

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

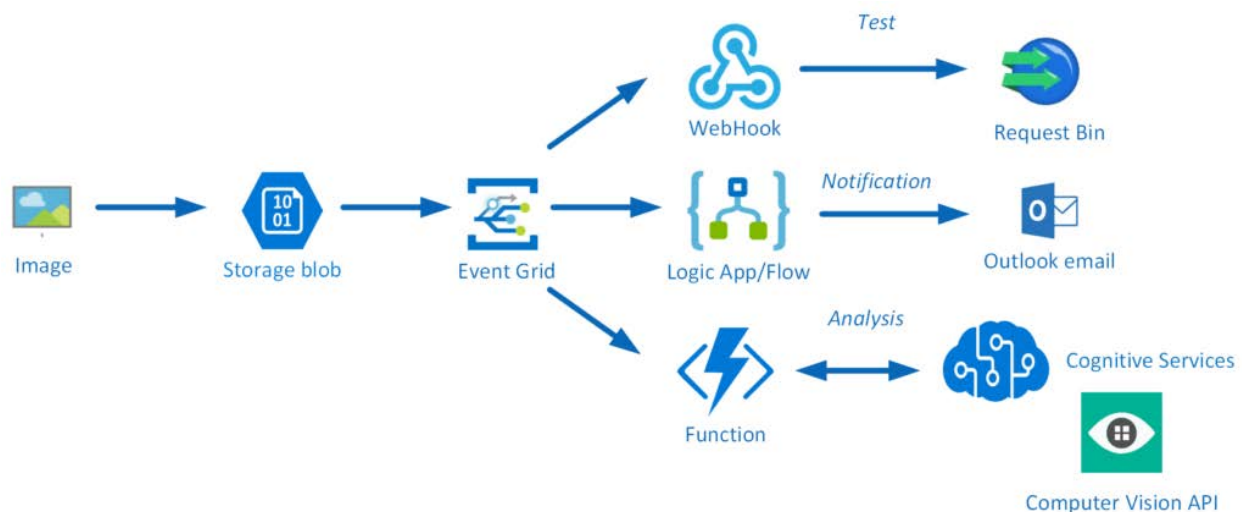
Steef-Jan Wiggers

Contents

Objective	3
Prerequisites	3
Step-by-step guide	4
Provision a Storage Account	5
Create a container	7
Provision Cognitive Service Computer Vision API.....	8
Provision a Function App	9
Create a Function.....	10
Create a Request Bin.....	14
Provision a Logic App.	15
Configure the Event Grid Topic in the Azure Storage Account.....	17
Test the solution.	21
Explanation of the Lab (What have we done)	26
Azure Event Grid	26
Cognitive Services	30
Logic Apps and Functions.....	30

Objective

In this lab, we will build solutions with Event Grid, Azure Storage, and Functions. 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 WebHook (RequestBin) will subscribe to this event. The Azure Function will handle the event by calling a Cognitive Service API (Computer Vision API). The WebHook (Request Bin) will receive the event too, and you can inspect the raw event format. 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/>

Step-by-step guide

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:

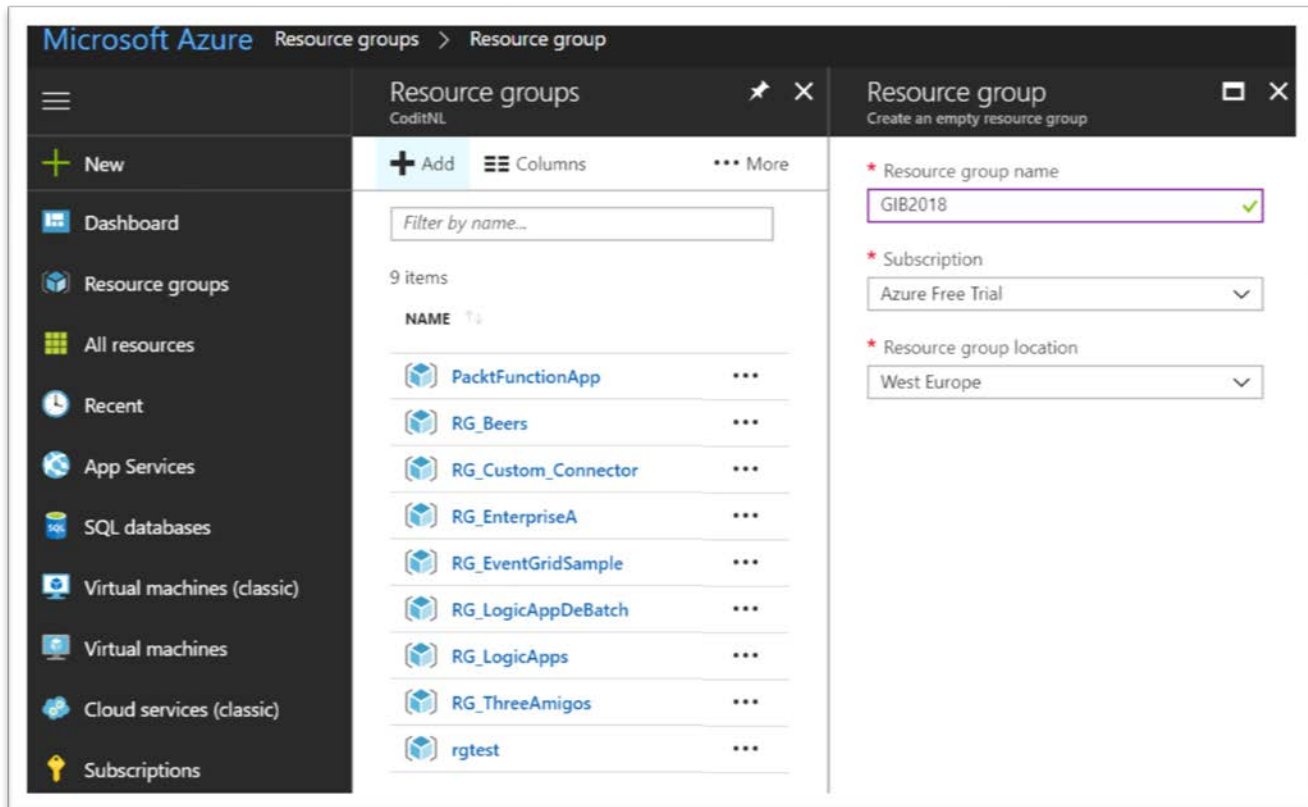
- Create a resource group with a name like **gibc2018-<initials>-rg<lab_no>**
- Provision a Storage Account with a name like **storage -<gib2018>-<labno>**
- Create a container with a name like **images**
- Provision Cognitive Service Computer Vision API with a name like **gib2018-<vision>-location**
- Provision a Function App with a name like **gib2018-<functions>-location**
- Create a Function with a name like **analyseimages**
- Create a RequestBin
- Provision a Logic App with a name like **gib2018-image-notification**
- Create Logic App definition
- Configure Event Grid Topic
- Test the solution

Lab duration: **60 minutes**

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. 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 (**gibc2018-<initials>-rg<lab_no>**), specify a Subscription, and a location. Note that the Event Grid Service we will use is globally available in the following regions: West US, East US, West US 2, East US 2, West Central US, Central US, West Europe, North Europe, Southeast Asia, and East Asia.



