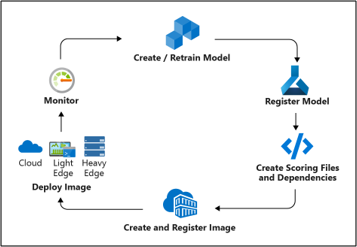
# Machine learning with Azure Databricks and Azure ML services

# Introduction lab

We will cover the Azure Machine Learning SDK lab to be found at:

<https://github.com/delange/databricks_aml_sdk>

A general overview how model creation and model deployment can be achieved within Azure can be found in below diagram.

 1:  Create / train model (with Azure Databricks)​

**1**

2: Register Image using the Image Registry ​  
(the Azure Container Registry)​

**4**

3: Deploy the Image to cloud or to edge devices​

**2**

**3**

4: Monitor models—you can monitor input, output, and other relevant data from your model.

This lab walks through 1-3 and is divided into 4 steps:

Step 1: Installation and configuration Azure ML services & Azure Databricks

Step 2: Ingest data to Azure Databricks

Step 3: Build model with Azure ML services SDK in Azure Databricks

Step 4: Deploy model with Azure ML services to Azure Container Instances

Required:

* Azure Databricks workspace + cluster (standard 6.5) – cluster takes around 5 min. to start

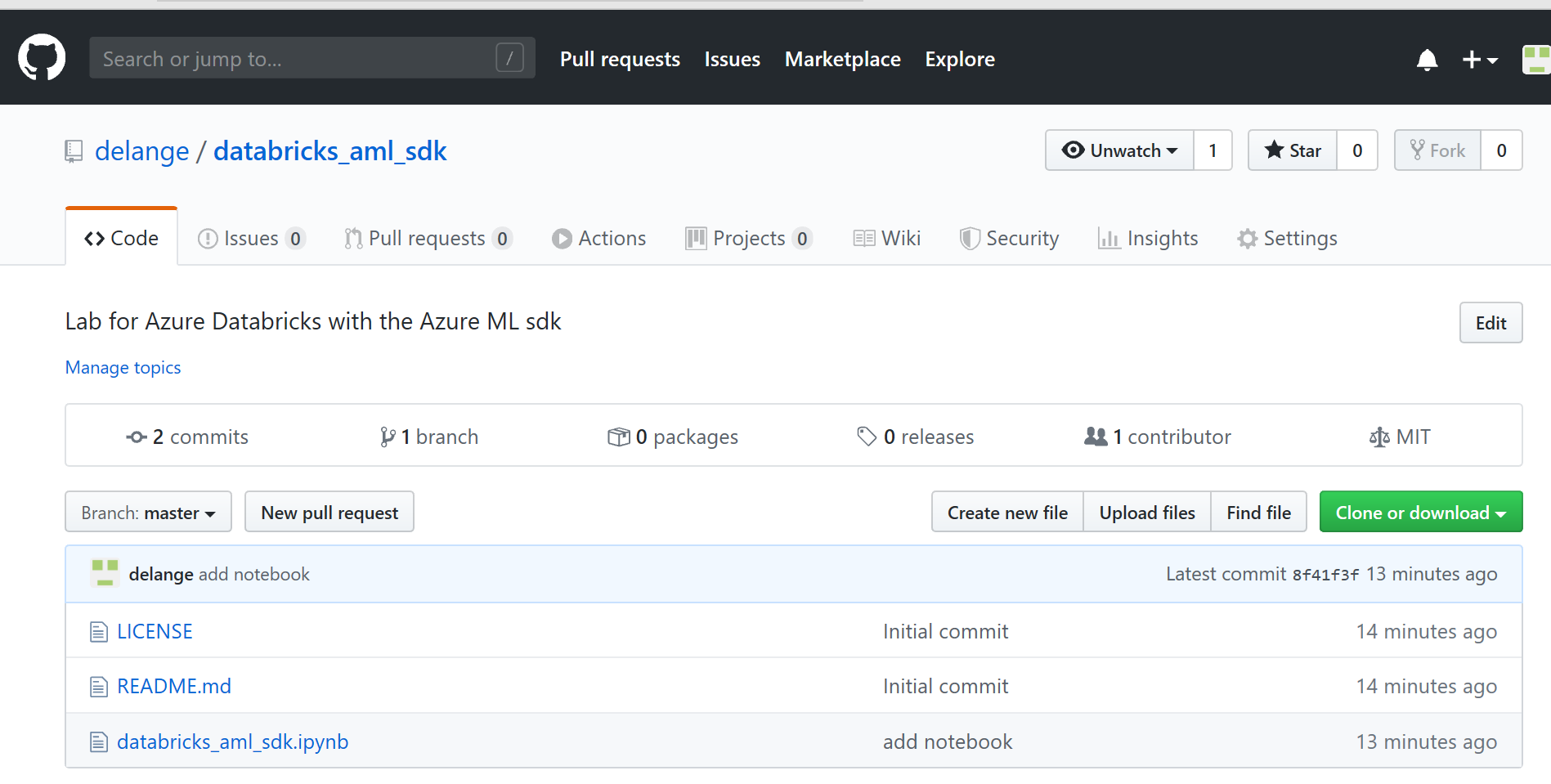
The model that we will create during this lab concerns a simple scoring model of American income for individuals given specific characteristics per person. You can explore the data set during the lab.

