**Managed Online Endpoints in Azure Machine Learning**

**Real-time Predictions**

* Deploy models to HTTPS endpoints for real-time predictions.
* Data sent to the endpoint is processed by a scoring script to return predictions.

**Types of Online Endpoints**

* **Managed Online Endpoints**: Azure handles infrastructure management.
* **Kubernetes Online Endpoints**: Users manage the Kubernetes cluster.

**Benefits of Managed Online Endpoints**

* Ideal for initial testing and validation.
* Requires specifying only VM type and scaling settings.
* Azure manages provisioning compute power and updating the host OS.

**Deploying a Model**

1. **Model Assets**: Model files (e.g., pickle files) or registered models in Azure ML workspace.
2. **Scoring Script**: Loads the model (automatically generated for MLFlow models).
3. **Environment**: Lists necessary packages for the endpoint's compute.
4. **Compute Configuration**: Specifies compute size and scaling settings.

**Blue/Green Deployment Strategy**

* Allows multiple model versions on the same endpoint.
* Example:
  + **Blue Deployment**: First version of the model.
  + **Green Deployment**: Second version of the model (for testing).
* Traffic distribution (e.g., 90% to blue, 10% to green) for testing new models.
* Seamlessly transition between model versions without service interruption.

**Creating an Endpoint**

* Use the ManagedOnlineEndpoint class.
* Required parameters:
  + name: Unique name in the Azure region.
  + auth\_mode: Authentication mode (key for key-based, aml\_token for token-based).

**Sample Code to Create an Endpoint**

python

Copy code

from azure.ai.ml.entities import ManagedOnlineEndpoint

# create an online endpoint

endpoint = ManagedOnlineEndpoint(

name="endpoint-example",

description="Online endpoint",

auth\_mode="key",

)

ml\_client.begin\_create\_or\_update(endpoint).result()

The easiest way to deploy a model to an online endpoint is to use an **MLflow** model and deploy it to a *managed* online endpoint. Azure Machine Learning will automatically generate the scoring script and environment for MLflow models.

To deploy an MLflow model, you need to have created an endpoint. Then you can deploy the model to the endpoint.

**Deploy an MLflow model to an endpoint**

When you deploy an MLflow model to a managed online endpoint, you don´t need to have the scoring script and environment.

To deploy an MLflow model, you must have model files stored on a local path or with a registered model. You can log model files when training a model by using MLflow tracking.

In this example, we're taking the model files from a local path. The files are all stored in a local folder called model. The folder must include the MLmodel file, which describes how the model can be loaded and used.

Next to the model, you also need to specify the compute configuration for the deployment:

* instance\_type: Virtual machine (VM) size to use. [Review the list of supported sizes](https://learn.microsoft.com/en-us/azure/machine-learning/reference-managed-online-endpoints-vm-sku-list).
* instance\_count: Number of instances to use.

Since only one model is deployed to the endpoint, you want this model to take 100% of the traffic. When you deploy multiple models to the same endpoint, you can distribute the traffic among the deployed models.

To route traffic to a specific deployment, use the following code:

# blue deployment takes 100 traffic

endpoint.traffic = {"blue": 100}

ml\_client.begin\_create\_or\_update(endpoint).result()

To delete the endpoint and all associated deployments, run the command:

ml\_client.online\_endpoints.begin\_delete(name="endpoint-example")

**Deploy a model to a managed online endpoint**

You can choose to deploy a model to a managed online endpoint without using the MLflow model format. To deploy a model, you'll need to create the scoring script and define the environment necessary during inferencing.

To deploy a model, you need to have created an endpoint. Then you can deploy the model to the endpoint.

**Deploy a model to an endpoint**

To deploy a model, you must have:

* Model files stored on local path or registered model.
* A scoring script.
* An execution environment.

The model files can be logged and stored when you train a model.

**Create the scoring script**

The scoring script needs to include two functions:

* init(): Called when the service is initialized.
* run(): Called when new data is submitted to the service.

The **init** function is called when the deployment is created or updated, to load and cache the model from the model registry. The **run** function is called for every time the endpoint is invoked, to generate predictions from the input data. The following example Python script shows this pattern:

import json

import joblib

import numpy as np

import os

# called when the deployment is created or updated

def init():

global model

# get the path to the registered model file and load it

model\_path = os.path.join(os.getenv('AZUREML\_MODEL\_DIR'), 'model.pkl')

model = joblib.load(model\_path)

# called when a request is received

def run(raw\_data):

# get the input data as a numpy array

data = np.array(json.loads(raw\_data)['data'])

# get a prediction from the model

predictions = model.predict(data)

# return the predictions as any JSON serializable format

return predictions.tolist()

**Create an environment**

Your deployment requires an execution environment in which to run the scoring script.