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.

Provision a Storage Account

Within the resource group, you can quickly 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 the screenshot below, you will see the details you need to specify.

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

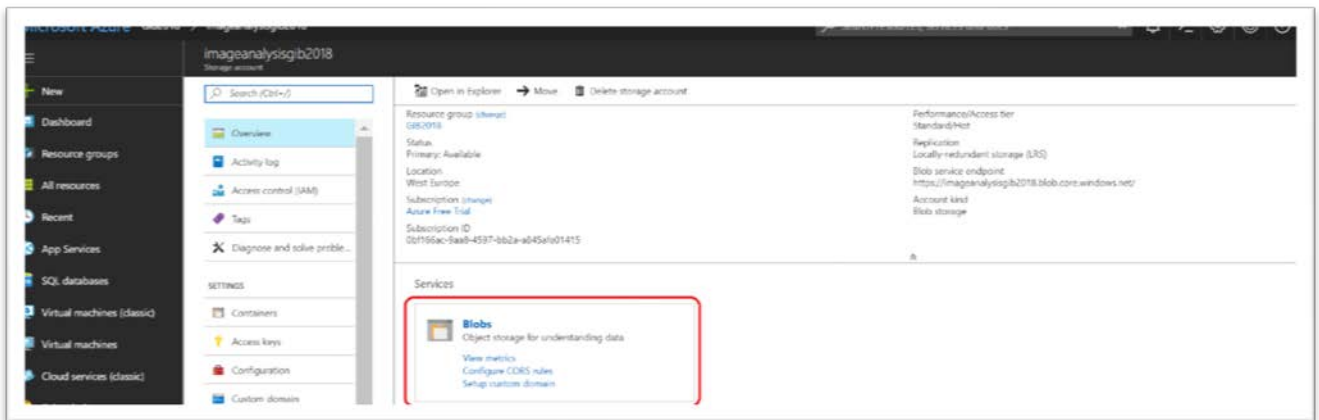
Automation options

10. Choose a useful unique name (**storage -<gib2018>-<labno>**).
11. Click **Create** once finishing specifying. Not keep every Azure resource in the same location to prevent unnecessary network charges for traffic between locations.

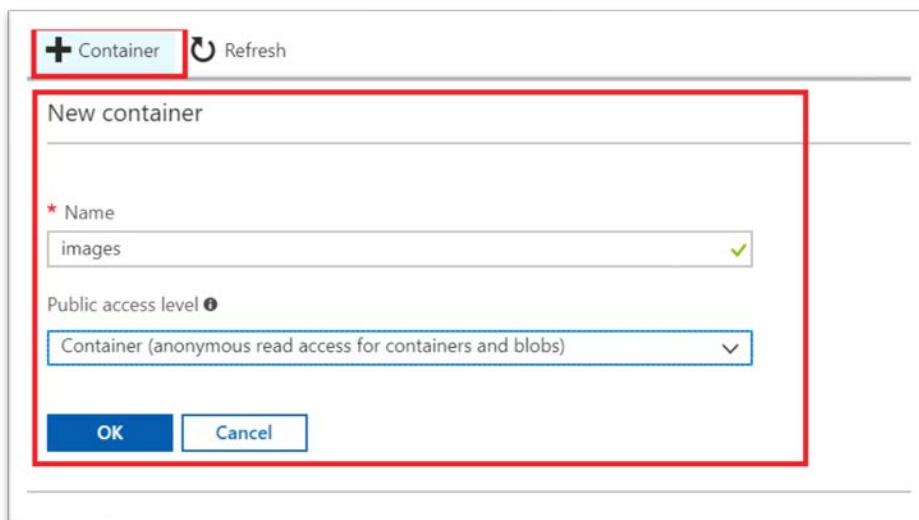
Create a container

Once the Storage Account (blob) is available for you, 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 images as the name for the container in the window that will appear.



6. Change the public access level to read access for containers and blobs only. This is necessary to give the function access your blob. If you keep it private then the function cannot find the blob and give you an **HTTP 404 Not Found**.
7. Click **OK**.

Provision Cognitive Service Computer Vision API

In this lab, we will leverage the pre-built AI capability of the 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*.
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 Cognitive Service like in the screenshot below. Choose a unique name (**gib2018-<vision>-location**), select Computer Vision API in API Type, the location, the cheapest pricing tier, and the correct resource group (Step 1).

The screenshot shows the 'Create Cognitive Services' dialog box. It contains the following fields and options:

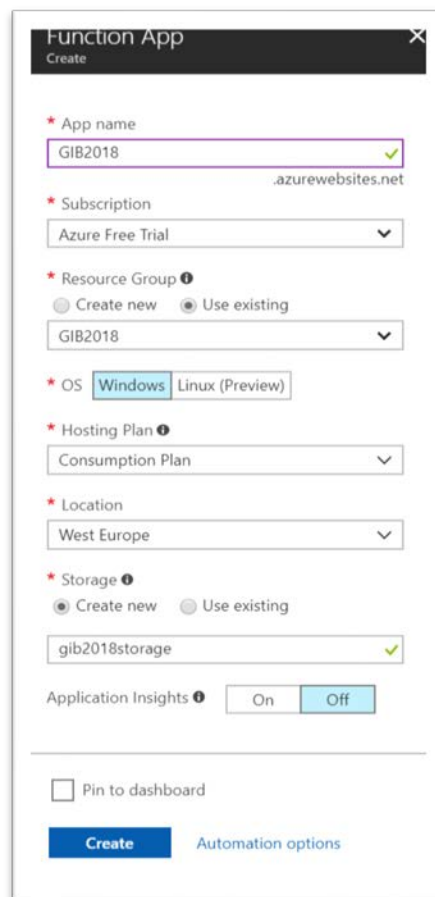
- Name:** GIB2018FaceDetection (with a green checkmark icon)
- Subscription:** Azure Free Trial (dropdown menu)
- API type:** Computer Vision API (dropdown menu)
- Location:** West Europe (dropdown menu)
- Pricing tier:** F0 (20 Calls per minute, 5K Calls per month) (dropdown menu, with a link to 'View full pricing details')
- Resource group:** GIB2018 (dropdown menu, with radio buttons for 'Create new' and 'Use existing' where 'Use existing' is selected)
- Pin to dashboard:** checkbox (unchecked)
- Create:** blue button
- Automation options:** text link

