

# Task 3 P: Design and Deploy Model using Azure Machine Learning

## Part 1:

### What is the Azure Machine Learning SDK for Python?

Azure Machine Learning Python SDK is a library for Python that allows users to build, train, deploy and manage machine learning models on Microsoft Azure. It provides different packages and APIs for working with Azure Machine Learning services. The SDK allows users to:

- Build, train and deploy machine learning and deep learning models
- Able to run discrete ML activity as a job that can be run locally or over the cloud
- Provides scalability and performance optimizations for training and deploying ML models
- Provides ease of monitoring and managing model performance, versions and more.

### What is Azure Machine Learning Workspace?

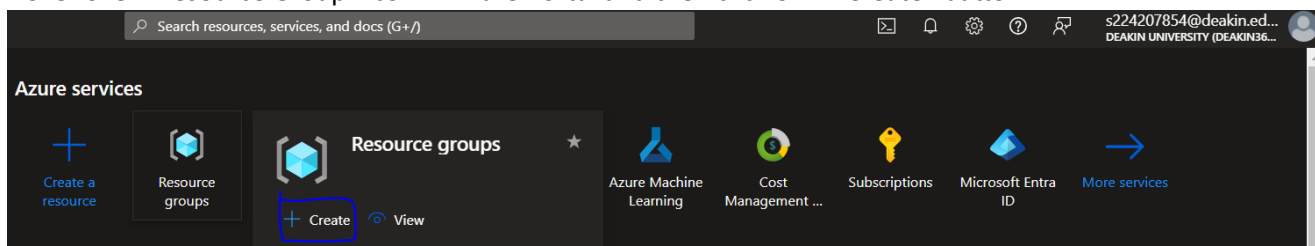
Azure Machine Learning workspace is a central place to manage resources and assets for training and managing models. The workspace keeps a history of all jobs, including logs, metrics, output and a snapshot of your scripts. It also stores references like data, models, environments and more. In workspace we can perform different tasks like:

- Jobs creation for training runs, which used to build models.
- Helps in managing data used for model training and pipelines.

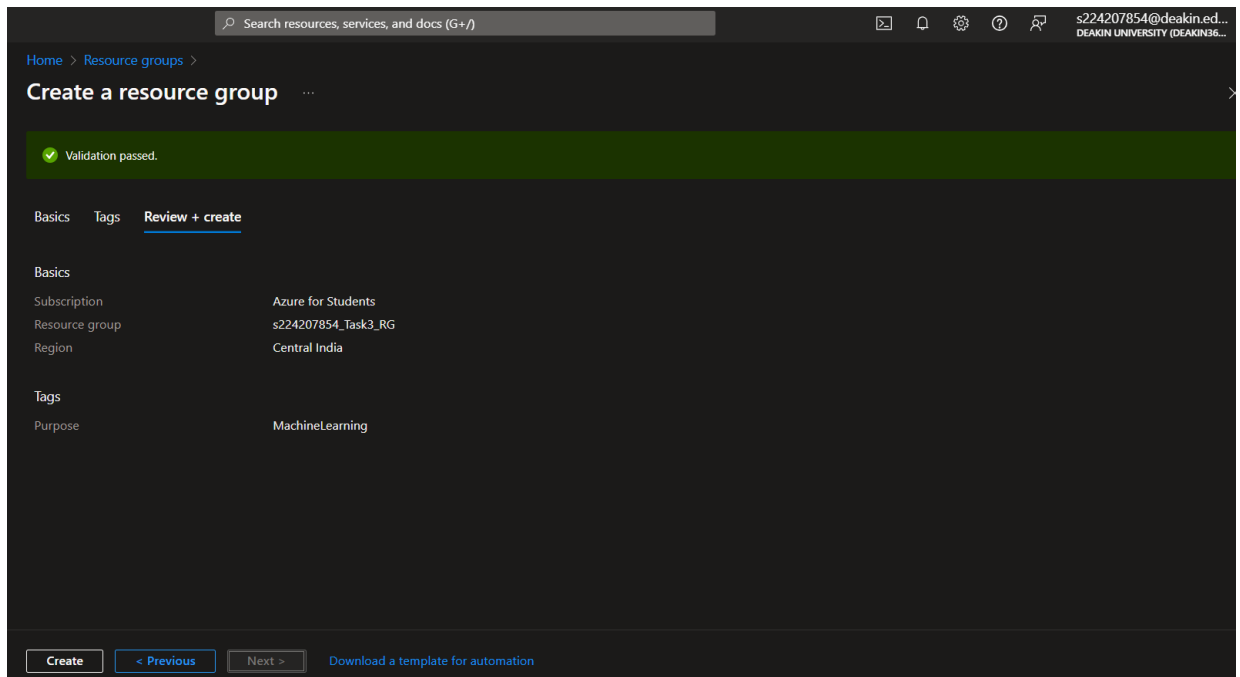
### How to create a workspace for machine learning using Azure Portal?

Steps to create Azure ML workspace using Azure Portal:

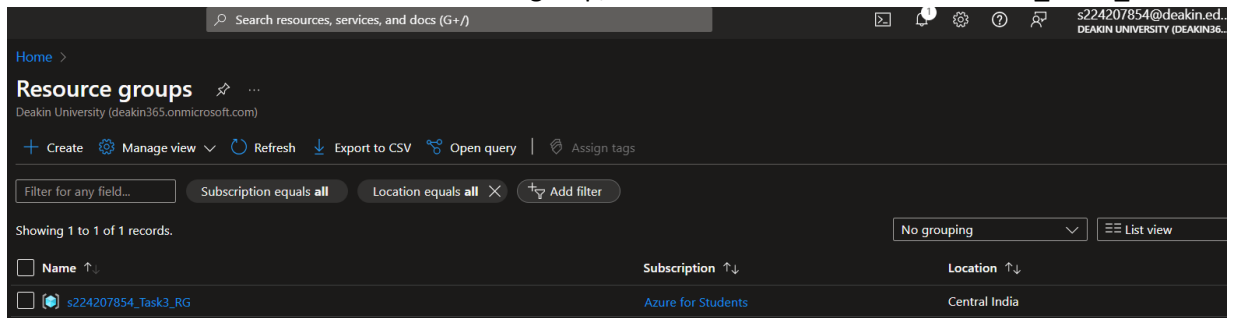
1. Sign in to Azure portal with your Deakin creds. Link -> [portal.azure.com](https://portal.azure.com)
2. Create a new resource group, using below steps:
  - a. Hover over “Resource Group” icon in Azure Portal and then click on “+ Create” button.



