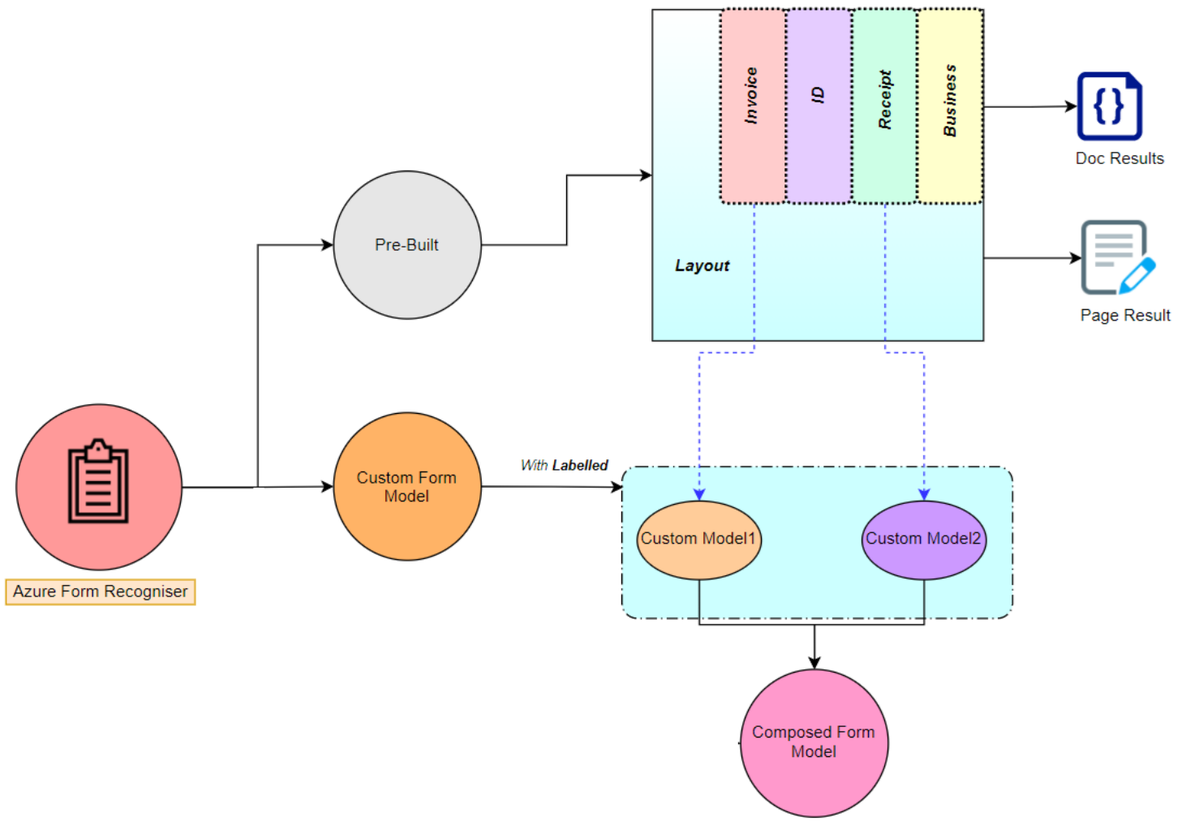
Document Name	HOL – Azure Form Recognizer Service v3.2
Author	Shiva S Tomar & Anupreet Kaur
Reviewer	
Executive Summary	Azure Cognitive APIs enable the developers of all skill levels to add human intelligence in their applications. The services are designed for developers interested in pursuing DS/AI/ML skills and people who want to acquire the deep technical knowledge on the Cognitive APIs of Azure, despite not having Machine Learning expertise.
Purpose	This document is created to help you gain level 350 working knowledge on Azure Form Recognizer Service. You will be able to explore each functionality offered by the service through the GUI Portal to train the model & REST APIs to observe the outcomes. We have also shared a sample dataset to replicate what we have used to create the content of this workshop. Once you complete these labs, you'll go from ***Zero to Hero*** on the respective Azure Cognitive service and should be able to ***D**emo, **D**evelop and **D**eploy* your own custom use cases. 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 explore all the features of a service offered through their APIs. The diagram shown in the beginning of the document is its functional Architecture; talking about the functionalities offered by the service in a flow. It also covers the Concepts, How-to and best practices about the service. This document is not intended to enable you with scenarios of deployment in production.

Service brief: Azure Form Recognizer Service

Azure Form Recognizer is a part of Azure Applied AI Services that lets you build automated data processing software using machine learning technology. Identify and extract text, key/value pairs, selection marks, tables, and structure from your documents—the service outputs structured data that includes the relationships in the original file, bounding boxes, confidence and more.

Diagram: Functional Architecture



The service offers various functionalities under 2 major categories, which are as follows :

1. Pre-built Models : These are out of the box models available to extract information from your forms & documents. Various models are available, such as :
    a. Layout – The layout model extracts the contents of the document in a structured format. Extract text, selection marks and tables structures, along with their bounding box coordinates, from documents. Results are obtained in a hierarchy such as page, line, word.
    b. Invoices – The Invoice model extracts key fields and line items from invoices and returns them in an organized structured JSON response. Invoices can be from various formats and quality, including phone-captured images, scanned documents, and digital PDFs. The invoice API will extract the structured output from all of these invoices.
    c. Receipts – The receipt model analyses and extracts information from sales receipts. It is a Deep Learning model leveraging Optical Character Recognition capabilities to extract key information such as merchant name, merchant phone number, transaction date, transaction total, and more from receipts written in English.
    d. Business Cards – The Business Card model analyses and extracts contact information from business cards. It is a Deep Learning model leveraging Optical Character Recognition capabilities with our business card understanding model to extract key information from business cards in English. It extracts personal contact info, company name, job title, email ID, Phone Numbers and more.

e. **Identity Documents** – The Identity Documents model analyses and extracts information from government-issued identification documents (IDs). It is a Deep Learning model leveraging Optical Character Recognition capabilities with ID recognition capabilities to extract key information from Worldwide Passports and U.S. Driver's Licenses (all 50 states and D.C.). The IDs API extracts key information from these identity documents, such as first name, last name, date of birth, document number and more.

2. **Custom Models** : Though Azure Form Recognizer service provides many pre-built models, however, in many scenarios they do not work as is and call for the need of building a custom model for your use case. For example, we have a pre-built ID card model which works well for some ID Cards but does not work so well for some geographies such as India. Therefore, in this workshop we have created custom models to read and extract the data of various Indian ID cards such as Aadhaar card & Driver's licence. You can use the custom model in standalone mode.

3. **Composed Models** : This is a combination of 2 or more custom models and it is most widely used due to the pragmatic nature of our production environments. For example, in production environments, we mostly work with a mixed bag of forms (receipts, IDs, invoices etc). In such scenarios, we can build 1 custom model for each form type and assimilate them together in a Composed model, to make all these Custom models work together and not in silos. Once your Composed model is ready, you can pass a new unprocessed form belonging to any of the Custom model categories comprising the composed model to generate the outcome. Behind the scenes, the Composed model automatically classifies the new input form and executes the relevant Custom model to generate the corresponding outcomes.

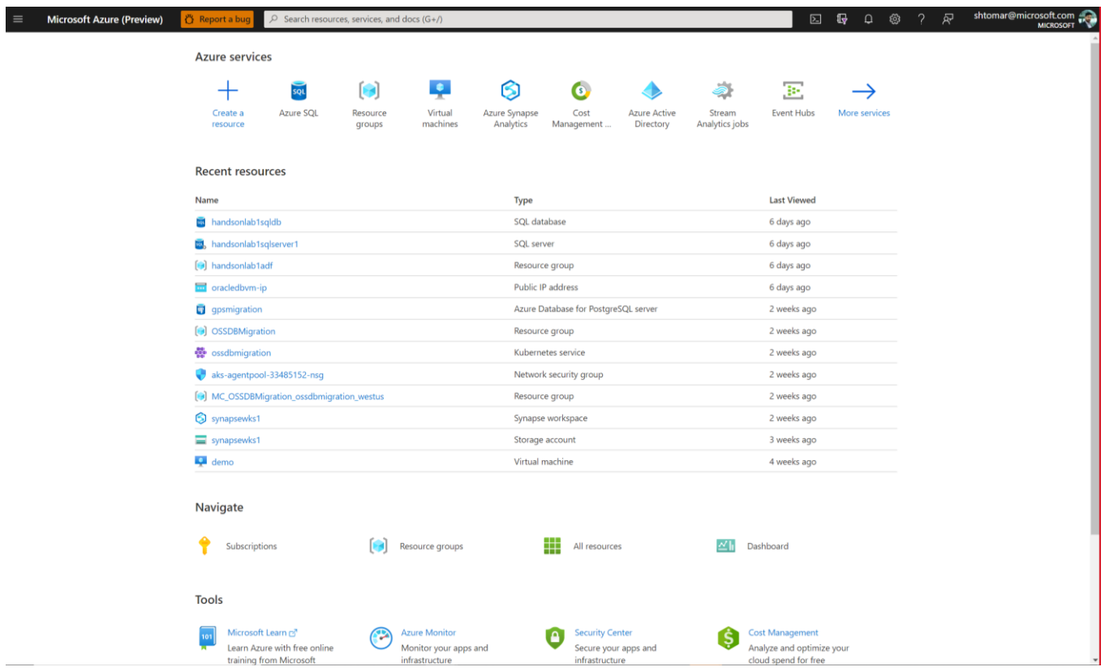
These APIs are available both as REST APIs and language specific SDKs.
You can deploy the models on the cloud or on the edge.

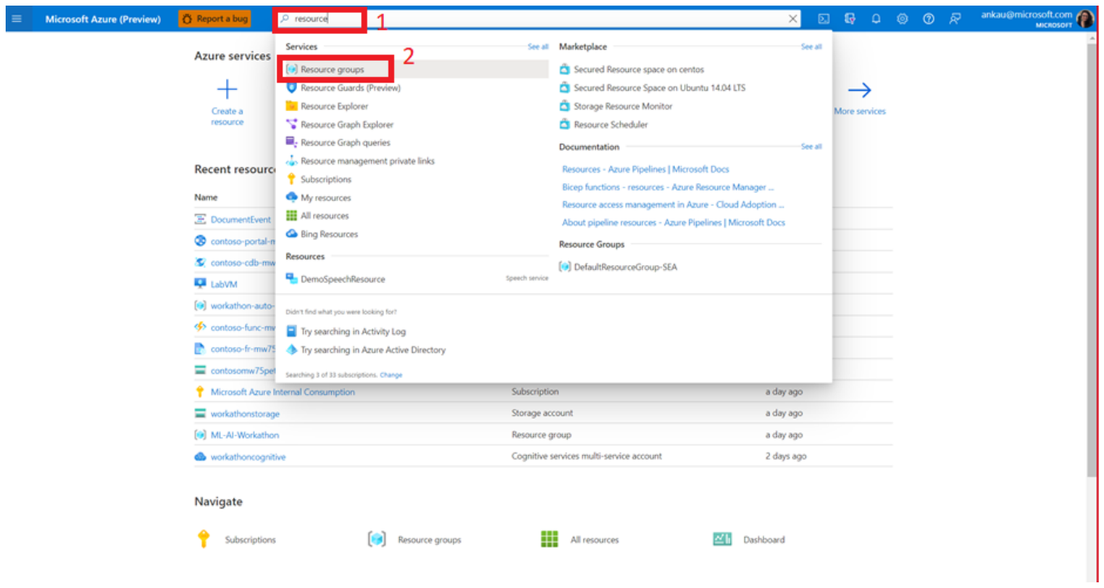
Step by step hands on guide to go from *Zero to Hero*

Pre-requisites

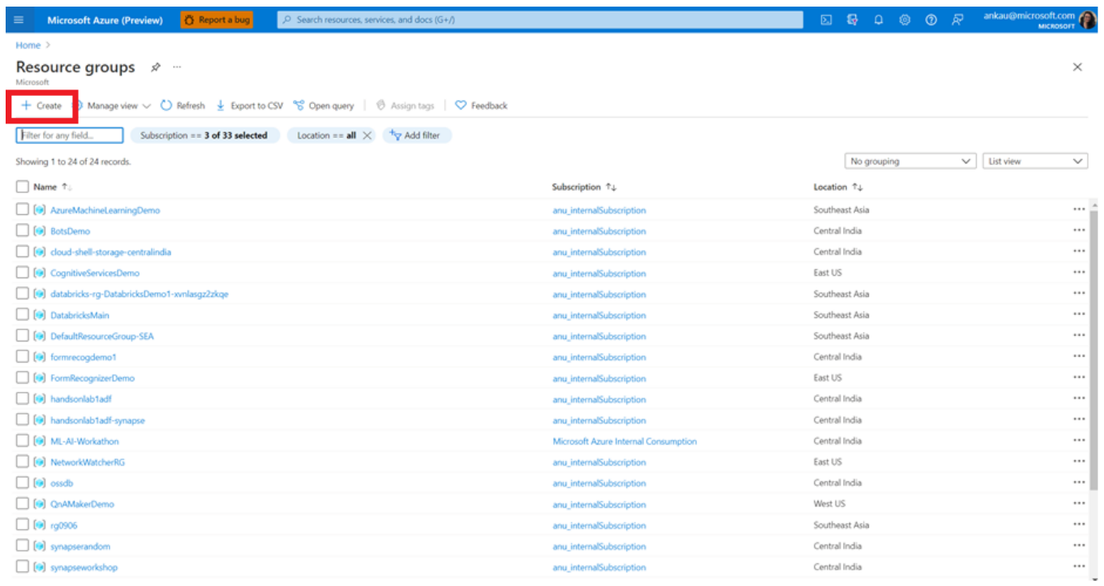
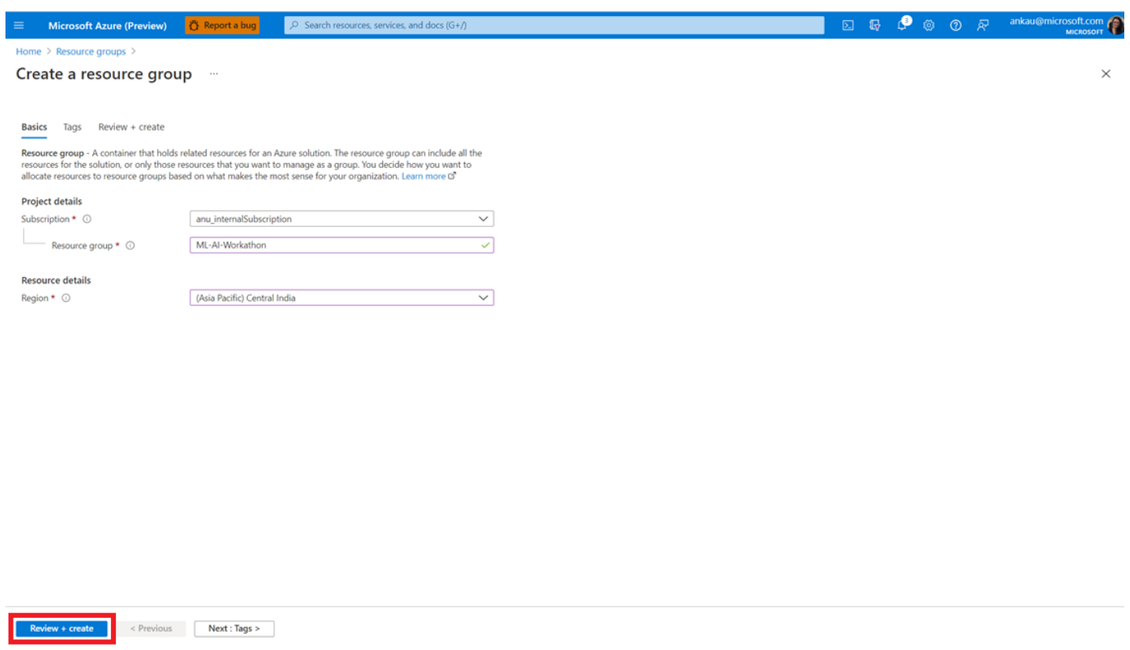
- Download & Install Postman
  - Postman is a free tool which allows you to make API calls
  - You can download the desktop application or get started using the web version (Download Postman | Try Postman for Free)
- An active Azure Account
  - You can use your current Azure Subscription or get started by creating a free trial account (https://azure.microsoft.com/en-in/free)
- Download the data for training & Testing from Data Folder in Form Recognizer folder.

Let's get started!

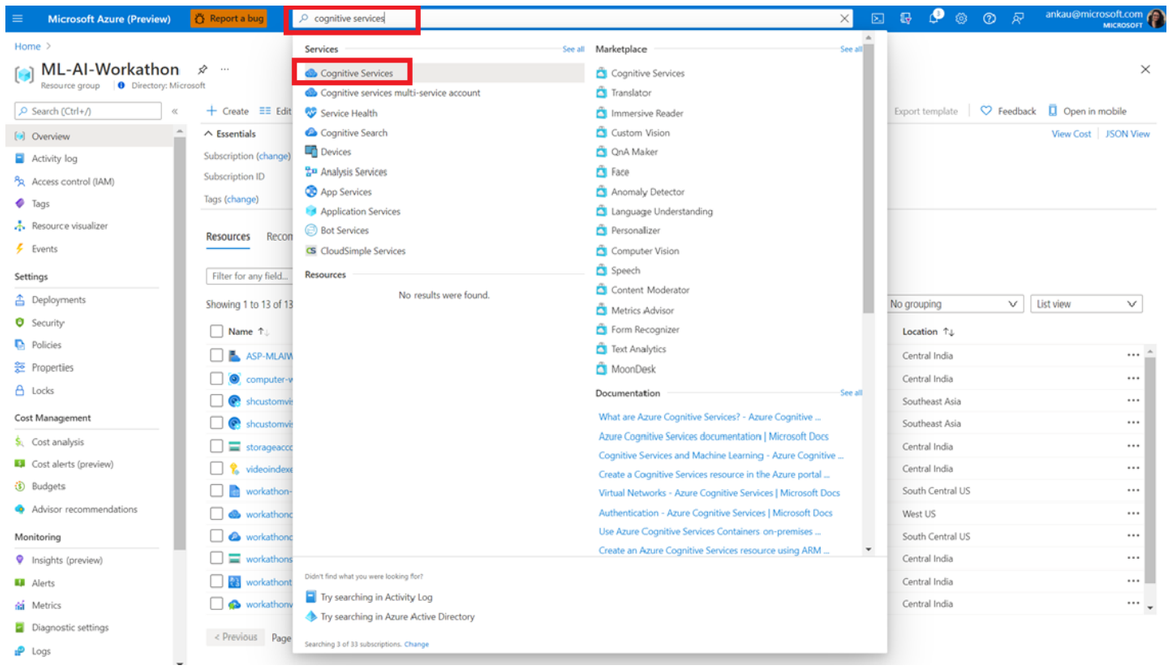
Screenshots	Steps & Significance
	Sign into your Azure Portal.

<table>
<tr>
<td> 
</td>
<td><u>Create a Resource Group</u>

Follow steps 1 & 2 to create a resource group.

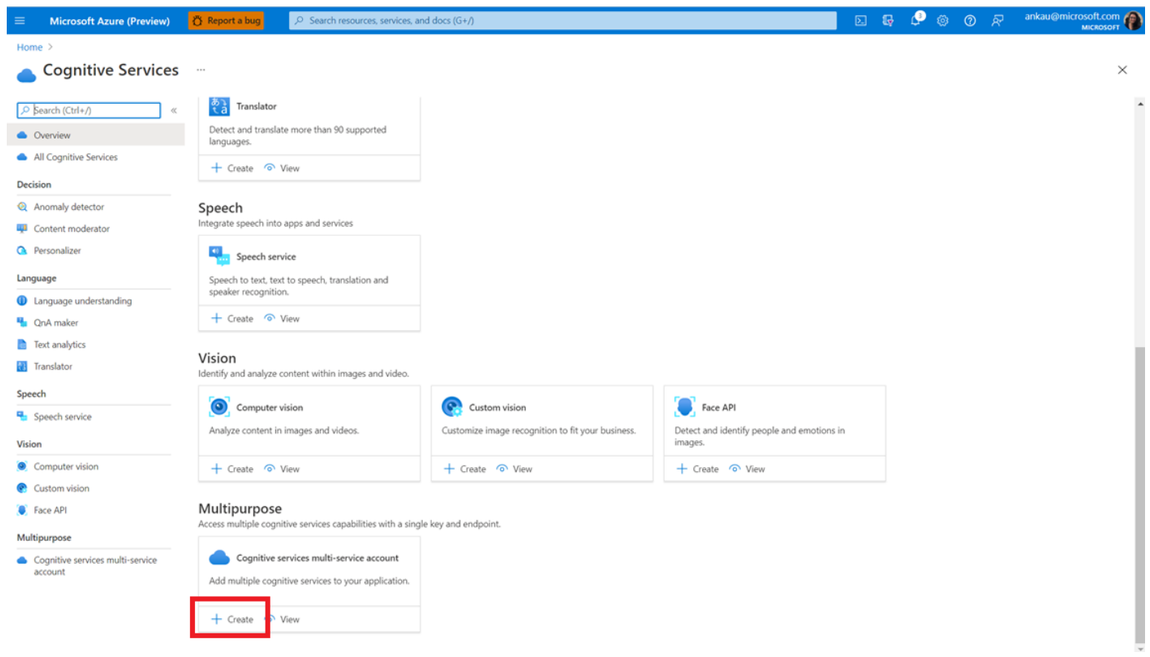
You can skip this step if you already have a Resource Group in place.</td>
</tr>
<tr>
<td> 
</td>
<td>Click create to create a new resource group.</td>
</tr>
<tr>
<td> 
</td>
<td>Enter the details –
1. Subscription : Azure subscription in which you want to deploy the resource group
2. Resource Group : Name of your choice for the resource group
3. Region : Region where you want to deploy the resource group

Click Review + Create.</td>
</tr>
</table>



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 Form Recognizer. This can be a multipurpose Azure Cognitive Resource or a Form Recognizer Resource.

