.NET Conf 2022

Deep Learning in .NET

Deep Learning in .NET

Kinfey Lo – Microsoft Regional Cloud Advocate
kinfeylo@microsoft.com
https://github.com/Kinfey

Hi

Kinfey Lo

.NET 18 year+

MVP Microsoft Most Valuable Professional 2012-2021

10 year+

Xamarin MOST VALUABLE PROFESSIONAL 2014

7 year+

Microsoft REGIONAL DIRECTOR 2018-2021

Since 2021.08
Microsoft Cloud Advocate

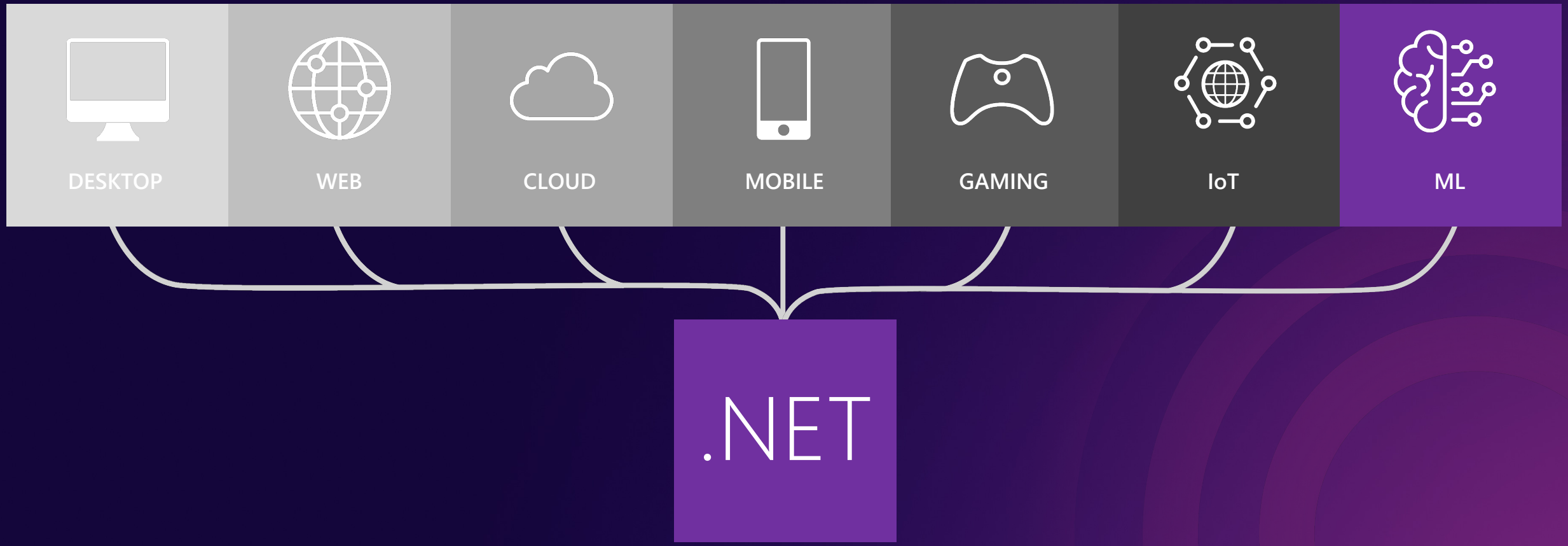
Today we will Talk

- Machine Learning in .NET
- Deep Learning in .NET
- MLOps in .NET

Face

Not a face

.NET can do everything

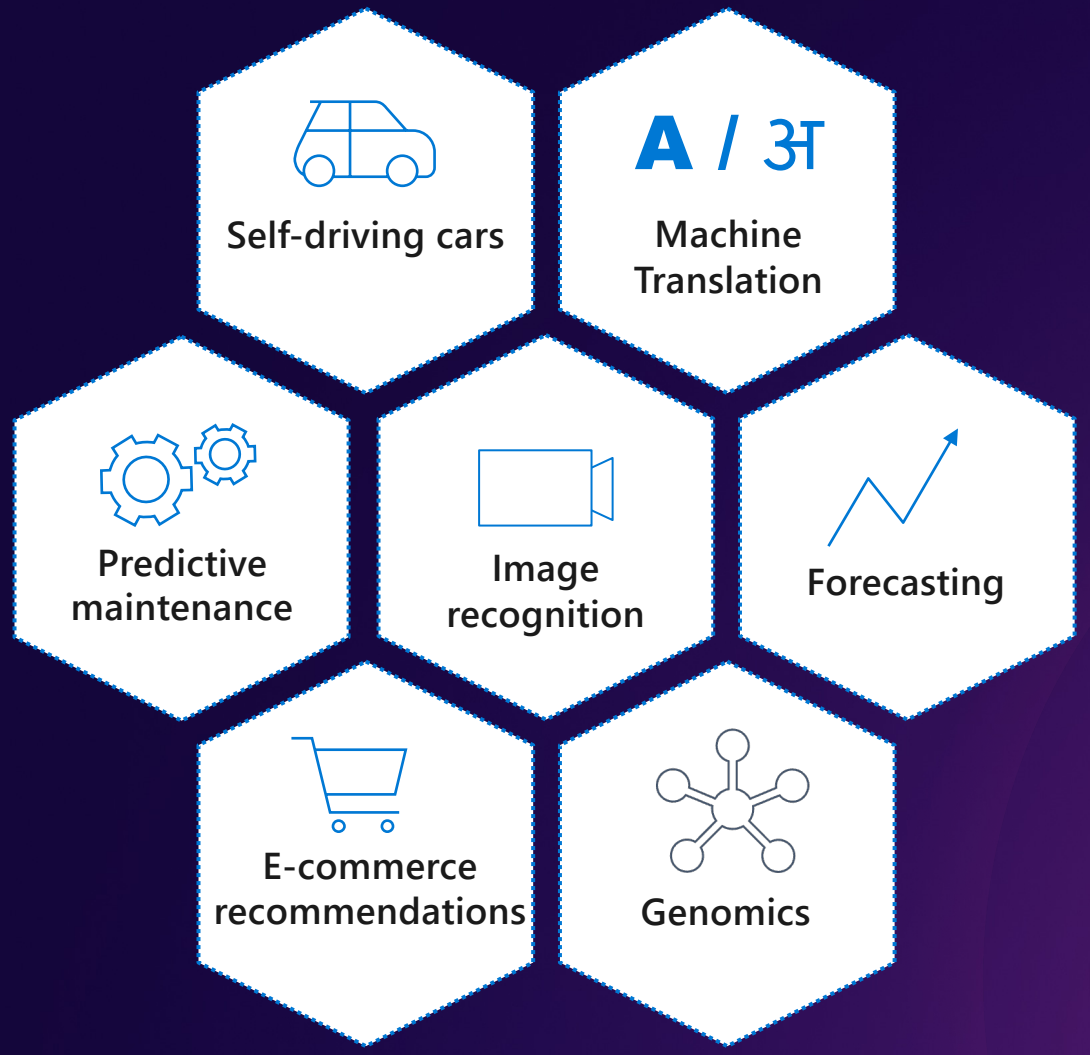


What’s Machine Learning ?

Machine learning (ML) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks.

