



Workshop: Developing Custom Object Detection Models with NVIDIA and Azure ML Studio

Using Automated ML for Vision and Triton Server or
NVIDIA Deepstream

AI at the edge

Is a growing market be it in manufacturing, energy, retail or smart cities.

In this workshop we will teach how to combine best of both worlds of the Azure and NVIDIA platforms.
The content is built up in the following setting:

First: set up. In this part we will guide you through the set up to get started.

Second: label a dataset using Azure Machine Learning Studio.

Third: Use AutoML for vision to train a labeled Dataset and develop a production model.

At the end of the workshop you will have a model suitable for NVIDIA's Triton Server or Deepstream.

Part one: Set up

The set up consists of:

- Creating a Storage Account and loading the image data
- Creating a workspace resource to get started with Azure Machine Learning
 - o Create a workspace
 - o Create a compute instance
 - o Create a datastore

Creating a Storage Account

1. Create a Storage account with Blob store container. If you are unsure how to create the storage account and upload the images use the following details and links to learn how.
2. **Please note:** As you create the storage account make sure you record the **Storage Account Name, Blob Container Name and Access Key** as these will be needed later on when we create a Datastore in AML.
 - a. **Storage Account overview (Concepts):** [Explore Azure Storage services - Learn | Microsoft Docs](#)

Storage Account an Blob Store creation:

To access Azure Storage, you'll need an Azure subscription. If you don't already have a subscription, create a [free account](#) before you begin.

All access to Azure Storage takes place through a storage account. For this quickstart, create a storage account using the [Azure portal](#), Azure PowerShell, or Azure CLI. For help creating a storage account, see [Create a storage account](#).

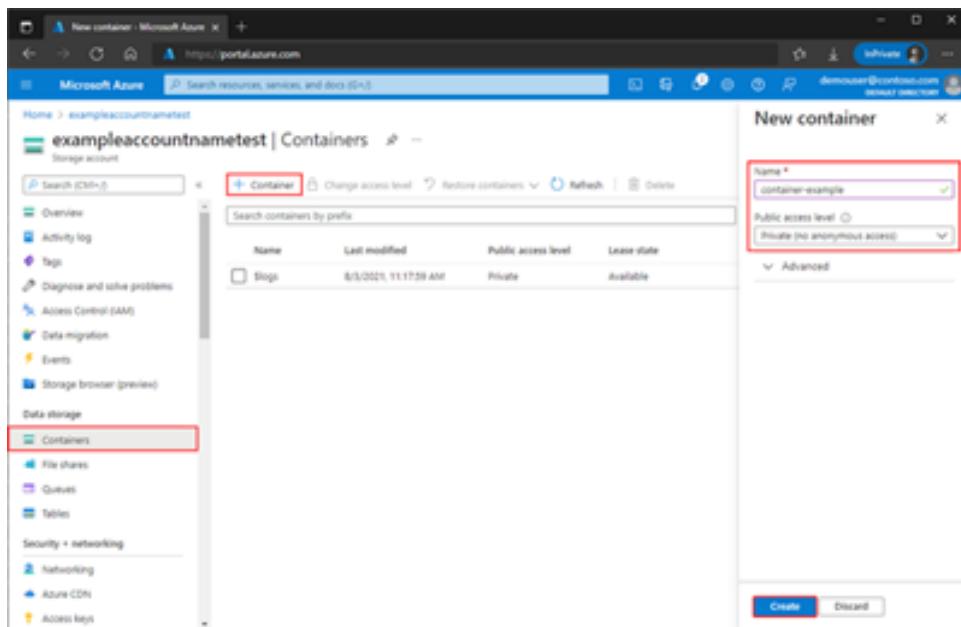
Name your storage account: < **your unique storage account name**>

!Record this name as it will be needed later.!

Create a container

To create a container in the Azure portal, follow these steps:

1. Navigate to your new storage account in the Azure portal.
2. In the left menu for the storage account, scroll to the **Data storage** section, then select Containers.
3. Select the **+ Container** button.
4. Type a name for your new container. **Name your container:** <your container name> The container name must be lowercase, must start with a letter or number, and can include only letters, numbers, and the dash (-) character. For more information about container and blob names, see [Naming and referencing containers, blobs, and metadata](#).
5. Set the level of public access to the container. The default level is **Private (no anonymous access)**.
6. Select Create to create the container.



7. Once the container is created go to the Access keys area under the Security + networking portion of the panel and select the icon.

This will bring up the following screen. Click on the "Show keys" icon (shown below as Hide keys) and copy the key to the clipboard and then record it in notepad. This key will be needed later.

Upload the provided image data. This can be found by downloading and uncompressing the zip file found in **GTC\soda_data.zip**. The image data from this zip

file can be found in the **soda_data\train_img** directory and uploaded into the Storage account container named **soda-raw**. This should be 243 images. These images are numbered **0.jpg – 244.jpg**. **Please note that at a minimum you will need at least 50 images to train an AutoML for Images model.**

For a reference on how to do this review the following link:

[Quickstart: Upload, download, and list blobs - Azure portal - Azure Storage | Microsoft Docs](#)

Create workspace resources you need to get started with Azure Machine Learning

Azure Machine Learning is a cloud service for accelerating and managing the machine learning project lifecycle. Machine learning professionals, data scientists, and engineers can use it in their day-to-day workflows: Train and deploy models, and manage MLOps.

You can create a model in Azure Machine Learning or use a model built from an open-source platform, such as Pytorch, TensorFlow, or scikit-learn. MLOps tools help you monitor, retrain, and redeploy models.

There are a number of advantages of using the Azure AML platform to create computer vision models. These include:

- An Enterprise grade platform service that facilitates the following capabilities when training and deploying CV models:
- A single platform to label, train and deploy models
- Scalability the ability to execute the code for the model training on one compute while the real training of the model happens on another compute that is completely scalable to align with the number of images and modeling tasks.
- Using the hyperdrive functionality of AutoML for images, it is possible to train hundreds of models using different algorithms and hyperparameters and then automatically have AML determine the best (champion) model automatically. This alone with the forementioned capabilities saves time and helps scale the overall model training/deployment process.

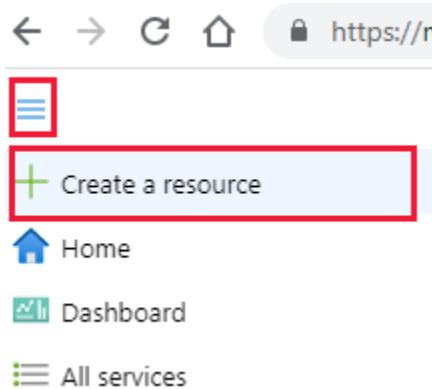
For more information on Azure Machine Learning: [Azure Machine Learning - ML as a Service | Microsoft Azure](#)

Create the workspace

If you already have a workspace, skip this section and continue to [Create a compute instance](#).

If you don't yet have a workspace, create one now:

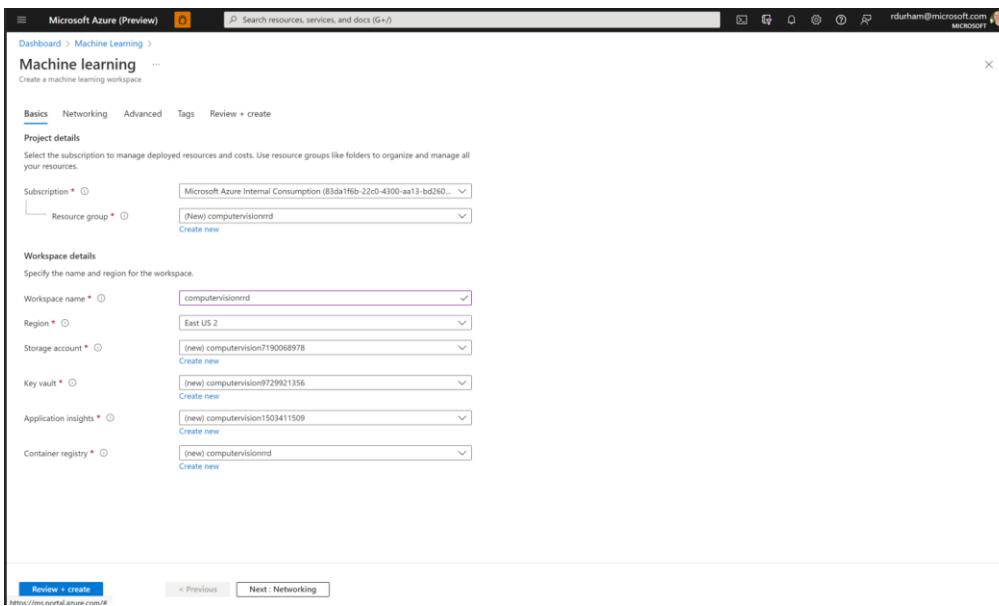
1. Sign in to the [Azure portal](#) by using the credentials for your Azure subscription.
2. In the upper-left corner of the Azure portal, select the three bars, then + **Create a resource**.



3. Use the search bar to find **Machine Learning**.
4. Select **Machine Learning**.
5. In the **Machine Learning** pane, select **Create** to begin.

A screenshot of the Azure Machine Learning service page. The top navigation bar shows 'Dashboard > Machine Learning'. The main content area features a large 'Machine Learning' card with a blue icon, a 'Create' button at the bottom left, and a '5.0 (1 Azure rating)' rating. Below this, there are tabs for 'Overview', 'Plans', 'Usage Information + Support', and 'Reviews'. A descriptive text block explains that Azure Machine Learning empowers developers and data scientists with a wide range of productive experiences for building, training, and deploying machine learning models. At the bottom, there's a 'Media' section with a thumbnail of a dashboard and a 'See All' link. Finally, there's a 'More products from Microsoft' section with cards for 'Device Update for IoT Hub', 'Front Door Standard/Premium', 'Azure VMware Solution', and 'API App'.

- Provide the following information to configure your new workspace: **Note you may need to create a new resource group, workspacename, storage account, key vault, application insights and container registry for the workspace.** An example label might be using computervision + <your initials> as the textbox entry.
- Select **Review and create** to create the workspace.



- To view the new workspace go the Azure Portal and enter "**Machine Learning**" in the search bar:

WARNING: It can take several minutes to create your workspace in the cloud. When the process is finished, a deployment success message appears.

- To view the new workspace go the Azure Portal and enter "**Machine Learning**" in the search bar:

The screenshot shows the Microsoft Azure (Preview) dashboard. At the top, there's a search bar with the text 'Machine Learning'. Below the search bar, there are several service tiles: Event Hubs, CreateVm-micros, IoT Hub, IoTSeCo, ADFTutorialDataF, yolord, Microsoft.MachineLearn, and aicamp. In the center, there's a 'Services' section with tabs for All, Services (11), Marketplace (20), Documentation (99+), Azure Active Directory (34), and Resources (0). The 'Services' tab is selected. Under 'Services', there are sections for Machine Learning Studio (classic) web service plans, Marketplace, and Documentation. A red box highlights the 'Machine learning' tile under the services section.

10. Download the **config.json** file and store it locally for use later in this tutorial (see the image below for download location)

The screenshot shows the 'Computer_Vision_Pipeline' workspace in the Azure Machine Learning studio. On the left, there's a navigation menu with sections like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings, Monitoring, Automation, Support + troubleshooting, Usage + quota, and New Support Request. The main area is titled 'Essentials' and contains information about the resource group, location, subscription, and storage. At the bottom of this blade, there's a 'Manage your machine learning lifecycle' section with a 'Launch studio' button. A red box highlights the 'Download config.json' button.

11. From the portal view of your workspace, select **Launch studio** to go to the Azure Machine Learning studio.

Create compute instance

You could install Azure Machine Learning on your own computer. But in this quick start, you'll create an online compute resource that has a development environment already installed and ready to go. You'll use this online machine, a **compute instance**, for your development environment to write and run code in Python scripts and Jupyter notebooks.

Create a **compute instance** to use this development environment for the rest of the tutorials and quickstarts.

1. If you didn't select **Go to workspace** in the previous section, sign in to [Azure Machine Learning studio](#) now, and select your workspace.
2. You can do this by typing in "**Azure Machine Learning**" on the search bar and choosing the Machine Learning icon

The screenshot shows the Microsoft Azure (Preview) dashboard. At the top, there is a search bar with the text 'Azure Machine Learning'. Below the search bar, there are tabs for 'All', 'Services (88)', 'Marketplace (3)', 'Documentation (99+)', 'Azure Active Directory (10)', and 'Resources (0)'. The 'Services' tab is selected. Under the 'Services' category, there is a section titled 'Virtual machines' which includes 'Machine learning'. This 'Machine learning' item is also highlighted with a red box. To the right of the search bar, there is a sidebar with sections for 'LearningLUIS Language understanding', 'LearningQnARRC QnA maker', 'LearningL... Language underst...', 'LearningQnARRC QnA maker', and 'Custom Domain Name'. At the bottom of the dashboard, there is a footer with the URL 'https://ms.portal.azure.com/#blade/HubsExtension/BrowseResourceBlade/resourceType/Microsoft.MachineLearningServices%2fworkspaces' and a status bar showing '12:02 PM 1/16/2022'.

3. This will give you a list of AML workspaces to choose from:

The screenshot shows the Microsoft Azure (Preview) Machine learning workspace list page. The title bar says 'Machine learning'. The main area displays a table of workspaces, each with a name, resource group, location, and subscription information. The first workspace listed is 'AloT-ML', located in 'AloT_Dev_Ops_RG' under 'Subscription == JC-IoT-Sandbox'. The table has columns for 'Name', 'Resource group', 'Location', and 'Subscription'. There are filters at the top: 'Subscription == 24 of 36 selected', 'Resource group == all', and 'Location == all'. The bottom of the page shows pagination with 'Page 1 of 1'.

