Contents

Logic App

What is Logic App?

Azure Logic Apps is a cloud-based platform for creating and running automated *workflows* that integrate your apps, data, services, and systems. With this platform, you can quickly develop highly scalable integration solutions for your enterprise and business-to-business (B2B) scenarios. As a member of Azure Integration Services, Azure Logic Apps simplifies the way that you connect legacy, modern, and cutting-edge systems across cloud, on premises, and hybrid environments.

The following list describes just a few example tasks, business processes, and workloads that you can automate using the Azure Logic Apps service:

- Schedule and send email notifications using Office 365 when a specific event happens, for example, a new file is uploaded.
- Route and process customer orders across on-premises systems and cloud services.
- Move uploaded files from an SFTP or FTP server to Azure Storage.
- Monitor tweets, analyze the sentiment, and create alerts or tasks for items that need review.

For more information about the ways workflows can access and work with apps, data, services, and systems, review the following documentation:

- Connectors for Azure Logic Apps
- Managed connectors for Azure Logic Apps
- Built-in triggers and actions for Azure Logic Apps
- B2B enterprise integration solutions with Azure Logic Apps

Key Terms used in the hackathon :

Logic app

A *logic app* is the Azure resource you create when you want to develop a workflow.

Workflow

A *workflow* is a series of steps that defines a task or process. Each workflow starts with a single trigger, after which you must add one or more actions

Trigger

A *trigger* is always the first step in any workflow and specifies the condition for running any further steps in that workflow. For example, a trigger event might be getting an email in your inbox or detecting a new file in a storage account.

Action

A *trigger* is always the first step in any workflow and specifies the condition for running any further steps in that workflow. For example, a trigger event might be getting an email in your inbox or detecting a new file in a storage account.

Built-in operations

A *built-in* trigger or action is an operation that runs natively in Azure Logic Apps. For example, built-in operations provide ways for you to control your workflow's schedule or structure, run your own code, manage and manipulate data, send or receive requests to an endpoint, and complete other tasks in your workflow.

How Logic app works

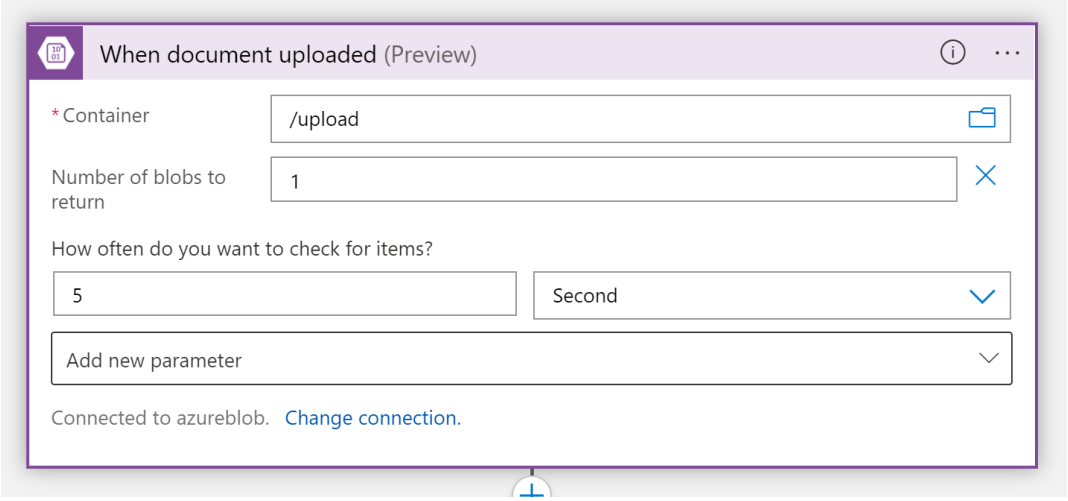
In a logic app, each workflow always starts with a single trigger. A trigger fires when a condition is met, for example, when a specific event happens or when data meets specific criteria. Many triggers include scheduling capabilities that control how often your workflow runs. Following the trigger, one or more actions run operations that, for example, process, handle, or convert data that travels through the workflow, or that advance the workflow to the next step.

Insurance claims automation Logic app

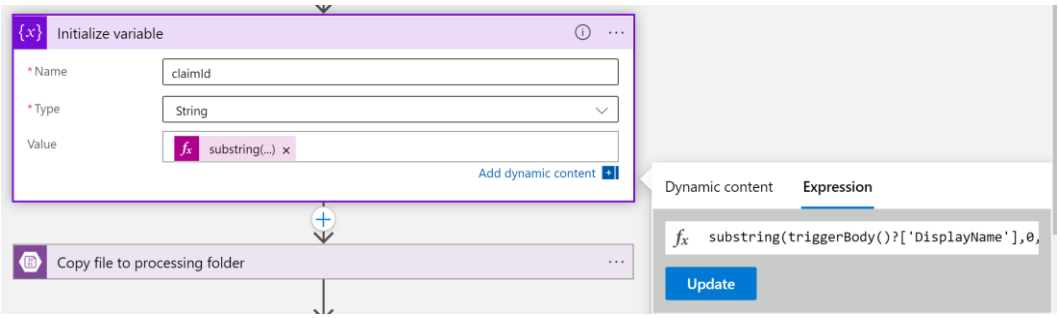
Logic app workflow to automate the insurance claims is already deployed in your resource group. Open the Logic app that is deployed and the designer should look like below:

