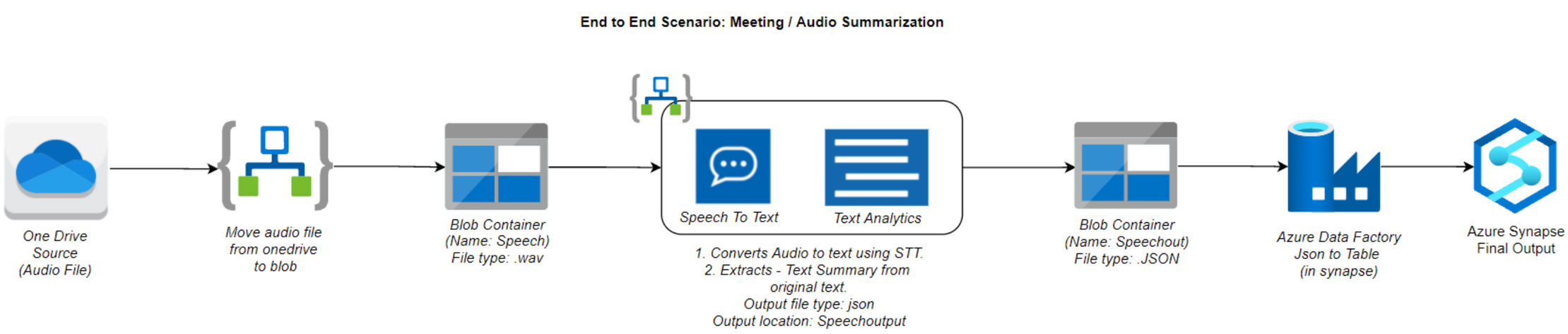
Document Name	HOL – Audio Summarization Use Case
Author	Shiva S Tomar & Anupreet Kaur
Reviewer	
Executive Summary	This workshop is an end-to-end simulation of processing an audio / video file and converting its audio part to summarized text, which then can be consumed via PowerBI or sent as an email. To accomplish this workshop, we will be using several Azure services like Cognitive Services (Speech and Text Analytics services), Logic Apps, Data Factory, Blob Storage and Synapse.
Purpose	The end-to-end scenario covered in this workshop will help you gain level 400 exposure on various Azure services, based on the knowledge you have gained by working on the previous workshops on individual cognitive services. Currently, Logic Apps does not provide out of the box modules to perform Speech to Text and Text Summarization capabilities (which is the most recent weapon in Text Analytics arsenal). Once you complete this lab, you will be the **Jedi Master** on the Azure services and should be able to ***D**emo, **D**evelop and **D**eploy* a POC or work in a production environment involving these Azure services. The important thing to note here is that you don't need to refer any other documents to complete this workshop.
Intent of Guide	This workshop is designed to help you integrate and automate your solutions using Azure services that lie in the Data & AI landscape. Also, we understand that Data folks come from a SQL or Python background and have limited exposure to application languages like .NET, Java etc. Hence, we have designed and developed this use case using no-code low-code and GUI driven services. It also covers the Concepts, How-to and best practices about the service.

Scenario brief: Audio Summarization

With audio content increasing at such a rapid pace, we seldom have enough time to go through the long videos, despite knowing they are significantly important and as a result we miss the knowledge in that content. Organizations today are saving audio and video content in different storage locations and at the same time there are some organizations which might not have the intent to do a long-term storage of such content, keeping the cost factor in mind. However, their employees might still need this content in the future.
For instance, you might have been in a situation when you missed an office meeting, or you attended and recorded one with the intent to view it back because it seemed important. However, you would rarely view the recording. At times, you might not have a rich collaboration solution like Microsoft Teams that allows you to systematically store and search the recordings or you might be leveraging 3rd party collaboration platforms which do not provide you the capability to transcribe your meeting. Despite all of this, most of us wish we had a summary of the entire meeting and instead of spending a few hours watching the complete thing, just skim though the key points & actions discussed. Keeping the content in audio format not only eats up your storage space but also limits the search capabilities on the content. Additionally, you lose out on all the advance search and analytics you could have performed had the content been in a text format.
The use case is built to help you automate the process to extract the audio transcription and perform text analytics on top of it, to extract artefacts like entities, summary, linked entities, sentiment from the audio content. This can then be published as reports via PowerBI or sent as an email. This use case can be implemented for any of these scenarios since it can be agnostic to the source platform.

Diagram: Functional Architecture



In this scenario, we will be leveraging the Teams meetings recording that are saved to OneDrive for Business and using Azure Cognitive Services, Logic Apps, Data Factory, Blob storage we will automate the process to save the extracted transcript and summary for each meeting to Azure Synapse Analytics.

To accomplish this, we will be building 2 Logic Apps workflow and 1 Data Factory pipeline, as follows :

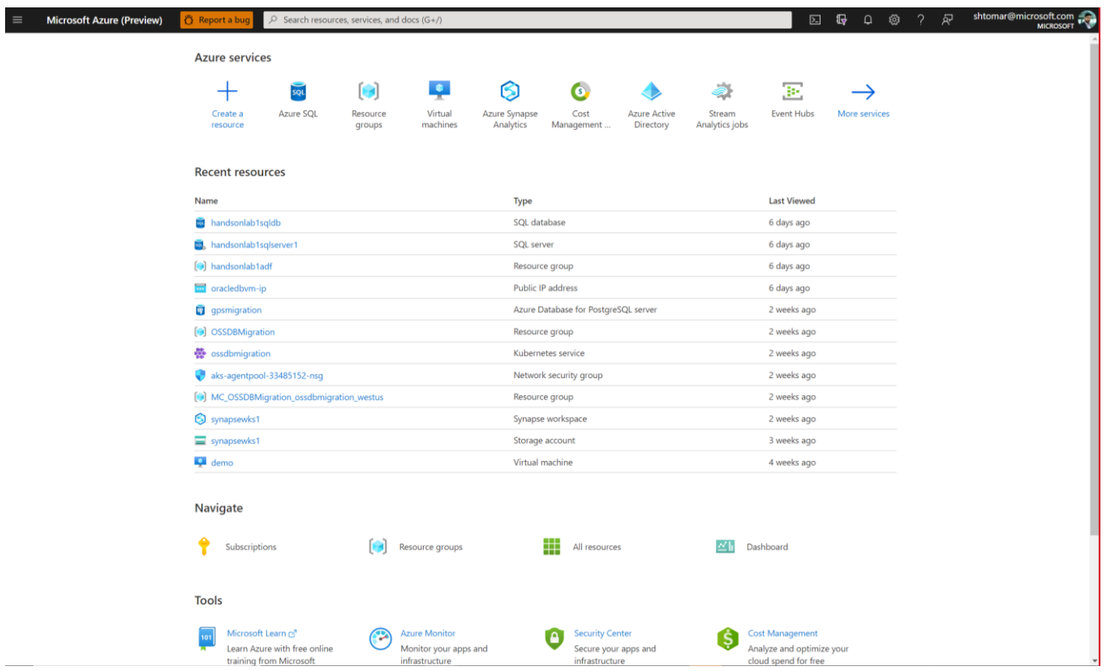
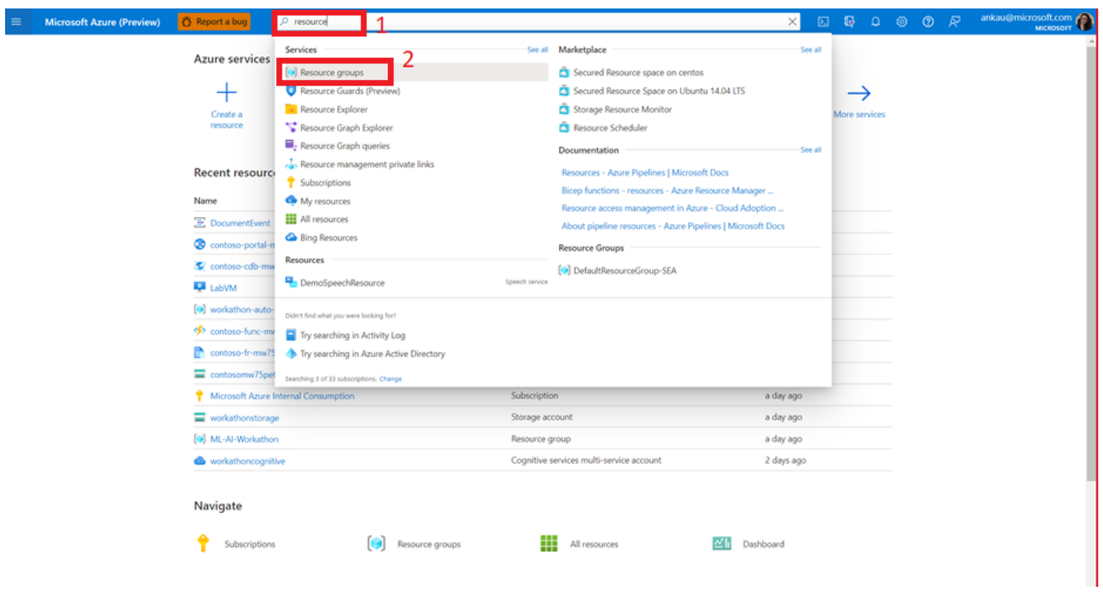
1. **Azure Logic Apps workflow 1 –** This workflow will extract the meeting recording from the source (OneDrive in this case) and copy it to a Blob Storage (speech container). Keeping the production scenario in mind, where in you might have the recordings stored at multiple locations. This workflow will help you streamline your sources and bring the recordings to a common storage location – Azure Blob Storage in our case. This flow is triggered as soon as a recording lands in the configured OneDrive folder. In case you have multiple sources, you will have to build similar additional workflows by just changing the source Trigger.
2. **Azure Logic Apps workflow 2 –** This workflow leverages 2 cognitive APIs (Speech to text and Text Analytics). This flow gets triggered when a media file lands in the speech container (via workflow 1). The speech to text API then gets triggered to generate the transcript for the incoming audio file and passes the transcript to Text Analytics API (v3.2-preview.1) to extract summary, linked entities, sentiment and entities from the original transcript text.
   Note : Both Speech to text and Text Analytics API (v3.2-preview.1) are not currently available as out of the box modules in Logic Apps, thus, we have used a series of HTTP modules to make the REST API calls to these Cognitive Services. This flow will create a consolidated JSON file containing the transcription and text analytics results and load it into Blob Storage (speechoutput container)
3. **Azure Data Factory pipeline –** This pipeline has 4 copy activities. 1 for each operation of Text Analytics like linked entities, summary, sentiment & entities to extract the key information from Text Analytics results saved in the JSON file in speechoutput container and saves it to the respective SQL table in a dedicated SQL Pool created in Azure Synapse Analytics. We will be using the Data Factory & pipeline functionality in Azure Synapse Analytics to create this pipeline.

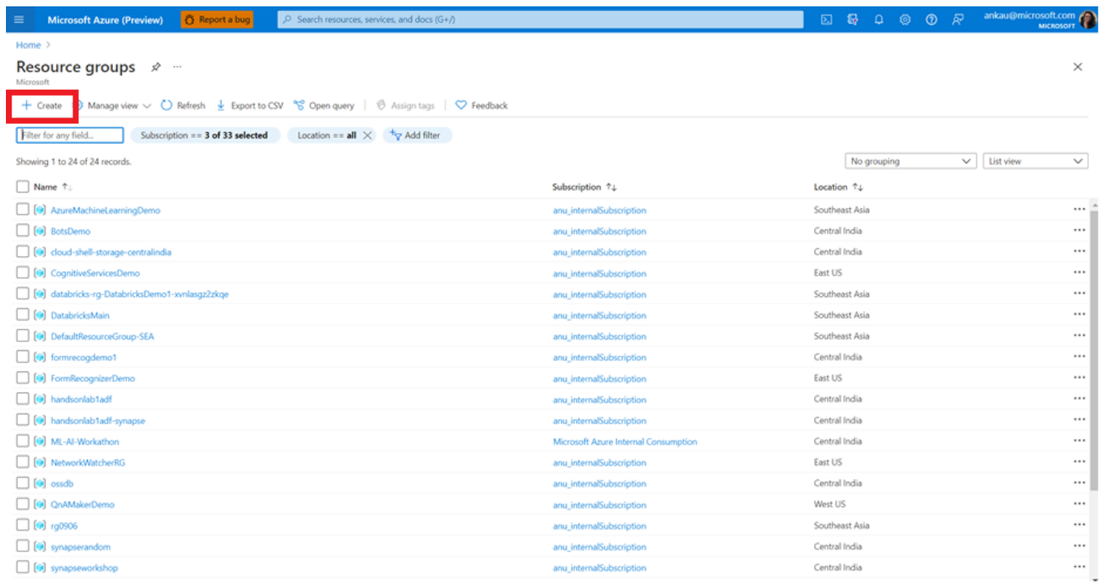
Step by step hands to become an Azure Jedi Master

Pre-requisites

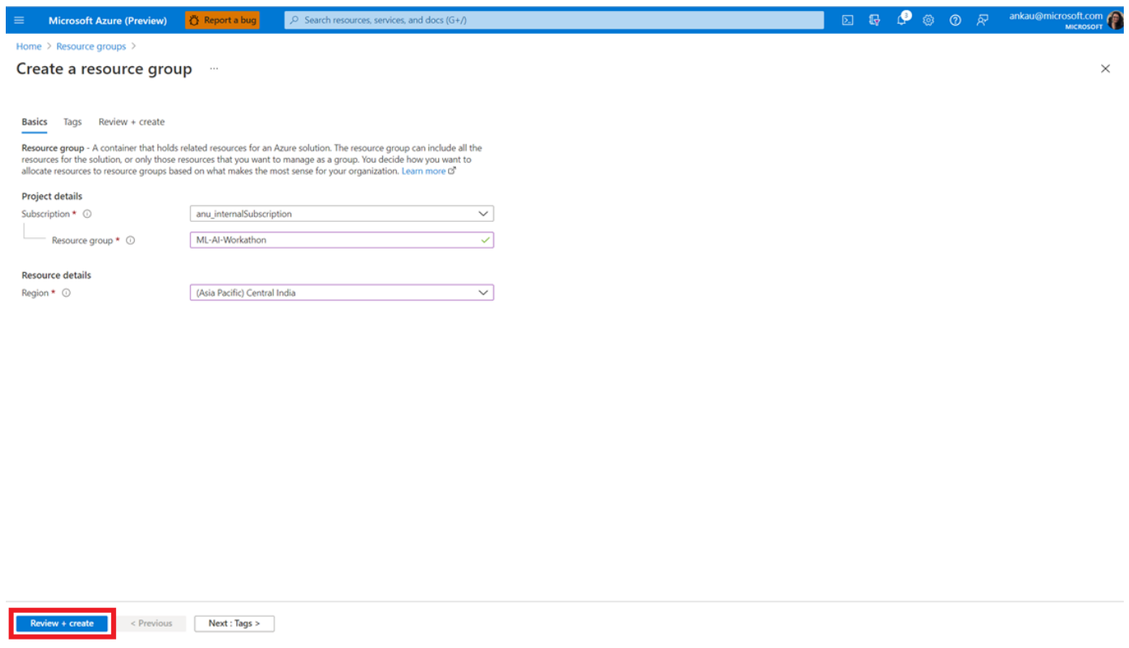
- An active Azure Account
  - You can use your current Azure Subscription or get started by creating a free trial account (https://azure.microsoft.com/en-in/free)
- Completed the workshops on Speech service & Text Analytics and understand the workflow for Speech-to-text batch transcription API and Text Analytics (analyze) API that make multiple async API calls to achieve the outcome.
- Use your own audio files (<50 MB in size; .wav, .mp3, .mp4 format)

Let's get started!

Screenshots	Steps & Significance
	Sign into your Azure Portal.
	Create a Resource Group Follow steps 1 & 2 to create a resource group. You can skip this step if you already have a Resource Group in place.



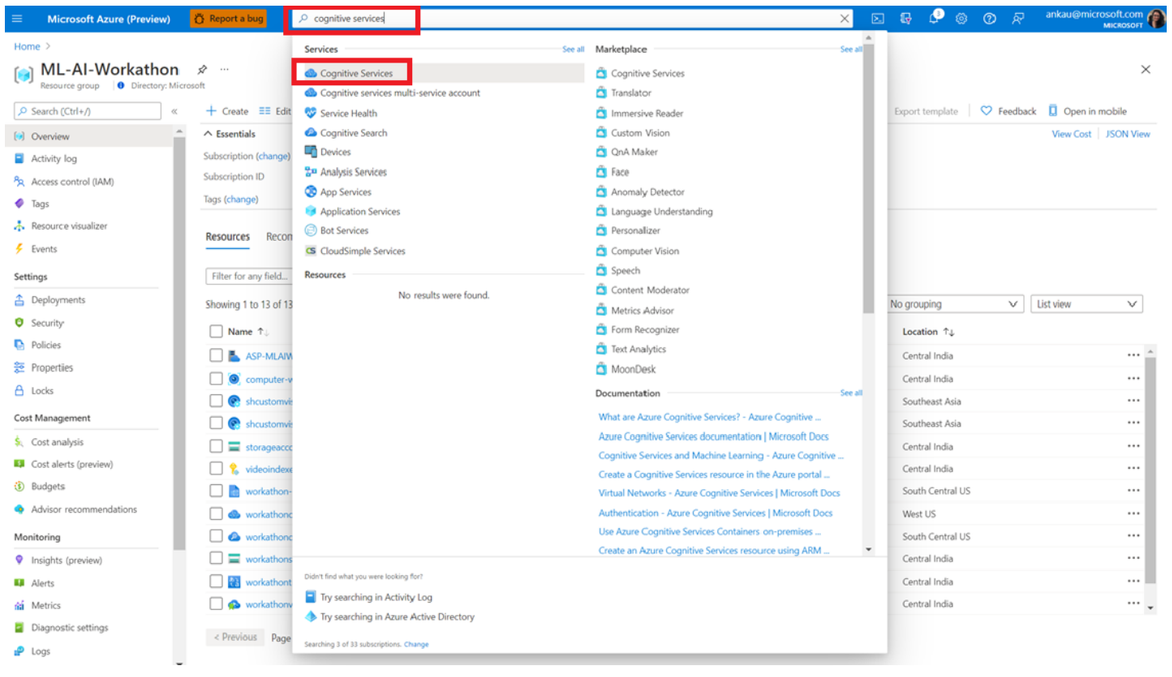
Click create to create a new resource group.



Enter the details –

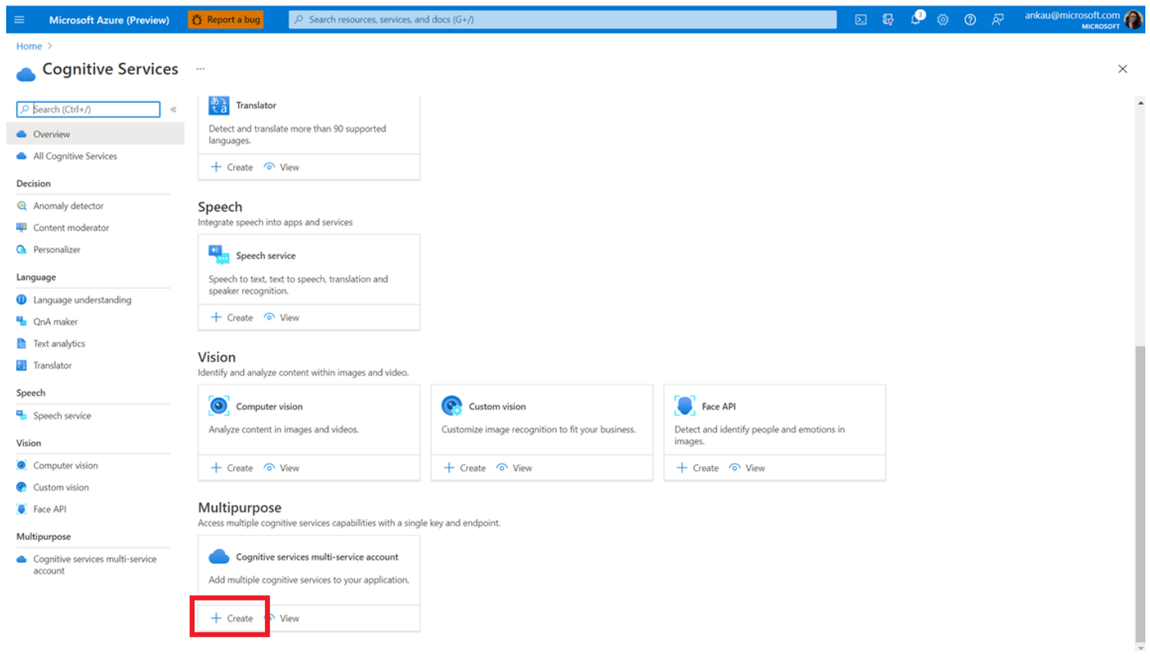
1. Subscription : Azure subscription in which you want to deploy the resource group
2. Resource Group : Name of your choice for the resource group
3. Region : Region where you want to deploy the resource group

Click Review + Create.



Once the resource group is created, search for Cognitive Services in the search bar above and select Cognitive Services.

You can skip this step if you already have a Cognitive Service in place for Text Analytics. This can be a multipurpose Azure Cognitive Resource or a Text Analytics Resource.

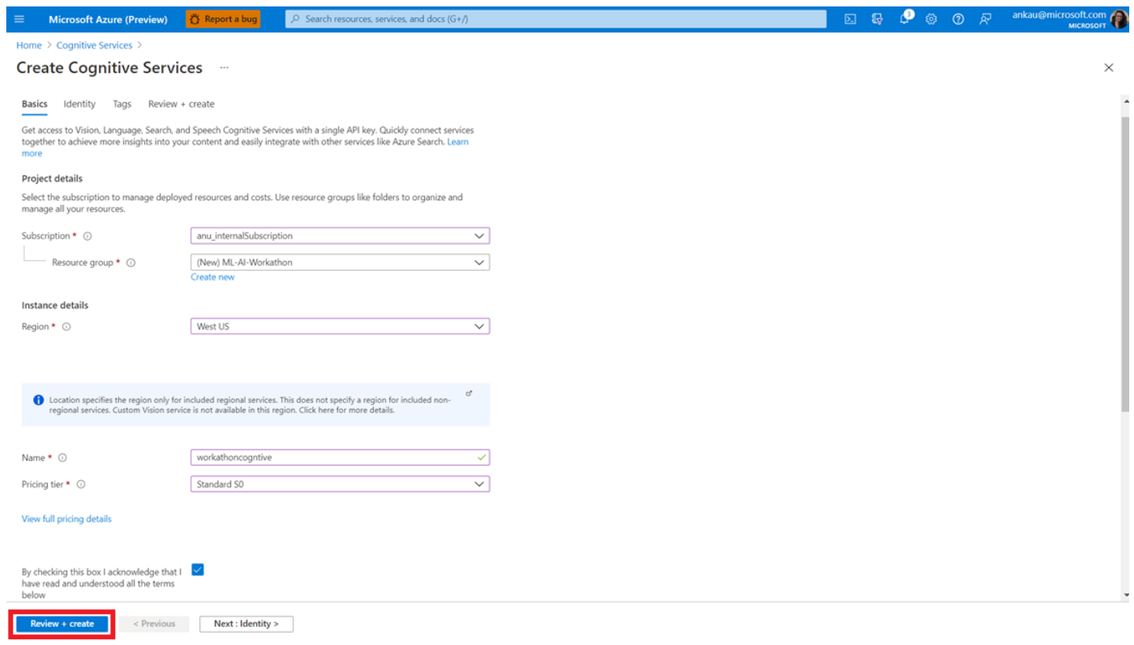


Create a multipurpose cognitive service

Significance : A multipurpose Cognitive Service account allows you to leverage the same resource for many cognitive services, which include :

Computer Vision - Analyze images
Content Moderator - Check text, image or videos for offensive or undesirable content
Face - Recognize people and their attributes in an image
Form Recognizer - Identify and extract text, key/value pairs and table data from form documents
Language Understanding - Extract meaning from natural language
Speech - Transform speech-to-text, text-to-speech and recognize speakers
Text Analytics - Detect sentiment, key phrases, entities and human language type in text

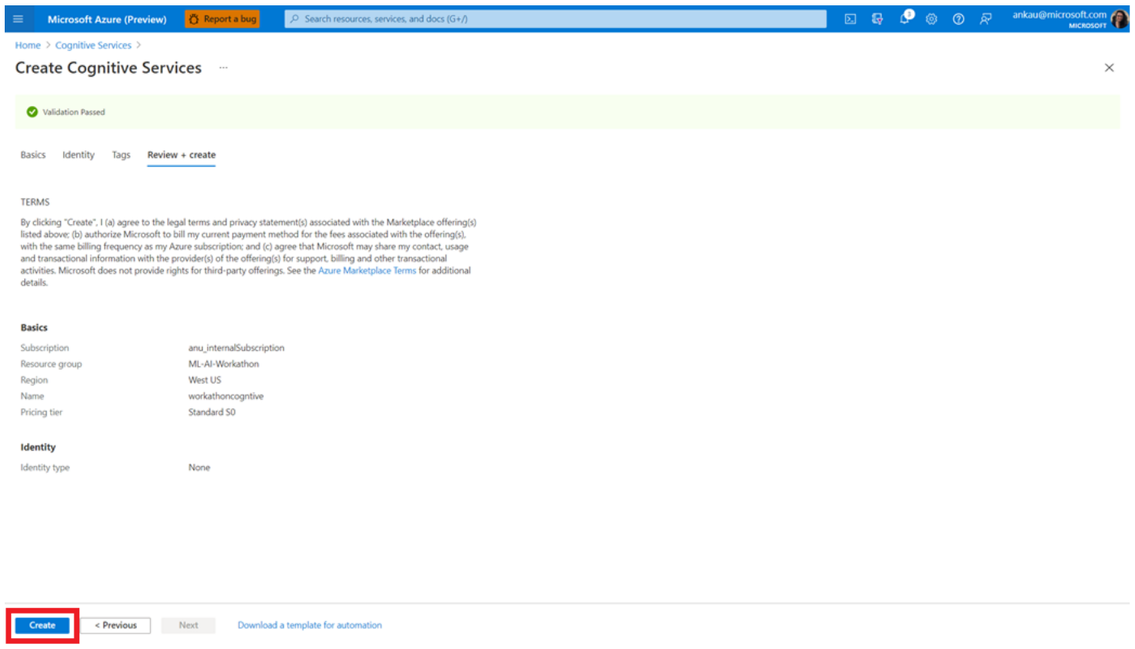
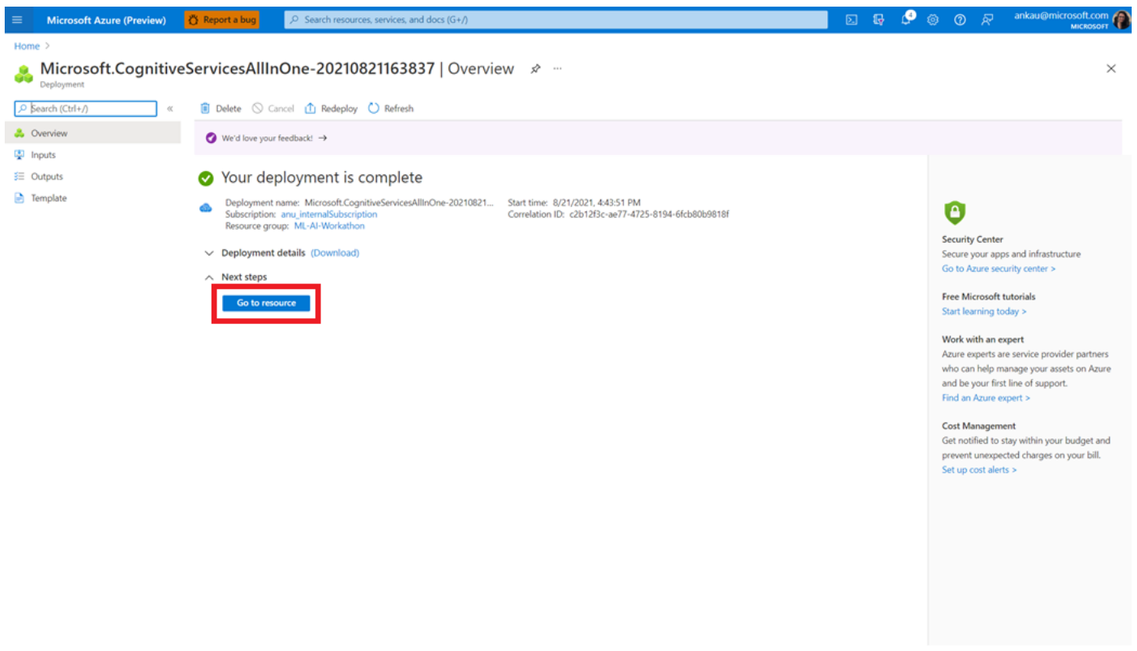
In this lab, we used a multipurpose Cognitive Service account since we would be learning about all the above-mentioned services.
However, you can also spin up individual services to execute these labs or for your development / production scenarios. The only difference is spinning up individual services allows logical separation from workspace standpoint and easy monitoring of billability.

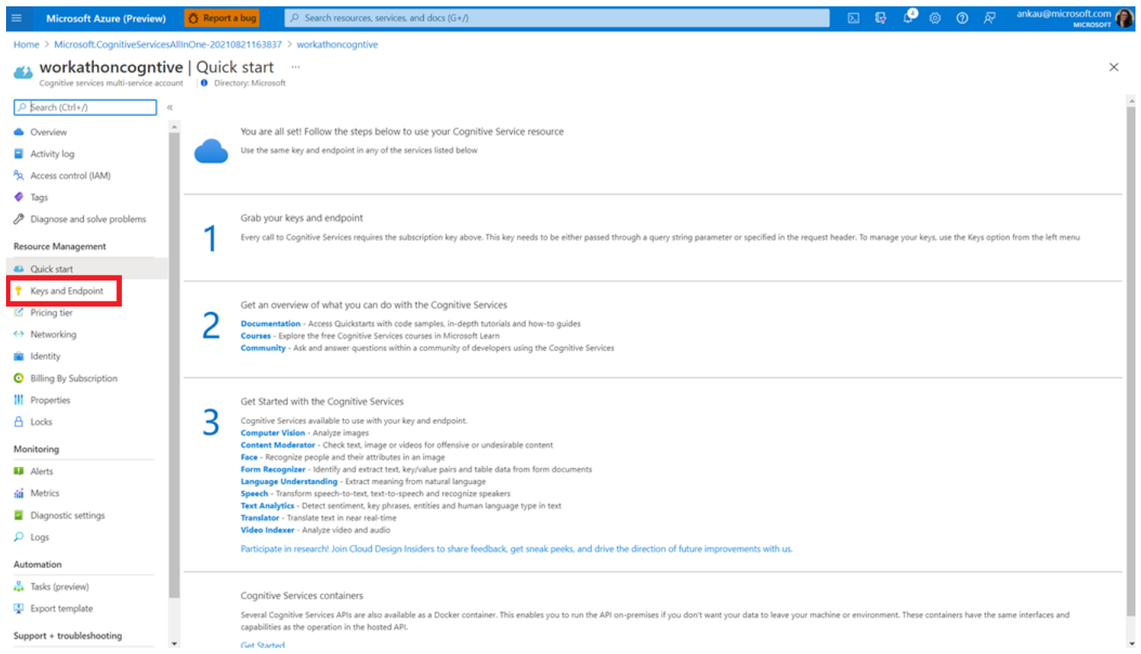
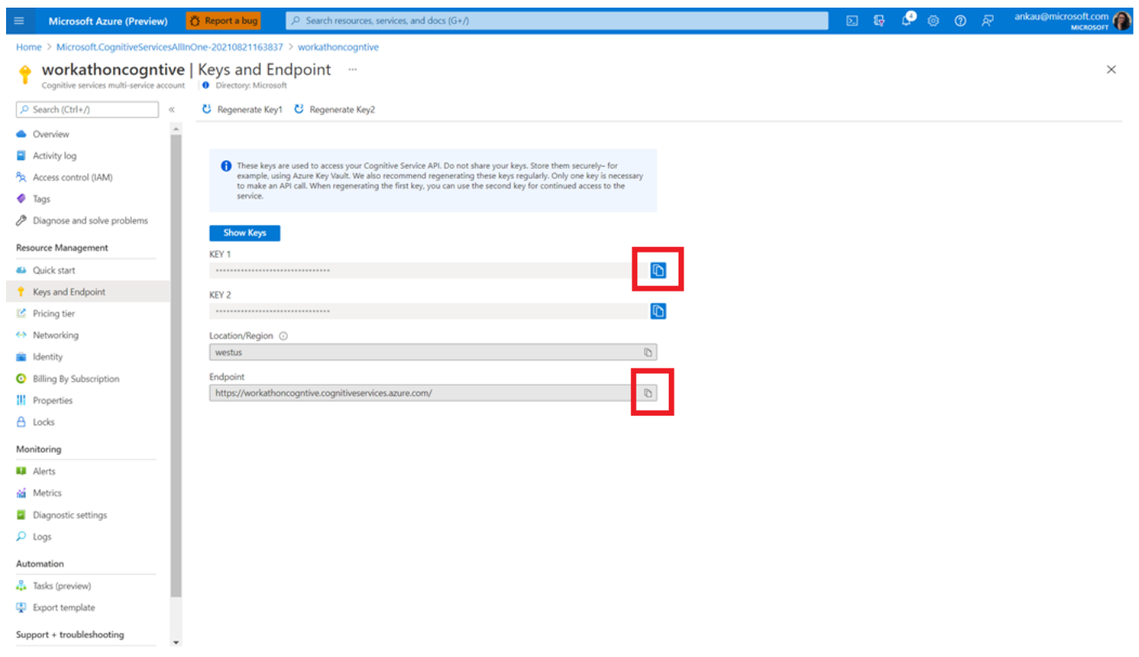


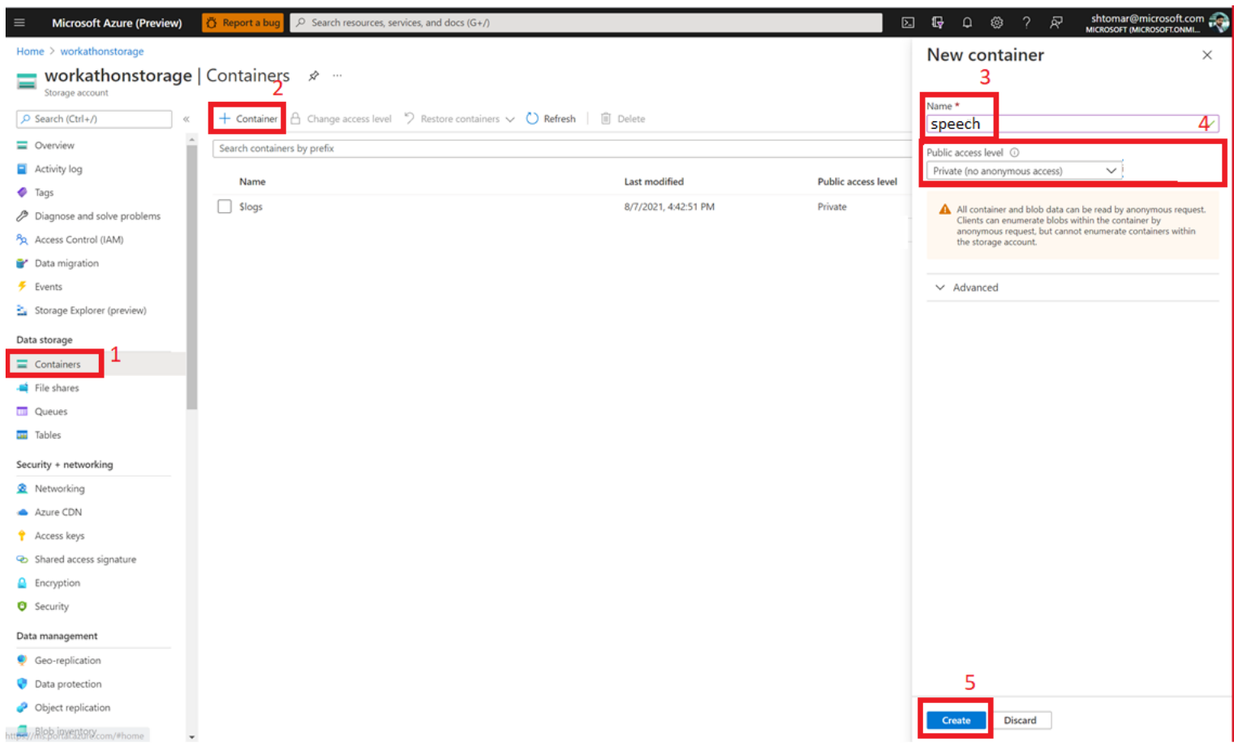
Enter the details to create a new cognitive service as follows -

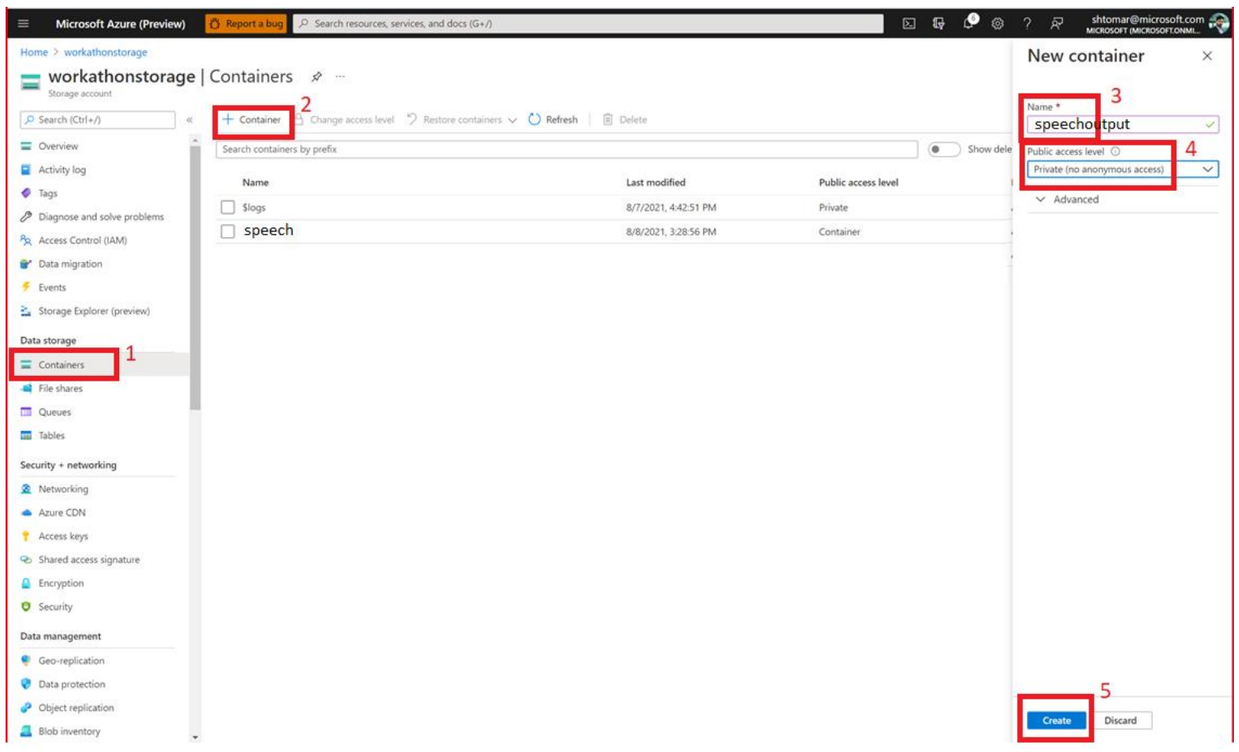
Project details	Description
Subscription	Select one of your available Azure subscriptions.
Resource group	The Azure resource group that will contain your Cognitive Services resource. You can create a new group or add it to a pre-existing group.
Region	The location of your cognitive service instance. Different locations may introduce latency but have no impact on the runtime availability of your resource.
Name	A descriptive name for your cognitive services resource.
Pricing tier	The cost of your Cognitive Services account depends on the options you choose and your usage.

Click Review + Create.

	Verify the details and click Create.
	After the resource has been deployed, click Go to Resource.

	<u>Copy keys & endpoints</u> On the Quick start page, you can find details about different cognitive services and can click the hyperlinks to learn more. Click Key and Endpoints.
	Copy the Key and Endpoint. Paste these in a notepad. You will leverage these in the next steps.





<u>Create blob containers</u>

In the Storage account, go to containers to create a blob container.
Follows the steps as numbered in the screenshot.

We will be creating 2 containers here :

1. <u>Container 1</u>

   Name : **speech** [You can choose a different name, however, we suggest you use the same name to avoid confusion since this is being referenced at a lot of steps going forward in this workshop. In case you still use a different name, make sure to make all the subsequent name changes]

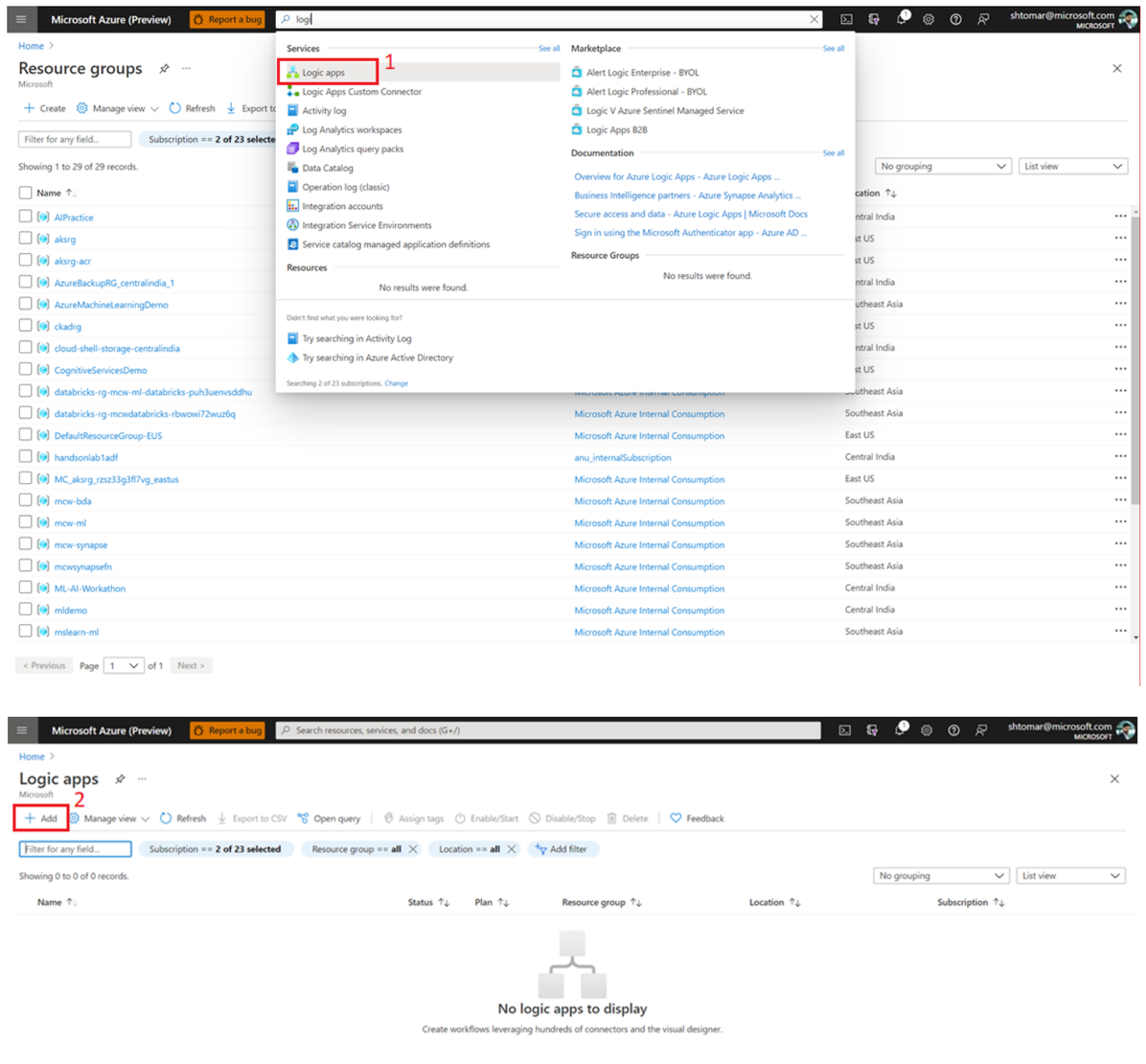
   Public access level : Private (no anonymous access)
   This container will be used to save the Speech output files from Logic Apps workflow 1, where we will be fetching the files from OneDrive and saving to blob container.

