **Upload**.
6. Go to your Azure Function and click on "Monitor".
7. You should see the event in it's raw format:



GLOBAL INTEGRATION BOOTCAMP

Invocation details

Parameter	
req	Method: POST, Uri: https://gib2018funapp.azurewebsites.net/api/ImageProcessor
log	
_context	dc6c9023-aa61-4358-9a22-f01fc6be021b
\$return	response

Logs

Webhook was triggered!
Event : [{
 "topic": "/subscriptions/b6091b73-42b2-430e-81a2-6397596c23b7/resourceGroups/GIB2018/providers/Microsoft.Storage/storageAccounts/gib2018imganalysis2",
 "subject": "/blobServices/default/containers/images/blobs/ForLab.JPG",
 "eventType": "Microsoft.Storage.BlobCreated",
 "eventTime": "2018-03-22T06:59:21.3851303Z",
 "id": "ec5d78f9-101e-00b8-41ab-c1da9a069455",
 "data": {
 "api": "PutBlob",
 "clientRequestId": "8b817540-2d9e-11e8-8401-a596a2b6f65b",

8. Inspect the message to see what basic attributes are part of this event.



Step 9: Update Function to generate Thumbnail images

1. Replace the code with the code below. (Available at <https://github.com/pacodelacruz/gib2018>)

```
#r "Newtonsoft.Json"
#r "System.Web"
#r "Microsoft.WindowsAzure.Storage"

using System;
using System.Net;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Web;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Blob;
using System.Configuration;

public static async Task<object> Run(HttpRequestMessage req, TraceWriter log)
{
    log.Info($"Webhook was triggered!");

    //get content
    var jsonContent = await req.Content.ReadAsStringAsync();

    log.Info($"Event : {jsonContent}");

    //event data is an Json Array
    var data = JObject.Parse(jsonContent);

    //get url from event data
    foreach (JObject item in data)
    {
        var blobEventData = item.GetValue("data");
        log.Info($"blobEventData : {blobEventData}");

        var imageUrl = blobEventData.Value<string>("url");
        log.Info($"imageUrl : {imageUrl}");

        //read image
        var webClient = new WebClient();
        var image = webClient.DownloadData(imageUrl);
        var thumbnailFileName = $"Thumbnail{imageUrl.LastIndexOf("/") + 1}";

        //generate image
        var thumbnailImageData = await GenerateThumbnail(image);

        //upload to blob
        UploadToBlob(thumbnailImageData, thumbnailFileName);
    }

    var response = req.CreateResponse(HttpStatusCode.OK);
    response.Content = new StringContent("Success", System.Text.Encoding.UTF8, "application/json");
    return response;
}

private static async Task<byte[]> GenerateThumbnail(byte[] image)
{
    var client = new HttpClient();
    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "<YOUR KEY>");
```

```
var requestParameters = "width=200&height=150&smartCropping=true";
var uri = "https://southeastasia.api.cognitive.microsoft.com/vision/v1.0/generateThumbnail?" +
requestParameters;

using (var content = new ByteArrayContent(image))
{
    content.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");

    var response = client.PostAsync(uri, content).Result;
    byte[] thumbnailImageData = await response.Content.ReadAsByteArrayAsync();

    return thumbnailImageData;
}
}

private static async Task UploadToBlob(byte[] thumbnail, string thumbnailFileName)
{
    var storageAccount = CloudStorageAccount.Parse(ConfigurationManager.AppSettings["AzureWebJobsStorage"]);
    var blobClient = storageAccount.CreateCloudBlobClient();
    var container = blobClient.GetContainerReference("thumbnails");
    var blockBlob = container.GetBlockBlobReference(thumbnailFileName);

    using (var memoryStream = new MemoryStream(thumbnail)) {
        await blockBlob.UploadFromStreamAsync(memoryStream);
    }
}
```

2. Click **Save**.
3. Note that the URI needs to **contain the location name** of your Cognitive Service. Moreover, you need to place the key in the section "**<YOUR KEY>**". The key can be obtained by navigating to your Cognitive Service (Step 4).
4. In Cognitive Service select **Keys** is the pane.
5. **Copy** Key1 and go to back to your function.
6. Paste it in the "**<YOUR KEY>**" section and **Save**.



Step 10 – Test the solution (Part 2)

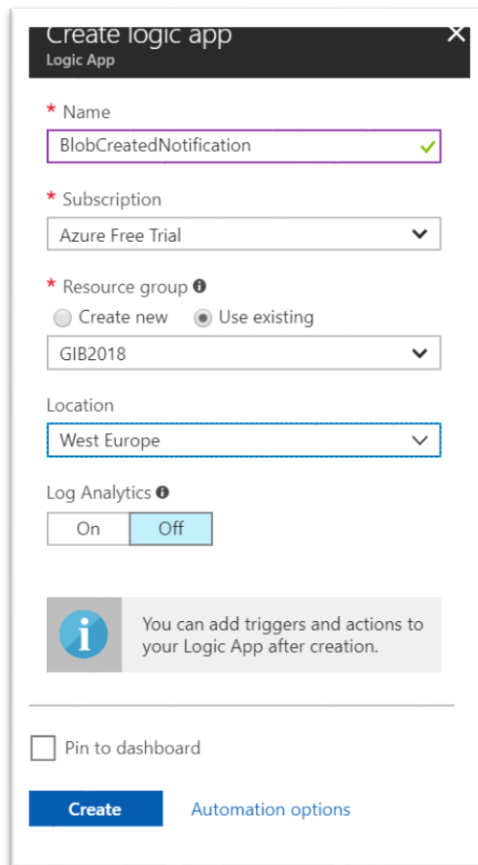
We can now test the thumbnail functionality that has just been built using Azure cognitive services.

