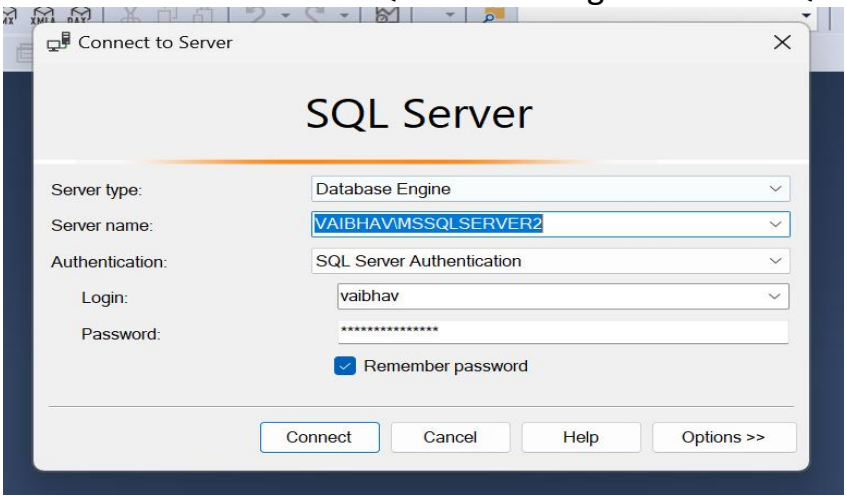
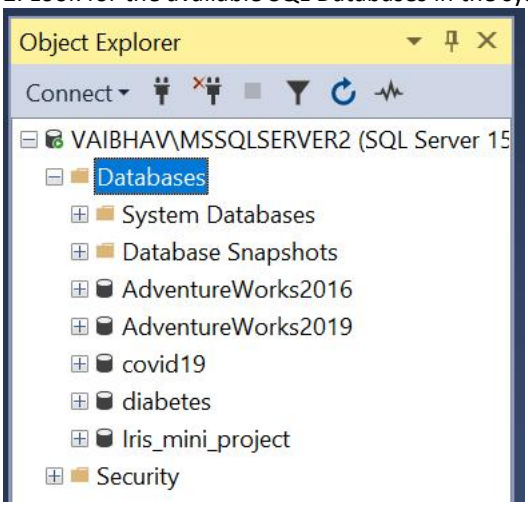
Step 1:- Import data from Microsoft Excel into Microsoft SQL Server

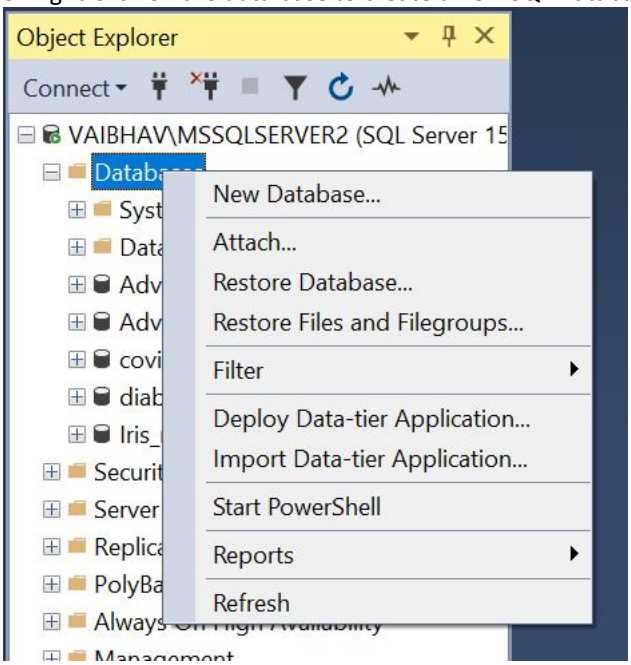
1. Connect to the local SQL Server using Microsoft SQL Server Management Studio



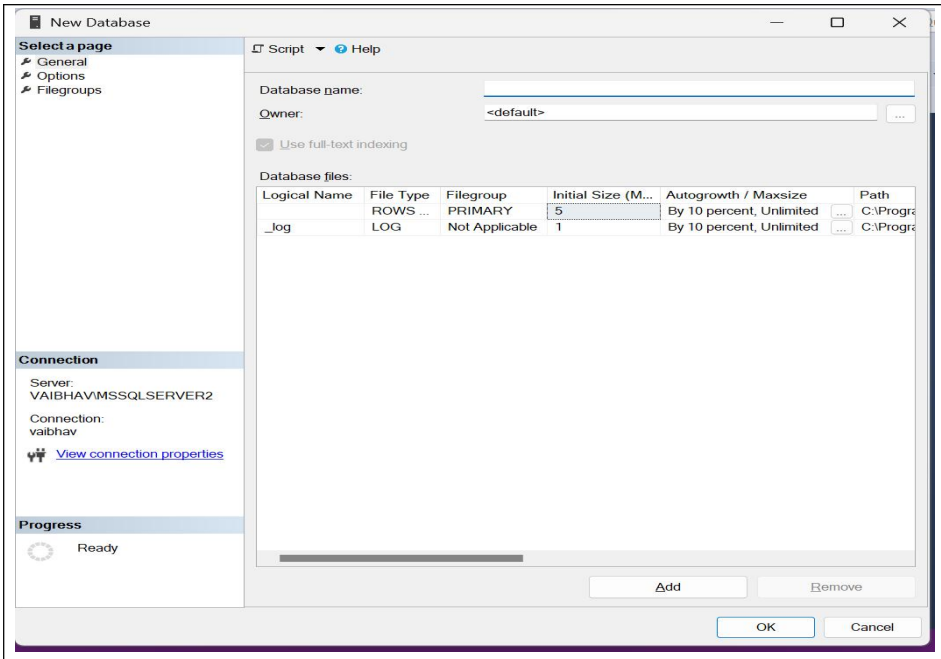
2. Look for the available SQL Databases in the system.



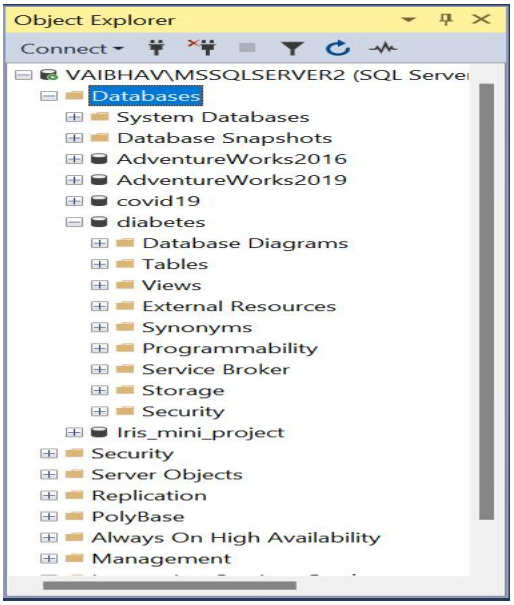
3. Right Click on the data base to create a new SQL Database for the file to be uploaded.



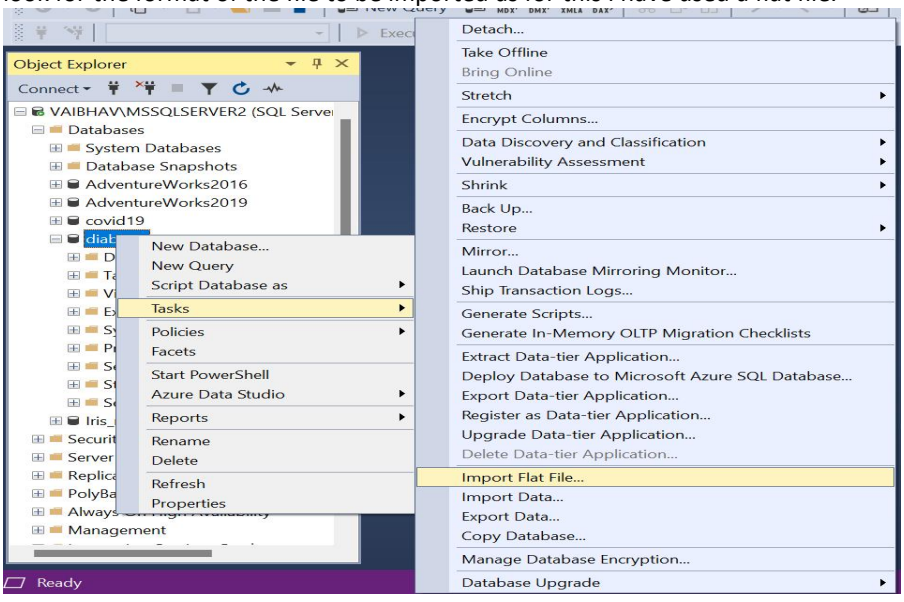
4. Enter the Name for the New Database which is needed to be created and keep rest of the things as default because in the further step the system will create the necessary details from the source file itself.



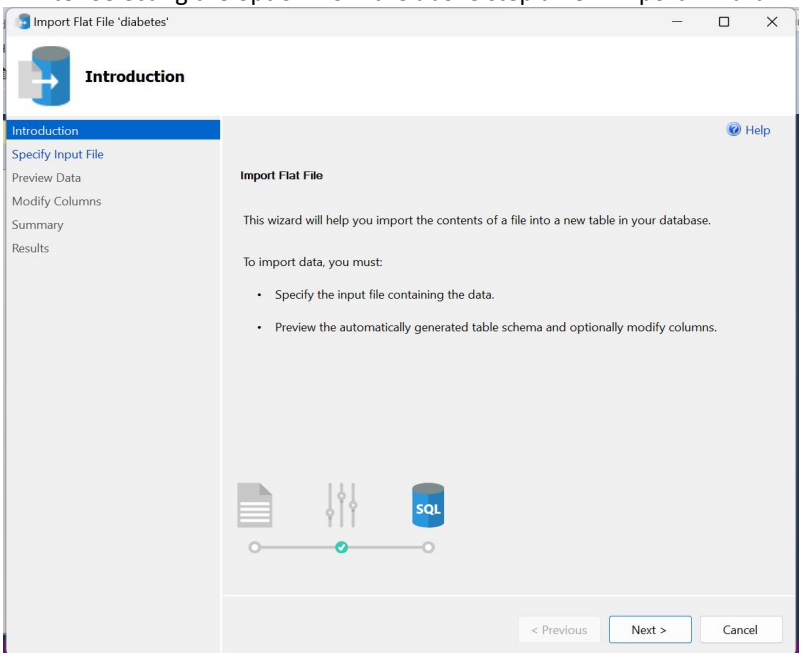
5. Once the Database is created refresh the Server and look for the newly created Database and expand it for further use.



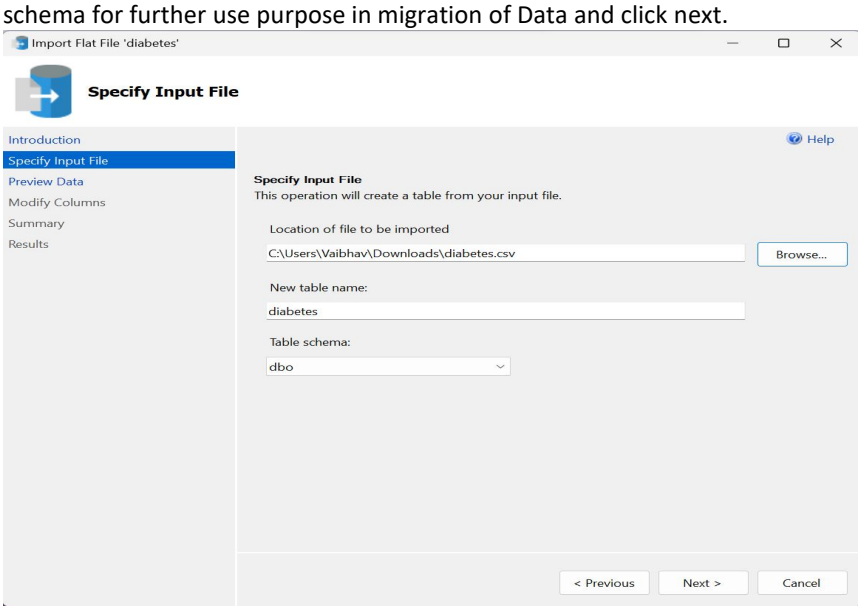
6. After expanding the Database, right click on the database name and then locate the task option, click on the task option and look for the format of the file to be imported as for this I have used a flat file.



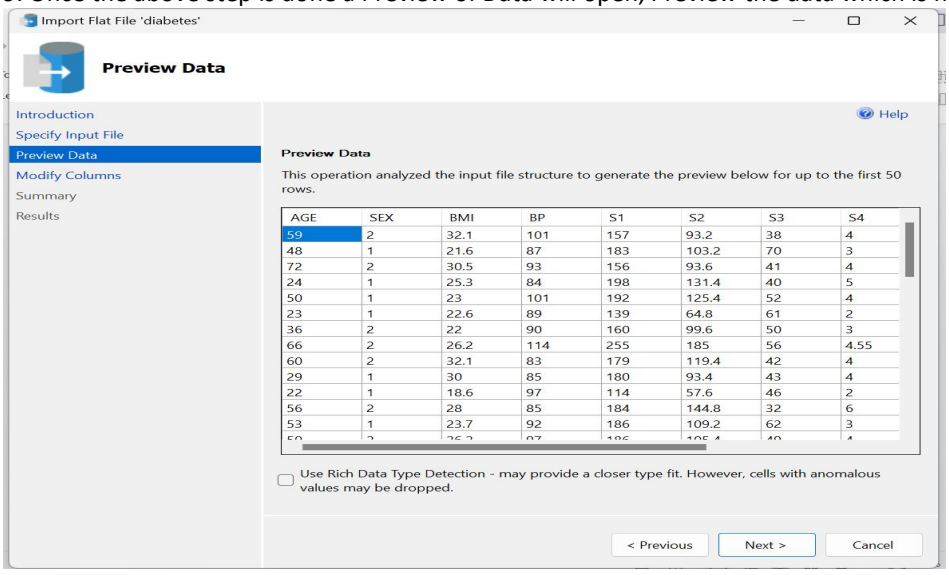
7. After Selecting the option from the above step a new Import Wizard will appear and click next to proceed.



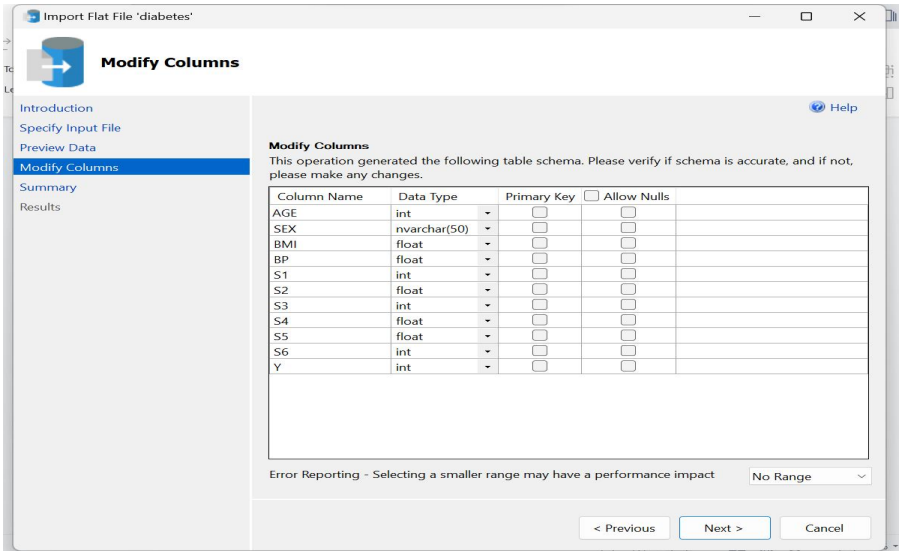
8. In this specify the path of the file which is needed to be imported and also mention the name of the table with the table schema for further use purpose in migration of Data and click next.



