4.6 - Build an Azure Machine Learning custom skill for Azure AI Search

---

Overview

Custom skills allow you to enhance datasets as they pass through the enrichment pipeline. Azure Machine Learning can build custom models for regression or classification to enrich your search indexes.

Learning objectives

In this module, you'll learn how to:

- Understand how to use a custom Azure Machine Learning skillset.
- Use Azure Machine Learning to enrich Azure AI Search indexes.

Introduction

Azure Machine Learning allows you to build powerful AI models and host them on Azure. These AI models have **endpoints that can be used in search solutions to enrich indexes.**

This module builds on Create a custom skill for Azure AI Search but uses the custom AML skill to enrich a search index with output from an AI model.

In this module, you'll see how to enrich a search index using the custom **AmlSkill**. You'll explore the steps needed to bring Azure AI Machine Learning Studio and Azure AI Search together. Finally, you'll complete an exercise to create and enrich your own search index.

By completing this module, you'll learn how to:

- Understand how to use a custom Azure Machine Learning skillset.
- Use Azure Machine Learning to enrich Azure AI Search indexes.

Understand how to use a custom Azure Machine Learning skillset

Using a machine learning custom skill works the same as adding any other custom skill to a search index.

Here, you'll see how using the `AmlSkill` custom skill is different and explore the considerations of how to effectively use it.

Custom Azure Machine Learning skill schema

When you enrich a search index with an Azure Machine Learning (AML) custom skill, the enrichment happens at the document level. The **skillset used by your document indexer needs to include an `AmlSkill`**. The schema for this skill is:

```
{
      "@odata.type": "#Microsoft.Skills.Custom.AmlSkill",
      "name": "AML name",
      "description": "AML description",
      "context": "/document",
      "uri": "https://[Your AML endpoint]",
      "key": "Your AML endpoint key",
      "resourceId": null,
      "region": null,
      "timeout": "PT30S",
      "degreeOfParallelism": 1,
      "inputs": [
        {
          "name": "field name in the AML model",
          "source": "field from the document in the index"
```

```
        },
        {
          "name": "field name in the AML model",
          "source": "field from the document in the index"
        },

      ],
      "outputs": [
        {
          "name": "result field from the AML model",
          "targetName": "result field in the document"
        }
      ]
    }
```

> Important: The URI has to use an **HTTPS** endpoint. This can either be a managed custom URL address provided by Microsoft or your own domain name and certificate.

Take note that the custom skill doesn't include settings for `batchSize`. **As the AML model will process a single document at a time.** The remaining settings that control the performance of the skill are `timeout` and `degreeOfParallelism`. The above schema has set 30 seconds as the timeout value. The degree of parallelism should start at one. Depending on your infrastructure, you might be able to increase this number.

The best way to manage the efficiency of an AML skill is to scale up the Kubernetes inference cluster appropriately to manage your workload.

The index for the document needs a field to store the results from the AML model. You'll then add an output field mapping to store the results from the custom skill set to the field on the document in the index.

The JSON to do this output field mapping is:

```
"outputFieldMappings": [
    {
      "sourceFieldName": "/result field in the document",
      "targetFieldName": "result field from the AML model"
    }
  ]
```

Enrich a search index using an Azure Machine Learning model

You create your Azure Machine Learning model using developer tools like the Python SDK, REST APIs, or Azure CLI. Another option is to take advantage of the **Azure AI Machine Learning studio, a graphical user interface that lets you create, train, and deploy models without writing any code.**

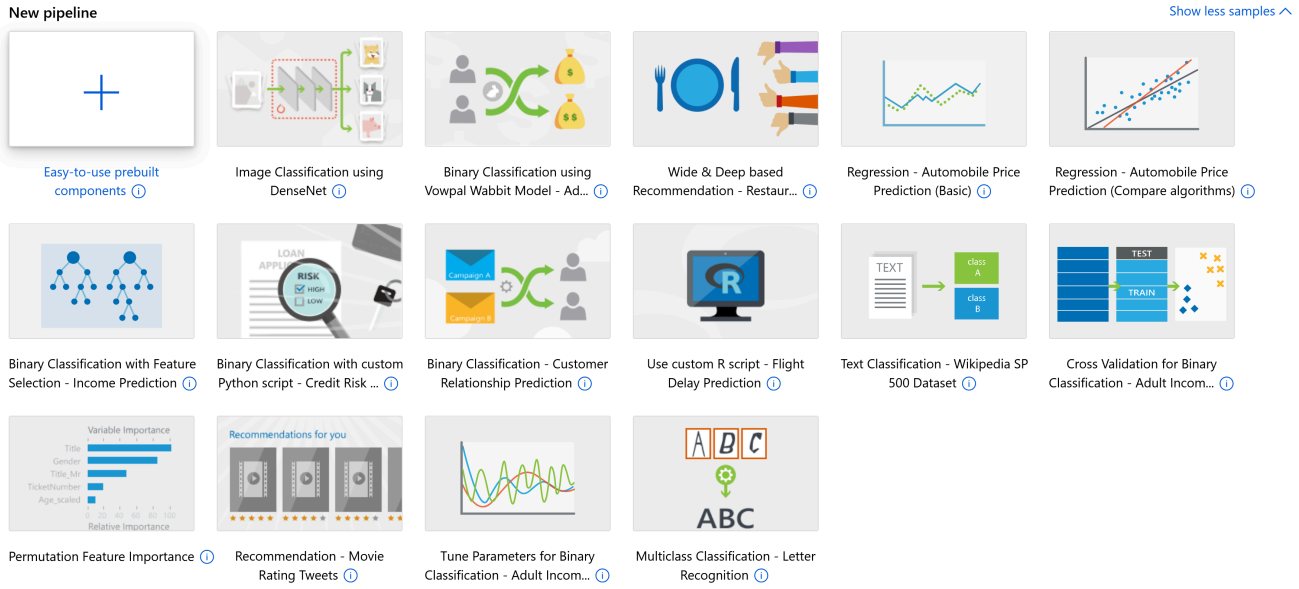
With a model created, you alter how the scoring code calls the model to allow it to be used by your custom search skill. The last steps are to **create a Kubernetes cluster to host an endpoint for your model.**

Create an AML workspace

When you create the AML workspace, Azure will also create storage accounts, a key store, and application insights resources. The AML workspace Overview pane gives you a link to launch the Azure AI Machine Learning Studio.

Create and train a model in Azure Machine Learning studio

Azure AI Machine Learning Studio lets you use a designer to use drag and drop to create pipelines that create and train models. There's an even easier way to create models by using prebuilt templates.



However you choose to create your models, they need to be registered in Azure AI Machine Learning Studio so that you can deploy the model to a web service.

Alter how the model works to allow it to be called by the AML custom skill

The models you train will normally use many examples of the data. The datasets will have many rows and be split and used to train and test the model. The code that handles this data and passes it to the model needs to be changed to handle single rows.

The JSON response from the model should also contain only the output prediction.

For example, if your data is an array of JSON objects:

```
[
    {
```

```
        "attribute-1": null,
        "attribute-2": null
    },
    {
        "attribute-1": null,
        "attribute-2": null
    },
    {
        "attribute-1": null,
        "attribute-2": null
    }
]
```

The python scoring code will have to process the data a row at a time:

```
data = json.loads(data)
for row in data:
    for key, val in row.items():
        input_entry[key].append(decode_nan(val))
```

To change the input dataset to a single record:

```
{
    "attribute-1": null,
    "attribute-2": null
}
```

The python code will need to change to:

```
data = json.loads(data)
for key, val in data.items():
    input_entry[key].append(decode_nan(val))
```

For the response from the scoring code, the default code returns the whole JSON document:

```
return json.dumps({"result": result.data_frame.values.tolist()})
```

The custom skill needs to be able to map a single response from the model. So the code should return JSON that is only the last attribute.

```
output = result.data_frame.values.tolist()
# return the last column of the the first row of the dataframe
return {
    "predicted_outcome": output[0][-1]
}
```

Create an endpoint for your model to use

The model is deployed to an endpoint. Azure AI Machine Learning Studio supports deploying a model to a real-time endpoint, a batch endpoint, or a web service. At the moment, the custom `AmlSkill` skill in Azure AI Search **only supports web service endpoints.**

The other restriction is that the endpoint has to be an Azure Kubernetes Service (AKS), container instances aren't supported.

