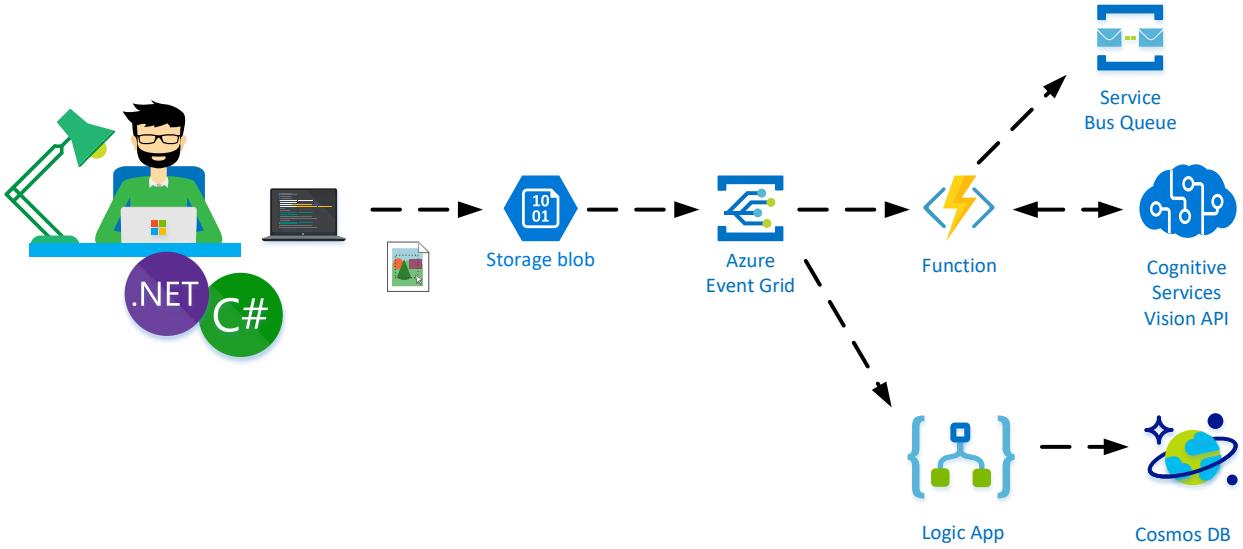
Lab 4 - Building a Smart solution

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 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/

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:

- Create a container with a name like **images2.**
- Add collection to Cosmos DB.
- Provision a Logic App with a name like **StoreImage.**
- Create Logic App definition.
- Create a Function with a name like **analyseimages.**
- Test the solution

Lab duration: **60 minutes**.

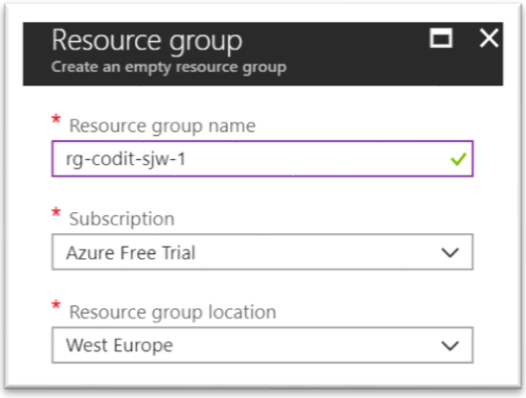
Created by Steef-Jan Wiggers, Microsoft Azure MVP, Codit Domain Lead Azure

Contact steef-jan.wiggers@codit.eu or twitter @steefJan

Step 1 - Create a resource group (optional)

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 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 (**rg- azurethursday -<initials>-<lab_no>)**, specify a Subscription, and a location.

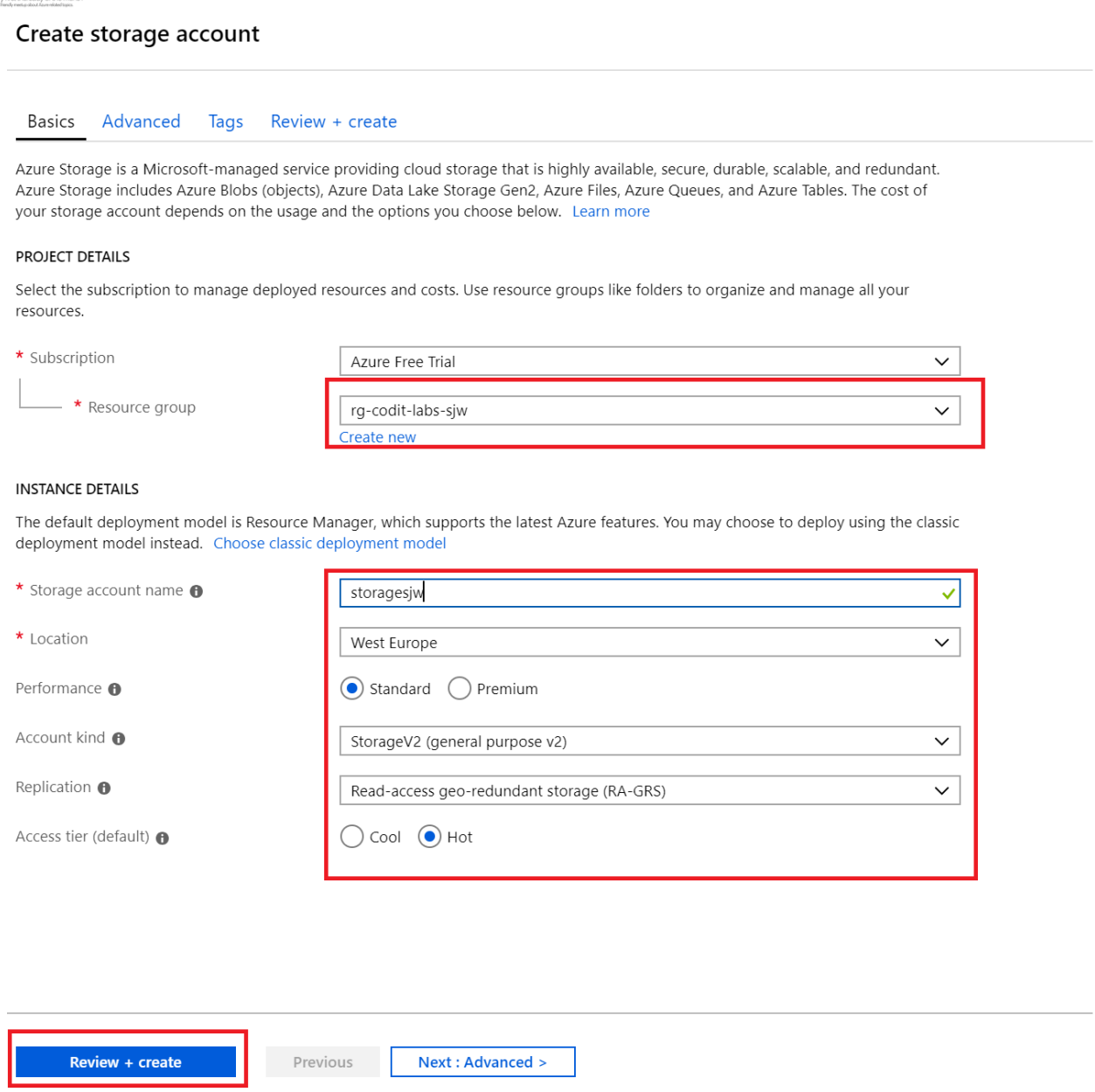


4. Finally, click **Create** and a resource group will be created for you.
5. In the top right corner, a pop-up will appear, which you can click to go to your resource group.

