Microsoft Cloud Workshop

Cognitive Services and deep learning

Hands-on lab step-by-step

March 2018

Contents

Cognitive Services and deep learning hands-on lab step-by-step 1

Abstract and learning objectives 1

Overview 1

Solution architecture 1

Requirements 2

Before the hands-on lab 3

Task 1: Provision the Windows Data Science Virtual Machine 3

Task 2: Verify remote desktop access to Data Science VM 8

Task 3: Initialize Azure Machine Learning Workbench 10

Task 4: Stop the Data Science VM 12

Exercise 1: Setup Azure Machine Learning accounts 13

Task 1: Provision Azure Machine Learning Experimentation service 13

Task 2: Create the Azure Machine Learning project 15

Task 3: Install dependencies 18

Exercise 2: Deploy the Summarizer as a Service 22

Task 1: Deploy your ACS cluster 22

Task 2: Set Visual Studio Code as the project IDE in Workbench 25

Task 3: Create the Summarization service 26

Task 4: Deploy the Summarization service 28

Exercise 3: Applying TensorFlow 30

Task 1: Prepare TensorFlow 30

Task 2: Train and deploy the TensorFlow model 30

Exercise 4: Completing the solution 36

Task 1: Deploy the Computer Vision API 36

Task 2: Deploy the Text Analytics API 38

Task 3: Completing the solution 41

After the hands-on lab 43

Task 1: Clean up lab resources 43

Cognitive Services and deep learning hands-on lab step-by-step

Abstract and learning objectives

In this workshop, you will learn to combine both pre-built artificial intelligence (AI) (in the form of various Cognitive Services) with custom AI (in the form of services built and deployed with Azure Machine Learning services). You will learn to create intelligent solutions atop unstructured text data by designing and implementing a text analytics pipeline. You will also learn how to build a binary classifier using a simple neural network that can be used to classify the textual data. Also, you will learn how to deploy multiple kinds of predictive services using Azure Machine Learning and learn to integrate with the Computer Vision API and the Text Analytics API from Cognitive Services.

Along the way, you will get to consider the following technologies and services:

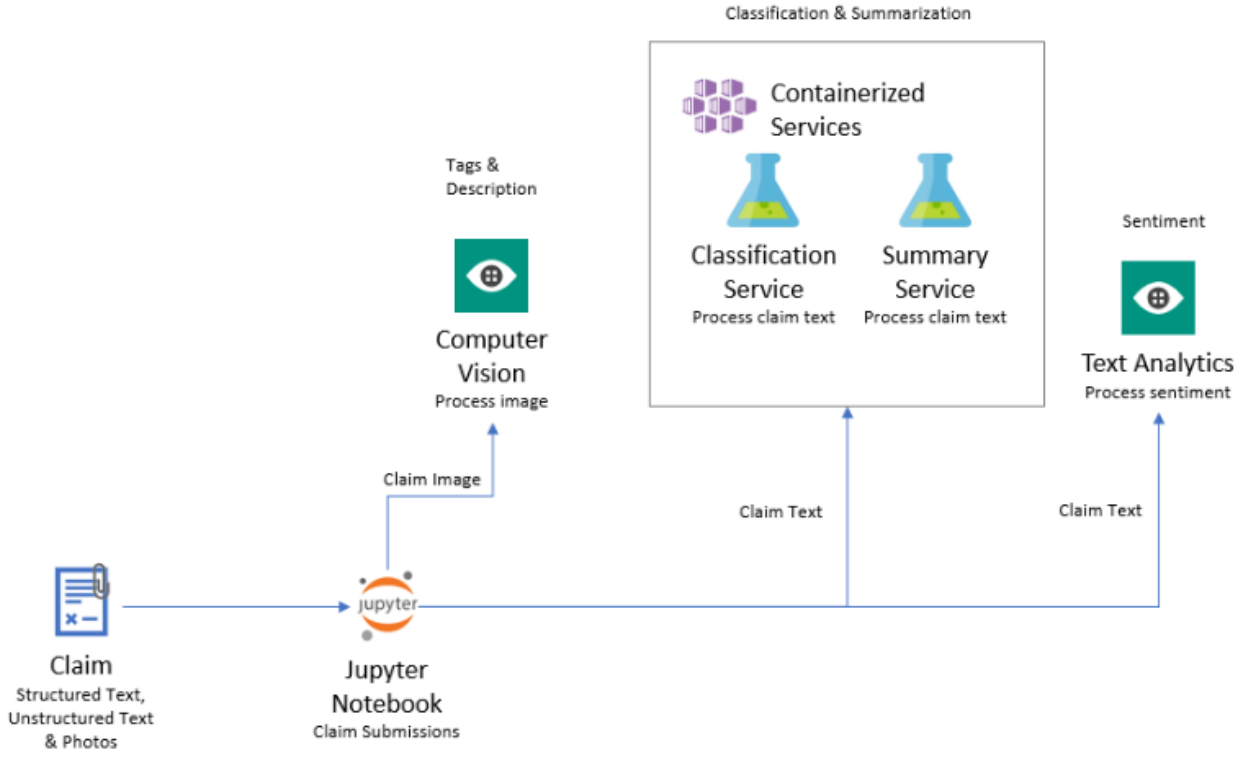
- Azure Machine Learning services
- Cognitive Services
- Computer Vision API
- Text Analytics API
- TensorFlow

Overview

In this workshop, you will help Contoso Ltd. Build a proof of concept that shows how they could build a solution that amplifies the claims processing capabilities of their agents.

Solution architecture

The high-level architecture of the solution is illustrated in the diagram. The lab is performed within the context of a Jupyter Notebook running within a Data Science VM on Azure. Various notebooks are built to test the integration with the Cognitive Services listed, to train customer ML services and to integrate the results in a simple user interface that shows the result of processing the claim with all of the AI services involved.



Requirements

1. Microsoft Azure subscription must be pay-as-you-go or MSDN
    a. Trial subscriptions will not work

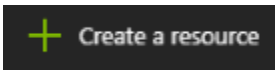
Before the hands-on lab

Duration: 30 minutes

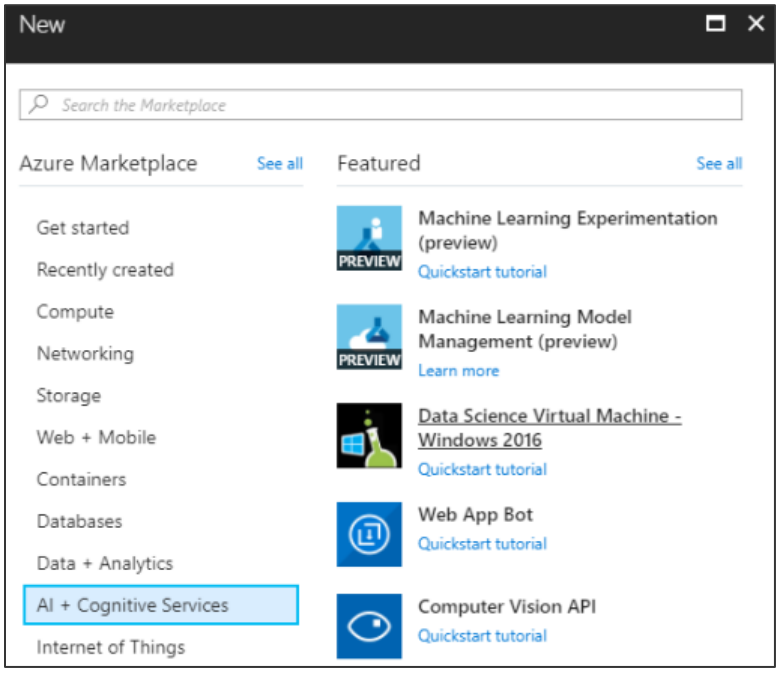
To maximize your lab time, the following steps which setup your environment should be performed before attending the lab.

Task 1: Provision the Windows Data Science Virtual Machine

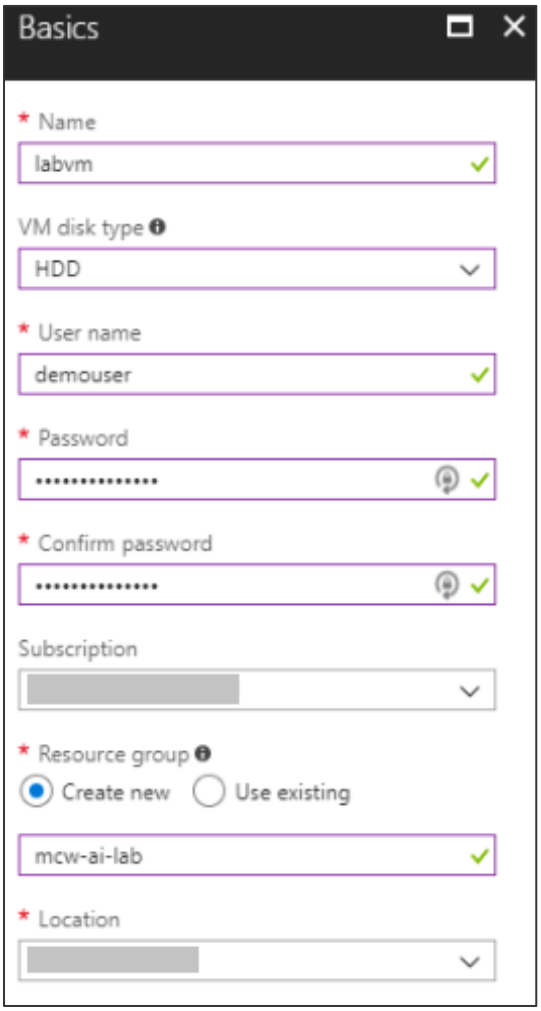
1. Navigate to the Azure Portal at https://portal.azure.com.
2. Select **Create a resource**.



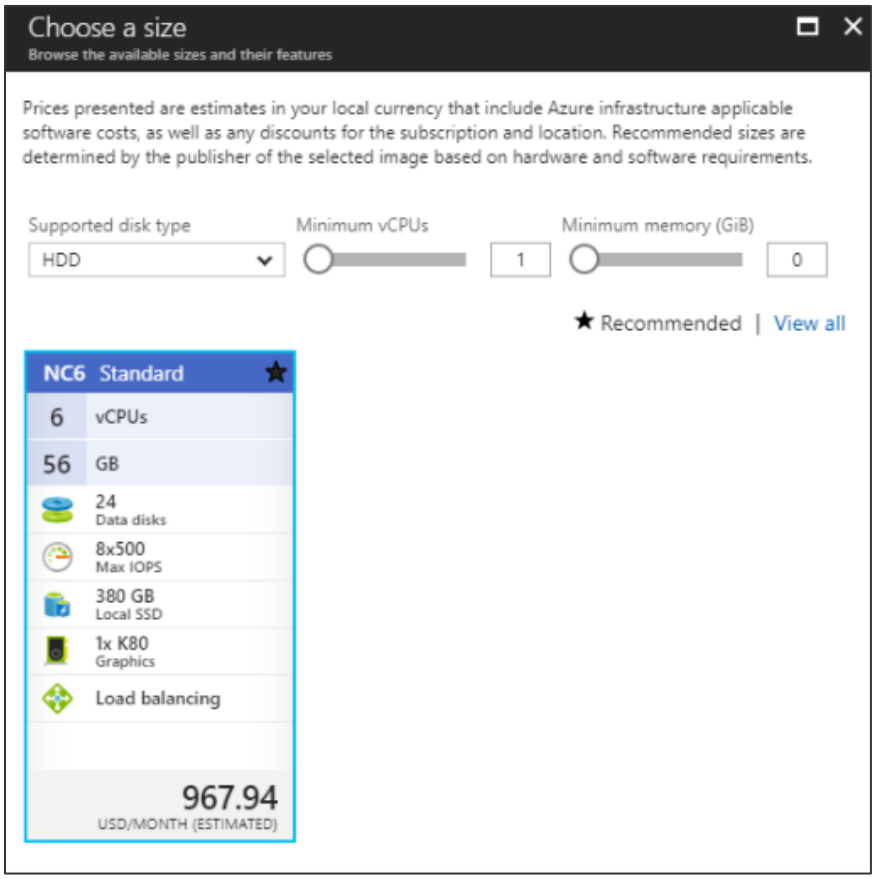
3. Select **AI + Cognitive Services** and then select **Data Science Virtual Machine Windows 2016**.



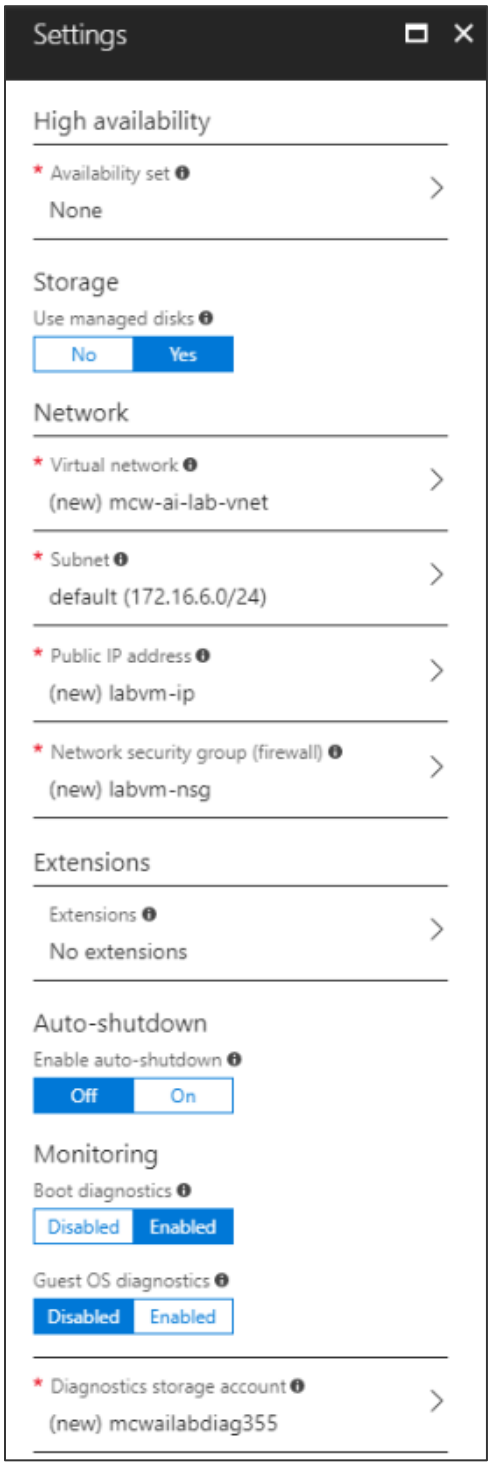
4. On the **Basics** blade provide the following inputs:
    a. **Name**: enter labvm
    b. **VM disk type**: select HDD. This will enable you to use a GPU based machine if you choose to in the subsequent step.
    c. **User name**: enter demouser
    d. **Password and Confirm Password:** enter Abc!1234567890
    e. **Subscription**: select your Azure subscription
    f. **Resource group**: select Create new and provide the name mcw-ai-lab
    g. **Location**: select either South Central US or East US (or any of the regions in which the NC-series VM's are currently available, see the regions service page for an up to date listing).