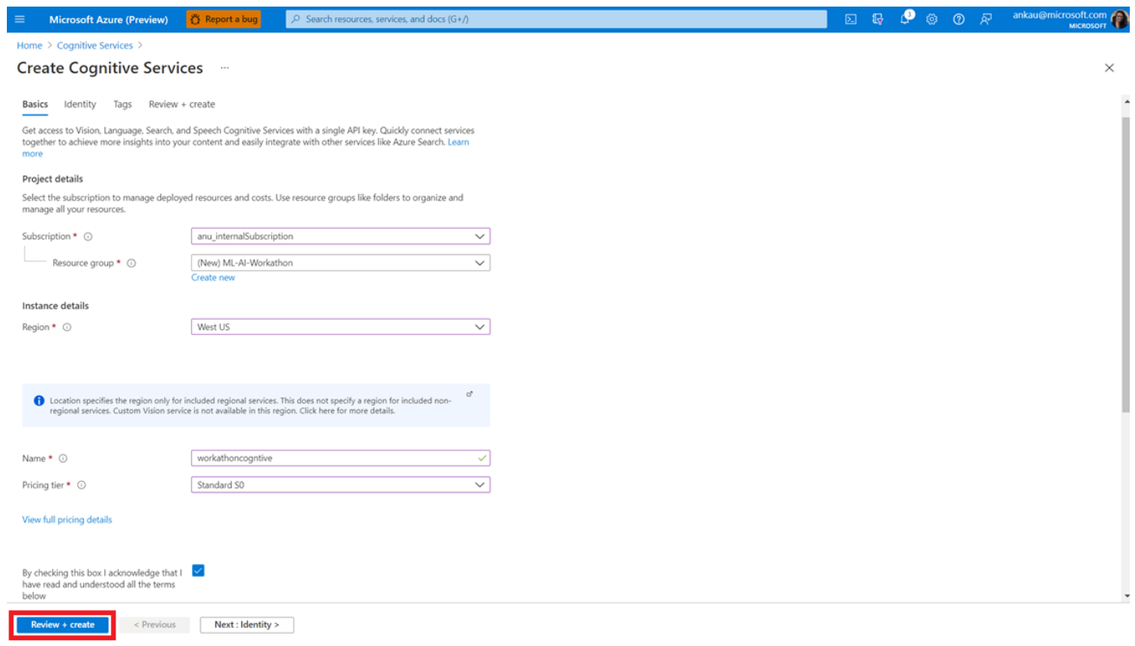


<u>Create a multipurpose cognitive service</u>

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

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

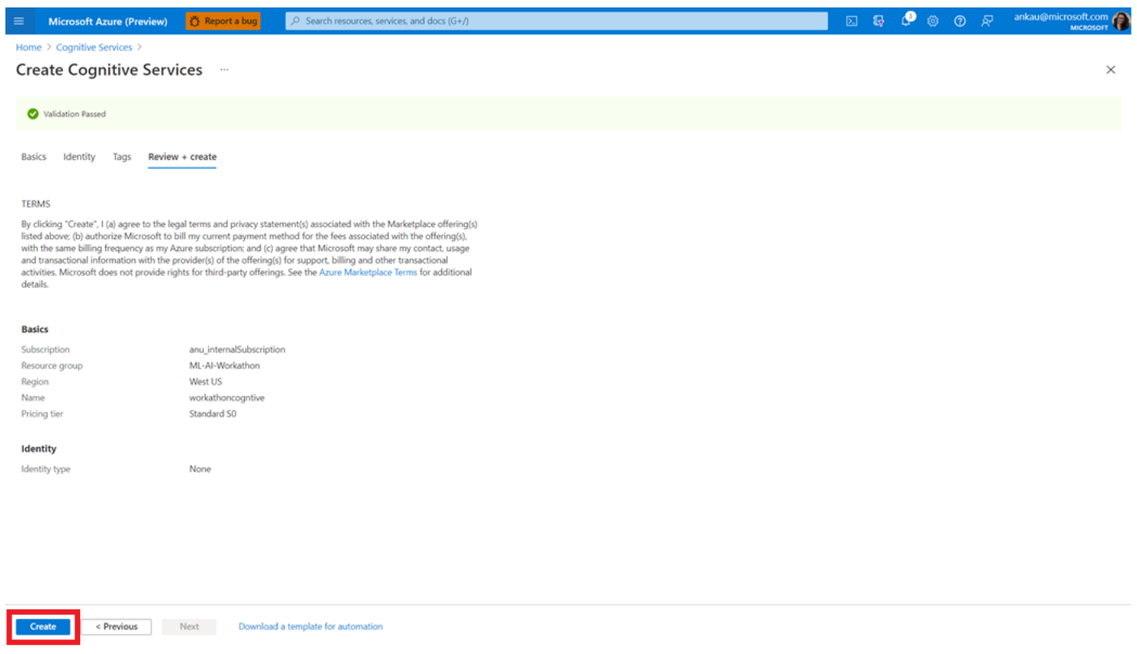
In this lab, we used a multipurpose Cognitive Service account since we would be learning about all the above-mentioned services.
However, you can also spin up individual services to execute these labs or for your development / production scenarios. The only difference is spinning up individual services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



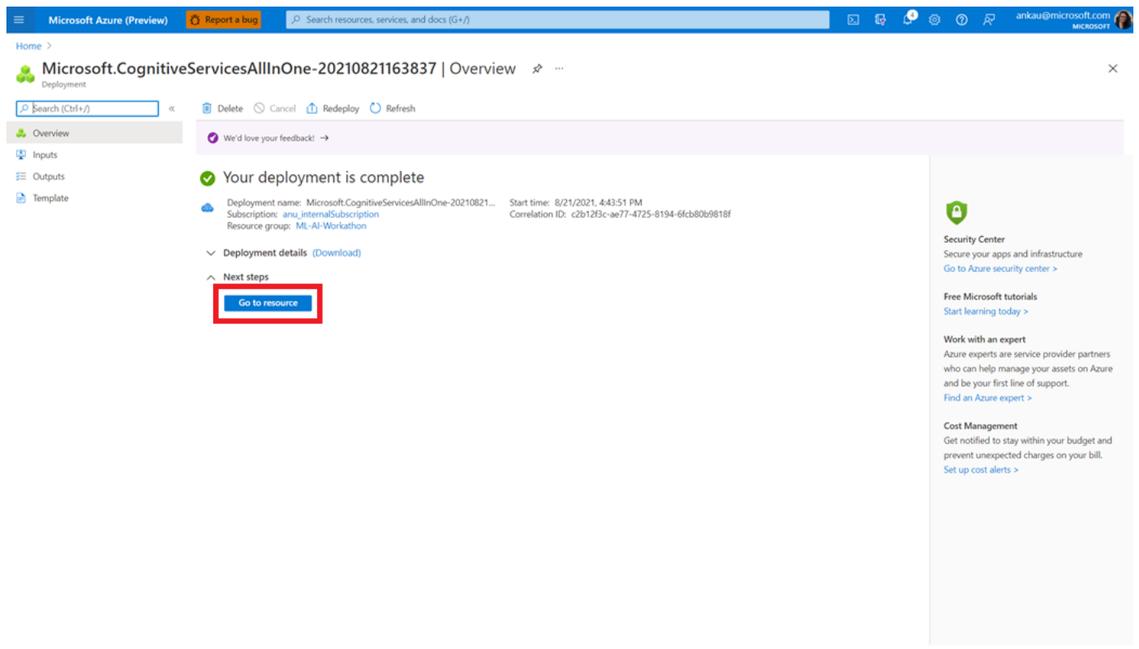
Enter the details to create a new cognitive service as follows -

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.

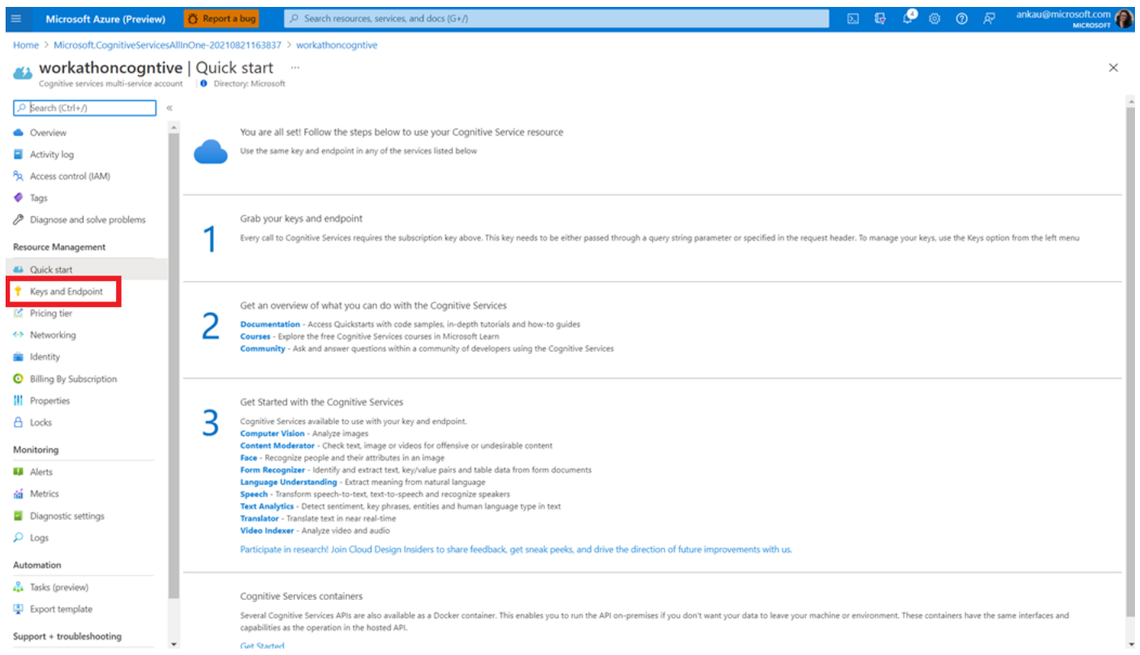
Click Review + Create.



Verify the details and click Create.



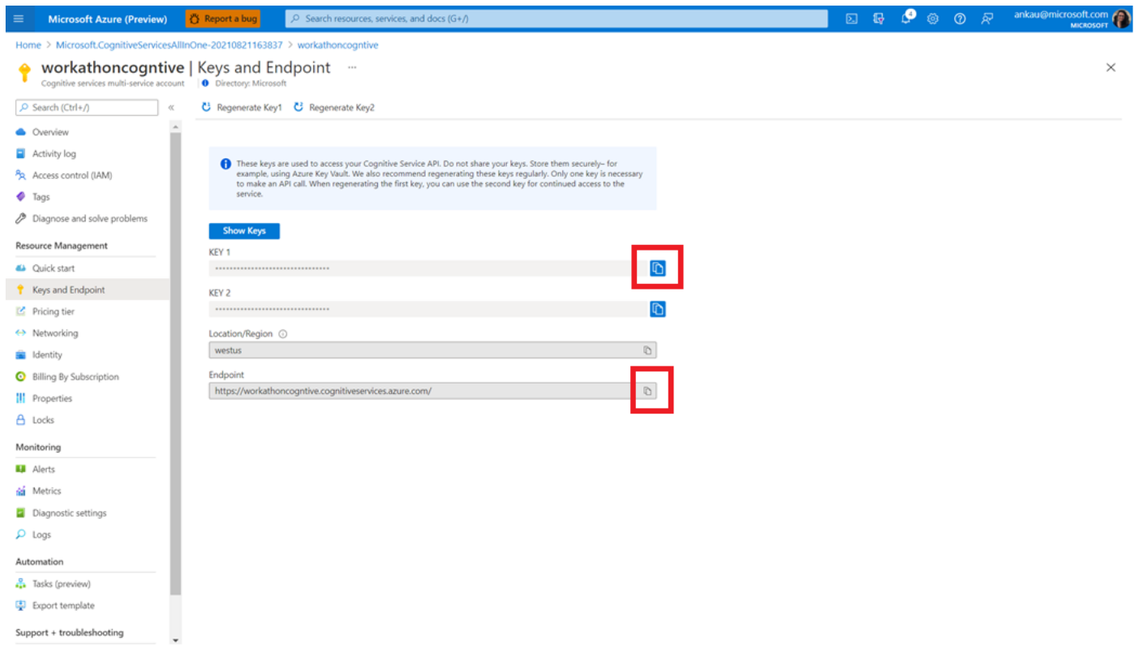
After the resource has been deployed, click Go to Resource.



<u>Copy keys & endpoints</u>

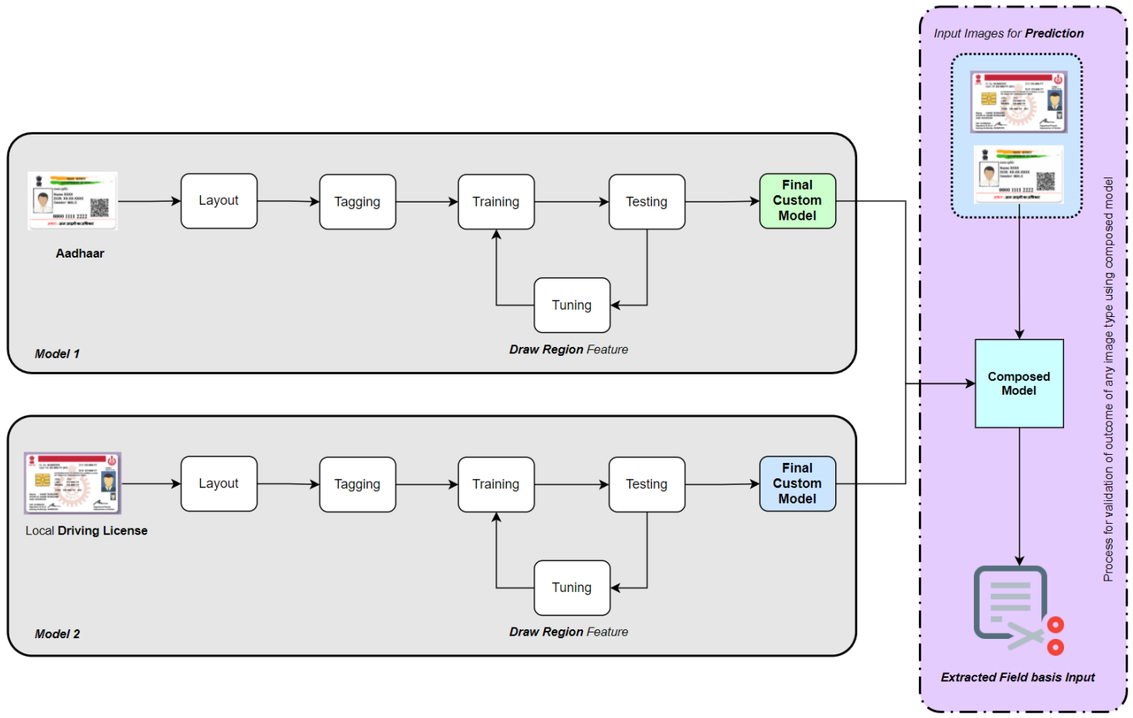
On the Quick start page, you can find details about different cognitive services and can click the hyperlinks to learn more.

Click Key and Endpoints.

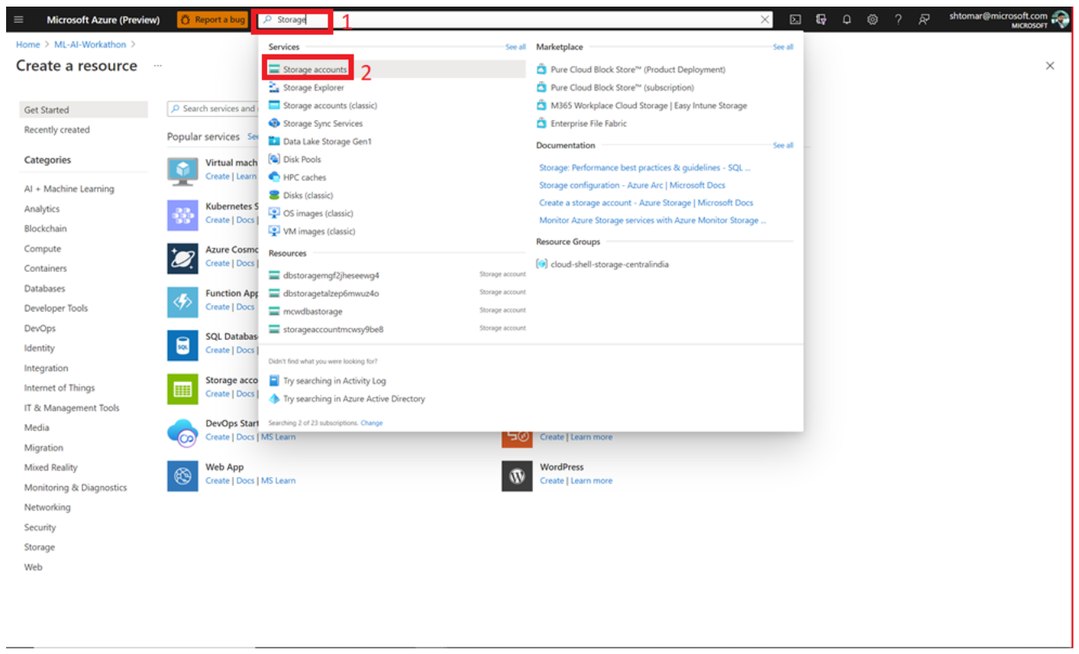


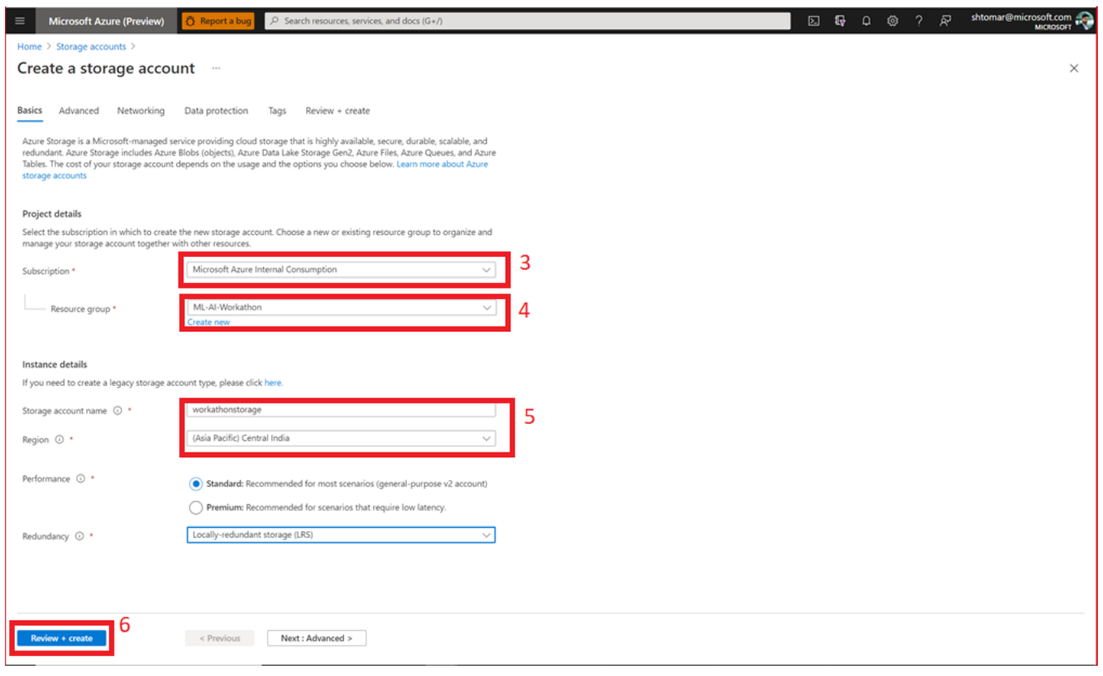
Copy the Key and Endpoint. Paste these in a notepad. You will leverage these at a later step, while creating a custom Form Recognizer project & setting up the global variables in Postman.

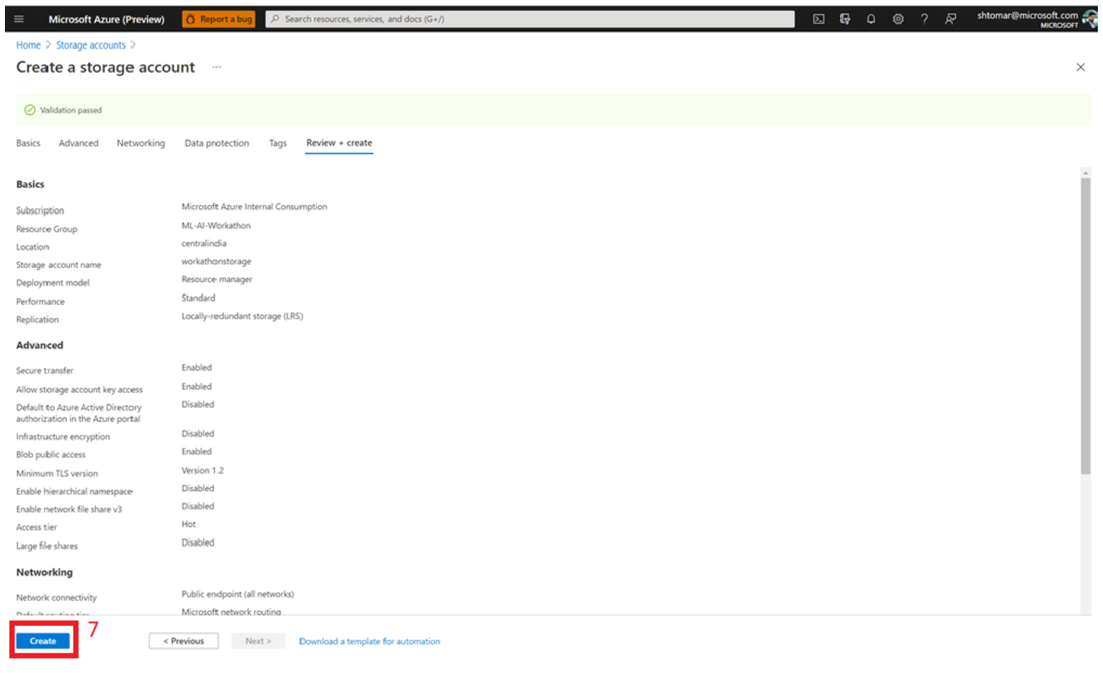
Workshop Workflow for Composed Model



In this workshop, we will build 2 custom models using local Identity card – Indian Aadhaar Card & Delhi State Driver's Licence. We will then use these custom models to create a Composed Model, which will act as sacrosanct single model for processing the 2 ID Cards, to generate outcomes specific to either card.







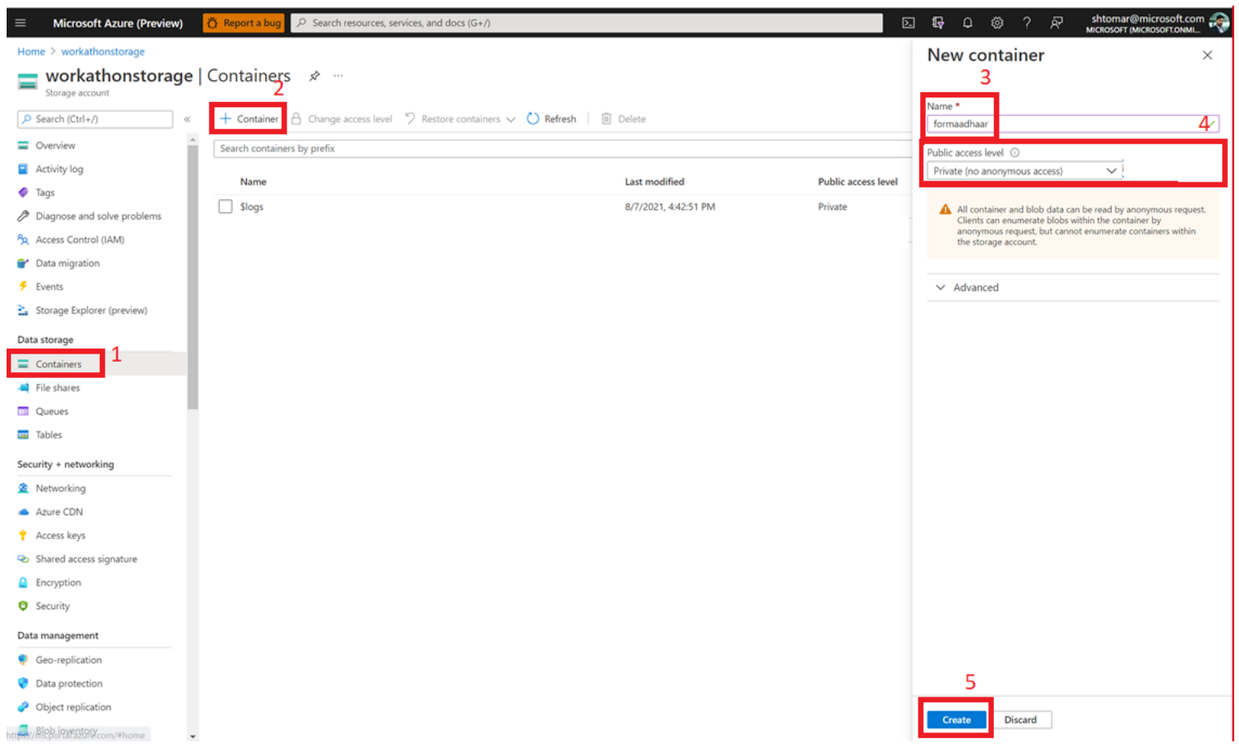
Create Storage Account in Azure Portal

This will be used to upload the Identity Cards used to train custom ID model.
If you already have a Storage Account in place, you can skip to Create blob container step.