Machine Learning

Applications

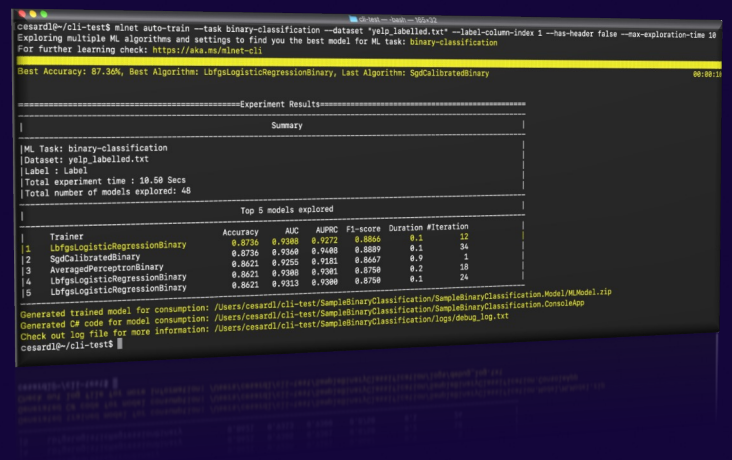


ML.NET

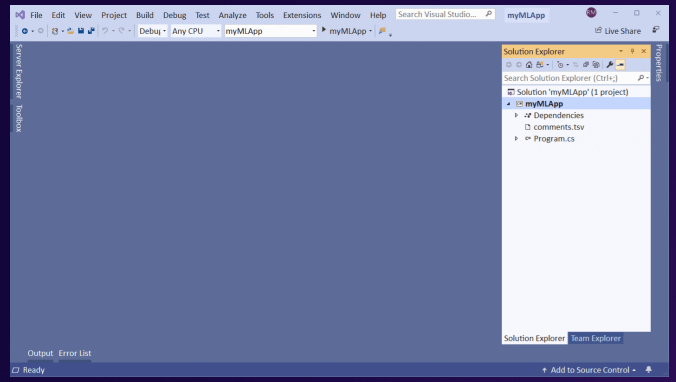
An open source and cross-platform machine learning framework for .NET

ML.NET

ML.NET CLI



Model Builder



AutoML

Can run an AutoML experiment on a given dataset to iterate over different data featurizations, ML algorithms, and hyperparameters to **select the best model**

All .NET Developers

Can use existing C# and F# skills to integrate ML into .NET apps

No data science or ML experience required

Tools and features to help developers easily build, train, and deploy high-quality custom ML models both locally and in Azure

*Goal is not to convert Python or R developers to .NET but instead to target 100% of .NET developers and enable them to add ML to their .NET apps

Developer friendly API for Machine Learning

ML Model Training & Consumption

Data

- **IDataView** (Dataset)
- **File Loaders**
- **Database Loader**
- **Image Loader**
- **Data Transforms**

Training ML Tasks

Classical ML
- Classification
- Regression
- Anomaly Detection
- Recommendation
- Time Series
- Ranking
- Clustering

Computer Vision
- Image Classification
- Object Detection

NLP
- Text Classification
- Sentence Similiar

Model Consumption & Evaluation

Consumption
- Model
- Prediction Engine
- Prediction Engine Pool
- ONNX Export

Evaluation
- Model Evaluators
- Quality Metrics

Extensions & Tools

- **ONNX** Consumption
- **TensorFlow** Consumption
- **AutoML**
- **CLI** (Command Line Interface)
- **Model Builder** in VS
- **Jupyter Notebooks**

What about Deep Learning in .NET ?

Deep Learning in .NET

https://github.com/SciSharp/TensorFlow.NET

TensorFlow.NET (TF.NET) provides a .NET Standard binding for TensorFlow. It aims to implement the complete Tensorflow API in C# which allows .NET developers to develop, train and deploy Machine Learning models with the cross-platform .NET Standard framework. TensorFlow.NET has built-in Keras high-level interface and is released as an independent package TensorFlow.Keras.

```
### install tensorflow C#/F# binding
PM> Install-Package TensorFlow.NET
### install keras for tensorflow
PM> Install-Package TensorFlow.Keras

### Install tensorflow binary
### For CPU version
PM> Install-Package SciSharp.TensorFlow.Redist

### For GPU version (CUDA and cuDNN are required)
PM> Install-Package SciSharp.TensorFlow.Redist-Windows-GPU
```

Deep Learning in .NET

TorchSharp is a .NET library that provides access to the library that powers PyTorch. It is part of the .NET Foundation.

