Customize a model with fine-tuning

Article • 11/17/2023

Azure OpenAI Service lets you tailor our models to your personal datasets by using a process known as *fine-tuning*. This customization step lets you get more out of the service by providing:

- Higher quality results than what you can get just from prompt engineering
- The ability to train on more examples than can fit into a model's max request context limit.
- Lower-latency requests, particularly when using smaller models.

A fine-tuned model improves on the few-shot learning approach by training the model's weights on your own data. A customized model lets you achieve better results on a wider number of tasks without needing to provide examples in your prompt. The result is less text sent and fewer tokens processed on every API call, potentially saving cost and improving request latency.

Prerequisites

- Read the When to use Azure OpenAI fine-tuning guide.
- An Azure subscription. Create one for free .
- Access granted to Azure OpenAI in the desired Azure subscription.
- An Azure OpenAI resource that's located in a region that supports fine-tuning of the Azure OpenAI model. Check the Model summary table and region availability for the list of available models by region and supported functionality. For more information, see Create a resource and deploy a model with Azure OpenAI.
- Fine-tuning access requires **Cognitive Services OpenAI Contributor**.
- If you do not already have access to view quota, and deploy models in Azure OpenAI Studio you will require additional permissions.

> **Note**
>
> Currently, you must submit an application to access Azure OpenAI Service. To apply for access, complete **this form** .

Models

The following models support fine-tuning:

- `gpt-35-turbo-0613`
- `babbage-002`
- `davinci-002`

Fine-tuning of `gpt-35-turbo-0613` is not available in every region where this model is available for inference. Consult the models page to check which regions currently support fine-tuning.

Review the workflow for Azure OpenAI Studio

Take a moment to review the fine-tuning workflow for using Azure OpenAI Studio:

1. Prepare your training and validation data.
2. Use the **Create custom model** wizard in Azure OpenAI Studio to train your custom model.
   a. Select a base model.
   b. Choose your training data.
   c. Optionally, choose your validation data.
   d. Optionally, configure advanced options for your fine-tuning job.
   e. Review your choices and train your new custom model.
3. Check the status of your custom fine-tuned model.
4. Deploy your custom model for use.
5. Use your custom model.
6. Optionally, analyze your custom model for performance and fit.

Prepare your training and validation data

Your training data and validation data sets consist of input and output examples for how you would like the model to perform.

Different model types require a different format of training data.

gpt-35-turbo 0613

The training and validation data you use **must** be formatted as a JSON Lines (JSONL) document. For `gpt-35-turbo-0613` the fine-tuning dataset must be formatted in the conversational format that is used by the Chat completions API.

If you would like a step-by-step walk-through of fine-tuning a `gpt-35-turbo-0613` model please refer to the Azure OpenAI fine-tuning tutorial

Example file format

JSON

```
{"messages": [{"role": "system", "content": "You are an Xbox customer sup-
port agent whose primary goal is to help users with issues they are expe-
riencing with their Xbox devices. You are friendly and concise. You only
provide factual answers to queries, and do not provide answers that are
not related to Xbox."}, {"role": "user", "content": "Is Xbox better than
PlayStation?"}, {"role": "assistant", "content": "I apologize, but I can-
not provide personal opinions. My primary job is to assist you with any
issues related to your Xbox device. Do you have any Xbox-related issues
that need addressing?"}]}
{"messages": [{"role": "system", "content": "You are an Xbox customer sup-
port agent whose primary goal is to help users with issues they are expe-
riencing with their Xbox devices. You are friendly and concise. You only
provide factual answers to queries, and do not provide answers that are
not related to Xbox."}, {"role": "user", "content": "Can I play PC games
on my Xbox?"}, {"role": "assistant", "content": "While Xbox consoles don't
natively support PC games, they can play an array of Xbox games, many of
which are also available on PC. Can I assist you with finding games com-
patible with your Xbox device?"}]}
{"messages": [{"role": "system", "content": "You are an Xbox customer sup-
port agent whose primary goal is to help users with issues they are expe-
riencing with their Xbox devices. You are friendly and concise. You only
provide factual answers to queries, and do not provide answers that are
not related to Xbox."}, {"role": "user", "content": "I'm having trouble
connecting my Xbox to the Wi-Fi."}, {"role": "assistant", "content": "No
worries, let's go through the network settings on your Xbox. Can you
please tell me what happens when you try to connect it to the Wi-Fi?"}]}
```

In addition to the JSONL format, training and validation data files must be encoded in UTF-8 and include a byte-order mark (BOM). The file must be less than 100 MB in size.

Create your training and validation datasets

The more training examples you have, the better. The minimum number of training examples is 10, but such a small number of examples is often not enough to noticeably influence model responses. OpenAI states it's best practice to have at least 50 high quality training examples. However, it is entirely possible to have a use case that might require 1,000's of high quality training examples to be successful.

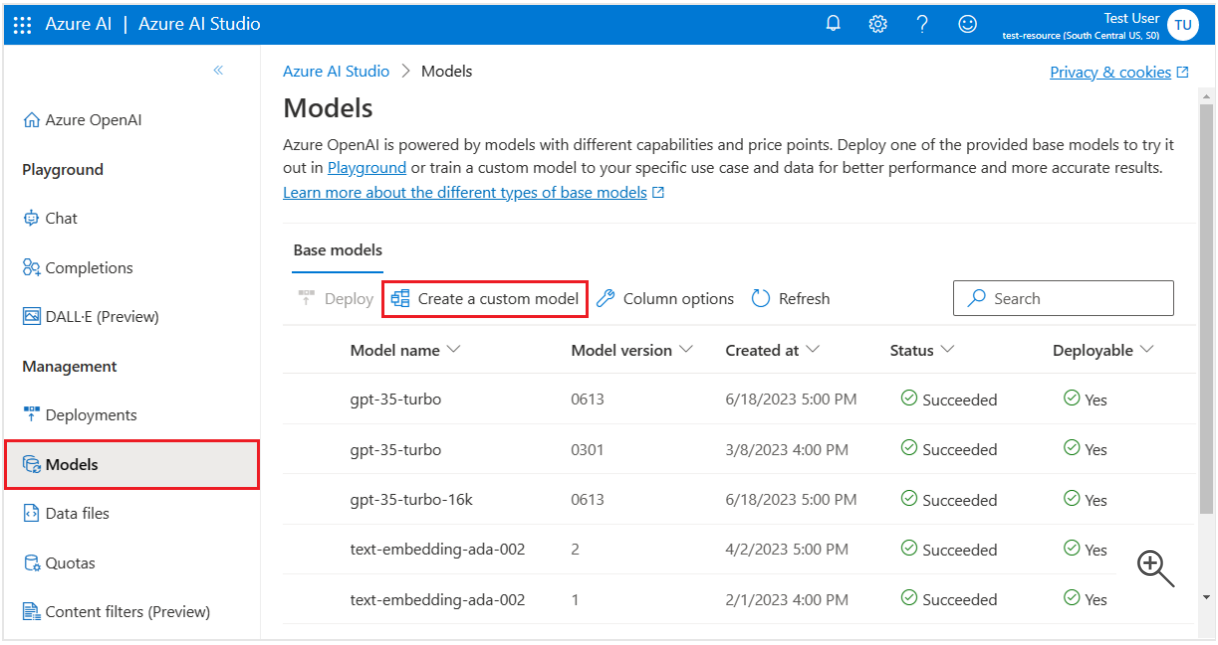
In general, doubling the dataset size can lead to a linear increase in model quality. But keep in mind, low quality examples can negatively impact performance. If you train the model on a large amount of internal data, without first pruning the dataset for only the highest quality examples you could end up with a model that performs much worse than expected.

Use the Create custom model wizard

Azure OpenAI Studio provides the **Create custom model** wizard, so you can interactively create and train a fine-tuned model for your Azure resource.

1. Open Azure OpenAI Studio at https://oai.azure.com/ and sign in with credentials that have access to your Azure OpenAI resource. During the sign-in workflow, select the appropriate directory, Azure subscription, and Azure OpenAI resource.

2. In Azure OpenAI Studio, browse to the **Management** > **Models** pane, and select **Create a custom model**.



The **Create custom model** wizard opens.