9. Click **Create** after you have specified the details.
10. In Cognitive Service select **Keys** is the pane.
11. **Copy** Key1 into a notepad.

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 resources 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 (**gib2018-<functions>-location**), select resource group, the location, Consumption Plan, and provide a name for a new Storage Account.



The screenshot shows the 'Function App Create' form in the Azure portal. The form is titled 'Function App Create' with a close button (X) in the top right corner. It contains the following fields and options:

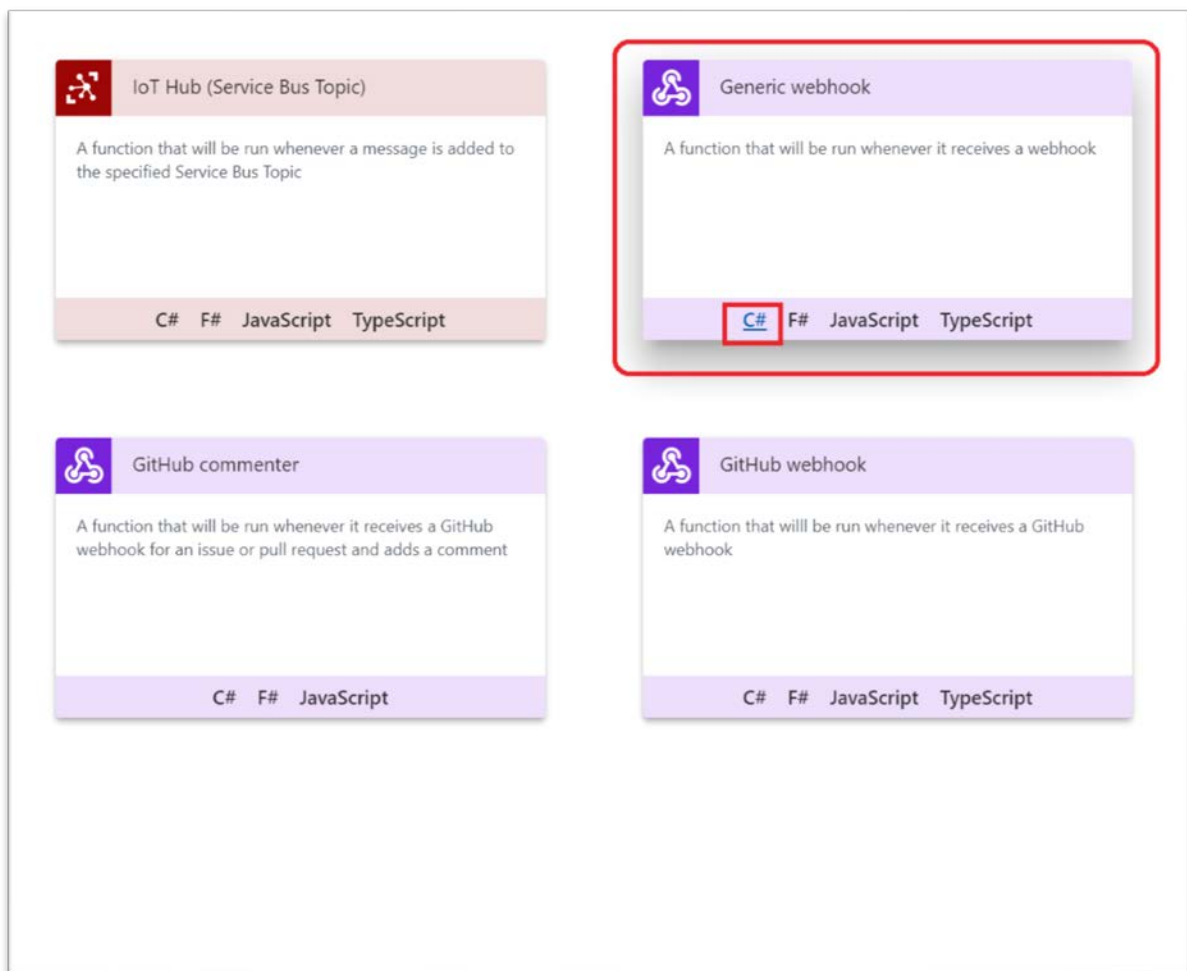
- * App name:** A text input field containing 'GIB2018' with a green checkmark icon to its right. Below the field, the text '.azurewebsites.net' is visible.
- * Subscription:** A dropdown menu showing 'Azure Free Trial'.
- * Resource Group:** A section with two radio buttons: 'Create new' (unselected) and 'Use existing' (selected). Below the radio buttons is a dropdown menu showing 'GIB2018'.
- * OS:** Two buttons: 'Windows' (selected) and 'Linux (Preview)'.
- * Hosting Plan:** A dropdown menu showing 'Consumption Plan'.
- * Location:** A dropdown menu showing 'West Europe'.
- * Storage:** A section with two radio buttons: 'Create new' (selected) and 'Use existing' (unselected). Below the radio buttons is a text input field containing 'gib2018storage' with a green checkmark icon to its right.
- Application Insights:** A section with a label 'Application Insights' and two buttons: 'On' and 'Off' (selected).
- Pin to dashboard:** A checkbox that is currently unchecked.
- Create:** A blue button at the bottom left.
- Automation options:** A link at the bottom right.