Following is the high-level process flow for the workflow:

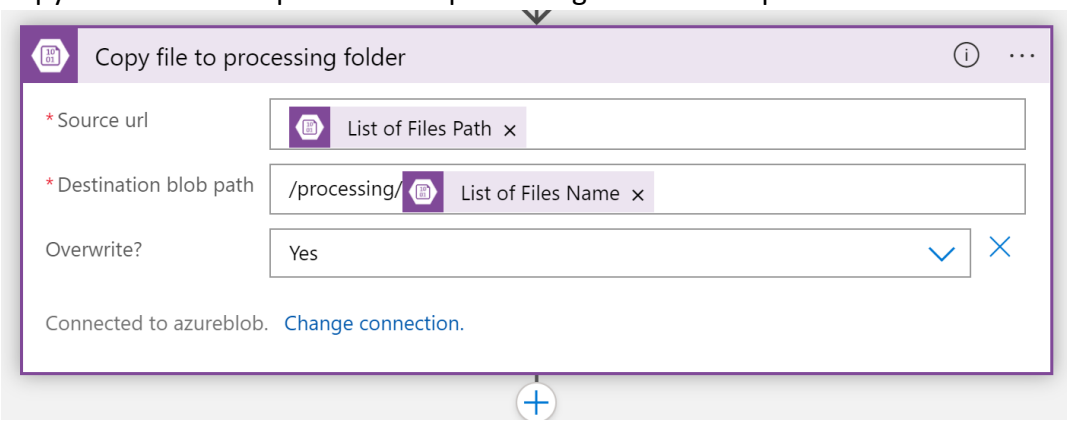
1. Trigger the logic app when a new document is uploaded into the storage account (/upload container)



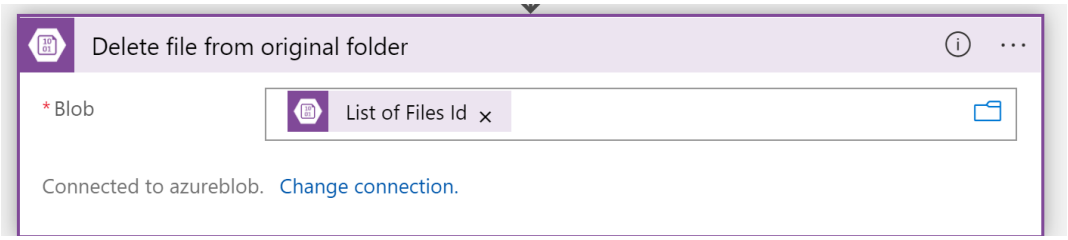
2. Initialize the ClaimId from the received blob image. Note we are using custom functions capability within logic app



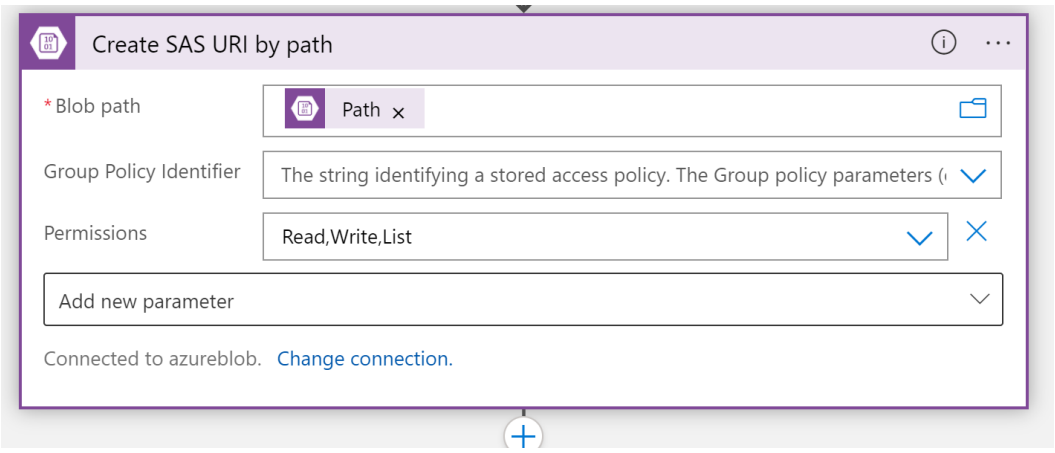
3. Copy the file that is uploaded into processing container to process the document



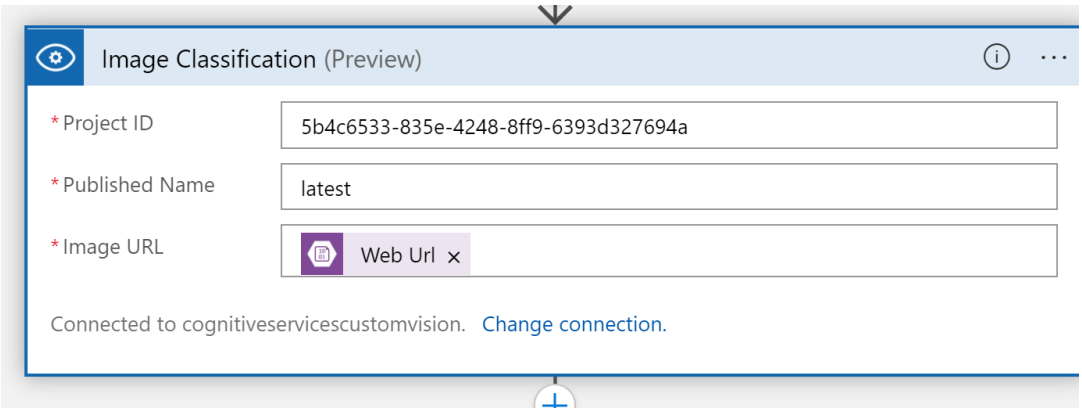
4. To avoid duplication, delete the original file from the /upload container