4. Once you are in your AML workspace On the left side, select **Compute**.

The screenshot shows the Microsoft Azure Machine Learning Studio interface. The URL in the address bar is https://ml.azure.com/compute/list?wsid=/subscriptions/83da16b-22c0-4300-aa13-bd260eab57e5/resourcegroups/Computer.Vision.Pipeline/worksheets/Computer.Vision.Pipeline&tid=72988bf-86f1-462c-a9c8-412d5a22a50c. The page title is "Microsoft Azure Machine Learning Studio". The main content area is titled "Compute" and shows a list of "Compute instances". The table includes columns for Name, State, Applications, Size, Created on, and Assigned to. Five instances are listed: rdurham1 (Stopped, JupyterLab, Jupyter, VS Code, RStudio, Terminal, STANDARD_DS14_V2, Jan 10, 2022 10:17 AM, Rick Durham), cdb89dd80 (Stopped, JupyterLab, Jupyter, VS Code, RStudio, Terminal, STANDARD_D3_V2, Dec 15, 2021 9:49 AM, Rick Durham), Imageprocessing (Stopped, JupyterLab, Jupyter, VS Code, RStudio, Terminal, STANDARD_DS3_V2, Oct 20, 2021 10:55 AM, Rick Durham), computervisiontrain1 (Running, JupyterLab, Jupyter, VS Code, RStudio, Terminal, STANDARD_NV6, Oct 20, 2021 10:46 AM, Rick Durham), and computer-vision-pipeline (Stopped, JupyterLab, Jupyter, VS Code, RStudio, Terminal, STANDARD_DS3_V2, May 17, 2021 4:55 PM, Rick Durham). On the left sidebar, there is a section with icons for Compute, Datasets, Experiments, and Model Registry. The "Compute" icon is highlighted with a red box.

Name	State	Applications	Size	Created on	Assigned to
rdurham1	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_DS14_V2	Jan 10, 2022 10:17 AM	Rick Durham
cdb89dd80	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_D3_V2	Dec 15, 2021 9:49 AM	Rick Durham
Imageprocessing	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_DS3_V2	Oct 20, 2021 10:55 AM	Rick Durham
computervisiontrain1	Running	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_NV6	Oct 20, 2021 10:46 AM	Rick Durham
computer-vision-pipeline	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_DS3_V2	May 17, 2021 4:55 PM	Rick Durham

5. Select **+New** to create a new compute instance.
6. Supply a name like **Imageprocessing**. Please select a NVIDIA GPU instance: **Standard_NC6**. We will use this compute instance as we execute the notebook and to train our model.
7. Select **Create**.

In about two minutes, you'll see the **State** of the compute instance change from **Creating to Running**. It's now ready to go. Now that we have an AML workspace created and a compute instance we are ready to setup our AML Data Store and AML Dataset. This Data Store and Data Set will point back to the image data we uploaded in previous steps.

Create a new datastore in a few steps with the Azure Machine Learning studio

Datastores securely connect to your storage service on Azure without putting your authentication credentials and the integrity of your original data source at risk. They store connection information, like your subscription ID and token authorization in your Key Vault that's associated with the workspace, so you can securely access your storage without having to hard code them in your scripts. You can create datastores that connect to these Azure storage solutions.

We are now going to create a new datastore in Azure Machine Learning Studio.

Important

8. Sign in to [Azure Machine Learning studio](#).
9. Select **Datastores** on the left pane under **Manage**.

Name	Type	Storage name	Created on	Created by
workspaceartifactstore	Azure Blob Storage	computervision5249774111	Sep 30, 2021 4:30 PM	--
computervisionwimagesraw	Azure Blob Storage	computervisionwimagesraw	Aug 16, 2021 12:55 PM	Rick Durham
azureml_globaldatasets	Azure Blob Storage	mmstorageeastus	Jun 1, 2021 9:24 AM	Service Principal
workspaceblobstore (Default)	Azure Blob Storage	computervision5249774111	May 17, 2021 4:54 PM	Service Principal
workspacefilestore	Azure file share	computervision5249774111	May 17, 2021 4:54 PM	Service Principal

10. Select **+ New datastore**.

11. Complete the form to create and register a new datastore. The form intelligently updates itself based on your selections for Azure storage type and authentication type. Name the Datastore using a name similar to the one shown below. You will recall we recorded the other form entries as *Storage Account Name, Blob Container Name and Access Key at the beginning of this workshop*. You will use them to verify the correct entries for the *Storage Account Name, Blob Container Name* dropdowns. You will enter the Access key into the text box.

12. See the [storage access and permissions section](#) to understand where to find the authentication credentials you need to populate this form.

The following example demonstrates what the form looks like when you create an **Azure blob datastore**:

New datastore

Datastore name *

Datastore type *

Account selection method

From Azure subscription Enter manually

Subscription ID *

Storage account *

Blob container *

Save credentials with the datastore for data access [\(?\)](#)

Authentication type * [\(?\)](#)

Account key

Account key * [\(?\)](#)
.....

Use workspace managed identity for data preview and profiling in Azure Machine Learning studio [\(?\)](#)

[\(?\) Note: Azure Machine Learning service does not validate whether the underlying data source ...](#)

Part 2: Create a Labeled Dataset using the Azure Machine Learning Data Labeling Tools

Azure Machine Learning data labeling is a central place to create, manage, and monitor data labeling projects:

Coordinate data, labels, and team members to efficiently manage labeling tasks.

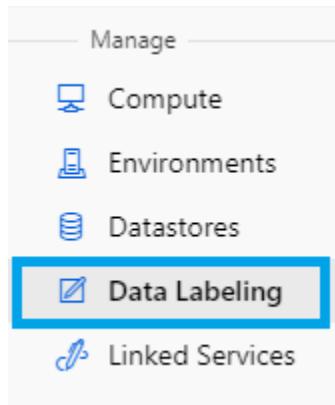
Tracks progress and maintains the queue of incomplete labeling tasks.

Start and stop the project and control the labeling progress.

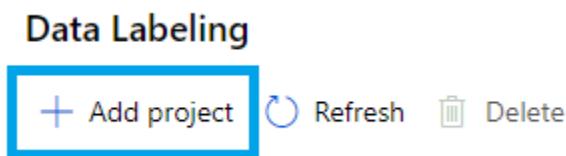
Review the labeled data and export labeled as an Azure Machine Learning dataset.

Here you will use Azure ML to create a dataset of labeled images using images collected on the edge, retrieved from your attached datastore.

13. Navigate to your Azure Machine Learning workspace and click *Data Labeling*.



14. From the top menu, click *+ Add project*.



15. Under the *Project details* section, give your project a name that is specific to the particular detection task at hand and select *Object Identification (Bounding Box)* from the menu below, then click *Next*.

The screenshot shows the 'Project details' and 'Labeling task type' sections of the Azure ML Data Labeling interface.

Project details:

- A message box indicates: "New feature: To make labeling faster, we've added a new feature to train an ML model while you label. This feature currently supports image or text classification."
- Project name ***: A text input field contains the value "soda".

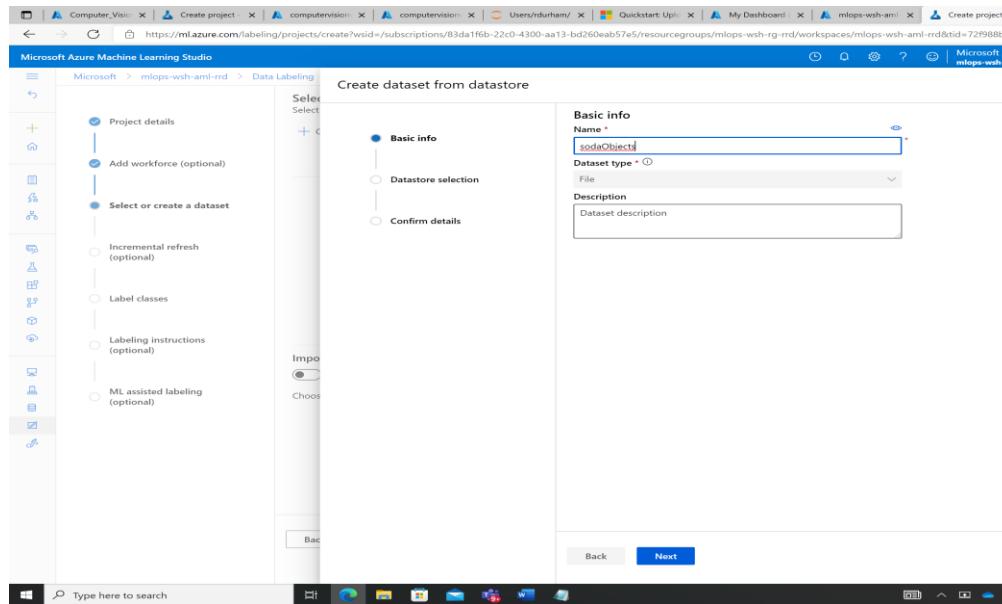
Media type *: A radio button group shows "Image" selected (indicated by a blue dot) and "Text" unselected.

Labeling task type *: A grid of four options:

- Image Classification Multi-class
- Image Classification Multi-label
- Object Identification (Bounding Box)** (selected, indicated by a blue border and a blue dot)
- Instance Segmentation (Polygon)

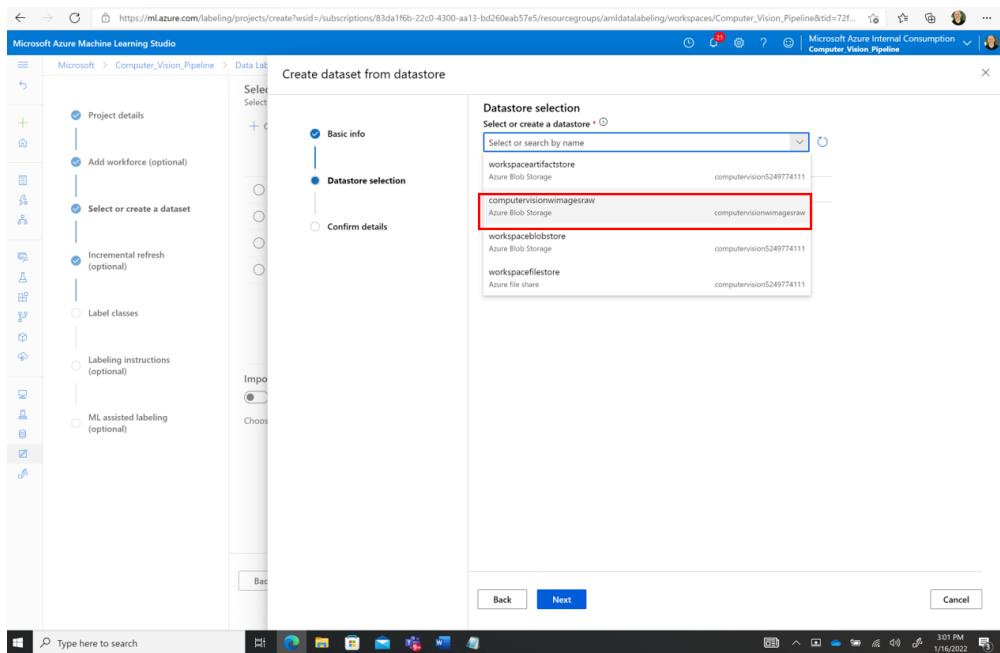
Below the grid, a note says: "Assign a class and a bounding box to each object within an image". A "Learn more" link is also present.

When prompted to "select or create a dataset choose" + **Create dataset** and select the **From datastore** option.



16. Give your new **dataset** the unique name e.g. **sodaObjects** that reflects the images captured in support of the detection task and click *Next*.

17. Under datastore selection choose the **datastore** name that you added previously which contains images captured. Here, you can also provide a wildcarded path if you wish to pull only images from specified partitions. If you wish to pull all images from the container, enter / as the path and click *Next*. Confirm details about your new dataset and click *Create*.



You will be prompted to Enable incremental refresh at regular intervals. This will automatically add newly captured images to your data labeling project.

You should **not** use ML-assisted labeling for this lab.

ML assisted labeling (optional)

Message: ML assisted labeling uses Azure Machine Learning compute for model training and inferencing. By enabling ML assisted labeling, you acknowledge that you will incur the related Azu... ▾

Accelerate your labeling project by enabling ML assisted labeling, which trains a model to pre-label data for your labelers to review.

Enable ML assisted labeling

You can enable ML assisted labeling on the project details page any time after project creation.

On the next panel, add label classes for all objects or defects you wish to detect - include positive and negative classes here.

Label classes

Enter the list of label classes

Label name(s)

coke	
diet_coke	
sprite	

+ Add label

You can **optionally use ML-assisted labeling** which will accelerate your data labeling process, particularly as more data is captured. This option is strongly recommended. **For this course we won't be using this option.**

ML assisted labeling (optional)

ⓘ Message: ML assisted labeling uses Azure Machine Learning compute for model training and inferencing. By enabling ML assisted labeling, you acknowledge that you will incur the related Azu... ▾

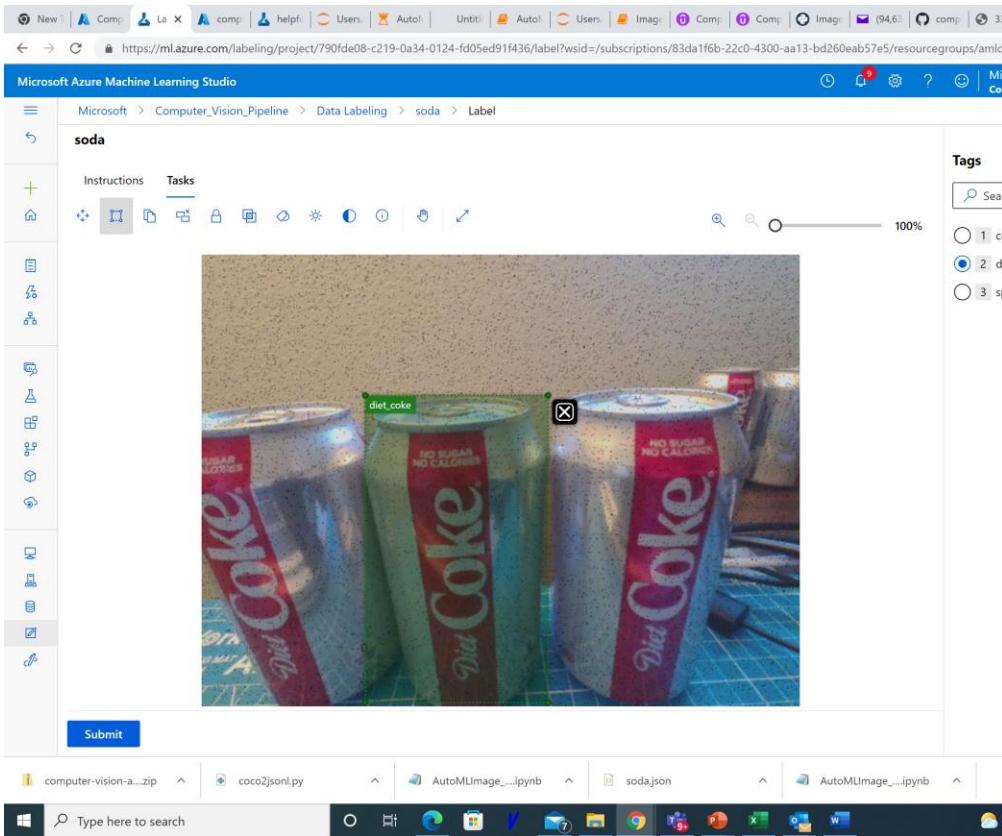
Accelerate your labeling project by enabling ML assisted labeling, which trains a model to pre-label data for your labelers to review.

