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

Service brief: Azure Machine Learning

Azure Machine Learning is a cloud service for accelerating and managing the machine learning project lifecycle. Machine learning professionals, data scientists and engineers can use it in their day-to-day workflows: Train, test & deploy models and manage MLOps. You can create a model in Azure Machine Learning or use a model built from an open-source platform, such as Pytorch, TensorFlow or scikit-learn. MLOps tools help you monitor, retrain, and redeploy models.

You can take the following major approaches to build a model :

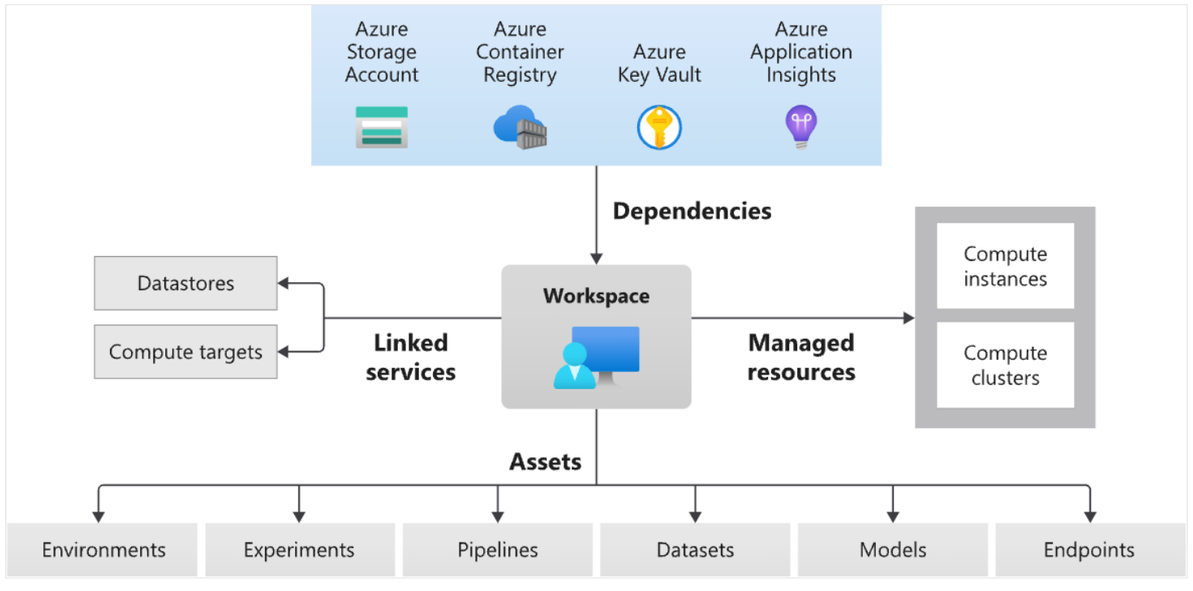
1. **AutoML (no-code)** : Automated machine learning, also referred to as automated ML or AutoML, is the process of automating the time-consuming, iterative tasks of machine learning model development. Traditional machine learning model development is resource-intensive, requiring significant domain knowledge and time to produce and compare dozens of models. With AutoML, you just need to provide the problem type you are trying to solve (Regression, Classification or Time Series) and the dataset. The model with automatically handle tasks like data cleaning, data pre-processing, model selection, hyper-parameter tuning, try out a combination of different models and present you with the model that gives the best results. You can then deploy these models for real time inference.
   You can also invoke AutoML experiments using pre-built functions in Azure ML Python SDK.

2. **Designer (low-code)** : Azure Machine Learning designer is a drag-and-drop interface used to train and deploy models in Azure Machine Learning. A model trained in AML Designer is referred to as a Pipeline, since you build this model by integrating various modules in the Designer, which act as separate steps. You have pre-built modules for tasks like data cleaning & transformation, feature selection, ML algorithms, Model scoring & evaluation that you can simply drag on to the screen to include in your pipelines. You also have modules for specific ML use cases like Computer Vision, Text Analytics, Anomaly detection etc. In case you want to write custom code or queries, you also have the option to include modules for Python, R and SQL Scripts. You can deploy these pipelines as real time or batch inference endpoints.

3. **Notebooks (code-only)** : Azure Machine Learning notebooks allow you to create and manage your custom models from scratch using a code first approach. All major frameworks like TensorFlow, PyTorch, ScikitLearn etc are supported in the AML Kernels. Azure also provides SDKs in languages like Python and R to easily manage your resources, end to end ML lifecycle and model artefact.

Azure Machine Learning provides you with a lot of functionality to help you manage you model and model lifecycle, both in a GUI based and a code-based approach. You can create & version notebooks, datasets, experiments, models, environments etc. You can also integrate Azure DevOps co create Continuous Integration and Continuous Development pipelines.

Diagram: Functional Architecture



We will be discussing all these components as and when we come across them in the workshop.

Scenario Brief

Banking Scenario

A bank ABC is on a digital transformation for all its departments. Bank has a growing customer base where majority of them are liability customers vs asset customers.

Liability Customer - the one who deposits into the bank (depositors).

Asset Customer - the one who borrows from the bank (borrowers).

The bank is interested in expanding the borrowers base rapidly to bring in more business via loan interests. A campaign that the bank ran in last quarter showed an average single digit conversion rate. Digital transformation being the core strength of the business strategy, marketing department wants to devise effective campaigns with better target marketing to increase the conversion ratio to double digit with same budget as per last campaign.

Information about the Data

The data consists of the following attributes:

1. ID - Customer ID
2. Age - Customer's age.
3. CustomerSince - Customer of the bank since (years).
4. HighestSpend - Customer's highest spend so far in single transaction.
5. ZipCode - Customer's zip code.
6. HiddenScore - A score associated to the customer which is masked by the bank.
7. MonthlyAverageSpend - Customer's monthly average spend.
8. Level - A level associated to the customer which is masked by the bank.
9. Mortgage - Customer's mortgage, if any.
10. Security - Customer's security asset with the bank.
11. FixedDepositAccount - Customer's fixed deposit account with the bank.
12. InternetBanking - whether the customer uses internet banking.
13. CreditCard - whether the customer uses bank's credit card.
14. LoanOnCard - whether the customer has a loan on credit card [Target Column]

Objective

Build a Machine Learning model to perform focused marketing by predicting the potential customers who will convert using the historical dataset. Steps and tasks:

1. Import and warehouse data:
   • Import all the given datasets and explore shape and size of each.
   • Merge all datasets onto one and explore final shape and size.

2. Data cleansing:
   • Explore and if required correct the datatypes of each attribute.
   • Explore for null values in the attributes and if required drop or impute values.

3. Data analysis & visualisation:
   • Perform detailed statistical analysis on the data.
   • Perform a detailed univariate, bivariate and multivariate analysis with appropriate detailed comments after each analysis.

4. Data pre-processing:
   • Segregate predictors vs target attributes.
   • Check for target balancing and fix it if found imbalanced.
   • Perform train-test split.

5. Model training, testing and tuning:
   • Design and train a Logistic regression and Naive Bayes classifiers.
   • Display the classification accuracies for train and test data.
   • Display and explain the classification report in detail.
   • Apply all the possible tuning techniques to train the best model for the given data.
   • Select the final best trained model with your comments for selecting this model.

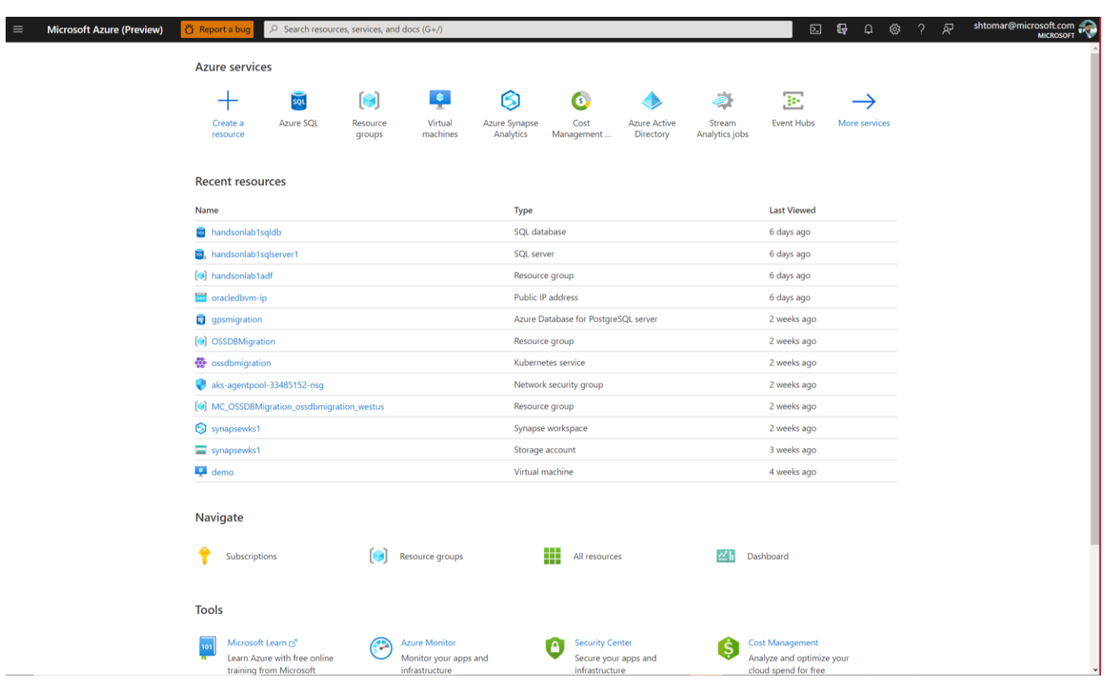
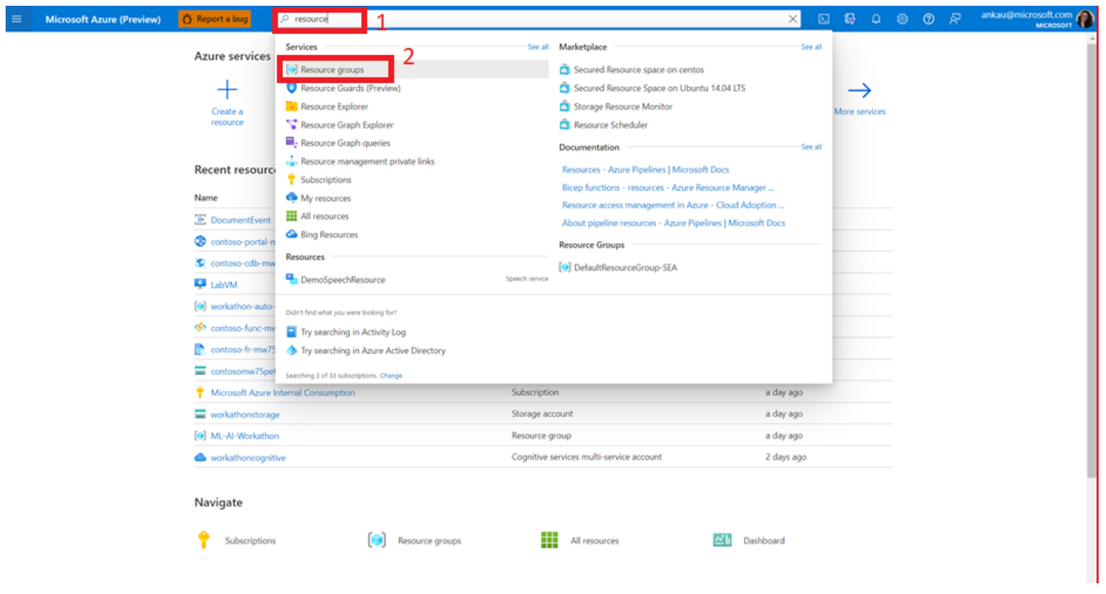
6. Conclusion and improvisation:
   • Write your conclusion on the results.

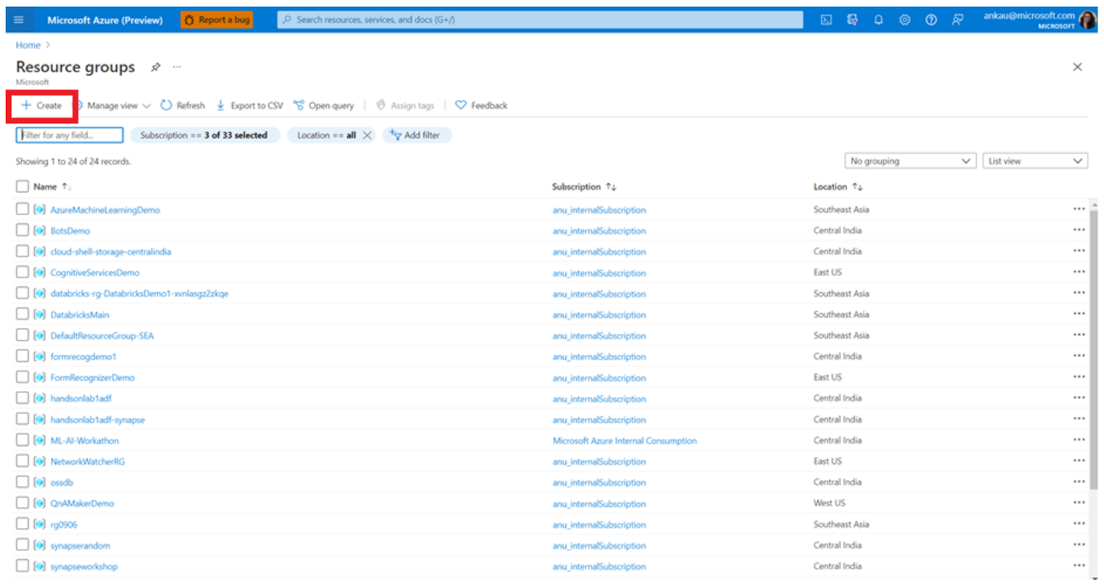
Step by step hands on guide to go from *Zero to Hero*

Pre-requisites

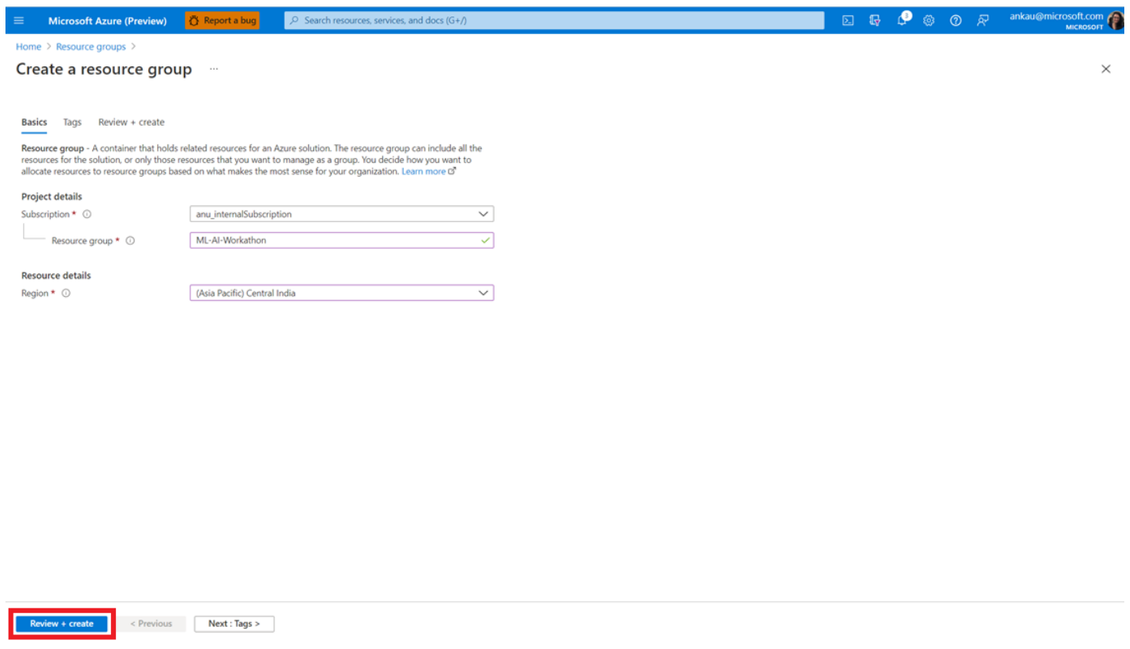
- An active Azure Account
  - You can use your current Azure Subscription or get started by creating a free trial account (https://azure.microsoft.com/en-in/free)
- Download the data for model from Data folder.

Let's get started!

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



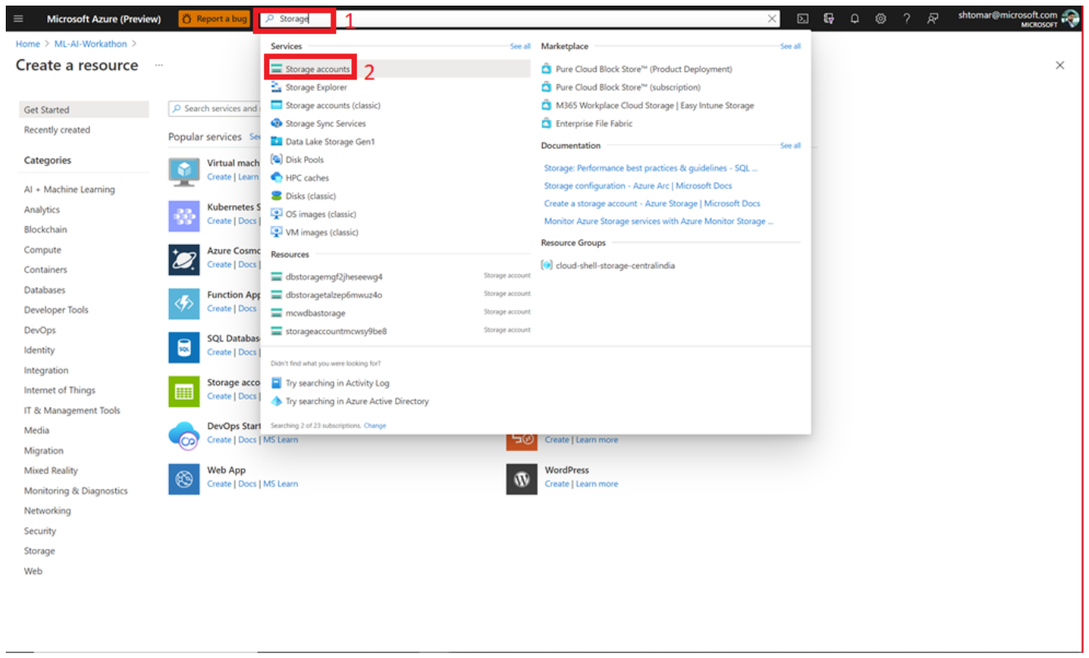
Click create to create a new resource group.

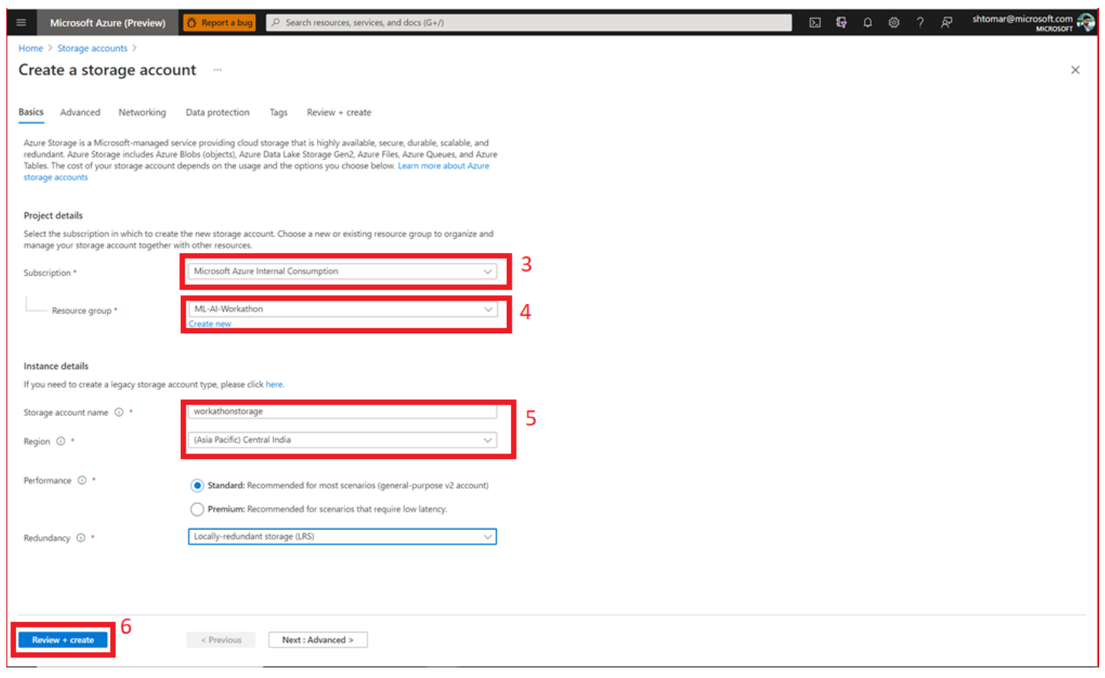


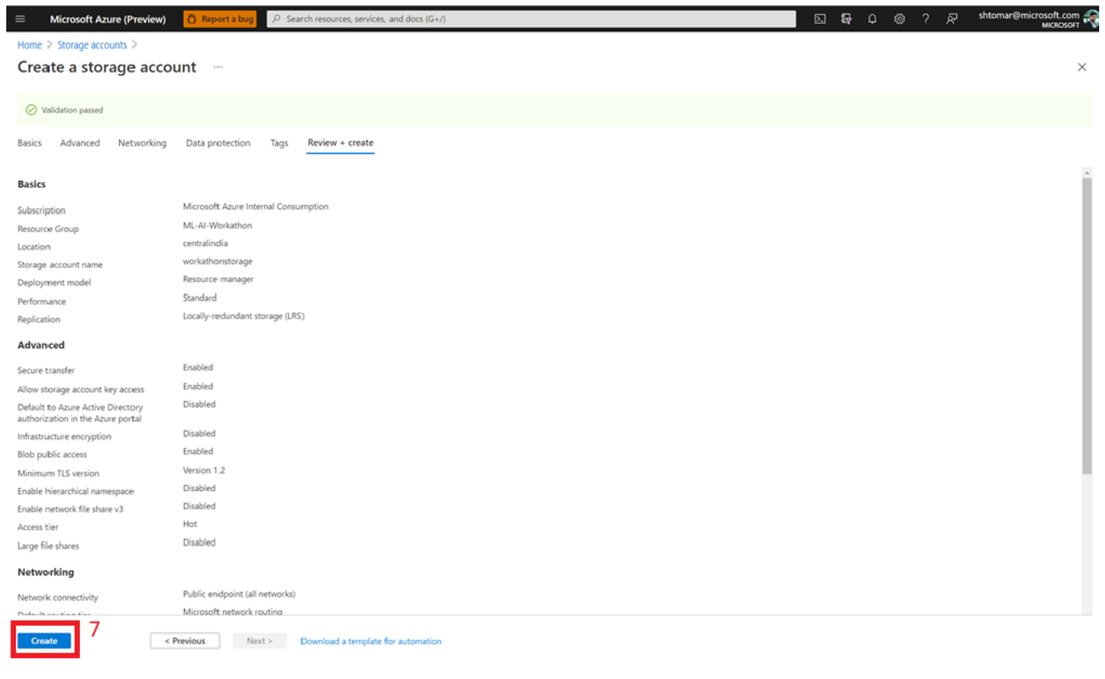
Enter the details –

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

Click Review + Create.



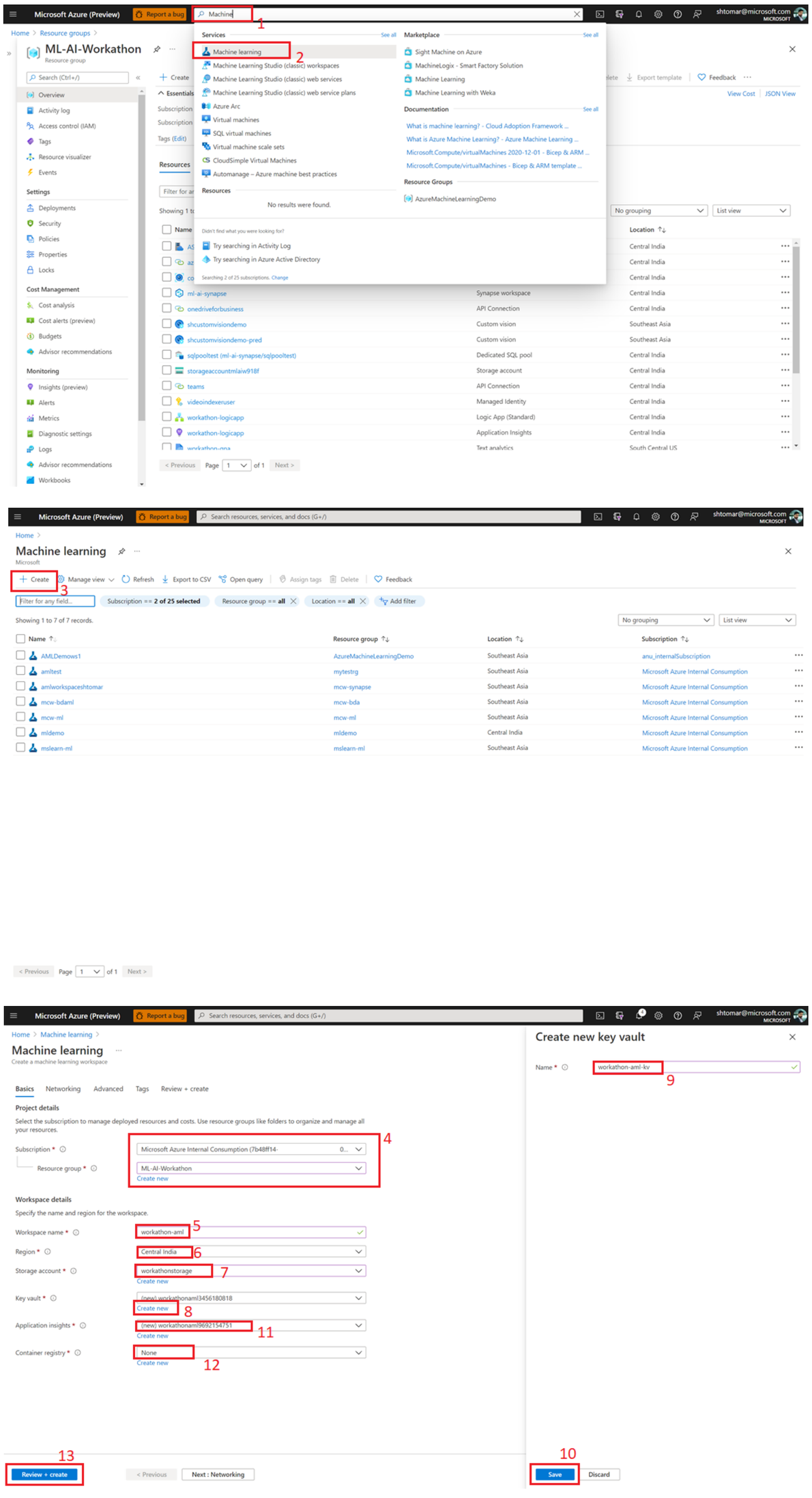




Create Storage Account in Azure Portal

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

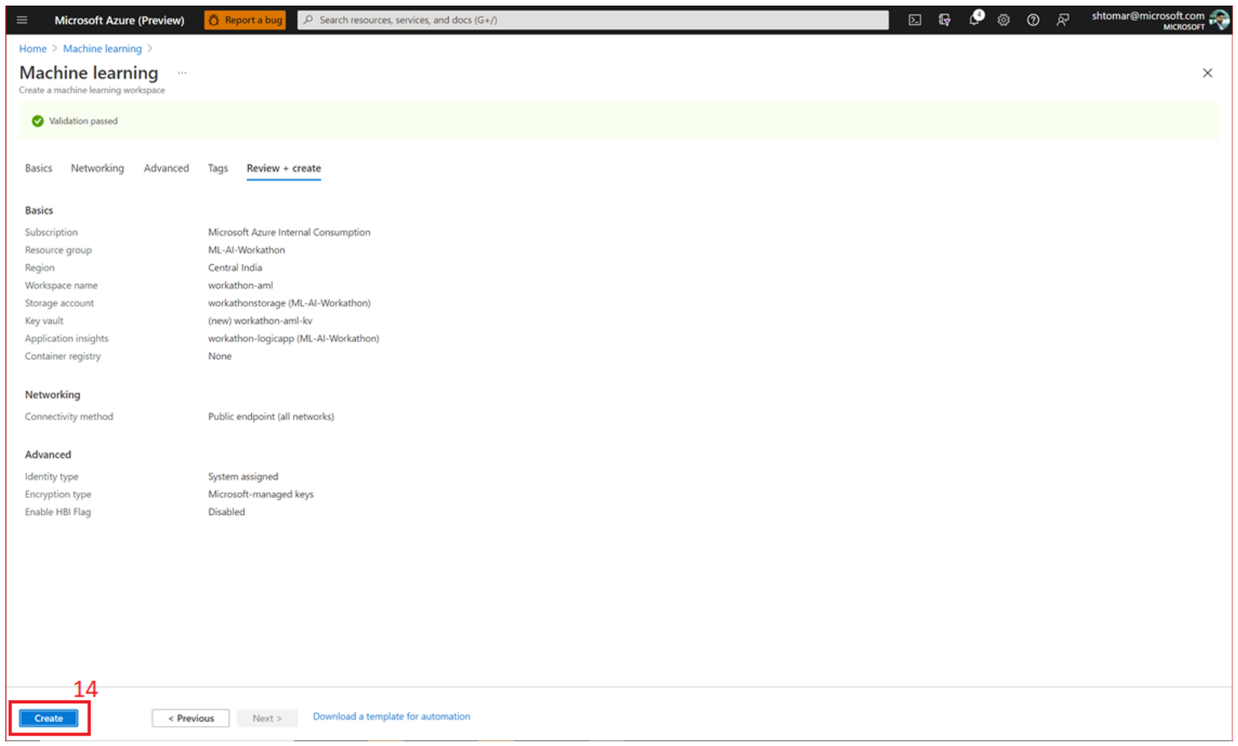
Search for Storage Account in the search bar and select Storage account resource. (Step 1,2)
Enter the details highlighted in the screenshot. (step 3-5)
Click Review + Create. (step 6)
Verify the details and hit Create. (step 7)



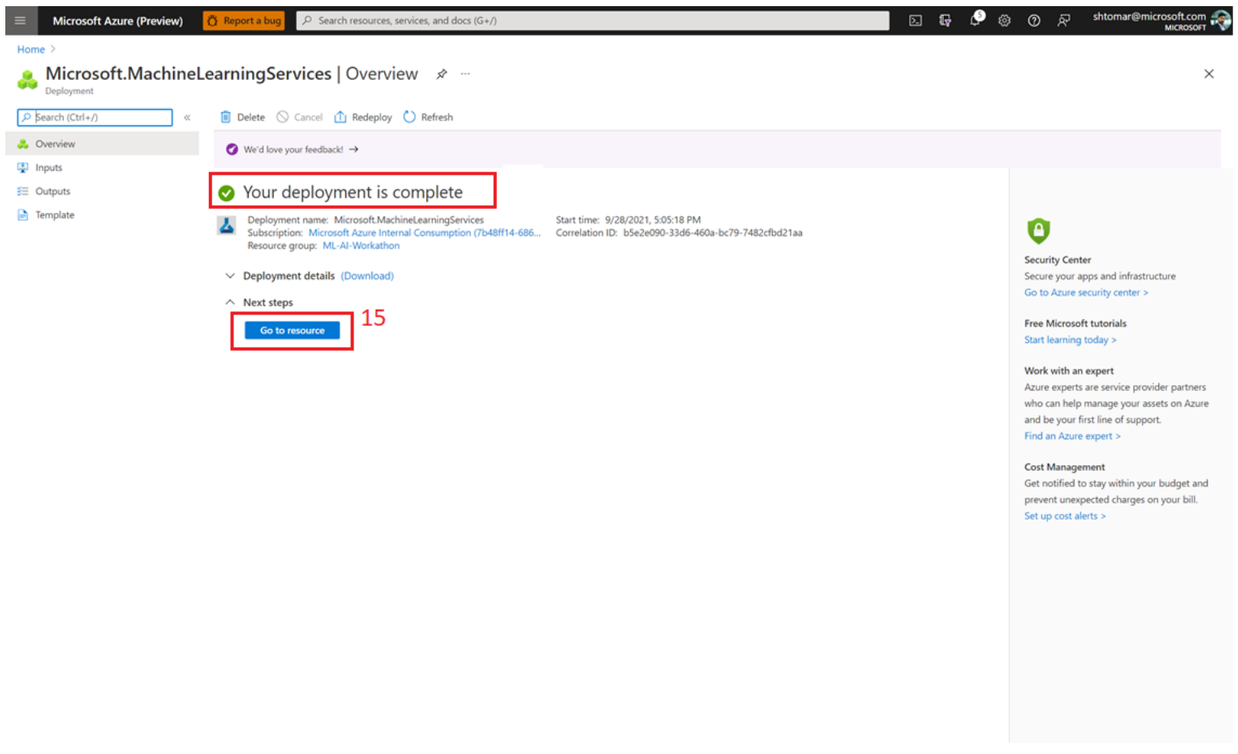
Create Azure Machine Learning resource

Azure Machine Learning workspace is like a one stop to create and manage all your resources such as notebooks, datasets, compute, experiments, models, endpoints etc.

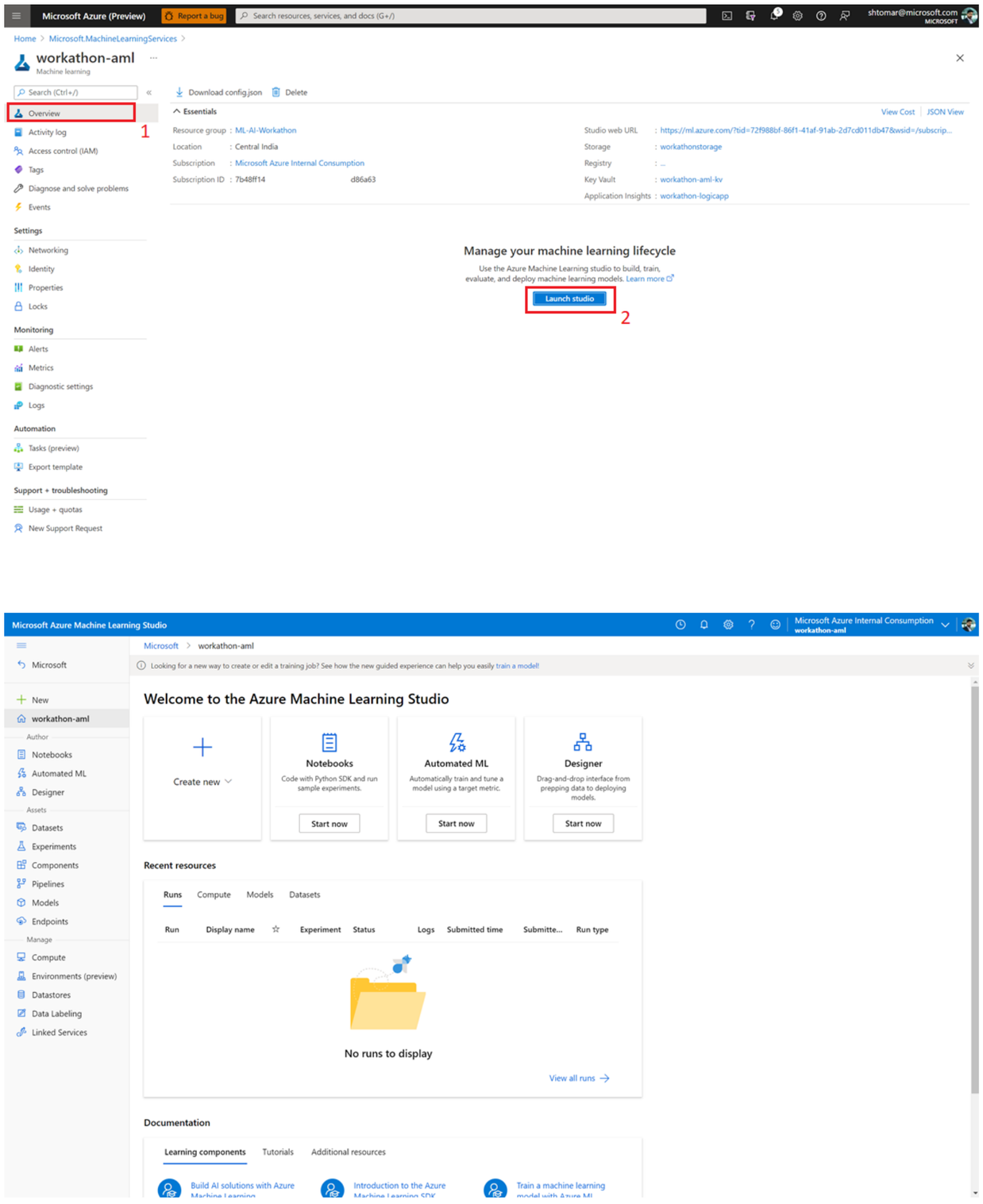
1. Search for Machine Learning
2. Select Azure Machine Learning resource
3. Click + Create
4. Select the subscription and resource group in which you want to deploy the resource
5. Provide a workspace name
6. Region : India
   [The Machine Learning resource and all the related resources will be provisioned in this region]
7. Select the Storage account you created above
   [Storage Account is used to save your artefacts such as datasets, notebooks, logs etc.]
8. Click create new for the Key Vault
   [Key vault is used to save the connection strings and secrets for any dataset or datastores you connect to you ML workspace]
9. Give the key vault a custom name in ‘Create new key vault’ wizard
10. Click Save
11. Let Azure automatically create the Application Insights resource for you.
    [This is used to monitor your applications. It will automatically detect performance anomalies, and includes powerful analytics tools to help you diagnose issues and to understand what users do with your app]
12. Container Registry : None
    [This is used to host the models that you deploy to Azure Container Instance. We will provision this later.]
13. Click Review + Create.