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

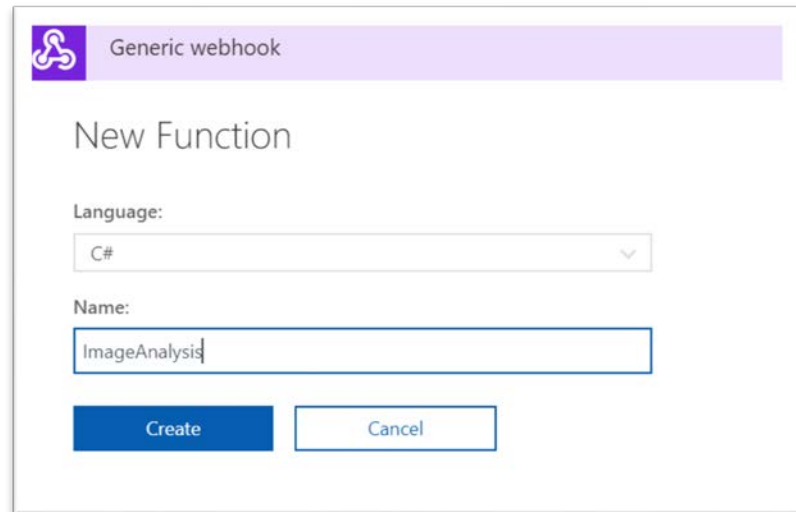
Create a Function.

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 App Service (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 adding 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-hand side. Provide a name (**analyseimages**) for your function and click **Create**.



6. A new pane will appear with some default code.
7. Click **Run** to examine the behavior. We will change this code for our objective in this lab.
8. Replace the code with the code below. This code handles the event from event grid that an image has been added, retrieves the image and sends it to cognitive services to get a description of what it contains.
9. Note that the <https://westeurope.api.cognitive.microsoft.com/vision/v1.0/describe?> URL you might need to change the location westeurope to your location!

```
#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}");
```

```

//event data is an Json Array
JSONArray data = JSONArray.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();
    byte[] image = webClient.DownloadData(imageUrl);

    //analyze image
    imageInfo = AnalyzeImage(image);

    //write to the console window
    log.Info(imageInfo);
}

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

private static string AnalyzeImage(byte[] imageLocation) {
    var client = new HttpClient();
    var queryString = HttpUtility.ParseQueryString(string.Empty);

    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "your key");

    queryString["maxCandidates"] = "1";
    var uri = " https://westeurope.api.cognitive.microsoft.com/vision/v1.0/describe?"
    + queryString;
    HttpResponseMessage response;

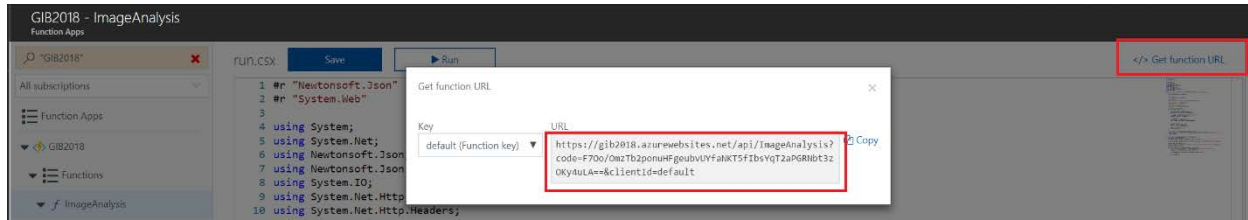
    using (var content = new ByteArrayContent(imageLocation)) {
        content.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");
        response = client.PostAsync(uri, content).Result;

        string imageInfo = response.Content.ReadAsStringAsync().Result;

        return imageInfo;
    }
}

```

10. Click **Save**.
11. 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).
12. **Copy** Key1 from notepad and go to back to your function.
13. Paste it into the "**your key**" section and **Save**.
14. Click **Get Function URL**.



15. **Save** the **URL** in a notepad.

Create a Request Bin.

In this lab, we will use Request Bin, which will enable us to see what your HTTP client is sending or to inspect and debug WebHook requests. We will use this to see the raw event pushed from Azure Storage once an image has been uploaded to a container through Storage Explorer.

1. Go to <https://gib2018requestbin.herokuapp.com/>
2. Click **Create a Request Bin**.
3. A new window will appear with a Bin URL.

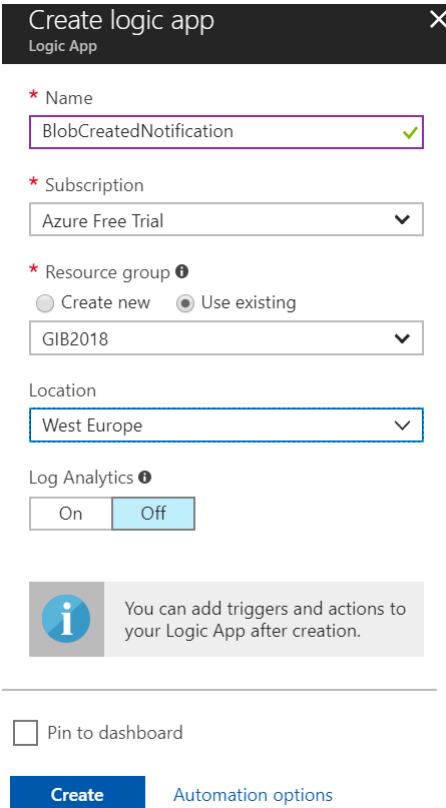


4. **Copy** the **URL** in a notepad.

Provision a Logic App.

The final part of our solution is a Logic App. A Logic App is a service in Azure that enables you to build a flow definition based on actions. The flow starts with an action trigger followed by one or more actions. We will provision a Logic App and build a definition to handle the BlobCreated event by 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 (**gib2018-image-notification**) in the Create Logic App pane, the subscription, resource group, and location.



Create logic app

Logic App


* Name
BlobCreatedNotification ✓

* Subscription
Azure Free Trial ▼

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

Location
West Europe ▼

Log Analytics ⓘ

 You can add triggers and actions to your Logic App after creation.