You can create an environment with a Docker image with Conda dependencies, or with a Dockerfile.

To create an environment using a base Docker image, you can define the Conda dependencies in a conda.yml file

**Create the deployment**

When you have your model files, scoring script, and environment, you can create the deployment.

To deploy a model to an endpoint, you can specify the compute configuration with two parameters:

* instance\_type: Virtual machine (VM) size to use. [Review the list of supported sizes](https://learn.microsoft.com/en-us/azure/machine-learning/reference-managed-online-endpoints-vm-sku-list).
* instance\_count: Number of instances to use.

To deploy the model, use the ManagedOnlineDeployment class

Use the Azure Machine Learning studio

You can list all endpoints in the Azure Machine Learning studio, by navigating to the Endpoints page. In the Real-time endpoints tab, all endpoints are shown.

You can select an endpoint to review its details and deployment logs.

Additionally, you can use the studio to test the endpoint.

For testing, you can also use the Azure Machine Learning Python SDK to invoke an endpoint.

In many production scenarios, long-running tasks that deal with large amounts of data are performed as batch operations. In machine learning, batch inferencing is used to asynchronously apply a predictive model to multiple cases and write the results to a file or database.

 **Batch Predictions**:

* Batch predictions involve deploying a model to a batch endpoint, which allows triggering batch scoring jobs asynchronously.
* The endpoint is an HTTPS endpoint that can be called to initiate batch scoring from services like Azure Synapse Analytics or Azure Databricks.
* Results from batch scoring jobs are typically stored in a datastore connected to Azure Machine Learning.

 **Creating a Batch Endpoint**:

* Use the BatchEndpoint class to create a batch endpoint in Azure Machine Learning.
* Endpoint names must be unique within an Azure region.

 **Deployment to Batch Endpoints**:

* You can deploy multiple models to a batch endpoint.
* When calling the batch endpoint, it triggers a batch scoring job using the default deployment configuration unless specified otherwise.

 **Compute Clusters for Batch Deployments**:

* **Ideal Compute**: Azure Machine Learning compute clusters are recommended for batch deployments.
* For efficient processing of new data in parallel batches, ensure the compute cluster has more than one maximum instance.

 **Creating a Compute Cluster**:

* Use the AMLCompute class to define and create an Azure Machine Learning compute cluster.
* Specify the VM size (vm\_size) and maximum nodes (max\_nodes) based on your workload requirements.

 **Managing Batch Endpoints**:

* Batch endpoints can be managed, including viewing details and configuring deployments, through the Azure Machine Learning studio or programmatically using the Azure Machine Learning Python SDK.

An easy way to deploy a model to a batch endpoint is to use an **MLflow** model. Azure Machine Learning will automatically generate the scoring script and environment for MLflow models.

To deploy an MLflow model, you need to have created an endpoint. Then you can deploy the model to the endpoint.

**Register an MLflow Model**

To deploy an MLflow model, you first need to register it in your Azure Machine Learning workspace. Registration allows Azure ML to manage and deploy the model effectively.

python

Copy code

from azure.ai.ml.entities import Model

from azure.ai.ml.constants import AssetTypes

# Define model details and register it

model\_name = 'mlflow-model'

model = ml\_client.models.create\_or\_update(

Model(name=model\_name, path='./model', type=AssetTypes.MLFLOW\_MODEL)

)

* **Model Registration**: Use ml\_client.models.create\_or\_update() to register the model.
* **Path**: Specify the local path where your MLflow model files (MLmodel file and associated artifacts) are stored.