14. Verify the details and click Create.

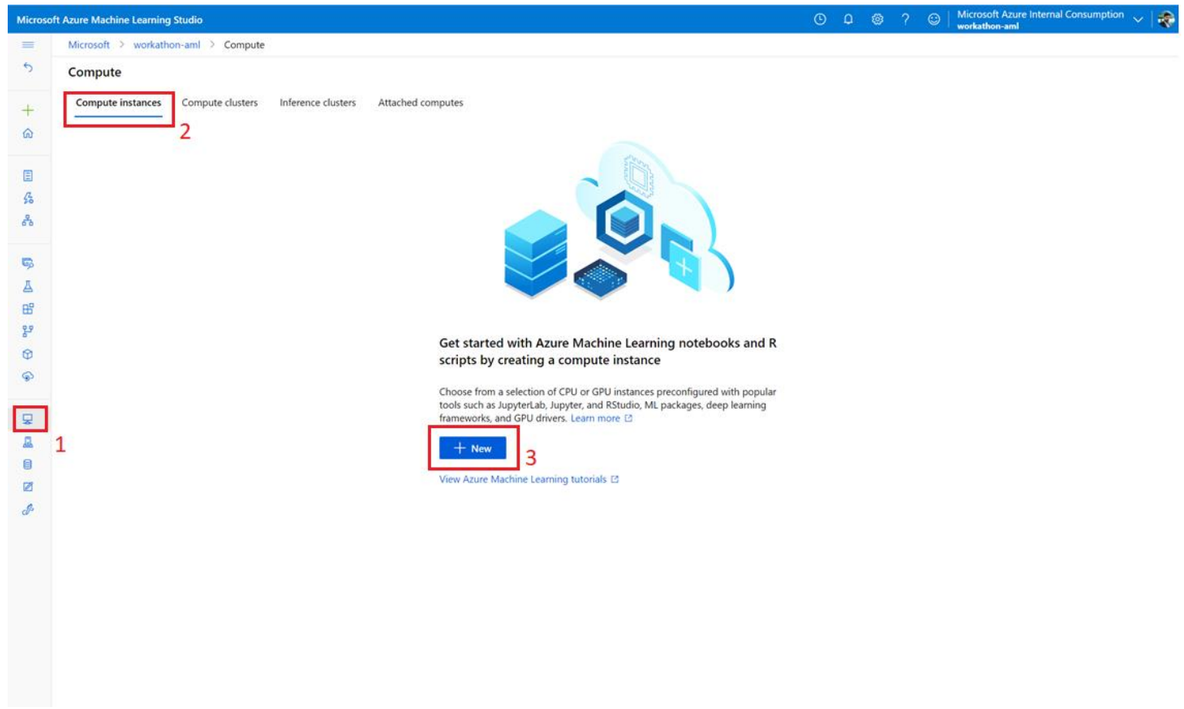


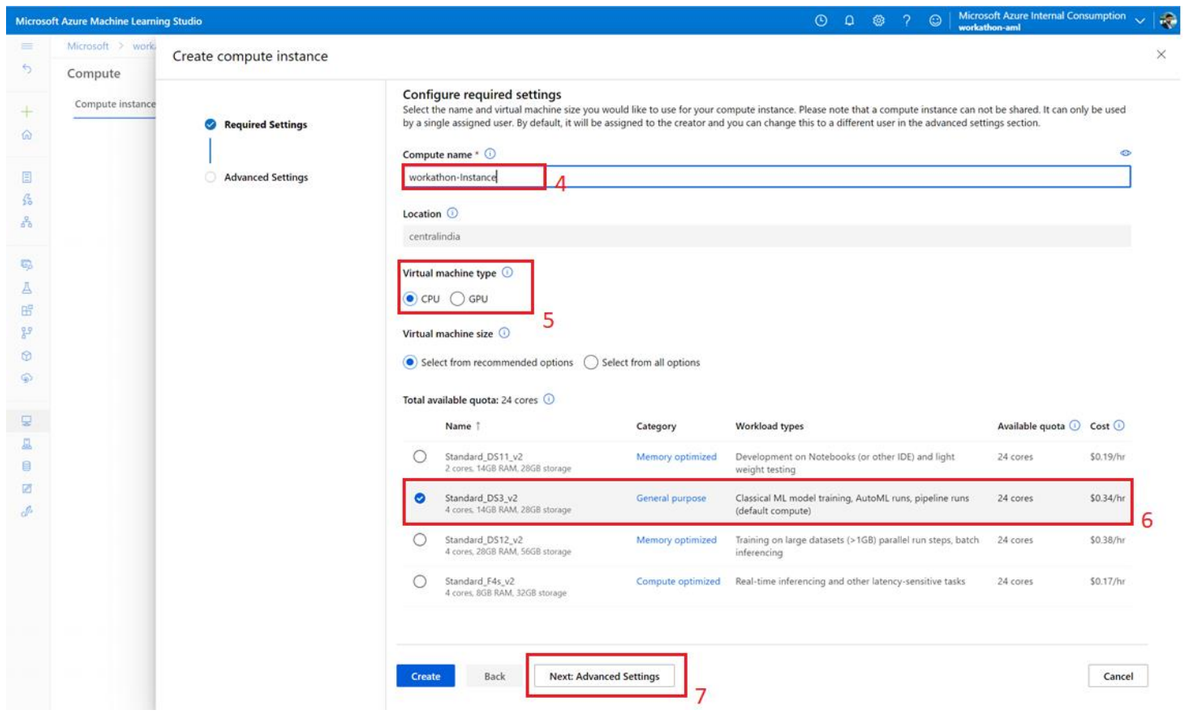
15. Once the resource is deployed, click ‘Go to resource’.

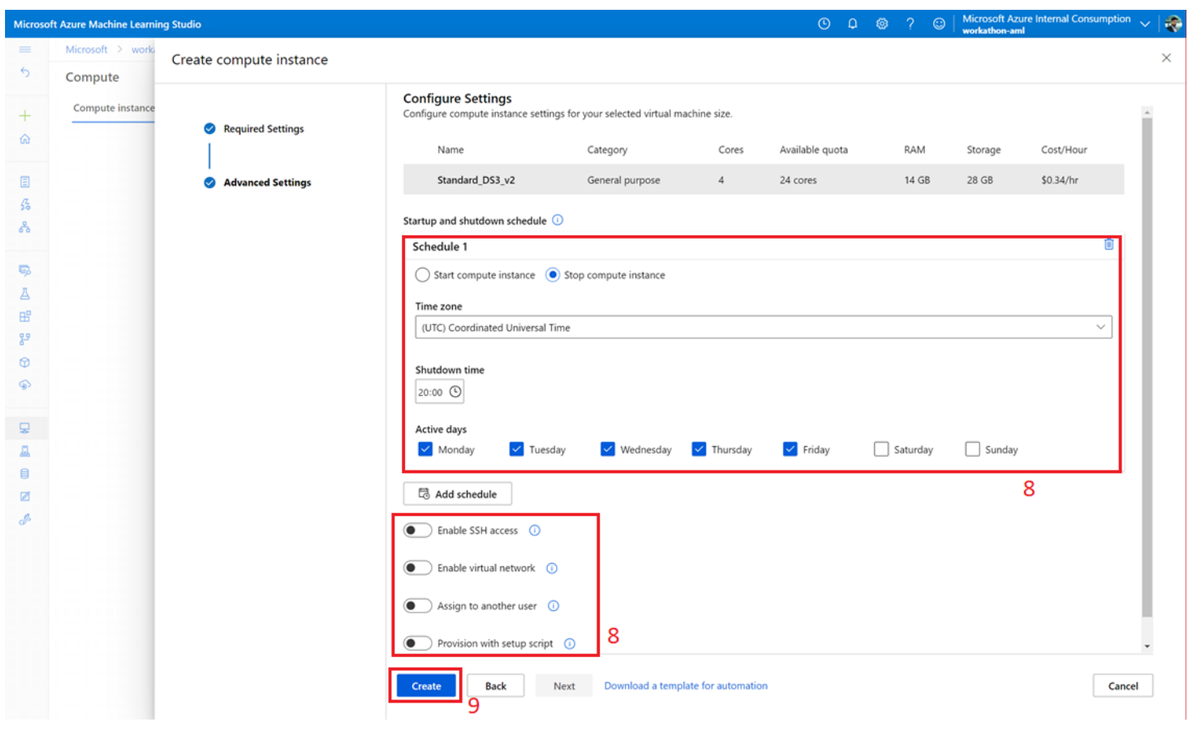


Launch AML Studio

1. Select the Overview tab in the AML resource that you provisioned
2. Click ‘Launch Studio’







Azure Machine Learning Compute

AML allows you to provision & use 4 different types of computes for your experiments :

a. **Compute Instance** : This is a fully configured and managed development environment in the cloud for machine learning. It can also be used as a compute target for training and inferencing for development and testing purposes. This compute is not shareable.

b. **Compute Cluster** : This is a production grade model training compute resource with multi-node scaling capabilities. This resource that can be shared with other users in your workspace. The compute scales up automatically when a job is submitted and can be put in an Azure Virtual Network.

c. **Inference Cluster** : When performing inference, Azure Machine Learning creates a Docker container that hosts the model and associated resources needed to use it. This container is then used in a compute target. The compute target you use to host your model will affect the cost and availability of your deployed endpoint. For real time inference, you can have 2 major compute target options :

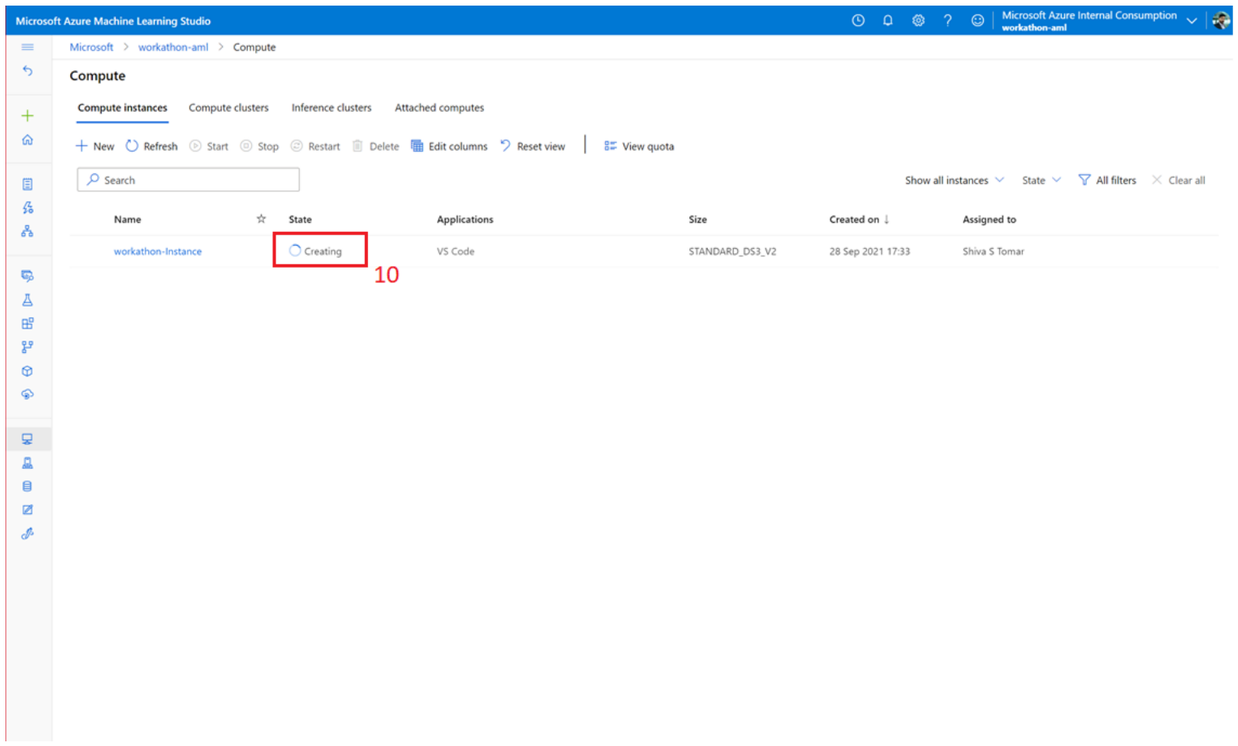
   i. **Azure Container Instance** : This is recommended for dev/test purposes only or for low-scale CPU-based workloads
   ii. **Azure Kubernetes Service** : This is recommended for production workload. Provides fast response time and autoscaling of the deployed service.

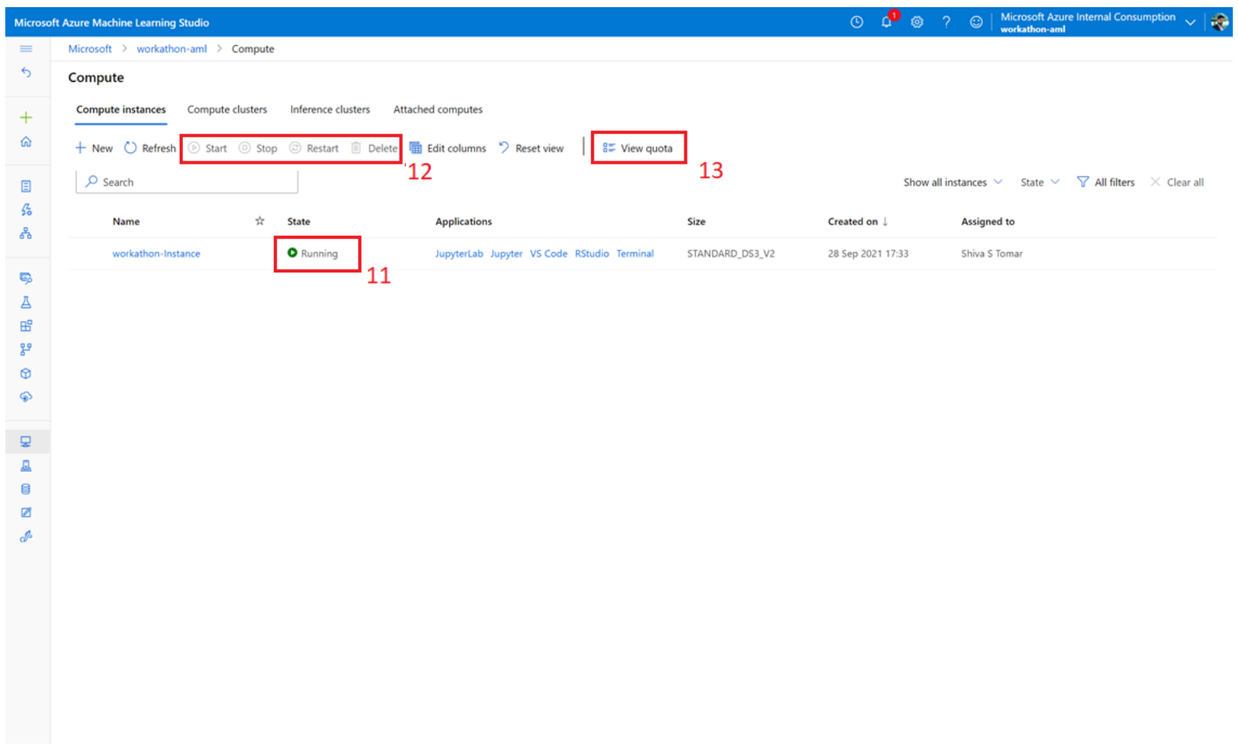
d. **Attached Compute** : This type of compute target is created outside Azure Machine Learning and then attach it to your workspace. This is like an unmanaged compute and required you to maintain them. This includes compute targets like Azure Synapse (Spark Pool), Azure Databricks, Azure Data Lake Analytics etc.

Create a compute instance

We will use a Compute Instance for data cleaning, preparation, exploration and training a couple of models for testing purposes.

1. Go to the compute tab
2. Select ‘Compute Instance’
3. Click + New
4. Provide a unique name to your compute instance
5. Virtual machine type : CPU
   [Choose this depending on your processing needs. GPU is used for workloads such as Deep Learning algorithms that require heavy compute.]
6. Virtual Machine size : Select a VM of your choice (We are selecting ‘Standard_DS3_v2’ from the recommended options)
   [AML offers a variety of VM categories such as General purpose, compute optimised, memory optimised, storage optimised, high performance. You can view all these by selecting all options. In a production scenario, you will choose this basis your workload requirement.]
7. Click Next : Advanced Settings
8. You are charged for the compute instance if it’s running. To save costs, you can set schedules to start & stop the compute instance basis your needs. You can also choose amongst other advanced options like Enable SSH, Virtual Networks etc.
9. Click Create



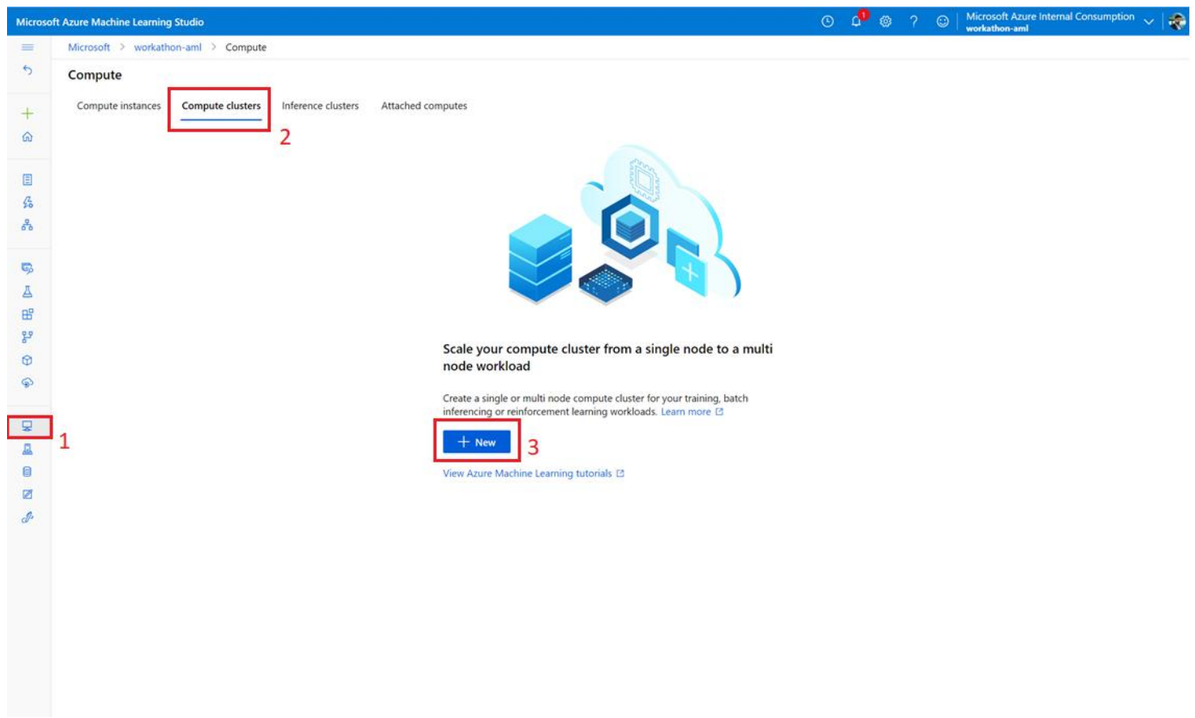


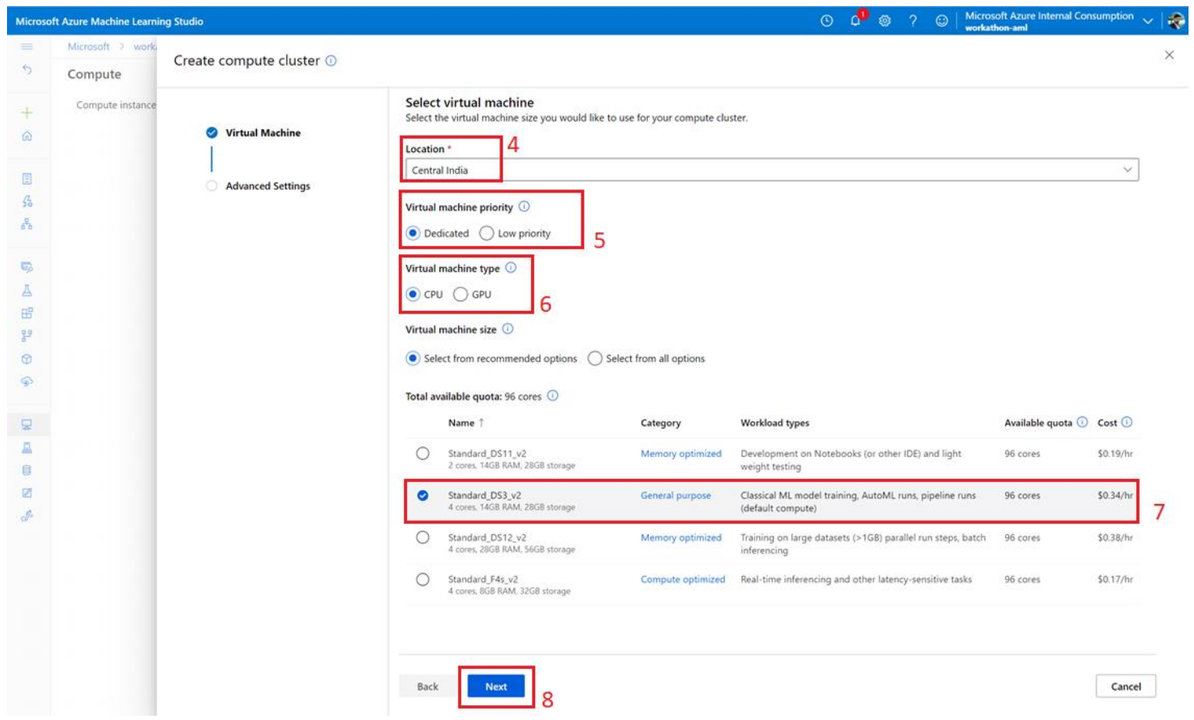
10. The Compute Instance will take a while to create

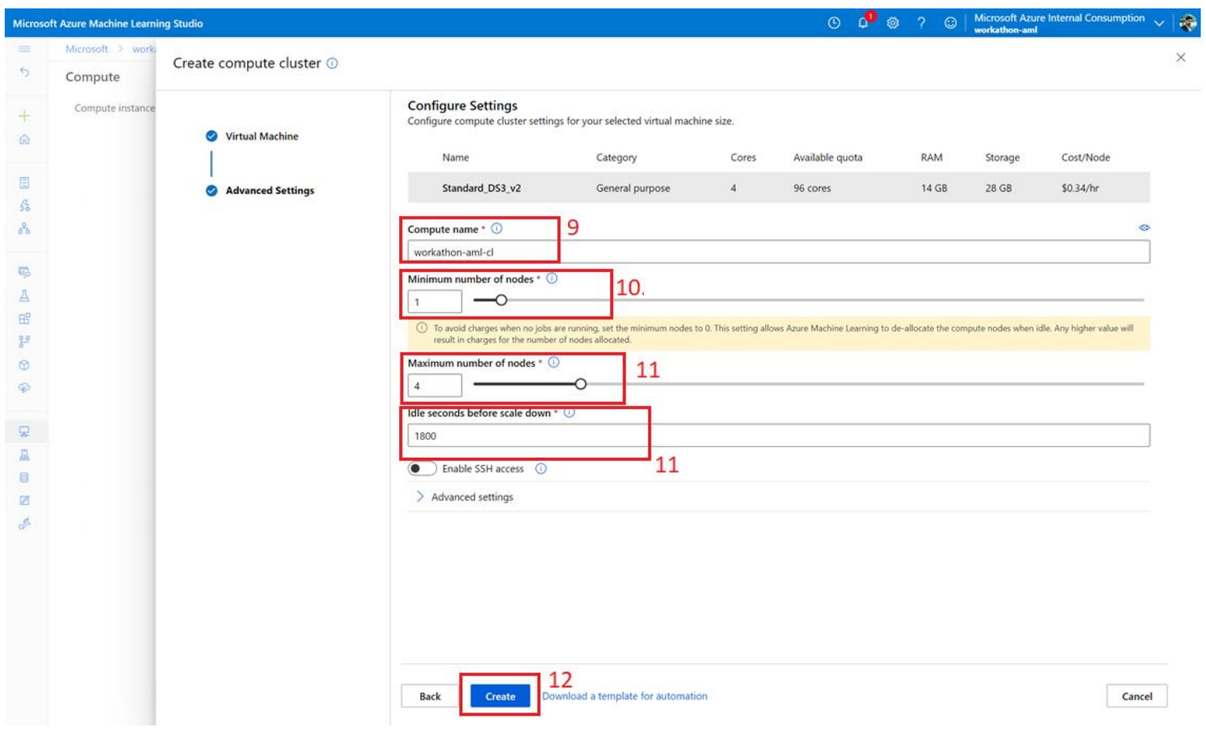
11. Once created, you will see the status as Running

12. The AML Studio GUI provides you various options to manage your compute, for example :
    a. Start – to start a stopped compute
    b. Stop – to stop a running compute
    c. Restart – To restart a running compute
    d. Delete – To delete a provisioned c

13. View quota : Azure Machine Learning quota are preconfigured limits, which specifies the maximum number of cores you can use at any given moment. This option shows the quota usage and limits across all workspaces in your subscription.







Create a Compute Cluster

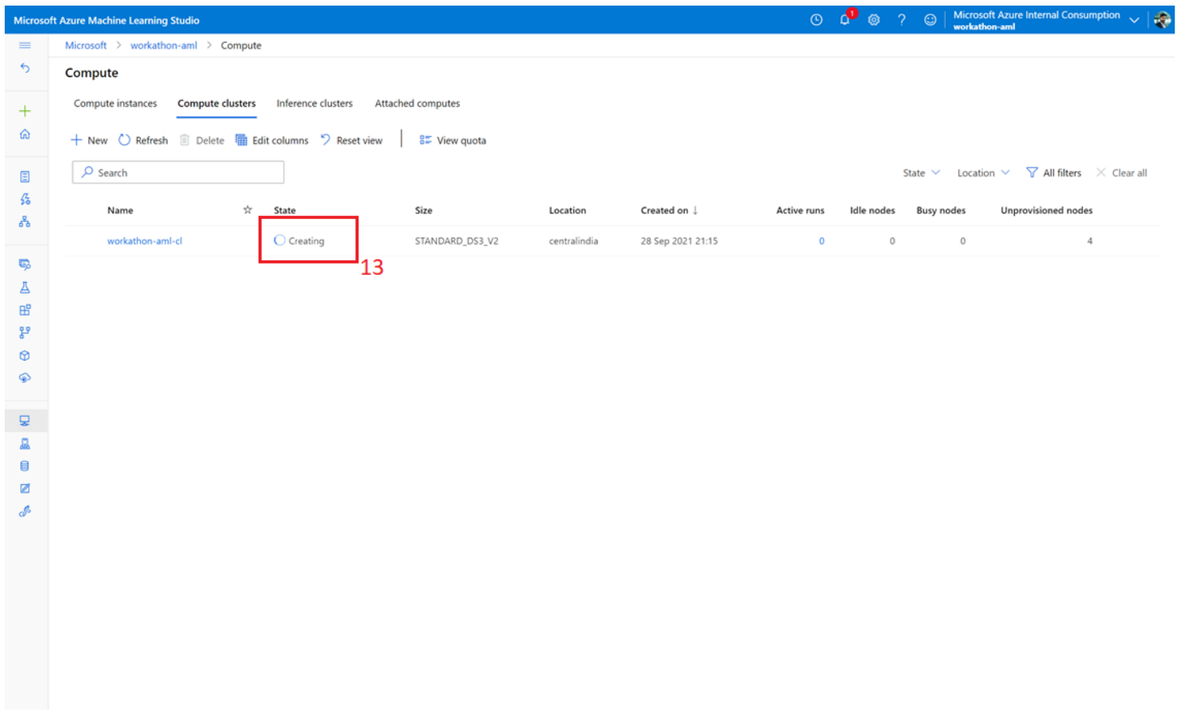
Compute Cluster can be used for development, training and testing of a production grade model. Since this is a multi-node compute, you can scale up and scale down the compute nodes either automatically or manually.

1. Select the compute tab
2. Go to the ‘Compute clusters’ tab
3. Click + New

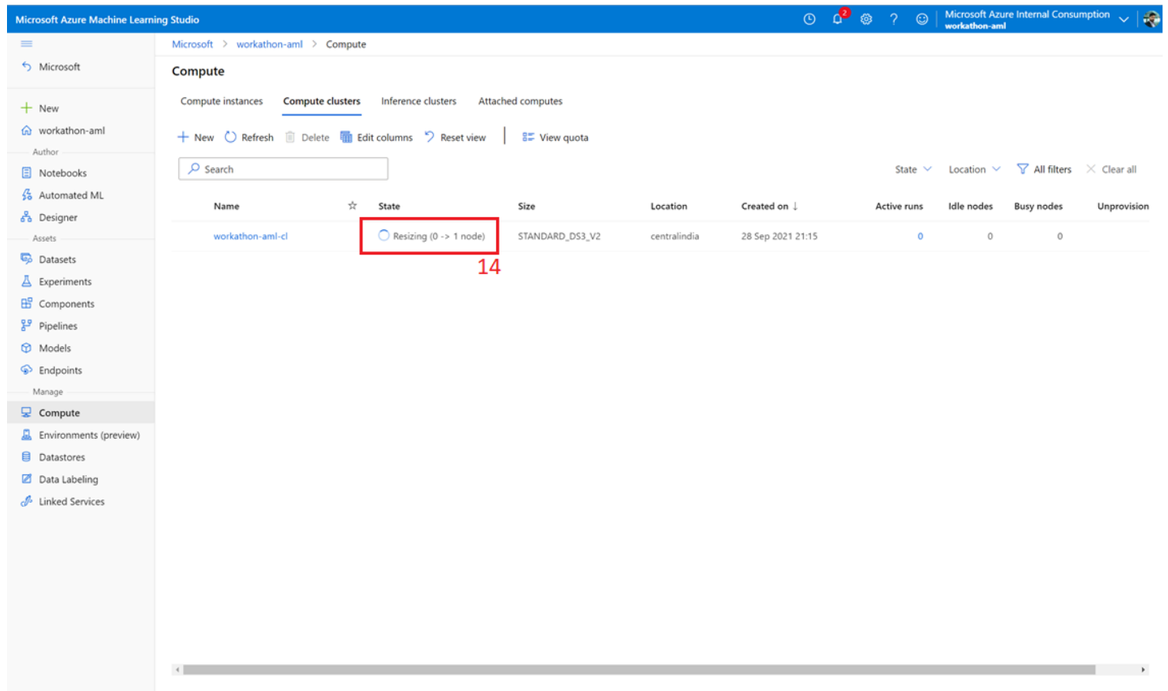
4. Location : select the region where you want to deploy this cluster on Azure
5. Virtual machine priority : this allows you to choose between 2 VM priority categories:
   a. Low priority : These VMs are cheaper but don't guarantee the compute node availability. You share the compute nodes with others and hence your job may be pre-empted. This can be used for testing and development purpose.
   b. Dedicated : These VM resources are not shared with others and are reserved for you, thus your jobs are never pre-empted. This is ideal for production scenarios.

   We will be choosing a dedicated VM.
6. Virtual machine type : Like compute instance, you can choose between CPU & GPU compute. We will select a CPU machine.
7. Virtual machine size : You get similar options as Compute instance in terms of VM categories. You can select the appropriate category and size depending upon your workload. We will select be selecting General purpose ‘Standard_DS3_v2’ VM in this workshop.
8. Click Next

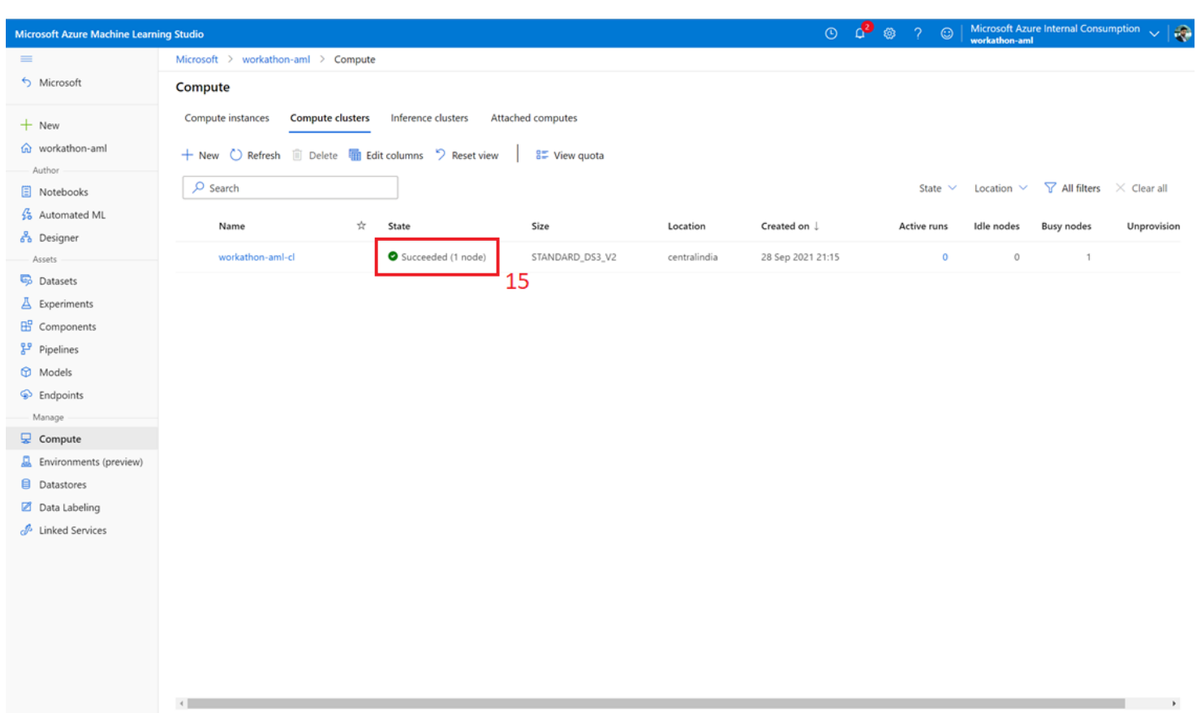
9. Compute name : Give your cluster a unique name
10. Minimum number of nodes : This is the number of compute nodes that are always running. Generally, you will set this to 0 to save costs when there are no jobs running on the cluster. It will scale up as soon as you set this as you send a job request to the cluster. For this workshop, we have set this to 1, to have 1 node always running and save on time.
11. Maximum number of nodes : This is the number of nodes that the cluster will scale up to. Basis the job you are running, AML will automatically scale up your compute nodes. The cluster will start scaling down to the minimum number of nodes if it is idle for the time set in ‘Idle seconds before scale down’ parameter.
12. Click Create.