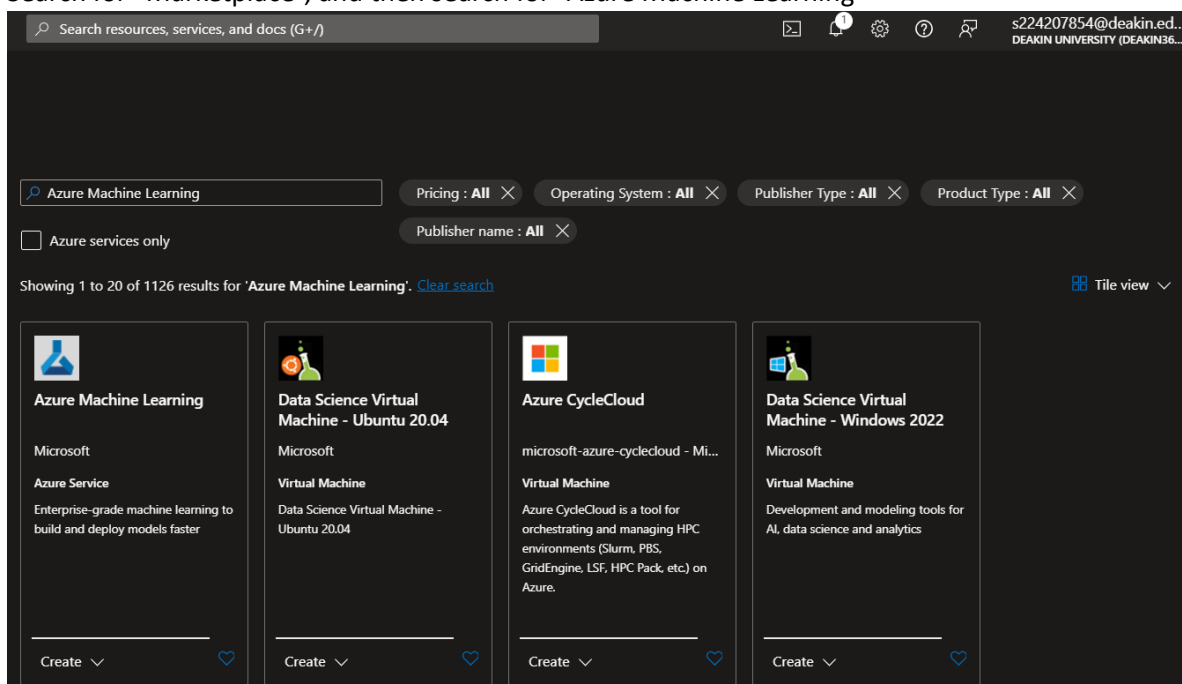
- b. Fill the required information for resource creation and before creating, please review, will be looking like below, after which click on “Create” button.



- c. You will be able to see created the resource group, here we have created “s224207854\_Task3\_RG”.



3. Search for “Marketplace”, and then search for “Azure Machine Learning”



4. Click on “Create” button in Azure Machine Learning, which will redirect to this page.

Home > Marketplace >

## Azure Machine Learning

Create a machine learning workspace

Basics Networking Encryption Identity Tags Review + create

Resource details

Every workspace must be assigned to an Azure subscription, which is where billing happens. You use resource groups like folders to organize and manage resources, including the workspace you're about to create. [Learn more about Azure resource groups](#)

Subscription \*

Resource group \*

Workspace details

Configure your basic workspace settings like its storage connection, authentication, container, and more. [Learn more](#)

Name \*

Region \*

Storage account \*

Key vault \*

[Review + create](#) < Previous Next : Networking

5. Once required details has been filled, please go for “Review + create” button and “Create” the workspace, here we have created with name “s224207854\_Task3\_WS”

Home >

## Microsoft.MachineLearningServices | Overview

Deployment

Search << Delete Cancel Redeploy Download Refresh

Overview Inputs Outputs Template

✓ Your deployment is complete

Deployment name: Microsoft.MachineLearningSe... Start time: 29/3/2024, 5:27:05 pm  
Subscription: Azure for Students Correlation ID: 9b6d9a1a-9f78-42c5-826b-19ff477f2c7d  
Resource group: s224207854\_Task3\_RG

Deployment details

Next steps

[Go to resource](#)

Cost Management  
Get notified to stay within your budget and prevent unexpected charges on your bill. [Set up cost alerts >](#)

6. Click on “Go to resource” and your newly created workspace can be viewed

Home > Microsoft.MachineLearningServices | Overview >

## s224207854\_Task3\_WS

Azure Machine Learning workspace

Search << Download config.json Delete

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Events

Essentials

Resource group : s224207854\_Task3\_RG Studio web URL : <https://ml.azure.com?tid=d02378ec-1688-46d5-8540...>

Location : Central India Container Registry : ...

Subscription : Azure for Students Key Vault : s224207854task0121527534

Storage : s224207854task8525422840 Application Insights : s224207854task8379656913

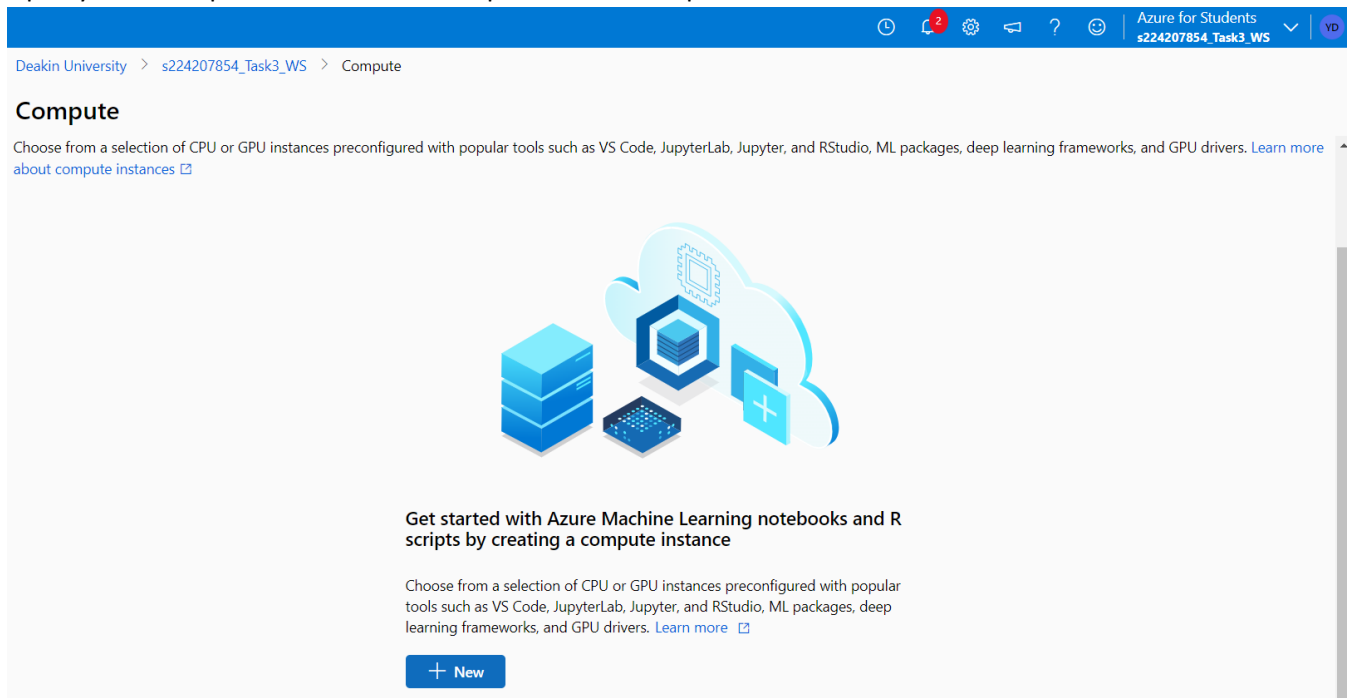
MLflow tracking URI : azureml://centralindia.api.azureml.ms/mlflow/v1.0/su...