If you have experience in creating and managing AKS clusters, you can manually create the clusters in the Azure portal and reference them when you create your endpoint. However, an **easier option is to let Azure AI Machine Learning Studio create and manage the cluster for you.**

If you navigate to the compute section of the studio, you can **create inference clusters**. AML studio will then guide you through choosing the size of the cluster and even enable HTTPS and create a domain name for you. It will be in the format of `location.cloudapp.azure.com:443`.

Connect the AML custom skill to the endpoint

With everything above in place, you need to **update your Azure AI Search service**. First, to enrich your search index you'll add a new field to your index to include the output for the model.

Then you'll update your index skillset and add the `#Microsoft.Skills.Custom.AmlSkill` custom skill.

Next, you'll change your indexer to map the output from the custom skill to the field you created on the index.

The **last step is to rerun your indexer** to enrich your index with the AML model.

Exercise: Enrich a search index using Azure Machine Learning model

You can use the power of machine learning to enrich a search index. To do this, you'll use a model trained in Azure AI Machine Learning studio and call it from a machine learning custom skillset.

In this exercise, you'll create an Azure AI Machine Learning Studio model, then train, deploy, and test an endpoint using the model. Then you'll create an Azure AI Search service, create sample data, and enrich an index using the Azure AI Machine Learning studio endpoint.

> **Note** To complete this exercise, you will need a Microsoft Azure subscription. If you don’t already have one, you can sign up for a free trial at https://azure.com/free.

Create an Azure Machine Learning workspace

Before you enrich your search index, create an **Azure Machine Learning workspace. The workspace will give you access to the Azure AI Machine Learning studio**, a graphical tool you can use to build AI models and deploy them for use.

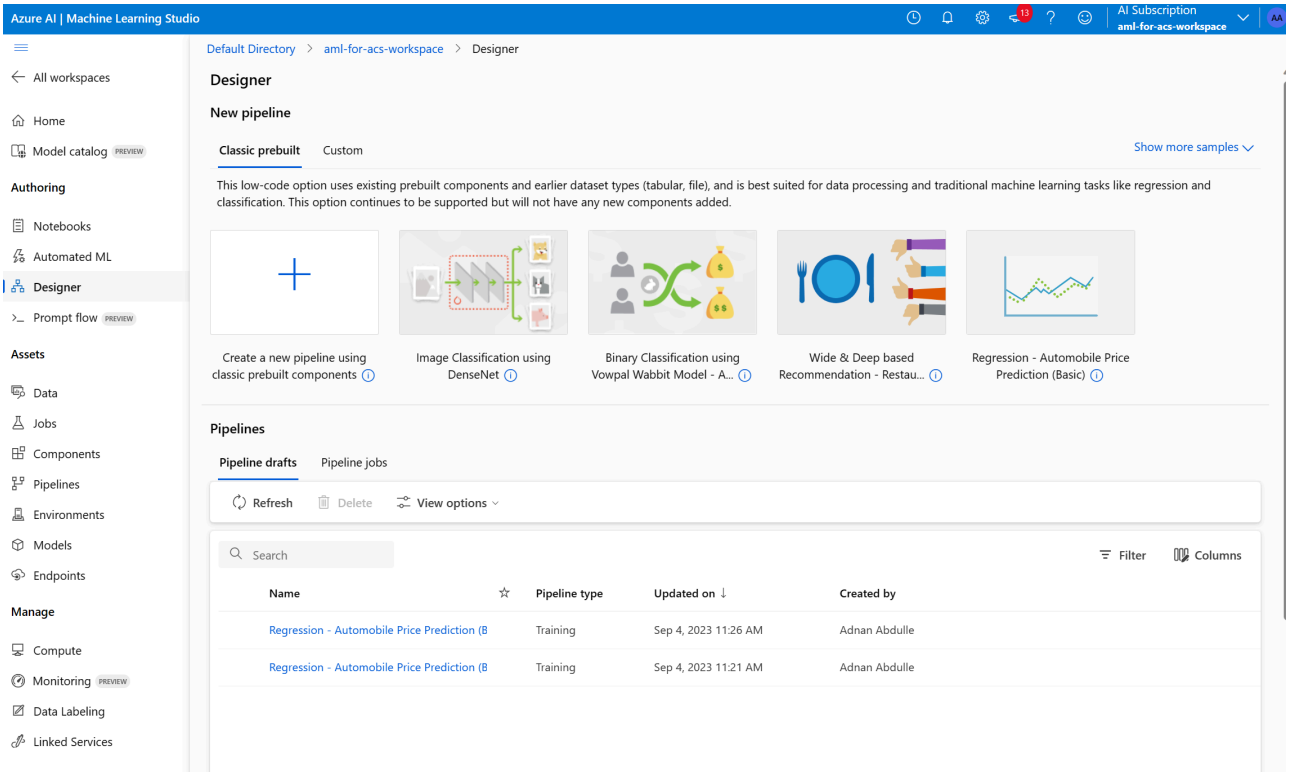
1. Sign into the Azure portal.
2. Select **+ Create a resource**.
3. Search for machine learning, and then select **Azure Machine Learning**.
4. Select **Create**.
5. Select **Create new** under **Resource group** and name it **aml-for-acs-enrichment**.
6. In the Workspace details section, for **Name**, enter **aml-for-acs-workspace**.
7. Select a supported **Region** near to you.

8. Use the default values for the **Storage account**, **Key vault**, **Application insights**, and **Container registry**.
9. Select **Review + create**.
10. Select **Create**.
11. Wait for the Azure Machine Learning workspace to be deployed, then select **Go to resource**.
12. On the Overview pane, select **Launch studio**.