Select the base model

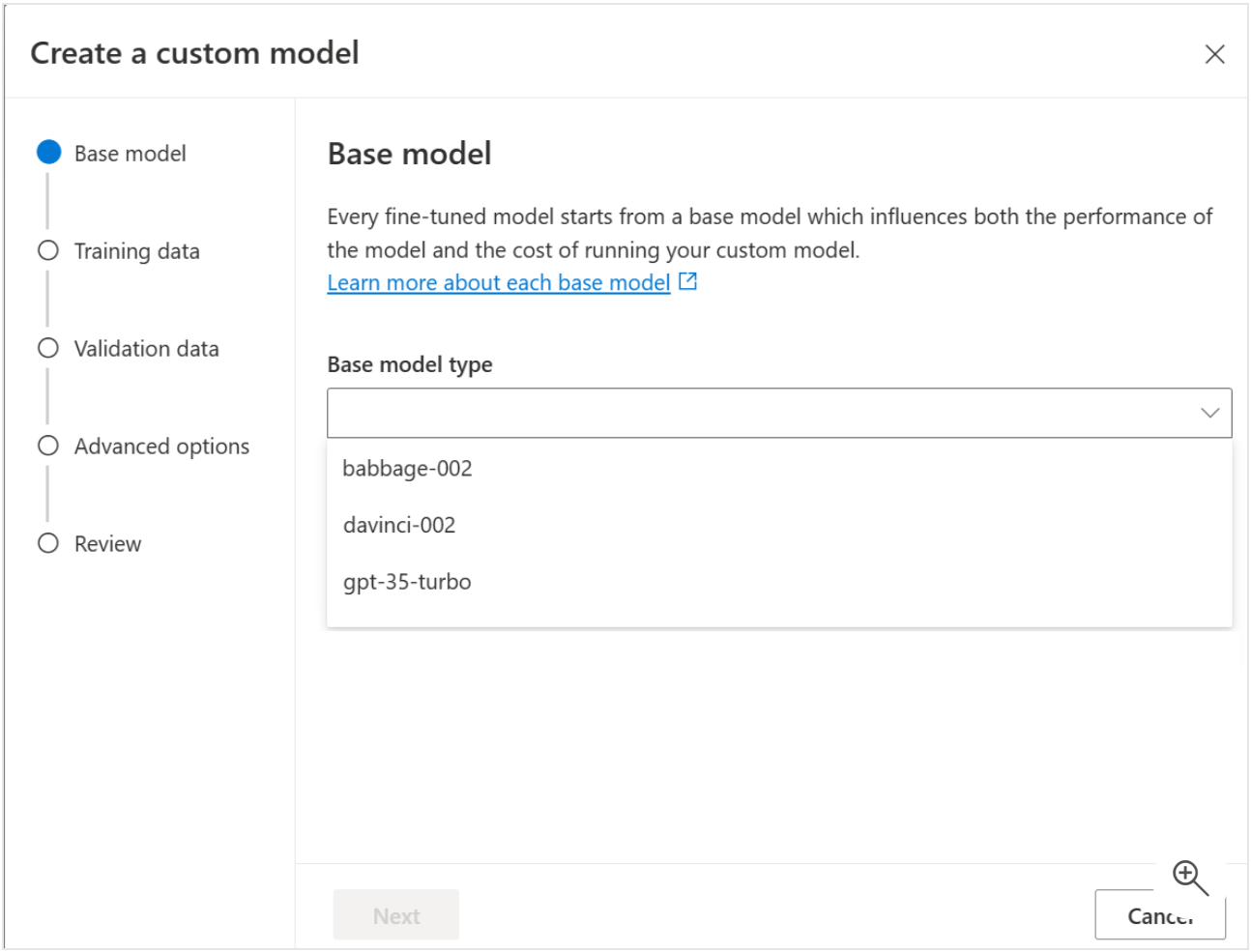
The first step in creating a custom model is to choose a base model. The **Base model** pane lets you choose a base model to use for your custom model. Your choice influences both the performance and the cost of your model.

Select the base model from the **Base model type** dropdown, and then select **Next** to continue.

You can create a custom model from one of the following available base models:

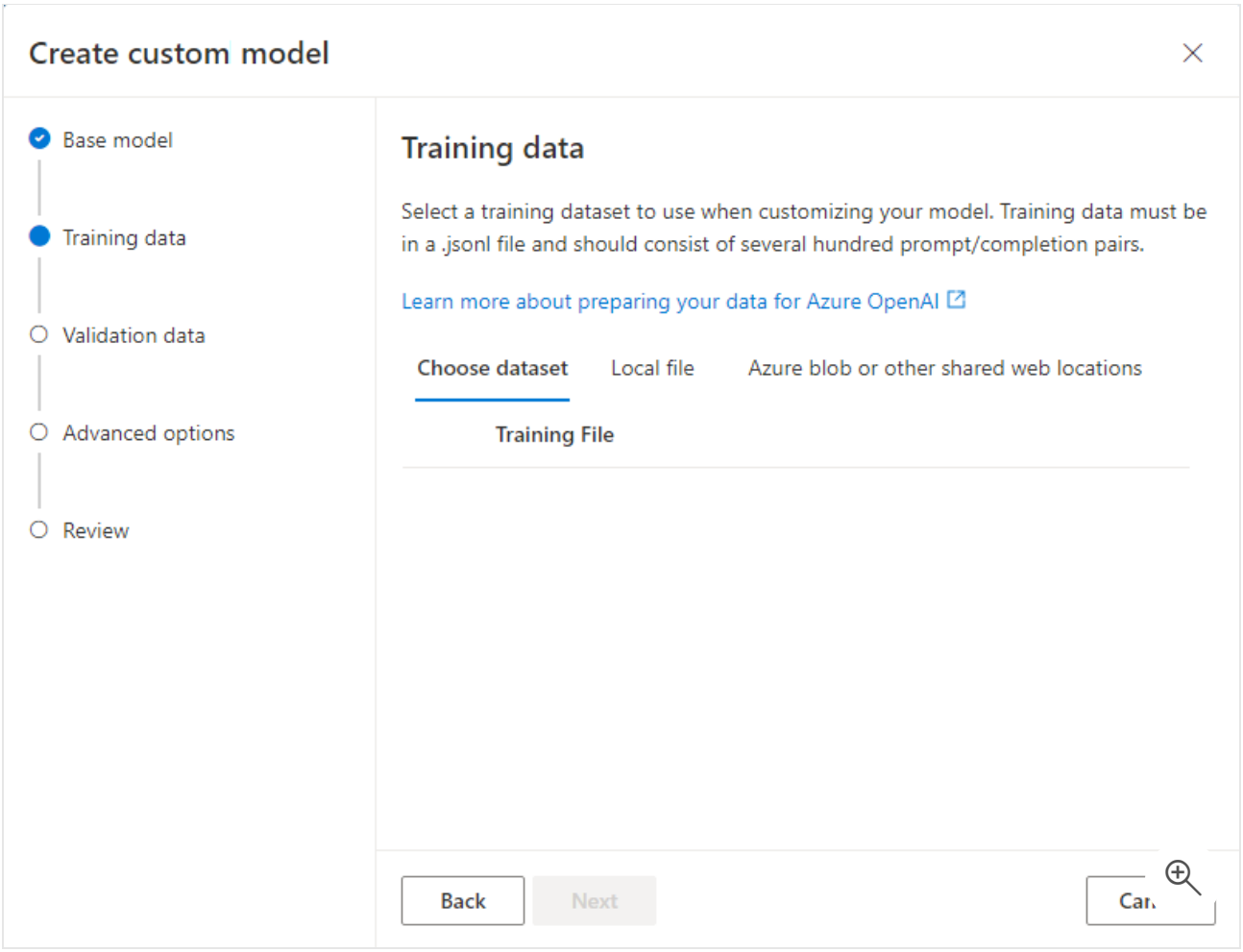
- `babbage-002`
- `davinci-002`
- `gpt-35-turbo`

For more information about our base models that can be fine-tuned, see Models.



Choose your training data

The next step is to either choose existing prepared training data or upload new prepared training data to use when customizing your model. The **Training data** pane displays any existing, previously uploaded datasets and also provides options to upload new training data.



- If your training data is already uploaded to the service, select **Choose dataset**.
  - Select the file from the list shown in the **Training data** pane.
- To upload new training data, use one of the following options:
  - Select **Local file** to upload training data from a local file.
  - Select **Azure blob or other shared web locations** to import training data from Azure Blob or another shared web location.

For large data files, we recommend that you import from an Azure Blob store. Large files can become unstable when uploaded through multipart forms because the requests are atomic and can't be retried or resumed. For more information about Azure Blob Storage, see What is Azure Blob Storage?