What is compute instance and how to create compute instance using Azure Portal?

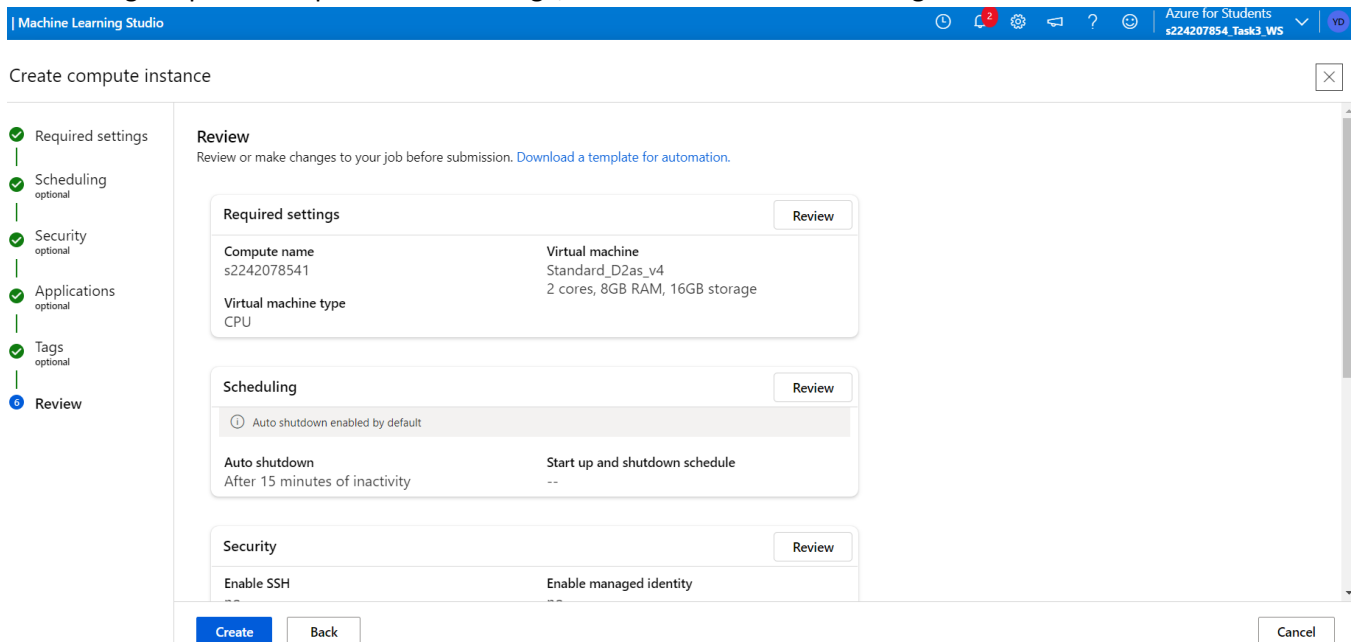
Compute Instance in Azure Machine Learning is a virtual machine which provides a development environment for users and we can start running sample notebooks without needing to set up anything.

Step to create compute instance:

- Open your workspace and click on “Compute” with “Compute instances” as tab. Click on “+ New” button.



- Use settings as per the requirement and usage, after which review the settings and then click on “Create”.



- Once deployment is completed we can check our instance status, here our instance name is “s2242078541”.

Machine Learning Studio

Azure for Students  
s224207854\_Task3\_WS

Deakin University > s224207854\_Task3\_WS > Compute

## Compute

The "Kubernetes clusters" tab is now where you can access previous versions of "inference clusters" (also known as "AKS clusters") and "attached Kubernetes" compute types along with any previously created compute targets using those types. [Learn more](#) about Kubernetes clusters.

Compute instances

Compute clusters

Kubernetes clusters

Attached computes

Choose from a selection of CPU or GPU instances preconfigured with popular tools such as VS Code, JupyterLab, Jupyter, and RStudio, ML packages, deep learning frameworks, and GPU drivers. [Learn more about compute instances](#)

+ New

Refresh

Start

Stop

Restart

Schedule and idle shutdown

Delete

Reset view

View quota

Search

Filter

Columns

Name	☆	State	Idle shutdown ⓘ	Applications ⓘ	Size	Created on ↓	Assigned to
s2242078541		Running	15 minutes	<a href="#">JupyterLab</a> <a href="#">Jupyter</a> <a href="#">VS Code (Web)</a> <span>PREVIEW</span> <span>...</span>	Standard_F4s_v2	Mar 30, 2024 3:02 PM	YATHARTH DEOLY

## Part 2:

### Introduction

In this case study we will be using python knowledge with Azure Machine Learning to train and deploy the model. Dataset that will be used here is of “**Chronic Kidney Disease**”. 400 records have been provided in this dataset.

This dataset contains 25 columns, i.e.,

- **Age:** This column contains numerical data and units is in years with missing values. ColumnName: ‘age’.
- **Blood Pressure:** This column contains numerical data and unit is in mm/Hg with missing values. ColumnName: ‘bp’.
- **Specific Gravity:** This column contains categorical data with unique values as: (1.005,1.010,1.015,1.020,1.025) and some missing values. ColumnName: ‘sg’.
- **Albumin:** This column contains categorical data with unique values as: (0,1,2,3,4,5) and some missing values. ColumnName: ‘al’.
- **Sugar:** This column contains categorical data with unique values as: (0,1,2,3,4,5) and some missing values. ColumnName: ‘su’.
- **Red Blood Cells:** This column contains categorical data with unique values as: (normal, abnormal) and some missing values. ColumnName: ‘rbc’.
- **Pus Cell:** This column contains categorical data with unique values as: (normal, abnormal) and some missing values. ColumnName: ‘pc’.
- **Pus Cell clumps:** This column contains categorical data with unique values as: (present, notpresent) and some missing values. ColumnName: ‘pcc’.
- **Bacteria:** This column contains categorical data with unique values as: (present, notpresent) and some missing values. ColumnName: ‘ba’.
- **Blood Glucose Random:** This column contains numerical data and units is in mgs/dl with missing values. ColumnName: ‘bgr’.
- **Blood Urea:** This column contains numerical data and units is in mgs/dl with missing values. ColumnName: ‘bu’.
- **Serum Creatinine:** This column contains numerical data and units is in mgs/dl with missing values. ColumnName: ‘sc’.
- **Sodium:** This column contains numerical data and units is in mEq/L with missing values. ColumnName: ‘sod’.
- **Potassium:** This column contains numerical data and units is in mEq/L with missing values. ColumnName: ‘pot’.
- **Hemoglobin:** This column contains numerical data and units is in gms with missing values. ColumnName: ‘hemo’.
- **Packed Cell Volume:** This column contains numerical data with missing values. ColumnName: ‘pcv’.
- **White Blood Cell Count:** This column contains numerical data and units is in cells/cmm with missing values. ColumnName: ‘wbcc’.
- **Red Blood Cell Count:** This column contains numerical data and units is in millions/cmm with missing values. ColumnName: ‘rbcc’.
- **Hypertension:** This column contains categorical data with unique values as: (yes, no) and some missing values. ColumnName: ‘htn’.
- **Diabetes Mellitus:** This column contains categorical data with unique values as: (yes, no) and some missing values. ColumnName: ‘dm’.
- **Coronary Artery Disease:** This column contains categorical data with unique values as: (yes, no) and some missing values. ColumnName: ‘cad’.
- **Appetite:** This column contains categorical data with unique values as: (good, poor) and some missing values. ColumnName: ‘appet’.