Enable ML assisted labeling

You can enable ML assisted labeling on the project details page any time after project creation.

18. After providing all requested information click *Create project* and wait for the project to initialize.

19. Click your newly created project and then select the **Label data** button. This will open a labeling utility which will allow you to draw bounding boxes and tag objects/defects present in your images. There are multiple keyboard shortcuts available which can be reviewed under the *Shortcut Keys* panel. As you label images hit the submit button and this will save the labelled image.



20.The next step after labeling all of your images is to export an annotation file.

After labeling a large number of images, navigate to the data labeling project homepage and click *Export* then select *Azure ML Dataset*. This will export your image dataset as an AutoML-compatible Azure ML dataset named according to the format 'NAME_DATE_TIME'.

This is done as follows:

1. Select Export from the Data Labeling Menu
2. Choose Azure ML dataset

3. Click on the View dataset link

This will open a view to the newly created dataset. Copy the name of the AML ML dataset in this case it is named "**SodaDemo_20220126_144158**"

The screenshot shows the Microsoft Azure Machine Learning Studio interface. In the top navigation bar, the path is 'Microsoft > Computer_Vision_Pipeline > Datasets > SodaDemo_20220126.144158'. Below the path, there's a dropdown menu showing 'Version 1 (latest)'. The main content area has tabs for 'Details', 'Consume', 'Explore', and 'Models'. A red box highlights the 'Tags' section on the right, which contains the following data:

labelingCreatedBy
Data Labeling
labelingProjectType
Object Identification (Bounding Box)
SourceDatastoreName
computervision\wimages\raw
SourceRelativePath
/
labelingLabelName
[“diet_coke”, “sprite”, “coke”]

Below the tags, there are sections for 'Description' and 'Data source'.

Part 3: Use Automated ML to training a computer vision model

Automated ML for images (preview) adds support for computer vision tasks, which allows you to easily generate models trained on image data for scenarios like image classification and object detection.

With this capability you can:

- Seamlessly integrate with the Azure Machine Learning data labeling capability
- Use labeled data for generating image models
- Optimize model performance by specifying the model algorithm and tuning the hyperparameters.
- Download or deploy the resulting model as a web service in Azure Machine Learning.

- Operationalize at scale, leveraging Azure Machine Learning MLOps and ML Pipelines capabilities.
- Authoring AutoML models for vision tasks is supported via the Azure ML Python SDK. The resulting experimentation runs, models, and outputs can be accessed from the Azure Machine Learning studio UI.

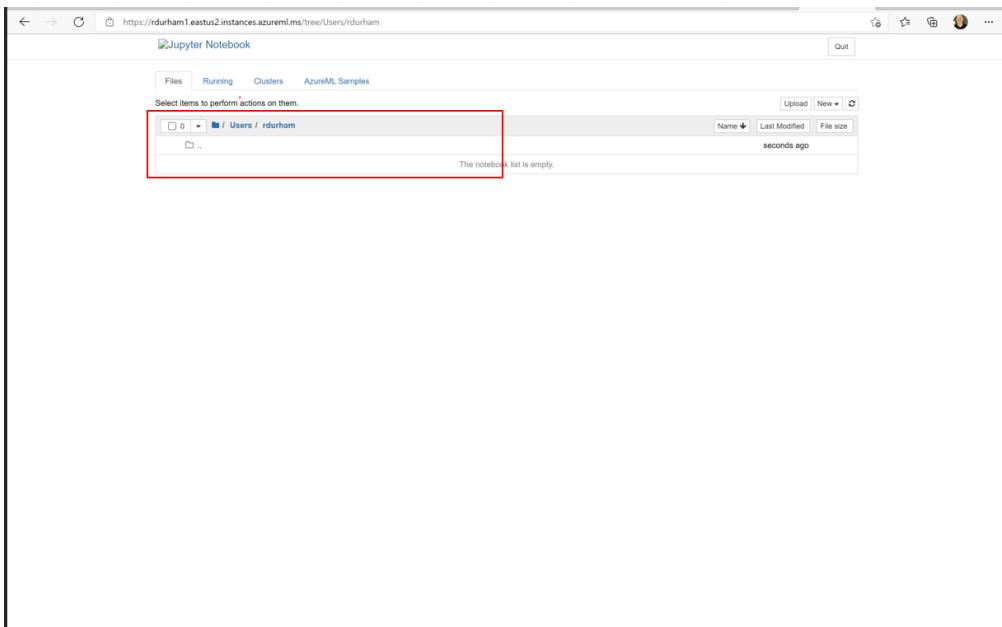
For more information on Automated ML: [What is automated ML? AutoML - Azure Machine Learning | Microsoft Docs](#)

Open the AML Workspace you created earlier:

4. **Go to workspace** in the previous section, sign in to [Azure Machine Learning studio](#) now, and select your workspace. Select the compute icon and click on the **Jupyter** link:

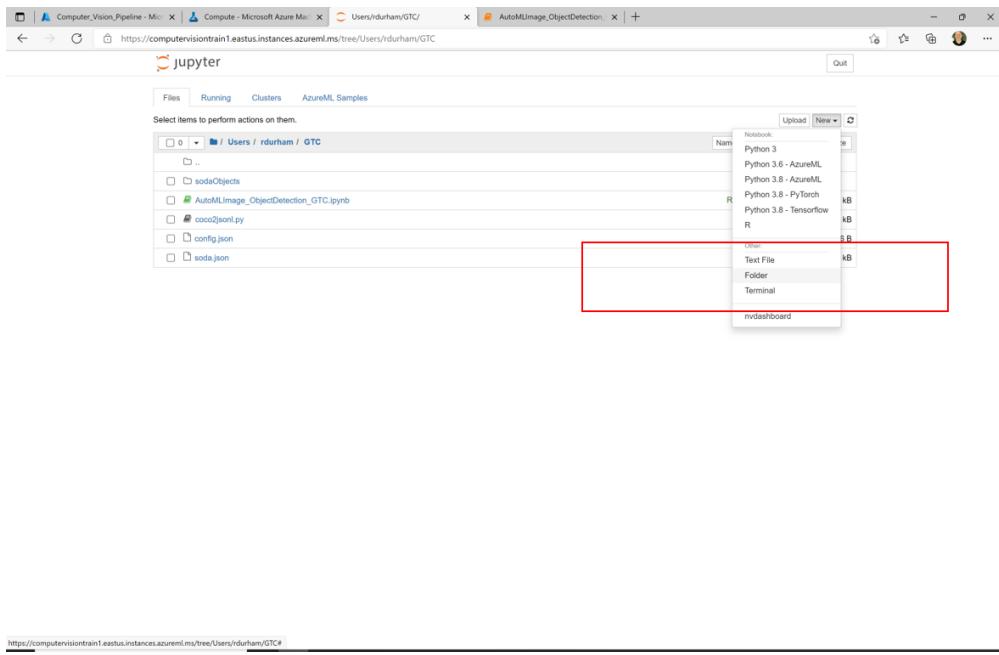
Name	State	Applications	Size	Created on	Assigned to
rdurham1	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_DS14_V2	Jan 10, 2022 10:17 AM	Rick Durham
cid866dd80	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_D3_V2	Dec 15, 2021 9:49 AM	Rick Durham
Imageprocessing	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_DS3_V2	Oct 20, 2021 10:55 AM	Rick Durham
computervisiontrain1	Running	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_NV6	Oct 20, 2021 10:46 AM	Rick Durham
computer-vision-pipeline	Stopped	JupyterLab Jupyter VS Code RStudio Terminal	STANDARD_DS3_V2	May 17, 2021 4:55 PM	Rick Durham

Click on the users directory and select the your user name directory e.g.:

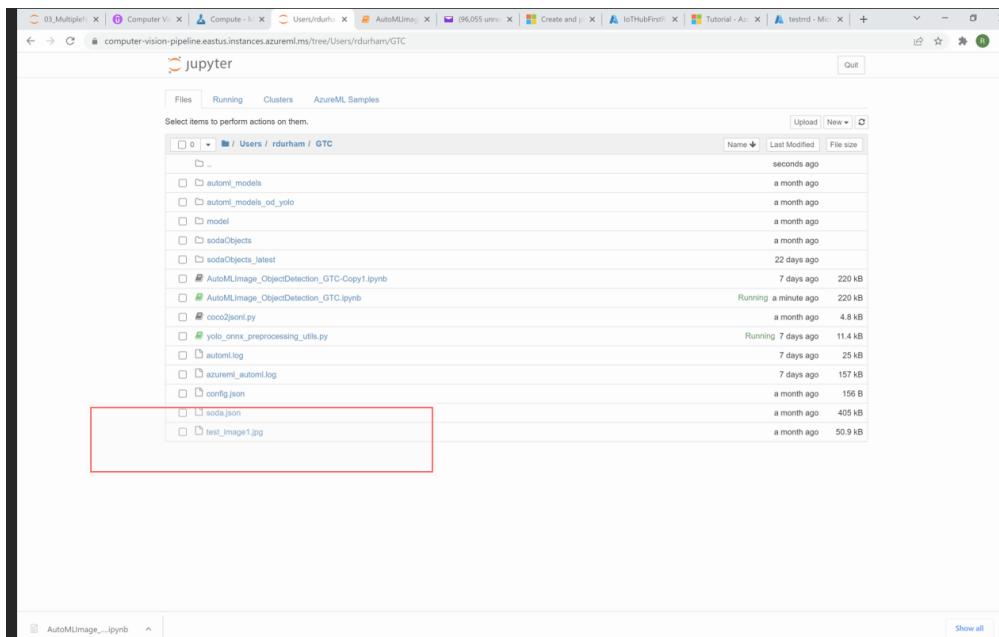


Use the upload button (top right) to upload the **auto-ml-image-object-detection_latest-V2.ipynb** Jupyter notebook from the local drive where you stored in the **GTC\ directory**

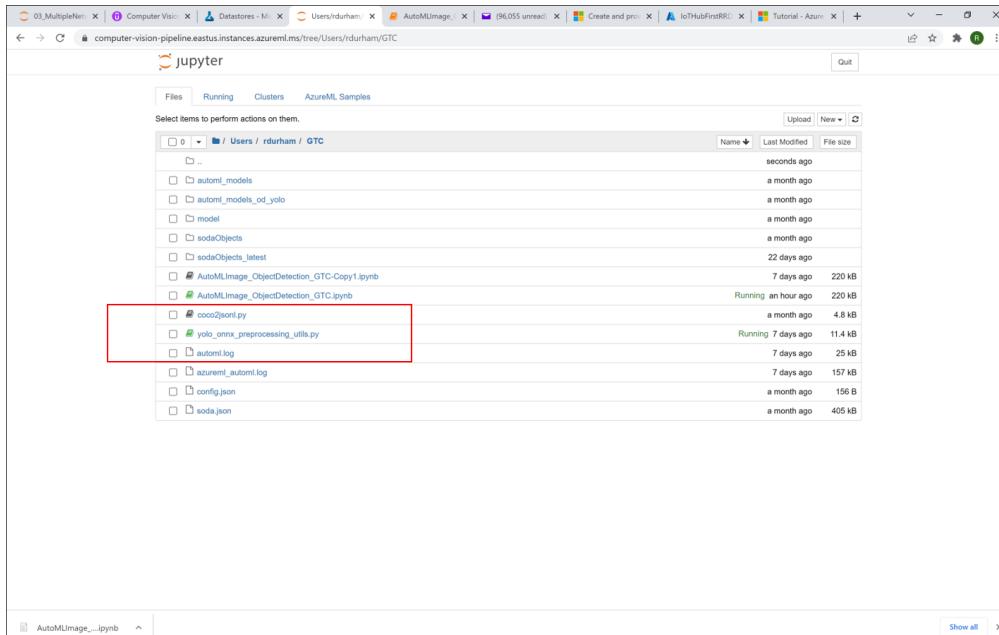
Upload the **config.json** Using the new menu dropdown create a new folder called **sodaObjects**. Once you finished your environment should look like the following:



Upload the file **test_image1.jpg** into the same folder where the notebook resides.