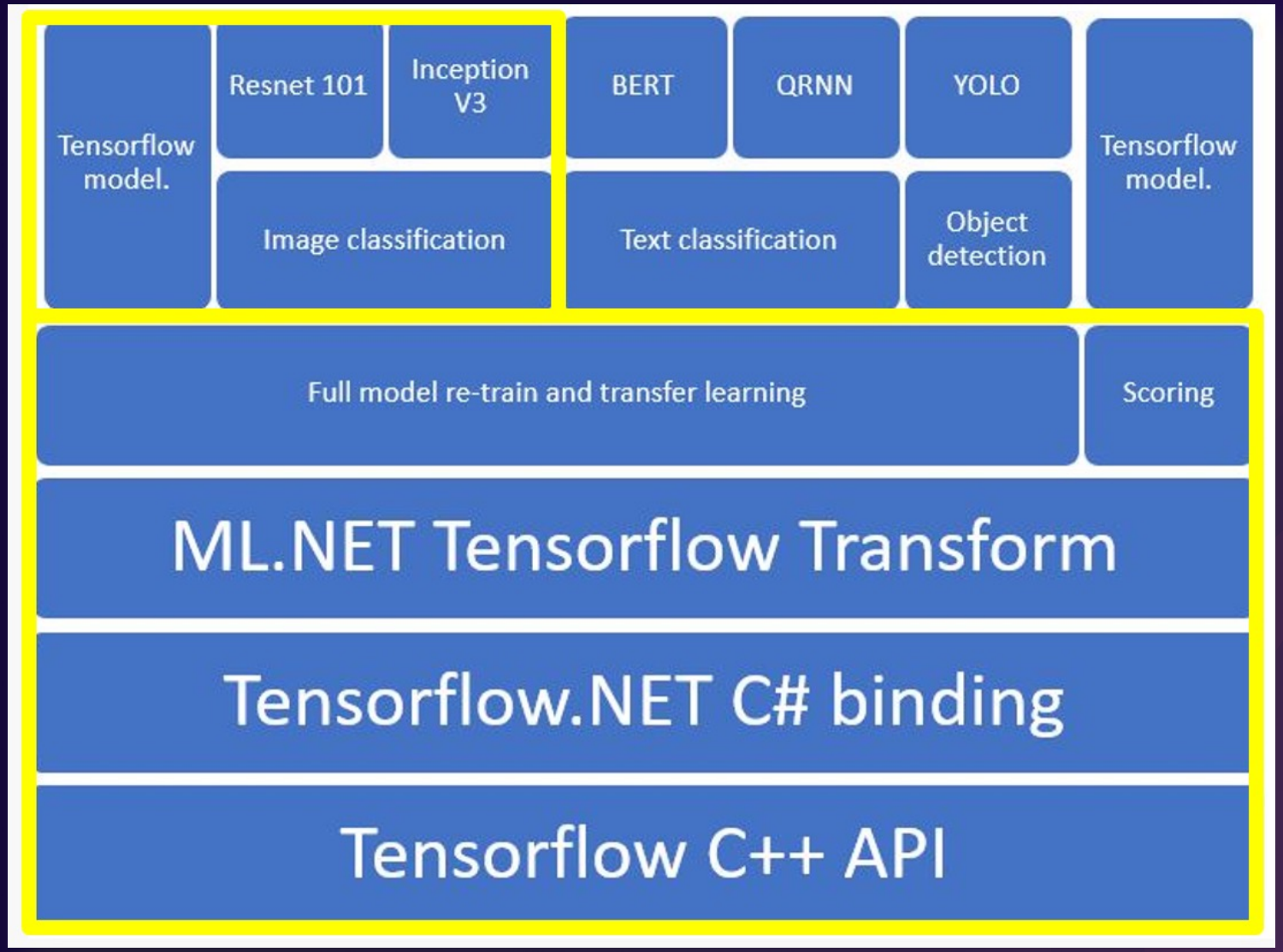
https://github.com/dotnet/TorchSharp

- `libtorch-cpu-linux-x64` (CPU, Linux)
- `libtorch-cpu-win-x64` (CPU, Windows)
- `libtorch-cpu-osx-x64` (CPU, OSX)
- `libtorch-cpu` (CPU, references all three, larger download but simpler)
- `libtorch-cuda-11.7-linux-x64` (CPU/CUDA 11.3, Linux)

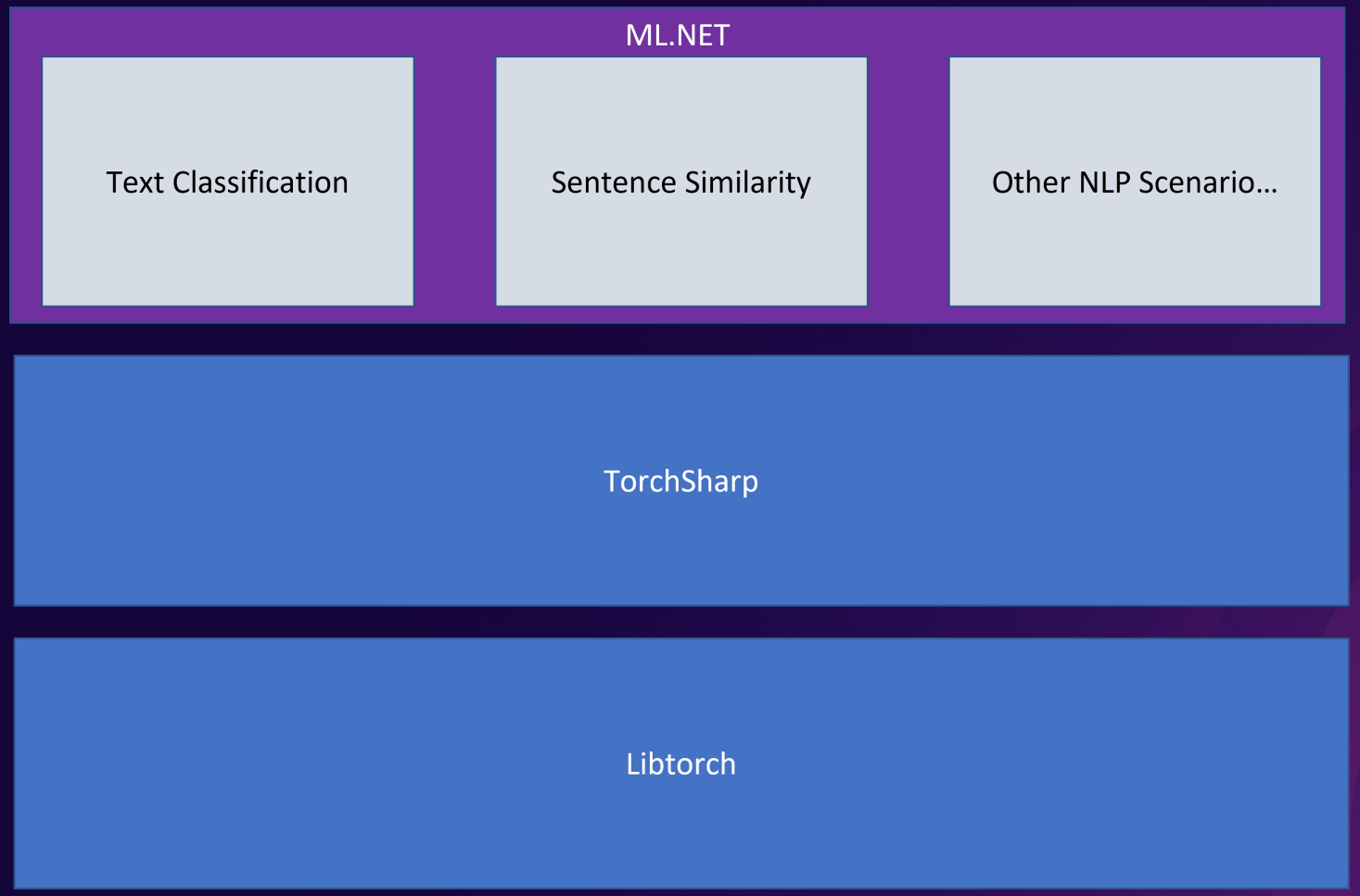
  > NOTE: Due to the presence of very large native binaries, using the `libtorch-cuda-11.7-linux-x64` package requires .NET 6, e.g. .NET SDK version `6.0.100-preview.5.21302.13` or greater.

- `libtorch-cuda-11.7-win-x64` (CPU/CUDA 11.3, Windows)