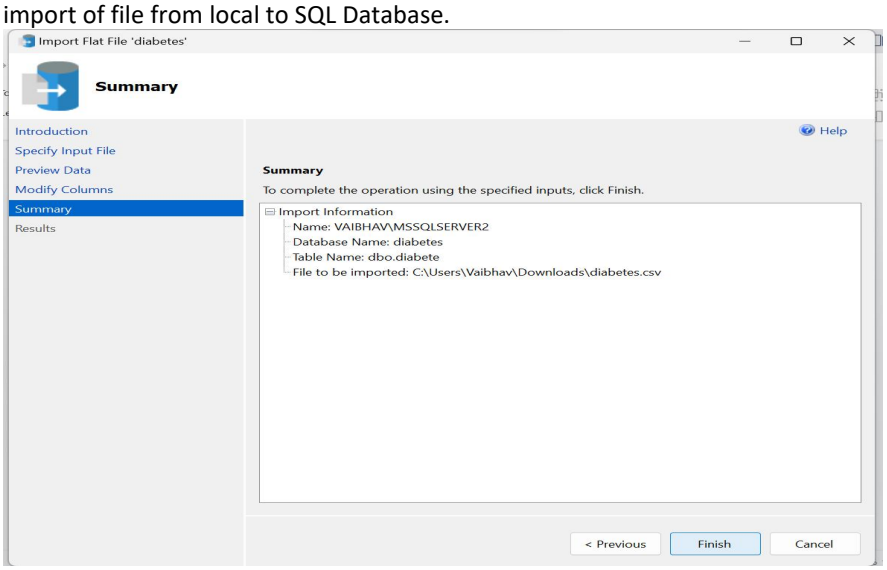
9. Once the above step is done a Preview of Data will open, Preview the data which is needed to be imported and click next.



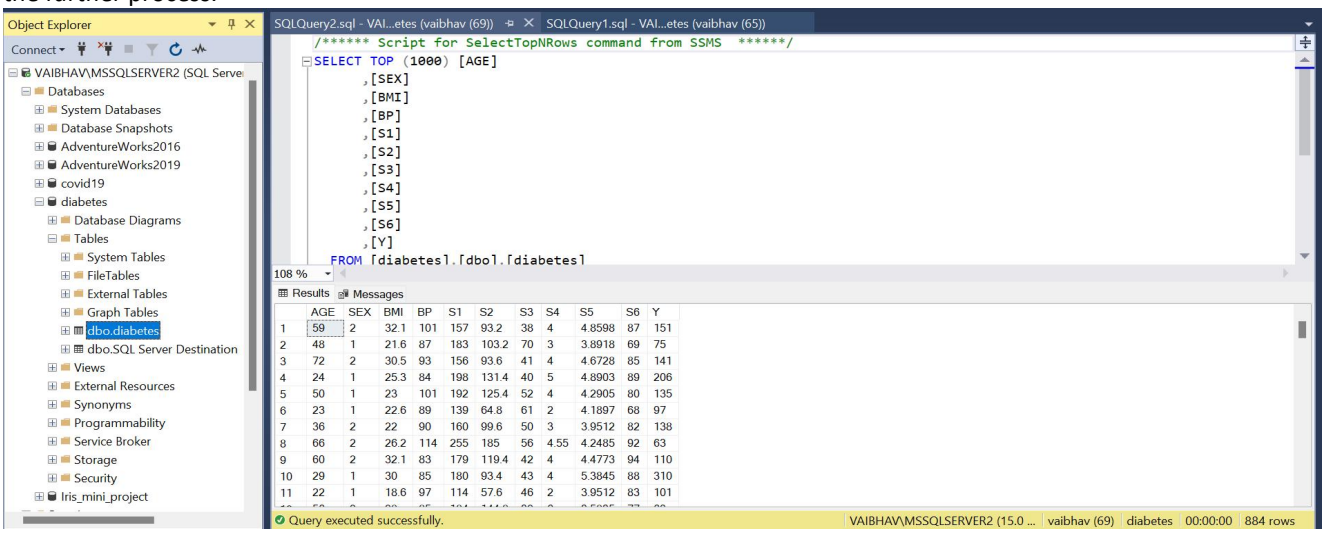
10. Define the necessary Keys and look for the data types are they correctly allocated if not then updated it in the wizard and click next.



11. Once the above steps are done correctly summary will be generated and click on the finish button to complete the task of import of file from local to SQL Database.



12. Once the task is finished refresh the Server and expand the Tables Option in the Database and right click on the table name which was created with schema in the import wizard step and then click on the Select Statement to check and use the data in the further process.



Step 2:- Copy data from a SQL Server database to Azure Blob storage

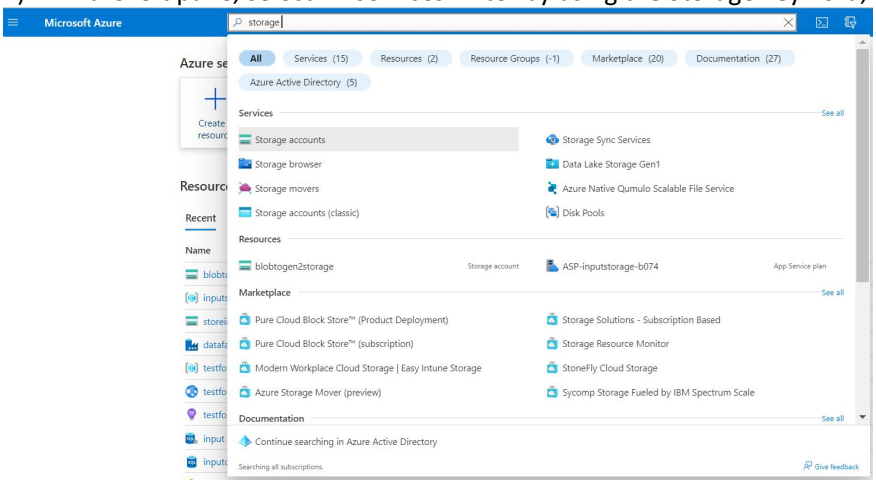
General Steps:-

1. Create a data factory.
2. Create a self-hosted integration runtime.
3. Create SQL Server and Azure Storage linked services.
4. Create SQL Server and Azure Blob datasets.
5. Create a pipeline with a copy activity to move the data.
6. Start a pipeline run.
7. Monitor the pipeline run.

Get the storage account name and account key

You use the name and key of your storage account in this. To get the name and key of your storage account, take the following steps:

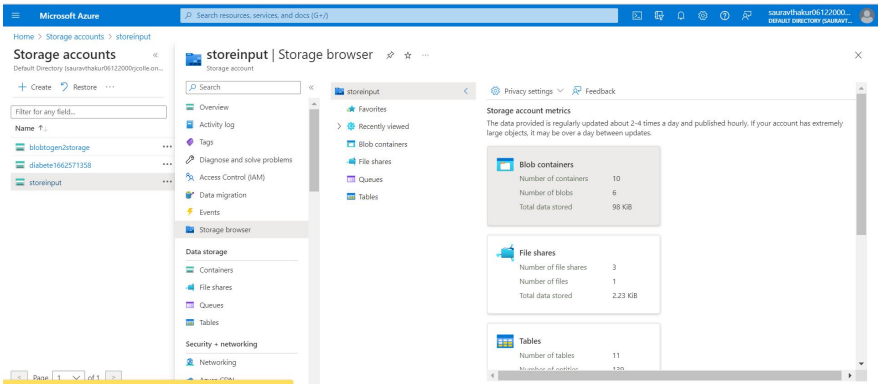
1) Sign in to the Azure portal with your Azure user name and password.
2) In the left pane, select All services. Filter by using the Storage keyword, and then select Storage accounts.



3) In the list of storage accounts, filter for your storage account if needed. Then select your storage account.
4) In the Storage account window, select Access keys.
5) In the Storage account name and key1 boxes, copy the values, and then paste them into Notepad or another editor for later use

Create the container

1) In this section, you create a blob container in your Blob storage.



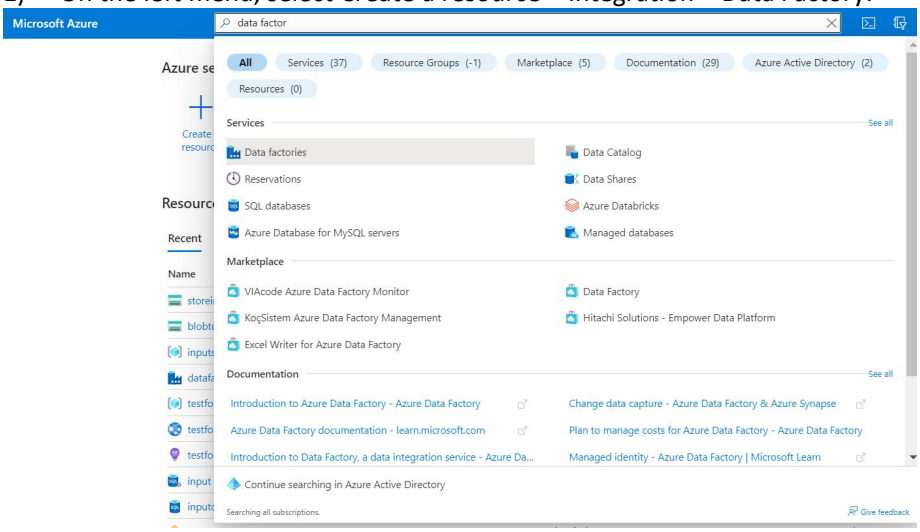
2) In the Storage account window, go to Overview, and then select Containers.
3) Select Blobs option
4) In the Containers window, select + Container to create a new one.
5) In the New container window, under Name, enter NAME_FOR_THE_CONTAINER. Then select Create.
6) In the list of containers, select container you just created.

Keep the container window open. You use it to verify the output at the end of the below step. Data Factory automatically creates the output folder in this container, so you don't need to create one.

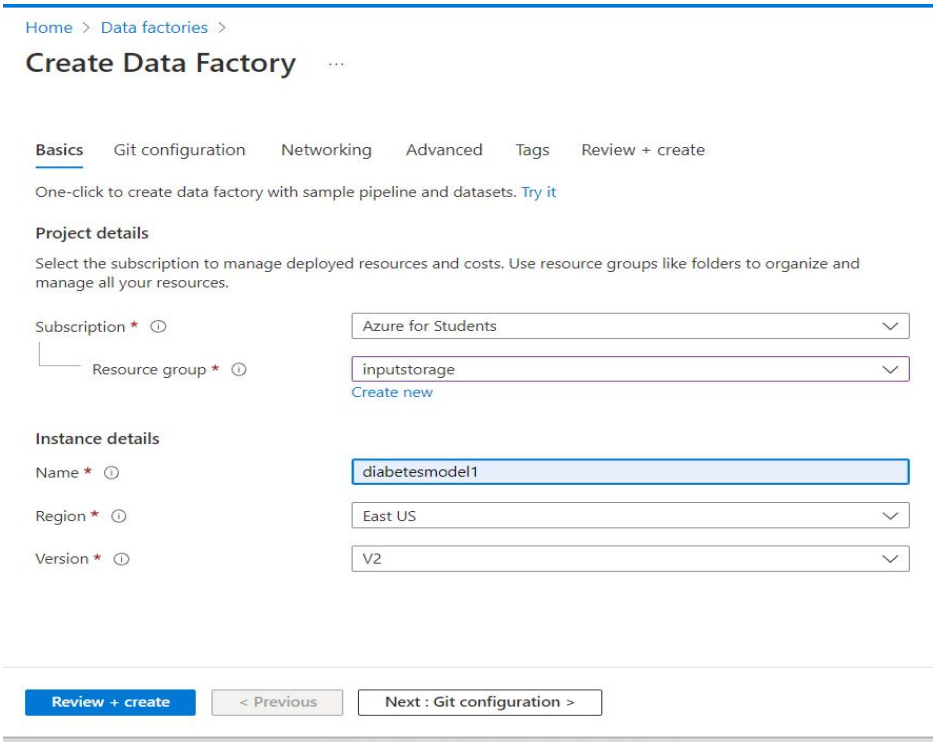
Create a data factory

In this step, you create a data factory and start the Data Factory UI to create a pipeline in the data factory.

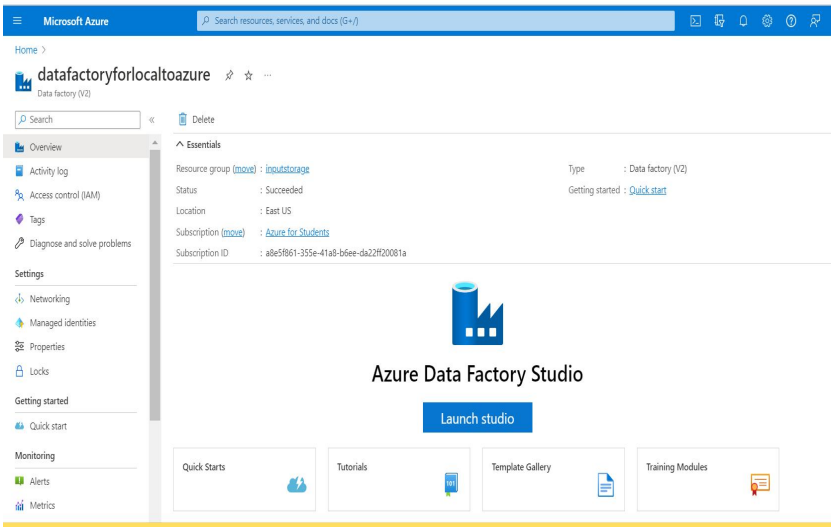
1) On the left menu, select Create a resource > Integration > Data Factory:



2) Data Factory selection in the "New" pane
3) On the New data factory page, under Name, enter a unique name.