1. Upload another image to the blob container using the **Azure Storage Explorer** as explained previously.
2. Wait for a minute and then open the **thumbnail** blob container.
3. You should see an image file by the name Thumbnail<Original File Name> created.
4. Download this file locally. It should be a 200 pixels X 150 pixels sized thumbnail for your original image file.

Step 11 – Provision a Logic App and add Event Grid subscription

The final part of our solution is a Logic App. A Logic App is a service in Azure that enable you to build a flow definition based on actions. The flow starts with an action trigger followed with one or more actions. We will provision a Logic App and build a definition to handle the BlobCreated event by analysing the image and sending an email notification.

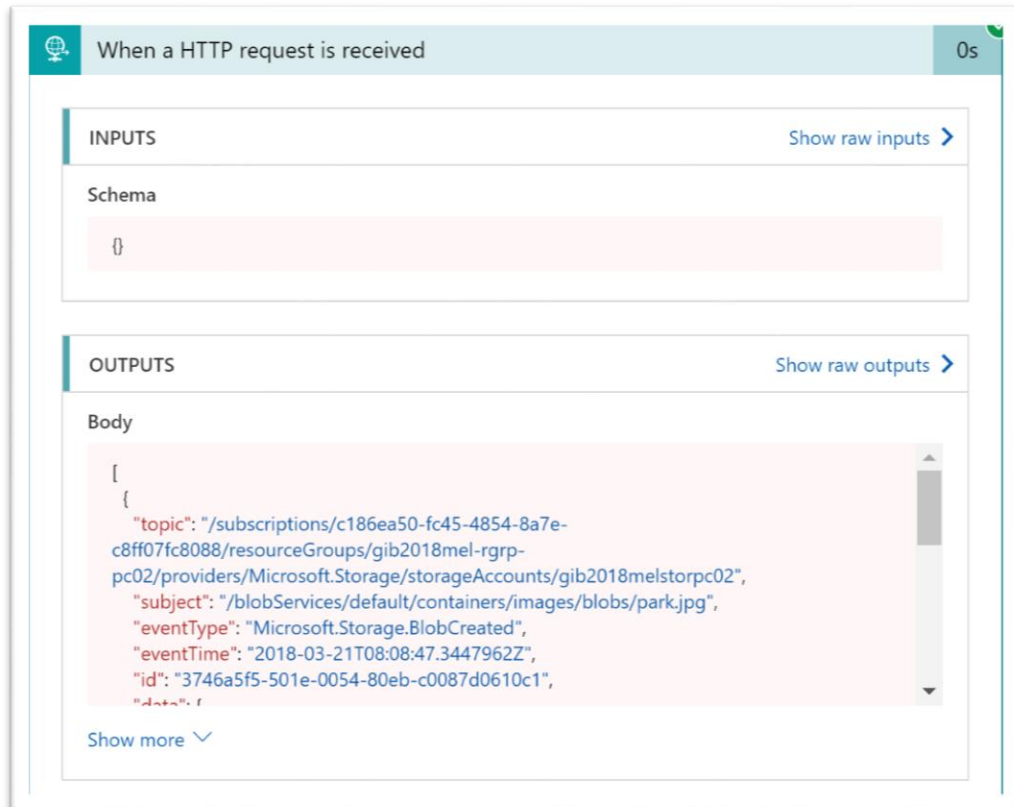
1. Go to the resource group you created earlier (Step 1).
2. Click **+ Add**.
3. A new pane will appear, where you can search for a resource (service).
4. Enter: **Logic App**.
5. Click the icon of Logic App.
6. Specify a name in the Create Logic App pane, the subscription, resource group, and location.



7. Click **Create**.
8. Open the Logic App once it is created.
9. Click **Blank Logic App**. We will build a Logic App definition from scratch and not choosing any of the pre-built ones.
10. In the first action (trigger) enter: *Http*
11. Choose **Request**.
12. **Save** the Logic App.

GLOBAL INTEGRATION BOOTCAMP

13. After saving the Logic App, we can trigger it and inspect the Event Grid message,
 - a. Copy the HTTP POST URL
 - b. Add a **new subscription** in the Storage Account **Event Grid Settings** (As explained in the Step 7) created before but now entering Logic App HTTP POST URL.
 - i. **Prefix:** /blobServices/default/containers/images/
 - ii. **Suffix:** .jpg
 - c. Test the Event Grid new subscription by adding another image to the **images** container
 - d. In the Logic App **overview** blade, under **Runs History**, click on the first run, and on the Run Details graphic view, click on the trigger box. Check the outputs of the Http Request Trigger.