Search for Storage Account in the search bar and select Storage account resource. (Step 1,2)

Enter the details highlighted in the screenshot. (step 3-5)
Click Review + Create. (step 6)

Verify the details and hit Create. (step 7)



Create blob container 1 (formaadhaar)

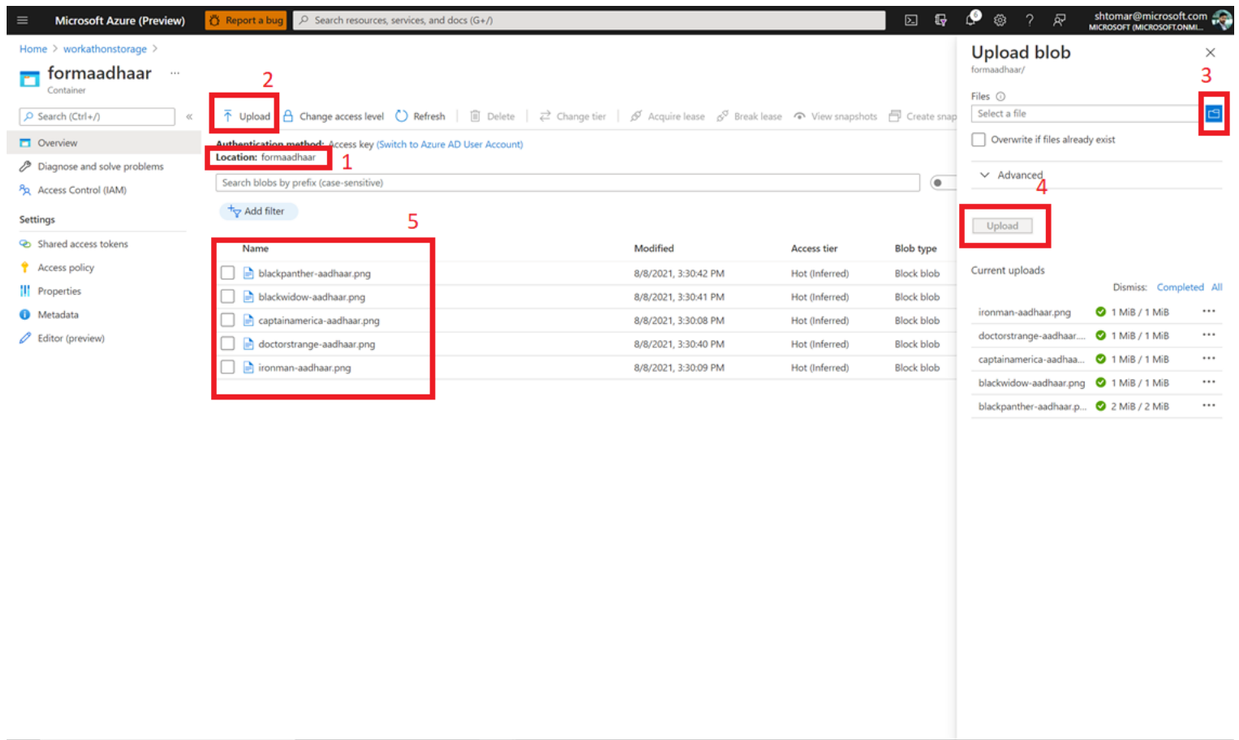
This container will be used to store the training data for the first document type : Aadhaar Card.

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

Create a container with the following details –

1. Name : **formaadhaar** [You can choose a different name, however, make sure to make all the subsequent name changes]
2. Public access level : Private (no anonymous access)

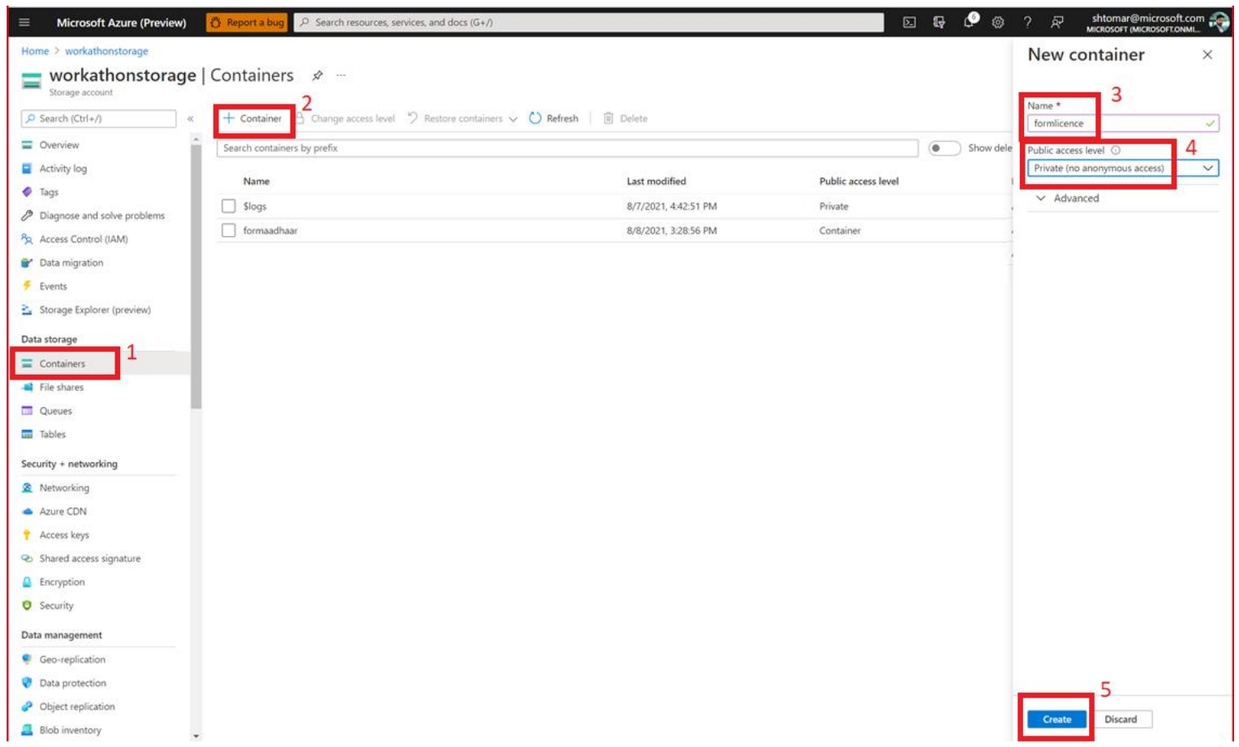
Click Create.



Upload documents

Make sure you are in the **formaadhaar** container (step 1)
Select Upload (step 2)
Click the file icon (step 3) and browse & select the images you want to upload for ID Card 1 (Avengers Aadhaar)
Click Upload button (step 4)

You should see the files uploaded as shown in step 5.

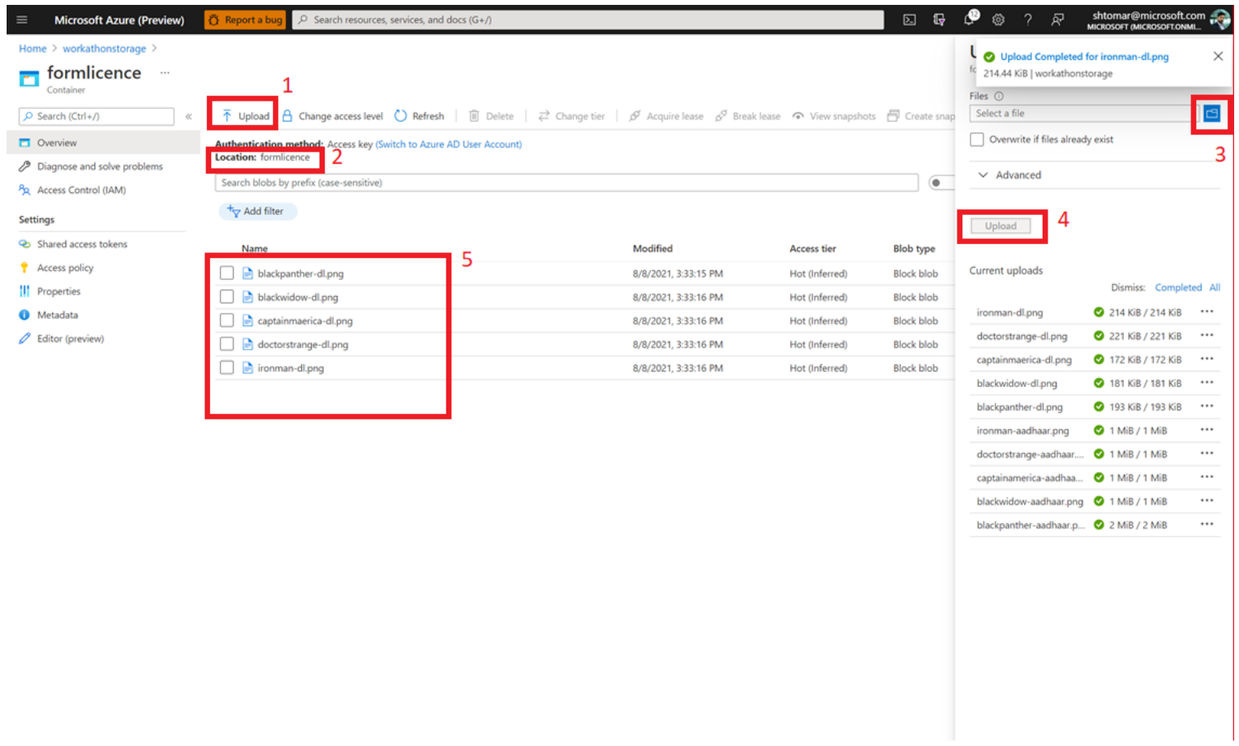


Similarly, create another container for licence cards by following the steps as highlighted.

Create a container with the following details –

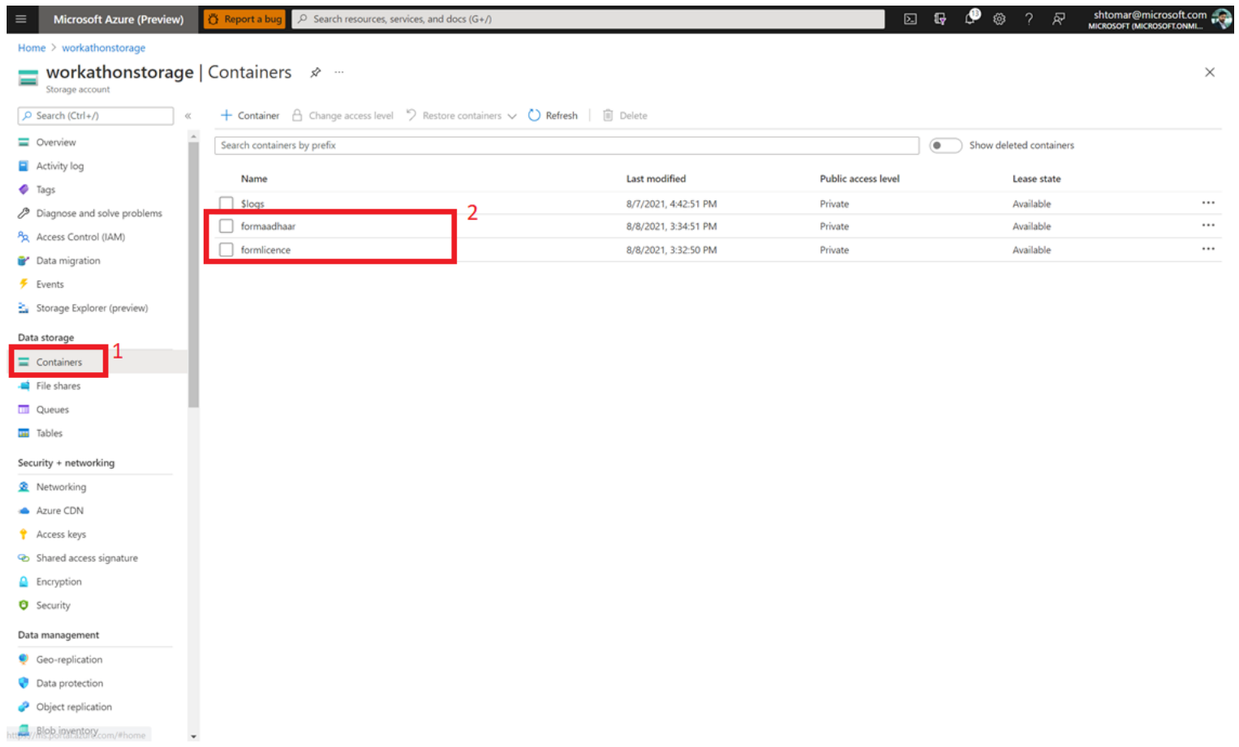
1. Name : **formlicence** [You can choose a different name, however, make sure to make all the subsequent name changes]
2. Public access level : Private (no anonymous access)

Click Create.

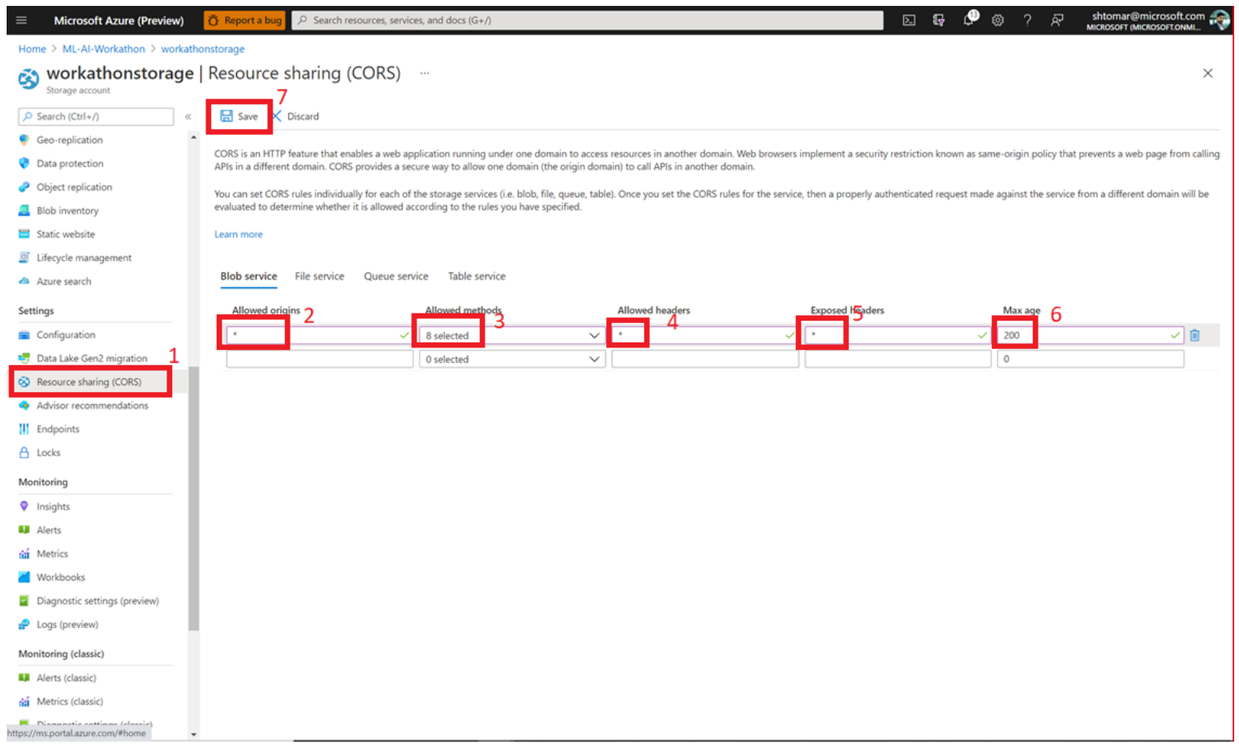


Make sure you are in the **formlicence** container (step 1)
Select Upload (step 2)
Click the file icon (step 3) and browse & select the images you want to upload for ID Card 2 (Avengers Licence)
Click Upload button (step 4)

You should see the files uploaded as shown in step 5.



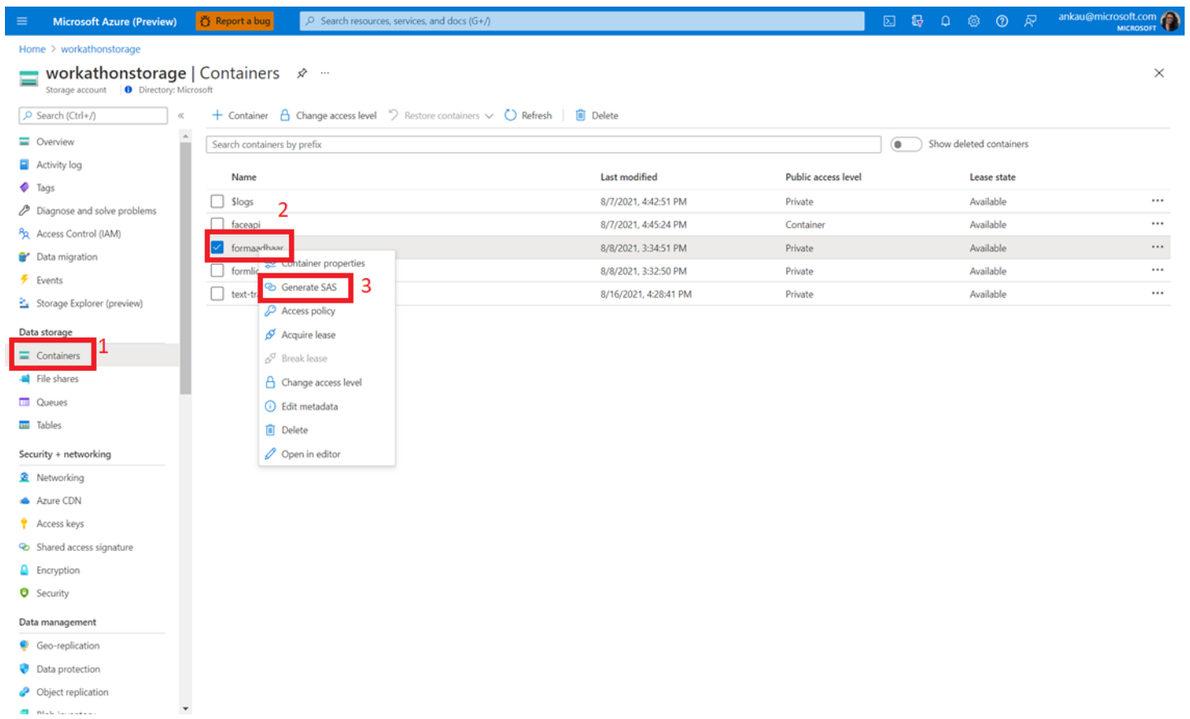
Once you have created both containers & uploaded the files, you should see 2 containers containing 5 files each.

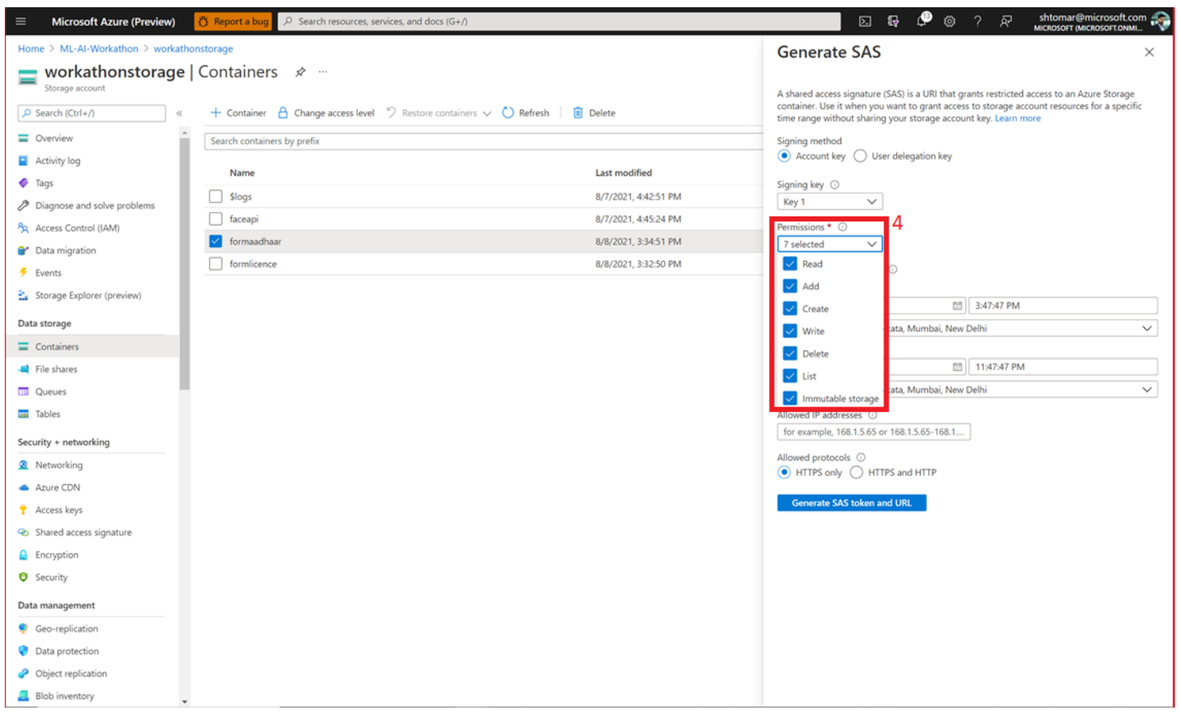


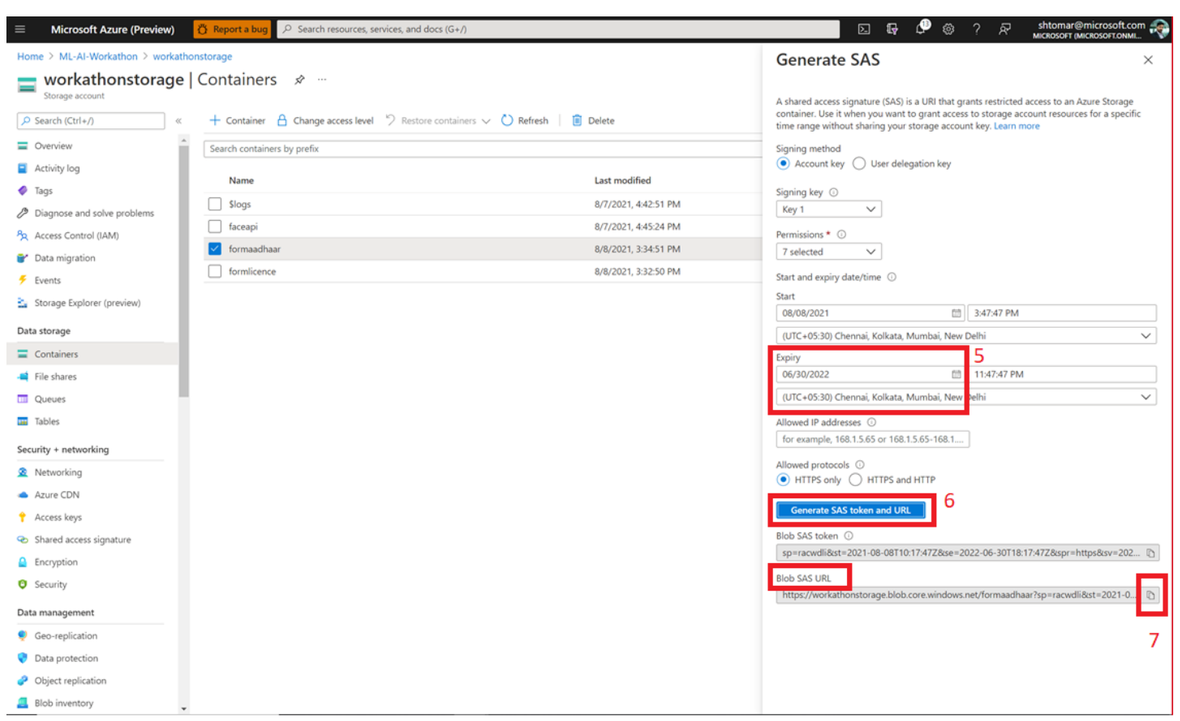
<u>Setup CORS for permission</u>

We will set up CORS permission to allow access to the blob storage from another domain & external source, such as Form Recognizer client & Postman.