ML.NET Image Classification

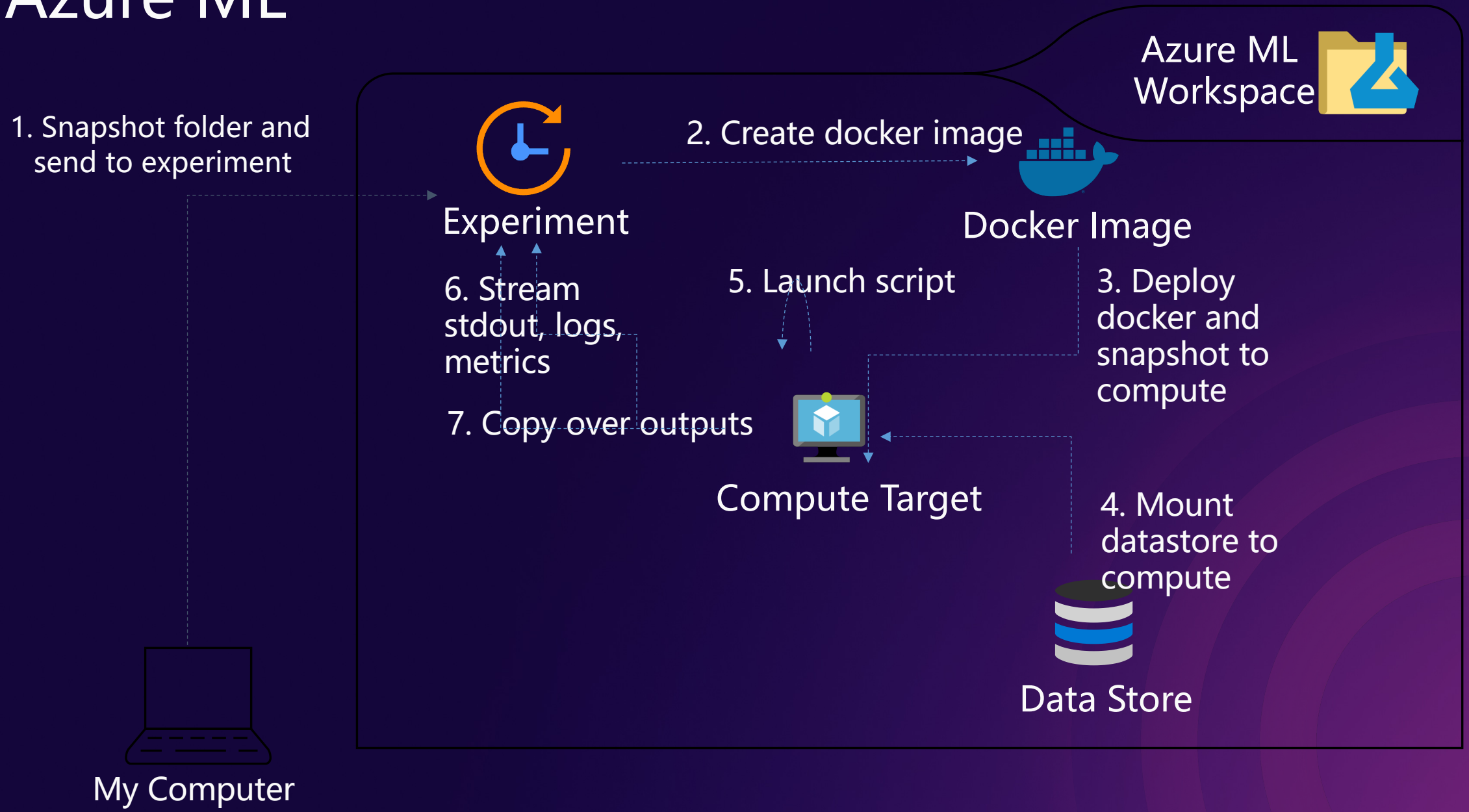


ML.NET Image Classification

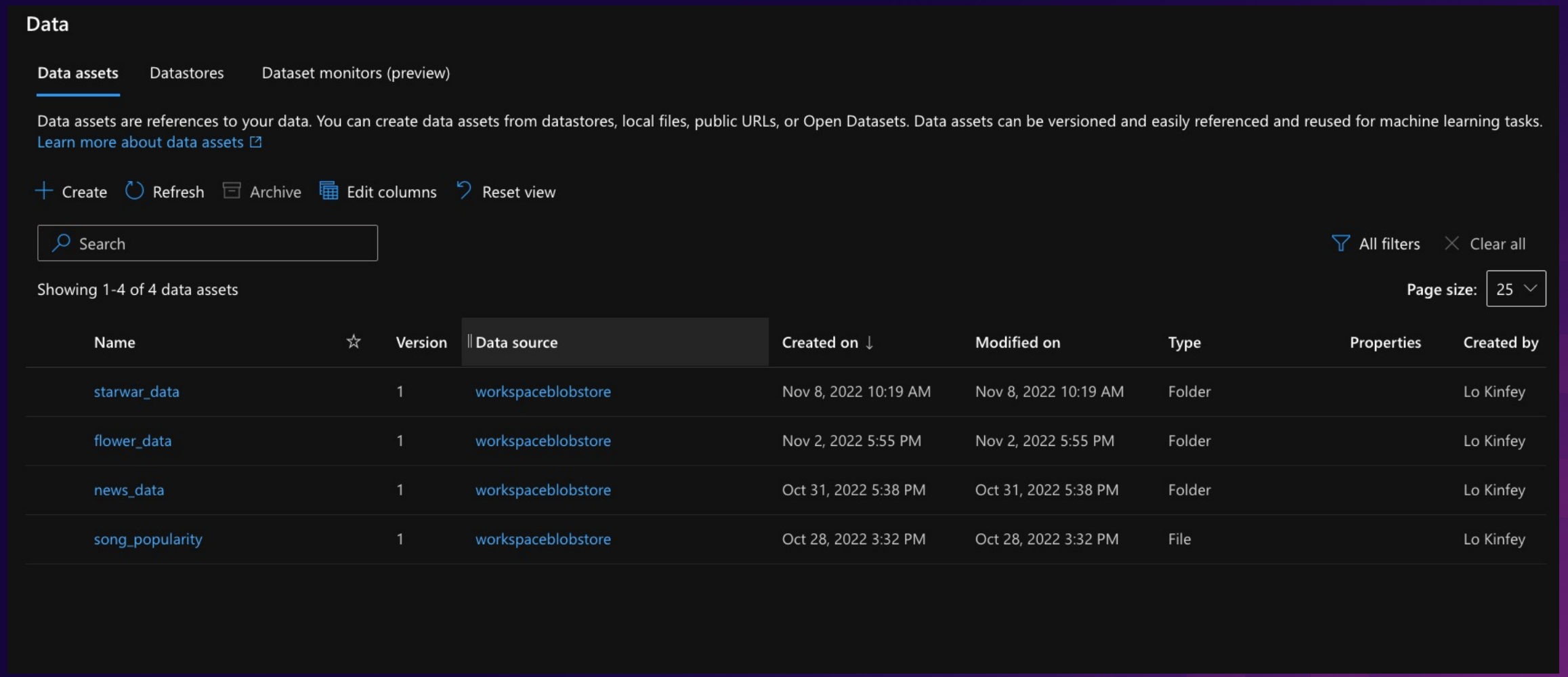


Maybe we need to more real Machine Learning scenario

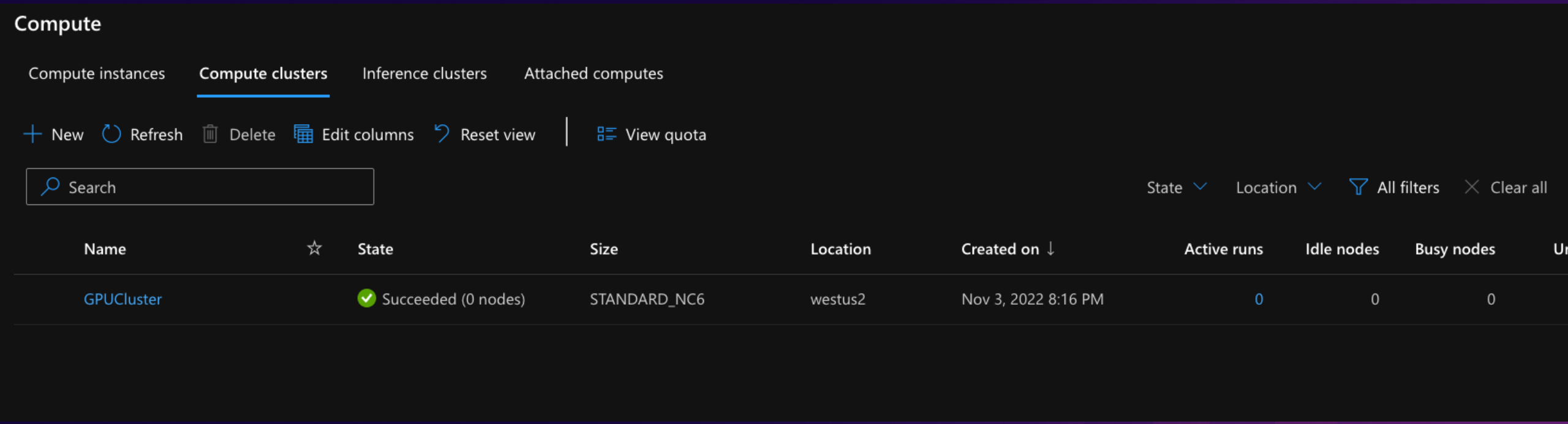
Azure ML


Azure ML Workspace
1. Snapshot folder and send to experiment
Experiment
2. Create docker image
Docker Image
6. Stream stdout, logs, metrics
5. Launch script
3. Deploy docker and snapshot to compute
7. Copy over outputs
Compute Target
4. Mount datastore to compute
Data Store
My Computer