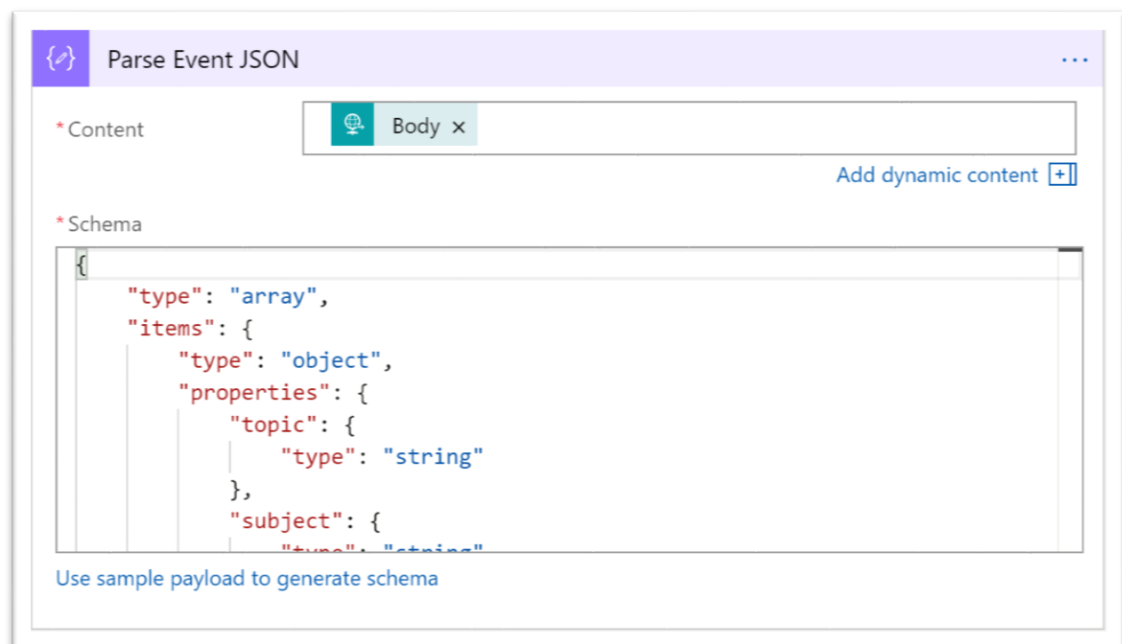
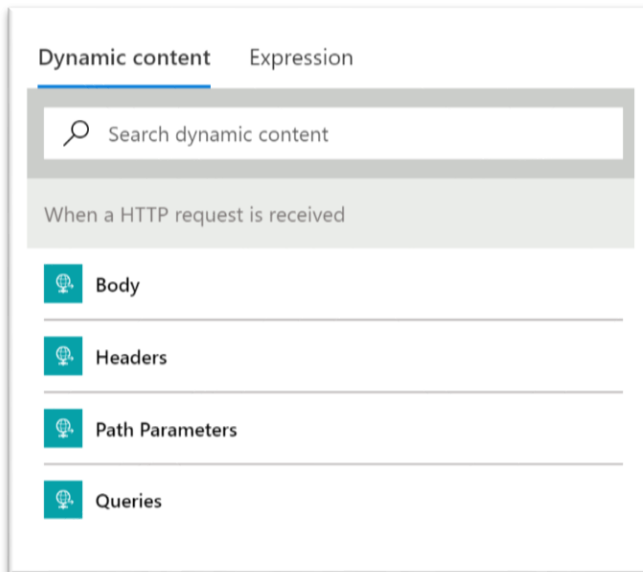
The screenshot shows the 'When a HTTP request is received' trigger configuration in the Logic App interface. The 'INPUTS' section shows an empty schema. The 'OUTPUTS' section shows the 'Body' of the Event Grid message, which is an array containing one event object. The event object has the following properties:

```
[
  {
    "topic": "/subscriptions/c186ea50-fc45-4854-8a7e-c8ff07fc8088/resourceGroups/gib2018mel-rgrp-pc02/providers/Microsoft.Storage/storageAccounts/gib2018melstorpc02",
    "subject": "/blobServices/default/containers/images/blobs/park.jpg",
    "eventType": "Microsoft.Storage.BlobCreated",
    "eventTime": "2018-03-21T08:08:47.3447962Z",
    "id": "3746a5f5-501e-0054-80eb-c0087d0610c1",
    "data": {}
  }
]
```

- e. We can inspect the body of the Event Grid message. As we can see, it's an array of events. In this case, an array of one element.
- f. **Copy** the output body which will be used in the next exercise.