Follow the steps as highlighted and enter the values as shown.







Get SAS URI of the blob containers

A shared access signature (SAS) provides secure delegated access to resources in your storage account. With a SAS, you have granular control over how a client can access your data. For example:

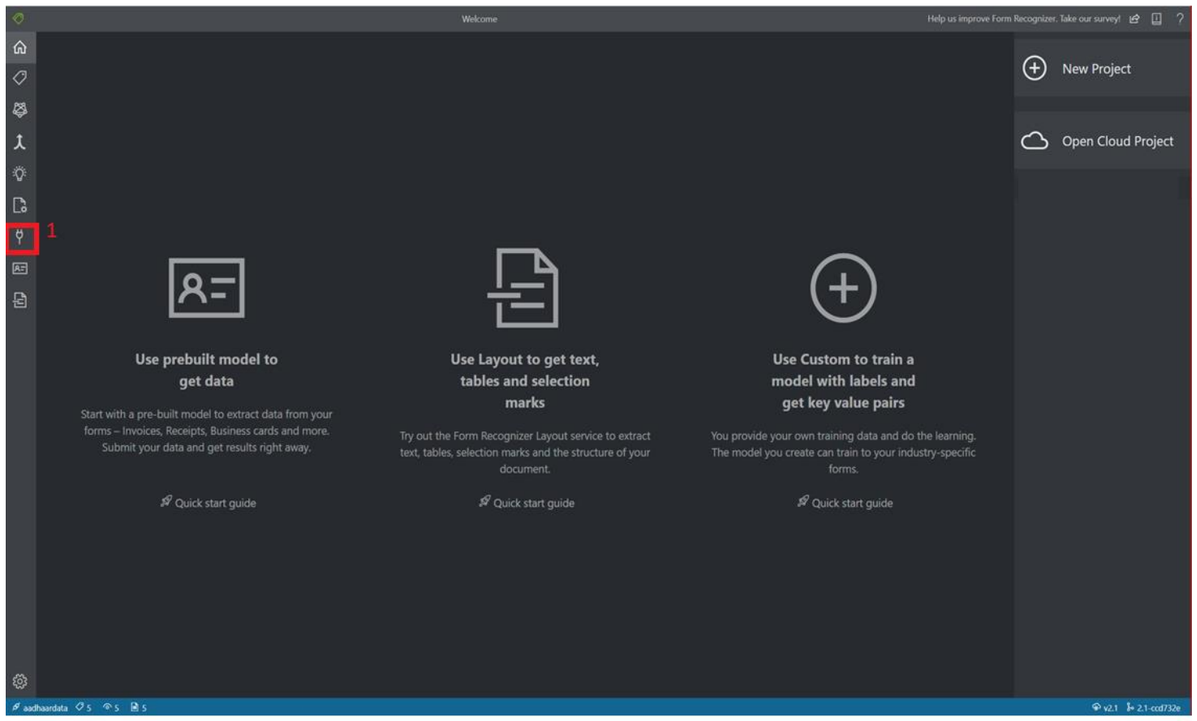
a. What resources the client may access
b. What permissions they have to those resources.
c. How long the SAS is valid

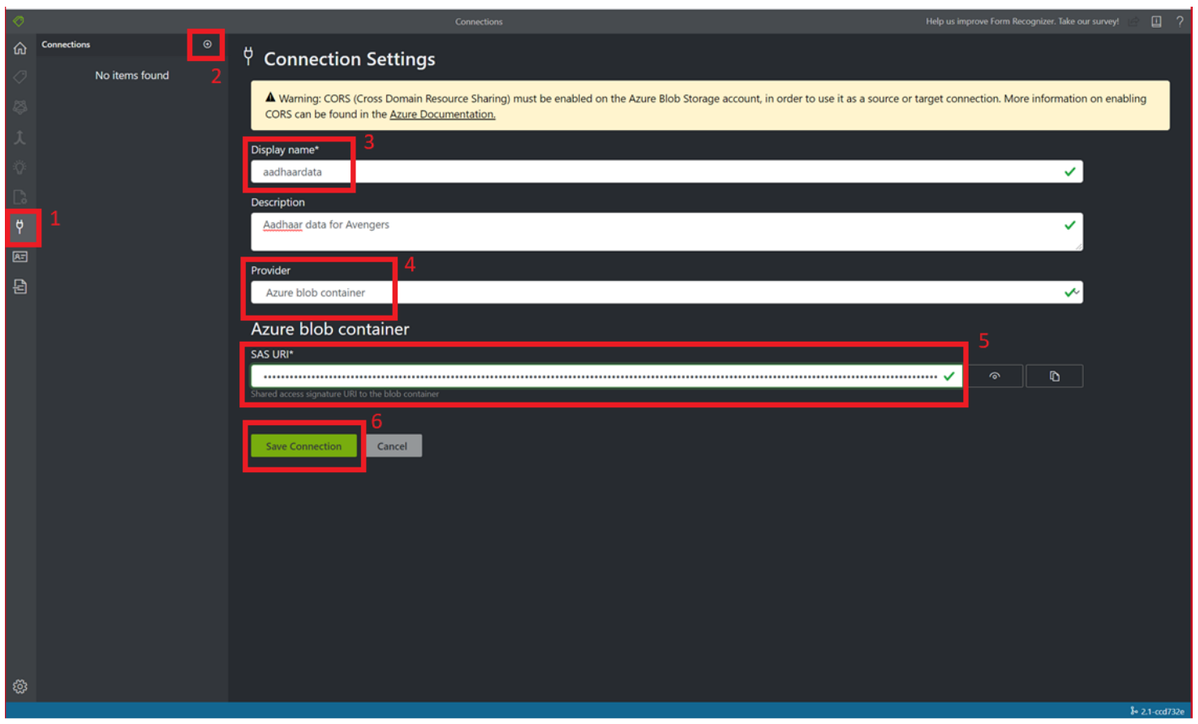
We require a SAS URI to secure access to the blob storage containers from Form Recognizer client.

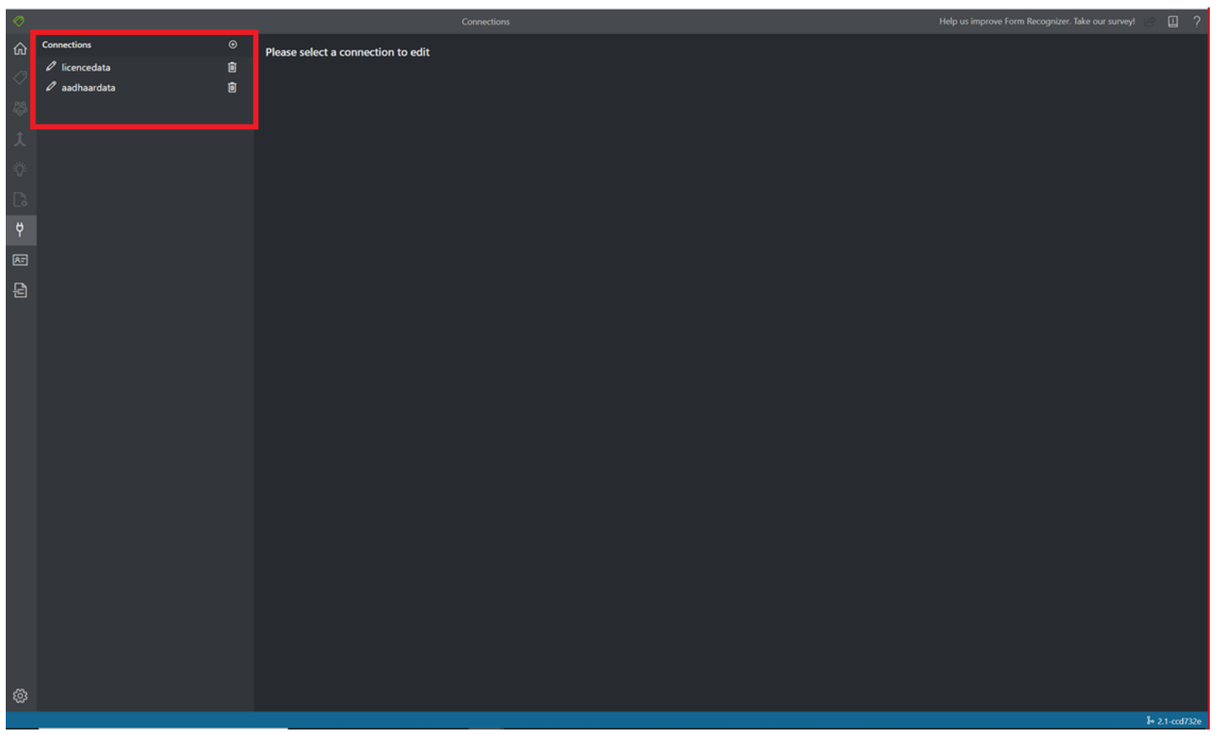
1. In the containers section, right click on formaadhaar container & select Generate SAS (steps 1-3)
2. Leave the signing method & Signing key to defaults
3. For Permissions select all (Step 4) **
4. Set up the SAS Expiry (Make sure to set this up to a later date since you will have to generate a new SAS key & change it in the Form Recognizer model to still access the model in case the keys expire) (step 5)
5. Click Generate SAS URL & Token (step 6)
6. Copy the Blob SAS URL and paste it in a notepad. We will leverage this at a later step

Similarly, obtain the Blob SAS URI for formlicence container.

** To maintain access with least privilege you might as well select only Read, Write, Delete and List. This workshop is not considering security set up as this is designed to familiarise you with service capabilities.
We will be sharing the security best practices in a separate document.







Create Custom Model

Jump to the Form Recognizer GUI Tool, to build a custom model.

Using Azure Form Recognizer GUI Tool, you can test out pre-built models or create your custom models. In this lab, we will be creating a custom model. We suggest you try out the pre-built models on your own.

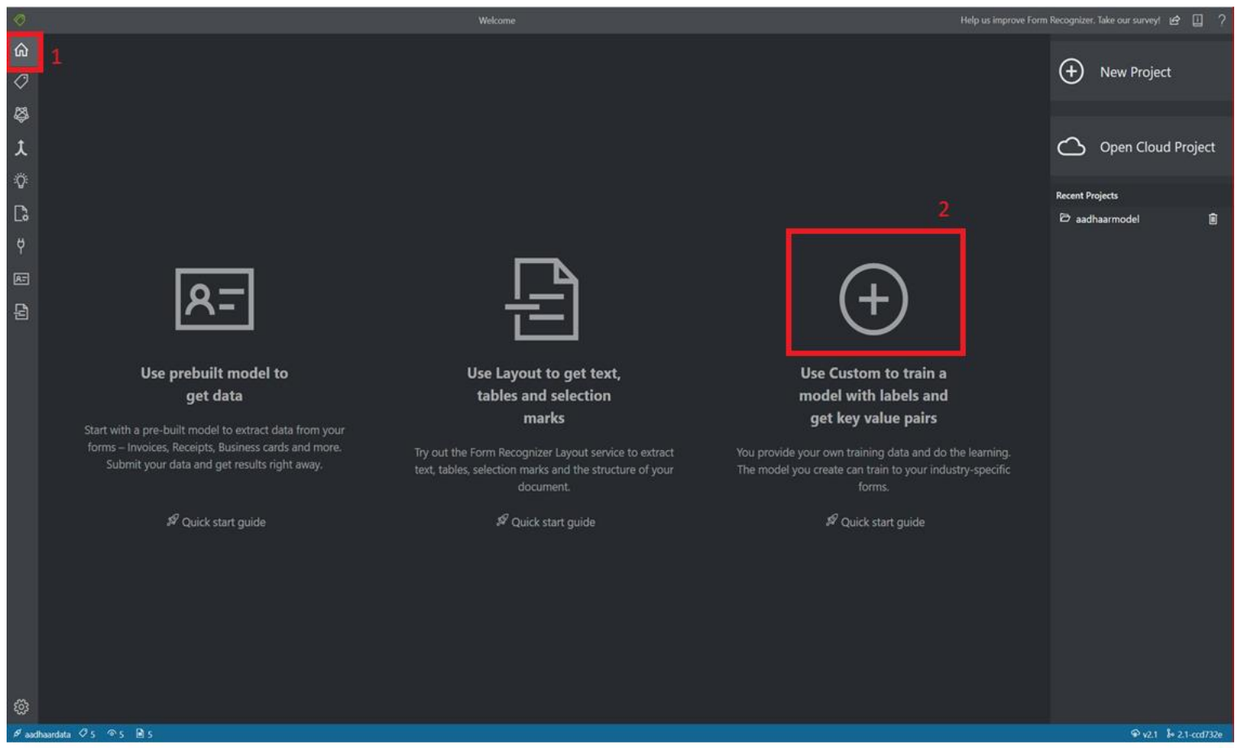
Create connections to Blob containers

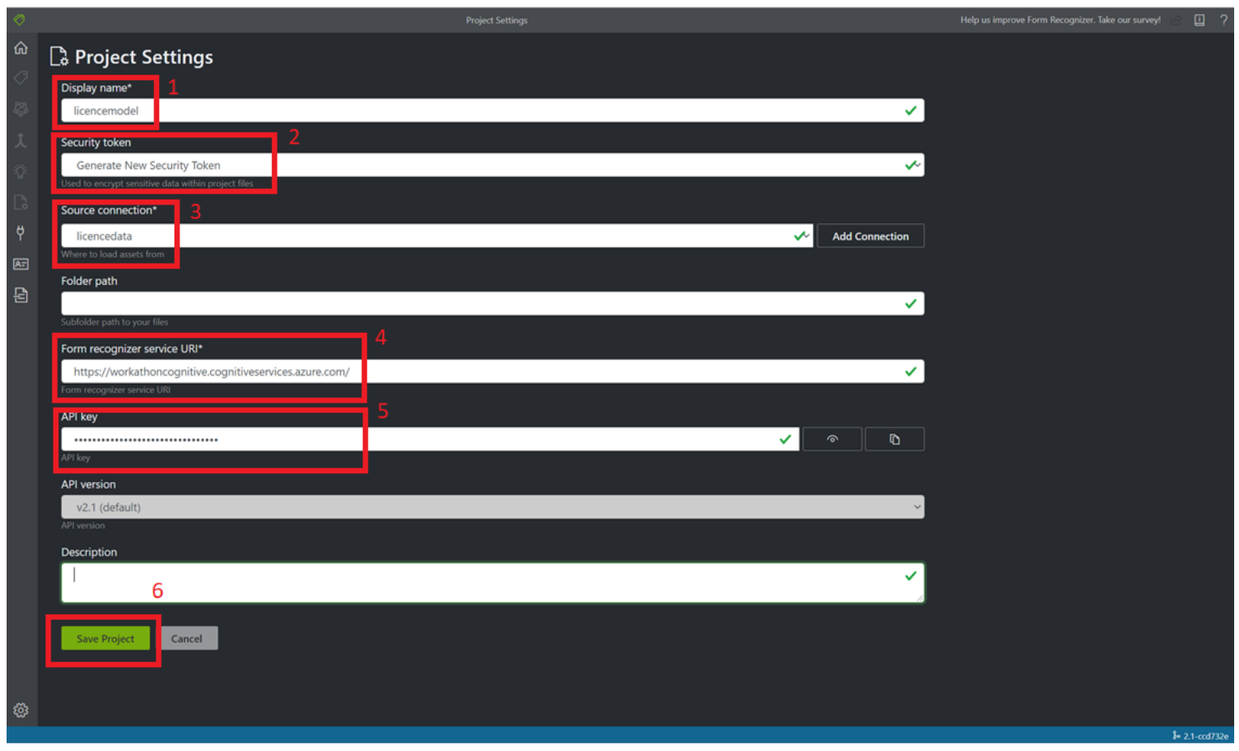
First, we will create a connection to the 2 blob containers we created, using their respective SAS URLs.

1. Select the tab as shown in step 1. This brings you to the connections page.
2. Select + (step 2) to add a new connection setting.
3. Display Name : aadhaardata. Optionally add a description to help you identify the connection.
4. Provider : Azure blob container
5. SAS URI : Paste the SAS URI you copied above for formaadhaar container.
6. Click Save connection.

Following the same steps, create another connection to formlicence container. Make sure to use the SAS URL generated for the formlicence container.

Once you have created both the connections, you should see a view with 2 connections, as shown in the last image in this step.





<u>Create a project to build custom model</u>

A project helps you manage you custom models. You can also share the project with other collaborators. We will see the steps to share a project later in the workshop.

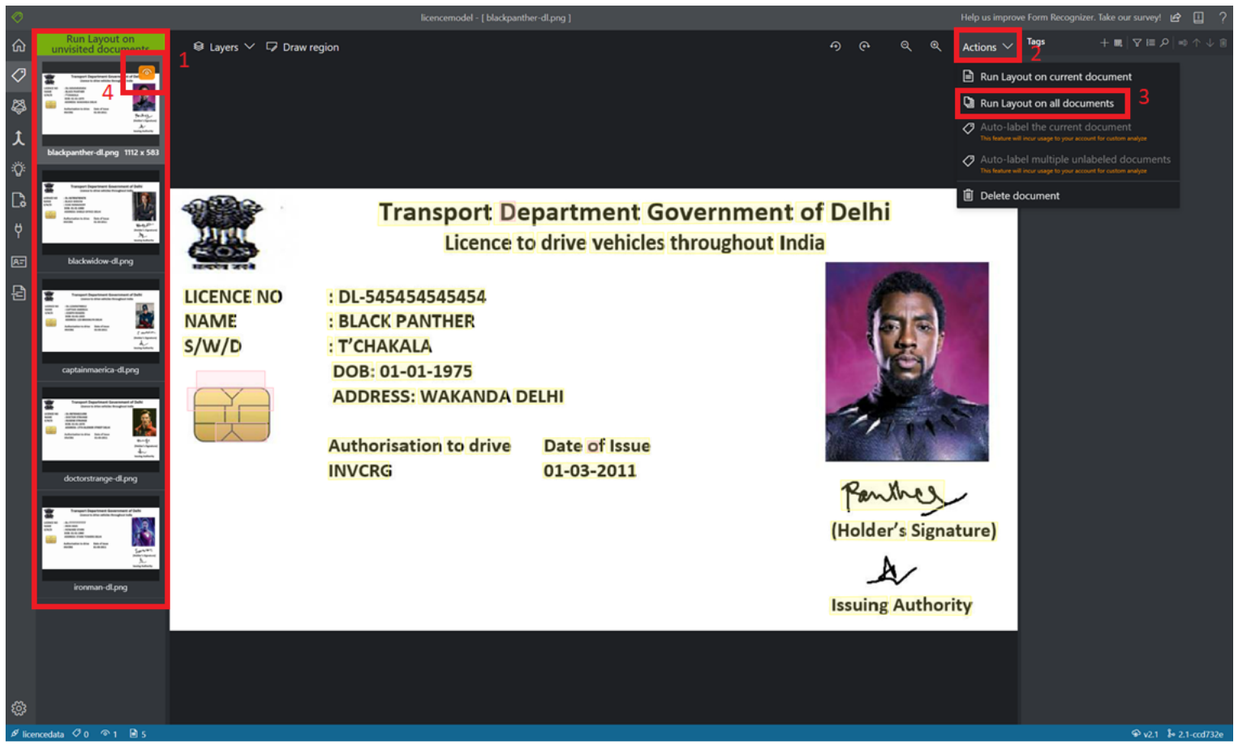
We will now start building the custom ID model for our use case.
We will be creating 2 projects : **licencemodel** & **aadhaarmodel** and then compose them to a single model.

<u>Let's start by creating the first project : licencemodel</u>

Select the home tab. (step 1)
Select + to create a new Custom model. (Step 2)

This will open the Project Settings page. Enter the details as follows –

1. Display name : **licencemodel**
2. Security Token : A new token will be generated by default. We will be using this token share models for collaboration & re-accessing the projects later, more on this later!
3. Source connection : From the drop down, select the connection created for formlicence container in the previous step.
4. Form recognizer service URI : Enter the Endpoint for the cognitive service we provisioned at the beginning of the workshop
5. API Key : Enter the Key for the cognitive service we provisioned at the beginning of the workshop
6. Click Save Project.