2. <u>Container 2</u>

   Name : **speechoutput** [You can choose a different name, however, we suggest you use the same name to avoid confusion since this is being referenced at a lot of steps going forward in this workshop. In case you still use a different name, make sure to make all the subsequent name changes]

   Public access level : Private (no anonymous access)
   This container will be used to save the JSON output files containing the complete meeting text analysis and summary from Logic Apps workflow 2, where we will be running STT to first extract the text from the input audio file and then run Text Analytics on top of it.

Also go to Access Keys tab and copy the Connection String. We will use this when we create a connection to this storage account from Logic Apps Workflow.



Create logic apps resource

Follow the steps as highlighted in the next few screenshots, to create a Logic Apps resource :

1. Search for Logic Apps in the search bar and select the resource

2. Click + Add on Logic Apps resource screen.

3. Select the corresponding subscription and resource group where you want to deploy the resource

Instance Details :

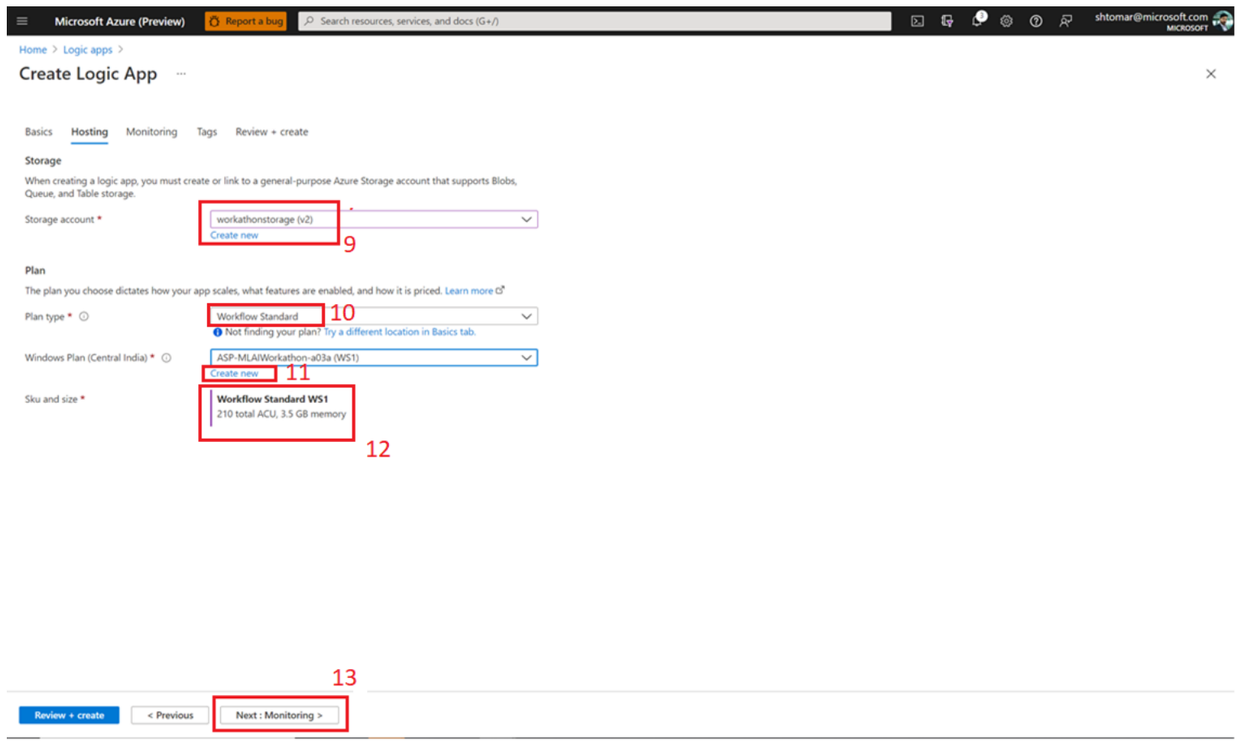
4. **Type** : Standard
   [We choose type as Standard since this allows us to build multiple workflows in the same resource, has more built-in connectors, allows more control and fine-tuning capability around runtime and performance settings, has integrated support for virtual networks and private endpoints etc.
   Consumption based Logic Apps, are fully managed easier to get started with but do not support the functionalities as supported by Standard type.
   In this workshop, we will be building multiple workflows, change limits & parameters and use advanced capabilities. Thus, we selected the type as Standard.]

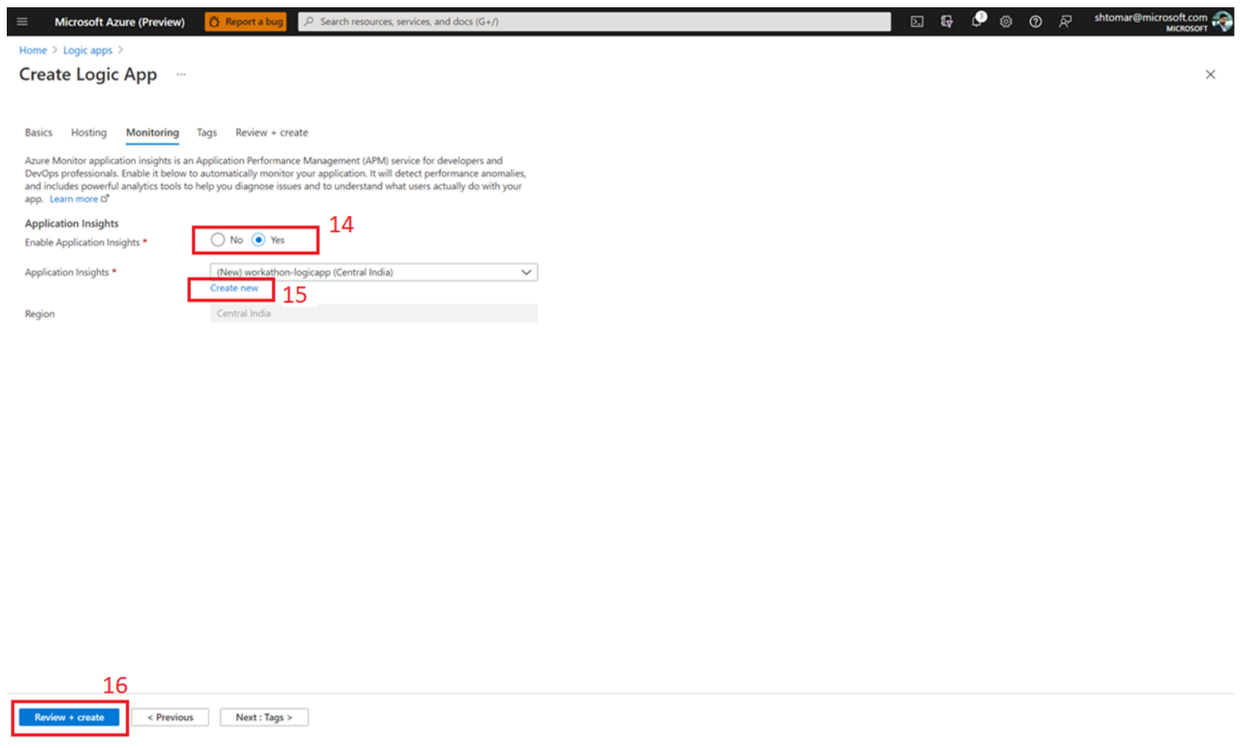
5. **Publish** : workflow
   [This defines the deployment destination for your logic app. Workflow is the default option for deployment to single-tenant Azure Logic Apps. You select Docker container in case you want to deploy your Logic Apps elsewhere, such as on Kubernetes.]

6. **Logic App name** : A unique name for your Logic Apps resource

7. **Region** : Central India [This is the region where your service will be deployed]

8. Click Next : Hosting





9. **Storage Account** : Select the Storage Account we created above
   [Logic App flows need a storage account to store artefacts and act as a staging area when the flows are run. Hence, we need to configure a Storage account]

10. **Plan type** : Workflow Standard
    [This defines the hosting plan to use for deploying your logic app]

11. Create a new plan in Windows Plan option
    [This defines the plan name to use. Either select an existing plan name or provide a name for a new plan.]

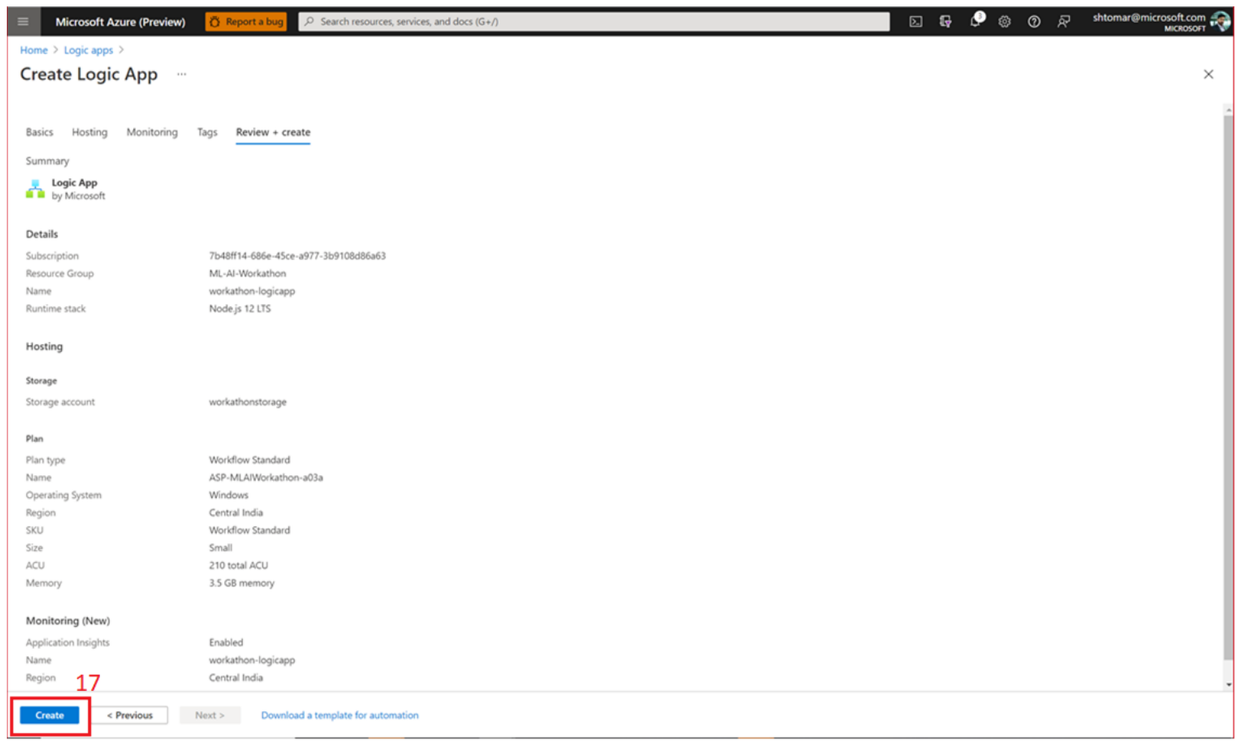
12. **SKU and Size** : Leave at default
    [The pricing tier to use for your logic app. Your selection affects the pricing, compute, memory, and storage that your logic app and workflows use.
    To change the default pricing tier, select Change size. You can then select other pricing tiers, based on the workload that you need.]

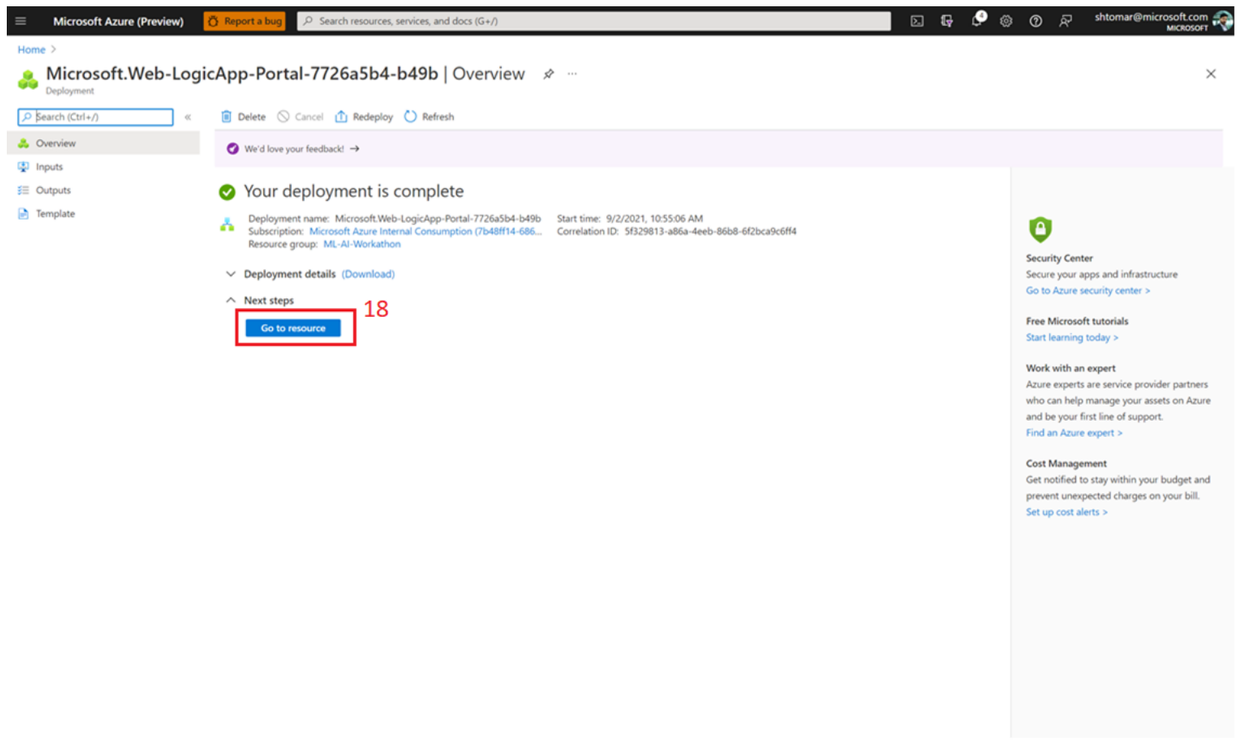
13. Click next : Monitoring

14. On the Monitoring tab, you can choose you enable or disable Application Insights

15. In case you enable Application Insights, select from an already existing Application Insights resource or create a new one.

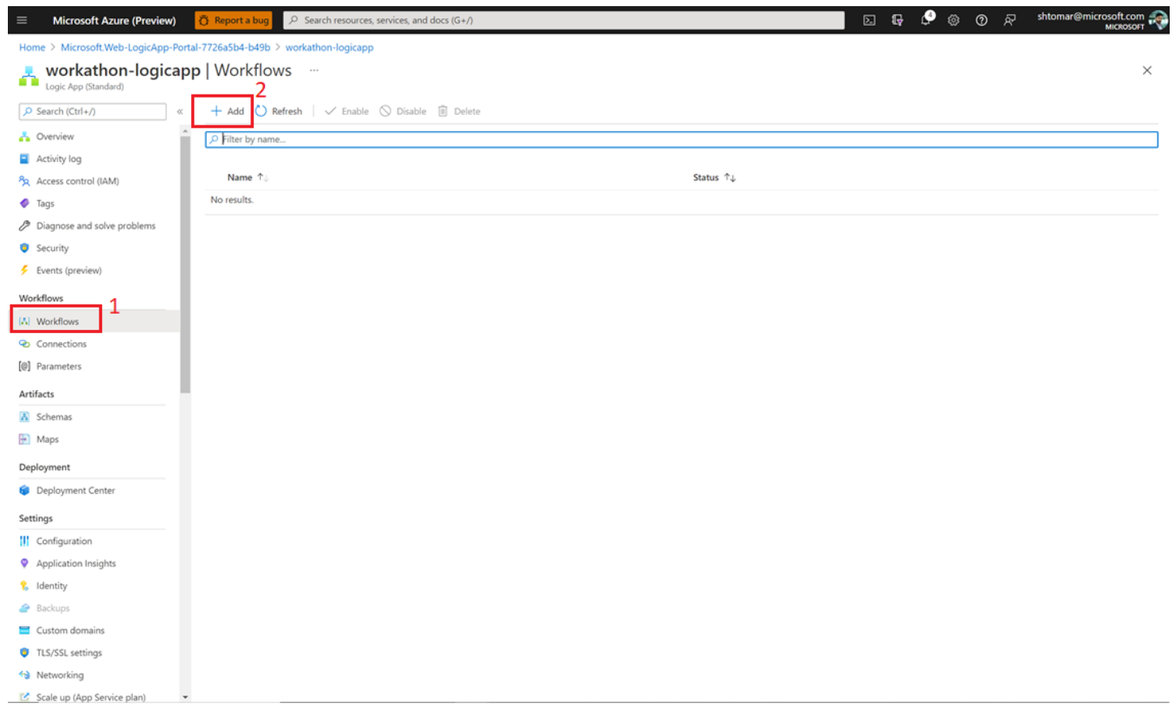
16. Click Review + Create

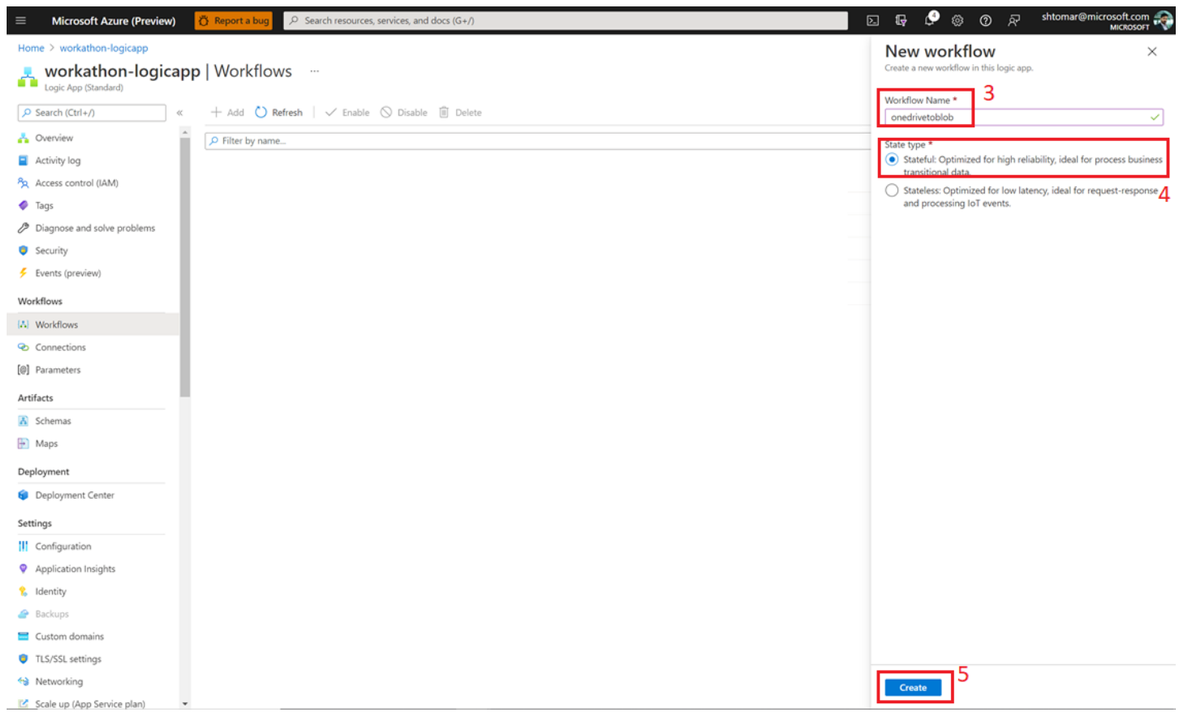


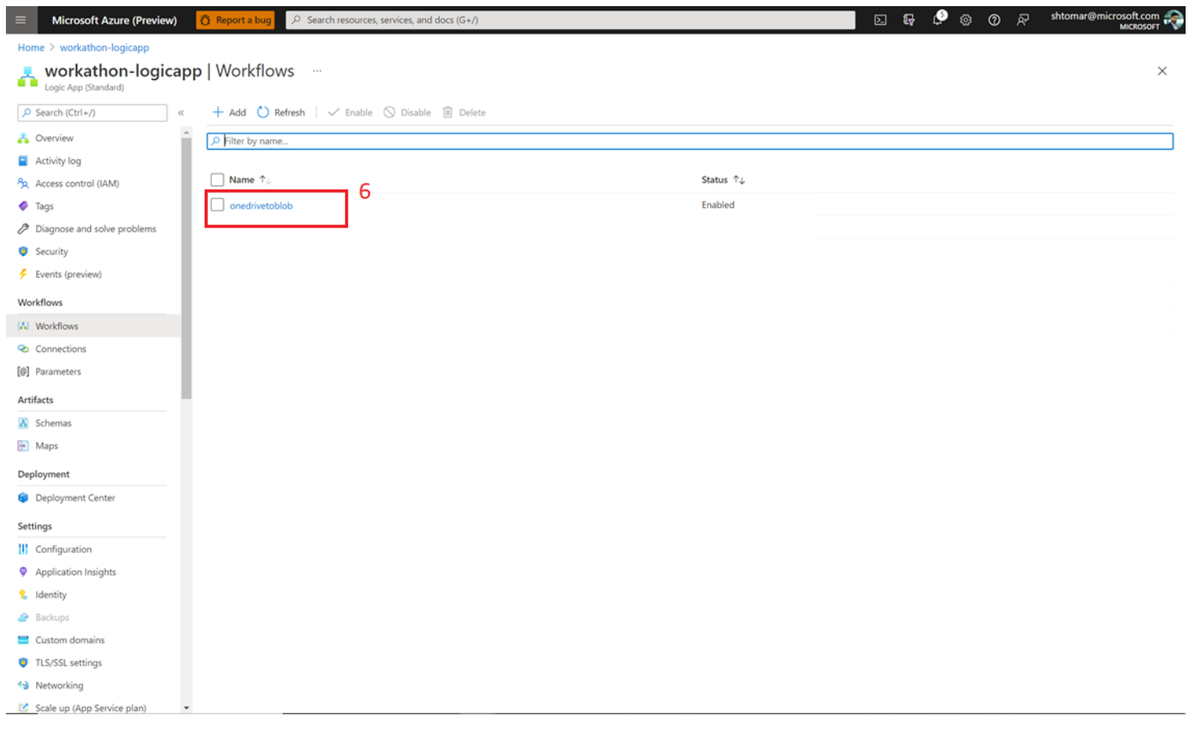


17. Go through the details and click Create

18. Once deployed, click Go to resource





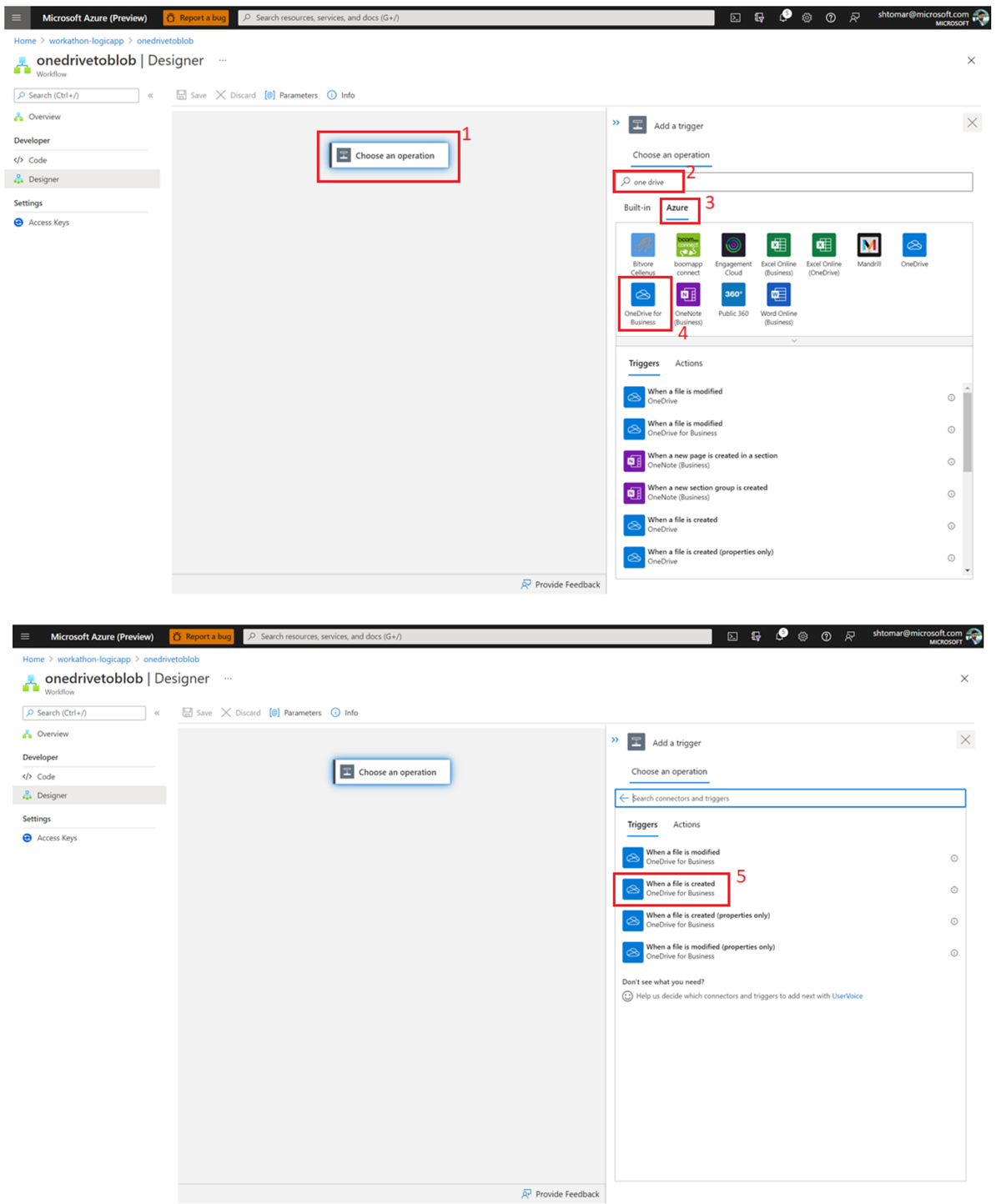


Create Logic App Workflow 1 : Onedrivetoblob

In Workflow 1, we will use source as OneDrive for Business (Recordings folder) since this is the default location for Teams meeting recording and we will use blob storage as destination to store the contents at one place.
In case you have recordings coming in from multiple sources, you can build additional Workflows like workflow 1 and use blob storage as a central repository.

To create a new Logic App workflow, follow the steps as highlighted :

1. Select the Workflows tab
2. Click + Add
3. Provide the Workflow name (onedrivetoblob)
4. Select State Type as Stateful
   [Create a stateful workflow when you need to keep, review, or reference data from previous events. These workflows save and transfer all the inputs and outputs for each action and their states to external storage, which makes reviewing the run details and history possible after each run finishes.]
5. Click create
6. Once created, click on the Workflow name to start building the flow



Adding a trigger

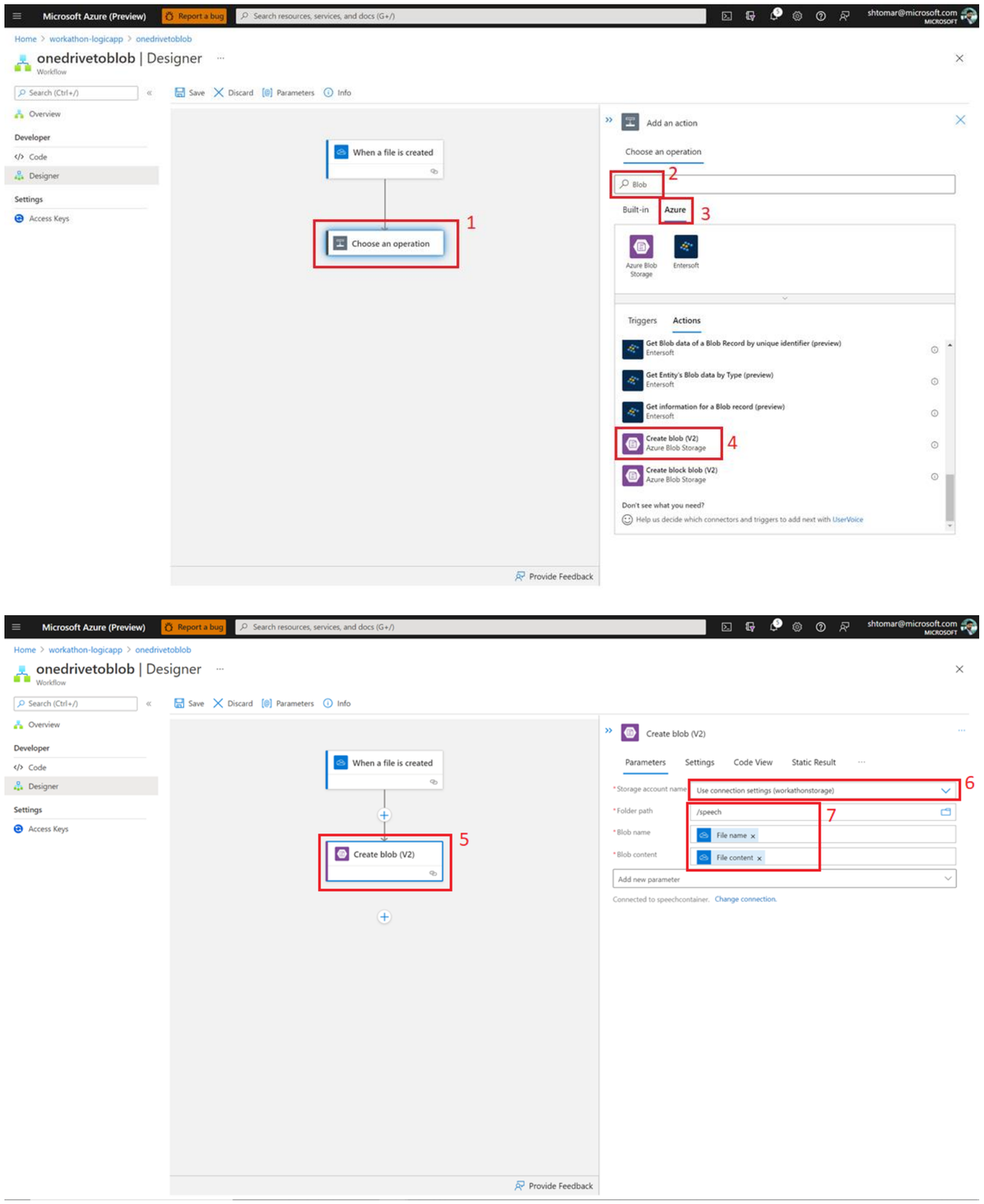
Logic Apps flows begin with a Trigger, which is an operation that executes the workflow as soon as it occurs. We are using 'OneDrive for business' as the initial source for meeting recordings and this flow will trigger every time a new file is added to the set OneDrive location.

Follow the steps as highlighted to add the trigger module :

1. Click on Choose an operation
2. Search for OneDrive
3. Change the tab to Azure
4. Select 'OneDrive for Business' [Change the source if you build additional similar flows or have a different source location]

5. Select 'When a file is created' [You might be asked to authenticate and create a connection to your resource if you're connecting for the first time]

6. Keep the module selected
7. Navigate to the corresponding folder or file location
8. Set the trigger check time to 15 seconds and set the time zone



Create blob step

In this step, we will upload the contents of the file to Speech container in Blob storage.

Follow the steps as highlighted to add the Create Blob module :

1. Click on Choose an operation
2. Search for Blob
3. Change the tab to Azure
4. Select Create blob (V2)

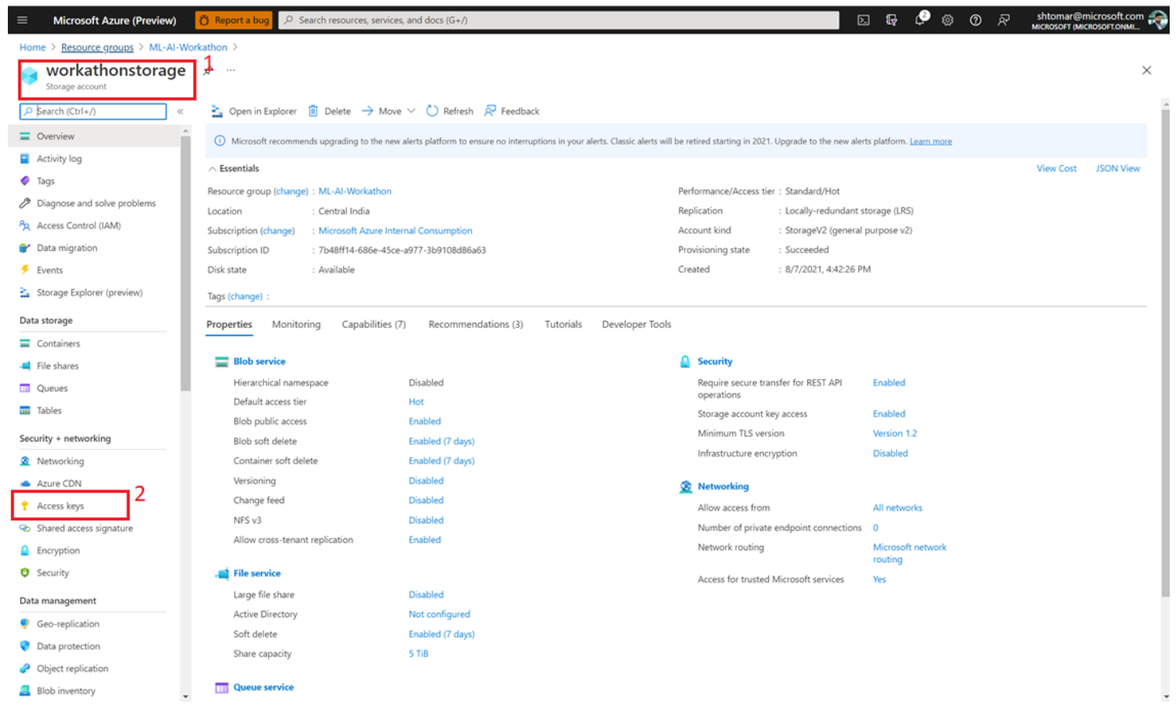
5. Keep the module selected
6. You will be asked to create a connection to the Blob Storage Account. [Refer the screenshots in 'Create a connection to Storage Account' section below to create this connection]

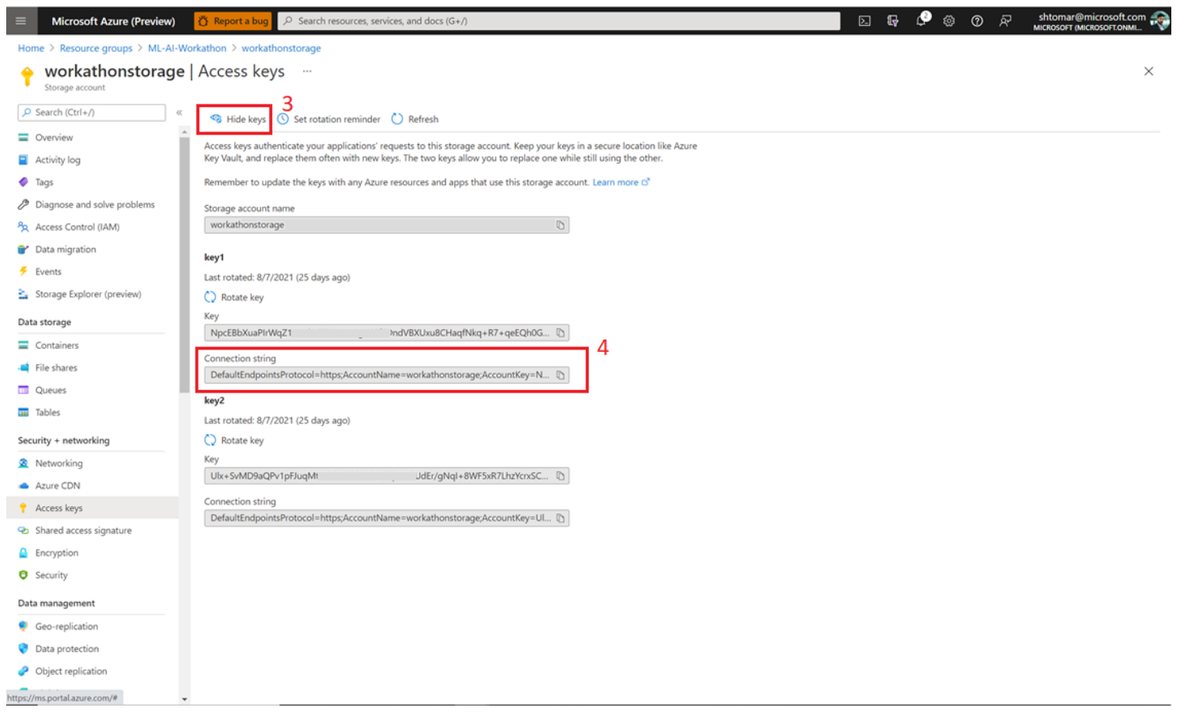
7. Folder Path : /speech [In case you named container 1 differently, make sure to change this]

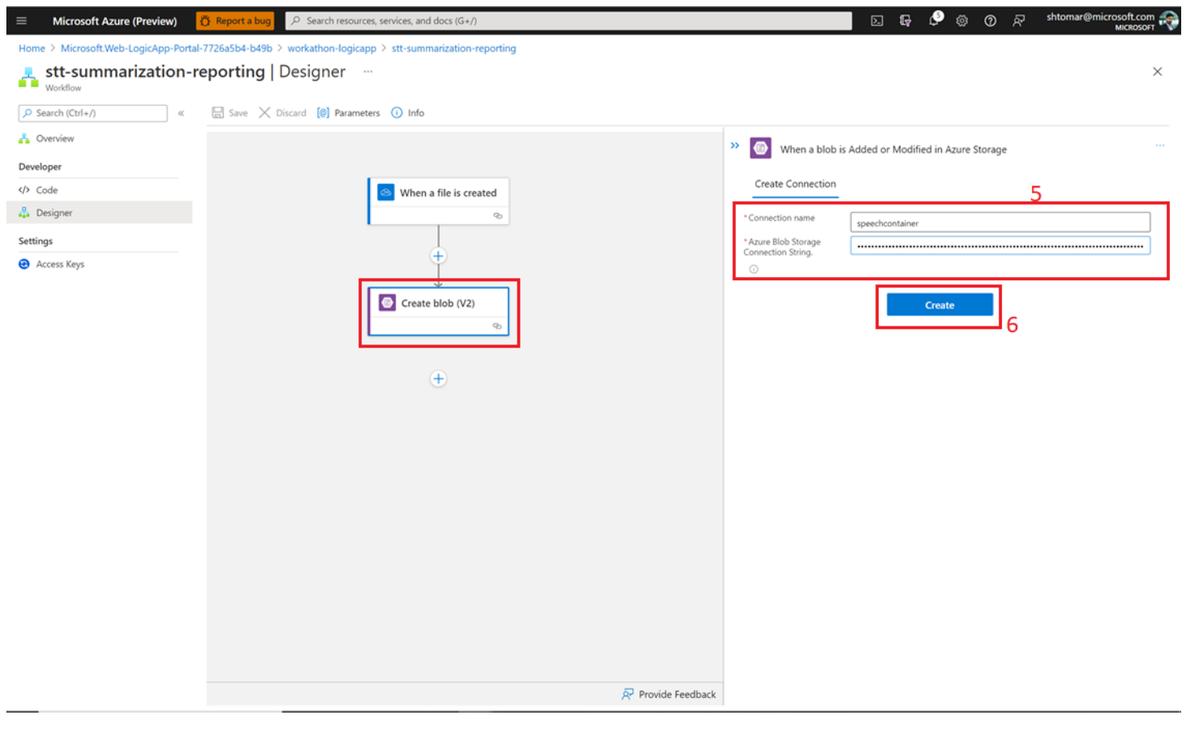
   Blob Name : Dynamically pick the 'File Name' from the trigger step

   Blob Content : Dynamically pick the 'File Content' from the trigger step

   [When you click in the input space for Blob name or Blob Content, you will see all the available outputs from the previous step. Select the outputs as mentioned]







Create a connection to Storage Account

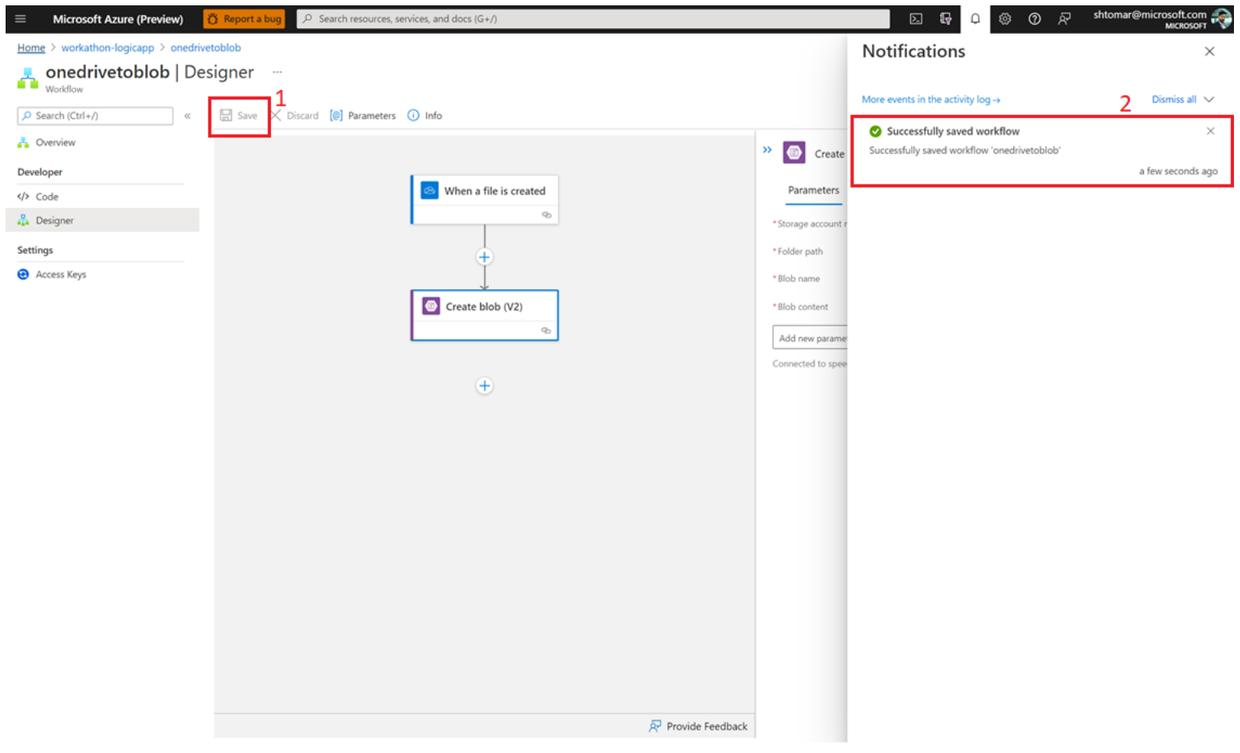
You need to create a connection to the storage account every time you leverage a new storage account in logic apps. We will need to do this once in this workshop since we will be using the same storage account [This connection is created at a storage account level, not at a container level]

1. Go to the storage account you created earlier
2. Select Access Keys

3. Click Show keys
4. Copy the Connection string

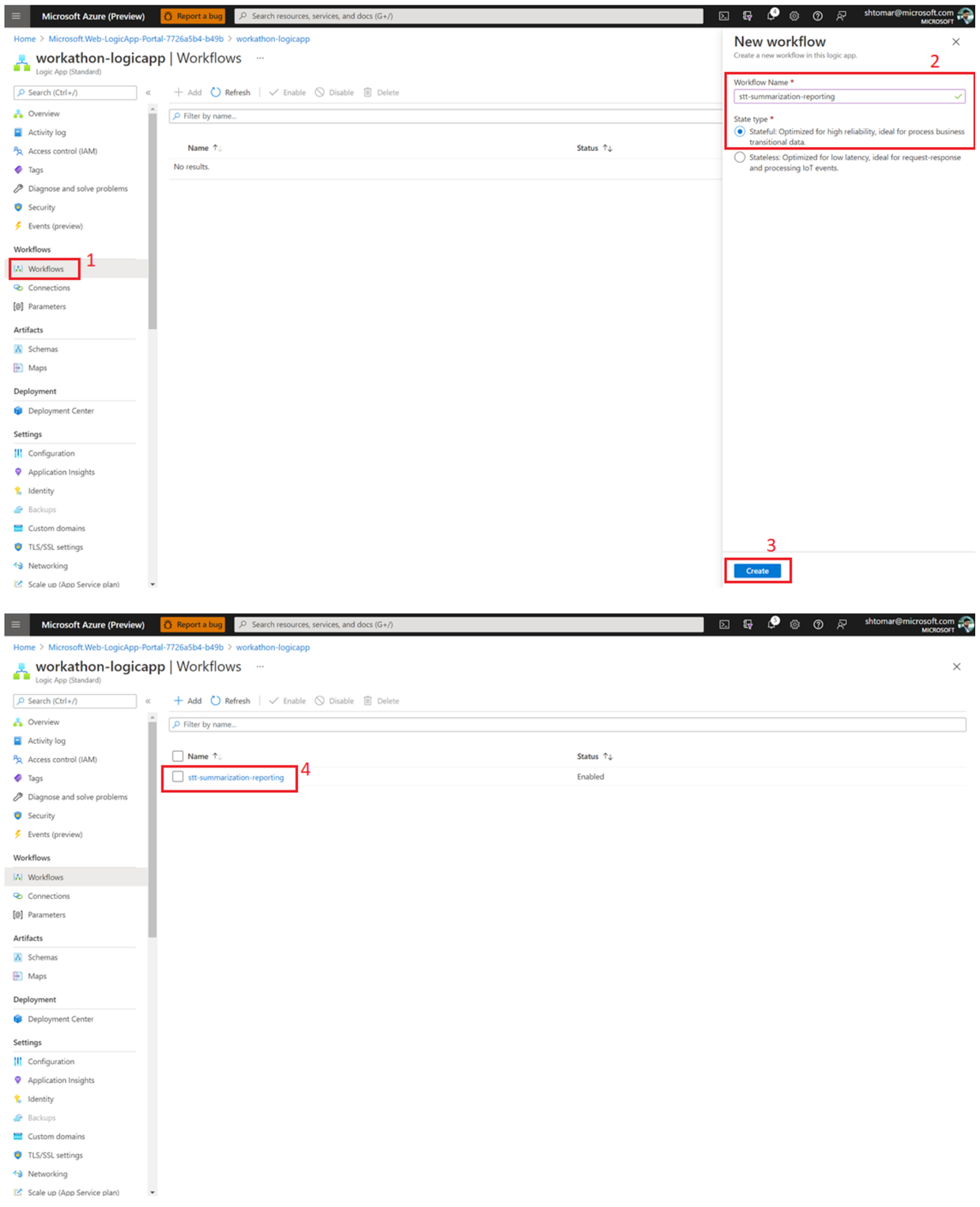
5. Go back to logic apps, give your connection a name and paste the connection string
6. Click Create

This creates a connection to the storage account. Continue with the steps above.