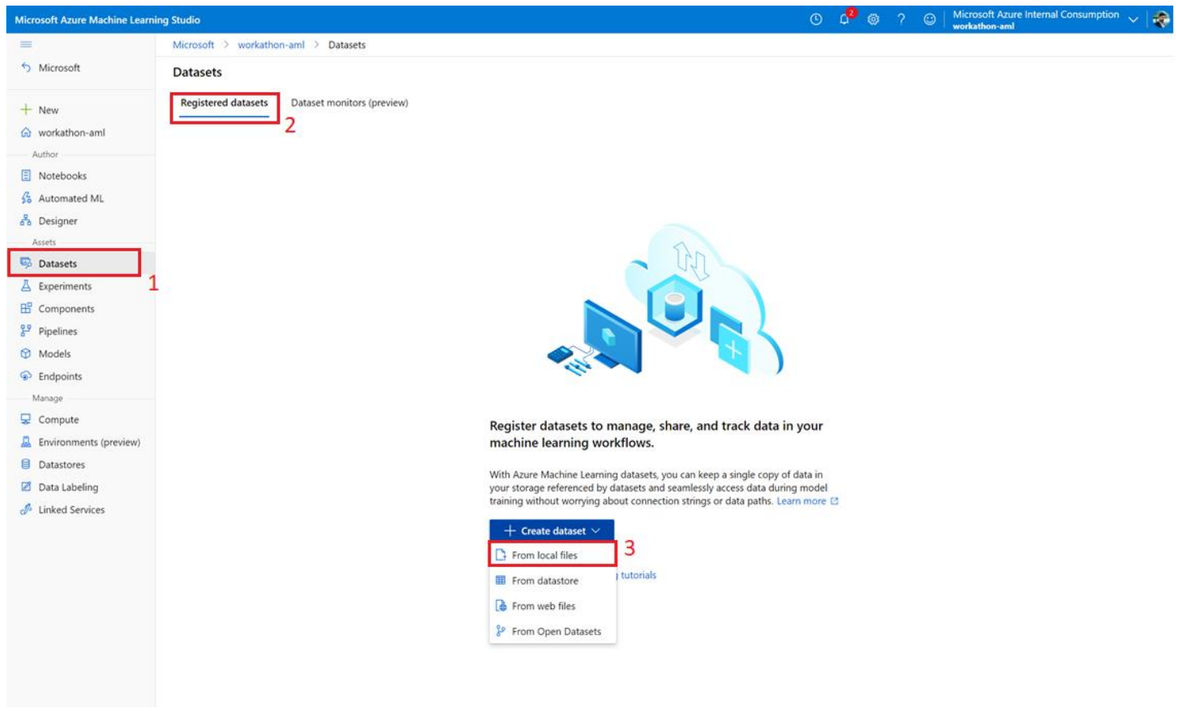
13. The cluster will take a while to spin up.

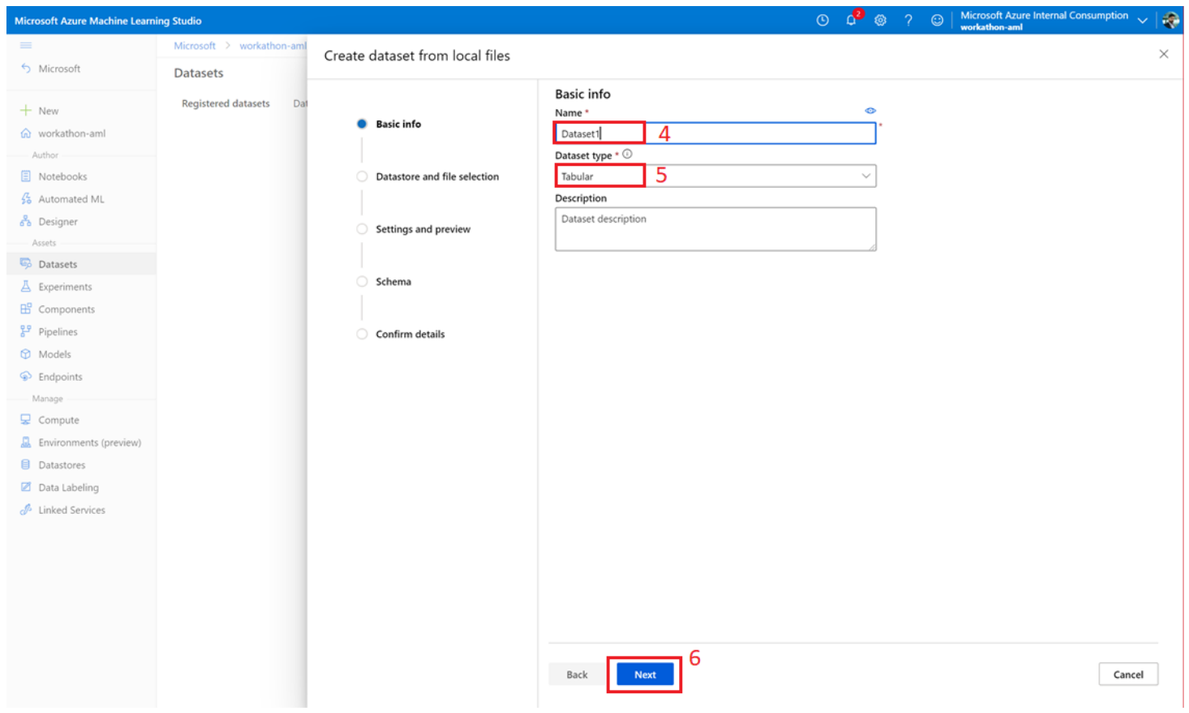


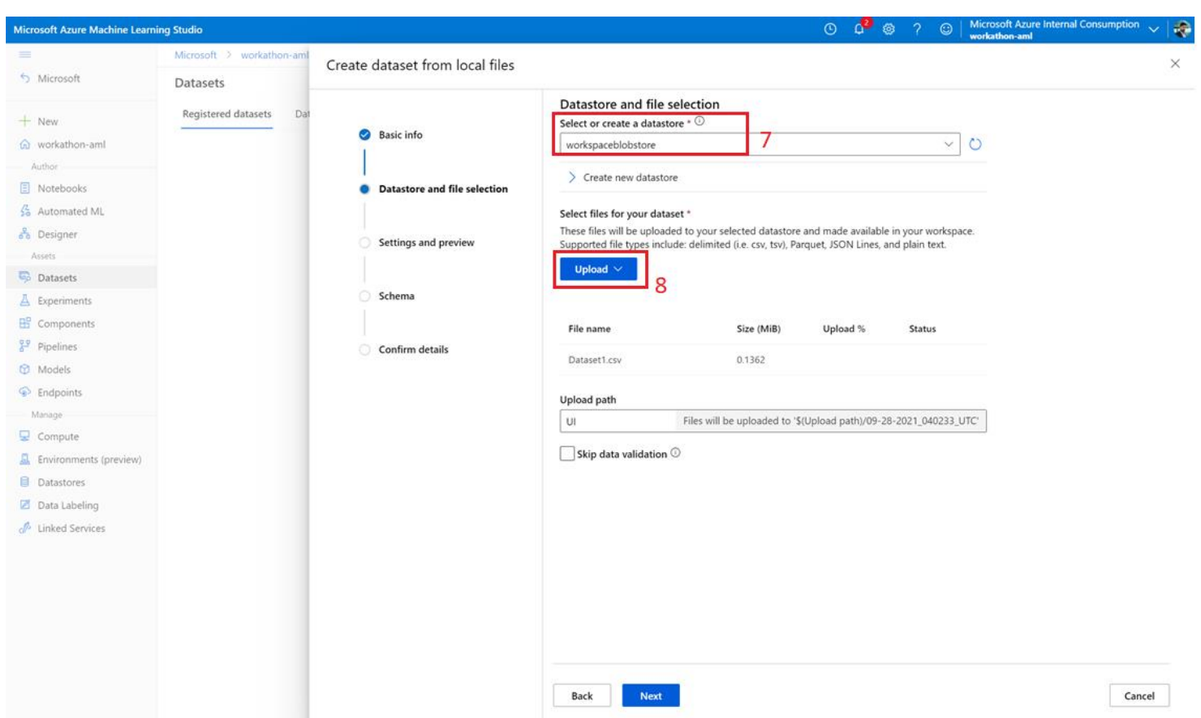
14. Since we had set the minimum number of nodes to 1, this will resize to 1 node as soon as it is provisioned.



15. You will see the state as succeeded once you cluster is ready.







Register a Dataset (aka upload)

To use data for model training & testing, we will first register the data as a dataset. These are 2 ways methods to register datasets :

a. GUI based approach – we have used this approach to upload data for now
b. Code based – you can use Python or R code to upload datasets. Here is a sample code snippet that we have picked up from the notebook we will be building in this workshop

```
local_path = 'data/finance.csv'
df_upsampled.to_csv(local_path)

# get the datastore to upload prepared data
datastore = ws.get_default_datastore()

# upload the local file from src_dir to the target_path in datastore
datastore.upload(src_dir='data', target_path='data')

#create a dataset referencing the cloud location
ds = Dataset.Tabular.from_delimited_files(datastore.path('data/finance.csv'))

financial = ds.register(workspace=ws, name='finance_ds', description='finance training data')
```

You can register datasets from 4 sources :

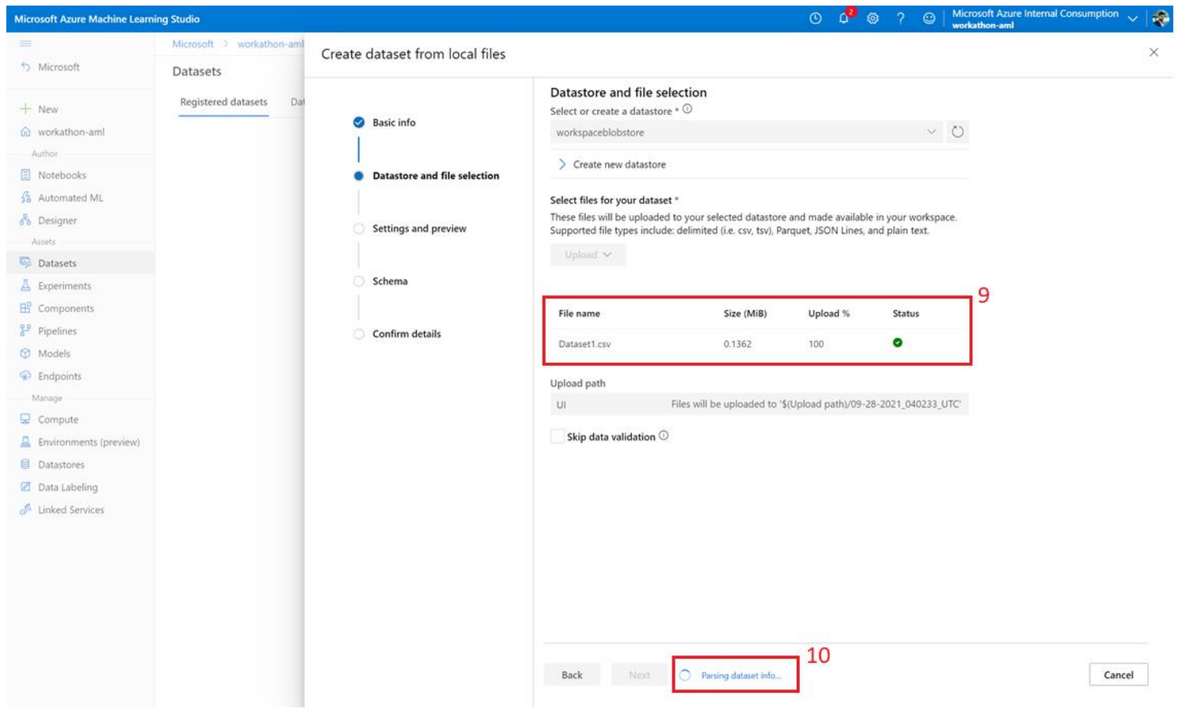
a. Local files – to register datasets using files from your local machines
b. Datastore – to register datasets using data saved in different Azure storage locations or databases. We will be discussing datastore in 'Explore Datastores' section
c. Web files – to register datasets using data saved on web URLs
d. Open Datasets – These are some pre-built datasets related to different domains that are available to you. You can leverage these for learning purposes. For example, you can find datasets related to healthcare, travel, census etc.

In this workshop, we will first be uploading the raw unprocessed data using local files & GUI based approach and later we will register the processed dataset using the Datastore & code-based approach.

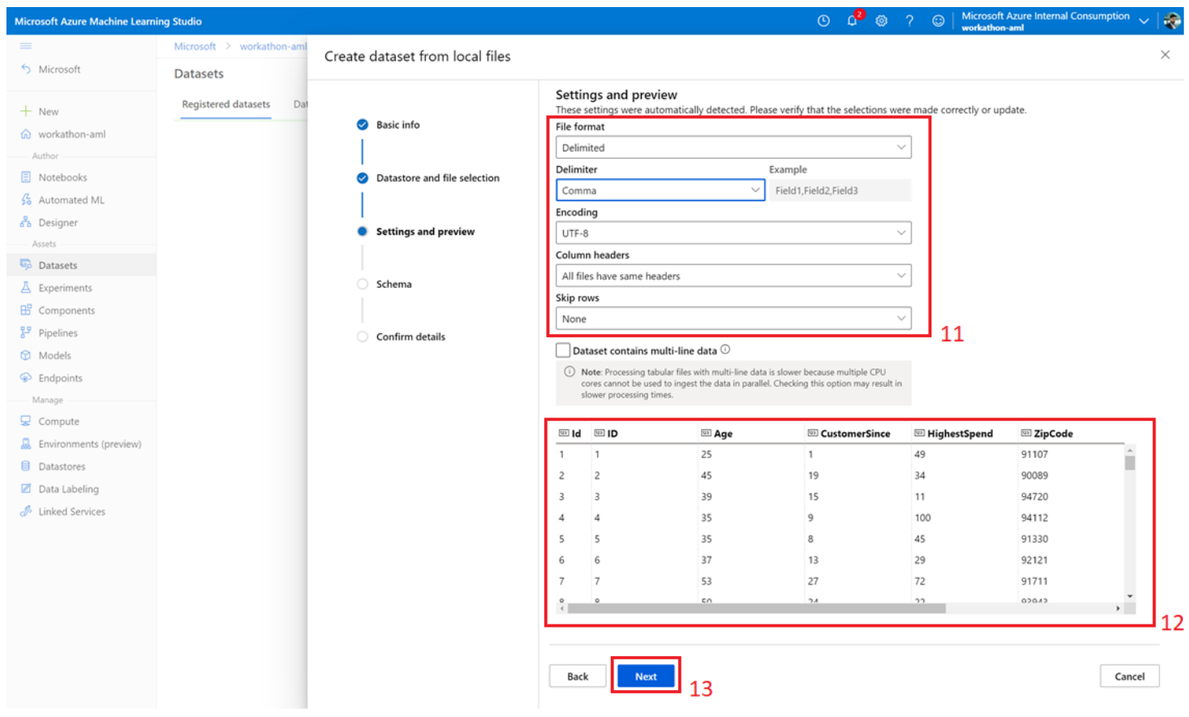
Steps to register a dataset

We will now upload 2 CSV files, each of which contain partial data for our Finance use case. We will be merging the 2 datasets when we are building the code.

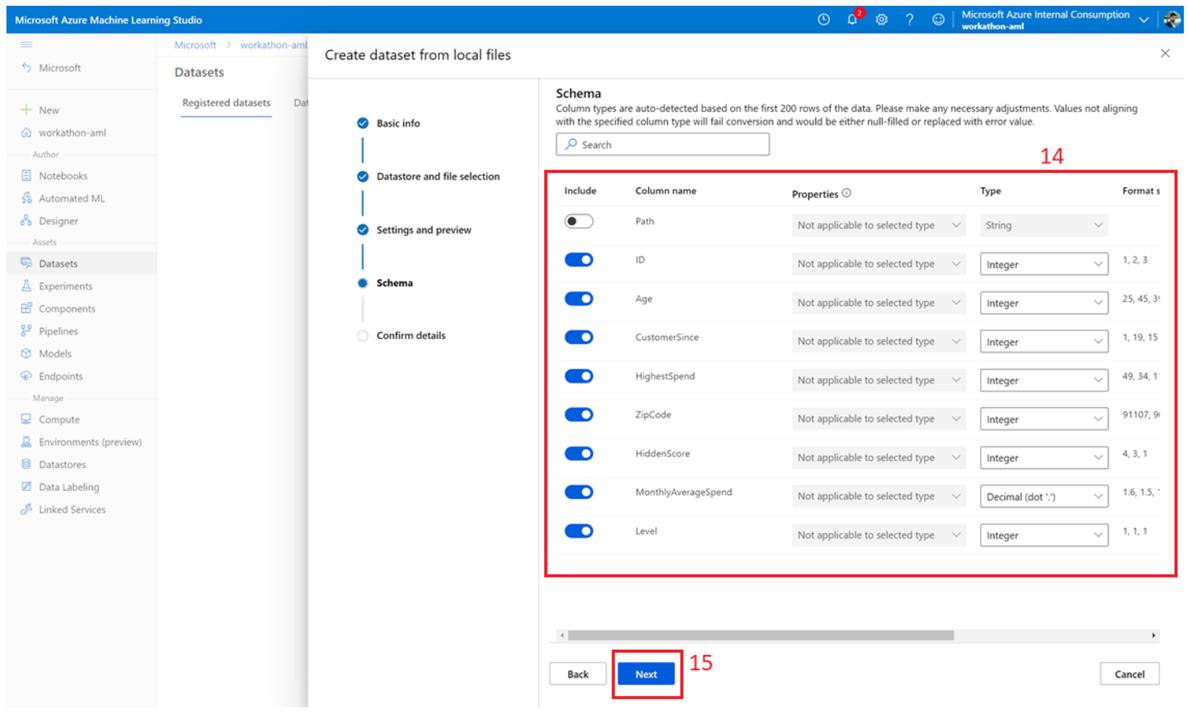
1. Go to the Datasets tab
2. Select Registered datasets
3. Click + Create dataset and select 'From local files'
4. Name : Give your dataset a unique name. This name is used to reference the dataset anytime you want to access it. We have used the name Dataset1 for the first data file.
5. Dataset type : Tabular
   [There are two dataset types, which provide different capabilities:
   A Tabular dataset represents data in a tabular format by parsing the provided file or list of files. This provides you with the ability to materialize the data into a Pandas or Spark DataFrame.
   A File dataset references a single or multiple files in your datastores or public URLs. This provides you with the ability to download or mount the files to your compute.]
6. Click Next
7. Select or create a datastore : select the default blob storage for your account
   [When you upload any new data, AML saves the data is a storage location or dataset. You can choose amongst any of the existing datastores or create a new one.]
8. Click upload and select the first data CSV file.



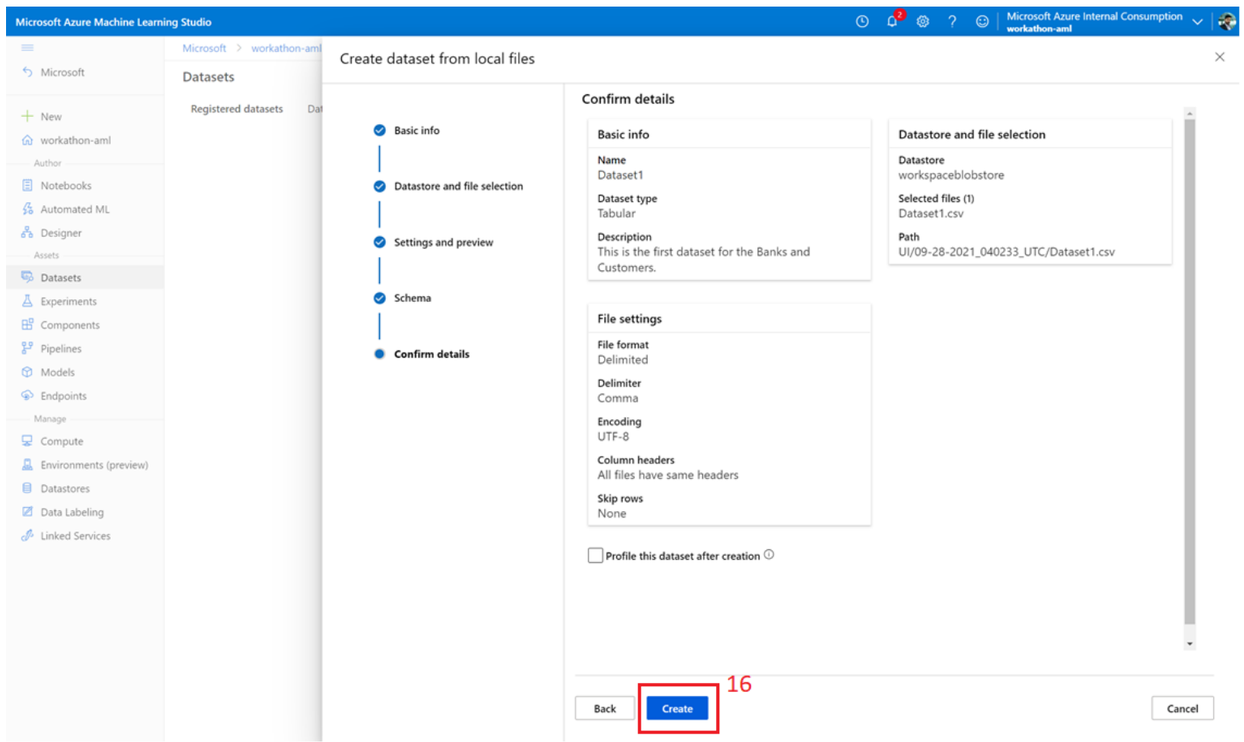
9. Once the data file is uploaded successfully, you can view it as highlighted
10. Wait while it's parsing dataset information

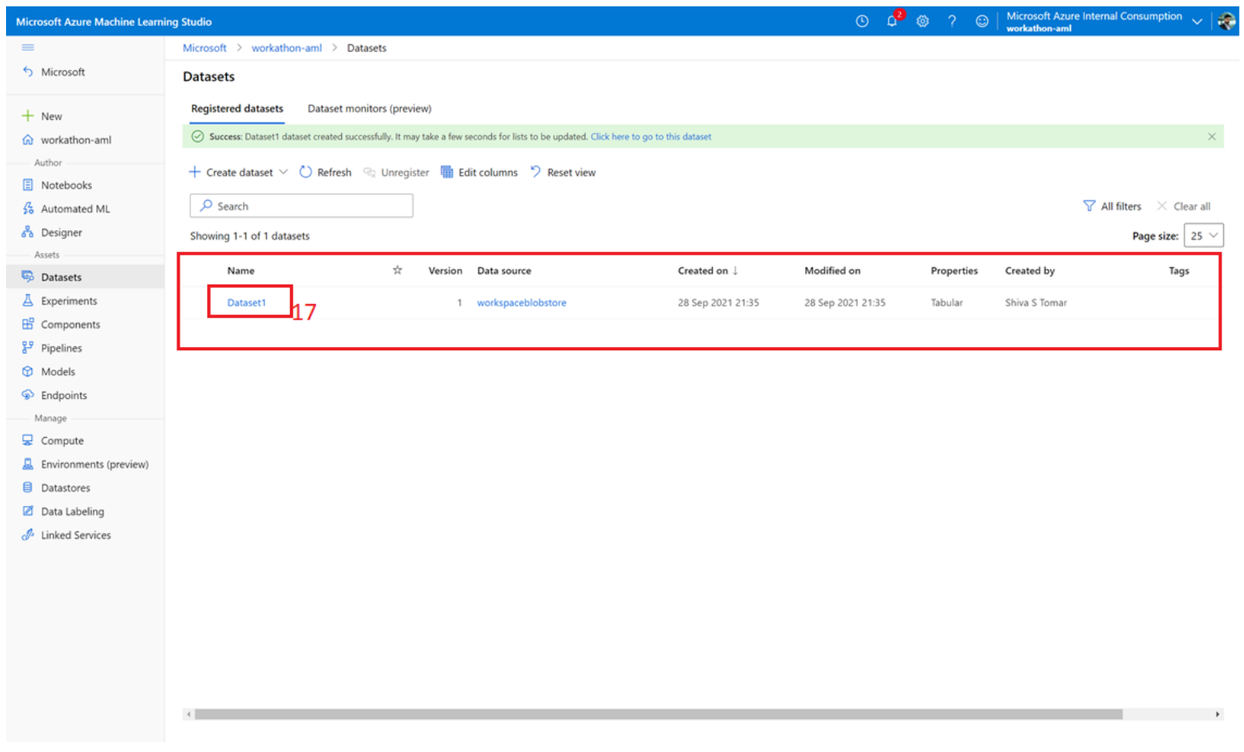


11. The details for the dataset will be populated automatically. Make sure to confirm that that the right details have been added. In case you want to change any of those, you can do so by selecting other options from the dropdown.
12. You can preview the data as highlighted
13. Click Next



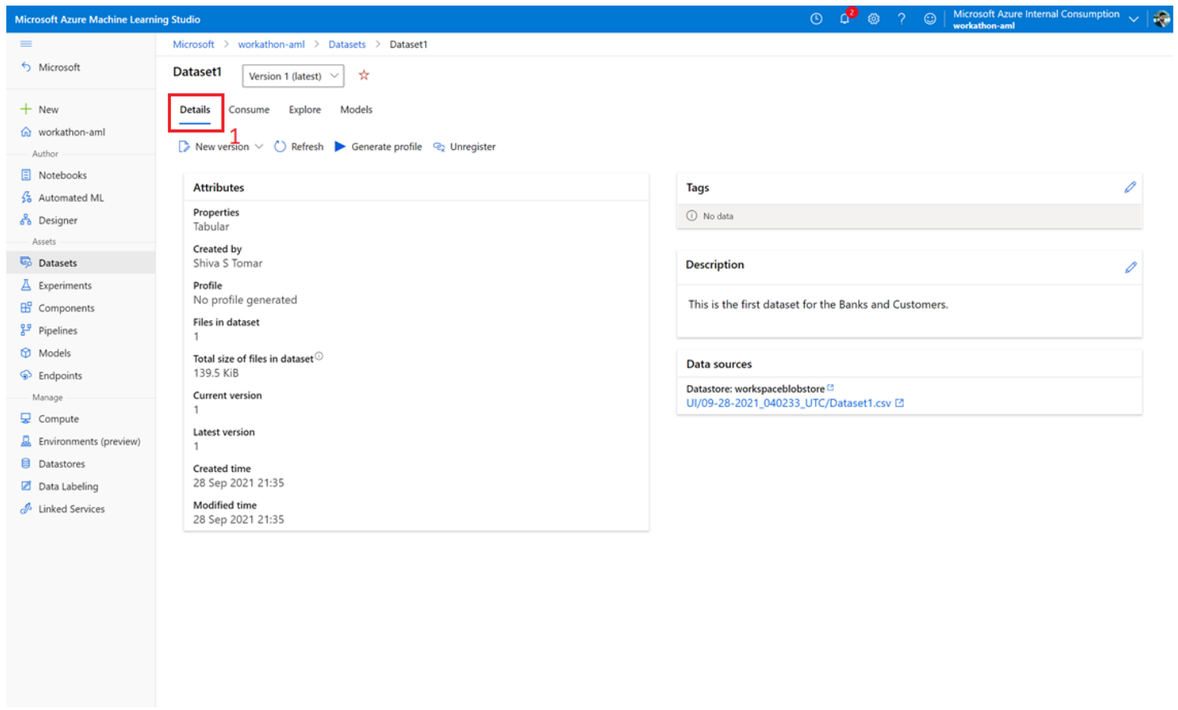
14. In the Schema page, you can choose to exclude any column you want by clicking on the toggle key for that column. Alternatively, you can also change the data type for any column if it has not been populated correctly.
15. Click Next

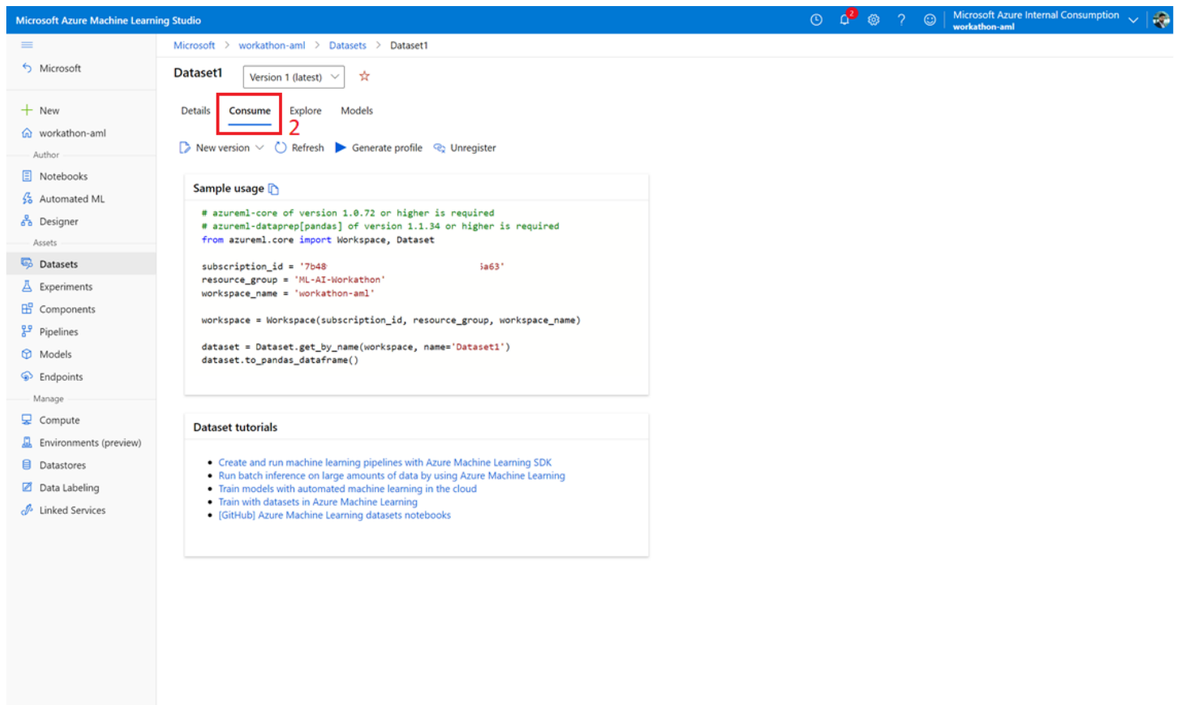


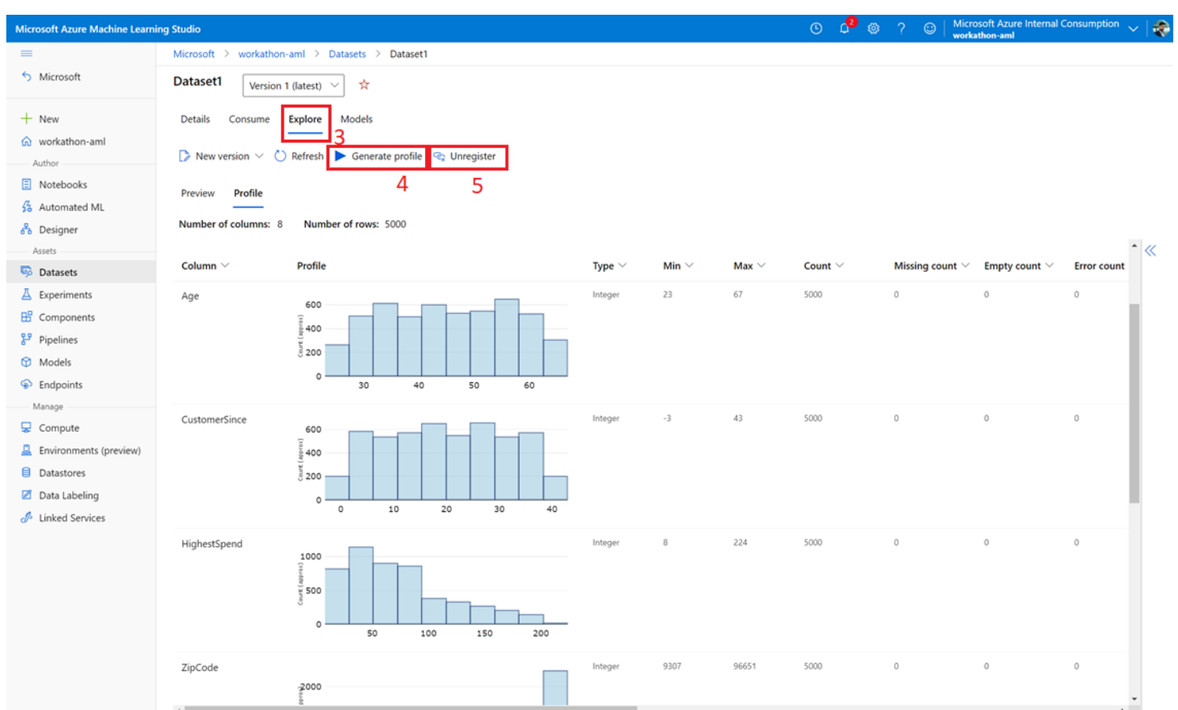


16. Verify the details and click Create.

17. Once the dataset is created, you can view it in the Datasets tab under Registered Datasets. Click on the Dataset name to explore it further.







Exploring the Dataset

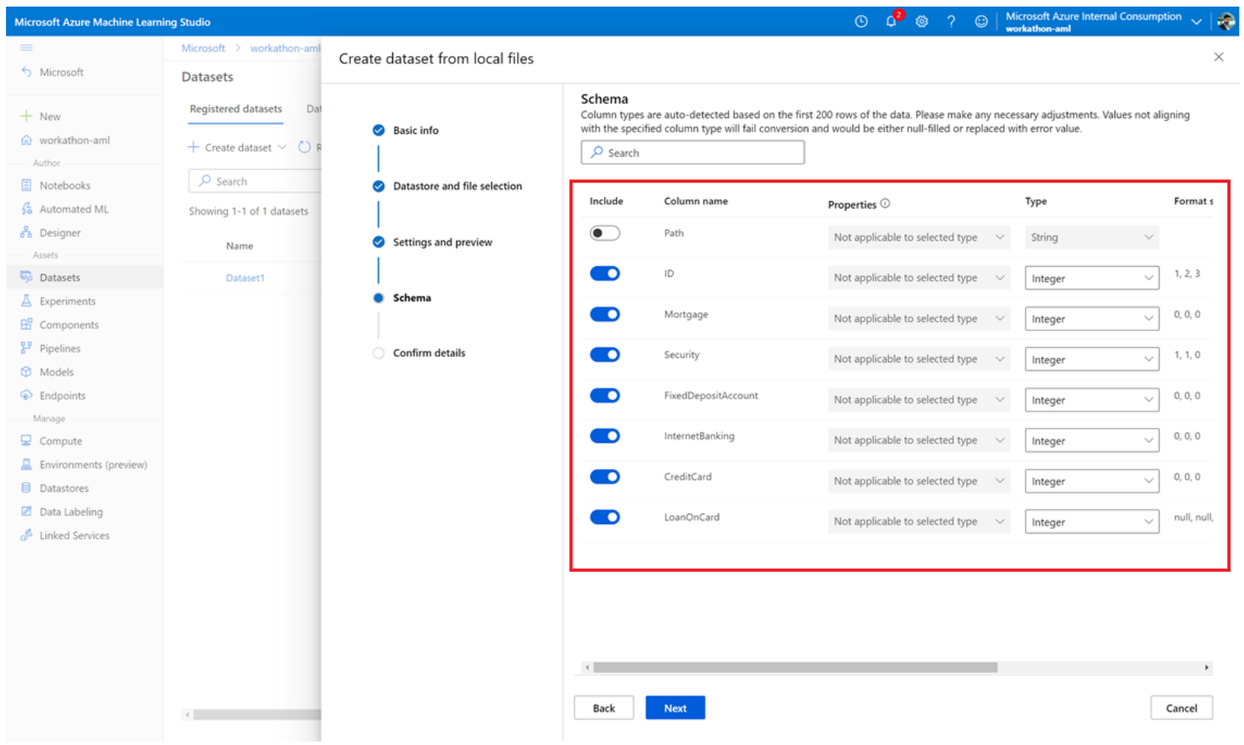
1. In the details tab, you can find the metadata such as total size, version, properties, created by, created time etc.