Step 12: Add Cognitive Service to analyze the image

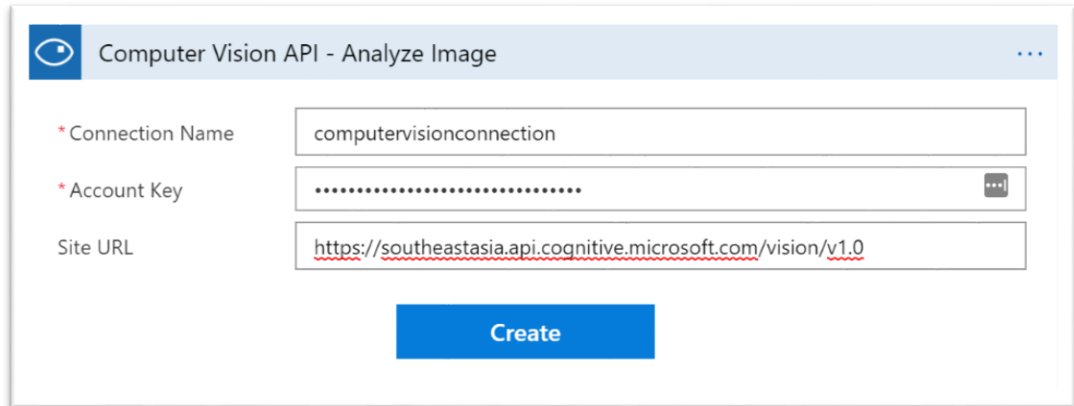
1. Go back to the Logic App Designer
2. Add an action of type **Data Operations - Parse JSON**, so we can use the Event message properties as tokens in the Designer.
3. Click **“Use sample payload to generate schema”**
4. **Paste** the Event Message we received in our previous test (last step of previous exercise) and click **Done**. This will generate the JSON schema for us.
5. In **Content**, from the **Dynamic Content**, select the Http Request **Body**



6. Then, add another action, **Computer Vision API - Analyze Image**
7. The designer will ask you to create a connection for the Computer Vision API.

GLOBAL INTEGRATION BOOTCAMP

- a. Give it a connection name
- b. Enter the key and URL of your Computer Vision API



Computer Vision API - Analyze Image

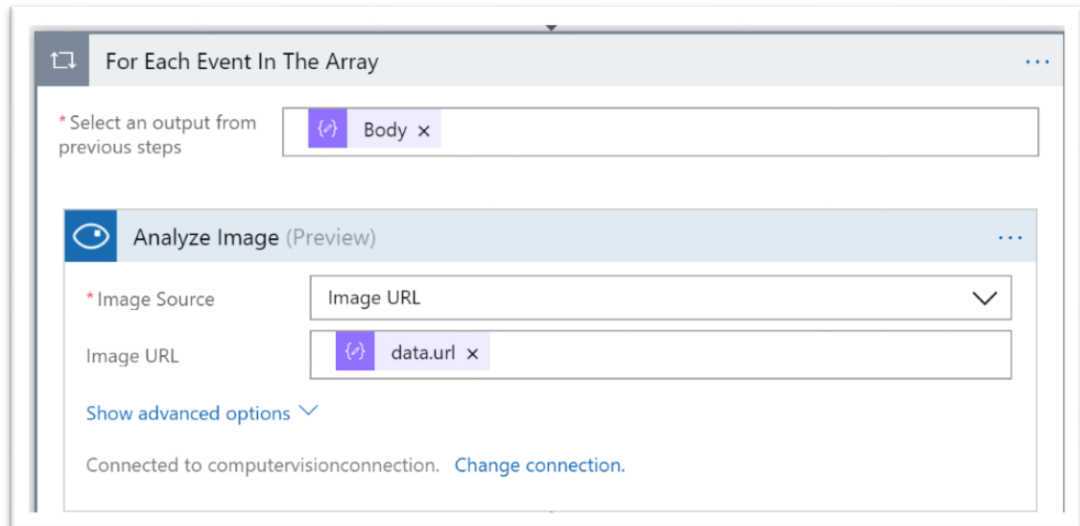
* Connection Name: computervisionconnection

* Account Key:

Site URL: <https://southeastasia.api.cognitive.microsoft.com/vision/v1.0>

Create

8. Then configure the Analyze Image action,
 - a. Select source as **Image URL**
 - b. Select the token **url** from the Parse JSON Action.
 - c. This will add a For Each loop, as the Event Grid message is an array of events which can contain one or more events.



For Each Event In The Array

* Select an output from previous steps: Body x

Analyze Image (Preview)