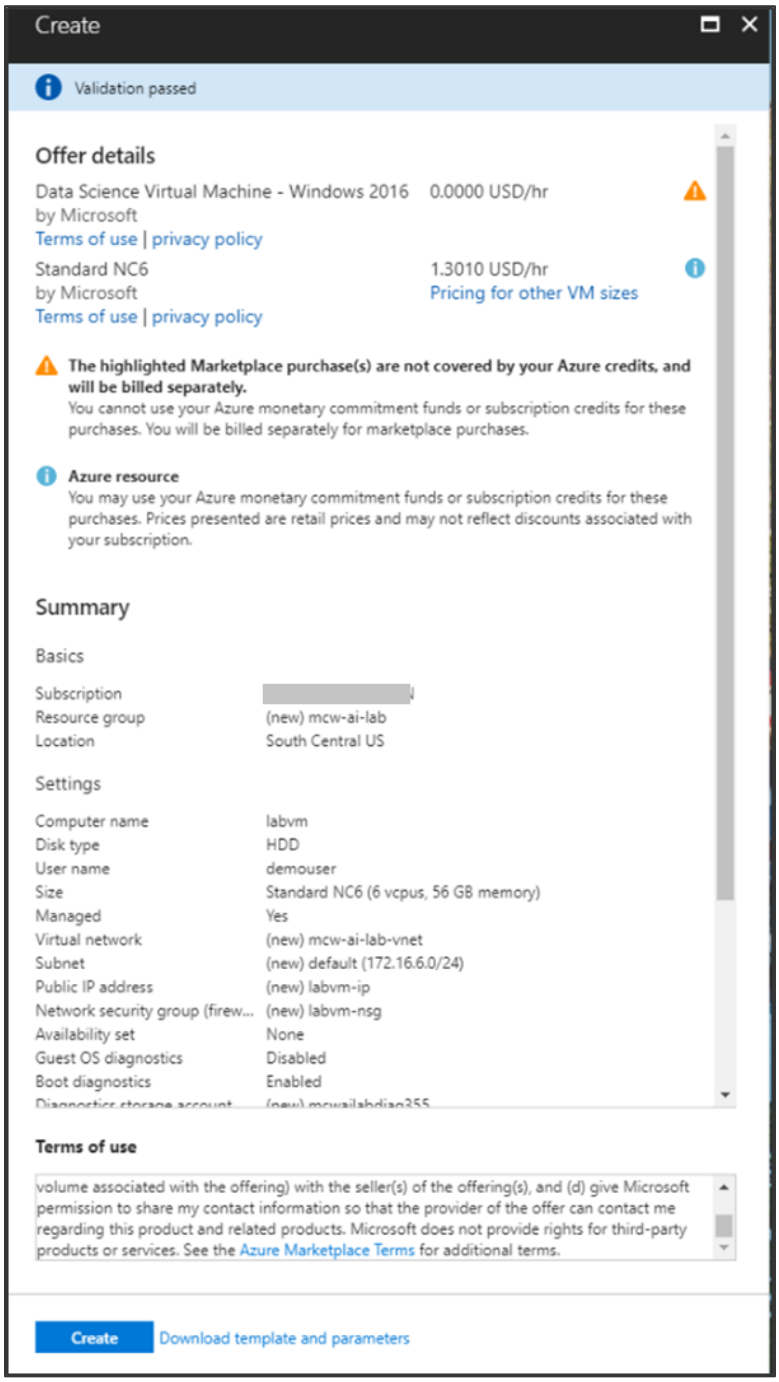
5. Select **OK**.
6. On the **Choose a size** blade, select **NC6 Standard** and choose **Select**.



7. Leave all values on the **Settings** blade at their defaults and select **OK**.



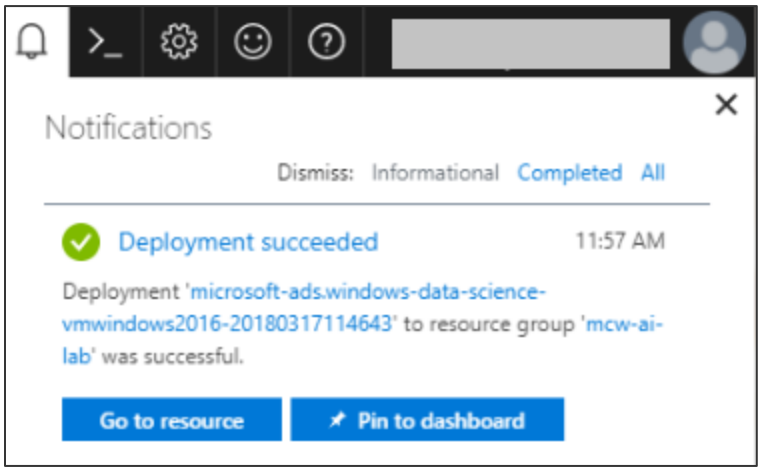
8. On the **Create** blade, review the summary and then select **Create.**



9. The VM should take 10-15 minutes to provision.

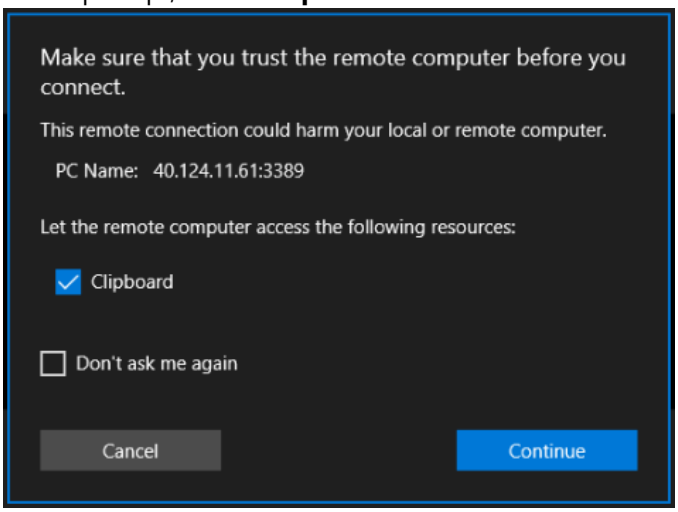
Task 2: Verify remote desktop access to Data Science VM

1. When the VM is ready, you should see a notification. Select **Go to resources** to view the deployed Data Science VM in the Portal.

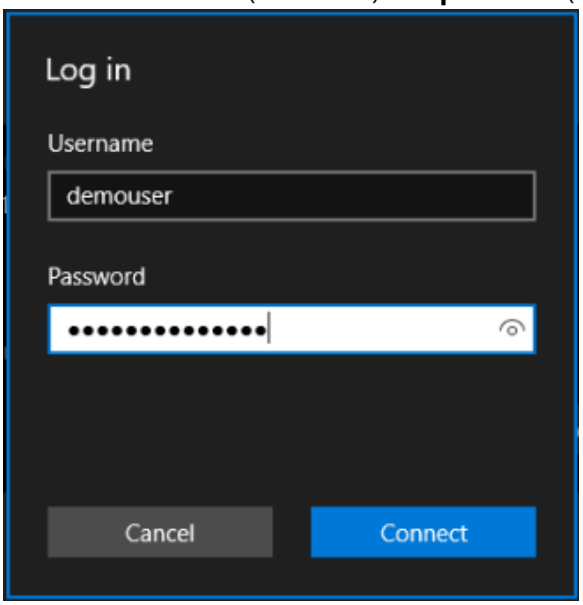


2. On the blade for the VM, select **Connect**. This will download a Remote Desktop (RDP) file.

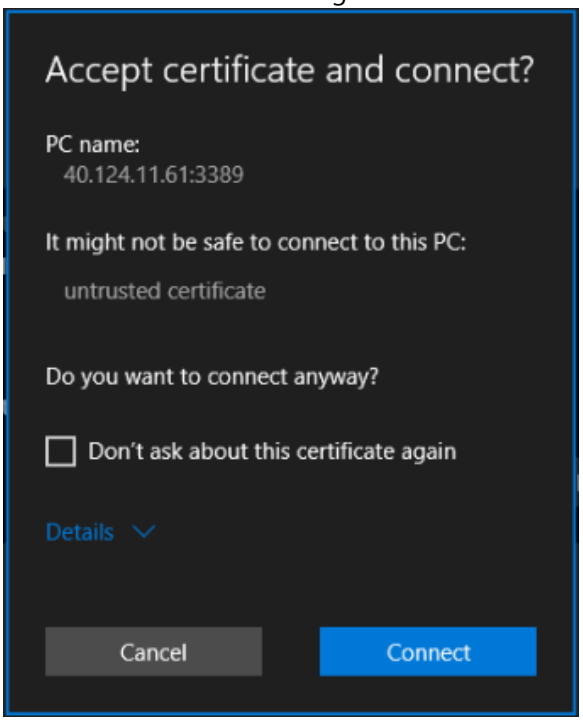
3. Open the downloaded RDP file.
4. At the prompt, ensure **Clipboard** is checked and select **Continue**.



5. Enter the **username** (demouser) and **password** (Abc!1234567890) and select **Connect** to login.



6. Select **Connect** on the dialog that follows.

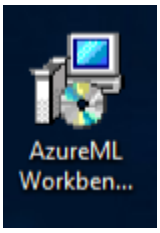


7. Within a few moments, you should see the desktop for your new Data Science Virtual Machine.

Task 3: Initialize Azure Machine Learning Workbench

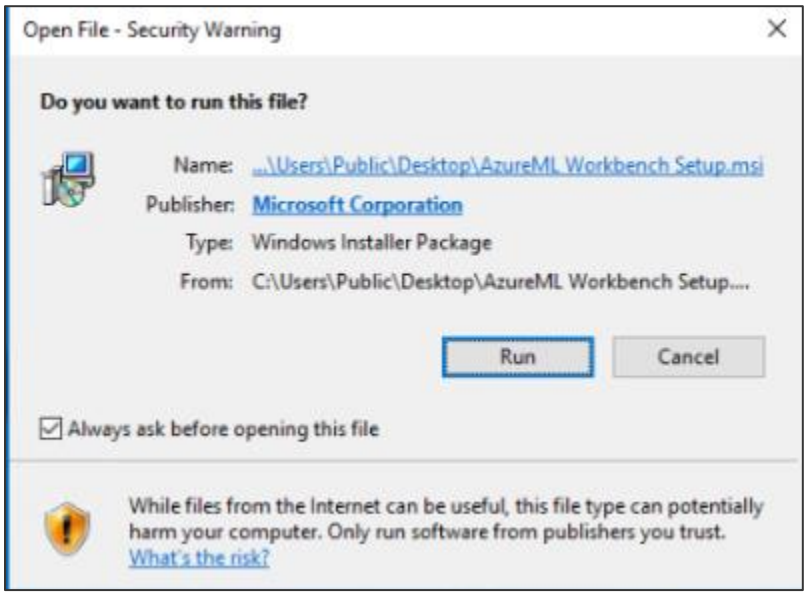
Before using the Azure Machine Learning Workbench on the Data Science VM, you will need to take the one-time action of double-clicking on the AzureML Workbench Setup icon on the desktop to install your instance of the workbench.

1. Within the RDP session to the Data Science VM, on the desktop locate the AzureML Workbench Setup.

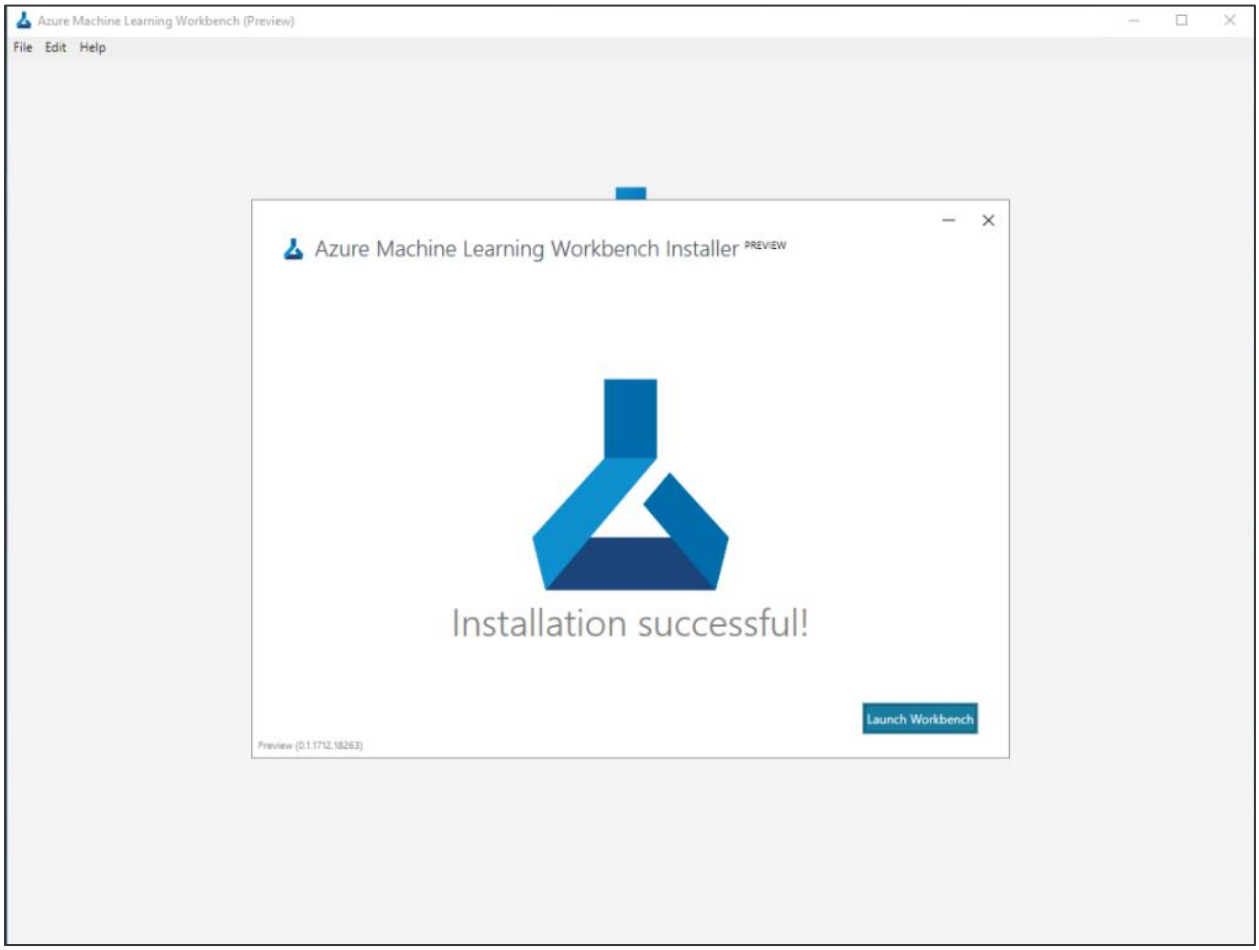


2. Double-click the icon to install the Workbench.

3. At the **Open File – Security Warning** dialog, select **Run**.



4. Step through all the prompts leaving all values at their defaults to complete the Workbench installation. The installation will take about 25 minutes. Use the **X** to close the install when it is finished.



Task 4: Stop the Data Science VM

If you are performing this setup the night before the hands-on lab, you can optionally Stop the VM to save on costs overnight and resume it when you are ready to start on the lab. Follow these steps to Stop the VM:

1. Return to the Azure Portal.
2. Navigate to the blade of your labvm.
3. Select the **Stop** button.

NOTE: When you are ready to resume the VM, simply follow the previous steps and instead of selecting **Stop**, select **Start**. Your VM will take about 5 minutes to start up, after which you can use the **Connect** button in the VM blade to RDP into the VM as before.

You should follow all steps provided *before* attending the Hands-on lab.

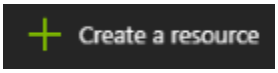
Exercise 1: Setup Azure Machine Learning accounts

Duration: 45 minutes

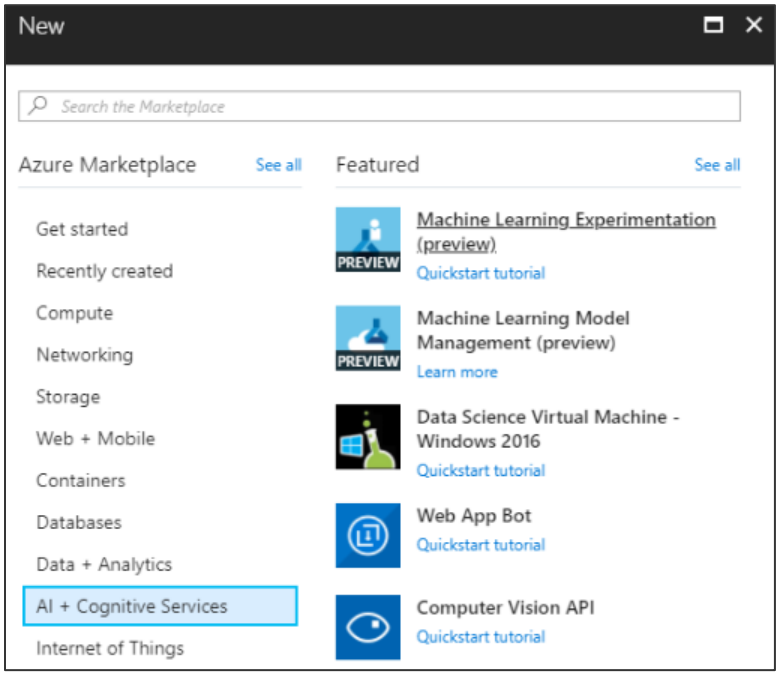
In this exercise, you will setup your Azure Machine Learning Experimentation and Model Management Accounts and get your project environment setup.

