One Month Project On

MLOPS Using Azure Cloud

Submitted by :-

Harsh Tripathi (MT2022048)

Vipul Manohar Ahuja (MT2022132)

Under Guidance of:-

Prof. B. Thangaraju

Problem Statement :-

Diabetes is a prevalent chronic disease with significant implications for patient health and healthcare costs. Efficiently managing and predicting diabetes progression is crucial for timely intervention and personalized care. The challenge is to develop a robust MLOps (Machine Learning Operations) solution that leverages data-driven insights to enhance diabetes prediction models, deploy them in a scalable and automated manner, and continuously monitor and update the models to ensure accuracy and relevance. The goal is to establish an end-to-end MLOps framework that seamlessly integrates data processing, model development, deployment, and ongoing monitoring to optimize diabetes management and healthcare outcomes.

Objective :-

Primary objective of this project are as follows :-

- Create a ML model for Diabets data which can be found in sklearn and apply MLOPS using two ways.
    1. Azure ML
    2. Databricks
- Create a free Azure account.
- Using Azure Machine Learning Studio for creating ML pipeline which can be monitor and executed.
- We will also Databricks for monitor and execution our models.
- At last we will compare both ways.

MLOPS using Azure Cloud

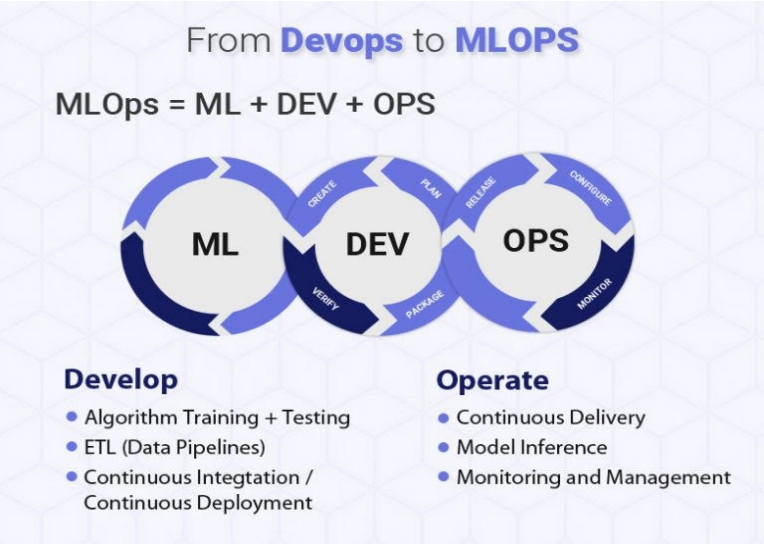
Introduction

In our day-to-day life, we are doing automation in every domain with the help of Artificial Intelligence and Machine Learning. Now, building a Machine Learning Model requires so many steps which are se follows: -

(1) Defining the problem. (2) Data Preprocessing. (3) Exploratory Data Analysis. (4) Splitting the data. (5) Applying different Models/Algorithms. (6) Feature Scaling. (7) Train the Model. (8) Hyper-Parameter Tuning. (8) Evaluate the Model. (9) Deploy the Model. (10) Monitor and Maintain.

And if we want to make changes in the model so we must perform every step one by one again for deployment. So, we use MLOPS concept which will automatically build everything after we make any changes in the model.

We will see the basic introduction of MLOPS which is as follows: -



MLOPS, short for Machine Learning Operations, is a set of practices that aims to streamline and enhance the collaboration between data science and operations teams in the development and deployment of machine learning (ML) models. It is an emerging discipline that combines aspects of machine learning, software engineering, and IT operations to create a more efficient and scalable ML lifecycle.

Key Components of MLOps:

1)Collaboration and Communication:

MLOps emphasizes effective communication and collaboration between data scientists, engineers, and operations teams. This helps ensure that the entire ML pipeline, from model development to deployment and monitoring, is well-coordinated and aligned with business objectives.

2)Automation:

Automation is a core principle of MLOps. It involves automating repetitive tasks throughout the ML lifecycle, such as data preprocessing, model training, testing, and deployment. Automation increases efficiency, reduces errors, and accelerates the overall development process.

3)Continuous Integration and Continuous Deployment (CI/CD):

MLOps leverages CI/CD practices to enable the continuous integration of code changes, automated testing, and seamless deployment of ML models into production. CI/CD ensures that the ML pipeline is always in a deployable state, promoting a faster and more reliable release process.

4)Model Monitoring and Management:

MLOps emphasizes the importance of monitoring ML models in production. This involves tracking model performance, detecting drift in data distribution, and addressing issues promptly. Additionally, MLOps provides tools for model versioning, enabling easy rollback and management of multiple model versions.

5)Scalability and Reproducibility:

MLOps addresses the challenges of scaling ML workflows by providing infrastructure and tools that support the deployment of models in diverse environments. It also emphasizes reproducibility, ensuring that ML experiments and results can be replicated consistently.

6)Security and Compliance:

Security is a critical consideration in MLOps, with a focus on safeguarding sensitive data and ensuring compliance with regulations. MLOps practices incorporate security measures throughout the ML lifecycle, from data handling to model deployment.

Why this is Project is Important:-

- Early Detection and Prevention: Predictive models can assist in identifying individuals at risk of developing diabetes before the onset of symptoms. Early detection allows for proactive intervention and lifestyle modifications, potentially preventing or delaying the progression of the disease.
- Personalized Treatment Plans: Machine learning models can help tailor treatment plans for individuals based on their unique characteristics and health data. Personalized approaches may lead to more effective and efficient management of diabetes, optimizing medication regimens and lifestyle recommendations.

- Resource Allocation in Healthcare: By accurately predicting which individuals are more likely to develop complications or require intensive medical interventions, healthcare resources can be allocated more efficiently. This can help prioritize patient care and reduce the burden on healthcare systems.
- Public Health Planning: Aggregating and analyzing diabetes data on a larger scale can contribute to public health planning. Identifying trends, risk factors, and demographic patterns can inform public health policies, educational campaigns, and resource allocation at a broader level.
- Research and Understanding: Machine learning models applied to diabetes data can contribute to a deeper understanding of the underlying factors influencing the disease. This insight can guide further research, leading to the development of new treatments, medications, or preventive strategies.
- Continuous Monitoring and Feedback: In the context of wearable devices and remote patient monitoring, machine learning models can provide continuous feedback on a patient's health status. This real-time monitoring allows for timely adjustments to treatment plans and facilitates better disease management.
- Reducing Healthcare Costs: By preventing complications, optimizing treatment plans, and promoting proactive management, machine learning applications in diabetes care have the potential to reduce overall healthcare costs. This is particularly important given the economic burden associated with diabetes and its complications.

In summary, leveraging machine learning for diabetes data is important because it can lead to earlier intervention, personalized care, more efficient resource allocation, and a deeper understanding of the disease. These advancements contribute to improved patient outcomes, better public health strategies, and the overall well-being of individuals affected by diabetes.

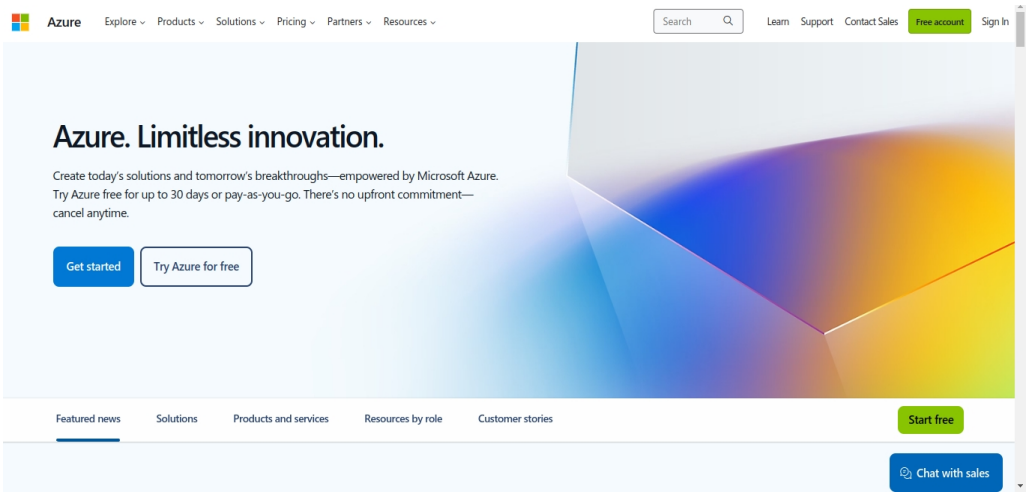
Why Azure ?

We are using Azure Cloud because our system has limitation of Storage and RAM, and we want to make the model which requires huge memory and RAM to build that model and second reason is that Azure has better seamless Integration with other tools and Hybrid Cloud Solutions. To make this project first, we make an account on Azure Cloud, which provides basic services for free for one year and advance services for free for 30 days.

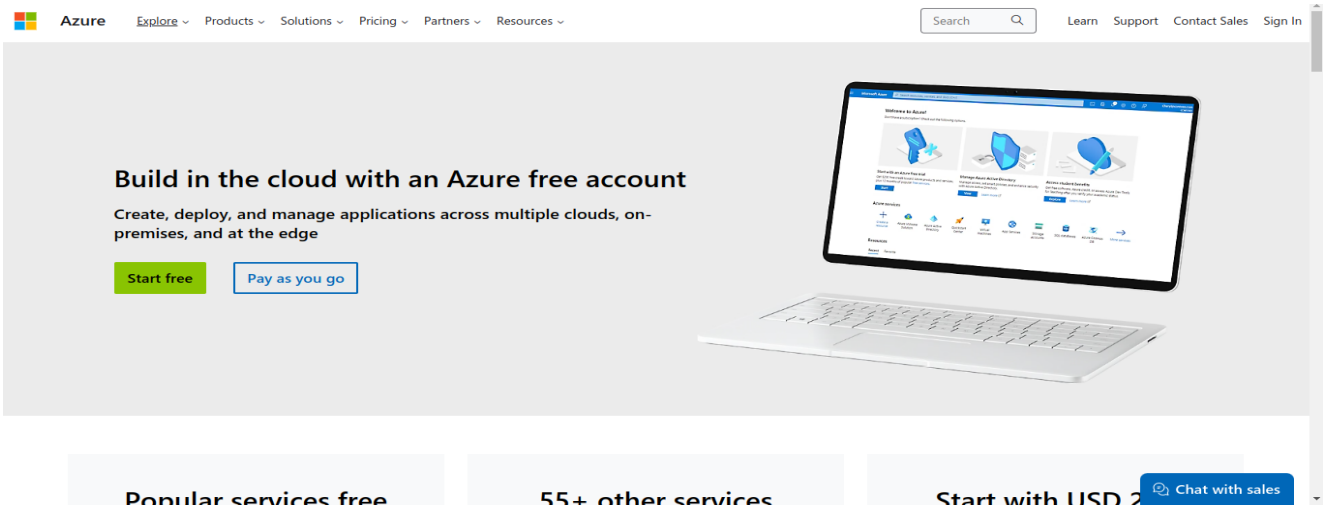
So, there are so many tools which we can use in MLOPS but in this project we are trying to build it using a single tool, i.e., Azure Cloud and Databricks.

To create the Account on Microsoft Azure, following are the steps given below: -

(1) Go to any web browser and type Microsoft Azure. Click on the first link. The window will appear whose screenshot is given below.