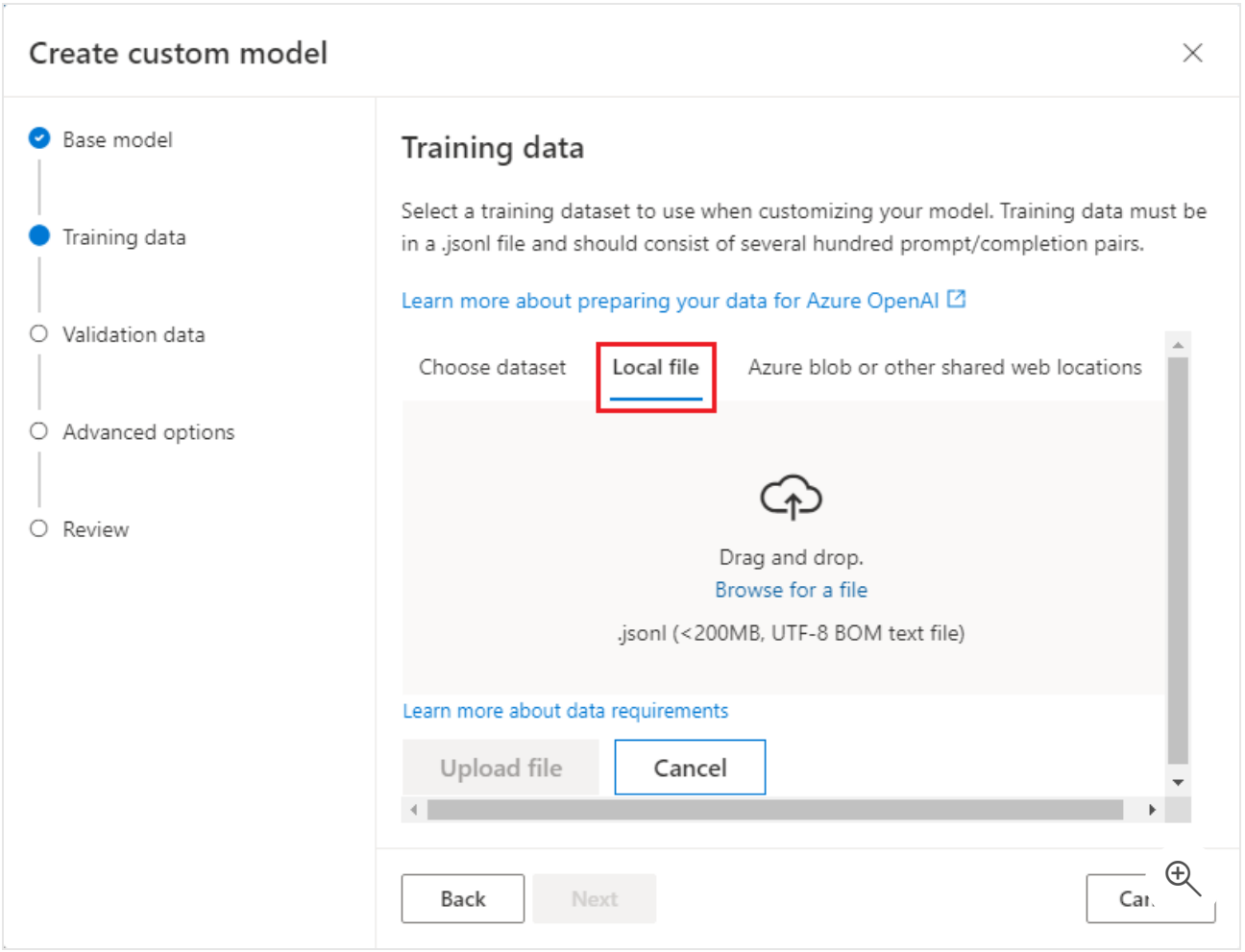
> **Note**
>
> Training data files must be formatted as JSONL files, encoded in UTF-8 with a byte-order mark (BOM). The file must be less than 100 MB in size.

Upload training data from local file

You can upload a new training dataset to the service from a local file by using one of the following methods:

- Drag and drop the file into the client area of the **Training data** pane, and then select **Upload file**.
- Select **Browse for a file** from the client area of the **Training data** pane, choose the file to upload from the **Open** dialog, and then select **Upload file**.

After you select and upload the training dataset, select **Next** to continue.

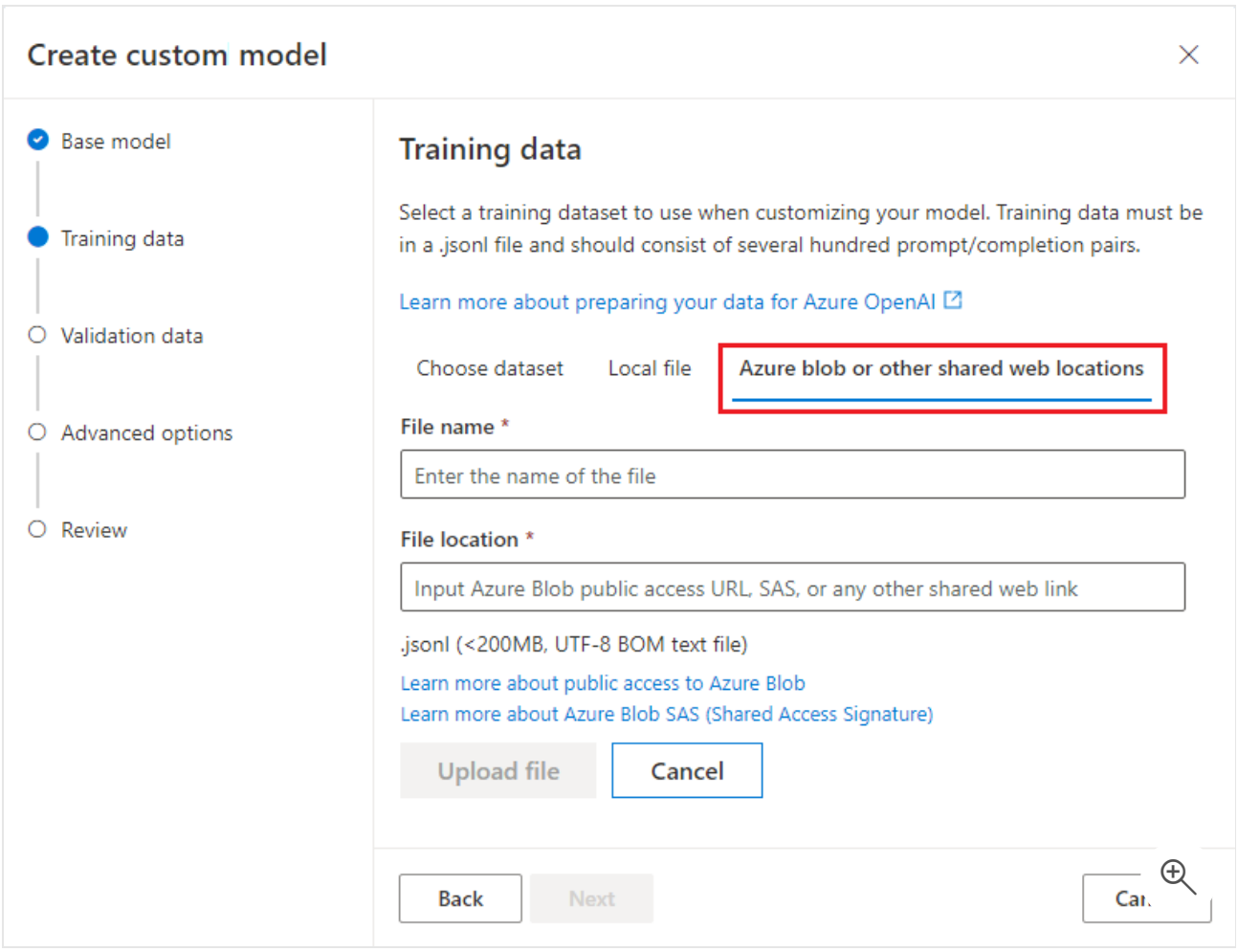


Import training data from Azure Blob store

You can import a training dataset from Azure Blob or another shared web location by providing the name and location of the file.

1. Enter the **File name** for the file.
2. For the **File location**, provide the Azure Blob URL, the Azure Storage shared access signature (SAS), or other link to an accessible shared web location.
3. Select **Upload file** to import the training dataset to the service.