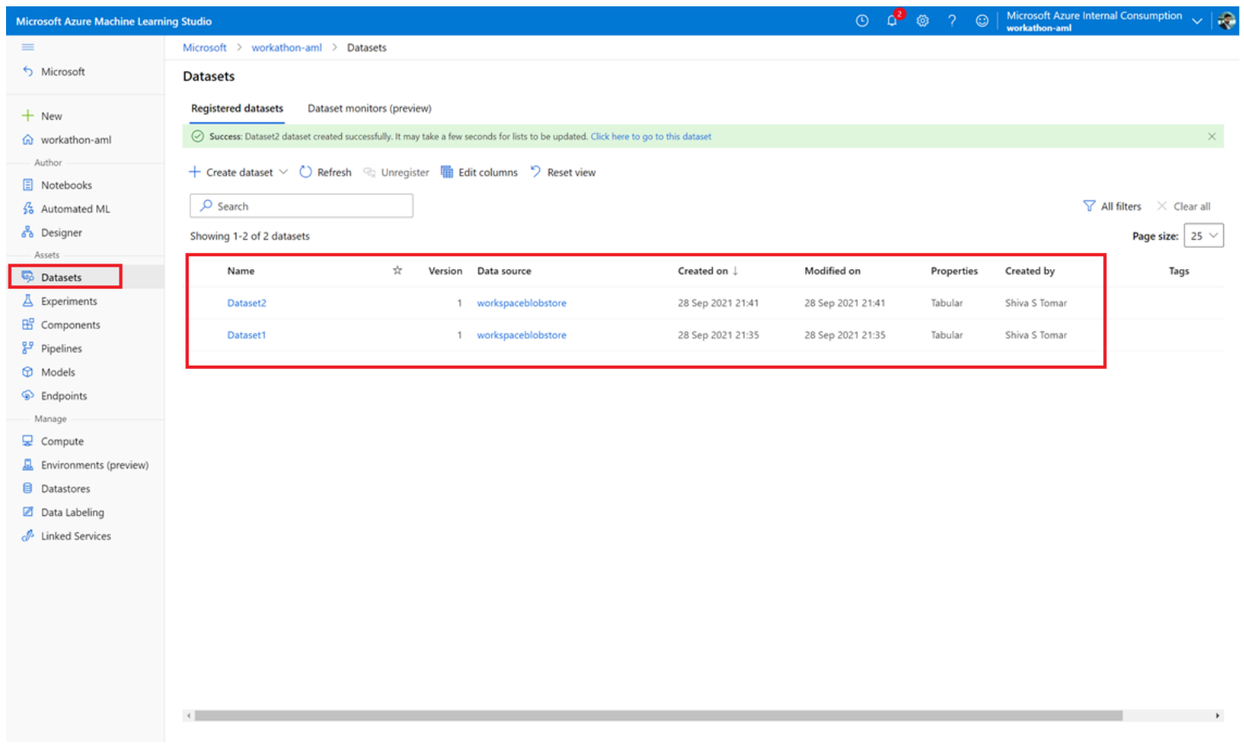
2. In the consume tab, you can find the code snippet to read the data from this dataset.

3. In the explore tab, you can view the automatically generated graphs and column statistics.

4. By default, the profile is generated only for the top 10K rows in the dataset. In case you have a larger dataset and want to view the profile based on all the rows, you can click ‘Generate profile’. This will ask you for a compute target and will create an experiment to generate statistics based on the complete dataset.

5. You can unregister (aka delete) the dataset anytime you feel it is not required anymore. [Do not click this option for now!]





<u>Register the second dataset</u>

Similarly, register the 2nd CSV file (Dataset2).
Refer the screenshot to confirm the Schema once you have uploaded the file.

Once you have registered both datasets, you will get a view similar in the Datasets tab.

There are 2 approaches to move forward with the workshop :

1. Build your code from scratch – If you are familiar with Python & data science, you can build your own code from scratch using the hints and expected output that we have shared.
2. Upload a pre-built notebook – If you are not so familiar with Python or data science but want to explore Azure Machine Learning functionality you can use our pre-built notebook (Banking Scenario.ipynb).

<u>Model development</u>

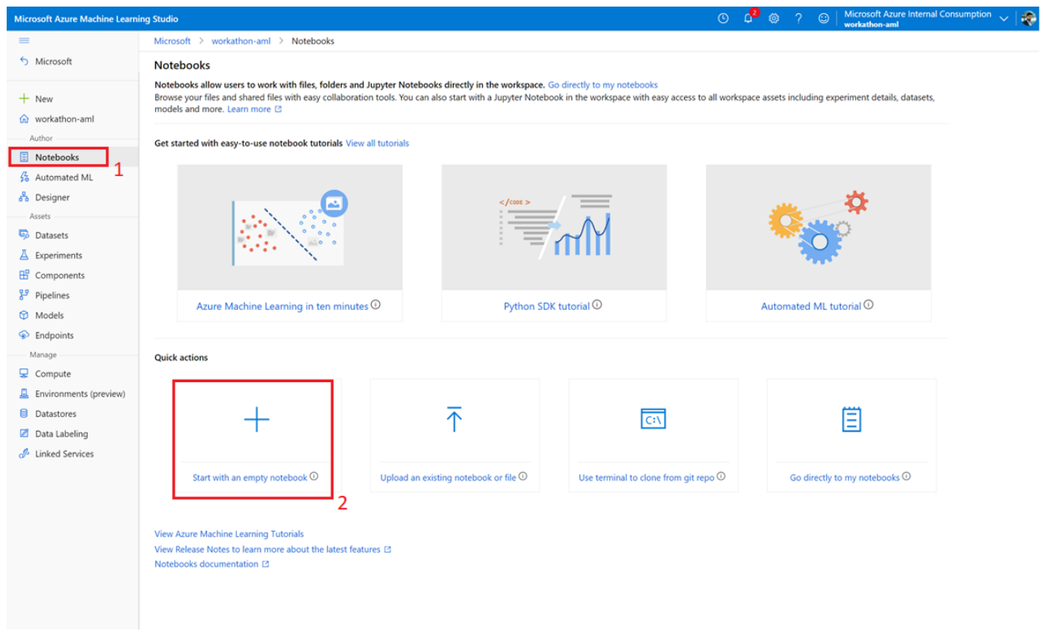
To develop a Machine Learning model, you can write code in the language of your choice such as Python, R etc. In this workshop, we will be developing our model using Python.

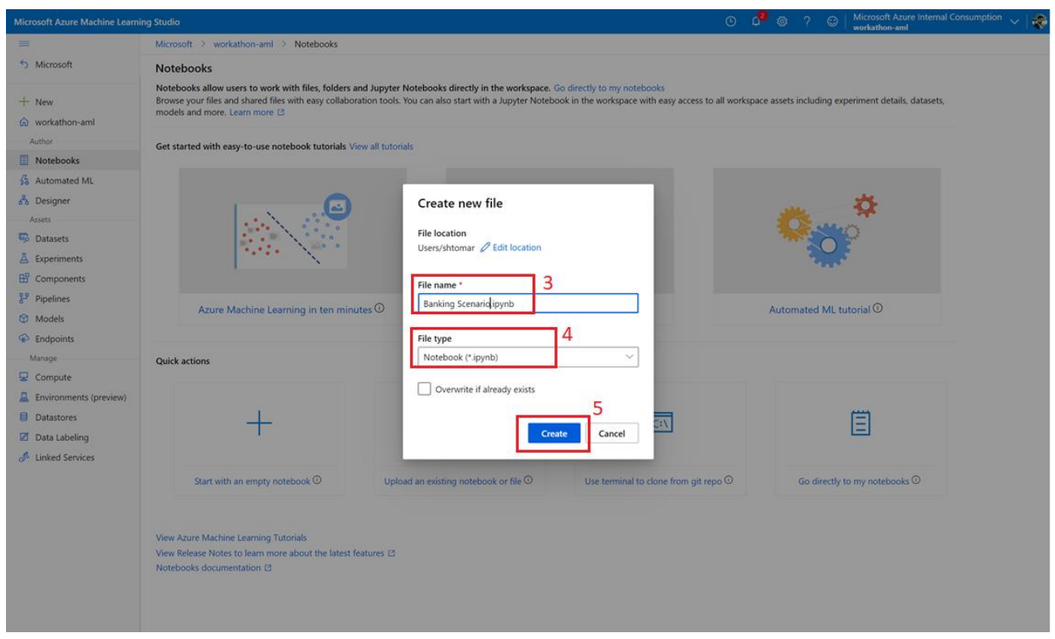
There are 2 major platforms you can use to build the code for your models :

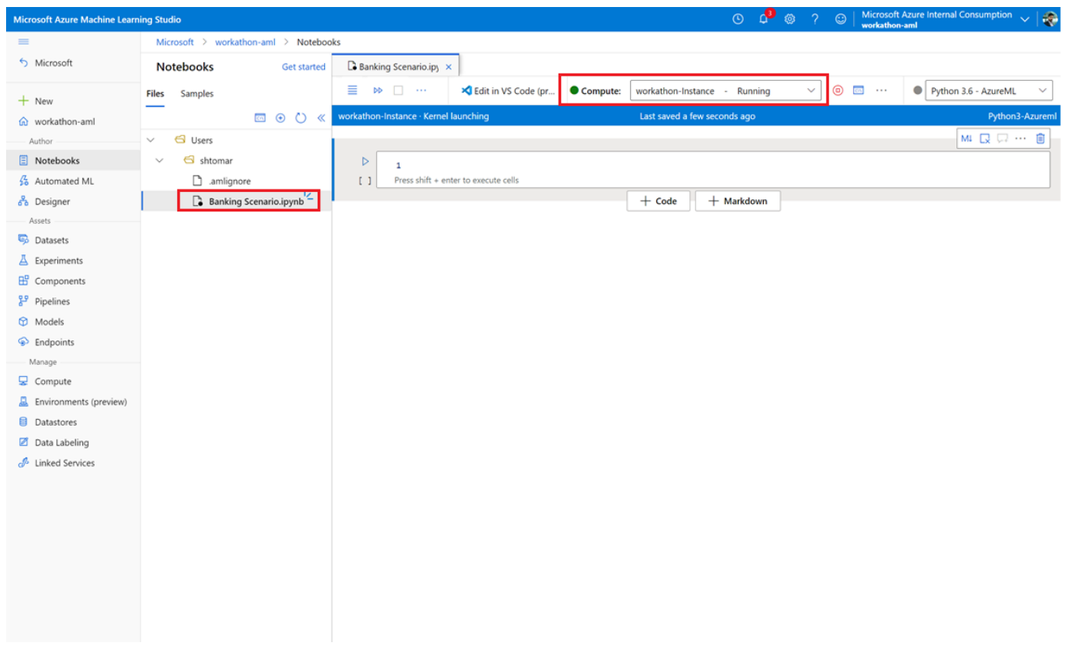
1. AML Notebooks : This is the native notebook functionality of AML that allows you to write code in an interactive platform. You can create python files & notebooks, R files, text script or Bash scripts.
2. Integrated Applications : You can access developer applications like JupyterLab, Jupyter notebooks, VS Code etc from within AML Studio, if you want to use the familiar experiences.

Alternatively, you can also create custom models using AML Designer. This is a no-code / low-code functionality of AML that allows you to build models using a drag & drop interface and pre-built modules or include code snippets if you want to. We will be discussing the Designer approach going forward.

Approach 1 : Build your code from scratch







Create a New Notebook

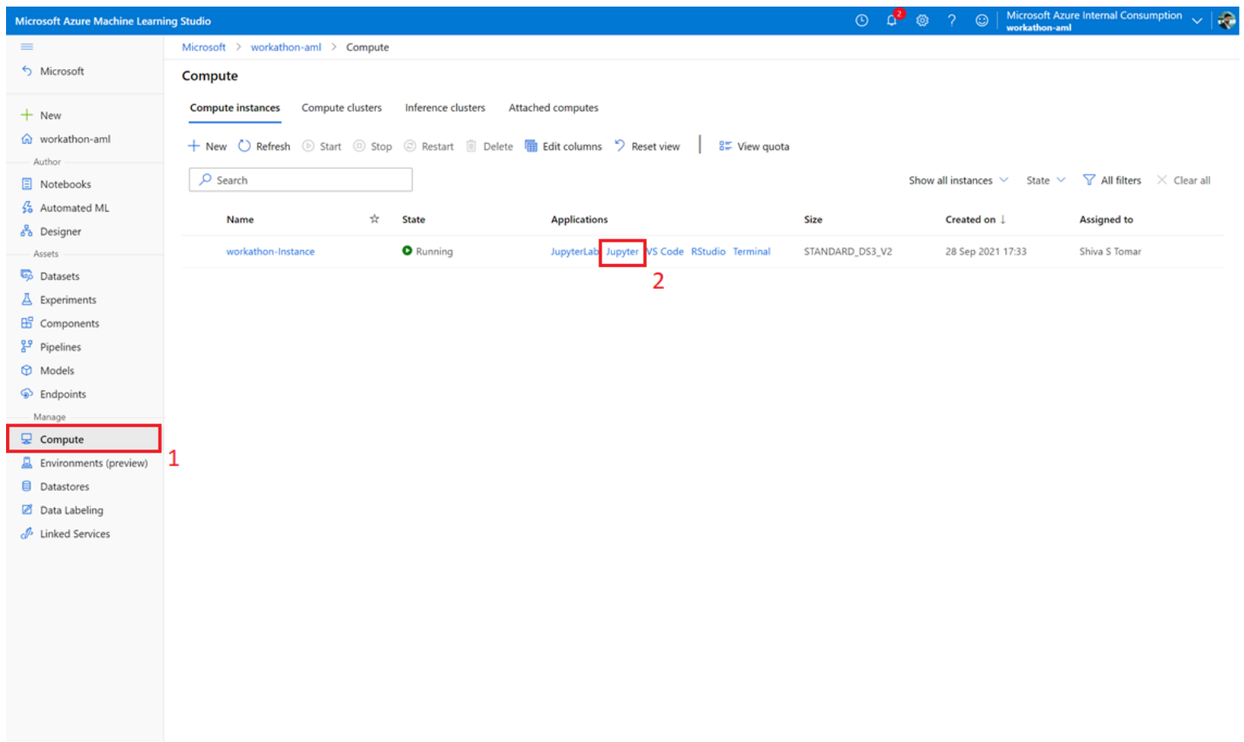
AML Notebooks and Jupyter notebooks have the same functionalities, however, in this lab we will be using Jupyter notebooks to build our models, since most developers might already be familiar with the Jupyter notebooks interface.

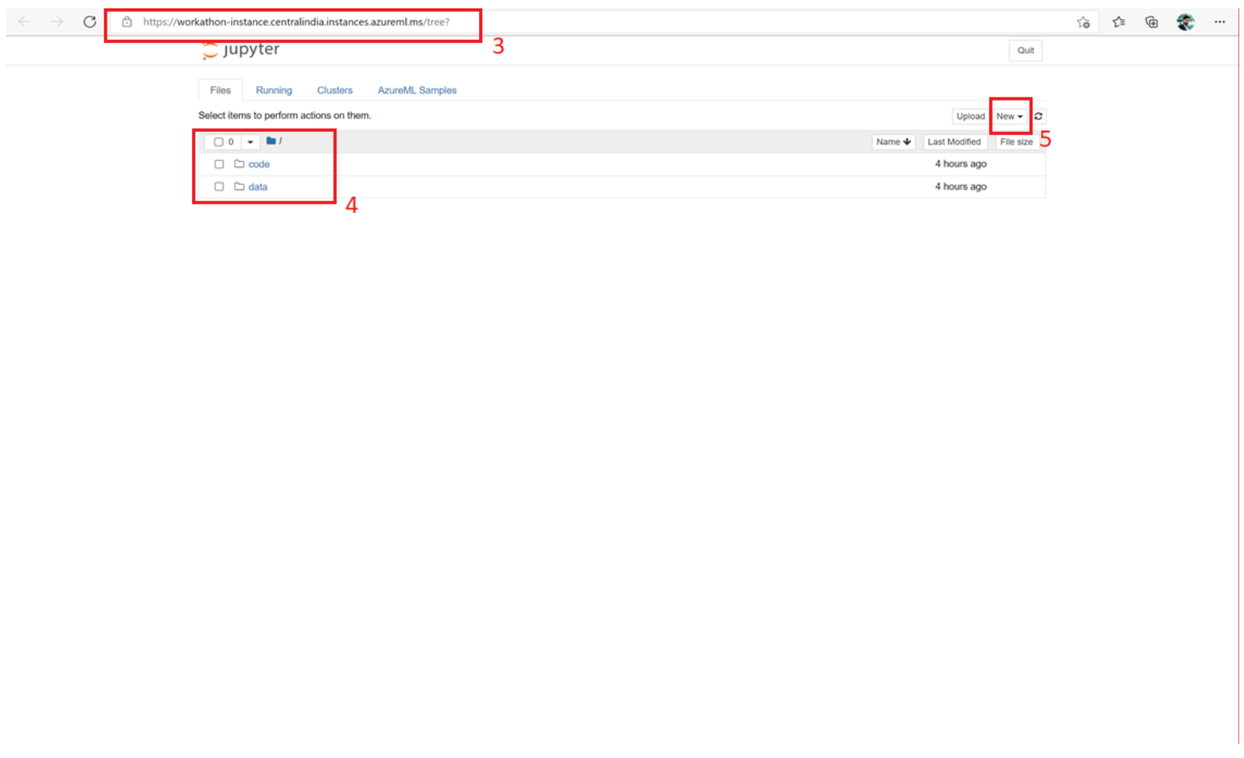
In case you want to create your models using the native AML notebooks, you can follow the steps as highlighted to create a new notebook :

1. Go to the Notebooks tab
2. Select 'Start with an empty notebook'

3. File Name : Provide your notebook a unique name
4. File Type : Select Notebook(*.ipynb)
5. Click Create

Once you create a new notebook, attach it to the Compute Instance you created earlier in case there are multiple instances existing in your environment.
You can now start building your code.





To get started with **model development on Jupyter Notebook**, follow the steps as highlighted:

1. Go to the Compute tab
2. Select Jupyter from the Application options

3. This opens a Jupyter instance that is hosted on your Compute instance
4. You have a folder hierarchy already created for you in terms of data & code folders. You can leverage this if you want or can define how you want to create and manage your notebooks, data & other artefacts.
5. Click New and create a 'Python 3.6 -AzureML' Kernel.
   [This Kernel comes pre-installed with all the AML libraries, thus saving you the overhead to install any of the libraries. You can get started by importing the required libraries]

Hints for building & deploying the model

Build the Model

1. Read Scenario, Information about the Data & Project Objective from the top of this Workshop Guide.

2. Import Libraries and Packages
   Hint:
   Use libraries like – numpy, pandas, matplotlib, seaborn, scipy, sklearn, statsmodel.api
   Note:
   you can also use a library of your choice to accomplish this task.

3. Import and Merge Data
   Hint:
   Read data sources and remove duplicates before merging
   Use merge function

   Expected output:

Out[7]:

	ID	Age	CustomerSince	HighestSpend	ZipCode	HiddenScore	MonthlyAverageSpend	Level	Mortgage	Security	FixedDepositAccount	InternetBanking
0	1	25	1	49	91107	4	1.6	1	0	1	0	0
1	2	45	19	34	90089	3	1.5	1	0	1	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0

4. a. Data Cleansing:
   Hint:
   Check for the nulls, NA and negative values.
   Do describe on data

   Expected outcome:

Out[14]:

	count	mean	std	min	25%	50%	75%	max
ID	5000.0	2500.500000	1443.520003	1.0	1250.75	2500.5	3750.25	5000.0
Age	5000.0	45.338400	11.463166	23.0	35.00	45.0	55.00	67.0
CustomerSince	5000.0	20.104600	11.467954	-3.0	10.00	20.0	30.00	43.0
HighestSpend	5000.0	73.774200	46.033729	8.0	39.00	64.0	98.00	224.0
ZipCode	5000.0	93152.503000	2121.852197	9307.0	91911.00	93437.0	94608.00	96651.0
HiddenScore	5000.0	2.396400	1.147663	1.0	1.00	2.0	3.00	4.0
MonthlyAverageSpend	5000.0	1.937938	1.747659	0.0	0.70	1.5	2.50	10.0
Level	5000.0	1.881000	0.839869	1.0	1.00	2.0	3.00	3.0
Mortgage	5000.0	56.498800	101.713802	0.0	0.00	0.0	101.00	635.0
Security	5000.0	0.104400	0.305809	0.0	0.00	0.0	0.00	1.0
FixedDepositAccount	5000.0	0.060400	0.238250	0.0	0.00	0.0	0.00	1.0
InternetBanking	5000.0	0.596800	0.490589	0.0	0.00	1.0	1.00	1.0
CreditCard	5000.0	0.294000	0.455637	0.0	0.00	0.0	1.00	1.0
LoanOnCard	4980.0	0.096386	0.295149	0.0	0.00	0.0	0.00	1.0

b. Drop insignificant columns and rows with null values

Expected outcome:

Out[24]:

	Age	CustomerSince	HighestSpend	HiddenScore	MonthlyAverageSpend	Level	Mortgage	Security	FixedDepositAccount	InternetBanking	CreditCar
9	34	9	180	1	8.9	3	0	0	0	0	
10	65	39	105	4	2.4	3	0	0	0	0	
11	29	5	45	3	0.1	2	0	0	0	1	
12	48	23	114	2	3.8	3	0	1	0	0	
13	59	32	40	4	2.5	2	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	
4995	29	3	40	1	1.9	3	0	0	0	1	
4996	30	4	15	4	0.4	1	85	0	0	1	
4997	63	39	24	2	0.3	3	0	0	0	0	
4998	65	40	49	3	0.5	2	0	0	0	1	
4999	28	4	83	3	0.8	1	0	0	0	1	

4980 rows × 12 columns

5. Data analysis & visualization
   Hint:
   Use techniques like – univariate, bivariate & multi-variate analysis to explore and visualise the data.

6. Data pre-processing for Modelling
   Hint:
   Treat Imbalanced Data – which means check for skewness of target variable (LoanonCard).
   Use resample function from sklearn library.

7. Model Preparation
   Split the data into train and test

8. Use various algorithms to check for the best outcome
   In this workshop we will be using the following algorithms
   - Apply Logistic Regression with lbfgs solver.
   - Apply Logistic Regression with linear solver.
   - Naive Bayes Classifier
   - Gaussian Naive Bayes

   Note: You can also use any other Algorithm for building your model like SVM, Random Forest, Decision Tree etc.
   Check for metrics like Precision, Recall, F1-Score and Accuracy to decide the best model.

9. Draw you conclusion based on the best outcomes from the models you ran

Note : Once you have trained your model using the above algorithms, pick the model that performed best. We will now be deploying this model as a real time endpoint.
In our case, we deployed the Logistic Regression model with linear solver.

<u>Deploy the model</u>

10. Define, Register and View the AML Environment.
    Hint:
    Use this link to accomplish this task.

11. Prepare the training script
    Hint:
    Use this link to prepare your training script.

12. Run an experiment on remote compute.
    Hint:
    Use this and this link to create your script for the remote compute and running the model.

13. Register the model
    Hint:
    Use this link.

14. Create the scoring script which will be deployed on the compute cluster.
    Hint: Use this link.

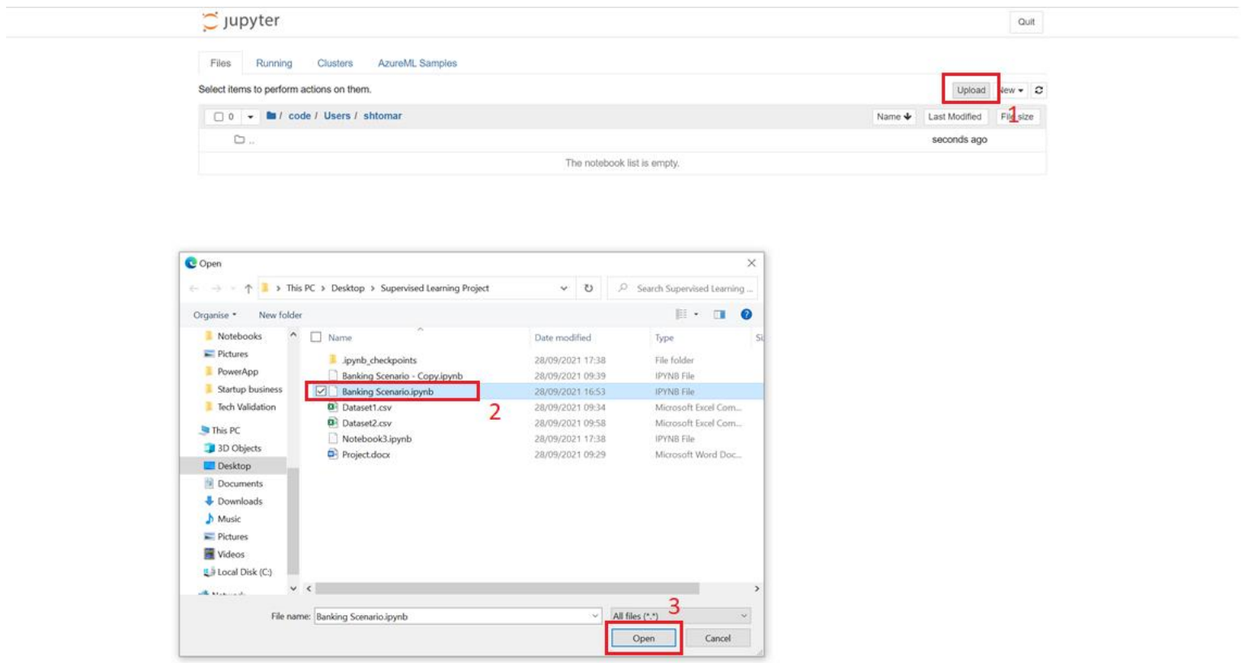
15. Configure scoring environment and Deploy
    Hint: Use this link.

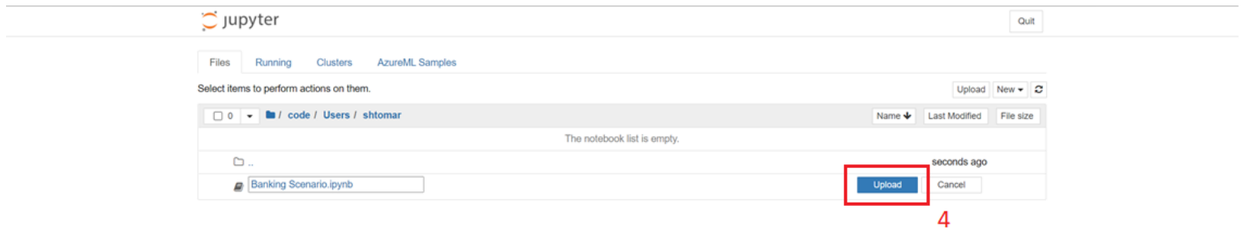
16. Call your web Service to test the model.
    Hint: Use this link.

Thank you!

If you were able to accomplish these steps, you can skip the ‘Approach 2 - Upload a pre-built notebook’ section and jump to the ‘Exploration’ section.
In case you want to leverage the code we have written, you can upload the notebook we built and run the cells to view the results. We recommend you read through the code carefully, to understand how model training and deployment works in Azure Machine Learning. Once you run the notebook, go to the ‘Exploration’ section.

Approach 2 - Upload a pre-built notebook

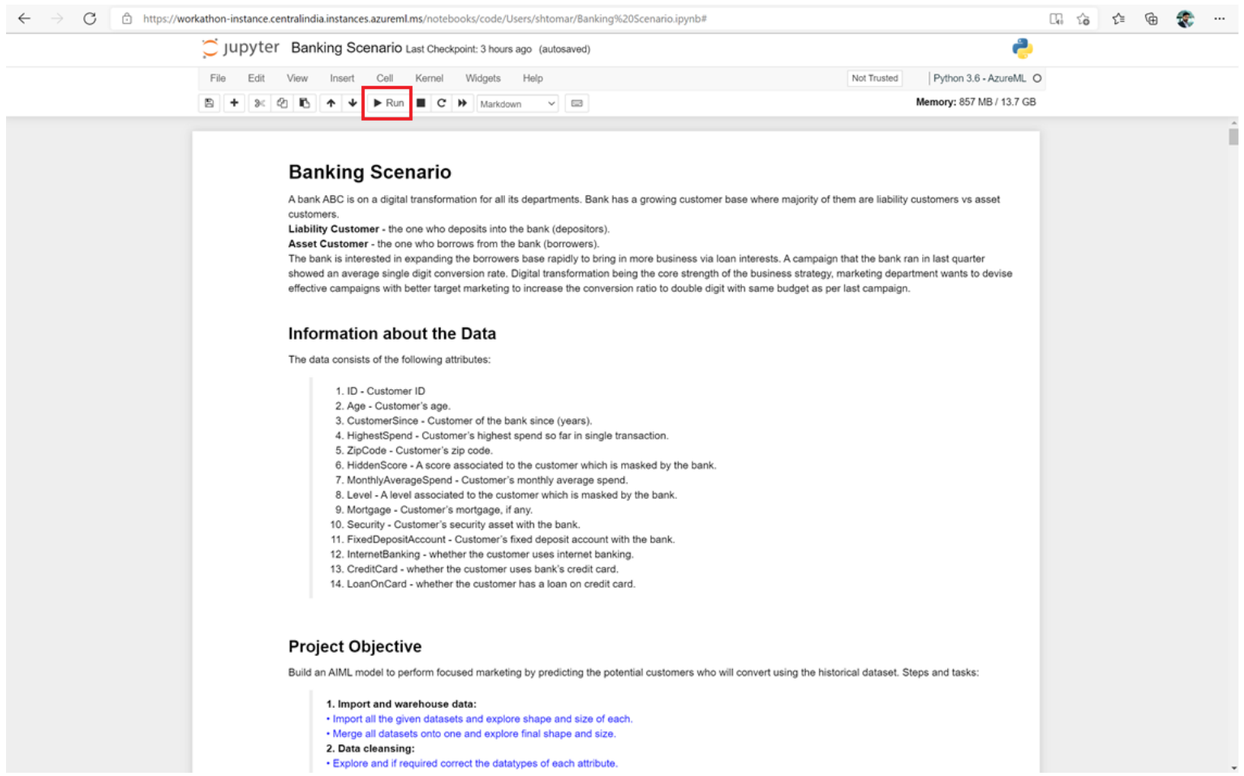




Upload the notebook

1. In the Jupyter Notebook, click on the Upload button
2. Browse to the location where you have saved the notebook
3. Click Open

4. Click on the Upload button to finally upload the notebook



Run the notebook

When you open the uploaded notebook, you will get such a view.

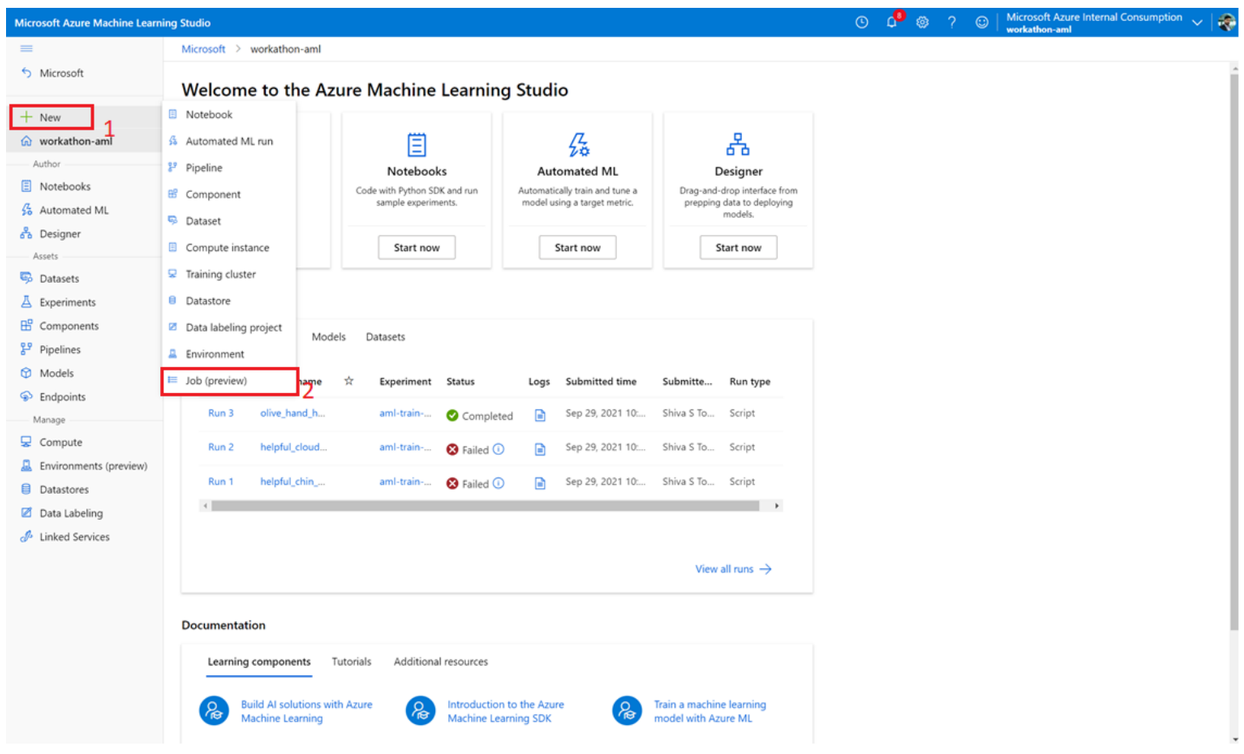
Observe the code snippets in each cell of the notebook, run the cells one by one and observe the outcome accordingly.
You can run each individual cell be clicking the 'run' button as highlighted or press 'Shift' + 'Enter'. These commands will run the cell that is currently selected.

Once you have run all the cells of our notebook successfully, you will see some content/artifacts generated in the AML environment.
These can be observed both through the code-based approach and GUI based approach in ML Studio.

We will explore all these artefacts one by one in the Azure ML Studio.

Exploration

In this section, we will help you explore various features of AML and define them for you to better understand how you can leverage them best, basis the objective that you are trying to achieve.



New

The new tab allows you to create and provision a variety of AML resources like Notebooks, AutoML experiments, Datasets, Compute etc.
You can also create these resources from their individual tabs.