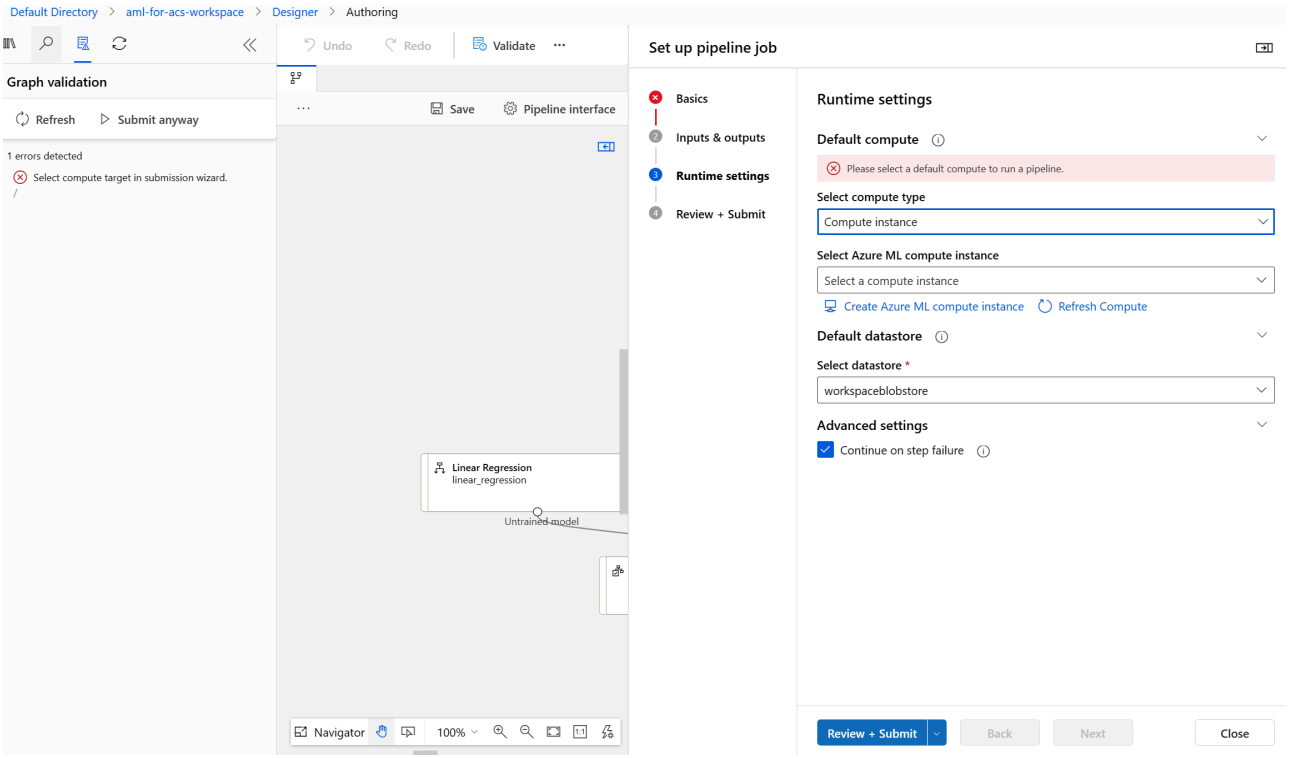
Create a regression training pipeline

You'll now create a regression model and train it using an Azure AI Machine Learning Studio pipeline. You'll train your model on automobile price data. The model, once trained, will predict the price of an automobile based on its attributes.

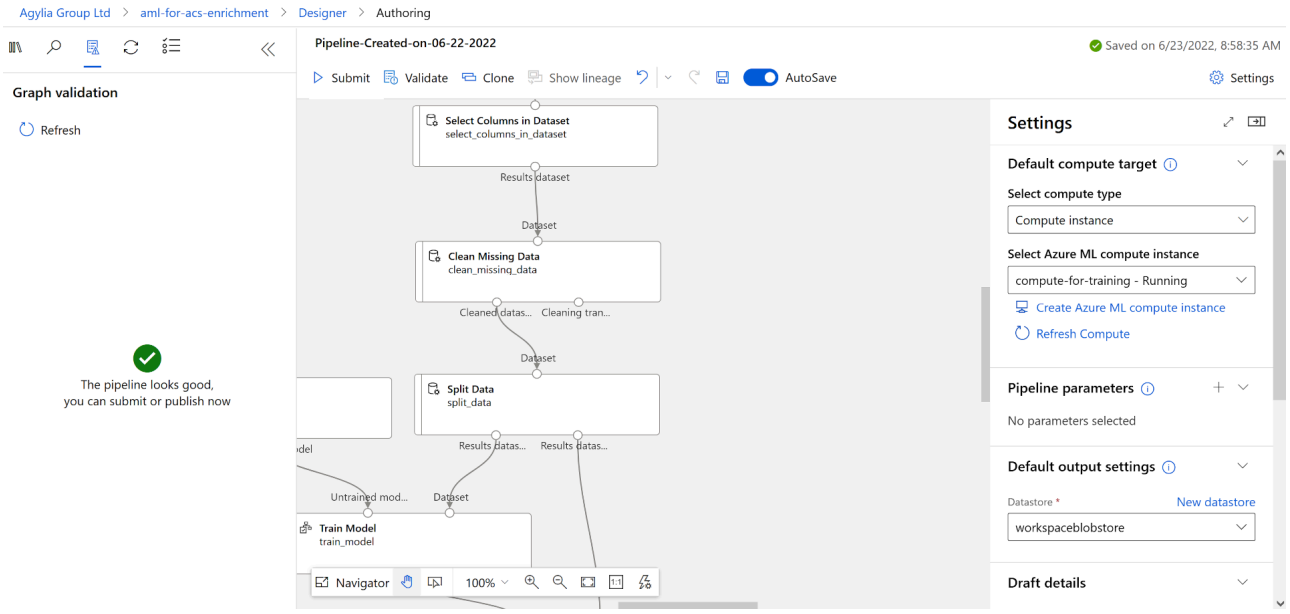
1. On the home page, select **Designer**.
2. From the list of prebuilt components, select **Regression - Automobile Price Prediction (Basic)**.



3. Select **Validate**.
4. On the **Graph validation** pane, select the error **Select compute target in submission wizard**.



5. In the **Select compute type** dropdown, choose **Compute instance**. Then select **Create Azure ML compute instance** underneath.
6. In the **Compute name** field, enter a unique name (such as **compute-for-training**).
7. Select **Review + create**, then select **Create**.
8. In the **Select Azure ML compute instance** field, select your instance from the dropdown. You might need to wait until it has finished provisioning.
9. Select **Validate** again, the pipeline should look good.



10. Select **Basics** in the **Set up pipeline job** pane.

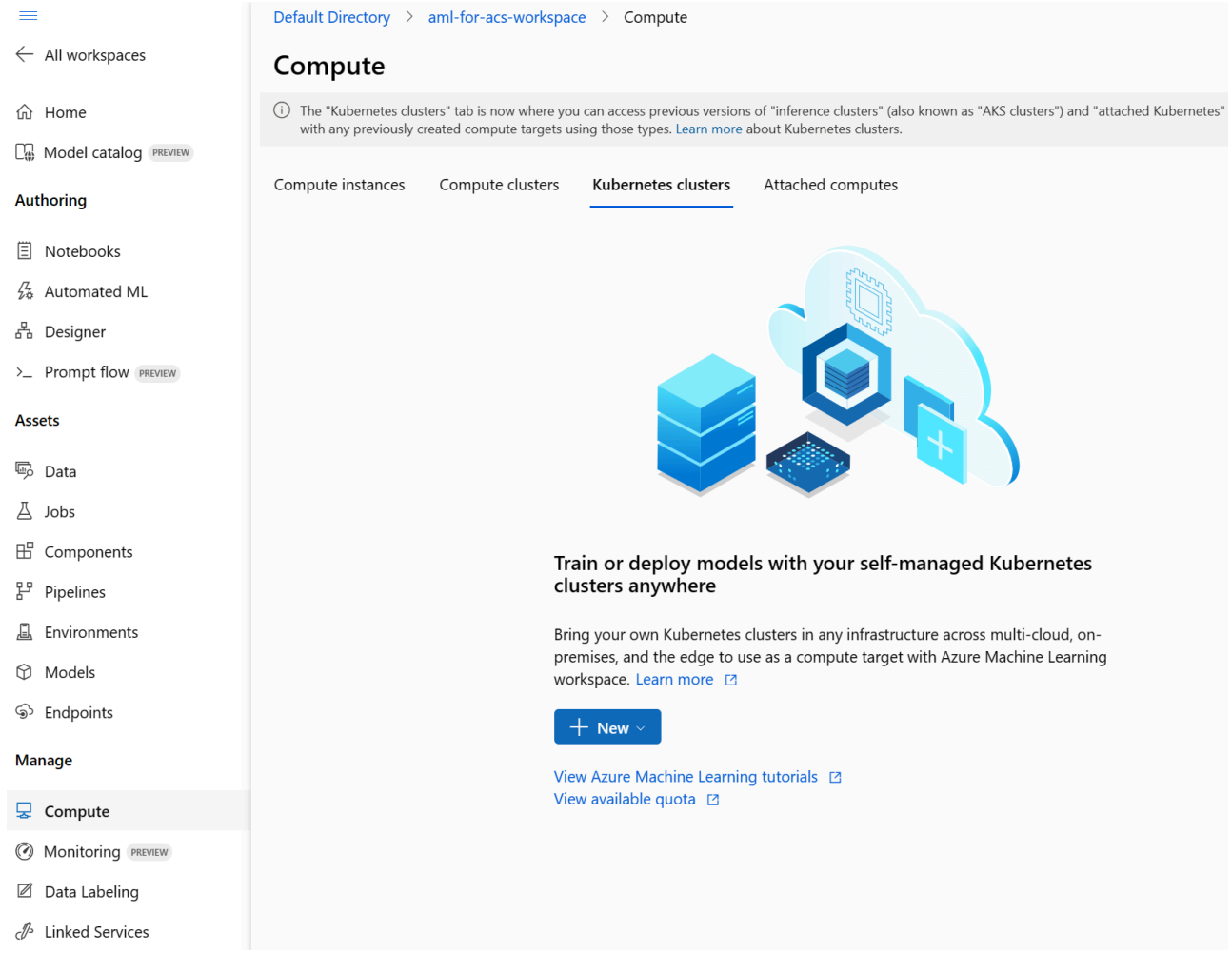
> Note: If you hid the **Set up pipeline job** pane before, you can open it again by selecting **Configure & Submit**.