After you select and upload the training dataset, select **Next** to continue.

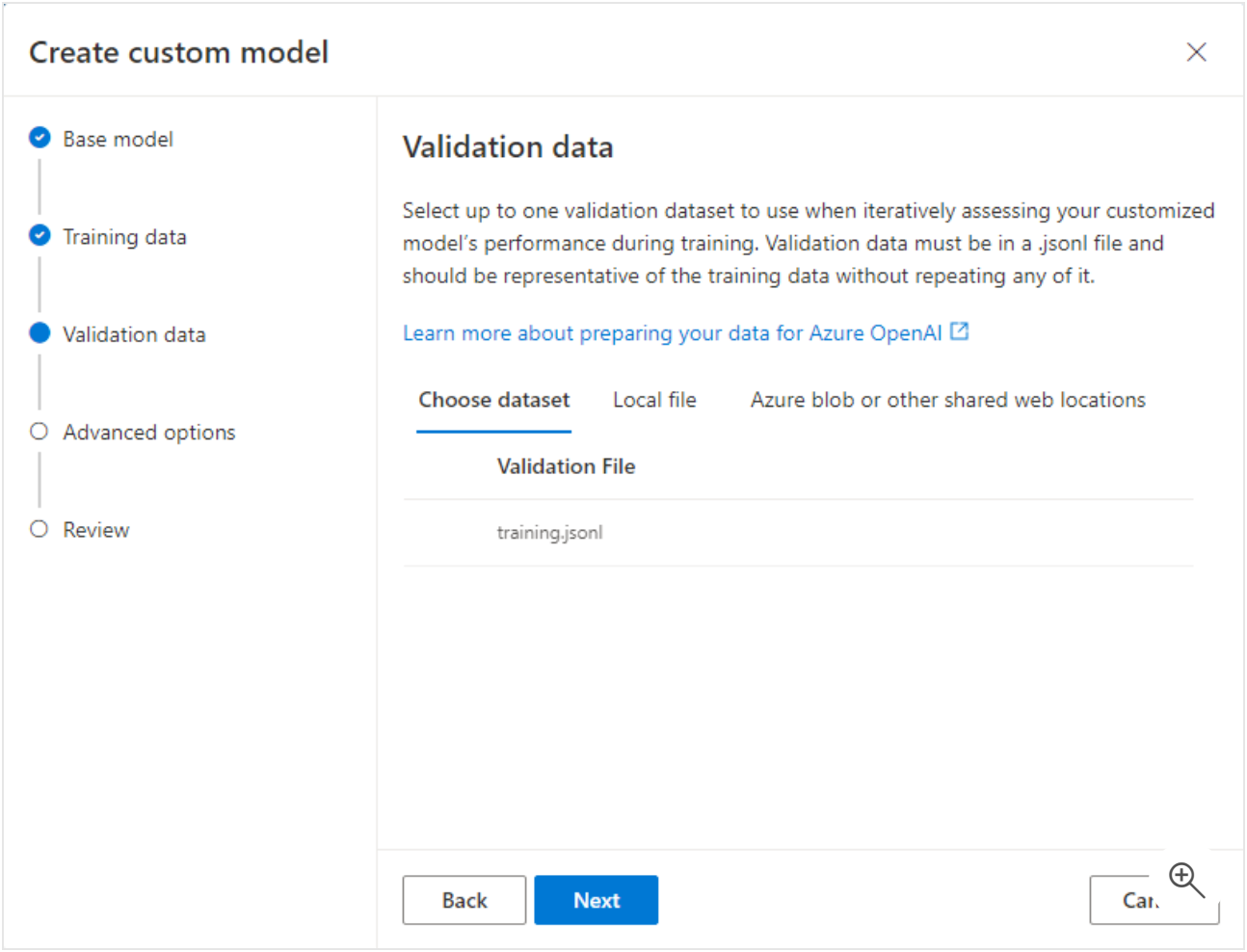


Choose your validation data

The next step provides options to configure the model to use validation data in the training process. If you don't want to use validation data, you can choose **Next** to continue to the advanced options for the model. Otherwise, if you have a validation dataset, you can either

choose existing prepared validation data or upload new prepared validation data to use when customizing your model.

The **Validation data** pane displays any existing, previously uploaded training and validation datasets and provides options by which you can upload new validation data.



- If your validation data is already uploaded to the service, select **Choose dataset**.
  - Select the file from the list shown in the **Validation data** pane.
- To upload new validation data, use one of the following options:
  - Select **Local file** to upload validation data from a local file.
  - Select **Azure blob or other shared web locations** to import validation data from Azure Blob or another shared web location.

For large data files, we recommend that you import from an Azure Blob store. Large files can become unstable when uploaded through multipart forms because the requests are atomic and can't be retried or resumed.

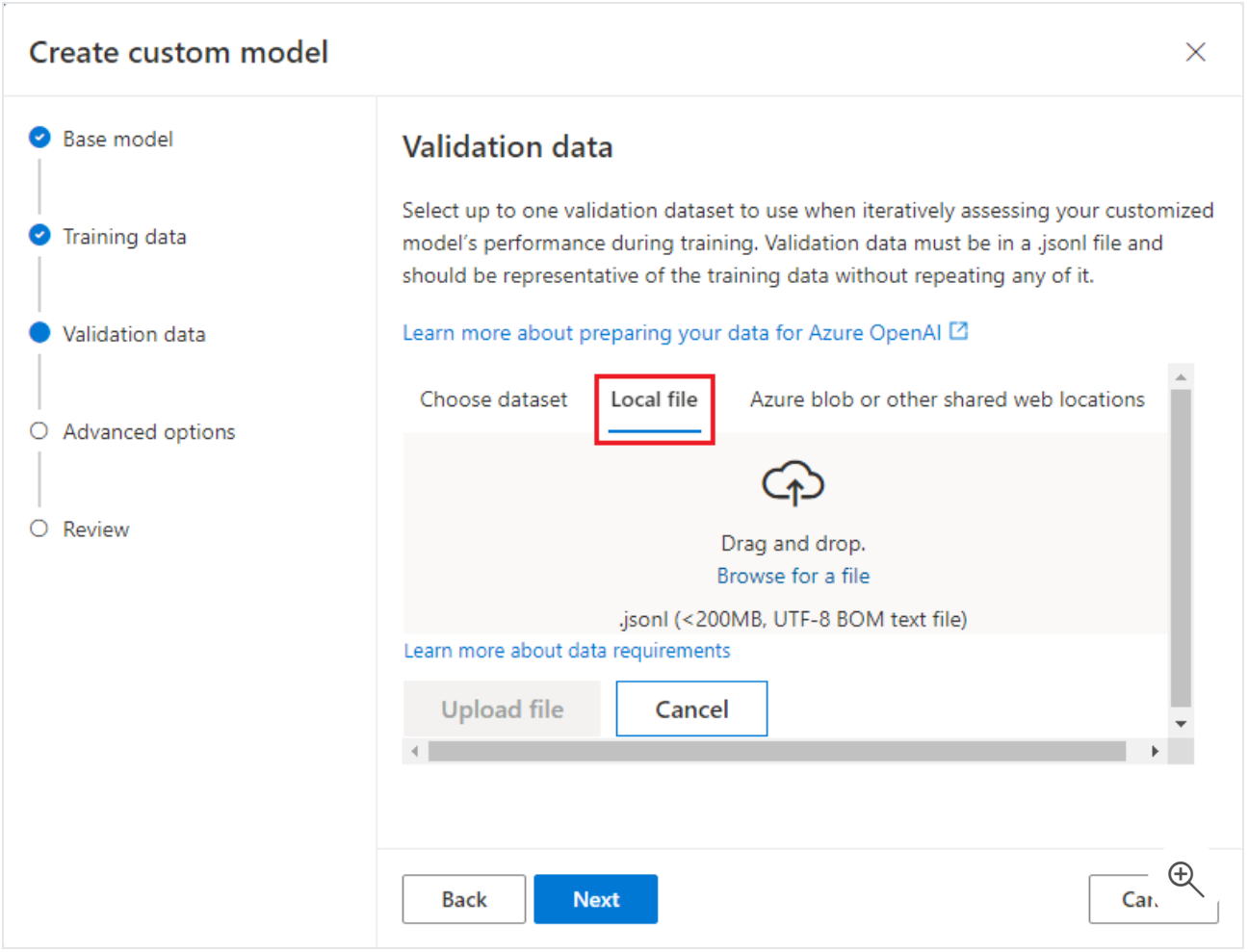
> **Note**
>
> Similar to training data files, validation data files must be formatted as JSONL files, encoded in UTF-8 with a byte-order mark (BOM). The file must be less than 100 MB in size.