Save the workflow

1. Click Save
2. Wait for the workflow to be saved successfully. In case you get any errors or warnings, cross check with the steps above.



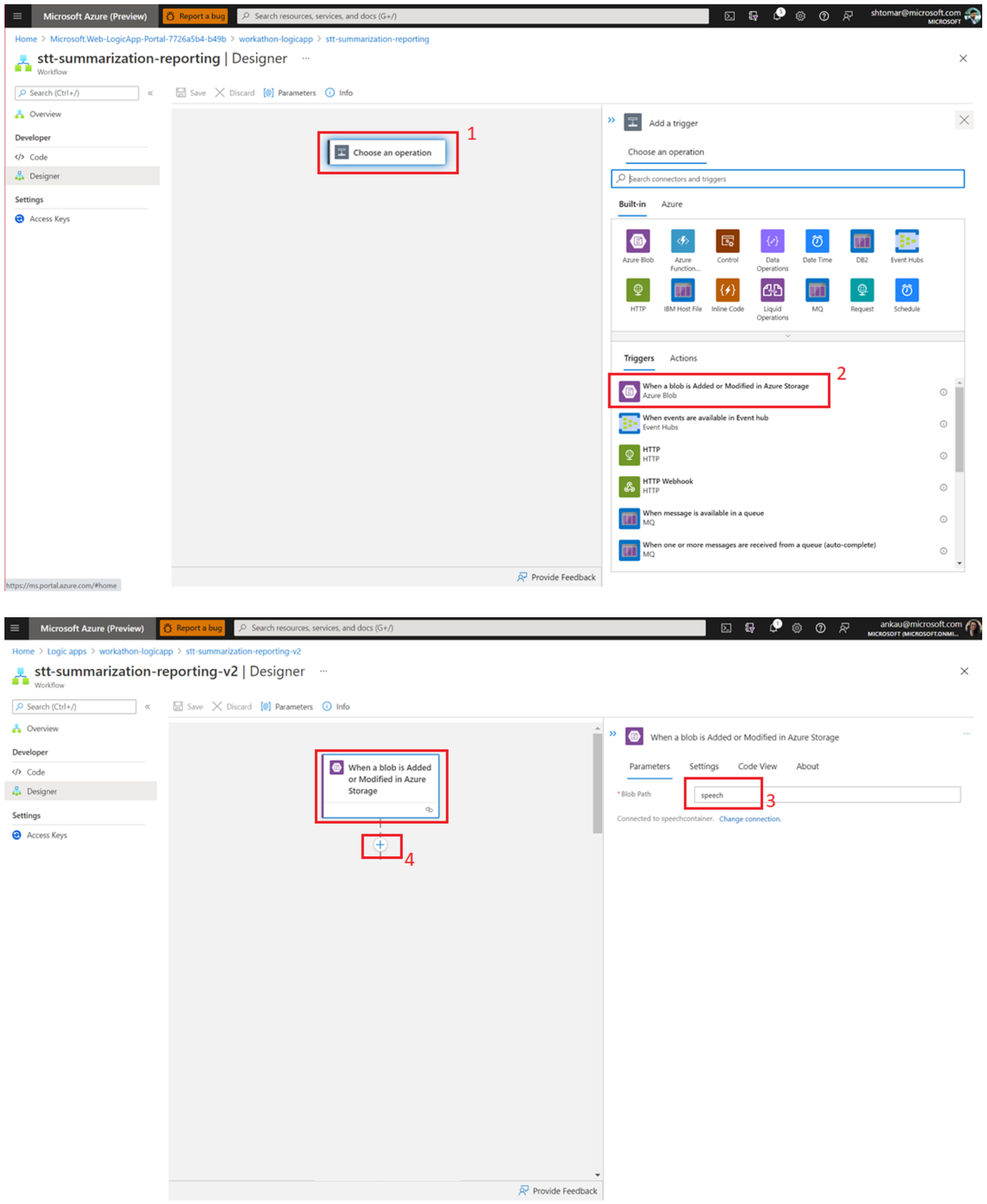
Create Logic App Workflow 2 : stt-summarization-reporting

Prior to moving on in the workshop, make sure you have performed Speech and Text Analytics workshop and clearly understand how the Asynchronous APIs [For STT and Text Analytics] that require multiple POST & GET calls work. If not, please revisit the individual workshops before continuing.

To create a new Logic App workflow, follow the steps as highlighted :

1. Select the Workflows tab
2. Provide the Workflow name (stt-summarization-reporting) & Select State Type as Stateful
3. Click create

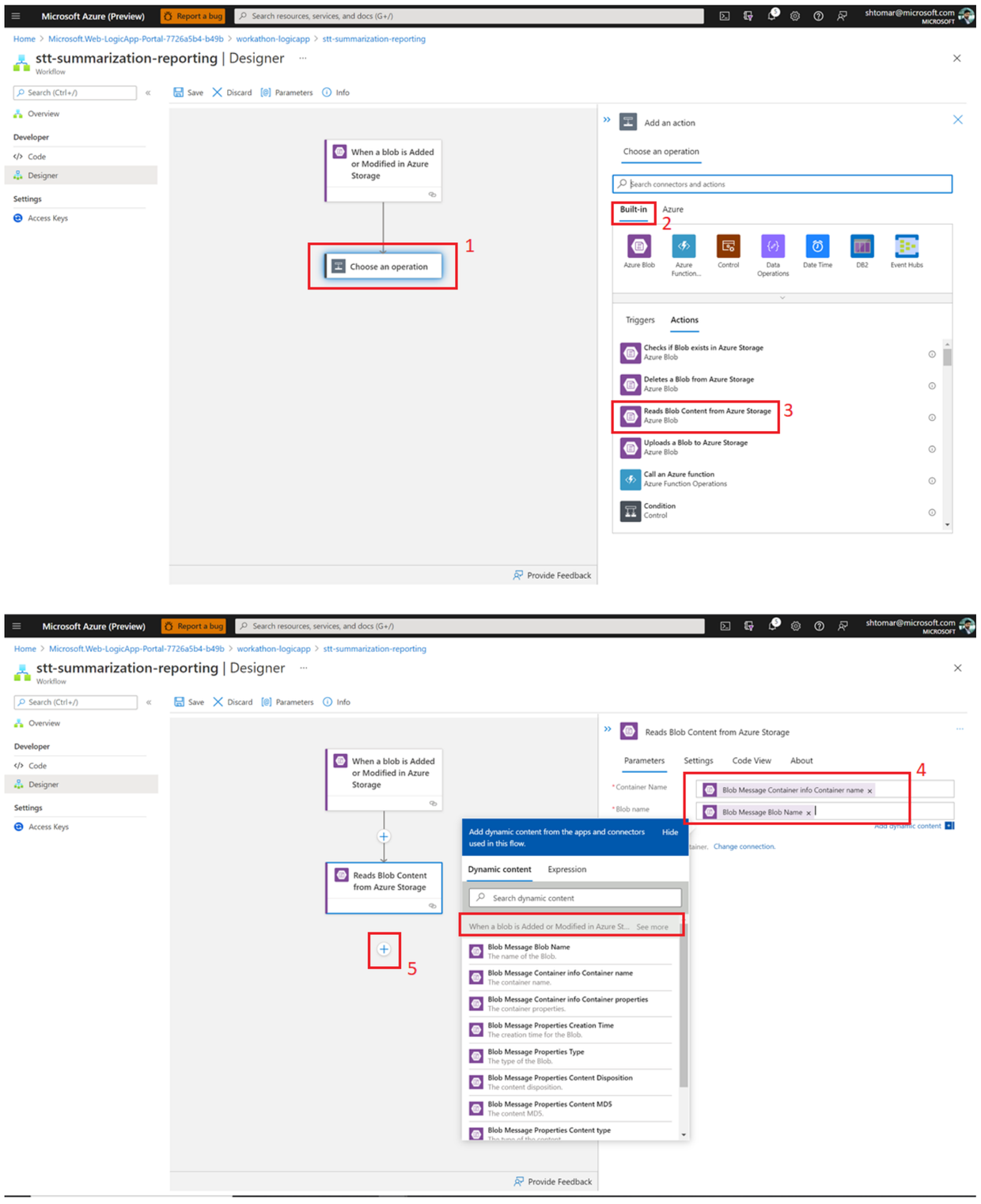
4. Once created, click on the Workflow name to start building the flow



Adding a trigger

Follow the steps as highlighted to add the trigger module :

1. Click on Choose an operation
2. In the built-in tab select ‘When a blob is Added or Modified in Azure Storage’ [This time, you shouldn’t be asked to authenticate since we created the connection in above flow]
3. Keep the module selected. This will show a message like ‘Connected to speechcontainer’ at the bottom of right tab
4. Blob path : speech [This is the name of container 1 that we created]

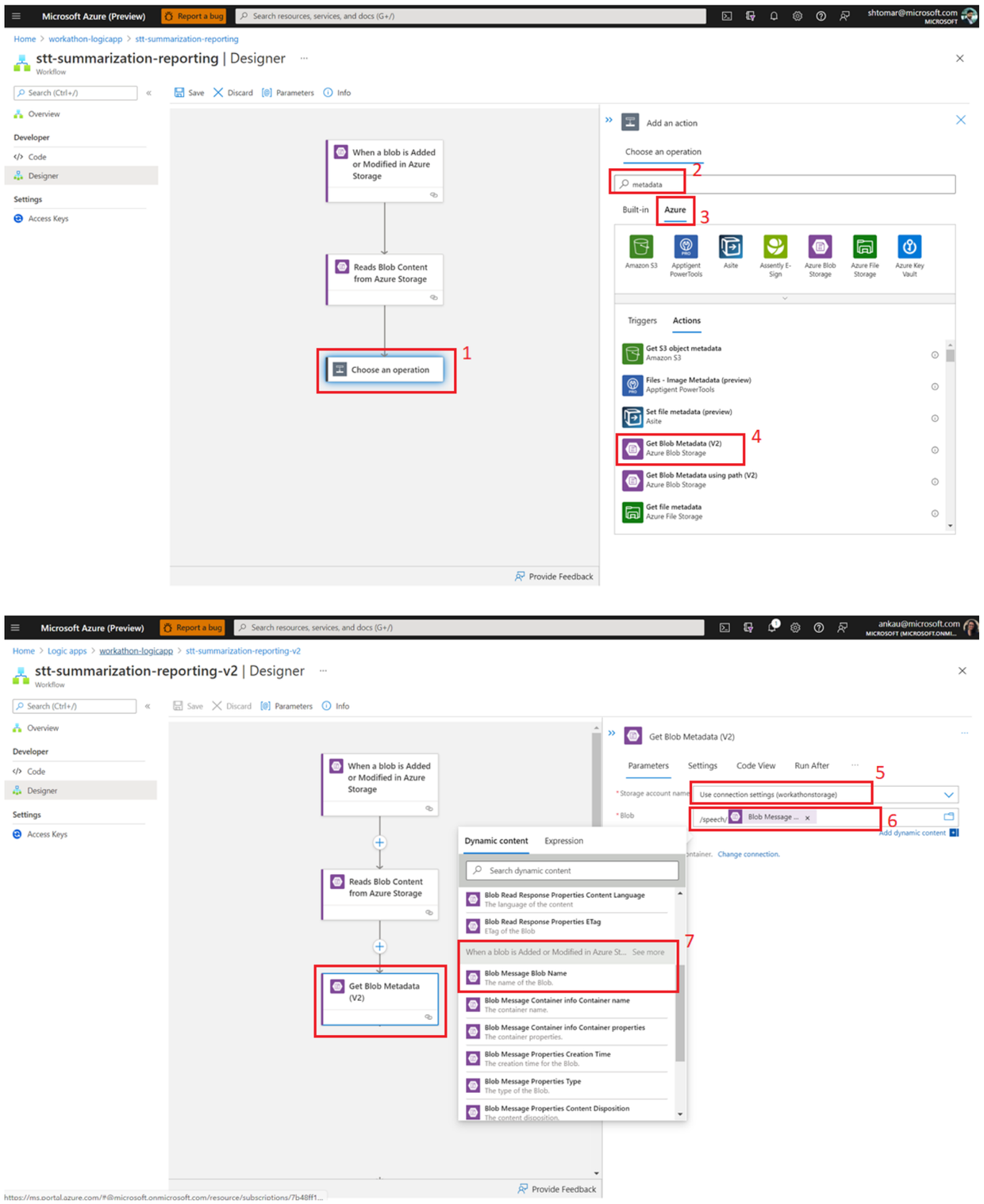


Create ‘Reads Blob Content from Azure Storage’ step

In this step, we are using the metadata from trigger step to read the contents of the blob that triggered this workflow.

1. Create a new step
2. Select built-in tab
3. Select ‘Reads Blob Content from Azure Storage’ module

4. Container Name : Blob Message Container info Container name (from output of previous step)
   Blob Name : Blob Message Blob Name

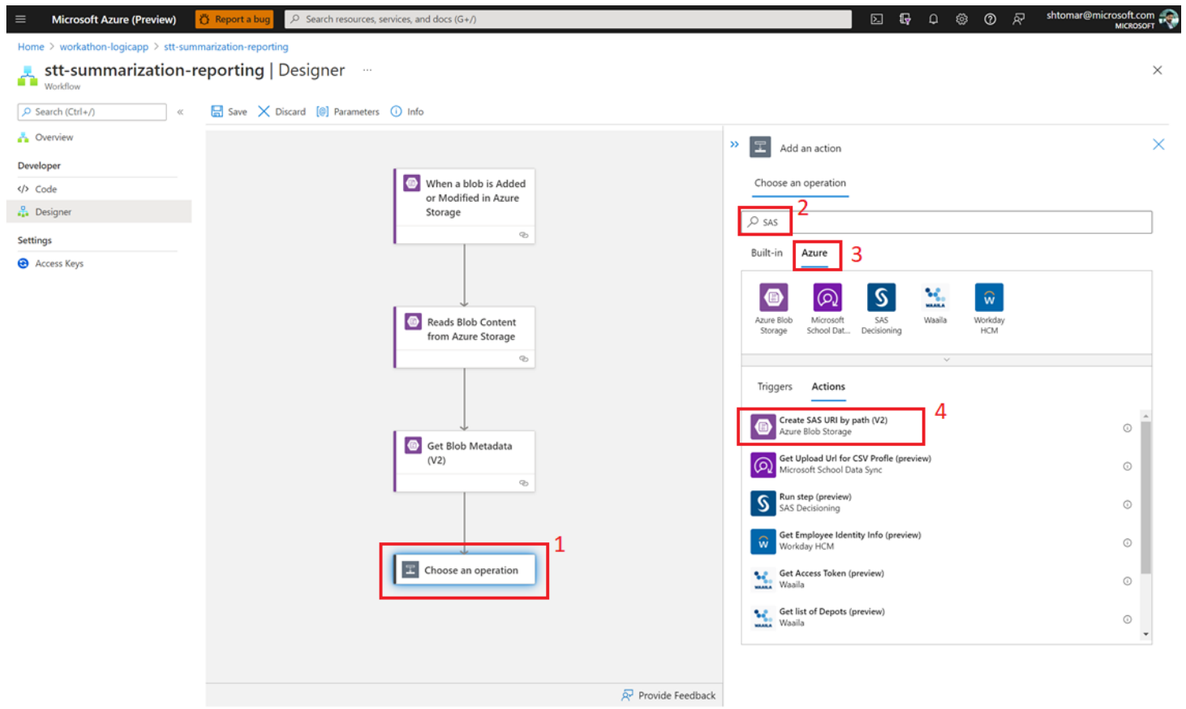


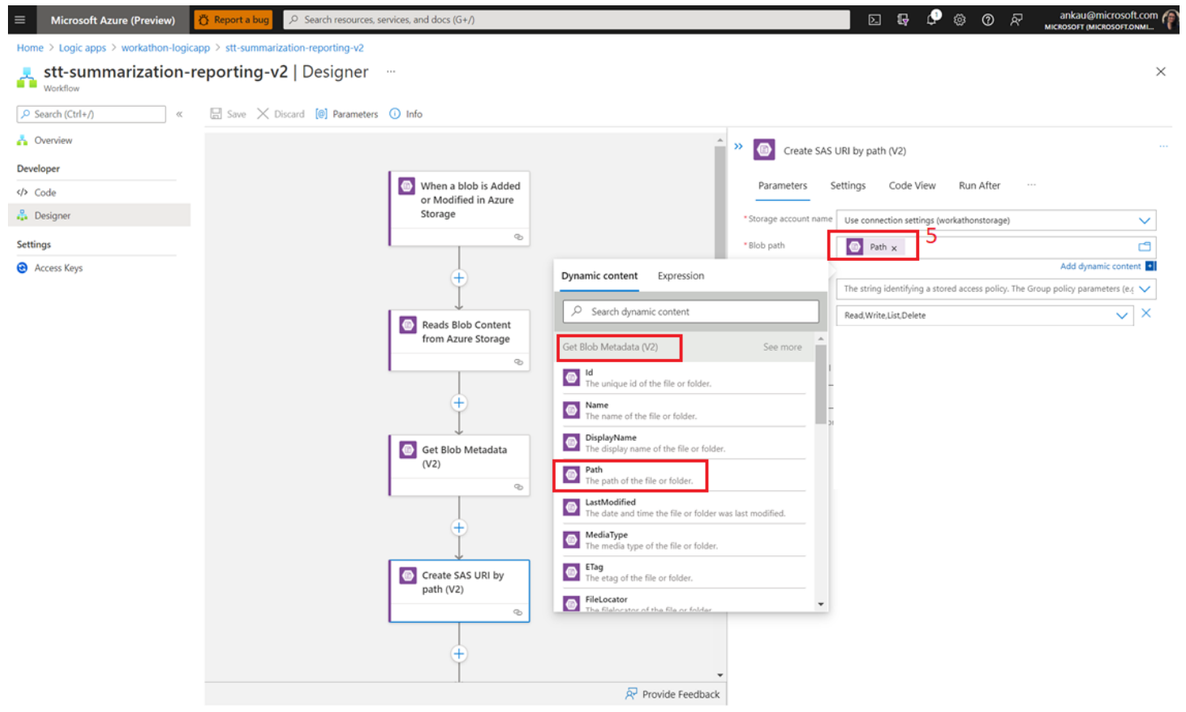
Create Get Blob Metadata step

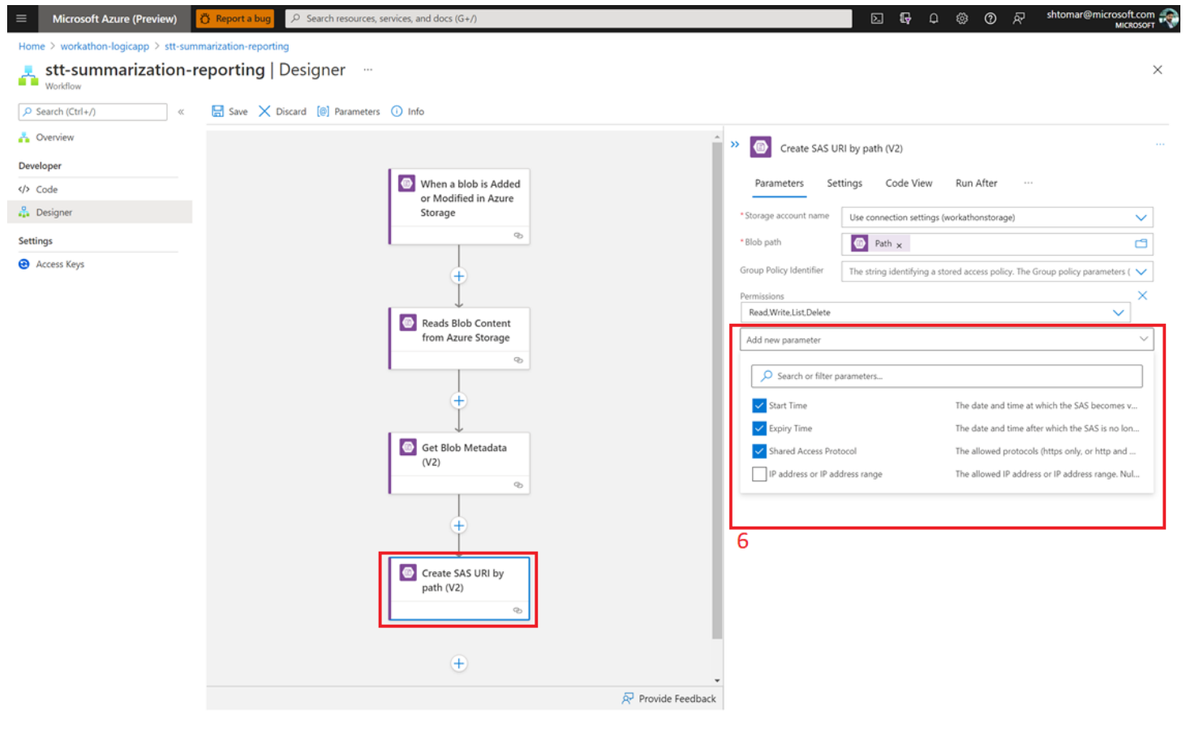
In this step, we are extracting the metadata of the blob that triggered the flow, to gain information like blob path that will be used in next step.

1. Create a new step and select Choose an operation
2. Search for metadata
3. Select Azure tab
4. Select ‘Get Blob Metadata(V2)’ module

5. Storage Account Name : From the dropdown select the storage account you previously created a connection to
6. Blob : /speech/<<Blob message name>>
   [Pick this from output of ‘when a blob is Added…’ step as shown in step 7]







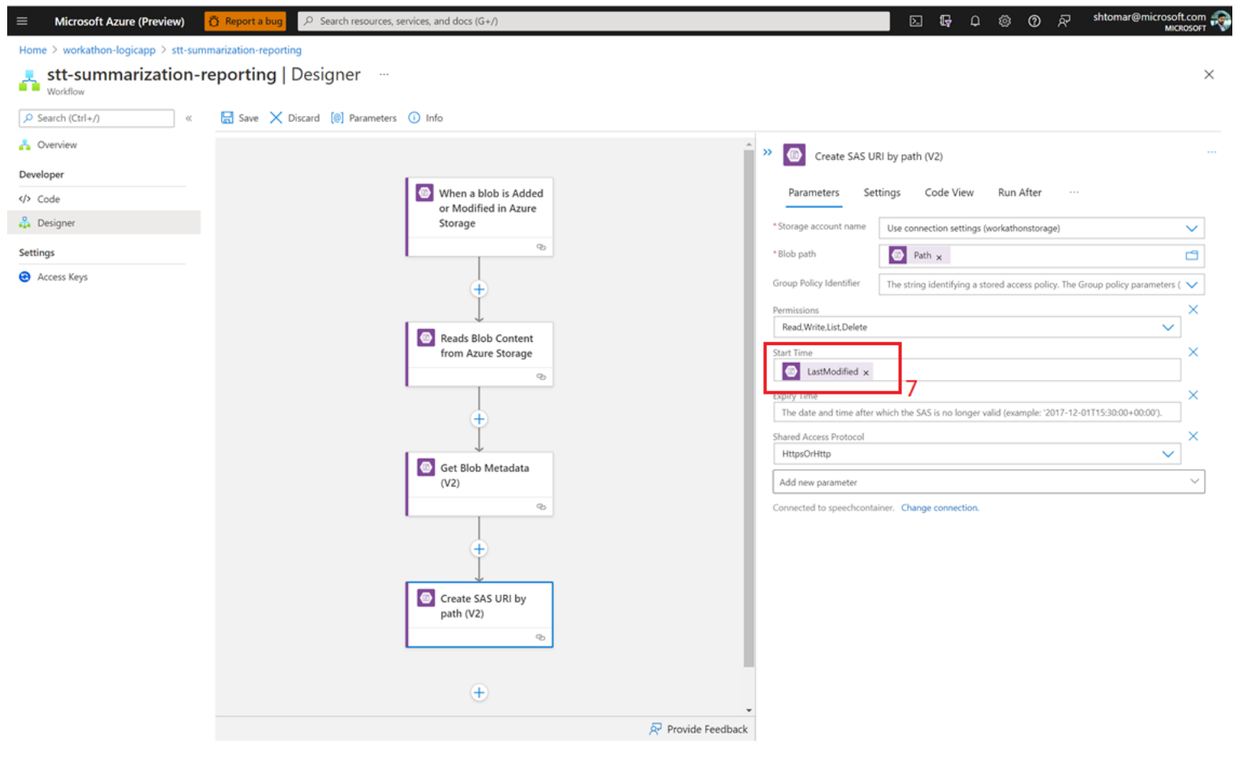
Create SAS URI step

In this step, we are creating a SAS URI for the blob that triggered this workflow. This will be passed in Speech-to-text API call in next steps.

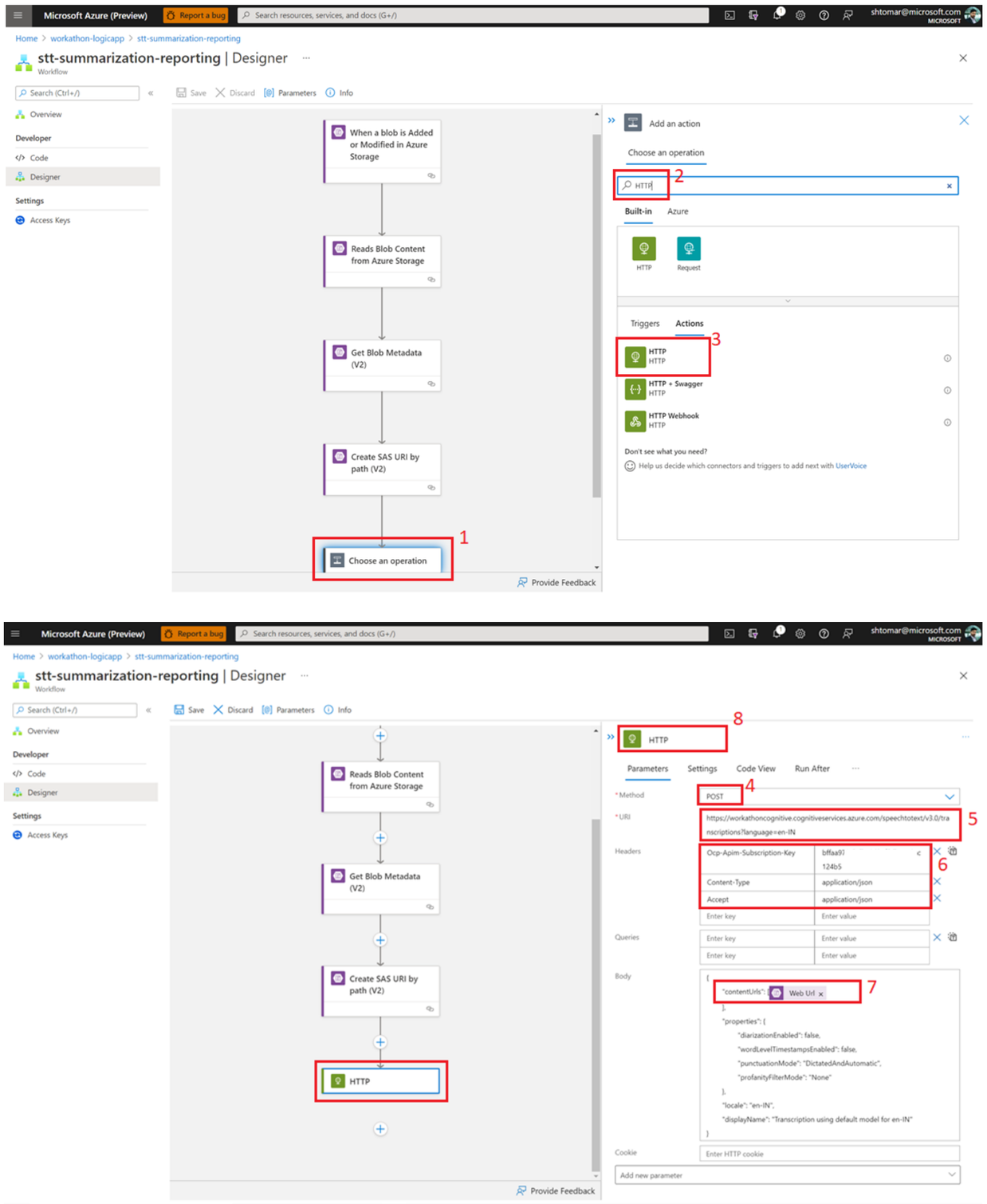
1. Create a new step and select ‘Choose and operation’
2. Search for SAS
3. Select Azure tab
4. Select ‘Create SAS URI by path (V2)’ module

5. Storage Account Name : From the dropdown select the storage account you previously created a connection to
   Blob path : Dynamically select ‘Path’ from output of ‘Get Blob Metadata’ step, as shown in screenshot

6. In add new parameters, select parameters as shown



7. Start time : Select ‘Last Modified’ from ‘Get Blob Metadata (V2)’ step
8. Leave other additional parameters as is



Speech-to-text async API call

In the next few steps, we will leverage modules like HTTP, JSON Parser, do until loops, delay etc to generate the transcript for incoming meeting recording. We will be using batch STT API call. This is a 3-step process :

1. 1 POST call
2. 2 GET calls

Since this is an async call, we have built some loops and added wait time to make sure we extract the results only after the call has been executed completely.

In case you are not familiar with the flow of Batch STT API calls, make sure to try out the Speech Service workshop prior to building this flow.

API CALL 1 : Speech-to-text-transcription

1. Create a new step
2. Search for HTTP
3. From built-in actions, select HTTP module
4. Method : POST

5. URI :
   https://workathoncognitive.cognitiveservices.azure.com/speechtotext/v3.0/transcriptions?language=en-IN
   [Replace workathoncognitive with the name of your cognitive service resource. This is the URL for the 1st POST call in STT batch transcription API]

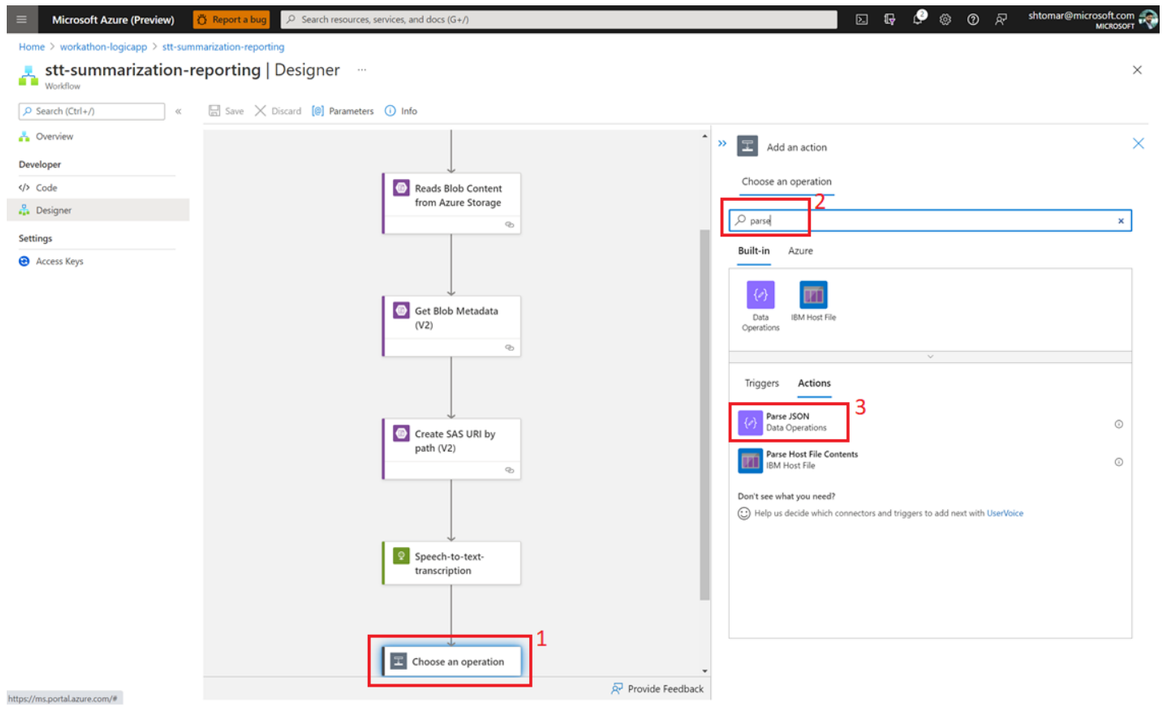
6. Headers :
   Ocp-Apim-Subscription-Key : <<Paste the API key for your cognitive service resource>>
   Content-Type : application/json
   Accept : application/json

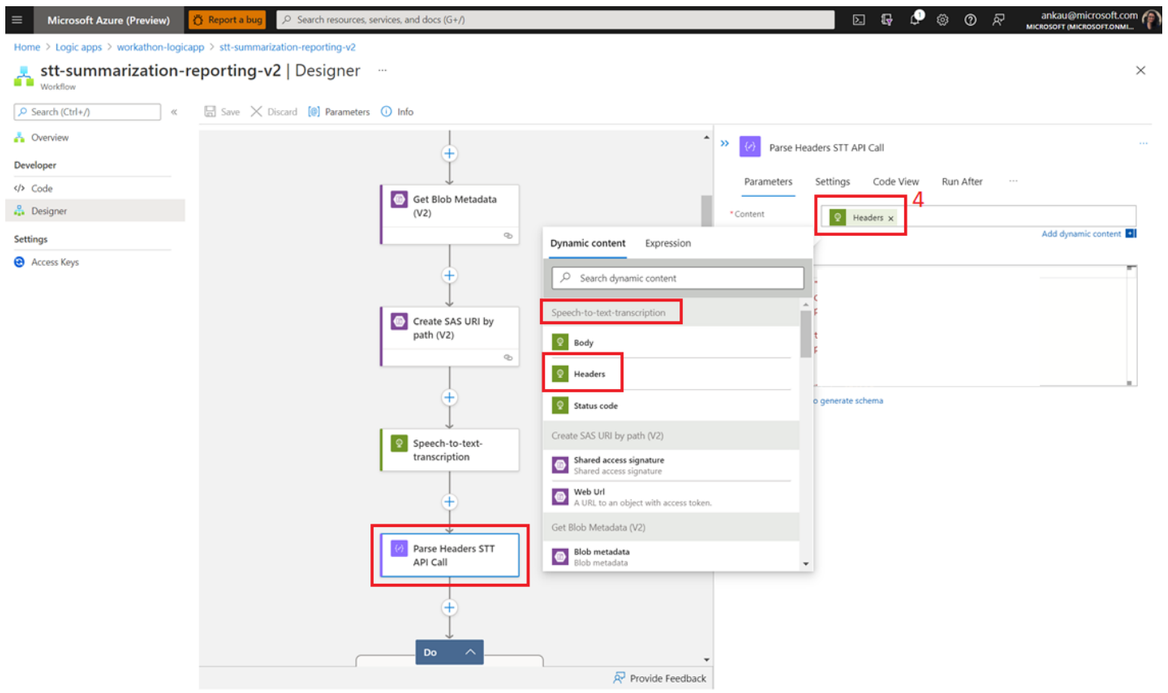
7. Body
   {
   "contentUrls": [
   <<Select the Web URL output from Create SAS URI output>>
   ],
   "displayName": "Transcription using default model for en-IN",
   "locale": "en-IN",
   "properties": {
   "diarizationEnabled": false,
   "profanityFilterMode": "None",
   "punctuationMode": "DictatedAndAutomatic",
   "wordLevelTimestampsEnabled": false
   }
   }

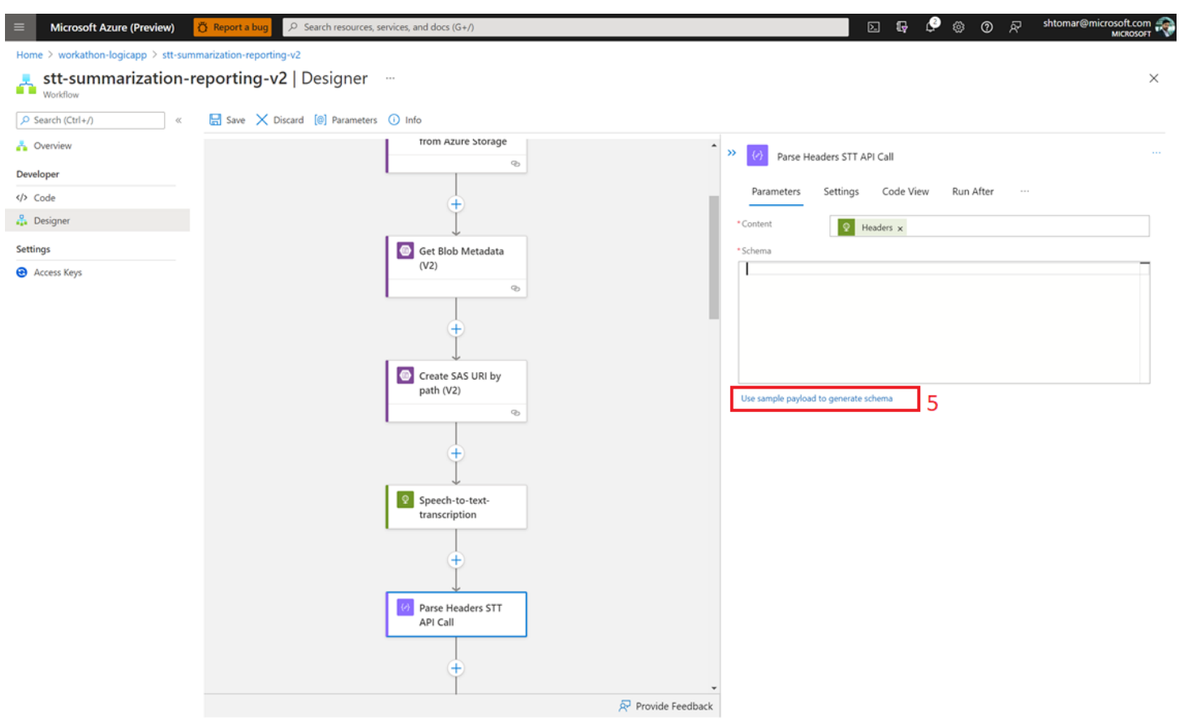
Alternatively, you can paste the below code as value for contentURL in the above body, *if you've used the same naming convention as ours so far - @{body('Create_SAS_URI_by_path_(V2)')?['WebUrl']}*

8. Rename module to Speech-to-text-transcription

See screenshot to verify the format and fields you enter.