Task 1: Provision Azure Machine Learning Experimentation service

1. Navigate to the Azure Portal.
2. Select **Create a resource**.

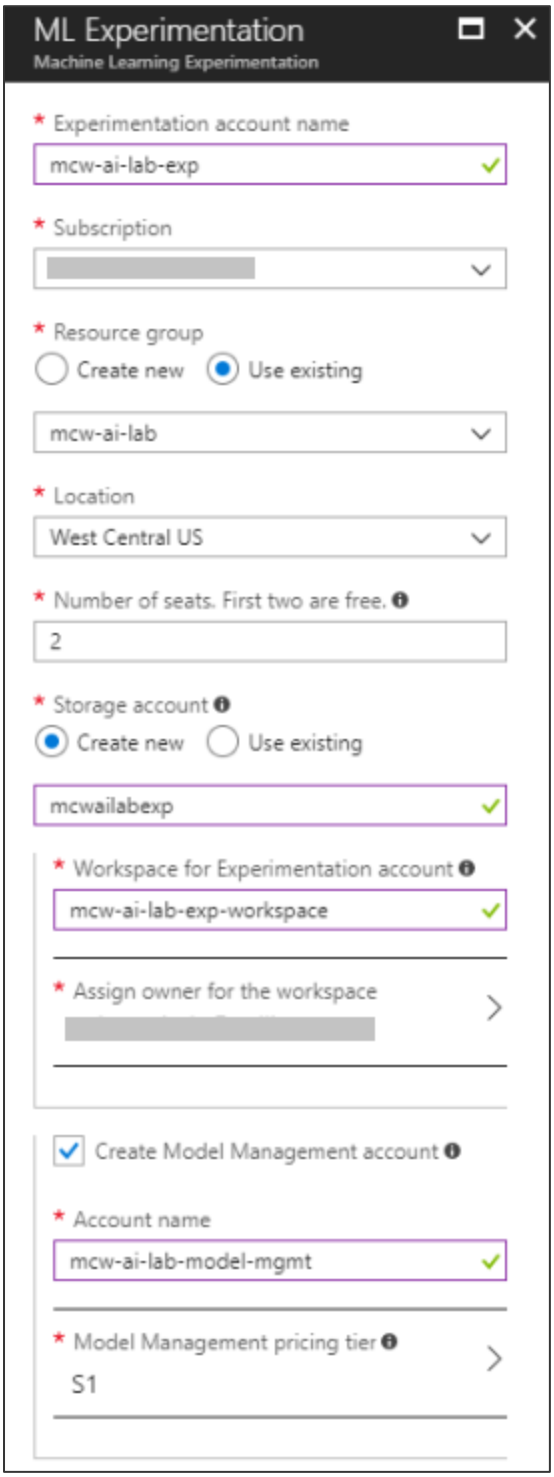


3. Select **AI + Cognitive Services** and then select **Machine Learning Experimentation**



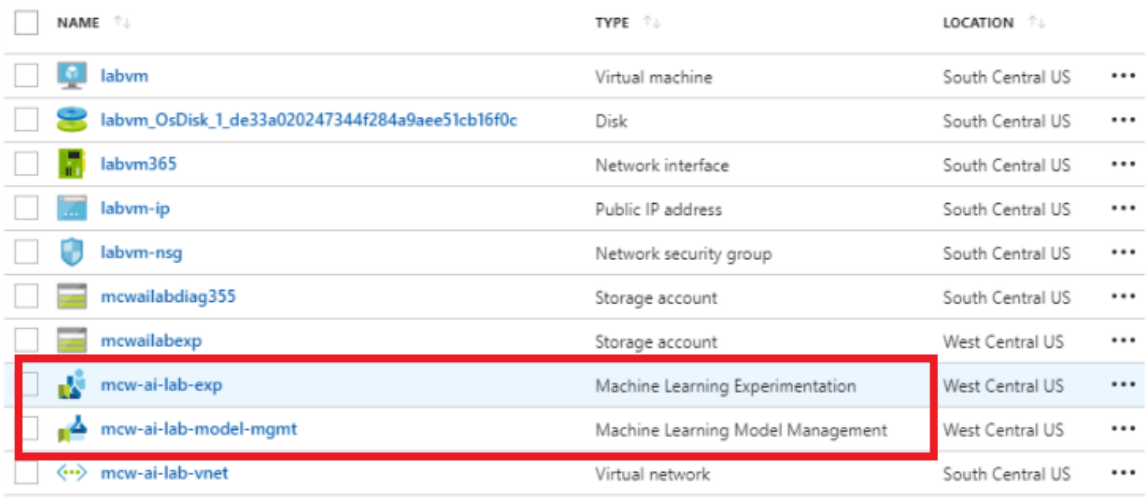
4. On the **ML Experimentation** blade, provide the following:
    a. **Experimentation account name**: provide a name for your experimentation account.
    b. **Subscription:** select your Azure subscription.
    c. **Resource group**: select the mcw-ai-lab resource group you previously created.
    d. **Location**: select the region nearest to where you deployed your Data Science VM. It's OK if they are not in exactly the same region, but try to select a region that is close to minimize latency.
    e. **Number of seats**: leave at 2.
    f. **Storage account**: select create new and provide a unique name for the new storage account.
    g. **Workspace for Experimentation account**: provide a unique name for the workspace.
    h. **Assign owner for the workspace**: leave the owner assigned to you.
    i. **Create Model Management account**: leave checked.
    j. **Account name**: provide a name for your model management account.

k. **Model Management pricing tier**: select the S1 pricing tier.



5. Select **Create** to provision the Experimentation and Model Management Service. The deployment should take about 2 minutes.

6. When the deployment completes, navigate to your mcw-ai-lab resource group and confirm that you see an instance of Machine Learning Experimentation and Machine Learning Model Management.

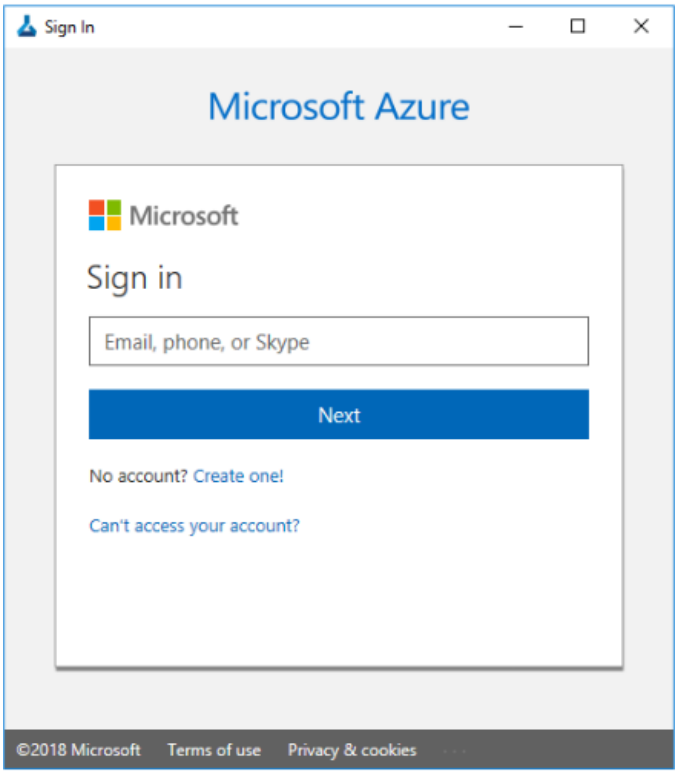


NAME	TYPE	LOCATION
labvm	Virtual machine	South Central US
labvm_OsDisk_1_de33a020247344f284a9aee51cb16f0c	Disk	South Central US
labvm365	Network interface	South Central US
labvm-ip	Public IP address	South Central US
labvm-nsg	Network security group	South Central US
mcwailabdiag355	Storage account	South Central US
mcwailabexp	Storage account	West Central US
mcw-ai-lab-exp	Machine Learning Experimentation	West Central US
mcw-ai-lab-model-mgmt	Machine Learning Model Management	West Central US
mcw-ai-lab-vnet	Virtual network	South Central US

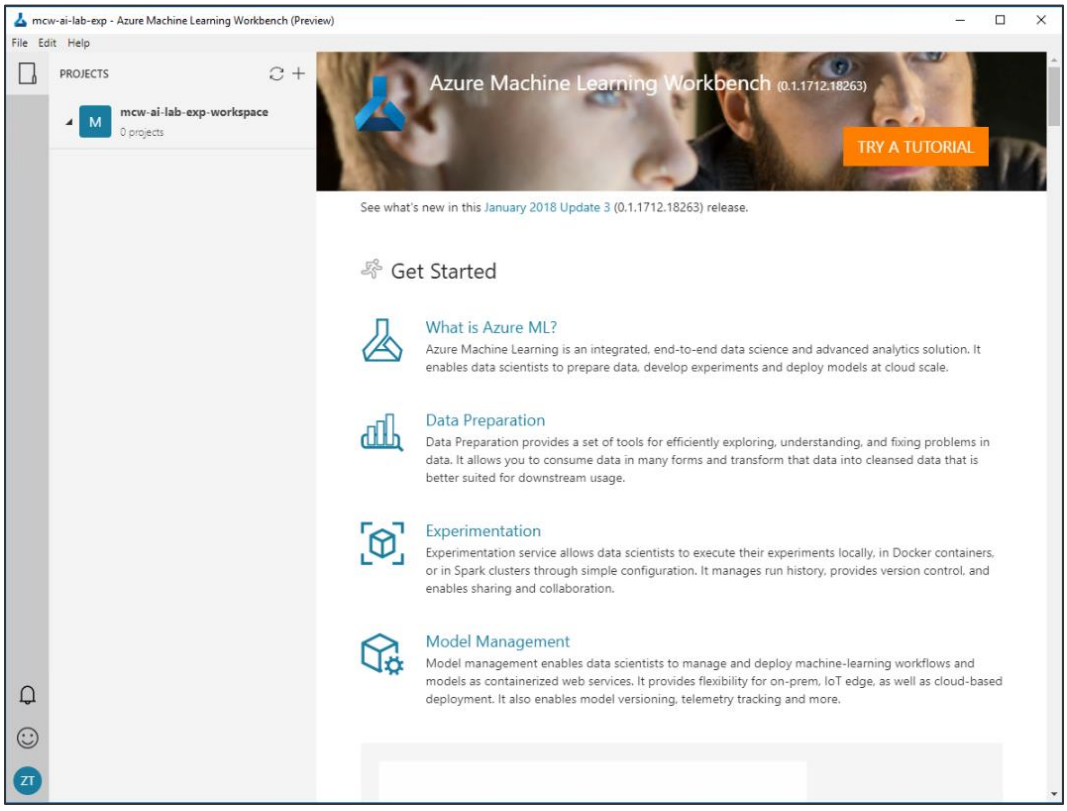
Task 2: Create the Azure Machine Learning project

1. Connect to the labvm via RDP. If you stopped the VM, remember to Start it up again before attempting to connect.
2. From the **Start** menu, launch **Azure Machine Learning Workbench**.
3. Select **Sign in with Microsoft.**

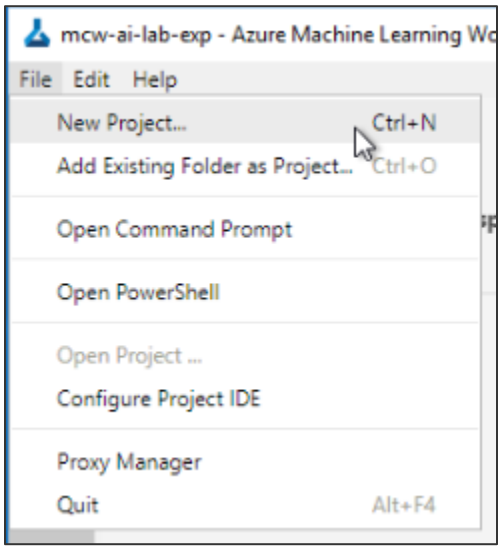
4. Sign in with the credentials you used when creating the Experimentation Service in the Azure Portal.



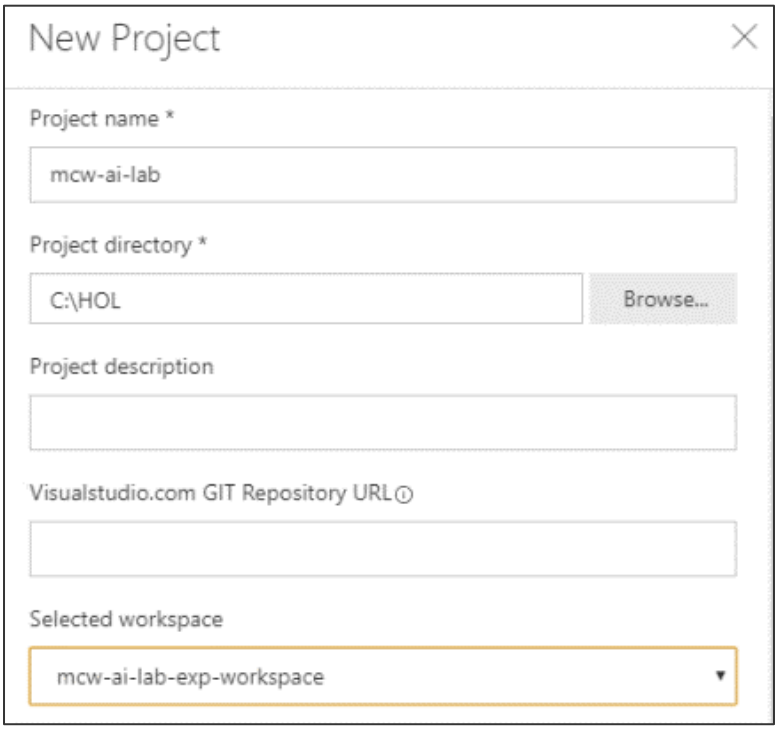
5. After successfully signing in, the Workbench interface should appear, listing the experimentation workspace that you created.



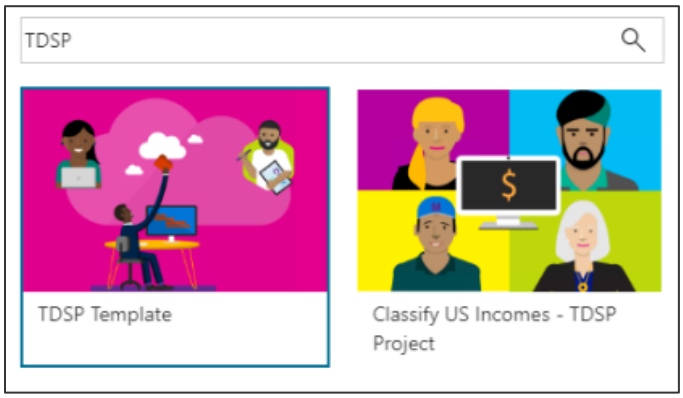
6. From the **File** Menu, select **New Project...**



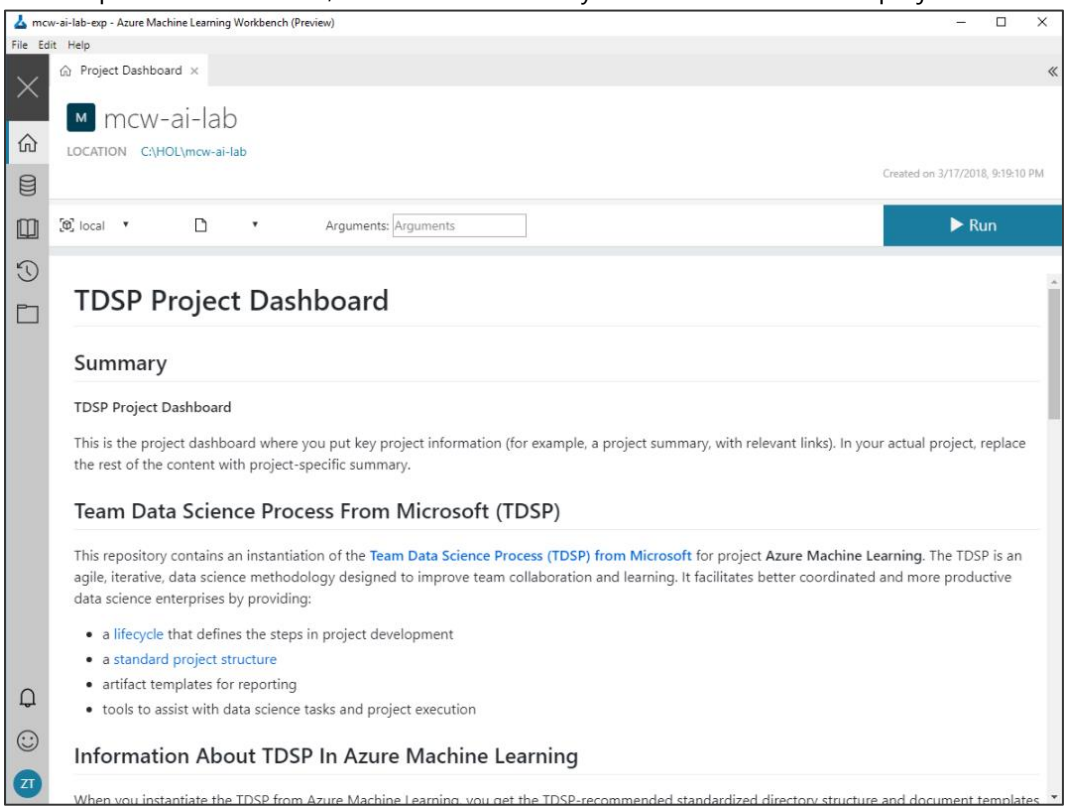
7. In the **New Project** blade that appears, provide the following:
   a. **Project name**: mcw-ai-lab
   b. **Project directory**: C:\HOL
   c. **Project description**: leave blank
   d. **Visualstudio.com GIT Repository URL**: leave blank
   e. **Selected workspace**: select the ML Experimentation Workspace you created.