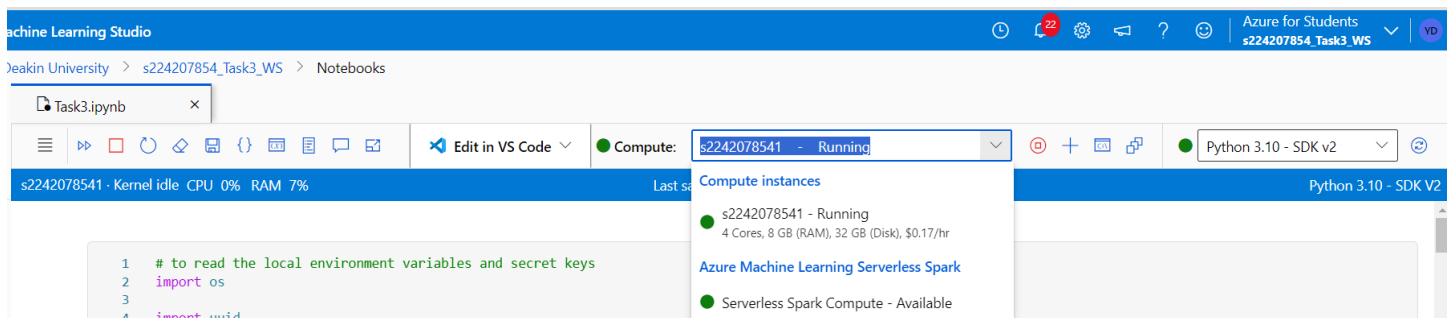
- **Pedal Edema:** This column contains categorical data with unique values as: (yes, no) and some missing values. ColumnName: 'pe'.
- **Anemia:** This column contains categorical data with unique values as: (yes, no) and some missing values. ColumnName: 'ane'.
- **Class:** This column contains categorical data with unique values as: (ckd, notckd) and some missing values. ColumnName: 'class'.

Brief information about dataset:

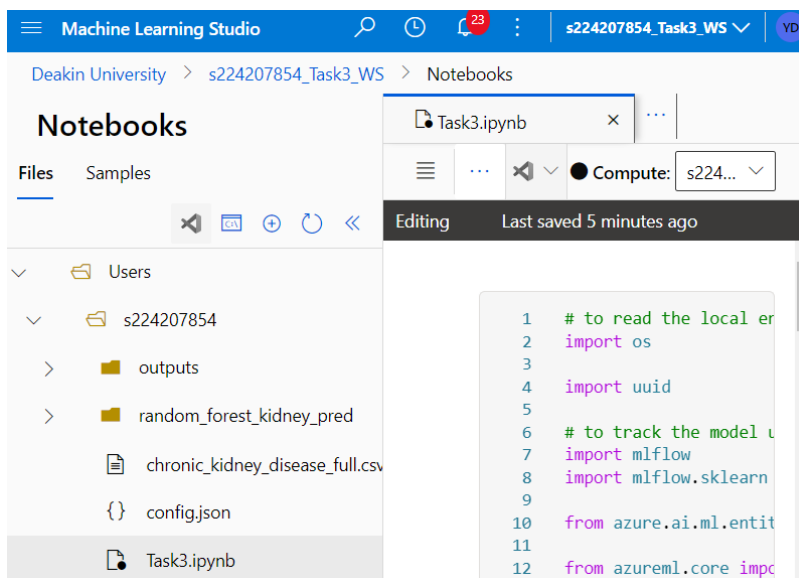
- Dataset contains missing values which has been denoted as '?'.
- Dataset have 24 features and 1 target class. These categories have been divided into as 11 numerical and 14 nominal.

## Azure Notebook

To do this task we will be using Azure ML notebook, which can be created from “Notebooks” section. Once notebook is created, we need to have a compute instance for our notebook and setting up our kernel as “Python 3.10 – SDK v2”