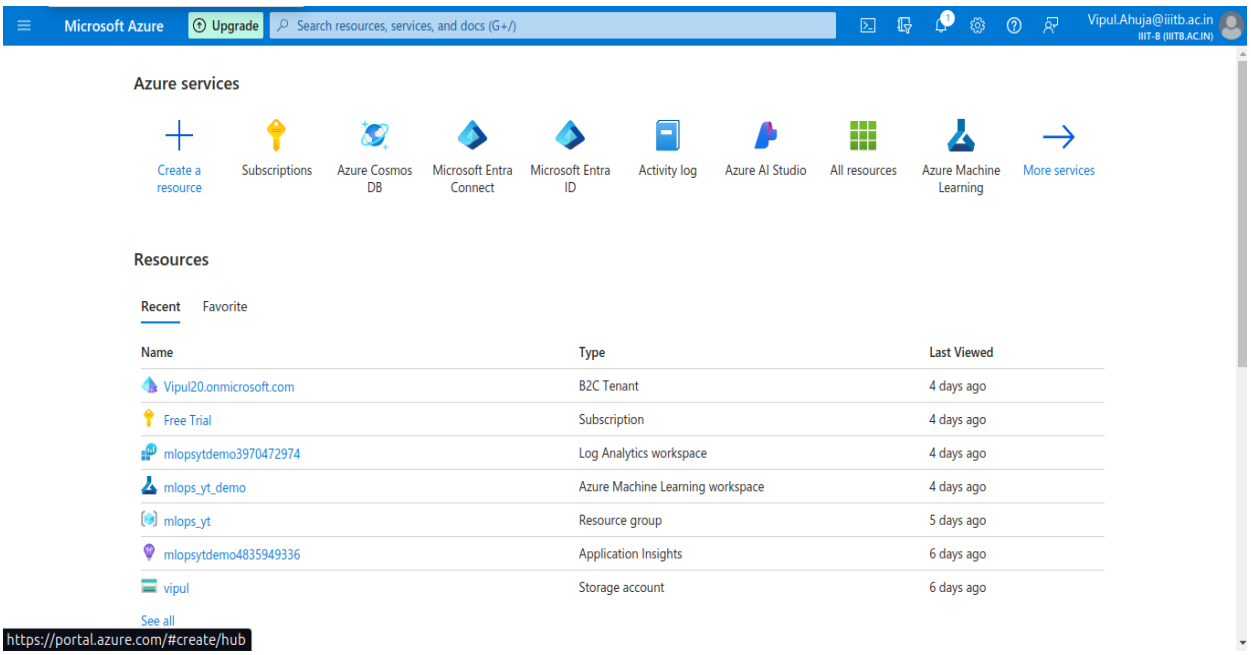


(2) Click on the green box Free Account on top Right Corner or Start Free on Bottom Right Corner. A new window will appear as follows: -



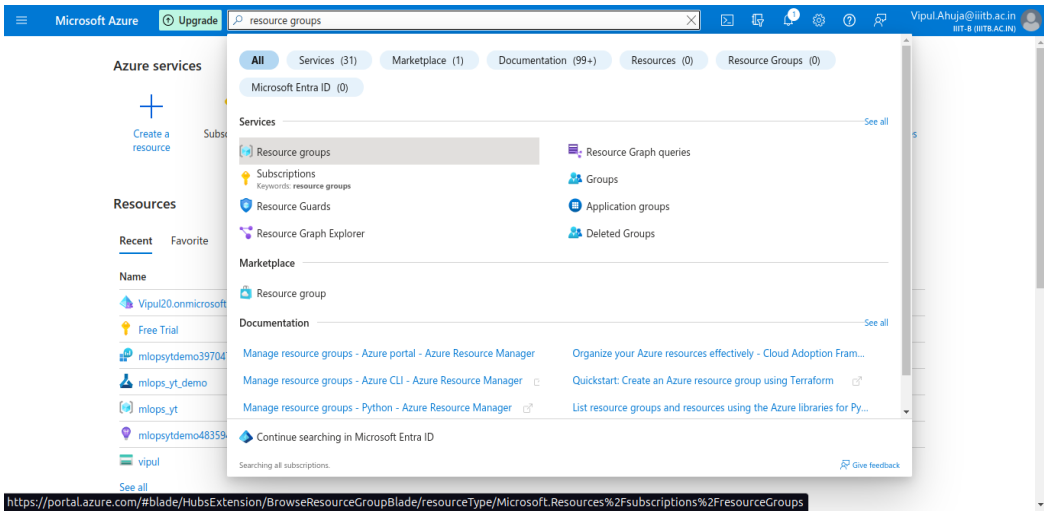
(3) Click on the Start Free Account. A New page will open which will ask for Sign Up. Put your Microsoft email id and get access to the free trial.

(4) Go to Microsoft Azure Portal. Your account will look like this in the below image.

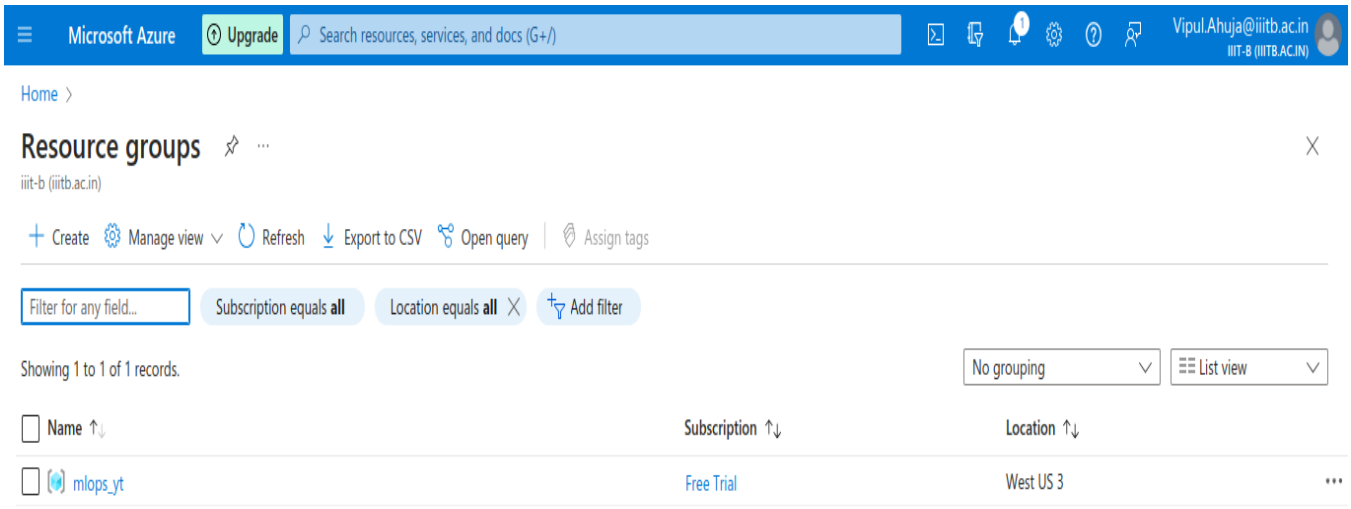


Now following are the steps for creating the Project: -

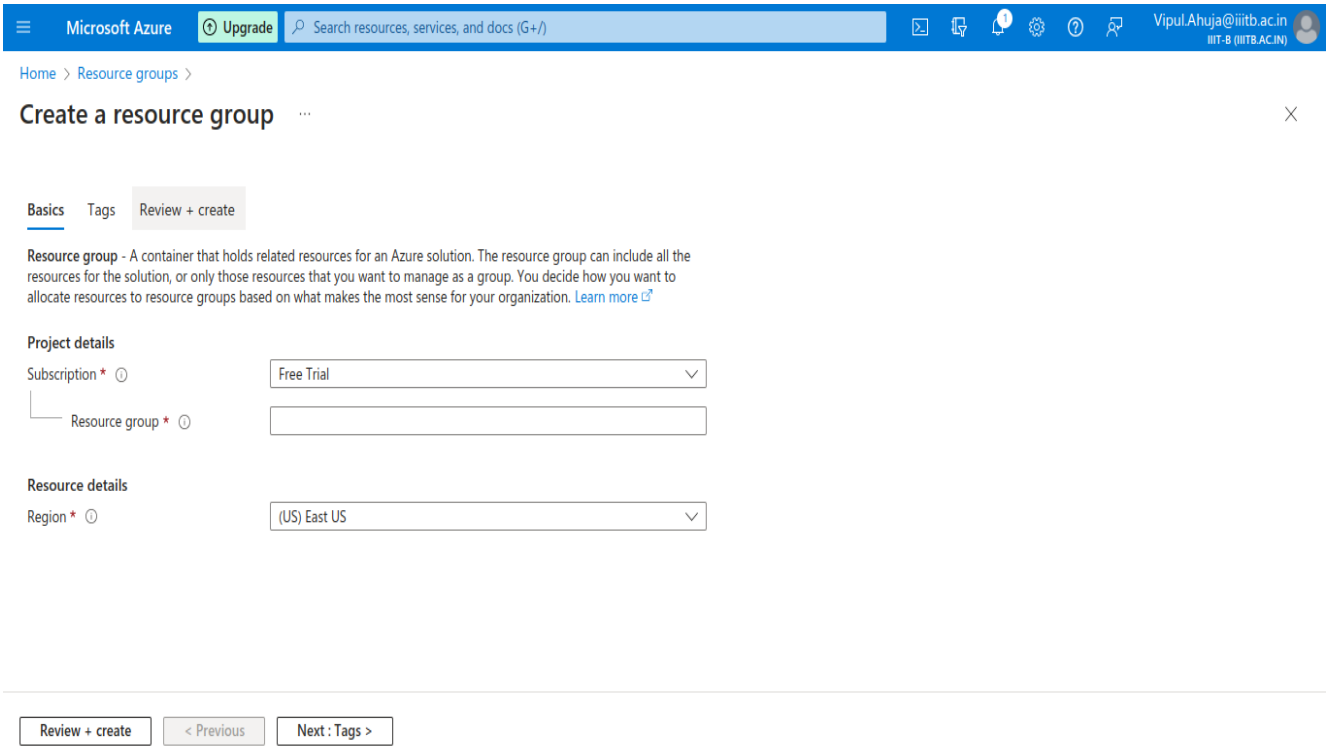
(1) Now search the resource group in the search bar at the top in the middle.



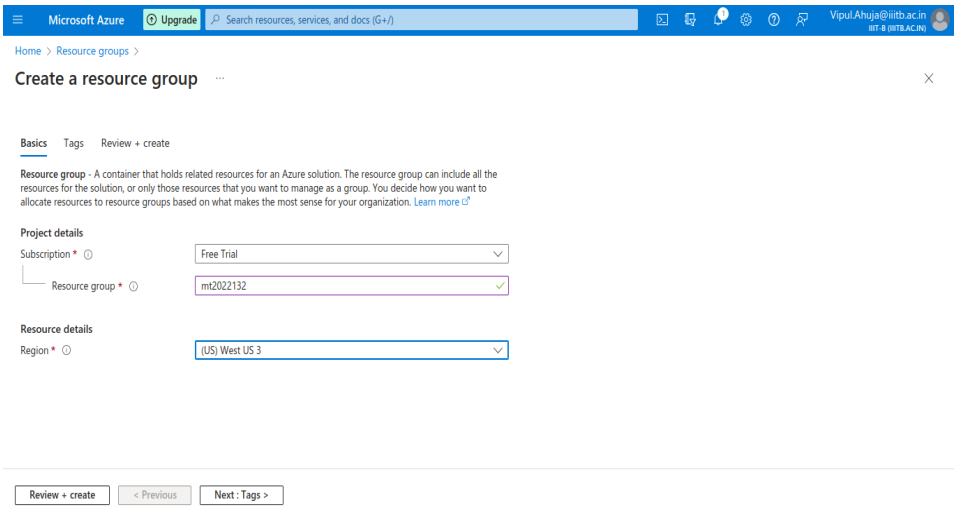
(2) Click on the resource group. Then New page will appear. On that click on Create.



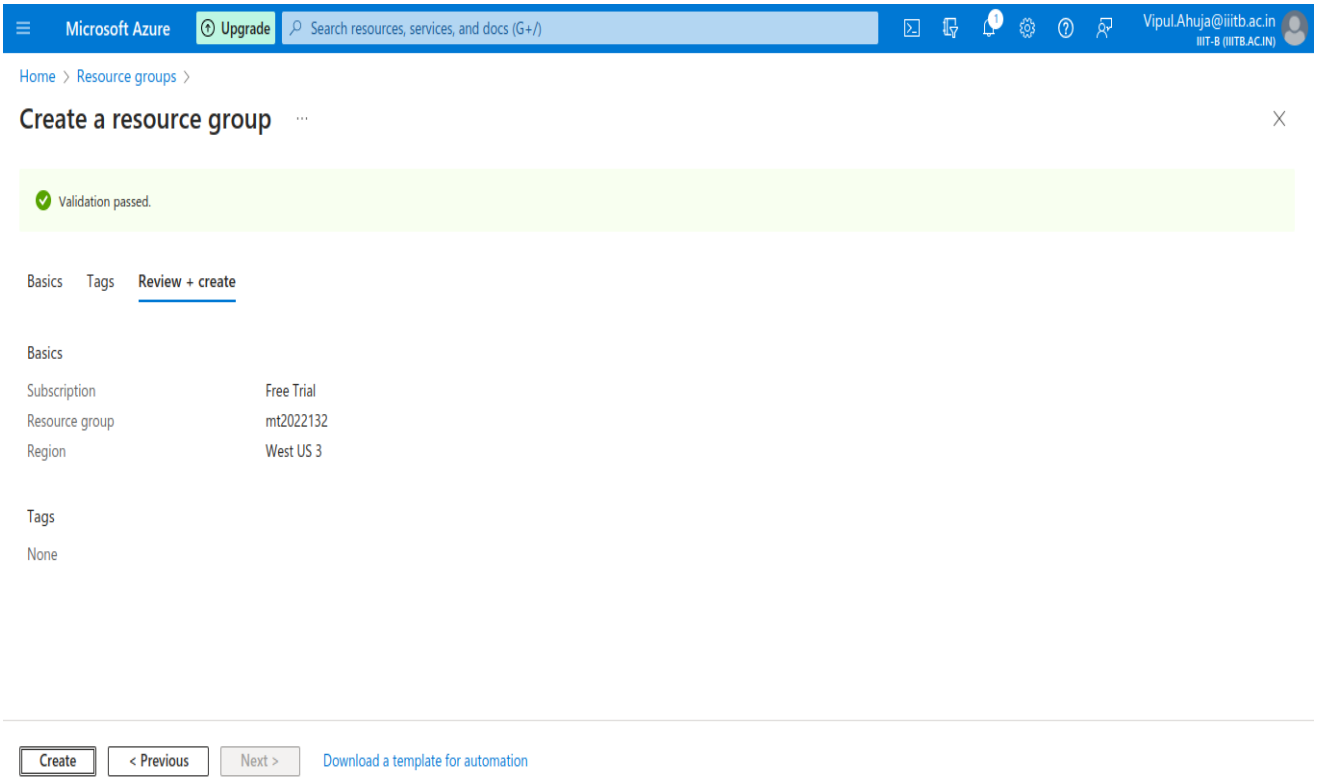
(3) After that, a new Window will appear.