8. In the Search Project Templates, enter **TDSP** and select the item called **TDSP Template**.



9. Select **Create.**
10. The template will download, and a few moments you should see the TDSP project dashboard.



Task 3: Install dependencies

The tasks that follow depend on Python libraries like nltk and gensim. The following steps ensure you have these installed in your environment.

1. From the **File** menu of Workbench, select **Open Command Prompt**.
2. Run the following command to install nltk:
   pip install nltk

3. NLTK should install, with a message similar to the following:

```
C:\HOL\mcw-ai-lab>pip install nltk
Collecting nltk
  Downloading nltk-3.2.5.tar.gz (1.2MB)
    100% |################################| 1.2MB 957kB/s
Requirement already satisfied (use --upgrade to upgrade): six in c:\users\demouser\appdata\local\amlworkbench\python\lib
\site-packages (from nltk)
Building wheels for collected packages: nltk
  Running setup.py bdist_wheel for nltk ... done
  Stored in directory: C:\Users\demouser\AppData\Local\pip\Cache\wheels\18\9c\1f\276bc3f421614062468cb1c9d695e6086d0c73d
67ea363c501
Successfully built nltk
Installing collected packages: nltk
Successfully installed nltk-3.2.5
You are using pip version 8.1.2, however version 9.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

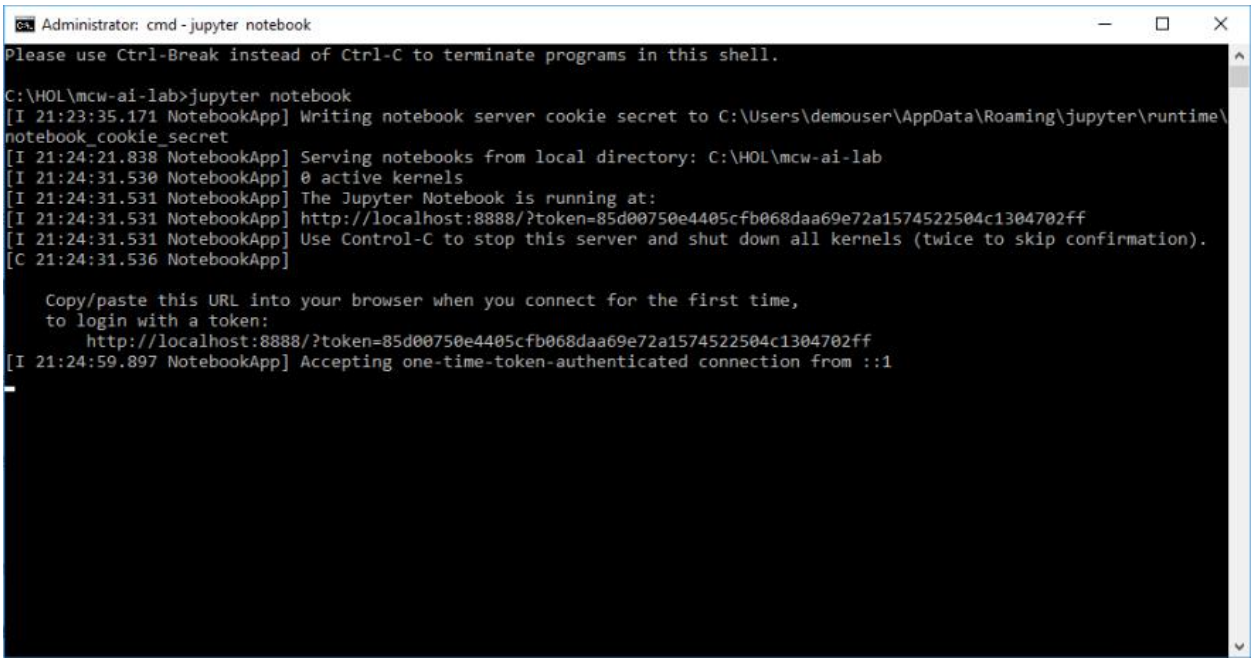
4. NLTK is a rich toolkit with modular components, many of which are not installed by default. To install all the components, run the python shell by entering **python** at the command prompt:
   python
5. Within the python shell, run the following two lines:
   import nltk
   nltk.download('all')
6. The downloader will take about 5 minutes to complete. Once it is finished, exit the python shell by entering:
   exit()
7. Run the following command to install genism:
   pip install gensim