**Deploy the MLflow Model to a Batch Endpoint**

After registering the model, you can deploy it to a batch endpoint for batch predictions.

python

Copy code

from azure.ai.ml.entities import BatchDeployment

# Define deployment details

batch\_deployment = BatchDeployment(

name="mlflow-batch-deployment",

model=model,

instance\_count=2, # Number of compute nodes for generating predictions

max\_concurrency\_per\_instance=5, # Maximum parallel runs per compute node

mini\_batch\_size=10, # Number of files per scoring script run

output\_action="append\_row", # Append predictions to output file

output\_file\_name="predictions.csv" # Output file for predictions

)

# Create or update the batch deployment

ml\_client.batch\_deployments.begin\_create\_or\_update(batch\_deployment)

* **Batch Deployment Configuration**:
  + **instance\_count**: Number of compute nodes to use.
  + **max\_concurrency\_per\_instance**: Maximum parallel scoring script runs per compute node.
  + **mini\_batch\_size**: Number of files processed per scoring script run.
  + **output\_action**: Defines how predictions are handled (summary\_only or append\_row).
  + **output\_file\_name**: Name of the file to which predictions are appended if append\_row is chosen for output\_action.

If you want to deploy a model to a batch endpoint without using the MLflow model format, you need to create the scoring script and environment.

To deploy a model, you must have already created an endpoint. Then you can deploy the model to the endpoint.

**Create the scoring script**

The scoring script is a file that reads the new data, loads the model, and performs the scoring.

The scoring script must include two functions:

* init(): Called once at the beginning of the process, so use for any costly or common preparation like loading the model.
* run(): Called for each mini batch to perform the scoring.

The run() method should return a pandas DataFrame or an array/list.

**Create an environment**

Your deployment requires an execution environment in which to run the scoring script. Any dependency your code requires should be included in the environment.

You can create an environment with a Docker image with Conda dependencies, or with a Dockerfile.

You'll also need to add the library azureml-core as it is required for batch deployments to work.

To create an environment using a base Docker image, you can define the Conda dependencies in a conda.yaml file:

**Configure and create the deployment**

Finally, you can configure and create the deployment with the BatchDeployment class.

**nvoke and troubleshoot batch endpoints**

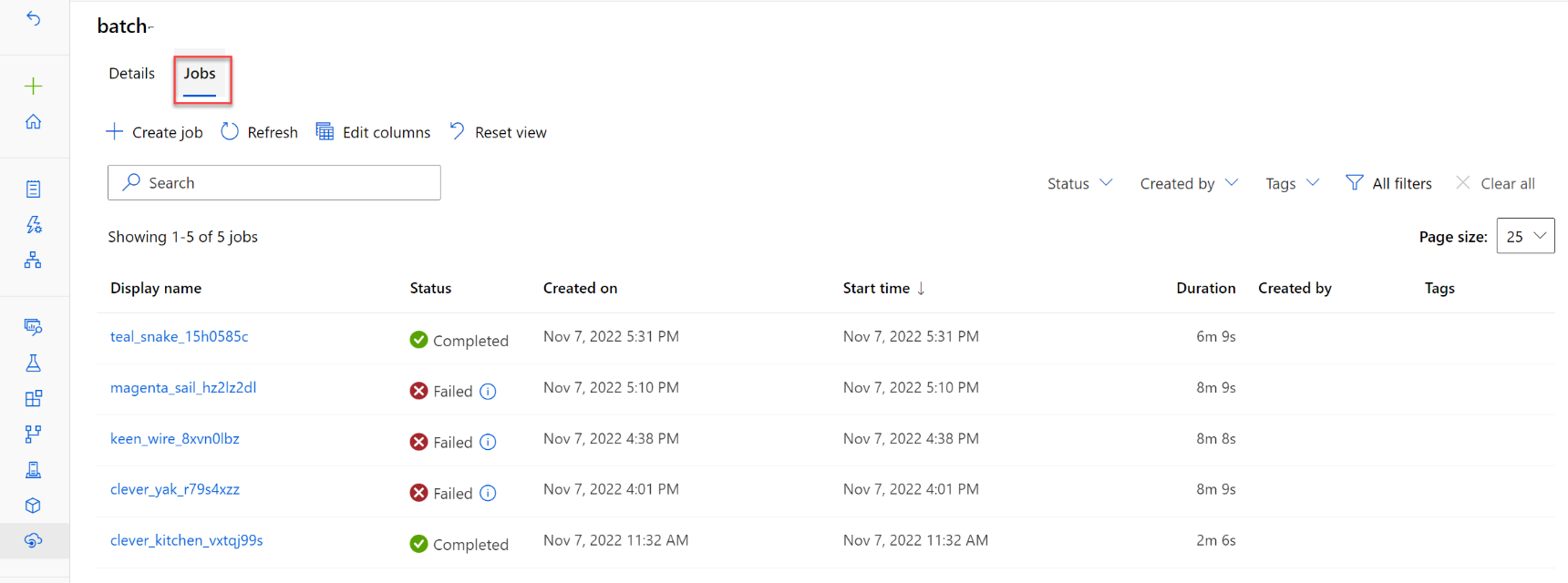
When you invoke a batch endpoint, you trigger an Azure Machine Learning **pipeline job**. The job will expect an input parameter pointing to the data set you want to score.