# Step 1. Installation and Configuration

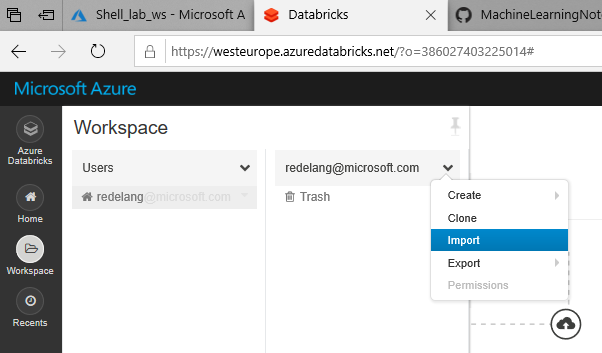
## 1.1 Import Notebook

The aim of this step is to import the first Jupyter Notebook from GitHub into Azure Databricks:

* Open from the [Azure Machine Learning SDK lab](https://github.com/delange/databricks_aml_sdk) databricks\_aml\_sdk.ipynb, and copy the URL from the browser (which should be [this](https://github.com/delange/databricks_aml_sdk/blob/master/databricks_aml_sdk.ipynb)).



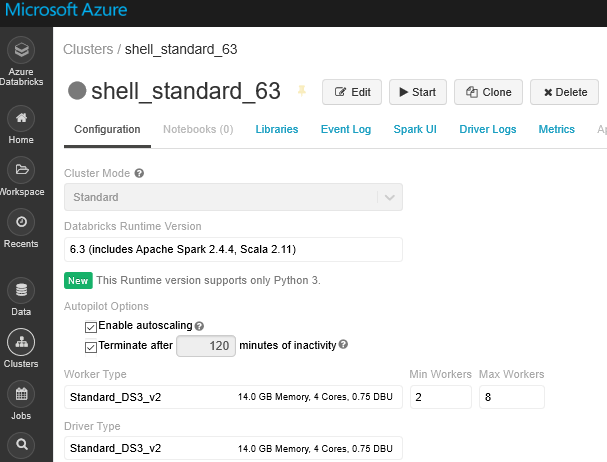
* In Azure Databricks, go to your Workspace and import the Jupyter Notebook by using the above copied URL.

****

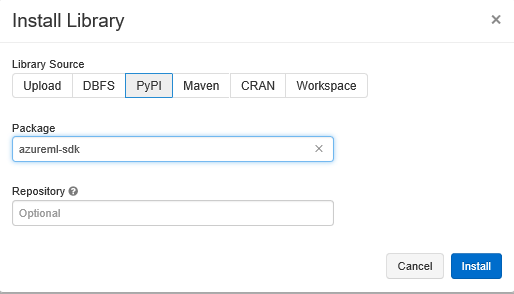
## 1.2 Install library

On the second cell of the notebook, one can read that we need to install the Azure ML SDK library. A GUI is used to install Python/R/Java/Scala libraries in Azure Databricks.