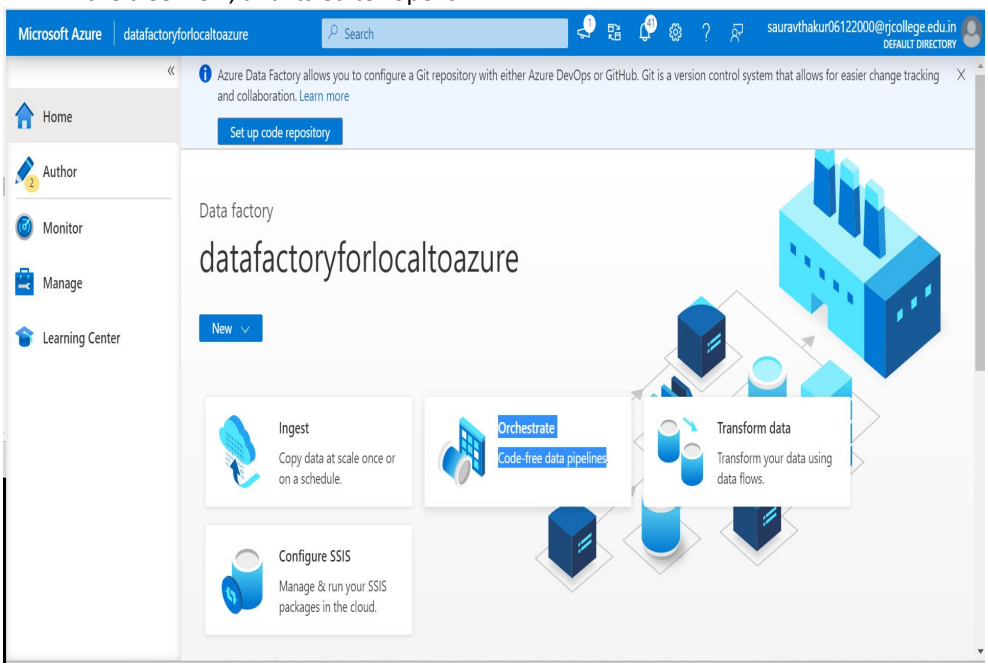
4) Select the Azure subscription in which you want to create the data factory.
5) For Resource Group, take one of the following steps:
   a) Select Use existing, and select an existing resource group from the drop-down list.
   b) Select Create new, and enter the name of a resource group.
6) Under Version, select V2.
7) Under Location, select the location for the data factory. Only locations that are supported are displayed in the drop-down list. The data stores and computes used by Data Factory can be in other regions.
8) Select Create.
9) After the creation is finished, you see the Data Factory page as shown in the image:



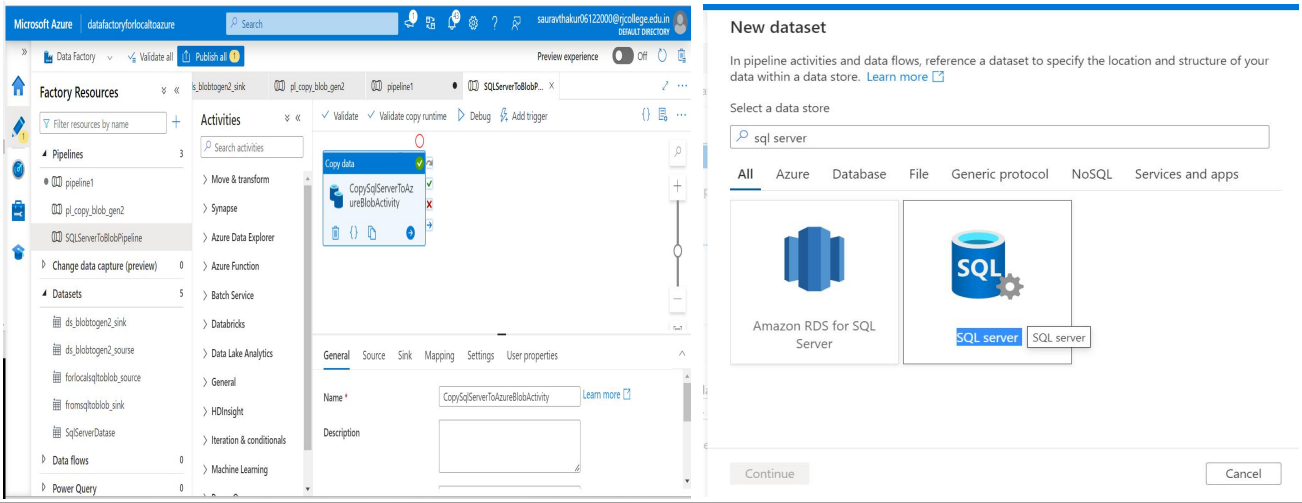
10) Select Open on the Open Azure Data Factory Studio tile to launch the Data Factory UI in a separate tab.

Create a pipeline

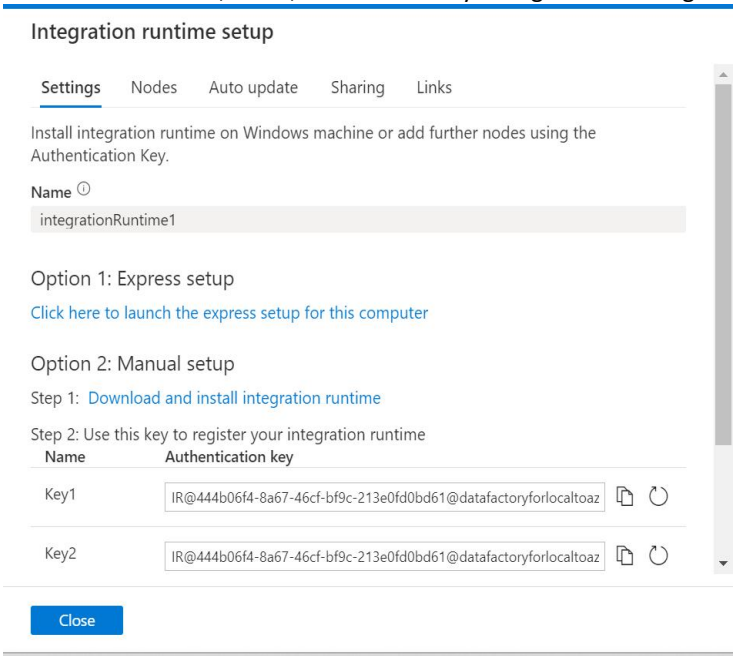
1) On the Azure Data Factory home page, select Orchestrate. A pipeline is automatically created for you. You see the pipeline in the tree view, and its editor opens.



2) In the General panel under Properties, specify SQLServerToBlobPipeline for Name. Then collapse the panel by clicking the Properties icon in the top-right corner.
3) In the Activities tool box, expand Move & Transform. Drag and drop the Copy activity to the pipeline design surface. Set the name of the activity to CopySqlServerToAzureBlobActivity.
4) In the Properties window, go to the Source tab, and select + New.
5) In the New Dataset dialog box, search for SQL Server. Select SQL Server, and then select Continue.



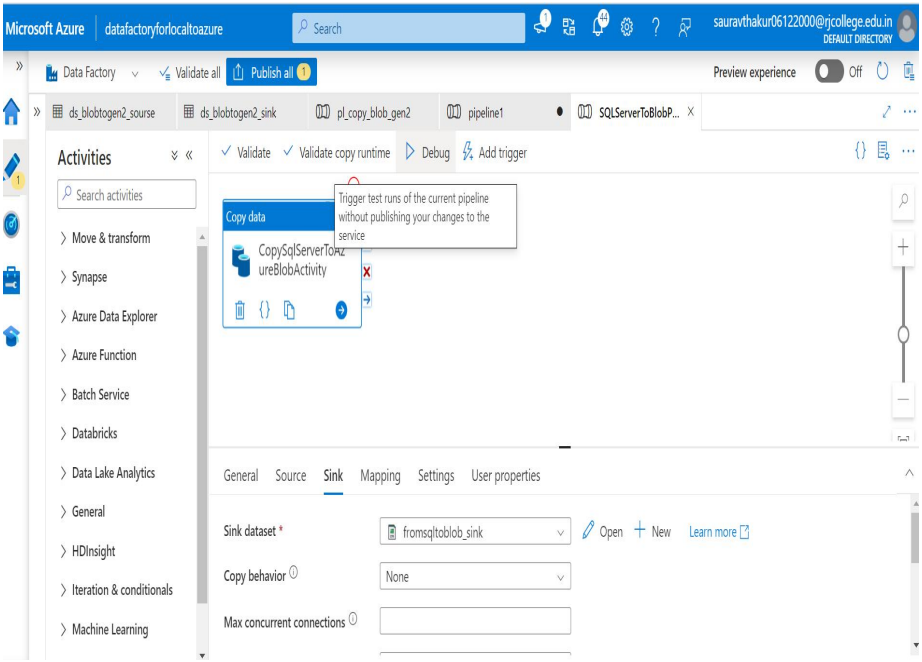
6) In the Set Properties dialog box, under Name, enter SqlServerDataset. Under Linked service, select + New. You create a connection to the source data store (SQL Server database) in this step.
7) In the New Linked Service dialog box, add Name as SqlServerLinkedService. Under Connect via integration runtime, select +New. In this section, you create a self-hosted integration runtime and associate it with an on-premises machine with the SQL Server database. The self-hosted integration runtime is the component that copies data from the SQL Server database on your machine to Blob storage.
8) In the Integration Runtime Setup dialog box, select Self-Hosted, and then select Continue.
9) Under name, enter TutorialIntegrationRuntime. Then select Create.
10) For Settings, select Click here to launch the express setup for this computer. This action installs the integration runtime on your machine and registers it with Data Factory. Alternatively, you can use the manual setup option to download the installation file, run it, and use the key to register the integration runtime.



11. In the Integration Runtime (Self-hosted) Express Setup window, select Close when the process is finished.
12. In the New linked service (SQL Server) dialog box, confirm that TutorialIntegrationRuntime is selected under Connect via integration runtime. Then, take the following steps:
    a. Under Name, enter SqlServerLinkedService.
    b. Under Server name, enter the name of your SQL Server instance.
    c. Under Database name, enter the name of the database with the emp table.
    d. Under Authentication type, select the appropriate authentication type that Data Factory should use to connect to your SQL Server database.
    e. Under User name and Password, enter the user name and password. Use mydomain\myuser as user name if needed.
    f. Select Test connection. This step is to confirm that Data Factory can connect to your SQL Server database by using the self-hosted integration runtime you created.
    g. To save the linked service, select Create.
13) After the linked service is created, you're back to the Set properties page for the SqlServerDataset. Take the following steps:
    a. In Linked service, confirm that you see SqlServerLinkedService.
    b. Under Table name, select [dbo].[table_name].
    c. Select OK.
14) Go to the tab with SQLServerToBlobPipeline, or select SQLServerToBlobPipeline in the tree view.
15) Go to the Sink tab at the bottom of the Properties window, and select + New.
16) In the New Dataset dialog box, select Azure Blob Storage. Then select Continue.
17) In Select Format dialog box, choose the format type of your data. Then select Continue.
18) In the Set Properties dialog box, enter AzureBlobDataset for Name. Next to the Linked service text box, select + New.
19) In the New Linked Service (Azure Blob Storage) dialog box, enter AzureStorageLinkedService as name, select your storage account from the Storage account name list. Test connection, and then select Create to deploy the linked service.
20) After the linked service is created, you're back to the Set properties page. Select OK.
21) Open the sink dataset. On the Connection tab, take the following steps:
    a. In Linked service, confirm that AzureStorageLinkedService is selected.
    b. In File path, enter 'container_created' /fromonprem for the Container/ Directory part. If the output folder doesn't exist in the 'container_created' container, Data Factory automatically creates the output folder.
    c. For the File part, select Add dynamic content. dynamic expression for resolving file name
    d. Add @CONCAT(pipeline().RunId, '.txt'), and then select Finish. This action will rename the file with PipelineRunID.txt.

22) Go to the tab with the pipeline opened, or select the pipeline in the tree view. In Sink Dataset, confirm that AzureBlobDataset is selected.

23) To validate the pipeline settings, select Validate on the toolbar for the pipeline. To close the Pipe validation output, select the >> icon. validate pipeline



24) To publish entities you created to Data Factory, select Publish all.

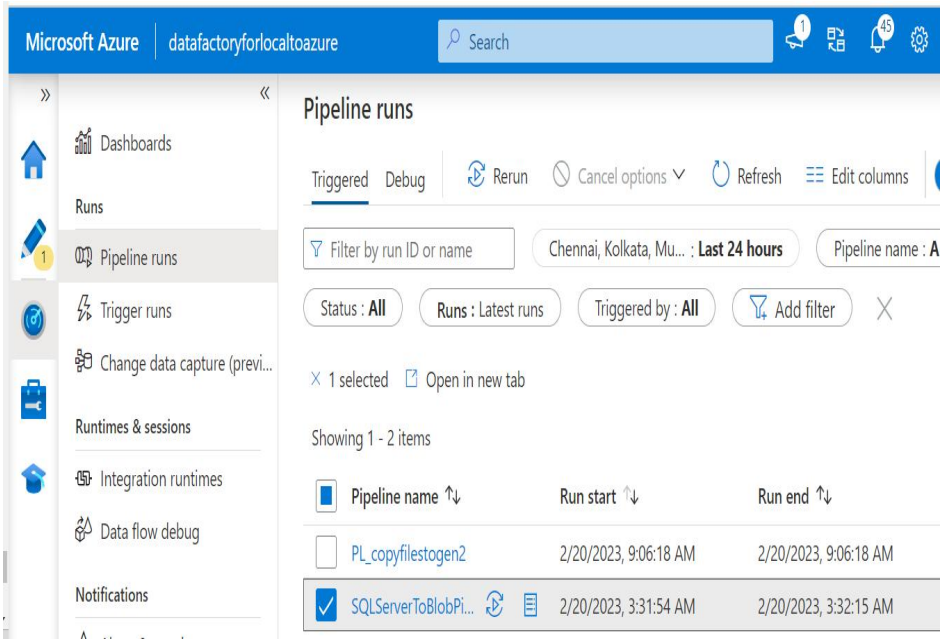
25) Wait until you see the Publishing completed pop-up. To check the status of publishing, select the Show notifications link on the top of the window. To close the notification window, select Close.

Trigger a pipeline run

Select Add Trigger on the toolbar for the pipeline, and then select Trigger Now.

Monitor the pipeline run

1) Go to the Monitor tab. You see the pipeline that you manually triggered in the previous step.
2) To view activity runs associated with the pipeline run, select the SQLServerToBlobPipeline link under PIPELINE NAME. Monitor pipeline runs

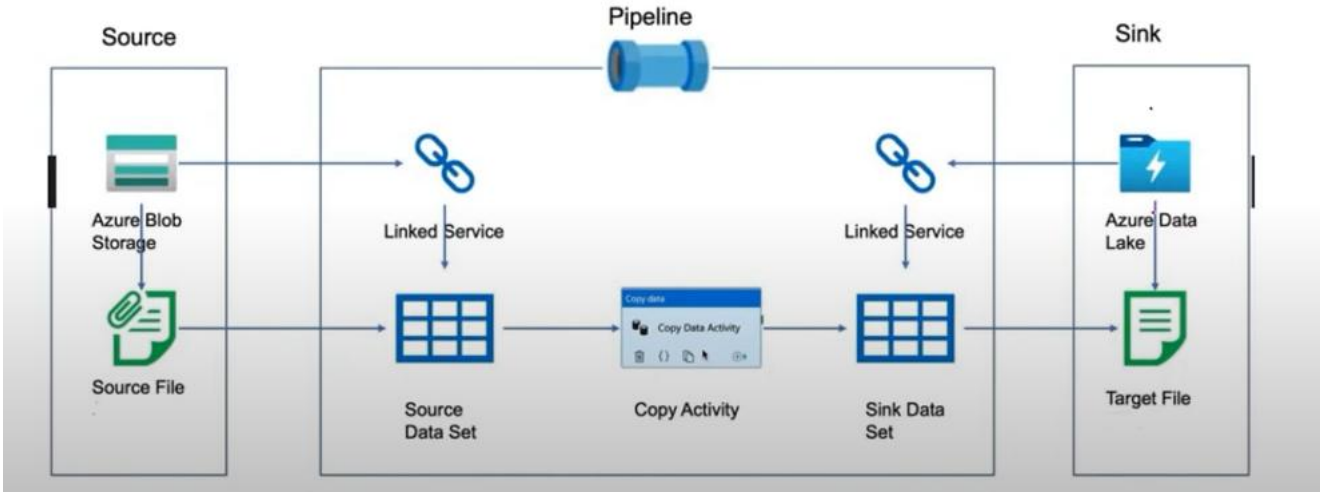


3) On the Activity runs page, select the Details (eyeglasses image) link to see details about the copy operation. To go back to the Pipeline Runs view, select All pipeline runs at the top.