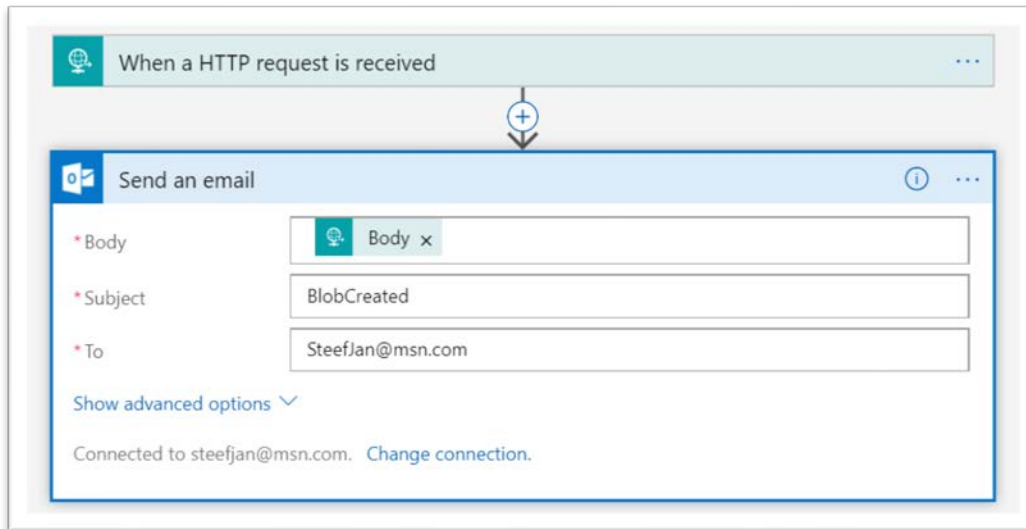
☐ Pin to dashboard

[Automation options](#)

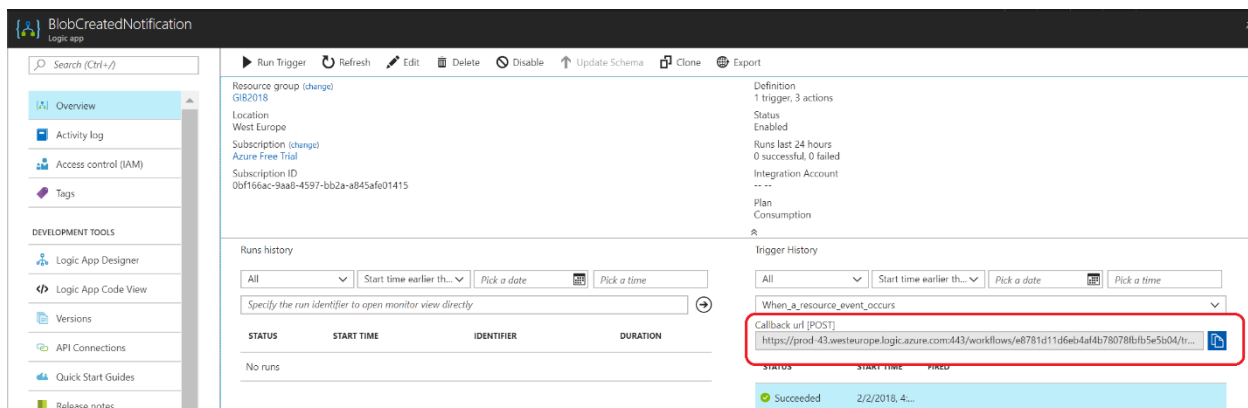
[Create](#)

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 choose any of the pre-built ones.
10. In the first action (trigger) enter: *Http*
11. Choose **Request**.
12. Next, add an action. In the second and final action enter: *Outlook*

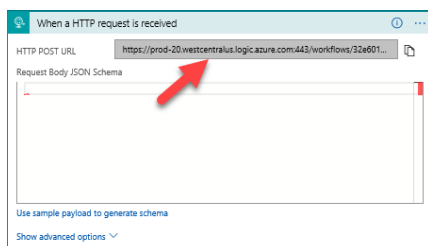
13. Select **Outlook.com**.
14. Select **Outlook.com – Send an email**.
15. Sign into your **Outlook.com** (with your Live/Microsoft account). You need to assign permission to the Logic App to use your email account.
16. Next specify To, Subject, and Body. You sent an email to yourself, Subject is from Event Grid, and Body (see screenshot below).



17. **Save** the Logic App.
18. In the overview copy the Callback URL (POST) and paste it in a notepad.



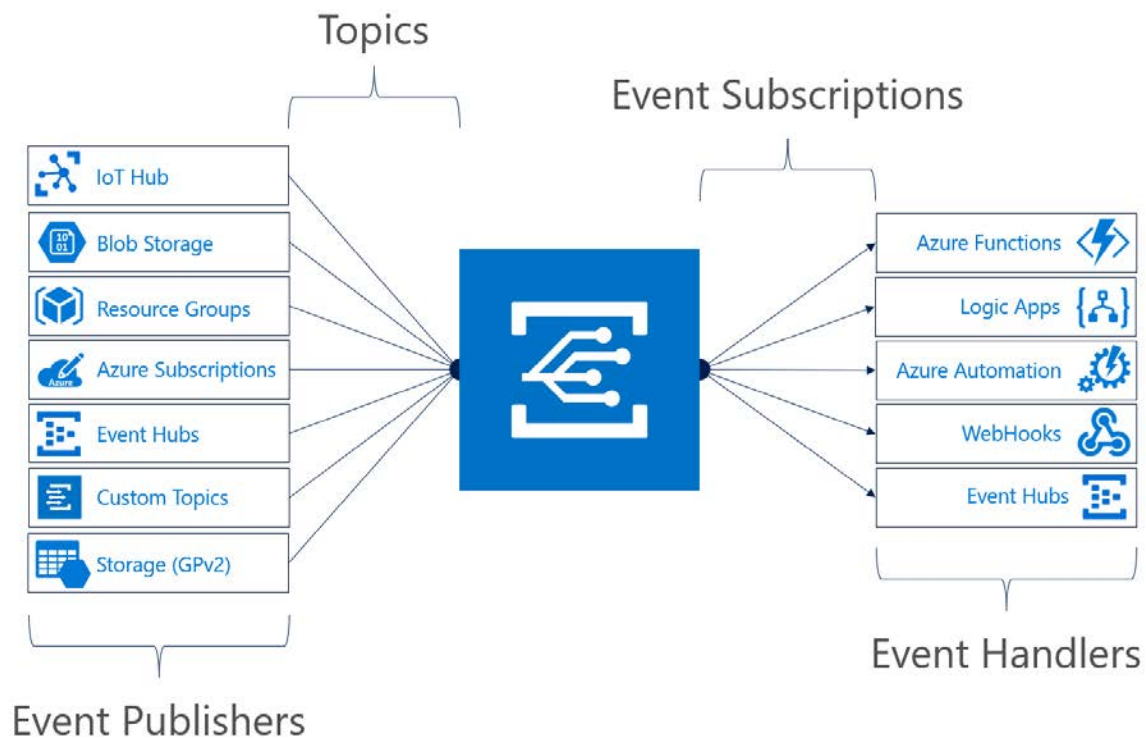
NOTE: If you don't see this callback URL on the Overview screen, you can get it from the trigger shape of the Logic App designer instead:



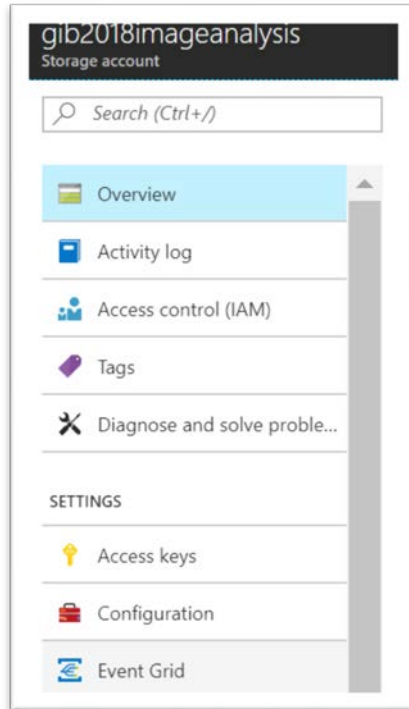
Configure the Event Grid Topic in the Azure Storage Account.

The resources for this lab has been set up 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 have to continuously poll an application or service for a change. Now it can receive an event and continue.

The concept of Event Grid evolves 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 are for example 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, and Event Hubs.



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




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 suffix filter like the screenshot below.

Prefix filter: **/blobServices/default/containers/<your container name>**

Suffix filter: **.jpg**

This is a subscription to the Function that will handle the **BlobCreated** event. Here you will need the URL of your Function (Step 6).



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

5. Click **Create** in the bottom right corner of the pane.
6. Hit **Refresh** and the subscription should appear.
7. Click **+ Event Subscription** for the next subscription we will configure.
8. A pane will appear, and here you specify the details for Request Bin (Step 7), which will subscribe to all the events.
9. Click **Create**.
10. Hit **Refresh** and the subscription should appear.
11. Click **+ Event Subscription** for the last subscription we will configure.
12. This will be the subscription to the Logic App endpoint.

Create Event Subscription

Event Grid

×

*

Name

LogicApp

✓

☐

Subscribe to all event types

*

Event Types

Blob Created

▼

Subscriber Type

Web Hook

▼

*

Subscriber Endpoint

https://prod-11.westeurope.logic.azure.co...

✓

Prefix Filter

/blobServices/default/containers/images/

Optional

Suffix Filter







.jpg

Optional


☐

Filter Case Sensitive

13. You should now have three subscriptions 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
 RequestBin	WebHook	https://requestb.in/r14w0tri			All	 Metrics
 LogicApp	WebHook	https://prod-11.westeurope.logic.azure.com/workflows/864e56e7014...	/blobServices/default...	.jpg	Microsoft.Storage.BlobCreated	 Metrics