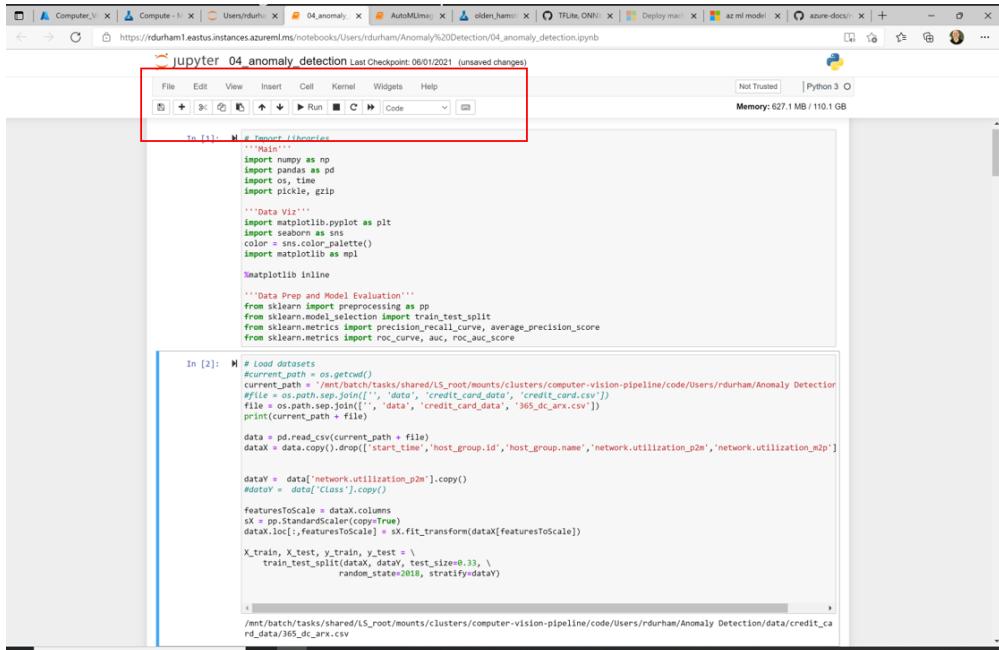


Upload the **yolo_onnx_preprocessing_utils.py** file into the same directory where the notebook is stored:



Execute the AutoMLImage_ObjectDetection_GTC notebook:

5. Click on the **auto-ml-image-object-detection_latest-V2.ipynb** notebook in the Jupyter environment. You will execute the following cells in order. In order to execute a cell click in the code cell and hold down on the shift key and press enter or execute the run command using the arrow: Example:



Execute the following Jupyter Notebook cells in order:

Environment Setup

1. Please note after you execute the cell with the code "**pip install torchvision==0.9.1**" you need to restart the Kernel by going to the menu item Kernel and clicking on the "**Restart**" menu item. After this you can execute the next cell pip freeze which will show you all of the python libraries installed.

The screenshot shows a Jupyter Notebook interface with the title "jupyter AutoMLImage_ObjectDetection_GTC Last Checkpoint: 4 minutes ago (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The Kernel dropdown is open, showing options: Interrupt, Restart, and Clear Output. The "Restart" option is highlighted with a red box. Below the menu is a toolbar with icons for New, Open, Save, Run, Insert, Cell, Kernel, Widgets, and Help. The main area contains two code cells:

In [122]:

```
import azureml.core
print("This notebook was created using version 1.35.0 of the Azure ML SDK.")
print("You are currently using version", azureml.core.VERSION, "of the Azure ML SDK.")
assert (
    azureml.core.VERSION >= "1.35",
), "Please upgrade the Azure ML SDK by running '!pip install --upgrade azureml-sdk' then restart the kernel."
This notebook was created using version 1.35.0 of the Azure ML SDK.
You are currently using version 1.35.0 of the Azure ML SDK.
```

In [123]:

```
!pip install torchvision==0.9.1
```

The output of the first cell shows the notebook was created with version 1.35.0 and is currently using version 1.35.0. It also includes an assertion to check for the minimum required version and a note to upgrade if it's not met. The second cell is a command to install the torchvision library with version 0.9.1.

2. Workspace Setup

3. Compute Target Setup

4. Experiment Setup

5. Dataset with input Training Data (****Please note you will need to replace the name in this step the AML dataset name you recorded earlier** in our example this was called **SodaDemo_20220126_144158**)

6. View Training Set (you should see a dataset as shown below)

```

jupyter AutoMLImage_ObjectDetection_GTC Last Checkpoint: 15 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3.6 - AzureML O
Memory: 627.6 MB / 110.1 GB
# validation_dataset = _labelledDatasetFactory.from_json_lines(
#     task=LabelledDatasetTask.OBJECT_DETECTION, path=ds.path['appleObjects/validation_annotations.json'])
# validation_dataset = validation_dataset.register(workspace=ws, name=validation_dataset_name)

#print("Training dataset name: " + training_dataset.name)
#print("Validation dataset name: " + validation_dataset.name)

Found the training dataset sodaObjects

Validation dataset is optional. If no validation dataset is specified, by default 20% of your training data will be used for validation. You can control the percentage using the split_ratio argument - please refer to the documentation for more details.

This is what the training dataset looks like

View Training Set
In [9]: M import pandas as pd
training_dataset.to_pandas_dataframe()
Out[9]:
   image_url           image_details      label
0  StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'sprite', 'topX': 0.153960129310344...
1  StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'sprite', 'topX': 0.18536407197044...
2  StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'sprite', 'topX': 0.221994119458128...
3  StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'coke', 'topX': 0.3115655972906404...
4  StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'dft_coke', 'topX': 0.317753232758...
...
160 StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'sprite', 'topX': 0.033886237684729...
161 StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'sprite', 'topX': 0.205068050394581...
162 StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'coke', 'topX': 0.2278517642610837...
163 StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'dft_coke', 'topX': 0.200142399532...
164 StreamInfo(AmIDatastore/sodaObjects/images0...  {'format': 'jpg', 'width': 816.0, 'height': 61...  ['label': 'dft_coke', 'topX': 0.32669968590...
165 rows × 3 columns

```

Configuring your AutoML run for image tasks

Please note that we are using a Compute Instance with a GPU as part of this workshop. As this is an example we didn't create a separate computer cluster.

A compute cluster is an additional compute target that can be used to train larger models with more data. A compute cluster can upscale or downscale the GPU target to align with the workload that could be as few as a few hundred to several thousand images.

Additional you only pay when the nodes are up and not when they are scaled down. For a compute instance you pay per hour even when there is no training happening but is still running.

For more information about compute inside Azure Machine learning:

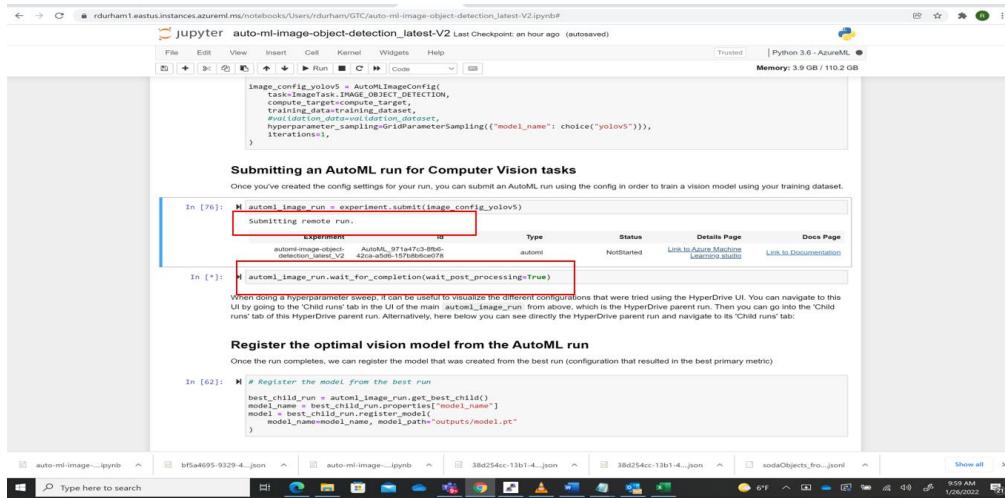
Compute instance: [What is an Azure Machine Learning compute instance? - Azure Machine Learning | Microsoft Docs](#)

Compute cluster: [Create compute clusters - Azure Machine Learning | Microsoft Docs](#)

Execute the following Jupyter Notebook cells in order:

7. Using default hyperparameter values for the specified algorithm.

- 8. Submitting an AutoML run for Computer Image tasks.** This will start the AML training job and provide a link to the AML workspace to monitor the job.
 Additionally, execute the next cell which will stay in a holding state until the AML job finishes



```

image_config_yolov5 = AutoMLImageConfig(
    task='image-object-detection',
    compute_target='compute_target',
    training_data='training_dataset',
    validation_data='validation_dataset',
    hyperparameter_sampling=GridParameterSampling({'model_name': choice('yolov5')}),
    iterations=3,
)

```

Submitting an AutoML run for Computer Vision tasks

Once you've created the config settings for your run, you can submit an AutoML run using the config in order to train a vision model using your training dataset.

In [76]:	# automl_image_run = experiment.submit(image_config_yolov5)	Experiment	Type	Status	Details Page	Docs Page
	Submitting remote run.	automl	automl	NotStarted	Link to Azure Machine Learning studio	Link to Documentation

In [*]: # automl_image_run.wait_for_completion(wait_post_processing=True)

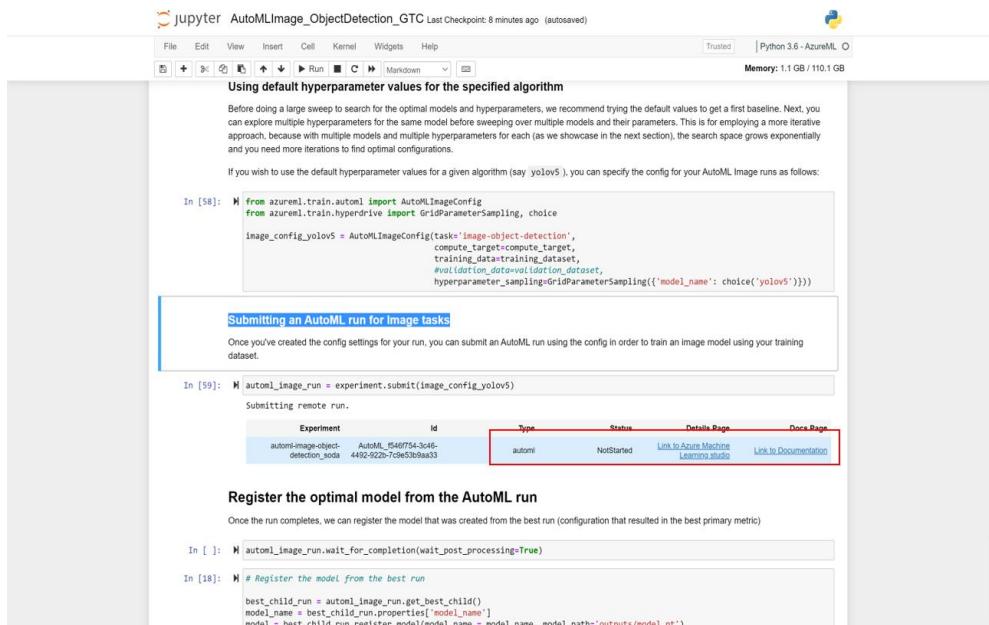
When doing a hyperparameter sweep, it can be useful to visualize the different configurations that were tried using the HyperDrive UI. You can navigate to this UI by going to the 'Child run' tab in the UI of the main 'automl_image_run' from above, which is the HyperDrive parent run. Then you can go into the 'Child' run tab of this HyperDrive parent run. Alternatively, here below you can see directly the HyperDrive parent run and navigate to its 'Child' runs tab:

Register the optimal vision model from the AutoML run

Once the run completes, we can register the model that was created from the best run (configuration that resulted in the best primary metric)

In [62]:	# Register the model from the best run	best_child_run = automl_image_run.get_best_child()	model_name = best_child_run.properties['model_name']	model = best_child_run.register_model(model_name, model_path='outputs/model.pt')		
----------	--	--	--	--	--	--

If you want to view the model training in AML click on the Details page shown below



Using default hyperparameter values for the specified algorithm

Before doing a large sweep to find the optimal models and hyperparameters, we recommend trying the default values to get a first baseline. Next, you can explore multiple hyperparameters for the same model before sweeping over multiple models and their parameters. This is for employing a more iterative approach, because with multiple models and multiple hyperparameters for each (as we showcase in the next section), the search space grows exponentially and you need more iterations to find optimal configurations.

If you wish to use the default hyperparameter values for a given algorithm (say 'yolov5'), you can specify the config for your AutoML Image runs as follows:

In [58]:	# from azureml.train.automl import AutoMLImageConfig from azureml.train.hyperdrive import GridParameterSampling, choice	image_config_yolov5 = AutoMLImageConfig(task='image-object-detection', compute_target='compute_target', training_data='training_dataset', validation_data='validation_dataset', hyperparameter_sampling=GridParameterSampling({'model_name': choice('yolov5')}))				
----------	--	--	--	--	--	--

Submitting an AutoML run for Image tasks

Once you've created the config settings for your run, you can submit an AutoML run using the config in order to train an image model using your training dataset.

In [59]:	# automl_image_run = experiment.submit(image_config_yolov5)	Experiment	Id	Type	Status	Details Page	Docs Page
	Submitting remote run.	automl-image-object-detection_000	4492-922b-Tce8d3e0a33	automl	NotStarted	Link to Azure Machine Learning studio	Link to Documentation

Register the optimal model from the AutoML run

Once the run completes, we can register the model that was created from the best run (configuration that resulted in the best primary metric)

In []:	# automl_image_run.wait_for_completion(wait_post_processing=True)					
In [18]:	# Register the model from the best run	best_child_run = automl_image_run.get_best_child()	model_name = best_child_run.properties['model_name']	model = best_child_run.register_model(model_name, model_path='outputs/model.pt')		

It should be in a “Running” state.

Microsoft Azure Machine Learning Studio

brave_lizard_6n7b308h

Status: Running

Created: Jan 26, 2022 9:53 AM

Started: Jan 26, 2022 9:54 AM

Compute target: gpu-cluster-nc6

Run ID: AutoML_971a47c3-8fb6-42ca-a5d6-157b8b6ce078

Script name: ...