11. Select **Create new** under the Experiment name.
12. In **New experiment name**, enter **linear-regression-training**.
13. Select **Review + Submit** , then select **Submit**.

Create an inference cluster for the endpoint

While your pipeline is training a linear regression model, you can create the resources you need for the endpoint. **This endpoint needs a Kubernetes cluster to process web requests to your model.**

1. On the left, select **Compute**.

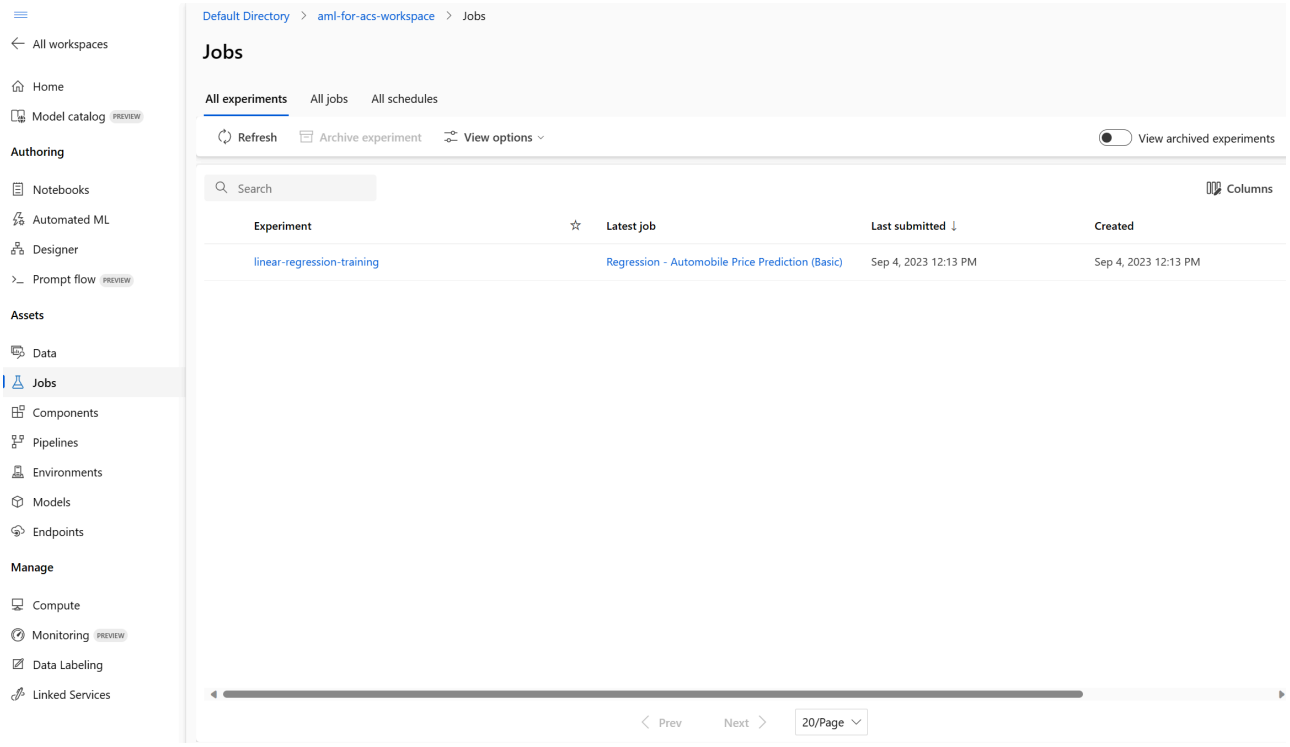


2. Select **Kubernetes clusters**, then select **+ New**.
3. In the dropdown, select **AksCompute**.
4. On the **Create AksCompute** pane, select **Create new**.
5. For **Location**, select the same region you used to create your other resources.
6. In the VM sizes list, select **Standard_A2_v2**.
7. Select **Next**.
8. In **Compute name**, enter **aml-acs-endpoint**.
9. Select **Enable SSL configuration**.
10. In **Leaf domain**, enter **aml-for-acs**.
11. Select **Create**.

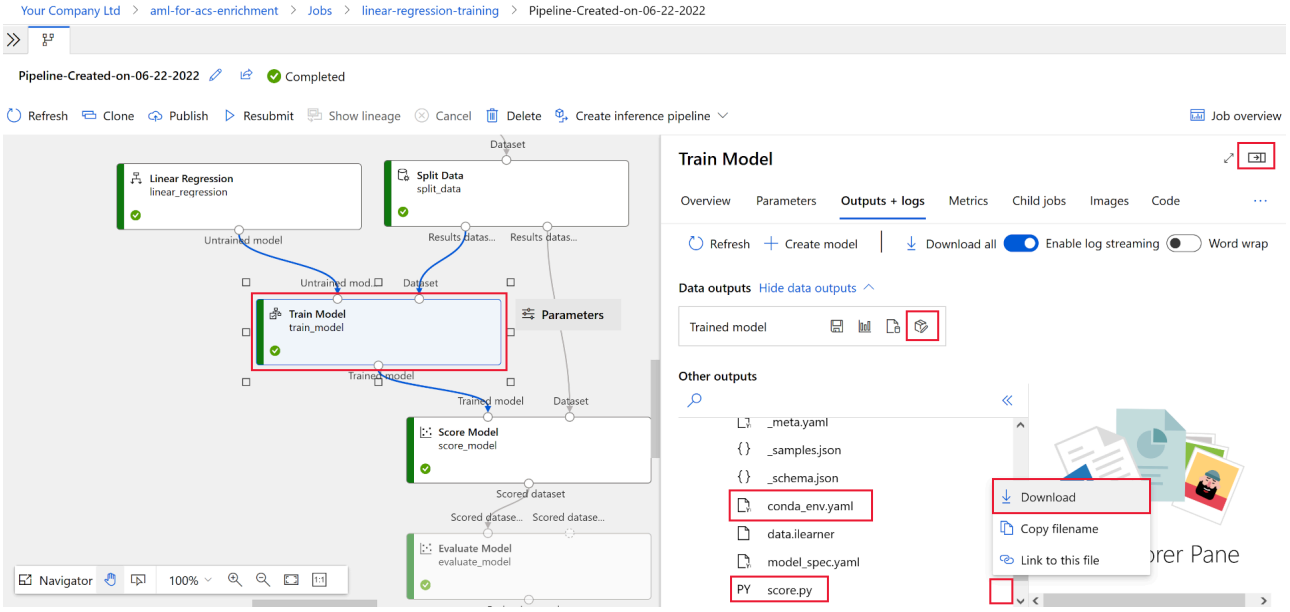
Register your trained model

Your pipeline job should have finished. You'll download the `score.py` and `conda_env.yaml` files. Then you'll register your trained model.