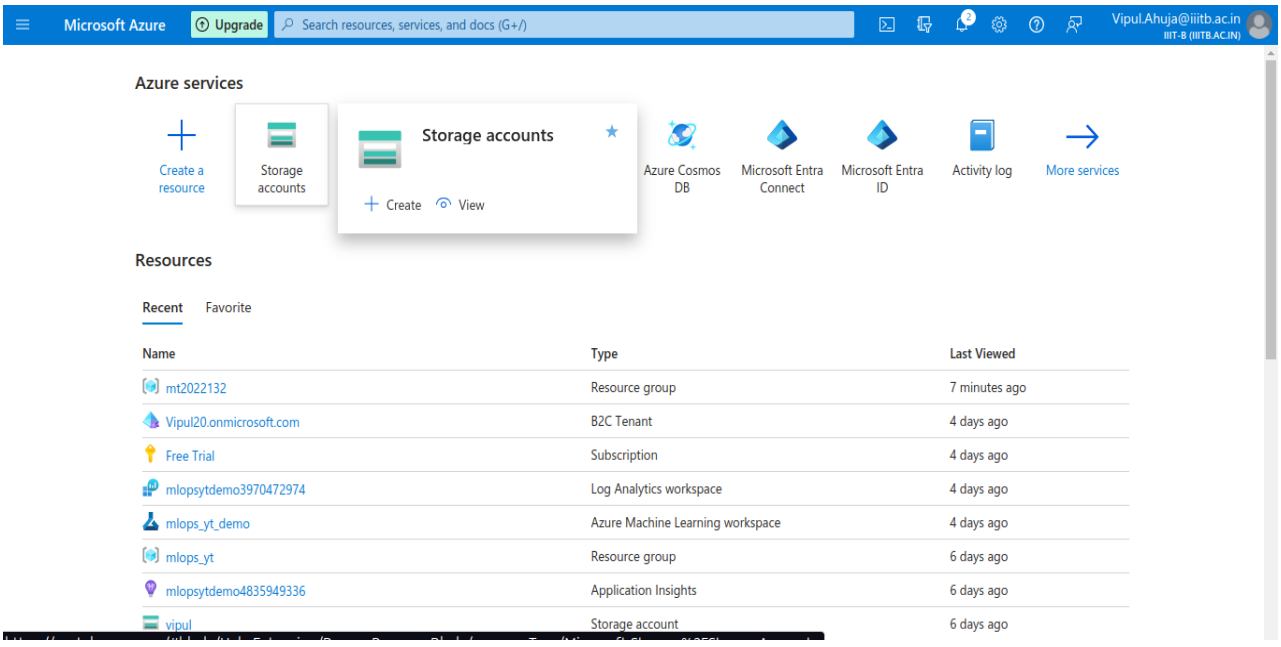
(4) In that set the Subscription Field as Free Trial. (By default, it is free trial). Free trial consists of $200 worth of resources. Once it is used, your free trail expires, so whatever you use you should delete that resource so that you can maximize your usage. In resource Group, give the valid name of your choice. Select any region of your choice. I am selecting West US 3. Then Click on Review + create.



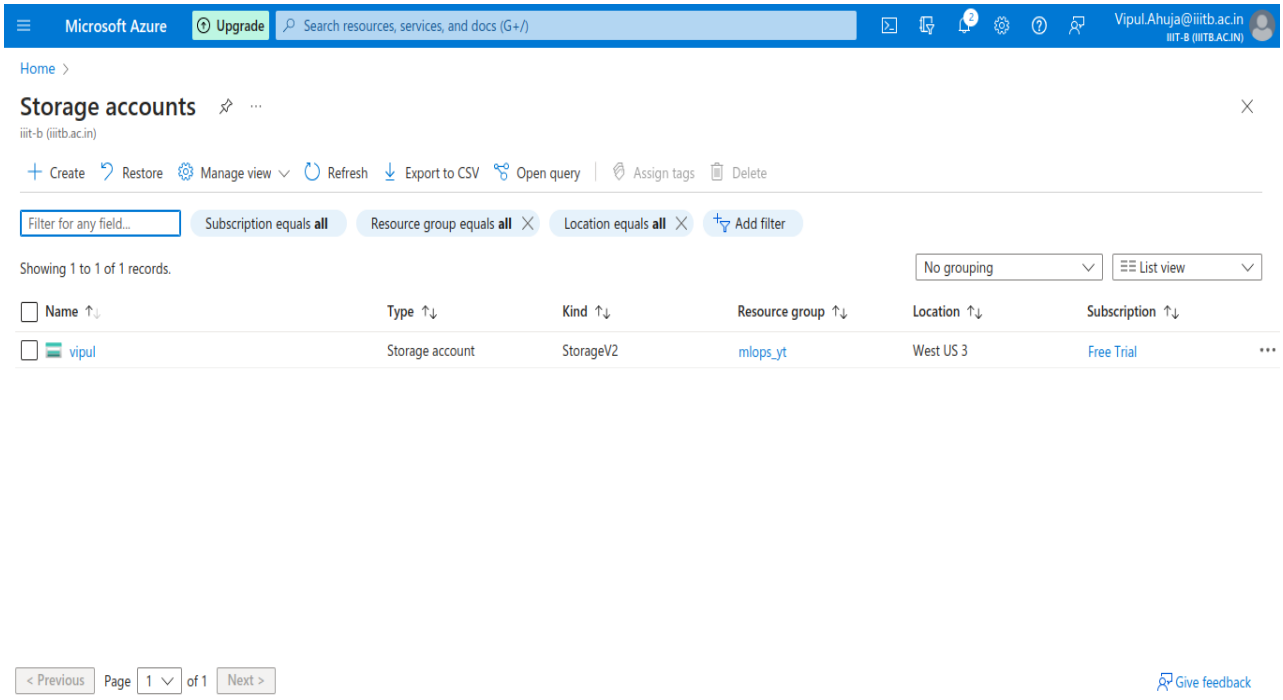
(5) Now, Check if the validation pass comes or not. If it comes click on create, otherwise try again the above procedure.



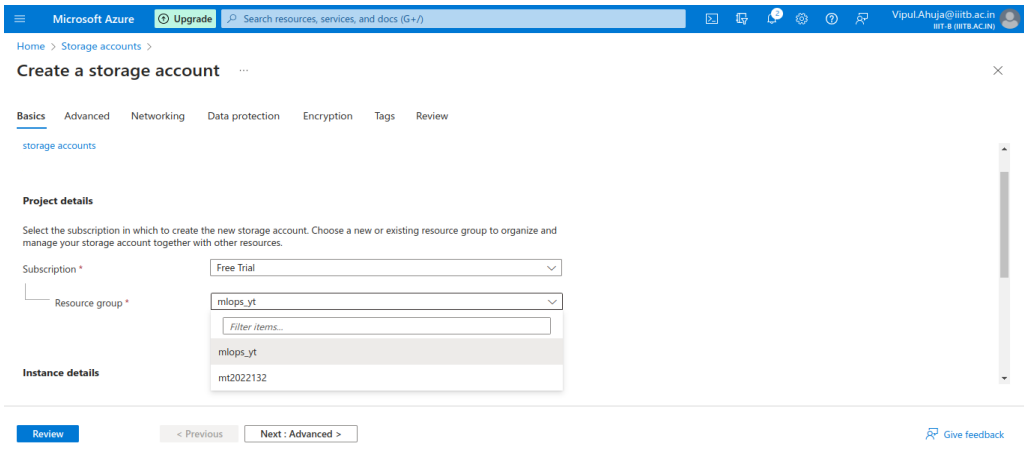
(6) Goto Home. Click on Storage Account in the resource section.



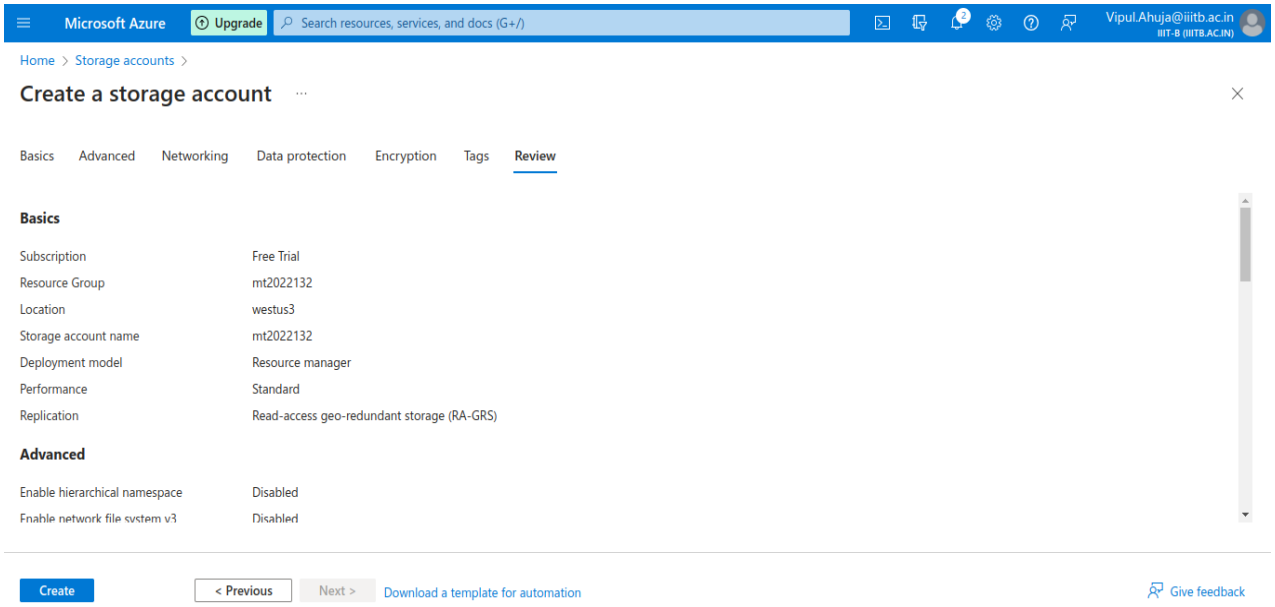
(7) A new window will appear. Click on Create.



(8) In the next step, set the subscription field as Free Trial. Set the Resource Group as the field name which you created in resource group in above steps. Here, I set it as “mt2022132”. So, I will select that. Set the valid storage name on your own. I am setting that as “mt2022132”. Click on next then again next till last next as rest field values are taken by default values.