Input datasets: Input name: training_data. Dataset: SodaDemo_20220126_131830; Version 1

Output datasets: None

Arguments: None

Primary metric: Mean average precision

Experiment name: automl-image-object-detection_latest_V2

Best model summary: No data

Run summary: Task type: Object identification (Bounding Box); Featureization: Auto; Primary metric: Mean average precision; Experiment name: automl-image-object-detection_latest_V2

Once the model training job finishes executing you should see the following screen. By clicking on the **Algorithm name** in the **Best model summary** we can see the results of the model training:

Microsoft Azure Machine Learning Studio

brave_lizard_6n7b308h

Status: Completed

Created: Jan 26, 2022 9:53 AM

Started: Jan 26, 2022 9:54 AM

Duration: 13m 18.97s

Compute duration: 13m 18.96s

Compute target: gpu-cluster-nc6

Run ID: AutoML_971a47c3-8fb6-42ca-a5d6-157b8b6ce078

Tags: is_gpu : True

Script name: ...

Input datasets: Input name: training_data. Dataset: SodaDemo_20220126_131830; Version 1

Output datasets: None

Arguments: None

See all properties: Raw JSON

Best model summary: Algorithm name: yolov5

Primary metric: 0.98891 View all other metrics

Sampling: 100.00 %

Registered models: No registration yet

Deploy status: No deployment yet

Run summary: Task type: Object identification (Bounding Box); Featureization: Auto; Primary metric: Mean average precision; Experiment name: automl-image-object-detection_latest_V2

Clicking on **Outputs + Logs** and opening the training_artifacts folder will show you the model outputs in terms of the model and the label files:

Please note we see the *label file (labels.json)* the *yolo5 weights file (model.pt)* and the *ONNX model (model.onnx)* in this folder

```

1 Final settings (CLI free):
2   0 device: device(type='cuda', index=0), 'primary_metric': 'mean_average_precision', 'number_of_epochs': 30, 'training_batch_size': 16,
3   1 'enable_run_restructure', 'giou', 'label_column_name', 'arguments', 'model', 'run_source', 'is_giou', 'cls_pw', 'data_folder', 'enable
4   2 Available vCPUs: 12
5   3 GPU info:b'Tesla K80': Mem (MB): mem_info_total:11441.19, mem_info_free:11438.19, mem_info_used:3.00, gpu_utilization:0%, avg_mem_info
6   4 Disk Usage ('/'): total: '684365.49MB', 'used': '14490.69MB', 'free': '63942.89MB', 'percent': 2.2)
7   5 Disk Usage ('/dev/shm'): total: '16384.00MB', 'used': '0.00MB', 'free': '16384.00MB', 'percent': 0.0)
8   6 Virtual Memory = (total: '112785.34MB', 'available': '109540.15MB', 'percent': 2.0, 'used': '2141.68MB', 'free': '94824.30MB', 'activ
9   7 Tensorboard package is not installed, no log will be created.
10  8 GPU memory validated to be above the minimum threshold of 11400.0 MB.
11  9 Stopping download of files in 5.48 seconds
12  10 Downloading image file [yolov5.3.onnx] to /root/.cache/torch/hub
13  11 [train dataset_id: f455480a-8afe-4983-b1bc-ab3d7979b1a6, validation dataset_id: None]
14  12 [# train images: 132, # validation images: 33, # labels: 3, image size: 640]
15  13 Using 8 num_workers.
16  14 Using 8 num_workers.
17  15 [downloading yolov5.3.onnx with ncc3]
18  16 [downloading "https://aka.ms/cvmln/resources/data/models-vision-pretrained/yolov5.3.onnx-a84eea56.pth" to /root/.cache/torch/hub
19  17 [model: yolov5 (medium), # layers: 263, # param: 21493896]
20  18 [grad_accumulation_step: 1]
21  19 [stder]
22  20 [stder]
23  21 [stder]
24  22 [stder]
25  23 [stder]
26  24 [stder]
27  25 [stder]
28  26 [stder]
29  27 [stder]
30  28 [stder]
31  29 [stder]
32  30 [stder]
33  31 [stder]
34  32 [stder]
35  33 [stder]
36  34 [stder]
37  35 [stder]
38  36 [stder]
39  37 [stder]
40  38 [stder]
41  39 [stder]
42  40 [stder]
43  41 [stder]
44  42 [stder]
45  43 [stder]
46  44 [stder]
47  45 [stder]
48  46 [stder]
49  47 [stder]
50  48 [stder]
51  49 [stder]
52  50 [stder]
53  51 [stder]
54  52 [stder]
55  53 [stder]
56  54 [stder]
57  55 [stder]
58  56 [stder]
59  57 [stder]
60  58 [stder]
61  59 [stder]
62  60 [stder]
63  61 [stder]
64  62 [stder]
65  63 [stder]
66  64 [stder]
67  65 [stder]
68  66 [stder]
69  67 [stder]
70  68 [stder]
71  69 [stder]
72  70 [stder]
73  71 [stder]
74  72 [stder]
75  73 [stder]
76  74 [stder]
77  75 [stder]
78  76 [stder]
79  77 [stder]
80  78 [stder]
81  79 [stder]
82  80 [stder]
83  81 [stder]
84  82 [stder]
85  83 [stder]
86  84 [stder]
87  85 [stder]
88  86 [stder]
89  87 [stder]
90  88 [stder]
91  89 [stder]
92  90 [stder]
93  91 [stder]
94  92 [stder]
95  93 [stder]
96  94 [stder]
97  95 [stder]
98  96 [stder]
99  97 [stder]
100 98 [stder]
101 99 [stder]
102 100 [stder]
103 101 [stder]
104 102 [stder]
105 103 [stder]
106 104 [stder]
107 105 [stder]
108 106 [stder]
109 107 [stder]
110 108 [stder]
111 109 [stder]
112 110 [stder]
113 111 [stder]
114 112 [stder]
115 113 [stder]
116 114 [stder]
117 115 [stder]
118 116 [stder]
119 117 [stder]
120 118 [stder]
121 119 [stder]
122 120 [stder]
123 121 [stder]
124 122 [stder]
125 123 [stder]
126 124 [stder]
127 125 [stder]
128 126 [stder]
129 127 [stder]
130 128 [stder]
131 129 [stder]
132 130 [stder]
133 131 [stder]
134 132 [stder]
135 133 [stder]
136 134 [stder]
137 135 [stder]
138 136 [stder]
139 137 [stder]
140 138 [stder]
141 139 [stder]
142 140 [stder]
143 141 [stder]
144 142 [stder]
145 143 [stder]
146 144 [stder]
147 145 [stder]
148 146 [stder]
149 147 [stder]
150 148 [stder]
151 149 [stder]
152 150 [stder]
153 151 [stder]
154 152 [stder]
155 153 [stder]
156 154 [stder]
157 155 [stder]
158 156 [stder]
159 157 [stder]
160 158 [stder]
161 159 [stder]
162 160 [stder]
163 161 [stder]
164 162 [stder]
165 163 [stder]
166 164 [stder]
167 165 [stder]
168 166 [stder]
169 167 [stder]
170 168 [stder]
171 169 [stder]
172 170 [stder]
173 171 [stder]
174 172 [stder]
175 173 [stder]
176 174 [stder]
177 175 [stder]
178 176 [stder]
179 177 [stder]
180 178 [stder]
181 179 [stder]
182 180 [stder]
183 181 [stder]
184 182 [stder]
185 183 [stder]
186 184 [stder]
187 185 [stder]
188 186 [stder]
189 187 [stder]
190 188 [stder]
191 189 [stder]
192 190 [stder]
193 191 [stder]
194 192 [stder]
195 193 [stder]
196 194 [stder]
197 195 [stder]
198 196 [stder]
199 197 [stder]
200 198 [stder]
201 199 [stder]
202 200 [stder]
203 201 [stder]
204 202 [stder]
205 203 [stder]
206 204 [stder]
207 205 [stder]
208 206 [stder]
209 207 [stder]
210 208 [stder]
211 209 [stder]
212 210 [stder]
213 211 [stder]
214 212 [stder]
215 213 [stder]
216 214 [stder]
217 215 [stder]
218 216 [stder]
219 217 [stder]
220 218 [stder]
221 219 [stder]
222 220 [stder]
223 221 [stder]
224 222 [stder]
225 223 [stder]
226 224 [stder]
227 225 [stder]
228 226 [stder]
229 227 [stder]
230 228 [stder]
231 229 [stder]
232 230 [stder]
233 231 [stder]
234 232 [stder]
235 233 [stder]
236 234 [stder]
237 235 [stder]
238 236 [stder]
239 237 [stder]
240 238 [stder]
241 239 [stder]
242 240 [stder]
243 241 [stder]
244 242 [stder]
245 243 [stder]
246 244 [stder]
247 245 [stder]
248 246 [stder]
249 247 [stder]
250 248 [stder]
251 249 [stder]
252 250 [stder]
253 251 [stder]
254 252 [stder]
255 253 [stder]
256 254 [stder]
257 255 [stder]
258 256 [stder]
259 257 [stder]
260 258 [stder]
261 259 [stder]
262 260 [stder]
263 261 [stder]
264 262 [stder]
265 263 [stder]
266 264 [stder]
267 265 [stder]
268 266 [stder]
269 267 [stder]
270 268 [stder]
271 269 [stder]
272 270 [stder]
273 271 [stder]
274 272 [stder]
275 273 [stder]
276 274 [stder]
277 275 [stder]
278 276 [stder]
279 277 [stder]
280 278 [stder]
281 279 [stder]
282 280 [stder]
283 281 [stder]
284 282 [stder]
285 283 [stder]
286 284 [stder]
287 285 [stder]
288 286 [stder]
289 287 [stder]
290 288 [stder]
291 289 [stder]
292 290 [stder]
293 291 [stder]
294 292 [stder]
295 293 [stder]
296 294 [stder]
297 295 [stder]
298 296 [stder]
299 297 [stder]
300 298 [stder]
301 299 [stder]
302 300 [stder]
303 301 [stder]
304 302 [stder]
305 303 [stder]
306 304 [stder]
307 305 [stder]
308 306 [stder]
309 307 [stder]
310 308 [stder]
311 309 [stder]
312 310 [stder]
313 311 [stder]
314 312 [stder]
315 313 [stder]
316 314 [stder]
317 315 [stder]
318 316 [stder]
319 317 [stder]
320 318 [stder]
321 319 [stder]
322 320 [stder]
323 321 [stder]
324 322 [stder]
325 323 [stder]
326 324 [stder]
327 325 [stder]
328 326 [stder]
329 327 [stder]
330 328 [stder]
331 329 [stder]
332 330 [stder]
333 331 [stder]
334 332 [stder]
335 333 [stder]
336 334 [stder]
337 335 [stder]
338 336 [stder]
339 337 [stder]
340 338 [stder]
341 339 [stder]
342 340 [stder]
343 341 [stder]
344 342 [stder]
345 343 [stder]
346 344 [stder]
347 345 [stder]
348 346 [stder]
349 347 [stder]
350 348 [stder]
351 349 [stder]
352 350 [stder]
353 351 [stder]
354 352 [stder]
355 353 [stder]
356 354 [stder]
357 355 [stder]
358 356 [stder]
359 357 [stder]
360 358 [stder]
361 359 [stder]
362 360 [stder]
363 361 [stder]
364 362 [stder]
365 363 [stder]
366 364 [stder]
367 365 [stder]
368 366 [stder]
369 367 [stder]
370 368 [stder]
371 369 [stder]
372 370 [stder]
373 371 [stder]
374 372 [stder]
375 373 [stder]
376 374 [stder]
377 375 [stder]
378 376 [stder]
379 377 [stder]
380 378 [stder]
381 379 [stder]
382 380 [stder]
383 381 [stder]
384 382 [stder]
385 383 [stder]
386 384 [stder]
387 385 [stder]
388 386 [stder]
389 387 [stder]
390 388 [stder]
391 389 [stder]
392 390 [stder]
393 391 [stder]
394 392 [stder]
395 393 [stder]
396 394 [stder]
397 395 [stder]
398 396 [stder]
399 397 [stder]
400 398 [stder]
401 399 [stder]
402 400 [stder]
403 401 [stder]
404 402 [stder]
405 403 [stder]
406 404 [stder]
407 405 [stder]
408 406 [stder]
409 407 [stder]
410 408 [stder]
411 409 [stder]
412 410 [stder]
413 411 [stder]
414 412 [stder]
415 413 [stder]
416 414 [stder]
417 415 [stder]
418 416 [stder]
419 417 [stder]
420 418 [stder]
421 419 [stder]
422 420 [stder]
423 421 [stder]
424 422 [stder]
425 423 [stder]
426 424 [stder]
427 425 [stder]
428 426 [stder]
429 427 [stder]
430 428 [stder]
431 429 [stder]
432 430 [stder]
433 431 [stder]
434 432 [stder]
435 433 [stder]
436 434 [stder]
437 435 [stder]
438 436 [stder]
439 437 [stder]
440 438 [stder]
441 439 [stder]
442 440 [stder]
443 441 [stder]
444 442 [stder]
445 443 [stder]
446 444 [stder]
447 445 [stder]
448 446 [stder]
449 447 [stder]
450 448 [stder]
451 449 [stder]
452 450 [stder]
453 451 [stder]
454 452 [stder]
455 453 [stder]
456 454 [stder]
457 455 [stder]
458 456 [stder]
459 457 [stder]
460 458 [stder]
461 459 [stder]
462 460 [stder]
463 461 [stder]
464 462 [stder]
465 463 [stder]
466 464 [stder]
467 465 [stder]
468 466 [stder]
469 467 [stder]
470 468 [stder]
471 469 [stder]
472 470 [stder]
473 471 [stder]
474 472 [stder]
475 473 [stder]
476 474 [stder]
477 475 [stder]
478 476 [stder]
479 477 [stder]
480 478 [stder]
481 479 [stder]
482 480 [stder]
483 481 [stder]
484 482 [stder]
485 483 [stder]
486 484 [stder]
487 485 [stder]
488 486 [stder]
489 487 [stder]
490 488 [stder]
491 489 [stder]
492 490 [stder]
493 491 [stder]
494 492 [stder]
495 493 [stder]
496 494 [stder]
497 495 [stder]
498 496 [stder]
499 497 [stder]
500 498 [stder]
501 499 [stder]
502 500 [stder]
503 501 [stder]
504 502 [stder]
505 503 [stder]
506 504 [stder]
507 505 [stder]
508 506 [stder]
509 507 [stder]
510 508 [stder]
511 509 [stder]
512 510 [stder]
513 511 [stder]
514 512 [stder]
515 513 [stder]
516 514 [stder]
517 515 [stder]
518 516 [stder]
519 517 [stder]
520 518 [stder]
521 519 [stder]
522 520 [stder]
523 521 [stder]
524 522 [stder]
525 523 [stder]
526 524 [stder]
527 525 [stder]
528 526 [stder]
529 527 [stder]
530 528 [stder]
531 529 [stder]
532 530 [stder]
533 531 [stder]
534 532 [stder]
535 533 [stder]
536 534 [stder]
537 535 [stder]
538 536 [stder]
539 537 [stder]
540 538 [stder]
541 539 [stder]
542 540 [stder]
543 541 [stder]
544 542 [stder]
545 543 [stder]
546 544 [stder]
547 545 [stder]
548 546 [stder]
549 547 [stder]
550 548 [stder]
551 549 [stder]
552 550 [stder]
553 551 [stder]
554 552 [stder]
555 553 [stder]
556 554 [stder]
557 555 [stder]
558 556 [stder]
559 557 [stder]
560 558 [stder]
561 559 [stder]
562 560 [stder]
563 561 [stder]
564 562 [stder]
565 563 [stder]
566 564 [stder]
567 565 [stder]
568 566 [stder]
569 567 [stder]
570 568 [stder]
571 569 [stder]
572 570 [stder]
573 571 [stder]
574 572 [stder]
575 573 [stder]
576 574 [stder]
577 575 [stder]
578 576 [stder]
579 577 [stder]
580 578 [stder]
581 579 [stder]
582 580 [stder]
583 581 [stder]
584 582 [stder]
585 583 [stder]
586 584 [stder]
587 585 [stder]
588 586 [stder]
589 587 [stder]
590 588 [stder]
591 589 [stder]
592 590 [stder]
593 591 [stder]
594 592 [stder]
595 593 [stder]
596 594 [stder]
597 595 [stder]
598 596 [stder]
599 597 [stder]
600 598 [stder]
601 599 [stder]
602 600 [stder]
603 601 [stder]
604 602 [stder]
605 603 [stder]
606 604 [stder]
607 605 [stder]
608 606 [stder]
609 607 [stder]
610 608 [stder]
611 609 [stder]
612 610 [stder]
613 611 [stder]
614 612 [stder]
615 613 [stder]
616 614 [stder]
617 615 [stder]
618 616 [stder]
619 617 [stder]
620 618 [stder]
621 619 [stder]
622 620 [stder]
623 621 [stder]
624 622 [stder]
625 623 [stder]
626 624 [stder]
627 625 [stder]
628 626 [stder]
629 627 [stder]
630 628 [stder]
631 629 [stder]
632 630 [stder]
633 631 [stder]
634 632 [stder]
635 633 [stder]
636 634 [stder]
637 635 [stder]
638 636 [stder]
639 637 [stder]
640 638 [stder]
641 639 [stder]
642 640 [stder]
643 641 [stder]
644 642 [stder]
645 643 [stder]
646 644 [stder]
647 645 [stder]
648 646 [stder]
649 647 [stder]
650 648 [stder]
651 649 [stder]
652 650 [stder]
653 651 [stder]
654 652 [stder]
655 653 [stder]
656 654 [stder]
657 655 [stder]
658 656 [stder]
659 657 [stder]
660 658 [stder]
661 659 [stder]
662 660 [stder]
663 661 [stder]
664 662 [stder]
665 663 [stder]
666 664 [stder]
667 665 [stder]
668 666 [stder]
669 667 [stder]
670 668 [stder]
671 669 [stder]
672 670 [stder]
673 671 [stder]
674 672 [stder]
675 673 [stder]
676 674 [stder]
677 675 [stder]
678 676 [stder]
679 677 [stder]
680 678 [stder]
681 679 [stder]
682 680 [stder]
683 681 [stder]
684 682 [stder]
685 683 [stder]
686 684 [stder]
687 685 [stder]
688 686 [stder]
689 687 [stder]
690 688 [stder]
691 689 [stder]
692 690 [stder]
693 691 [stder]
694 692 [stder]
695 693 [stder]
696 694 [stder]
697 695 [stder]
698 696 [stder]
699 697 [stder]
700 698 [stder]
701 699 [stder]
702 700 [stder]
703 701 [stder]
704 702 [stder]
705 703 [stder]
706 704 [stder]
707 705 [stder]
708 706 [stder]
709 707 [stder]
710 708 [stder]
711 709 [stder]
712 710 [stder]
713 711 [stder]
714 712 [stder]
715 713 [stder]
716 714 [stder]
717 715 [stder]
718 716 [stder]
719 717 [stder]
720 718 [stder]
721 719 [stder]
722 720 [stder]
723 721 [stder]
724 722 [stder]
725 723 [stder]
726 724 [stder]
727 725 [stder]
728 726 [stder]
729 727 [stder]
730 728 [stder]
731 729 [stder]
732 730 [stder]
733 731 [stder]
734 732 [stder]
735 733 [stder]
736 734 [stder]
737 735 [stder]
738 736 [stder]
739 737 [stder]
740 738 [stder]
741 739 [stder]
742 740 [stder]
743 741 [stder]
744 742 [stder]
745 743 [stder]
746 744 [stder]
747 745 [stder]
748 746 [stder]
749 747 [stder]
750 748 [stder]
751 749 [stder]
752 750 [stder]
753 751 [stder]
754 752 [stder]
755 753 [stder]
756 754 [stder]
757 755 [stder]
758 756 [stder]
759 757 [stder]
760 758 [stder]
761 759 [stder]
762 760 [stder]
763 761 [stder]
764 762 [stder]
765 763 [stder]
766 764 [stder]
767 765 [stder]
768 766 [stder]
769 767 [stder]
770 768 [stder]
771 769 [stder]
772 770 [stder]
773 771 [stder]
774 772 [stder]
775 773 [stder]
776 774 [stder]
777 775 [stder]
778 776 [stder]
779 777 [stder]
780 778 [stder]
781 779 [stder]
782 780 [stder]
783 781 [stder]
784 782 [stder]
785 783 [stder]
786 784 [stder]
787 785 [stder]
788 786 [stder]
789 787 [stder]
790 788 [stder]
791 789 [stder]
792 790 [stder]
793 791 [stder]
794 792 [stder]
795 793 [stder]
796 794 [stder]
797 795 [stder]
798 796 [stder]
799 797 [stder]
800 798 [stder]
801 799 [stder]
802 800 [stder]
803 801 [stder]
804 802 [stder]
805 803 [stder]
806 804 [stder]
807 805 [stder]
808 806 [stder]
809 807 [stder]
810 808 [stder]
811 809 [stder]
812 810 [stder]
813 811 [stder]
814 812 [stder]
815 813 [stder]
816 814 [stder]
817 815 [stder]
818 816 [stder]
819 817 [stder]
820 818 [stder]
821 819 [stder]
822 820 [stder]
823 821 [stder]
824 822 [stder]
825 823 [stder]
826 824 [stder]
827 825 [stder]
828 826 [stder]
829 827 [stder]
830 828 [stder]
831 829 [stder]
832 830 [stder]
833 831 [stder]
834 832 [stder]
835 833 [stder]
836 834 [stder]
837 835 [stder]
838 836 [stder]
839 837 [stder]
840 838 [stder]
841 839 [stder]
842 840 [stder]
843 841 [stder]
844 842 [stder]
845 843 [stder]
846 844 [stder]
847 845 [stder]
848 846 [stder]
849 847 [stder]
850 848 [stder]
851 849 [stder]
852 850 [stder]
853 851 [stder]
854 852 [stder]
855 853 [stder]
856 854 [stder]
857 855 [stder]
858 856 [stder]
859 857 [stder]
860 858 [stder]
861 859 [stder]
862 860 [stder]
863 861 [stder]
864 862 [stder]
865 863 [stder]
866 864 [stder]
867 865 [stder]
868 866 [stder]
869 867 [stder]
870 868 [stder]
871 869 [stder]
872 870 [stder]
873 871 [stder]
874 872 [stder]
875 873 [stder]
876 874 [stder]
877 875 [stder]
878 876 [stder]
879 877 [stder]
880 878 [stder]
881 879 [stder]
882 880 [stder]
883 881 [stder]
884 882 [stder]
885 883 [stder]
886 884 [stder]
887 885 [stder]
888 886 [stder]
889 887 [stder]
890 888 [stder]
891 889 [stder]
892 890 [stder]
893 891 [stder]
894 892 [stder]
895 893 [stder]
896 894 [stder]
897 895 [stder]
898 896 [stder]
899 897 [stder]
900 898 [stder]
901 899 [stder]
902 900 [stder]
903 901 [stder]
904 902 [stder]
905 903 [stder]
906 904 [stder]
907 905 [stder]
908 906 [stder]
909 907 [stder]
910 908 [stder]
911 909 [stder]
912 910 [stder]
913 911 [stder]
914 912 [stder]
915 913 [stder]
916 914 [stder]
917 915 [stder]
918 916 [stder]
919 917 [stder]
920 918 [stder]
921 919 [stder]
922 920 [stder]
923 921 [stder]
924 922 [stder]
925 923 [stder]
926 924 [stder]
927 925 [stder]
928 926 [stder]
929 927 [stder]
930 928 [stder]
931 929 [stder]
932 930 [stder]
933 931 [stder]
934 932 [stder]
935 933 [stder]
936 934 [stder]
937 935 [stder]
938 936 [stder]
939 937 [stder]
940 938 [stder]
941 939 [stder]
942 940 [stder]
943 941 [stder]
944 942 [stder]
945 943 [stder]
946 944 [stder]
947 945 [stder]
948 946 [stder]
949 947 [stder]
950 948 [stder]
951 949 [stder]
952 950 [stder]
953 951 [stder]
954 952 [stder]
955 953 [stder]
956 954 [stder]
957 955 [stder]
958 956 [stder]
959 957 [stder]
960 958 [stder]
961 959 [stder]
962 960 [stder]
963 961 [stder]
964 962 [stder]
965 963 [stder]
966 964 [stder]
967 965 [stder]
968 966 [stder]
969 967 [stder]
970 968 [stder]
971 969 [stder]
972 970 [stder]
973 971 [stder]
974 972 [stder]
975 973 [stder]
976 974 [stder]
977 975 [stder]
978 976 [stder]
979 977 [stder]
980 978 [stder]
981 979 [stder]
982 980 [stder]
983 981 [stder]
984 982 [stder]
985 983 [stder]
986 984 [stder]
987 985 [stder]
988 986 [stder]
989 987 [stder]
990 988 [stder]
991 98
```