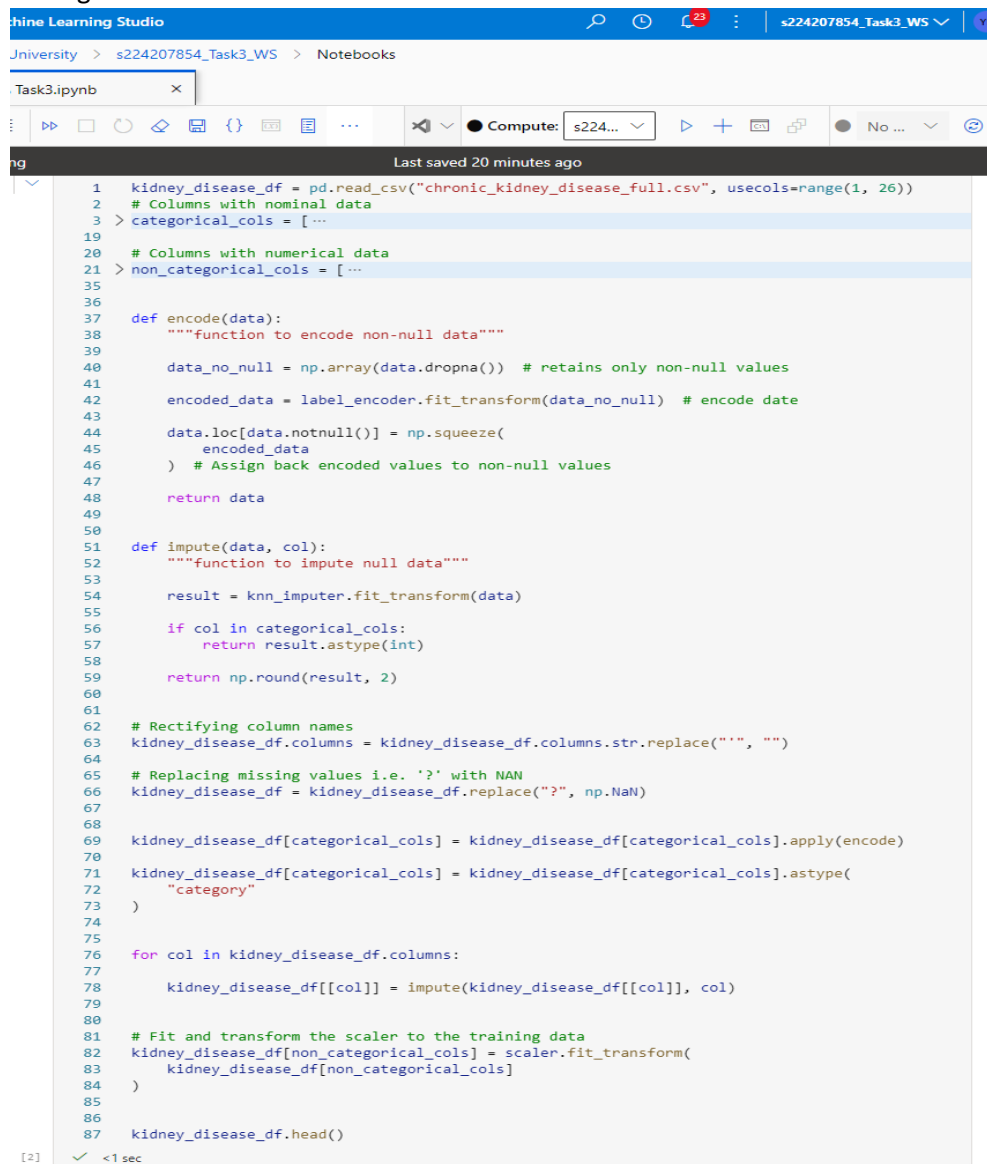


In notebook section, we can upload different resources that will be used in our case study.



## Data Loading and Data Pre-processing

- Data has been cleansed and ready for future analytics and modelling use, by using different ML techniques.
- Different steps followed here are:
  - Extracting data into pandas Dataframe.
  - Rectifying column names, as col names contains quotes, which needs to be removed.
  - Replacing missing values i.e. '?' with NAN.
  - As we have 14 categorical data that needs to be encoded, so for this we will be using 'LabelEncoder' technique from 'sklearn' package. This is a technique that converts categorical variables into numerical values.
  - As we have ample missing data in every column, so for that we will be using 'KNNImputer' technique from 'sklearn' package. This is a scikit-learn class that uses the K-Nearest Neighbors (KNN) algorithm to predict or fill in missing values in a dataset. It's a multivariate technique that considers multiple features in the dataset to estimate the missing values.
  - As we have 11 non categorical data whose values ranges a lot, so to normalize we will be using 'StandardScaler' technique from 'sklearn' package. It normalizes features by removing the mean and scaling them to unit variance.

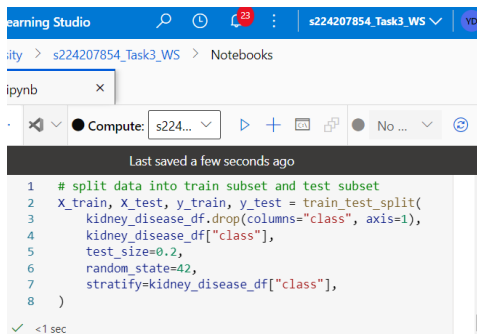


```
1 kidney_disease_df = pd.read_csv("chronic_kidney_disease_full.csv", usecols=range(1, 26))
2 # Columns with nominal data
3 > categorical_cols = [...]
19
20 # Columns with numerical data
21 > non_categorical_cols = [...]
35
36
37 def encode(data):
38     """function to encode non-null data"""
39
40     data_no_null = np.array(data.dropna()) # retains only non-null values
41
42     encoded_data = label_encoder.fit_transform(data_no_null) # encode date
43
44     data.loc[data.notnull()] = np.squeeze(
45         encoded_data
46     ) # Assign back encoded values to non-null values
47
48     return data
49
50
51 def impute(data, col):
52     """function to impute null data"""
53
54     result = knn_imputer.fit_transform(data)
55
56     if col in categorical_cols:
57         return result.astype(int)
58
59     return np.round(result, 2)
60
61
62 # Rectifying column names
63 kidney_disease_df.columns = kidney_disease_df.columns.str.replace("'", "")
64
65 # Replacing missing values i.e. '?' with NAN
66 kidney_disease_df = kidney_disease_df.replace("?", np.NaN)
67
68
69 kidney_disease_df[categorical_cols] = kidney_disease_df[categorical_cols].apply(encode)
70
71 kidney_disease_df[categorical_cols] = kidney_disease_df[categorical_cols].astype(
72     "category"
73 )
74
75
76 for col in kidney_disease_df.columns:
77     kidney_disease_df[[col]] = impute(kidney_disease_df[[col]], col)
78
79
80
81 # Fit and transform the scaler to the training data
82 kidney_disease_df[non_categorical_cols] = scaler.fit_transform(
83     kidney_disease_df[non_categorical_cols]
84 )
85
86
87 kidney_disease_df.head()
```



## Data Split

Now we will be splitting our dataframe into 80% training dataset and 20% as testing dataset. To achieve this, we will be using `train\_test\_split` from `sklearn` package. This is a function in the scikit-learn library that splits a dataset into two sets: a training set and a test set. The training set is used to fit a machine learning model, while the test set is used to evaluate the model's performance.



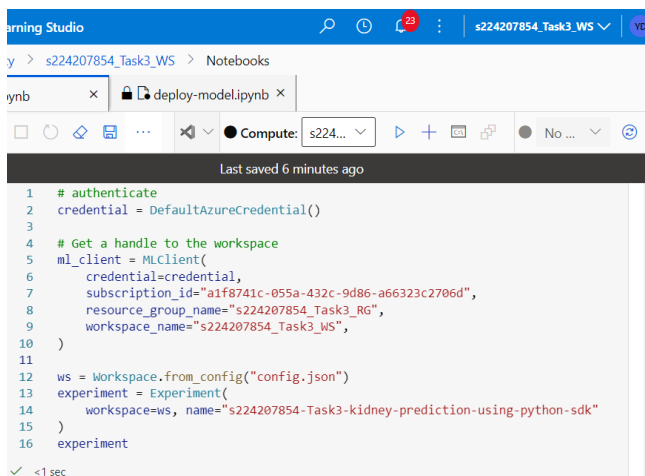
The screenshot shows the Azure ML Studio interface with a notebook titled 's224207854\_Task3\_WS'. The code in the notebook is as follows:

```
1 # split data into train subset and test subset
2 X_train, X_test, y_train, y_test = train_test_split(
3     kidney_disease_df.drop(columns="class", axis=1),
4     kidney_disease_df["class"],
5     test_size=0.2,
6     random_state=42,
7     stratify=kidney_disease_df["class"],
8 )
```

The code is executed successfully, with a status bar indicating '<1 sec'.