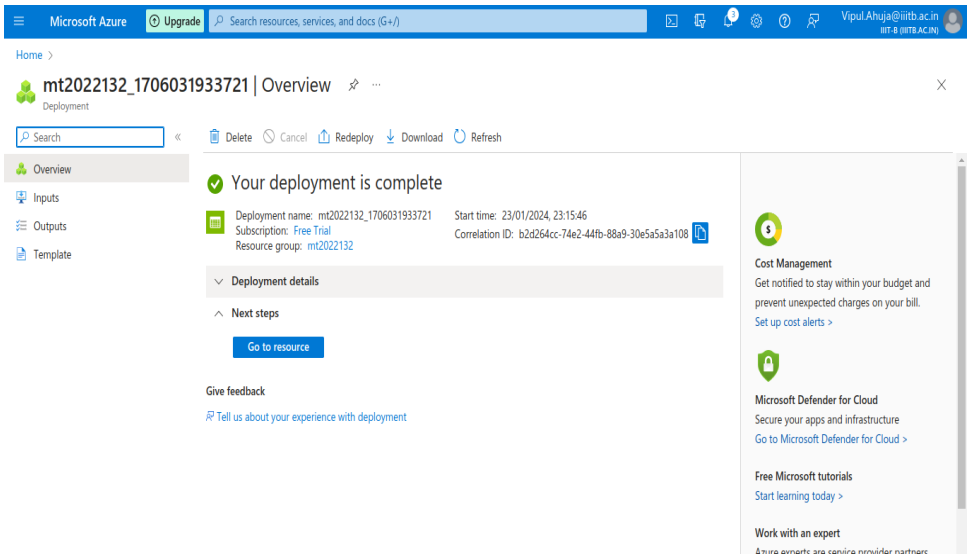


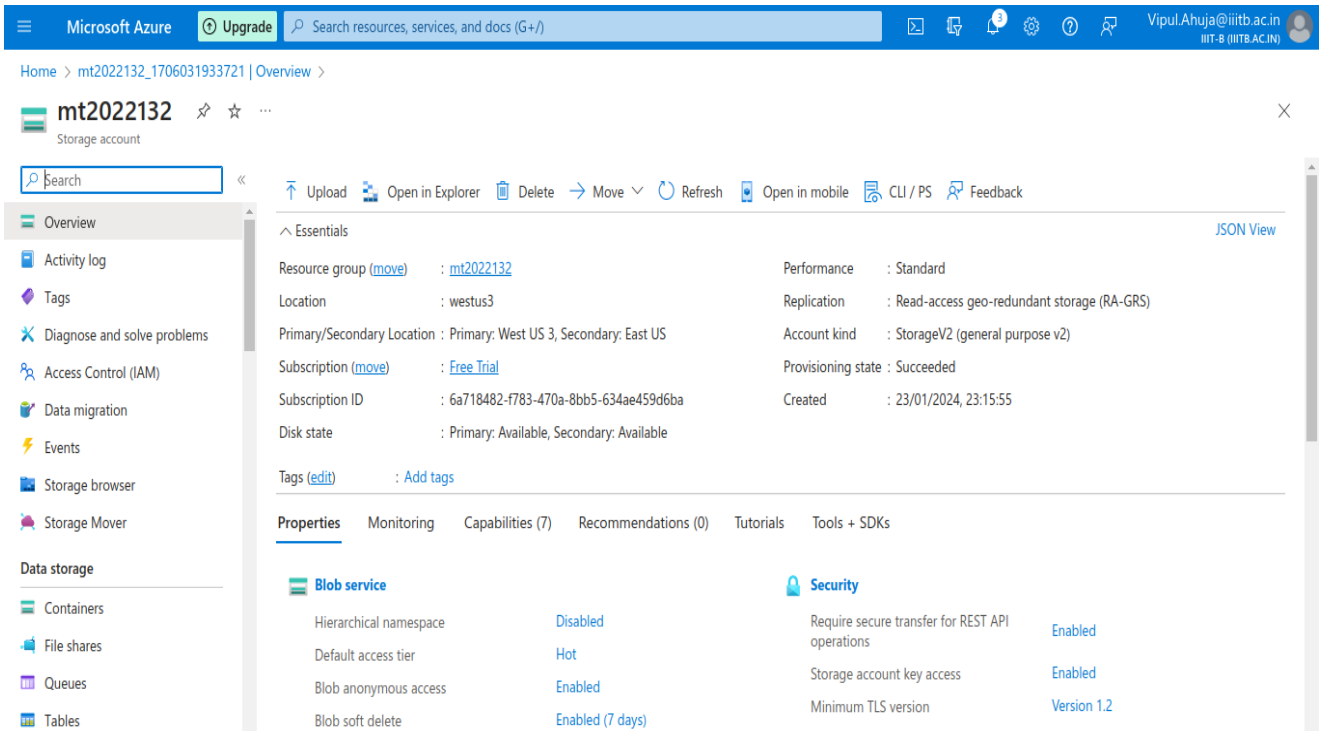
(9) After setting default values of rest of the fields, you will see the image as follows



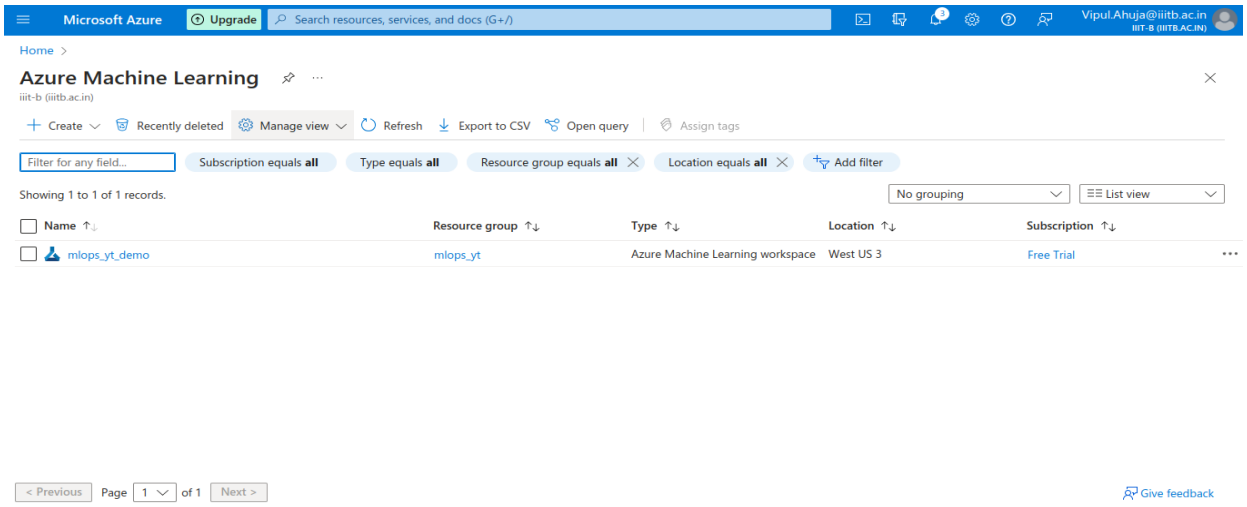
Click on Create now and wait for a few minutes as it takes some 3-5 minutes to create.

(10) As storage account has been created, click on Goto Resource button. You will see the details.

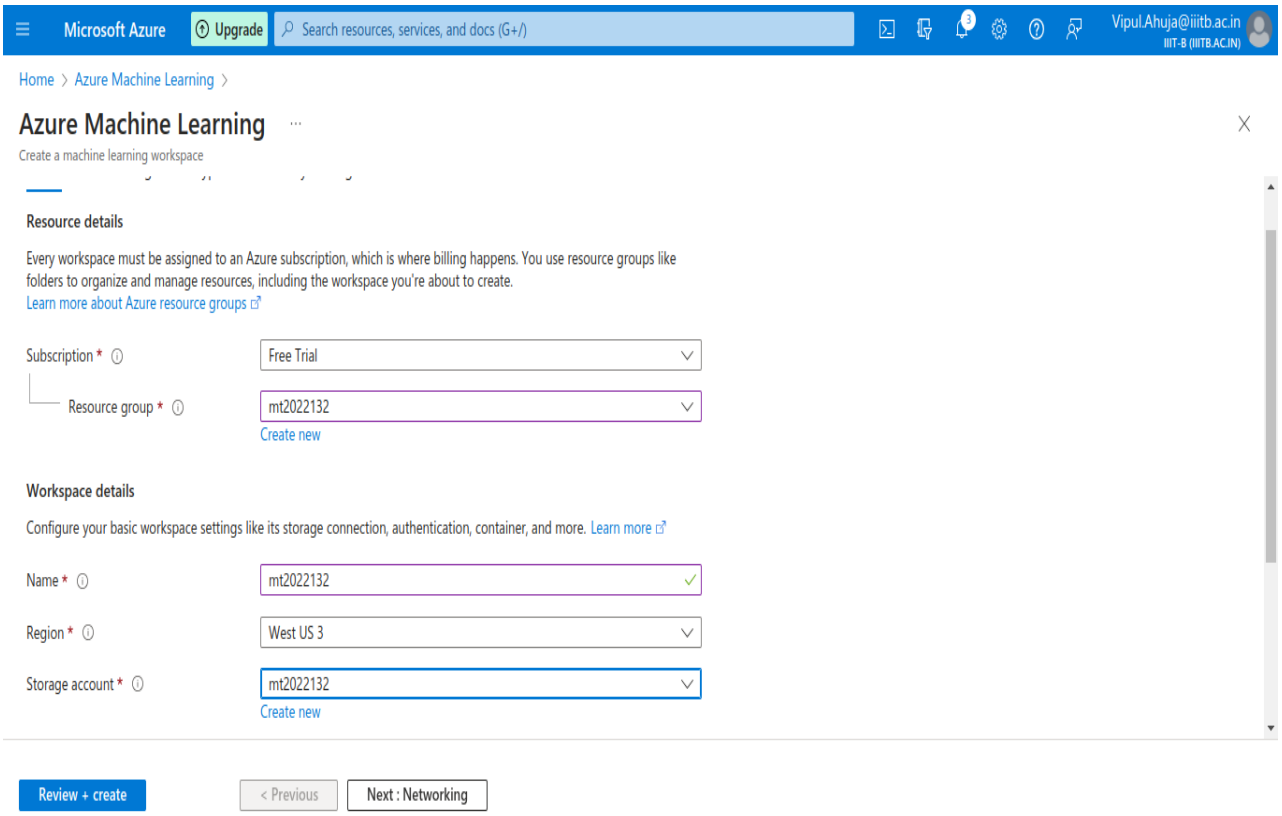




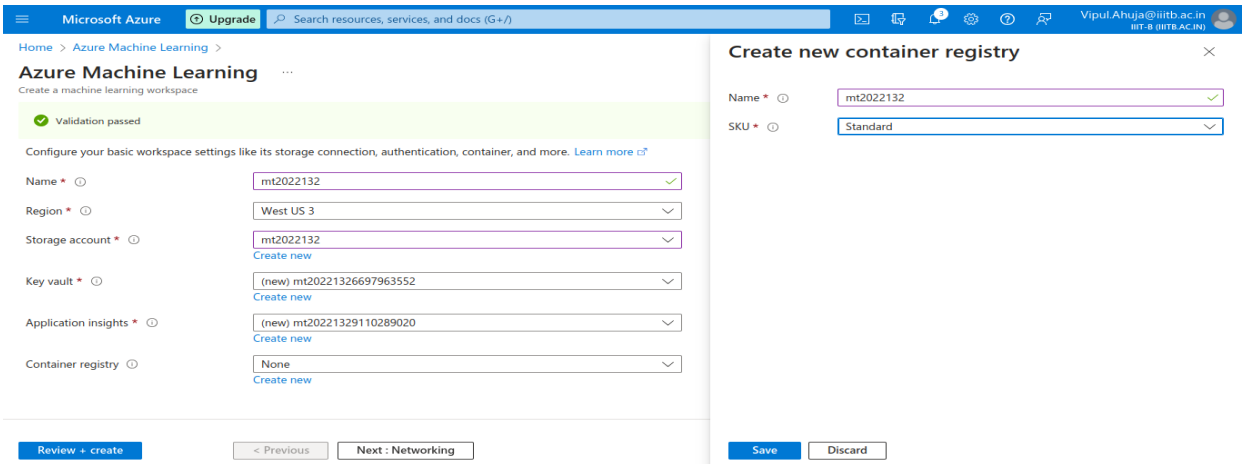
(11) Now, go to the Home page. Search “Azure Machine Learning” in the search bar. Click on it. Then Click on Create button. Two options will appear. New Workspace and New Registry. Select 1st one that is New Workspace.



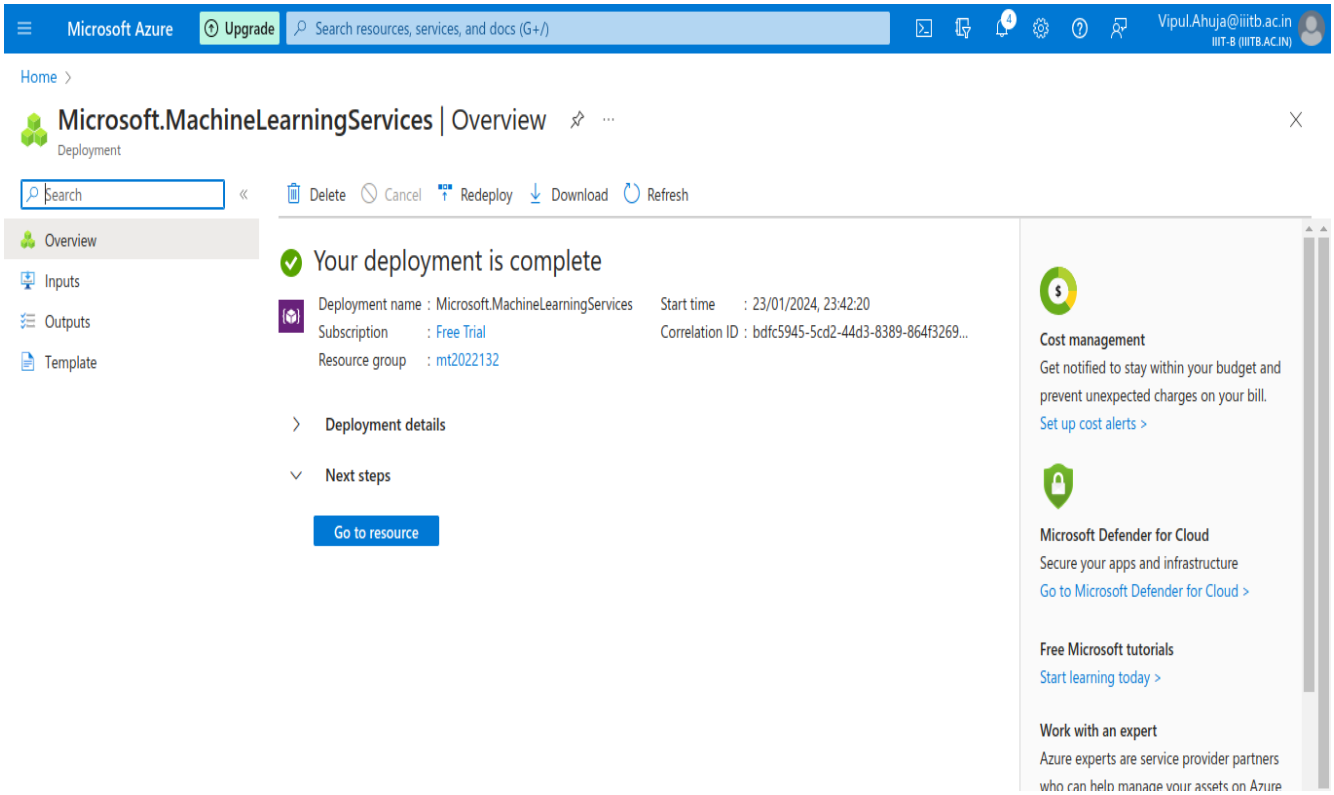
(12) Set the field as follows in the image.



(13) The container registry is by default None. Click on Create below Content Registry dialog box to create new registry. Set the name as “mt2022132”. Set SKU as Standard. Click Save. Then click on Next: Networking.