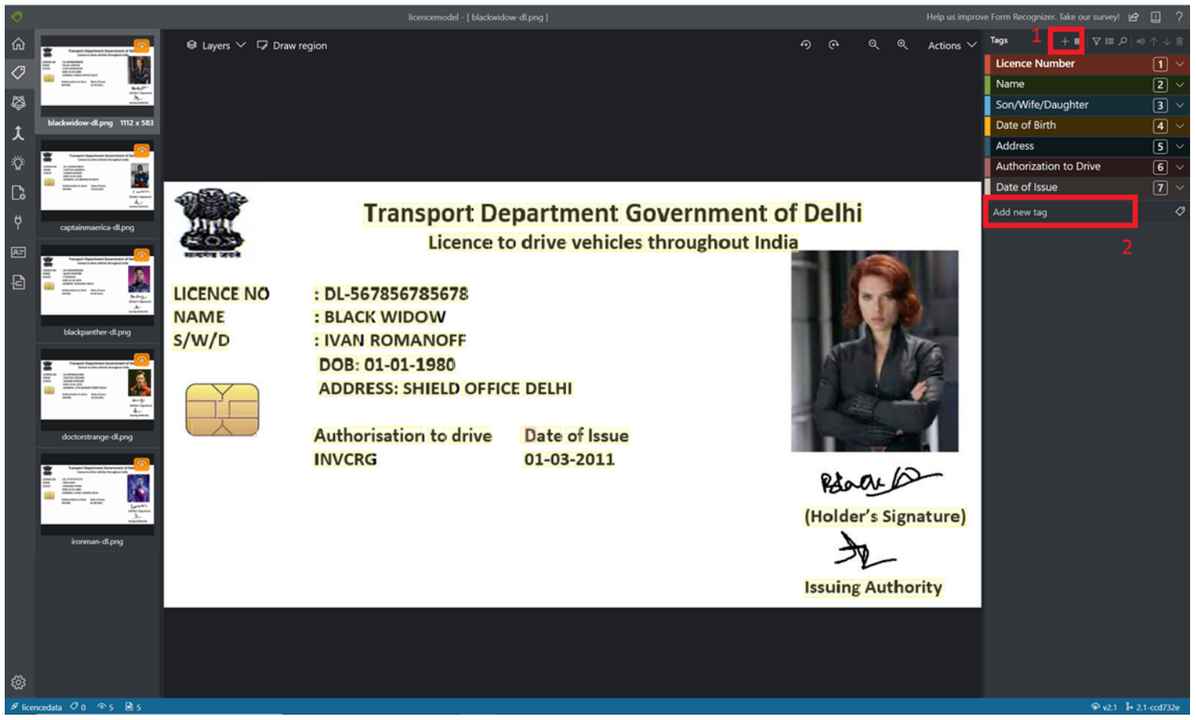
Custom Labelling & Training

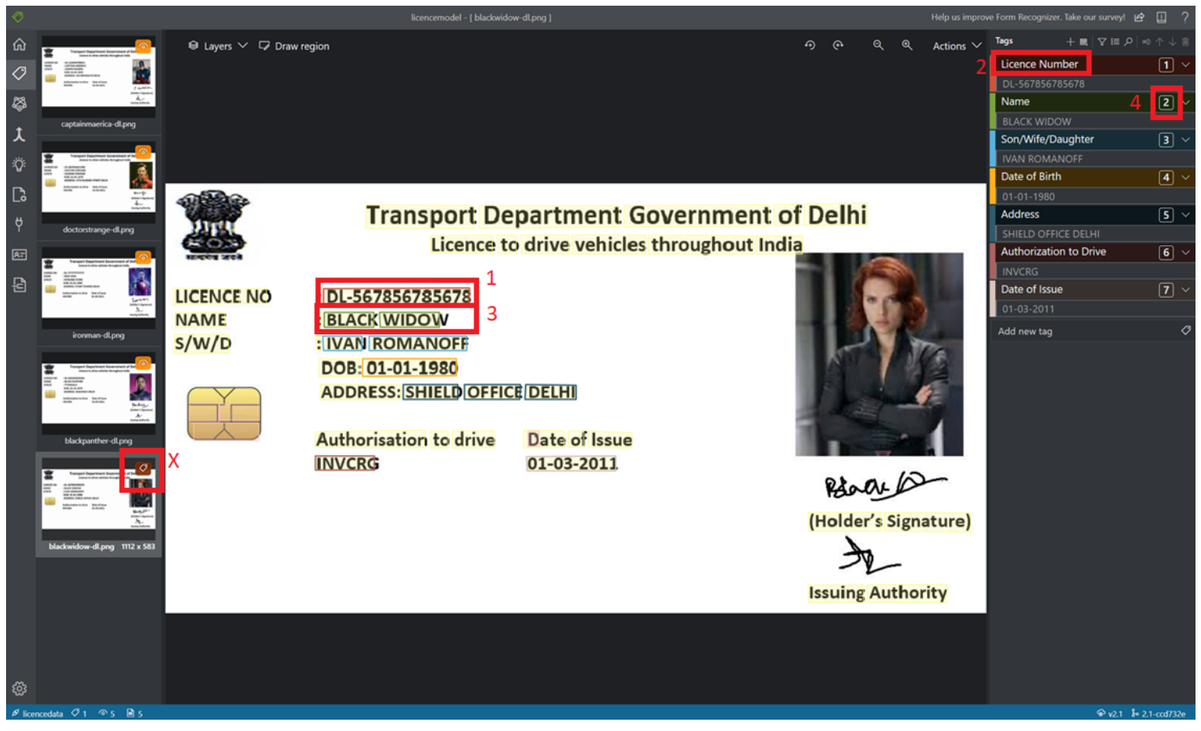
Once you have created a new project, you should get a similar view, with all the images displayed on the left of the screen(step 1). If the images are not being rendered or you are getting an error, re-check the steps performed by far : generating SAS URL, creating connections to both blob containers, creating a new project by leveraging blob connection & cognitive resource key & endpoint.

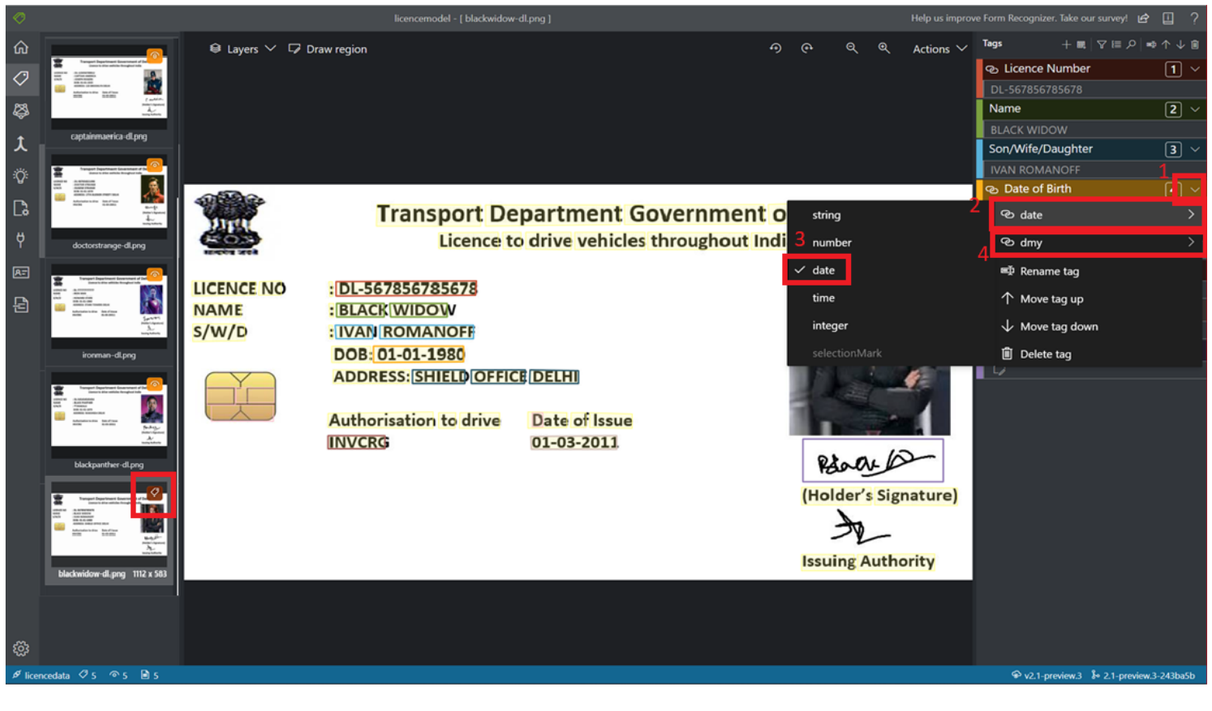
Extracting document layouts

Layout feature extracts text, selection marks and tables structures, along with their bounding box coordinates, from the documents.
It will draw bounding boxes around each text element. This is helpful when we label the fields in the next step.

Select Actions drop down(step 2).
Select Run Layout on all documents (Step 3).
You will see yellow eye like icon (step 4) on all the images once step 3 is complete.







Adding Custom Tags to model

We will now create our custom tags and tag the content in the ID cards to their respective fields & tags.

Select the + icon (step 1)
Type the name of the tag (step 2)
Repeat the above steps to create multiple tags. Once you have the required tags, you will get a view like image 1.

Tagging ID Card fields to tags

Click on the Licence No value (step 1), then click on Licence Number (step 2).
You will notice the Licence No value is displayed under Licence Number tag.

Now, Black Widow (step 3) and press key 2 on keyboard (step 4).
You will notice BLACK WIDOW is displayed under Name tag.

This way, by selecting the value and clicking on the tag or pressing the corresponding tag key, you will be able to tag all the fields.
As & when you complete the image tagging, the yellow eye like icon will change to a brown tag like icon. (Step X)
Repeat this for all the images.

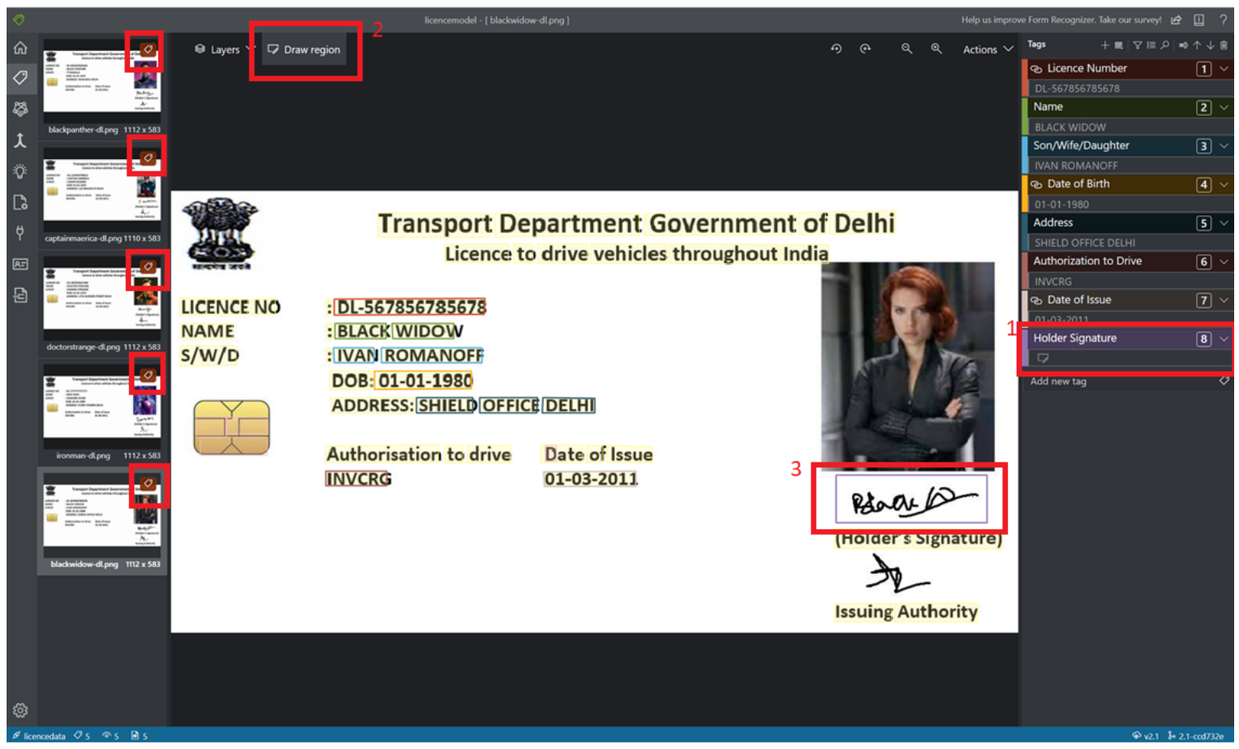
Changing Data Type for tag

There are a few different data types that you can select for each tag, like string, number, date, time etc.
Tagging each field to the corresponding data type helps increase accuracy since the model is better able to identify the field.

We will change the data type for Date of Birth tag.

Follow steps 1 through 4 in image 3, to change the data type to Date and format to dmy.
Similarly, change data type for Date of Issue Tag.

Explore the different data types and check if you can replace it for other tags.

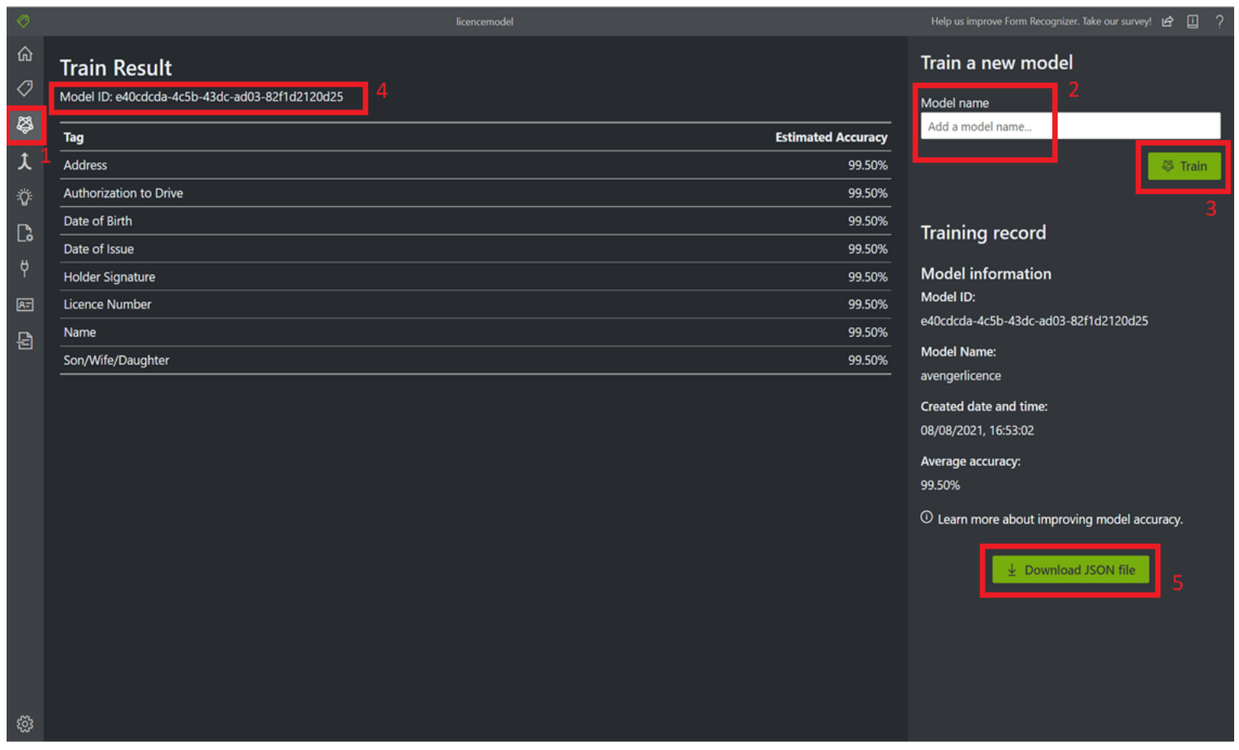


Draw Region feature

If your training document does not have a value filled in, you can draw a box where the value should be. This also allows you to select regions in the image where the layout feature was not able to detect text or values, such as signatures ** etc.

Create a new tag – Holder Signature (step 1)
Select Draw Region tool (step 2)
Draw a bounding box around the signature (step 3)
Tag this to Holder Signature tag.

** Since this is not a signature matching tool, it will not extract the signature exactly like it is. But this can be used to fetch a binary value for whether a signature is present.
If you use bounding box in cases where text is clearly visible (printed or handwritten), it will be able to extract the text as is.



Train Model

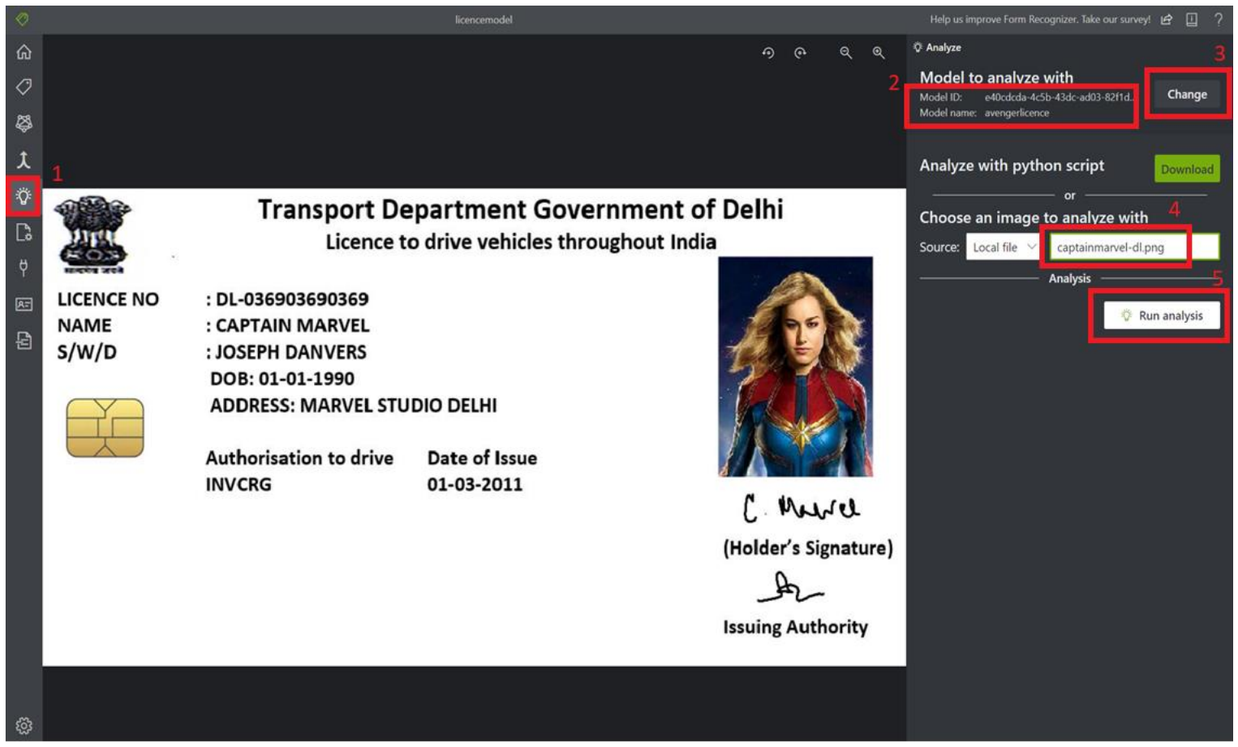
We will now train the model so we can leverage it to generate outcomes from new forms & documents.

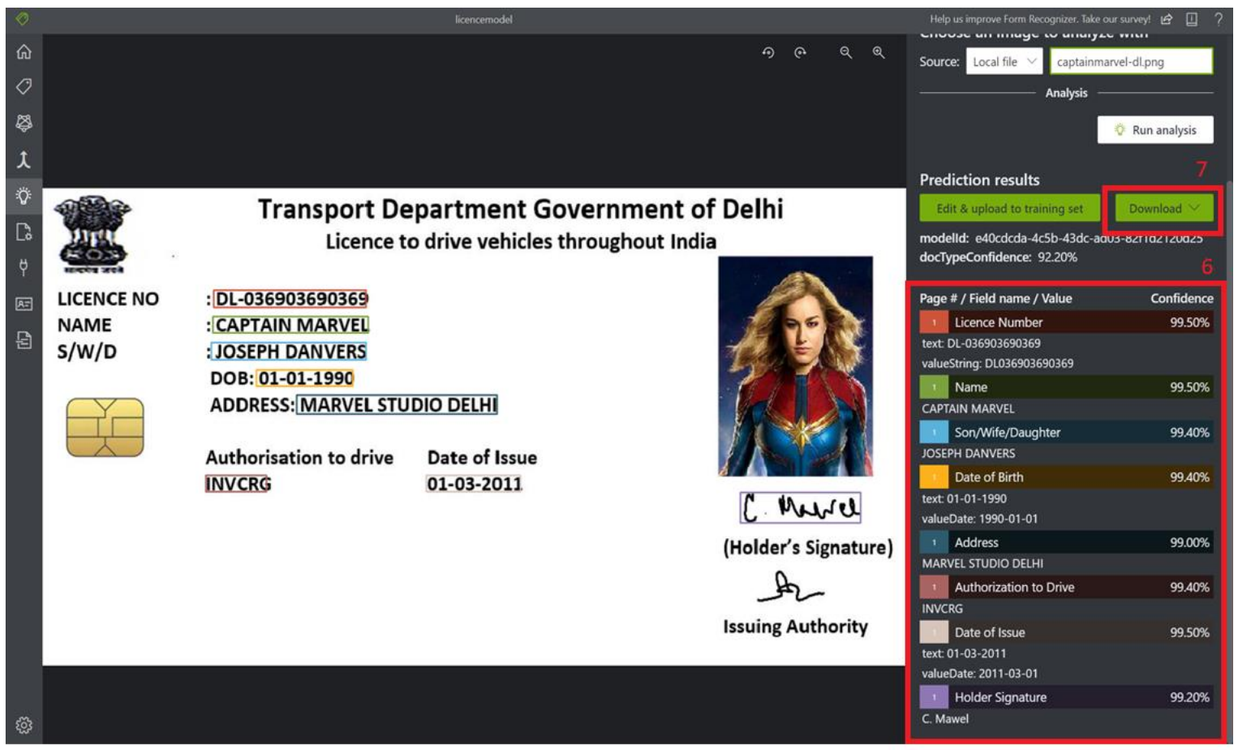
Follow the highlighted steps to train the model –

1. Select the Train model tab
2. Enter a Model Name (**avengerlicence**)
3. Click Train
4. Note & save the Model ID (we will leverage this later)

Observe the accuracy your model obtained as a whole and for each tag. Basis this, you can further decide how to tune & retrain your model.