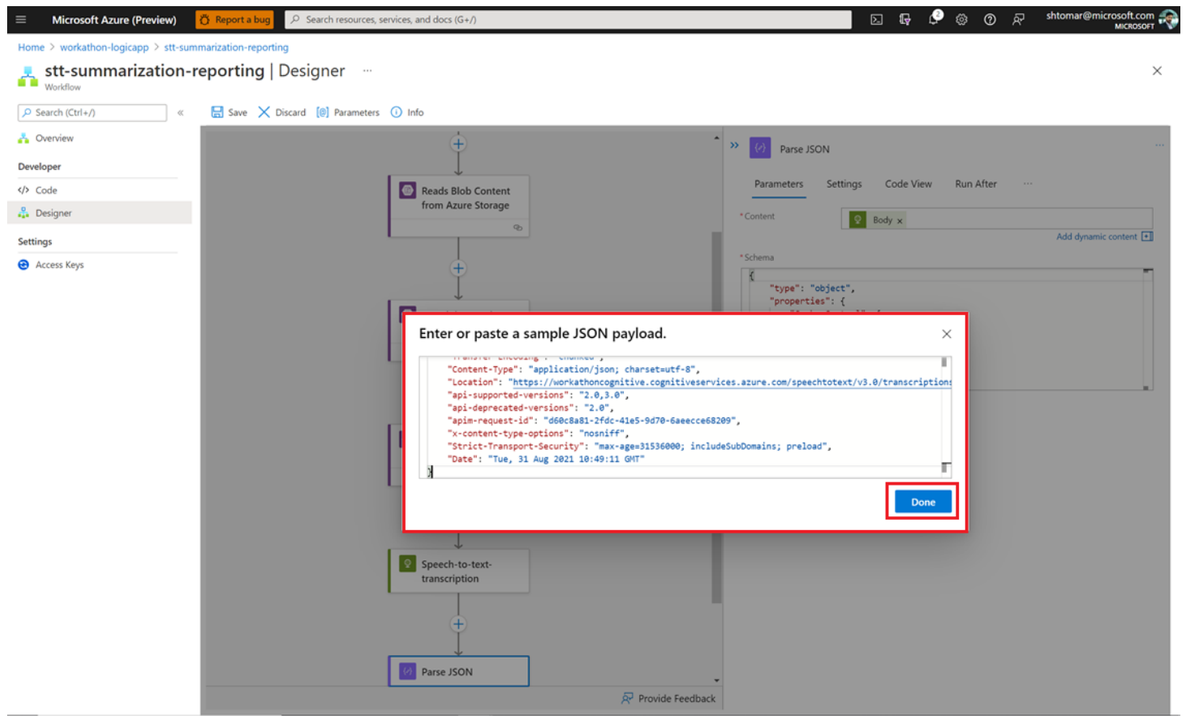
Parse Headers STT API step

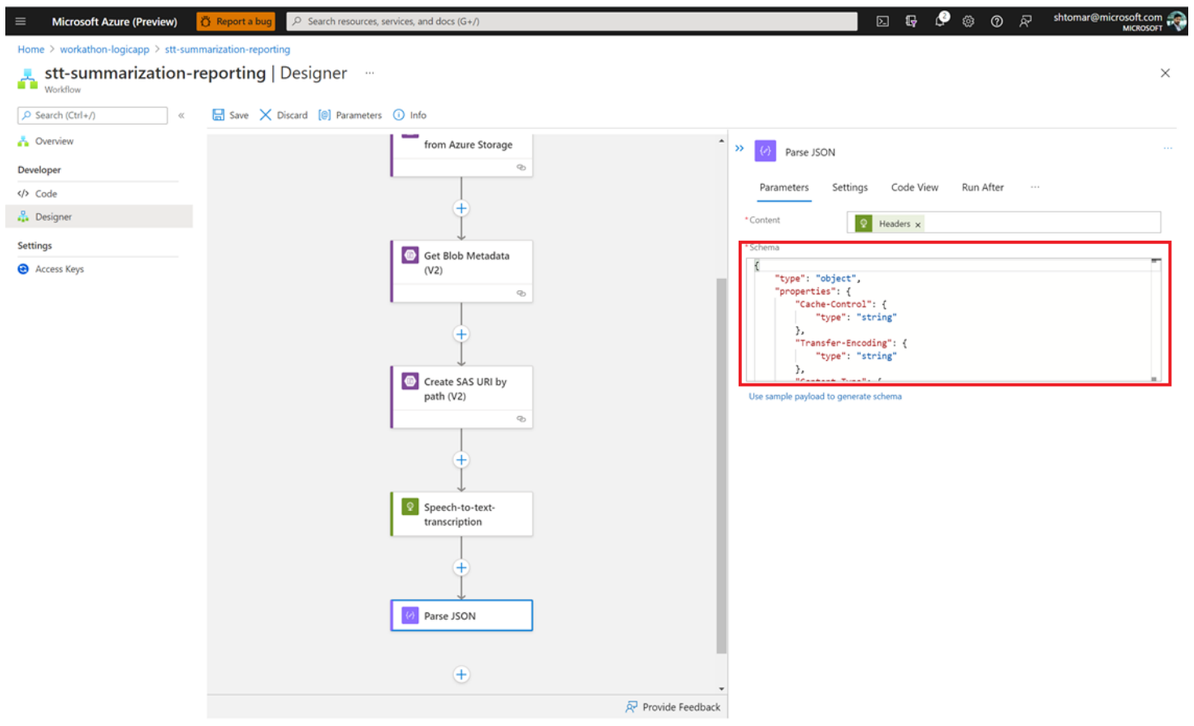
In this step, we will be parsing the 'header' content received from the previous POST call, to extract the parameters in it.
You can't leverage the individual parameters unless you the Parse the 'headers' output as a JSON, for which you require the JSON schema definition as shown in step 6.

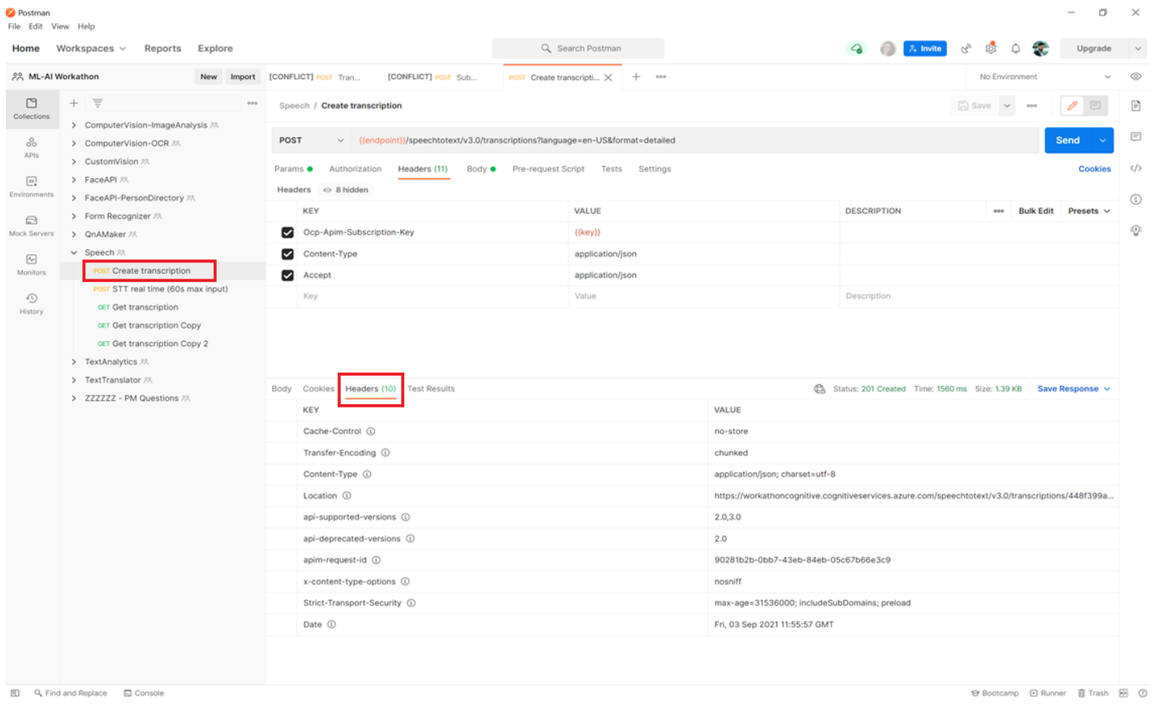
1. Create a new step
2. Search for parse
3. Select 'Parse JSON' module from built-in actions and rename the step to 'Parse Headers STT API Call'.

4. Content : Dynamically, select 'Headers' as part of 'Speech-to-text-transcription' step

5. Select 'use sample payload to generate schema'







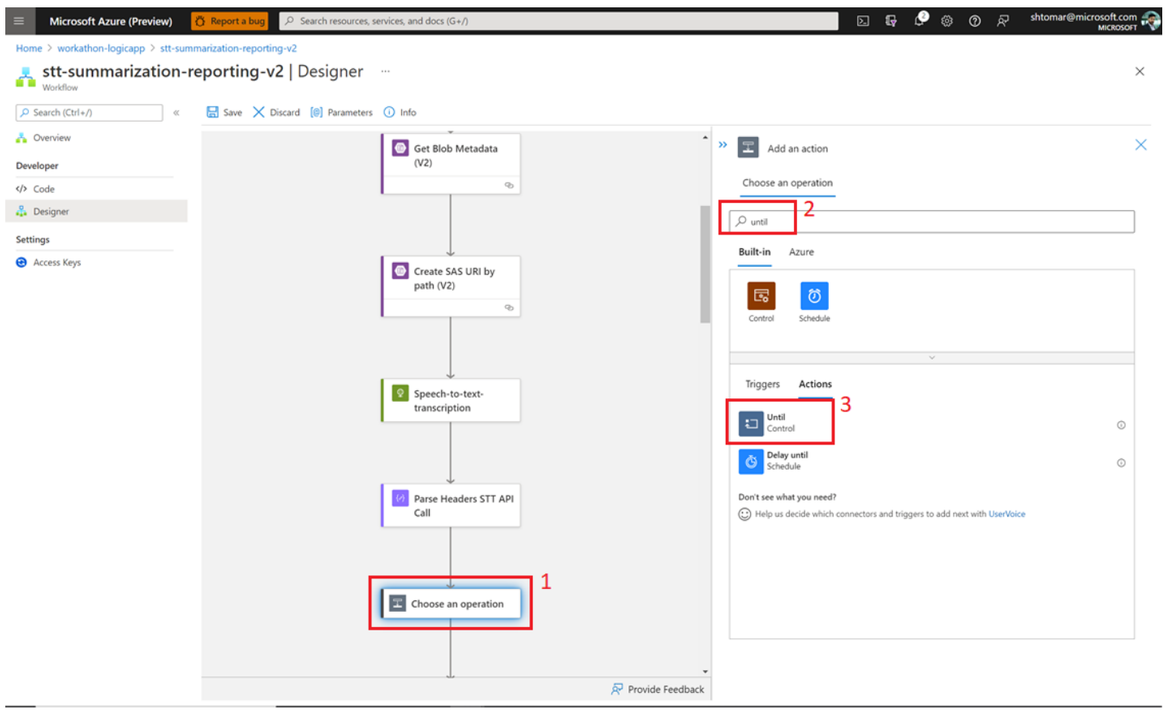
6. Paste the following and click Done

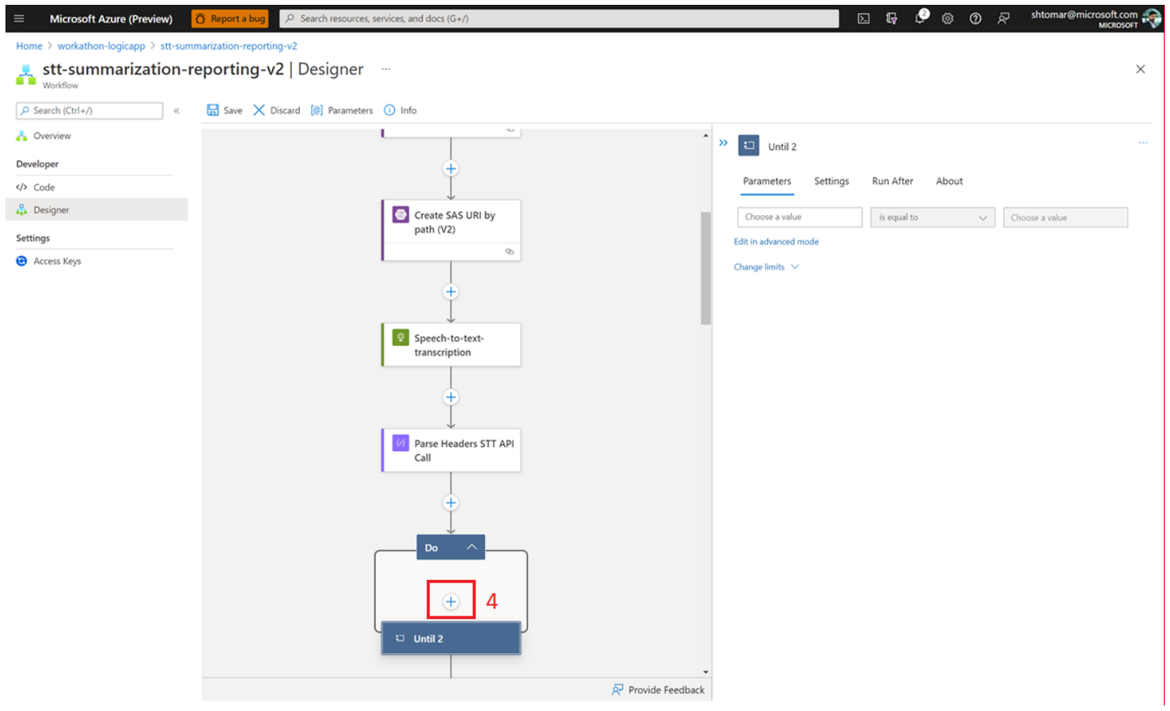
{
"Cache-Control": "no-store",
"Transfer-Encoding": "chunked",
"Content-Type" : "application/json; charset=utf-8"
"Location": "https://workathoncognitive.cognitiveservices.azure.com/speechtotext/v3.0/transcriptions/3b409298-250c-4bcf-a7ec-116188105558",
"api-supported-versions" : "2.0,3.0"
"api-deprecated-versions" : "2.0"
"apim-request-id": "d60c8a81-2fdc-41e5-9d70-6aeecce68209",
"x-content-type-options": "nosniff",
"Strict-Transport-Security": "max-age=31536000; includeSubDomains; preload",
"Date": "Tue, 31 Aug 2021 10:49:11 GMT",
}

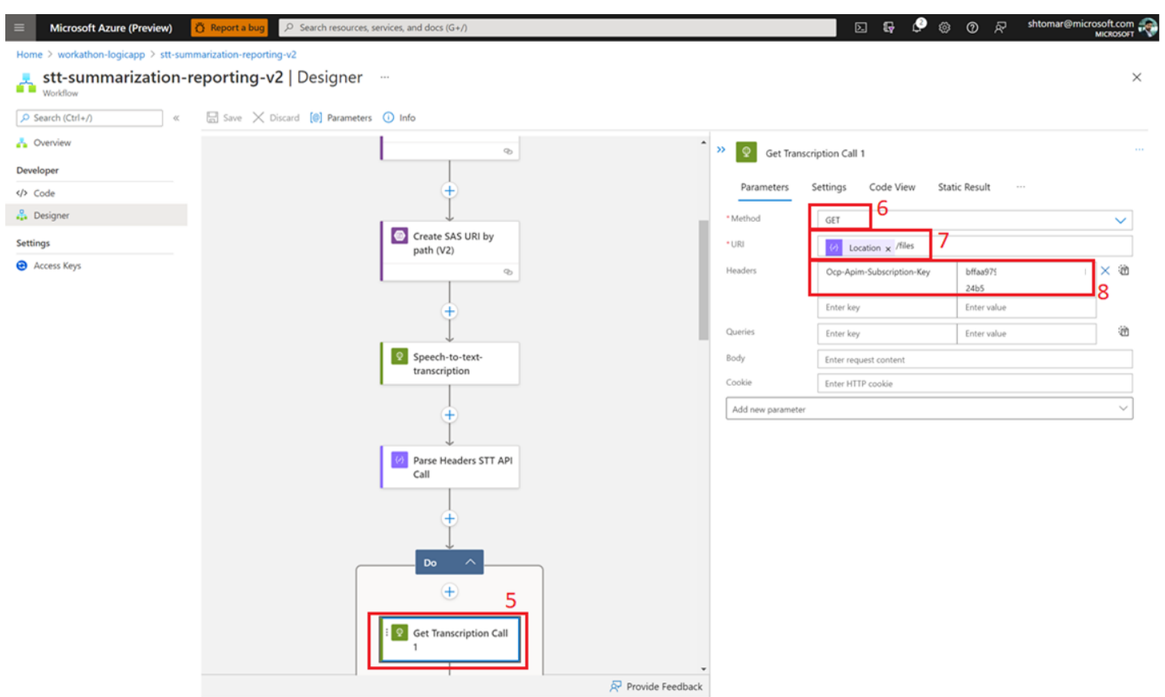
The Schema will be auto populated basis the Payload you provide.

The above payload JSON Schema has been defined using the parameters appearing under the 'Headers' section as returned by the 1st API POST call.

This screen should be familiar if you've performed the Speech Service workshop!







Add Until step

The next GET API call that we will hit always returns Status 200 OK on successful execution. However, the contents of the Body returned are not populated until the STT transcription is generated. Hence, we will add an 'until' loop that will repeatedly make the GET API call.

1. Create a new step
2. Search for until
3. Select 'Until' module from built-in actions

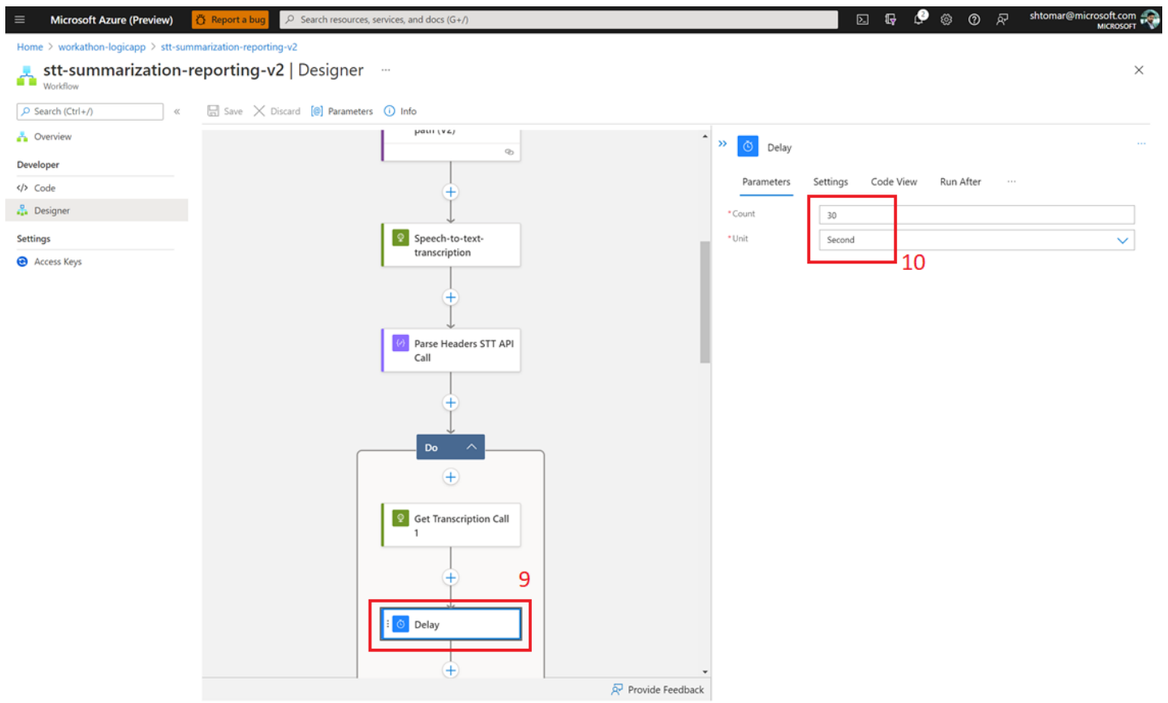
4. Click on + within do until loop, to add a step within it

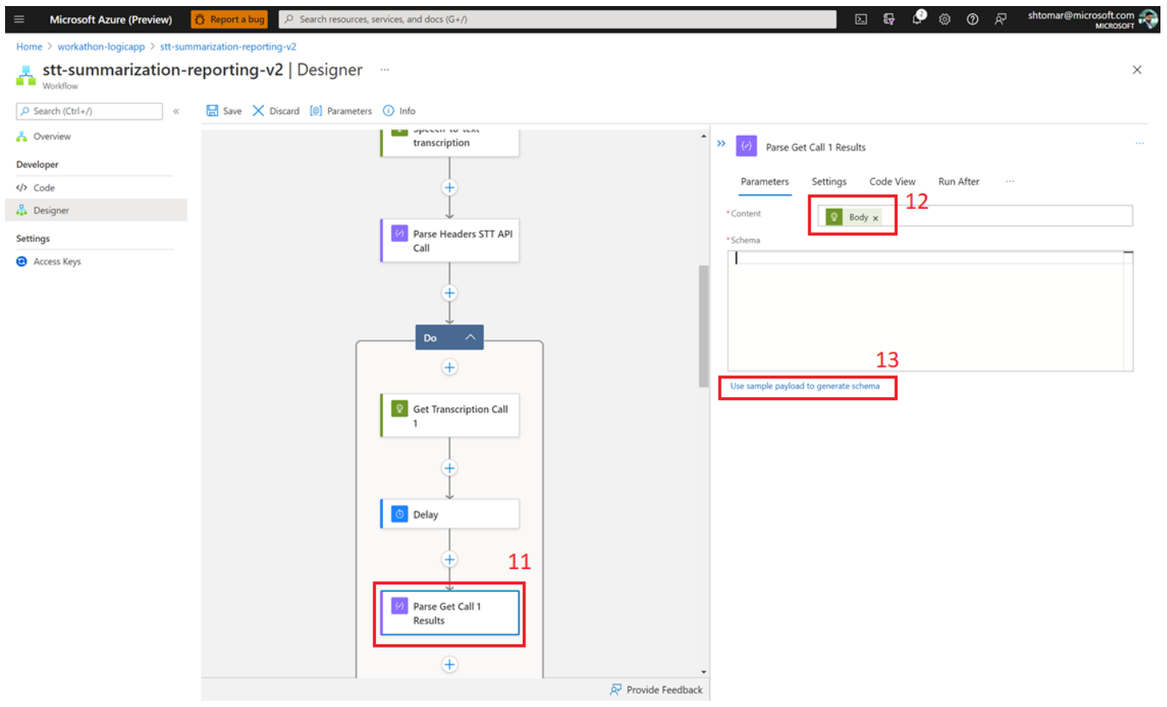
We will be adding the loop condition after adding all the modules in the loop, since the condition is dependent on one of the modules.

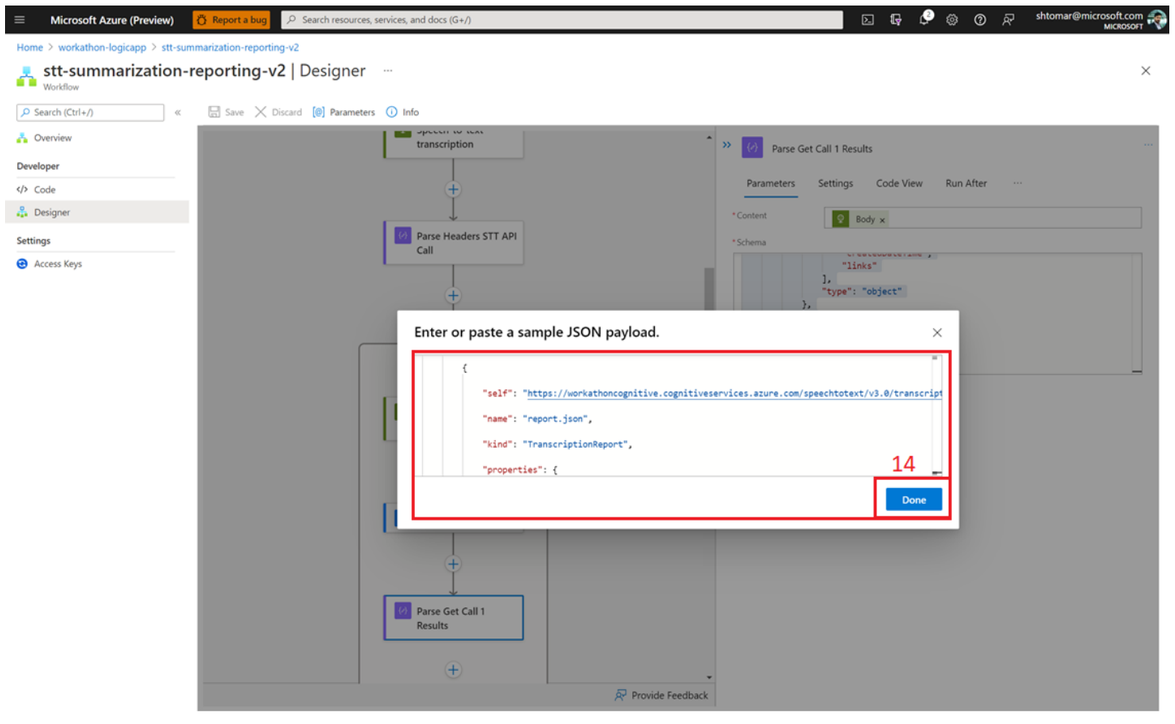
API CALL 2 : Get Transcription Call 1

5. Create a new step, Search for HTTP and from built-in actions, select HTTP module. Rename the step to 'Get Transcription Call 1'
6. Method : GET
7. URI : <<Dynamically pick 'location' from 'Parse Headers STT API Call' ouput>> and append '/files' to it [Refer screenshot to see format]
8. Headers :
   Ocp-Apim-Subscription-Key : <<Paste the API key for your cognitive service resource>>

Save the flow.
If you receive any errors or warnings, confirm you followed the right steps by referring the attached screenshots.







Add Delay step

The GET API call (Get Transcription Call 1) that we just hit, will take some time to return the intended output. The time depends on the size and length of the input file, which will differ for every file. Hence, we are adding a delay step, so that the call is not hit innumerable times. Thus, saving costs and load on the endpoint.

9. Create a new step, search for delay and select 'Delay' module from built-in actions
10. Fill in the following values for the parameters -
    Count : 30
    Unit : Second

[This will add a delay of 30 seconds between the recursive calls]

Add 'Parse Get Call 1 Results' step

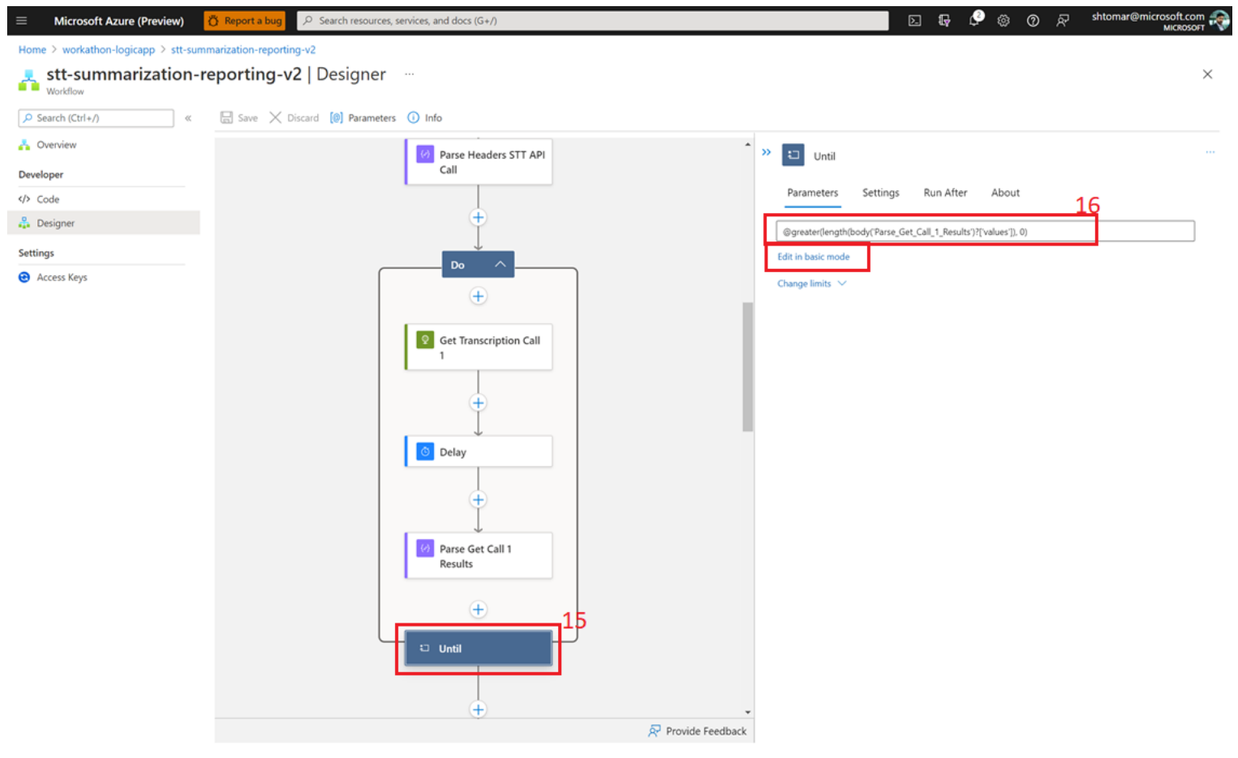
In this step, we will be parsing the Body content received from the previous 'GET transcription call 1', to extract the URL which points to the transcribed content of the input file. This URL will be used in the next GET API call.
You can't leverage the individual parameters unless you the Parse the 'Body' output as a JSON, for which you require the JSON schema definition as shown in step 14.

11. Create a new step, search for parse and select 'Parse JSON' module from built-in actions
12. Content : Dynamically, select 'Body' as part of 'GET transcription call 1' step. [Make sure you don't select the Body output that came from 'Speech-to-text-transcription' API Call]
13. Select 'use sample payload to generate schema'
14. Paste the following and click Done


```
{
  "values": [{
      "self": "https://workathoncognitive.cognitiveservices.azure.com/speechtotext/v3.0/
      transcriptions/fee10c94-9dcd-4c29-863c-47f31a2d80b2/files/
      28d4f3d0-c602-425f-92a9-b9afea338c38",
      "name": "report.json",
      "kind": "TranscriptionReport",
      "properties": {
        "size": 362
      },
      "createdDateTime": "2021-09-03T11:42:02Z",
      "links": {
        "contentUrl": "https://spsvcprodusw.blob.core.windows.net/
        bestor-338cca94-e7c5-4fc1-8a7b-7b8c19fd291a/TranscriptionData/
        fee10c94-9dcd-4c29-863c-47f31a2d80b2_report.json?sv=2020-08-04&st=2021-09-
03T12%3A11%3A01Z&se=2021-09-04T00%3A16%3A01Z&sr=b&sp=rl&
        sig=jFneRDK6GRMGhjoc6mEt%2FO0C6cXO%2FCCcvuqwBT%2B5GDw%3D"
      }
    }, {
      "self": "https://workathoncognitive.cognitiveservices.azure.com/speechtotext/v3.0/transcriptions/
      fee10c94-9dcd-4c29-863c-47f31a2d80b2/files/4607fd40-2f9f-460d-825a-cee793b14ba5",
      "name": "contenturl_0.json",
      "kind": "Transcription",
      "properties": {
        "size": 46385
      },
      "createdDateTime": "2021-09-03T11:42:02Z",
      "links": {
        "contentUrl": "https://spsvcprodusw.blob.core.windows.net/
        bestor-338cca94-e7c5-4fc1-8a7b-7b8c19fd291a/TranscriptionData/fee10c94-9dcd-4c29-863c-
47f31a2d80b2_0_0.json?sv=2020-08-04&st=2021-09-03T12%3A11%3A01Z&se=2021-09-
04T00%3A16%3A01Z&sr=b&sp=rl&
        sig=PlhGrUB%2BCl204Hwgguq%2FWYg5A3MwljinGB0mZHWIRME%3D"
      }
    }
  ]
}
```


The sample payload given above was fetched the Body of the Get transcription API call when executed via Postman in the Speech Service Workshop.

Also rename the step to 'Parse Get Call 1 Results'



Add condition for Until step

15. Click on Until module and click on Edit in Advanced Mode
16. Add the following condition :

@greater(length(body('Parse_Get_Call_1_Results')?['values']), 0)

Alternatively, you can create this expression from the GUI of “Add Dynamic Content” in basic mode.

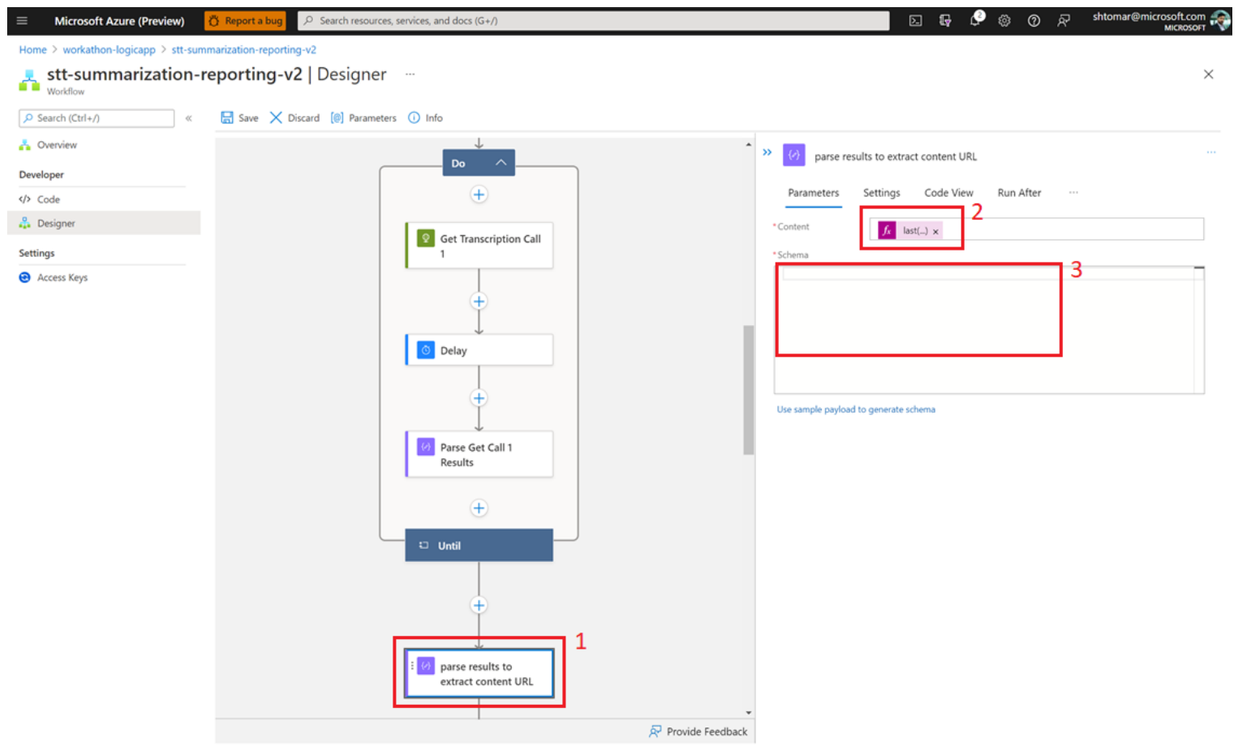
Note : The above expression will only work if you followed the naming convention the same as ours.

Significance :
The GET call keeps returning the following Body until the transcription is complete :

```
{
   "values": []
}
```

In the until condition, we are checking for the length of the ‘values’ array as returned in the Body. The loop will keep executing while the length is not greater than 0.
We will exit the loop once we obtain the values in the ‘values’ array .

Select change limits and enter the following :
Count : 5000 [You can increase this is you have longer files]
Timeout : PT8H [The amount of time that the Until loop can run before exiting and is specified in ISO 8601 format]



Parse results to extract content URL step

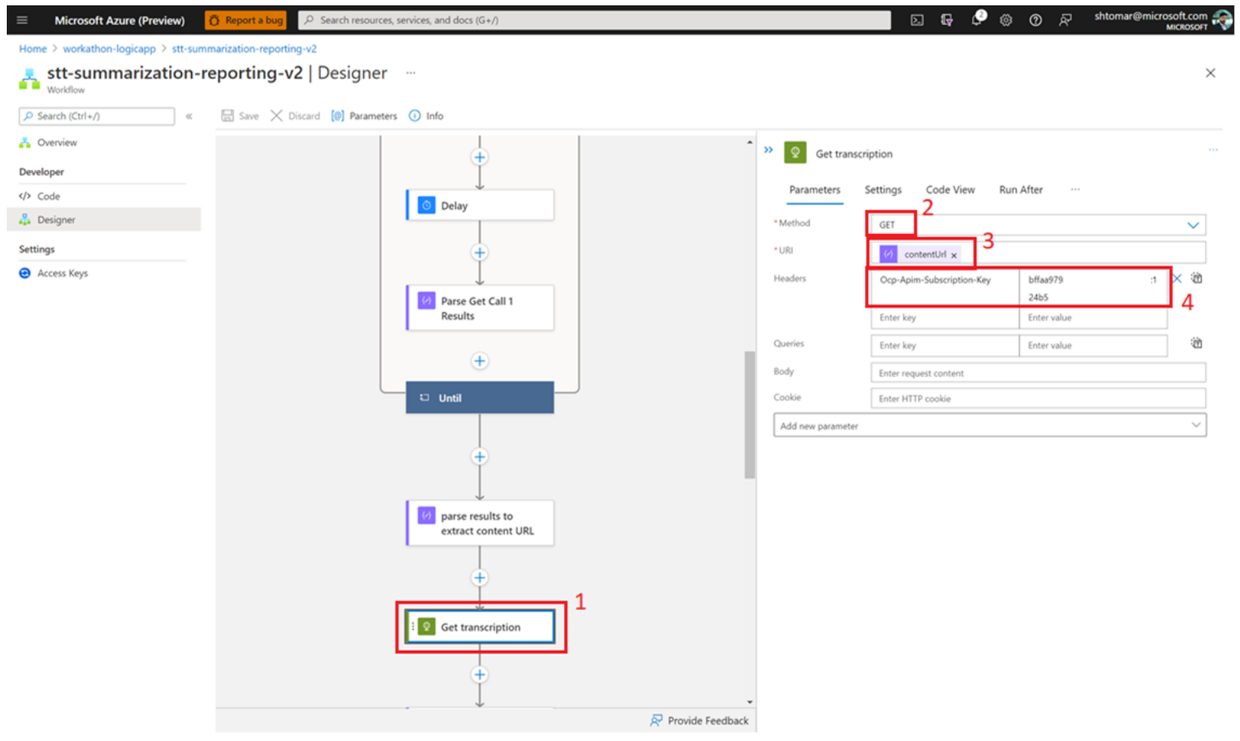
The ‘values’ array received in the body of ‘GET transcription call 1’ contains 2 elements. The first element contains ContentURL for the transcription report metadata. The last element contains ContentURL for the actual transcription.

In this step, we will be parsing the last element in the ‘values’ array, to retrieve the URL that points to the actual transcription for the final API call.

1. Create a new step, search for parse, select ‘Parse JSON’ module from built-in actions and rename step to ‘parse results to extract content URL’
2. Content : *@last(body('Parse_Get_Call_1_Results')?['values'])*
3. Paste the following in the schema input :

```
{
   "properties": {
      "createdDateTime": {
         "type": "string"
      },
      "kind": {
         "type": "string"
      },
      "links": {
         "properties": {
            "contentUrl": {
               "type": "string"
            }
         },
         "type": "object"
      },
      "name": {
         "type": "string"
      },
      "properties": {
         "properties": {
            "size": {
               "type": "integer"
            }
         },
         "type": "object"
      },
      "self": {
         "type": "string"
      }
   },
   "type": "object"
}
```

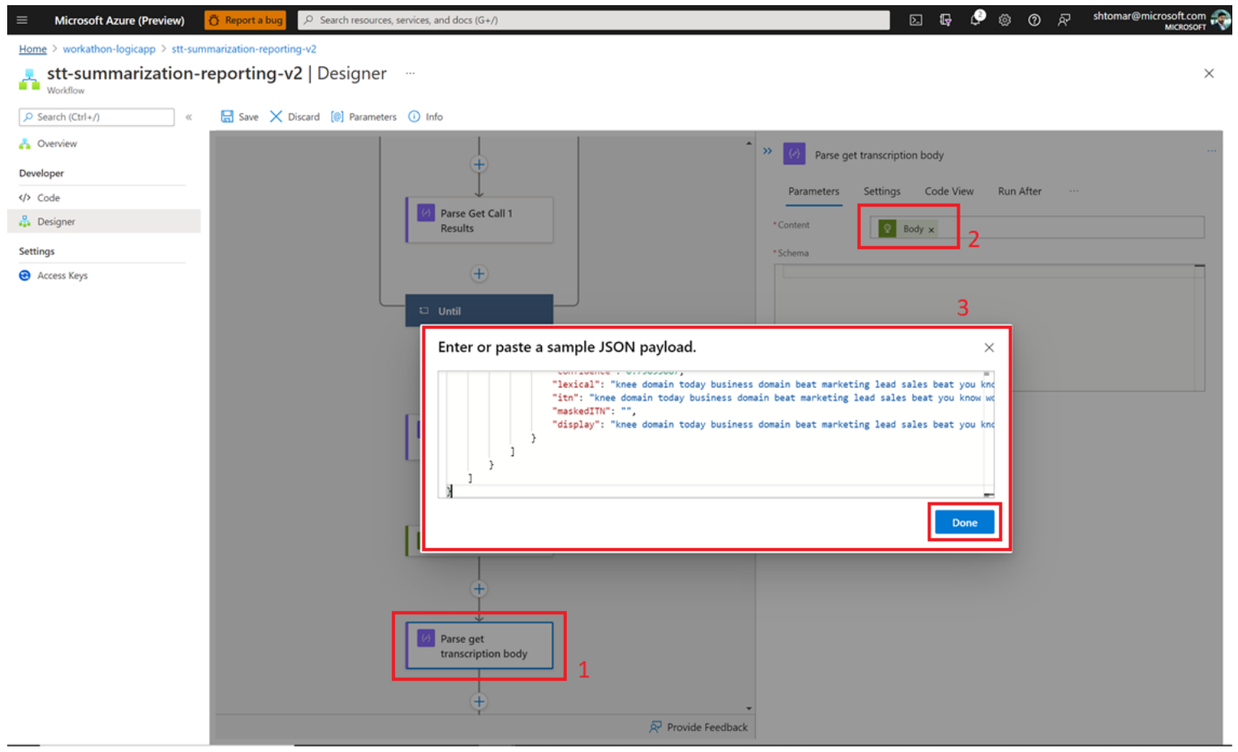
Please note this time we have directly shared the schema, not the JSON payload, hence we are pasting it in schema input and not JSON payload input.

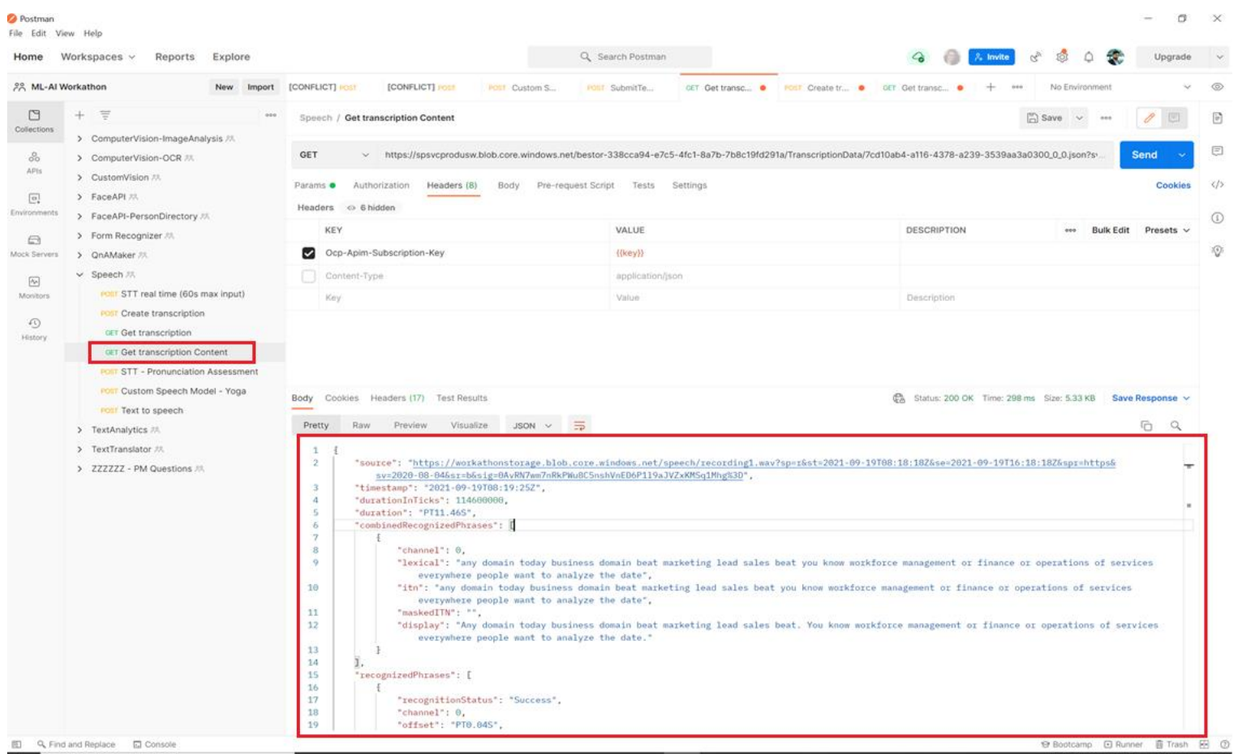


API CALL 3 : Get Transcription

1. Create a new step, search for HTTP and from built-in actions select HTTP module. Rename the step to 'Get Transcription'
2. Method : GET
3. URI : <<Dynamically pick 'ContentURL' from 'parse results to extract content URL' output.>>
4. Headers :
   Ocp-Apim-Subscription-Key : <<Paste the API key for your cognitive service resource>>

Save the flow. If you receive any errors or warnings, confirm you followed the right steps by referring the attached screenshots.