Step 3:- Copy data from a Azure Blob storage to azure data lake gen2 storage in Azure Data Factory

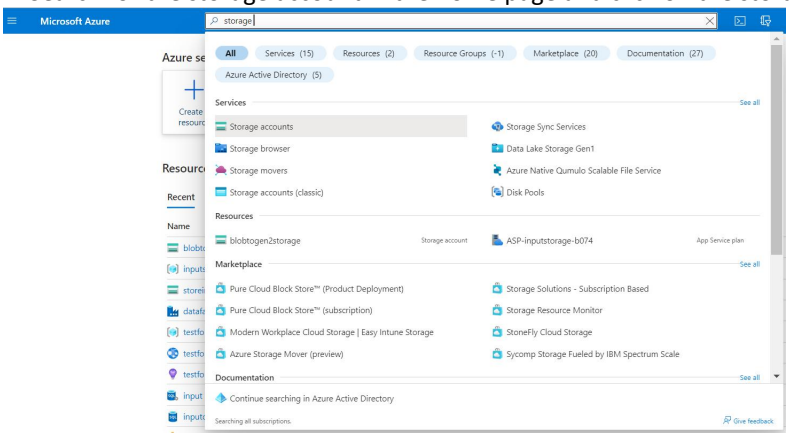
General Steps:-

1) Create a data factory.
2) Create a self-hosted integration runtime.
3) Create Azure Data Lake Gen2 Storage and Azure Storage linked services.
4) Create Azure Blob datasets(source) and Azure Data Lake Gen2(sink)
5) Create a pipeline with a copy activity to move the data.
6) Start a pipeline run.
7) Monitor the pipeline run.

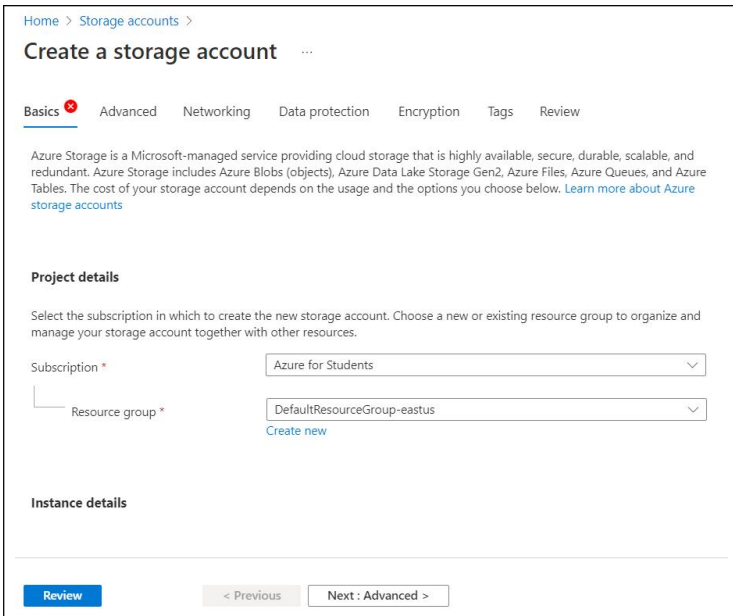


Note:- ** the steps upto to the Create Pipeline is same as above. Kindly refer that.
Just one new task to be done apasrt from above as creating a Azure Data Lake Gen2 Storage.
Steps:

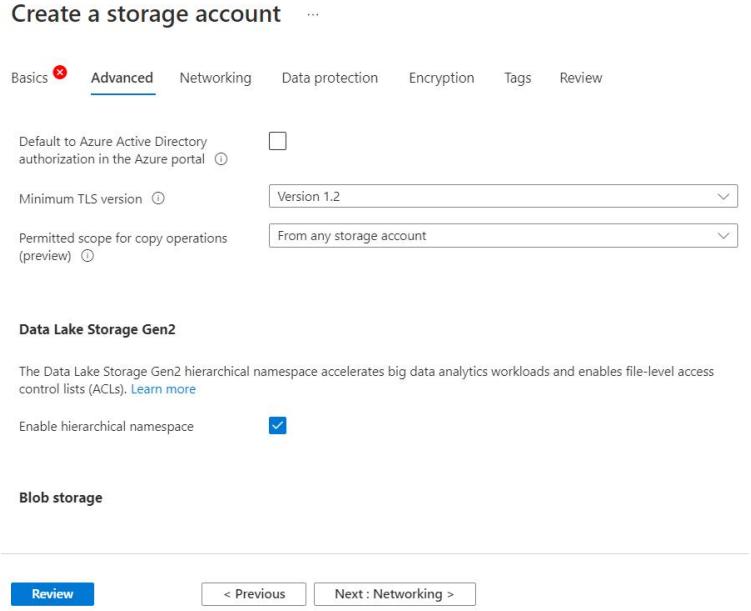
1. Search for the storage account in the home page and click on the storage account.



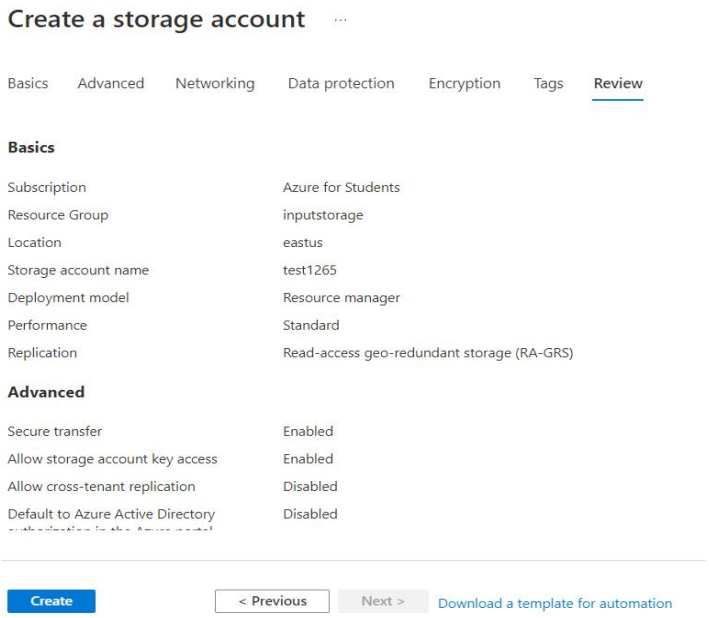
2. Once the Storage account open, look for create option
3. Clcik on the create option and fill the required details for the creation of the account.
4. Provide a Unique name to the storage account so that it can be easily recognizable.
5. Click on the Next: Advance once the details are filled.



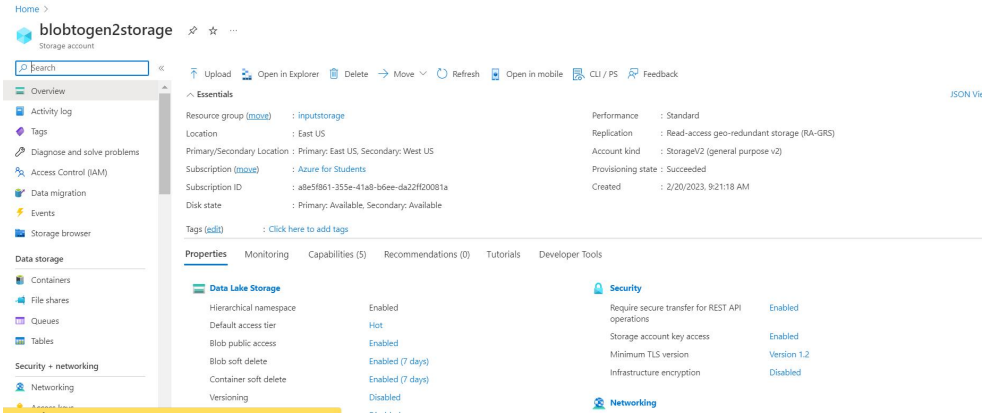
6. In the Advance tab scroll down and look for Data Lake Storage Gen2 and enable the same to create the storage account.



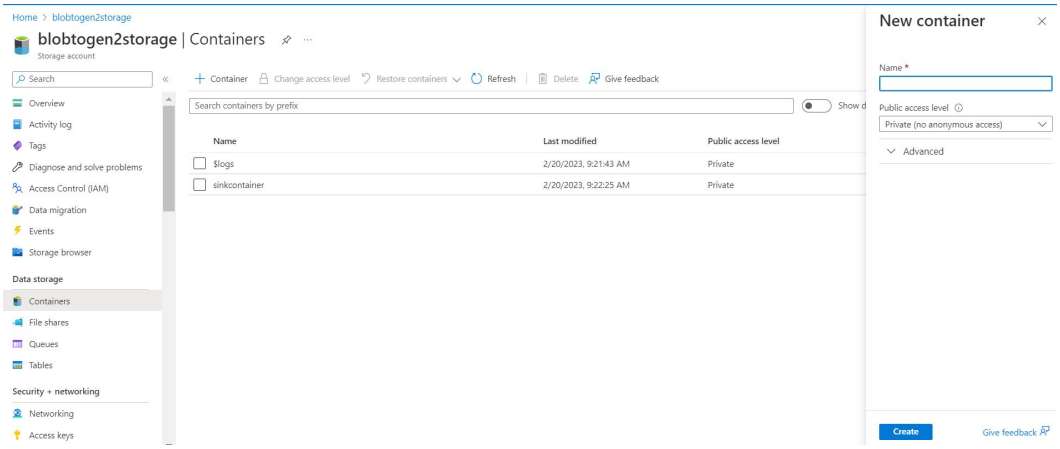
7. After this click on review and create Azure Data Lake Gen2 Storage account.



8. Once the process is completed wait for the deployment to be completed and then open the storage account after deployment and click on the container tab in left side panel of the storage account.

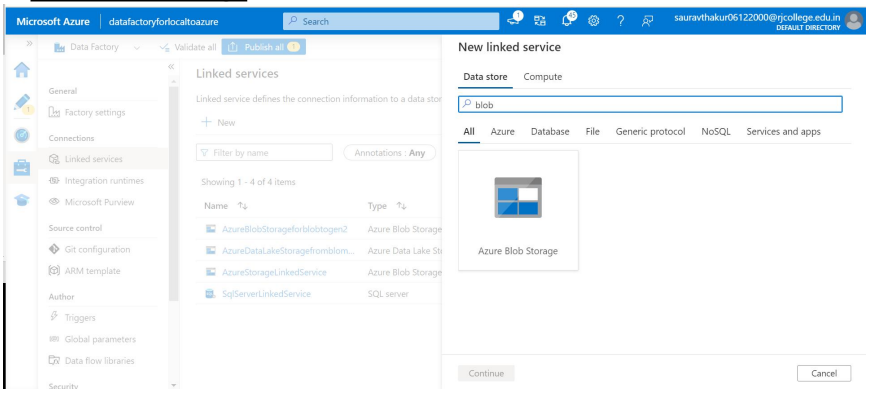


9. Once the container tab open look for the header and click on the + container option and create a new container for the file location where the file is needed to be store.

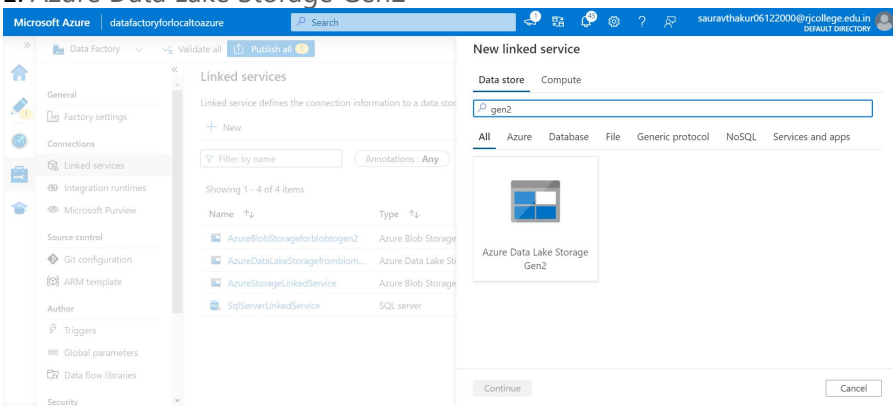


Create a Linked Services:

1. Azure Blob Storage

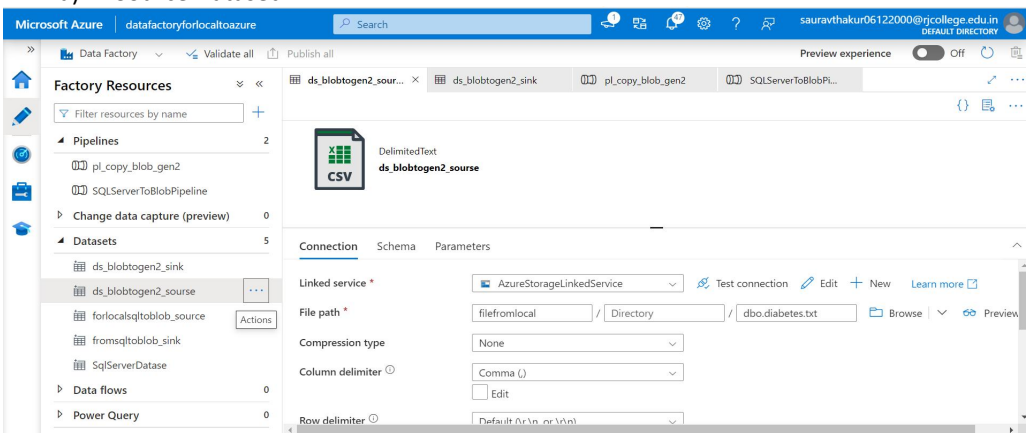


2. Azure Data Lake Storage Gen2

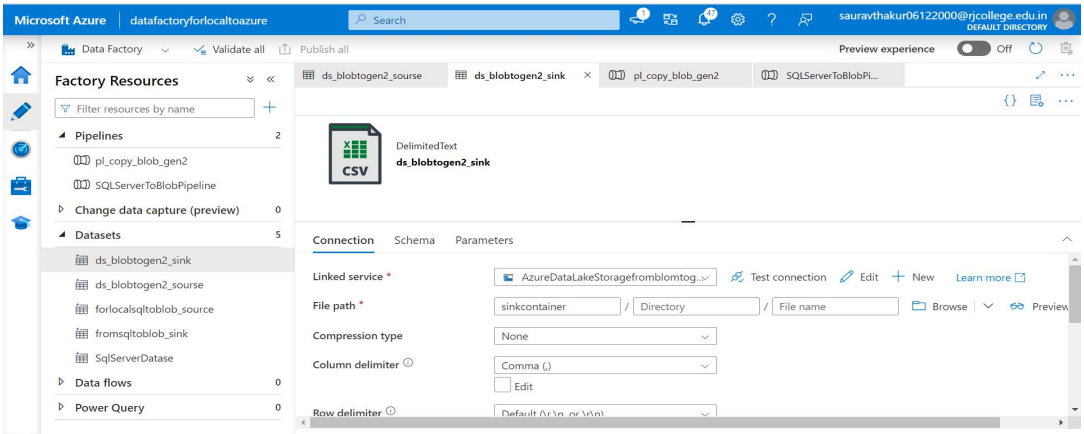


Create Factory Resources:
1. Datasets
2. Pipeline

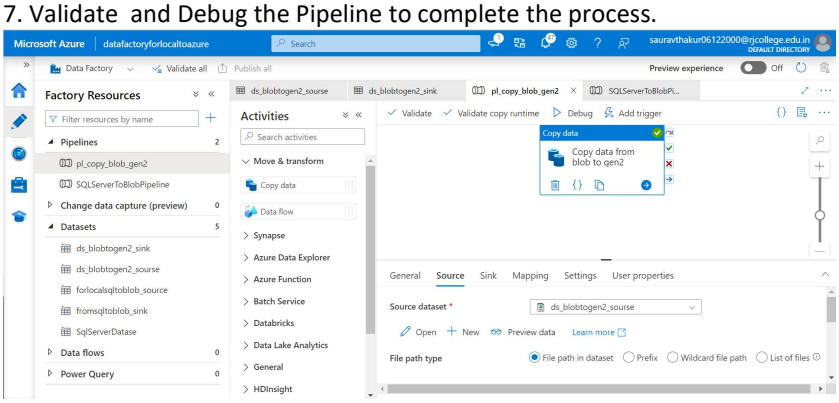
1. Datasets
   a) Source Dataset.



   b) Sink Dataset



Pipeline:-
1. Create a new pipeline
2. Search for the copy option in the activities section.
3. Drag and drop the copy option
4. Provide a name to the Pipeline
5. Use the source and sink data set in defining the pipeline properties.
6. Check the connection after using the datasets to avoid any error.
7. Validate and Debug the Pipeline to complete the process.