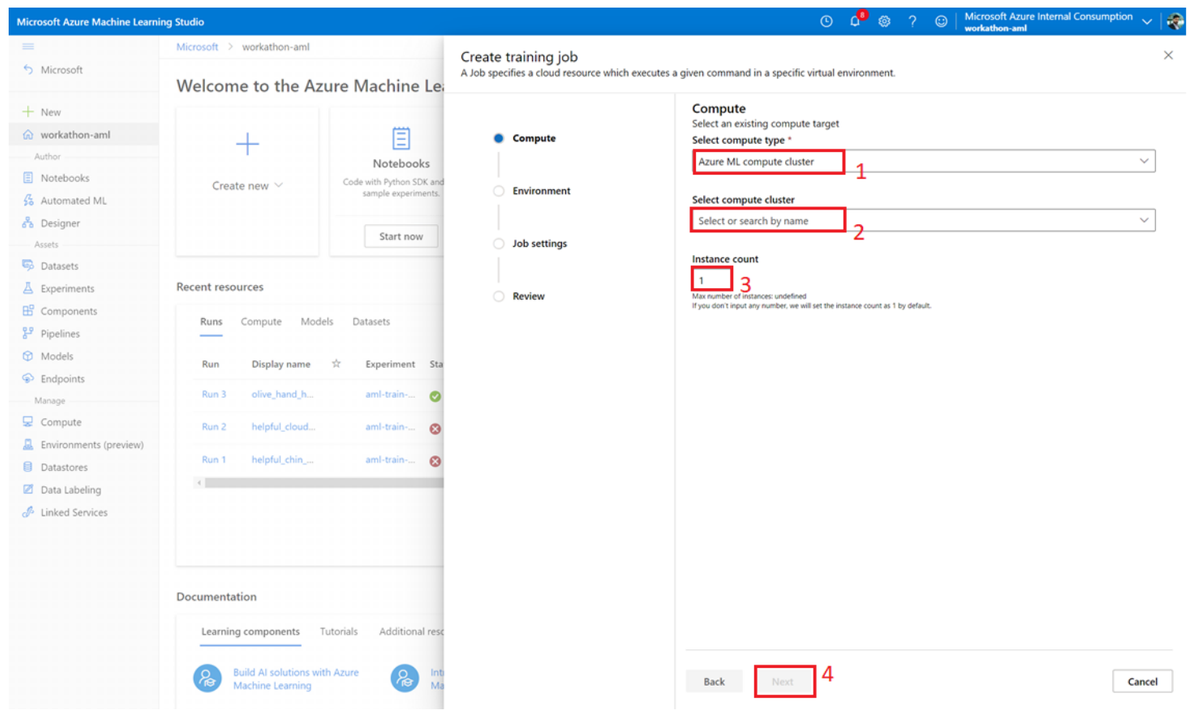
We will be discussing all the artefacts in their individual resource tabs. Here, let's discuss a new feature to create a Job (this feature is in preview)

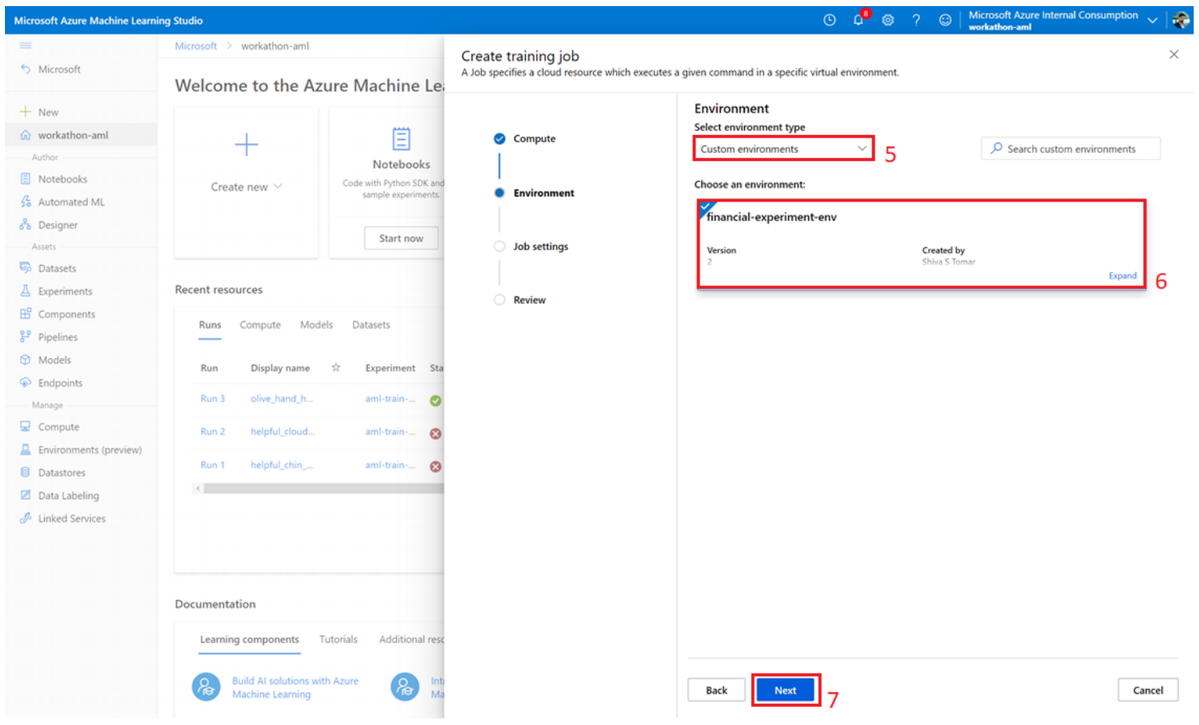
Creating Jobs

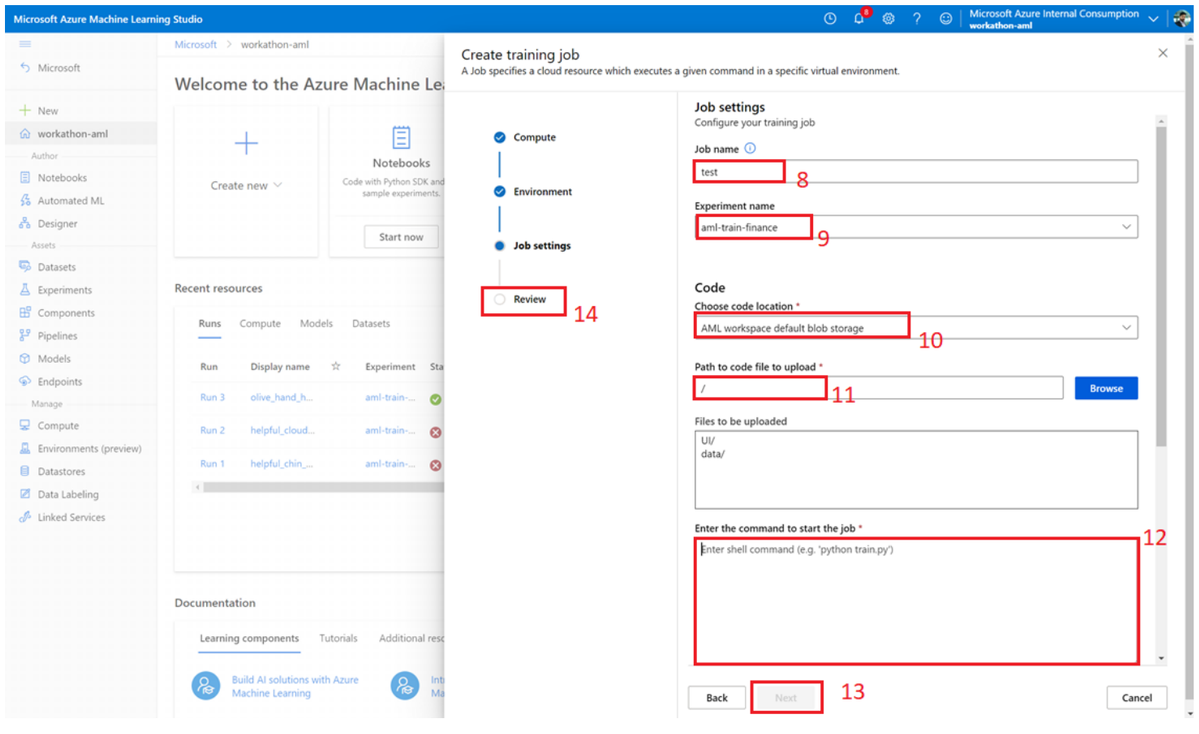
Creating a job allows you to execute a given command or script in a specific virtual environment and cloud resource.
You can use the UI to directly create a training job.

Note : We will not be creating a job here but let's explore the high-level steps to create a job:

1. Click the + New Tab
2. Go to Job (preview)







Configuring training job parameters

Compute

1. **Compute Type** : This defines the compute target on which the job will run. You can choose out of the following compute targets :
   a. Compute cluster
   b. Compute instance
   c. Kubernetes cluster
   You can select the target basis the workload you want to run. We have chosen the Compute cluster we created earlier.

2. **Compute cluster** : From the dropdown, select the name of the compute cluster.

3. **Instance count** : For a compute cluster or a Kubernetes cluster, you may also specify how many nodes you want for the job in Instance count. The default number of instances is 1.

4. Click Next

Environment

5. **Environment type** : Here you can choose between a curated environment (pre-built), custom environment like the one we created in the workshop or a Container registry image.

6. **Environment** : Basis the environment type you choose, you will be given the available environment options. We chose ‘Custom environments’ hence have just 1 option to choose from.

7. Click Next

Job settings

8. **Job name** : Give your job a unique name

9. **Experiment Name** : This helps you to organize each job’s run record under the corresponding experiment in the studio's ‘Experiment’ tab. By default, Azure will put the job in the Default experiment.

Code

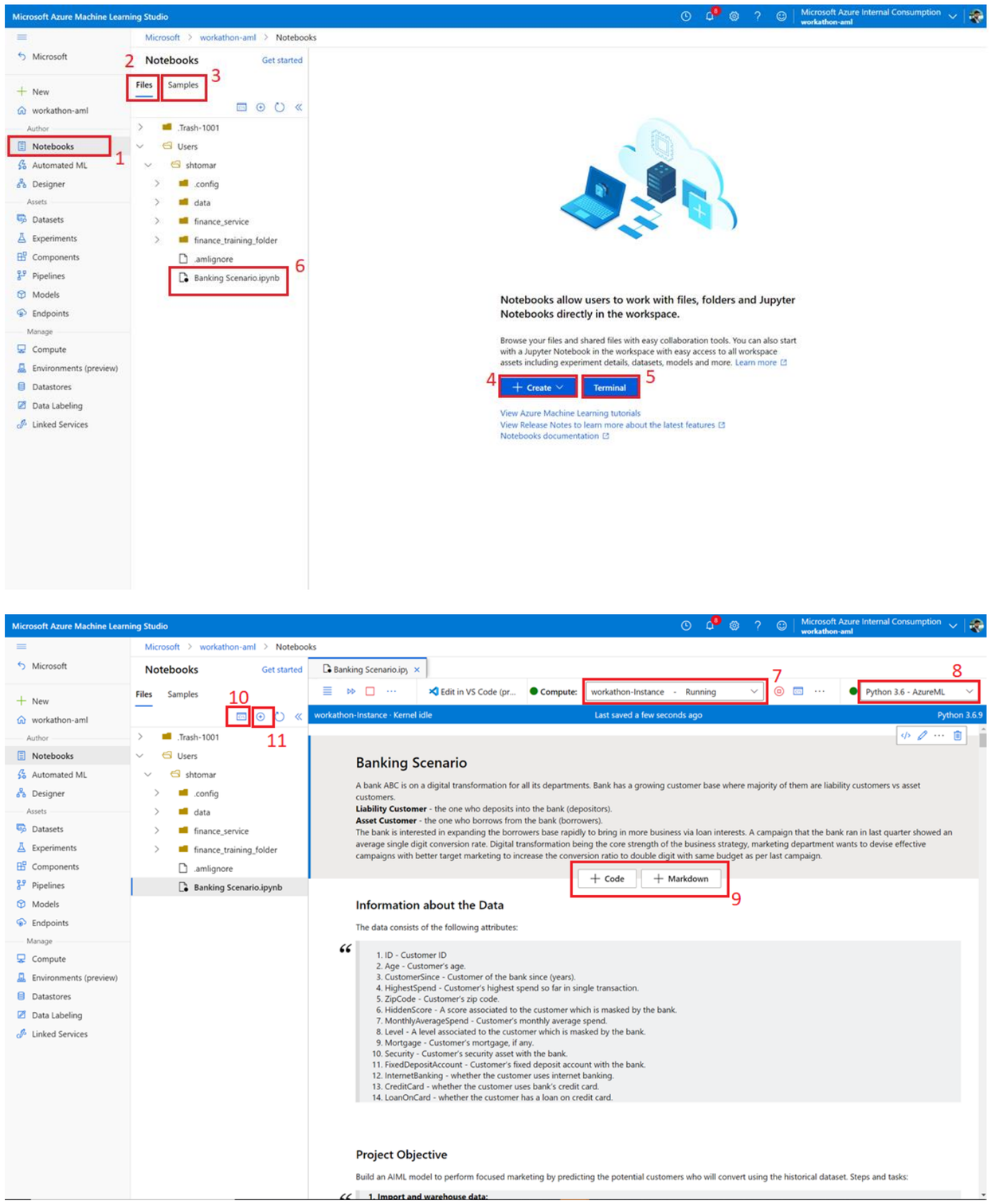
10. **Code location** : You can upload a code file or a folder from your machine or upload a code file from the workspace's default blob storage. For eg : The location for the python training script file.

11. **Code Path** : You can also browse to the specific path in the location where your file is kept. Azure will show the files to be uploaded after you make the selection.

12. **Command** : Enter command to execute. Command-line arguments can be explicitly written into the command or inferred from other sections, for example : ‘python train.py’

13. Once you have entered all the settings, hit Next.

14. Review the job parameters and run the job.



Explore the Notebook

In this workshop, we learnt 2 major approaches to create notebooks –

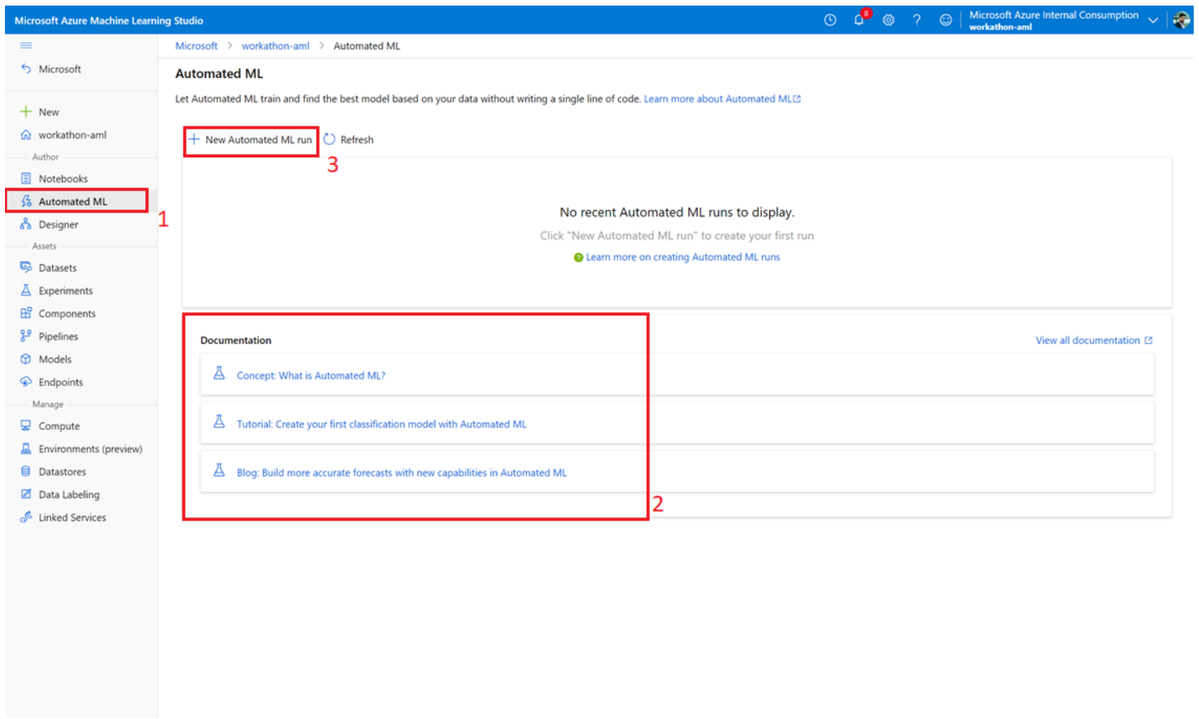
a. AML Notebooks
b. Integrated Applications like Jupyter

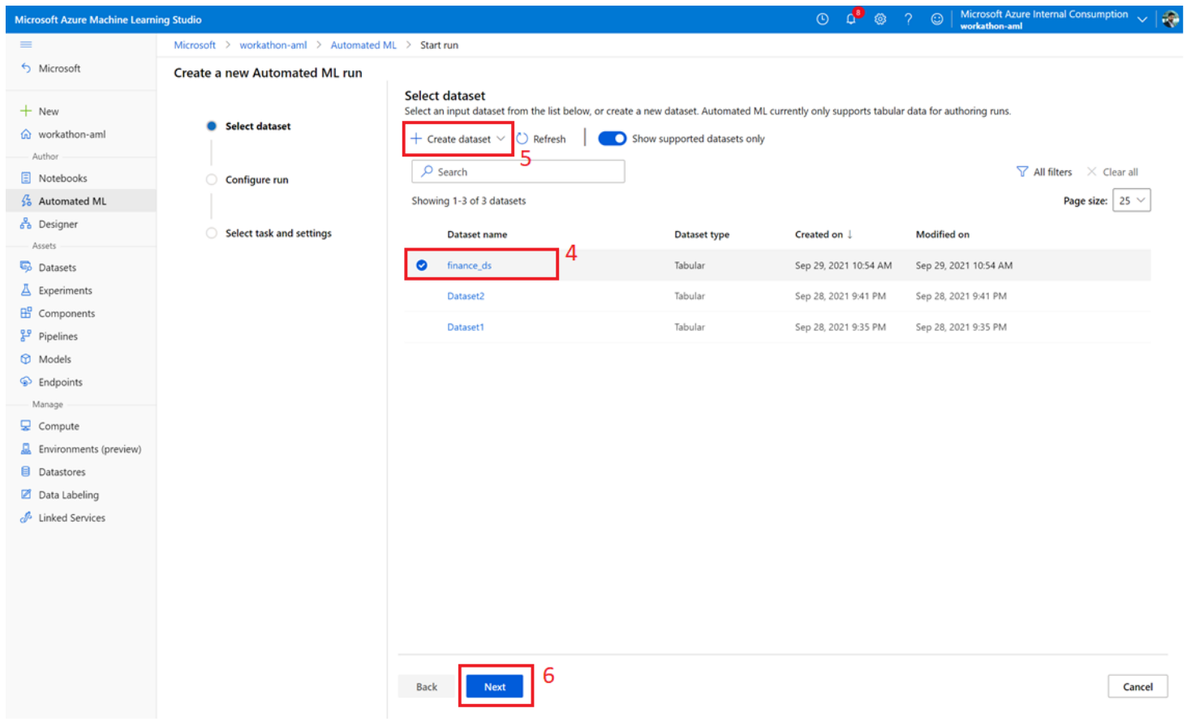
AML uses the same underlying default blob storage container to save the notebooks and artefacts created using any of the above approaches.

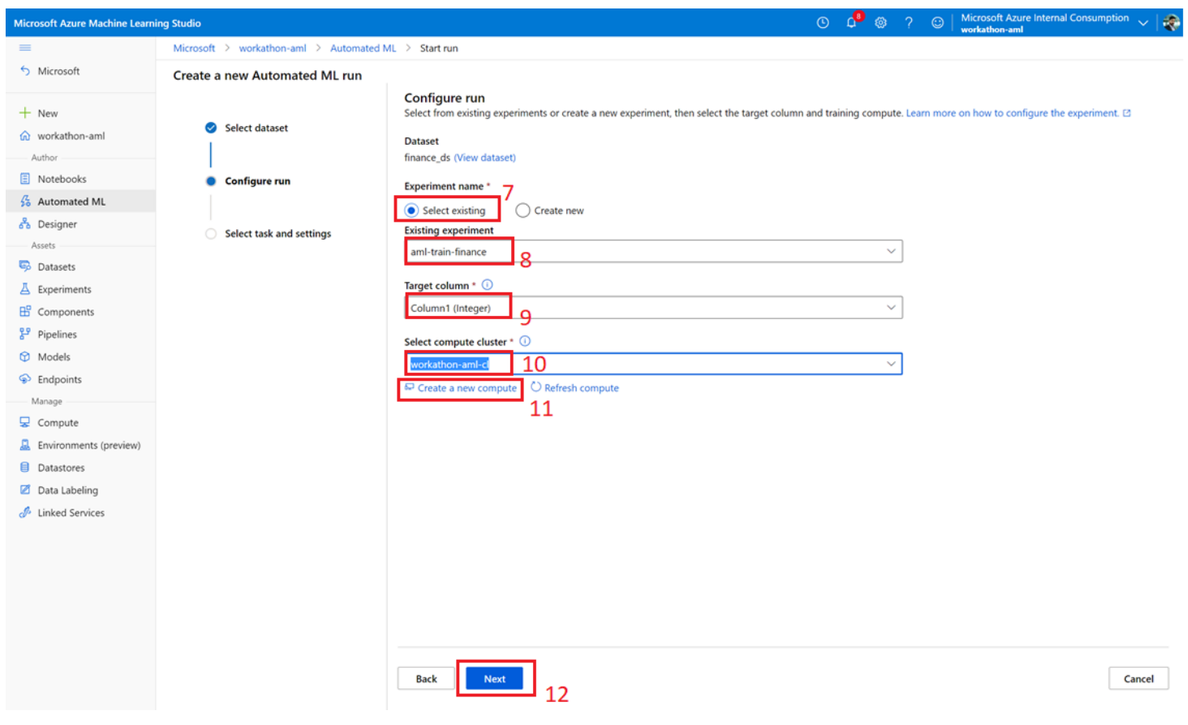
In this workshop, we created our Notebook using Jupyter Notebooks, so you can always access it by launching the Jupyter Experience from the Compute Instance page, the way we did earlier.
However, you can also view the same notebook in the native AML Notebooks experience and build on it right here if you want to.

Let's explore this a little further.

1. Select the Notebooks tab
2. Select the **files** tab : this tab allows you to view the notebooks, datasets and other artefacts that you might have created any of the 2 approaches. By default, a folder is created for each user under the 'Users' folder.
3. **Samples** : this tab contains pre-built notebooks demonstrating different capabilities of AML and industry use cases. You can leverage this for learning purposes or to reference advanced capabilities in a production scenario.
4. **+ Create** Option : This allows you to create new files & folders or upload files & folders from your local machine. These include the following file types : .ipynb, .py, .r, .txt, .sh.
   You can upload other formats as well, however, you won't be able to preview them in this experience.
5. **Terminal** : This tab allows you to execute any commands that you can execute in a traditional terminal experience. AML have native integration with Terminal capabilities.
6. Select the Banking Scenario notebook that you created
7. **Compute** : Each notebook will require a compute instance to run on. In case you have multiple compute instances spun up, choose the one on which you want to run the notebook.
8. **Kernel** : You can choose between the R & Python Kernels as appropriate.
9. As in a traditional notebook experience, you can add new Code & Markdown cells
10. **Terminal** : This also allows you to access the Terminal functionality
11. You can create new file & folders or upload files & folders using this option as well







Explore Automated ML

Though we will not be creating an AutoML experiment in this workshop (This is given as a homework exercise), we want to walk you through the AutoML experience to help you get started.

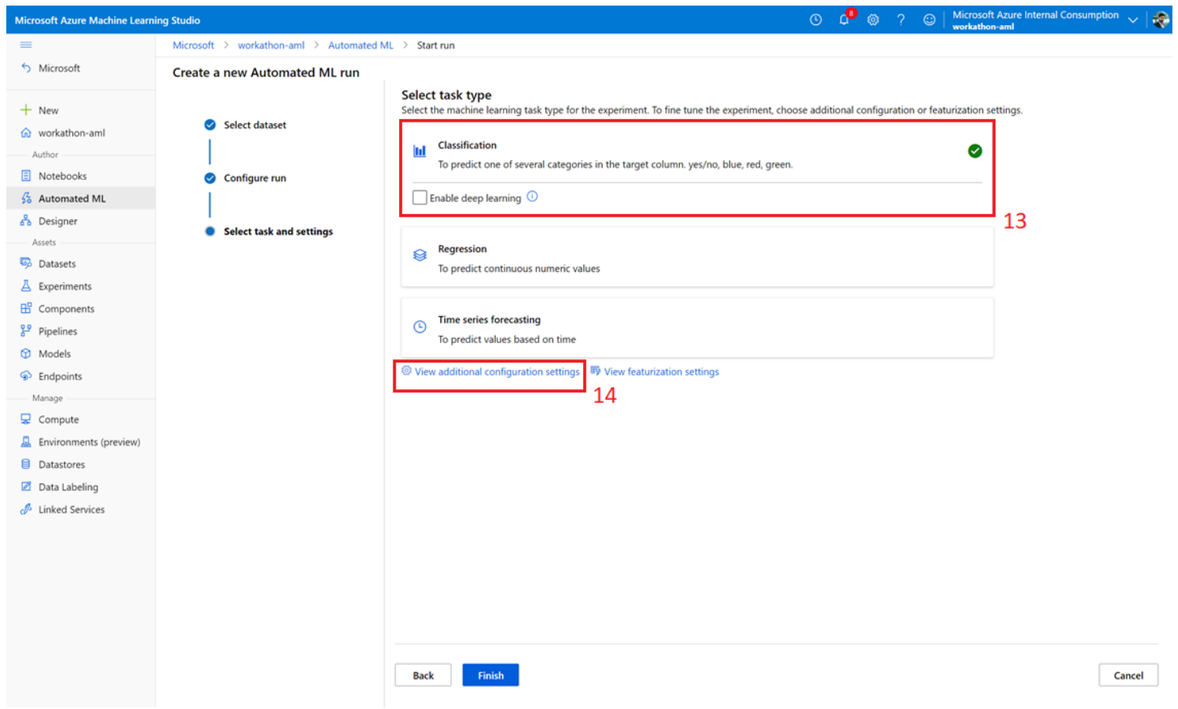
1. Go to the AutoML tab
2. If you are starting off for the first time, you can access some learning resources as highlighted
3. Click + New Automated ML run

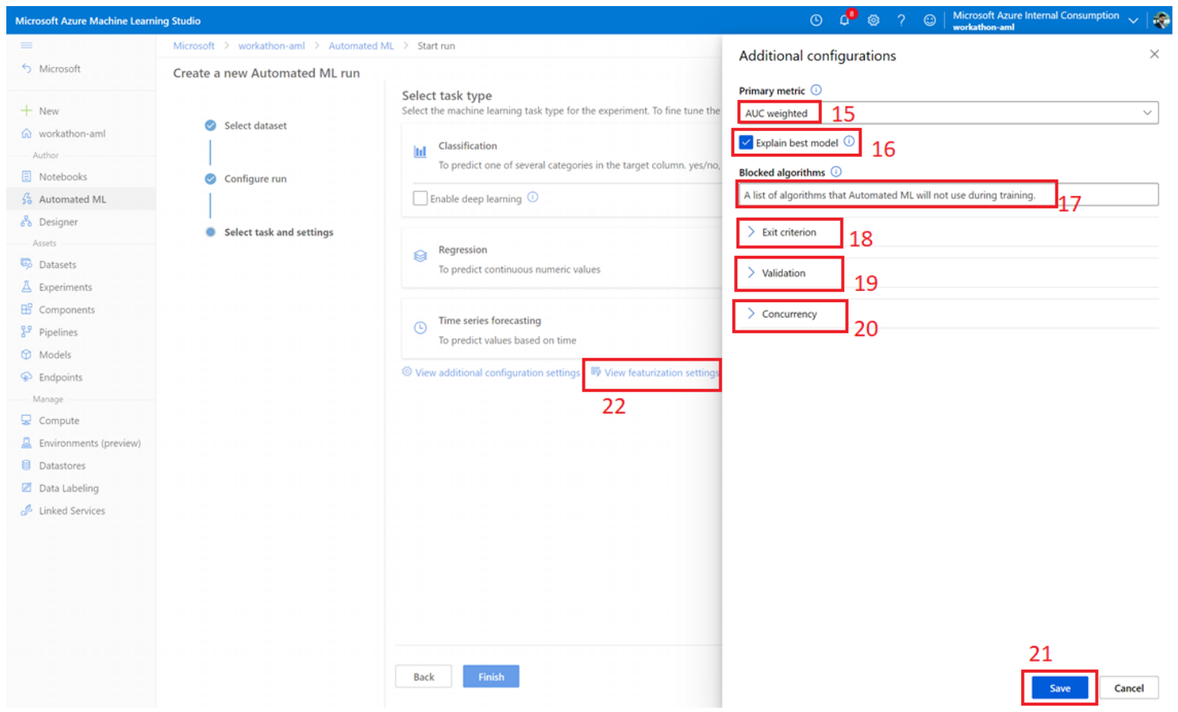
Select dataset

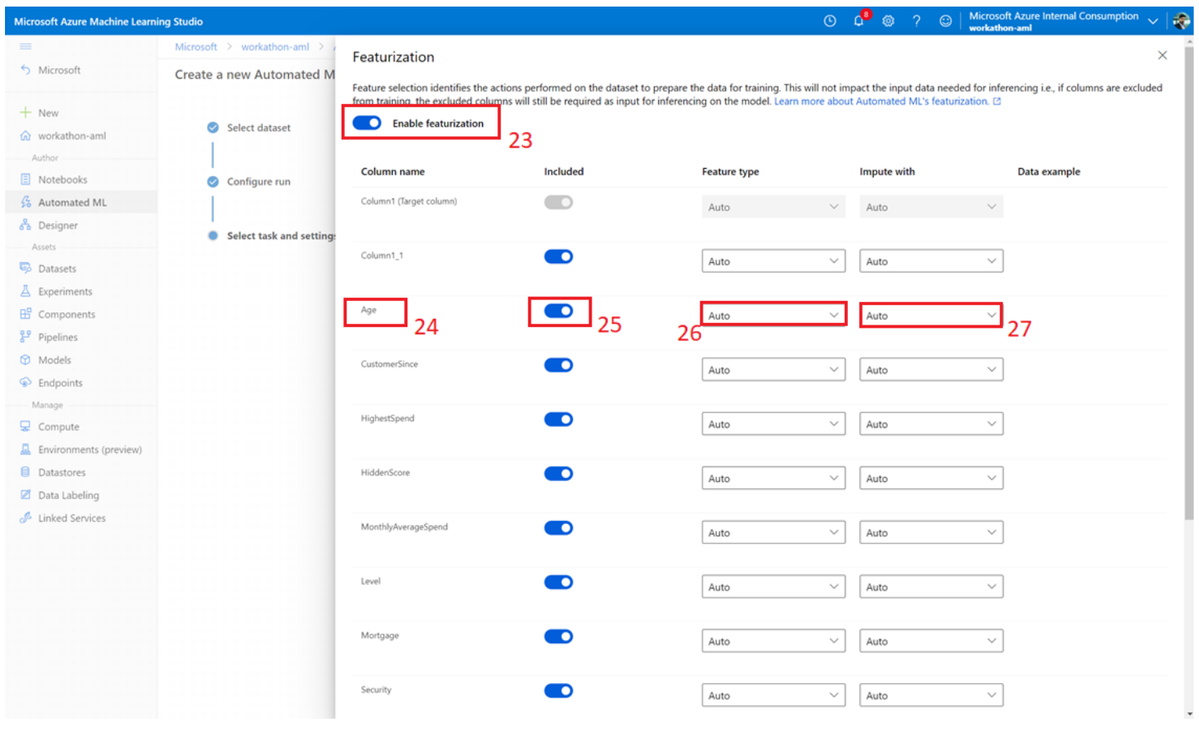
4. In the 'Select dataset' tab, you select the appropriate dataset on which you want to run the AutoML experiment. For instance, you can select the finance_ds dataset that we created in the workshop [You will see this dataset only if you ran the notebook we shared. Else you can select the dataset you created]
5. In case you don't have a dataset created, you can create a new dataset right from here, the way we created dataset1 and dataset2.
6. Click Next

Configure run

7. Experiment name : Every AutoML run is associated with an experiment, to help you track and manage the runs. You can select from an existing experiment or create a new experiment.
8. Experiment : For an existing experiment, select the experiment name from the dropdown. For a new experiment, provide a new name.
9. Target Column : This refers to the dependent column that you are trying to predict when creating a model. For instance, in our use case, LoanOnCard will be the target column
10. Compute cluster : From the dropdown, select the compute cluster on which you want to run the AutoML experiment.
11. Create a new compute : In case you don't have an existing compute cluster or want to provision a new cluster for this experiment, you can do it by clicking on this option and filling the cluster details like we did earlier.
12. Click Next after you have provided the run configuration







Select task type

The next step is to select the problem type you are trying to solve and configure the model settings. For now, AutoML supports Classification, Regression and Time series forecasting.

13. Select the problem type. Our use case falls under Classification, since we are trying to predict a discrete value for each customer.
14. Click 'View additional configuration settings'

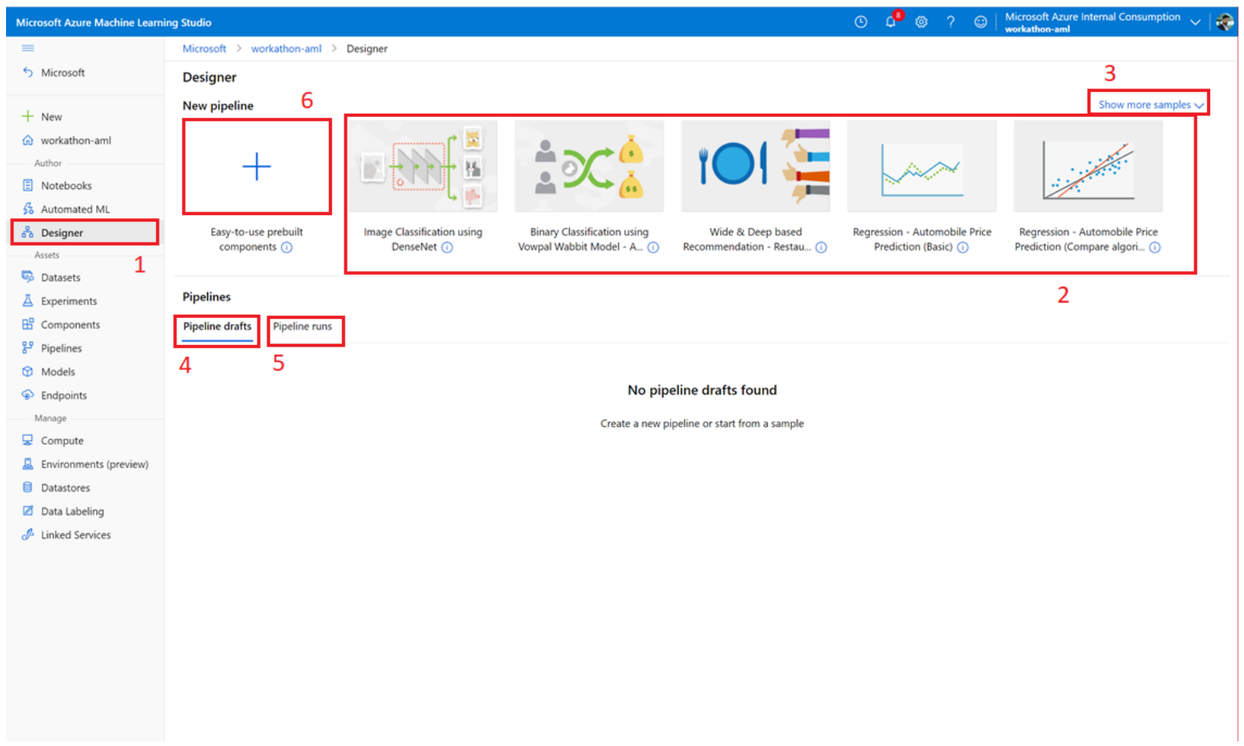
Additional configurations

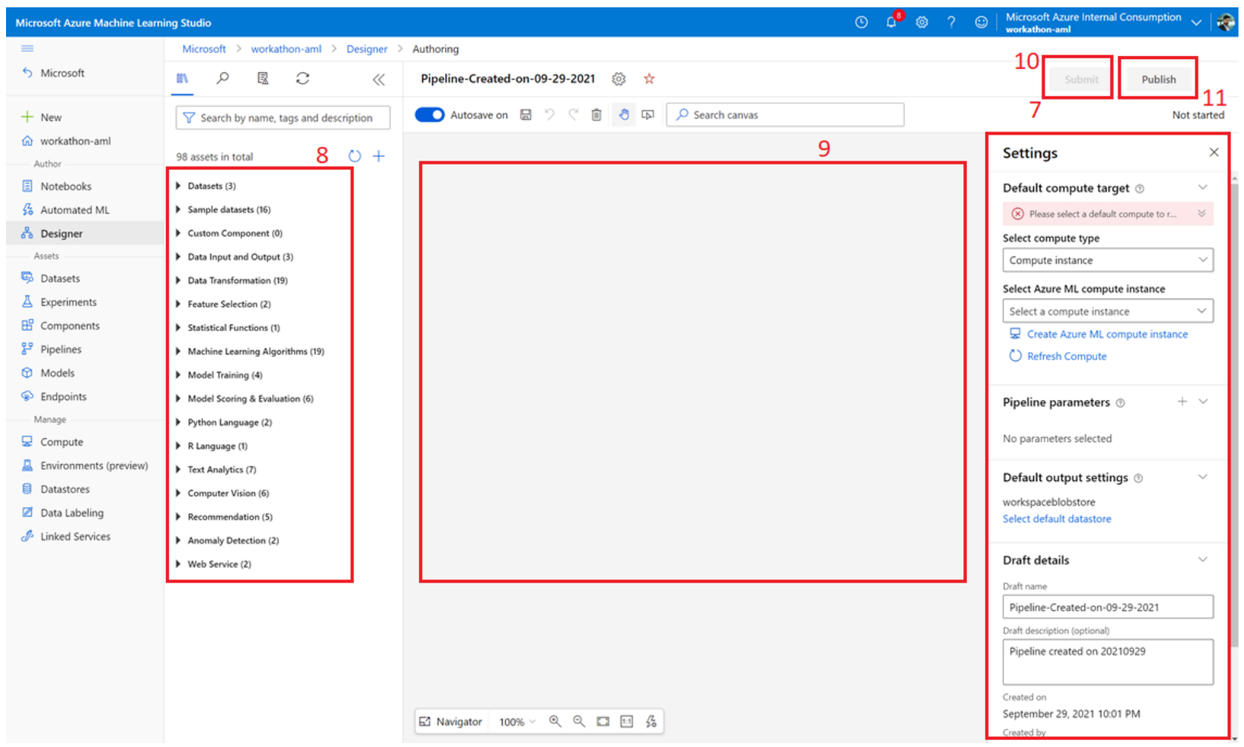
15. **Primary metric** : Out of all the models that AutoML will try out, the model that achieves the best value of Primary metrics will be selected as the best algorithm. You will get different metrics in the dropdown, basis the task type that you chose.
16. **Explain best model** : Checking this option will automatically generate & explain the results for the best model, in terms of which are the most important features, how a particular feature affects the prediction etc
17. **Blocked algorithms** : By default, AutoML tries a lot of different algorithms that are suitable basis the task type. In case you want the model to skip any algorithms, you can block them here. This will save you time and compute resources, however, it is recommended to let AutoML try all the algorithms so that it can provide you with the best model.
18. **Exit Criterion** : You can cause the AutoML experiment to terminate early using any of the following parameters –
    a. Training job time : This defines the maximum time for which the experiment will run. The model that performs best in that time frame will be selected as the best model
    b. Metric score threshold : This defines the threshold score for the Primary metrics. The experiment will terminate as soon as an algorithm crosses this threshold and that model will be selected as the best model
19. **Validation** : Model validation is referred to as the process where a trained model is evaluated with a testing data set. This parameter defines the validation technique to be used while training the model.
20. **Concurrency** : Since we run an AutoML experiment on a Compute cluster, we can set this parameter greater than 1 to run multiple algorithms in parallel. This parameter defines the maximum number of iterations that should be executed in parallel. This should be less than the number of nodes of the compute target (training cluster), 100 being the maximum value for Automated ML.
21. Once you have defined the configurations, click Save
22. Click '**View featurization settings**'
    [Feature selection identifies the actions performed on the dataset to prepare the data for training. This will not impact the input data needed for inferencing. i.e., if columns are excluded from training, the excluded columns will still be required as input for inferencing on the model. Similar transformations will also be applied to data that is sent as input during inference.]