Parse Get transcription body step

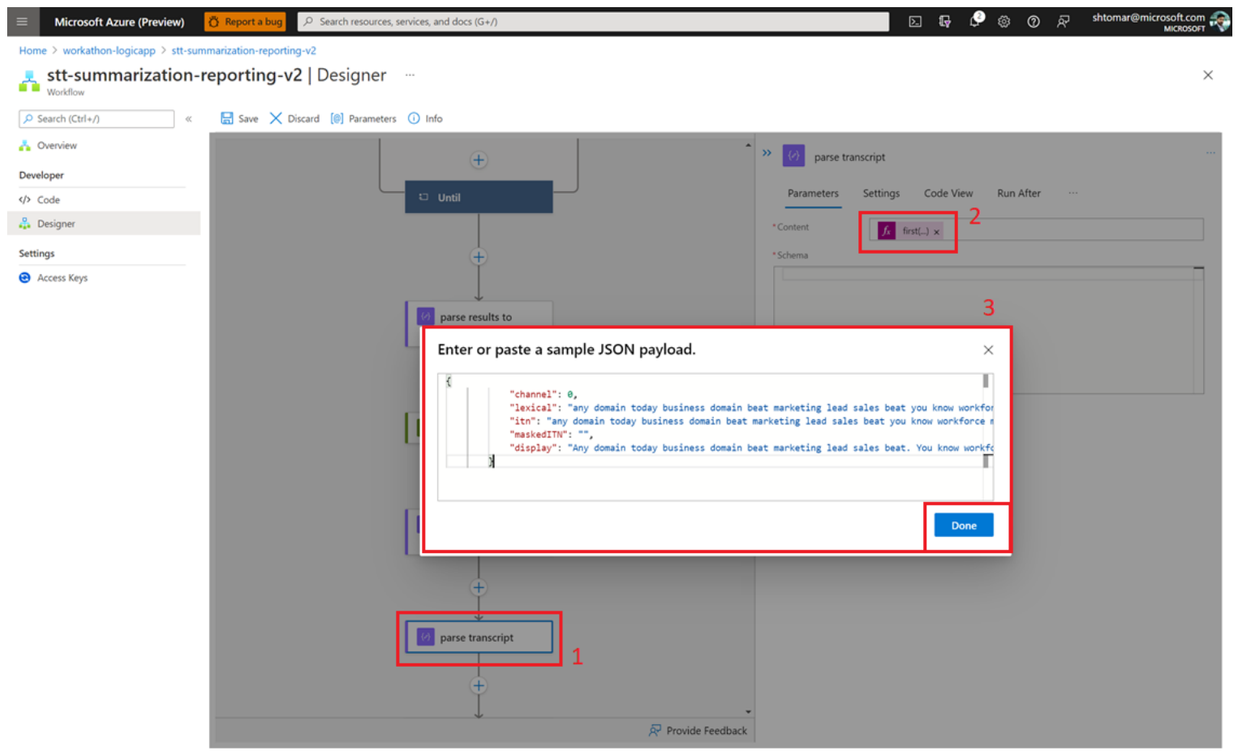
In this step, we will be parsing the Body content received from the previous 'GET transcription', to extract the final transcription.

You can't leverage the individual parameters unless you the Parse the 'Body' output as a JSON, for which you require the JSON schema definition.

1. Create a new step, search for parse, select 'Parse JSON' module from built-in actions and rename the step to 'Parse get transcription body'
2. Content : *<< Dynamically, select 'Body' as part of 'GET transcription' step >>* [Make sure you don't select the body output from any of the other API calls]
3. Select 'use sample payload to generate schema'
4. Fetch the sample JSON payload from the body output of the 3rd API call as made from Postman for Batch STT service in the Speech Service workshop.

Alternatively, you can paste the contents from the 'Parse get transcript body schema input' file directly in the 'schema' input. Download this file from the Code folder.

[Do note we have directly provided you the 'schema'. This is not the JSON payload. If you use this, paste this in 'Schema' input, not in JSON payload sample input.]

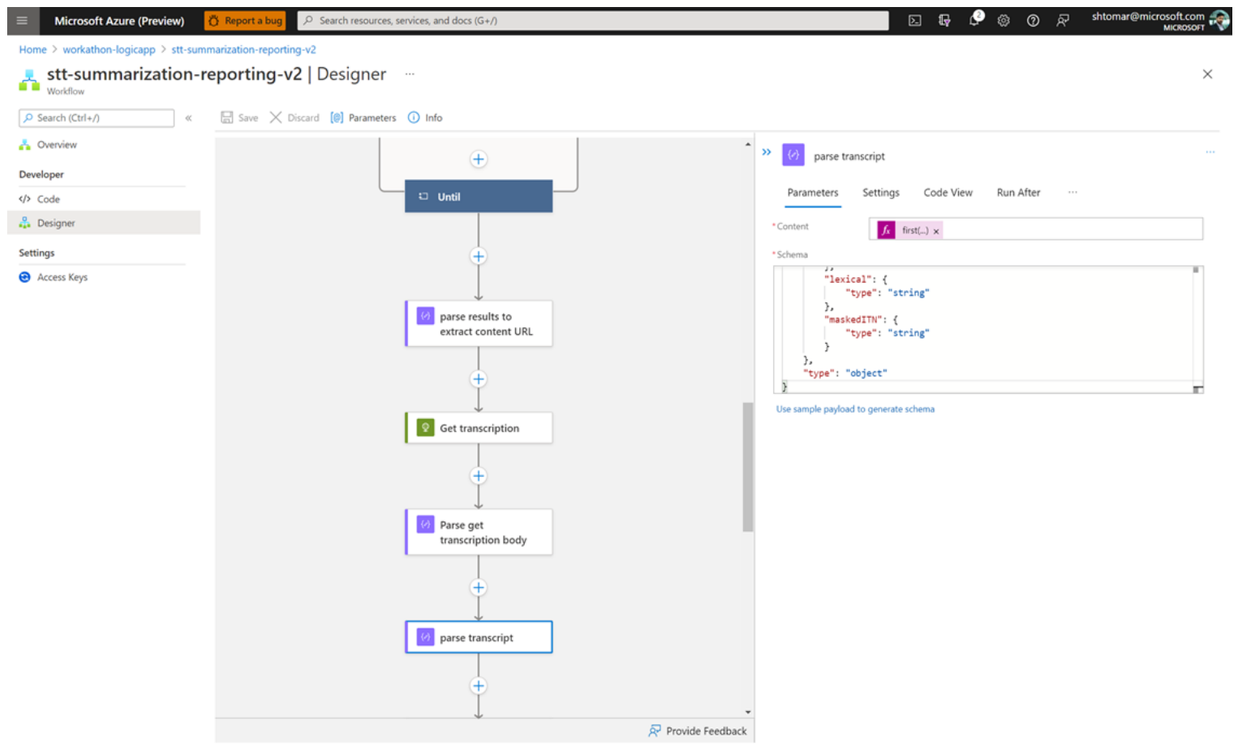


Parse transcript step

In this step, we will be extracting the *'combinedRecognizedPhrases'* as received from the final get call. Despite containing a single JSON, the value for the key *'combinedRecognizedPhrases'* is an array, because of which Logic Apps flow automatically creates a 'for loop' when trying to use the main transcription. Hence, we will parse this value as a standalone JSON, to avoid the unnecessary looping. [Refer the Postman screenshot above to view the value for *'combinedRecognizedPhrases'* in body output.]

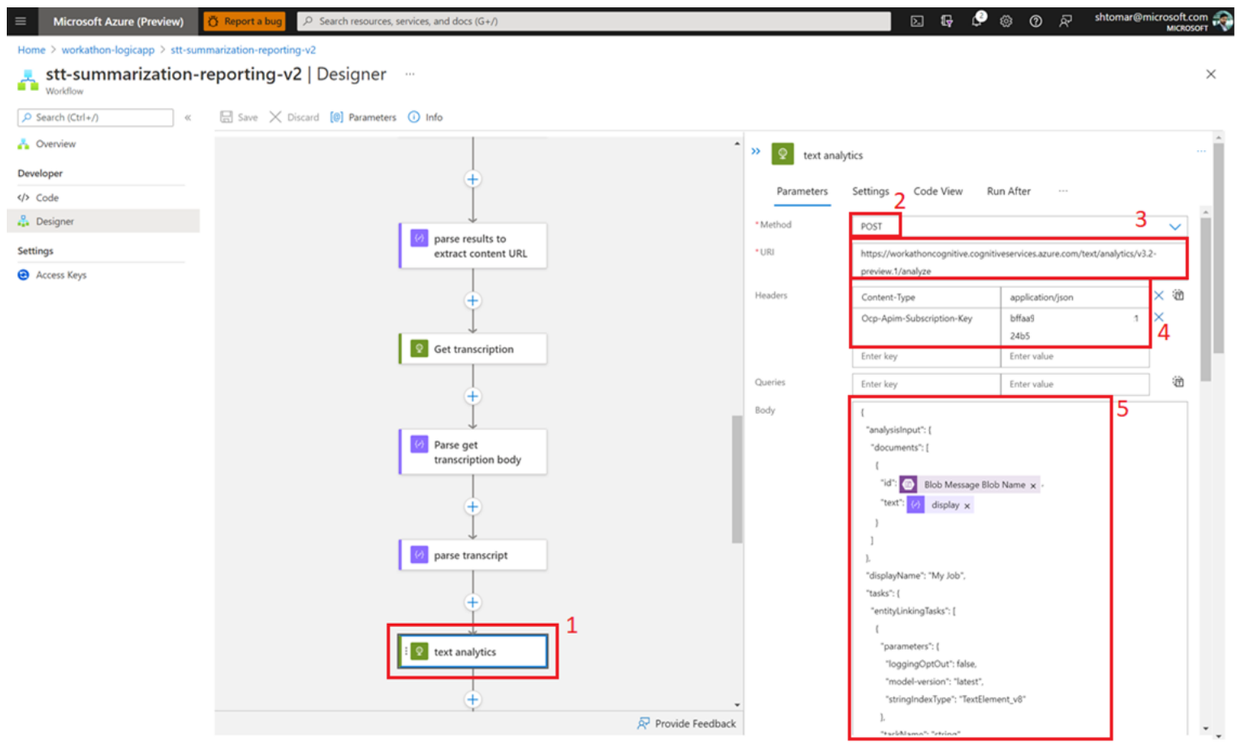
1. Create a new step, search for parse, select 'Parse JSON' module from built-in actions and rename the step to 'Parse transcript'
2. Content : @first(body('Parse_get_transcription_body')?['combinedRecognizedPhrases'])
3. Select 'use sample payload to generate schema' and paste the following payload :

{
"channel": 0,
"lexical": "any domain today business domain beat marketing lead sales beat you know workforce management or finance or operations of services everywhere people want to analyze the date",
"itn": "any domain today business domain beat marketing lead sales beat you know workforce management or finance or operations of services everywhere people want to analyze the date",
"maskedITN": "",
"display": "Any domain today business domain beat marketing lead sales beat. You know workforce management or finance or operations of services everywhere people want to analyze the date."
}



This will help us in retrieving the final cleaned transcription.

We will now pass this output to Text Analytics API as input.



Text Analytics API

Once the text is extracted from speech, let's use the Async text analytics API (/analyze) to extract summary and other features like key phrases, sentiments etc.

Since this is also an asynchronous multi-API call (1 POST call, 1 GET call), we will again add loops and conditions to make sure we extract the results only after the call has been executed completely.

In case you are not familiar with the flow of Batch Text Analytics API, make sure to try out the text Analytics workshop prior to building this flow further.

API CALL 1 : text analytics

1. Create a new step, search for HTTP, select HTTP module from built-in actions and rename the step to 'text analytics'

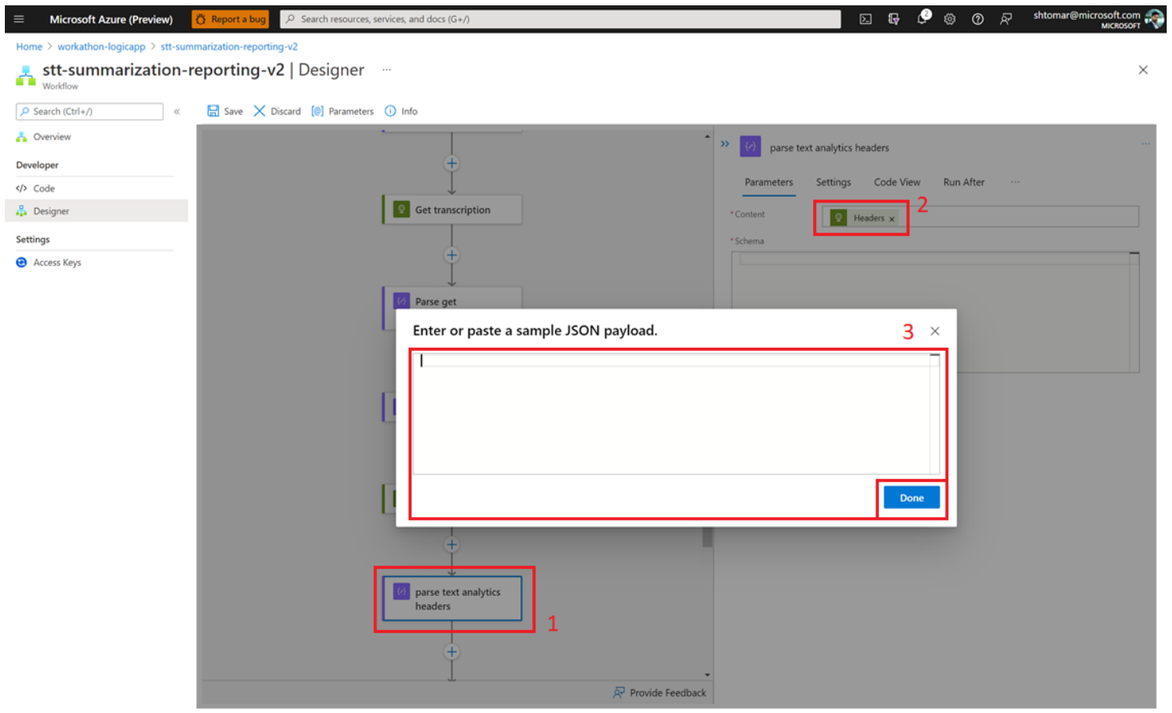
2. Method : POST

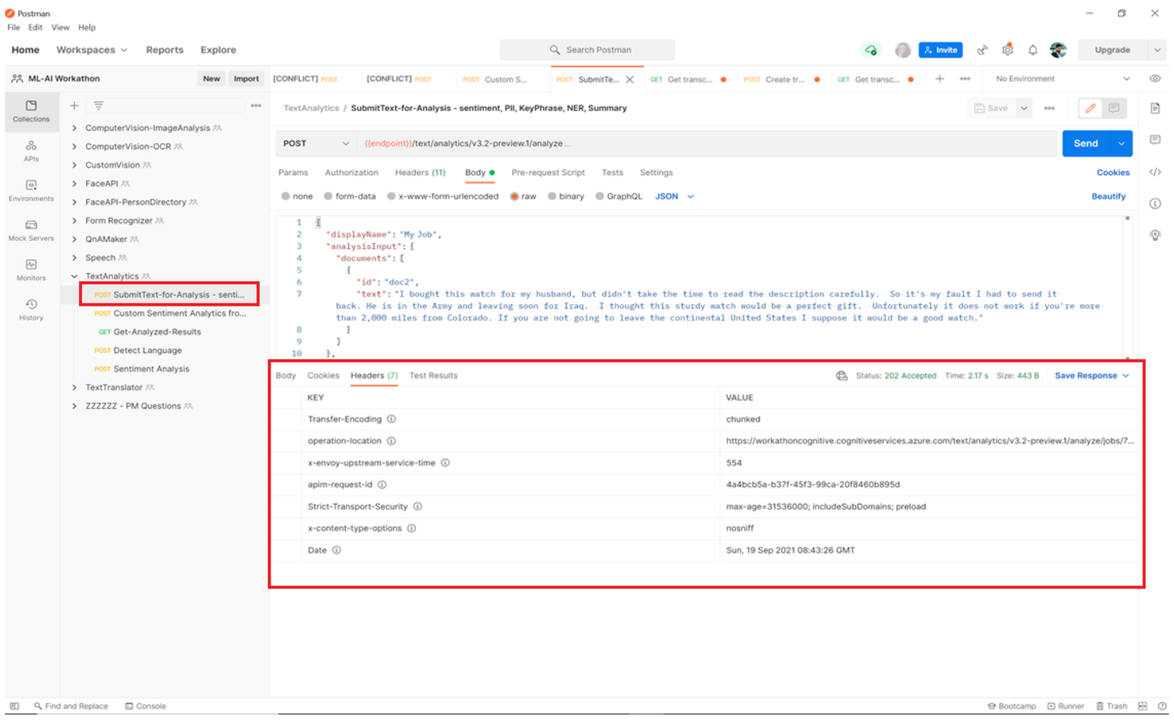
3. URI : https://workathoncognitive.cognitiveservices.azure.com/text/analytics/v3.2-preview.1/analyze
   Replace workathoncognitive with the name of your cognitive service resource. This is URL for the 1st POST call in text analytics (/analyze API)

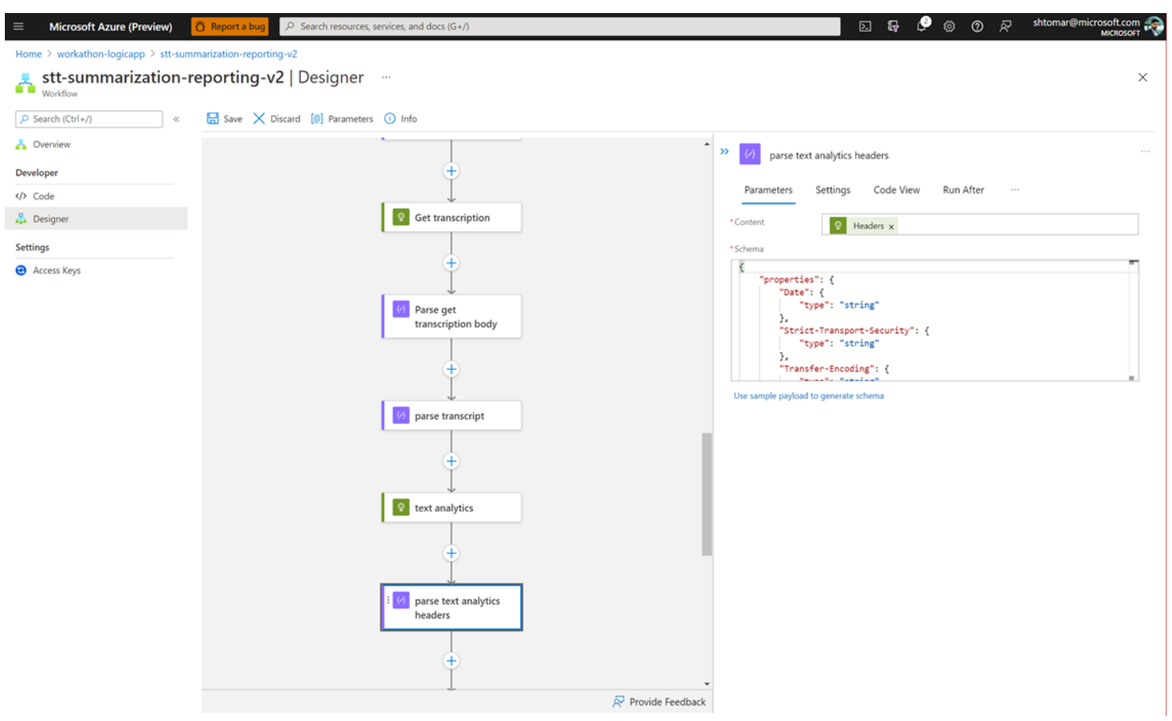
4. Headers :
   Ocp-Apim-Subscription-Key : <<Paste the API key for your cognitive service resource>>
   Content-Type : application/json

5. Body :
   The Body is the same as the one we used in Postman while performing Text Analytics workshop. The difference being that we will be passing the id and text dynamically.
   "id" : *<< dynamically pick 'blob message blob name' from the trigger output 'When a blob is Added or Modified in Azure Storage' >>*
   "text" : *<< dynamically pick 'display' output from 'parse transcript' step'* >> [make sure you don't pick display output from 'Parse get transcription body' output step]

Alternatively, you can paste the content from the 'text analytics api call body' file. Make sure to confirm the dynamic variables have been picked up correctly.







Parse text analytics header step

In this step, we will be parsing the 'header' content received from the previous POST call, to extract the 'operation-location' which contains the results for Text Analytics and use it to make next GET API Call.

You can't leverage the individual parameters unless you the Parse the 'Headers' output as a JSON, for which you require the JSON schema definition.

1. Create a new step, search for parse, select 'Parse JSON' module from built-in actions and rename step to 'parse text analytics headers'

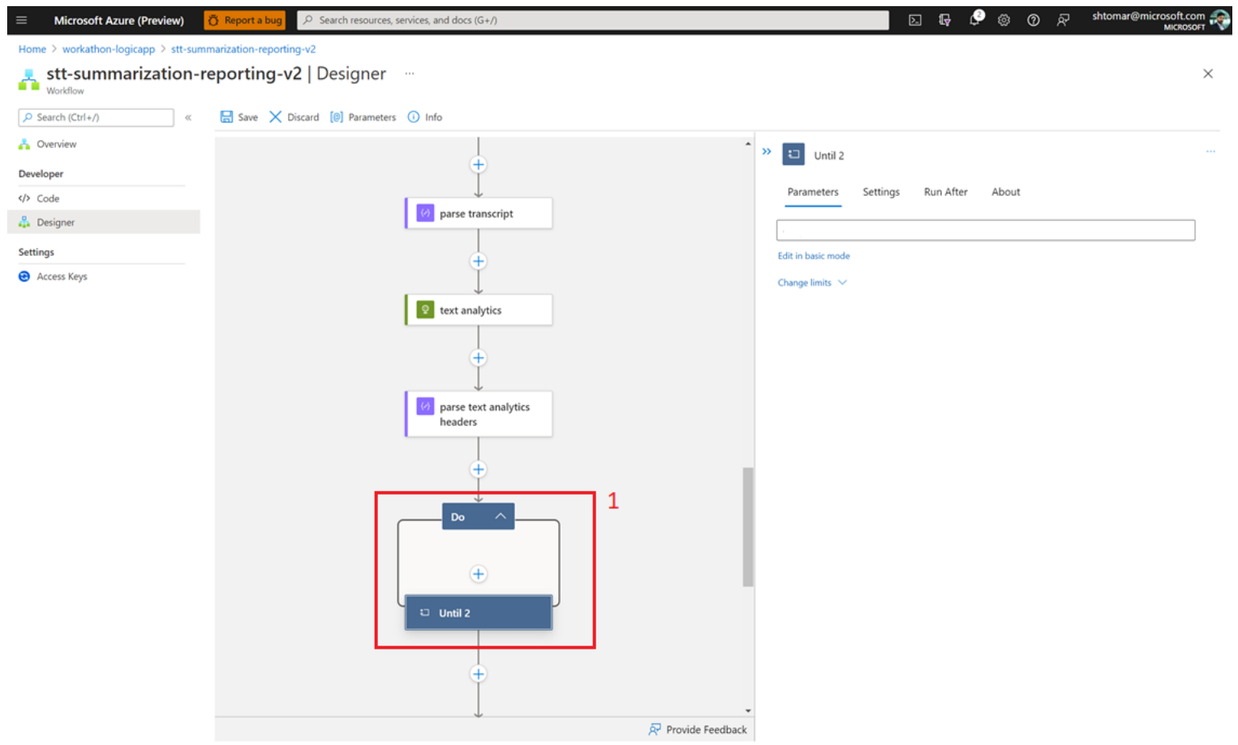
2. Content : << Dynamically, select 'Headers' as part of 'text analytics' output >> [By now, we have a couple of API calls giving the header output. Make sure to select the right header]

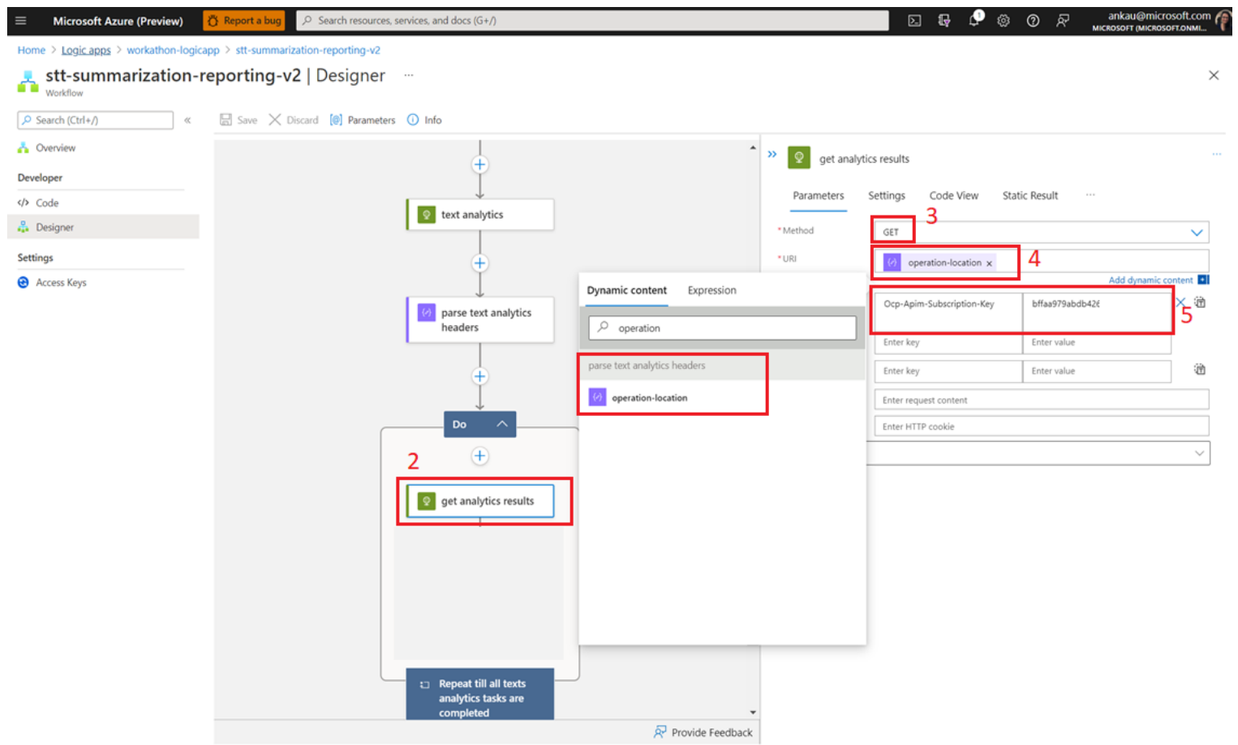
3. Create the JSON payload using the Headers output as obtained by executing the call in Postman, like the ones we shared earlier. This will help you in learning how to create one, in case you decide to create your own custom flows and call other APIs!
   Select 'use sample payload to generate schema' and paste the JSON payload you generate.

Alternatively, you can directly paste the below schema into the schema input –

```
{
  "properties": {
    "Date": {
      "type": "string"
    },
    "Strict-Transport-Security": {
      "type": "string"
    },
    "Transfer-Encoding": {
      "type": "string"
    },
    "apim-request-id": {
      "type": "string"
    },
    "operation-location": {
      "type": "string"
    },
    "x-content-type-options": {
      "type": "string"
    },
    "x-envoy-upstream-service-time": {
      "type": "string"
    }
  },
  "type": "object"
}
```

[Do note we have directly provided you the 'schema'. This is not the JSON payload. Paste this in 'Schema' input, not in JSON payload sample input.]





Add Until step

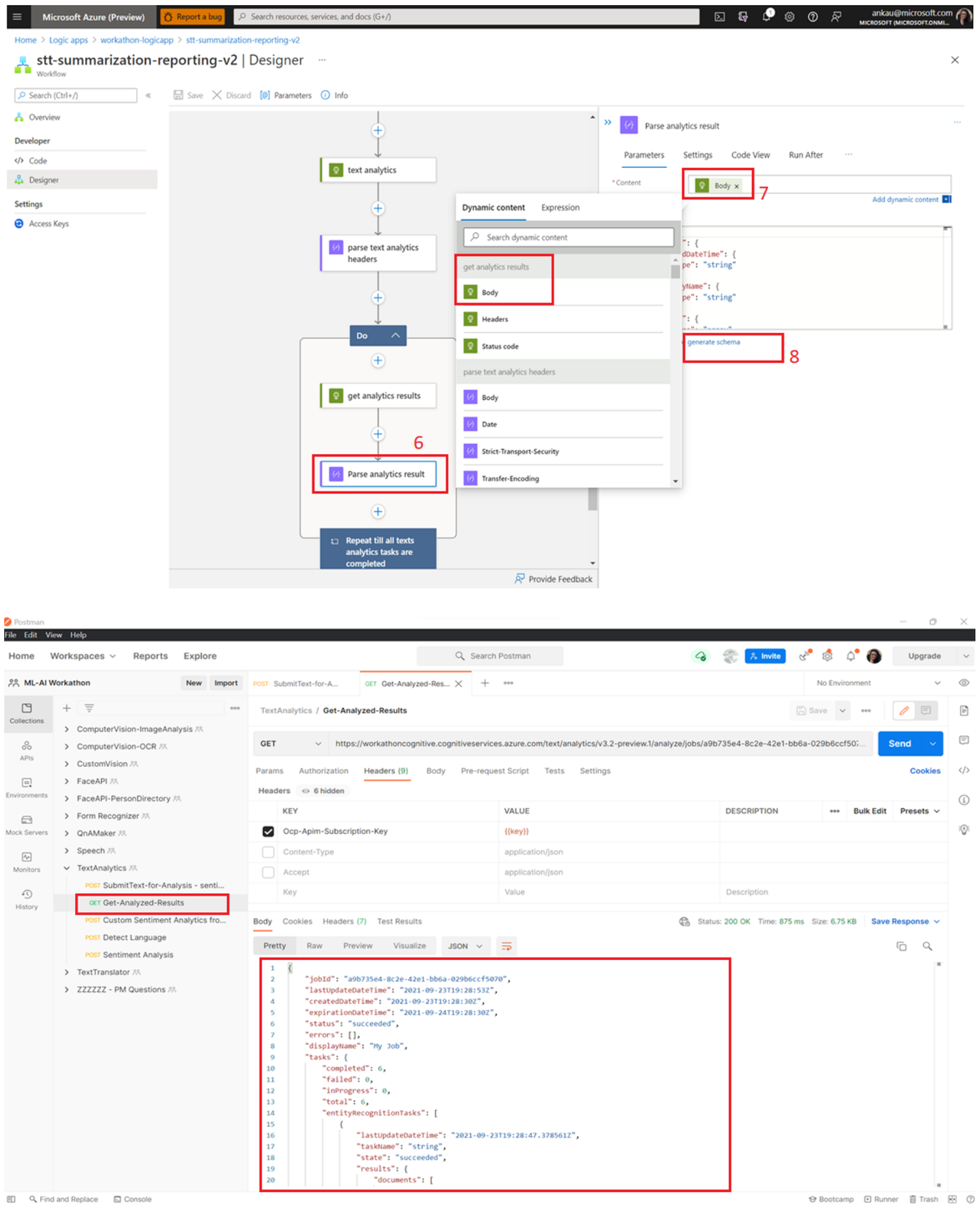
The next GET API call that we will hit, always returns Status 200 OK on successful execution. However, since this is an asynchronous call, the GET call output body keeps updating and changing till all 6 text analytics tasks are completed. Each task associated with 1 activity, namely : linked entity, key phrases, named entity extraction, sentiment analytics, summarization, PII entity recognition.

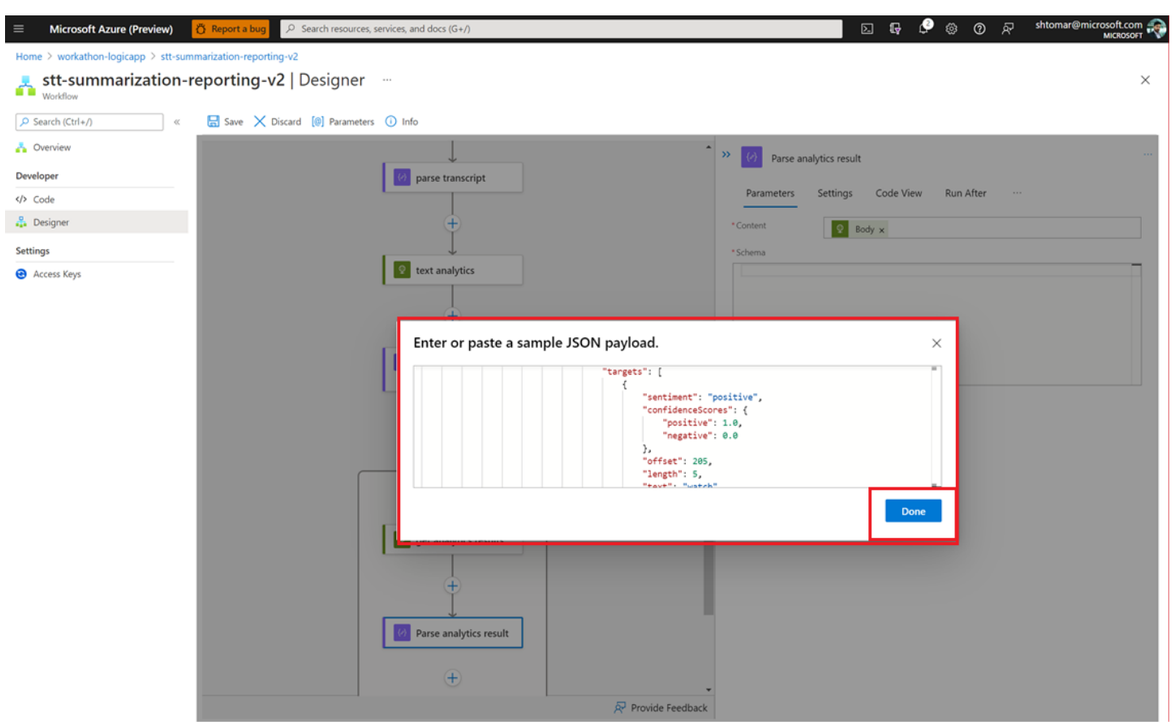
Hence, we will add an 'until' loop that will repeatedly make the GET API call until we get the intended output Body.

1. Create a new step, search for until, select 'Until' module from built-in actions and rename it to 'Repeat till all texts analytics tasks are completed'

API CALL 2 : Get analytics results

2. Click on + within do until loop, add an HTTP Module and rename it to 'get analytics results'
3. Method : GET
4. URI : <<Dynamically pick 'operation-location' from 'parse text analytics headers' output>>
5. Headers :
   Ocp-Apim-Subscription-Key : <<Paste the API key for your cognitive service resource>>





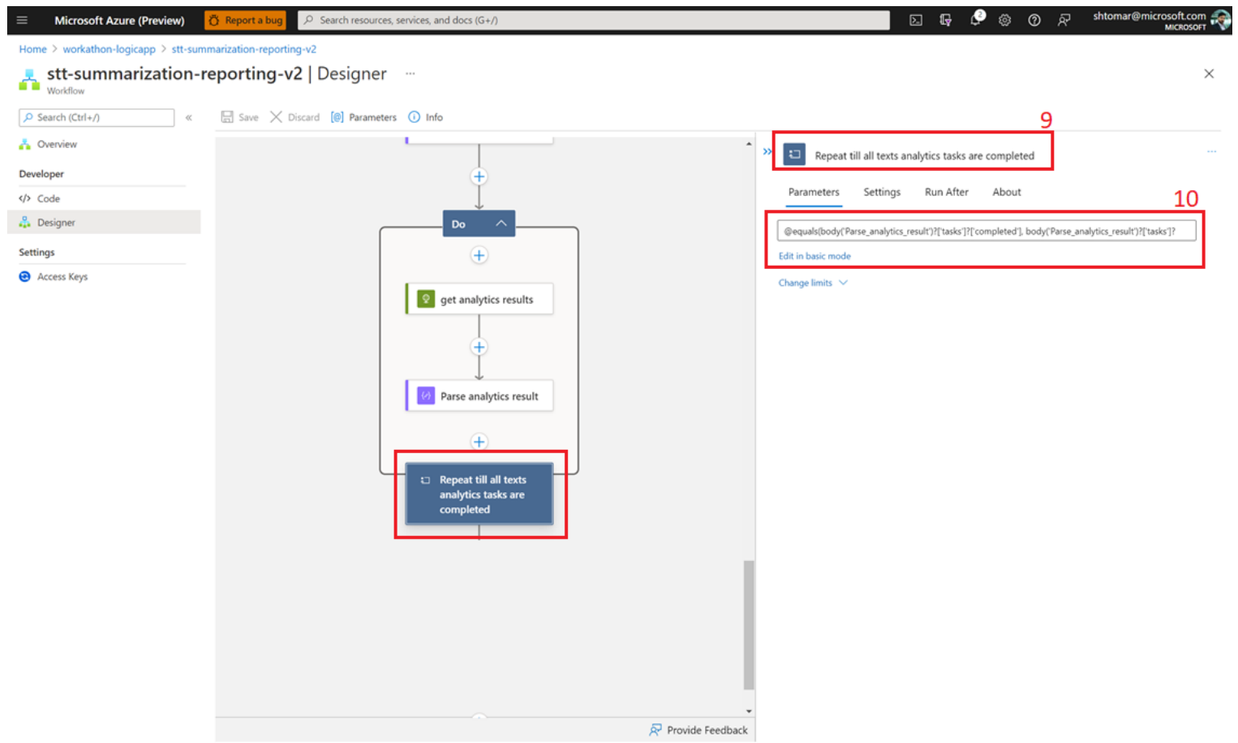
Parse analytics result step

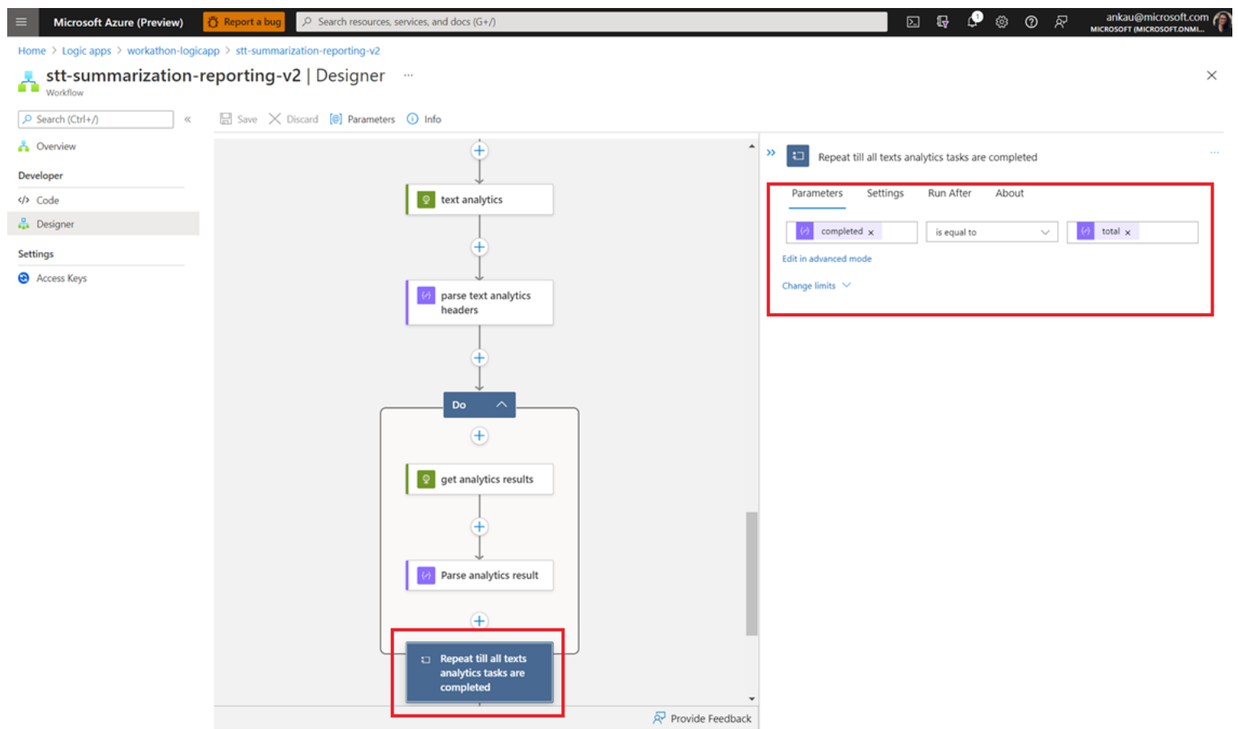
In this step, we will be parsing the 'Body' content received from the previous 'GET analytics result' API call, to extract the final text analytics results.

6. Create a new step, search for parse and select 'Parse JSON' module from built-in actions
7. Content : *<< Dynamically, select 'Body' as part of 'GET analytics result' step. >>* [Make sure you don't select the body output from any of the other API calls]
8. Select 'use sample payload to generate schema'
9. Fetch the output for the body after executing GET analytics result API call in Postman and paste that as sample JSON payload.

Alternatively, you can copy the schema from the 'parse analytics result schema' file and paste it in the schema input directly

[Do note we have directly provided you the 'schema'. This is not the JSON payload. Paste this in 'Schema' input, not in JSON payload sample input.]





Add condition for Until step

1. Click on Until module and click on Edit in Advanced Mode
2. Add the following condition :

@equals(body('Parse_analytics_result')?['tasks']?['completed'], body('Parse_analytics_result')?['tasks']?['total'])

Alternatively, you can create this expression from the GUI of “Add Dynamic Content” [See screenshot below]

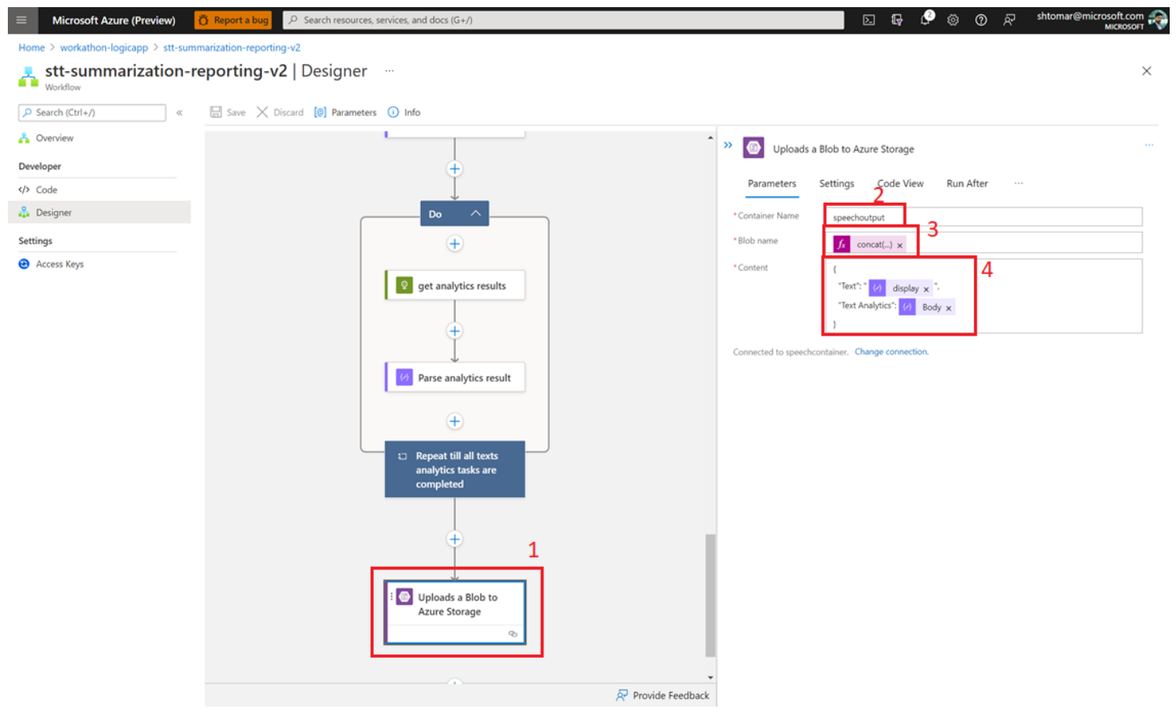
Note : The above expression will only work if you followed the naming convention the same as ours.