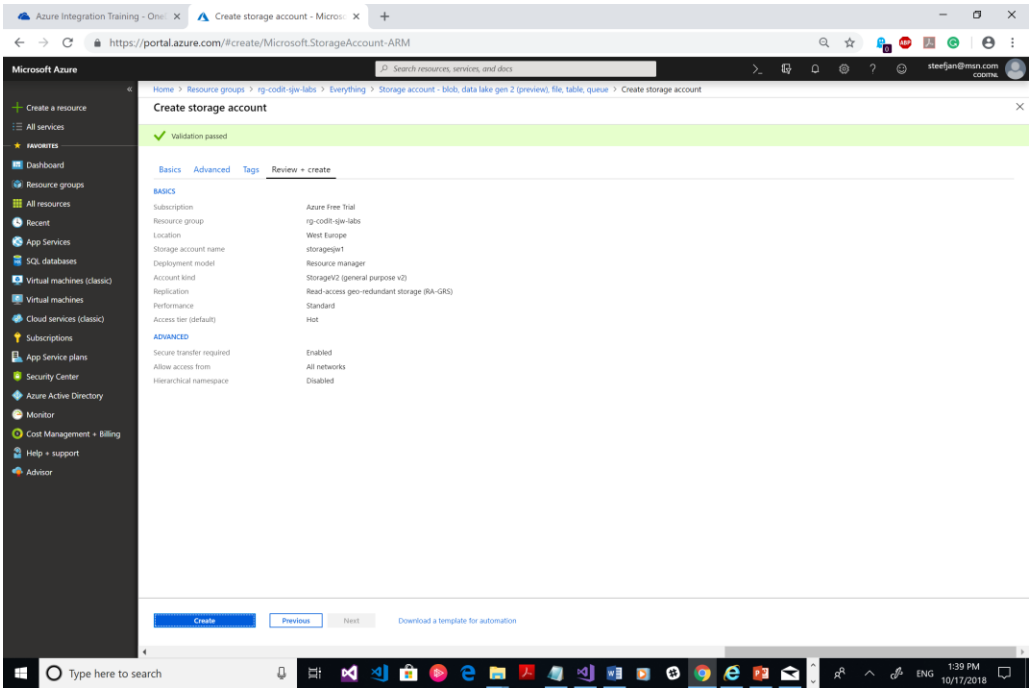
Step 2 – Provision a Storage Account

Within the resource group, you can 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.



10. Choose a useful unique name (**storage<intials><labno>)**.
11. Click **Create + Review** once finishing specifying.



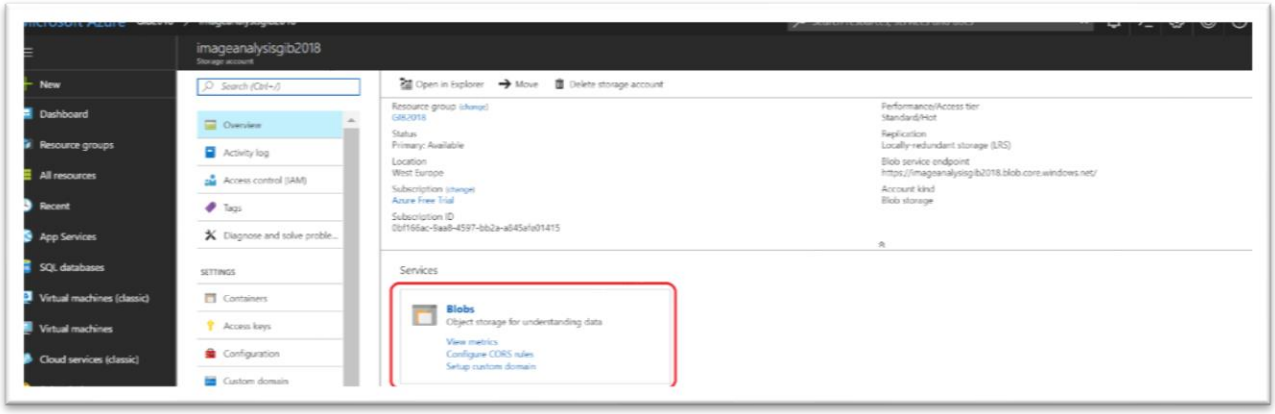
12. Click **Create**.

    Note: Keep every Azure resource in the same location to prevent unnecessary network charges for traffic between regions.

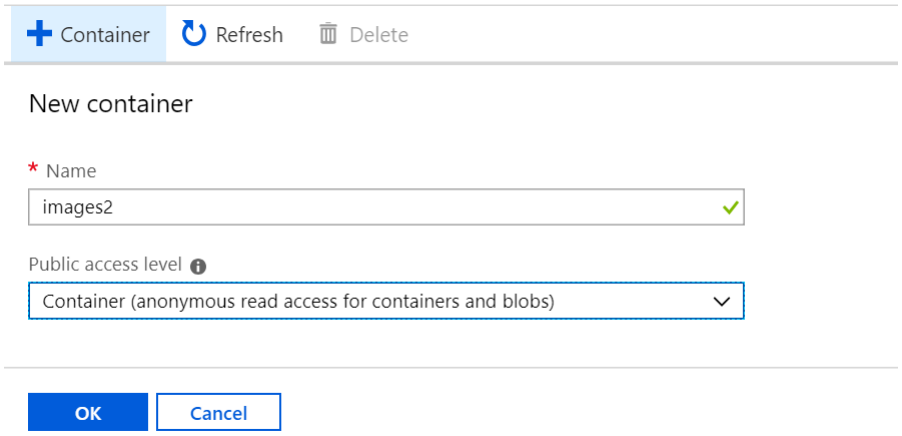
Step 3 – Create a container

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

1. Click the Storage Account you created in a previous step.
2. Click **Blobs**.



3. Click on **+ Container**.
4. Specify images as the name for the container in the window that will appear.



5. Change the public access level to read access for containers and blobs only. This level 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**.
6. Click **OK**.

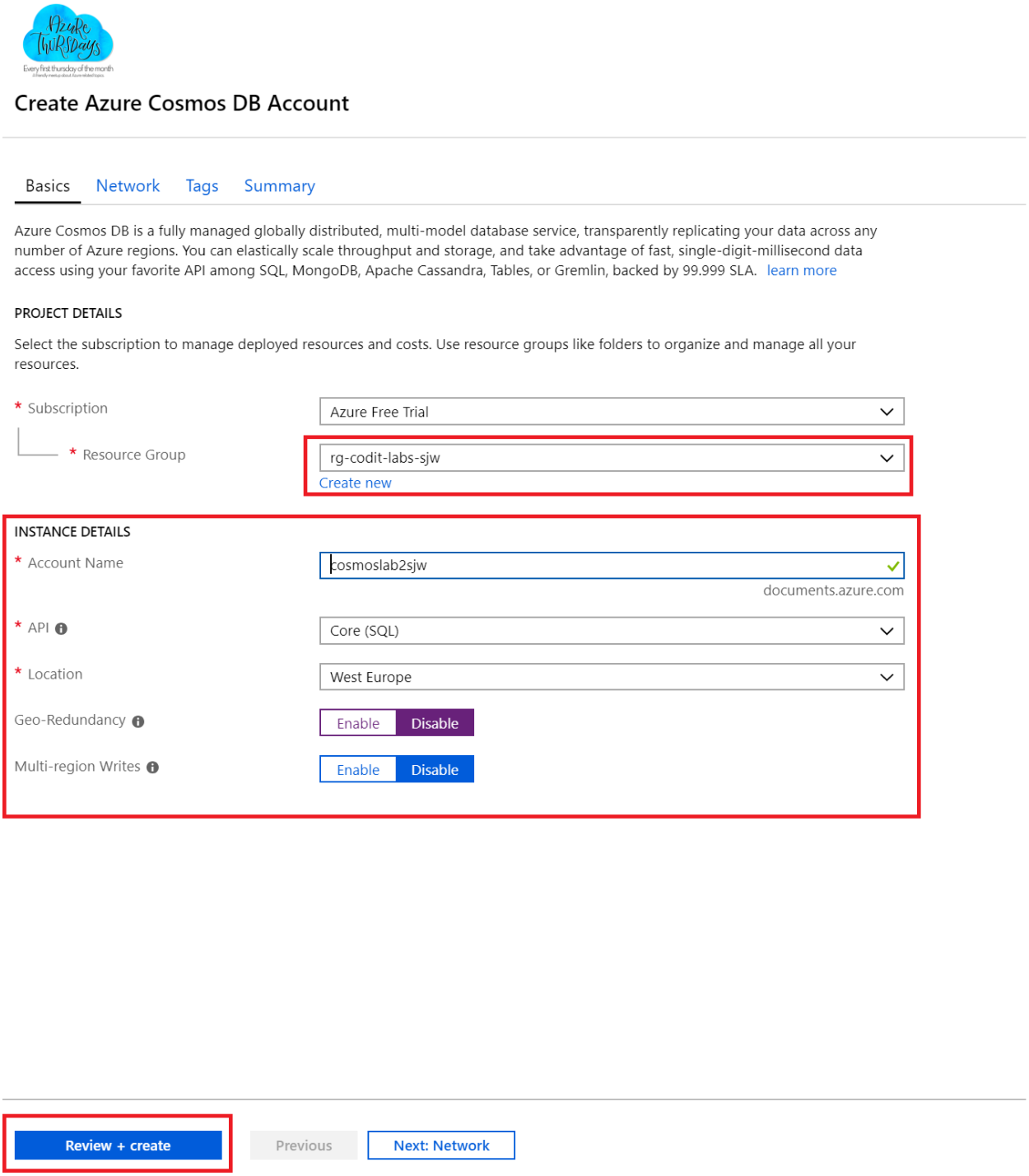
Step 4 – Provision a Cosmos DB Instance

Within the resource group, you can quickly add various types of Azure Resources. For this lab, we will need an instance of Cosmos DB.