If we click on the model folder we can see the following two files:

A screenshot of a web browser window showing a file listing. The URL is https://rdurham1.eastus.instances.azureml.ms/tree/Users/rdurham/GTC/model. The browser tabs are visible at the top. The main content area shows a file tree with three items: 'labels.json', 'model.onnx', and a folder named 'model'. A red box highlights the 'model' folder. Below the tree is a table with columns 'Name', 'Last Modified', and 'File size'. The 'labels.json' file was modified 'seconds ago' and is 31 B. The 'model.onnx' file was modified '6 minutes ago' and is 86 MB.

Name	Last Modified	File size
labels.json	seconds ago	31 B
model.onnx	6 minutes ago	86 MB



We can click on the text box and download each file to our local workstation.

A screenshot of a web browser window showing a file listing. The URL is https://rdurham1.eastus.instances.azureml.ms/tree/Users/rdurham/GTC/model. The browser tabs are visible at the top. The main content area shows a file tree with three items: 'labels.json', 'model.onnx', and a folder named 'model'. A red box highlights the 'Clusters' tab in the navigation bar. Below the tree is a table with columns 'Name', 'Last Modified', and 'File size'. The 'labels.json' file was modified 'seconds ago' and is 31 B. The 'model.onnx' file was modified '8 minutes ago' and is 86 MB.

Name	Last Modified	File size
labels.json	seconds ago	31 B
model.onnx	8 minutes ago	86 MB

Now we have the label and model files needed to deploy to our device. We will use these later for our deployment to the target environment.

If for some reason you need to retrieve your model and label files after your training run is over or you reopen the jupyter notebook later you can always execute the **"Optional: if you trained the model earlier you can still download the model and label using Run id. This will be the highest level run id from a child run perspective"** cell. You will need to retrieve the **run_id** for the experiment to get the best model and labels file and replace the one in the sample notebook (see below).

The screenshot shows a Jupyter Notebook interface with several tabs at the top: 'Exercises | Exploring the Deep...', 'What is GStreamer?', 'Session expired', 'Data Labeling - Microsoft Az...', 'Users/rdurham/GTC...', and 'AutoMLImage_ObjectDetection_GTC.ipynb'. The main area contains Python code:

```
# Select the best child run
from azureml.train.automl import AutoMLRun
import json

run_id = "AutoML_e16605d-d6a-41f7-93c6-607705c9ff0" # Specify the run ID
automl_image_run = AutoMLRun(experiment.experiment, run_id=run_id)
best_child_run = automl_image_run.get_best_child()

labels_file = "./model/labels.json"
best_child_run.download_file(name="train_artifacts/labels.json", output_file_path=labels_file)

onnx_model_path = "./model/model.onnx"
best_child_run.download_file(name="train_artifacts/model.onnx", output_file_path=onnx_model_path)
```

A callout box highlights the line `best_child_run = automl_image_run.get_best_child()` with the text: *****Optional: if you trained the model earlier you can still download the model and label using Run id. This will be the highest level run id from a child run perspective**.