```
C:\HOL\mcw-ai-lab>pip install gensim
Collecting gensim
  Downloading gensim-3.4.0-cp35-cp35m-win_amd64.whl (22.5MB)
    100% |################################| 22.5MB 46kB/s
Collecting smart-open>=1.2.1 (from gensim)
  Downloading smart_open-1.5.6.tar.gz
Requirement already satisfied (use --upgrade to upgrade): scipy>=0.18.1 in c:\users\demouser\appdata\local\amlworkbench\
python\lib\site-packages (from gensim)
Requirement already satisfied (use --upgrade to upgrade): six>=1.5.0 in c:\users\demouser\appdata\local\amlworkbench\pyt
hon\lib\site-packages (from gensim)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.11.3 in c:\users\demouser\appdata\local\amlworkbench\
python\lib\site-packages (from gensim)
Collecting boto>=2.32 (from smart-open>=1.2.1->gensim)
  Downloading boto-2.48.0-py2.py3-none-any.whl (1.4MB)
    100% |################################| 1.4MB 846kB/s
Collecting bz2file (from smart-open>=1.2.1->gensim)
  Downloading bz2file-0.98.tar.gz
Requirement already satisfied (use --upgrade to upgrade): requests in c:\users\demouser\appdata\local\amlworkbench\pytho
n\lib\site-packages (from smart-open>=1.2.1->gensim)
Collecting boto3 (from smart-open>=1.2.1->gensim)
  Downloading boto3-1.6.11-py2.py3-none-any.whl (128kB)
    100% |################################| 133kB 5.4MB/s
Requirement already satisfied (use --upgrade to upgrade): jmespath<1.0.0,>=0.7.1 in c:\users\demouser\appdata\local\amlw
orkbench\python\lib\site-packages (from boto3->smart-open>=1.2.1->gensim)
Collecting s3transfer<0.2.0,>=0.1.10 (from boto3->smart-open>=1.2.1->gensim)
  Downloading s3transfer-0.1.13-py2.py3-none-any.whl (59kB)
    100% |################################| 61kB 5.1MB/s
Collecting botocore<1.10.0,>=1.9.11 (from boto3->smart-open>=1.2.1->gensim)
  Downloading botocore-1.9.11-py2.py3-none-any.whl (4.1MB)
    100% |################################| 4.1MB 290kB/s
Collecting docutils>=0.10 (from botocore<1.10.0,>=1.9.11->boto3->smart-open>=1.2.1->gensim)
  Downloading docutils-0.14-py3-none-any.whl (543kB)
    100% |################################| 552kB 1.9MB/s
Requirement already satisfied (use --upgrade to upgrade): python-dateutil<2.7.0,>=2.1 in c:\users\demouser\appdata\local
\amlworkbench\python\lib\site-packages (from botocore<1.10.0,>=1.9.11->boto3->smart-open>=1.2.1->gensim)
Building wheels for collected packages: smart-open, bz2file
  Running setup.py bdist_wheel for smart-open ... done
  Stored in directory: C:\Users\demouser\AppData\Local\pip\Cache\wheels\36\48\35\97efc2bd1b233627131c9a936c9de23681846db
707b907d353
  Running setup.py bdist_wheel for bz2file ... done
  Stored in directory: C:\Users\demouser\AppData\Local\pip\Cache\wheels\31\9c\20\996d65ca104cbca940b1b053299b68459391c01
c774d073126
Successfully built smart-open bz2file
Installing collected packages: boto, bz2file, docutils, botocore, s3transfer, boto3, smart-open, gensim
Successfully installed boto-2.48.0 boto3-1.6.11 botocore-1.9.11 bz2file-0.98 docutils-0.14 gensim-3.4.0 s3transfer-0.1.1
3 smart-open-1.5.6
```

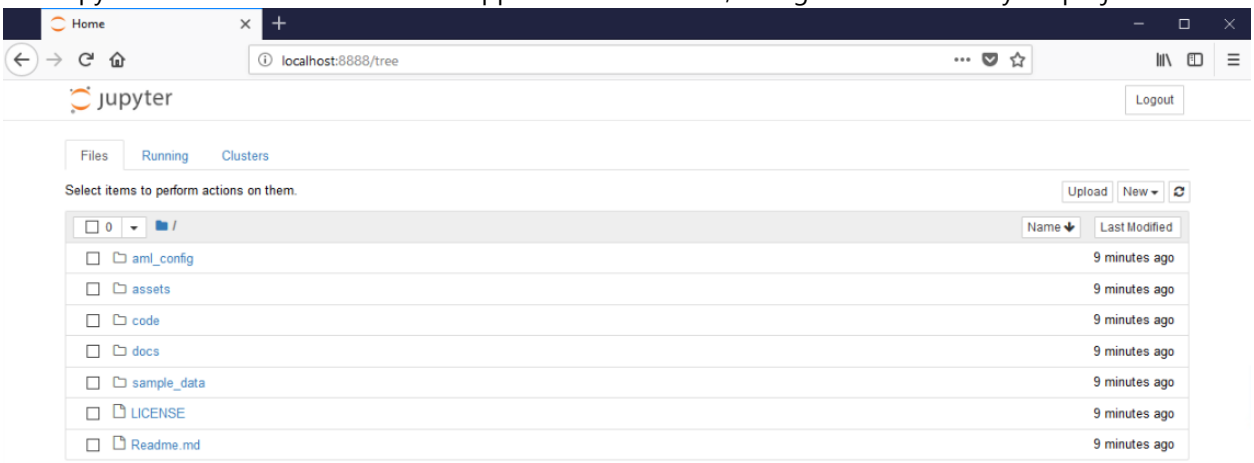
8. Next, download a pre-built Jupyter Notebook that you will step through to understand the process used to summarize the text of claims documents. In the **Firefox** browser on your VM, navigate to the following (note that the URL is case sensitive). Note, if using IE you will need to modify the default security settings, which prevent files from being downloaded.

   http://bit.ly/2G4hAQz

9. In the command prompt, enter the following and press enter to launch the Jupyter Notebook:
   jupyter notebook

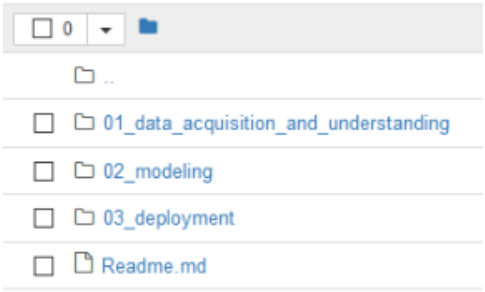


10. In a few moments, you should be prompted for which browser to use to open the link, select **Firefox**.
11. The Jupyter Notebook interface should appear in the browser, listing the contents of your project folder.



12. Select the **code** folder.

13. Select **01_data_acquisition_and_understanding**.



14. Select the **Upload** button.
15. Open the **Summarize.ipynb** notebook and follow the instructions within it.

Exercise 2: Deploy the Summarizer as a Service

Duration: 45 minutes

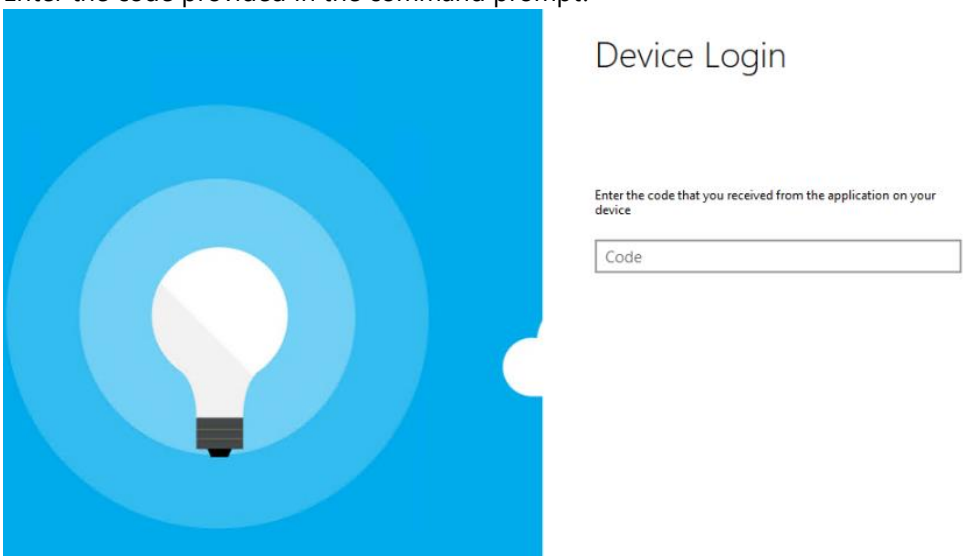
In this exercise you will create and deploy a web service that uses a pre-trained model to summarize long paragraphs of text.

Task 1: Deploy your ACS cluster

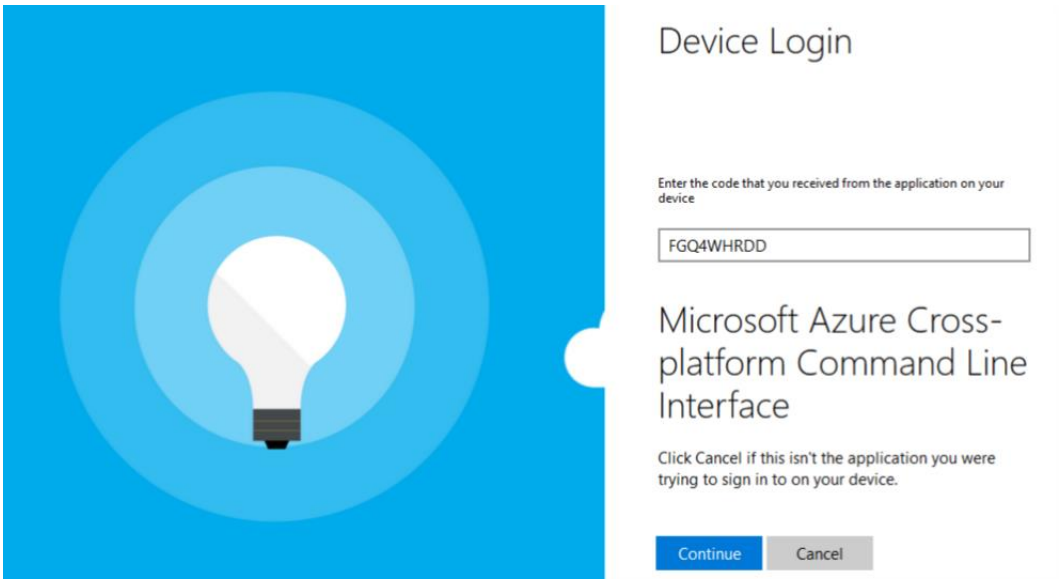
1. Within Workbench, from the **File** menu, select **Open Command Prompt.**
2. Create the cluster environment by running the following command, replacing the values indicated in angle brackets with appropriate values. This will create new resources groups for the cluster.
    a. For <environment name> enter mcwailabenv, or something similar. This value can only contain lowercase alphanumeric characters.
    b. For location, use eastus2, westcentralus, australiaeast, westeurope, or southeastasia, as those are the only acceptable values at this time.

    az ml env setup -c -n <environment name> --location <e.g. eastus2>

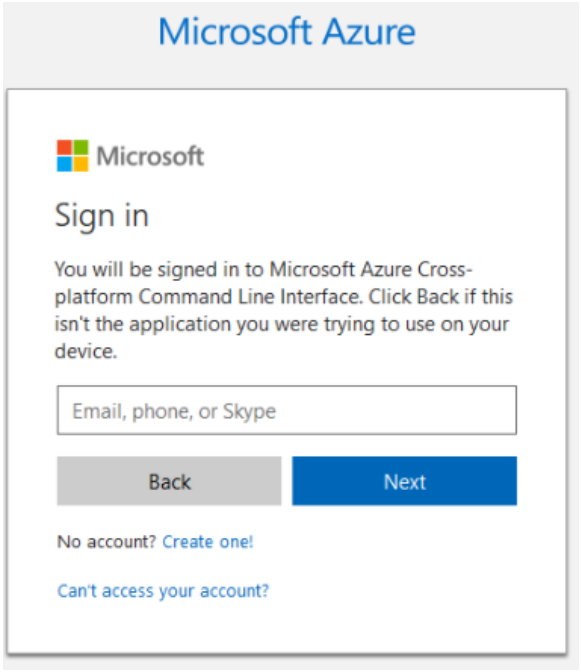
3. When prompted, copy the URL presented and sign in using your web browser.
4. Enter the code provided in the command prompt.



5. Select **Continue.**

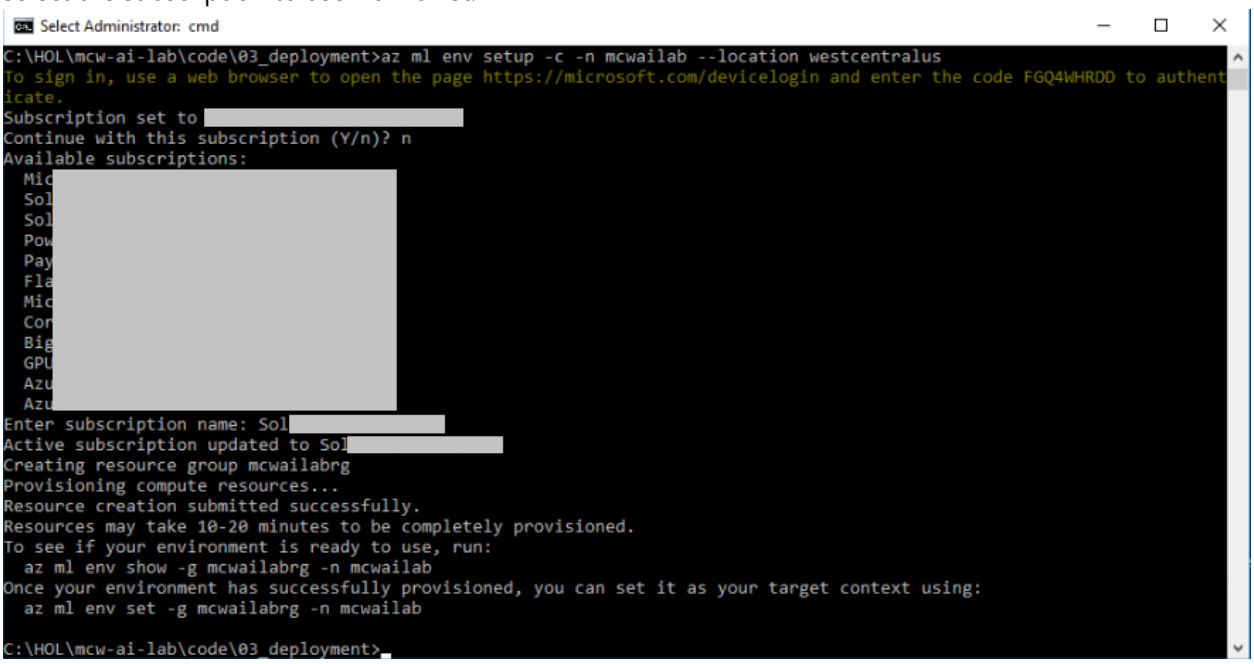


6. Sign in with your Azure Credentials.

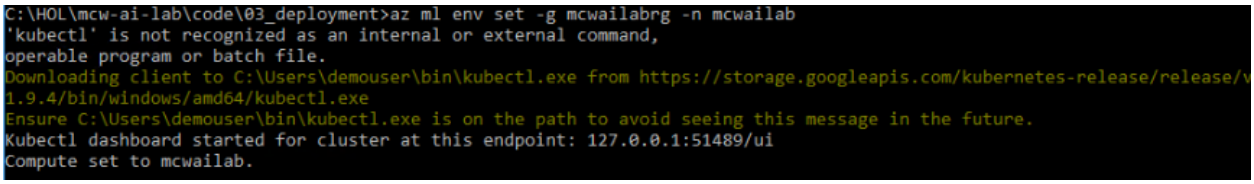


7. Return to the command prompt, which should automatically update after you log in.

8. At the "Subscription set to <subscription name>" prompt, enter **Y** if the subscription name is correct, or **N** to select the subscription to use from a list.



9. It will take 10-20 minutes for your ACS cluster to be ready. You can periodically check on the status by running the command shown in the output to the previous step, which is of the form:
   az ml env show -g <resourceGroupName> -n <clusterName>
10. Once the environment has successfully provisioned (the Provisioning State in the above command will read "Succeeded"), run the other command provided in step 8, which is of the form:
    az ml env set -g <resourceGroupName> -n <clusterName>

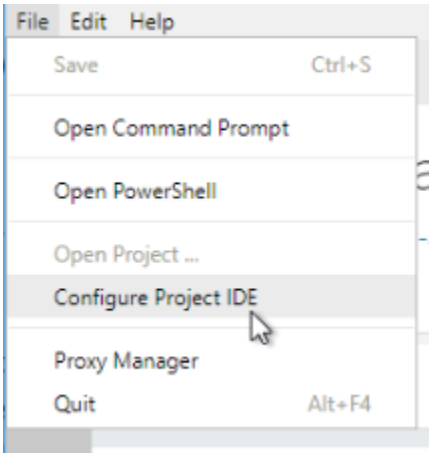


11. This will set the context of the command line to target this environment.
12. Finally, set the model management account to be used by the command line, to be the one you created previously (mcw-ai-lab-model-mgmt). Run the following command, replacing the values indicated in angle brackets with appropriate values.
    a. For <acctname>, enter the name of the Machine Learning Model Management resource in your mcw-ai-lab resource group.
    b. For <resourcegroupname>, use your mcw-ai-lab resource group name.

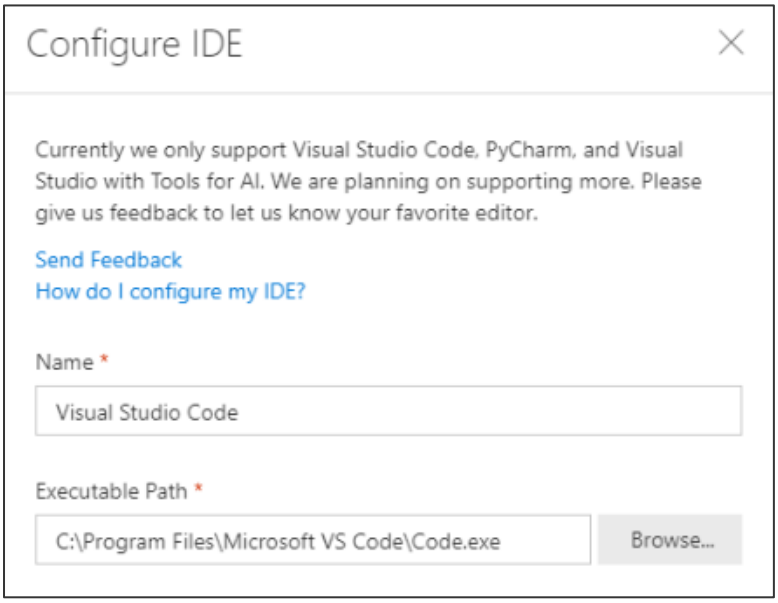
    az ml account modelmanagement set -n <acctname> -g <resourcegroupname>

Task 2: Set Visual Studio Code as the project IDE in Workbench

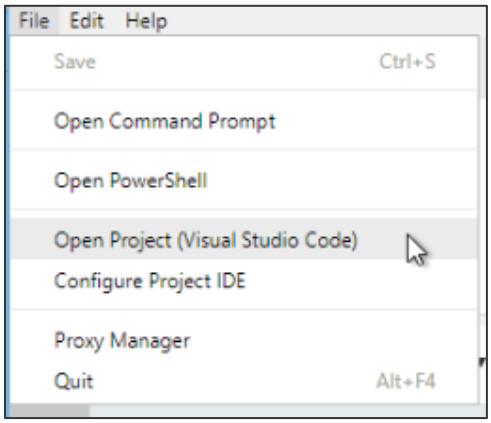
1. Within Workbench, from the **File** menu, select **Configure Project IDE.**



2. In the **Configure IDE** blade that appears, set the following properties:
   a. **Name**: Visual Studio Code
   b. **Executable Path**: C:\Program Files\Microsoft VS Code\Code.exe



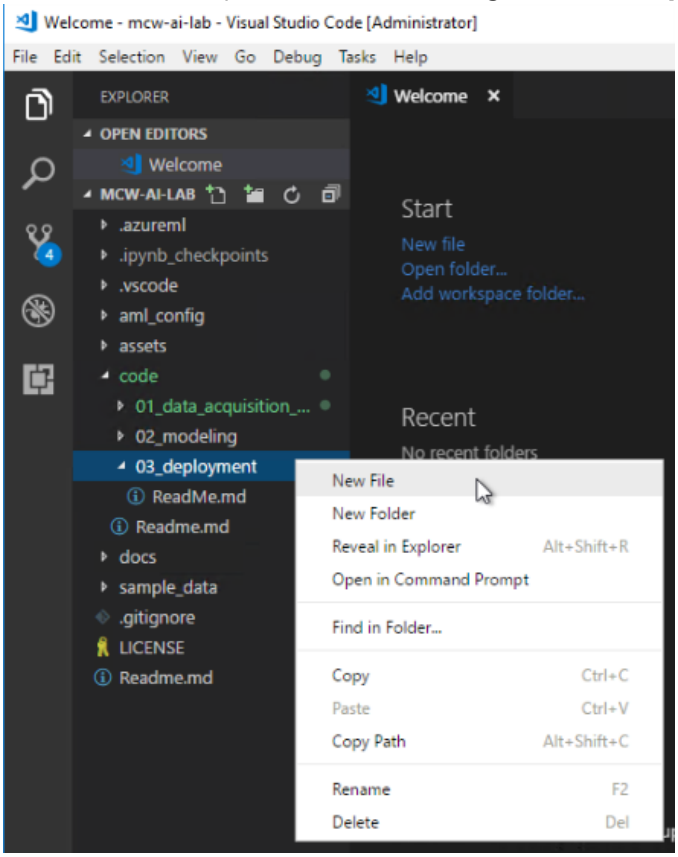
3. Select **OK**.
4. Launch Visual Studio Code for the project by selecting **Open Project (Visual Studio Code)** from the **File** menu.



5. You are now ready to author service script.

Task 3: Create the Summarization service

1. Visual Studio Code will open against the project directory.
2. In the tree view, expand **code** and then right-click **03_deployment** and select **New File.**

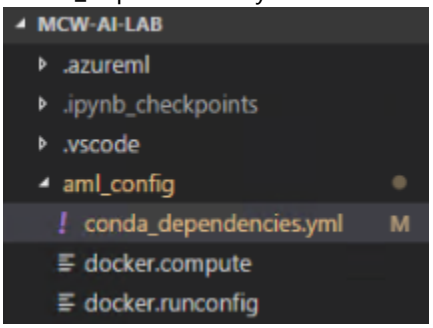


3. For the file name, enter summarizer_service.py and press **Enter**.
4. In a browser, navigate to http://bit.ly/2FLJn8Y and copy the contents of the file.
5. Paste the contents of this script into your summarizer_service.py. Take a moment to review the script, as it is effectively the same code you were running in the Jupyter notebook, except that is has been modified to follow the format required by services in Azure Machine Learning. The init method is called once per container by the Azure Machine Learning infrastructure when the service is deployed. It is here that we need to load all of the modules required by NLTK in the call to nltk.download. The run method is where any scoring (or in our case

summarization) activity takes place.

```python
from azureml.datacollector import ModelDataCollector
import re
import nltk
import unicodedata
import numpy as np
from gensim.summarization import summarize, keywords

def clean_and_parse_document(document):
    document = re.sub('\n', ' ', document)
    if isinstance(document, str):
        document = document
    elif isinstance(document, unicode):
        return unicodedata.normalize('NFKD', document).encode('ascii', 'ignore')
    else:
        raise ValueError("Document is not string or unicode.")
    document = document.strip()
    sentences = nltk.sent_tokenize(document)
    sentences = [sentence.strip() for sentence in sentences]
    return sentences

def summarize_text(text, summary_ratio=None, word_count=30):
    sentences = clean_and_parse_document(text)
    cleaned_text = ' '.join(sentences)
    summary = summarize(cleaned_text, split=True, ratio=summary_ratio, word_count=word_count)
    return summary

def init():
    nltk.download('all')
    return

def run(input_str):
    try:
        return summarize_text(input_str)
    except Exception as e:
        return (str(e))
```

6. Next, we need to capture the dependencies for the modules used by the script. These are declared in a conda environment file, which you can generate from an environment or create by hand. In this case, we will edit the default conda environment provided by the TDSP project by hand, and add a configuration that will pip install gensim as required by our script. To do this, in Visual Studio Code, expand, aml_config and open conda_dependencies.yml.



7. At the last line, under azure-ml-api-sdk add another line with -gensim to the pip configuration. You should also add entries for tensorflow and tflearn, which we will need later in the lab. Your final configuration should look as

follows:

```
name: project_environment
dependencies:
  - python=3.5.2
  - scikit-learn
  - pip:
    # The API for Azure Machine Learning Model Management Service.
    # Details: https://github.com/Azure/Machine-Learning-Operationalization
    - azure-ml-api-sdk==0.1.0a11
    - gensim
    - tensorflow
    - tflearn
```

8. Save the file. When we go to create the image in a later step, this file will be included with command.
9. Next, create an empty file called dummy_model.bin in the 03_deployment folder. In this case, we don't have a model to deploy with this service, but we still need to provide one to the CLI as we will see in a moment. An empty file will do.

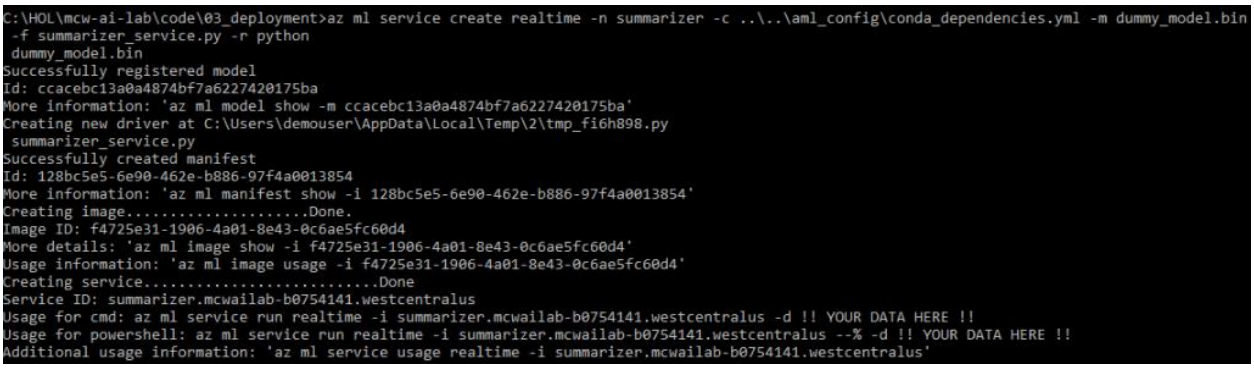
Task 4: Deploy the Summarization service

1. Return to the Workbench and use the **File** menu to open another command prompt.
2. At the command prompt, change directories to the code\03_deployment directory by executing the following command:

   ```
   cd code\03_deployment
   ```

3. You can deploy the service using a single command (which orchestrates the multiple steps of creating a docker manifest, creating a docker image, and deploying a container instance from the image). The command needs to refer to all the components required for the service including the dummy model file, the service script, the conda dependencies and the runtime to use (python in this case). Run the following command to deploy the summarizer service:

   ```
   az ml service create realtime -n summarizer -c ..\..\aml_config\conda_dependencies.yml -m dummy_model.bin -f summarizer_service.py -r python
   ```



4. Notice in the output of the preceding command, you are provided with instructions (third line from last) on how you can invoke the deployed service using the CLI. Try executing the following command (modify the Service ID of

you service as indicated in the previous command output):

az ml service run realtime -i summarizer.**[mcwailab-xyz.location]** -d "I was driving down El Camino and stopped at a red light. It was about 3pm in the afternoon. The sun was bright and shining just behind the stoplight. This made it hard to see the lights. There was a car on my left in the left turn lane. A few moments later another car, a black sedan pulled up behind me. When the left turn light changed green, the black sedan hit me thinking that the light had changed for us, but I had not moved because the light was still red. After hitting my car, the black sedan backed up and then sped past me. I did manage to catch its license plate. The license plate of the black sedan was ABC123."