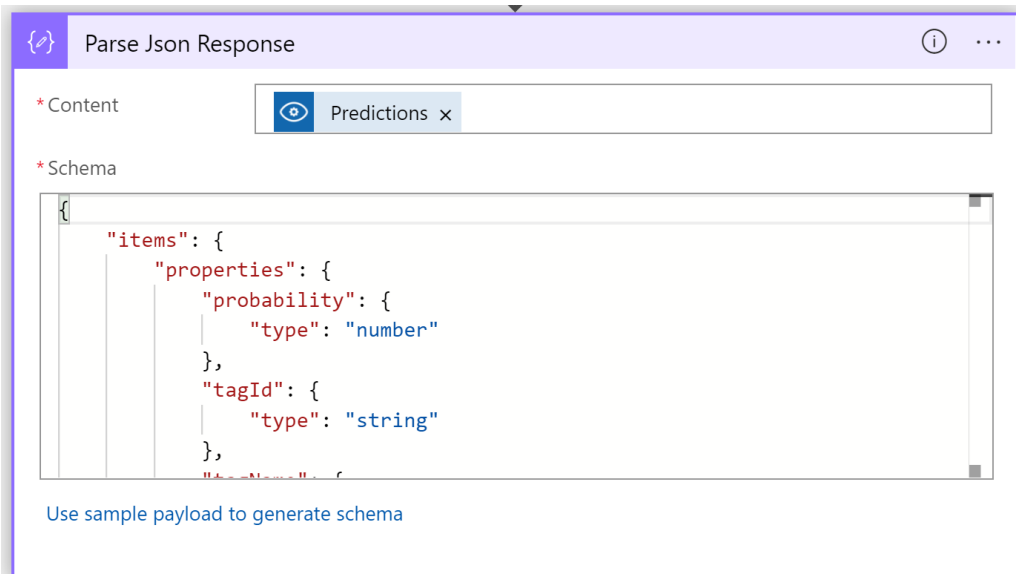
5. Dynamically create the SAS URI so that we can use that to call our custom vision model.



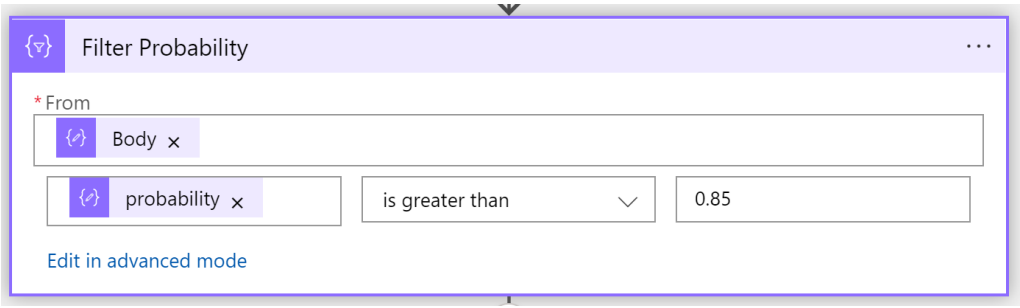
6. Use the inbuilt connector to call custom vision model. Note: Ensure you change the ProjectId to match to what you captured from "Custom Vision" section. Also do notice the published name is "latest" which is what we called our model when training custom model. Image URL is the SAS URI path that was created in Step 5.



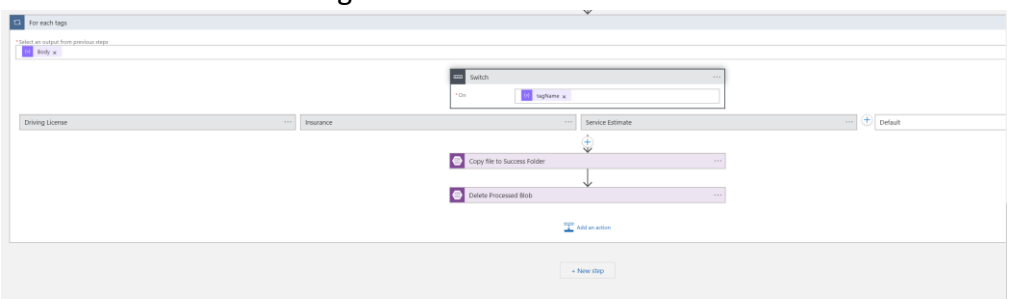
7. Because we are getting the JSON response from custom vision, use the capability within Logic app to parse the JSON (from object predictions)