Featurization

23. **Enable featurization** : You can choose to enable or disable featurization on the complete dataset
24. **Column name** : You can have different featurization settings for each column in the dataset
25. **Included** : You can choose to enable or disable featurization on a particular column
26. **Feature type** : AutoML automatically detects the datatype for each column basis the values. If required, you can explicitly set the Feature type.
27. **Impute with** : AutoML tries different imputation techniques for each column in case you have missing values. If required, you can explicitly set the Impute with column to functions like mean, median, most frequent etc.

Once you have set all the parameters, hit Finish.
The AutoML experiment will take a couple of minutes to finish executing. Observe the results by clicking on the run.

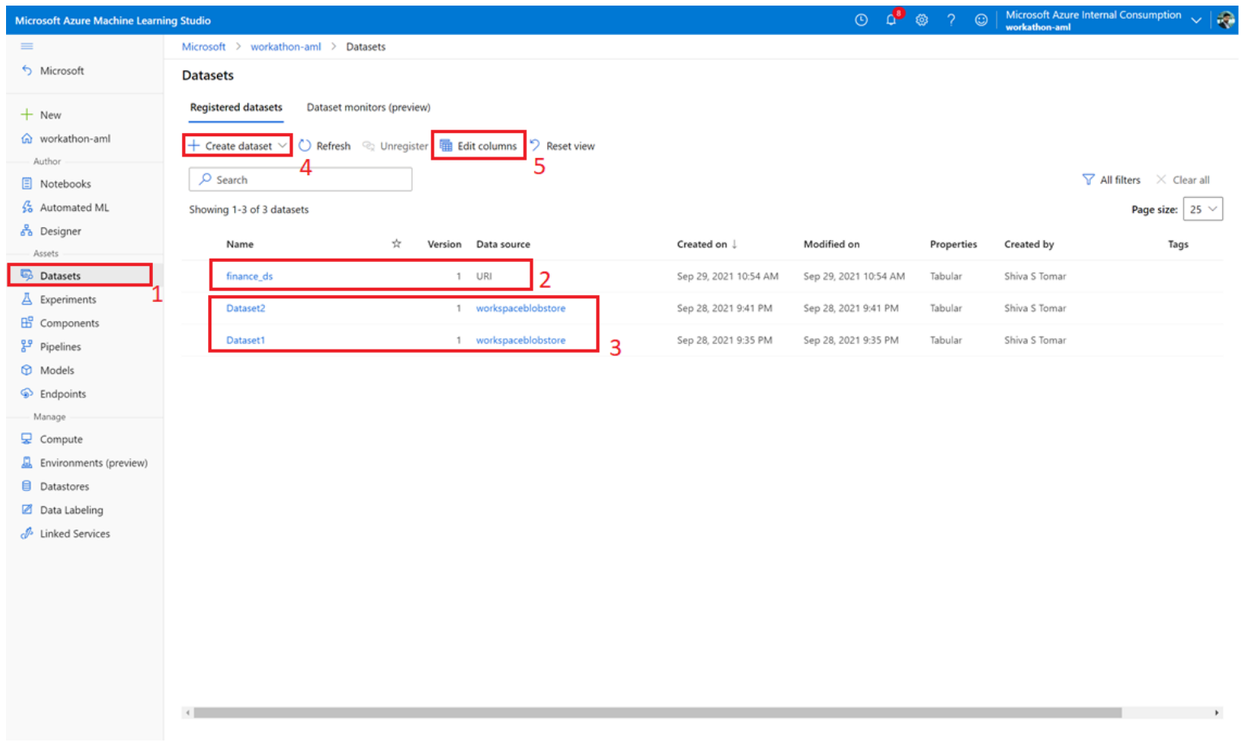




Explore Designer

Though we will not be creating a model using the Designer functionality in this workshop (This is given as a homework exercise), we want to walk you through the Designer experience to help you get started.

1. Click on the Desginer tab
2. Explore the pre-built pipelines for different use cases, to accelerate your understanding of AML Designer
3. Click 'Show more examples' to view additional pre-built use cases
4. **Pipeline drafts** : This tab will show all the pipelines that you have created. You can further build on them anytime
5. **Pipeline runs** : This tab will show all the runs for these pipelines. This gives information like the input, output, logs, underlying environment & code for each step, information about failed steps etc
6. Click the + icon to create a new pipeline from scratch
7. Once you open a new pipeline, you can see the settings in terms of Compute target, pipeline parameters, output settings etc
8. AML Desginer offers 100+ pre-built modules to choose from. These include modules across data transformation, model training, model evaluation and use cases like Computer Vision, Text Analytics, Anomaly Detection etc.
9. Drag the modules from step 8 to the canvas to get started with creating your pipeline.
10. **Submit** : Once you have created the pipeline, you can submit it. This will run all the pipeline steps. Rectify any errors that you receive.
11. **Publish** : Once you have created and run a pipeline, you can publish it as a real time inference pipeline and batch inference pipeline.



Explore the Datasets

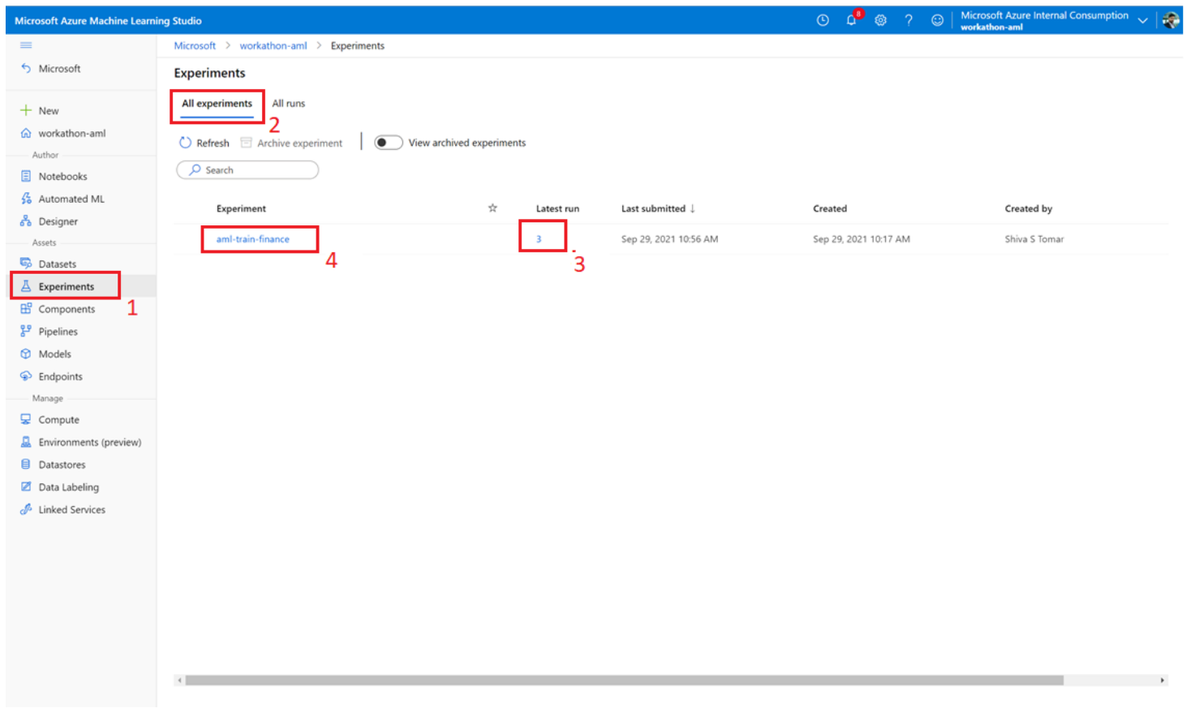
We explored Datasets section in the beginning of the workshop. Let's now explore how the view has changed post running the notebook.

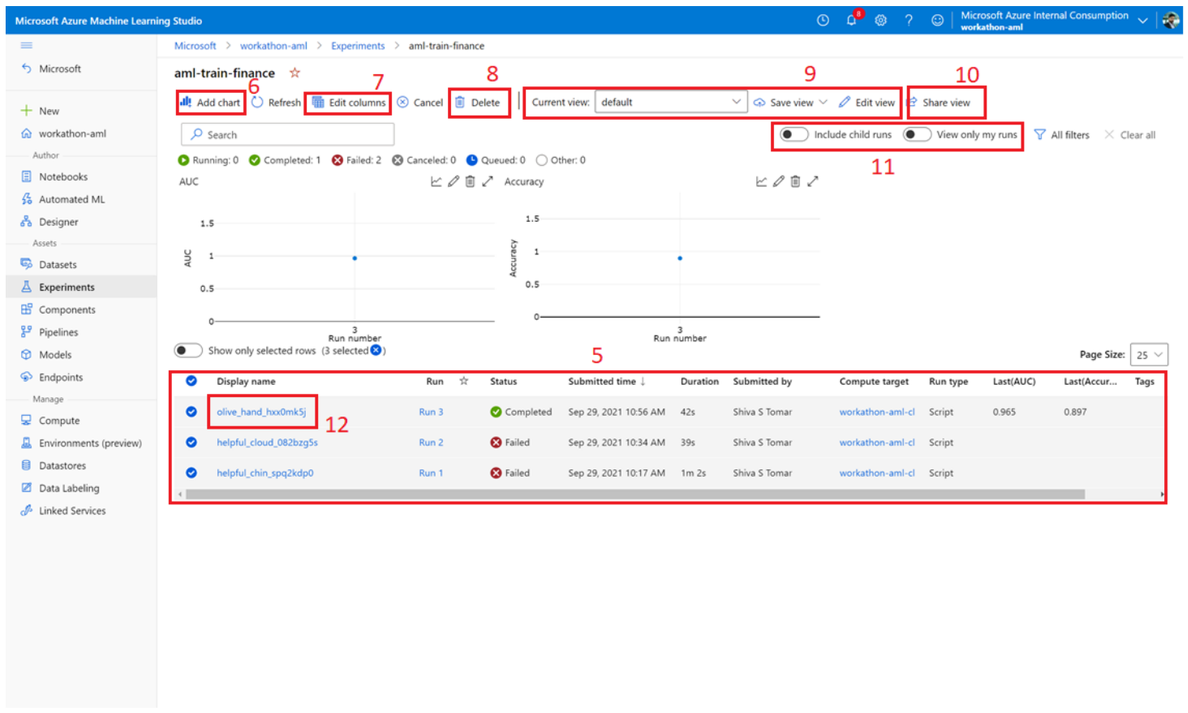
1. Go to Datasets tab
2. You will see a new dataset 'finance_ds' added to the dataset list. This is the dataset we registered on the fly by writing Python code from the notebook.

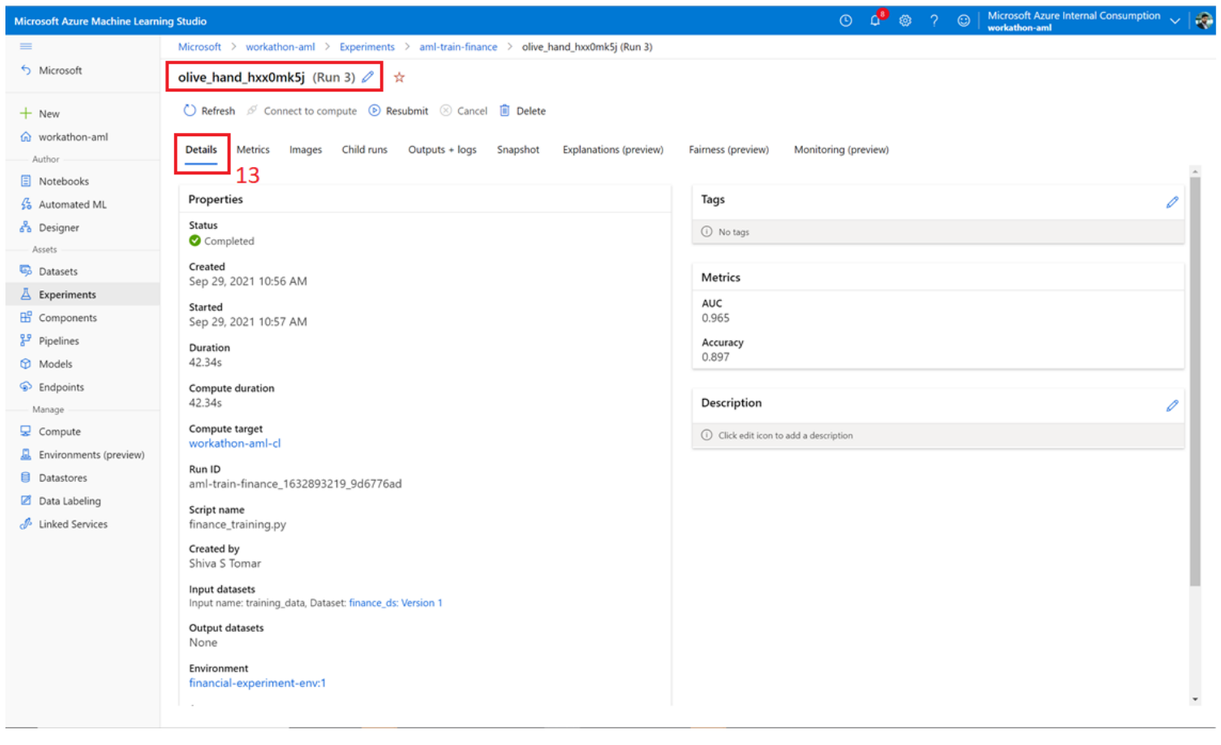
   The following code snippet from the notebook is responsible for registering the dataset in the AML workspace, which can be view through the Datasets tab in ML Studio :

```
local_path = 'data/finance.csv'
df_upsampled.to_csv(local_path)

# get the datastore to upload prepared data
datastore = ws.get_default_datastore()

# upload the local file from src_dir to the target_path in datastore
datastore.upload(src_dir='data', target_path='data')

#create a dataset referencing the cloud location
ds = Dataset.Tabular.from_delimited_files(datastore.path('data/finance.csv'))

financial = ds.register(workspace=ws, name='finance_ds', description='finance training data')
```

3. Dataset1 & Dataset2 are the ones we created earlier from the Studio GUI itself
4. You can create new datasets, if required.
5. Edit Columns : This option allows you to add or remove columns from your Dataset metadata view on the Studio GUI.







Explore the Experiments

An Experiment is like a compilation of your code which a job uses during the run to provide outcomes.

The experiments tab contains all the information about each experiment that you have run. This could be an AutoML experiment, a pipeline experiment or an experiment triggered by writing a script, the way we did in this workshop.

Once you have run the notebook and deployed the model, you'll see an experiment getting created here. The following code snippet defines the experiment configuration & submits the experiment :

```
#import libraries
from azureml.core import Experiment, ScriptRunConfig, Environment
from azureml.core.conda_dependencies import CondaDependencies
from azureml.widgets import RunDetails

# Get the training dataset
financial_ds = ws.datasets.get("finance_ds")

# get the registered environment
registered_env = Environment.get(ws, 'financial-experiment-env')

# Create a script config
script_config = ScriptRunConfig(source_directory=experiment_folder,
                                script='finance_training.py',
                                arguments = ['--input-data', financial_ds.as_named_input('training_data')], # Reference to dataset
                                environment=registered_env,
                                compute_target=cluster_name)

# submit the experiment
experiment_name = 'aml-train-finance'
experiment = Experiment(workspace=ws, name=experiment_name)
run = experiment.submit(config=script_config)
RunDetails(run).show()
```

Let's explore the information that is stored for each experiment run.

1. Go to the Experiments tab
2. Select All experiments
   [This will list all the experiments, which in turn contain their respective runs. The 'All runs' tab lists the runs for all the experiments in the same view.]
3. The latest run column displays the hyperlink to the most recent run in the experiment.
4. You can view the experiment line item created with the name you provided. Click on the experiment name

Viewing experiment details

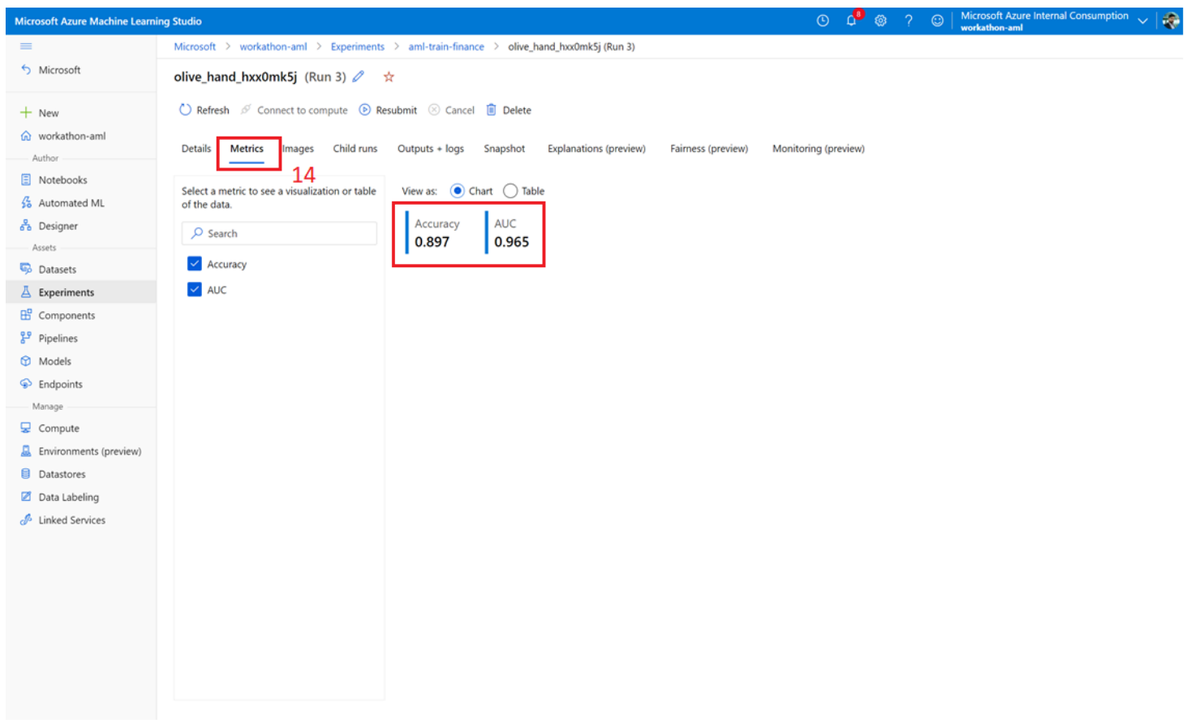
5. You can view all the runs for the selected experiment
   [Even we had 2 failed runs. So, it's absolutely okay if you are not able to run the model in the first go!]
6. **Add chart** : Though the experiment details tab does provide you with some basic charts for the metrics, you can create custom charts as you like.
7. **Edit Columns** : This option allows you to add or delete column for runs metadata & change the view of the table that lists the runs.
8. **Delete** : You can delete a particular run if you no longer want to retain it.
9. **Views** : As and when you customise the details page by adding charts, editing columns etc, you can save the customisations as 'Views'. This is helpful when multiple people are working on the same experiment and each of them wants to customise the details page according to their convenience.
10. **Share View** : You can share the default or custom views using a shareable link. This does not generate a public URL. Access to this view will require the user to have IAM access to this workspace.
11. You can toggle between the runs that you want to view, such as if you want to view all the child or view runs in this experiment that were created by anyone or just by you.
12. Click on the latest successful run in your experiment.

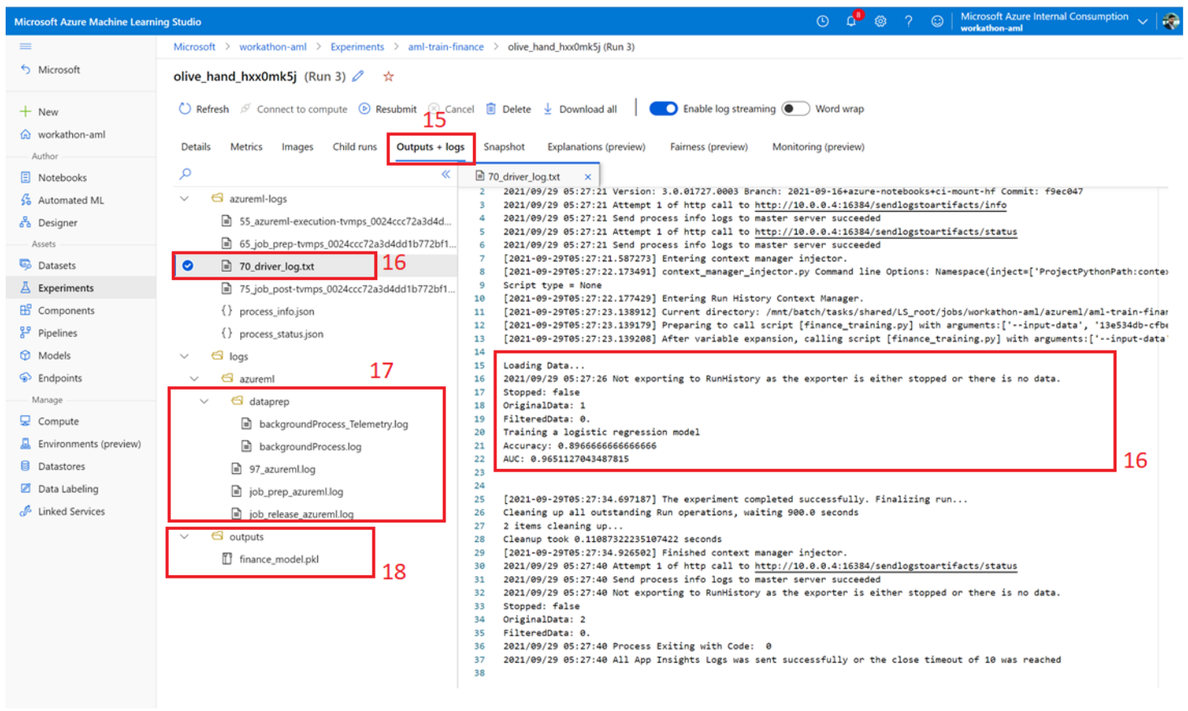
View run details

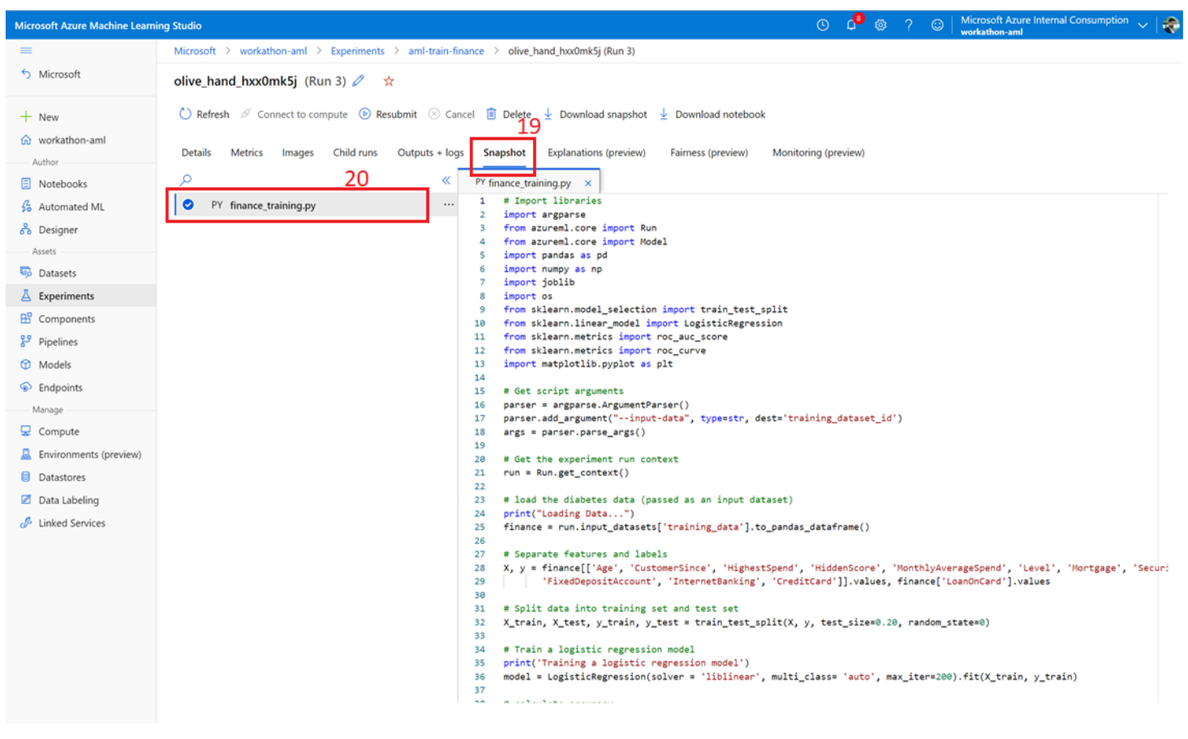
Each run has an associated name and run number. You can edit the run name anytime you like.

Details tab

13. In the details tab, you can view the metadata for the run such as it's status, created time, duration, compute target, run ID, metrics, tags etc.







Metrics

14. The metrics tab displays all the metrics you logged during training the experiment. In our notebook, we logged the Accuracy & AUC Score metrics.

The following code snippet under 'Prepare your training script' section is responsible to log these metrics :

```
# calculate accuracy
y_hat = model.predict(X_test)
acc = np.average(y_hat == y_test)
print('Accuracy:', acc)
run.log('Accuracy', np.float(acc))

# calculate AUC
y_scores = model.predict_proba(X_test)
auc = roc_auc_score(y_test,y_scores[:,1])
print('AUC: ' + str(auc))
run.log('AUC', np.float(auc))
```

Outputs + logs

15. The Outputs + logs tab contains the log files for your end-to-end model execution and any outputs that were saved during model execution, such as model dumps etc.
16. Select the 70_driver_log.txt file. This is 1 of the most important log files. It helps you see the execution of your training script. Any print statements that you wrote in the training script that was run on a remote compute (Compute Cluster in our case) can be viewed here. This easily helps you to debug your model training scripts by adding print statements as checkpoints.
17. You can also view the logs for job preparation, background telemetry etc
18. We also dumped the model as a pickle file to be leveraged later. Anything that's uploaded during the remote training run will be saved under the outputs folder of that run.

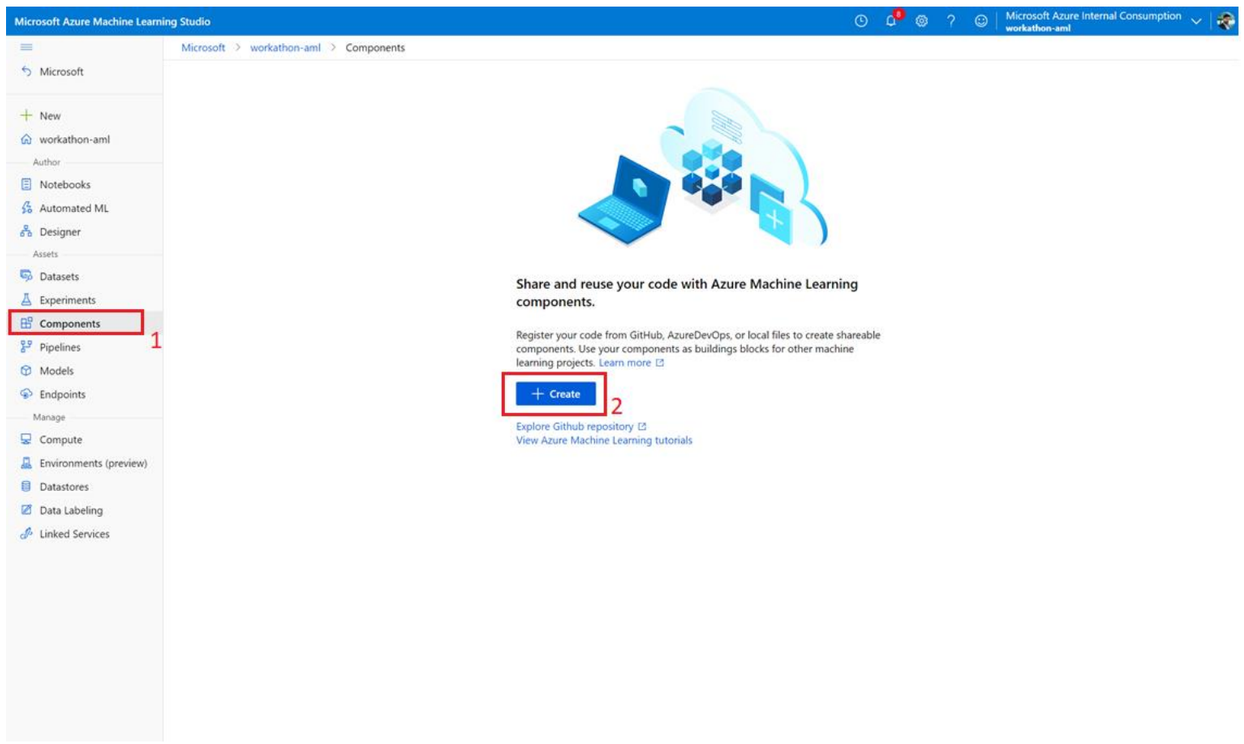
The following code snippet under 'Prepare your training script' section is responsible for saving the model dump file :

```
os.makedirs('outputs', exist_ok=True)
# note file saved in the outputs folder is automatically uploaded into experiment record
joblib.dump(value=model, filename='outputs/finance_model.pkl')
```

Snapshot

19. This tab contains a snapshot of the parent folder of your training script. (In our use case, this is the folder that contains the finance_training.py file i.e finance_training_folder. We just had 1 file in this folder. However, if you have other scripts, data files or other files in that folder, those will also be uploaded.)
20. Open the finance_training.py file. Notice this is the training script we created.

Snapshot helps you to version and save the scripts and code for each run.

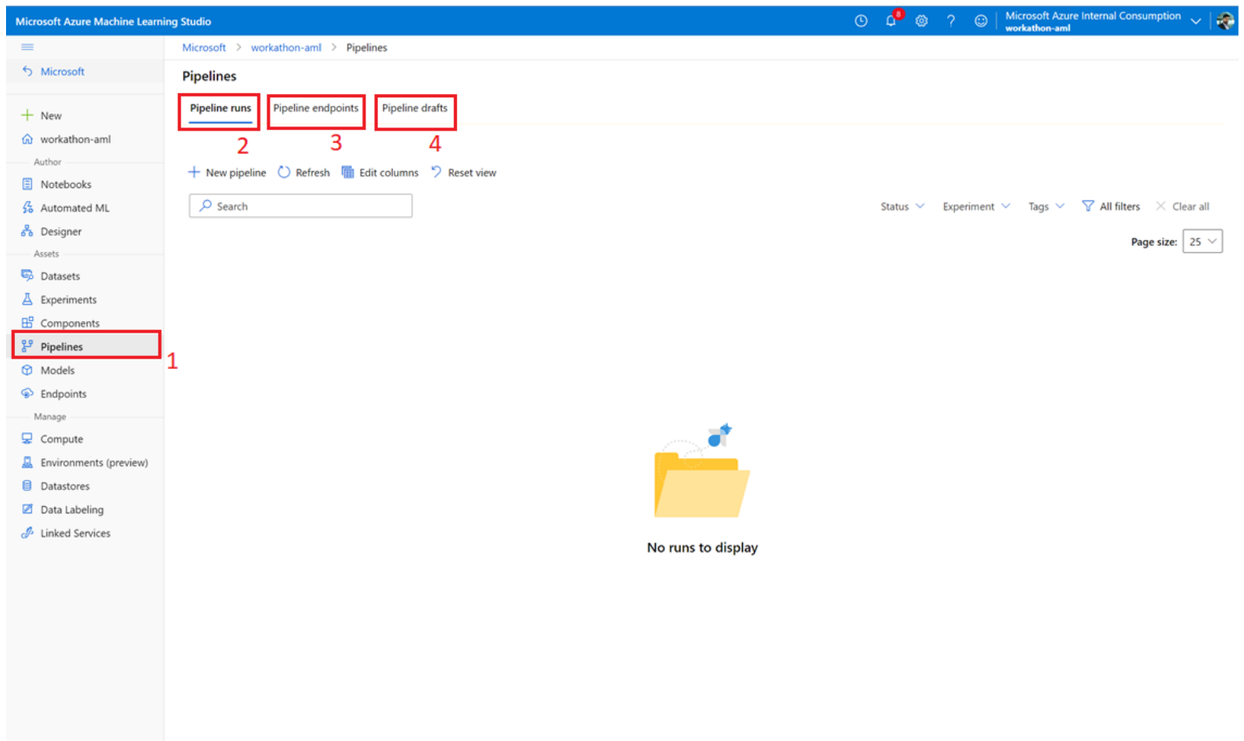


Explore Components

A component is a self-contained set of code that performs one step in machine learning pipeline, such as data pre-processing, model training, model scoring and so on. A component is analogous to a function, in that it has a name, parameters, expects certain input and returns some value. Any python script can be wrapped as a component.