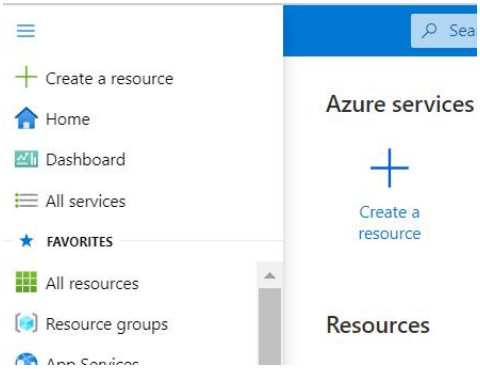


** note for detailed steps look for the above step in copying data from sql to blob storage and replace the properties of SQL with Blob and Blob with Azure Data Lake Gen2 Storage.

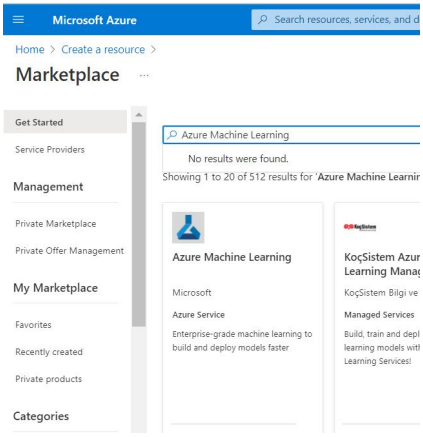
Step 4:- Creating a Automl model.

Create a workspace

1) Sign in to the Azure portal by using the credentials for your Azure subscription.
2) In the upper-left corner of the Azure portal, select the three bars, then + Create a resource.



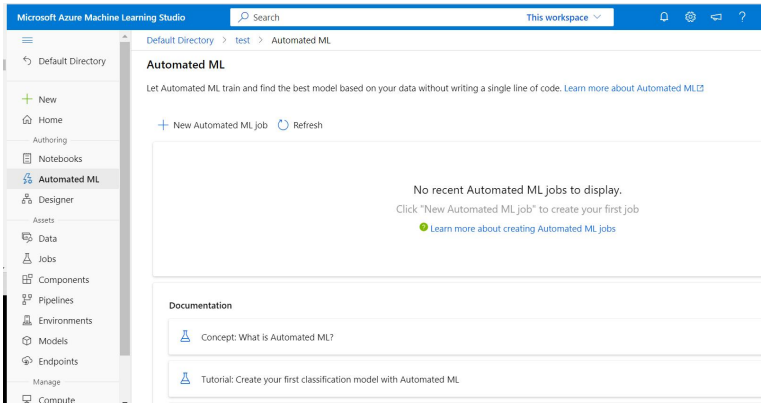
3) Use the search bar to find Azure Machine Learning.
4) Select Azure Machine Learning.



5) In the Machine Learning pane, select Create to begin.
6) Provide the necessary information to configure your new workspace:
7) After you're finished configuring the workspace, select Review + Create.
8) Select Create to create the workspace. When the process is finished, a deployment success message appears.
9) To view the new workspace, select Go to resource.
10) From the portal view of your workspace, select Launch studio to go to the Azure Machine Learning studio.

Sign in to the studio

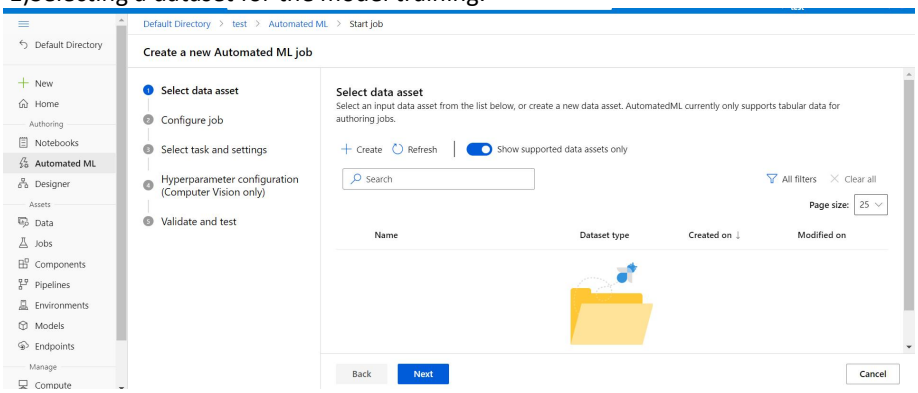
1) Sign in to Azure Machine Learning studio.
2) Select your subscription and the workspace you created.
3) Select Get started.
4) In the left pane, select Automated ML under the Author section.



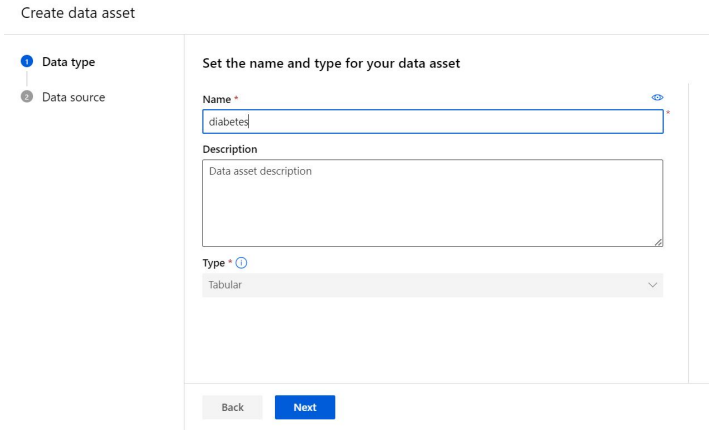
5) Select +New automated ML job.

Steps for the Automated ML job.

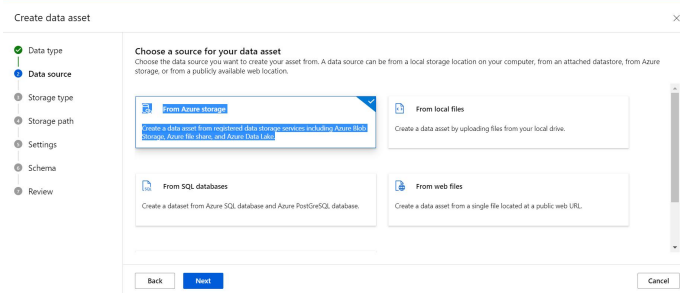
1)Selecting a dataset for the model training.



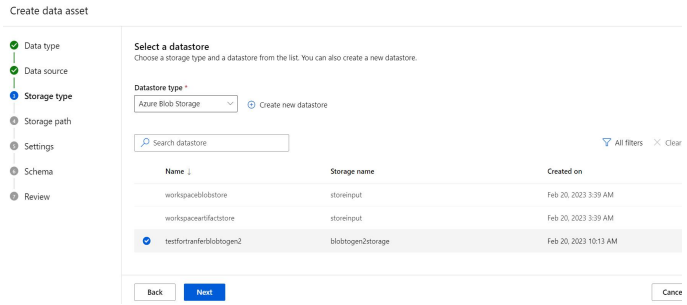
2)Providing a name for the dataset.



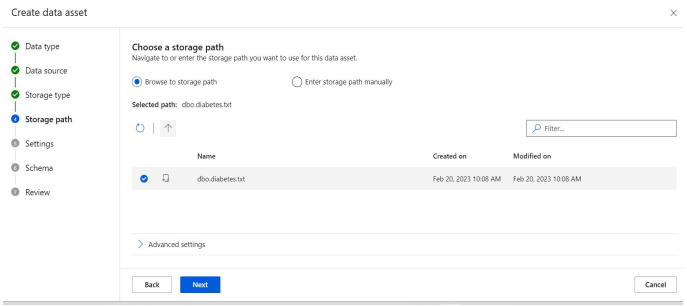
3)Provind the path where we have copied the data set in the previous step



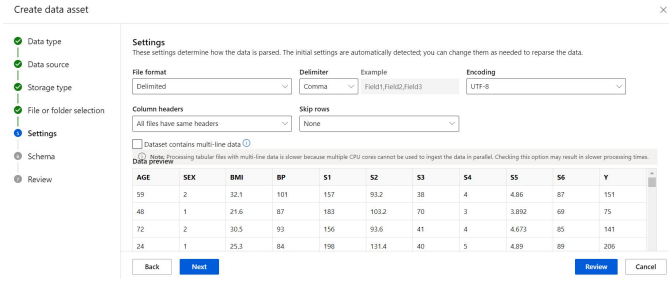
4)Selecting the storage account.



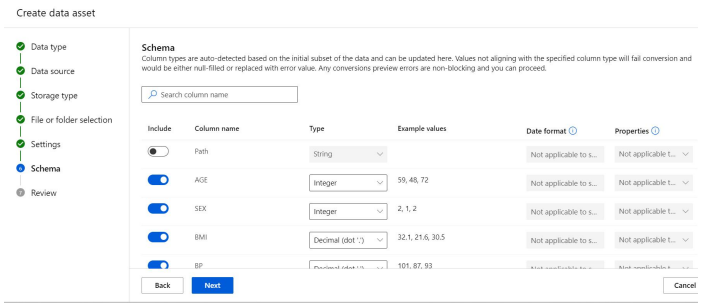
5)Selecting the Data set file.



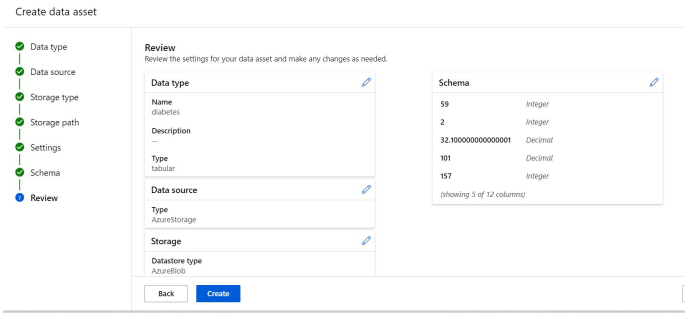
6)Setting the necessary properties



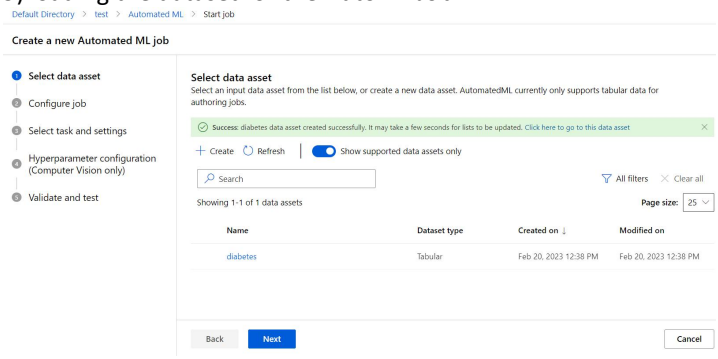
7)Checking the Schema and Data Type



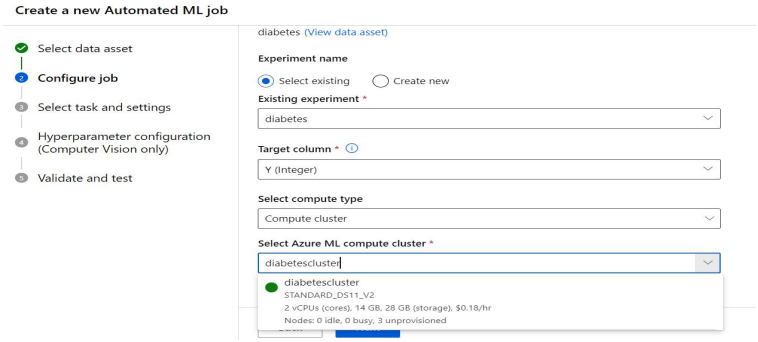
8)Creating a Data Asset



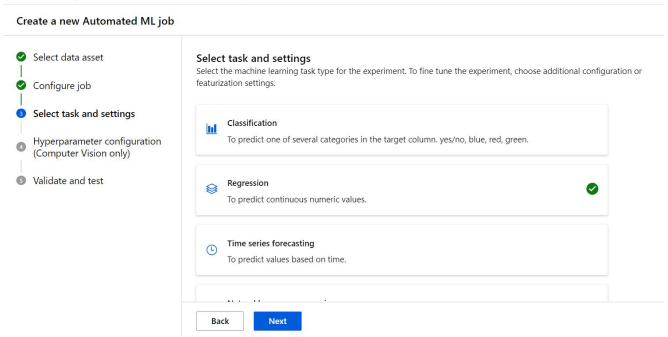
9)Loading the dataset for the AutoML Job



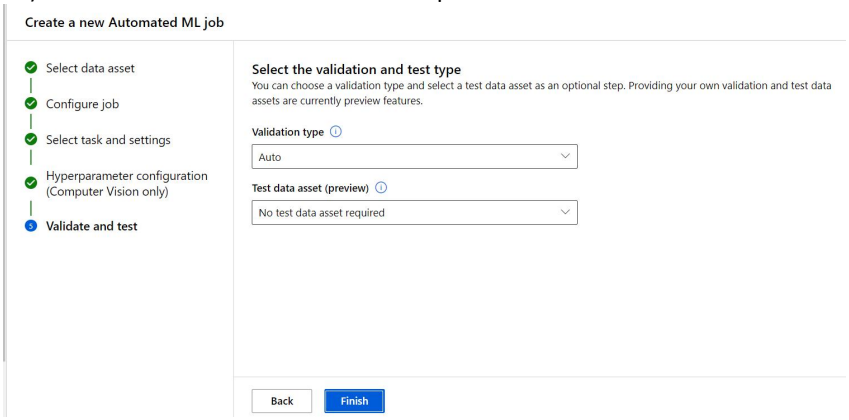
10)Defining the Target column, giving a name to the experiment, and selecting the compute cluster.