## Prerequisites for Azure ML

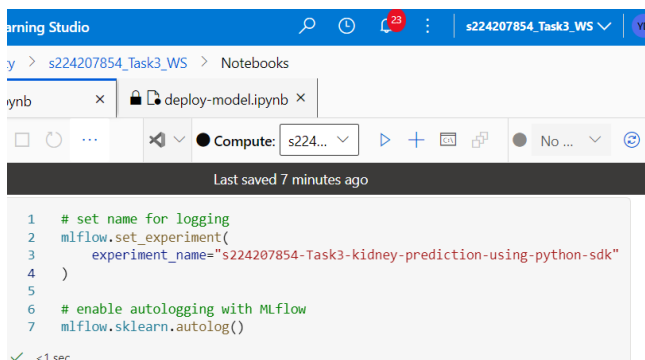
Now will be declaring MLClient which will help in managing resources, jobs and ml-flow. Once workspace is created will create an experiment, which is another foundational cloud resource that represents a collection of trials (individual model runs). Once experiment is set, will be enabling auto logging with MLflow.



The screenshot shows the Azure ML Studio interface with a notebook titled 's224207854\_Task3\_WS'. The code in the notebook is as follows:

```
1 # authenticate
2 credential = DefaultAzureCredential()
3
4 # Get a handle to the workspace
5 ml_client = MLClient(
6     credential=credential,
7     subscription_id="a1f8741c-055a-432c-9d86-a66323c2706d",
8     resource_group_name="s224207854_Task3_RG",
9     workspace_name="s224207854_Task3_WS",
10 )
11
12 ws = Workspace.from_config("config.json")
13 experiment = Experiment(
14     workspace=ws, name="s224207854-Task3-kidney-prediction-using-python-sdk"
15 )
16 experiment
```

The code is executed successfully, with a status bar indicating '<1 sec'.



The screenshot shows the Azure ML Studio interface with a notebook titled 's224207854\_Task3\_WS'. The code in the notebook is as follows:

```
1 # set name for logging
2 mlflow.set_experiment(
3     experiment_name="s224207854-Task3-kidney-prediction-using-python-sdk"
4 )
5
6 # enable autologging with MLflow
7 mlflow.sklearn.autolog()
```

The code is executed successfully, with a status bar indicating '<1 sec'.

## Registering and creation of model

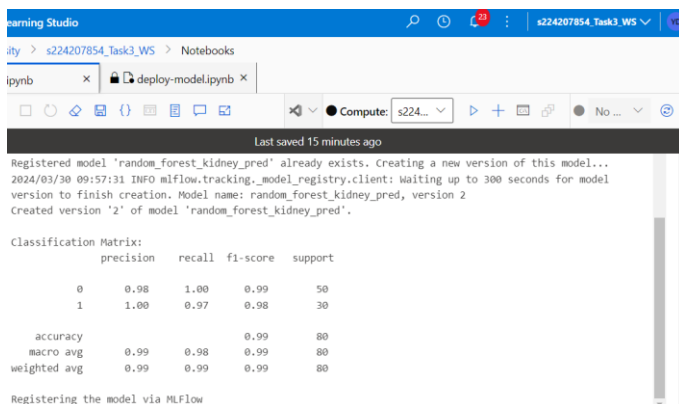
As our best model, will be setting up RandomForestClassifier with different set of hyperparameters. Using sklearn and mlflow libraries, will be running our ML model and will be showing report of our model. Once model is ready, will register that model to our Azure ML portal. Model will be registered using MLflow.



```
1 mlflow.start_run()
2 run = experiment.start_logging()
3
4 random_forest_classifier = RandomForestClassifier(
5     criterion="gini", max_depth=7, n_estimators=20, random_state=42
6 )
7
8 run.log("num_samples", kidney_disease_df.shape[0])
9 run.log("num_features", kidney_disease_df.shape[1] - 1)
10 run.log("criterion", "gini")
11
12 # fit the model using fit() on train data
13 random_forest = random_forest_classifier.fit(X_train, y_train)
14 y_pred = random_forest.predict(X_test)
15
16 print("\nClassification Matrix:\n", classification_report(y_test, y_pred))
17
18 # Logging all metrics of classification_report
19 cr = classification_report(y_test, y_pred, output_dict=True)
20
21 run.log("accuracy", cr.pop("accuracy"))
22
23 for class_or_avg, metrics_dict in cr.items():
24     for metric, value in metrics_dict.items():
25         run.log(class_or_avg + "_" + metric, value)
26
27 model_name = "random_forest_kidney_pred"
28
29 print("Registering the model via MLFlow")
30 mlflow.sklearn.log_model(
31     sk_model=random_forest_classifier,
32     registered_model_name=model_name,
33     artifact_path=model_name,
34 )
35
36 # Saving the model to a file
37 mlflow.sklearn.save_model(
38     sk_model=random_forest_classifier,
39     path=os.path.join(model_name, "trained_model"),
40 )
41
42 run.complete()
43
44 mlflow.end_run()
```

✓ 17 sec

## Model output



```
Registered model 'random_forest_kidney_pred' already exists. Creating a new version of this model...
2024/03/30 09:57:31 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model
version to finish creation. Model name: random_forest_kidney_pred, version 2
Created version '2' of model 'random_forest_kidney_pred'.
```

Classification Matrix:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	50
1	1.00	0.97	0.98	30
accuracy			0.99	80
macro avg	0.99	0.98	0.99	80
weighted avg	0.99	0.99	0.99	80

Registering the model via MLFlow

Once model is registered, we can check same over Azure Portal in “Model List”.