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: *Azure Cosmos DB*.
5. Select Azure Cosmos DB and in the right pane that will appear click **Create**.
6. In the **New** Account page, enter the settings for the new Azure Cosmos DB account.

Setting	Value	Description
Account Name	*Enter a unique name*	Enter a unique name to identify this Azure Cosmos DB account. Because *documents.azure.com* is appended to the ID that you provide to create your URI, use a unique but identifiable ID. The ID can contain only lowercase letters, numbers, and the hyphen (-) character, and it must contain 3 to 50 characters.
API	Core(SQL)	The API determines the type of account to create. Azure Cosmos DB provides five APIs to suits the needs of your application: SQL (document database), Gremlin (graph database), MongoDB (document database), Azure Table, and Cassandra, each which currently require a separate account. Select **SQL** because in this quickstart you are creating a document database that is queryable using SQL syntax and accessible with the SQL API.
Subscription	*Your subscription*	Select Azure subscription that you want to use for this Azure Cosmos DB account.
Resource Group	Create new *Then enter the same unique name as provided above in ID*	Select **Create New**, then enter a new resource-group name for your account. For simplicity, you can use the same name as your ID.

Setting	Value	Description
Location	*Select the region closest to your users*	Select geographic location in which to host your Azure Cosmos DB account. Use the location that's closest to your users to give them the fastest access to the data.
Enable geo-redundancy	Disable	This creates a replicated version of your database in a second (paired) region.
Multi-region Writes	Disable	Multi-region write capability allows you to take advantage of the provisioned throughput for your databases and containers across the globe.



7. Click **Create**. The account creation takes a few minutes.

Create Azure Cosmos DB Account

Validation Success

Basics | Network | Tags | Summary

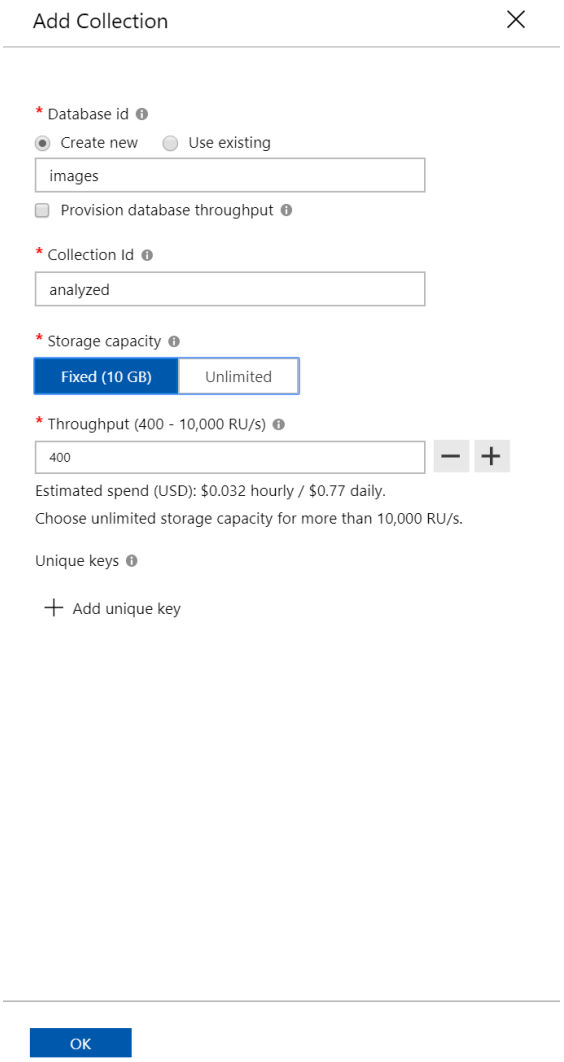
BASICS

Subscription	Azure Free Trial
Resource Group	rg-codit-labs-sjw
Location	West Europe
Account Name	(new) cosmoslab2sjw
API	DocumentDB
Geo-Redundancy	Disable
Multi-region Writes	Disable

Create | Previous | Next | Download a template for automation

Step 5 – Add a collection

1. Click the Cosmos DB instance you created in a previous step.
2. Click **Add Collection**.
3. Click **New Collection**.
4. Fill in details:
    a. Database id: images
    b. Collection id: analyzed
    c. Storage Capacity: **10 Gb**
    d. RU: **400**

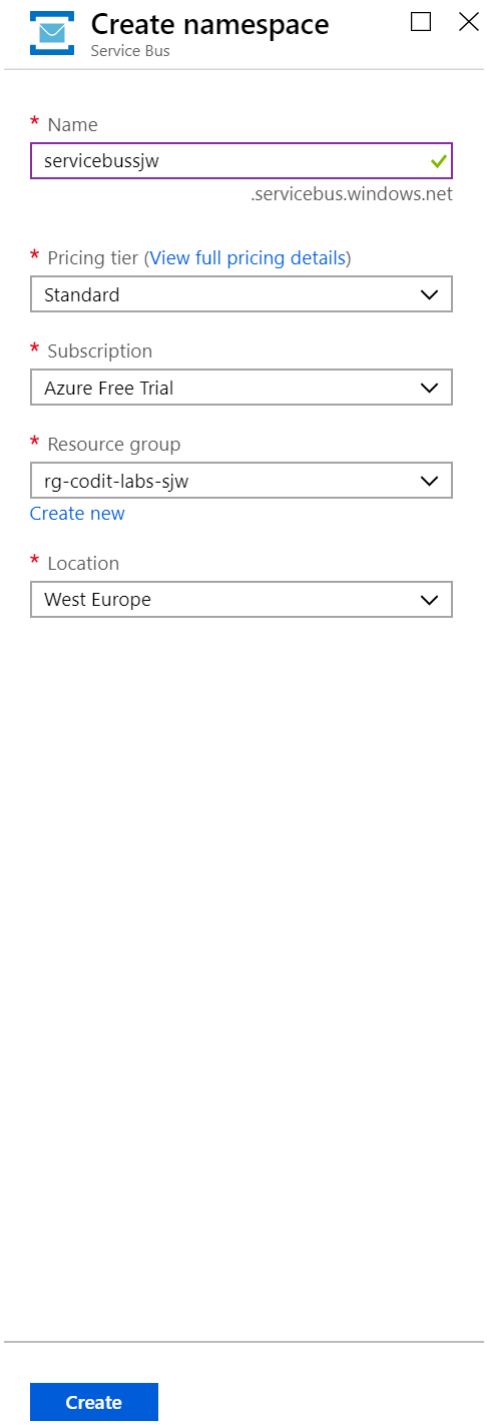


5. Click **Ok**.

Step 6 – Provision a Service Bus Namespace

Within the resource group, you can quickly add various types of Azure Resources. For this lab, we will need a service bus namespace.

