Deploy to an online endpoint

To consume a model from an application, you can deploy the model to an online endpoint. You'll create an MLflow model from local files and test the endpoint.

Before you start

You'll need the latest version of the azureml-ai-ml package to run the code in this notebook. Run the cell below to verify that it is installed.

> **Note: If the azure-ai-ml package is not installed, run `pip install azure-ai-ml` to install it.**

In [ ]:

```
pip show azure-ai-ml
```

Connect to your workspace

With the required SDK packages installed, now you're ready to connect to your workspace.

To connect to a workspace, we need identifier parameters - a subscription ID, resource group name, and workspace name. Since you're working with a compute instance, managed by Azure Machine Learning, you can use the default values to connect to the workspace.

In [ ]:

```
from azure.identity import DefaultAzureCredential, InteractiveBrowserCredential
from azure.ai.ml import MLClient

try:
    credential = DefaultAzureCredential()
    # Check if given credential can get token successfully.
    credential.get_token("https://management.azure.com/.default")
except Exception as ex:
    # Fall back to InteractiveBrowserCredential in case DefaultAzureCredential not work
    credential = InteractiveBrowserCredential()
```

In [ ]:

```
# Get a handle to workspace
ml_client = MLClient.from_config(credential=credential)
```

Define and create an endpoint

Ultimately, the goal is to deploy a model to an endpoint. Therefore, you first need to create an endpoint. The endpoint will be a HTTPS endpoint that an application can call to receive predictions from the model. An application can consume an endpoint by using its URI, and authenticating with a key or token.

Run the following cell to define the endpoint. Note that the name of the endpoint has to be unique. You'll use the `datetime` function to generate a unique name.

In [ ]:

```
from azure.ai.ml.entities import ManagedOnlineEndpoint
import datetime

online_endpoint_name = "endpoint-" + datetime.datetime.now().strftime("%m%d%H%M%f")
```

```
# create an online endpoint
endpoint = ManagedOnlineEndpoint(
    name=online_endpoint_name,
    description="Online endpoint for MLflow diabetes model",
    auth_mode="key",
)
```

Next, you'll create the endpoint by running the following cell. This may take several minutes. While your endpoint is being created, you can read about what are Azure Machine Learning endpoints.

In [ ]:

```
ml_client.begin_create_or_update(endpoint).result()
```

Configure the deployment

You can deploy multiple models to an endpoint. This is mostly useful when you want to update the deployed model while keeping the current model in production. You'll need to configure the deployment to specify which model needs to be deployed to an endpoint. In the following cell, you'll refer to the model trained and stored in the local `model` folder (stored in the same folder as this notebook). Note that since you're working with an MLflow model, you don't need to specify the environment or scoring script.

You'll also specify the infrastructure needed for the model to be deployed.

In [ ]:

```
from azure.ai.ml.entities import Model, ManagedOnlineDeployment
from azure.ai.ml.constants import AssetTypes

# create a blue deployment
model = Model(
    path="./model",
    type=AssetTypes.MLFLOW_MODEL,
    description="my sample mlflow model",
)

blue_deployment = ManagedOnlineDeployment(
    name="blue",
    endpoint_name=online_endpoint_name,
    model=model,
    instance_type="Standard_F4s_v2",
    instance_count=1,
)
```

Create the deployment

Finally, you can actually deploy the model to the endpoint by running the following cell:

In [ ]:

```
ml_client.online_deployments.begin_create_or_update(blue_deployment).result()
```

The deployment of the model may take 10-15 minutes. While waiting for the model to be deployed, you can learn more about managed endpoints in this video.

Since you only have one model deployed to the endpoint, you want this deployment to take 100% of the traffic. If you deploy multiple models to the endpoint, you could use the same approach to distribute traffic across the deployed models.

deployed models.

In [ ]:

```
# blue deployment takes 100 traffic
endpoint.traffic = {"blue": 100}
ml_client.begin_create_or_update(endpoint).result()
```

Test the deployment

Let's test the deployed model by invoking the endpoint. A JSON file with sample data is used as input. The trained model predicts whether a patient has diabetes or not, based on medical data like age, BMI, and the number of pregnancies. A `[0]` indicates a patient doesn't have diabetes. A `[1]` means a patient does have diabetes.

In [ ]:

```
# test the blue deployment with some sample data
response = ml_client.online_endpoints.invoke(
    endpoint_name=online_endpoint_name,
    deployment_name="blue",
    request_file="sample-data.json",
)

if response[1]=='1':
    print("Diabetic")
else:
    print ("Not diabetic")
```

Optionally, you can change the values in the `sample-data.json` file to try and get a different prediction.

List endpoints

Although you can view all endpoints in the Studio, you can also list all endpoints using the SDK:

In [ ]:

```
endpoints = ml_client.online_endpoints.list()
for endp in endpoints:
    print(endp.name)
```

Get endpoint details

If you want more information about a specific endpoint, you can explore the details using the SDK too.

In [ ]:

```
# Get the details for online endpoint
endpoint = ml_client.online_endpoints.get(name=online_endpoint_name)

# existing traffic details
print(endpoint.traffic)

# Get the scoring URI
print(endpoint.scoring_uri)
```

Delete the endpoint and deployment

As an endpoint is always available, it can't be paused to save costs. To avoid unnecessary costs, delete the endpoint.

In [ ]:

```
ml_client.online_endpoints.begin_delete(name=online_endpoint_name)
```