```
C:\HOL\mcw-ai-lab\code\03_deployment>az ml service run realtime -i summarizer.mcwailab-b0754141.westcentralus -d "I was driving down El Camino
 and stopped at a red light. It was about 3pm in the afternoon. The sun was bright and shining just behind the stoplight. This made it hard to
 see the lights. There was a car on my left in the left turn lane. A few moments later another car, a black sedan pulled up behind me. When th
e left turn light changed green, the black sedan hit me thinking that the light had changed for us, but I had not moved because the light was
still red. After hitting my car, the black sedan backed up and then sped past me. I did manage to catch its license plate. The license plate o
f the black sedan was ABC123."
['When the left turn light changed green, the black sedan hit me thinking that the light had changed for us, but I had not moved because the l
ight was still red.']
```

5. If you get a summary back, your service is working! Try calling the service with other text and observe the summary returned. Note that the service tries to build a summary of about 30 words, so if you provide too short a text, an empty summary will be returned.
6. Finally, in a notepad or other location take note of the full Service ID (e.g., summarizer.mcwailab-xyz.location) and the authorization key which you will need later in the lab. To get the authorization key for your deployed service, run the following command and take note of the PrimaryKey value in the output:

   az ml service keys realtime -i summarizer.**[mcwailab-xyz.location]**

```
C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service>az ml service keys realtime -i summarizer.mcwailab-b0754141.wes
tcentralus
PrimaryKey: 24c60eb8a60d483286b78f37441cbd72
SecondaryKey: f23b7d3e9884470880970d8c5a6b4914
```

Exercise 3: Applying TensorFlow

Duration: 60 minutes

In this exercise, you use TensorFlow to construct and train a simple deep neural network classification model that will classify claim text as belonging to a home insurance claim or an automobile claim. You will then deploy this trained model as a web service.

Task 1: Prepare TensorFlow

1. Return to your RDP session to the Data Science VM.
2. Switch to the command prompt that is running the Jupyter Notebook command and press **Control + Break**. This will stop the Jupyter Notebook process while you update TensorFlow.
3. From the command line run:

   pip install tensorflow

4. In a few moments, the install should complete, and you should see output ending similar to the following:

```
Successfully installed absl-py-0.1.11 astor-0.6.2 bleach-1.5.0 gast-0.2.0 grpcio-1.10.0 html5lib-0.9999999 markdown-2.6.
11 protobuf-3.5.2.post1 tensorboard-1.6.0 tensorflow-1.6.0 termcolor-1.1.0 werkzeug-0.14.1
You are using pip version 8.1.2, however version 9.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\HOL\mcw-ai-lab>
```

5. We will be using the TFLearn library which sits atop TensorFlow. To install it run:

   pip install tflearn

```
C:\HOL\mcw-ai-lab>pip install tflearn
Collecting tflearn
  Downloading tflearn-0.3.2.tar.gz (98kB)
    100% |################################| 102kB 2.6MB/s
Requirement already satisfied (use --upgrade to upgrade): numpy in c:\users\demouser\appdata\local\amlworkbench\python\l
ib\site-packages (from tflearn)
Requirement already satisfied (use --upgrade to upgrade): six in c:\users\demouser\appdata\local\amlworkbench\python\lib
\site-packages (from tflearn)
Collecting Pillow (from tflearn)
  Downloading Pillow-5.0.0-cp35-cp35m-win_amd64.whl (1.6MB)
    100% |################################| 1.6MB 747kB/s
Building wheels for collected packages: tflearn
  Running setup.py bdist_wheel for tflearn ... done
  Stored in directory: C:\Users\demouser\AppData\Local\pip\Cache\wheels\fb\06\72\0478c938ca315c6fddcce8233b80ec91a115ce4
496a27e8c90
Successfully built tflearn
Installing collected packages: Pillow, tflearn
Successfully installed Pillow-5.0.0 tflearn-0.3.2
```

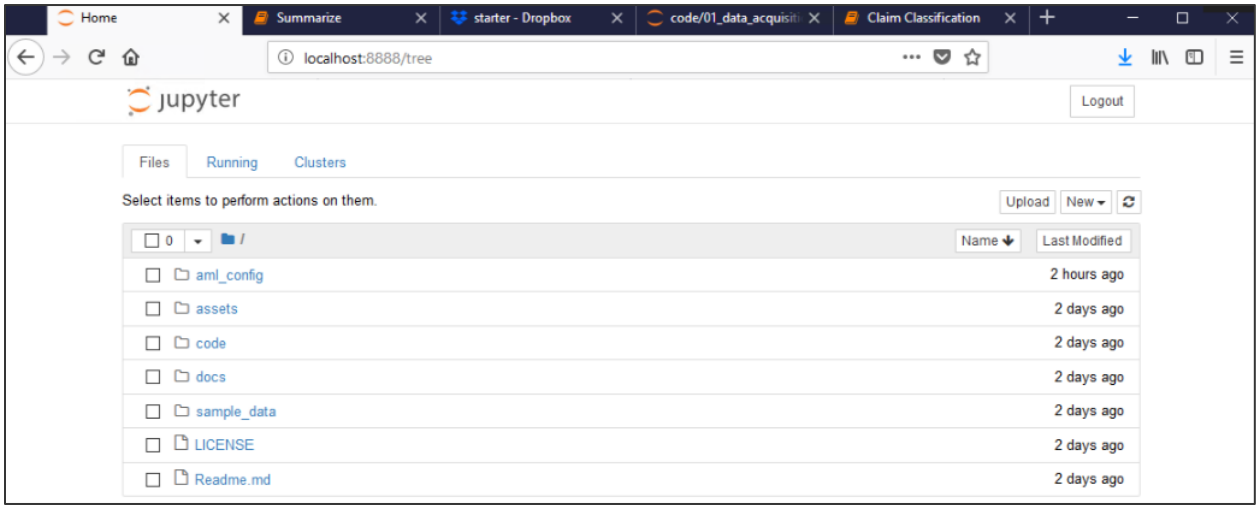
6. Run "Jupyter Notebook" to re-start the process.
7. You should now be ready to use TensorFlow on your Data Science VM.

Task 2: Train and deploy the TensorFlow model

1. Return to your RDP session to the Data Science VM.
2. Download the TensorFlow notebook, text analytics helper module and sample data from the following link:

http://bit.ly/2pucpje

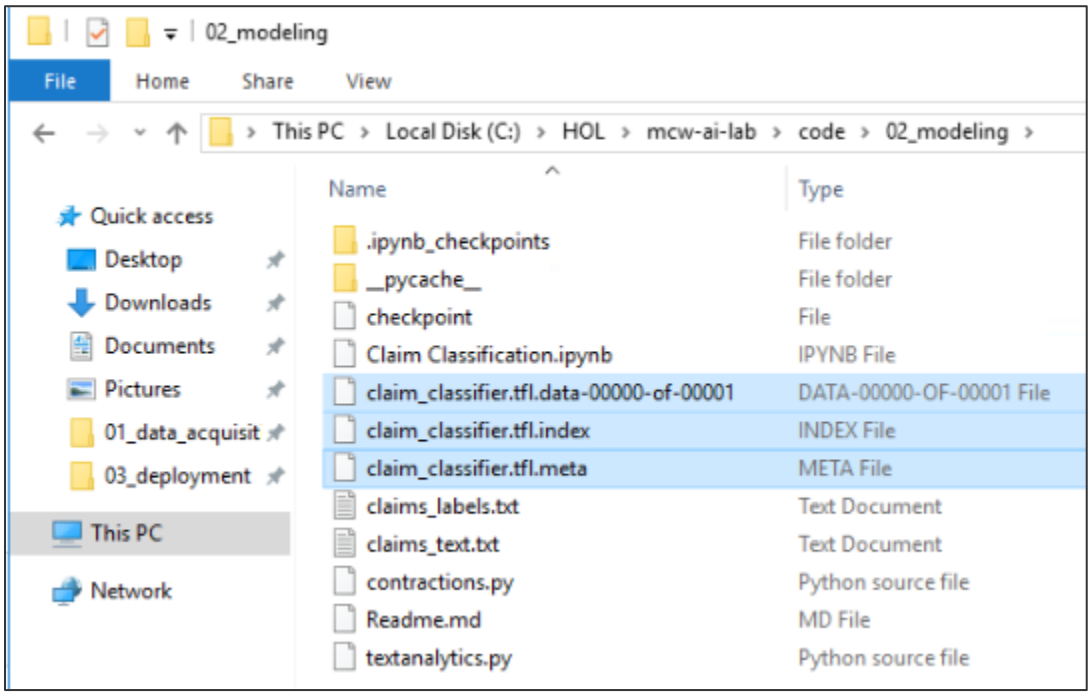
3. Extract this zip and copy the contents to C:\HOL\mcw-ai-lab\code\02_modeling.
4. Return to the instance of the Jupyter Notebook home that should be open in your browser.



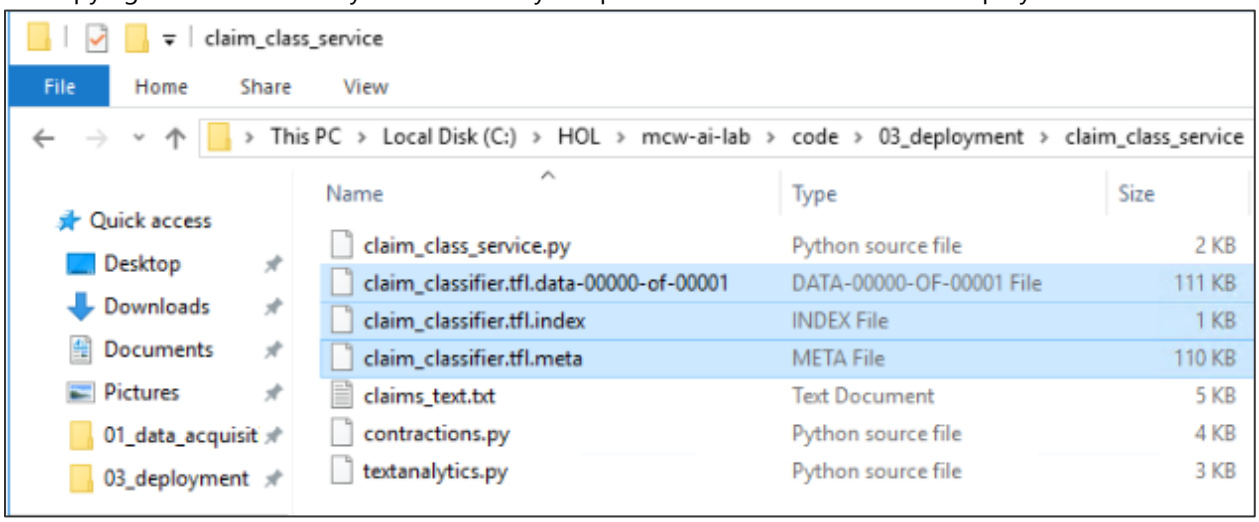
5. Select the code folder**, 02_modeling**. You should see a folder listing similar to the following. Select **Claim Classification.ipynb**.

6. The Claim Classification notebook will appear. Step through this notebook to read how the data is prepared and the neural network model is trained. Be sure to execute each cell as you get to it.
7. When you have finished executing the notebook, some model files will have been produced. Using File Explorer, navigate to **C:\HOL\mcw-ai-lab\code\02_modeling**, you should see the three new files (each beginning with

claim_classifier.tfl).



8. Copy these three files and paste them under C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service. You are copying these over so they can be used by the predictive web service we will deploy.

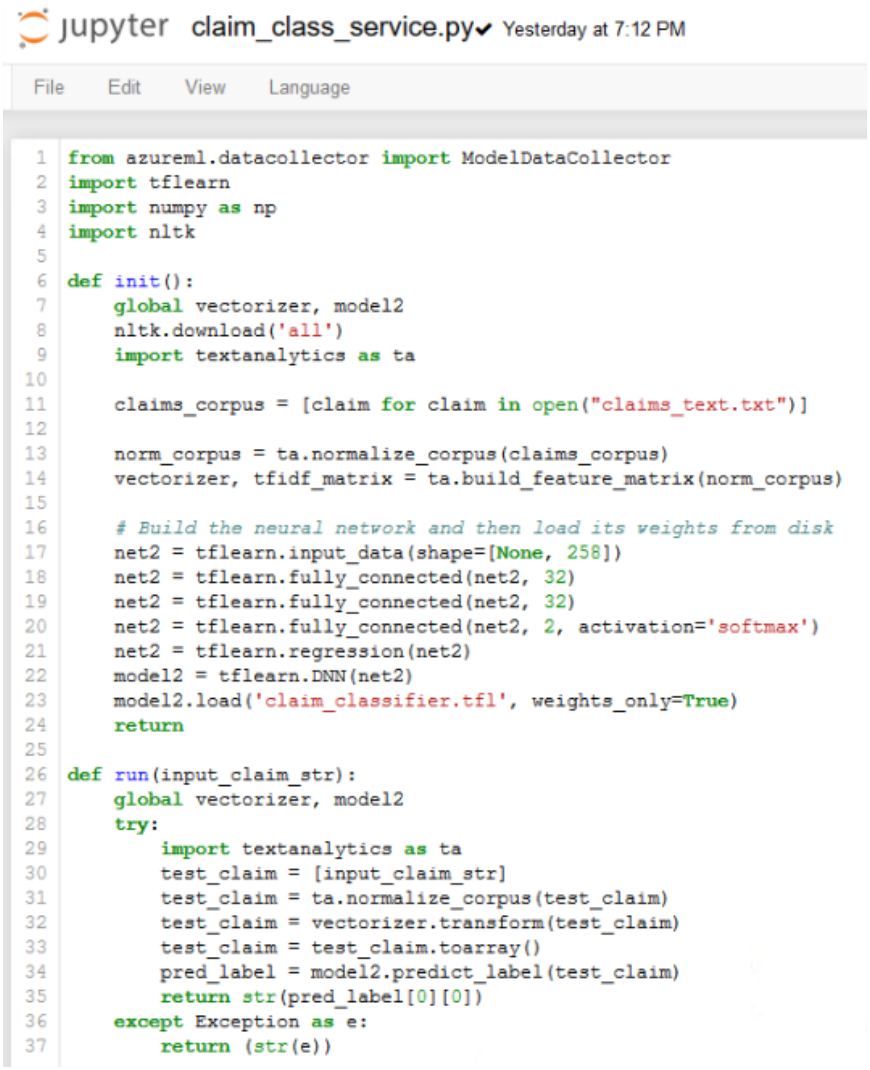


9. Next, download the supporting files for the claim_class_service from:

   http://bit.ly/2u5DoGH

10. Extract the files and copy them into C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service.
11. Return to the instance of the Jupyter Notebook home that should be open in your browser.
12. Select the code folder, **03_deployment** and then **claim_class_service**.
13. Open **claim_class_service.py**. Observe that the code it uses is like what you ran in the Claim Classification notebook, only formatted to fit the structure of an Azure Machine Learning web service (with init and run

methods).