* Image Source: Image URL

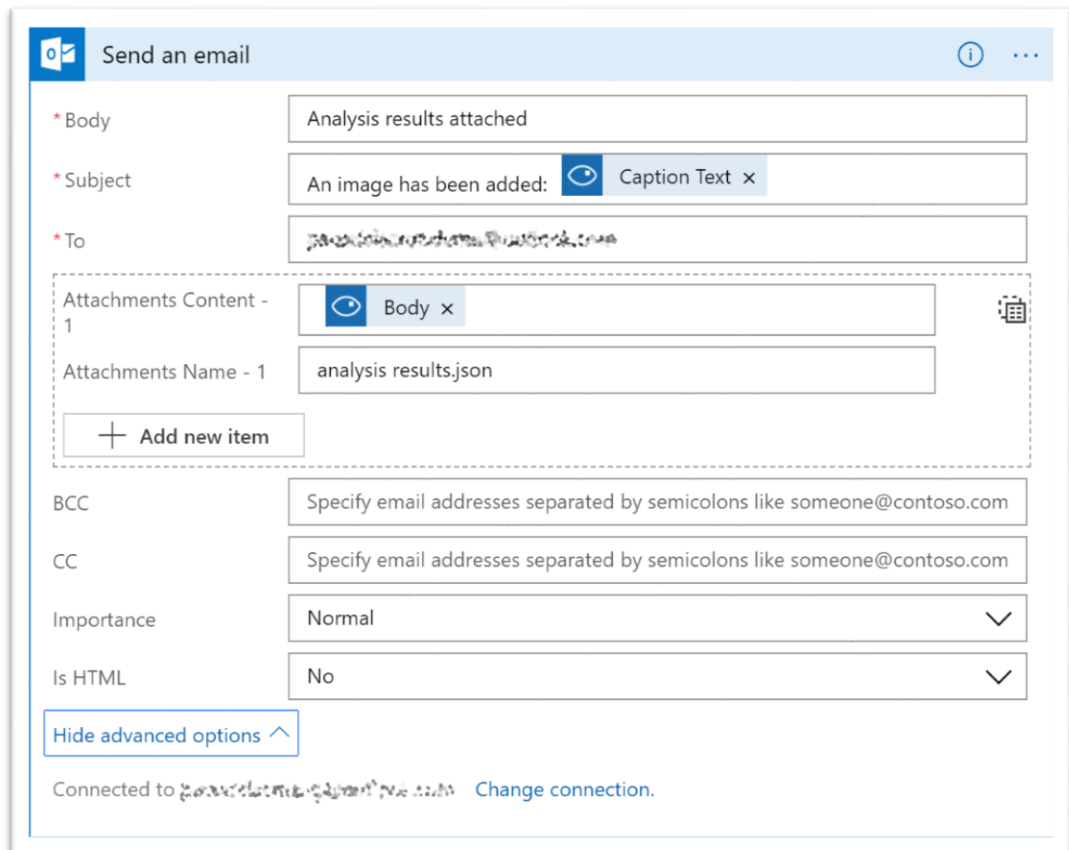
Image URL: data.url x

Show advanced options

Connected to computervisionconnection. [Change connection.](#)


Step 13: Send email notifications


1. Go back to the Logic App Designer and within the ForEach action, add an action of type **Outlook.com - Send an email**
2. Sign into your Outlook.com (with your Live/Microsoft account).
3. Next specify
 - a. **To:** your own email address
 - b. **Subject:** An image has been added, and select **Caption Text** from the previous action. This will add another ForEach, as the Caption Text is also an array, in many cases of only 1 element.
 - c. **In Body**, enter "Analysis results attached".
 - d. Click **Show advanced options**
 - e. In **Attachments Content**: type in in the **Expression editor**: `body('Analyze_Image')`
 - f. Enter the **Attachment Name**, like **results.json**.




Send an email

* Body: Analysis results attached

* Subject: An image has been added:  Caption Text x

* To: 