5. Download the JSON to observe its contents



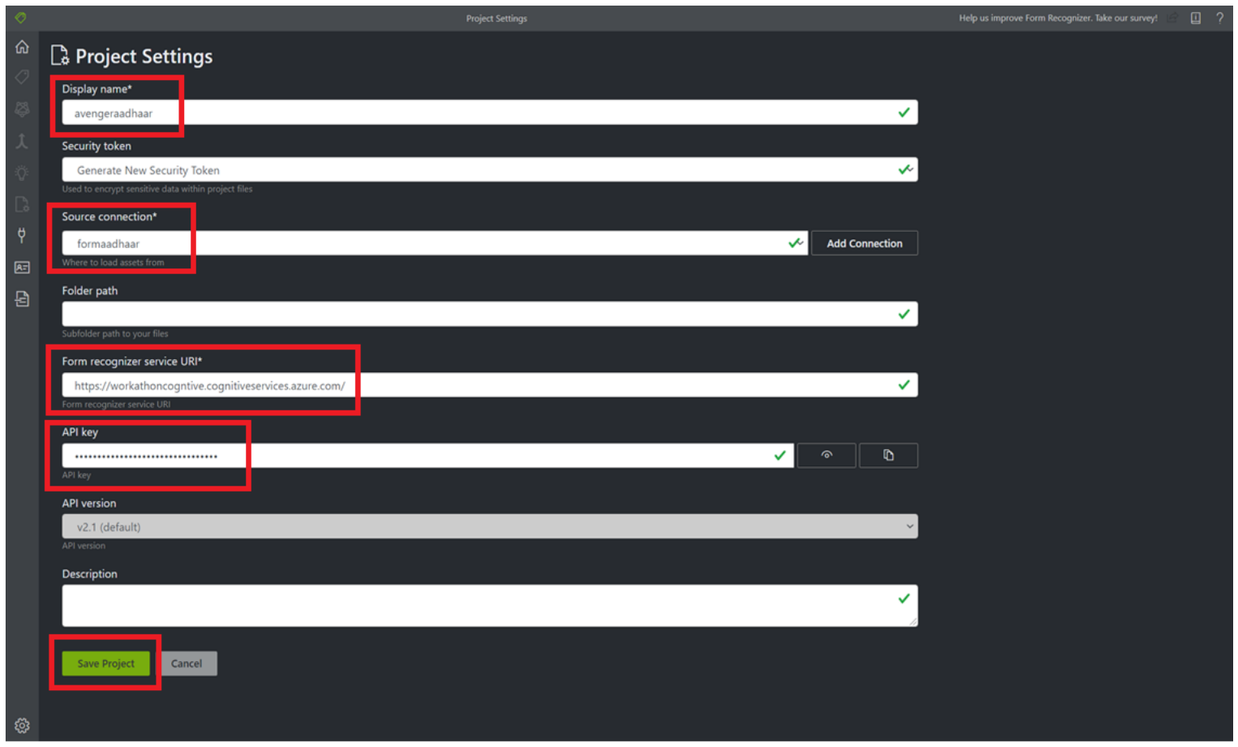


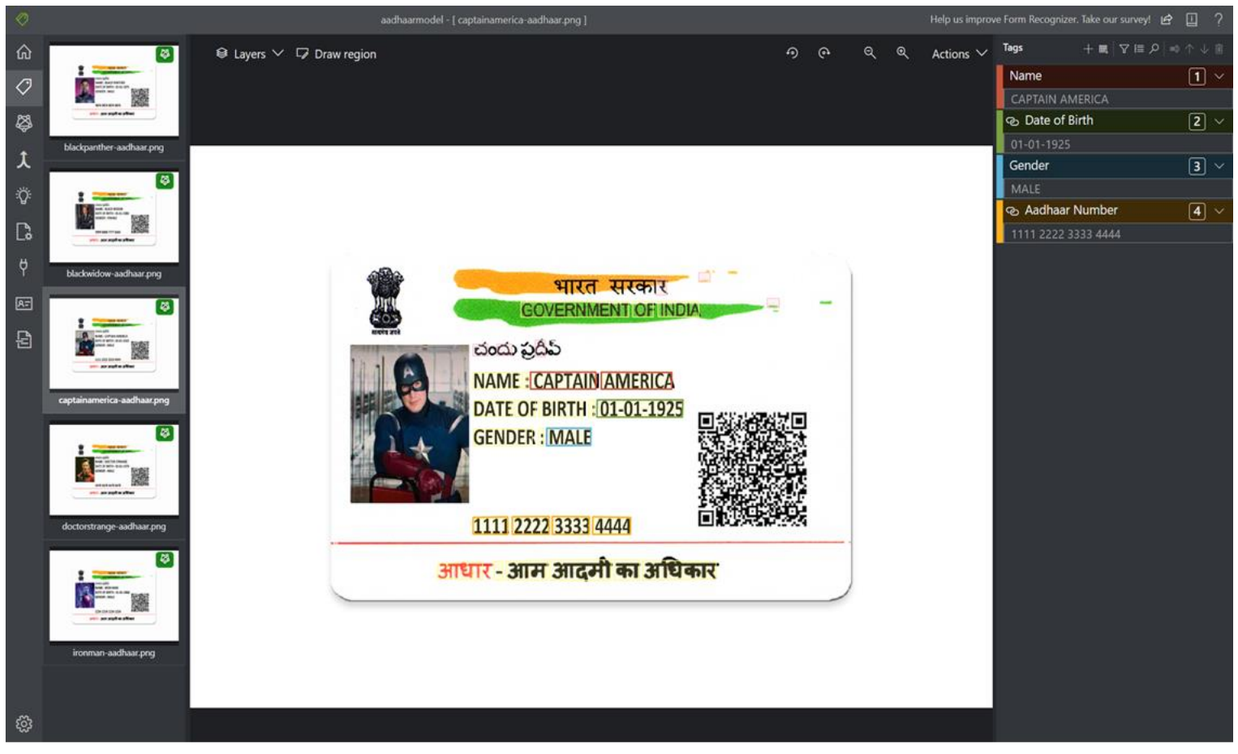
Test Model

We will now test the model we created for a new ID card.

Follow the highlighted steps to test the model –

1. Select the Test model tab
2. In case you have multiple models, make sure you have the right model selected.
3. If not, change it to select the right model by clicking on the Change button
4. Select source as Local file and browse the test image. If you want to test using an accessible image URL, change the source drop down to URL.
5. Click Run analysis
6. Observe the Field Names & values and corresponding Confidence in the result
7. You can also download the JSON & CSV to see the elaborate results for both Key value pairs extracted and the complete document layout. You can leverage the JSON for post processing bases your use case.





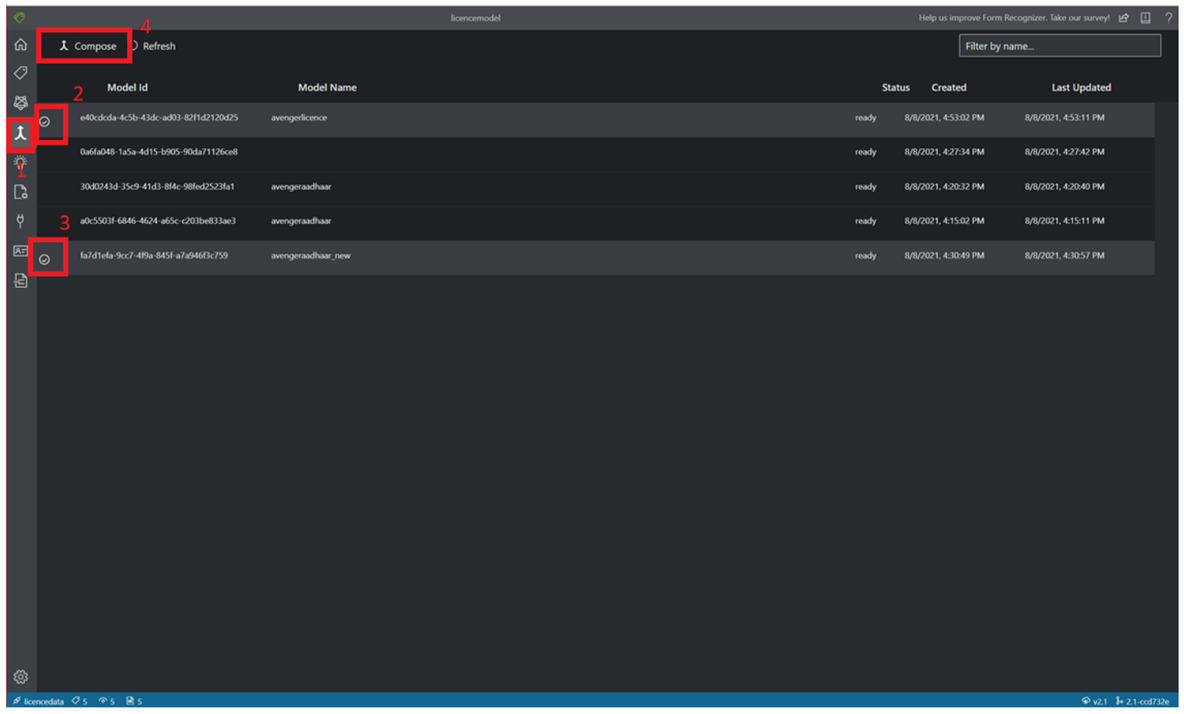
Similarly, create another model for Aadhaar Identity cards.

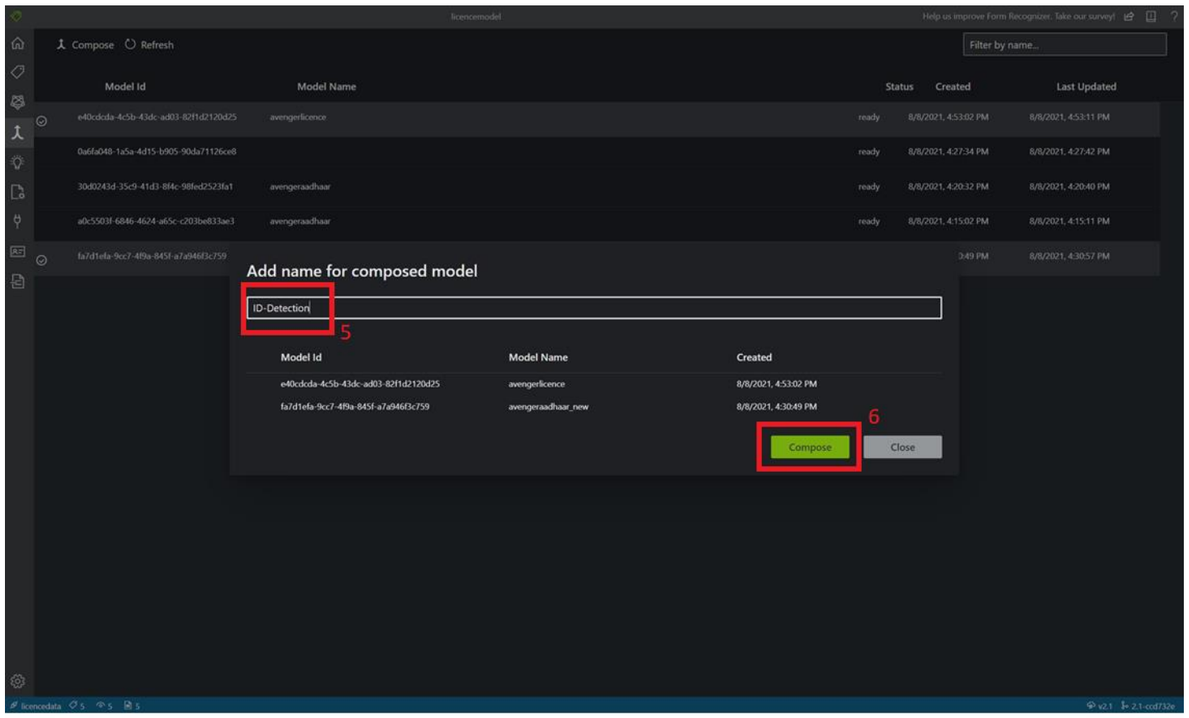
We are creating 2 models because we will be combining these 2 models, as shown in the workshop workflow diagram.

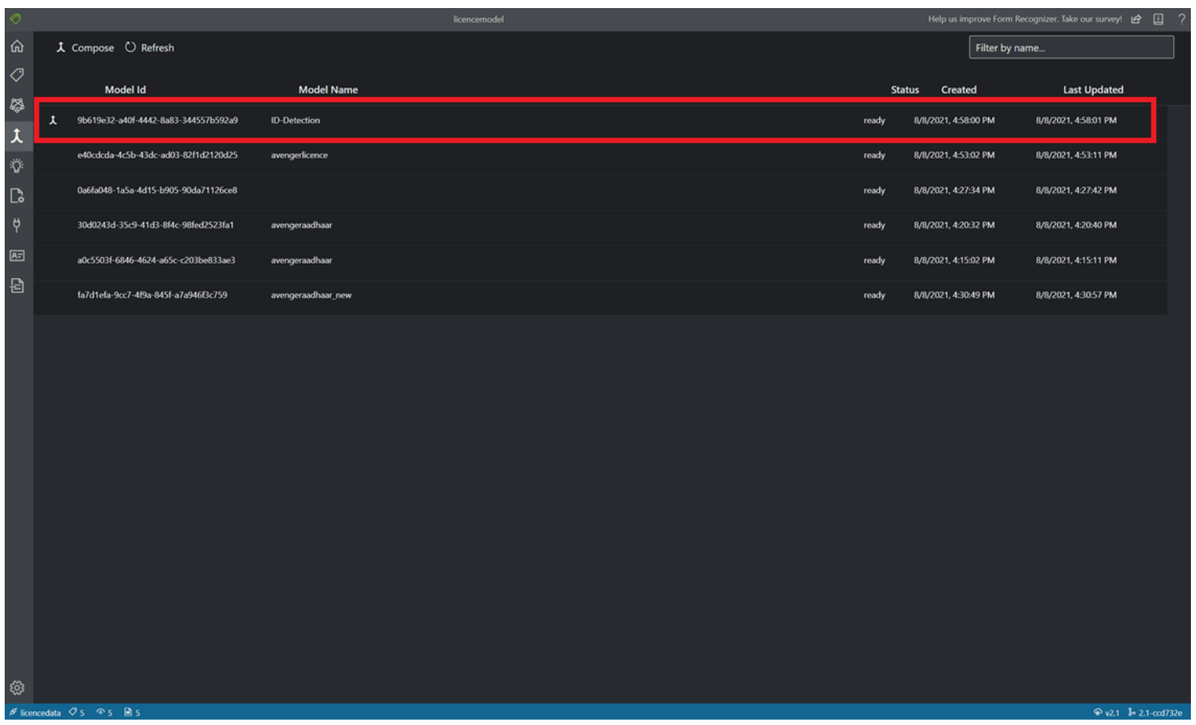
Follow the same steps for Aadhaar model.
We've already created the connection to blob storage. Leverage the 1st image to define the project settings.

Leverage the 2nd image to tag the images.

Train & Test this model.







Compose Model

We have built 2 separate models for Licence IDs & Aadhaar IDs, these 2 models are identified by their model IDs. If you want to leverage them separately, you can use the individual model IDs.

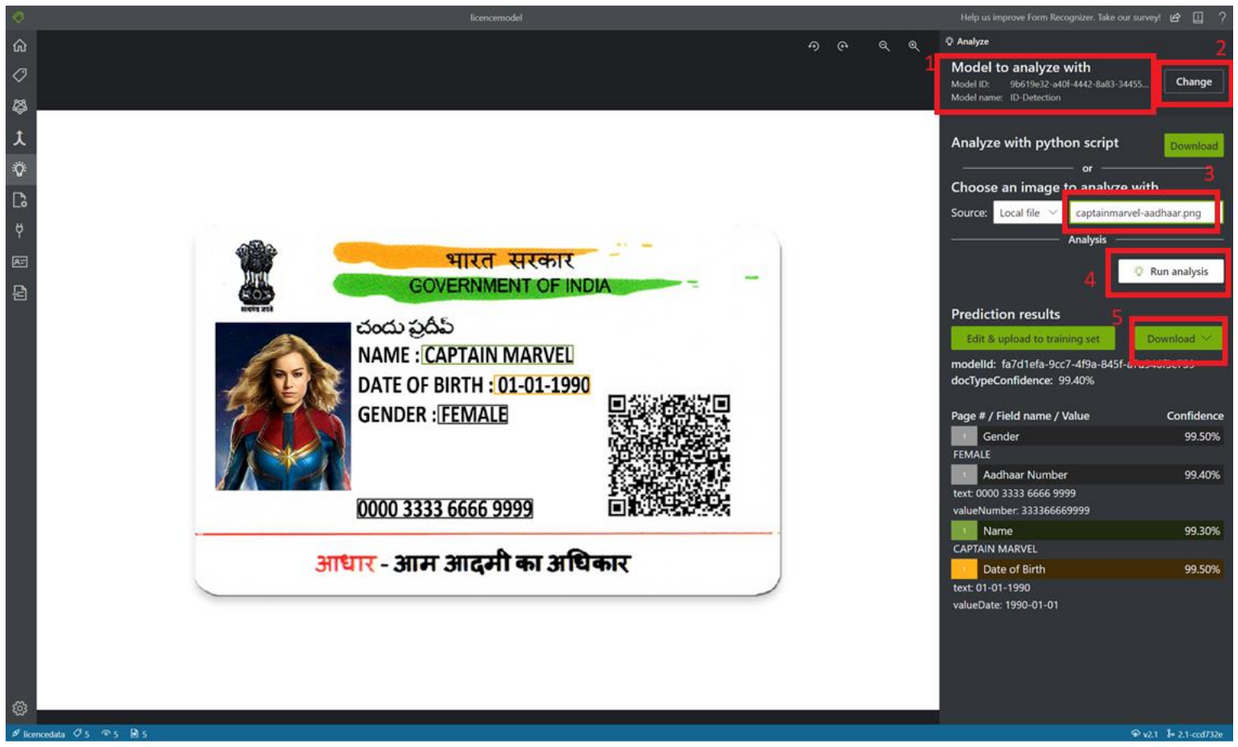
In a real-life scenario, however, if people are submitting ID cards for verification, they might be submitting different cards. This is where Compose model feature can be leveraged.
You can club the underlying models into a single model with 1 unique model ID.
Any new image submitted for processing will be automatically classified by the composed model and the underlying model will be called to extract the fields. This will reduce the overhead to manually sort & classify the incoming images & then calling the relevant model.

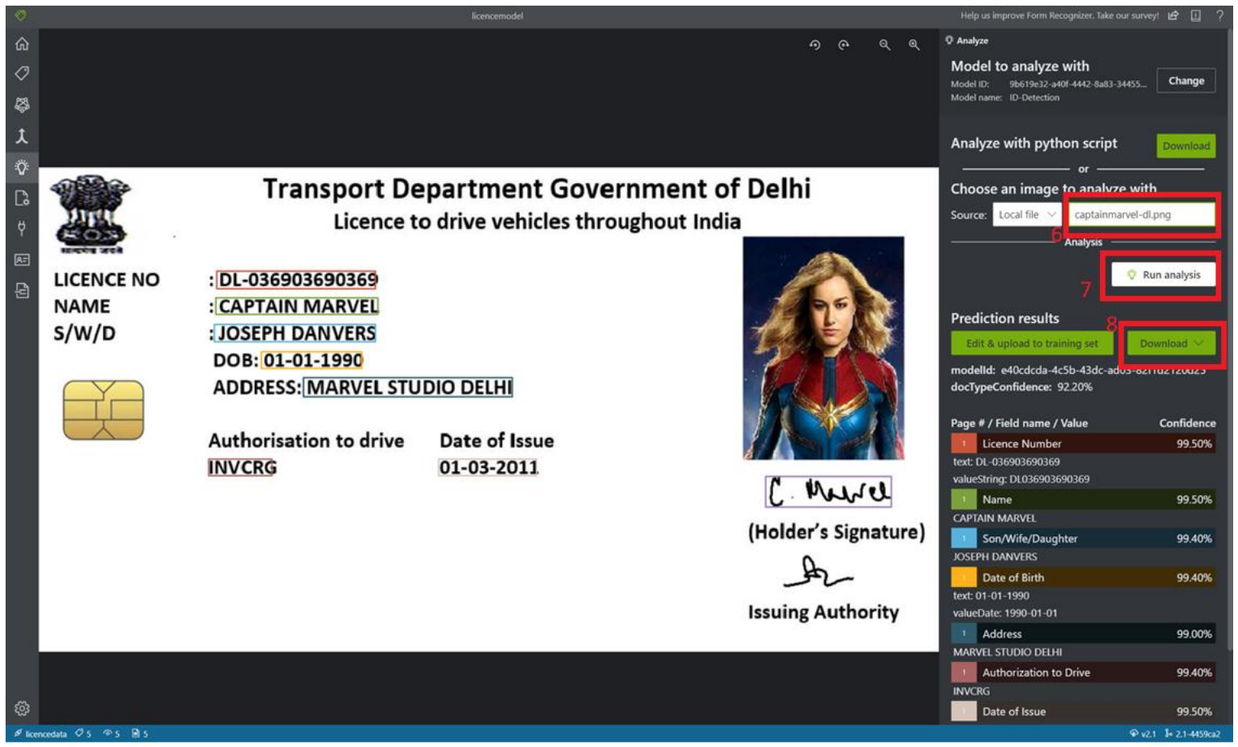
To create a Composed model, follow the steps as highlighted –

1. Switch to Compose model tab
2. Select model 1 you want to club
3. Select model 2 to club
4. Click compose

5. Provide a name for the composed model
6. Click compose

7. You will now see a model with the composed icon to it's left. That's the composed model.





<u>Testing the Composed model to process ID cards</u>

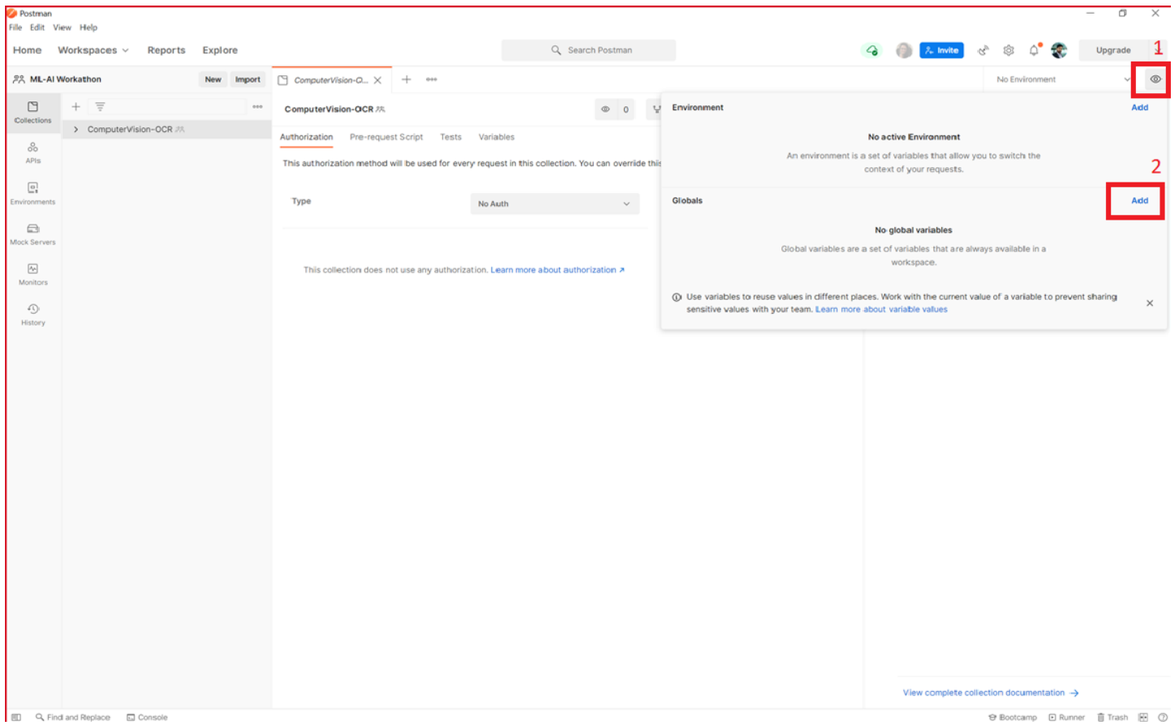
The created composed model can cater to both ID types. Hence, we will test out both.