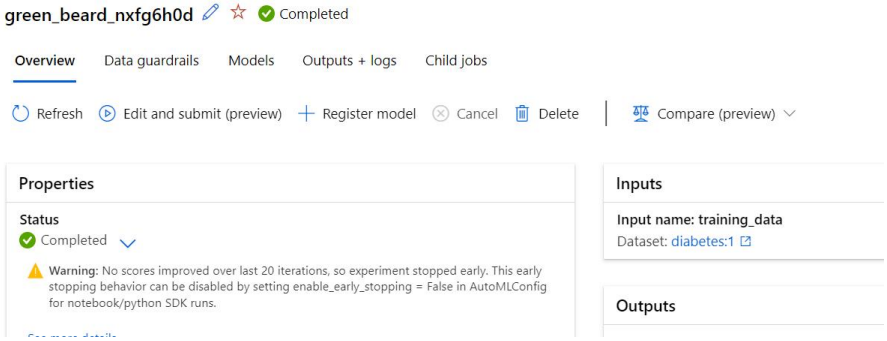
11)Seleting the task for the AutoML of the Azure Machine learning studio



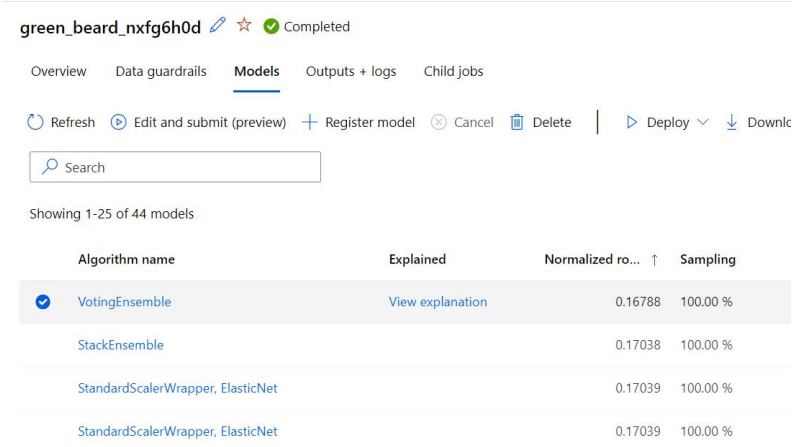
12)Select the Validation and if available provide the test dataset



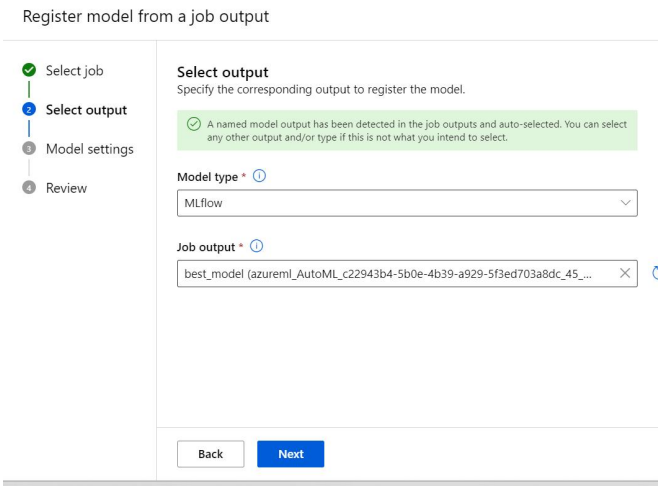
13) Once the experiment gets completed we get a Completed notification(The average time required for a model to gets trained is 45-50 minutes)

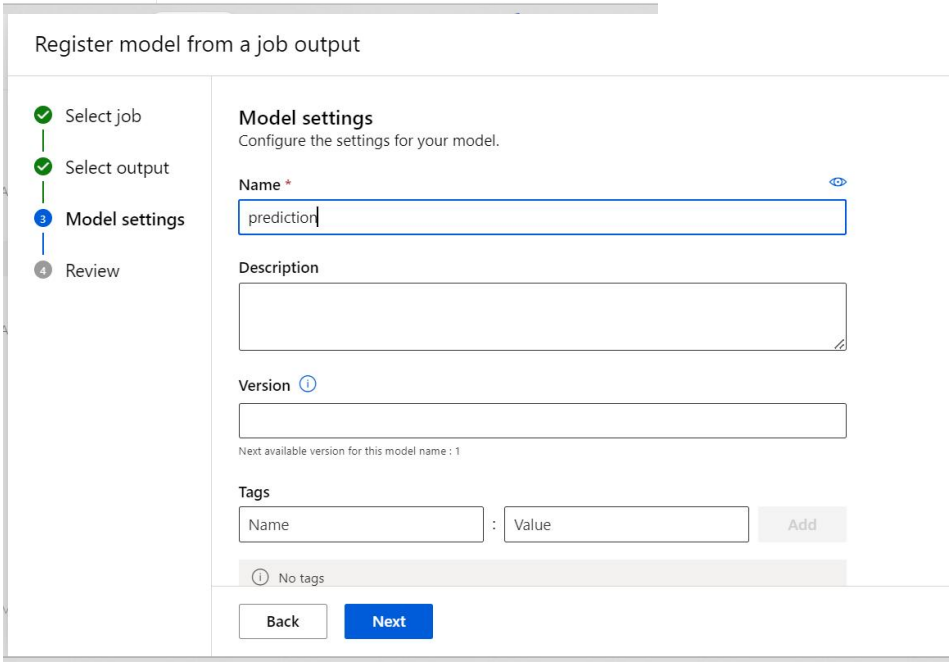


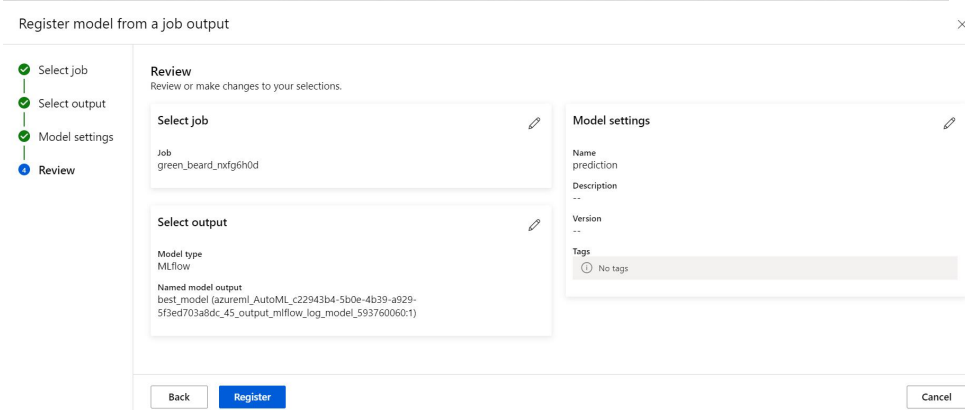
14) Go in the model tab in the experiment to get the best model for the experiment.



Register the best model which you get after completion of the azure automl experiment.

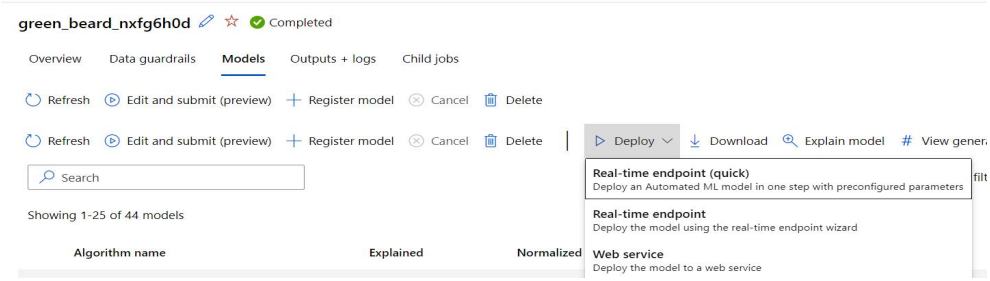




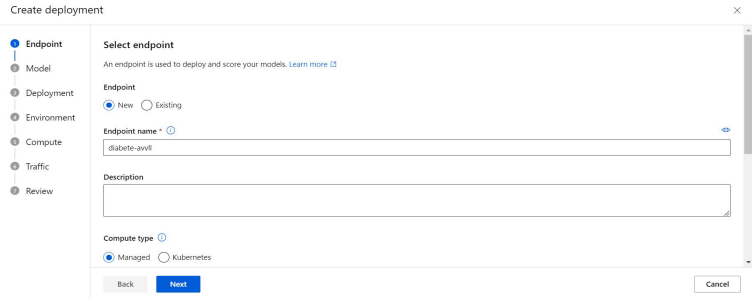


Model Deployment

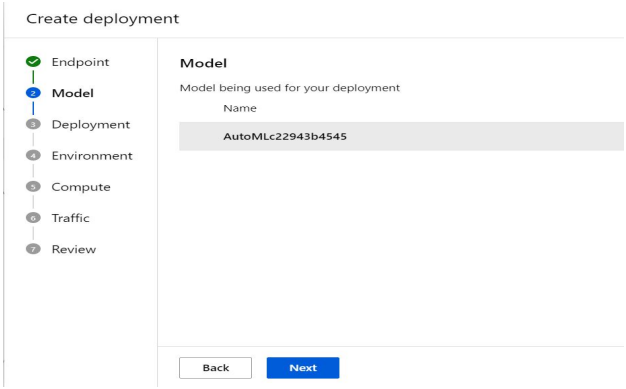
1. Once the best the model is registerd select the model and the deployment can be done using this model.



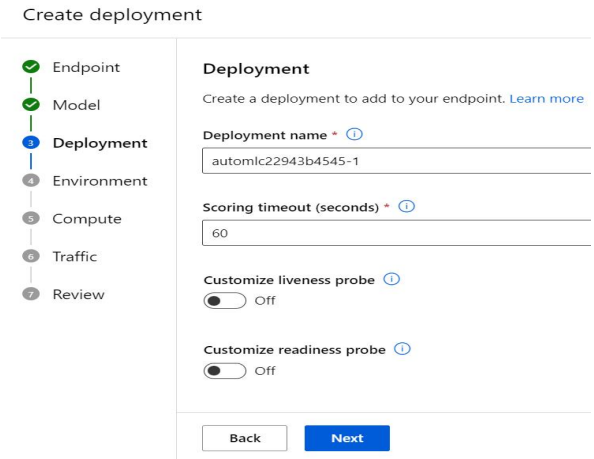
2. Provide a Endpoint Name for the deployment process and click next.



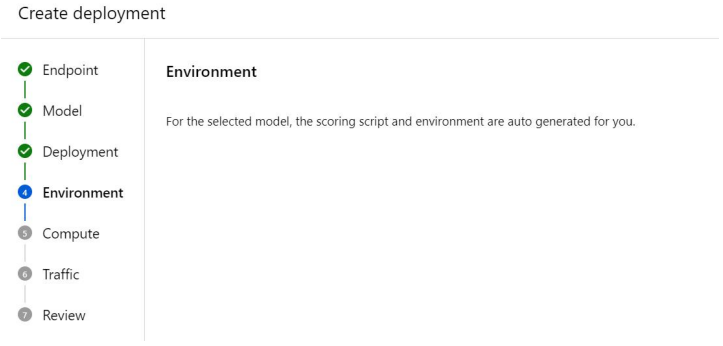
3. The selected and registed model name will appear click next.



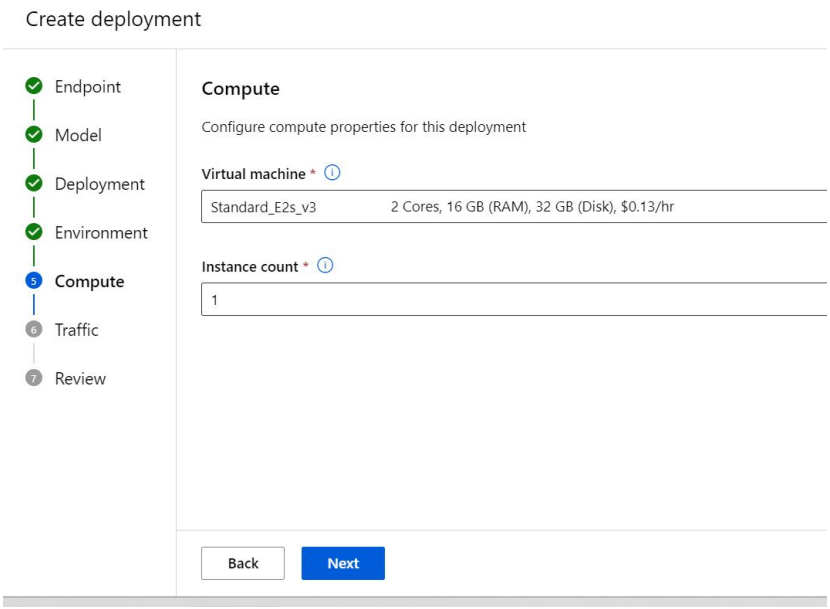
4. Provide a deployment Name scoring time and click next.



5. The environment will auto generated or else the environment can be generated using the conda env file which can be downloaded from the artifact of the best model registered and click next.