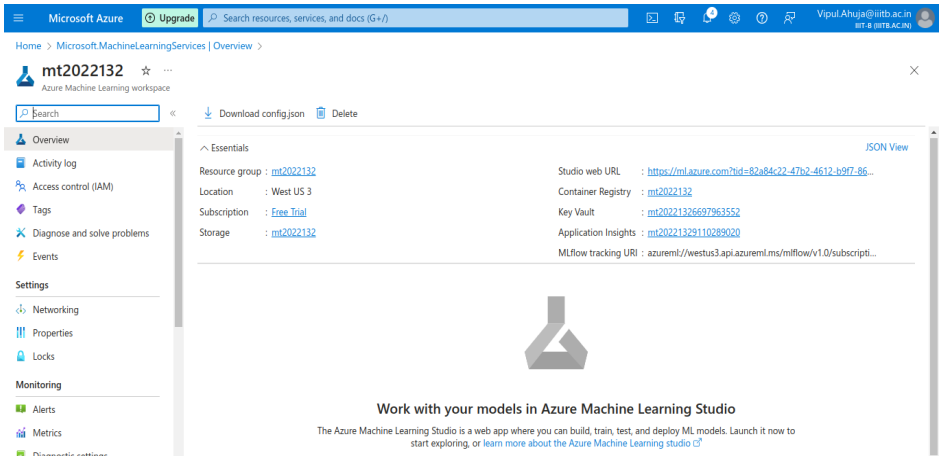


(14) Click on Next and Next until last Next as rest field are set to be default value. In the end Validation passed will come. Then Click on create. Wait for a few minutes.

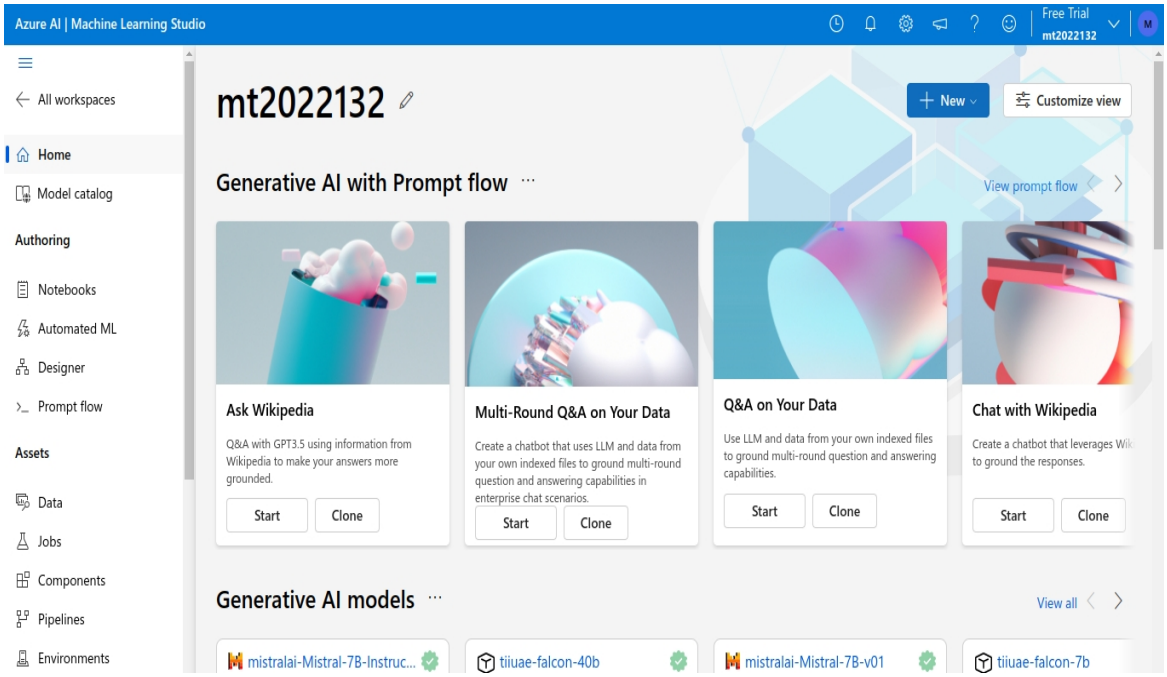


(15) Click on the Go to resource button.

(16) A new window will appear. Click on the Studio Web URL link.

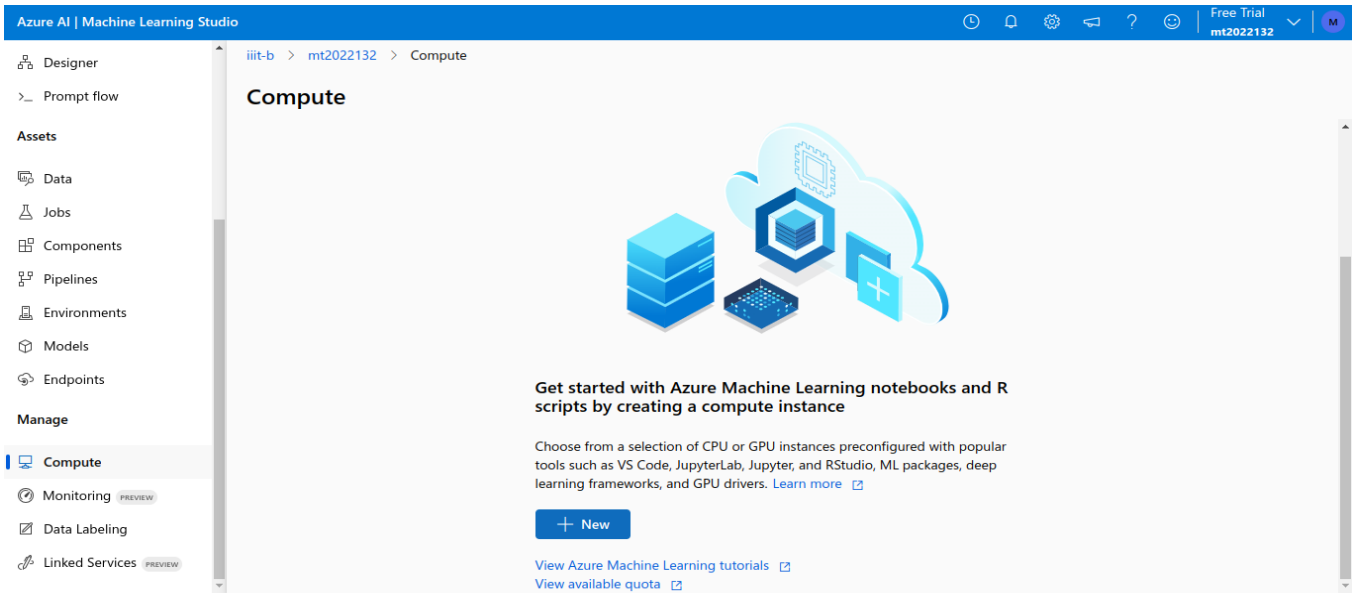


(17) It will Launch Azure AI/Machine Learning Studio in New Window. You can see that in the image below.

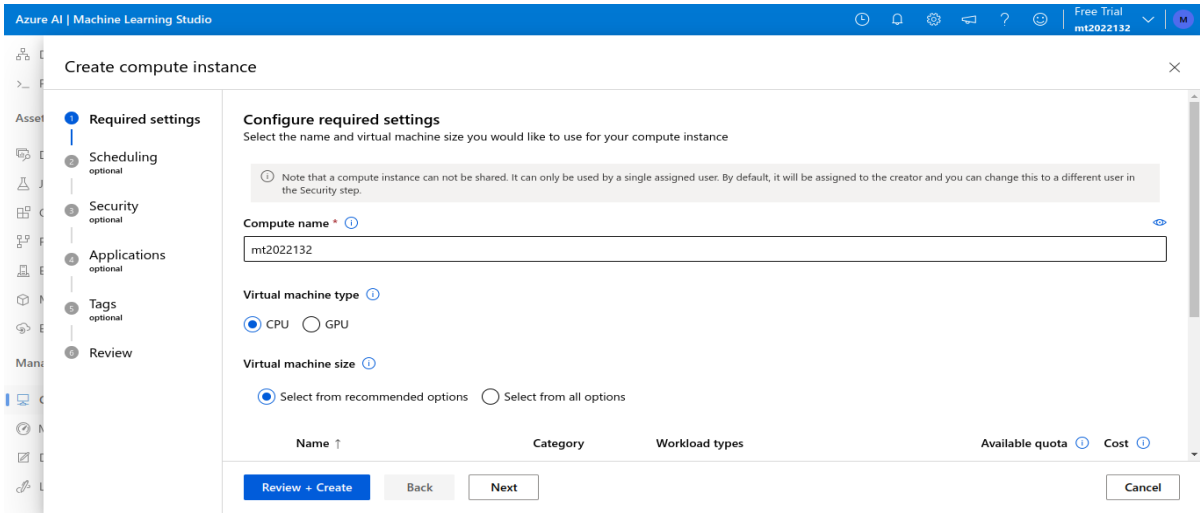


(18) In Studio, you will see so many options on the left panel which are divided into sections which can be seen on scrolling down. Using these options, we can integrate other MLOPS tools, or can create automated scripts, etc. But we are using only Azure Cloud for MLOPS so it has used very few options from that section because we are not using any different MLOPS tools (outside tools like Kubeflow, etc.)

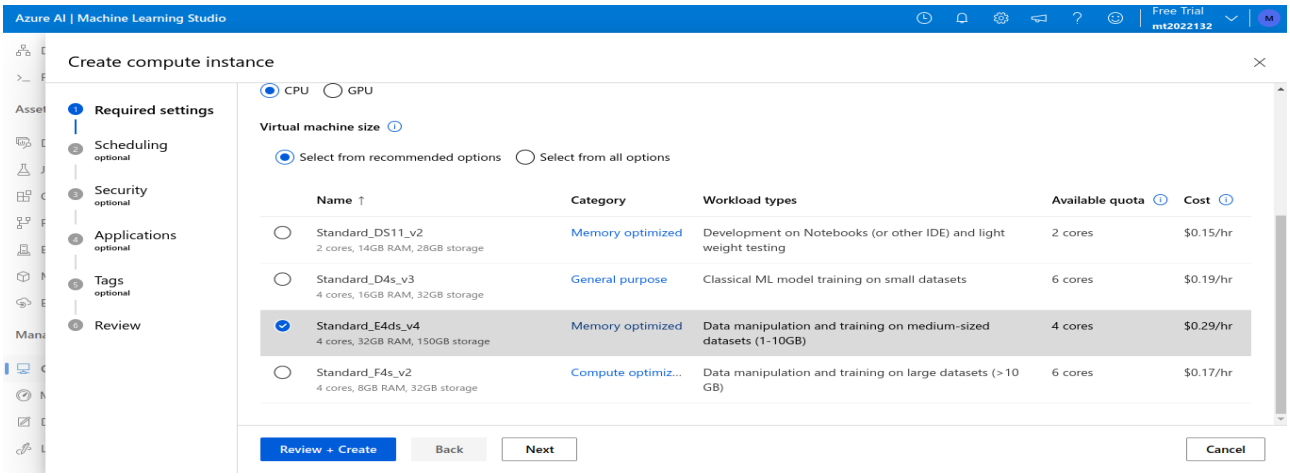
(19) Now we will create an instance under Compute Section to assign some memory to our project. Go to Compute Section. Click on New.



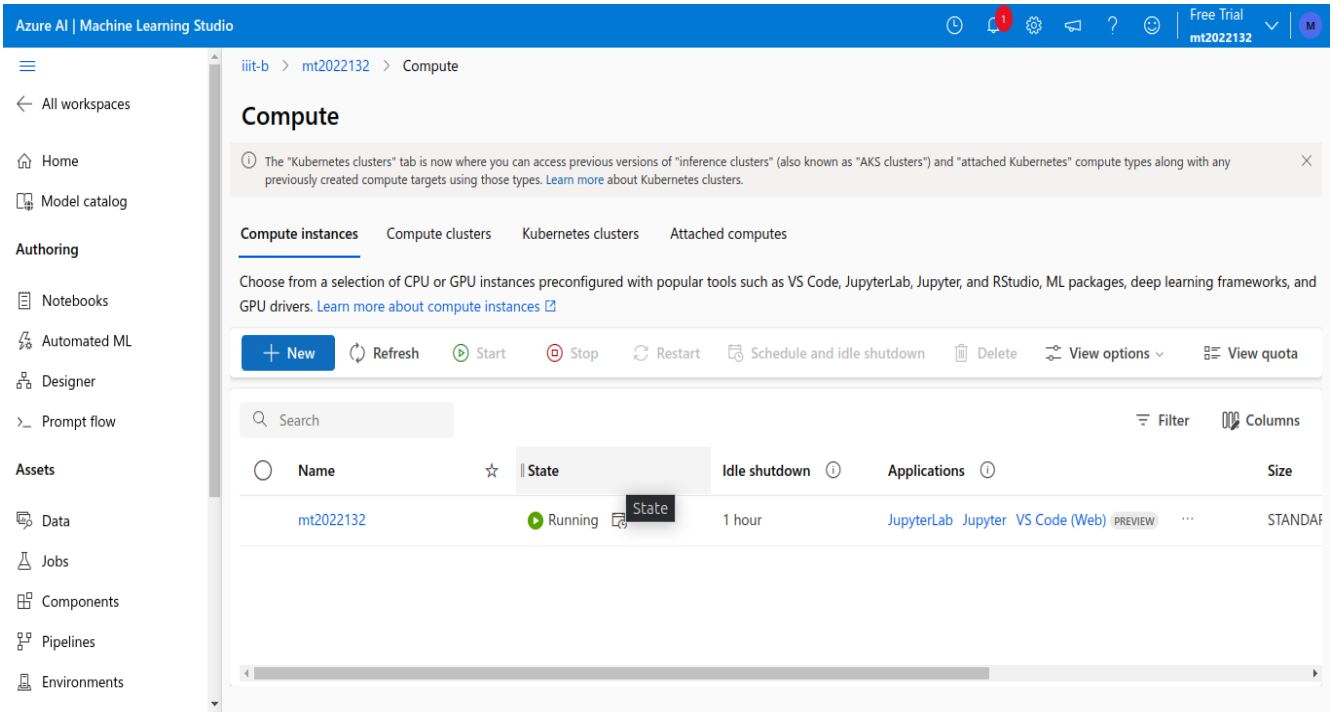
(20) Give the name of Compute. I am giving it as ”mt2022132”.