Prepare : Upload your data to Azure ML Data



Prepare : Set Azure ML Compute



Prepare : Create a docker and yml

```
FROM mcr.microsoft.com/dotnet/sdk:6.0
RUN dotnet tool install --global mlnet-linux-x64
ENV PATH="$PATH:/root/.dotnet/tools"
```

Dockerfile

```
1   FROM mcr.microsoft.com/dotnet/sdk:6.0
2   RUN  apt install wget
```

Prepare : Create a docker and yml

```yaml
$schema: https://azuremlschemas.azureedge.net/latest/commandJob.schema.json
command: |
  FILENAME=libtensorflow-gpu-linux-x86_64-2.5.0.tar.gz
  wget -q --no-check-certificate https://storage.googleapis.com/tensorflow/libtensorflow/${FILENAME}
  tar -C /usr/local -xzf ${FILENAME}
  ldconfig /usr/local/lib
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11.3/targets/x86_64-linux/lib
  cd code
  dotnet restore
  dotnet build
  dotnet run --dataPath=${{inputs.data_dir}} --outputPath="outputs"
code: .
inputs:
  data_dir:
    type: uri_folder
    path: azureml:flower_data:1
experiment_name: tf-image-training
environment:
  build:
    path: .
    dockerfile_path: Dockerfile
compute: azureml:GPUCluster
```

Prepare : Create a docker and yml

```yaml
$schema: https://azuremlschemas.azureedge.net/latest/commandJob.schema.json
command: mlnet image-classification --dataset ${{inputs.data_dir}} --output outputs --name "StarWarML"
code: .
inputs:
  data_dir:
    type: uri_folder
    path: azureml:starwar_data:1
experiment_name: mlnet-imageclassification-training
environment:
  build:
    path: .
    dockerfile_path: Dockerfile
compute: azureml:GPUCluster
```

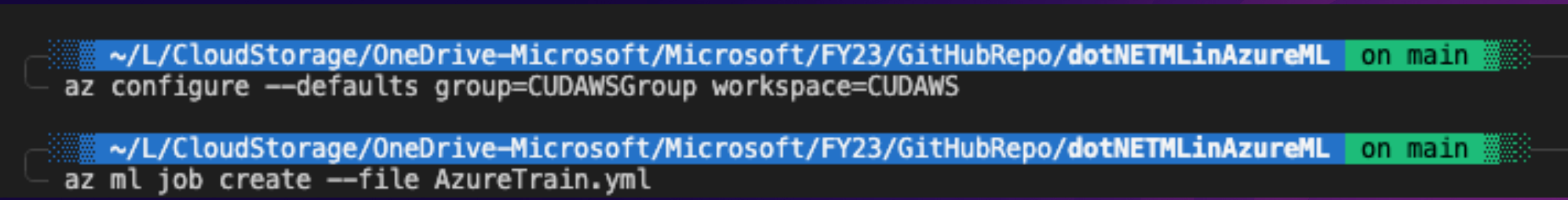
1 : Snapshot folder and send to experiment

The ml extension to the Azure CLI is the enhanced interface for Azure Machine Learning. It enables you to train and deploy models from the command line, with features that accelerate scaling data science up and out while tracking the model lifecycle.

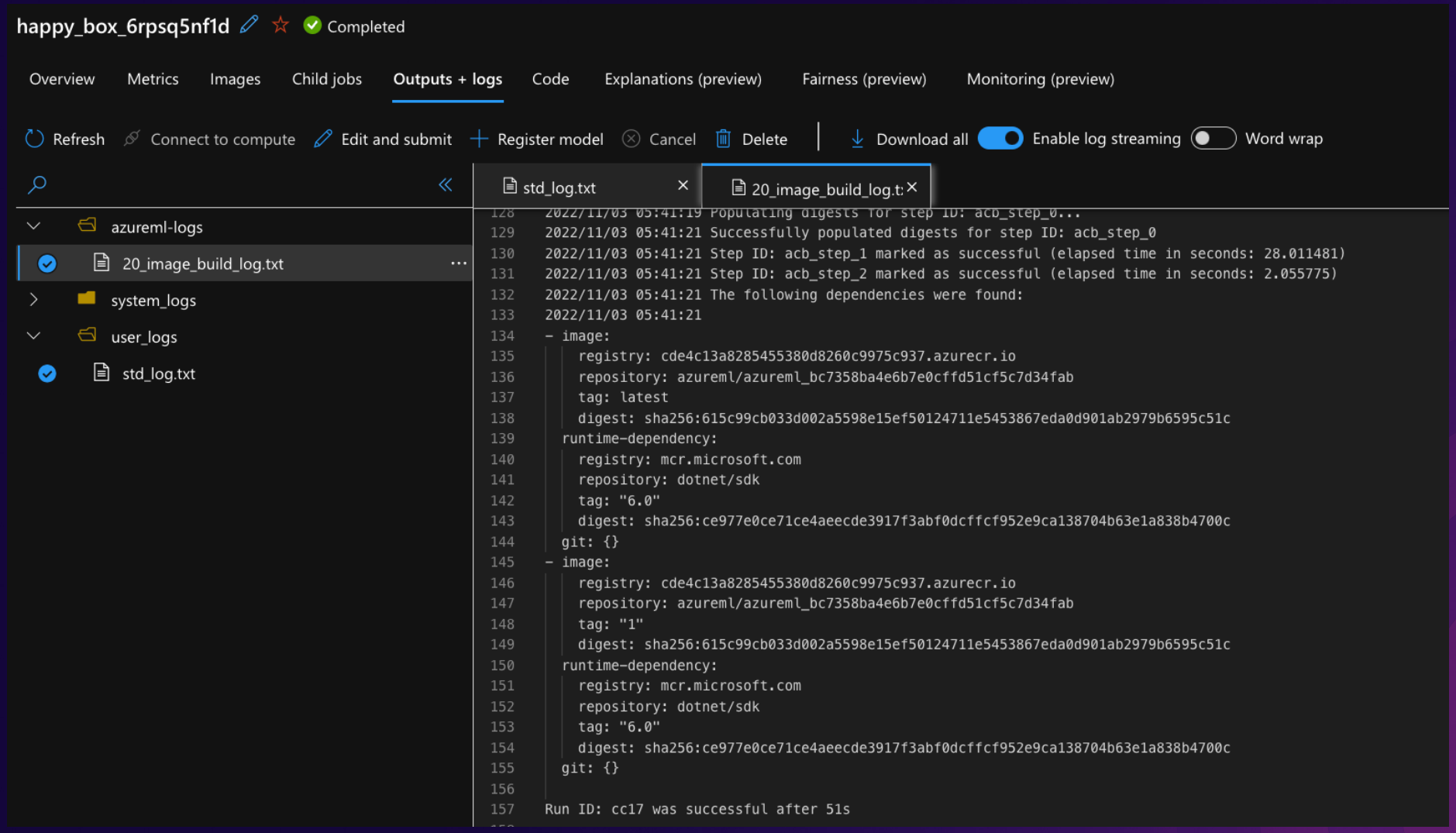
https://learn.microsoft.com/en-us/azure/machine-learning/how-to-configure-cli

az configure --defaults group=Your Resource Group workspace=Azure ML Workspace Name