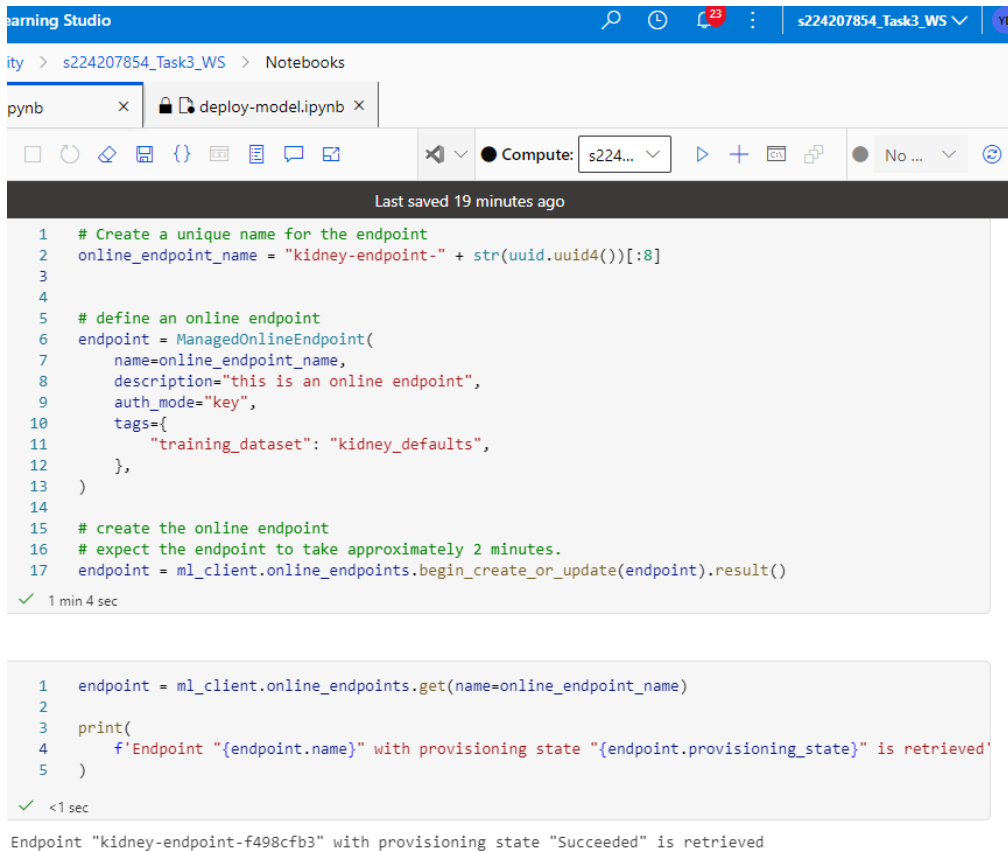
The screenshot shows the Azure Portal interface for a registered model. The breadcrumb navigation is: Deakin University > s224207854\_Task3\_WS > Models > random\_forest\_kidney\_pred:2. The model name is 'random\_forest\_kidney\_pred:2' with a star icon. Below the name are tabs: Details (selected), Versions, Artifacts, Endpoints, Jobs, Data, Feature sets, Responsible AI, Explanations (preview), and Fairness (preview). There are action buttons: Refresh, Archive, Deploy, Download all, and Share model. The 'Attributes' section on the left lists: Name (random\_forest\_kidney\_pred), Version (2), Created on (Mar 30, 2024 3:27 PM), Created by (YATHARTH DEOLY), Type (MLFLOW), Created by job (058b3712-1363-46f6-8334-7a6e74c17bdf), and Asset ID (azureml://locations/centralindia/workspaces/3d47cddc-6c4f-44be-af55-100fec43efed/models/random\_forest\_kidney\_pred/versions/2). The 'Tags' section shows 'No tags'. The 'Properties' section contains JSON-like data for azureml.artifactPrefix, azureml.storagePath, flavors (python\_function, sklearn), flavors.python\_function (model path, predict fn, loader module, python version, env, conda yaml, virtualenv), and flavors.sklearn (pickled model, sklearn version, serialization format, code).

Model metrics can also be checked In Azure portal inside jobs section as we have logged all the metrics.

The screenshot shows the Azure Portal interface for a job named 'g\_screw\_nxwt41rp'. The breadcrumb navigation is: University > s224207854\_Task3\_WS > Jobs > s224207854-Task3-kidney-prediction-using-python-sdk > boring\_screw\_nxwt41rp. The job name is 'g\_screw\_nxwt41rp' with a link icon, a star icon, and a green checkmark indicating it is 'Completed'. Below the job name are tabs: Metrics (selected), Images, Child jobs, Outputs + logs, Code, Explanations (preview), Fairness (preview), and Monitoring. The 'Metrics' section shows a grid of metrics. The top row includes: 0\_f1-score (0.9900990), 0\_precision (0.9803922), 0\_recall (1), 0\_support (50), 1\_f1-score (0.9830508), 1\_precision (1), and 1\_recall (0.9666667). The second row includes: 1\_support (30), accuracy (0.9875), criterion (gini), macro avg\_f1-score (0.9865749), macro avg\_precision (0.9901961), macro avg\_recall (0.9833333), and macro avg\_support (80). The third row includes: num\_features (24), num\_samples (400), weighted avg\_f1-score (0.9874559), weighted avg\_precision (0.9877451), weighted avg\_recall (0.9875), and weighted avg\_support (80). Each metric card has a refresh icon and a trash icon.

## Endpoint Creation

After model registration, we will be creating an endpoint which will be used for model deployment. Here we will be creating dynamic endpoint using 'uuid'. Once endpoint is up and running, we can check the status via coding way or checking over Azure ML portal.

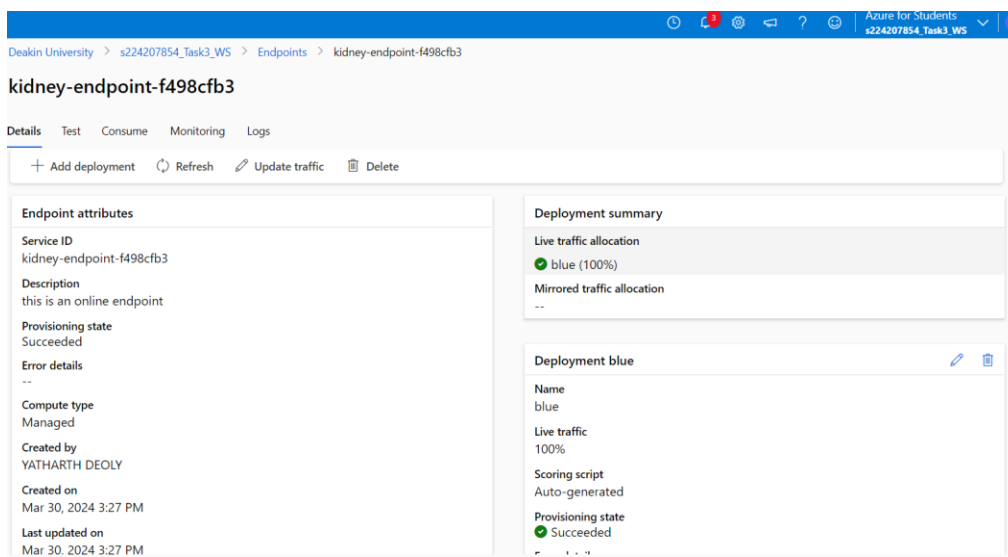


The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains code to create a unique endpoint name using a UUID, define an online endpoint with specific attributes, and then create it. The second cell contains code to retrieve the endpoint by name and print its provisioning state. The output of the first cell shows the endpoint was created successfully in 1 minute and 4 seconds. The output of the second cell shows the endpoint's provisioning state is 'Succeeded'.

```
1 # Create a unique name for the endpoint
2 online_endpoint_name = "kidney-endpoint-" + str(uuid.uuid4())[:8]
3
4
5 # define an online endpoint
6 endpoint = ManagedOnlineEndpoint(
7     name=online_endpoint_name,
8     description="this is an online endpoint",
9     auth_mode="key",
10     tags={
11         "training_dataset": "kidney_defaults",
12     },
13 )
14
15 # create the online endpoint
16 # expect the endpoint to take approximately 2 minutes.
17 endpoint = ml_client.online_endpoints.begin_create_or_update(endpoint).result()
✓ 1 min 4 sec

1 endpoint = ml_client.online_endpoints.get(name=online_endpoint_name)
2
3 print(
4     f'Endpoint "{endpoint.name}" with provisioning state "{endpoint.provisioning_state}" is retrieved'
5 )
✓ <1 sec
```

Endpoint "kidney-endpoint-f498cfb3" with provisioning state "Succeeded" is retrieved



The screenshot shows the Azure ML portal interface for the endpoint 'kidney-endpoint-f498cfb3'. The 'Details' tab is selected, showing various attributes and deployment information.