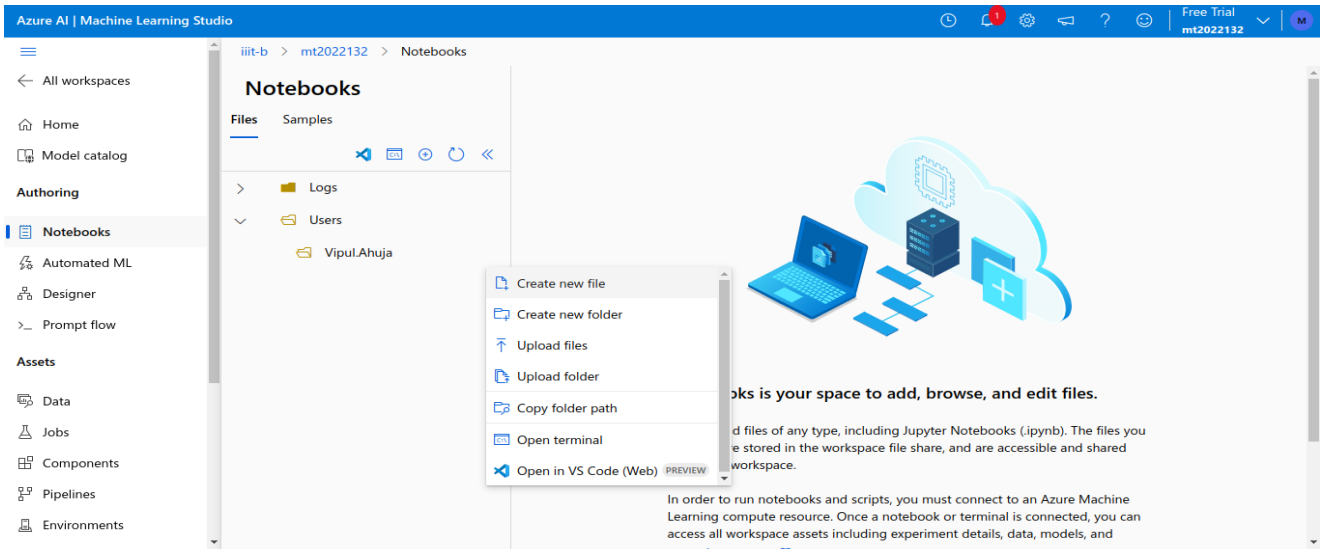
(21) Select the standard virtual machine size and delete another machine if you have created it, as it will cost you and you are not using it then you must delete that as creating 2 virtual machines instead of 1 will cost you double.



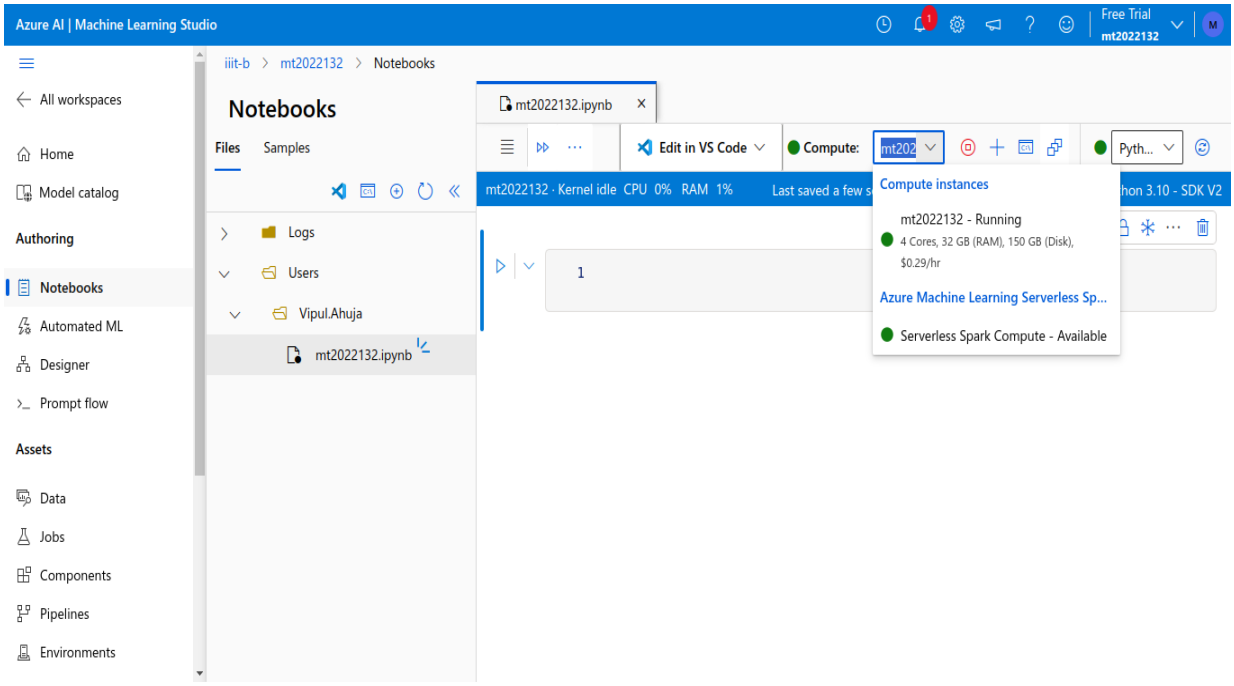
(22) Click on Next and set all the field values to be default. Then Click on Create. Wait for a few minutes. You can see the image below for the same.



(23) Now go to Notebook Section and Click on create new file under your Username folder as shown in the image. Then Click “Create new file” option. Create ipynb file.



(24) Write the valid name of the file on your own. I have set the name as “mt2022132”. Now, Select the compute instance that you have created. And make sure it is running and if it is not then start that.



(25) Now, write the code in ipynb file that you have created.

Explanations :-

MLOPS Using Azure ML Studio

Now we will use our diabetes data for Azure ML

Steps:-

- First we will import our data from sklearn

```
import joblib
import sklearn

from sklearn.datasets import load_diabetes
from sklearn.linear_model import BayesianRidge, Ridge
```

✓

```
x, y = load_diabetes(return_X_y=True)
```

✓

```
first_model = Ridge().fit(x, y)
joblib.dump(first_model, "model_ridge.pkl")
```

✓

```
['model_ridge.pkl']
```

```
from azureml.core.model import Model
model = Model.register(model_path="model_ridge.pkl",
                       model_name="model_ridge",
                       workspace=ws)
```

✓

- The score2.py script is designed for scoring with a Ridge regression model in Azure Machine Learning. The init() function initializes the model, loading it from the Azure Model Registry. The run() function handles inference by

parsing input JSON data, making predictions using the Ridge model, and returning results in JSON format. The script is intended for use in Azure ML scoring environments.

```
%%writefile score2.py
import joblib
import json
import numpy as np

from azureml.core.model import Model

def init():
    global model_3
    model_3_path = Model.get_model_path(model_name='model_ridge')
    model_3 = joblib.load(model_3_path)

def run(raw_data):
    try:
        data = json.loads(raw_data)['data']
        data = np.array(data)
        result_1 = model_3.predict(data)

        return {"prediction1": result_1.tolist()}
    except Exception as e:
        result = str(e)
        return result
```

- This code deploys a machine learning model as an Azure Container Instances (ACI) web service using Azure Machine Learning. It configures a Python environment, specifies an inference script (score2.py), and deploys the model to ACI with defined CPU and memory resources. The code waits for the deployment to complete and prints the service state, providing an end-to-end demonstration of model deployment in Azure ML.

```
from azureml.core.webservice import AciWebservice

aci_service_name = "aciservice-modelridge"

deployment_config = AciWebservice.deploy_configuration(cpu_cores=1, memory_gb=1)

service = Model.deploy(ws, aci_service_name, [model], inference_config, deployment_config, overwrite=True)
service.wait_for_deployment(True)

print(service.state)
```

- At last we will get our results

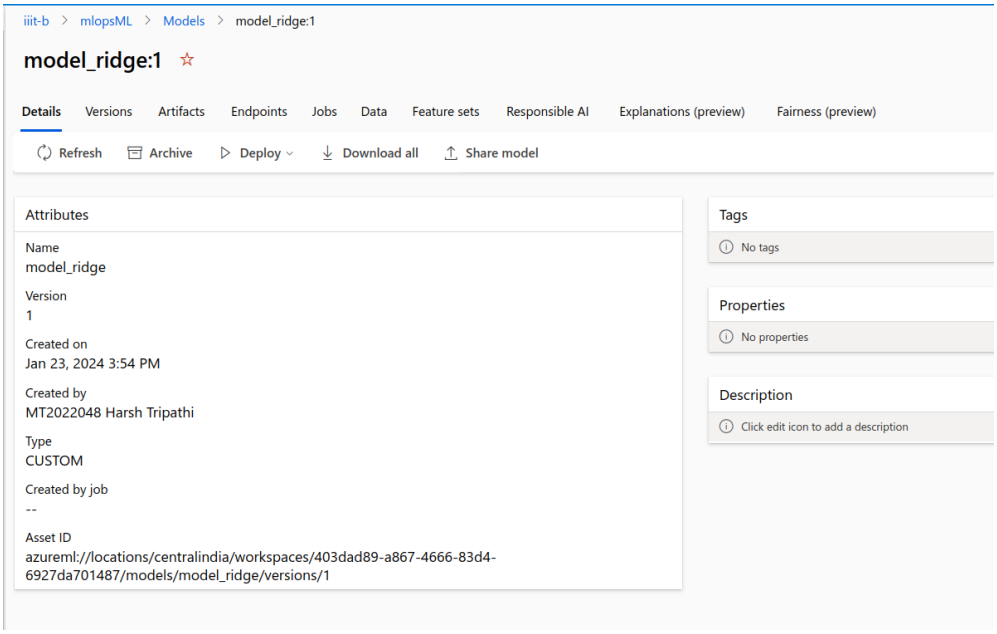
```
import json
test_sample = json.dumps({'data': x[0:2].tolist()})
predictions = service.run(test_sample)
predictions
```

```
{'prediction1': [182.67335420683418, 90.99860655841785]}
```

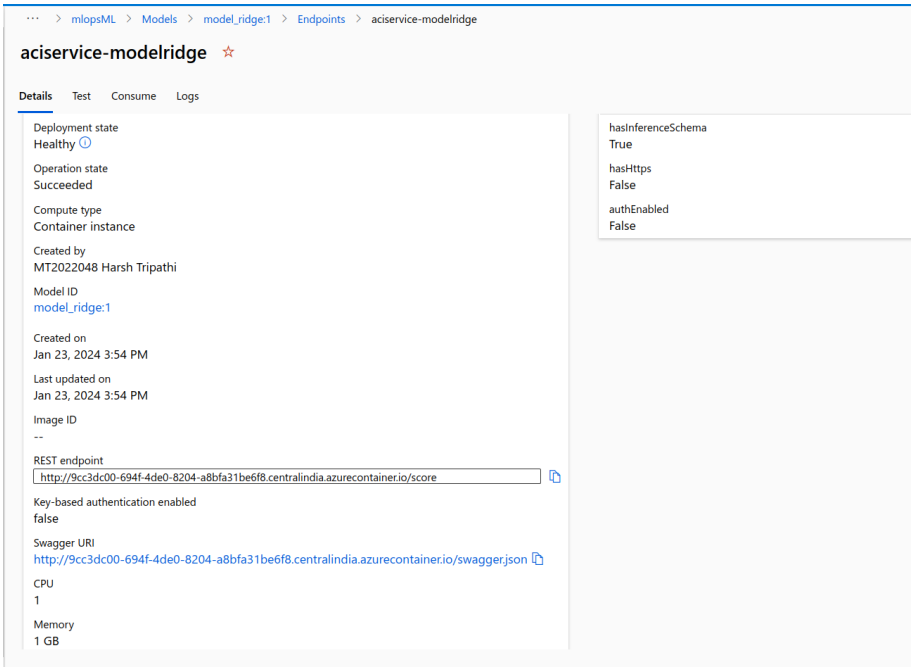
Now we will use of Model tab in Azure ML studio which helps in following ways

- Model Registration:
  - Register trained models in the "Models" section.
  - Create model references, store metadata, and enable versioning.
- Model Deployment:
  - Deploy registered models as web services.
  - Deployment options include Azure Kubernetes Service (AKS) and Azure Container Instances (ACI).
- Endpoint Management:
  - Manage model endpoints for monitoring, scaling, and updates.
  - Ensure deployed models meet performance requirements.
- Model Versioning:
  - Utilize versioning to track different model iterations.
  - Facilitates maintaining a history of model changes and enables rollback.
- Experimentation and Run History:
  - Access experiment tracking and run history features.
  - Monitor performance and results of various model training runs.
- Pipeline Integration:

    - Integrate model deployment into end-to-end ML pipelines.
    - Automate the entire ML lifecycle, from data preparation to deployment.
- Comprehensive Platform:
    - Azure Machine Learning Studio serves as a central platform for model management.
    - Offers a range of features for registration, deployment, versioning, and integration.