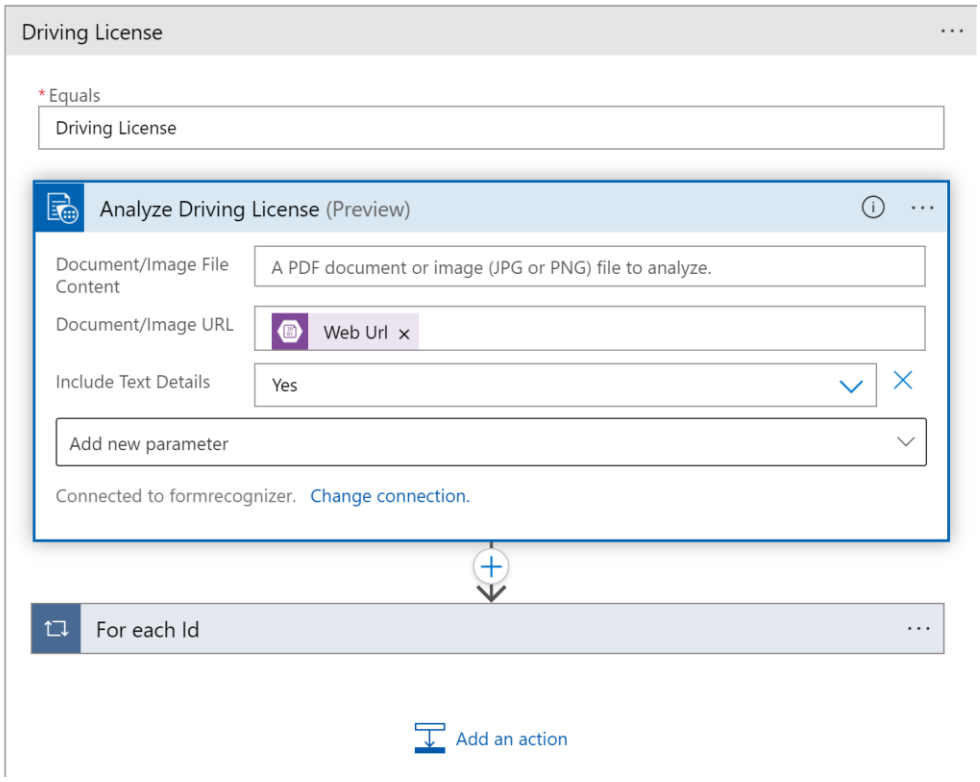
8. Let's filter the response so that the probability is > 0.85

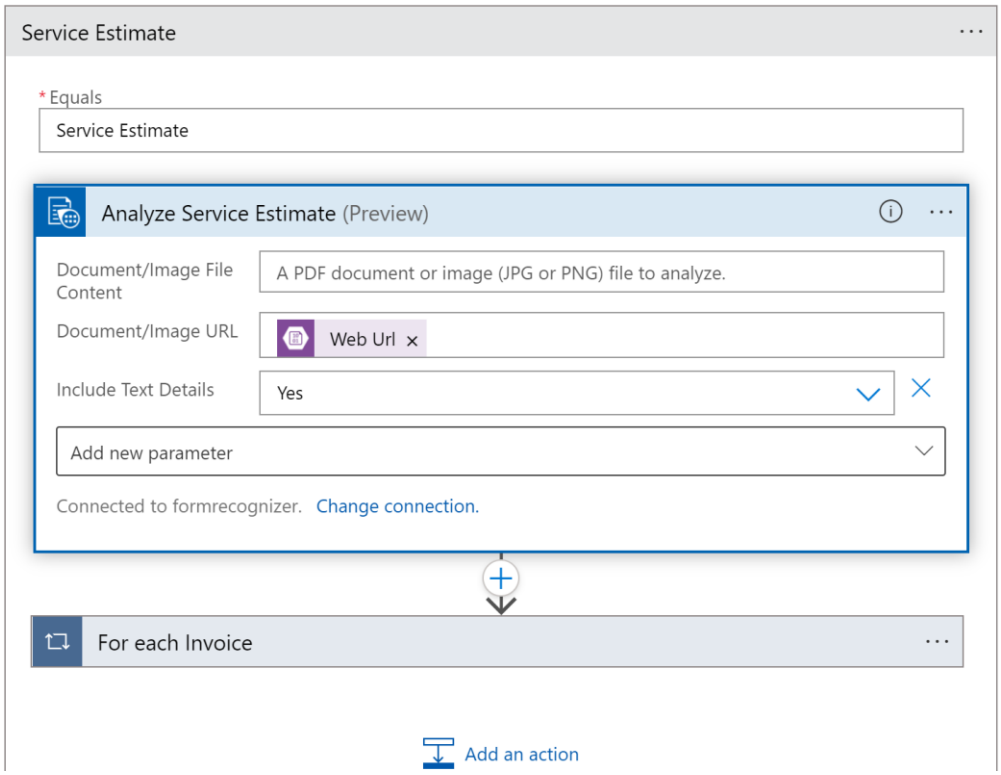


9. For each (ideally there should be only one with probability > 0.85) tag, execute the conditional statement. Conditional statement will (based on the document type) will call either the out of the box model (ID or invoice) or the custom model (insurance) that we created in "Form Recognizer" Section