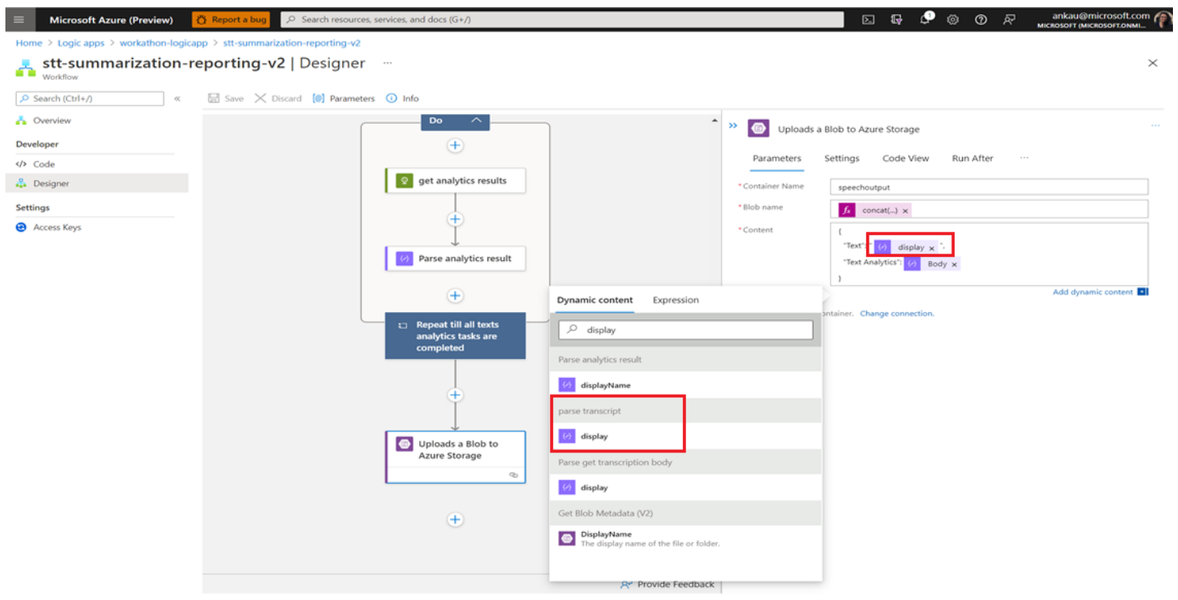
Significance :
The GET call keeps returning partial Body until all required text analytics tasks are complete. The number of tasks completed can be viewed in the ‘completed’ and ‘total task’ values, as shown in the following snippet from GET call body :

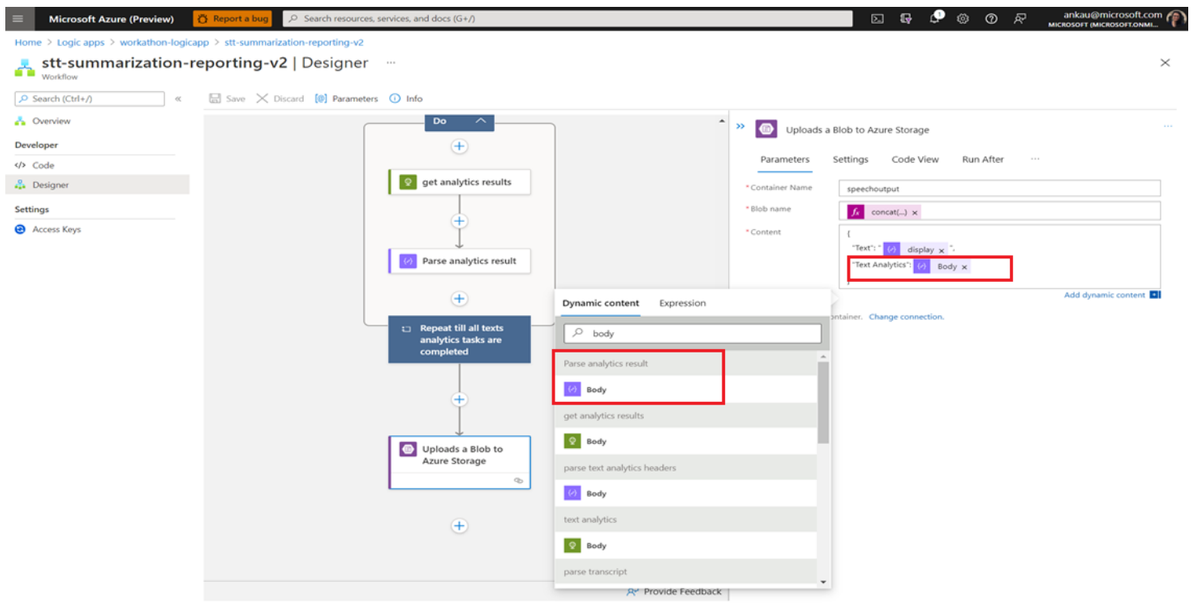

```
{ . . .
   "completed": 6,
  "failed": 0,
  "inProgress": 0,
  "total": 6,
. . .
}
```


In the until condition, we are checking for the number of tasks completed and comparing it to the total tasks. The loop will keep executing until the value of completed tasks equals the value of total tasks.
We will exit the loop once the 2 values are equal.

Select change limits and add the following values :
Count : 5000 [You can increase this is you have longer files]
Timeout : PT8H [This refers to the maximum time the loop will run for]







'Uploads a Blob to Azure Storage' step

As the final step, we will now upload the complete text generated from Speech-to-text API and the Text Analytics results as a JSON to speechoutput container in the storage account.

Follow the steps as highlighted to add the Blob module :

8. Click on Choose an operation, search for Blob, select 'Uploads a Blob to Azure Storage' module from the built-in actions. Keep the module selected. This will show a message like 'Connected to speechcontainer' at the bottom of right tab

9. Container Name : speechoutput [This is the name of the 2nd container we created early on in the workshop. Change the name in case you named it differently.

10. Blob Name :
    @concat(triggerOutputs()?['body']?['name'],body('Parse_analytics_result')?['createdDateTime'],'.json')

    Here, we are concatenating the name of the blob from the Trigger output and the 'createdDateTime' from 'parse analytics result' output to give the blob a unique name.

11. Content :
    {
    "Text": " @{body('parse_transcript')?['display']}",
    "Text Analytics": @{body('Parse_analytics_result')}
    }

Note : The above expression will only work if you followed the naming convention the same as ours.

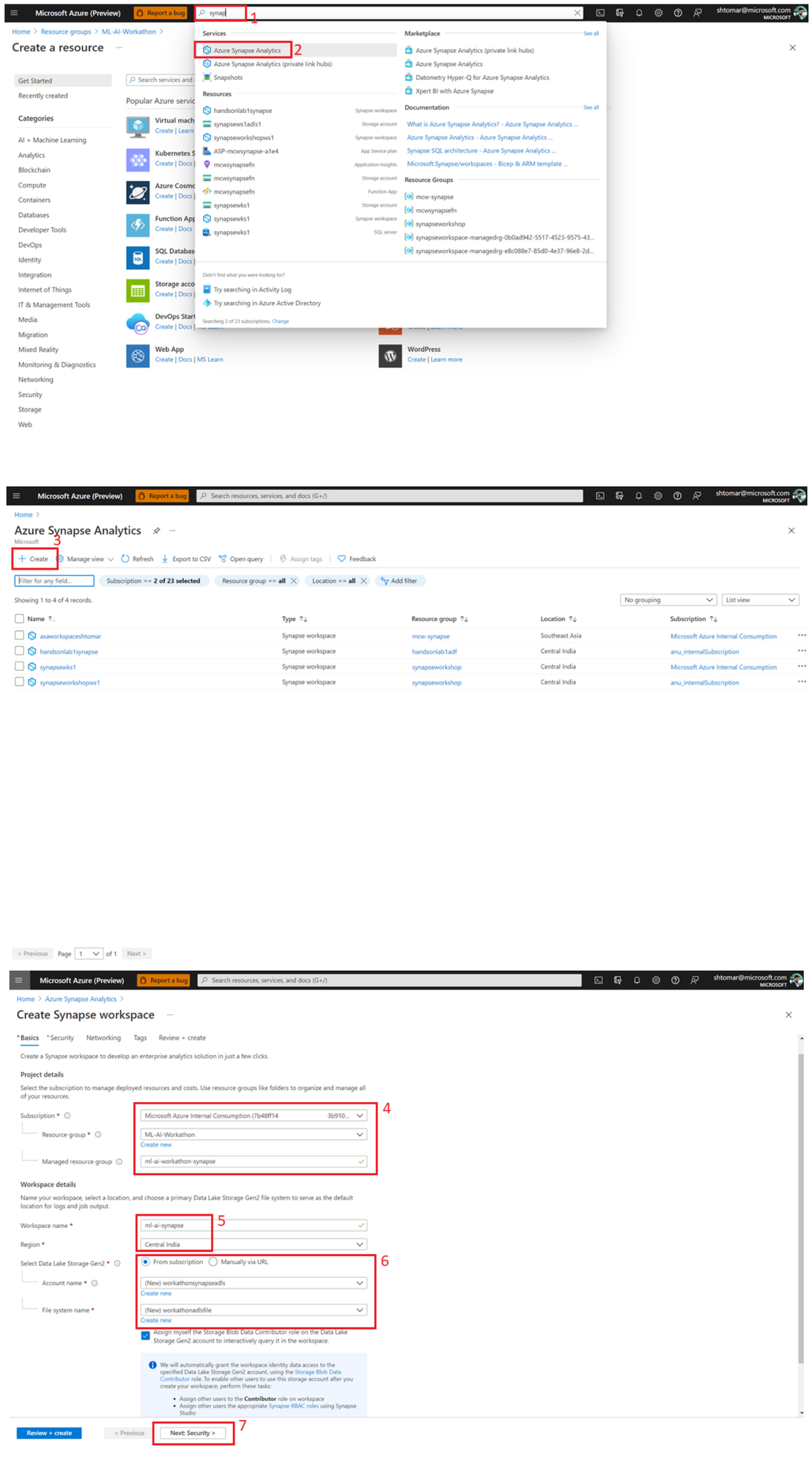
For Content, we are creating a JSON which contains 2 main keys :

a. "Text" : This takes the value of 'display' output from 'parse transcript' step [This is the main transcript generated from STT call]
b. "Text Analytics" : This takes the value of the 'body' output from 'parse analytics result' step.

You can create this as shown in the screenshot or paste the above code for content.

Alternatively, you can create the above flow simply by copy-pasting the code from the ‘speech summarization’ file in the code section of the flow. This will add all the modules and values in them. However, the workflow will not run for you unless you change the configuration and connection parameters according to your resources, such as OneDrive, Cognitive Services, Blob Storage etc. Make sure to run through the entire workshop and make the changes at each step as required.

Data Factory Pipeline



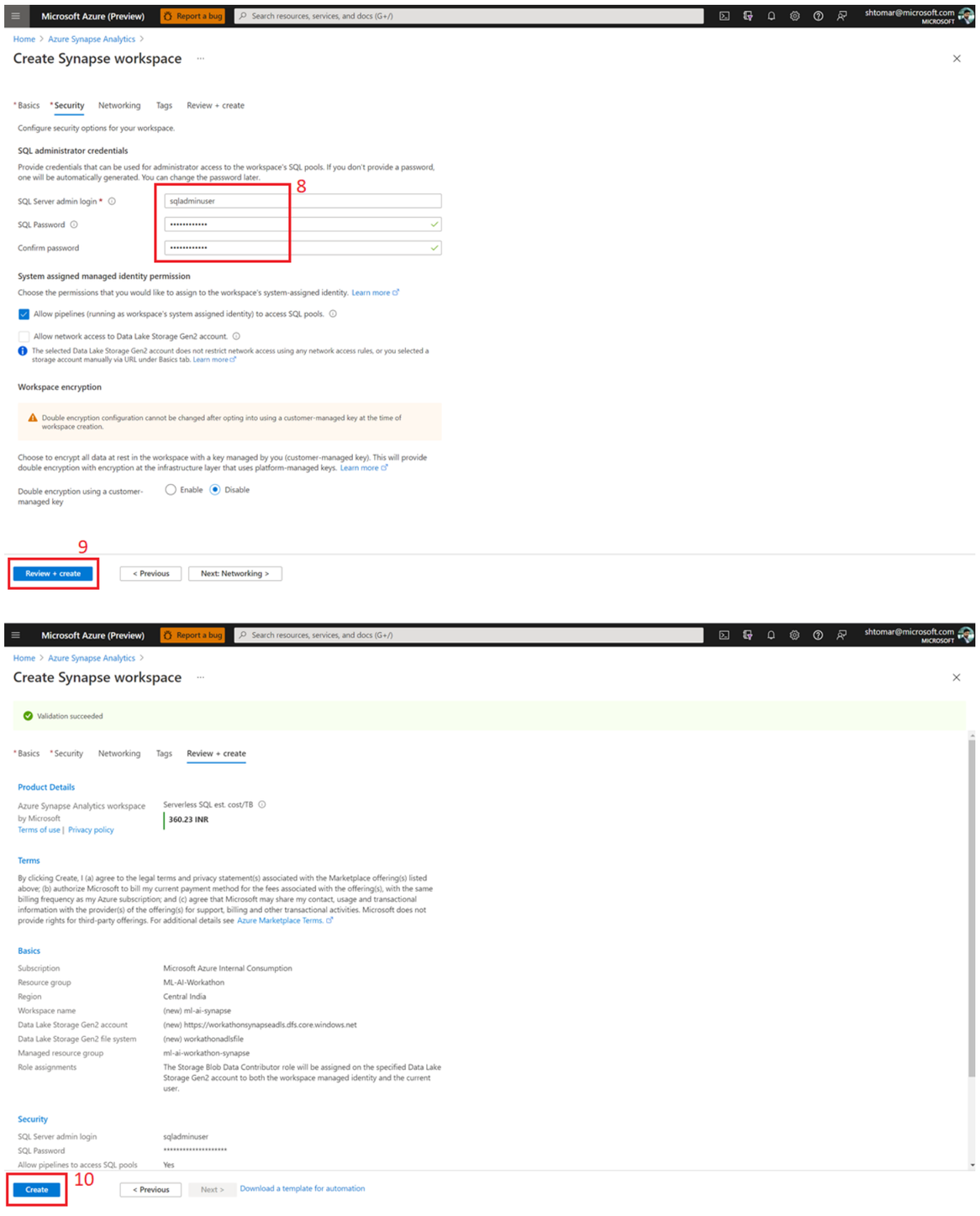
Create Synapse Analytics resource in Azure Portal

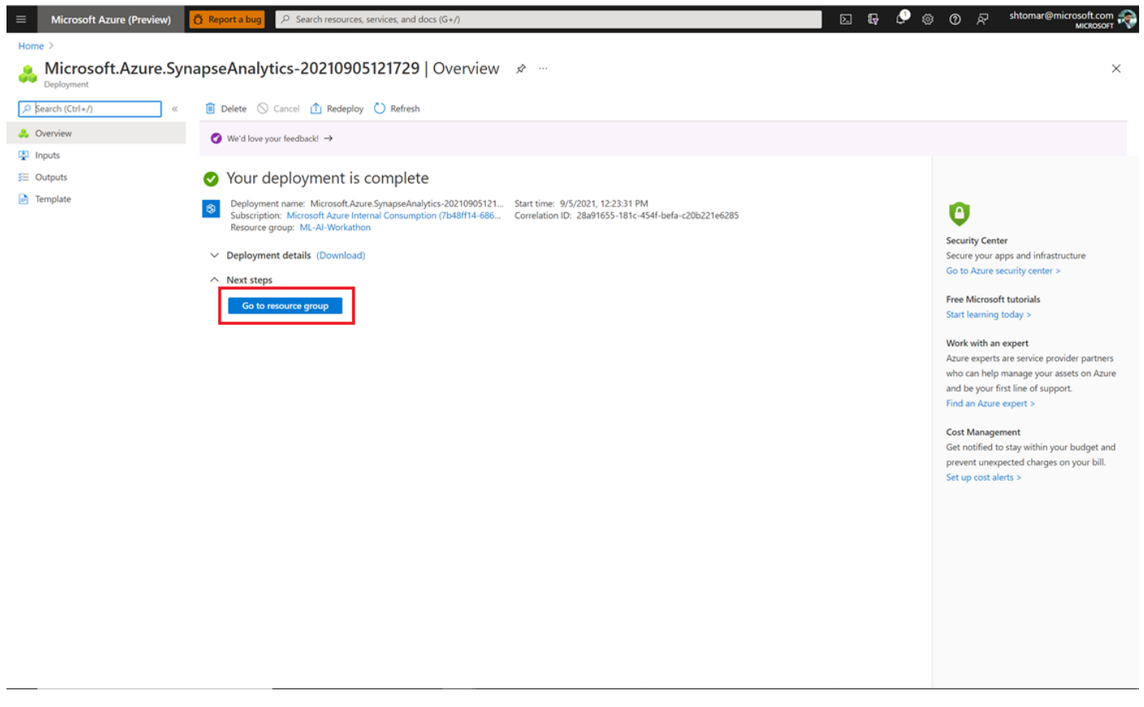
This will be used to build copy activities using pipelines, to copy data from speechoutput blob containers to SQL tables.

1. Search for Synapse in the search bar
2. Select Azure Synapse Analytics resource

3. On Azure Synapse Analytics page, click + Create

4. Enter the subscription and resource group details
5. Provide the workspace name and region
6. For data Lake Gen 2, select ‘From subscription’ and create a new Account and Container as shown
7. Click Next : Security

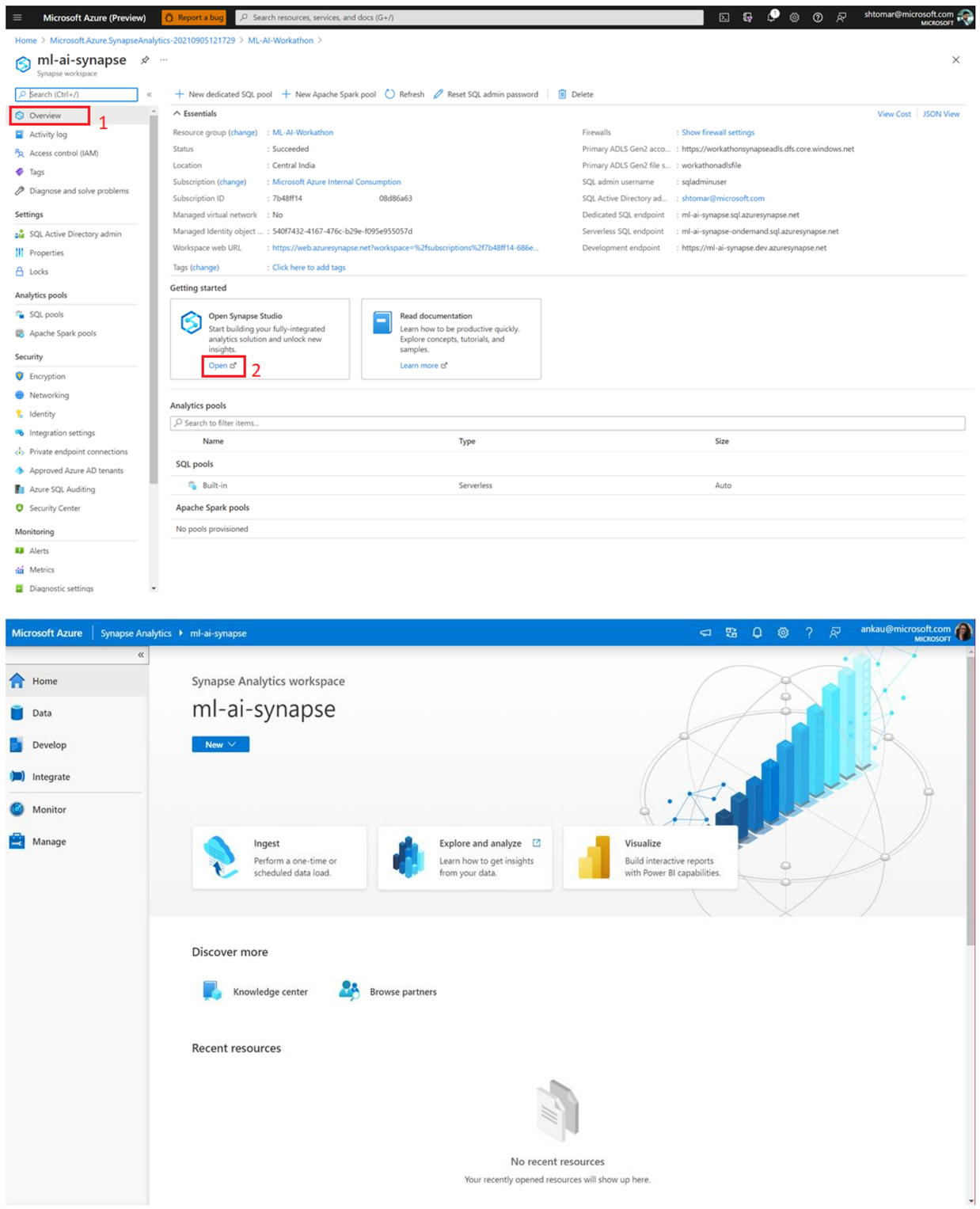




8. Provide a username and password for SQL login
9. Click Review + Create

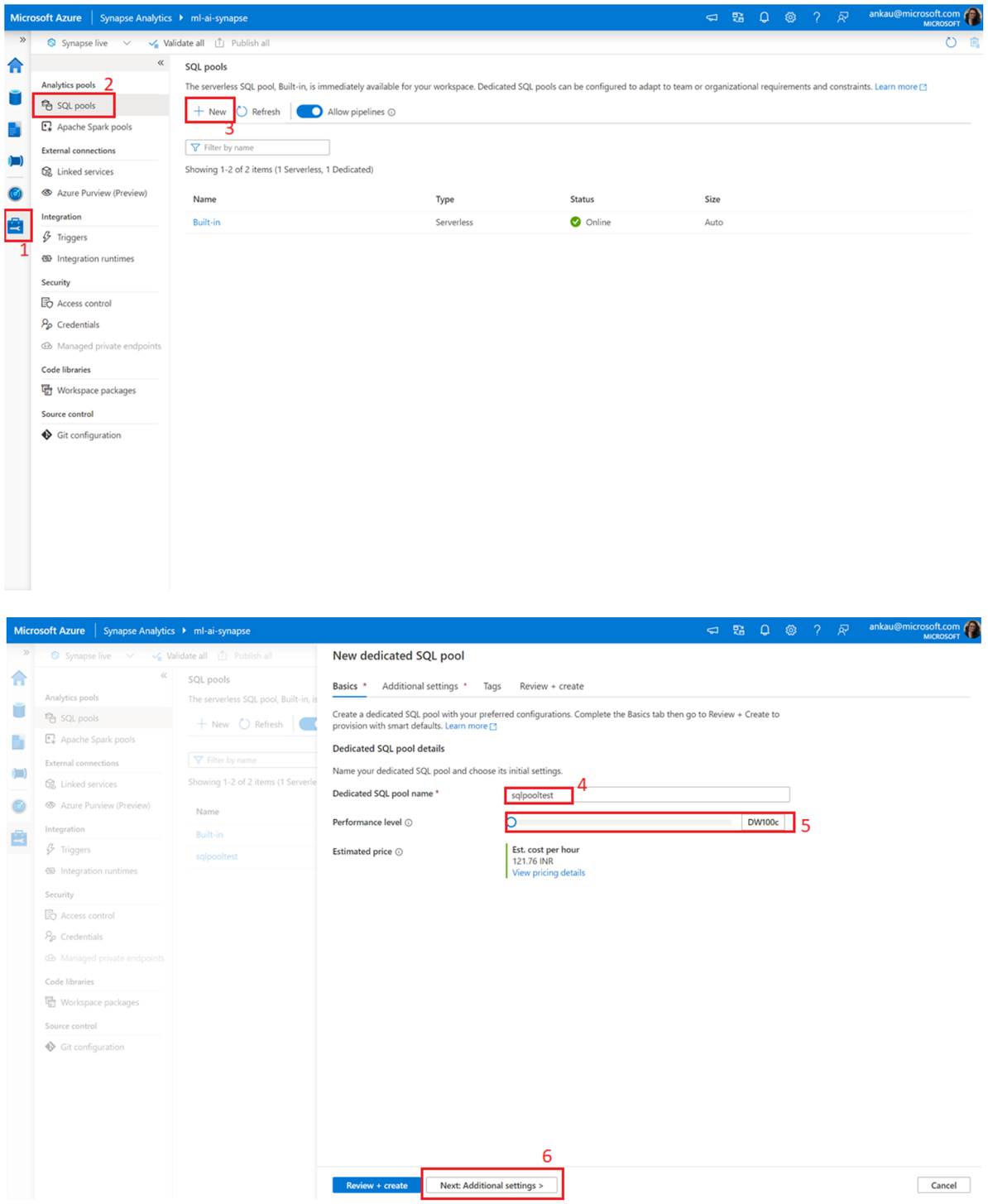
10. Click Create

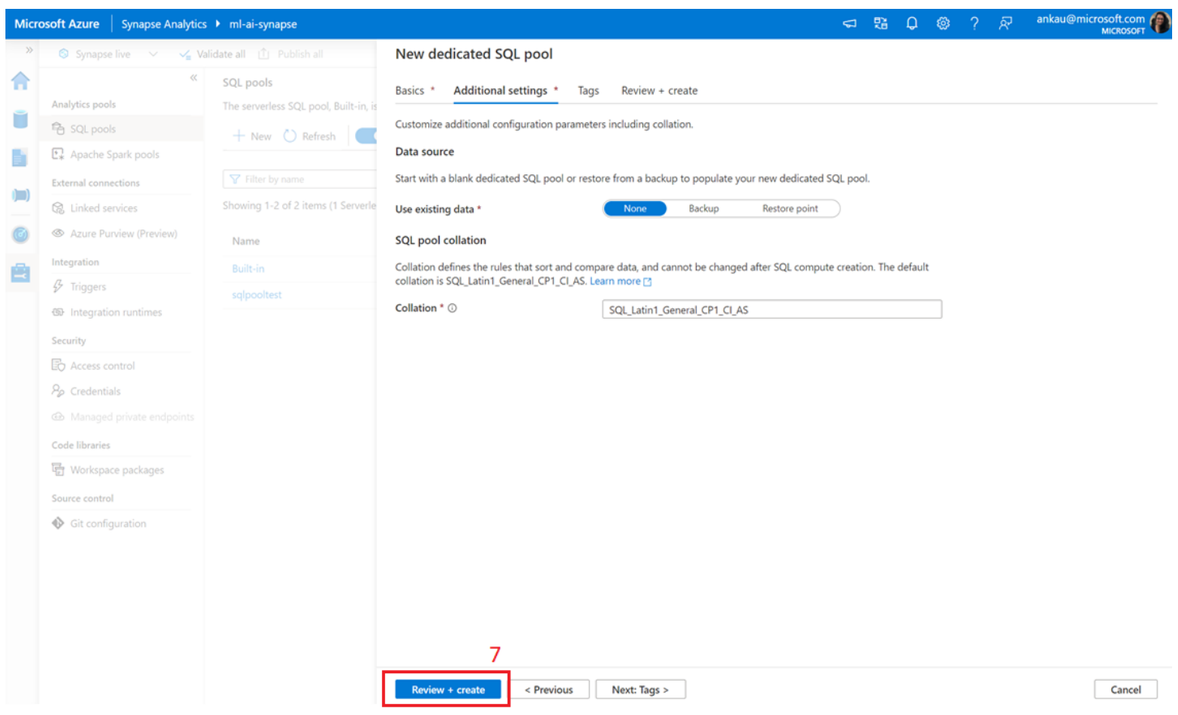
11. The deployment will take a couple of minutes. Once deployed, select ‘Go to resource group’



Open Synapse Studio

1. Select the overview tab in Synapse workspace
2. Click ‘Open’ to open the Synapse Studio





Create dedicated SQL pool

There are 2 major computes that Azure Synapse Analytics provides us –

1. Spark Pool : This is leveraged to perform Big Data & Advanced Analytics queries.
2. SQL Pool : This is leveraged to write SQL queries.

In this workshop, we will be working with SQL Pools.

Azure Synapse offers 2 capabilities as part of SQL pools :

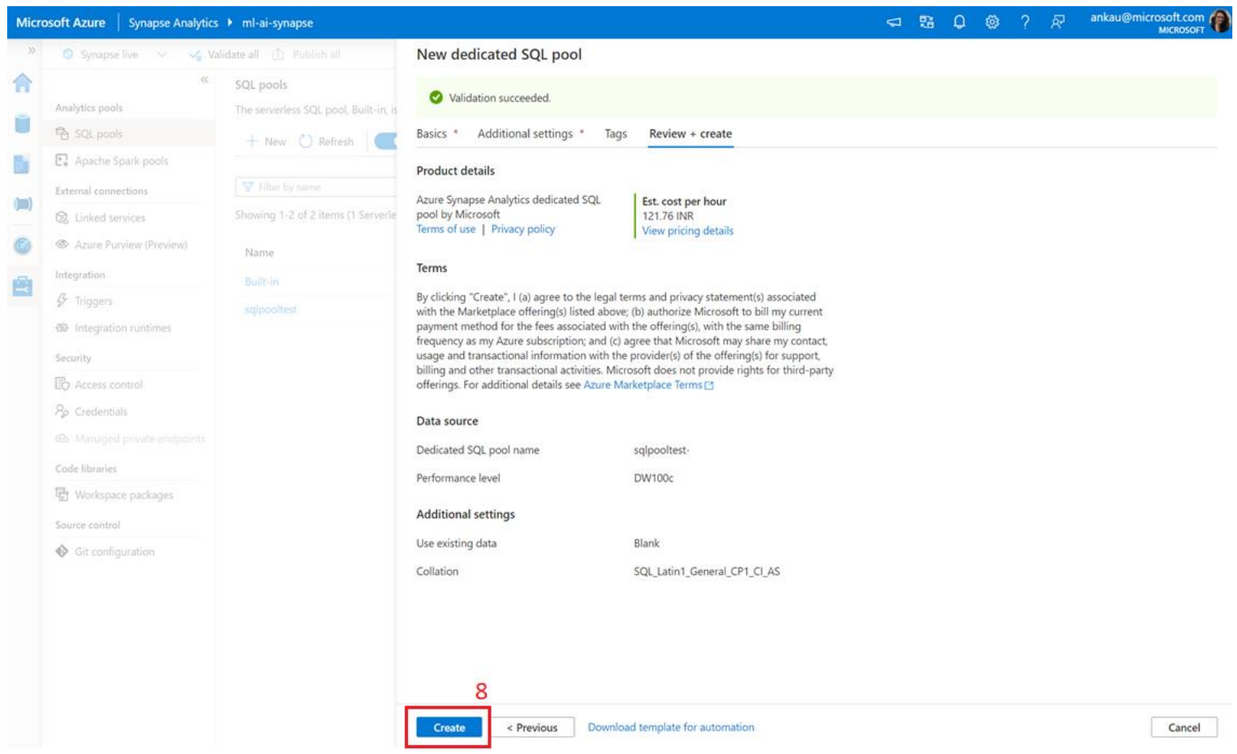
1. **Built-in serverless SQL pool** - This lets you use SQL without having to reserve capacity. Every workspace comes with a pre-configured serverless SQL pool called Built-in. You can't create additional serverless SQL Pools.
2. **Dedicated SQL pool** – These are similar to traditional SQL compute. You have reserve capacity when you create dedicated SQL pools. Workspaces don't come with pre-configured dedicated pools. You can create as many dedicated pools as you want.

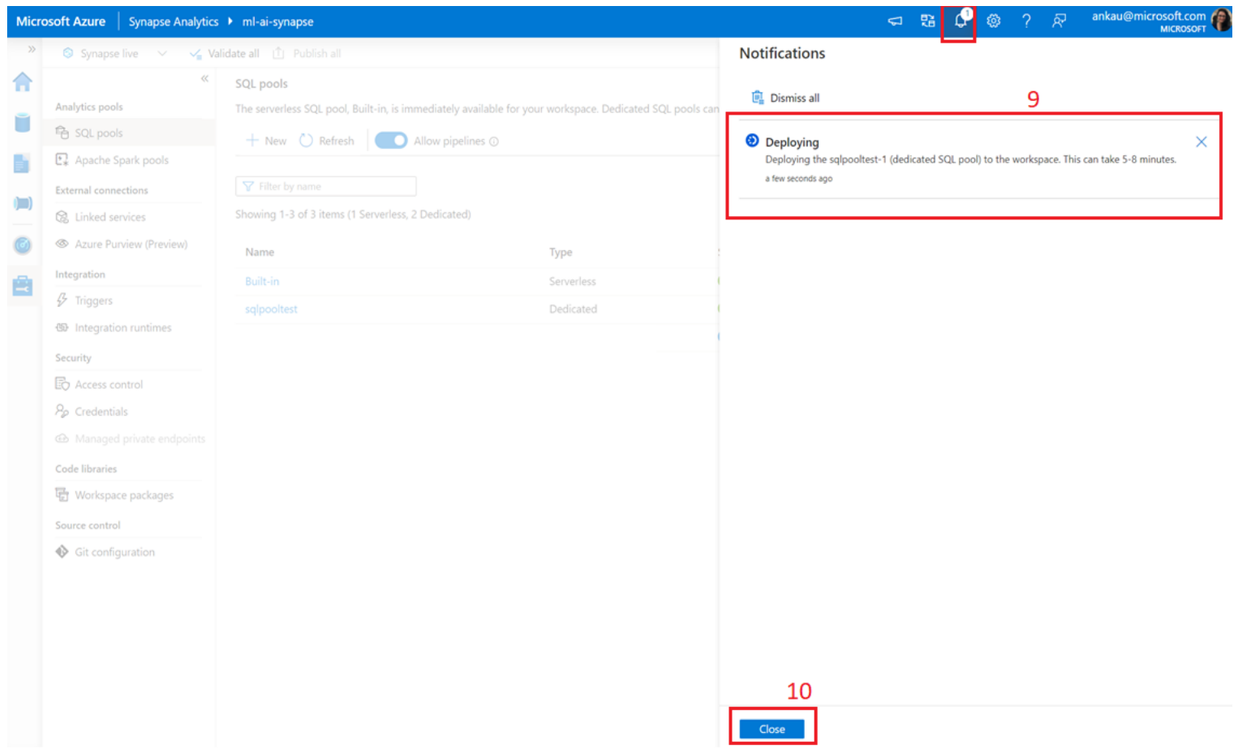
Follow the steps as highlighted to create a dedicated SQL pool :

1. Go to the Manage tab
2. Select SQL Pools
3. Click + New

4. Provide the SQL Pool a name
5. Performance level : DW100c
6. Click Next : Additional Settings

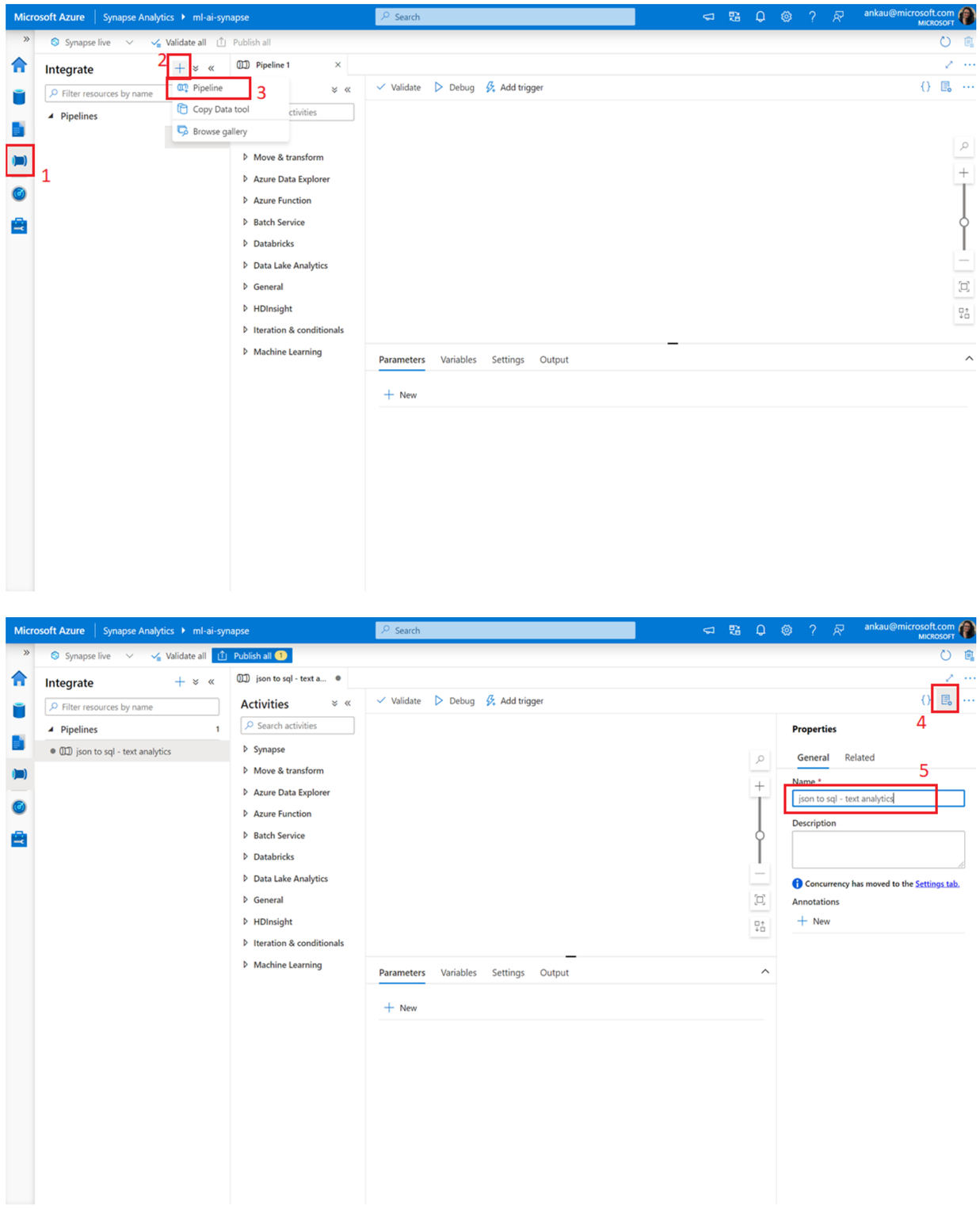
7. Leave the settings as is and click Review + Create





8. Verify the settings and click Create

9. Open the notifications tab to see the deployment status
10. Once the SQL Pool is deployed, click Close.



Create pipeline

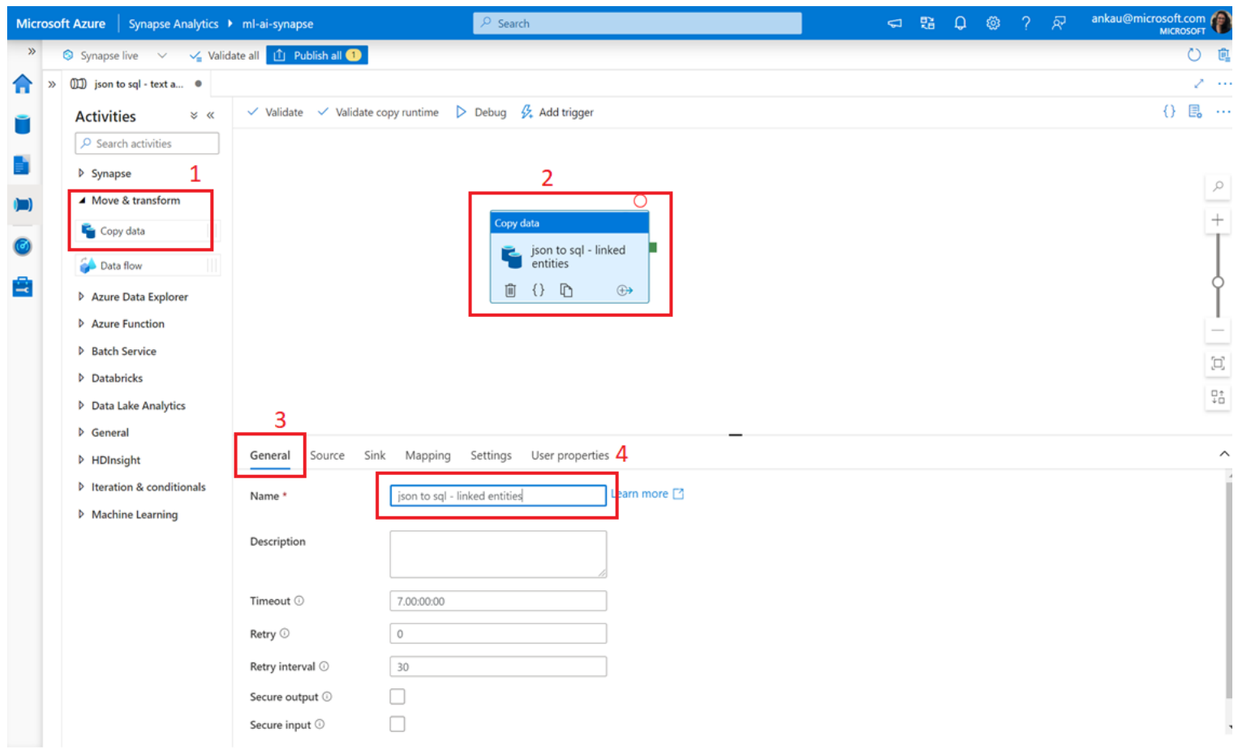
Pipeline is set of activities arranged together in a sequential or parallel manner to perform a certain task. Pipelines can contain activities like Copy, Mapping data flows, running Machine Learning jobs, Running Azure functions and many more.

We will be copying data for 4 Text Analytics tasks (Linked entities, Sentiment analytics, summarization & key entity extraction) from the JSON we saved in speechoutput container to 4 separate SQL tables in dedicated SQL Pools in Synapse.
You can create 4 separate pipelines or create 4 copy activity tasks in the same pipeline.
We will be taking the latter approach and create 1 pipeline with 4 copy activities.

The source for each of the copy activities will be the same (JSON files stored in speechoutput container). However, the destination will be 4 separate SQL tables and we will pick the respective content from each JSON file.