Creating components in AML allows you to share and reuse your code with others.
You can register your code from GitHub, AzureDevOps or local files to create shareable components.

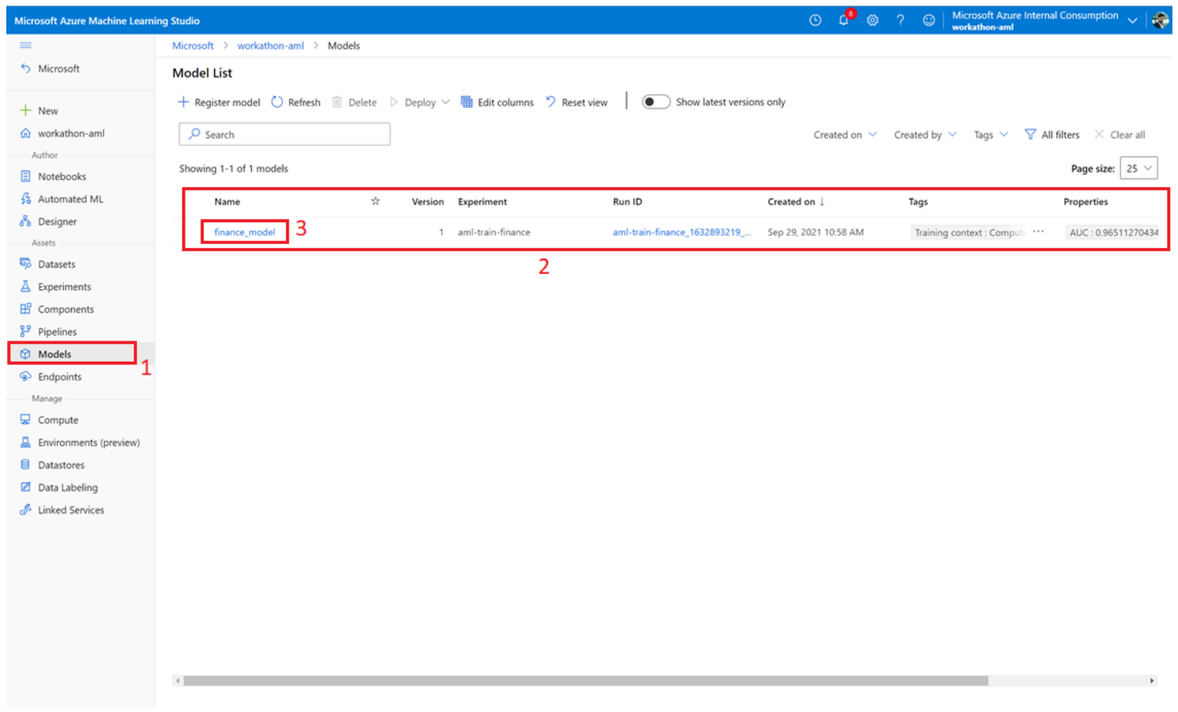
1. Go to the components tab
2. Click + Create to get started

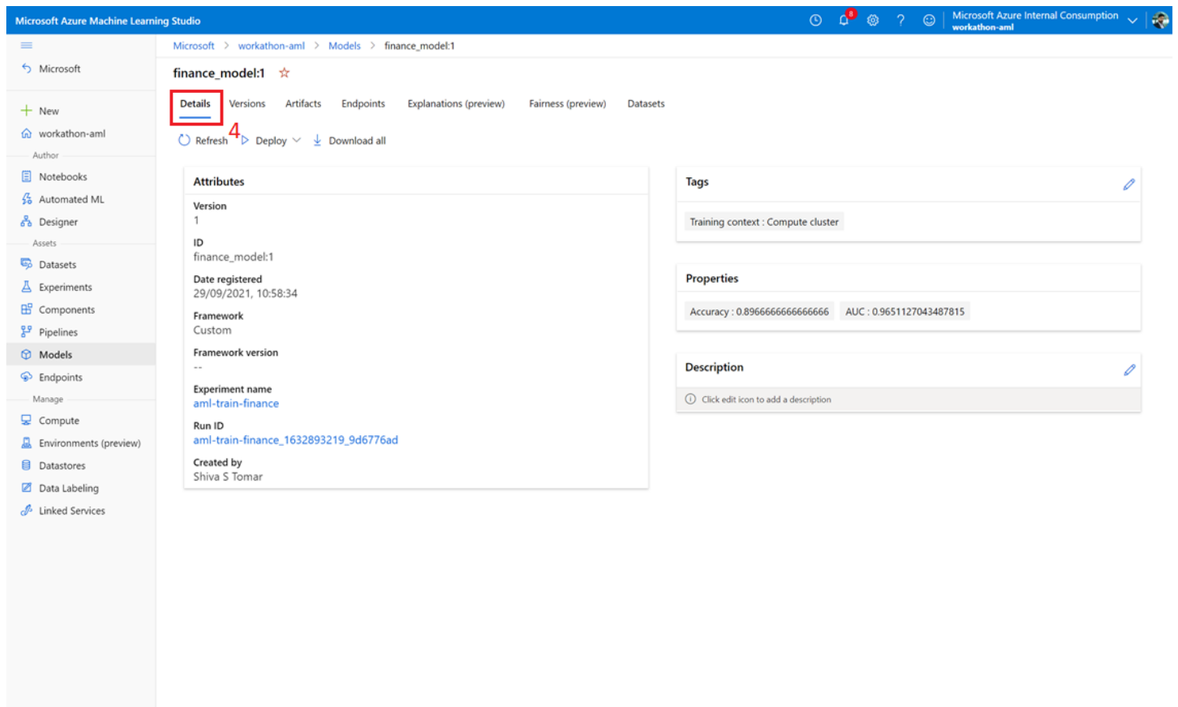


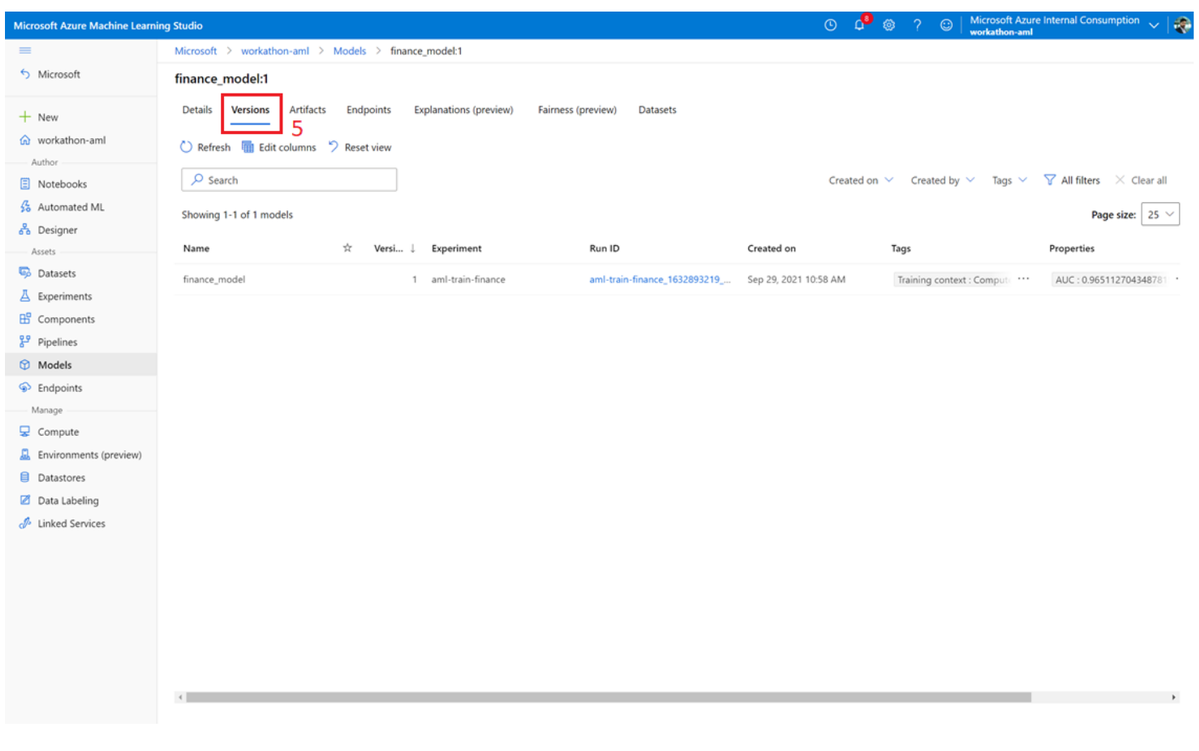
Explore Pipelines

A pipeline is a series of activities integrated together to perform a specific operation like data exploration & preparation, model building etc.
A pipeline can be built using AML designer or AML Python SDK.

1. Go to **Pipelines** tab : This displays all the pipelines you have built.
2. **Pipeline runs** : This tab displays the run details for all the pipelines.
3. **Pipeline endpoint** : This tab displays all active pipeline endpoints. Unlike real time endpoints, pipelines are published to leverage them later for batch processing. During batch processing, all the steps defined in the pipeline are executed on the input data. This could be a simple pipeline that just performs data cleaning or a complex pipeline that trains a model and publishes it as a real time endpoint.
4. **Pipeline drafts** : This tab is like Pipeline drafts tab in Designer and displays all the pipelines you have created using the Designer.







Explore Models

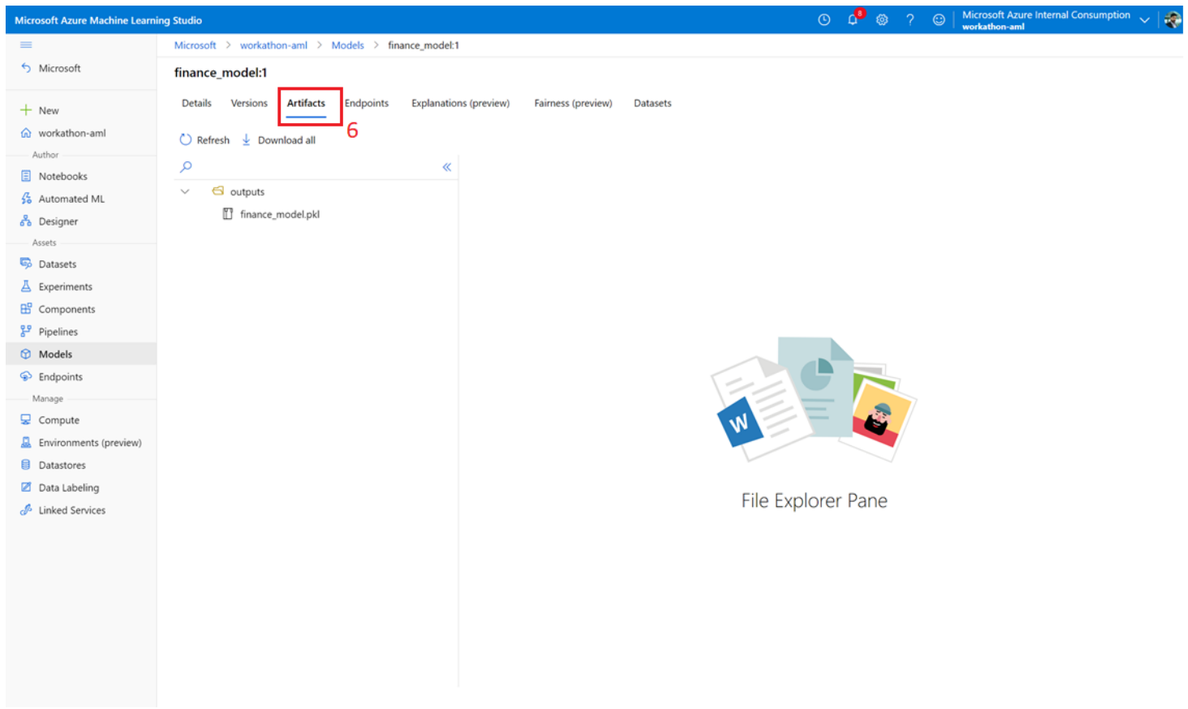
All models that you register will appear in the Models tab. In our use case, we registered the Logistic Regression model with the name finance_model.
These Models are versioned and every time you register a new model with an existing model name, it will create a new version.
A model registered with a new name will appear as version 1 for that model.

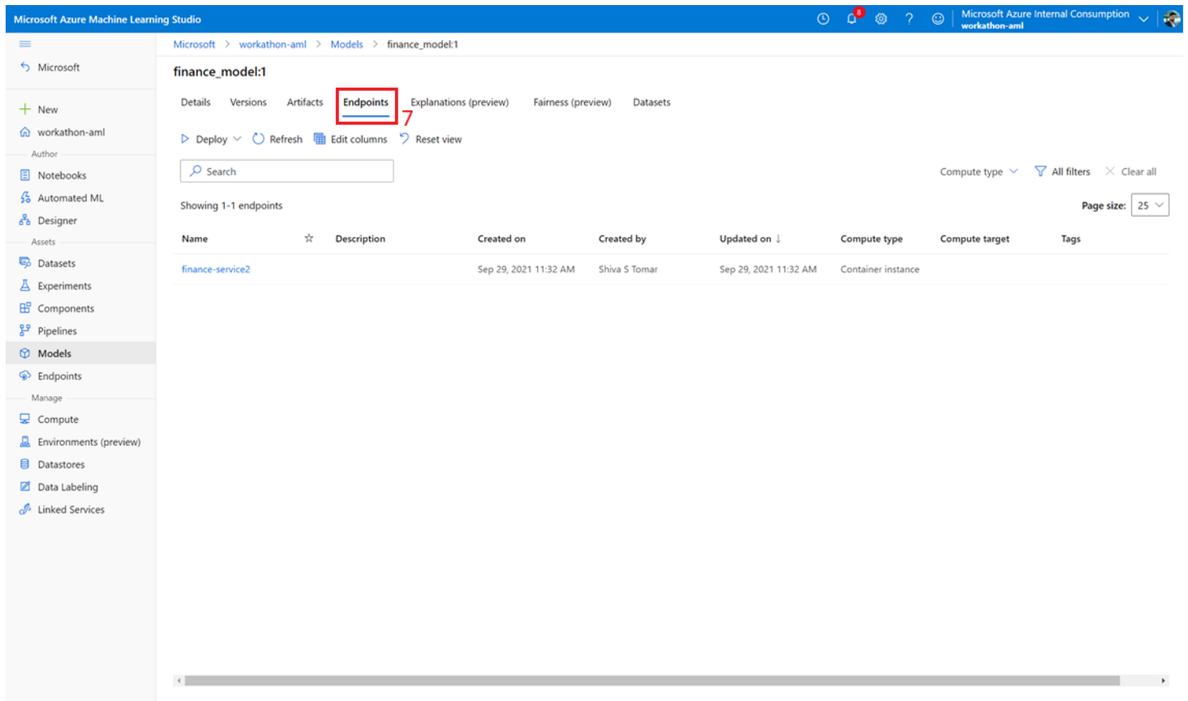
The following code snippet in the notebook is responsible for registering the model in the AML workspace :

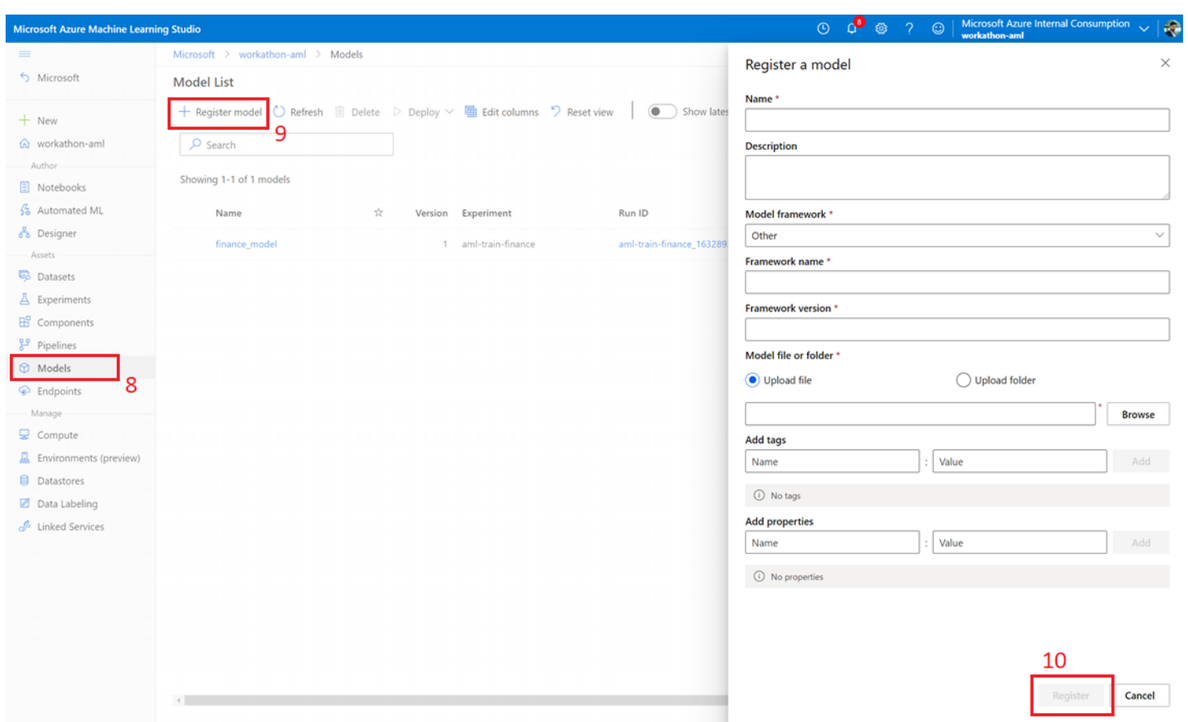
```
from azureml.core import Model

# Register the model
run.register_model(model_path='outputs/finance_model.pkl',
                   model_name='finance_model',
                   tags={'Training context':'Compute cluster'},
                   properties={'AUC': run.get_metrics()['AUC'],
                               'Accuracy': run.get_metrics()['Accuracy']})
```

1. Go to the Models tab
2. You can view the metadata for models such as name, version, experiment, tags, properties etc
3. Select the model you registered to further explore it.
4. In the details tab, you can view metadata for the model such as version, ID, date, tags, properties etc

5. The Versions tab also lists all the other versions for the selected model and their metadata. In our case, we just have 1 version, hence can view only 1 line item.



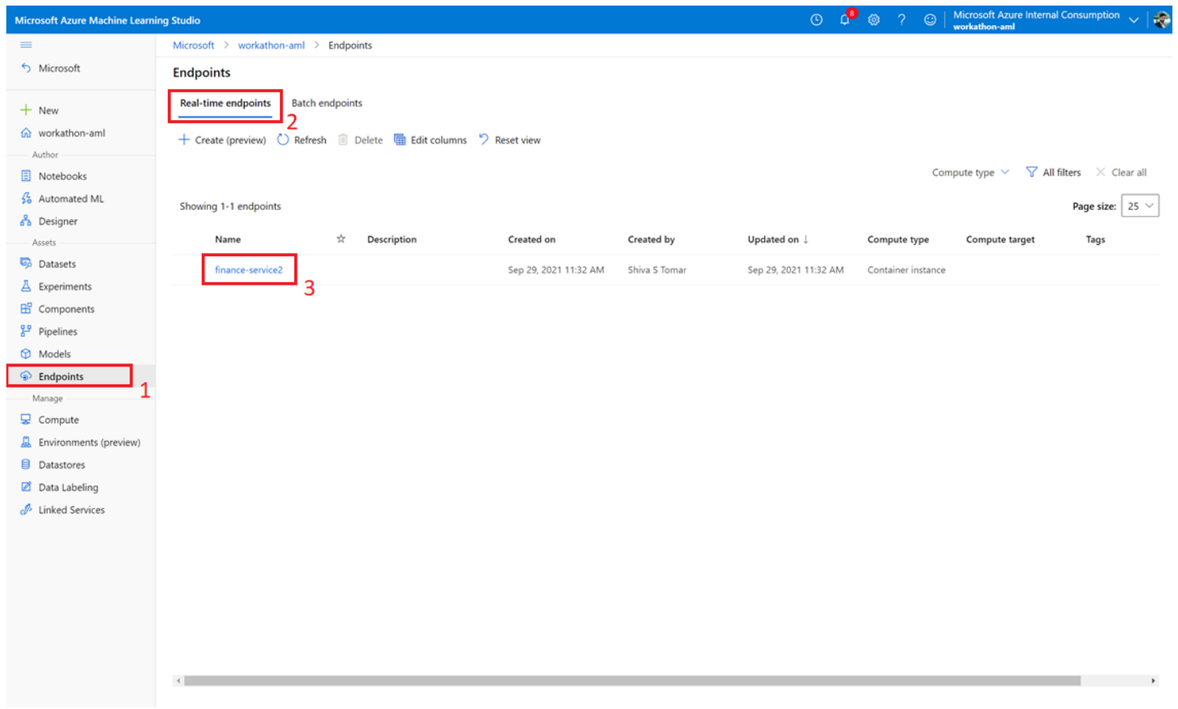


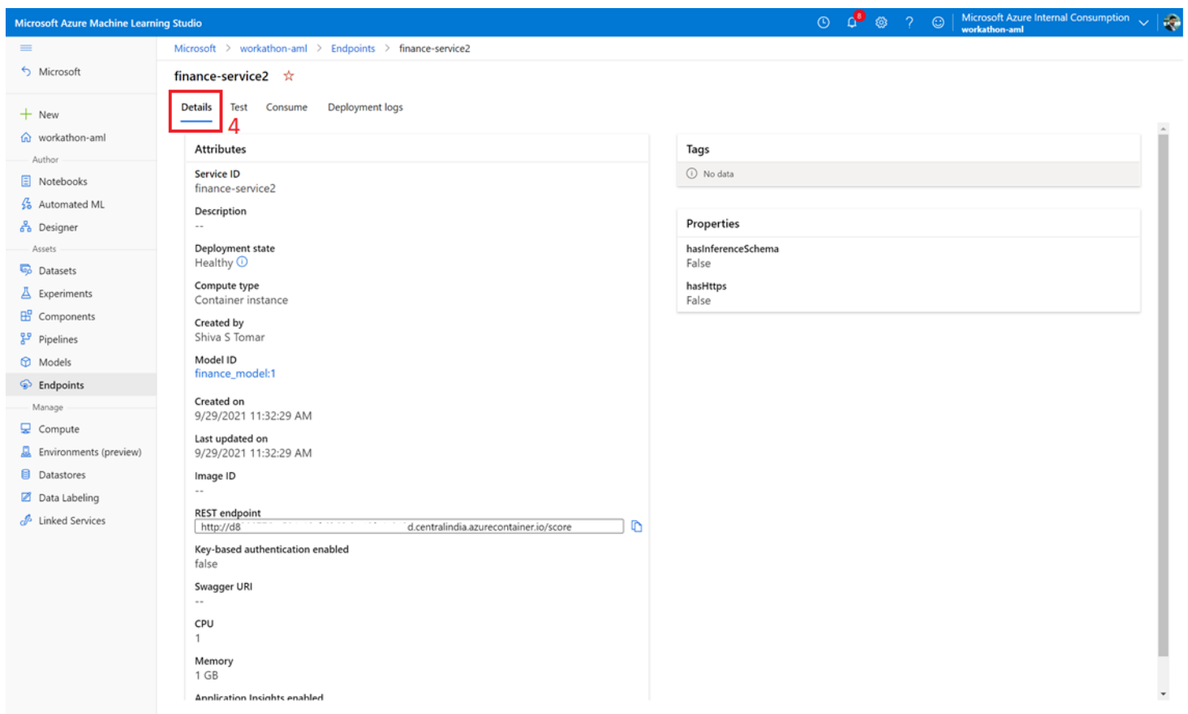


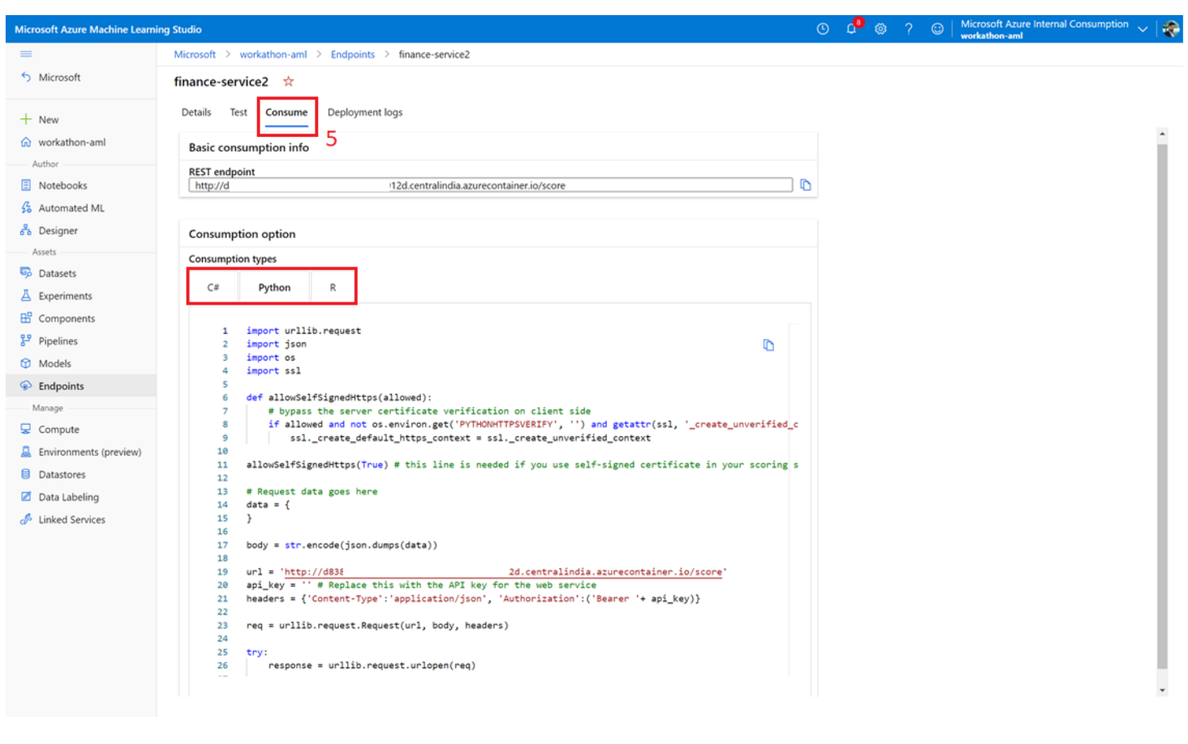
6. The artefacts tab contains the underlying model resources, such as model dumps (finance_model.pkl file in our case).

7. The Endpoints tab displays a list of endpoints that were deployed using the selected model. The deployed finance-service2 endpoint using version 1 of finance_model.

8. Go back to the main Models tab
9. Just like we registered the model using Python code, you can also register a new model from the GUI. You will need to provide the details such as model name, description, framework, the file that contains the model dump. Optionally, you can provide tags and properties.
10. Once you have configured the details, click Register. You will find the model gets added to the Model List.







Explore Endpoints

In Azure ML, you can provision 2 types of endpoints –

a. Real-time : These are used to make real-time calls for inference. Each endpoint has an underlying model that is leveraged to make the inference.
b. Batch : These are used to run pipelines in batch mode. You can have pipelines that provide inferences for bulk data or pipelines that retrain & deploy a model using new input data. Each batch endpoint has an underlying pipeline that it leverages. These pipelines can be built using Designer or Code based approach.

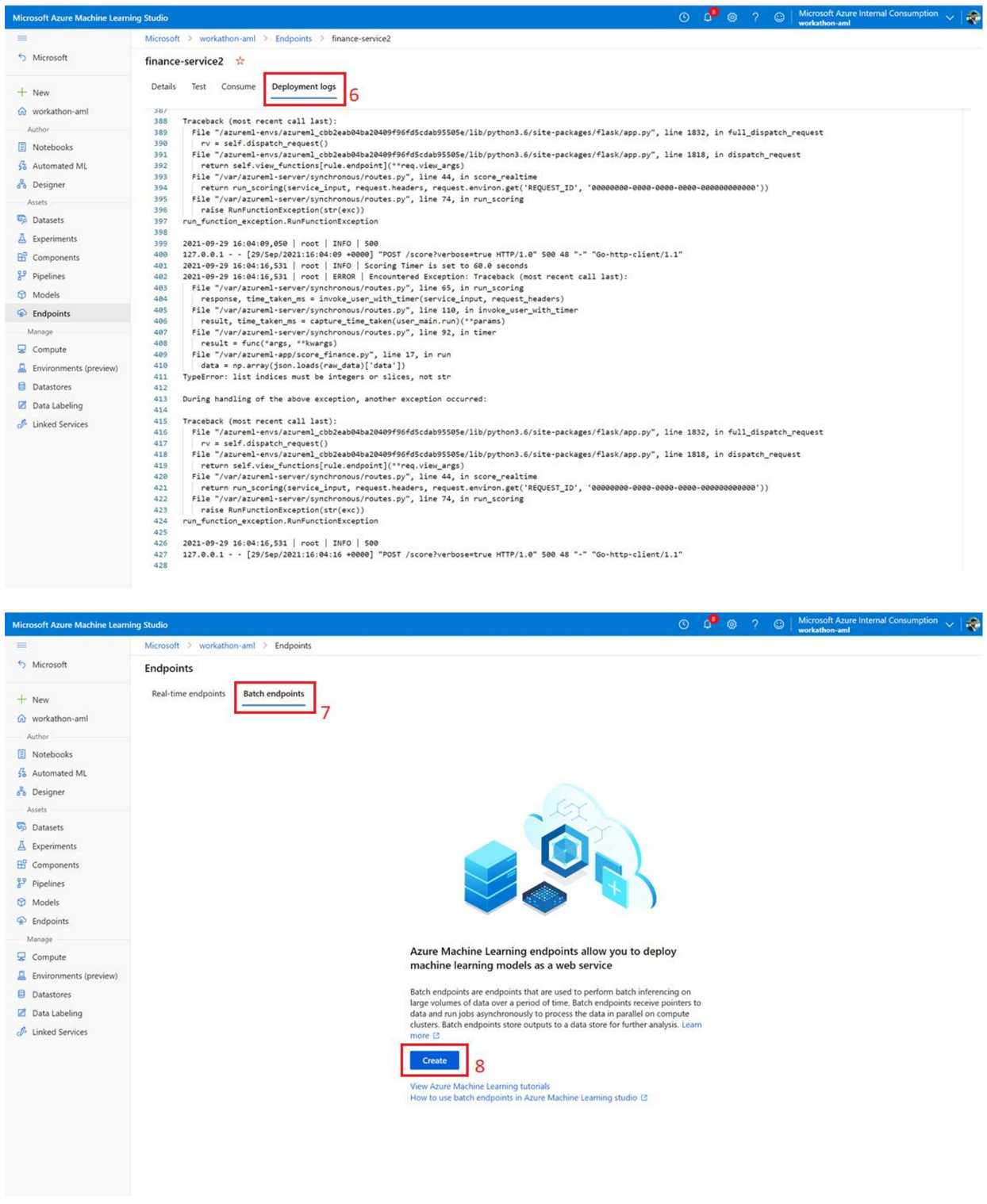
The following code snippet in 'Deploy the model as a web service' section is responsible for deploying the model as a real time end-point :

```
from azureml.core.webservice import AciWebservice
from azureml.core.model import InferenceConfig

# Configure the scoring environment
inference_config = InferenceConfig(entry_script=script_file,
                                   environment=registered_env)

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

service_name = "finance-service2"

service = Model.deploy(ws, service_name, [model], inference_config, deployment_config)

service.wait_for_deployment(True)
print(service.state)
```

Let's explore the endpoint details

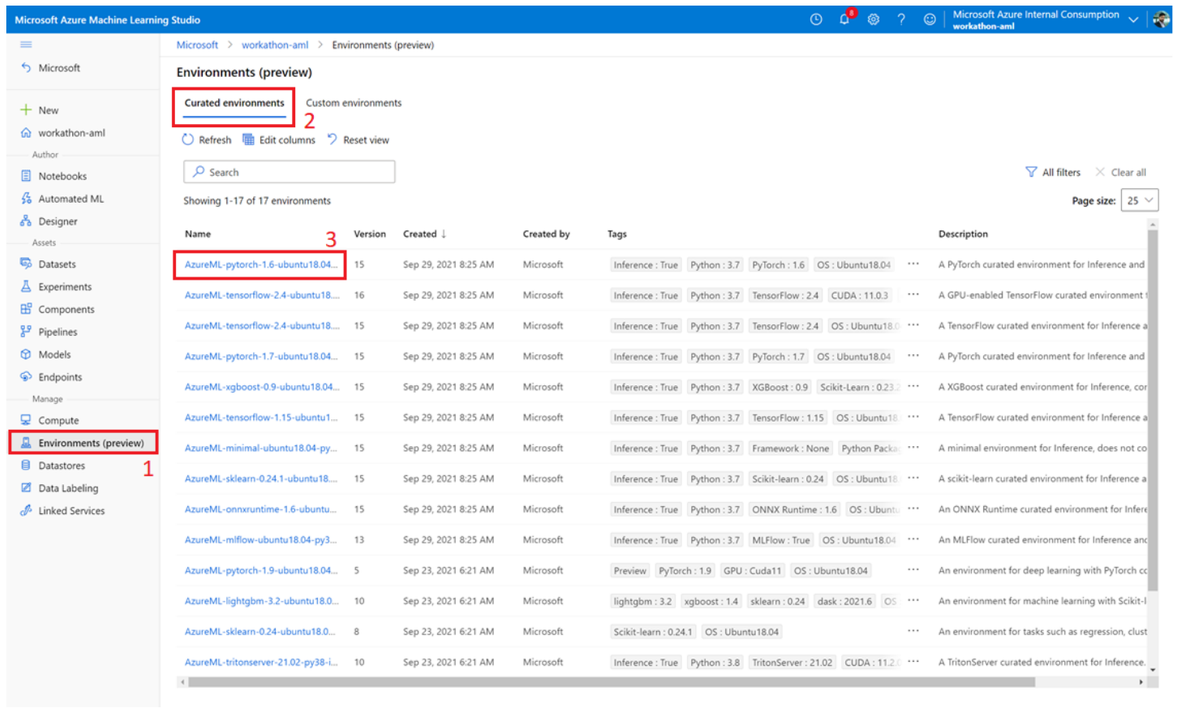
1. Go to the Endpoints tab
2. Select Real-time endpoints. Here you can find all the endpoints you have currently deployed. You see metadata like endpoint name, created on, created by, updated on, compute type etc.
3. Select the endpoint you deployed
4. In the details tab, you can view endpoint metadata like service ID, status, compute type, created by, REST endpoint, compute configuration in terms of CPU & Memory etc.