Attachments Content - 1:  Body x

Attachments Name - 1: analysis results.json

+ Add new item


BCC: Specify email addresses separated by semicolons like someone@contoso.com

CC: Specify email addresses separated by semicolons like someone@contoso.com

Importance: Normal

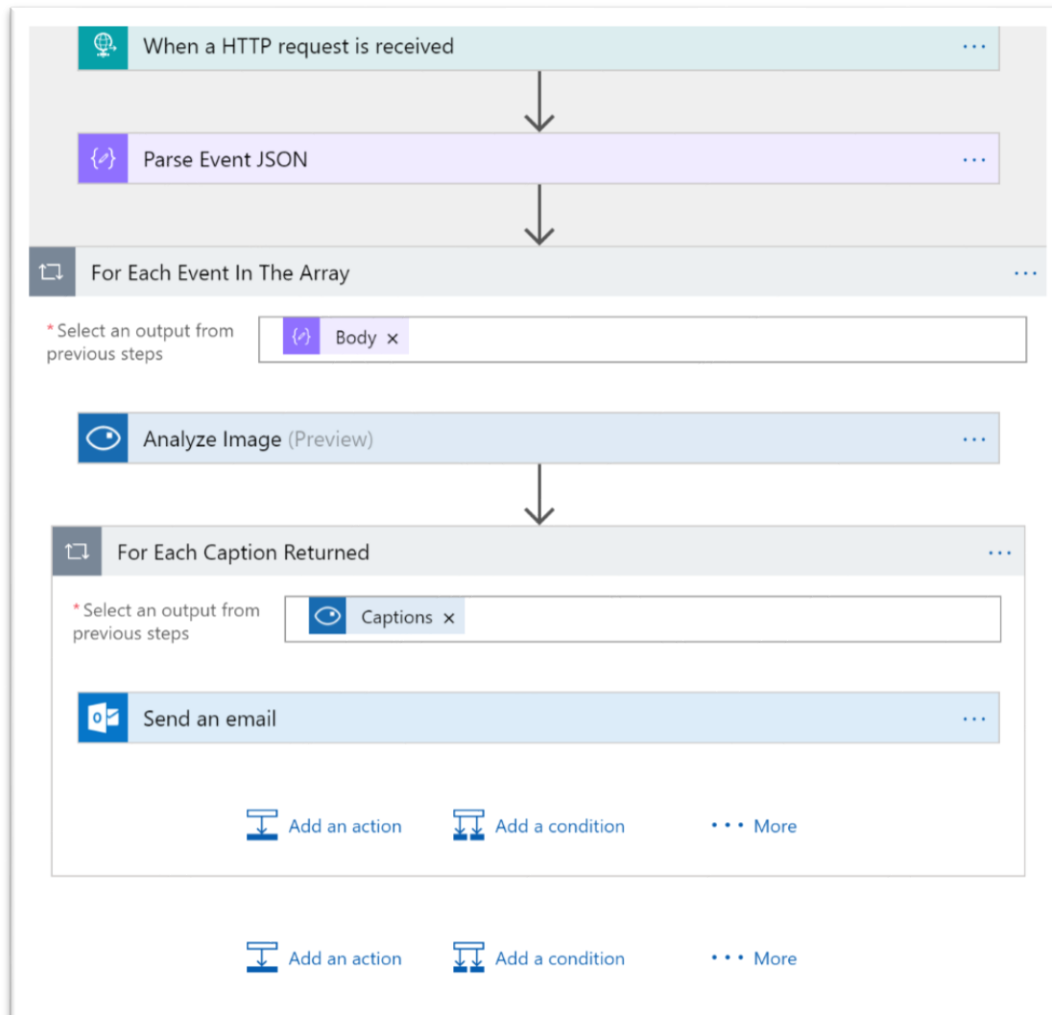
Is HTML: No

Hide advanced options ^

Connected to  Change connection.

4. Save the Logic App.

The final Logic App should look like this:



5. Now we are ready to test it.



GLOBAL INTEGRATION BOOTCAMP

Step 14 – Test the solution (Part 3)

We can now test the solution again by uploading a .jpg file using the Azure Storage Explorer. Repeat the steps described in the Step 8 and see how the Logic App is executed. You should receive an email.

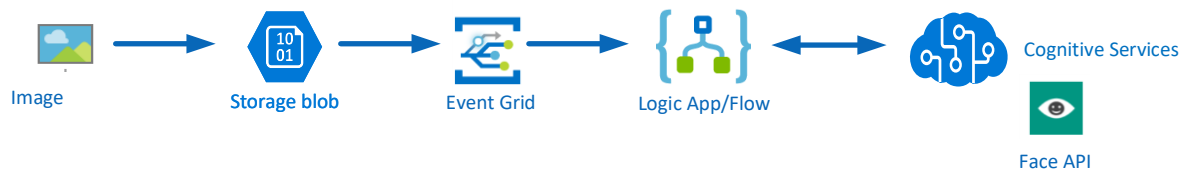
Test the solution with different images, check the caption given to each image, and also explore the run history and inputs and outputs of each action within each run.

We've finished the Lab :)

GLOBAL INTEGRATION BOOTCAMP

Explanation of the Lab (What have we done)

With this lab so far you have build a solution using Event Grid, Functions, Cognitive Service and Azure Storage. With Event Grid, GA since end of January 2018, you can build a sophisticated serverless solution, where Event Grid has its role and value. For instance you can run image analysis on let's say a picture of someone is being added to blob storage. The event, a picture added to blob storage can be pushed as an event to Event Grid , where a function or Logic App can handle the event by picking up the image from the blob storage and sent it to a Cognitive Service API face API. See the diagram below.



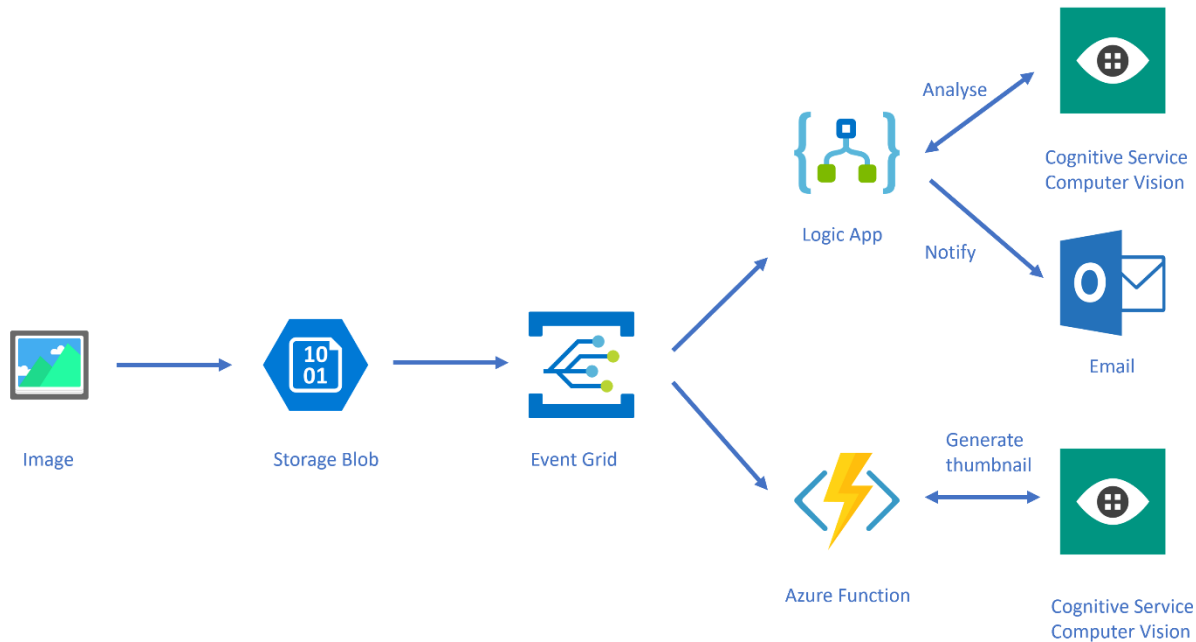
We have done something similar yet we also used a Function as event handler.