1. Go to the Integrate tab
2. Click +
3. Select Pipeline

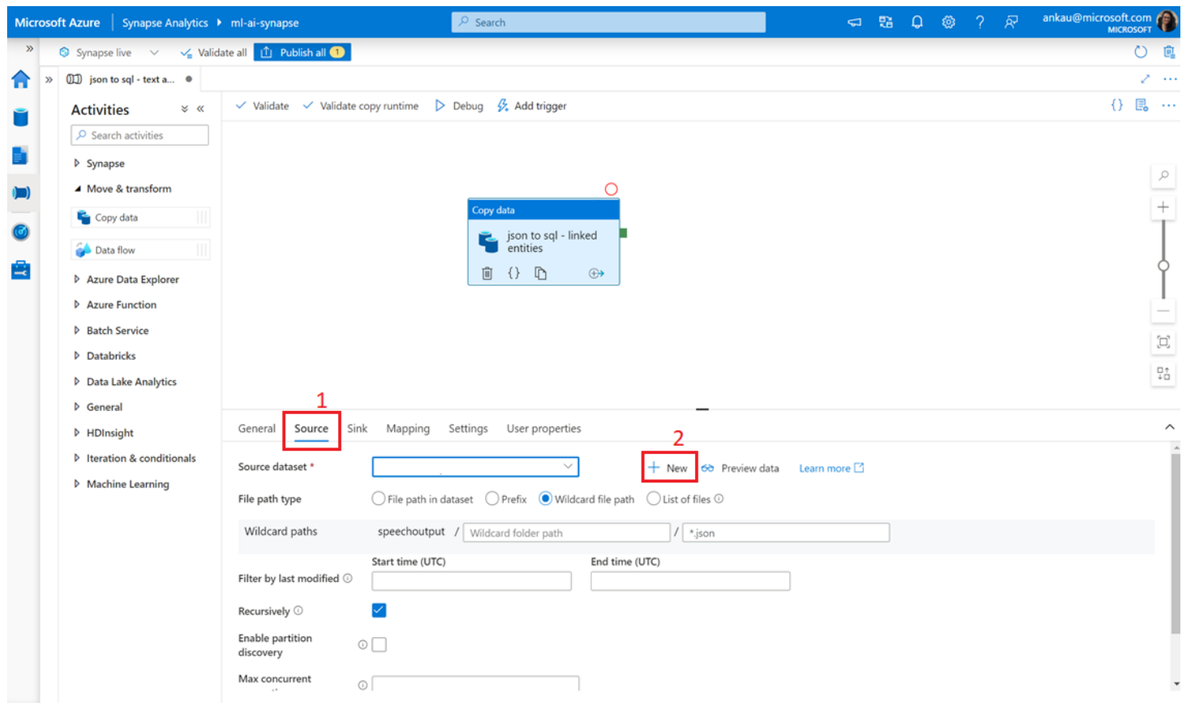
4. Open the settings tab
5. Rename the Pipeline to : json to sql – text analytics

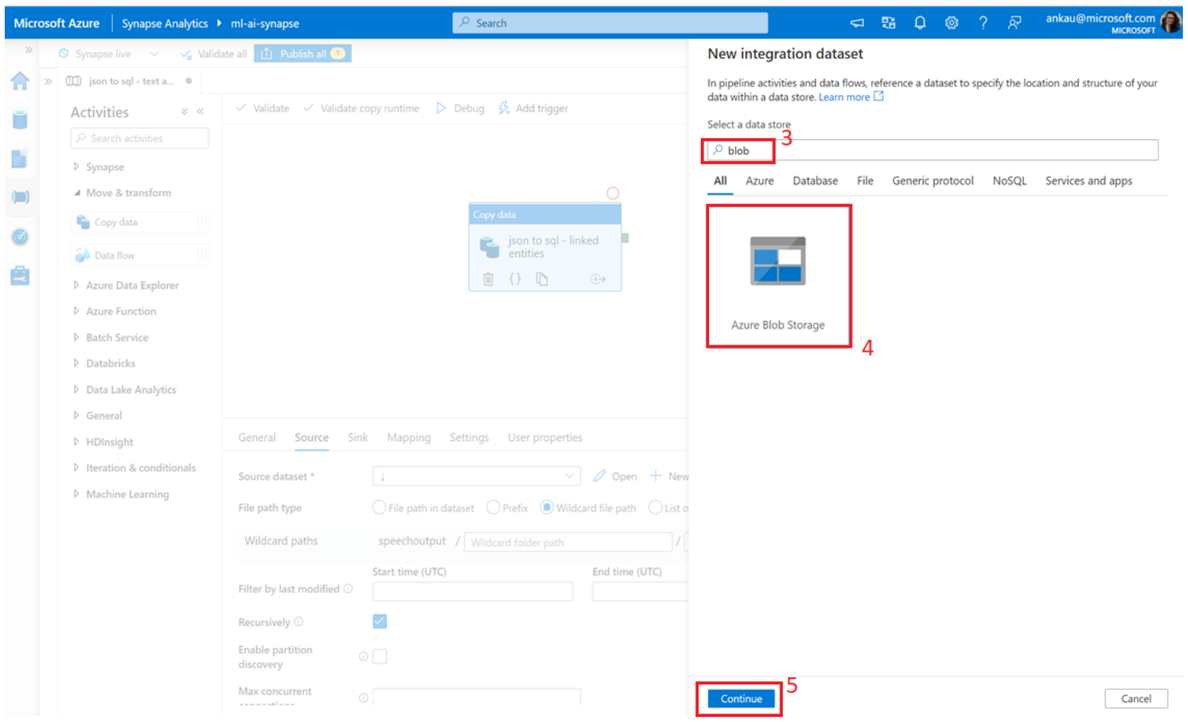


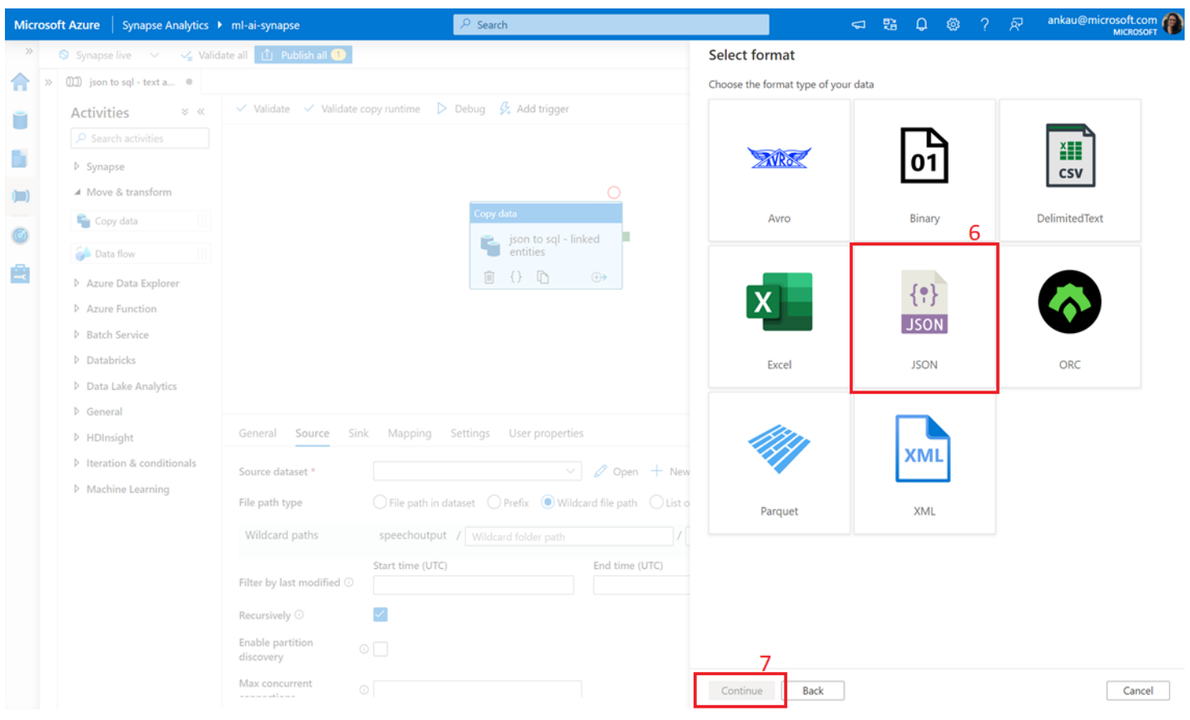
Create Copy Activity for Linked Entities

In this activity, we will be copying values pertaining to Linked Entity task from the complete Text Analytics JSON output to 'linkedentities' table in the SQL Pool.

1. From the Move & Transform tab, drag the Copy Activity module to the canvas
2. Select the copy data module on canvas
3. Go to General Tab
4. Name : json to sql – linked entities





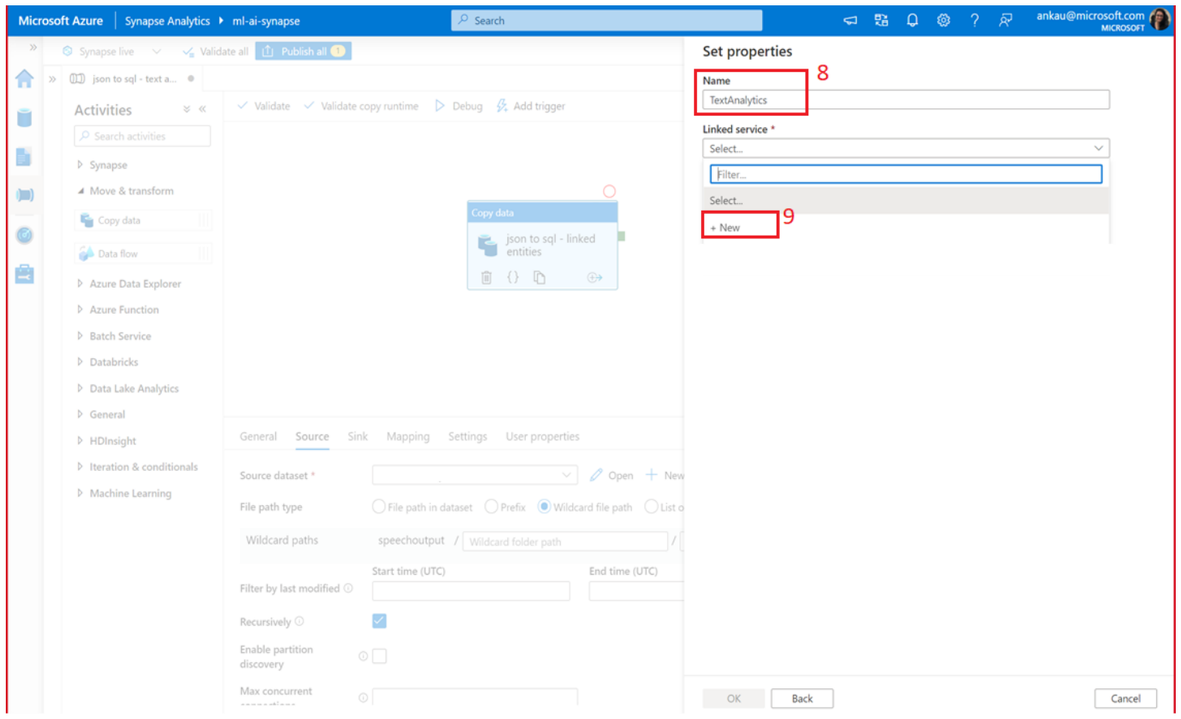


Source settings

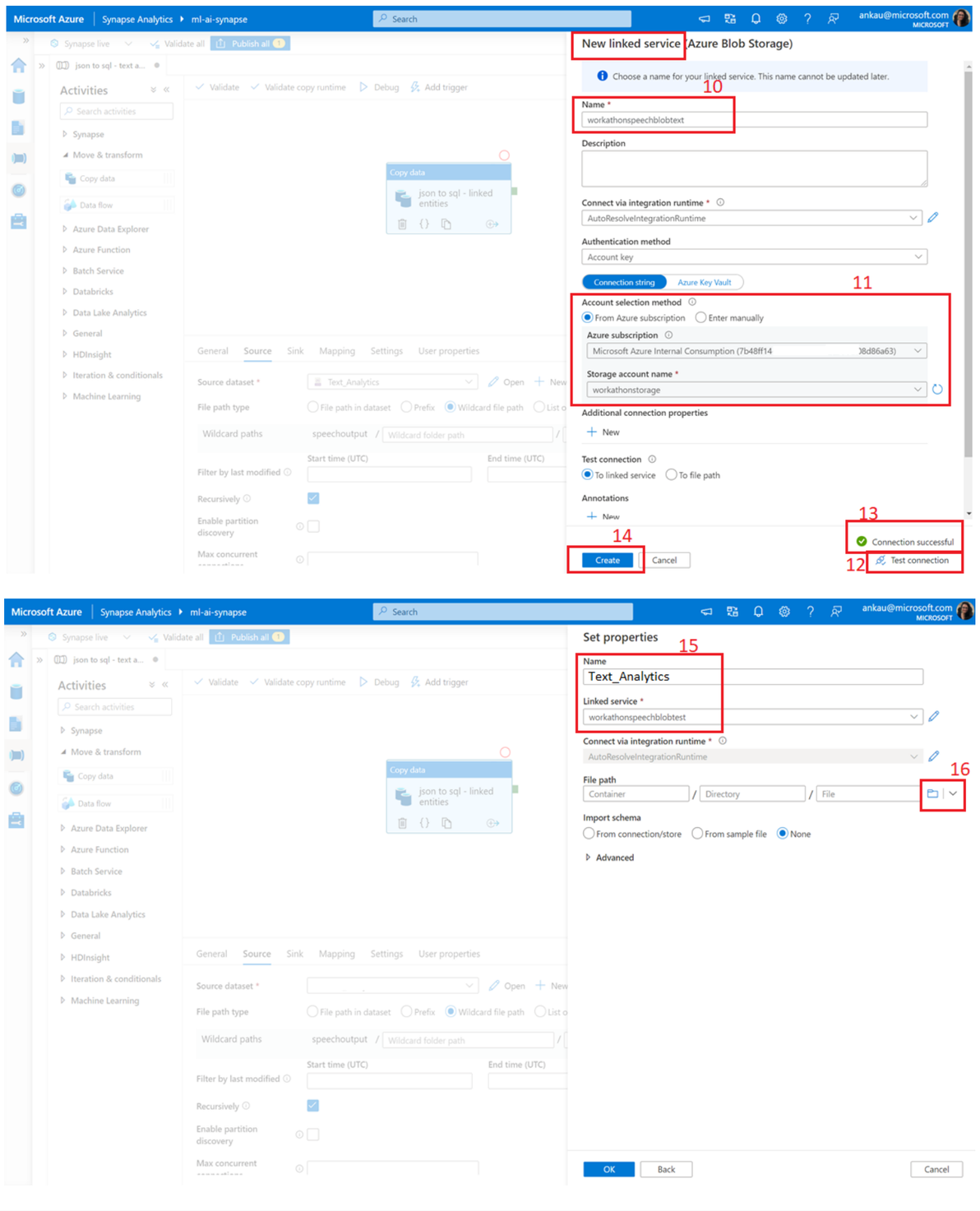
Speechoutput container in which we saved the final JSONs is the source for each of the copy modules.

Follow the steps as highlighted to create a connection to the source dataset:

1. Go to the Source tab
2. Click + New (since the source is the same for every copy activity, we will have to create the dataset just once)
3. Seach for Blob
4. Select Azure Blob Storage
5. Click Continue
6. Select JSON for format type
7. Click Continue

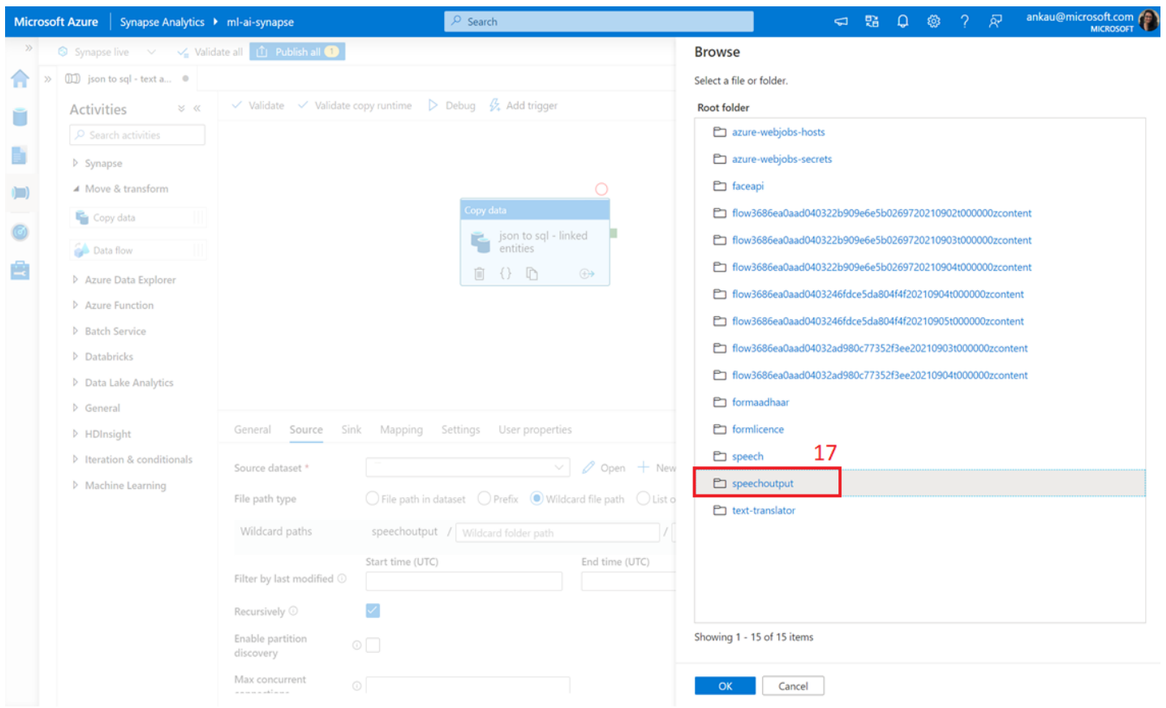


8. Name : Text Analytics
9. Linked Service : Create New. This will take you to New Linked Service wizard
   [Linked Service is a logical connection mapping to a data source, containing the metadata and connection keys for the resource. Once you create a Linked Service, you don't have to provide the connection strings time and again. You can use the same Linked Service at multiple places.
   Here, we are creating a Linked Service to the speechoutput container in our blob storage account.]

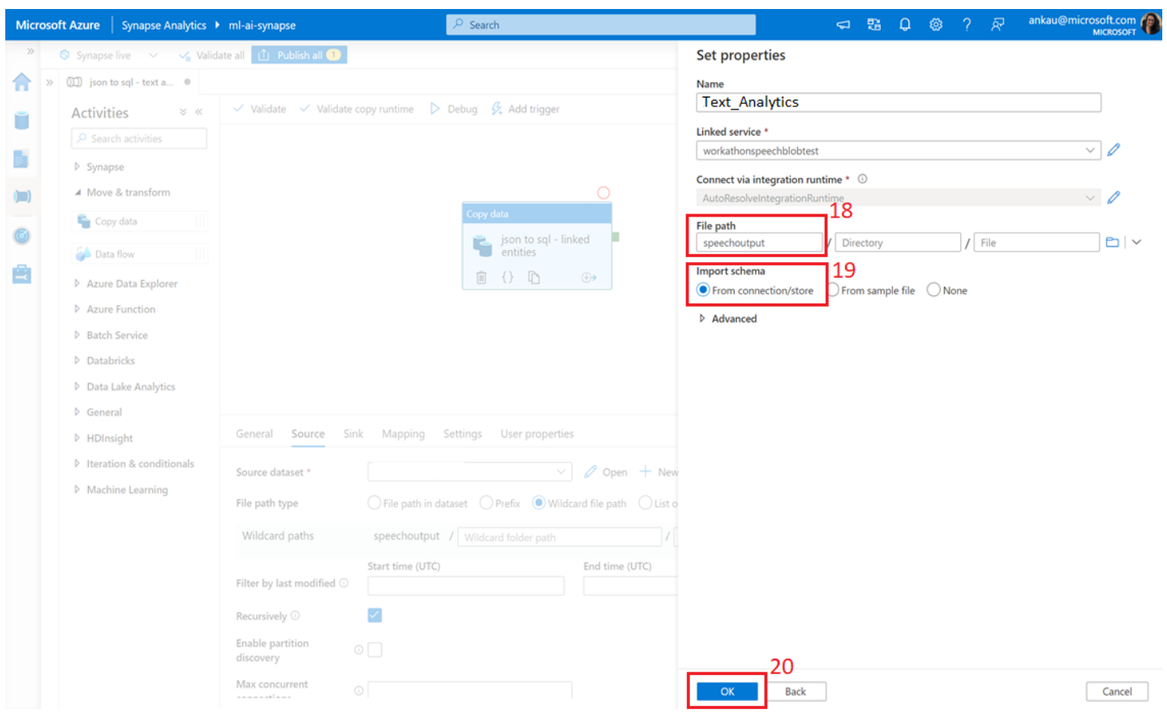


10. Name : workothonspeechblob
11. In the Account Selection Method, select From Azure subscription and select the storage account you created and used for Logic Apps flow that contains the speechoutput container
12. Click Test Connection
13. Make sure you get a 'Connection successful' message
14. Click Create. This will create the Linked Service and take you back to setting Dataset properties

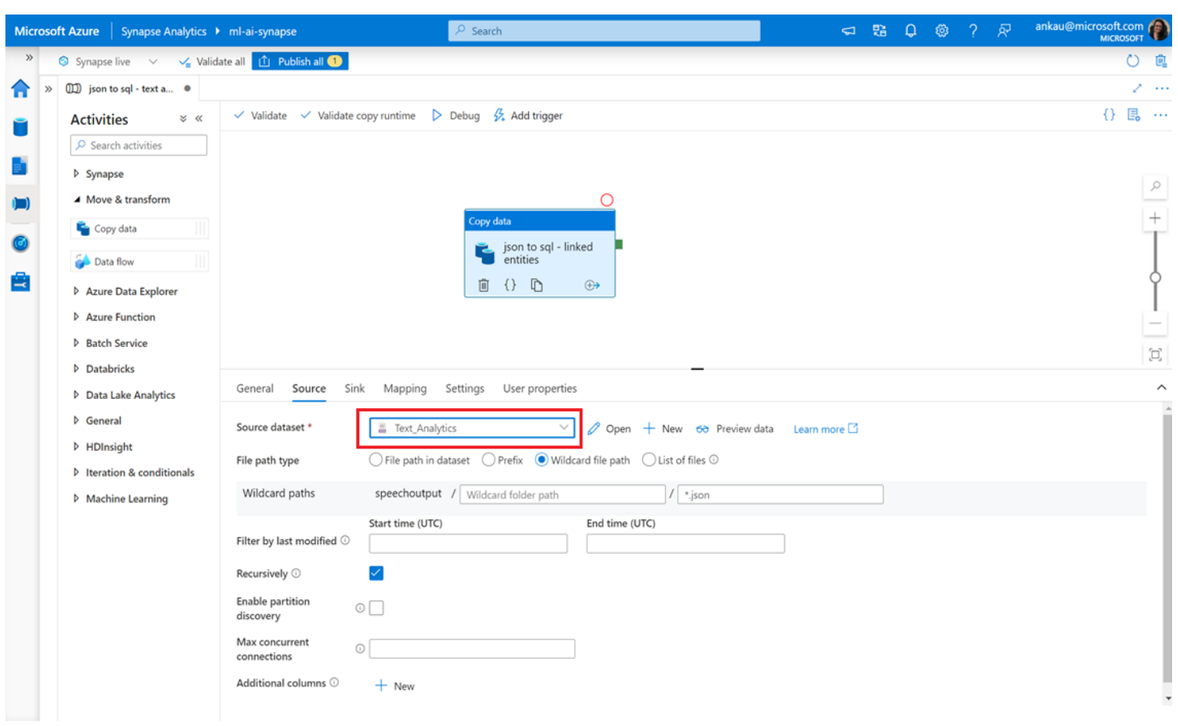
15. On the Set properties page, give the name Text_Analytics and the linked service you just created should be auto populated
16. To select the file path, click on the Folder icon



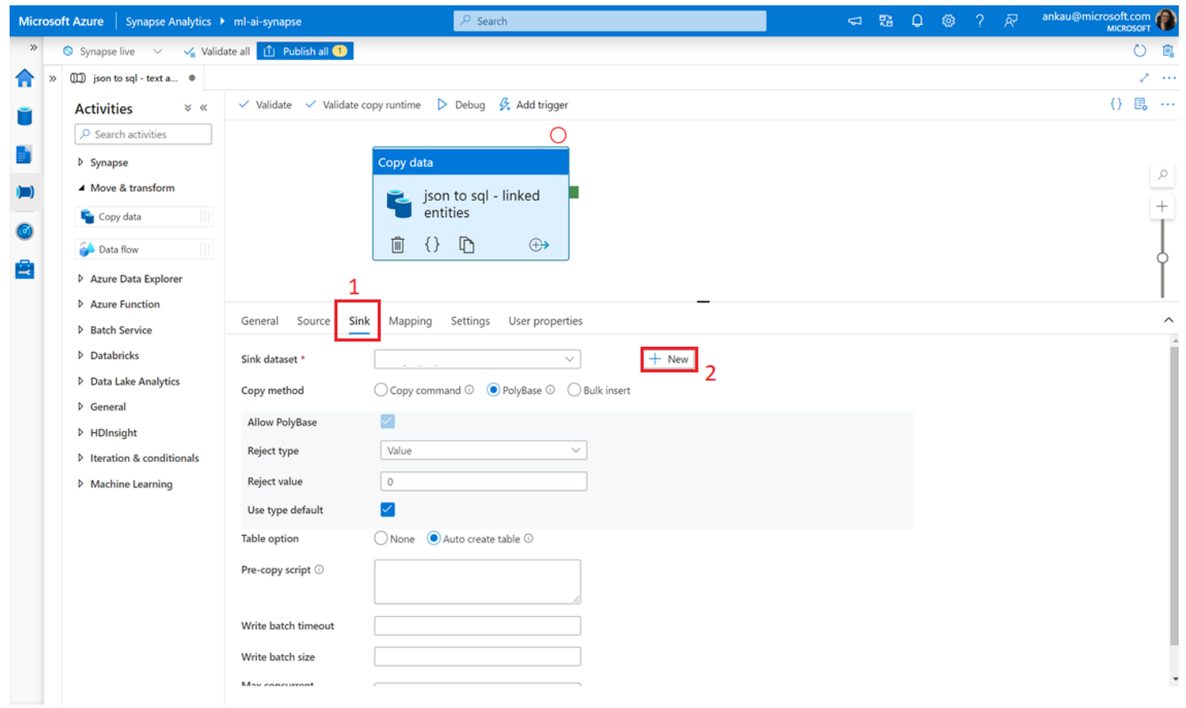
17. Select speechoutput folder and Click OK [This is the folder where our final JSONs files were stored in the 2nd Logic Apps workflow]

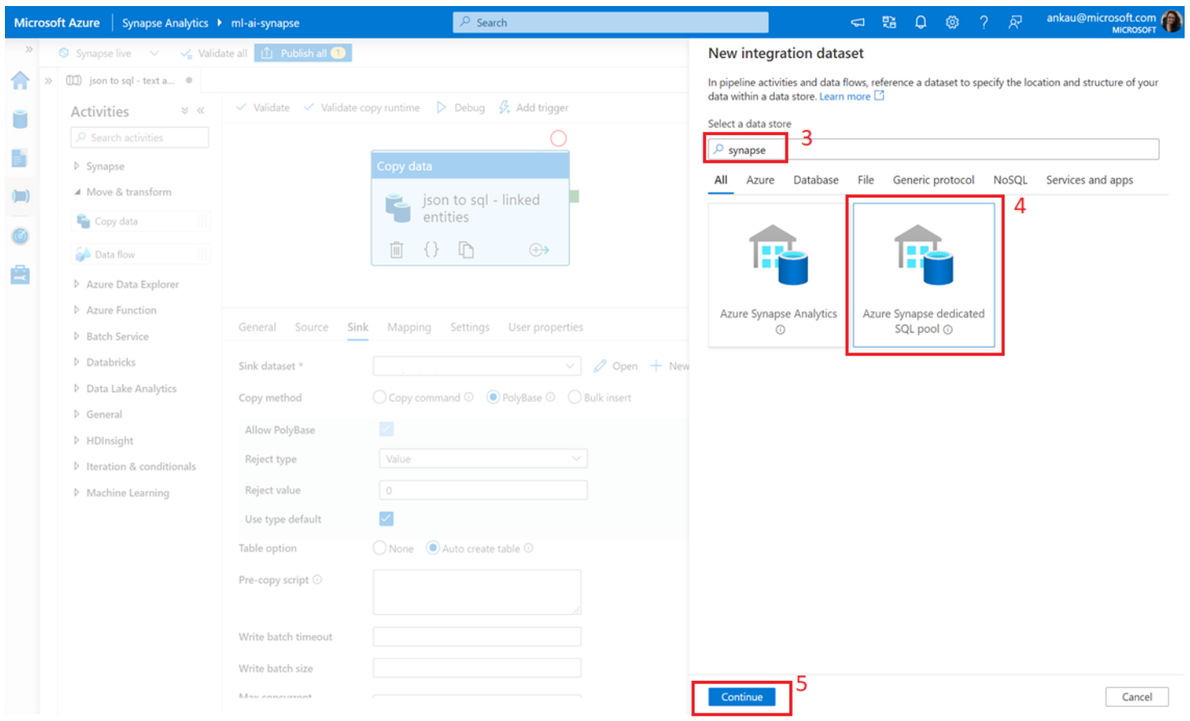


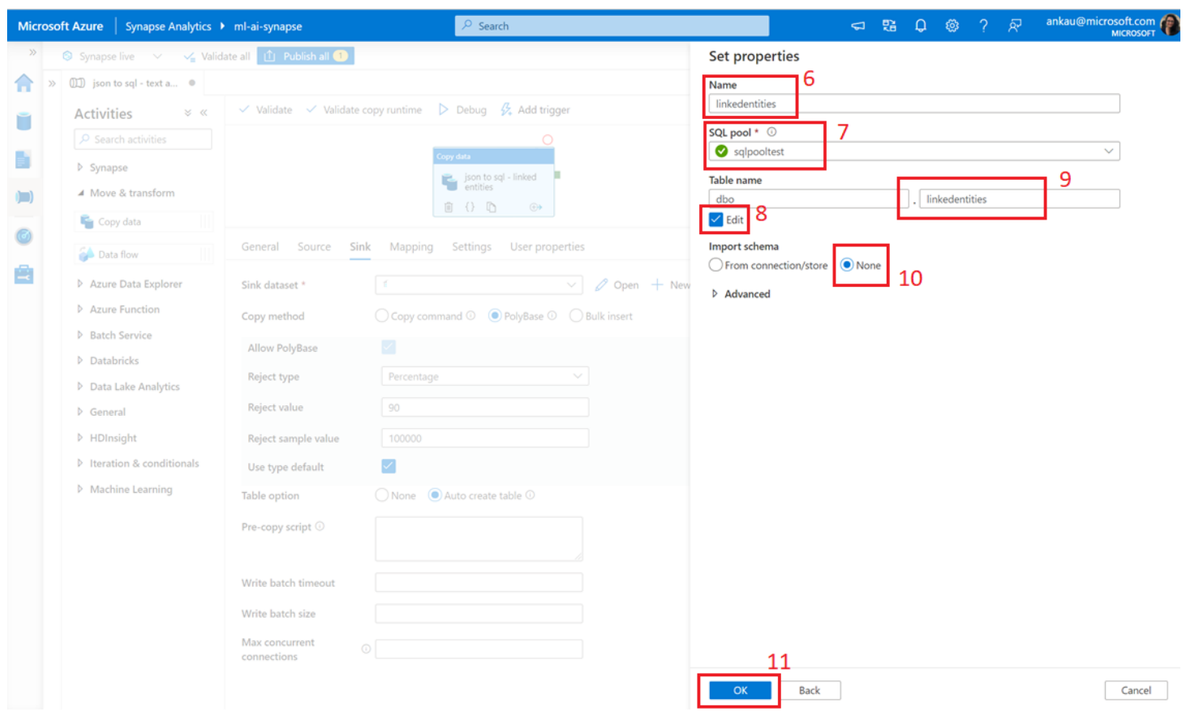
18. File path will now contain speechoutput in the container path box. Leave the directory and folder blank
19. In Import schema, select 'From connection/store'
    [This will pick the schema and metadata information from the source files and pass it on to the destination. We have chosen this option since our JSON files have a proper hierarchy that we can leverage to decide the schema.]
20. Click OK



21. Text_Analytics dataset just configured is now set as Source dataset. Confirm the other properties are similar to the ones in the screenshot





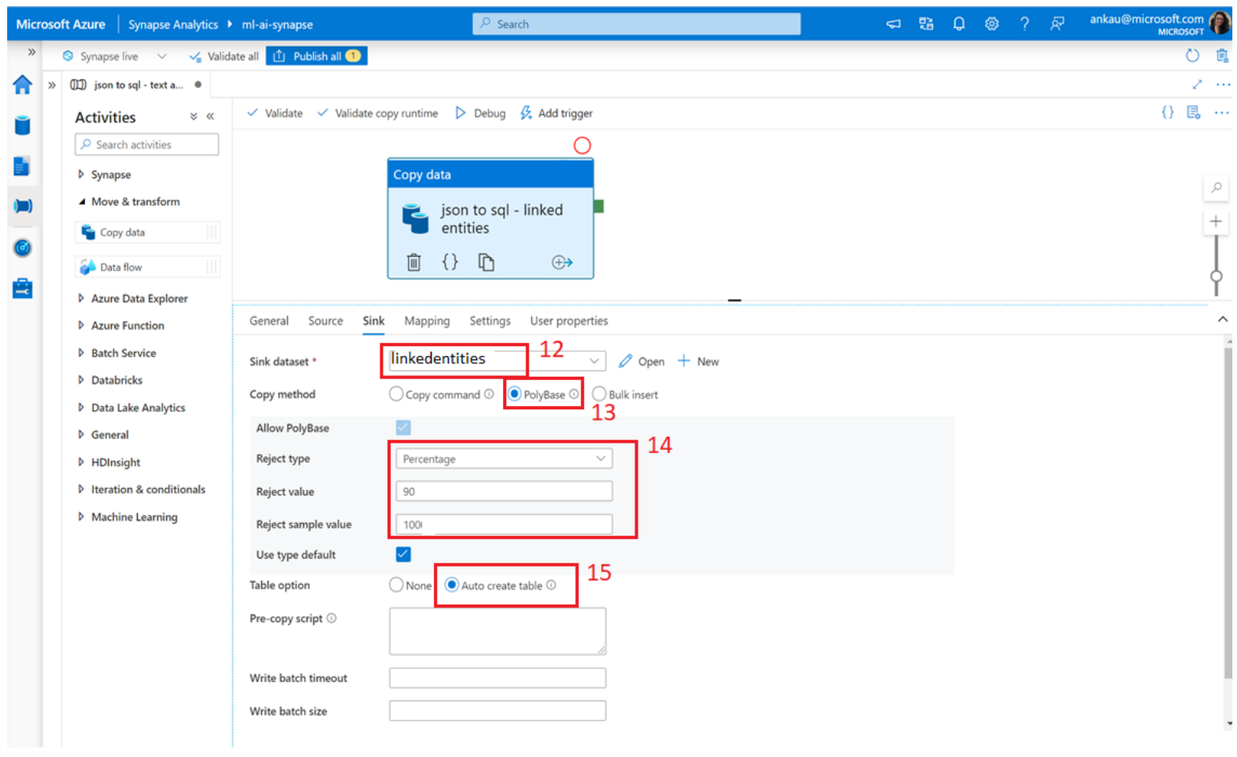


Sink settings

For each of the copy activities, we will be creating a new Sink dataset, since the destination of each will be a different table in the dedicated SQL pool.

Follow the steps as highlighted to create a connection to the sink dataset:

1. Go to the Sink tab
2. Click + New (since the sink is different for every copy activity, we will have to create the dataset for each copy activity separately)
3. Seach for Synape
4. Select Azure Synapse dedicated SQL pool
5. Click Continue
6. On the Set properties page, give the name linkedentities
7. SQL Pool : select the dedicated SQL pool you created earlier
8. Click Edit to enter a custom table name. Since this is a new table and the first time this pipeline runs, this will be created on the fly, hence we can't find it in the dropdown.
9. Table Name : dbo.linkedentities
10. In Import schema, select None
    [Since we are not pre-defining the target schema for the SQL table, to leverage the Source schema definition and apply it on Target while loading the data we need to select None.]
11. Click OK

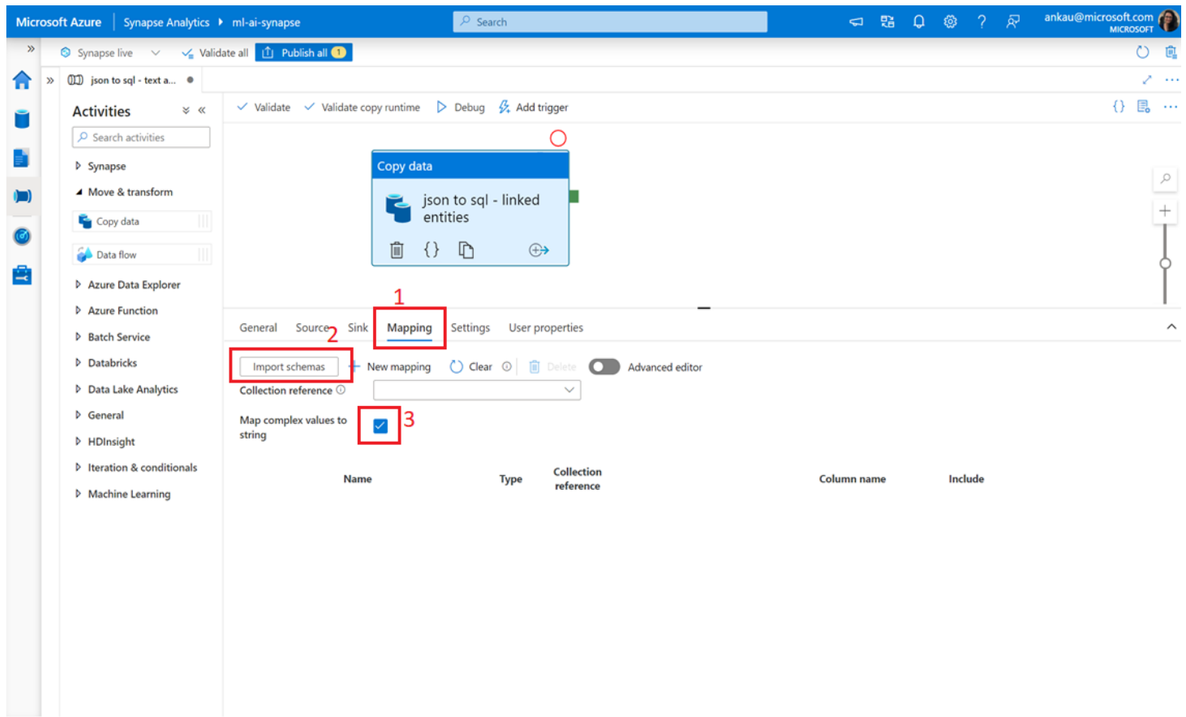


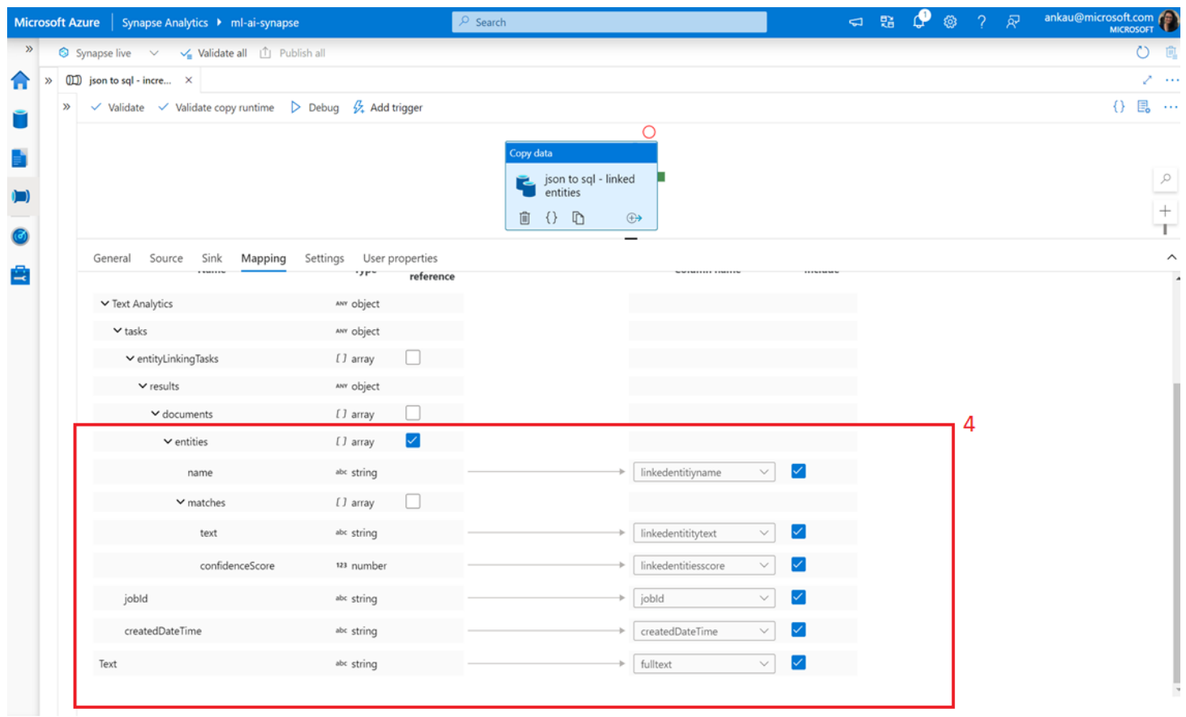
12. linkedentities dataset just configured is now set as Sink dataset.

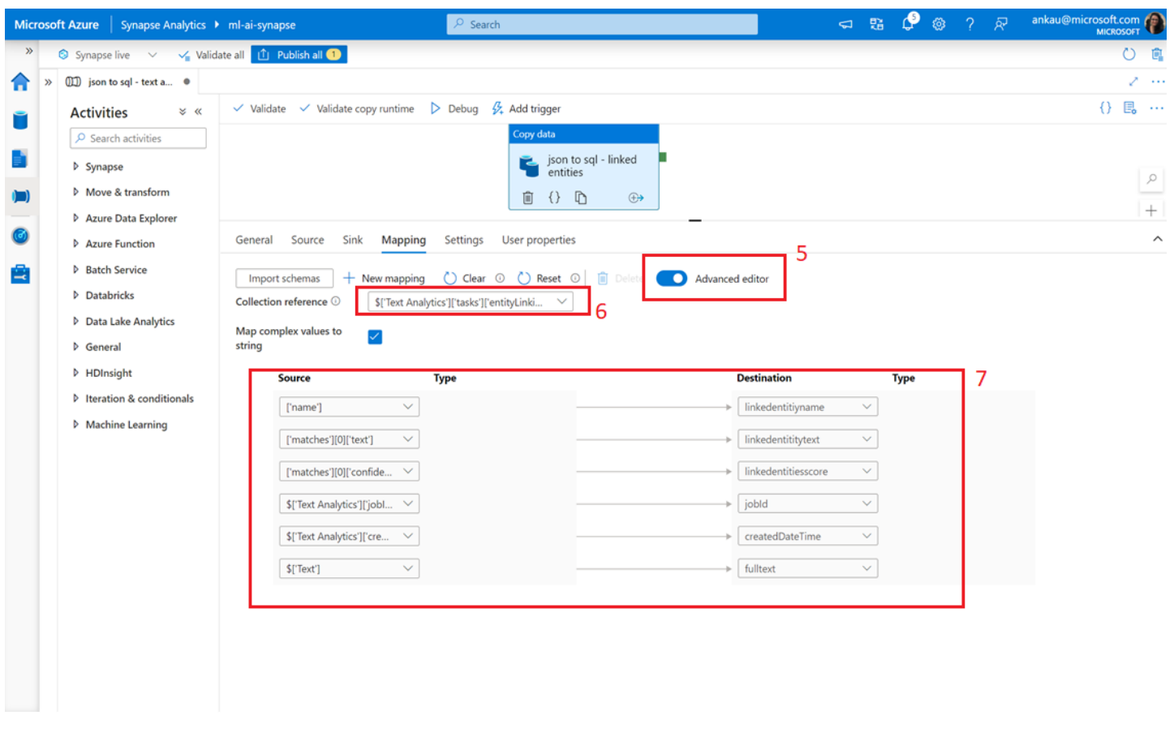
13. Copy Method : Polybase
    [Polybase is a technology that accesses external data stored in Azure Blob storage, Hadoop, or Azure Data Lake store using the Transact-SQL language. This is the most scalable and fastest way of loading data into an Azure Synapse SQL Pool.]