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: *Service Bus*
5. Click **Create**
6. Fill the details like below (i.e. unique name, correct location, pricing details)



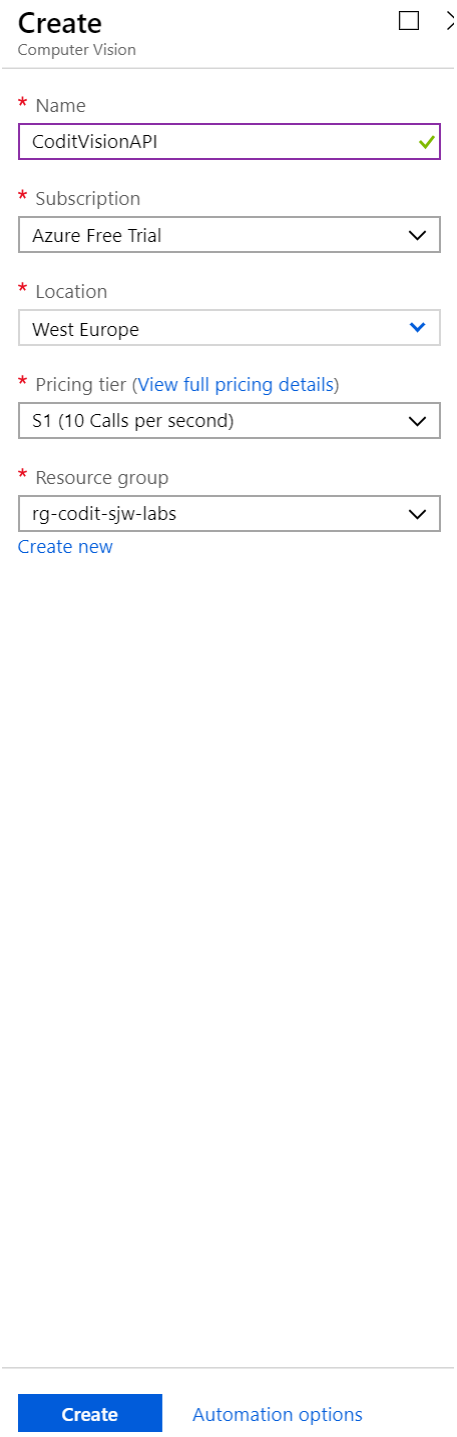
7. Click **Create**
8. Once the service namespace is provisioned open it.
9. Go to Shared Access policies.
10. Click **RootManageSharedAccessKey** and copy the primary connection string and paste it in a notepad.
11. Click Overview.
12. Click **+ Queue**

13. Give the queue a name: **outbound**

Step 7 – Create Computer Vision API Instance

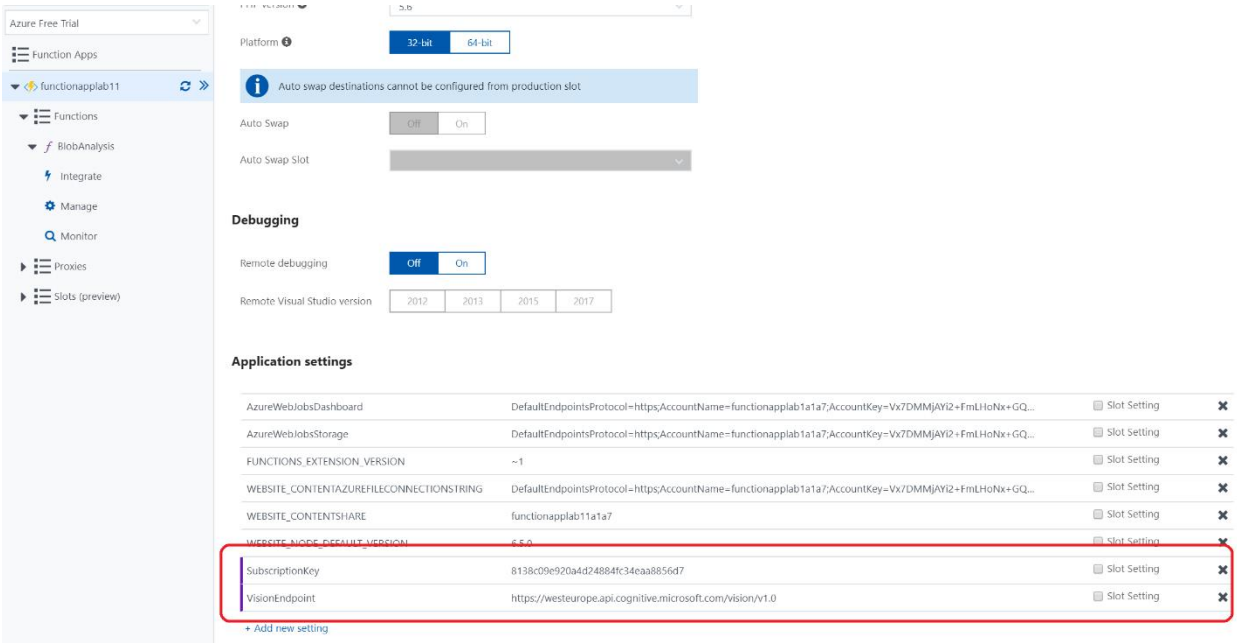
In this step, you will create a Computer Vision API instance, and then copy an access key and a base URL to notepad for step 9.

1. In the Azure Portal, go to your resource group, click **+ Create a resource**, in the search window enter **Computer Vision API**.
2. Select Computer Vision API and click create.
3. Name the service, select the correct location, tier, and resource group. Note that you need to choose F0 is possible.



4. Return to the blade for you resource group and click the **Computer Vision API** subscription that you just created.
5. Copy the URL under **Endpoint** into your favourite text editor so you can quickly retrieve it in a moment. The complete endpoint should be: https://westeurope.api.cognitive.microsoft.com/vision/v2.0
6. Then click **Show access keys**.
7. Click the **Copy** button to the right of **KEY 1** to copy the access key to the clipboard.

8. Return to the **Function App** in the Azure Portal and click the app name in the ribbon on the left. Then click **Application settings**.
9. Scroll down to the "**Application settings**" section. Add a new app setting named "**Subscription Key**" (without quotation marks), and paste the subscription key that is on the clipboard into the Value box. Then add a setting named "**VisionEndpoint**" and set its value to the endpoint URL you saved in Step 5. Finish up by clicking **Save** at the top of the blade.

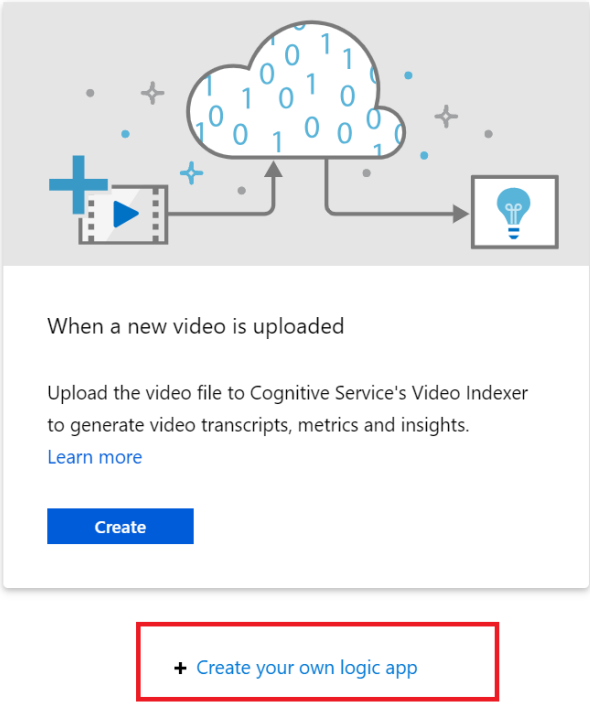


10. The app settings are now configured for your Azure Function.

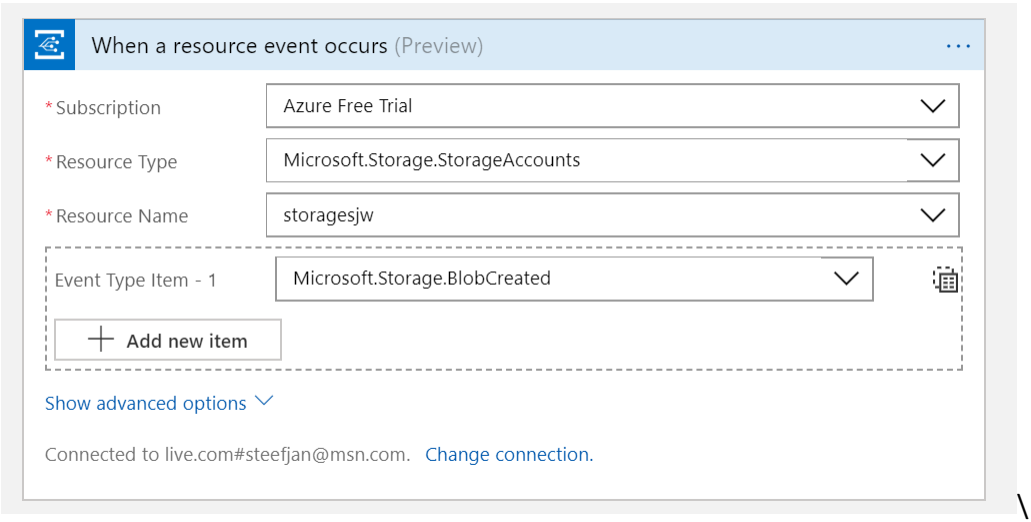
The work of writing and configuring the Azure Function is complete. Now comes the fun part: testing it out.