1. On the left, select **Jobs**.



2. Select your experiment, then select your completed job in the table, for example, **Regression - Automobile Price Prediction (Basic)**. If you're prompted to save changes, select **Discard** for changes.
3. In the designer, select **Job overview** in the top right, then select the **Train Model** node.



4. In the **Outputs + logs** tab, expand the **trained_model_outputs** folder.
5. Next to `score.py`, select the more menu (**...**), then select **Download**.
6. Next to `conda_env.yaml`, select the more menu (**...**), then select **Download**.
7. Select **+ Register model** at the top of the tab.
8. In the **Job output** field, select the **trained_model_outputs** folder. Then select **Next** at the bottom of the pane.
9. For model **Name**, enter **carevalmodel**.
10. In **Description**, enter **A linear regression model to predict the price of cars.**.
11. Select **Next**.
12. Select **Register**.

Edit the scoring script to respond to Azure AI Search correctly

Azure Machine Learning Studio has downloaded two files to your web browser's default download location.

You need to edit the score.py file to change how the JSON request and response are handled. You can use a text editor or a code editor like Visual Studio Code.

1. In your editor, open the score.py file.
2. Replace all the contents of the run function:

```
def run(data):
data = json.loads(data)
input_entry = defaultdict(list)
for row in data:
    for key, val in row.items():
        input_entry[key].append(decode_nan(val))

data_frame_directory = create_dfd_from_dict(input_entry, schema_data)
score_module = ScoreModelModule()
result, = score_module.run(
    learner=model,
    test_data=DataTable.from_dfd(data_frame_directory),
    append_or_result_only=True)
return json.dumps({"result": result.data_frame.values.tolist()})
```

   With this Python code:

```
def run(data):
    data = json.loads(data)
    input_entry = defaultdict(list)

    for key, val in data.items():
            input_entry[key].append(decode_nan(val))

    data_frame_directory = create_dfd_from_dict(input_entry, schema_data)
    score_module = ScoreModelModule()
    result, = score_module.run(
        learner=model,
        test_data=DataTable.from_dfd(data_frame_directory),
        append_or_result_only=True)
    output = result.data_frame.values.tolist()

    return {
            "predicted_price": output[0][-1]
    }
```

   The changes above allow the model to receive a single JSON object with car attributes instead of an array of cars.

   The other change is to only return the predicted price of the car instead of the whole JSON response.
3. Save the changes in your text editor.

Create a custom environment

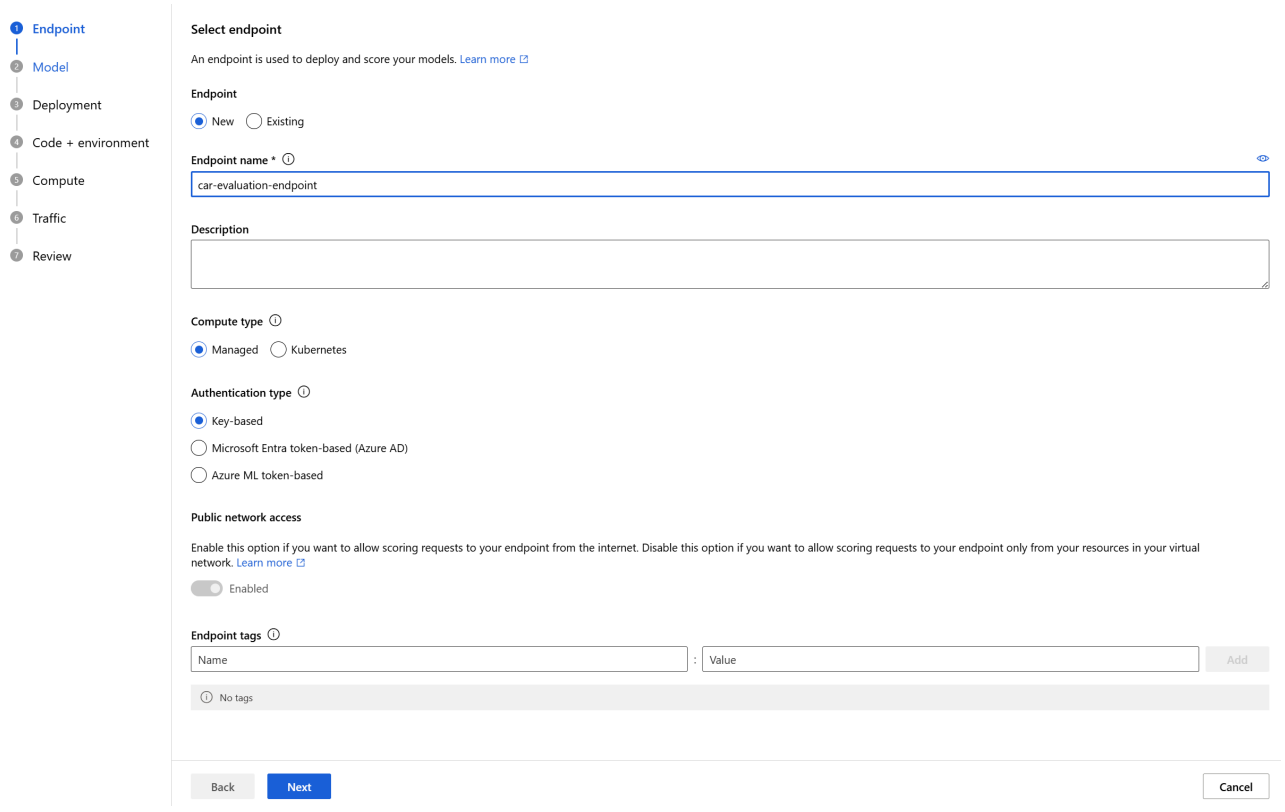
Next, you'll create a custom environment so you can deploy to a real-time endpoint.

1. Select **Environments** in the navigation pane.
2. Select the **Custom environments** tab.
3. Select **+ Create**.
4. For **Name**, enter **my-custom-environment**.
5. In the list of *Curated environments* under **Select environment type**, select the latest **automl-gpu** version.
6. Select **Next**.
7. On your local machine, open the `conda_env.yaml` file you downloaded earlier and copy its contents.
8. Return to browser, and select **conda_dependencies.yaml** in the Customize pane.
9. In the pane on the right, replace its contents with the code you copied earlier.
10. Select **Next**, then select **Next** again.
11. Select **Create** to create your custom environment.

Deploy the model with the updated scoring code

Your inference cluster should now be ready to use. You've also edited the scoring code to handle requests from your Azure Cognitive Search custom skillset. Let's create and test an endpoint for the model.

1. On the left, select **Models**.
2. Select the model you registered, **carevalmodel**.
3. Select **Deploy**, then select **Real-time endpoint**.



4. For **Name**, enter a unique name, for example **car-evaluation-endpoint-1440637584** .
5. For **Compute type**, select **Managed**.
6. For **Authentication type**, select **Key-based authentication**.
7. Select **Next**, then select **Next**.

8. Select **Next** again.
9. In the **Select a scoring script for inferencing** field, browse to your updated `score.py` file and select it.
10. In the **Select environment type** dropdown, select **Custom enviroments**.
11. Select the checkbox on your custom environment from the list.
12. Select **Next**.
13. For Virtual machine, select **Standard_D2as_v4**.
14. Set **Instance count** to **1**.
15. Select **Next**, then select **Next** again.
16. Select **Create**.