```
from azureml.datacollector import ModelDataCollector
import tflearn
import numpy as np
import nltk

def init():
    global vectorizer, model2
    nltk.download('all')
    import textanalytics as ta

    claims_corpus = [claim for claim in open("claims_text.txt")]

    norm_corpus = ta.normalize_corpus(claims_corpus)
    vectorizer, tfidf_matrix = ta.build_feature_matrix(norm_corpus)

    # Build the neural network and then load its weights from disk
    net2 = tflearn.input_data(shape=[None, 258])
    net2 = tflearn.fully_connected(net2, 32)
    net2 = tflearn.fully_connected(net2, 32)
    net2 = tflearn.fully_connected(net2, 2, activation='softmax')
    net2 = tflearn.regression(net2)
    model2 = tflearn.DNN(net2)
    model2.load('claim_classifier.tfl', weights_only=True)
    return

def run(input_claim_str):
    global vectorizer, model2
    try:
        import textanalytics as ta
        test_claim = [input_claim_str]
        test_claim = ta.normalize_corpus(test_claim)
        test_claim = vectorizer.transform(test_claim)
        test_claim = test_claim.toarray()
        pred_label = model2.predict_label(test_claim)
        return str(pred_label[0][0])
    except Exception as e:
        return (str(e))
```

14. Next, you will deploy this service. Switch to your command line window and navigate to **C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service**.
15. Run the following command in the context of the claim_class_service folder to deploy the service:

    az ml service create realtime -n claimclassifier -c ..\..\..\aml_config\conda_dependencies.yml -m claim_classifier.tfl.meta -f claim_class_service.py -r python -d claim_classifier.tfl.data-00000-of-00001 -d claim_classifier.tfl.index -d claims_text.txt -d textanalytics.py -d contractions.py

```
C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service>az ml service create realtime -n claimclassifier -c ..\..\..\am
l_config\conda_dependencies.yml -m claim_classifier.tfl.meta -f claim_class_service.py -r python -d claim_classifier.tfl
.data-00000-of-00001 -d claim_classifier.tfl.index -d claims_text.txt -d textanalytics.py -d contractions.py
 claim_classifier.tfl.meta
Successfully registered model
Id: 0ece759a08944797bba74da0475ce00b
More information: 'az ml model show -m 0ece759a08944797bba74da0475ce00b'
Creating new driver at C:\Users\demouser\AppData\Local\Temp\2\tmpx1539kqx.py
 claim_classifier.tfl.data-00000-of-00001
 claim_classifier.tfl.index
 claims_text.txt
 textanalytics.py
 contractions.py
 claim_class_service.py
Successfully created manifest
Id: 2aba20ea-a84d-49f0-b391-1641c90cff08
More information: 'az ml manifest show -i 2aba20ea-a84d-49f0-b391-1641c90cff08'
Creating image.............................Done.
Image ID: 6911d076-422e-4b7b-9452-2d445d3ad198
More details: 'az ml image show -i 6911d076-422e-4b7b-9452-2d445d3ad198'
Usage information: 'az ml image usage -i 6911d076-422e-4b7b-9452-2d445d3ad198'
Creating service.............................Done
Service ID: claimclassifier.mcwailab-b0754141.westcentralus
Usage for cmd: az ml service run realtime -i claimclassifier.mcwailab-b0754141.westcentralus -d !! YOUR DATA HERE !!
Usage for powershell: az ml service run realtime -i claimclassifier.mcwailab-b0754141.westcentralus --% -d !! YOUR DATA
HERE !!
Additional usage information: 'az ml service usage realtime -i claimclassifier.mcwailab-b0754141.westcentralus'
```

16. Next, test the deployed service by running the following command (substitute the values of the Service ID as indicated in the last line of the previous):

    az ml service run realtime -i claimclassifier.**[mcwailab-xyz.location]** -d "A tornado ripped through my home."

```
C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service>az ml service run realtime -i claimclassifier.mcwailab-b0754141
.westcentralus -d "I destroyed my car by running into a deer"
1

C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service>az ml service run realtime -i claimclassifier.mcwailab-b0754141
.westcentralus -d "A tornado ripped thru my home"
0
```

17. Recall the classifier will return 1 if the text is classified as related to a car insurance claim and 0 if the claim pertains to a home insurance claim. Try submitting a few different sentences to the service.
18. Next, in a notepad or other location take note of the full Service ID (e.g., claimclassifier.mcwailab-xyz.location) and the authorization key which you will need later in the lab. To get the authorization key for your deployed service, run the following command and take note of the PrimaryKey value in the output:

    az ml service keys realtime -i claimclassifier.**[mcwailab-xyz.location]**