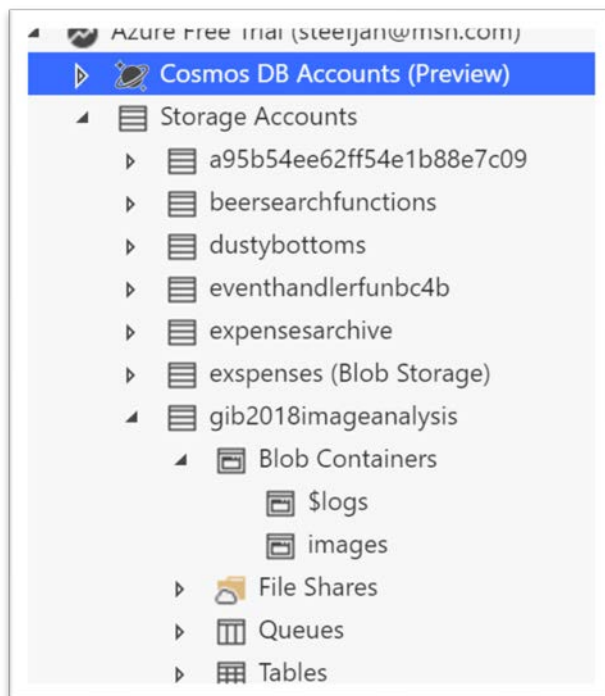
20


 GLOBAL INTEGRATION BOOTCAMP

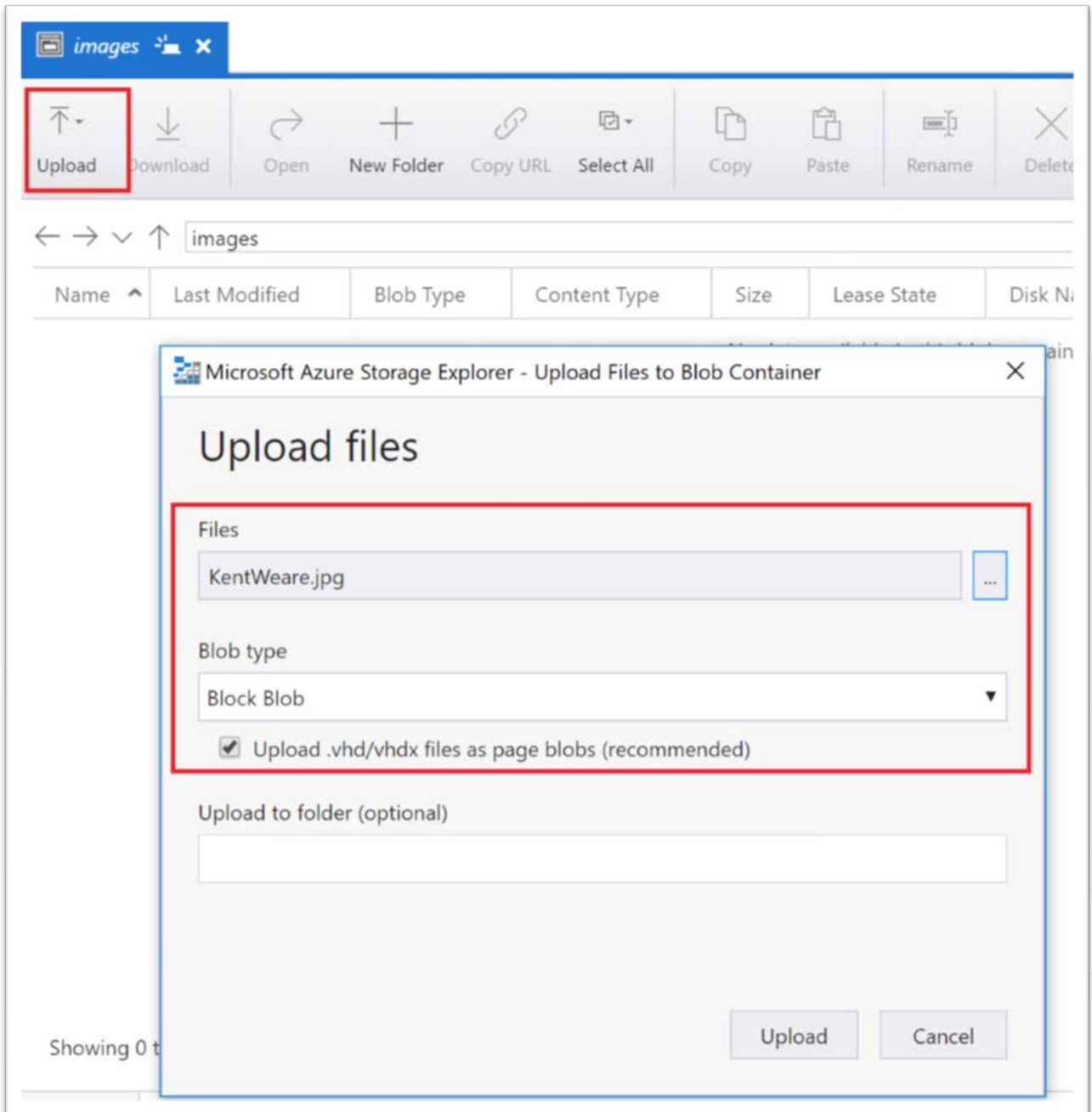
Test the solution.

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

1. Start the **Azure Storage Explorer**.
2. Note that you have to log in 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 Request Bin and inspect.


```

"api": "PutBlob",
"clientRequestId": "b41af620-0828-11e8-a417-2b437bef514e",
"requestId": "5f905981-001e-0062-0635-9c145a000000",
"eTag": "0x8D56A4C98E6E7A8",
"contentType": "image/jpeg",
"contentLength": 13034,
"blobType": "BlockBlob",
"url": "https://gib2018imageanalysis.blob.core.windows.net/images/KentWeare.jpg",
"sequencer": "00000000000000AF00000000001E8E0F",
"storageDiagnostics": {
  "batchId": "1b9ae595-1561-431f-be3c-bfd840e81b33"
}
},
"dataVersion": "",
"metadataVersion": "1"
}]
blobEventData : {
  "api": "PutBlob",
  "clientRequestId": "b41af620-0828-11e8-a417-2b437bef514e",
  "requestId": "5f905981-001e-0062-0635-9c145a000000",
  "eTag": "0x8D56A4C98E6E7A8",
  "contentType": "image/jpeg",
  "contentLength": 13034,
  "blobType": "BlockBlob",
  "url": "https://gib2018imageanalysis.blob.core.windows.net/images/KentWeare.jpg",
  "sequencer": "00000000000000AF00000000001E8E0F",
  "storageDiagnostics": {
    "batchId": "1b9ae595-1561-431f-be3c-bfd840e81b33"
  }
}
imageUrl : https://gib2018imageanalysis.blob.core.windows.net/images/KentWeare.jpg
{"description":{"tags":["person","man","indoor","smiling","holding","posing","photo","front","camera","standing","food","table","glasses","shirt","wearing","large","young","phone","sign","red","plate","white","blue"],"captions":[{"text":"a man smiling for the camera","confidence":0.9801141977309823}]},"requestId":"102095c8-6bc2-471c-9eb6-7b95c38c0f22","metadata":{"height":200,"width":200,"format":"Jpeg"}}

```

13. Finally, you should also receive an email.

BlobCreated



Steef-Jan Wiggers

5:43 PM

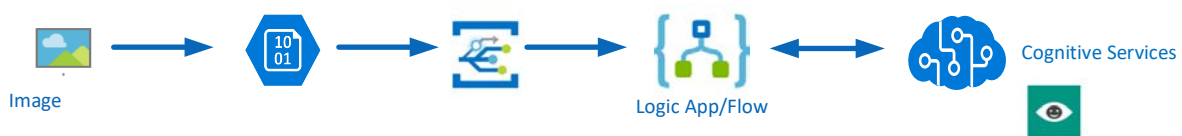


To: Steef-Jan Wiggers

```
[{"topic":"/subscriptions/0bf166ac-9aa8-4597-bb2a-a845afe01415/resourceGroups/GIB2018/providers/Microsoft.Storage/storageAccounts/gib2018imageanalysis", "subject":"/blobServices/default/containers/images/blobs/KentWeare.jpg", "eventType":"Microsoft.Storage.BlobCreated", "eventTime":"2018-02-02T16:36:40.1742479Z", "id":"6772e6cc-001e-00ac-5e44-9cc5d4069a70", "data":{"api":"PutBlob", "clientRequestId":"3e64d860-0837-11e8-a417-2b437bef514e", "requestId":"6772e6cc-001e-00ac-5e44-9cc5d4000000", "eTag":"0x8D56A5B22D4F28F", "contentType":"image/jpeg", "contentLength":13034, "blobType":"BlockBlob", "url":"https://gib2018imageanalysis.blob.core.windows.net/images/KentWeare.jpg", "sequenceNumber":"0000000000000000AF00000000001FB453", "storageDiagnostics":{"batchId":"81b60cfe-dacc-4ff9-b392-55d032013be4"}}, "dataVersion":"","metadataVersion":"1"}]
```

Explanation of the Lab (What have we done)

In this lab so far, you have built a solution using Event Grid, Functions, Cognitive Service and Azure Storage. With Event Grid, GA since the 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.