Wait for the model to be deployed, it can take up to 10 minutes. You can check the status in **Notifications** or the endpoints section of the Azure Machine Learning Studio.

Test your trained model's endpoint

1. On the left, select **Endpoints**.
2. Select **car-evaluation-endpoint**.
3. Select **Test**, in **Input data to test endpoint** paste this example JSON.

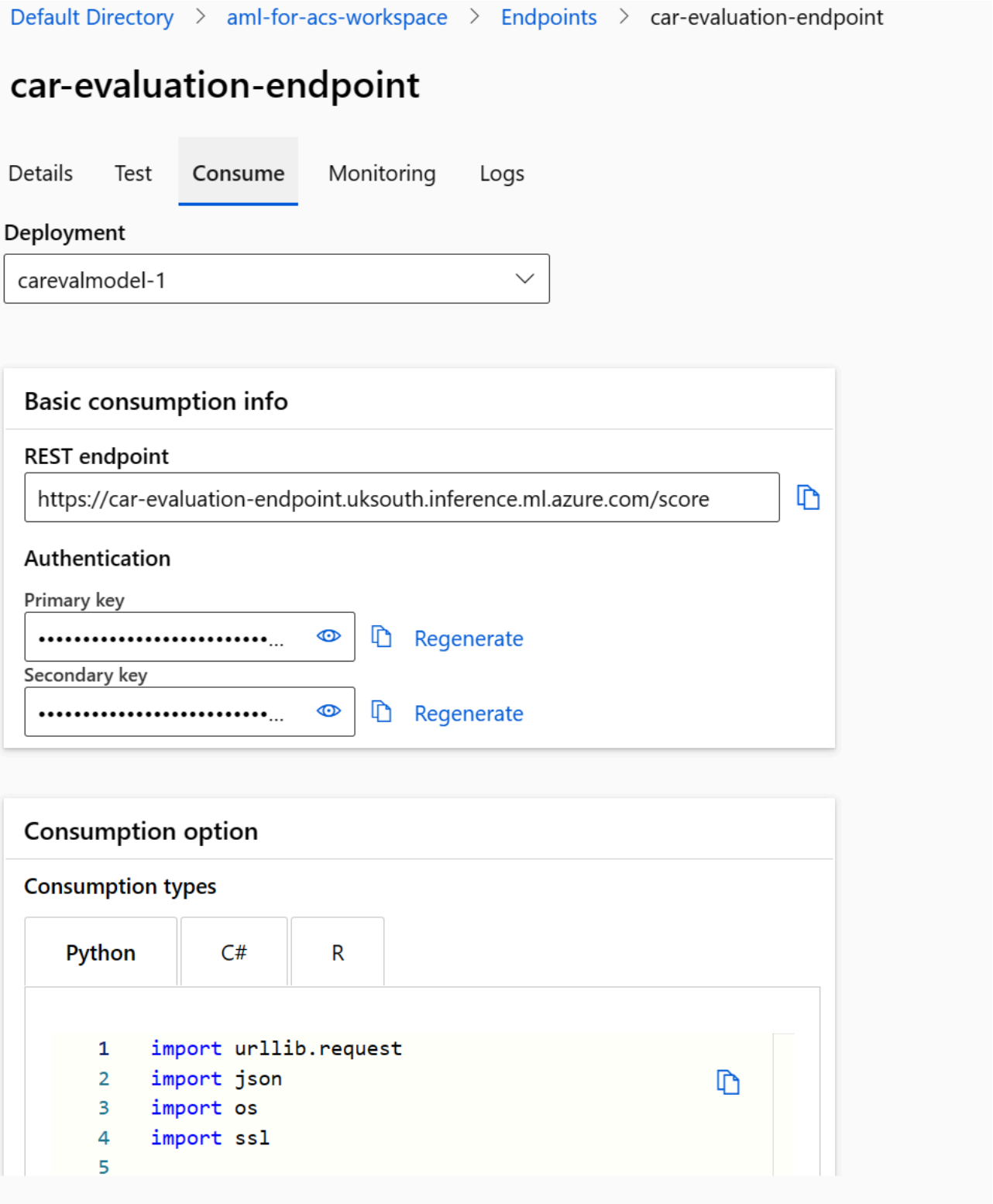
```
{
    "symboling": 2,
    "make": "mitsubishi",
    "fuel-type": "gas",
    "aspiration": "std",
    "num-of-doors": "two",
    "body-style": "hatchback",
    "drive-wheels": "fwd",
    "engine-location": "front",
    "wheel-base": 93.7,
    "length": 157.3,
    "width": 64.4,
    "height": 50.8,
    "curb-weight": 1944,
    "engine-type": "ohc",
    "num-of-cylinders": "four",
    "engine-size": 92,
    "fuel-system": "2bbl",
    "bore": 2.97,
    "stroke": 3.23,
    "compression-ratio": 9.4,
    "horsepower": 68.0,
    "peak-rpm": 5500.0,
    "city-mpg": 31,
    "highway-mpg": 38,
    "price": 0.0
}
```

4. Select **Test**, and you should see a response:

```
{
    "predicted_price": 5852.823214312815
```

```
}
```

5. Select **Consume**.



6. Copy the **REST endpoint** and **Primary key**.

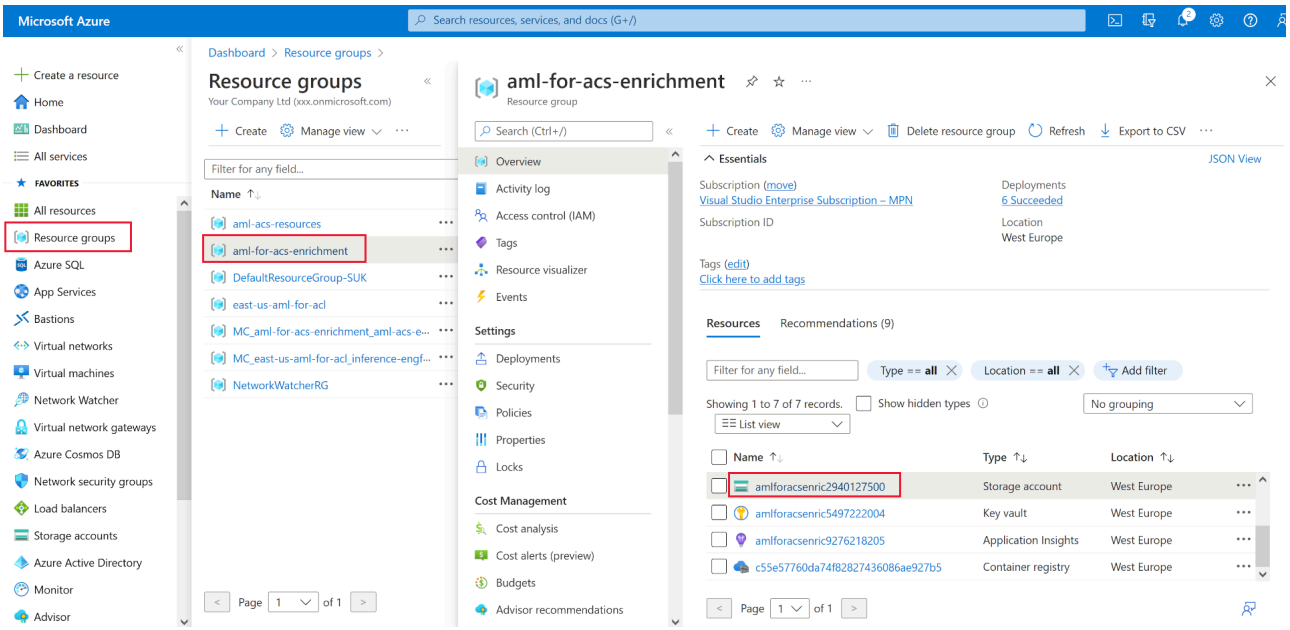
Integrate an Azure Machine Learning model with Azure AI Search

Next, you create a new Cognitive Search service and enrich an index using a custom skillset.