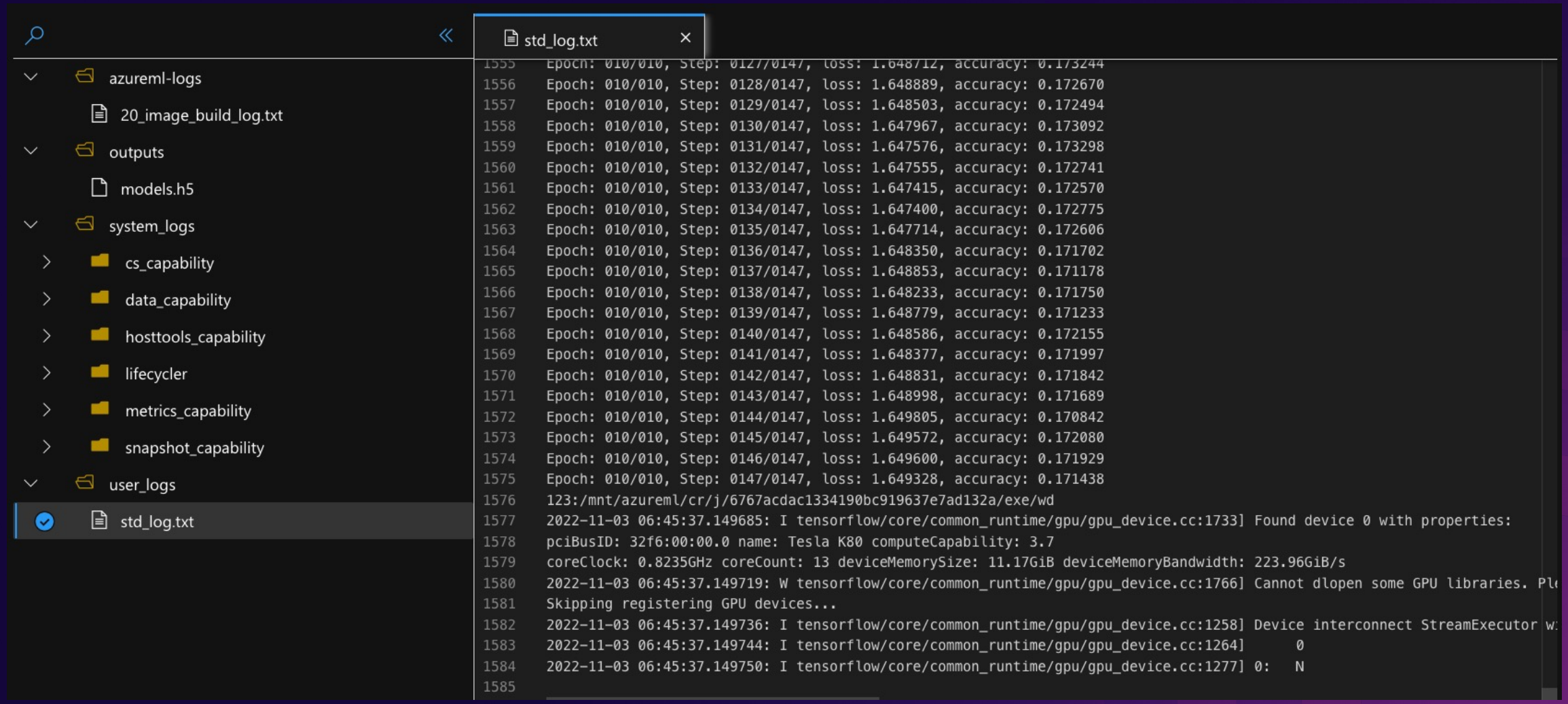
az ml job create --file Your.yml



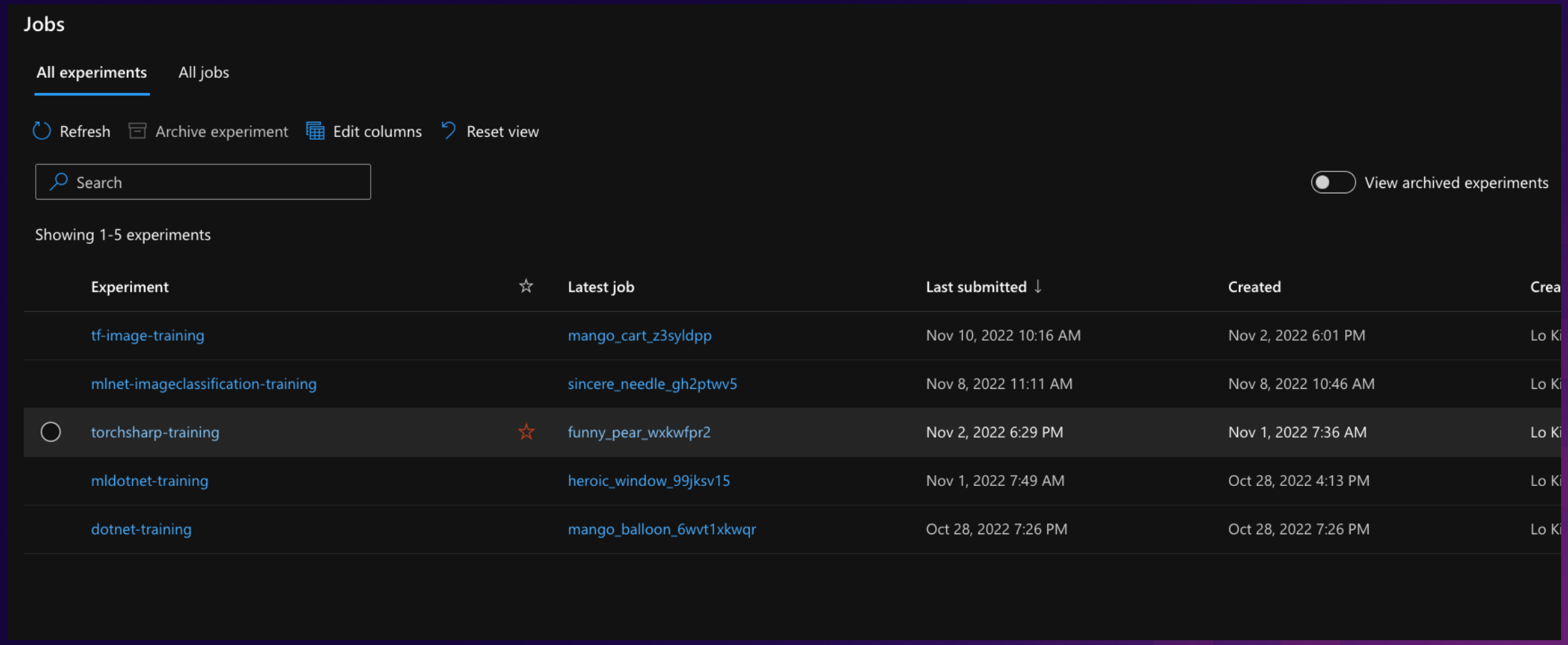
2 : Experiment



2 : Experiment

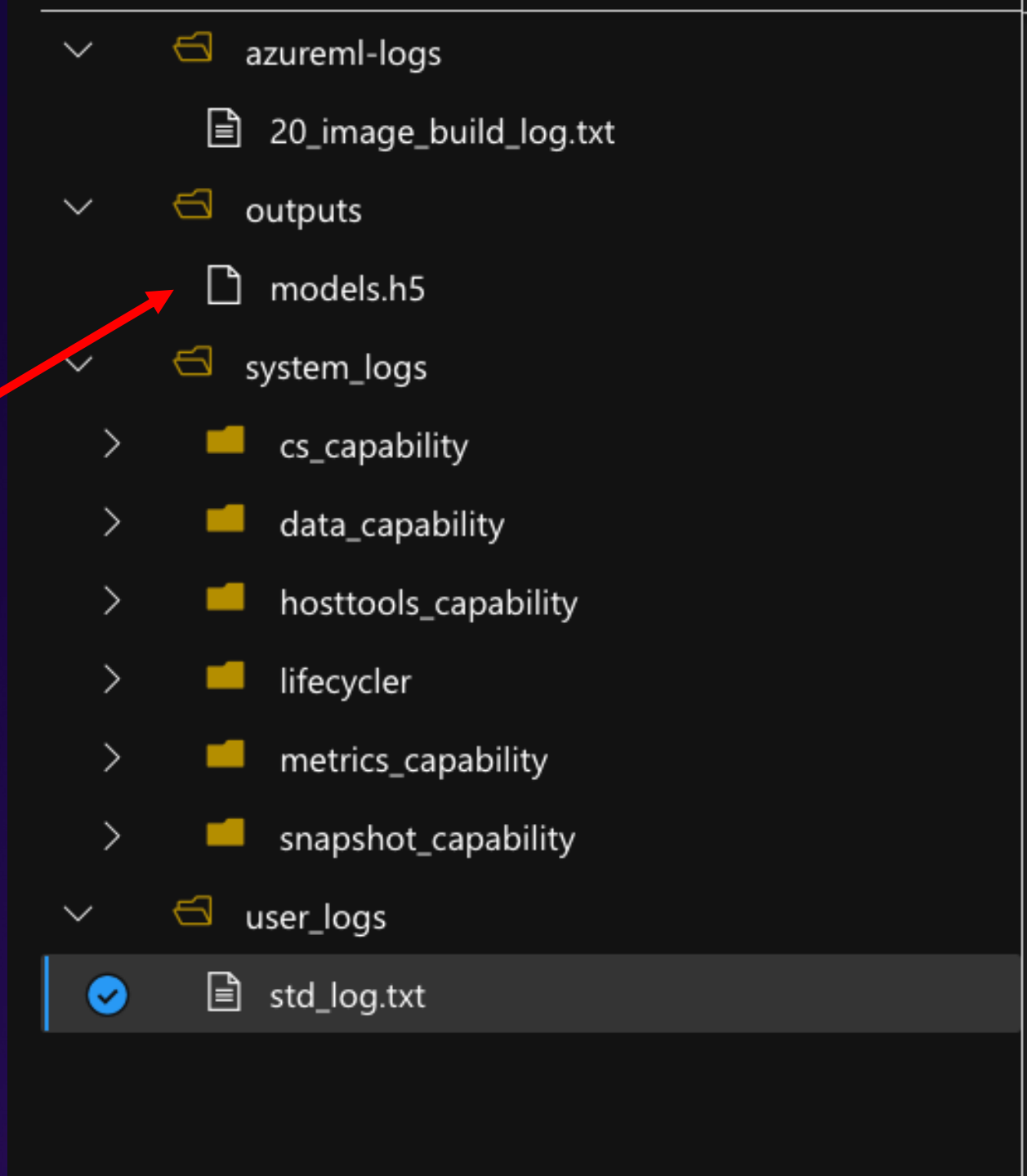