* Go to the Clusters overview and select your cluster, for example:

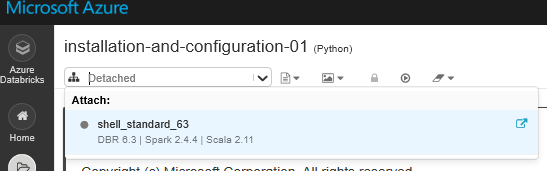
****

* Go to Libraries and select Install New (note that when this option is greyed-out, the cluster is not running).
* If the cluster is not running, start the cluster now.
* During the start-up process of the cluster, or when the cluster is running, one can install new libraries: select Install New.
* From PyPi install the azureml-sdk- package [note that the library name differs from the notebook, use the one named here (azureml-sdk)]:



## 1.3 Attach notebook to cluster

* Notebooks can be attached to up and running clusters, attach your notebook to your cluster:



* When attached, check if Azure ML SDK has been installed correctly, run cell/cmd3 (to run a cell, one can use the ‘play-button’ at the right corner of the cell, or use the [control]+[enter] combination, or [control]+[shift] combination.



## 1.4 Connect to AML workspace – collect data

We will use the AML SDK to connect the Azure Machine Learning Workspace to the Databricks workspace. With the provided code, a new AML workspace can be created, but it also checks whether it already exists, then it only makes the connection (that is what we are doing here).

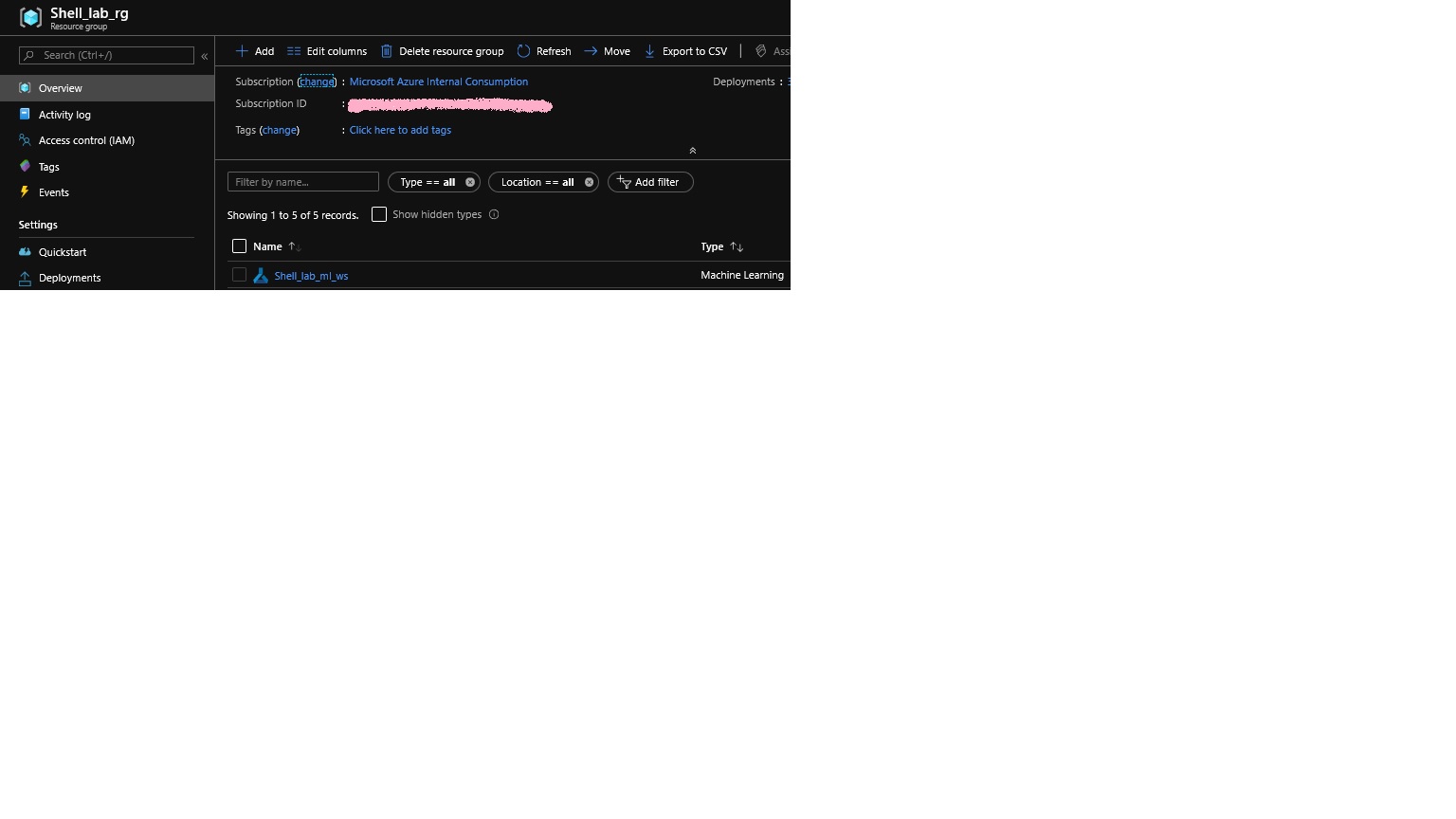
For setting up the connection, we need some resource references.