Create a test file

1. In the Azure portal, select Resource groups.

2. Select **aml-for-acs-enrichment**.



3. Select the storage account, for example **amlforacsworks1440637584**.
4. Select **Configuration** under **Settings**. Then set **Allow Blob anonymous access** to **Enabled**.
5. Select **Save**.
6. Under **Data storage**, select **Containers**.
7. Create a new container to store index data, select **+ Container**.
8. In the **New container** pane, in **Name**, enter **docs-to-search**.
9. In **Anonymous access level**, select **Container (anonymous read access for containers and blobs)**.
10. Select **Create**.
11. Select the **docs-to-search** container you created.
12. In a text editor, create a JSON document:

```
{
  "symboling": 0,
  "make": "toyota",
  "fueltype": "gas",
  "aspiration": "std",
  "numdoors": "four",
  "bodystyle": "wagon",
  "drivewheels": "fwd",
  "enginelocation": "front",
  "wheelbase": 95.7,
  "length": 169.7,
  "width": 63.6,
  "height": 59.1,
  "curbweight": 2280,
  "enginetype": "ohc",
  "numcylinders": "four",
  "enginesize": 92,
  "fuelsystem": "2bbl",
  "bore": 3.05,
  "stroke": 3.03,
  "compressionratio": 9.0,
  "horsepower": 62.0,
```

```
    "peakrpm": 4800.0,
    "citympg": 31,
    "highwaympg": 37,
    "price": 0
}
```

Save the document to your machine as `test-car.json` extension.

13. In the portal, select **Upload**.
14. In the **Upload blob** pane, select **Browse for files**, navigate to where you saved the JSON document, and select it.
15. Select **Upload**.

Create an Azure AI Search resource

1. In the Azure portal, on the home page, select **+ Create a resource**.
2. Search for **Azure AI Search**, then select **Azure AI Search**.
3. Select **Create**.
4. In **Resource Group**, select **aml-for-acs-enrichment**.
5. In Service name, enter a unique name, for example **acs-enriched-1440637584**.
6. For **Location**, select the same region you used earlier.
7. Select **Review + create**, then select **Create**.
8. Wait for the resources to be deployed, then select **Go to resource**.
9. Select **Import data**.
10. On the **Connect to your data** pane, for the **Data source** field, select **Azure Blob Storage**.
11. In **Data source name**, enter **import-docs**.
12. In **Parsing mode**, select **JSON**.
13. In **Connection string**, select **Choose an existing connection**.
14. Select the storage account you uploaded to, for example, **amlforacsworks1440637584**.
15. In the **Containers** pane, select **docs-to-search**.
16. Select **Select**.
17. Select **Next: Add cognitive skills (Optional)**.

Add cognitive skills

1. Expand **Add enrichments**, then select **Extract people names**.
2. Select **Next: Customize target index**.
3. Select **+ Add field**, in the **Field name** enter **predicted_price** at the bottom of the list.
4. In **Type**, select **Edm.Double** for your new entry.
5. Select **Retrievable** for all fields.
6. Select **Searchable** for **make**.
7. Select **Next: Create an indexer**.
8. Select **Submit**.

Add the AML Skill to the skillset

You'll now replace the people names enrichment with the Azure Machine Learning custom skillset.

1. On the Overview pane, select **Skillsets** under **Search management**.

2. Under **Name**, select **azureblob-skillset**.
3. Replace the skills definition for the `EntityRecognitionSkill` with this JSON, remember to replace your copied endpoint and primary key values:

```
"@odata.type": "#Microsoft.Skills.Custom.AmlSkill",
"name": "AMLenricher",
"description": "AML studio enrichment example",
"context": "/document",
"uri": "PASTE YOUR AML ENDPOINT HERE",
"key": "PASTE YOUR PRIMARY KEY HERE",
"resourceId": null,
"region": null,
"timeout": "PT30S",
"degreeOfParallelism": 1,
"inputs": [
  {
    "name": "symboling",
    "source": "/document/symboling"
  },
  {
    "name": "make",
    "source": "/document/make"
  },
  {
    "name": "fuel-type",
    "source": "/document/fueltype"
  },
  {
    "name": "aspiration",
    "source": "/document/aspiration"
  },
  {
    "name": "num-of-doors",
    "source": "/document/numdoors"
  },
  {
    "name": "body-style",
    "source": "/document/bodystyle"
  },
  {
    "name": "drive-wheels",
    "source": "/document/drivewheels"
  },
  {
    "name": "engine-location",
    "source": "/document/enginelocation"
  },
  {
    "name": "wheel-base",
    "source": "/document/wheelbase"
  },
  {
    "name": "length",
    "source": "/document/length"
  },
```

```
    {
      "name": "width",
      "source": "/document/width"
    },
    {
      "name": "height",
      "source": "/document/height"
    },
    {
      "name": "curb-weight",
      "source": "/document/curbweight"
    },
    {
      "name": "engine-type",
      "source": "/document/enginetype"
    },
    {
      "name": "num-of-cylinders",
      "source": "/document/numcylinders"
    },
    {
      "name": "engine-size",
      "source": "/document/enginesize"
    },
    {
      "name": "fuel-system",
      "source": "/document/fuelsystem"
    },
    {
      "name": "bore",
      "source": "/document/bore"
    },
    {
      "name": "stroke",
      "source": "/document/stroke"
    },
    {
      "name": "compression-ratio",
      "source": "/document/compressionratio"
    },
    {
      "name": "horsepower",
      "source": "/document/horsepower"
    },
    {
      "name": "peak-rpm",
      "source": "/document/peakrpm"
    },
    {
      "name": "city-mpg",
      "source": "/document/citympg"
    },
    {
      "name": "highway-mpg",
      "source": "/document/highwaympg"
    },
```

```
    {
      "name": "price",
      "source": "/document/price"
    }
  ],
  "outputs": [
    {
      "name": "predicted_price",
      "targetName": "predicted_price"
    }
  ]
```

4. Select **Save**.

Update the output field mappings

1. Go back to the **Overview** pane, and select **Indexers**, then select the **azureblob-indexer**.
2. Select the **Indexer Definition (JSON)** tab, then change the **outputFieldMappings** value to:

```
  "outputFieldMappings": [
      {
        "sourceFieldName": "/document/predicted_price",
        "targetFieldName": "predicted_price"
      }
    ]
```