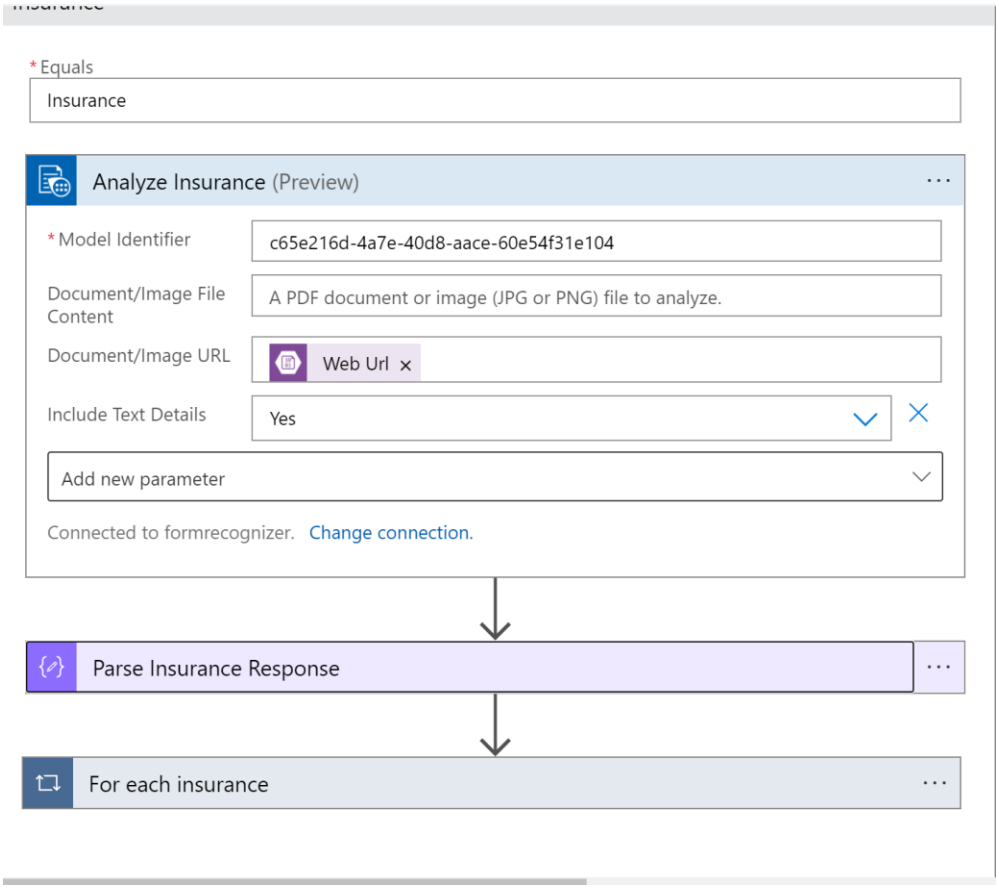


10. For out of the box model, we call Analyze method of the form recognizer model, passing on the URL

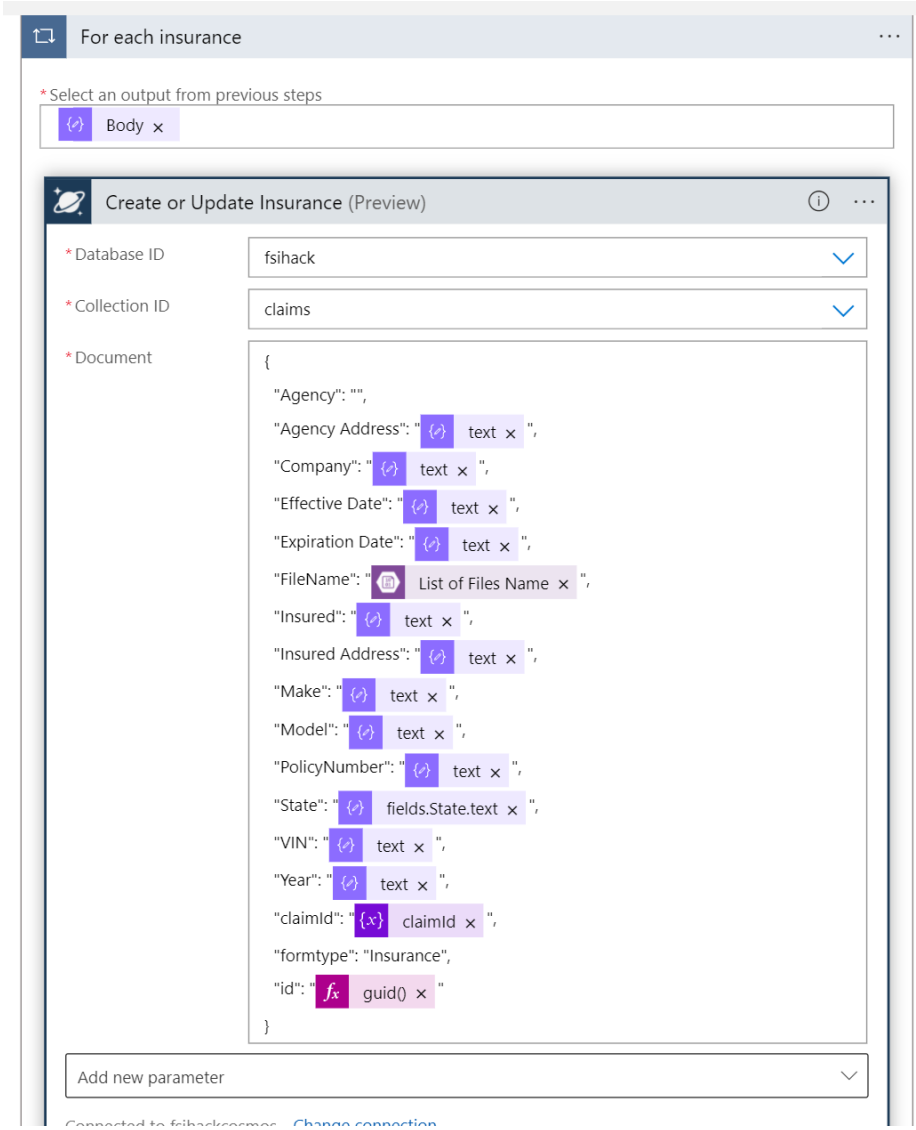




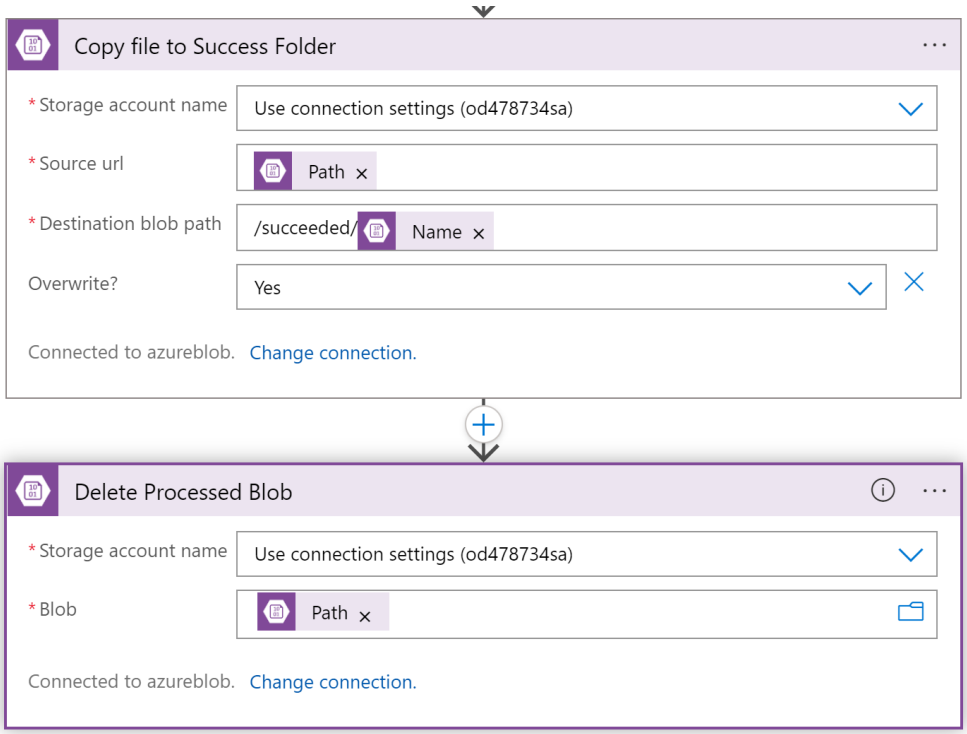
11. For custom model (insurance card), we are calling Analyze method on the custom model API. Since it's custom model, we need the "ModelId" that we captured in "Form Recognizer" Section. Copy the Form Recognizer modelId



12. Once the output from the form recognizer is parsed, we call CosmosDb connector to insert the data in CosmosDb collection