14. Reject type : Percentage [Specifies whether reject value is a percentage or literal value]
    Reject value : 90 [Specifies the number of rows that can be rejected before query fails. In production scenario, this value will be quite low]
    Reject sample value : 100 [Number of rows to retrieve before percentage of rejected rows is recalcuted]

15. Table Option : Auto create table [Since this is a new table and the first time this pipeline runs this will be created on the fly. For the table definition, it will pick the schema from the Source data]







Mapping settings

The source JSON contains a lot of different key value pairs under various tasks like extractiveSummarizationTasks, sentimentAnalysisTasks, entityLinkingTasks, entityRecognitionTasks etc.

Since each copy activity caters to 1 of these tasks, we will process just the relevant key value pair and copy them to the corresponding SQL table.
Hence, this is an important step that helps in mapping the input JSON keys to the SQL table columns.

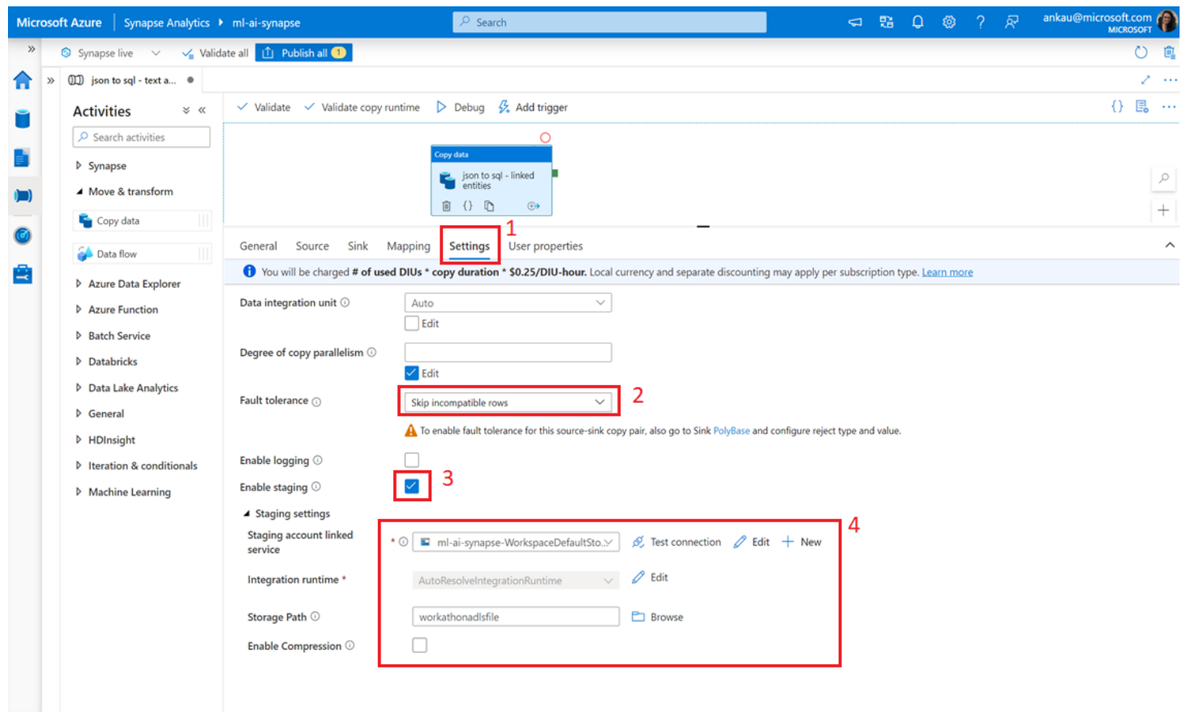
This step will be similar in all the 4 copy tasks, except for the keys that we pick to be dumped into the corresponding table.

Prior to going forward, do observe the JSON output of the text analytics results and for each of the tasks, observe the JSON and Array structure

1. Go to Mapping tab
2. Click on Import Schema. This will import the complete JSON hierarchy for you. We will now select the important values related to linked entities task that we want to copy to the linkedentities table that we created as the sink
3. Select the checkbox for 'Map complex values to string'

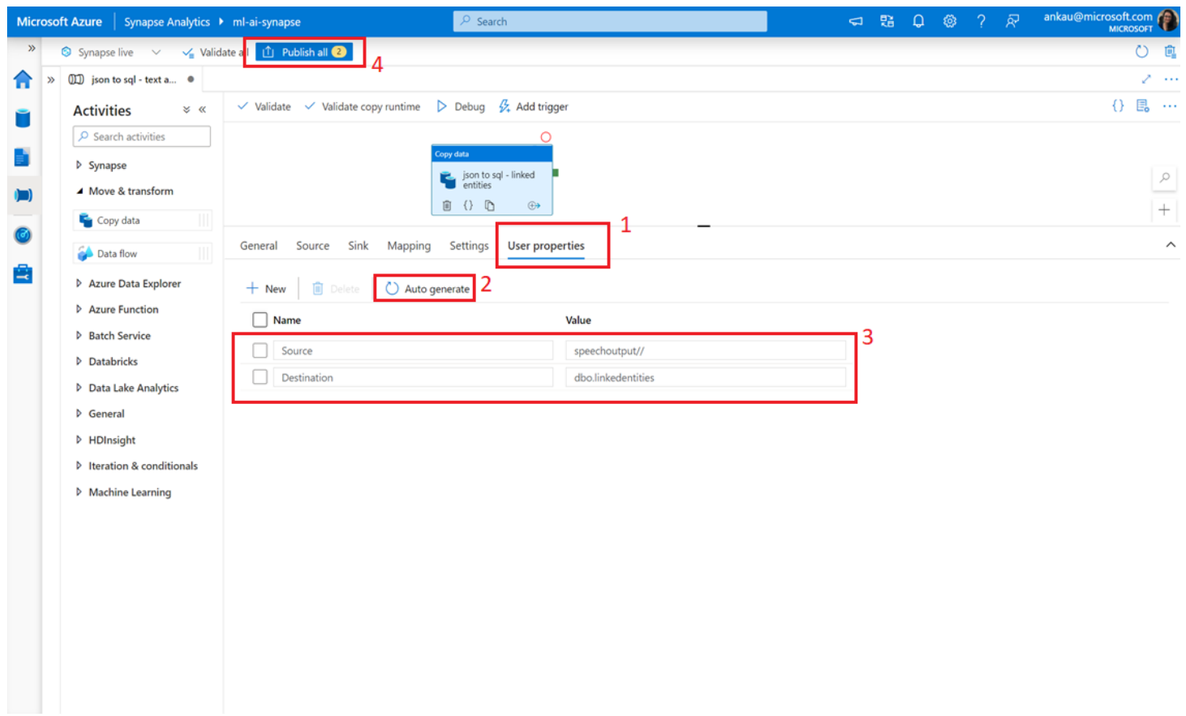
4. Select the checkbox Collection reference for entities.
   [Some of the Keys in JSON contains a single value while some contain an Array, which is again a combination of multiple key-value pairs bundled under 1 key. Thus, to extract all the key values from an Array, we have to process it iteratively (aka in loop).
   'Collection Reference' feature allows us to extract these values without implementing a loop separately.
   In this case, we want to iterate at the entities level, to extract the name and text & confidence score in the 'matches' Key.
   Also check the box for jobid – this will act as a unique ID for each document, in case you want to perform group by on document level or perform joins etc
   Check the box for createdDateTime.
   Also select the main 'Text' of the document.
   Rename the columns as shown.
   See step 4 in screenshot for reference. Make sure to select from the right hierarchy of key value pairs.]

5. Toggle to Advance Editor in order to review your column mappings
6. The resolved document hierarchy for entities will be populated in the Collection reference dropdown.
7. Confirm the structure of the key value pairs you selected and column names for the Destination



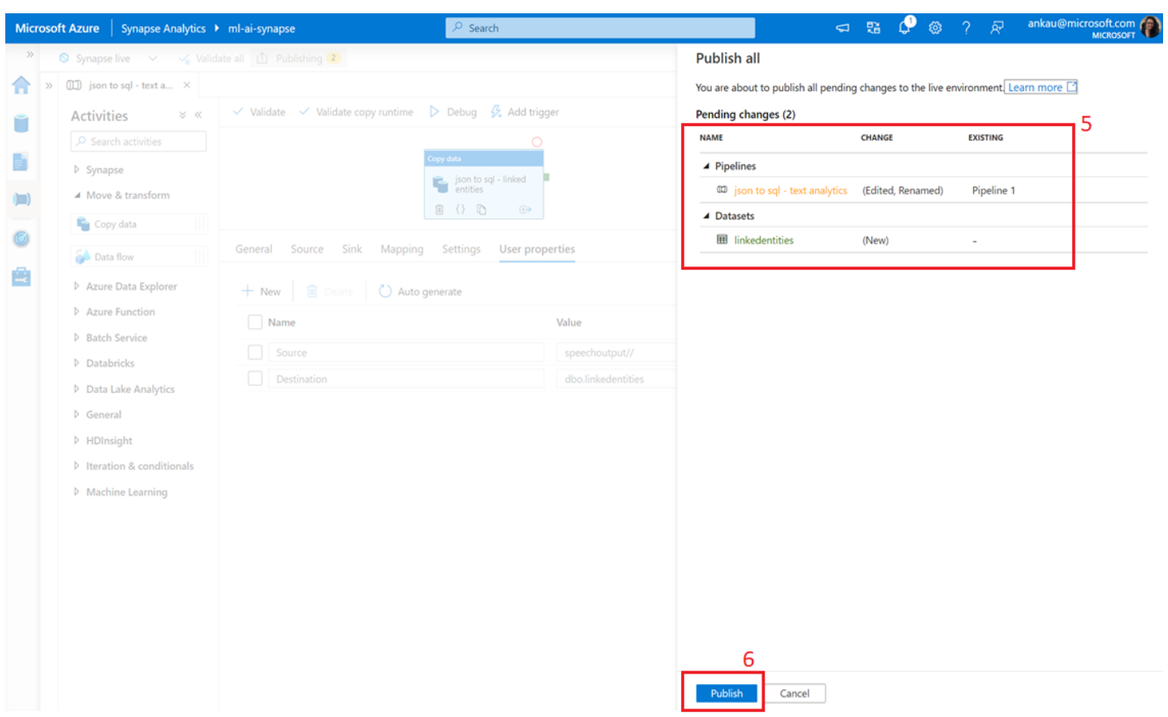
Settings

1. Go to the Settings tab
2. Set Fault tolerance to ‘Skip incompatible rows’
3. Enable Staging – Check
4. Staging settings :
   Linked service – select the default storage account
   Storage Path – Select the container you created when you provisioned Synapse Workspace
   [While executing pipelines, they require a temporary space where they can create and store temporary objects for the time the execution of the pipeline is ongoing.]



User properties

1. Go to User properties tab
2. Click Auto generate
   [In user properties, you can pass parameters and their values in key-value pairs and auto-generate helps generate a JSON file dynamically using these parameters at the time of execution]
3. Confirm that the Source & Destination values are as highlighted
4. Publish the changes

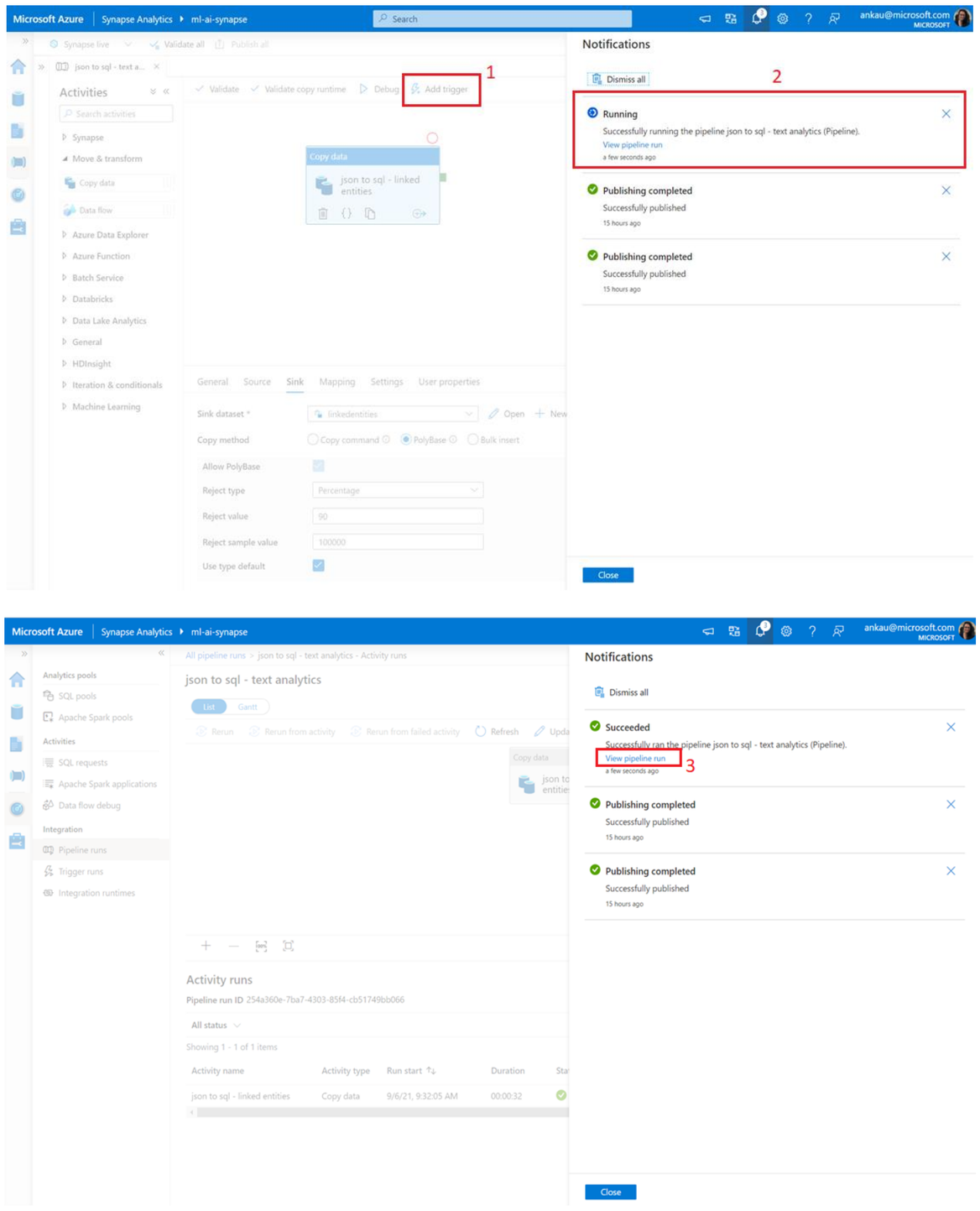


5. The ‘Publish all’ tab will show you the changes you’ve made
6. Click Publish

Like we discussed earlier, you can create 4 different pipelines with 1 copy activity each or a single pipeline with 4 independent copy activities.

We will be creating the other 3 copy activities in this pipeline, however, for now we will publish and trigger the pipeline to confirm that we haven’t made any errors and it’s working as intended.

Similarly, we can create other copy pipelines, by copying the ‘linked entities’ copy task and make changes in the Sink and Mapping tabs.



Trigger the pipeline

A trigger operation causes the pipeline to run.

Types of triggers -

1. **Trigger now** : This is a manual trigger to start the execution of pipeline at that very moment. This is generally used during testing & development.
2. **Scheduled trigger** : This causes the pipeline to run periodically as defined. There is also an option for using a Tumbling window trigger, that operates on a periodic interval, while also retaining state.
3. **Event based trigger** : This causes the pipeline to trigger when a particular event occurs.

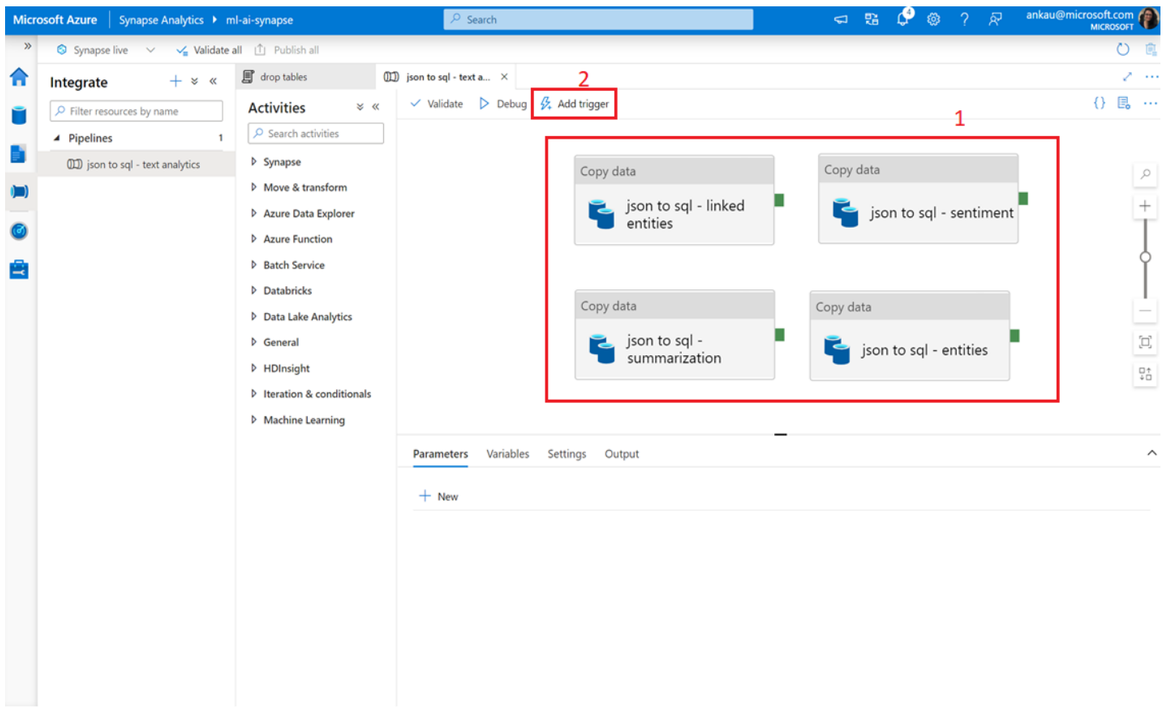
We will now trigger the pipeline to see whether it executes as intended.

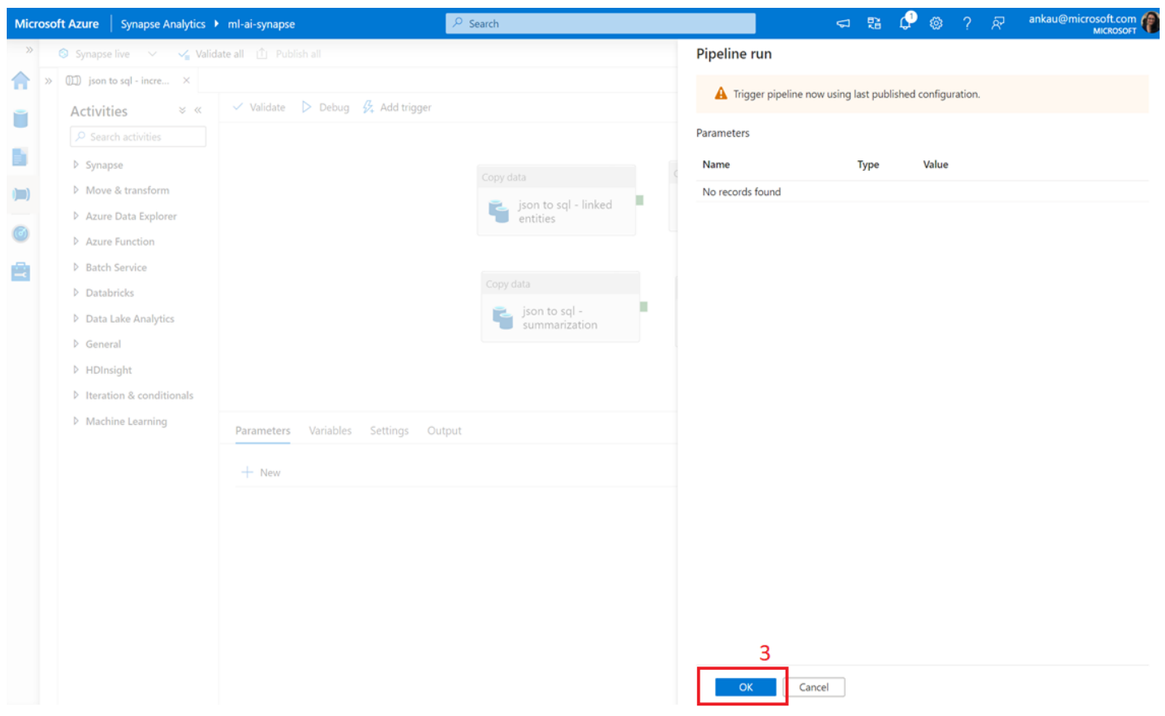
Prior to running this, make sure your 'speechoutput' container has a couple of JSON files with the Text Analytics results.

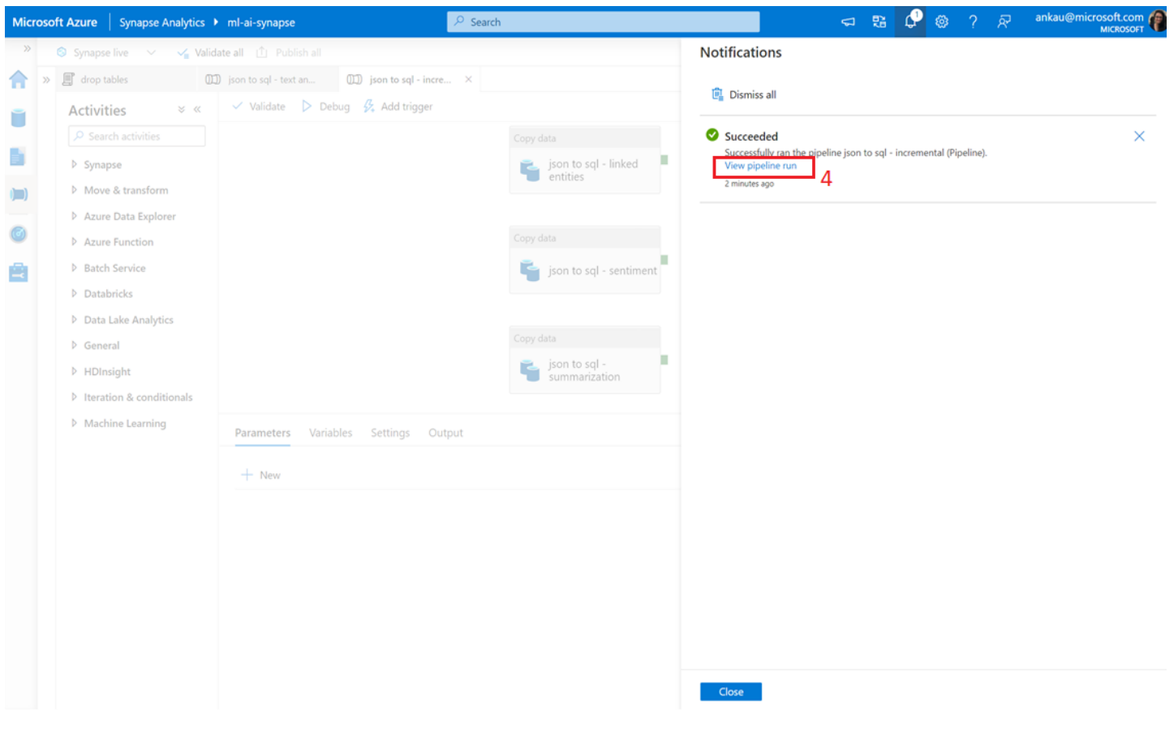
1. Click Add trigger and select Trigger now
2. The notifications tab will show the pipeline as running
3. Once successful, select 'View pipeline run'
   This will take you to the pipeline runs in Monitor tab. Observe the pipeline input and output, number of files executed etc.

View SQL table

1. Go to the Data tab
2. Select Workspace
3. Right click on db.linkedentities table
4. New SQL script > Select TOP 100 rows
5. Observe the table outcomes







Trigger final pipeline

Once you have created linked entities copy tasks, also create the tasks for summarization, sentiment and entities copy activity.

Hints :

- Copy previous task (linked entities) as a starting point

- Go to the Sink settings in each task and create new Sink dataset

- Carefully observe the hierarchy in the JSON file to decide which Keys to iterate over in the Mapping Tab and resolve that array using Collection Reference.
  Solution :
  1. Select the check box for collection reference on sentences array level for summarization copy task
  2. Select the check box for collection reference on sentences array level for sentiment copy task
  3. Select the check box for collection reference on entities array level in entity copy task

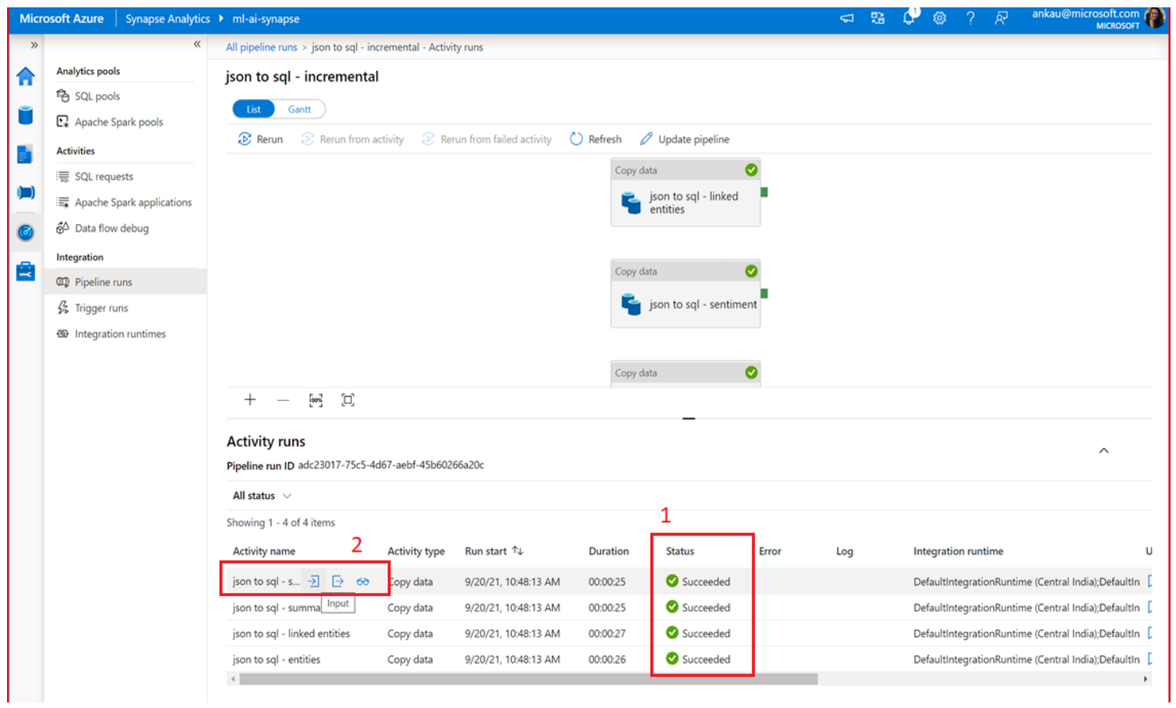
- View the values obtained for each task in the text analytics JSON output and select the values you want to copy into the SQL tables
  For example :
  1. Text, rankScore for summarization copy task
  2. sentiment at document & sentence level, confidence score for positive, negative, neutral sentiment etc for sentiment copy task
  3. Text, category, confidence score etc for entities copy task

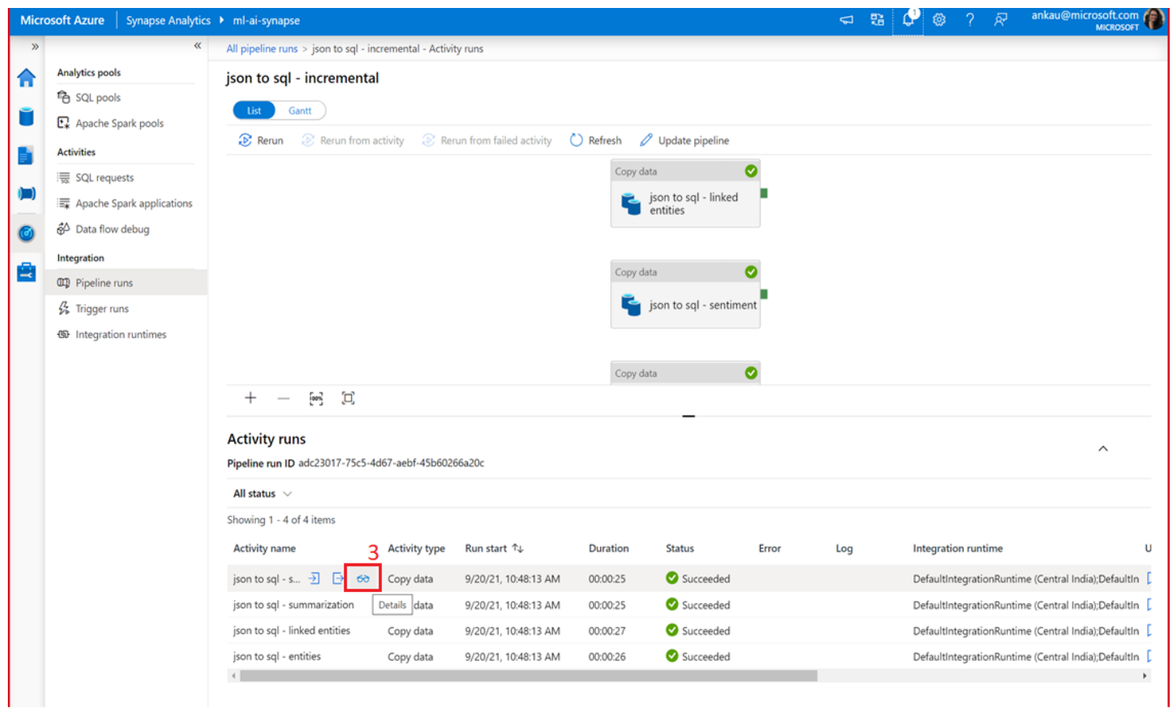
  Make sure to include jobId, createdDateTime, Text in all the copy activities, the way we included them in 'linked entities' copy task

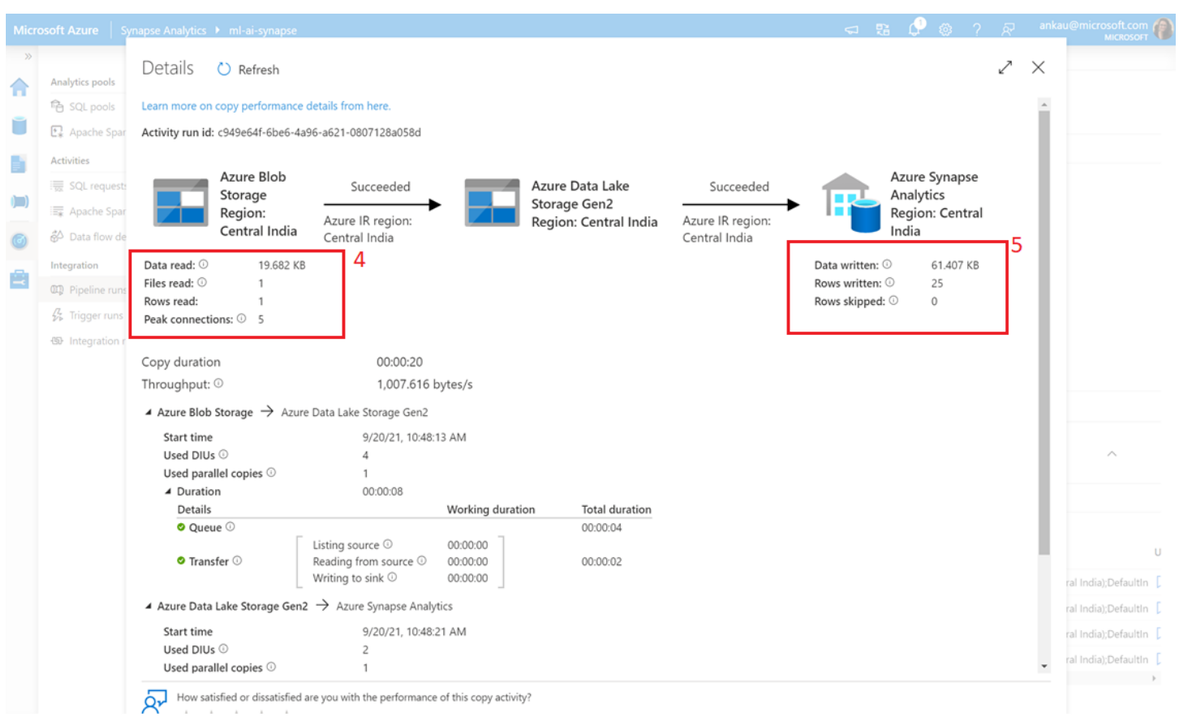
Trigger the pipeline once you are ready with all 4 copy tasks.

In case you ran the pipeline for linked entity task earlier, you might want to drop the linked entities table in the SQL Pool by going into the data tab of Synapse Studio, to avoid duplicate values.

4. Go to 'View pipeline run' once the pipeline is run successfully.







Monitor pipeline run

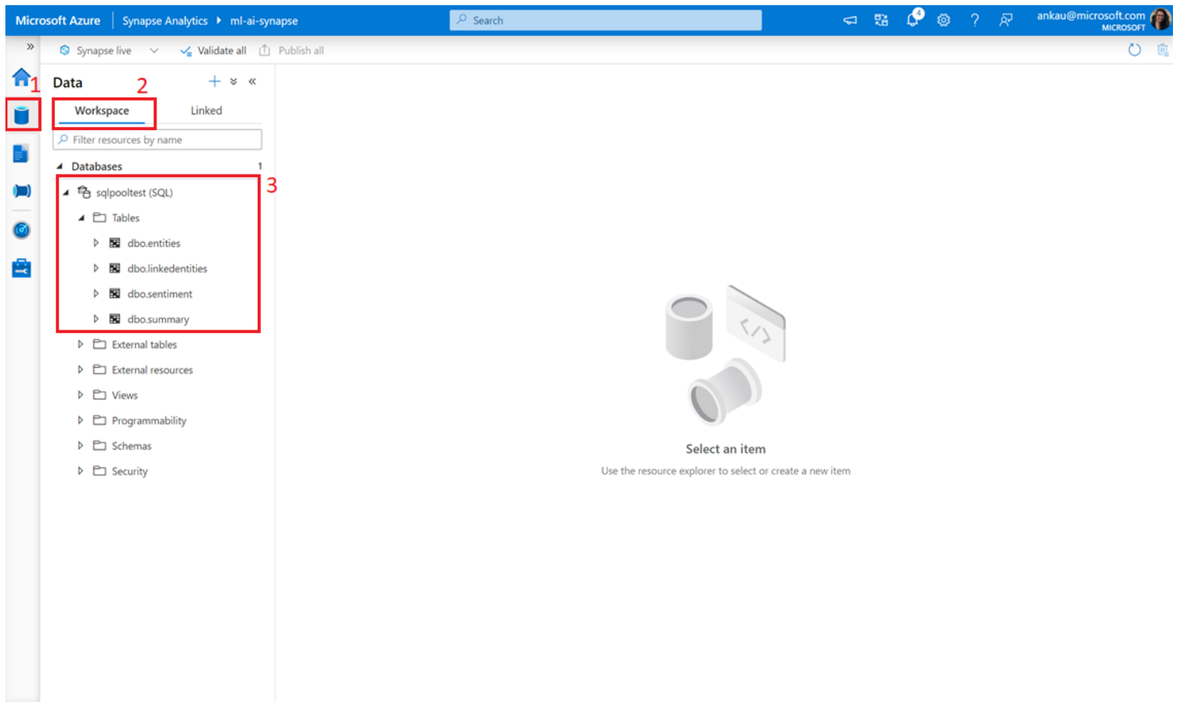
This functionality helps you to monitor the status of your pipelines, in terms of whether it is in process, has succeeded or has failed. This also provides additional information like, input and outputs for each step, duration for which the pipeline or steps ran, the compute used etc.

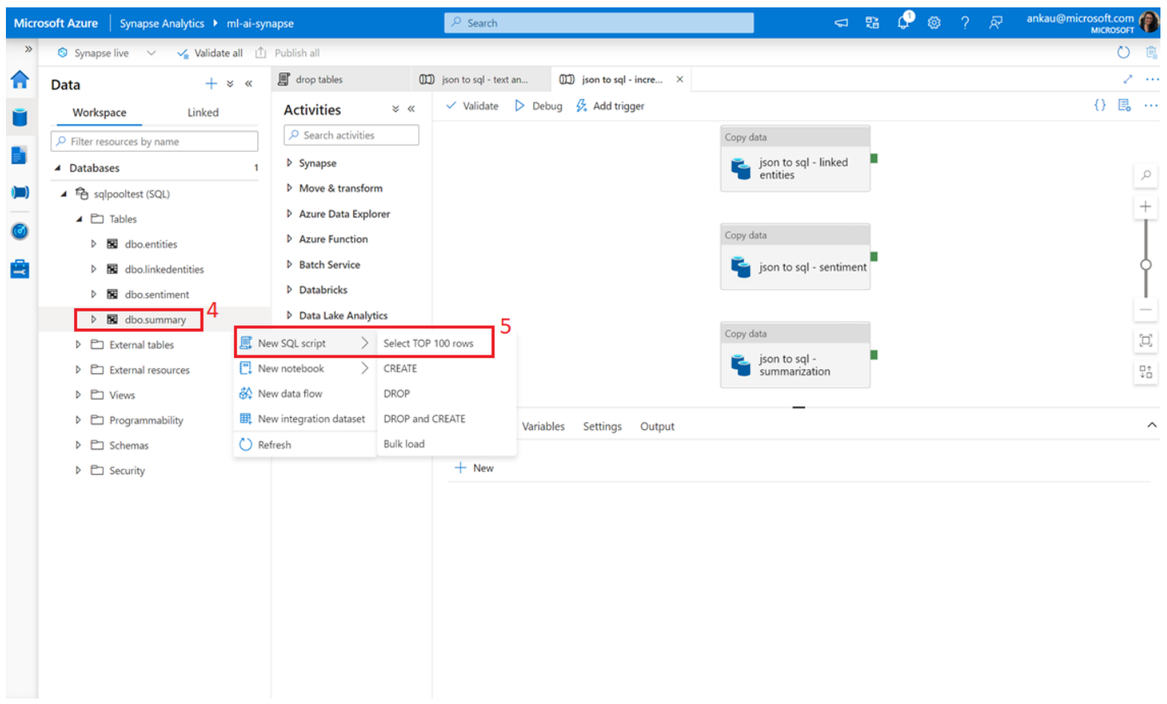
Selecting ‘View pipeline run’ will take you to Pipeline runs in Monitor tab.

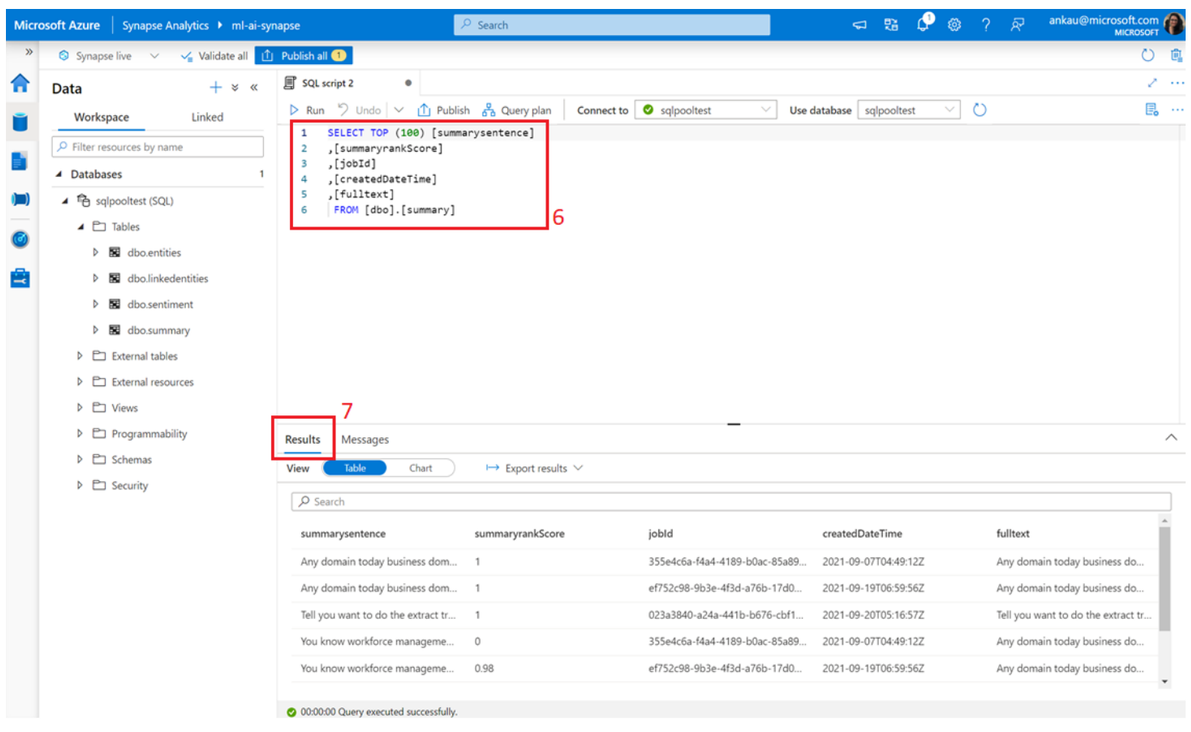
1. View the status for each of the copy tasks. Revisit pipeline in case any of them fails.
2. Hover over any of the tasks and view each of the input and output tabs. Observe JSON and file structures for input & output.

3. Select the details tab

4. View the input data in terms of data, files, rows read etc
5. View the output details in terms of data & rows written, rows skipped etc







View SQL table

Once the pipeline with all copy activity tasks is run successfully, you will have 4 SQL tables, 1 for each copy task.

1. Go to the Data tab
2. Select Workspace
3. View the SQL tables for each copy task

4. Right click on any of the tables
5. New SQL script > Select TOP 100 rows

6. Observe the SQL script
7. View the table outcomes in the Results tab

Note : Perform the task for other tables as well.

Homework

1. Once you have done the batch processing, we would recommend you try executing incremental load of data – which means process additional audio files in order to load the incremental data through the end-to-end scenario by creating an incremental load pipeline in Data Factory.
2. Hints :
   a. You can add event-based triggers for the speechoutput blob container to trigger the pipeline as soon as a new file is added to that container (If you take this approach, make sure to move the processed files to another location so they aren't processed repeatedly)
   b. You can add schedule based triggers to run the pipeline every few minutes or hours and process the files than came in during that time interval (You can alter the 'Filter by last modified' field in the Source tab and filter files to the ones that were added in the last scheduling interval.)

Additional recommended resources

Responsible AI being a part of best practices, we encourage you to read this.

Security best practices

Security best practices Azure Cognitive Services

Security best practices Data Factory

Security best practices Logic Apps

Security best practices Blob Storage

Security best practices Synapse Analytics

[There is more to security in each service, however, the given links talk about the baseline security guidelines]

Documentation References

Text Analytics Documentation , Text Analytics API & Error references

Speech-to-text Documentation, Speech-to-text API references

Logic Apps Documentation

Synapse Analytics Documentation

Blob Storage Documentation