Azure Machine Learning CLI: CHEAT SHEET

Microsoft

Azure

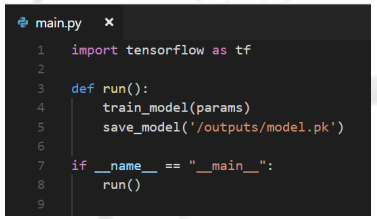
Intro

Azure Machine Learning services is an integrated, end-to-end data science and advanced analytics solution. It helps professional data scientists to prepare data, develop experiments, and deploy models at cloud scale.

To get the CLI, download and install Azure Machine Learning Workbench:
For Windows: https://aka.ms/azureml-wb-msi
For Mac: https://aka.ms/azureml-wb-dmg

To start the CLI, from Workbench, click File -> Open Command Prompt.

Train in compute target



Model management

Deploy with Kubernetes

Tutorials

Step by step tutorials: https://docs.microsoft.com/en-us/azure/machine-learning/preview/#Step-by-Step-Tutorials

Set up compute target

You can compute your ML model in local mode, in a Data Science VM (DSVM) or in a HDInsight cluster.
Attach a DSVM target:

```
$ az ml computetarget attach remotedocker -n <target name> -a <ip address or FQDN> -u <username> -w <password>
```

Attach HDInsight target:

```
$ az ml computetarget attach cluster -n <target name> -a <cluster name, e.g. myhdicluster-ssh.azurehdinsight.net> -u <ssh username> -w <ssh password>
```

Inside folder aml_config, you can change the conda dependencies. Also, you can operate with PySpark, Python or Python in a GPU DSVM. To define the operation mode for Python:
in <target name>.runconfig add `Framework:Python`
For PySpark:
in <target name>.runconfig add `Framework:PySpark`
For Python in a GPU DSVM:
in <target name>.runconfig add `Framework:Python`
in <target name>.compute add `baseDockerImage: microsoft/mmlspark:plus-gpu-0.9.9` and `nvidiaDocker:true`

To prepare the compute target:

```
$ az ml experiment prepare -c <target name>
```

HINT: you can add `--help` to any command to get a help message

```
$ az ml <xyz> --help
```

Submit job

To submit a job to a remote target:

```
$ az ml experiment submit -c <target name> myscript.py
```

Work with Jupyter notebooks

To start a notebook:

```
$ az ml notebook start
```

This will start a notebook in localhost. You can work in local, selecting the kernel Python 3, or in your remote VM, selecting the kernel <target name>.

Operate with run history

To list the run history:

```
$ az ml history list -o table
```

List all completed runs:

```
$ az ml history list --query "[?status=='Completed']" -o table
```

Find runs with the best accuracy

```
$ az ml history list --query "@[?Accuracy != null] | max_by(@, &Accuracy).Accuracy”
```

We can also download files generated in each run. A model saved in folder outputs can be promoted:

```
$ az ml history promote -r <run id> -ap outputs/model.pkl -n <model name>
```

Then downloaded by:

```
$ az ml asset download -l assets/model.pkl.link -d <model folder path>
```

Set up o16n environment

To set the operationalization environment you need to create:
– A resource group
– A storage account
– An Azure Container Registry (ACR)
– An Application insights account
– A Kubernetes deployment on an Azure Container Service (ACS) cluster

Local deployment for testing (requires to have Docker in your local machine)

```
$az ml env setup -l <region, e.g. eastus2> -n <env name> -g <resource group>
```

Cluster deployment: sets up an ACS cluster with Kubernetes

```
$az ml env setup -l <region, e.g. eastus2> -n <env name> -g <resource group> --cluster
```

See the status while creating

```
$ az ml env show -n <environment name> -g <resource group>
```

Set the environment to be used

```
$ az ml env set -n <environment name> -g <resource group>
```

HINT: you can add `--debug` to any command to get verbose messages

```
$ az ml <xyz> --debug
```

Model management

With this tool, you can efficiently deploy and manage ML models

```
$ az ml account modelmanagement create -l <region, e.g. eastus2> -n <name> -g <resource group> --sku-name <S1|S2|S3|DevTest>
$ az ml account modelmanagement set -n <name> -g <resource group>
```

To register one of the trained models obtained during any of the experiments.

```
$ az ml model register -m <path to model file> -n <name>
```

Deploy web service

Create the manifest

```
$ az ml manifest create -n <manifest name> -f <path to code file> -r <runtime: spark-py|python> -i <model id> -s <path to schema>
```

Create an image

```
$ az ml image create -n <name> --manifest-id <id> -c aml_config/conda_dependencies.yml
```

Create the web service:

```
$ az ml service create realtime --image-id <image id> -n <service name>
```

Get endpoint details and test the web service:

```
$ az ml service usage realtime -i <service id>
$ az ml service run realtime -i <service id> -d "input data"
```