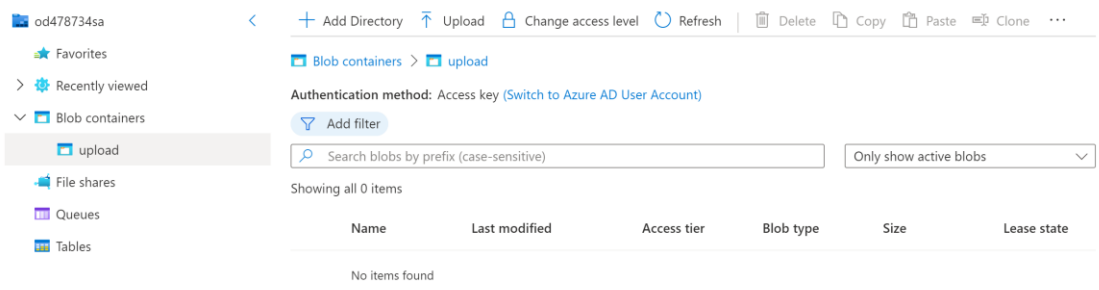
13. After successfully processing the workflow, copy the file to “Success” container and delete the file from "Processi”g" container


Copy file to Success Folder
* Storage account name
Use connection settings (od478734sa)
* Source url
Path
* Destination blob path
/succeeded/
Name
Overwrite?
Yes
Connected to azureblob.
Change connection.
Delete Processed Blob
* Storage account name
Use connection settings (od478734sa)
* Blob
Path
Connected to azureblob.
Change connection.