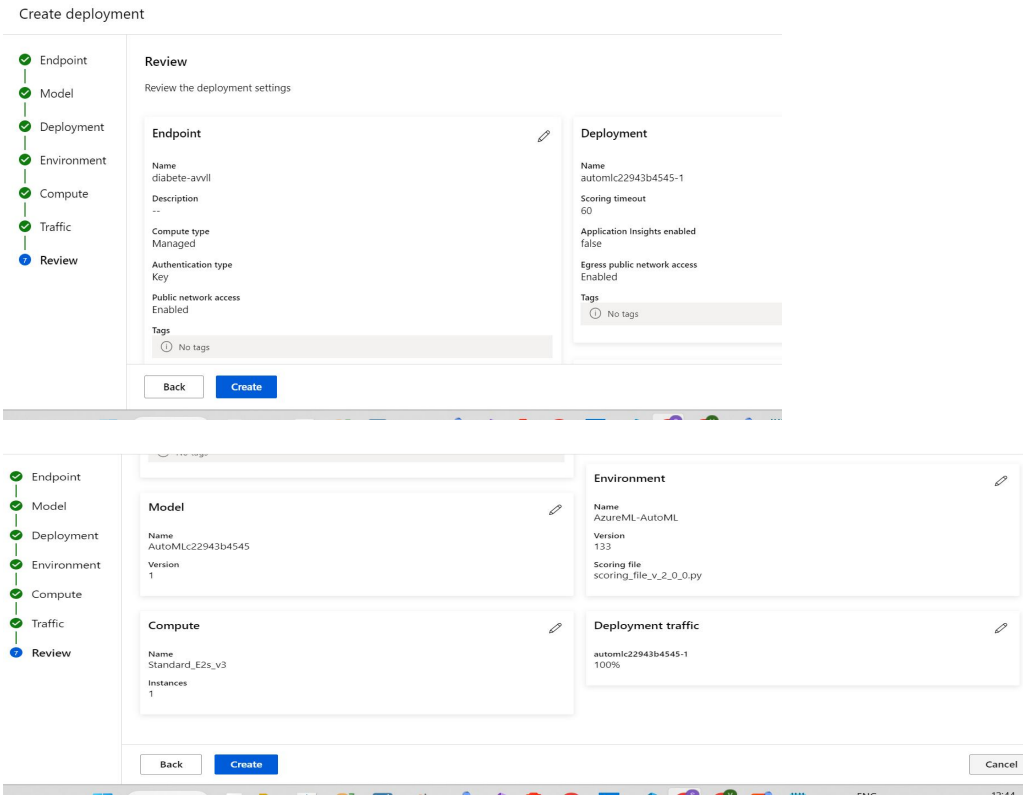


6. Select the Virtual Machine plan as per the need and provide the number of instance count needed and click next.

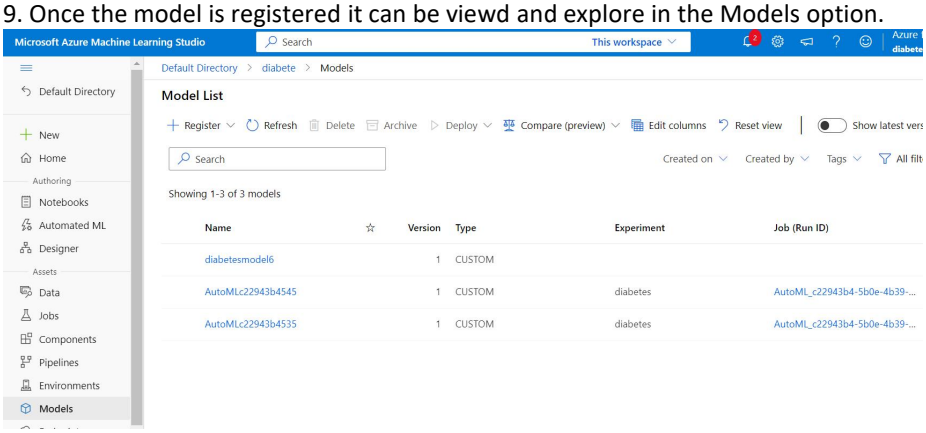


7. Allocated the amount of traffic it must be minimum 25 % and click next.

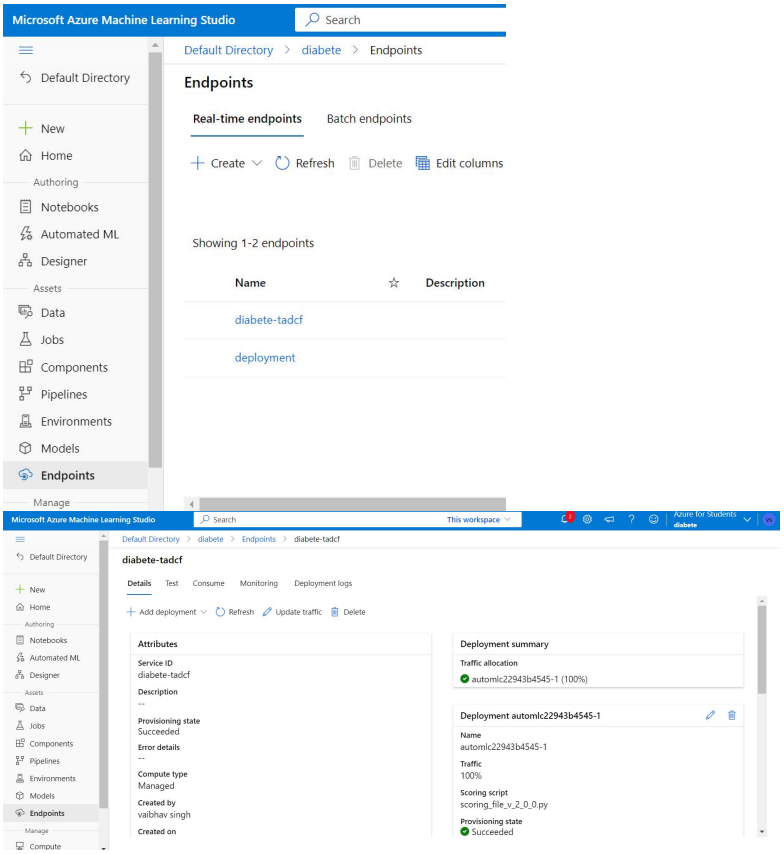
8. Review all the details provided and click create.



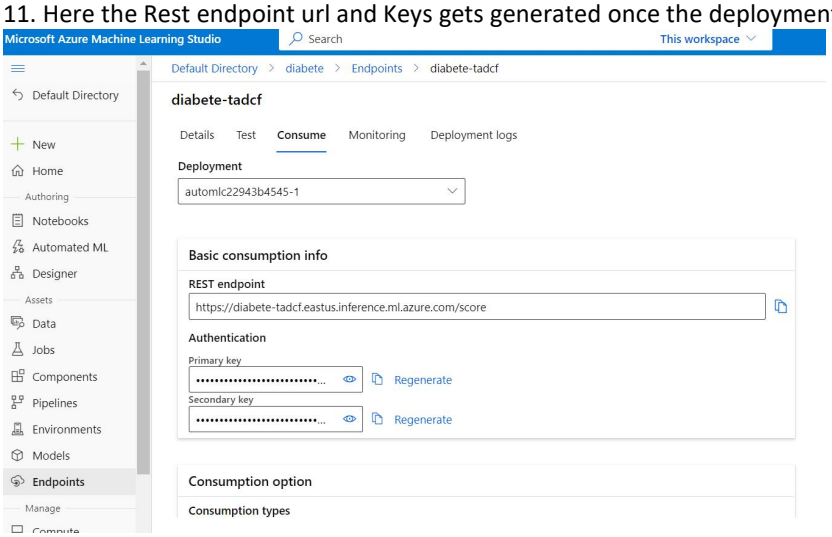
9. Once the model is registered it can be viewd and explore in the Models option.



10. Once the model is deployed successfully an end will be created and this end point can be used to deploy the model using web services.



11. Here the Rest endpoint url and Keys gets generated once the deployment is done correctly.



Sample Consume code:-

```
import urllib.request
```

```
import json
import os
import ssl

def allowSelfSignedHttps(allowed):
    # bypass the server certificate verification on client side
    if allowed and not os.environ.get('PYTHONHTTPSVERIFY', '') and getattr(ssl, '_create_unverified_context', None):
        ssl._create_default_https_context = ssl._create_unverified_context

allowSelfSignedHttps(True) # this line is needed if you use self-signed certificate in your scoring service.

# Request data goes here
# The example below assumes JSON formatting which may be updated
# depending on the format your endpoint expects.
# More information can be found here:
# https://docs.microsoft.com/azure/machine-learning/how-to-deploy-advanced-entry-script
data =  {
  "Inputs": {
    "data": [
      {
        "AGE": 0,
        "SEX": 0,
        "BMI": 0.0,
        "BP": 0.0,
        "S1": 0,
        "S2": 0.0,
        "S3": 0.0,
        "S4": 0.0,
        "S5": 0.0,
        "S6": 0
      }
    ]
  },
  "GlobalParameters": 0.0
}

body = str.encode(json.dumps(data))

url = 'https://diabete-tadcf.eastus.inference.ml.azure.com/score'
# Replace this with the primary/secondary key or AMLToken for the endpoint
api_key = ''
if not api_key:
    raise Exception("A key should be provided to invoke the endpoint")

# The azureml-model-deployment header will force the request to go to a specific deployment.
# Remove this header to have the request observe the endpoint traffic rules
headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key), 'azureml-model-deployment':
'automlc22943b4545-1' }

req = urllib.request.Request(url, body, headers)

try:
    response = urllib.request.urlopen(req)

    result = response.read()
    print(result)
except urllib.error.HTTPError as error:
    print("The request failed with status code: " + str(error.code))

    # Print the headers - they include the requert ID and the timestamp, which are useful for debugging the failure
    print(error.info())
    print(error.read().decode("utf8", 'ignore'))
```

Theoritical explanation and creation of an Automl Model and consuming the model in generating a extra column in PowerBi and visualizing it.

1. A. How to create an Azure Machine Learning compute cluster?

Azure Machine Learning compute cluster is a managed-compute infrastructure that allows you to easily create a single or multi-node compute.

The compute cluster is a resource that can be shared with other users in your workspace.

Automated machine learning trains many machine learning models to find the "best" algorithm and parameters.

Azure Machine Learning parallelized the running of the model training over a compute cluster.

1. B. Steps to create an Azure Machine Learning compute cluster

In Azure Machine Learning Studio, on the left menu, select Compute.

Open the Compute clusters tab. Then select New.

On the Create compute cluster page: -

Select a VM size. For this tutorial, a Standard_D11_v2 machine is fine.

Select Next.

Provide a valid compute name.

Keep Minimum number of nodes at 0.

Change Maximum number of nodes to 4.

Select Create.

2. A. Steps to create an Automated Machine learning run

Automated machine learning, also referred to as automated ML or AutoML, is the process of automating the time-consuming, iterative tasks of machine learning model development.

With automated machine learning, we'll accelerate the time it takes to get production-ready ML models with great ease and efficiency.

In Azure Machine Learning Studio, in the menu on the left, select Automated ML.

Select New Automated ML run.

Next, select the diabetes dataset we've created earlier.

Then select Next

2.B Steps to create an Automated Machine learning run

On the Configure run page: -

Under Experiment name, select Create new.

Name the experiment.

In the Target column field, select Y.

In the Select compute cluster field, select the compute cluster we've created earlier.

Finally, select a machine learning task. In this case, the task is Regression.

Select Finish.

Note: - Automated machine learning takes more than 30 minutes to finish training the 100 models.

3.A. How to deploy the best model at real time?

In order to start using a model for practical decision-making, it needs to be effectively deployed into production.

The goal of building a machine learning model is to solve a problem, and a machine learning model can only do so when it is in production and actively in use by consumers. As such, model deployment is as important as model building

When automated machine learning finishes, we can see all the machine learning models that have been tried by selecting the Models tab.

The models are ordered by performance; the best-performing model is shown first.

3.B. Steps to deploy the best model at real time

After we select the best model, the Deploy button is enabled.

Select Deploy to open a Deploy a model window: -

Name your model service.

Select Azure Container Service.

Select Deploy.

We should see a message that states that the model was deployed successfully

4.A. How to connect to an Azure Machine Learning model in the Power Query editor?

Power Query is a data transformation and data preparation engine.

Power Query comes with a graphical interface for getting data from sources and a Power Query Editor for applying transformations.
Using Power Query, you can perform the extract, transform, and load (ETL) processing of data.

5. Steps to connect to an Azure Machine Learning model in the Power Query editor
--Create the data model
Open Power BI Desktop and select Get Data.
In the Get Data dialog box, search for web.
Select the Web source > Connect.
In the From Web dialog box, copy and paste the following URL in the box: -
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.tab.txt
Select OK.
In Access Web content, select Anonymous > Connect.
Select Transform data to open the Power Query Editor window.
In the Home ribbon of the Power Query Editor, select the Azure Machine Learning button.

6.Steps to connect to an Azure Machine Learning model in the Power Query editor
After signing in to your Azure account using single sign-on, we see a list of available services.
Select the service that you created from the Train and deploy a machine learning model tutorial.
Power Query populates the columns automatically for you.
Select OK.

For time series models, Power BI may not automatically detect the date format for the time column. To proceed, convert the time column to Date/Time type in Power BI before invoking Azure Machine Learning.

7.Steps to connect to an Azure Machine Learning model in the Power Query editor
Selecting OK calls the Azure Machine Learning service.
It triggers a warning on data privacy for both the data and the endpoint.
Select Continue. In the next screen, select Ignore Privacy Levels checks for this file > Save.
Once the data is scored, Power Query creates an additional column named AzureML.(your model name).
The data that the service returns is a list.
To get the predictions, select the double-headed arrow in the AzureML.my-diabetes-model column header > Expand to New Rows.

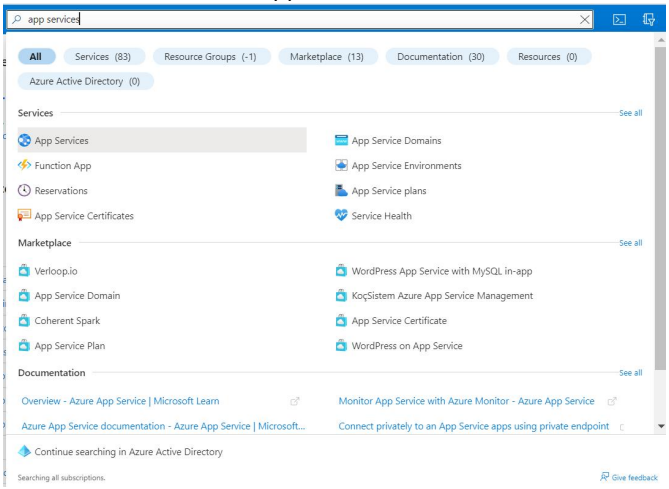
8.Steps to connect to an Azure Machine Learning model in the Power Query editor
Follow these next steps to finish cleaning up your data model: -
Rename the AzureML.my-diabetes-model column to predicted.
Rename the Y column to actual.
Change the type of the actual column: Select the column, then on the Transform ribbon select Data Type > Decimal Number.
Change the type of the predicted column: Select that column, then on the Transform ribbon select Data Type > Decimal Number.
On the Home ribbon, select Close & Apply.

9.How to create a report with a visualization based on Azure ML Model?
Now we can create some visualizations to show our data.
In the Visualizations pane, select a Line chart.
With the line chart visual selected:
Drag the AGE field to the Axis.
Drag the actual field to Values.
Drag the predicted field to Values.

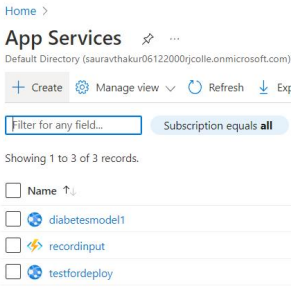
10.Publishing Azure ML report to the Power BI service.
We can add more visualizations if we wish. In the interest of brevity, in we'll publish the report.
Save the report.
Select File > Publish > Publish to Power BI.
Sign in to the Power BI service.
Select My Workspace.
When the report is published successfully, select the Open <MY_PBIX_FILE.pbix> in Power BI link. The report opens the report in Power BI in our browser.

Step 5:- a) Create a web app in Azure

Sign in to the Azure portal and follow these steps to create your Azure App Service resources.
In the Azure portal:
Enter app services in the search bar at the top of the Azure portal.
Select the item labeled App Services under the under Services heading on the menu that appears below the search bar.

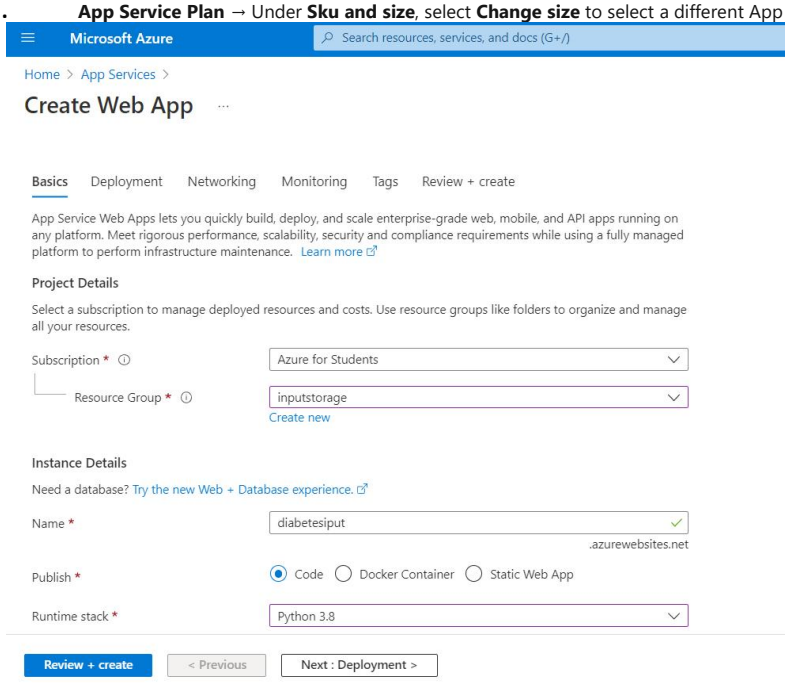


On the **App Services** page, select **+ Create**



On the **Create Web App** page, fill out the form as follows.