2 : Experiment

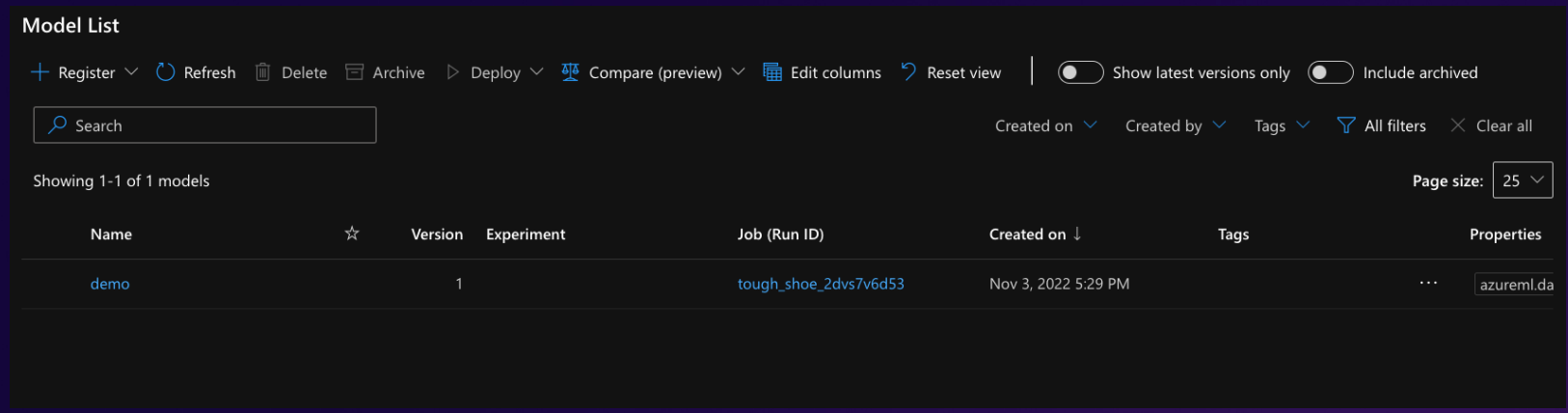


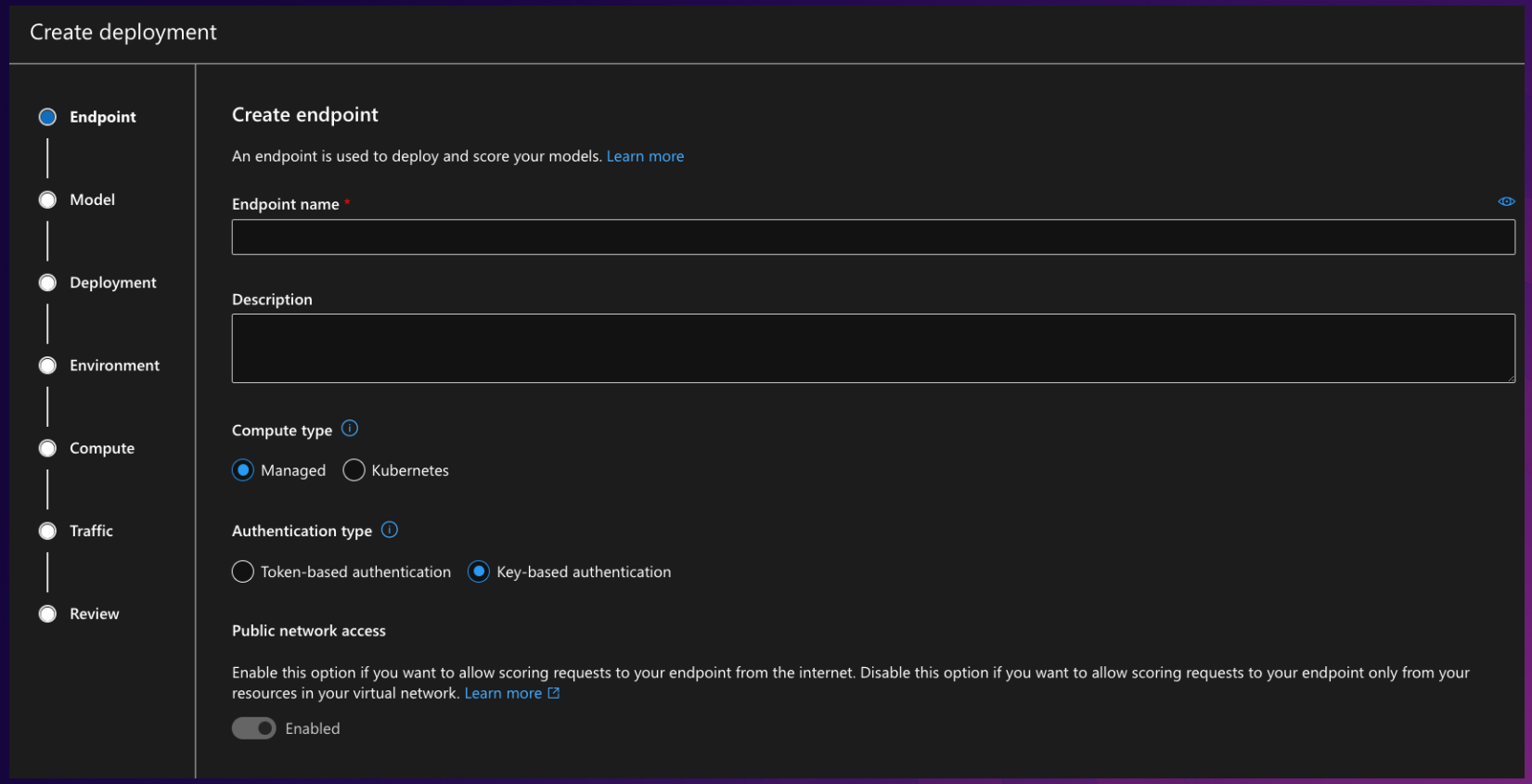
3 : Output Model

You can see the model

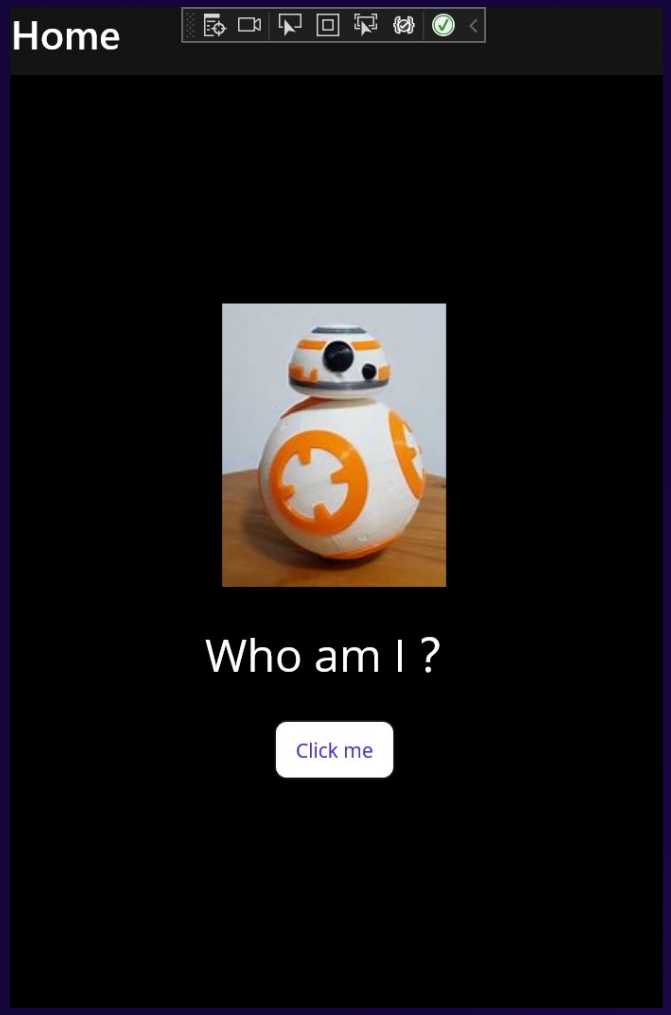


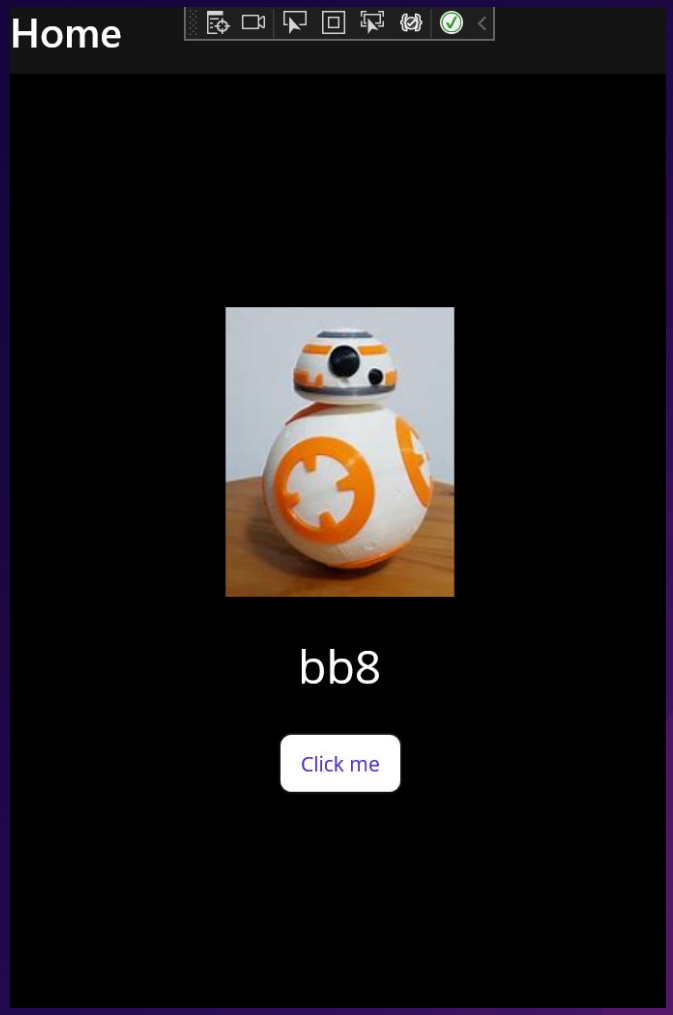
4 : Register Model





5 : Download Model and use it in your .NET applications





We can use ML.NET in any .NET solutions

Thank you

https://github.com/kinfey/dotNETMLinAzureML