Step 8 – Create a Logic App

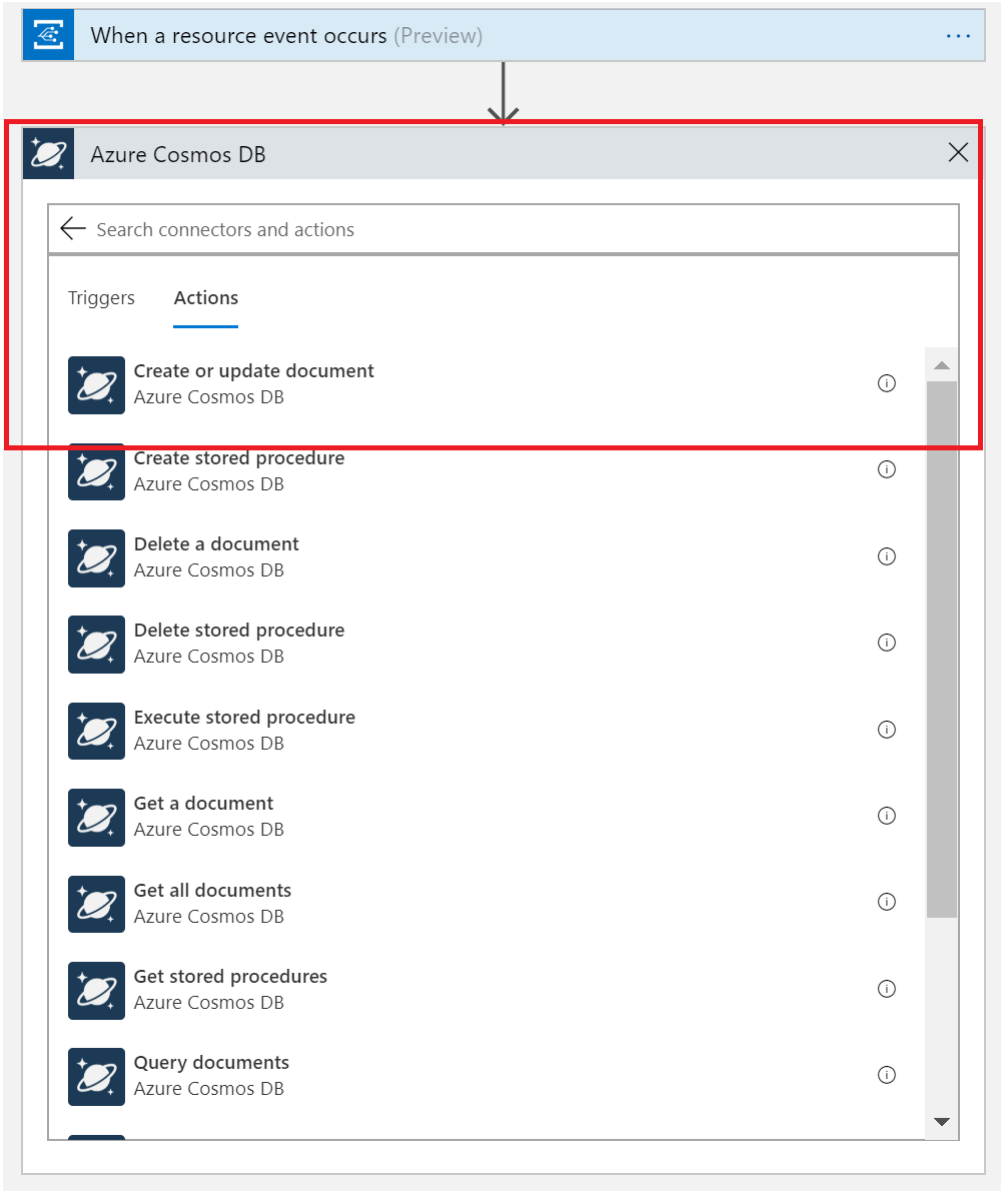
1. Go to the storage account.
2. Click **Events**.
3. Choose to *Create your own logic app* at the bottom.



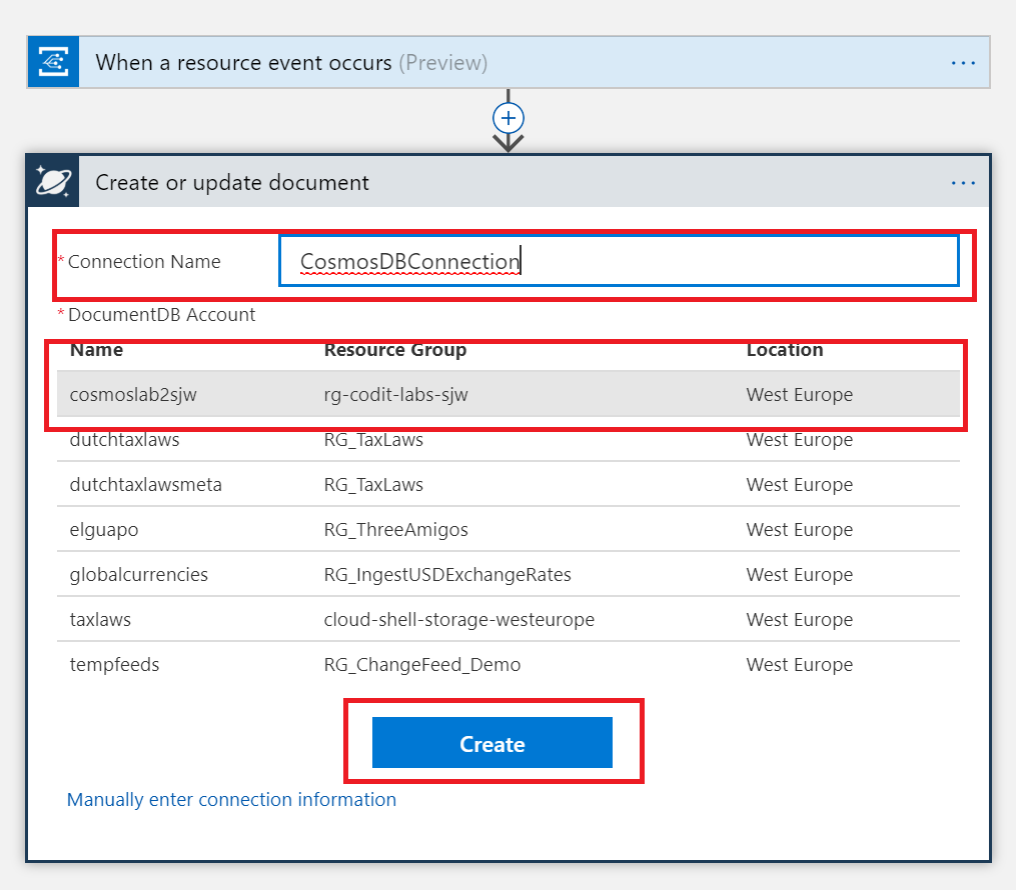
4. Click **Create**.
5. A Logic App Designer will appear.
6. Continue with Azure Event Grid Trigger in the Logic App Designer.
7. Click **Edit**.