Below this, another code cell is shown:

```
Load the labels and ONNX model files
```

```
import onnruntime
import json

labels_file = "./model/labels.json"
onnx_model_path = "./model/model.onnx"

with open(labels_file) as f:
    classes = json.load(f)
print(classes)
try:
```

Now let's take a look at how to use the model and the labels file to inference a test image:

Execute the **Load the labels and ONNX model files** cell. This will load the classes and session objects. Review the output results:

```
['coke', 'diet_coke', 'sprite']
ONNX model loaded...
```

Execute the **Get expected input and output details for an ONNX model** cell. This will output show the details of inputs and outputs of the onnx model.

```
No. of inputs : 1, No. of outputs : 4
0 Input name : input, Input shape : ['batch', 3, 640, 640],
Input type   : tensor(float)
```

```
0 Output name : output, Output shape : ['batch', 25200, 8]
,   Output type : tensor(float)
1 Output name : 1397, Output shape : [1, 3, 80, 80, 8],
Output type : tensor(float)
2 Output name : 1745, Output shape : [1, 3, 40, 40, 8],
Output type : tensor(float)
3 Output name : 2093, Output shape : [1, 3, 20, 20, 8],
Output type : tensor(float)
```

Execute the cells one-by-one following the **Image Inferencing Preprocessing** cell. These cells will perform the following inferencing functions:

- Add helper functions and prediction functions
- Set the test image path that points to the test image
- Get predictions from the onnx model
- Invoke the **non_max_suppression** function to remove unwanted lower confidence bounding boxes
- Get the bounding boxes, labels and scores for each object in the image
- Plot the image with its respective objects, bounding boxes, labels

Part 4: Deploy the model onto Triton Server

Triton is multi-framework, open-source software that is optimized for inference. It supports popular machine learning frameworks like TensorFlow, ONNX Runtime, PyTorch, NVIDIA TensorRT, and more. It can be used for your CPU or GPU workloads.

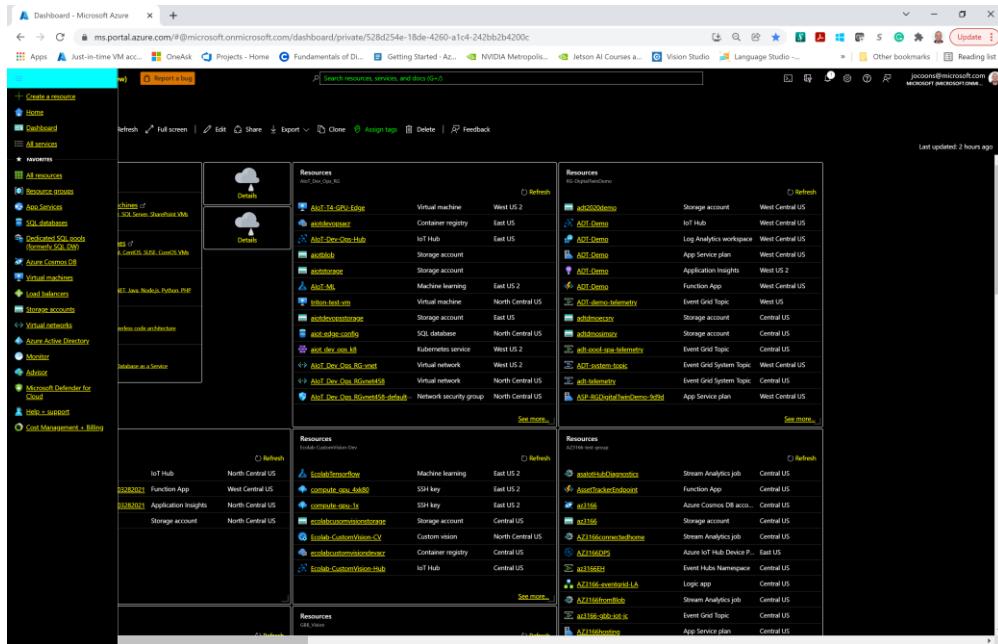
For more information how to use Triton Server in Azure Machine Learning:

[High-performance model serving with Triton \(preview\) - Azure Machine Learning | Microsoft Docs](#)

Create a VM:

1. Create a non-GPU VM in Azure (you can also use a dGpu-based machine if you have access to one) To create a VM, you'll need an Azure subscription. If you don't already have a subscription, create a free account before you begin.

2. Log into the Azure Portal, from there you can click I the upper left corner of the screen to bring up the menu, and then click add resource. From there you can either choose Virtual Machine, or the Ubuntu 18.04 Server VM option.

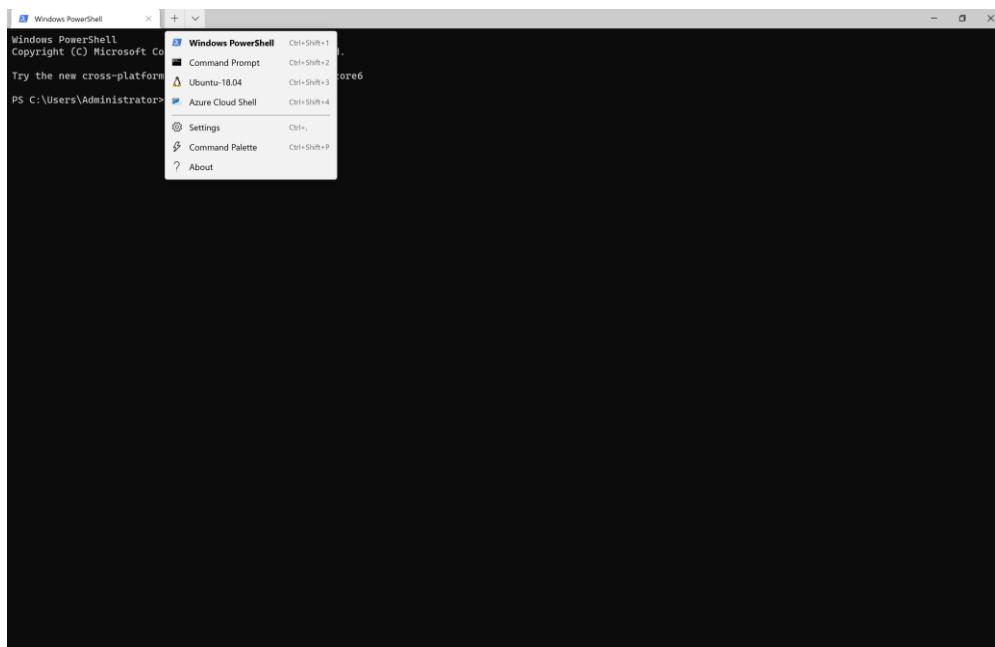


3. In the 'Create a virtual machine' blade, you will select your subscription, select or create your Resource Group, create a friendly name for your VM, choose the Region and availability options. Since this is temporary deployment, chose 'No infrastructure redundancy required'.
4. Since we will be accessing this VM in the workshop remotely, we do want to enable SSH access over Port 22. Under Administrator Account -> Authentication type, choose Password and enter a username and password. We'll use these quite often throughout the workshop, so please make a note of these or create a temporary copy of each in your text editor of choice.
5. Under 'Public Inbound Ports' allow selected ports should be already selected with SSH(22) as the selected port.
6. Since the aim is to use the simplest VM available, we can skip the additional setup blades you would normally go through, and just select the 'Review & Create' button. Once validated by the system, select 'Create' at the bottom of the screen.
7. This will now create several resources: the virtual machine, a network security group and public IP addresses. When provisioning is complete, select the 'Go to resource' button at the bottom.

8. Copy the Public IP Address in the Overview blade, and save this to your text editor of choice. We'll use this for accessing the VM remotely via a terminal emulator, i.e. TeraTerm or [Windows Terminal](#).

Log into the VM: 1.

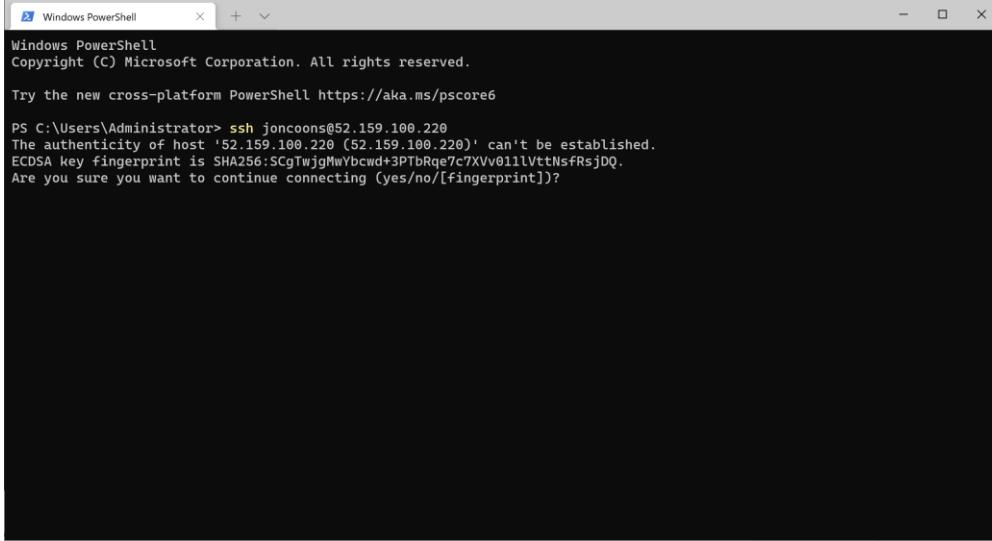
Open your terminal emulator of choice. For illustration, we'll be using Windows Terminal, as it allows for multiple windows to be simultaneously connected concurrently to the VM. We'll be using one window to start the Triton Server, one window to execute a Python script and one to copy images to a directory for processing via the CLI. With Windows Terminal, you also have your choice of CLI experience, PowerShell, Command Prompt, Ubuntu-18.04 (if WSL-2 is installed) or Azure Cloud Shell.



9. Copy the username you used to set up the VM in the previous stage, and in the command line run:

```
ssh <username>@<your VM IP address>
```

This will prompt you for the password you saved previously to your text editor. Copy this, and right click in the command line to paste. If logging in for the first time, you'll see the following message:

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the command "ssh joncoons@52.159.100.220" being run. The output indicates that the host's authenticity cannot be established, and it asks if the user is sure they want to continue connecting. The user has not yet responded.

Type 'yes' and press enter to log into the VM.

- 10.** Now we're going to load in a few packages the Python script needs to execute properly. On the command line, enter:

sudo apt update

sudo apt install -y python3-pip python3-dev nano wget

Prior to installing the Python packages required, we'll want to add '/home/<your username>/.local/bin' to the PATH by taking the following steps in the CLI:

sudo nano ~/.bashrc

Arrow to the bottom of this file in the editor, and add the following line:

export PATH=/home/<uname>/.local/bin:\$PATH

Press Ctrl + O and press enter to save the file, then press Ctrl + X to exit. On the command line, run:

source ~/.bashrc

This will reload the configuration for the server to include .local/bin in the PATH.

Now that we've loaded the Ubuntu package requirements and added the directory to PATH, we're going to install the required Python packages. Copy each line individually to run in the terminal window.

```
python3 -m pip install --upgrade pip wheel setuptools
```

```
python3 -m pip install numpy>=1.19.0 opencv-contrib-
python-headless tritonclient geventhttpclient
```

```
python3 -m pip install torch torchvision pandas tqdm
PyYAML scipy seaborn requests pybind11 pytest protobuf
objdict onnxruntime
```

If you are using a Nvidia GPU-capable VM, you can use onnxruntime-gpu instead of onnxruntime to take advantage of the CUDA/cuDNN acceleration.

11. To run the Triton Server container from Nvidia, we're going to need a container engine. If using a Nvidia GPU-based VM, nvidia-docker2 is the correct runtime, and this step can be skipped. For the non-GPU VMs, however, we'll utilize Moby, which is the OSS on which Docker is based. Microsoft has a distribution of this container runtime which can be installed using the following commands:

```
wget
https://packages.microsoft.com/config/ubuntu/18.04/multi-
arch/packages-microsoft-prod.deb -O packages-microsoft-
prod.deb
```

```
sudo dpkg -i packages-microsoft-prod.deb
```

```
rm packages-microsoft-prod.deb
```

Now we can install Moby:

```
sudo apt update
```

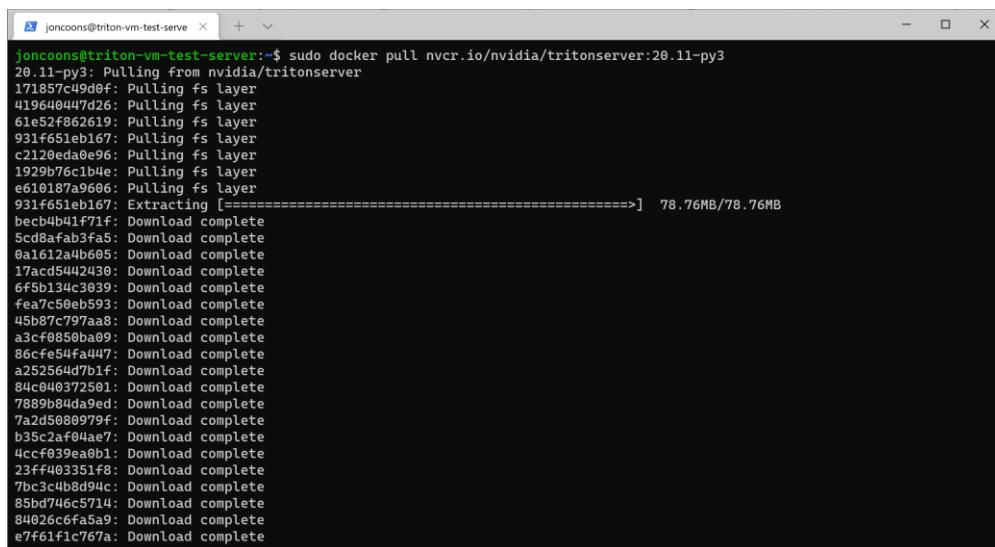
```
sudo apt install -y moby-engine
```

sudo apt update

- 12.** Now, we're ready to pull the container for the Triton Server from the Nividia NGC repository. You can also pull the container during the 'docker run' statement when we get to that step, but for simplicity, we'll do the pull now. In the terminal emulator, run:

sudo docker pull nvcr.io/nvidia/tritonserver:20.11-py3

This will take some time to download the container layers and extract them.



```
joncoons@triton-vm-test-server:~$ sudo docker pull nvcr.io/nvidia/tritonserver:20.11-py3
20.11-py3: Pulling from nvidia/tritonserver
171857c49d0f: Pulling fs layer
4196404407d26: Pulling fs layer
61e52f862619: Pulling fs layer
931f651eb167: Pulling fs layer
c2120eda0e96: Pulling fs layer
1929b76c1b4e: Pulling fs layer
e610187a9606: Pulling fs layer
931f651eb167: Extracting [=====] 78.76MB/78.76MB
beccb4b41f71f: Download complete
5cd8afab3fa5: Download complete
0a1612a4b605: Download complete
17acd5442430: Download complete
6f5b134c3039: Download complete
fea7c50eb593: Download complete
45b87c797aa8: Download complete
a3cf0850ba99: Download complete
86cfe54fa447: Download complete
a252564d7bf1f: Download complete
84c040372501: Download complete
7889b84da9ed: Download complete
7a2d5080979f: Download complete
b35c2af04ae7: Download complete
4ccf039ea0b1: Download complete
23ff403351f8: Download complete
7bc3c4b8d94c: Download complete
85bd746c5714: Download complete
84026c6fa59: Download complete
e7f61f1c767a: Download complete
```