Azure Event Grid

Azure Event Grid can be described as an event broker that has one or more event publishers and subscribers as we have learned from the introduction. Event publishers are currently Azure blob storage, resource groups, subscriptions, event hubs and custom events. More will be added in the coming months like IoT Hub, Service Bus, and Azure Active Directory. Subsequently, there are consumers of events (subscribers) like Azure Functions, Logic Apps, and WebHooks. And more will be added on the subscriber side too with Azure Data Factory, Service Bus and Storage Queues for instance.

We have uploaded an image to a Storage blob container, which will be the event source (publisher). Subsequently, the Storage blob container belongs to a Storage Account containing the Event Grid capability. And the Event Grid has two subscribers, a Logic App to notify me a blob has been created and an Azure Function that will analyze the image created in the blob storage, by extracting the URL from the event and use it to analyze the actual image.

GLOBAL INTEGRATION BOOTCAMP




The screenshot below depicts the subscriptions on the events on the Blob Storage account. The WebHook will subscribe to each event, while the Logic App and Azure Function are only interested in the **BlobCreated** event, in a particular container(prefix filter) and type (suffix filter).

NAME	ENDPOINT TYPE	ENDPOINT	PREFIX FILTER	SUFFIX FILTER	EVENT TYPES	
ImageAnalysisFunction	WebHook	https://gib2018funapp.azurewebsites.net/api/ThumbnailCreator	/blobServices/default/conta...	.jpg	Microsoft.Storage.BlobCreated	
ImageAnalysisLogicApp	WebHook	https://prod-09.australiasoutheast.logic.azure.com/workflows/b32dac95bafc419c9a2d23da8a4...	/blobServices/default/cont...	.jpg	Microsoft.Storage.BlobCreated	

Besides being centrally managed Event Grid offers intelligent routing, which is the core feature of Event Grid. And you can use filters for event type, or subject pattern (pre- and suffix). Moreover, the filters are intended for the subscribers to indicate what type of event and/or subject they are interested in. You have configured the filters in Step 7.





No Event Subscriptions Found!

Enable event based programming in Azure by using Event Subscriptions to connect your resources.
Connect native event sources to event handlers, or bring your own.
[Learn more about Azure Event Grid.](#)

Create one

Create Event Subscription

Event Grid

* Name
ImageAnalysisFunction ✓

☐ Subscribe to all event types

* Event Types
Blob Created ✓

Subscriber Type
Web Hook ✓

* Subscriber Endpoint
https://gib2018.azurewebsites.net/api/lma... ✓

Prefix Filter
/blobServices/default/containers/images/ Optional

Suffix Filter
.jpg Optional

☐ Filter Case Sensitive

- **Event Type** : Blob Created
- **Prefix** : /blobServices/default/containers/images/
- **Suffix** : .jpg

The prefix, a filter object, looks for the **beginsWith** in the subject field in the event. And the suffix looks for the **subjectEndsWith** in again the subject. In the event above you see that the subject has the specified **Prefix** and **Suffix**. See also [Event Grid subscription schema](#) in the documentation as it will explain the properties of the subscription schema. The subscription schema of the function is as follows:



```
{
  "properties": {
    "destination": {
      "endpointType": "webhook",
      "properties": {
        "endpointUrl":
"https://imageanalysisfunctions.azurewebsites.net/api/AnalyseImage?code=Nf301gnvyHy4J44JAKssv23
578D5D492f7KbRCaAhcEKkWw/vEM/9Q=="
      }
    },
    "filter": {
      "includedEventTypes": [ "blobCreated"],
      "subjectBeginsWith": "/blobServices/default/containers/testcontainer/",
      "subjectEndsWith": ".jpg",
      "subjectIsCaseSensitive": "true"
    }
  }
}
```