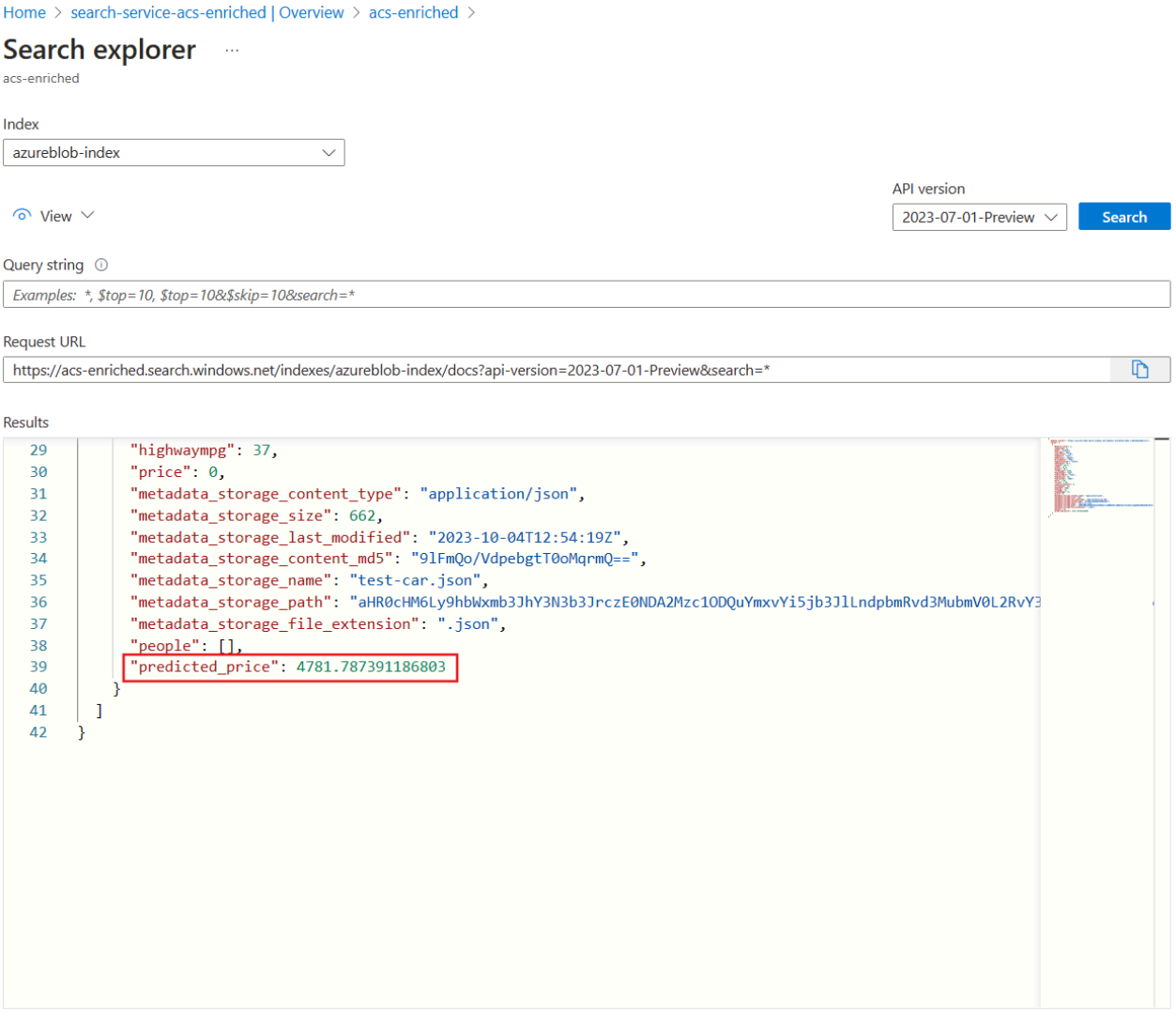
3. Select **Save**.
4. Select **Reset**, then select **Yes**.
5. Select **Run**, then select **Yes**.

Test index enrichment

The updated skillset will now add a predicted value to the test car document in your index. To test this, follow these steps.

1. On the **Overview** pane of your search service, select **Search explorer** at the top of the pane.
2. Select **Search**.

3. Scroll to the bottom of the results.



You should see the populate field `predicted_price`.

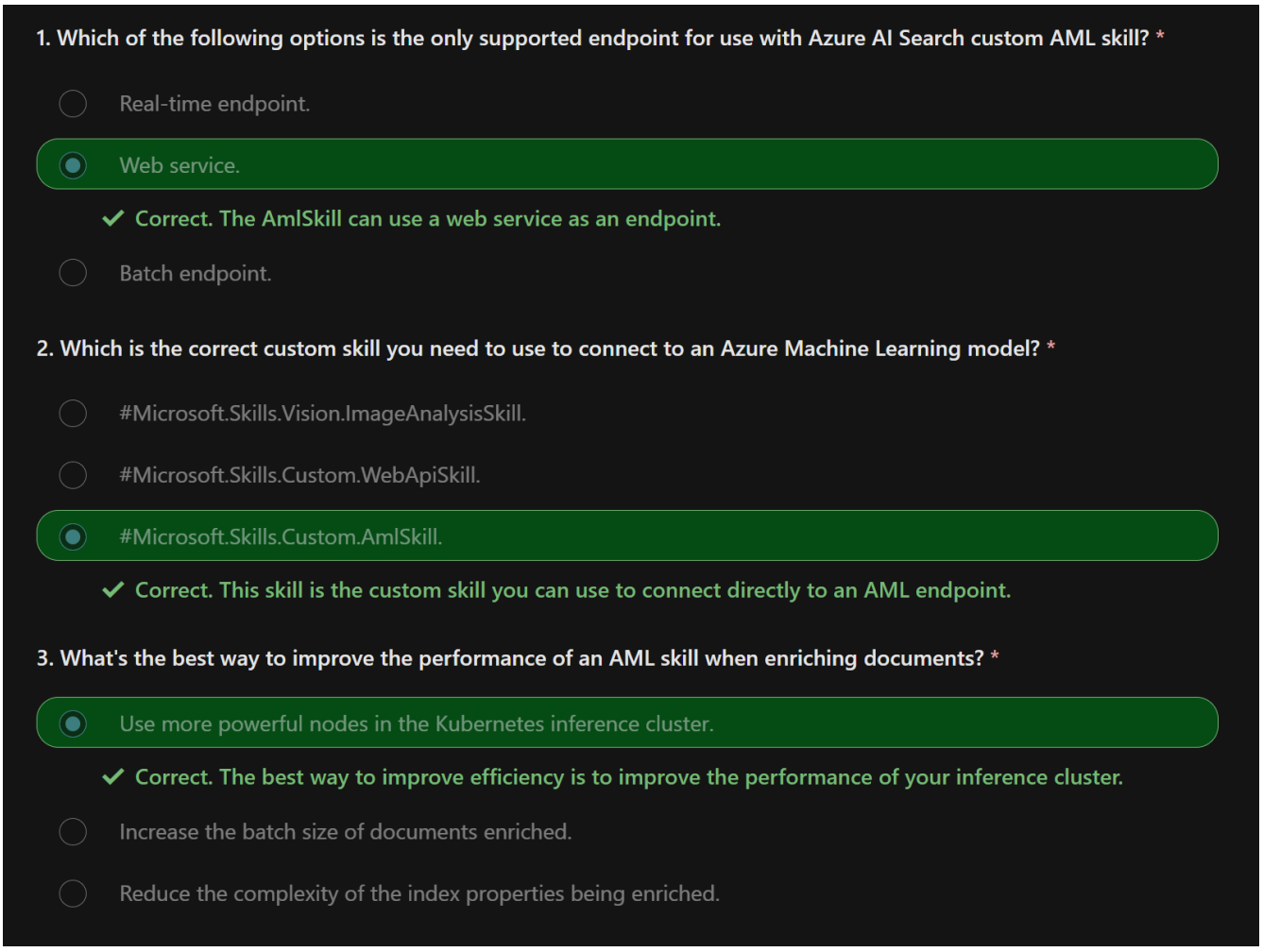
Delete exercise resources

Now that you've completed the exercise, delete all the resources you no longer need. Delete the Azure resources:

1. In the Azure portal, select **Resource groups**.
2. Select the resource group you don't need, then select **Delete resource group**.

Reference: [mslearn-knowledge-mining]

Knowledge Check



Summary

In this module, you've seen how to use the custom `AmlSkill` to enrich a model created in AML studio. Then you built your own Azure AI Search solution and enriched a sample car index with a predicted price.

Now that you've completed this module, you learned how to:

- Understand how to use a custom Azure Machine Learning skillset.
- Use Azure Machine Learning to enrich Azure AI Search indexes.

If you'd like to learn more about Azure Machine Learning studio, see What is Azure Machine Learning designer?.

✍️ Compiled by Kenneth Leung (2025)