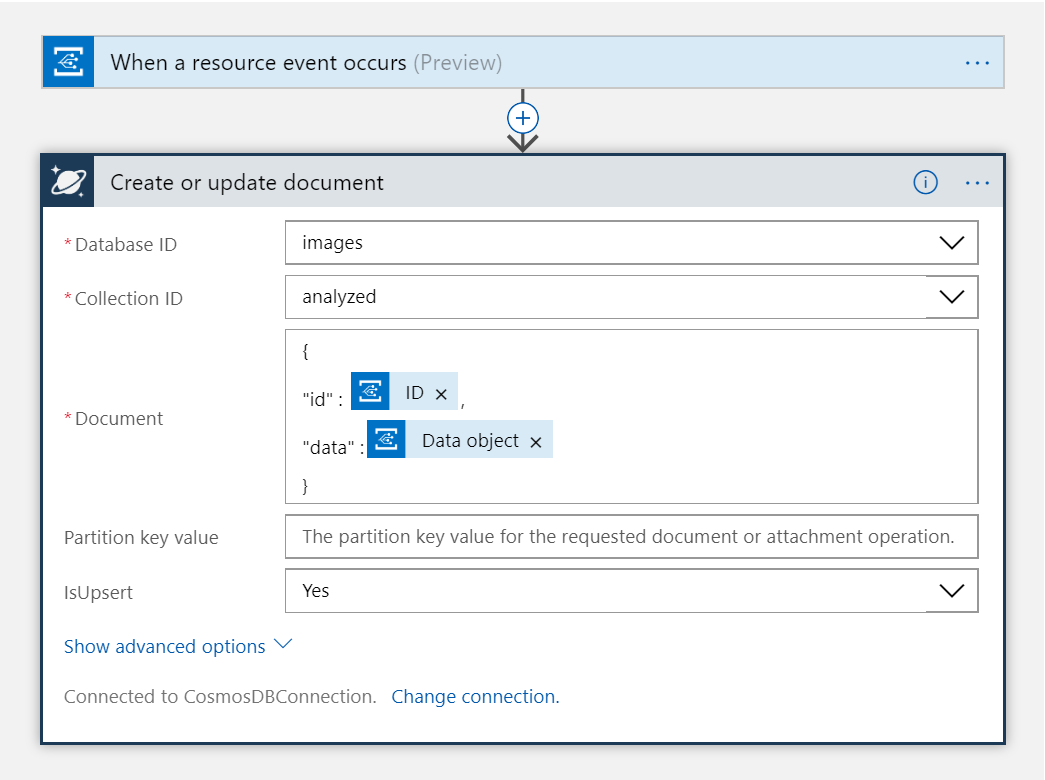
8. Select the resource name, which should be your storage account.
9. In the Event Type Item – 1 select Microsoft.Storage.BlobCreated.
10. Add action and search for CosmosDB.
11. Select Cosmos DB connector.
12. Choose create or update document action.



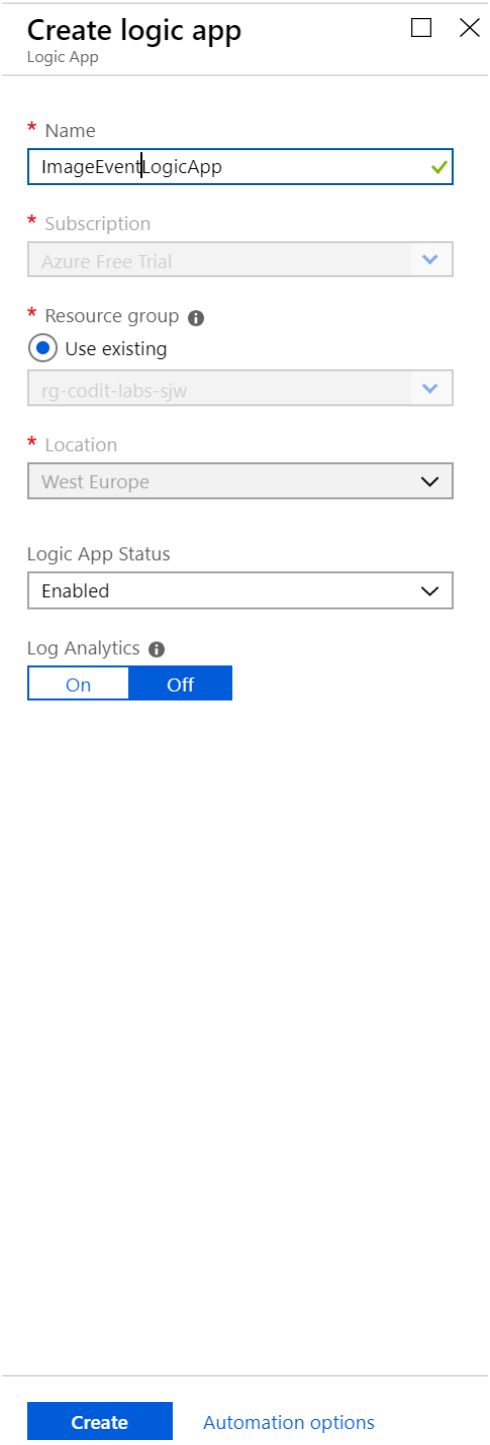
13. Provide a connection name and select your CosmosDB instance.



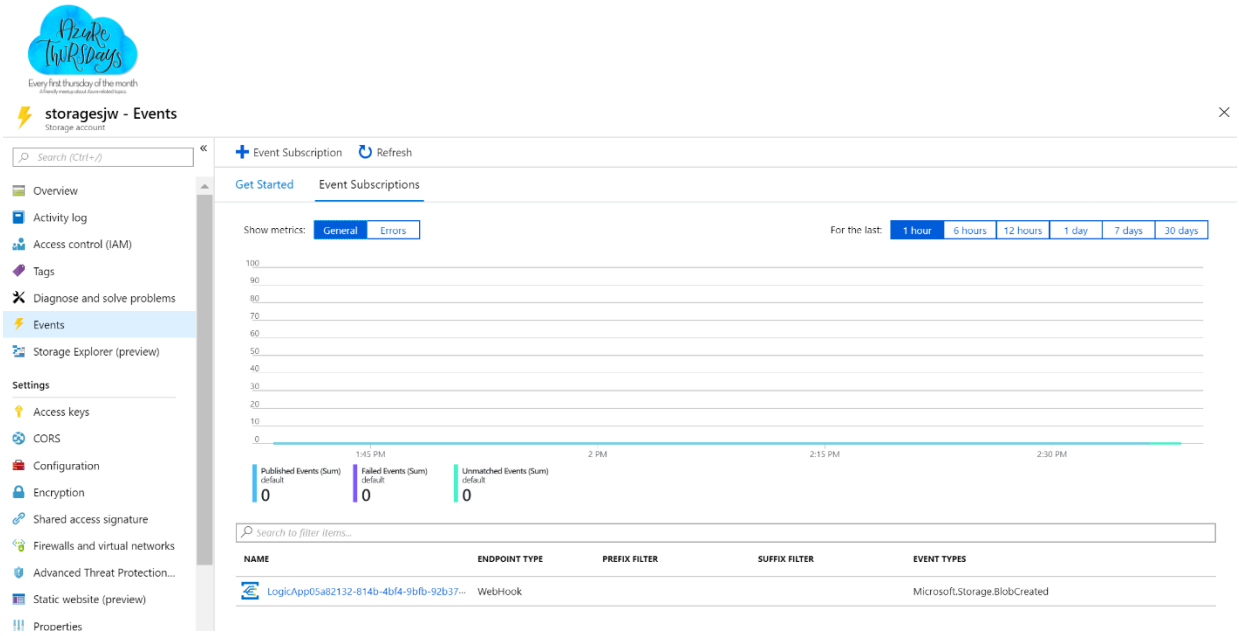
14. Click **Create**.
15. Configure the action as shown below.



16. Click **Save As**.
17. Specify Name.

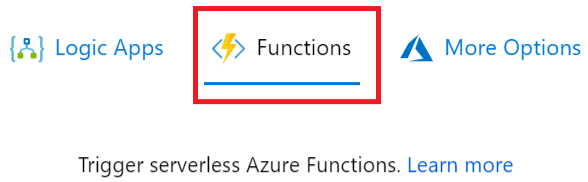


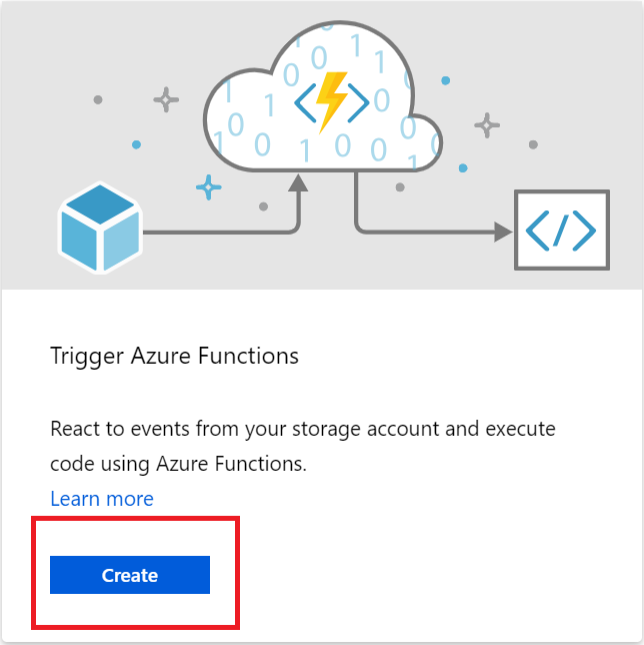
18. Click **Create**.
19. Go back to your Storage Account.
20. Click Events.
21. You should see an Event Subscription now.