We can also use endpoints to deploy the model.



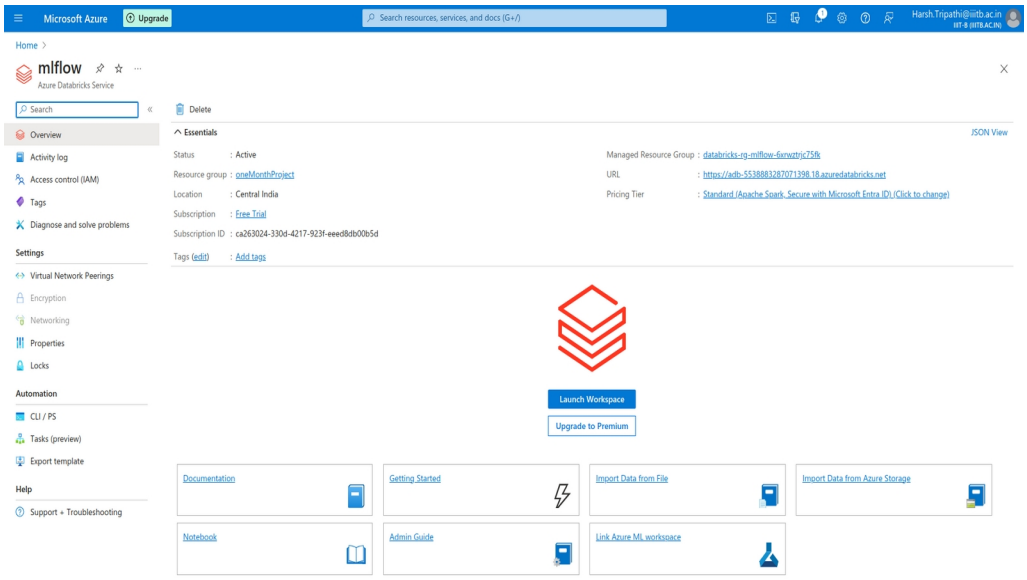
So by this way we can deploy our model to real world and feed live data.

We Will also implement the project in Databricks

Databricks is a cloud-based platform designed for big data analytics and machine learning. It was founded by the creators of Apache Spark, a widely used open-source distributed computing system. Databricks provides an integrated environment that facilitates data engineering, data science, and collaborative data analytics.

Steps for creating databricks workspace in azure:-

- In the Azure Portal, click on "Create a resource."
- Search for "Databricks" in the Azure Marketplace and select "Azure Databricks."
- Click "Create" to start the creation process.
- Fill in the required information, such as subscription, resource group, workspace name, region, and pricing tier.
- Configure additional settings and click "Review + create."
- Review your configuration and click "Create" to deploy the Databricks workspace.



At last, launch the workspace to open Databricks. We need to create a new notebook for writing our codes, for using mlops we will MLFlow.

MLflow is an open-source platform designed to manage the end-to-end machine learning lifecycle. Developed by Databricks, MLflow provides tools for tracking experiments, packaging code into reproducible runs, and sharing and deploying models.

MLflow supports various machine learning libraries and frameworks, making it agnostic to the underlying technology. It has integrations with popular tools like scikit-learn, TensorFlow, and PyTorch. MLflow is cloud-agnostic and can be used both on-premises and in cloud environments.

Overall, MLflow enhances collaboration and reproducibility in machine learning projects, addressing challenges across the entire lifecycle, from experimentation and development to deployment and monitoring. It has gained popularity in the machine learning community as a versatile and user-friendly tool for managing the complexities of machine learning workflows.

Steps:-

- Analysis of data

```
Y=np.array([y]).transpose()
d=np.concatenate((X,Y),axis=1)

columns = ['age','sex','bmi','bp','s1','s2','s3','s4','s5','s6','progression']
data=pd.DataFrame(d,columns=columns)
```

Command took 0.10 seconds -- by harsh.tripathi@iiitb.ac.in at 1/28/2024, 11:40:51 PM on MT2022048 Tripathi's Cluster

Cmd 8

```
data.head()
```

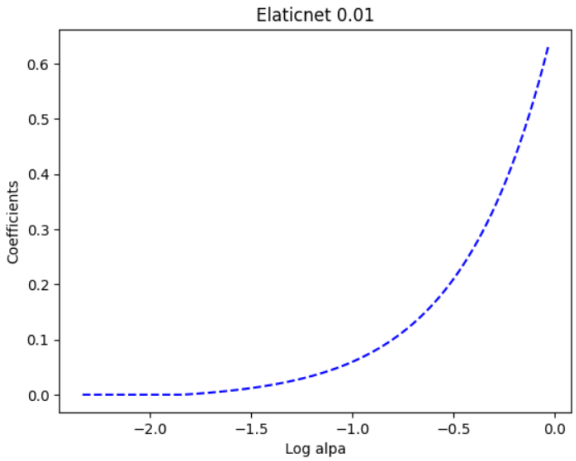
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	progression
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641	135.0

Command took 0.10 seconds -- by harsh.tripathi@iiitb.ac.in at 1/28/2024, 11:40:51 PM on MT2022048 Tripathi's Cluster

- The plot_elastic function generates and saves a plot illustrating the regularization path of coefficients for an Elastic Net model. It uses the enet_path function to calculate the model coefficients for different regularization strengths. The resulting plot, showing how coefficients change with varying regularization, is saved as 'Elasticnet_path.png'.

```python
def plot_elastic(X,y,l1_ratio):
    eps=5e-3
    alpha_enet, coef_enet, _ =enet_path(X , y, eps=eps, l1_ratio=l1_ratio)
    global image

    fig= plt.figure()
    az=plt.gca()

    colors = cycle(['b','r','g','c','k'])
    neg_alpha_ents = -np.log10(alpha_enet)

    for coef_e , c in zip(coef_enet, colors):
        l1=plt.plot(neg_alpha_ents, coef_e, c=c, linestyle = '--')
        plt.xlabel('Log alpa')
        plt.ylabel('Coefficients')

        title="Elaticnet "+ str(l1_ratio)
        plt.title(title)
        plt.axis('tight')
        image=fig
        fig.savefig('Elasticnet_path.png')
        plt.close(fig)
        return image
```

- The train_diab_model function trains an Elastic Net regression model on diabetes data, evaluates its performance, logs the results using MLflow, and saves the trained model. It splits the data, handles hyperparameters, trains the model, logs hyperparameters and metrics, saves the model and its artifacts, and returns information about the MLflow run.



- Once our model is ready, we will print some info, The print_model_info function prints details about each model version. The subsequent code uses the MLflow tracking client to create a registered model and a new model version associated with that model in the MLflow model registry. If the model already exists, it catches the exception. A delay is added to allow time for the model version to be fully registered.
- The code retrieves information about the latest version of a registered model, prints the details (though not captured due to the function not returning anything), and transitions the staging stage of that model version to "staging" in the MLflow model registry.

```
1  model_info = print_model_info(client.get_latest_versions(name=model_name))

name diab
version 3
run id a39902d944624f56a72317791d8d0a66
current_stage Staging
name diab
version 4
run id 50036fb3ba794a8b9a095fe321be89e4
current_stage None

Command took 0.20 seconds -- by harsh.tripathi@iiitb.ac.in at 1/28/2024, 11:40:51 PM on MT2022048 Tripathi's Cluster
```

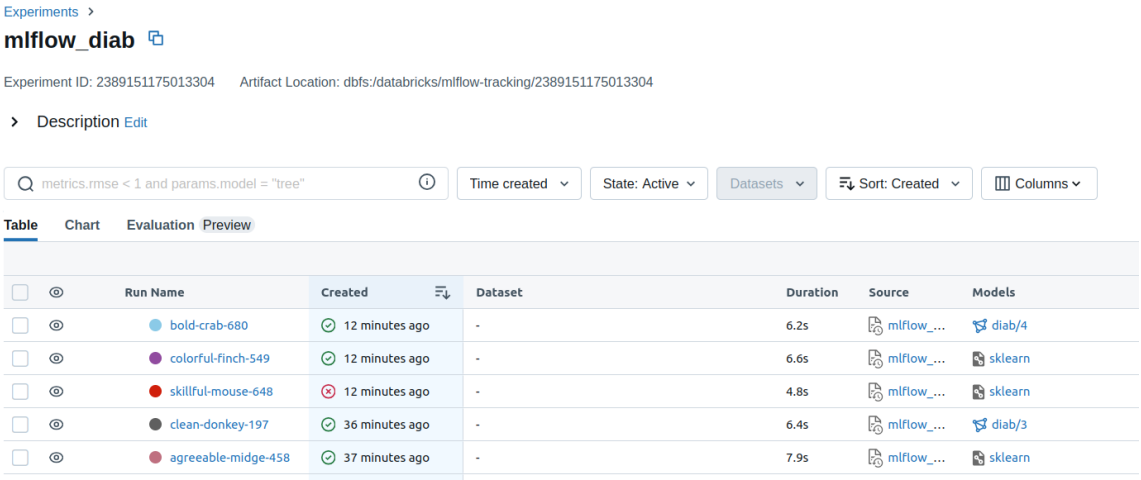
Cmd 19

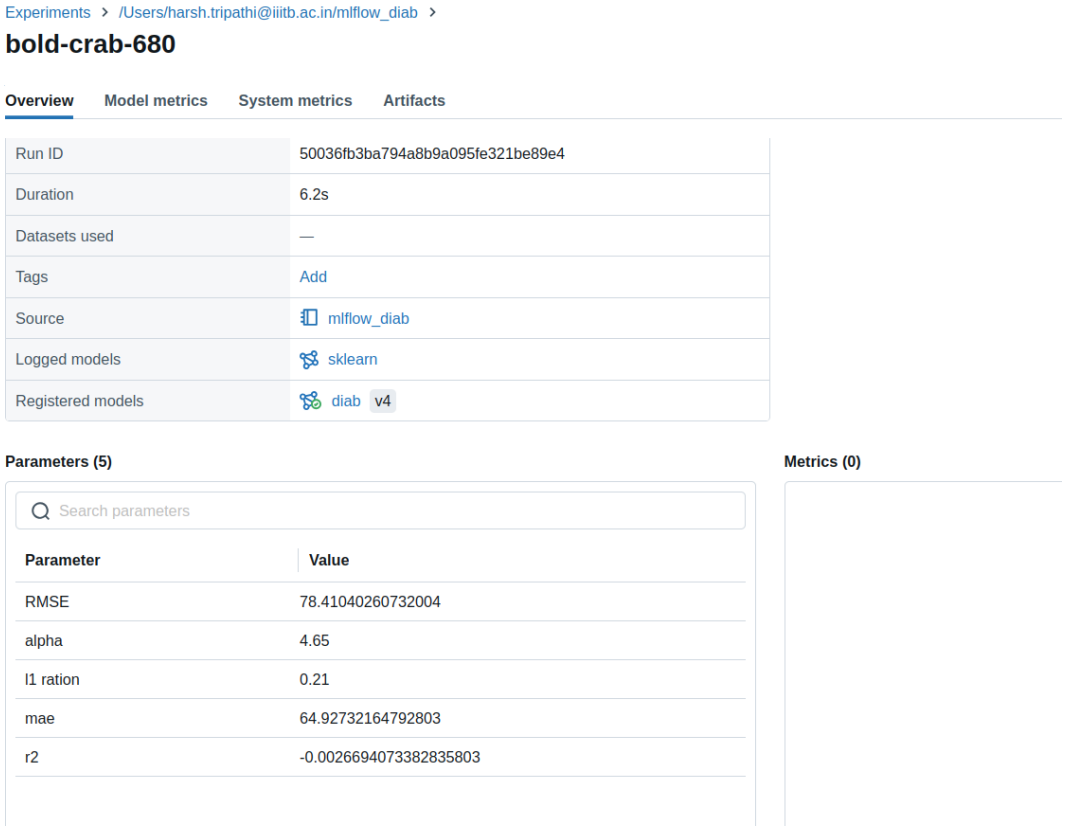
```
1  client.transition_model_version_stage(model_name, model_version.version, stage="staging")

<ModelVersion: aliases=[], creation_timestamp=1706465473907, current_stage='Staging', description='',
e321be89e4', run_link='', source='dbfs:/databricks/mlflow-tracking/2389151175013304/50036fb3ba794a8b9
_id='4490419882523069', version='4'>

Command took 0.29 seconds -- by harsh.tripathi@iiitb.ac.in at 1/28/2024, 11:40:51 PM on MT2022048 Tripathi's Cluster
```