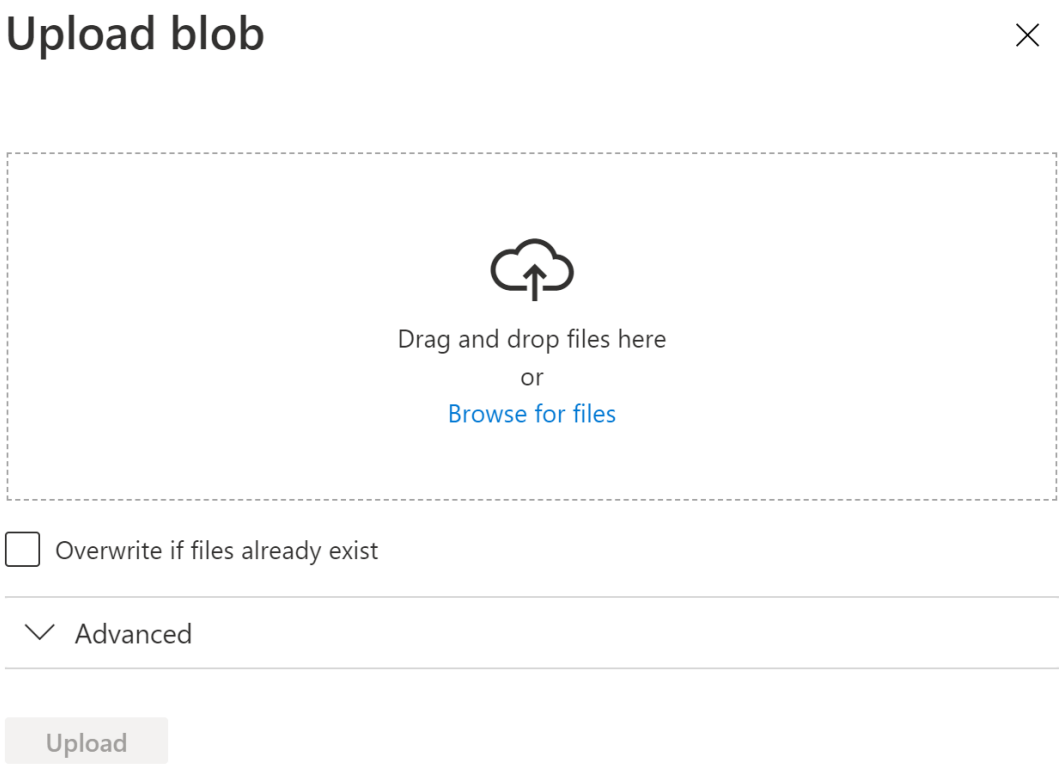
Test Logic app

Since our logic trigger is when the document is uploaded into storage account, we can test our logic app to invoke that trigger.

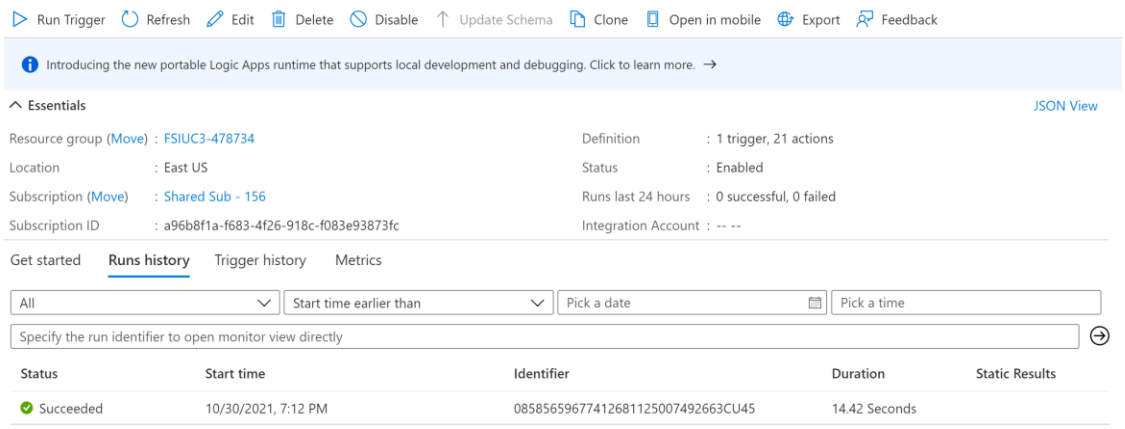
1. Go to Azure Portal -> Storage Account -> Storage Browser (Preview)
2. Click on Blob Container -> Upload