Upload validation data from local file

You can upload a new validation dataset to the service from a local file by using one of the following methods:

- Drag and drop the file into the client area of the **Validation data** pane, and then select **Upload file**.
- Select **Browse for a file** from the client area of the **Validation data** pane, choose the file to upload from the **Open** dialog, and then select **Upload file**.

After you select and upload the validation dataset, select **Next** to continue.

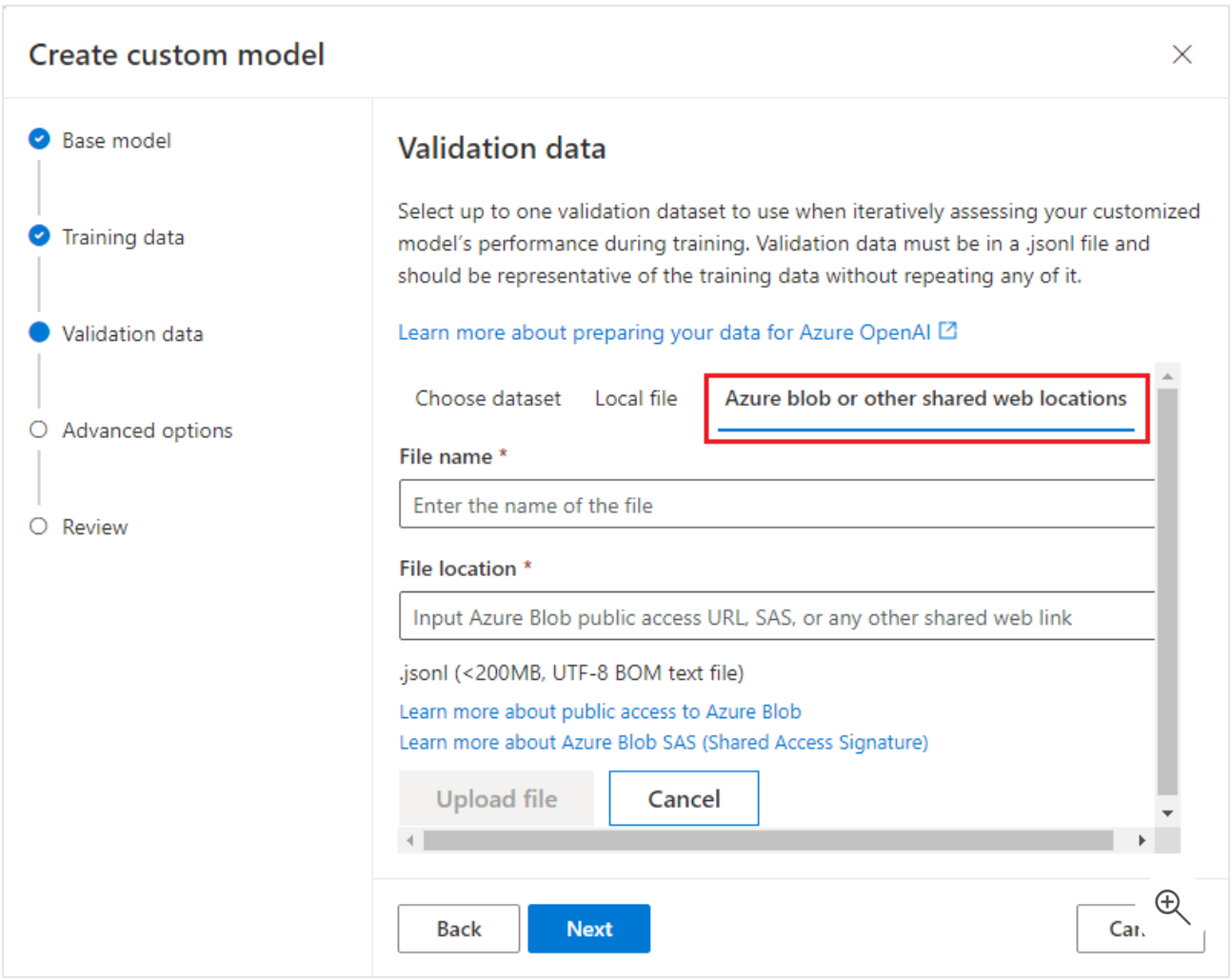


Import validation data from Azure Blob store

You can import a validation dataset from Azure Blob or another shared web location by providing the name and location of the file.

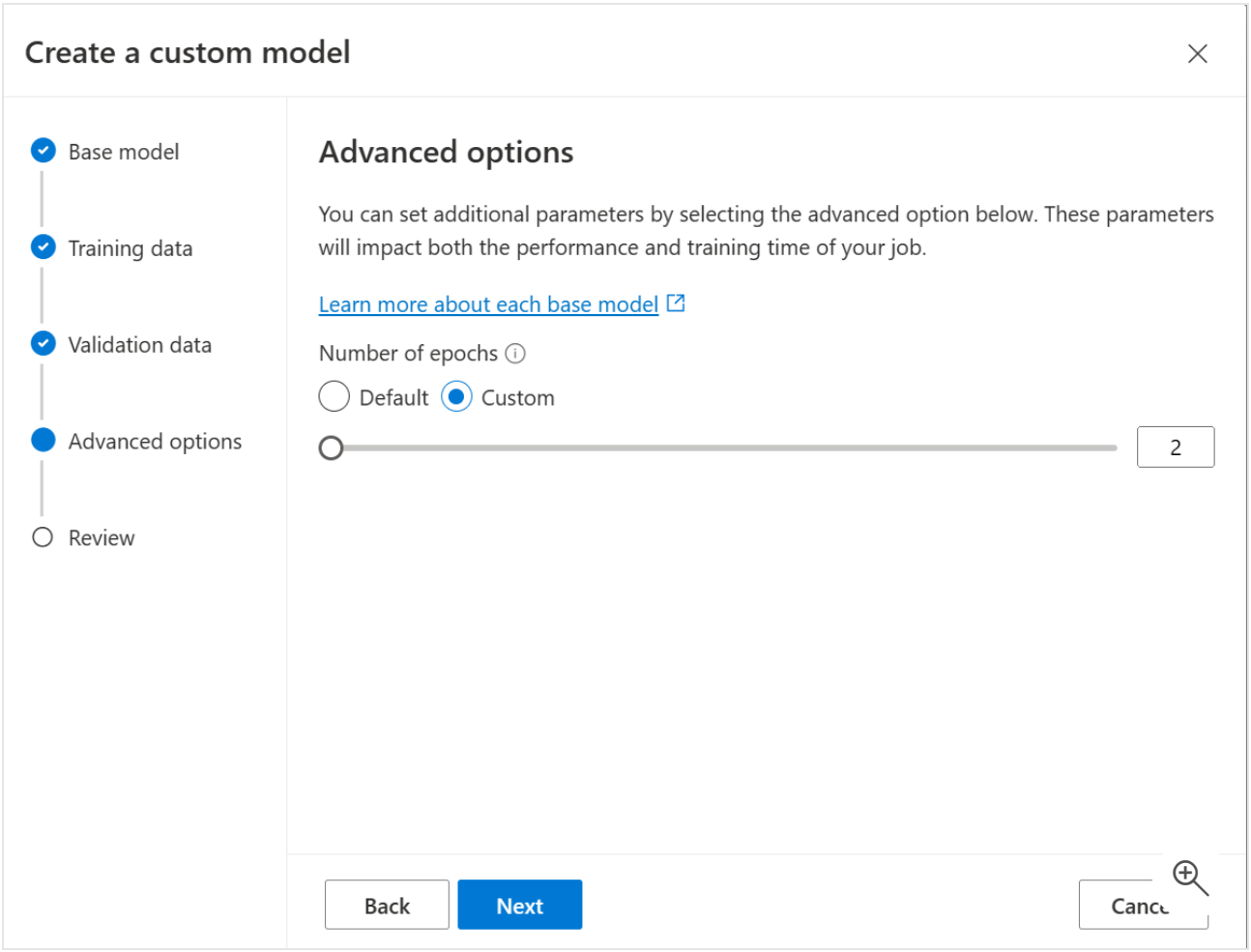
1. Enter the **File name** for the file.
2. For the **File location**, provide the Azure Blob URL, the Azure Storage shared access signature (SAS), or other link to an accessible shared web location.
3. Select **Upload file** to import the training dataset to the service.