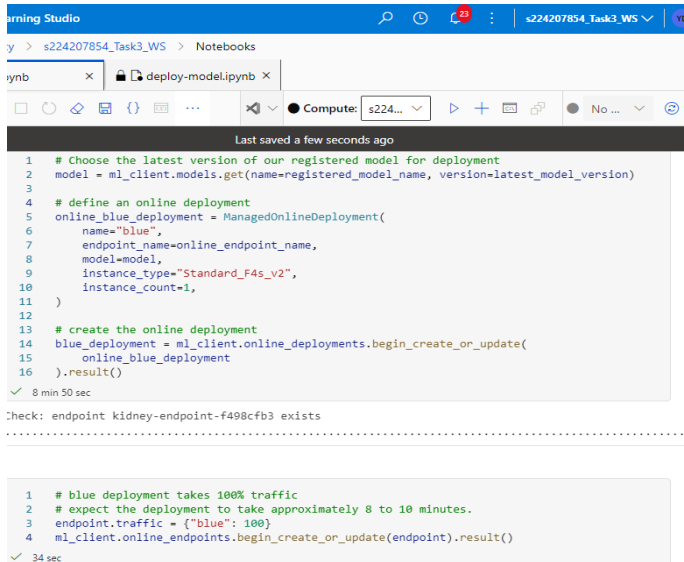
Endpoint attributes	
Service ID	kidney-endpoint-f498cfb3
Description	this is an online endpoint
Provisioning state	Succeeded
Error details	--
Compute type	Managed
Created by	YATHARTH DEOLY
Created on	Mar 30, 2024 3:27 PM
Last updated on	Mar 30, 2024 3:27 PM

Deployment summary	
Live traffic allocation	blue (100%)
Mirrored traffic allocation	--

Deployment blue	
Name	blue
Live traffic	100%
Scoring script	Auto-generated
Provisioning state	Succeeded

## Model Deployment

As our model and endpoint is ready, we will start our blue deployment with the latest version of model over endpoint. As we have created the cluster of instance type as “Standard\_F4s\_v2”, will be using the same instance for deployment also.



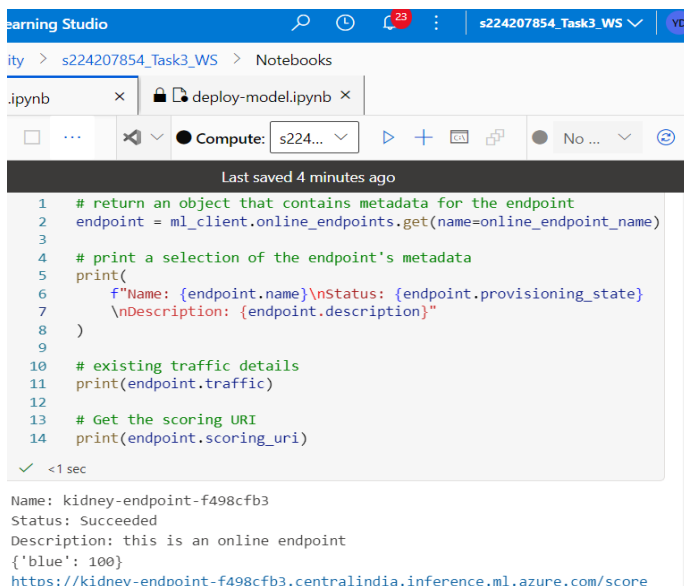
The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains Python code to create a managed online deployment. The second cell contains code to update the traffic allocation to 100% for the 'blue' deployment. The output of the first cell shows the deployment creation progress and a check for the endpoint existence. The output of the second cell shows the traffic update progress.

```
1 # Choose the latest version of our registered model for deployment
2 model = ml_client.models.get(name=registered_model_name, version=latest_model_version)
3
4 # define an online deployment
5 online_blue_deployment = ManagedOnlineDeployment(
6     name="blue",
7     endpoint_name=online_endpoint_name,
8     model=model,
9     instance_type="Standard_F4s_v2",
10    instance_count=1,
11 )
12
13 # create the online deployment
14 blue_deployment = ml_client.online_deployments.begin_create_or_update(
15     online_blue_deployment
16 ).result()
✓ 8 min 50 sec

Check: endpoint kidney-endpoint-f498cfb3 exists
.....

1 # blue deployment takes 100% traffic
2 # expect the deployment to take approximately 8 to 10 minutes.
3 endpoint.traffic = {"blue": 100}
4 ml_client.online_endpoints.begin_create_or_update(endpoint).result()
✓ 34 sec
```

Checking the status of deployment over endpoint.

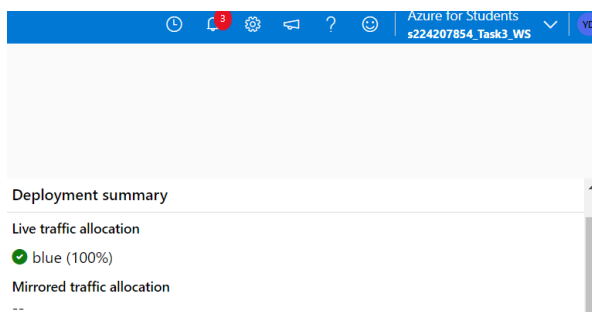


The screenshot shows a Jupyter Notebook interface with a single code cell. The code retrieves the endpoint metadata, prints it, and prints the existing traffic details. The output shows the endpoint name, status, description, and traffic allocation.

```
1 # return an object that contains metadata for the endpoint
2 endpoint = ml_client.online_endpoints.get(name=online_endpoint_name)
3
4 # print a selection of the endpoint's metadata
5 print(
6     f"Name: {endpoint.name}\nStatus: {endpoint.provisioning_state}\nDescription: {endpoint.description}"
7 )
8
9
10 # existing traffic details
11 print(endpoint.traffic)
12
13 # Get the scoring URI
14 print(endpoint.scoring_uri)
✓ <1 sec

Name: kidney-endpoint-f498cfb3
Status: Succeeded
Description: this is an online endpoint
{'blue': 100}
https://kidney-endpoint-f498cfb3.centralindia.inference.ml.azure.com/score
```

Deployment status can also be checked from Azure portal from “Endpoints”



## Summary

In this task we have learnt about usage of Azure with machine learning and how to train, manage and deploy model in Azure ML workspace.

## References

- <https://olympus.mygreatlearning.com/courses/103482/modules/items/5568439>
- <https://learn.microsoft.com/en-us/azure/machine-learning/tutorial-train-model?view=azureml-api-2>
- <https://learn.microsoft.com/en-us/azure/machine-learning/tutorial-deploy-model?view=azureml-api-2>