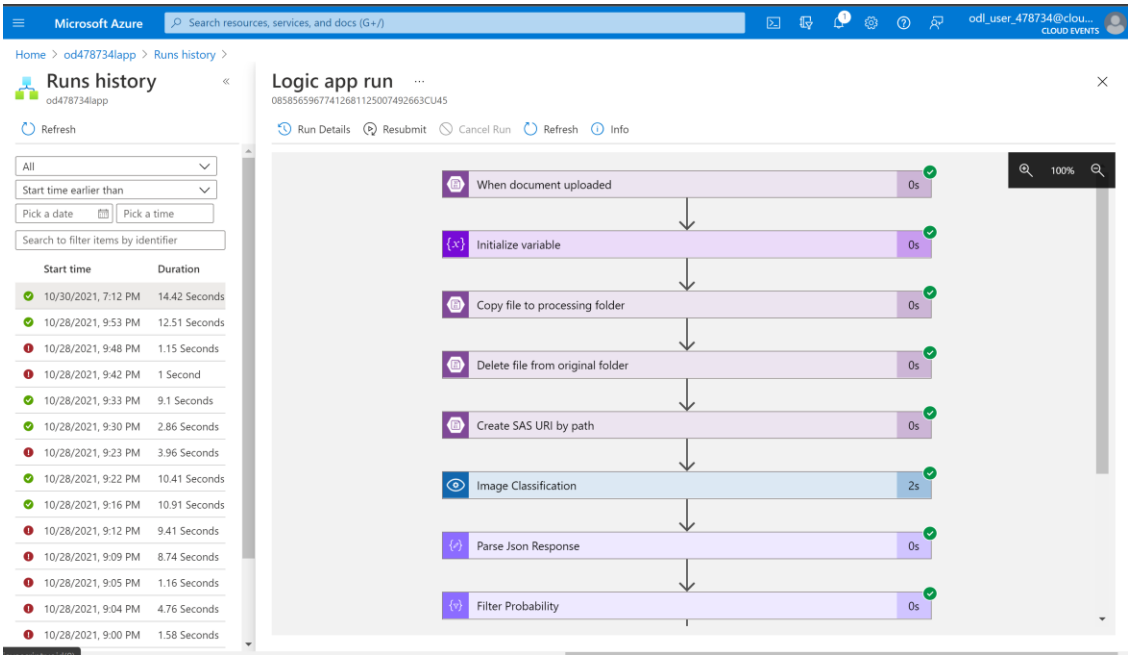
3. Click on Upload -> Browse for Files



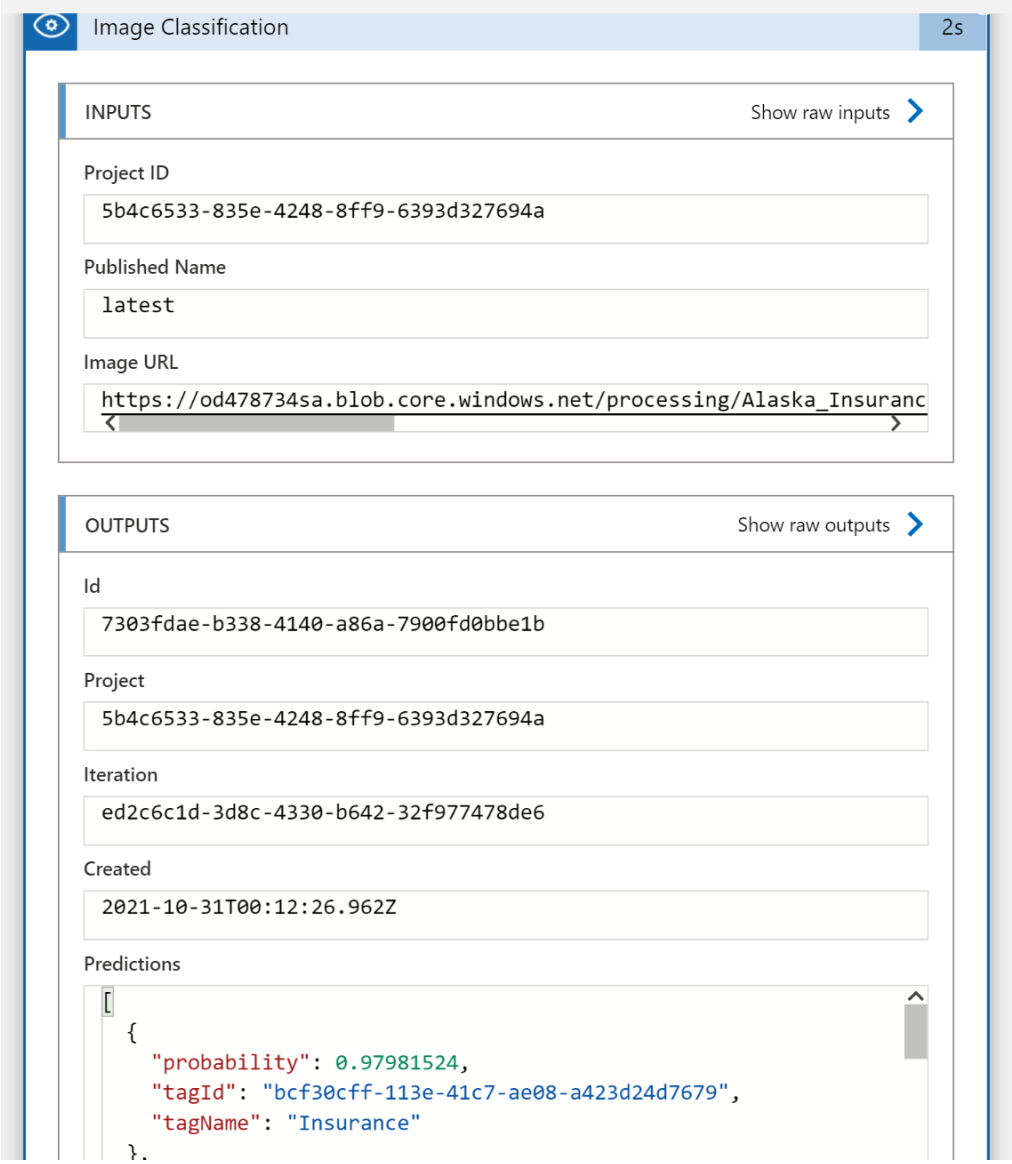
4. Select one of the document type (Insurance, Service Estimate, ID)
5. If you go back to Logic app and view the run history, you will see trigger executed the workflow



6. Clicking on the workflow, it will show you the run history for both triggers and all actions



7. Feel free to click on each action to view the “Input”, “Output” and additional details and understand the workflow execution and logic.



8. You can also go to cosmosdb collection to view the data that was inserted.