**Trigger the batch scoring job**

To prepare data for batch predictions, you can register a folder as a data asset in the Azure Machine Learning workspace.

You can then use the registered data asset as input when invoking the batch endpoint with the Python SDK:

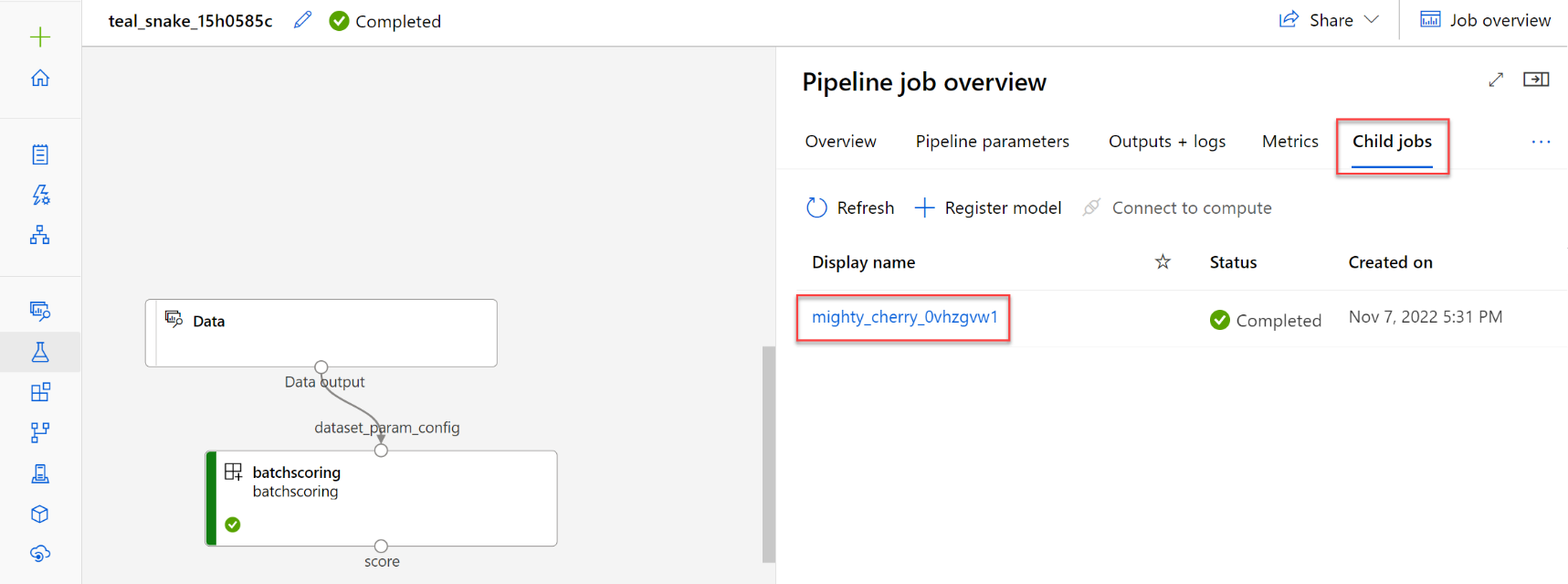
You can monitor the run of the pipeline job in the Azure Machine Learning studio. All jobs that are triggered by invoking the batch endpoint will show in the **Jobs** tab of the batch endpoint.



The predictions will be stored in the default datastore.

**Troubleshoot a batch scoring job**

The batch scoring job runs as a *pipeline job*. If you want to troubleshoot the pipeline job, you can review its details and the outputs and logs of the pipeline job itself.



If you want to troubleshoot the scoring script, you can select the child job and review its outputs and logs.

Navigate to the **Outputs + logs** tab. The **logs/user/** folder contains three files that will help you troubleshoot:

* job\_error.txt: Summarize the errors in your script.
* job\_progress\_overview.txt: Provides high-level information about the number of mini-batches processed so far.
* job\_result.txt: Shows errors in calling the init() and run() function in the scoring script.