After you select and upload the validation dataset, select **Next** to continue.



Configure advanced options

The **Create custom model** wizard shows the hyperparameters for training your fine-tuned model on the **Advanced options** pane. The following hyperparameter is currently available:



Select **Default** to use the default values for the fine-tuning job, or select **Advanced** to display and edit the hyperparameter values.

The **Advanced** option lets you configure the following hyperparameter:

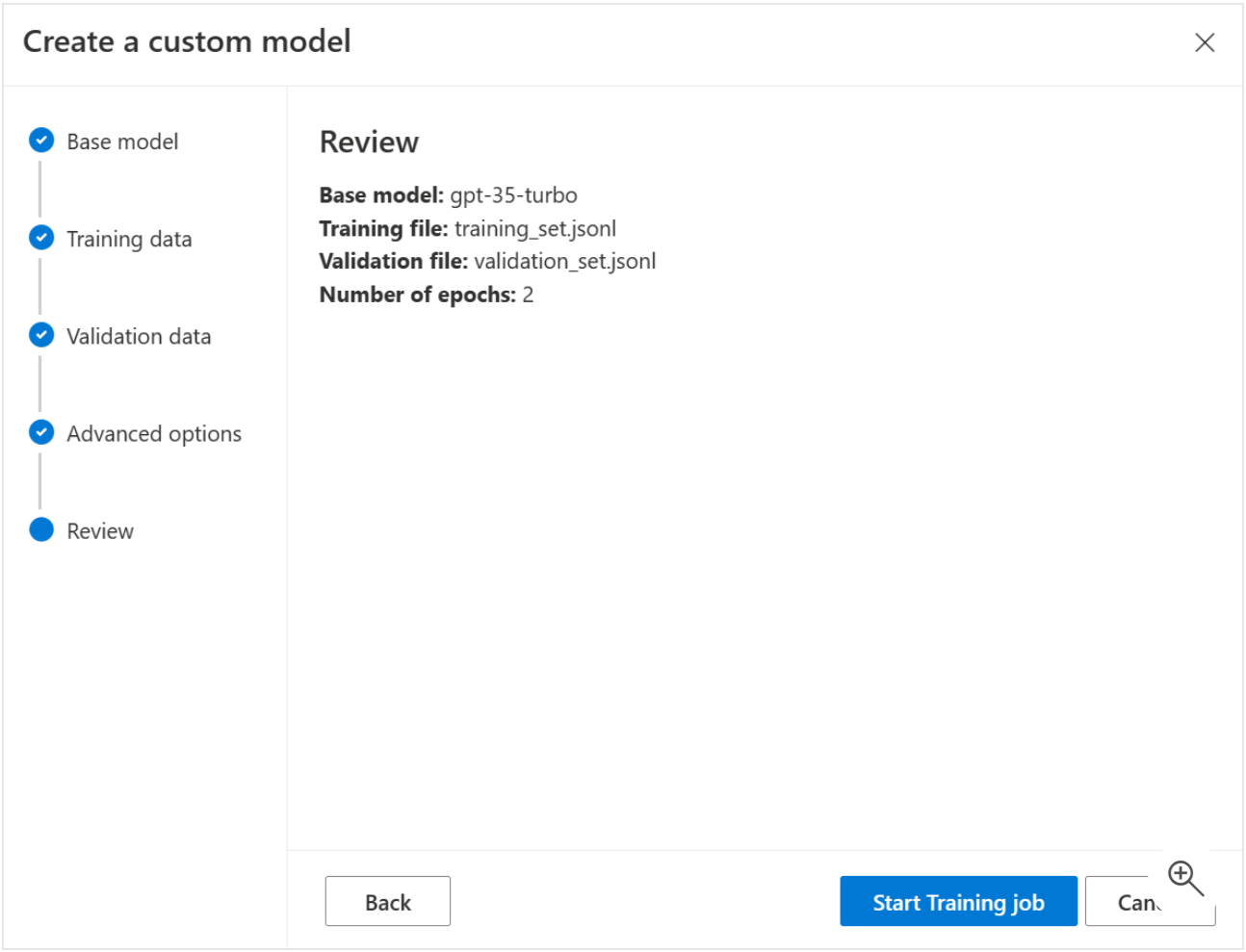
Expand table

Parameter name	Description
Number of epochs	The number of epochs to use for training the model. An epoch refers to one full cycle through the training dataset.

After you configure the advanced options, select **Next** to review your choices and train your fine-tuned model.

Review your choices and train your model

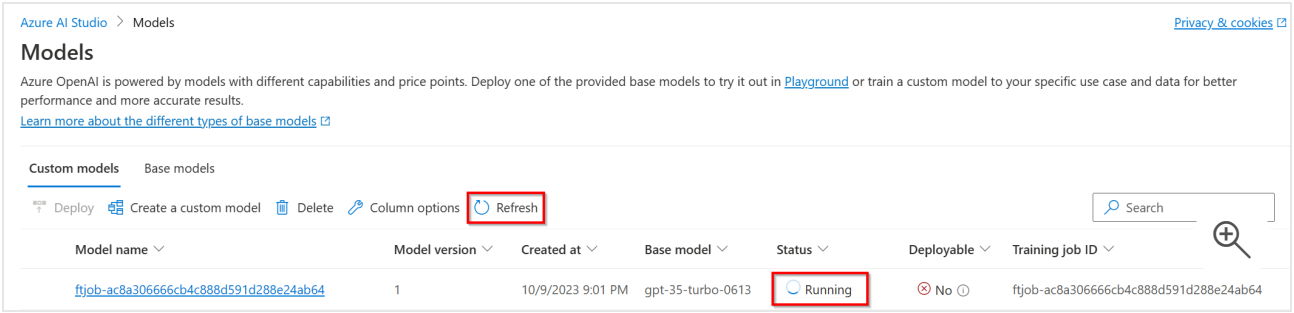
The **Review** pane of the wizard displays information about your configuration choices.



If you're ready to train your model, select **Start Training job** to start the fine-tuning job and return to the **Models** pane.

Check the status of your custom model

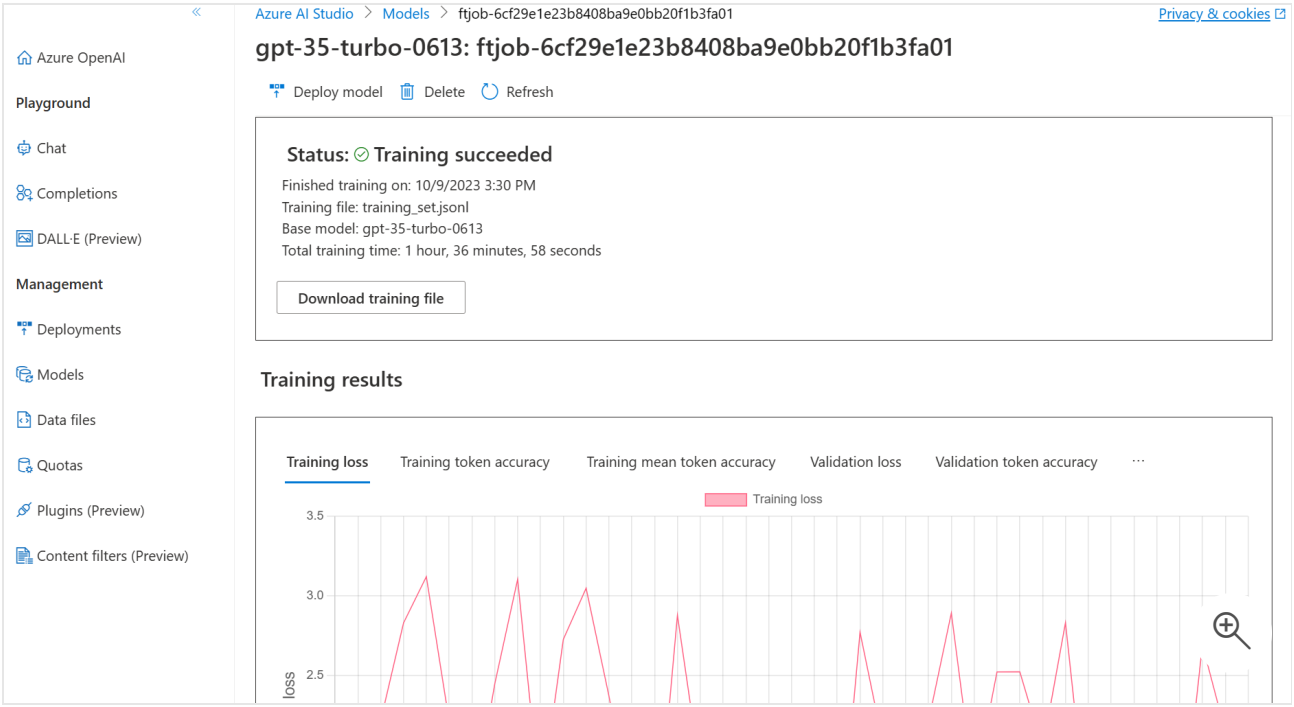
The **Models** pane displays information about your custom model in the **Customized models** tab. The tab includes information about the status and job ID of the fine-tune job for your custom model. When the job completes, the tab displays the file ID of the result file. You might need to select **Refresh** in order to see an updated status for the model training job.