Select the Test model tab & follow the highlighted steps to test the model –

1. Verify the Model name to sure you have the composed model selected.
2. If not, change it to select the right model by clicking the Change button
3. Select source as Local file and browse an Aadhaar card test image. If you want to test using an accessible image URL, change the source drop down to URL.
4. Click Run analysis and observe the Field Names & values and corresponding Confidence in the result
5. Download the JSON & CSV to see the elaborate results for both Key value pairs extracted and the complete document layout. You can leverage the JSON for post processing bases your use case.

Similarly, use the same model to process the Driver's Licence. Observe the outcomes. (Steps 6-8)

Notice how the same model and endpoint can now cater to multiple input form or document types, thus reducing overhead.



We have now switched the interface to Postman to test the models we created. If you haven't downloaded the Postman client, you can use web version.

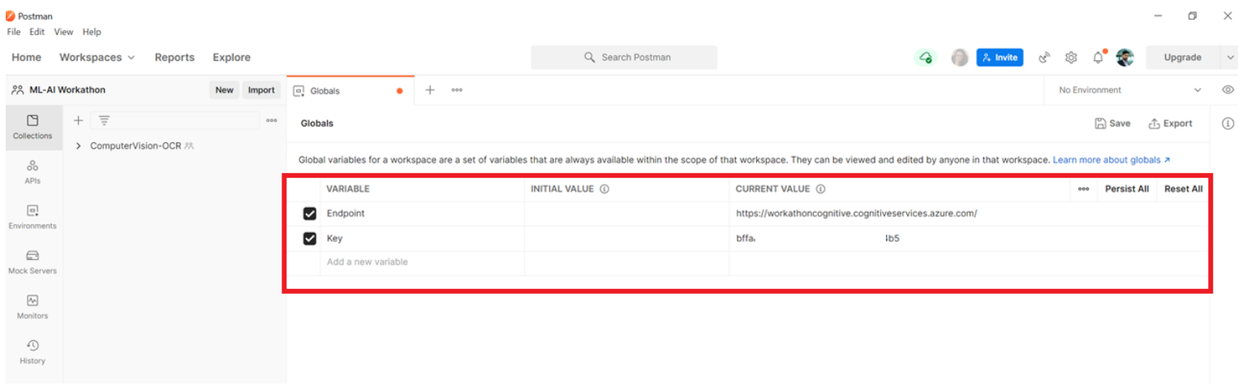
<u>Configure global variables in Postman</u>

Significance :
The global variables will be created once and leveraged time & again, in each API request that we make through Postman.
This way, you will not have to hard code the Endpoint & Key for every request you make, thereby, making it more secure. This will also save time and effort.

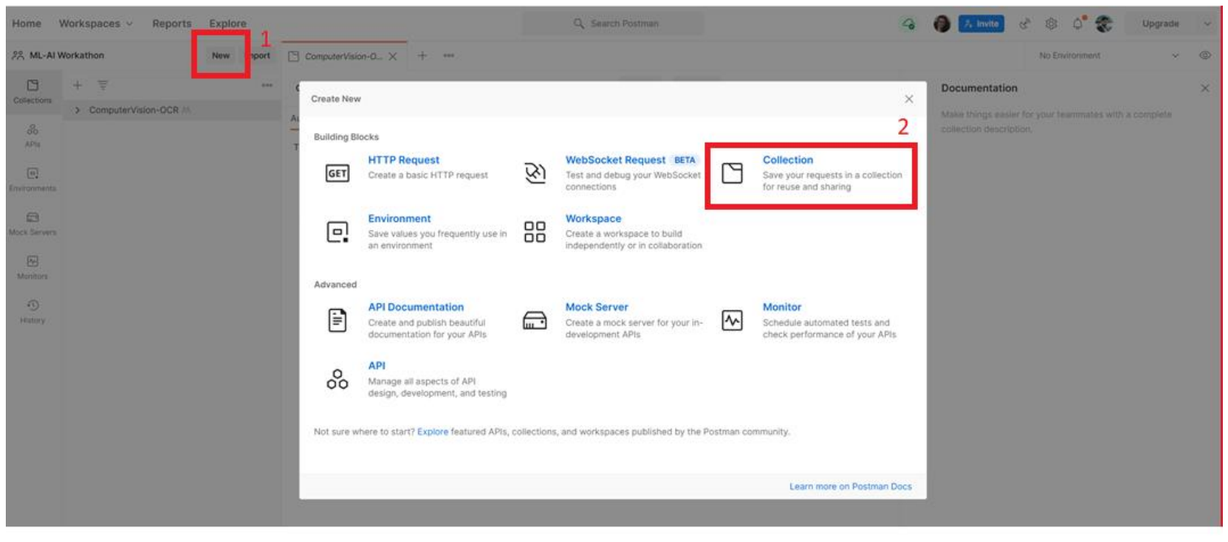
Follow step 1 & 2 to add global variables for :

1. Endpoint – Paste the endpoint of the Multipurpose service account copied earlier
2. Key - Paste the Key of the Multipurpose service account copied earlier

If you didn't copy the keys earlier, you can fetch them now from Keys and Endpoint section of the Cognitive Service resource.



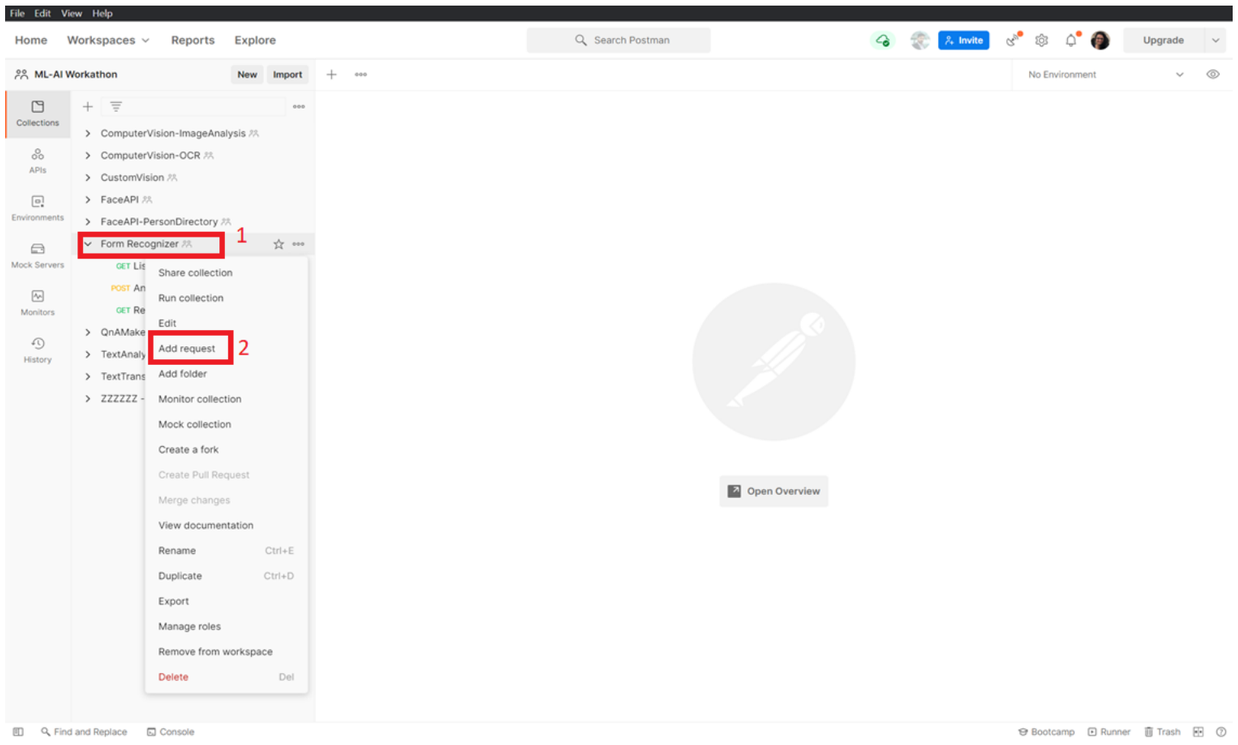
Once you have added the global variables, they will appear this way in your global variables section.



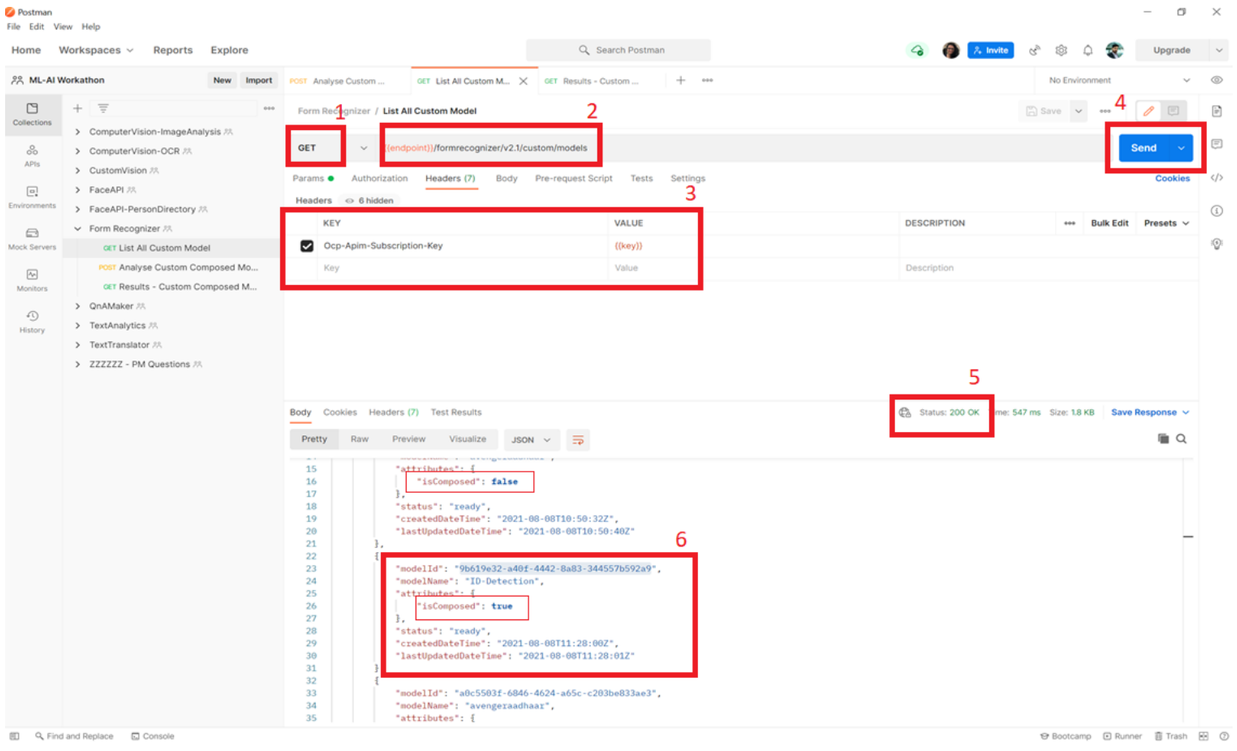
<u>Create new collection in Postman</u>

Open Postman > select New.
On the pop up select Collection.
Name the collection Form Recognizer.

Collection is like a folder for managing the API call requests.



Once you have created the collection, follow steps 1 & 2, to create a new request.



<u>Get details for all Models</u>

You can call this API request to get the details of all the custom & composed models you have created.

URL : {{endpoint}}/formrecognizer/v2.1/custom/models

Headers :
Ocp-Apim-Subscription-Key : {{key}}

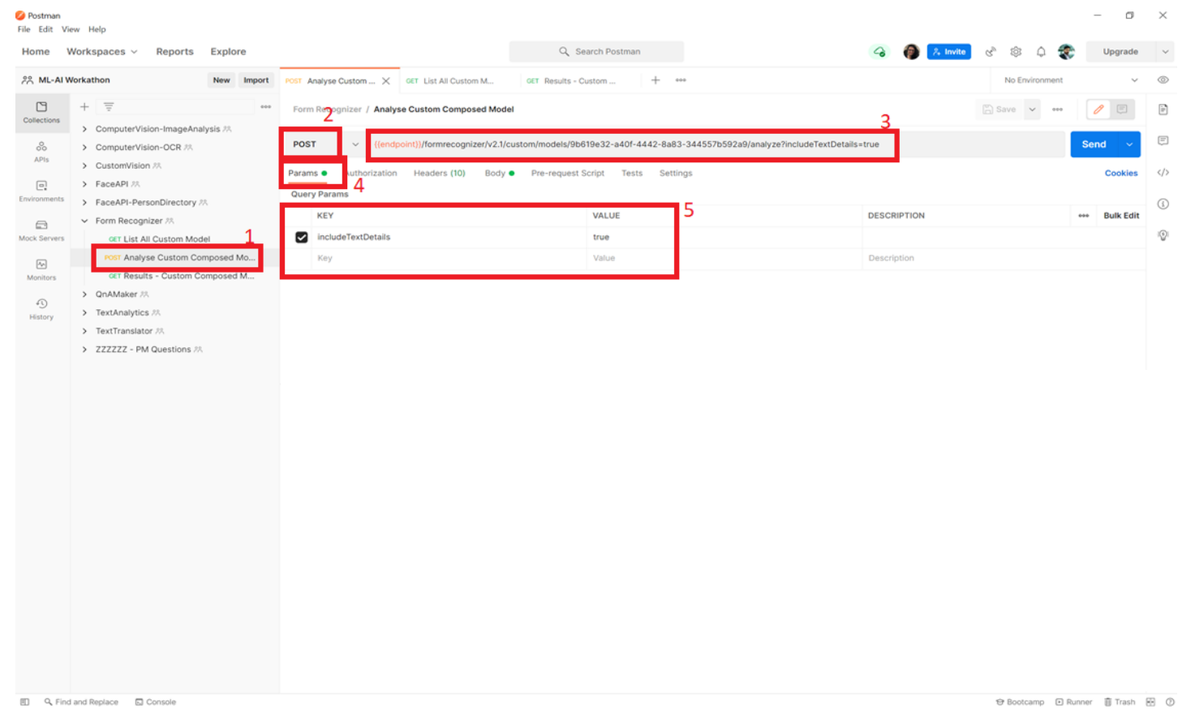
<u>Significance of input & output</u>

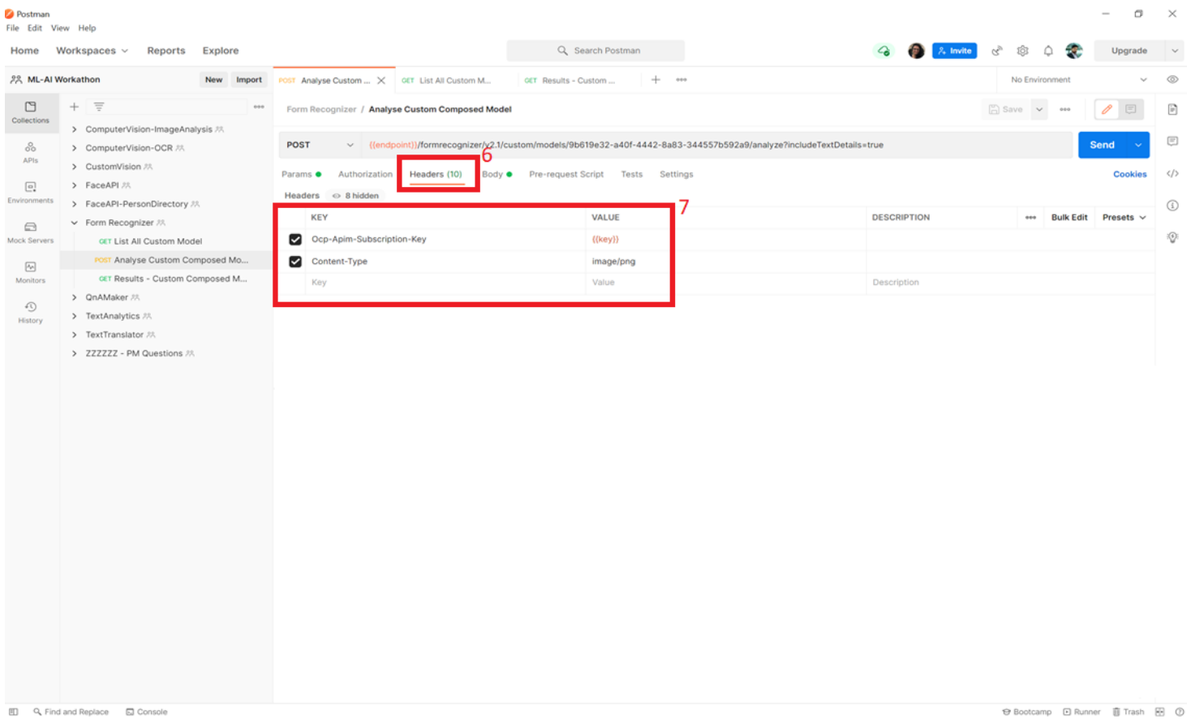
1. {{endpoint}} , {{key}} : Values being picked from global variables

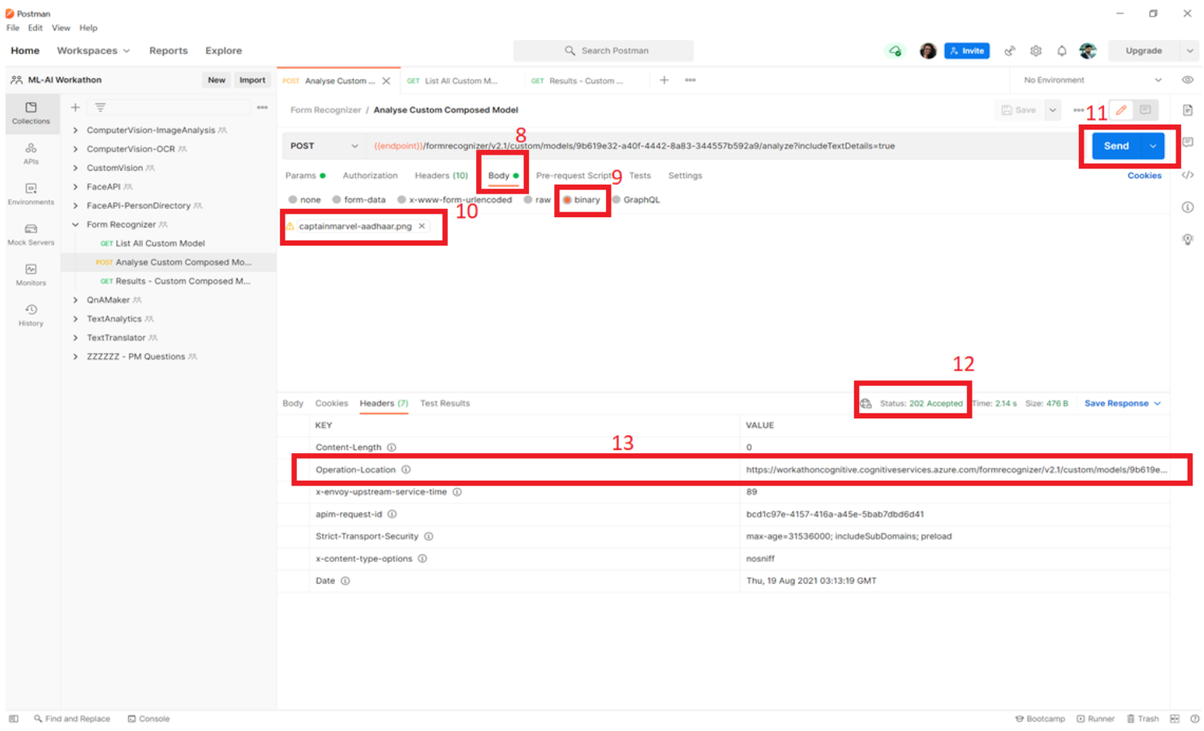
2. After you execute the call, observe the status returned, as shown in step 5. This should reflect 200 OK.

3. Observe the output received (step 6). You will see details for 3 models. You are returned the ModelId, ModelName, the relevant model created & updated dates. The isComposed value is True for Composed models & False for individual custom models.

Copy the Model ID for Composed model. We will leverage this in the next step to make REST API call.
You can also copy the individual custom model IDs, to try them out using REST APIs.







<u>Processing ID card using composed model</u>

The way we tested the composed model in GUI, we will now make a REST API call to test the model.

This is an async process, hence requires 2 API calls. The first call, as shown in step 3, generates the Operation-Location URL, shown in step 13.
The second call is a GET operation to this URL.

API CALL 1 (POST)

URL 1: {{endpoint}}/formrecognizer/v2.1/custom/models/<u>9b619e32-a40f-4442-8a83-344557b592a9</u>/analyze? includeTextDetails=true
Replace the underlined part with the Composed model key you copied in above step.

Headers :
Ocp-Apim-Subscription-Key : {{key}}
Content-Type : image/png

Body :
Select Binary and upload the test image for either Driver's Licence or Aadhaar card.

Copy the Operation-Location from the Headers section to view the result. We will use this in the next API call.