The Azure Function is only interested in a Blob Created event with a particular subject and content type (image .jpg). And this will be apparent once you inspect the incoming event to the function.

```
{
  "topic": "/subscriptions/0bf166ac-9aa8-4597-bb2a-
a845afe01415/resourceGroups/GIB2018/providers/Microsoft.Storage/storageAccounts/gib2018imag
eanalysis",
  "subject": "/blobServices/default/containers/images/blobs/KentWeare.jpg",
  "eventType": "Microsoft.Storage.BlobCreated",
  "eventTime": "2018-02-02T14:35:24.2530868Z",
  "id": "850b8ff2-001e-0038-5233-9c72bd069501",
  "data": {
    "api": "PutBlob",
    "clientRequestId": "4d7cbcc0-0826-11e8-a417-2b437bef514e",
    "requestId": "850b8ff2-001e-0038-5233-9c72bd000000",
    "eTag": "0x8D56A4A320B7C34",
    "contentType": "image/jpeg",
    "contentLength": 13034,
    "blobType": "BlockBlob",
    "url": "https://gib2018imageanalysis.blob.core.windows.net/images/KentWeare.jpg",
    "sequencer": "00000000000000AF00000000001E5CD2",
    "storageDiagnostics": {
      "batchId": "fb978160-259a-44a0-888b-91bc8ced5320"
    }
  },
  "dataVersion": "",
  "metadataVersion": "1"
}
```

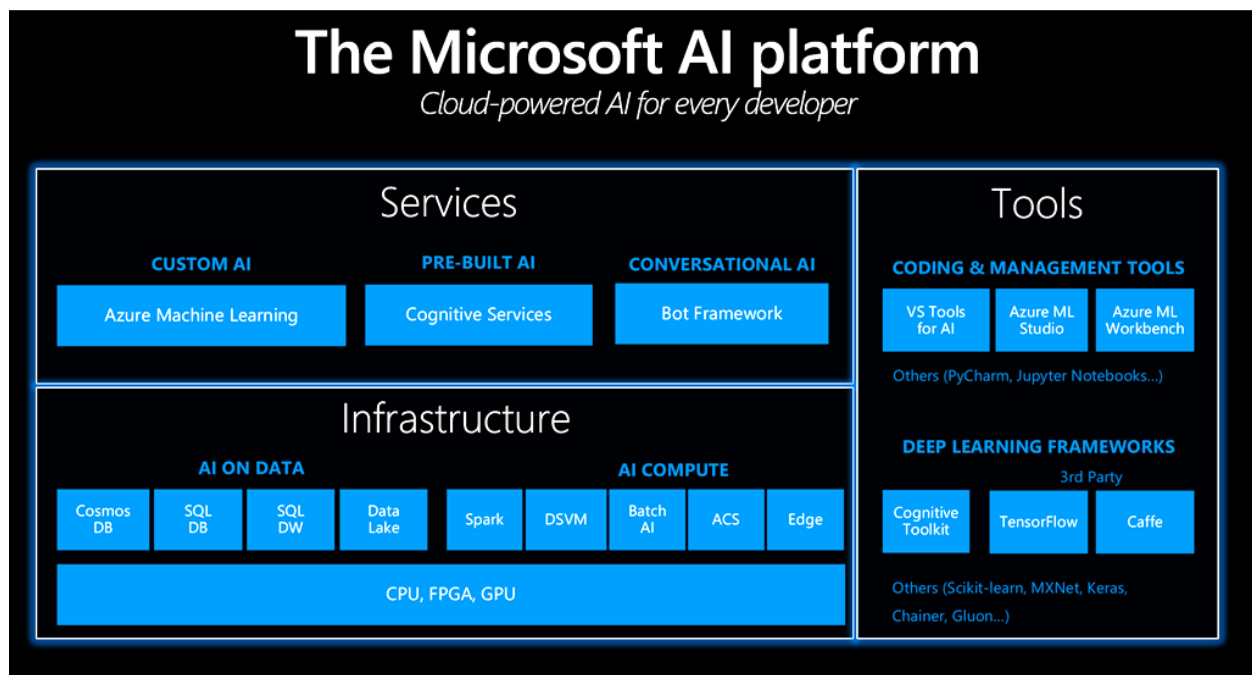
GLOBAL INTEGRATION BOOTCAMP

If you would like to learn more about Azure Event Grid you go to the following resources:

- Azure Event Grid: <https://azure.microsoft.com/en-us/services/event-grid/>
- InfoQ: <https://www.infoq.com/news/2018/02/microsoft-azure-event-grid-ga>
- White paper: <https://www.biztalk360.com/understanding-microsoft-azure-event-grid/>

Cognitive Services

Microsoft Cognitive Services are a part of the [Microsoft AI Platform](#). This service has several pre-built AI capabilities including speech, text, and images. The Cognitive Services Computer Vision API is one of the services, which you observe its behavior through the following URL: <https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/>



In this lab we have used the Computer Vision API through an Azure Function by calling its REST Endpoint for describing an image.

Logic Apps and Functions

The Logic App and Function we have used in this lab were event handlers. Both handle the **BlobCreated** event in a different manner. The function called the Computer Vision API, and the Logic App send out a notification through email. If you like to learn more about both these serverless capabilities visit:

- Logic App: <https://azure.microsoft.com/en-us/services/logic-apps/>
- Azure Function: <https://azure.microsoft.com/en-us/services/functions/>

Contact SteefJan@msn.com or twitter @steefJan (Original Author)