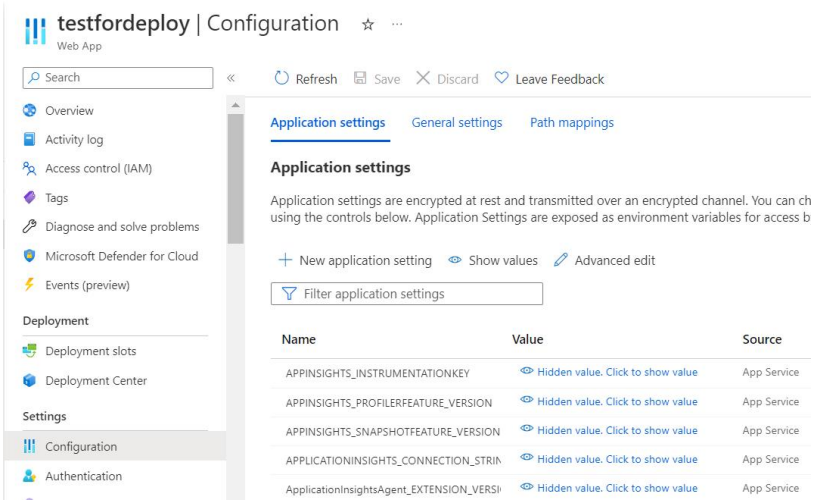
1. **Resource Group** → Select **Create new** and use a name of msdocs-python-webapp-quickstart.
2. **Name** → msdocs-python-webapp-quickstart-XYZ where XYZ is any three random characters. This name must be unique across Azure.
3. **Runtime stack** → **Python 3.8**.
4. **Region** → Any Azure region near you.
5. **App Service Plan** → Under **Sku and size**, select **Change size** to select a different App Service plan.



Step 5 :- b) Deploy your application code to Azure

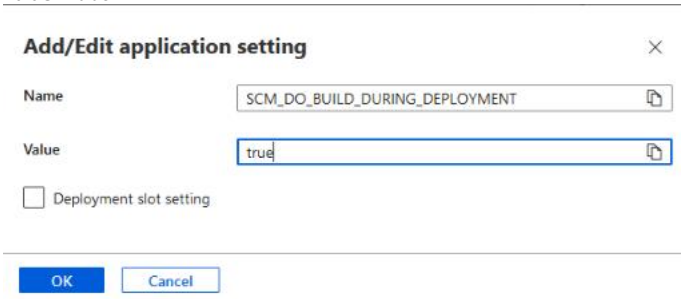
On the page for the web app in the Azure portal:

1. Select **Configuration** under the **Settings** header in the left toolbar to bring up the Application settings.
2. Under **Application settings**, select **New application setting**.



Using the dialog, enter a new setting with:

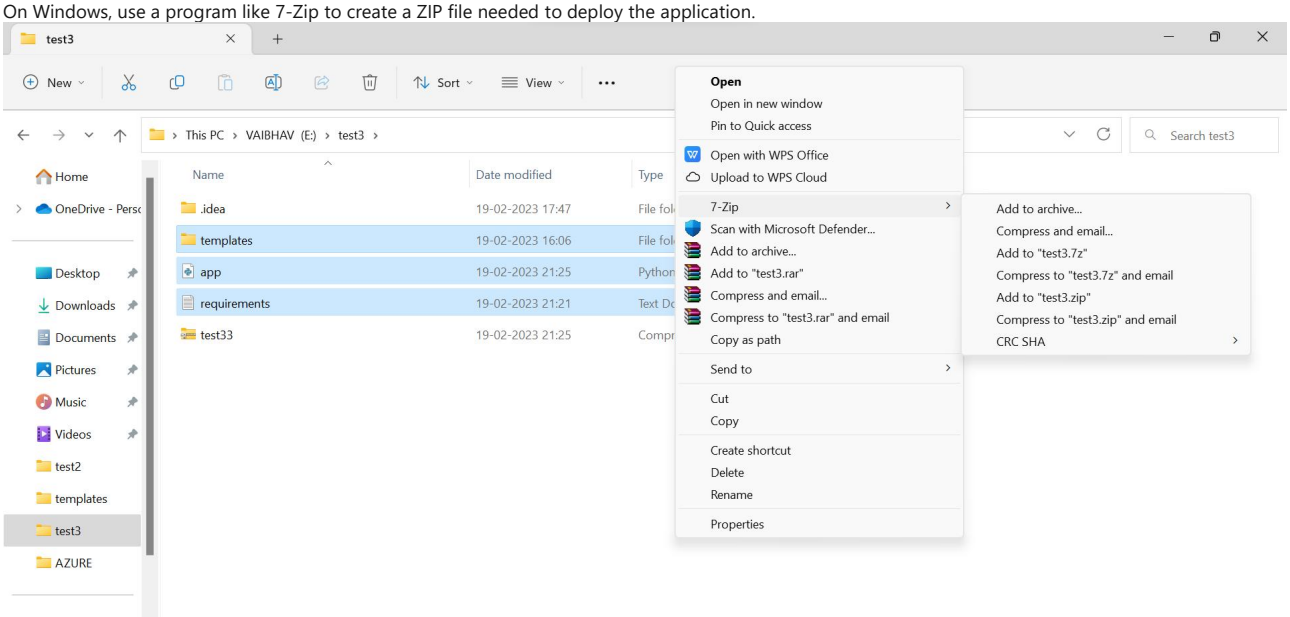
Name → *SCM_DO_BUILD_DURING_DEPLOYMENT*
Value → *true*



Select the **Save** to save your settings.

Step 5:- c) I)Create a ZIP file of your application

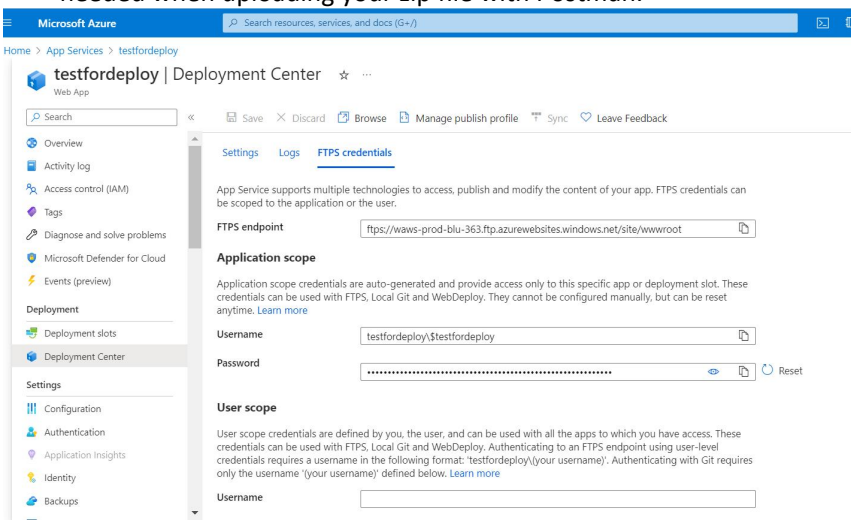
On Windows, use a program like 7-Zip to create a ZIP file needed to deploy the application.



Step 5:- c) II)Upload the ZIP file to Azure

To use Postman to upload your ZIP file to Azure, you will need the deployment username and password for your App Service. These credentials can be obtained from the Azure portal.

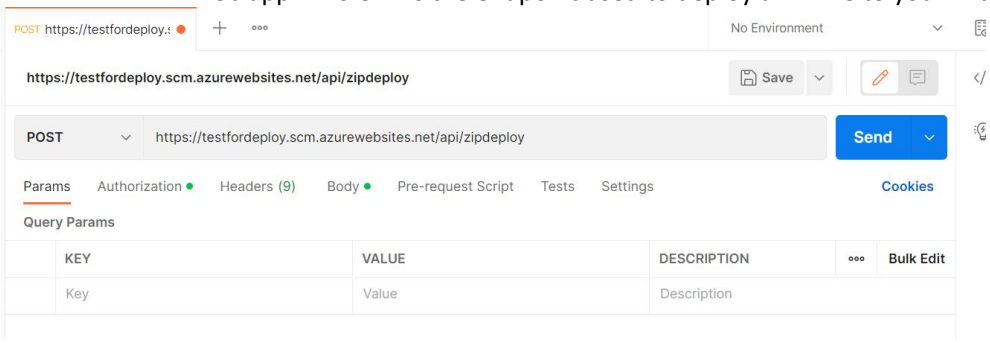
1. On the page for the web app, select Deployment center from the menu on the left side of the page.
2. Select the FTPS credentials tab.
3. The Username and Password are shown under the Application scope heading. For zip file deployments, only use the part of the username after the \ character that starts with a $, for example $testfordeploy. These credentials will be needed when uploading your zip file with Postman.



Step 5:- c) III) Postman

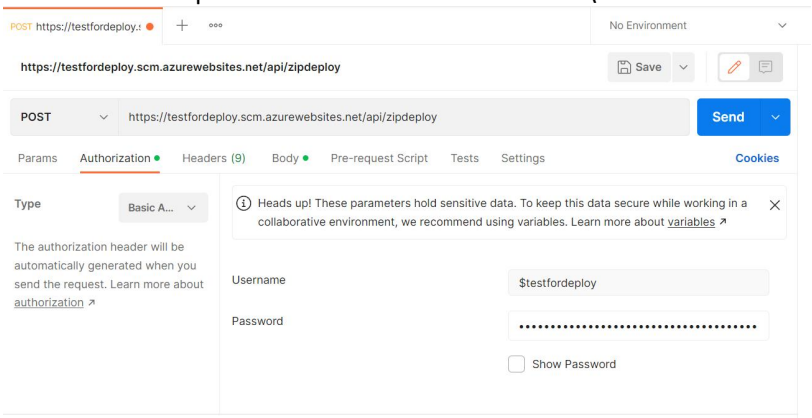
In Postman, upload your file using the following steps.
Click on the plus (+) sign icon to create a new request.

a) Select POST for the request type.
b) Enter the url https://\<app-name\>.scm.azurewebsites.net/api/zipdeploy where <app-name> is the name of the web app. This URL is the endpoint used to deploy a ZIP file to your Azure service.

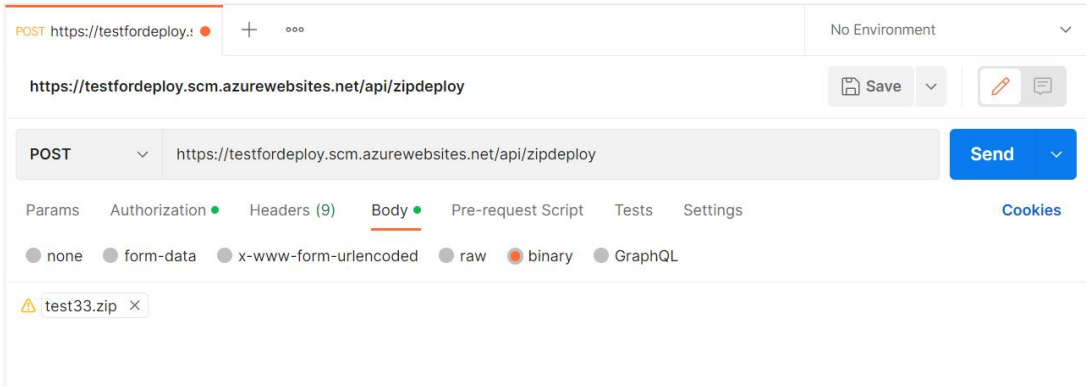


In the Authorization tab:

a) Set the Type to Basic.
b) Enter the deployment username and password obtained from the Azure portal above. Be sure to only use the portion of the username after the \ character that starts with a $.



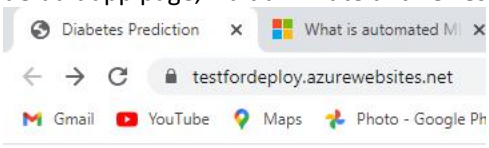
In the Body tab:
1. Select binary as the content type.
2. Use the Select File button to select your zip file.

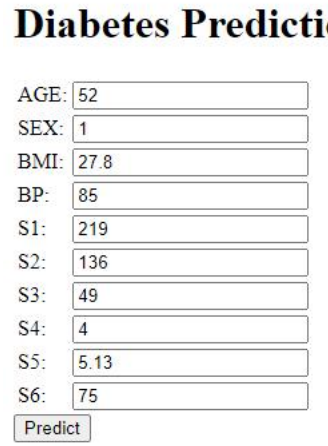


The filename of the file to be uploaded will be shown in the Body section.
Select the Send button to upload your zip file to Azure.
Depending on your network bandwidth, files usually take between 10 and 30 seconds to upload to Azure

Step 5:- d) Browse to the app

Browse to the deployed application in your web browser at the URL http://<app-name>.azurewebsites.net. If you see a default app page, wait a minute and refresh the browser.





app.py Code:-

```
from flask import Flask, render_template, request, jsonify
import json
import urllib.request

app = Flask(__name__)

url = "https://diabete-tadcf.eastus.inference.ml.azure.com/score"
api_key = '5l3xsfu0X0Hy4cww3J8rColjXRu0eKta'

@app.route('/')
def home():
   return render_template('index.html')

@app.route('/predict', methods=['POST','GET'])
def predict():
      data = { "Inputs" :
            {
        "data": [
```

```
        {
          "AGE": request.form['AGE'],
          "SEX": request.form['SEX'],
          "BMI": request.form['BMI'],
          "BP": request.form['BP'],
          "S1": request.form['S1'],
          "S2": request.form['S2'],
          "S3": request.form['S3'],
          "S4": request.form['S4'],
          "S5": request.form['S5'],
          "S6": request.form['S6']
        }
        ]
    }
  }
    headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key),'azureml-model-deployment':
'automlc22943b4545-1'}
    body = str.encode(json.dumps(data))
    req = urllib.request.Request(url, body, headers)

    try:
      response = urllib.request.urlopen(req)
      result = response.read().decode()
      return render_template('predict.html', prediction=result)
    except:
      error_msg = response.json().get('message', 'Prediction failed')
      print(f"Prediction failed: {error_msg}")
      return render_template('error.html', message='Prediction failed')

if __name__ == "__main__":
  app.run(debug=True)
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Diabetes Prediction</title>
</head>
<body>
        <h1>Diabetes Prediction</h1>
        <form action="/predict" method="post">
                <table>
                        <tr>
                                <td>AGE:</td>
                                <td><input type="number" name="AGE" ></td>
                        </tr>
                        <tr>
                                <td>SEX:</td>
                                <td><input type="number"  name="SEX" required></td>
                        </tr>
                        <tr>
                                <td>BMI:</td>
                                <td><input type="number" step="0.01" name="BMI" required></td>
                        </tr>
                        <tr>
                                <td>BP:</td>
                                <td><input type="number" step="0.01" name="BP" required></td>
                        </tr>
                        <tr>
                                <td>S1:</td>
```

```
                        <td><input type="number" name="S1" required></td>
                    </tr>
                    <tr>
                        <td>S2:</td>
                        <td><input type="number" step="0.01" name="S2" required></td>
                    </tr>
                    <tr>
                        <td>S3:</td>
                        <td><input type="number" step="0.01" name="S3" required></td>
                    </tr>
                    <tr>
                        <td>S4:</td>
                        <td><input type="number" step="0.01" name="S4" required></td>
                    </tr>
                    <tr>
                        <td>S5:</td>
                        <td><input type="number" step="0.01" name="S5" required></td>
                    </tr>
                    <tr>
                        <td>S6:</td>
                        <td><input type="number" name="S6" required></td>
                    </tr>
                </table>
                <input type="submit" value="Predict">
        </form>
</body>
</html>
```

Predict.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="html">
  <title>Diabetes Prediction Result</title>
 </head>
 <body>
  <h1>Diabetes Prediction Result</h1>
  <p>{{ prediction }}</p>
  <p><a href="/">Try Again</a></p>
 </body>
</html>
```

Error.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Error </title>
</head>
<body>
  <h1>Error page hai ye</h1>
  <p>{{ error }}</p>
</body>
</html>
```