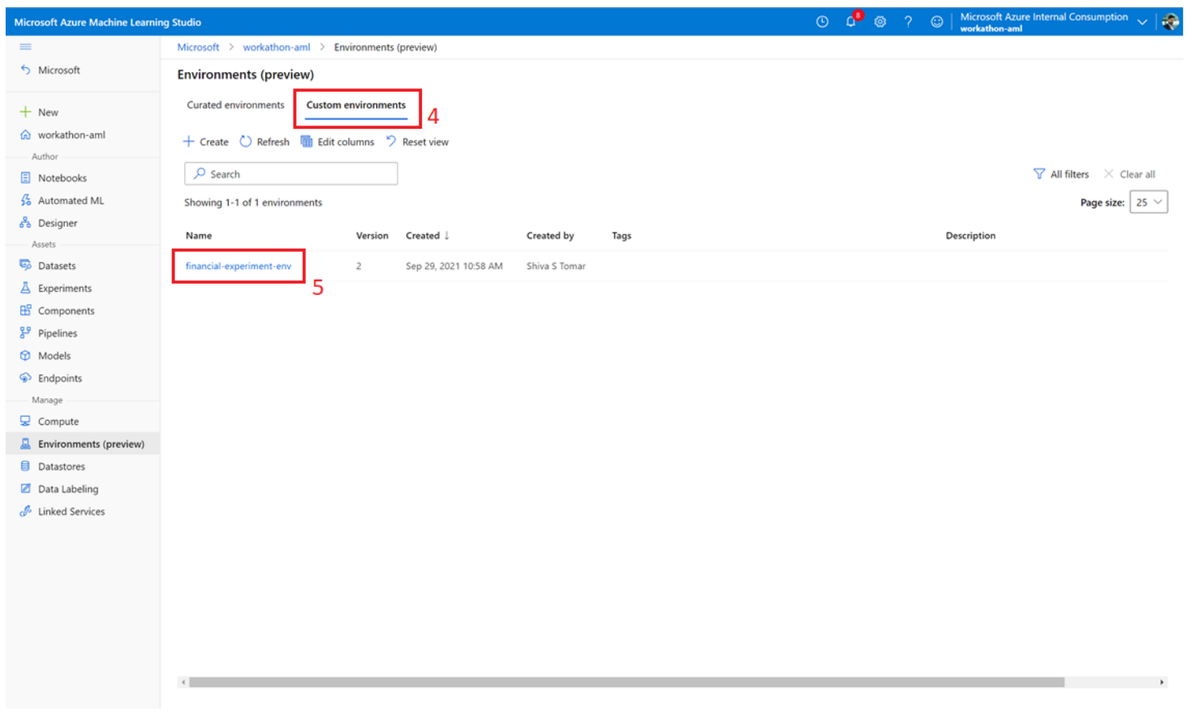
5. The Consume tab displays the REST endpoint and code snippets in various languages that you can leverage to make real time calls to the endpoint to get inferences.

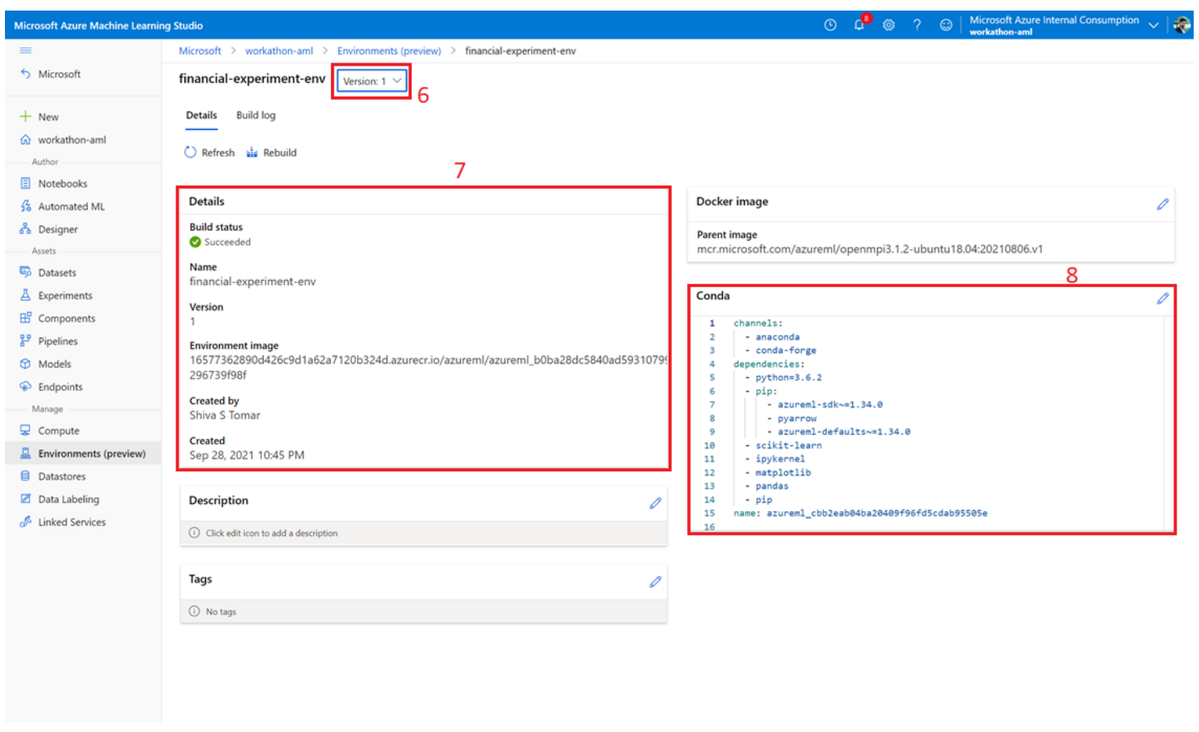


6. The Deployment logs tab display the logs for endpoint deployment. Any errors that occur during endpoint deployment can also be viewed here.

7. Go to Batch endpoints tab. In this workshop, we did not create any pipeline or batch endpoint, hence we see an empty list. However, we can view the batch endpoint details in the same way that we explored real-time endpoints.
8. You can create a new batch endpoint from the GUI as well, by clicking on the Create button.







Explore Environments

AML environments define the virtual environment where your ML training happens. They specify the Python packages, environment variables, and software settings around your training and scoring scripts. They also specify run times (Python, Spark, or Docker). The environments are managed and versioned entities within your Machine Learning workspace that enable reproducible, auditable, and portable machine learning workflows across a variety of compute targets.

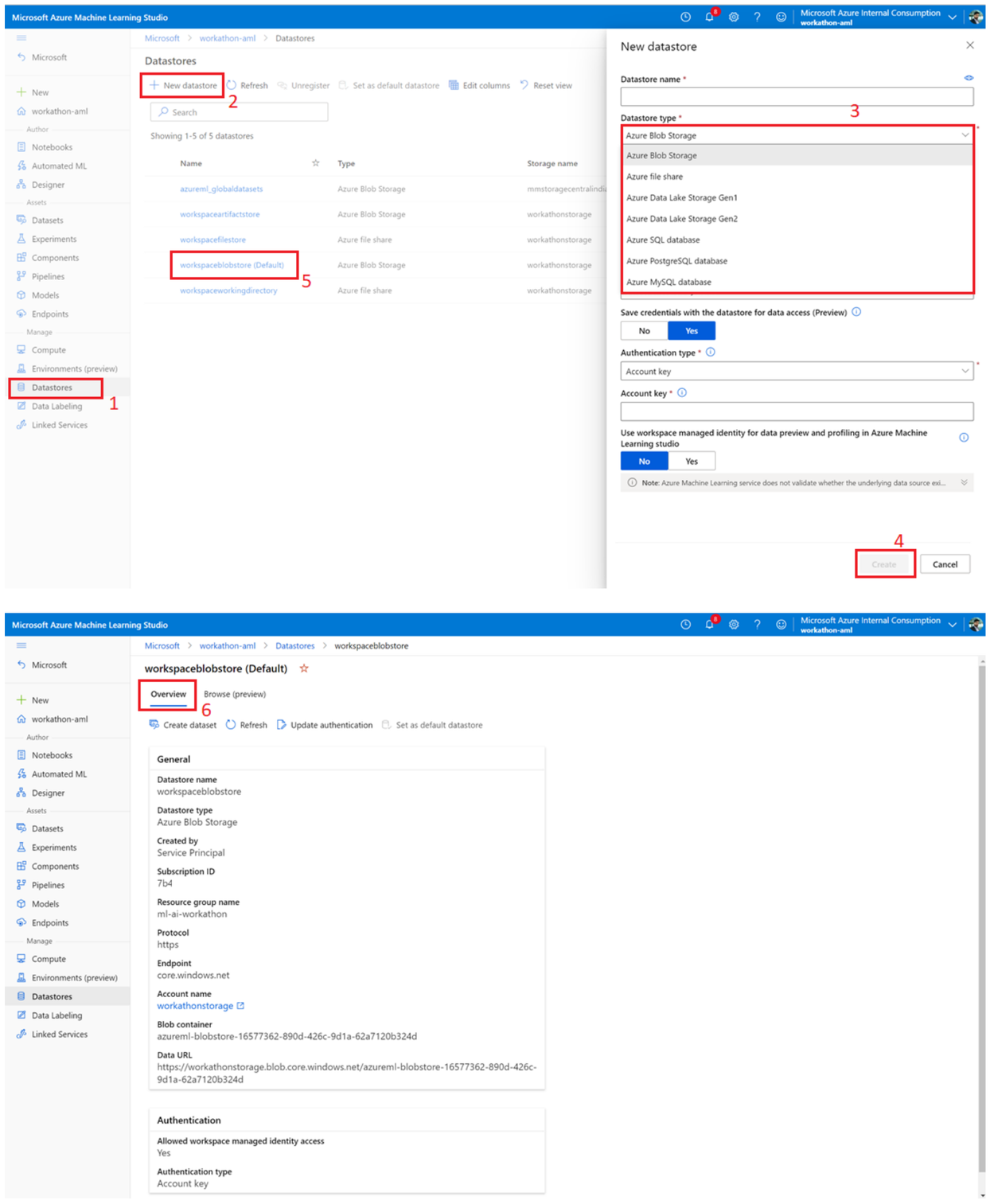
There are 2 types of environments in AML -

a. **Curated environments** : These provided by pre-built environments and are available in your workspace by default. Intended to be used as is, they contain collections of Python packages and settings to help you get started with various machine learning frameworks. These pre-created environments also allow for faster deployment time.
b. **Custom environments** : These are the environments that you create basis your requirements. For instance, for a Python environment, you provide the Conda & Pip dependencies in a YAML file to define the environment. Environments can be created & registered using GUI or code-based approach.

Let's explore the Environments now

1. Go to the Environments (preview) tab
2. Select Curated environments. This tab lists all the pre-built environments that are available. You can leverage these while building the model by referring to the Environment name.
3. Click on any of the pre-built environments to explore them further.
4. Select Custom environments tab
5. You will see the financial-experiment-env environment that we created for model training. Click the environment name to explore further.

6. **Environment Version :** Just like datasets, models etc, AML allows you to manage environments using versions. Every time you register an environment with the same name, it creates a new version.
7. You can view the environment details metadata such as Build status, name , version, image, created by etc.
8. You can also view the Conda file that was created for the environment you provisioned and registered via the notebook.



Explore Datastores

AML datastores securely keep the connection information to your data storage on Azure, so you don't have to code it in your scripts. You can create a datastore to easily connect to your data storage location.

Supported cloud-based storage services in Azure that can be registered as datastores:

- Azure Blob Storage
- Azure File Share
- Azure Data Lake Gen1
- Azure Data Lake Gen2
- Azure SQL Database
- Azure Database for PostgreSQL
- Azure Database for MySQL

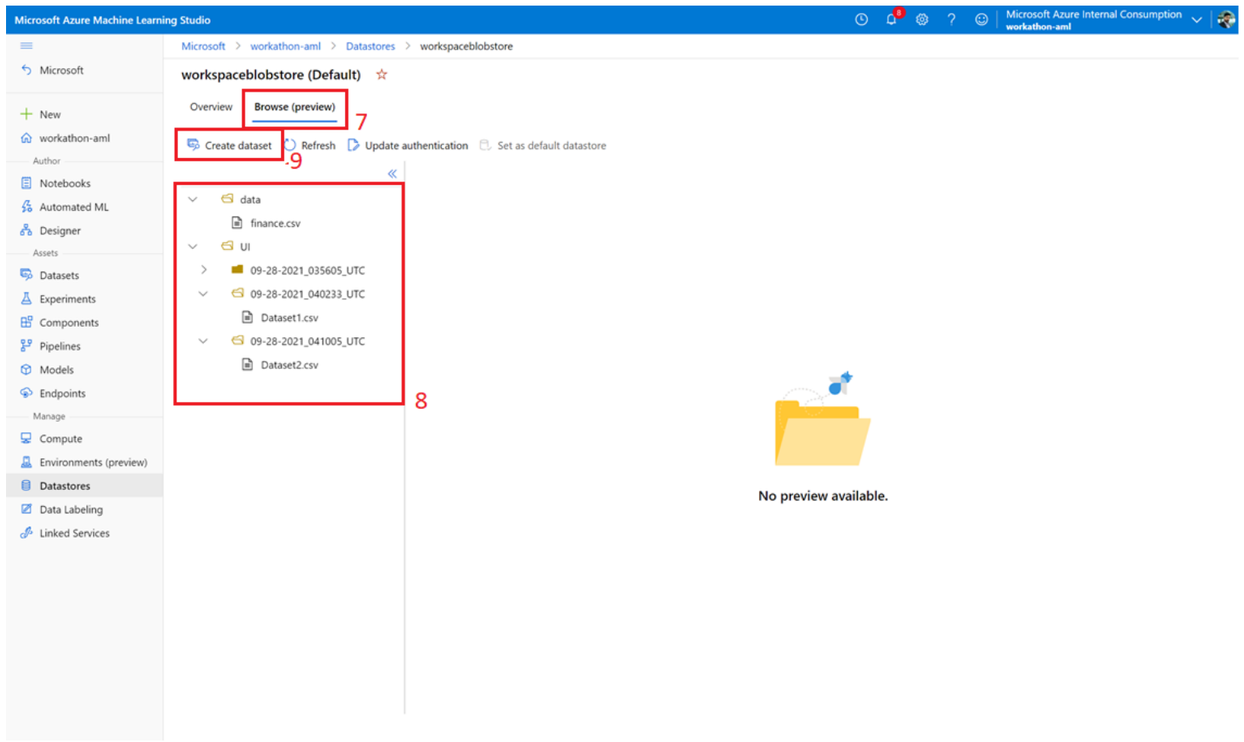
Let's explore how to create a new datastore –

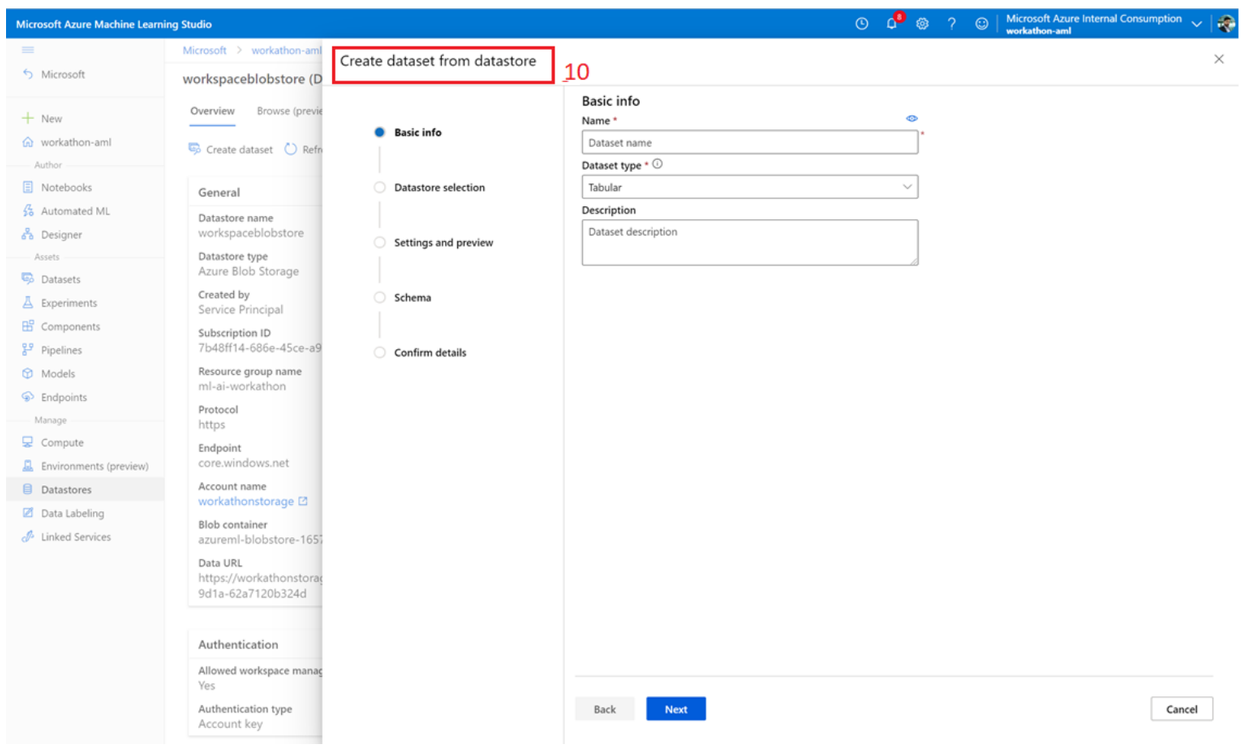
1. Go to Datastores tab
2. Select + New datastore
3. In the 'New datastore' wizard, give your datastore a name & select the Datastore type. You will need to enter the authentication information to connect to the Datastore basis the Datastore type. For instance, Connection string for blob storage, User ID & Password for databases etc
   You can select the resource & authenticate using 2 methods :
   a. From Azure subscription – This method used AAD and allows you to register any resource that you have access you.
   b. Manually – This method requires you to enter the connection details for the datastore. For instance, URL & subscription ID for blob storage, Server name & Database name for databases etc.
4. Once you have entered the details, click create to register the Datastore.

Explore an existing datastore

Let's explore the details we can view for any Datastore that we provision.

5. Click on the 'workspaceblobstore (Default)'. This is the default blob storage account for AML.
6. In the overview tab, you can view the metadata for the datastore such as name, type, created by, endpoint details & URLs etc.

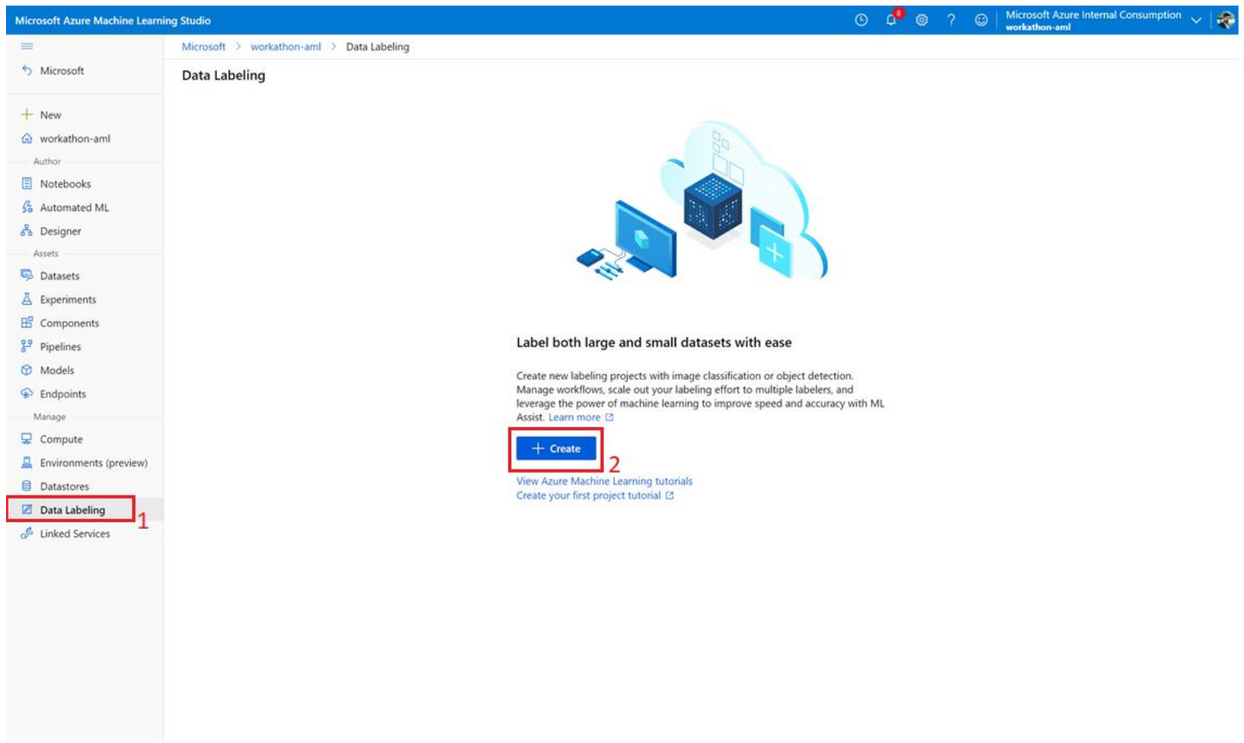


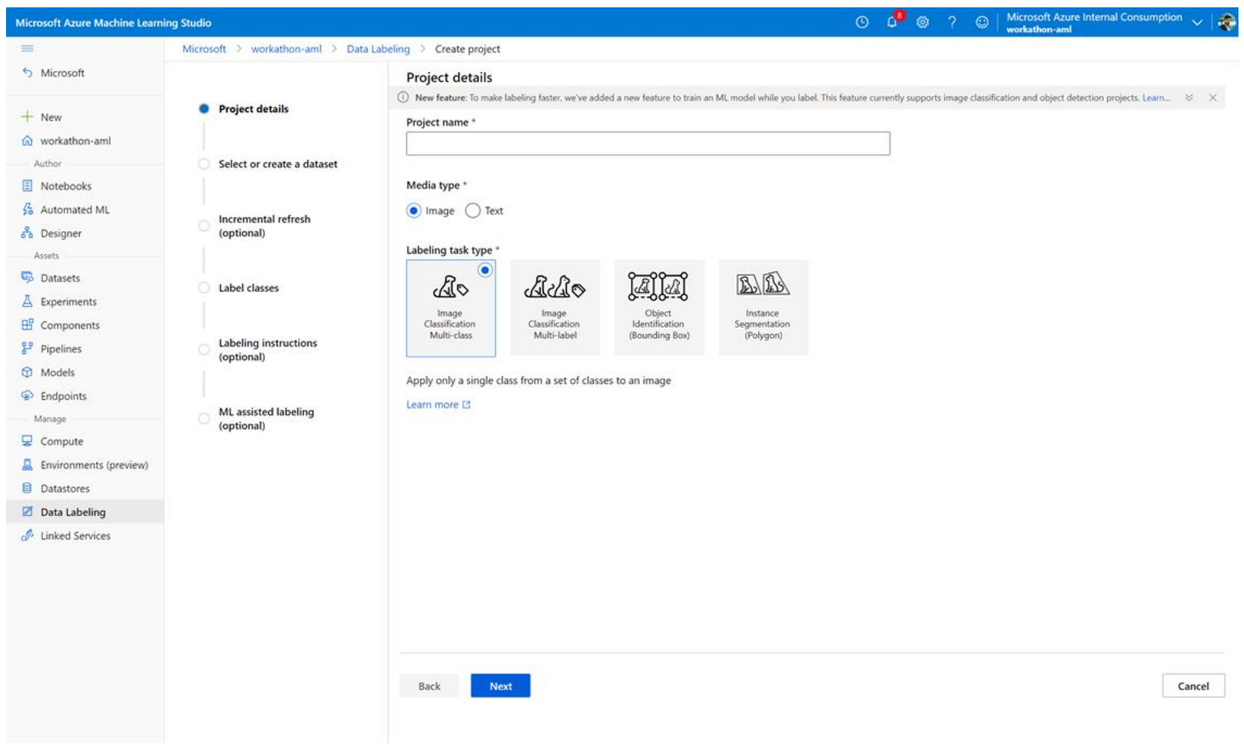


7. Go to the Browse (preview) tab.
8. This tab gives you a view of the data that is stored in the selected Datastore. Notice that you can view the datasets that we've created in the workshop.
9. You can also create a new dataset in this datastore using the 'Create dataset' option.

10. This will open the 'Create dataset from datastore' wizard that we are familiar with. The dataset details will be pre-configured for you.

Explore the metadata saved for any new Datastore that you create.





Explore Data Labeling

The Data Labeling feature of AML allows you to streamline the process of labelling your training data for tasks like Text Classification, Image Classification & Object Detection.
This is helpful when you have a lot of unlabelled data and want to leverage it to train supervised learning ML Models. The model keeps learning while you train and labels the rest of the images / text items for you. You can quickly verify the labels provided and choose to keep them or change them in case they are labelled incorrectly.

This feature helps you expedite the data labelling process, which is an integral part of preparing data for training ML Models.

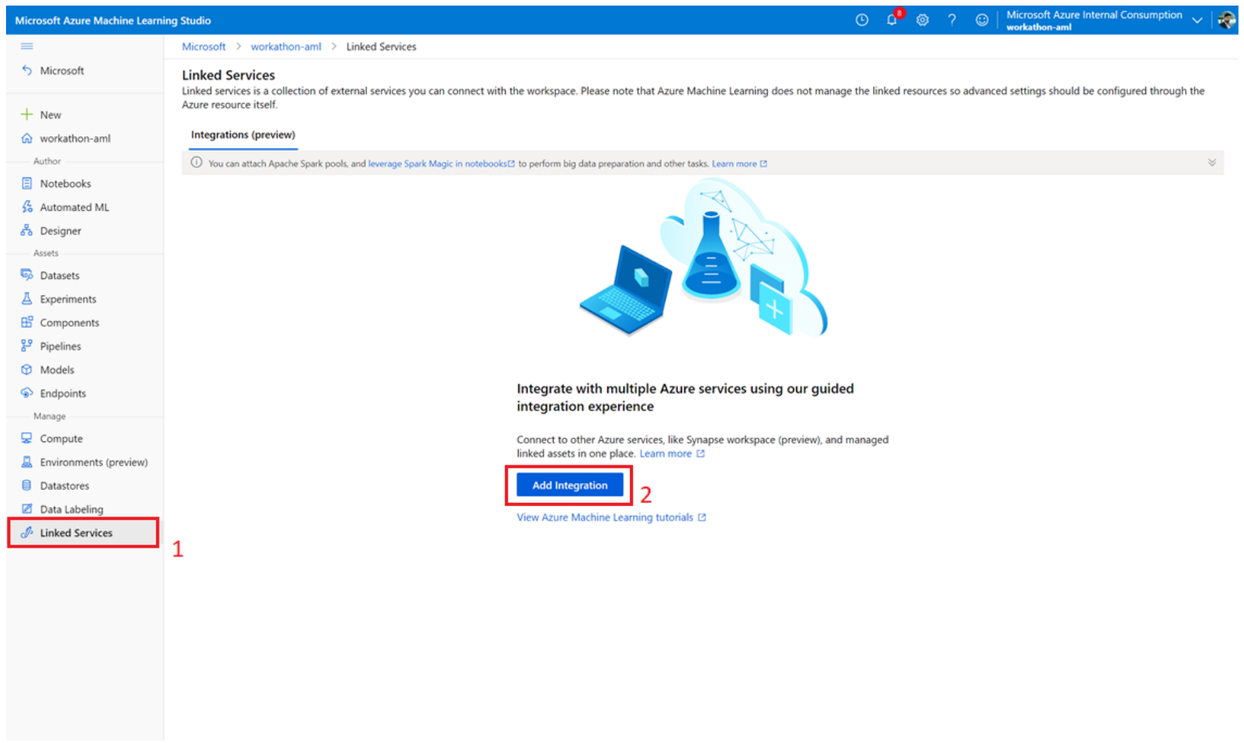
You can perform the following tasks :

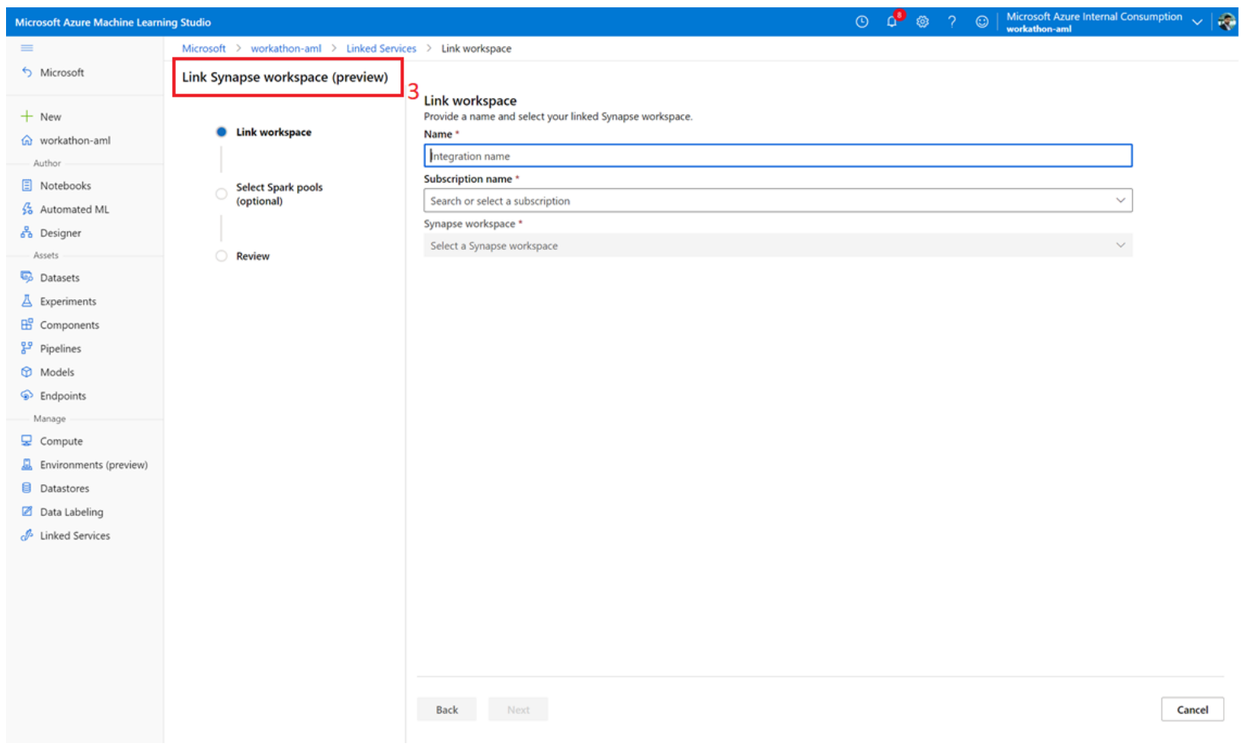
a. Image Related –
   Image Classification Multi-class
   Image Classification Multi-label
   Object Identification (Bounding Box)
   Object Identification (Polygon)

b. Text Related –
   Text Classification Multi-class
   Text Classification Multi-label

We will not be creating an end-to-end Data labeling project, however, let's explore a bit to help you get started :

1. Go to data Labeling tab
2. Click + Create
3. Just like any other project, you will need to configure the Project details page with information like the Type of task you want to perform, the dataset, labelling classes etc.





Explore Linked Services

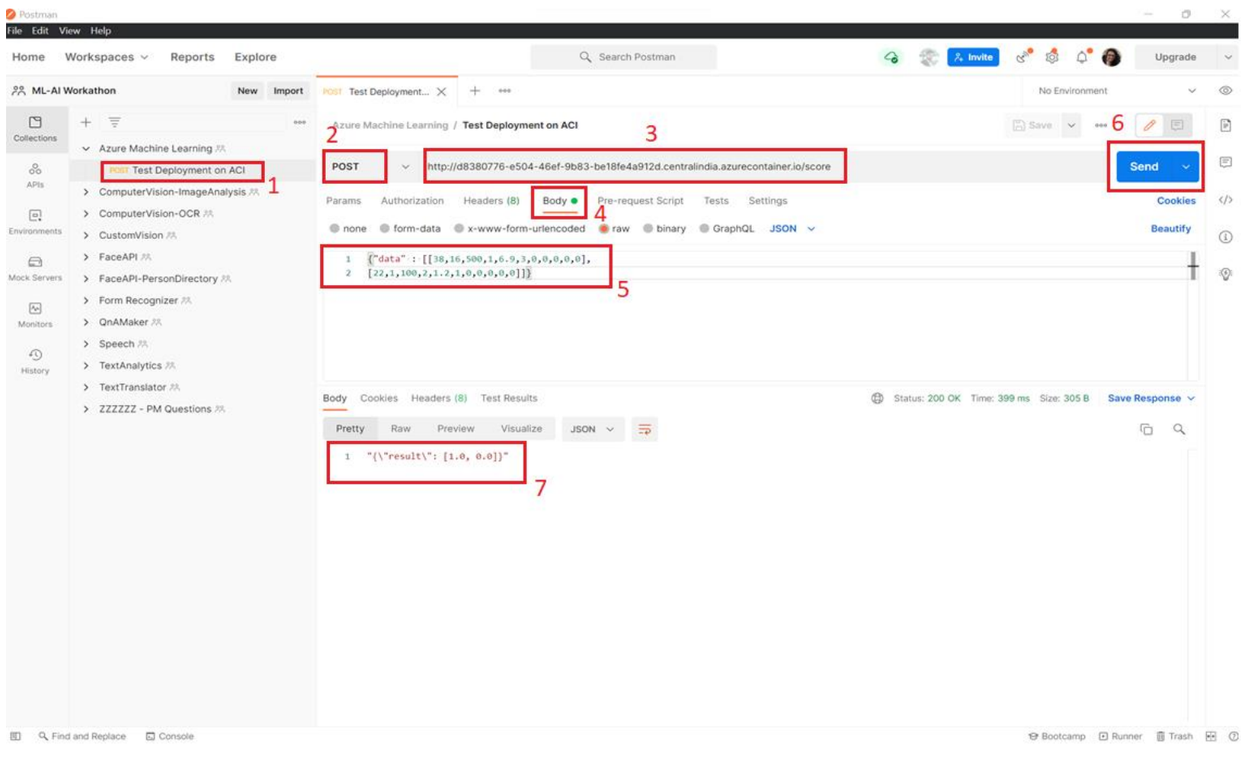
Like we learnt earlier, AML allows you to leverage Compute resources such as Databricks, HDInsights, Synapse etc that are external to AML.

The Linked Services tab allows you to attach Apache Spark pools provisioned in your Synapse Analytics workspace to your AML workspace and leverage the power of Spark for data preparation, model training and other tasks.

We will not be running a spark experiment in this workshop. This is shared as homework! However, let's explore how to link a Synapse resource to help you get started –

1. Go to Linked Services tab
2. Click 'Add Integration'

3. In the 'Link Synapse Workspace (preview), select the Synapse Workspace and Spark Pool that you want to attach to this workspace.



Call inference API from Postman

We will now switch the interface to Postman, to make the real time API call to the custom model we built using Azure Machine Learning.

Method : POST

URL : Copy the URL for your model endpoint from the 'Endpoints' tab

Body :
{"data" : [[38,16,500,1,6.9,3,0,0,0,0,0],
[22,1,100,2,1.2,1,0,0,0,0,0]]}

You can try adding additional datapoints using the same format.

Headers :
We did not enable authentication when we deployed the endpoint. However, in a production scenario, you will add the authentication parameter in the Headers.

Result :
You will receive an array with 2 inference values, since we passed 2 data input points.

Homework

1. Create an AutoML Classification experiment on the dataset you curated. [You can refer this tutorial]
2. Create a pipeline using AML Designer. Deploy this both as a real time endpoint and batch endpoint. [You can refer this tutorial]
3. If you are familiar with Synapse & Spark, attach a Spark pool to your AML resource and try executing some Spark commands from within AML. [You can refer this tutorial]

Additional recommended resources

Service Name	Category	Links
Machine Learning	Programming Language	R, Python, .NET
	Tiers	Refer the pricing Link
	Pricing	https://azure.microsoft.com/en-in/pricing/details/machine-learning/
	Limits	https://docs.microsoft.com/en-us/azure/machine-learning/resource-limits-quotas-capacity
	Language Support	Not Applicable
	Sample Apps	Examples
	Regional Availability & Support	https://azure.microsoft.com/en-us/global-infrastructure/services/?products=machine-learning-service&regions=all
	SLAs for Cognitive Services	https://azure.microsoft.com/en-in/support/legal/sla/machine-learning-service/v1_0/
	Compliance & Certificates	https://docs.microsoft.com/en-us/azure/machine-learning/security-controls-policy
	Machine Learning Services Updates	https://azure.microsoft.com/en-us/updates/?product=cognitive-services&category=ai-machine-learning

Best Practices

Azure Machine Learning Documentation

GitHub Labs for Azure Machine Learning

Use visual tools to create ML models with AML

Foundations of data science for ML

Understand data science for ML

Create machine learning models