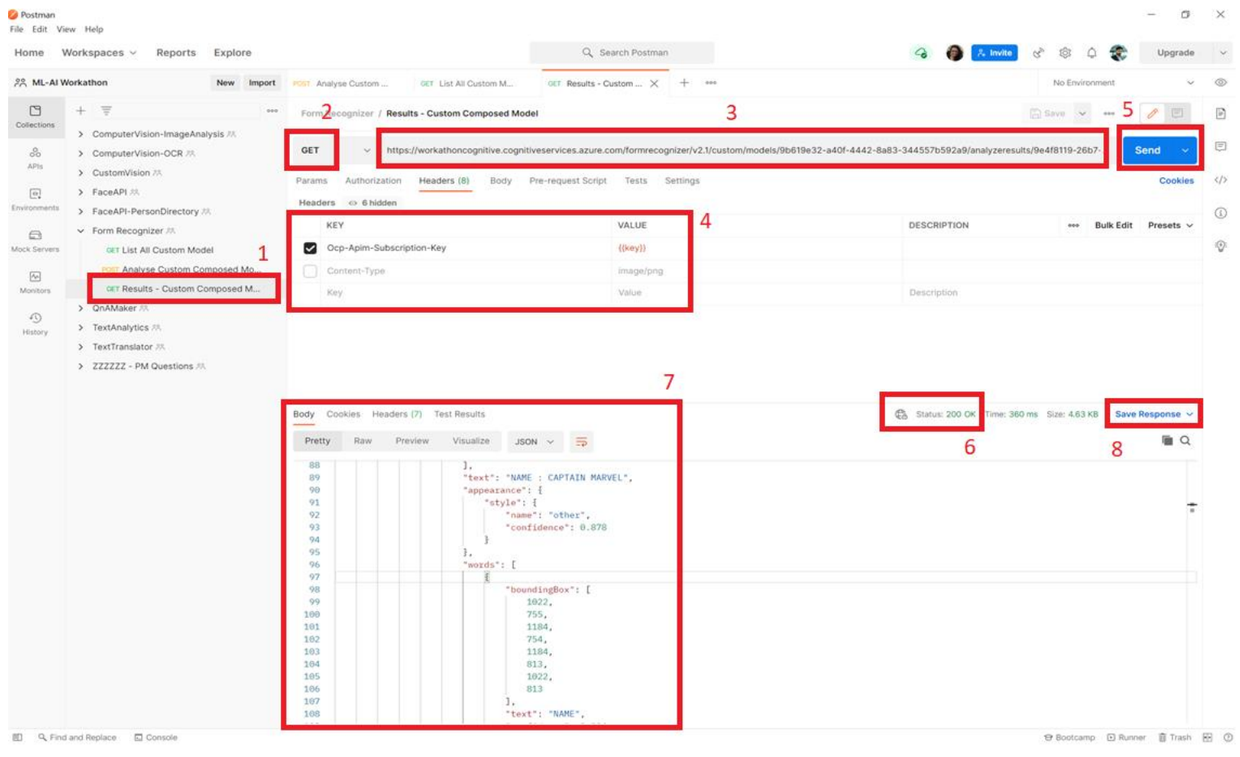
<u>Significance of input & output</u>

1. {{endpoint}} , {{key}} : Values being picked from global variables

2. We selected Content-Type as image/png since we are providing input as a binary object. If you want to provide a URL, choose Content-Type as application/json and add body in raw format as – {"url" : "<<enter url here>>"}

3. After you execute the call, observe the status returned, as shown in step 12. This should reflect 202 Accepted.

Proceed to 'GET' call in next step to obtain the results.



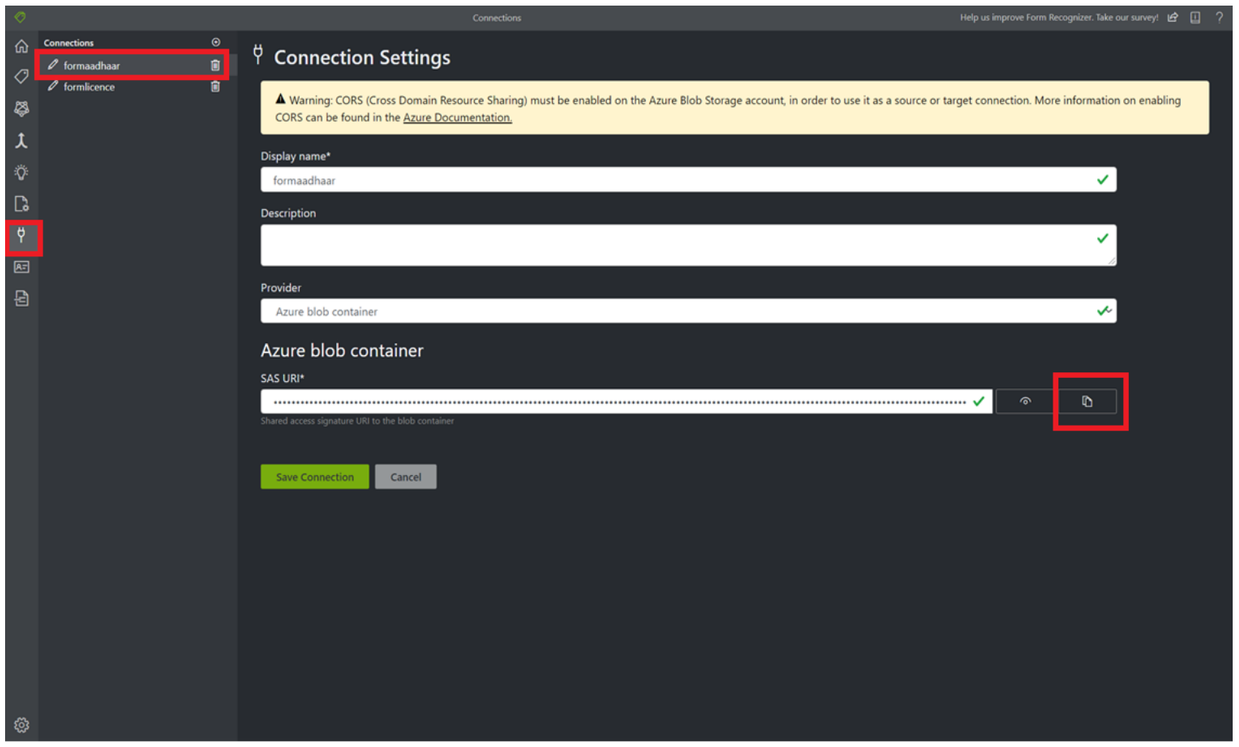
API CALL 2 (GET)

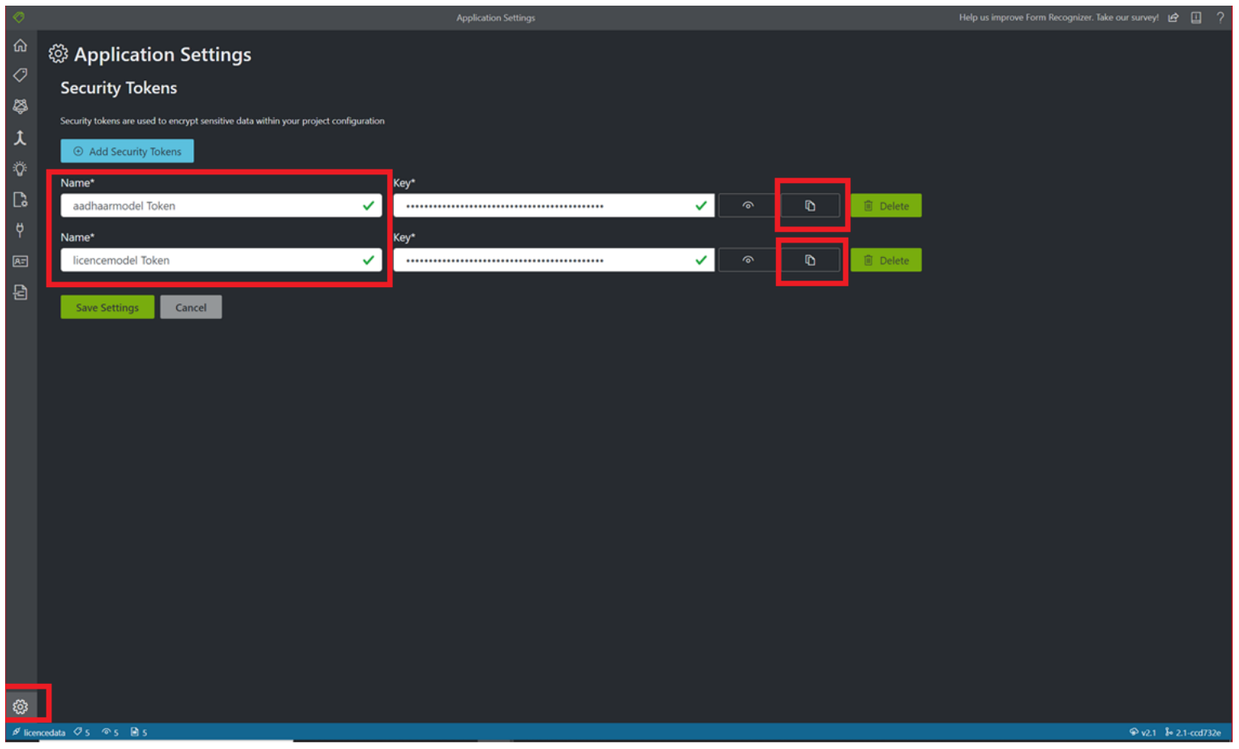
URL2 : Operation-Location fetched from step 13. Make sure to change the call to GET.

Headers :
Ocp-Apim-Subscription-Key : {{key}}

Significance of input & output

1. {{endpoint}} , {{key}} : Values being picked from global variables
2. After you execute the call, observe the status returned, as shown in step 6. This should reflect 200 OK.
3. Observe the JSON output you receive, as shown in step 7. For easier readability, select Save Response as a file (step 8). Observe the readResults & documentResults section carefully.





Sharing models / rendering them in a new browser client

Since the GUI Portal doesn't have a particular authentication mechanism and leverages browser cache while working with custom model(s), as a best practice we would recommend you store the following relevant information as backup to retrieve the model any time, in case it is not appearing on the portal -

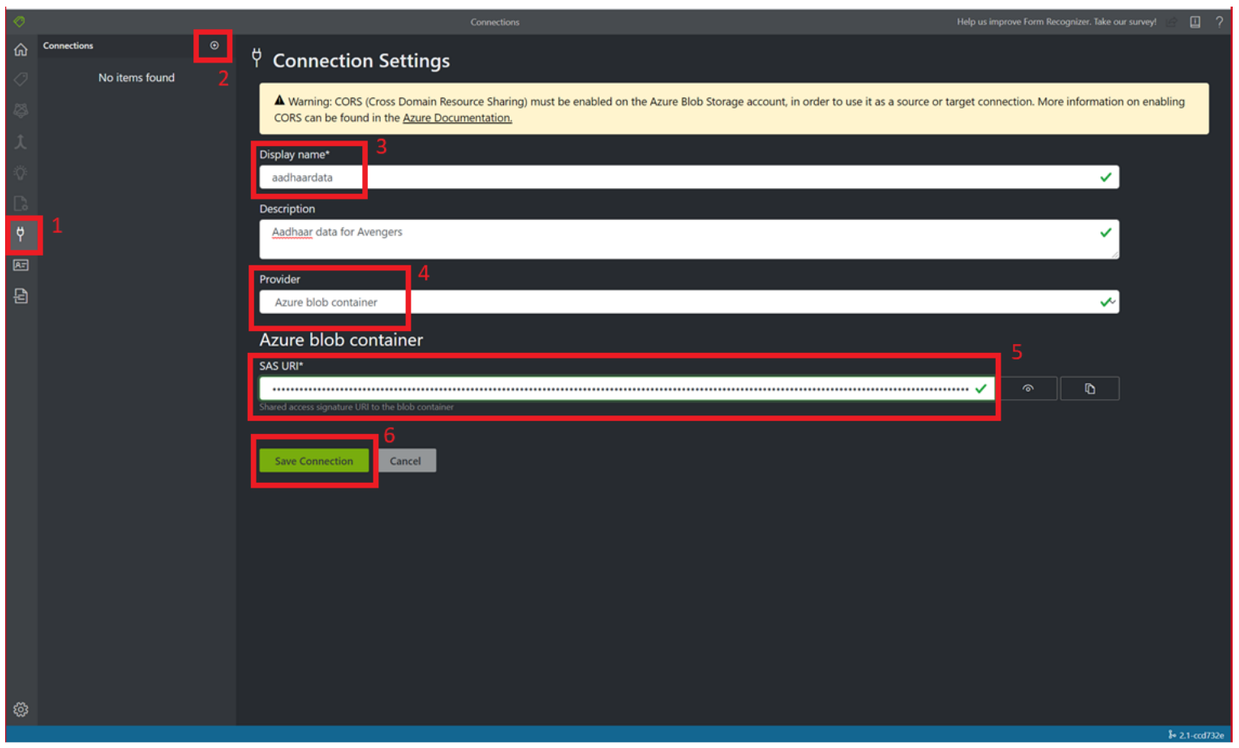
1. Blob SAS URL
2. Security Token

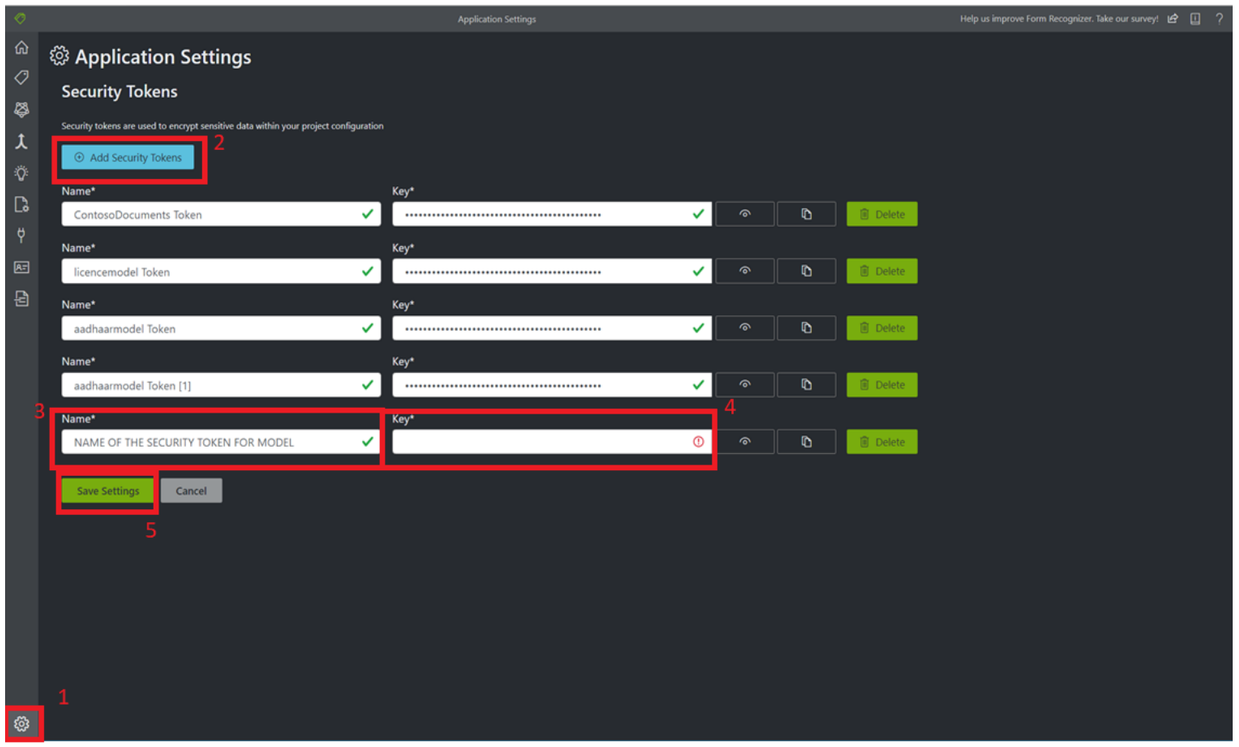
This is also applicable in case you want to share the project for collaboration with others.

Follow the steps to retrieve the above information –

1. **Blob SAS URL** : Follow highlighted steps in image 1. Copy the SAS URI for all the necessary containers.
2. **Security Token** : Follow the highlighted steps in image 2. Copy the Security Tokens for all the relevant models.

Once you have these 2 bits of information, you can retrieve the project anytime.
In the next steps, let's see how.



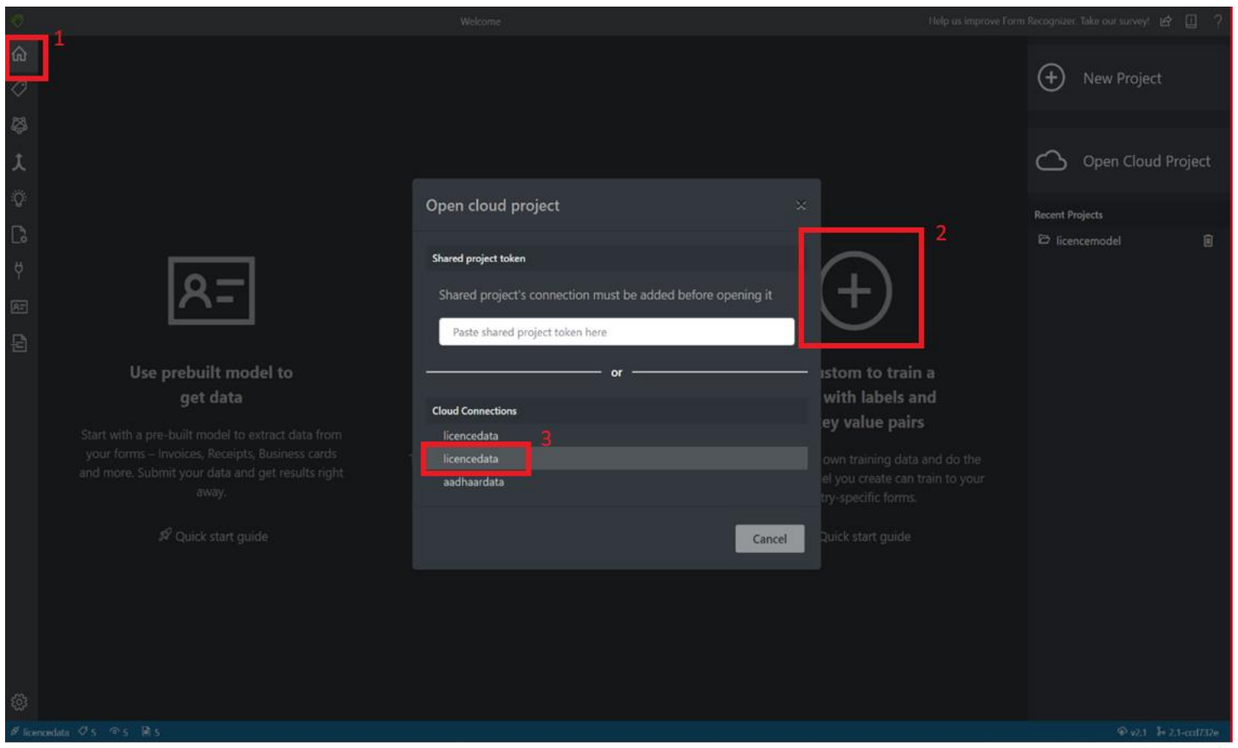


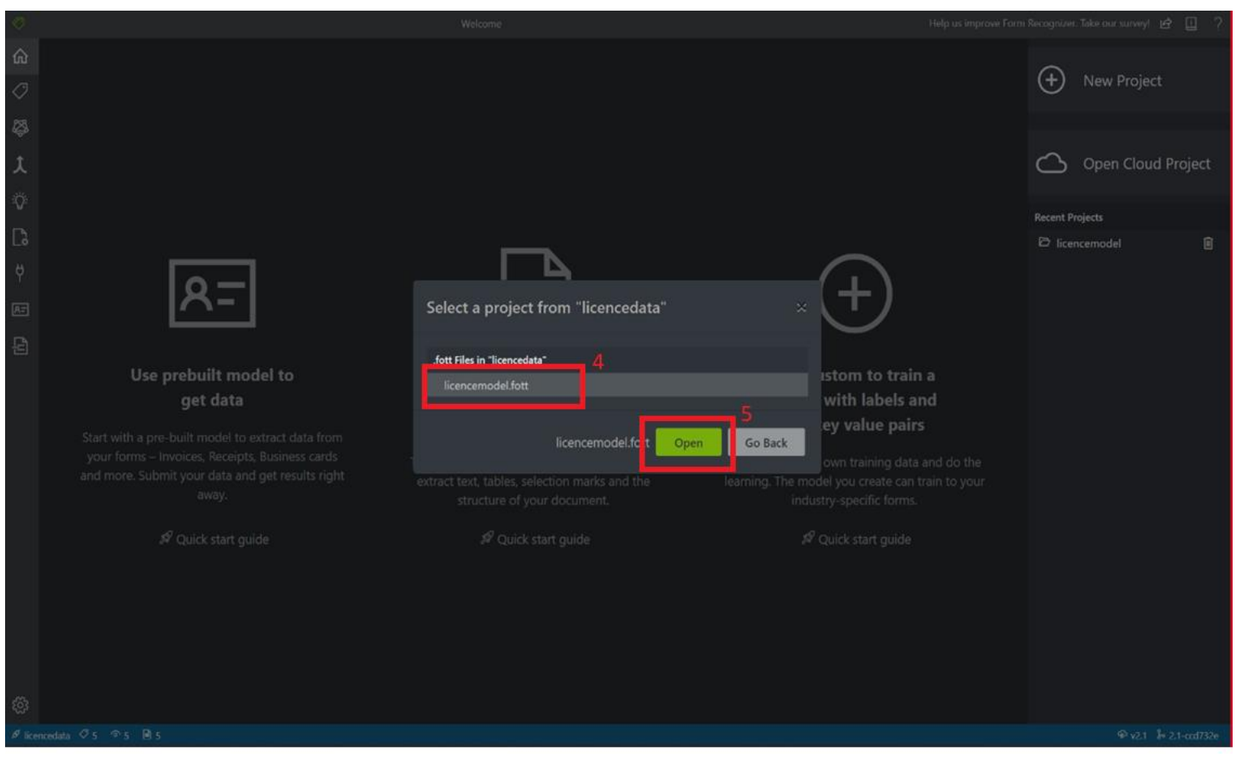
Retrieving the Model

Follow these steps to retrieve previously created models:

a. Create connection to blob container
   1. Go to Connections tab
   2. Select + to add a new connection setting.
   3. Enter the details for the model you want to retrieve. Set the SAS URI to the blob container URI copied above [Step 3-5]
   4. Save connection

b. Add Security token
   1. Go to settings tab
   2. Select Add security token
   3. Enter the Name for the token copied earlier
   4. Enter the Key for the token Copied above
   5. Save settings





Open saved Cloud Project

Once you have established the connection to the blob storage container & added the relevant Security Tokens, you are ready to retrieve the project.

1. Go to home tab
2. Select + to create project & select Cloud Project
3. On the pop up, select the blob connection's name

4. Select the available fott file
5. Click Open

Homework

1. In this workshop, we have explored the custom form models. We would recommend you to also try out the pre-built model for :
   a. Layout
   b. Invoices
   c. Receipts
   d. Business Cards
   e. ID Cards
   You can do this using the GUI Portal used above, REST APIs or language specific SDKs.
2. Try building the custom model using REST APIs or language SDKs.

Additional recommended resources

Service Name	Category	Links
Form Recognizer	Programming Language	C#, Java, Python, Javascript
	Tiers	Free (Not for Production), Standard
	Pricing	https://azure.microsoft.com/en-in/pricing/details/form-recognizer/
	Limits	
	Language Support	https://docs.microsoft.com/en-in/azure/applied-ai-services/form-recognizer/language-support
	Sample Apps	Sample Applications
	Regional Availability & Support	https://azure.microsoft.com/en-us/global-infrastructure/services/?products=cognitive-services&regions=all
	SLAs for Cognitive Services	https://azure.microsoft.com/en-in/support/legal/sla/cognitive-services/v1_1/
	Compliance & Certificates	https://azure.microsoft.com/en-us/support/legal/cognitive-services-compliance-and-privacy/
	Cognitive Services Updates	https://azure.microsoft.com/en-us/updates/?product=cognitive-services

Security best practices

1. Azure Cognitive Services security
2. Networking
3. Authentication
4. Key Management
5. Data loss prevention
6. Azure security baseline
7. Regulatory Compliance controls

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

Form Recognizer Documentation

API & Error references