* Go to the Azure Portal, search for Resource Groups, and select the group created for this lab, select Properties (at the left side):



Collect the following from this screen:

* Azure Subscription ID (here greyed out)
* Azure Resource Group
* Azure resource group workspace region (make sure you make use of the one-word version of the region name, see Appendix 1 for mapping Region Display Names against Name)
* Azure Machine Learning Workspace name: collect this name from the Overview tab in the Resource Group:



* Fill the collected information in cell 5

## 1.5 Connect to AML workspace – build connection

* When running cell 7, you might run in an authentication request, like below. Follow the instructions and sign-in at the provided web page:



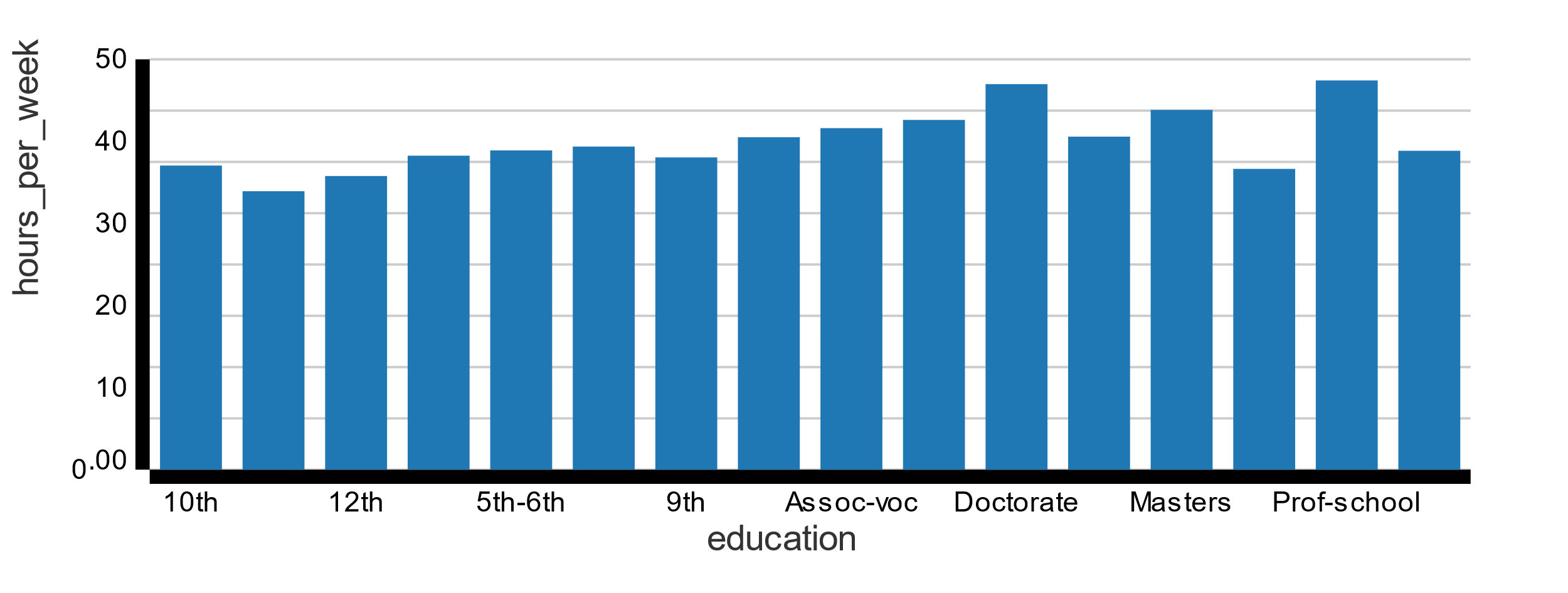
* Run the cells one-by-one (until section 02 Ingest Data), and verify whether the AML workspace has been configured correctly against the input data.