Azure ThuRSdays
Every first thursday of the month
storagesjw - Events
Storage account
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Events
Storage Explorer (preview)
Settings
Access keys
CORS
Configuration
Encryption
Shared access signature
Firewalls and virtual networks
Advanced Threat Protection...
Static website (preview)
Properties
Event Subscription
Refresh
Get Started
Event Subscriptions
Show metrics:
General
Errors
For the last:
1 hour
6 hours
12 hours
1 day
7 days
30 days
100
90
80
70
60
50
40
30
20
10
0
1:45 PM
2 PM
2:15 PM
2:30 PM
Published Events (Sum)
default
0
Failed Events (Sum)
default
0
Unmatched Events (Sum)
default
0
Search to filter items...
NAME
ENDPOINT TYPE
PREFIX FILTER
SUFFIX FILTER
EVENT TYPES
LogicApp05a82132-814b-4bf4-9bfb-92b37...
WebHook
Microsoft.Storage.BlobCreated

Step 9 – Add a Function

1. In the storage account.
2. Click Events.
3. Click **Get Started**.
4. Select **Functions**.





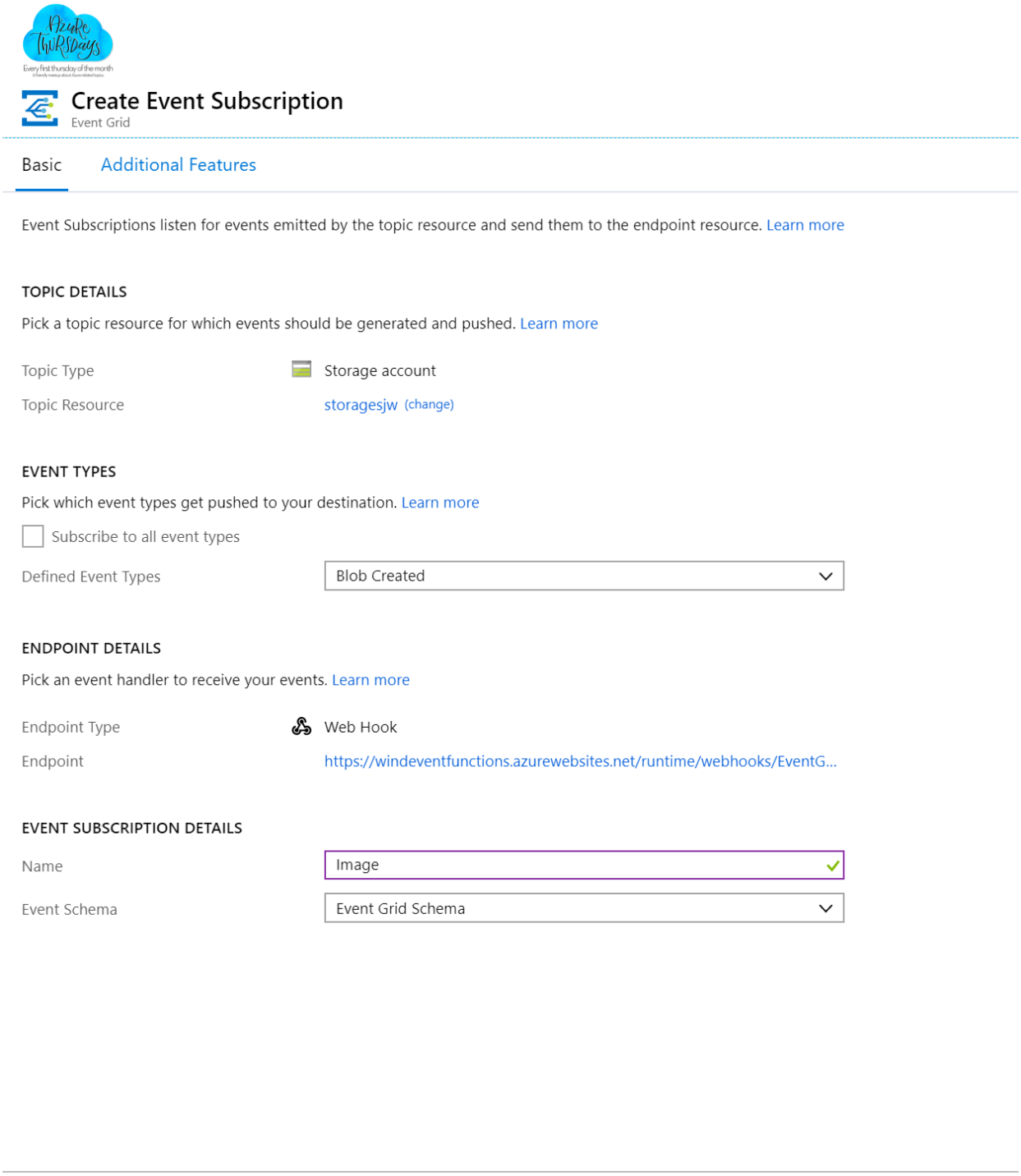
5. Click **Create**.
6. Choose an existing Function App or create a new one.
7. Click + on Functions.
8. Choose **Azure EventGrid Trigger** template.
9. Name the function and click **Create**.
10. Paste the following code:

```
#r "Microsoft.Azure.EventGrid"
#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;
using Microsoft.Azure.EventGrid.Models;

public static void Run(EventGridEvent eventGridEvent, ICollector<string>
outputSbMsg, ILogger log)
{
    log.LogInformation(eventGridEvent.Data.ToString());

    //intiliaze
    string imageInfo = string.Empty;

    //get content
    string jsonContent = eventGridEvent.Data.ToString();

    var objEvent = JsonConvert.DeserializeObject<Event>(jsonContent);

    //read image
    var webClient = new WebClient();
    byte[] image = webClient.DownloadData(objEvent.url);

    //analyze image
    imageInfo = AnalyzeImage(image);

    log.LogInformation(imageInfo);

    //output to service bus queue
    outputSbMsg.Add(imageInfo);
}

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;
    }
}

    public class StorageDiagnostics
    {
        public string batchId { get; set; }
    }

    public class Event
    {
        public string api { get; set; }
        public string clientRequestId { get; set; }
        public string requestId { get; set; }
        public string eTag { get; set; }
        public string contentType { get; set; }
        public int contentLength { get; set; }
        public string blobType { get; set; }
        public string url { get; set; }
        public string sequencer { get; set; }
        public StorageDiagnostics storageDiagnostics { get; set; }
    }
```

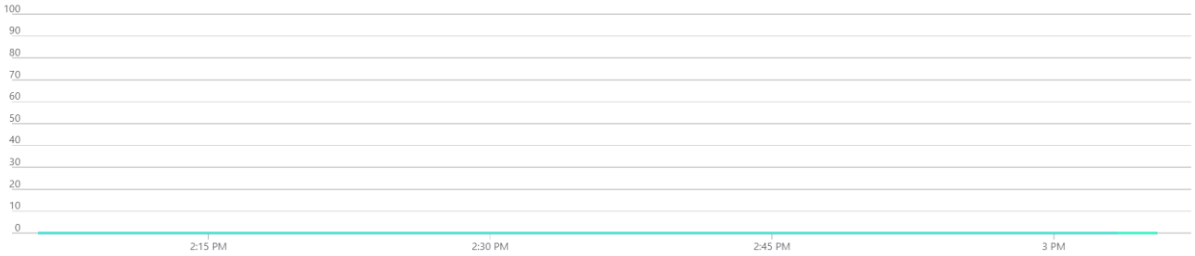
11. Change *"your key"* to the value of **Computer Vision API** key (step 6).
12. Go the integrate tab.
13. Add new output, and choose **Service Bus Queue**.
14. Choose Message Type: **Service Bus Queue**
15. Queue Name: **outbound** (see also step 6)
16. Click **Save**.
17. Go to your Function.
18. Click **Add Event Grid Subscription**.
19. Fill in the details like below:



20. Click **Create**.
21. Go to your storage account.
22. Click **Events**.
23. You should now have two subscriptions.

Show metrics: General | Errors

For the last: 1 hour | 6 hours | 12 hours | 1 day | 7 days | 30 days



Published Events (Sum)
default
0

Failed Events (Sum)
default
0

Unmatched Events (Sum)
default
0

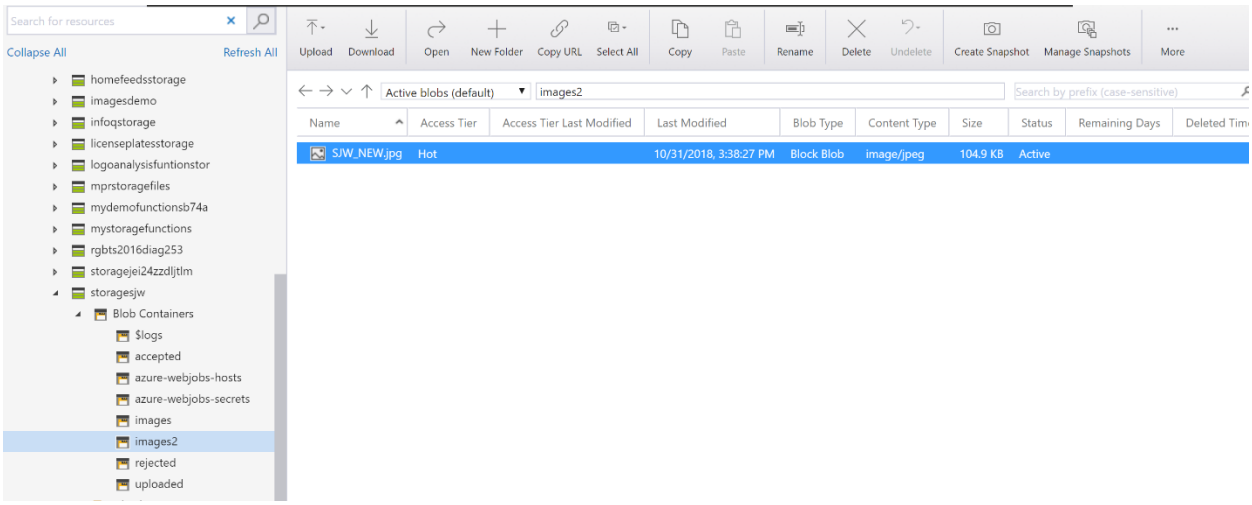
Search to filter items...

NAME	ENDPOINT TYPE	PREFIX FILTER	SUFFIX FILTER	EVENT TYPES
LogicApp05a82132-814b-4bf4-9bfb-92b37···	WebHook			Microsoft.Storage.BlobCreated
Image	WebHook			Microsoft.Storage.BlobCreated

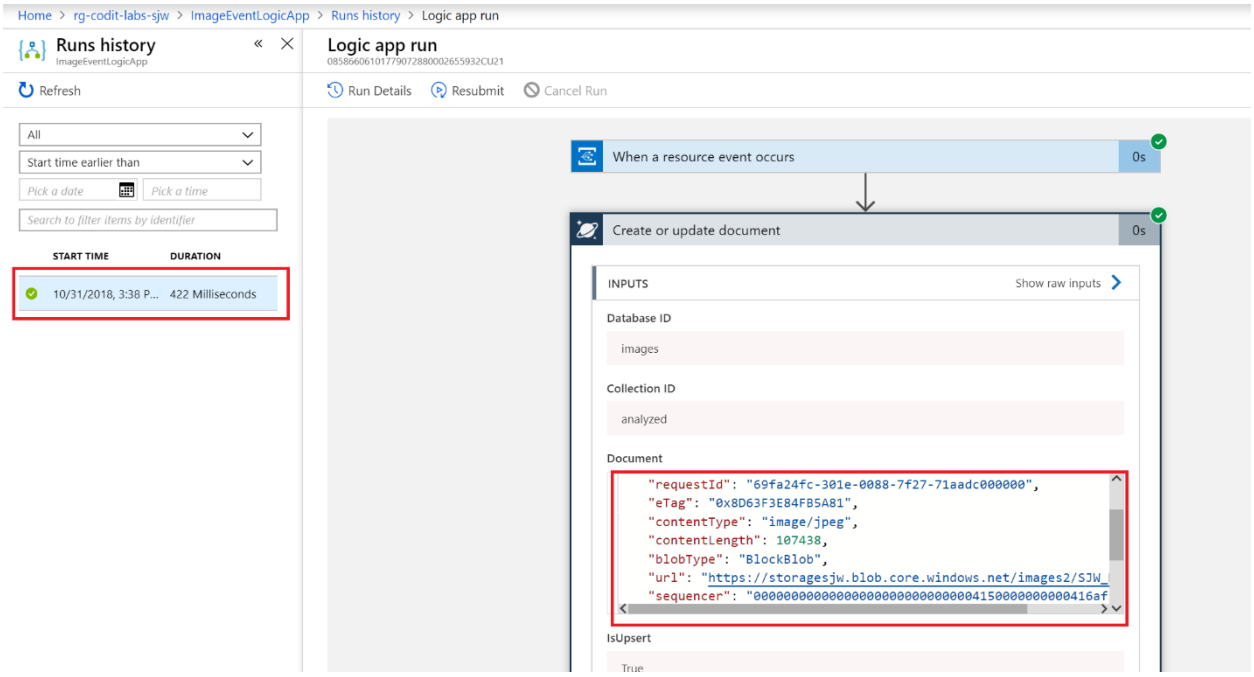
Step 10 – Test the solution

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 the .jpg picture on your machine that is a picture of yourself or someone else.

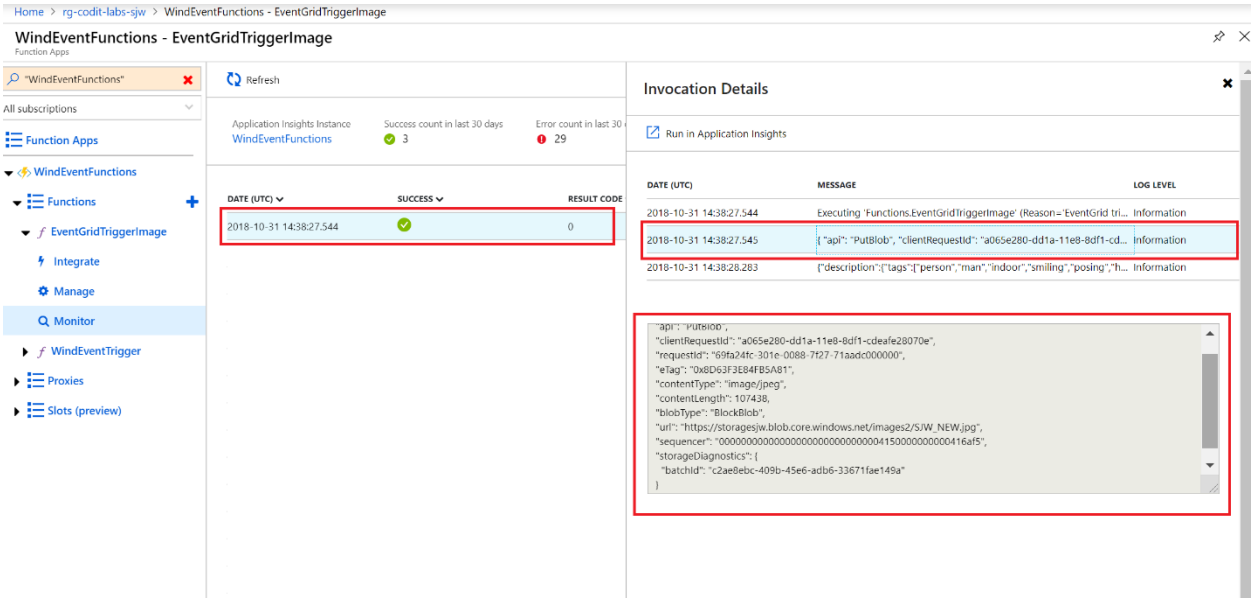


5. Go to your Logic App, and examine the run history and execution of the Logic App.



6. Go to your Function App and function.

7. Go to the monitor tab.



8. You can examine the Cosmos DB collection.
9. You can also examine the events in the storage account and see the graphs for the subscriptions.