After you start a fine-tuning job, it can take some time to complete. Your job might be queued behind other jobs on the system. Training your model can take minutes or hours depending on the model and dataset size.

Here are some of the tasks you can do on the **Models** pane:

- Check the status of the fine-tuning job for your custom model in the **Status** column of the **Customized models** tab.
- In the **Model name** column, select the model name to view more information about the custom model. You can see the status of the fine-tuning job, training results, training events, and hyperparameters used in the job.
- Select **Download training file** to download the training data you used for the model.
- Select **Download results** to download the result file attached to the fine-tuning job for your model and analyze your custom model for training and validation performance.
- Select **Refresh** to update the information on the page.



Deploy a custom model

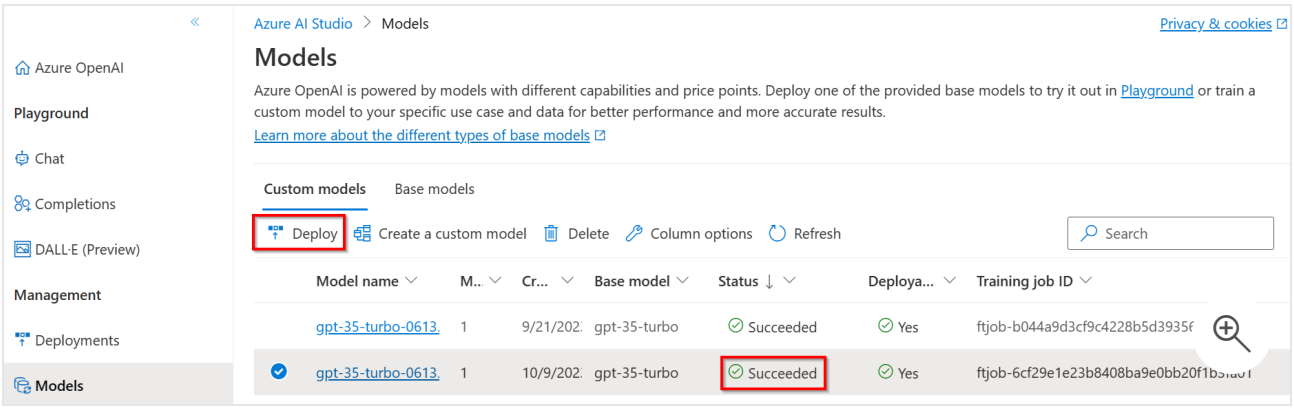
When the fine-tuning job succeeds, you can deploy the custom model from the **Models** pane. You must deploy your custom model to make it available for use with completion

calls.

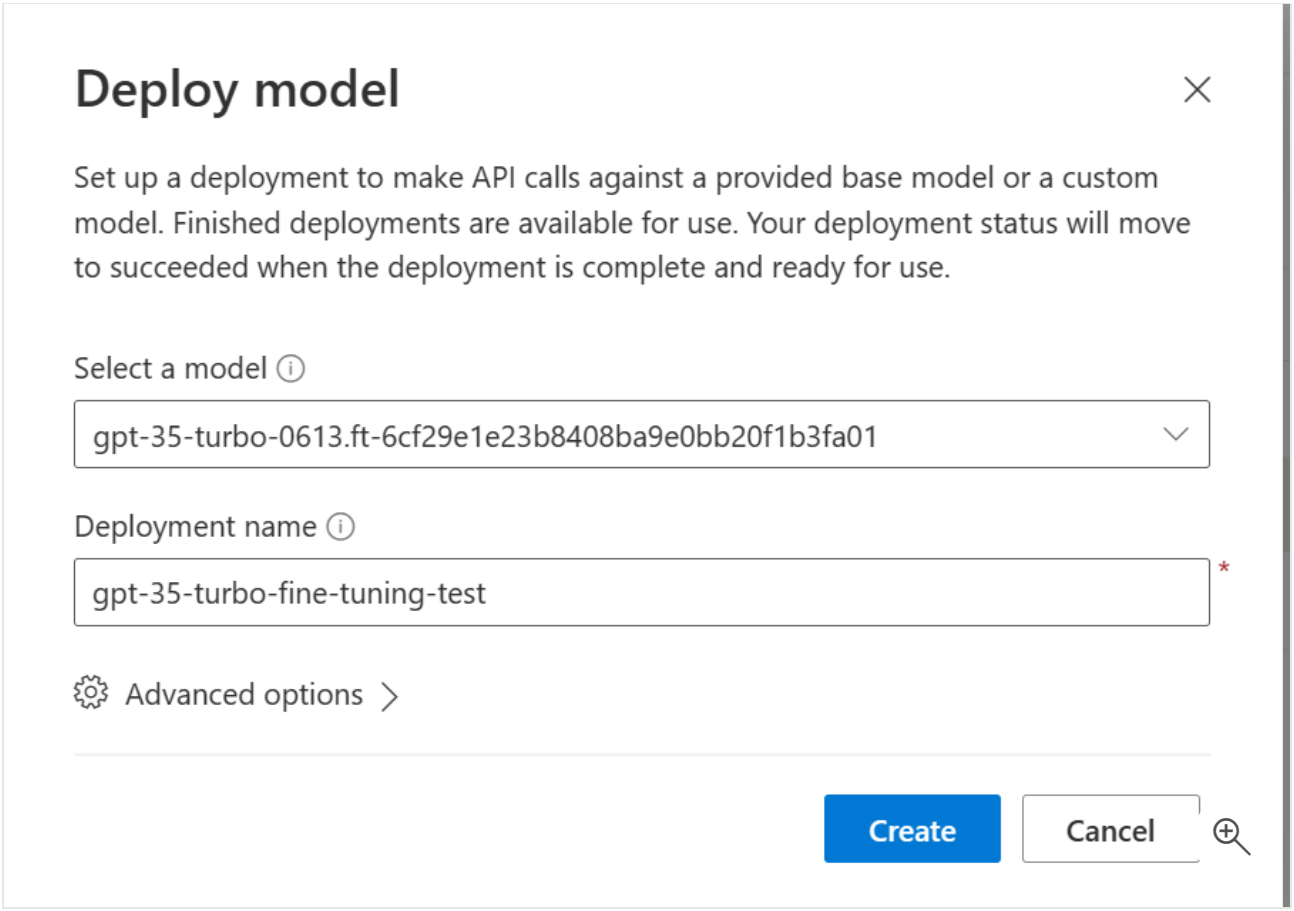
> **Important**
>
> After you deploy a customized model, if at any time the deployment remains inactive for greater than fifteen (15) days, the deployment is deleted. The deployment of a customized model is *inactive* if the model was deployed more than fifteen (15) days ago and no completions or chat completions calls were made to it during a continuous 15-day period.
>
> The deletion of an inactive deployment doesn't delete or affect the underlying customized model, and the customized model can be redeployed at any time. As described in **Azure OpenAI Service pricing**, each customized (fine-tuned) model that's deployed incurs an hourly hosting cost regardless of whether completions or chat completions calls are being made to the model. To learn more about planning and managing costs with Azure OpenAI, refer to the guidance in **Plan to manage costs for Azure OpenAI Service**.

> **Note**
>
> Only one deployment is permitted for a custom model. An error message is displayed if you select an already-deployed custom model.

To deploy your custom model, select the custom model to deploy, and then select **Deploy model**.