# Step 2. Ingest data

## 2.1 Import Notebook

## 2.2 Data ingestion

* In cell 14 and 15, you will download the datafile “AdultCensusIncome.csv” to the Databricks Filesystem[[1]](#footnote-2) and made available to the Spark cluster. Familiarize yourself with the data set on a schema level (cell 16) and the content itself (cell 17), and play around with the display settings to get better insights. Note that the a limit parameter is given, providing only a partial set of the data population, one can remove this setting to get full overview of the data (random example below).



## 2.3 Data preparation

* In cell 19, you will set a column that should be used as label – this is the metric that the model will attempt to predict.
* In cell 19, the data set is randomly split into two data sets, a train and test data set.

## 2.4 Data persistence

* In cell 21 DataFrames from the test and train datasets are stored on the Databricks filesystem in parquet format.

# Step 3. Build model and run history

## 3.1 Import Notebook

## 3.2 Model Building / load data sets

## 3.3 Define model

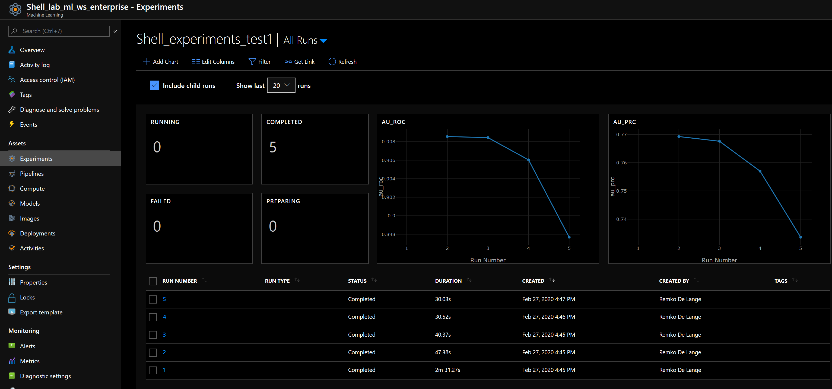
* In cell 28 a couple of model training runs will be executed, together they form an Azure Machine Learning experiment. You might want to give the myexperiment a more useful name (it’s now named “Ignite\_AI\_Talk”).

In this example [Logistic Regression](https://spark.apache.org/docs/2.4.4/ml-classification-regression.html#logistic-regression) models are created with the use of [Spark ML](https://spark.apache.org/docs/2.4.4/ml-guide.html) . Several runs are executed with different regularization values. Some log metrics are collected and send to Azure Machine Learning services for each individually, like the area under the receiver operating characteristics (AUROC) and area under the precision recall curves (AUPRC).

* Cell 29-31 collects the best scoring run and retrieve the model via the Azure ML SDK (azureml.core.run) – here, AUROC is used to define ‘best’.

## 3.4 Model Evaluation

* In cell 34, the best scoring model will be evaluated against the test data. Change the displayed number of results lines, and visualize for example prediction vs income for the two income groups.
* Cell 35 provide values to evaluate the model – are you able to improve the model further?
* Another way to obtain overview of all you experiment runs is in the Azure portal, you can make changes to the charts:

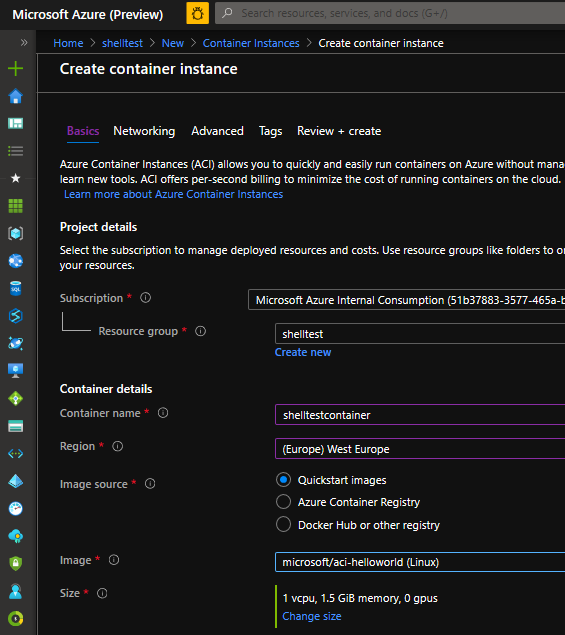


# Step 4. Deploy to Azure Container Instances

## 4.1 Import Notebook and load workspace

## 4.2 Register Azure Container Instance (ACI)

* Cell 41: - create a Container Instance via the Azure portal, in the same resource group. You can make of the Quickstart images options at Image source:



4.3 Register Model

* At cell 43, the model will be registered at the Azure Machine Learning Service; you can change the description with your own text. Note that the model gets automatically versioned.
* The registration can be verified via the Azure Portal; pay attention the model version.