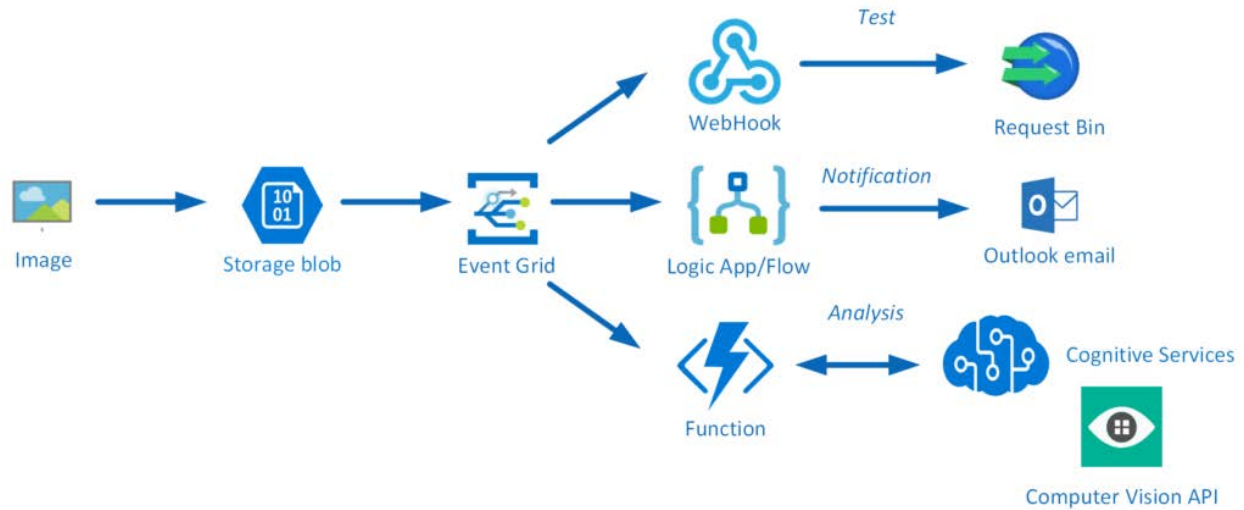


We have done something similar, yet we also used a Function as event handler and included Request Bin to see the actual raw event.

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 to the subscriber side too with Azure Data Factory, Service Bus and Storage Queues for instance.

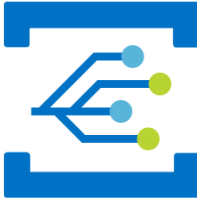
We have uploaded an image to a Storage blob container, which will be the event source (publisher). Subsequently, the Storage blob container belongs to a Storage Account containing the Event Grid capability. And the Event Grid has three subscribers, a WebHook (Request Bin) to capture the output of the event, 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 for the event and use it to analyze the actual image.



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://gib2018.azurewebsites.net/api/ImageAnalysis	/blobServices/default...	.jpg	Microsoft.Storage.BlobCreated	Metrics
RequestBin	WebHook	https://requestb.in/ri4w0tri			All	Metrics
LogicApp	WebHook	https://prod-11.westeurope.logic.azure.com/workflows/864e56e7014...	/blobServices/default...	.jpg	Microsoft.Storage.BlobCreated	Metrics

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 8.



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**
 ✓

☐ Subscribe to all event types

*** Event Types**
 ▼

Subscriber Type
 ▼

*** Subscriber Endpoint**
 ✓

Prefix Filter
 Optional

Suffix Filter
 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"
}]

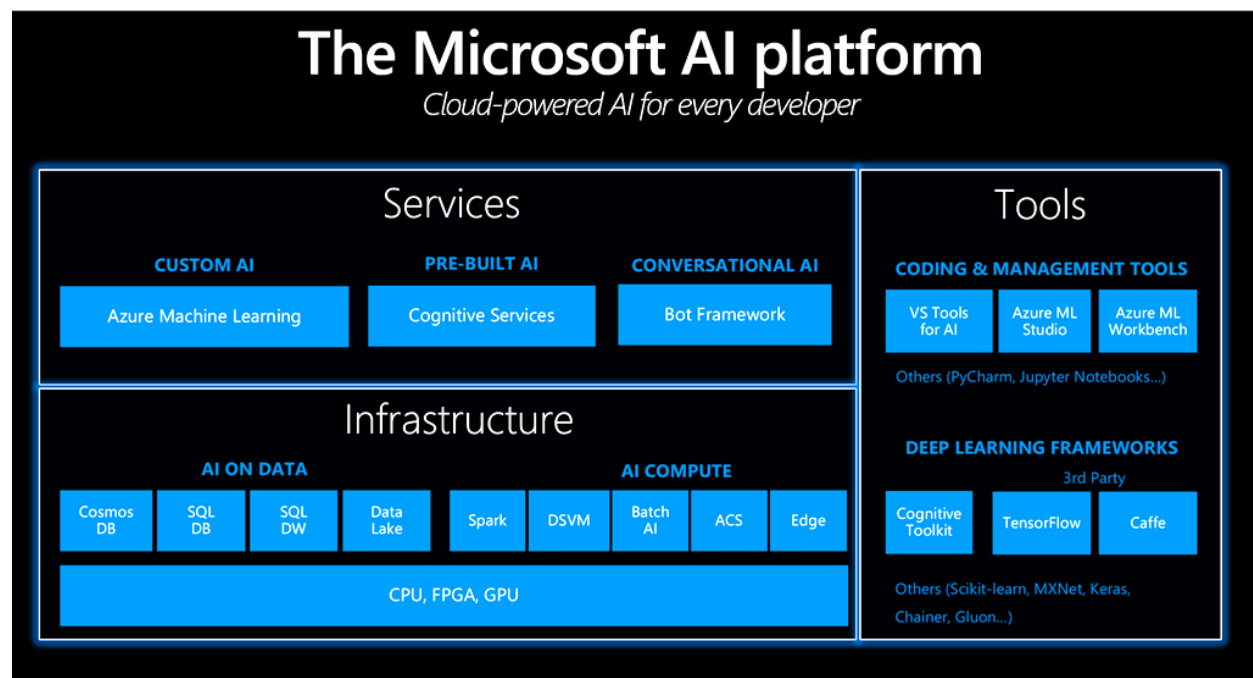
```

If you 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/>
- MSDN Magazine: <https://msdn.microsoft.com/en-us/magazine/mt829271.aspx>

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