```
C:\HOL\mcw-ai-lab\code\03_deployment\claim_class_service>az ml service keys realtime -i claimclassifier.mcwailab-b075414
1.westcentralus
PrimaryKey: 7b4028a1178642e8bcc6aa1485395004
SecondaryKey: d307080d48c94d51a420c12e90bb63a0
```

19. Finally, run the following command to retrieve the IP address of your claimclassifier and summarizer services, and note the value in notepad or other location for use later in the lab. The IP address will be the same for both services.

    az ml service usage realtime -i claimclassifier.**[mcwailab-xyz.location]**

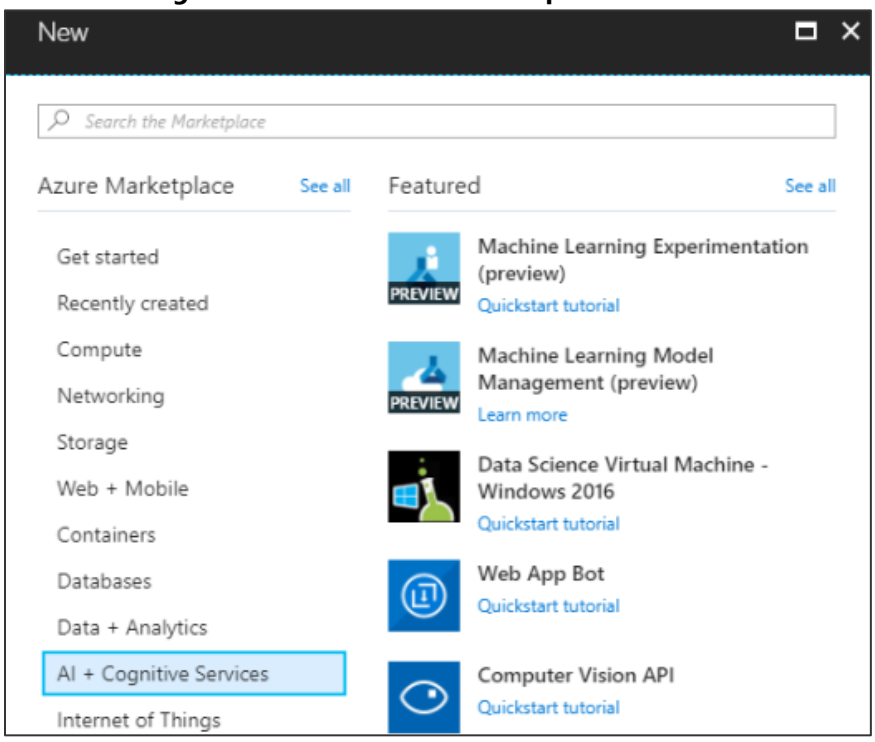
Exercise 4: Completing the solution

Duration: 45 minutes

In this exercise, you perform the final integration with the Computer Vision API and the Text Analytics API along with the Azure Machine Learning Services you previously deployed to deliver the completed proof of concept solution.

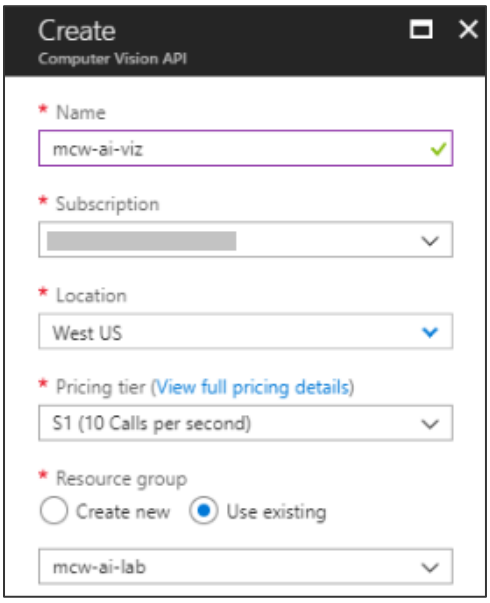
Task 1: Deploy the Computer Vision API

1. Navigate to the Azure Portal in your browser.
2. Select **Create a resource.**
3. Select **AI + Cognitive Services** and then **Computer Vision API.**

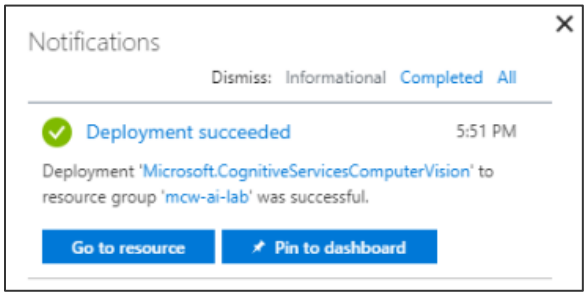


4. On the **Create** blade, provide the following:
    a. **Name:** provide a unique name for this instance.
    b. **Subscription:** select your Azure subscription.
    c. **Location**: select a location nearest your other deployed services.
    d. **Pricing tier**: select S1.

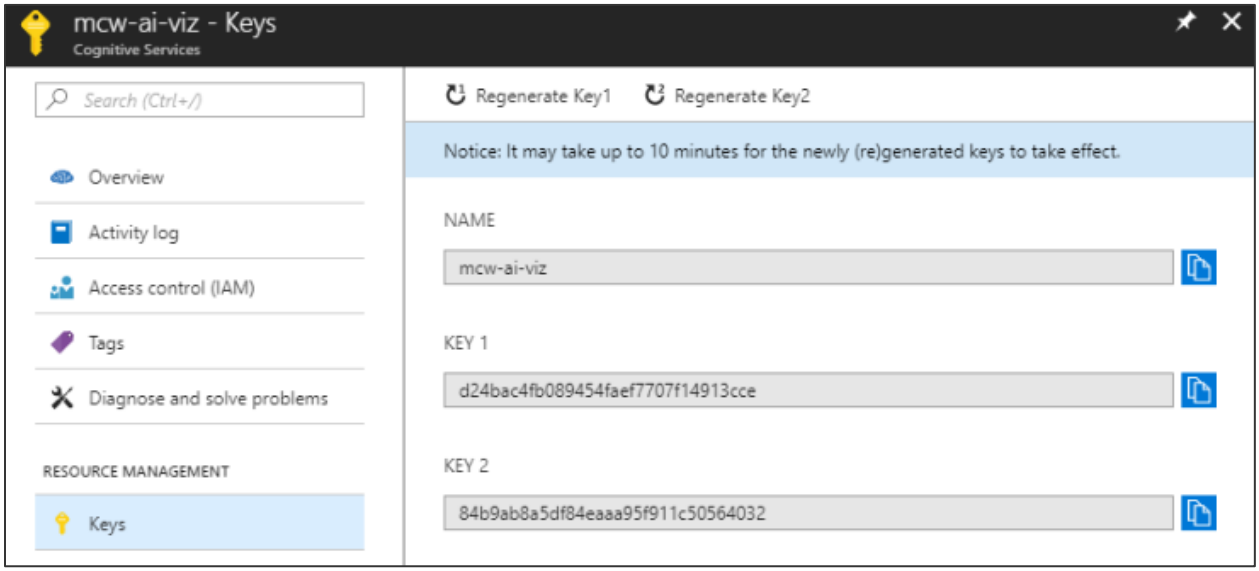
e. **Resource group**: select the existing mcw-ai-lab resource group.



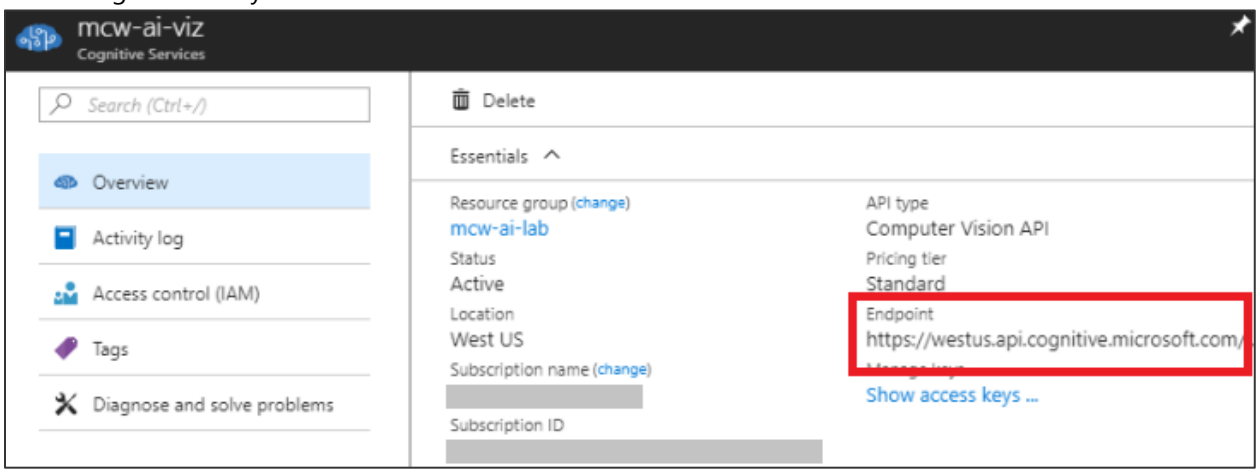
5. Select **Create**.
6. When the notification appears that the deployment succeeded, select **Go to resource**.



7. Select **Keys** and then copy the value of Key 1 into notepad or something similar as you will need this value later in the lab.



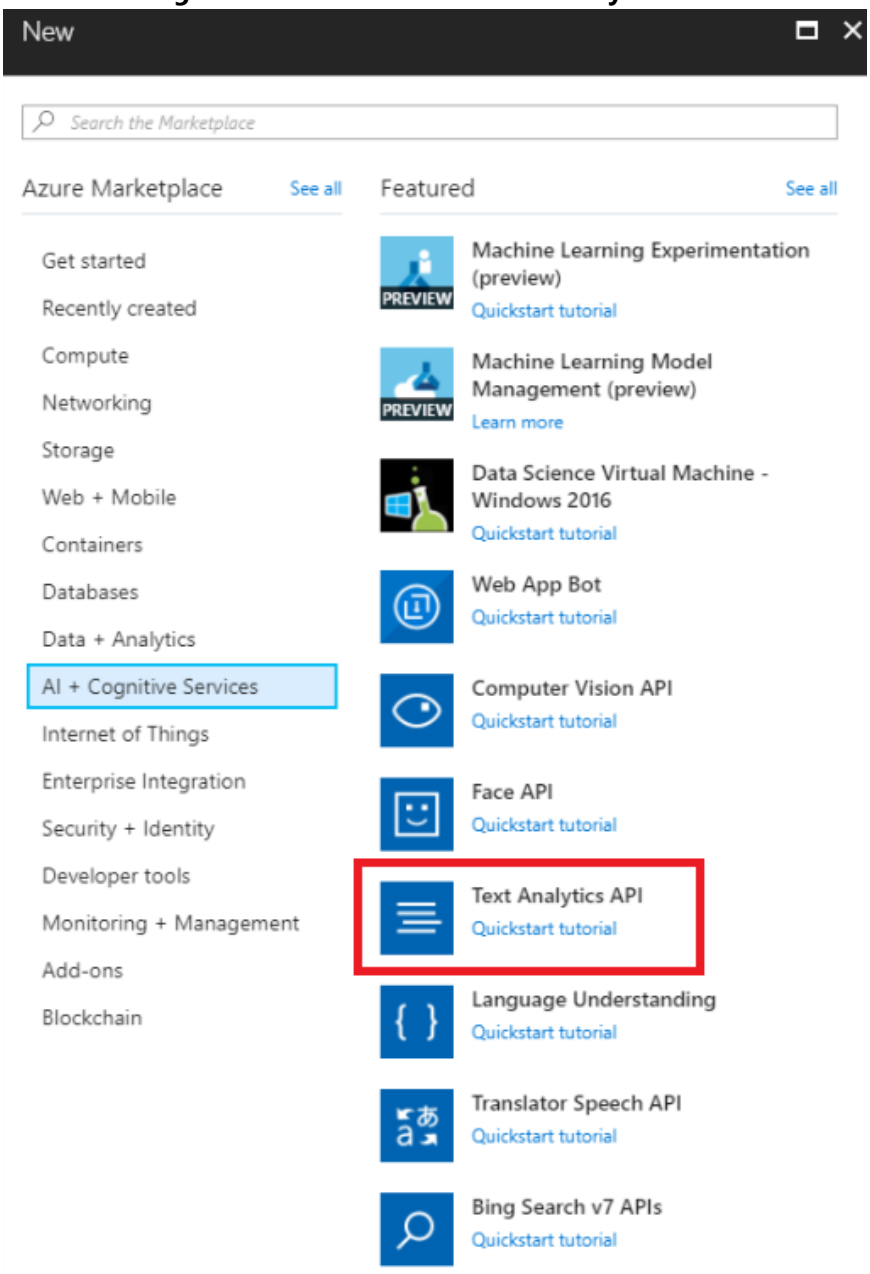
8. Select **Overview** and copy the value of Endpoint from the Essentials panel. Store this value in notepad or something similar as you will need this value later in the lab.



Task 2: Deploy the Text Analytics API

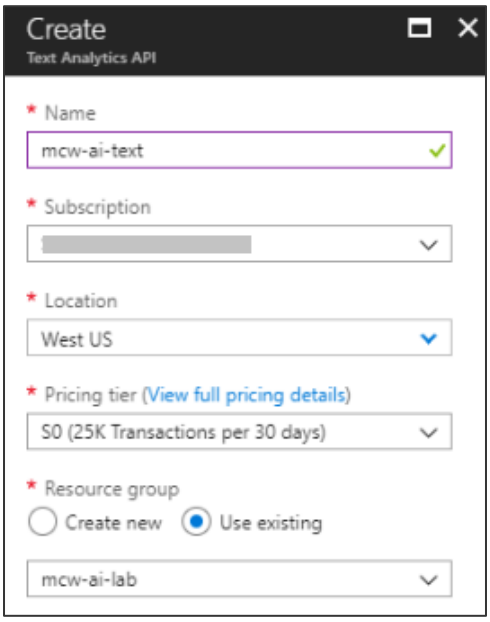
1. Navigate to the Azure Portal in your browser.
2. Select **Create a resource**.

3. Select **AI + Cognitive Services** and then **Text Analytics API.**

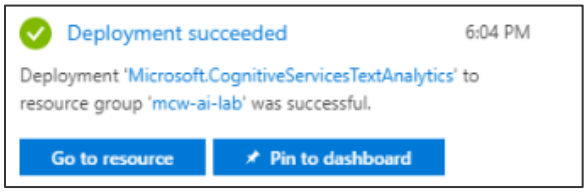


4. On the **Create** blade, provide the following:
    a. **Name**: provide a unique name for this instance.
    b. **Subscription**: select your Azure subscription.
    c. **Location**: select a location nearest your other deployed services.
    d. **Pricing tier**: select S0.

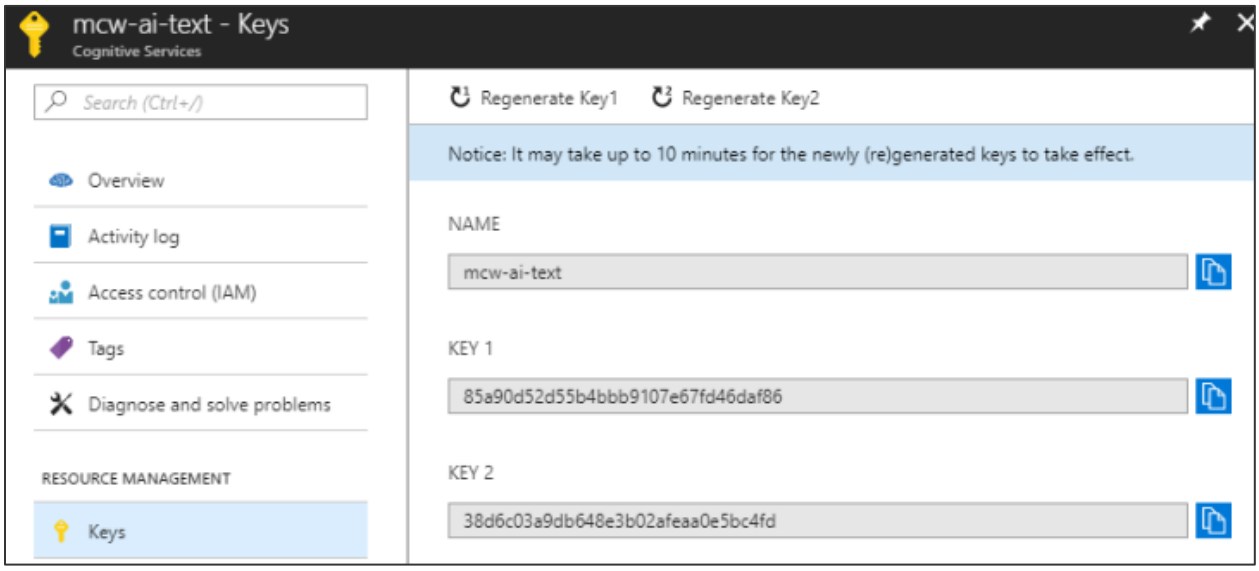
e. **Resource group**: select the existing mcw-ai-lab resource group.



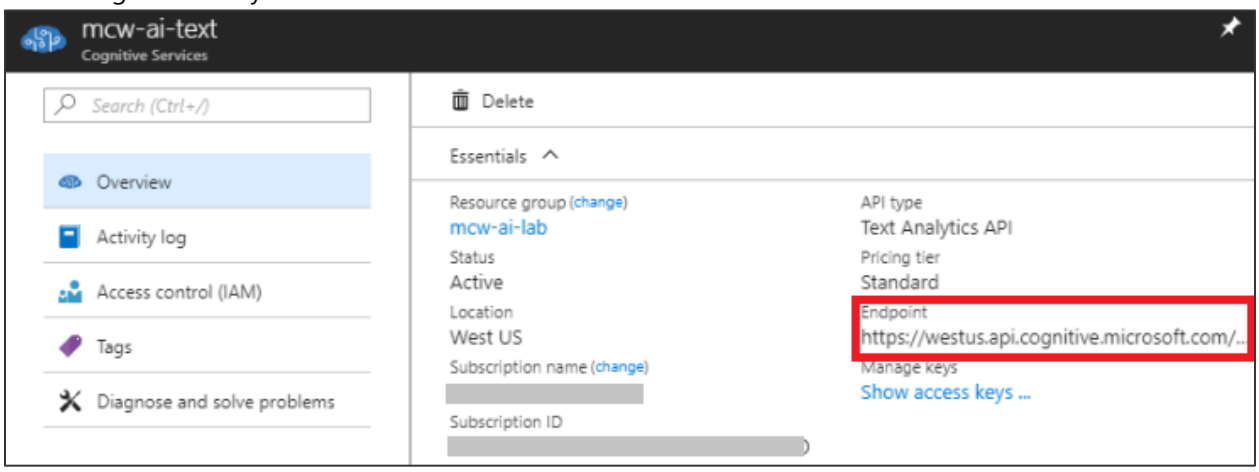
5. Select **Create**.
6. When the notification appears that the deployment succeeded, select **Go to resource**.



7. Select **Keys** and then copy the value of Key 1 in to notepad or something similar as you will need this value later in the lab.



8. Select **Overview** and copy the value of Endpoint from the Essentials panel. Store this value in notepad or something similar as you will need this value later in the lab.

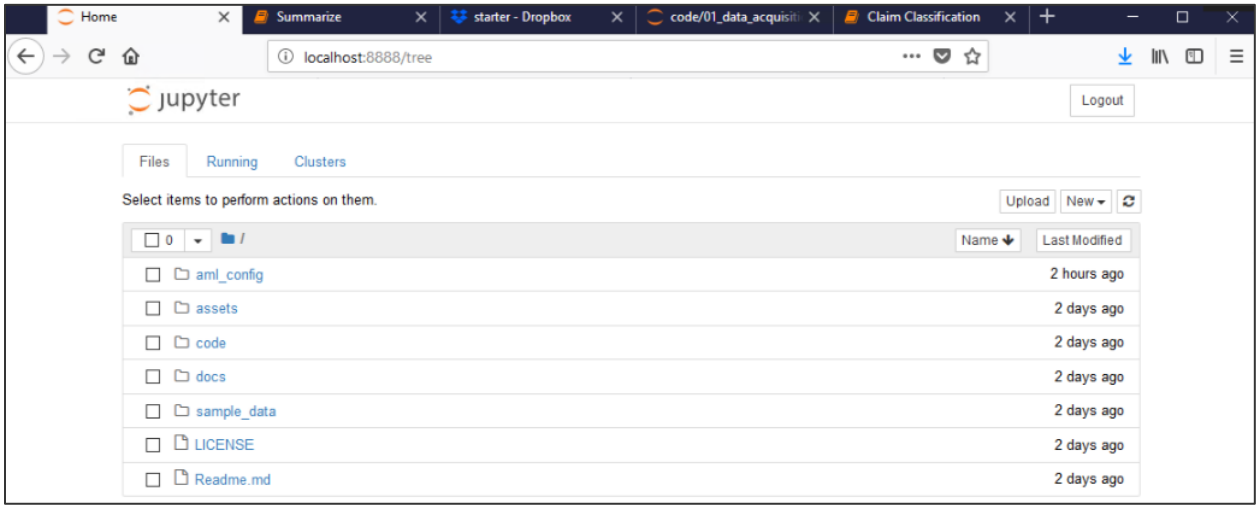


Task 3: Completing the solution

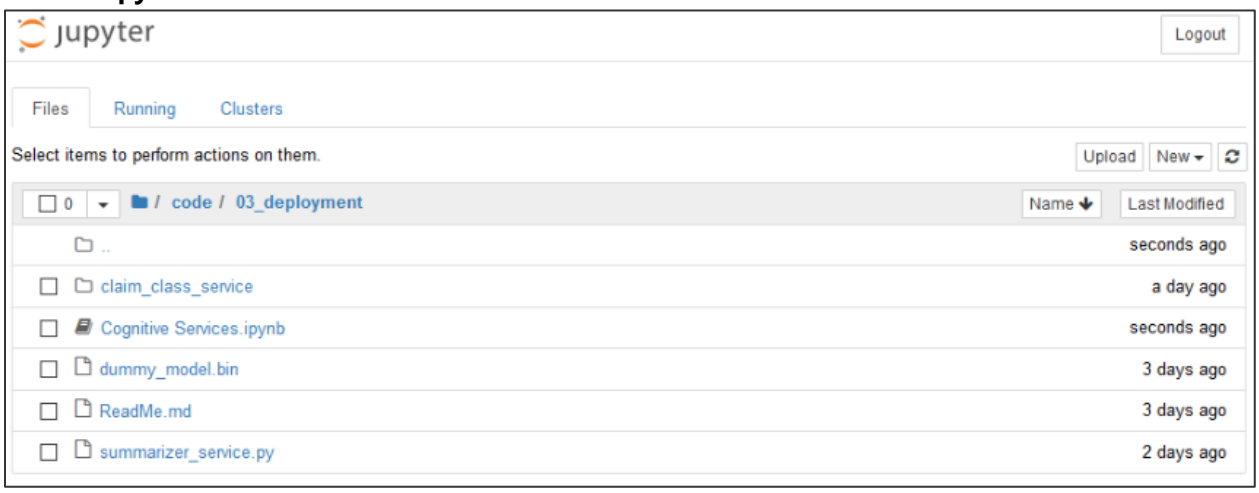
1. Return to your RDP session to the Data Science VM.
2. Download the starter files for this task from:

   http://bit.ly/2puj7oL

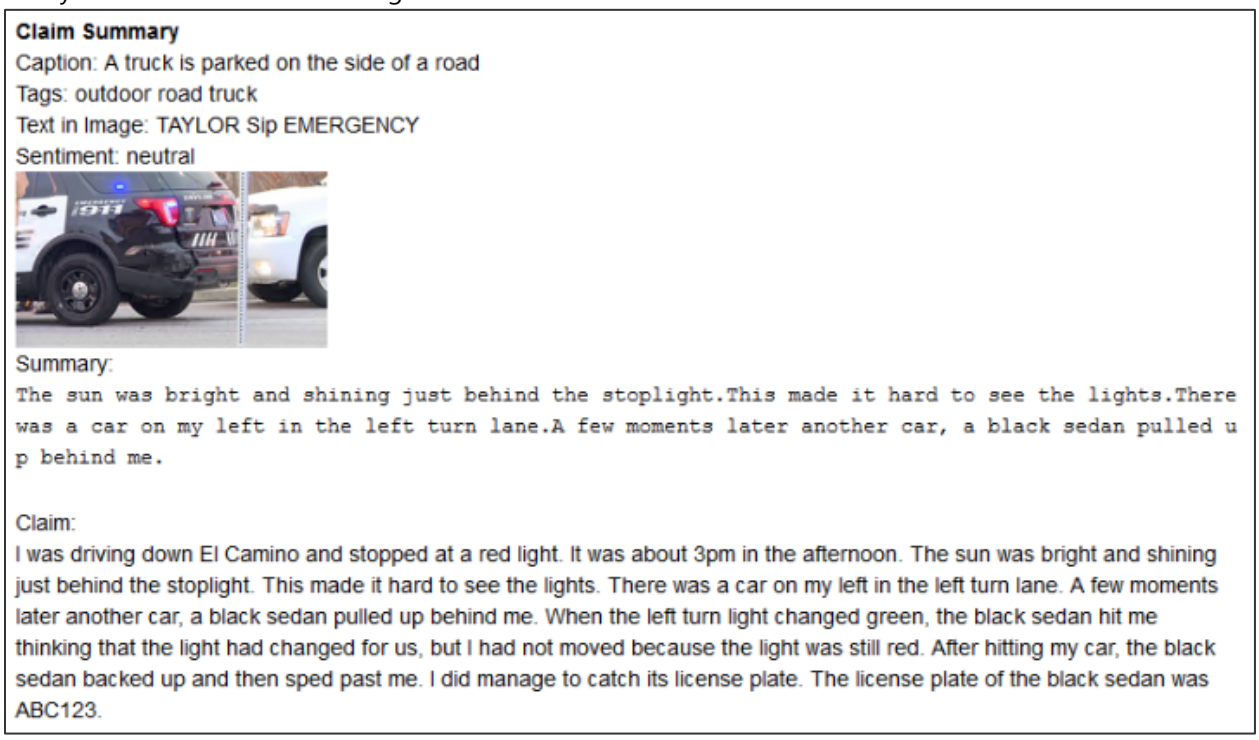
3. Extract the contents of this zip file to C:\HOL\mcw-ai-lab\code\03_deployment.
4. Return to the instance of the Jupyter Notebook home that should be open in your browser.



5. Select the code folder, **03_deployment**. You should see a folder listing like the following. Select **Cognitive Services.ipynb**.



6. Follow the steps within the notebook to complete the lab and view the result of combining Cognitive Services with your Azure Machine Learning Services.



After the hands-on lab

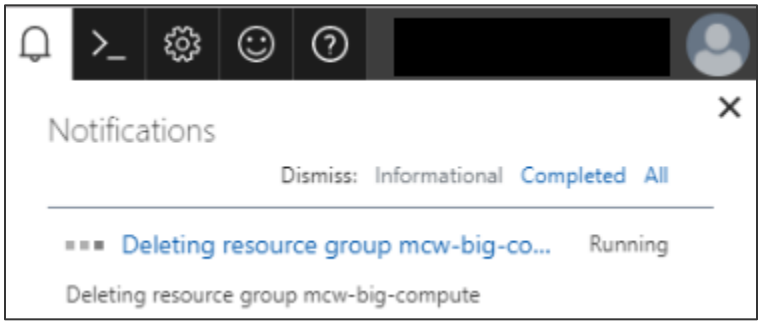
Duration: 5 minutes

To avoid unexpected charges, it is recommended you clean up all of your lab resources when you complete the lab.

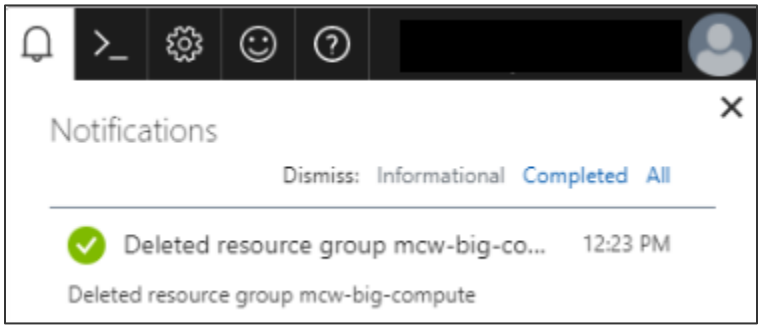
Task 1: Clean up lab resources

1. Navigate to the Azure Portal and locate the Resource Groups you created for this lab
    a. mcw-ai-lab
    b. mcwailabenv (note there are two resources groups starting with this name, so delete both)
2. Select **Delete resource group** from the command bar.

3. In the confirmation dialog that appears, enter the name of the resource group and select **Delete**.
4. Wait for the confirmation that the Resource Group has been successfully deleted. If you don't wait, and the delete fails for some reason, you may be left with resources running that were not expected. You can monitor using the Notifications dialog, accessible from the Alarm icon.



5. When the Notification indicates success, the cleanup is complete.



You should follow all steps provided *after* attending the Hands-on lab.