4.4 Create score file

* At cell 44, the scoring file get created.

You might want to edit the score file, i.e. appName.

4.5. Deploy to ACI

* Run cell 45 to deploy to an ACI webservice, this can take up to 5 minutes.

4.6 Test the web service

* At cell 48, some data are obtained from the test data set.
* At cell 49, the web service will be used with the data. Note that we are scoring the data on income:

0.0 means <50K

1.0 means >50K

# Appendix 1 – Region names

|  |  |  |  |
| --- | --- | --- | --- |
| DisplayName | Latitude | Longitude | Name |
| -------------------- | ---------- | ----------- | ------------------ |
| East Asia | 22.267 | 114.188 | eastasia |
| Southeast Asia | 1.283 | 103.833 | southeastasia |
| Central US | 415.908 | -936.208 | centralus |
| East US | 373.719 | -798.164 | eastus |
| East US 2 | 366.681 | -783.889 | eastus2 |
| West US | 37.783 | -122.417 | westus |
| North Central US | 418.819 | -876.278 | northcentralus |
| South Central US | 294.167 | -98.5 | southcentralus |
| North Europe | 533.478 | -62.597 | northeurope |
| West Europe | 523.667 | 4.9 | westeurope |
| Japan West | 346.939 | 1.355.022 | japanwest |
| Japan East | 35.68 | 139.77 | japaneast |
| Brazil South | -23.55 | -46.633 | brazilsouth |
| Australia East | -33.86 | 1.512.094 | australiaeast |
| Australia Southeast | -378.136 | 1.449.631 | australiasoutheast |
| South India | 129.822 | 801.636 | southindia |
| Central India | 185.822 | 739.197 | centralindia |
| West India | 19.088 | 72.868 | westindia |
| Canada Central | 43.653 | -79.383 | canadacentral |
| Canada East | 46.817 | -71.217 | canadaeast |
| UK South | 50.941 | -0.799 | uksouth |
| UK West | 53.427 | -3.084 | ukwest |
| West Central US | 40.890 | -110.234 | westcentralus |
| West US 2 | 47.233 | -119.852 | westus2 |
| Korea Central | 375.665 | 1.269.780 | koreacentral |
| Korea South | 351.796 | 1.290.756 | koreasouth |
| France Central | 463.772 | 23.730 | francecentral |
| France South | 438.345 | 21.972 | francesouth |
| Australia Central | -353.075 | 1.491.244 | australiacentral |
| Australia Central 2 | -353.075 | 1.491.244 | australiacentral2 |
| UAE Central | 24.466.667 | 54.366.669 | uaecentral |
| UAE North | 25.266.666 | 55.316.666 | uaenorth |
| South Africa North | -25.731.340 | 28.218.370 | southafricanorth |
| South Africa West | -34.075.691 | 18.843.266 | southafricawest |
| Switzerland North | 47.451.542 | 8.564.572 | switzerlandnorth |
| Switzerland West | 46.204.391 | 6.143.158 | switzerlandwest |
| Germany North | 53.073.635 | 8.806.422 | germanynorth |
| Germany West Central | 50.110.924 | 8.682.127 | germanywestcentral |
| Norway West | 58.969.975 | 5.733.107 | norwaywest |
| Norway East | 59.913.868 | 10.752.245 | norwayeast |

Created with:

PS Azure:\> az account list-locations -o table

1. [Databricks File System](https://docs.microsoft.com/en-us/azure/databricks/data/databricks-file-system) (DBFS) is a distributed file system mounted into an Azure Databricks workspace and available on Azure Databricks clusters. DBFS is an abstraction on top of scalable object storage and offers the following benefits:

   * Allows you to mount storage objects so that you can seamlessly access data without requiring credentials.
   * Allows you to interact with object storage using directory and file semantics instead of storage URLs.
   * Persists files to object storage, so you won’t lose data after you terminate a cluster.

   [↑](#footnote-ref-2)