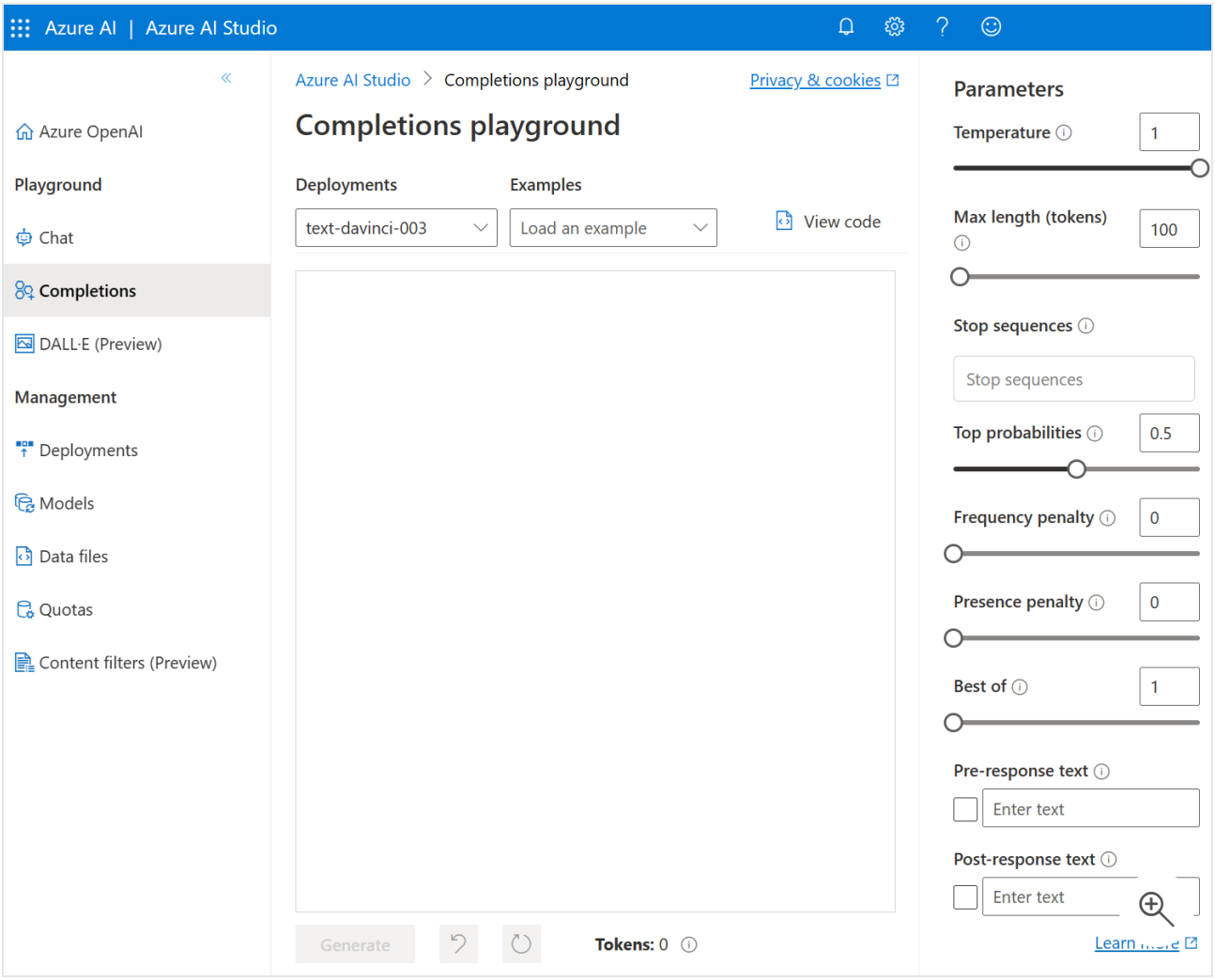
The **Deploy model** dialog box opens. In the dialog box, enter your **Deployment name** and then select **Create** to start the deployment of your custom model.



You can monitor the progress of your deployment on the **Deployments** pane in Azure OpenAI Studio.

Use a deployed custom model

After your custom model deploys, you can use it like any other deployed model. You can use the **Playgrounds** in Azure OpenAI Studio to experiment with your new deployment. You can continue to use the same parameters with your custom model, such as `temperature` and `max_tokens`, as you can with other deployed models. For fine-tuned `babbage-002` and `davinci-002` models you will use the Completions playground and the Completions API. For fine-tuned `gpt-35-turbo-0613` models you will use the Chat playground and the Chat completion API.



Analyze your custom model

Azure OpenAI attaches a result file named *results.csv* to each fine-tuning job after it completes. You can use the result file to analyze the training and validation performance of your custom model. The file ID for the result file is listed for each custom model in the **Result file Id** column on the **Models** pane for Azure OpenAI Studio. You can use the file ID to identify and download the result file from the **Data files** pane of Azure OpenAI Studio.

The result file is a CSV file that contains a header row and a row for each training step performed by the fine-tuning job. The result file contains the following columns:

Expand table

Column name	Description
`step`	The number of the training step. A training step represents a single pass, forward and backward, on a batch of training data.

Column name	Description
`train_loss`	The loss for the training batch.
`training_accuracy`	The percentage of completions in the training batch for which the model's predicted tokens exactly matched the true completion tokens. For example, if the batch size is set to 3 and your data contains completions `[[1, 2], [0, 5], [4, 2]]`, this value is set to 0.67 (2 of 3) if the model predicted `[[1, 1], [0, 5], [4, 2]]`.
`train_mean_token_accuracy`	The percentage of tokens in the training batch correctly predicted by the model. For example, if the batch size is set to 3 and your data contains completions `[[1, 2], [0, 5], [4, 2]]`, this value is set to 0.83 (5 of 6) if the model predicted `[[1, 1], [0, 5], [4, 2]]`.
`valid_loss`	The loss for the validation batch.
`valid_accuracy`	The percentage of completions in the validation batch for which the model's predicted tokens exactly matched the true completion tokens. For example, if the batch size is set to 3 and your data contains completions `[[1, 2], [0, 5], [4, 2]]`, this value is set to 0.67 (2 of 3) if the model predicted `[[1, 1], [0, 5], [4, 2]]`.
`validation_mean_token_accuracy`	The percentage of tokens in the validation batch correctly predicted by the model. For example, if the batch size is set to 3 and your data contains completions `[[1, 2], [0, 5], [4, 2]]`, this value is set to 0.83 (5 of 6) if the model predicted `[[1, 1], [0, 5], [4, 2]]`.

Clean up your deployments, custom models, and training files

When you're done with your custom model, you can delete the deployment and model. You can also delete the training and validation files you uploaded to the service, if needed.

Delete your model deployment

> **Important**

After you deploy a customized model, if at any time the deployment remains inactive for greater than fifteen (15) days, the deployment is deleted. The deployment of a customized model is *inactive* if the model was deployed more than fifteen (15) days ago and no completions or chat completions calls were made to it during a continuous 15-day period.

The deletion of an inactive deployment doesn't delete or affect the underlying customized model, and the customized model can be redeployed at any time. As described in **Azure OpenAI Service pricing** , each customized (fine-tuned) model that's deployed incurs an hourly hosting cost regardless of whether completions or chat completions calls are being made to the model. To learn more about planning and managing costs with Azure OpenAI, refer to the guidance in **Plan to manage costs for Azure OpenAI Service**.

You can delete the deployment for your custom model on the **Deployments** pane in Azure OpenAI Studio. Select the deployment to delete, and then select **Delete** to delete the deployment.

Delete your custom model

You can delete a custom model on the **Models** pane in Azure OpenAI Studio. Select the custom model to delete from the **Customized models** tab, and then select **Delete** to delete the custom model.

> ⓘ **Note**
>
> You can't delete a custom model if it has an existing deployment. You must first **delete your model deployment** before you can delete your custom model.

Delete your training files

You can optionally delete training and validation files that you uploaded for training, and result files generated during training, on the **Management** > **Data files** pane in Azure OpenAI Studio. Select the file to delete, and then select **Delete** to delete the file.

Troubleshooting

How do I enable fine-tuning? Create a custom model is greyed out in Azure OpenAI Studio?

In order to successfully access fine-tuning you need **Cognitive Services OpenAI Contributor assigned**. Even someone with high-level Service Administrator permissions would still need this account explicitly set in order to access fine-tuning. For more information please review the role-based access control guidance.

Next steps

- Explore the fine-tuning capabilities in the Azure OpenAI fine-tuning tutorial.
- Review fine-tuning model regional availability