Details about the Experiment:-

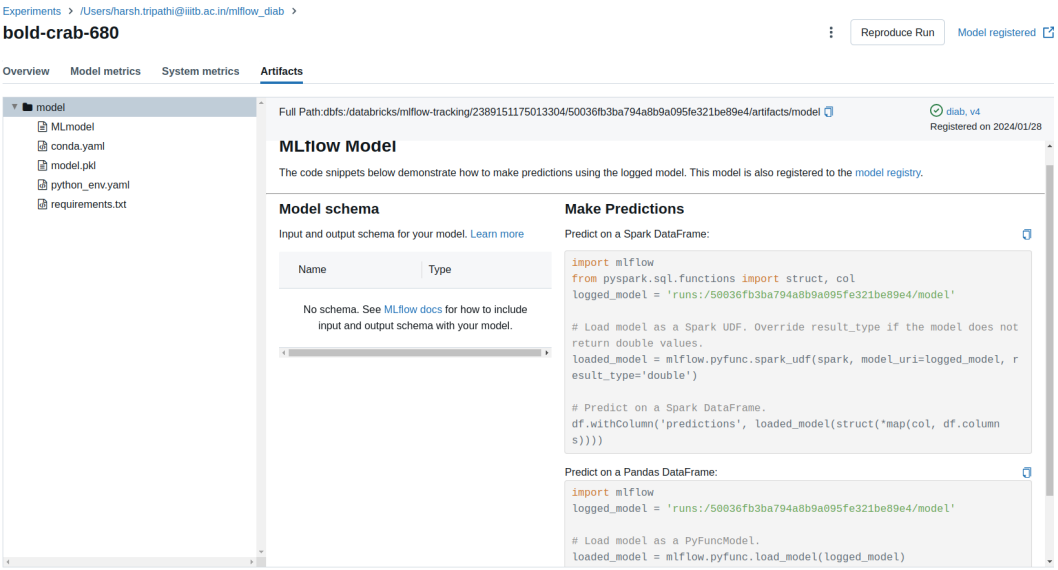
- The "Experiment" tab in Databricks is part of the MLflow integration.
- Users can create different experiments to organize and track machine learning runs.
- It allows logging runs with parameters, metrics, and artifacts for each machine learning task.
- Users can compare runs within the same experiment based on metrics, parameters, or metadata.
- The tab provides visualization of metrics over time, aiding in understanding model performance variations.
- Artifacts, such as model files, are automatically logged and can be managed within the "Experiment" tab.
- Runs within experiments are versioned, enabling tracking of changes over time.
- Experiments can be shared with collaborators for collaborative machine learning development.

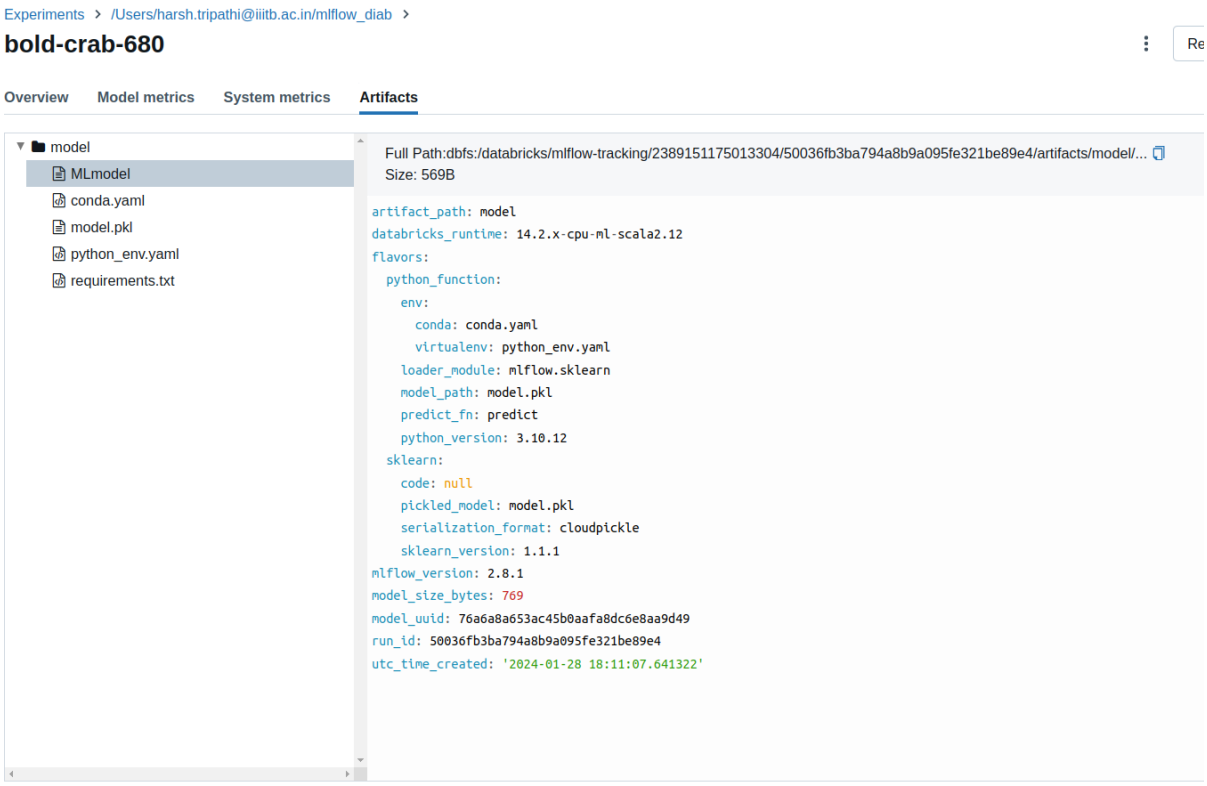
- The tab supports the entire machine learning lifecycle, including experimentation, model packaging, deployment, and monitoring.





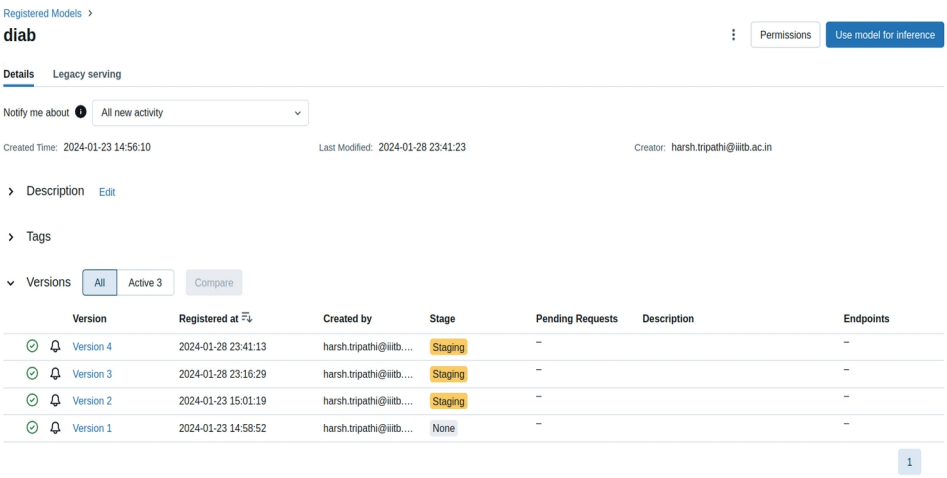
In this we can go to Artifacts tab to see further details of our model like yaml files:-



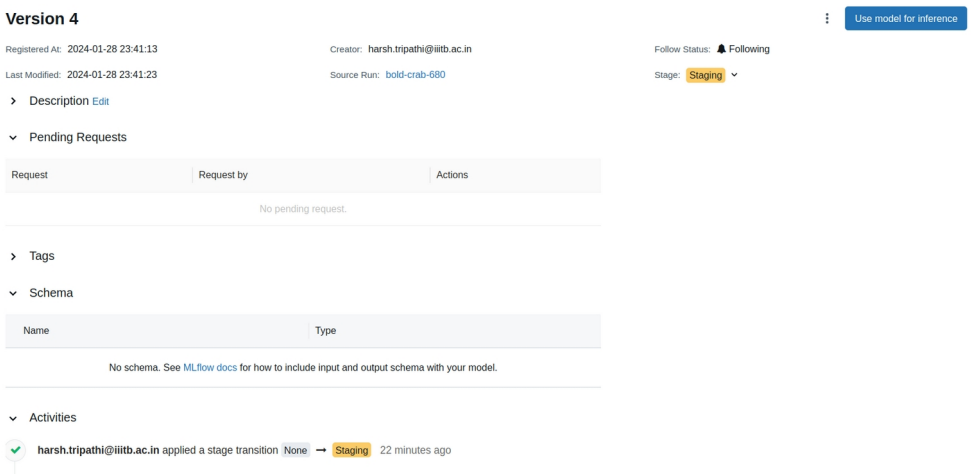


Now we will see Models tab in Databrick,

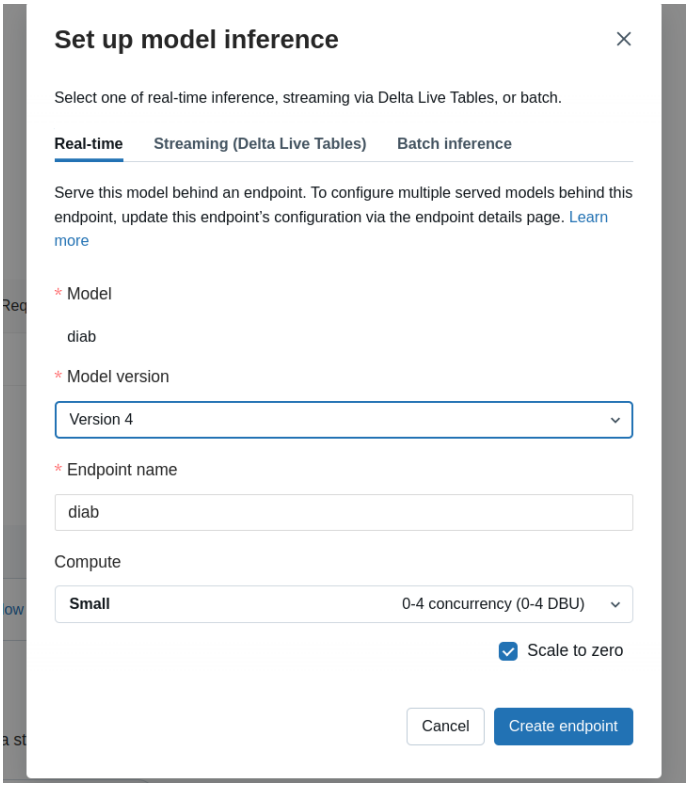
- Model Registration: The "Model" tab in Databricks, often associated with MLflow's Model Registry, allows users to register and organize machine learning models.
- Version Control: Users can manage different versions of a model, enabling version control, comparisons, and rollbacks to maintain consistency and reliability.
- Approval Workflow: The Model Registry supports approval workflows, where models can progress through stages like "Staging" and "Production" with appropriate approvals.
- Deployment: Registered models can be easily deployed from the Model Registry to production environments for serving predictions.
- Access Control: Access controls are implemented to ensure that only authorized users can modify or deploy models, enhancing security and governance.
- Monitoring and Logging: The Model Registry provides capabilities for monitoring model performance and logging relevant metrics during the model lifecycle.
- Collaboration: Facilitates collaboration among data scientists and stakeholders by serving as a central hub for managing and tracking the lifecycle of machine learning models.



As per our model we can change the states of it,



We can also use our model for inference such that we can create endpoint so that we can feed data from other sources.



Conclusion : -

A comparision between the two ways to apply MLOPS :-

Feature	Azure ML Studio	Databricks
Purpose	Comprehensive ML platform for model development and deployment.	Unified analytics platform for big data processing, analytics, and collaborative data science.
Model Building	Drag-and-drop interface for building ML models. Supports automated machine learning (AutoML).	Supports machine learning using MLlib and other popular ML libraries.
Integration with Azure	Tightly integrated with Azure services for data storage, integration, and deployment.	Can be integrated with Azure, providing access to Azure Data Lake Storage, Azure Blob Storage, and other services.
Data Preparation	Provides data preparation tools within the studio.	Offers interactive notebooks for data exploration and analysis.
Deployment Options	Supports deployment of models as web services (e.g., Azure Kubernetes Service, Azure Container Instances).	Focuses on big data processing and analytics rather than direct model deployment.
Collaboration	Suitable for individual or team-based ML model development.	Facilitates collaboration among data scientists, data engineers, and analysts in a shared workspace.
Experimentation and Tracking	Allows tracking and logging of experiments	Provides collaborative notebook development

	for understanding model performance.	with tracking features.
Scalability	Well-suited for smaller to medium-scale ML tasks.	Highly scalable, optimized for distributed data processing using Apache Spark.
Big Data Processing	Less emphasis on big data processing.	Primary focus on big data processing, leveraging Apache Spark.
Use Case	Ideal for end-to-end ML workflows.	Ideal for big data analytics, data engineering, and collaborative data science.

In summary, Azure ML Studio is tailored for end-to-end machine learning workflows, while Databricks is a unified analytics platform with a strong emphasis on big data processing and collaborative data science. The choice between them depends on the specific requirements of the project and the desired focus, whether it's on machine learning or big data analytics.

At the End it’s use choice.