- 13.** Now we're ready to copy the 'demo' directory over to the VM. If you download the demo.zip file from the repository, unzip this locally on your PC. Open a command prompt window, either in the utility, or open another window in Windows Terminal. Depending on where you unzipped the files, run the following command in the CLI:

*scp -r <path to unzipped>/demo <vm
username>@<x.x.x.x vm IP address>:/home/<vm
username>/*

Here is an example:

```

PS C:\Users\Administrator> scp -r Documents/AIoT_Dev_Ops/'Nvidia Triton' demo joncoons@52.159.100.220:/home/joncoons/
joncoons@52.159.100.220's password:
frame_grabber.py                                100% 9950   140.7KB/s  00:00
frame_grabber_onnxruntime.py                     100% 9391   172.6KB/s  00:00
activations.py                                  100% 3820   119.0KB/s  00:00
autoanchor.py                                   100% 7299   190.0KB/s  00:00
mime.sh                                         100% 806    25.0KB/s  00:00
resume.py                                       100% 1132   35.2KB/s  00:00
userdata.sh                                     100% 1328   27.6KB/s  00:00
__init__.py                                     100% 0       0.0KB/s  00:00
datasets.py                                     100% 48KB   297.4KB/s 00:00
example_request.py                            100% 312    4.4KB/s  00:00
README.md                                       100% 1777   55.5KB/s  00:00
restapi.py                                      100% 1115   34.9KB/s  00:00
general.py                                      100% 31KB   263.3KB/s 00:00
additional_requirements.txt                   100% 109    3.4KB/s  00:00
app.yaml                                        100% 186    8.4KB/s  00:00
Dockerfile                                       100% 846    38.0KB/s  00:00
google_utils.py                                100% 6107   157.4KB/s 00:00
loss.py                                           100% 9676   176.1KB/s 00:00
metrics.py                                      100% 9467   215.1KB/s 00:00
plots.py                                         100% 19KB   223.2KB/s 00:00
torch_utils.py                                 100% 13KB   183.5KB/s 00:00
log_dataset.py                                100% 896    28.4KB/s  00:00
wandb_utils.py                                100% 18KB   289.5KB/s 00:00

```

- 14.** Once we have this copied over to the VM, let's switch back over to the terminal window connected to the VM and set the permissions for this directory. In that CLI, enter:

sudo chmod -R 777 demo

Now we're ready to run the example Python script on the Triton Server. If you look in the 'demo' directory, you'll see a number of additional folders and files.

In the 'app' folder, there are two Python scripts – frame_grabber.py that uses the Triton Inference Server, and frame_grabber_onnxruntime.py that can be used standalone. The 'utils' folder inside of the 'app' directory contains python scripts to enable the interpretation of the model's output tensor.

Both python scripts are set to *watch* the 'image_sink' directory for any image files that are placed there. In the images-sample, you'll find a number of images we will copy via command line to the 'image_sink' for processing. The python scripts automatically delete the files from the 'image_sink' after the inference has been completed.

In the model folder, you'll find a folder for the name of the model, which holds the model configuration file for the Triton Inference server, as well as the label file. Also included is a folder denoting the version of the model, which contains the ONNX model that the server uses to inference.

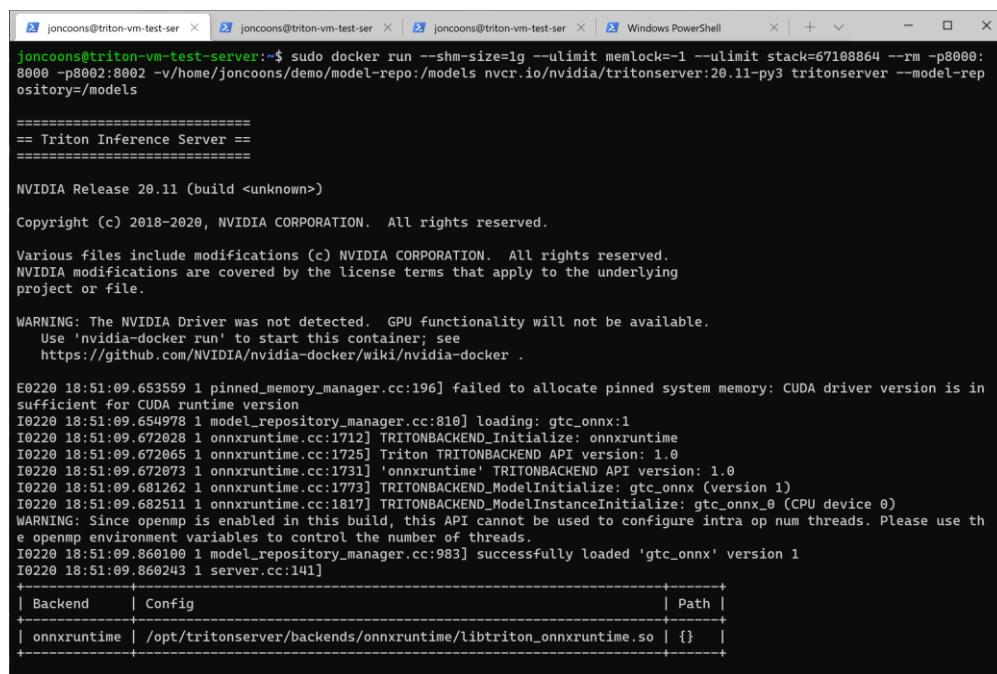
If the model detects the objects it was trained on, the python script will create an annotation of that inference with a bounding box, tag name and confidence score. The script saves the image into the 'images-annotated' folder, using a unique name using a timestamp, which we can download to view locally. That way, you can copy

the same images over and over again to the ‘image_sink’ but have new annotated images created each run for illustration purposes.

To get started on the inferencing, we’ll want to open two additional windows in the Windows Terminal, and ssh into the VM from each window.

- 15.** In the first window, run the following command, but first change out the `username` place holder with the username for the VM:

```
sudo docker run --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 --rm -p8000:8000 -p8002:8002 -v/home/<vm username>/demo/model-repo:/models nvcr.io/nvidia/tritonserver:20.11-py3 tritonserver --model-repository=/models
```



```
joncoons@triton-vm-test-ser ~$ sudo docker run --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 --rm -p8000:8000 -p8002:8002 -v/home/joncoons/demo/model-repo:/models nvcr.io/nvidia/tritonserver:20.11-py3 tritonserver --model-repository=/models
=====
== Triton Inference Server ==
=====

NVIDIA Release 20.11 (build <unknown>)

Copyright (c) 2018-2020, NVIDIA CORPORATION. All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying
project or file.

WARNING: The NVIDIA Driver was not detected. GPU functionality will not be available.
Use 'nvidia-docker run' to start this container; see
https://github.com/NVIDIA/nvidia-docker/wiki/nvidia-docker

E0220 18:51:09.653559 1 pinned_memory_manager.cc:196] failed to allocate pinned system memory: CUDA driver version is insufficient for CUDA runtime version
I0220 18:51:09.654978 1 model_repository_manager.cc:810] loading: gtc_onnx:1
I0220 18:51:09.672028 1 onnxruntime.cc:1712] TRITONBACKEND_Initialize: onnxruntime
I0220 18:51:09.672065 1 onnxruntime.cc:1725] Triton TRITONBACKEND API version: 1.0
I0220 18:51:09.672073 1 onnxruntime.cc:1731] 'onnxruntime' TRITONBACKEND API version: 1.0
I0220 18:51:09.681262 1 onnxruntime.cc:1773] TRITONBACKEND_ModelInitialize: gtc_onnx (version 1)
I0220 18:51:09.682511 1 onnxruntime.cc:1817] TRITONBACKEND_ModelInstanceInitialize: gtc_onnx_0 (CPU device 0)
WARNING: Since openmp is enabled in this build, this API cannot be used to configure intra op num threads. Please use the openmp environment variables to control the number of threads.
I0220 18:51:09.860100 1 model_repository_manager.cc:983] successfully loaded 'gtc_onnx' version 1
I0220 18:51:09.860243 1 server.cc:141]

+-----+
| Backend | Config | Path |
+-----+
| onnxruntime | /opt/tritonserver/backends/onnxruntime/libtriton_onnxruntime.so | {} |
```

- 16.** In the second window, copy the following command, changing the `<vm username>` to your value, and set the `<probability threshold>` to your desired confidence level between 0 and 1 (by default, this is set to .6)

```
python3 demo/app/frame_grabber.py -u <vm username> -p .07
```

- 17.** In the third window, copy and paste this command to copy the image files from the 'images_sample' folder to the 'image_sink' folder:

```
cp demo/images_sample/* demo/image_sink/
```

If you go back to your second window, you can see the execution of the model, including the model statistics and the returned inference in the form of the following Python dictionary:

```
{
    'model_name': self.model_name,
    'inferencing_time': t_infer,
    'object_detected': "True",
    'camera_id': self.camID,
    'camera_name': f'{self.camLocation}-{self.camPosition}',
    'annotated_image_name': annotatedName,
    'annotated_image_path': annotatedPath,
    'created': created,
    'detected_objects': result['predictions']
}
```

Here is a sample view of what you should see in the second window as the script executes:

```
Deleted image: /home/joncoons/demo/image_sink/98fae924-b598-11eb-8f73-0242ac110002.jpg

[{"model_stats": [{"name": "gtc_onnx", "version": "1", "last_inference": 1645385230275, "inference_count": 116, "executed_count": 116, "inference_stats": {"success": {"count": 116, "ns": 13441076349}, "fail": {"count": 0, "ns": 0}, "queue": {"count": 116, "ns": 7575881}, "compute_input": {"count": 116, "ns": 134861830}, "compute_infer": {"count": 116, "ns": 13271916772}, "compute_output": {"count": 116, "ns": 184484791}, "batch_stats": [{"batch_size": 1, "compute_input": {"count": 116, "ns": 134861830}, "compute_infer": {"count": 116, "ns": 13271916772}, "compute_output": {"count": 116, "ns": 184484791}}]}}, {"Detection Count: 12
Inference Message:
{
    "model_name": "gtc_onnx", "inferencing_time": 182.97052383422852, "object_detected": "True", "camera_id": "image_file", "camera_name": "table_top-side", "annotated_image_name": "table_top-side-2022002192710281417-annotated.jpg", "annotated_image_path": "/home/joncoons/demo/images_annotated/table_top-side-2022002192710281417-annotated.jpg", "created": "2022-02-20T19:27:10.281417", "detected_objects": [{"probability": 84.693116, "labelId": 0, "labelName": "coke", "bbox": {"left": 400.55175781, "top": 253.426651, "width": 464.94250488, "height": 367.72769165}, {"probability": 83.713382, "labelId": 1, "labelName": "diet_coke", "bbox": {"left": 528.04998779, "top": 363.43780518, "width": 584.56451416, "height": 462.15820312}, {"probability": 82.81498, "labelId": 1, "labelName": "diet_coke", "bbox": {"left": 269.56283569, "top": 376.00695801, "width": 321.26956177, "height": 470.84552002}, {"probability": 82.723194, "labelId": 0, "labelName": "coke", "bbox": {"left": 260.46252441, "top": 259.53399658, "width": 323.97149658, "height": 369.66400146}, {"probability": 81.409341, "labelId": 1, "labelName": "diet_coke", "bbox": {"left": 340.33187866, "top": 373.08662905, "width": 383.19277954, "height": 460.13253784}, {"probability": 80.9778813, "labelId": 1, "labelName": "diet_coke", "bbox": {"left": 451.75698853, "top": 375.85040283, "width": 462.93267823}, {"probability": 79.75741, "labelId": 1, "labelName": "diet_coke", "bbox": {"left": 464.8838501, "top": 365.34536743, "width": 521.6660791, "height": 462.63473511}, {"probability": 79.421031, "labelId": 1, "labelName": "diet_coke", "bbox": {"left": 197.00305176, "top": 381.88150024, "width": 252.59799194, "height": 468.22762026}, {"probability": 79.19423599999999, "labelId": 0, "labelName": "coke", "bbox": {"left": 475.00402832, "top": 250.71531677, "width": 530.42126465, "height": 358.79223633}, {"probability": 79.06372, "labelId": 0, "labelName": "coke", "bbox": {"left": 538.23297119, "top": 254.81051636, "width": 606.93927002, "height": 358.27236938}, {"probability": 73.989487, "labelId": 0, "labelName": "coke", "bbox": {"left": 331.27261353, "top": 257.70022583, "width": 391.5118103, "height": 361.55038452}, {"probability": 73.61843, "labelId": 0, "labelName": "coke", "bbox": {"left": 184.52938843, "top": 264.27941895, "width": 254.4085083, "height": 371.78729248}}]}}, {"Deleted image: /home/joncoons/demo/image_sink/13.jpg
```

- 18.** If you want to see a list of your annotated images, you can run this simple command:

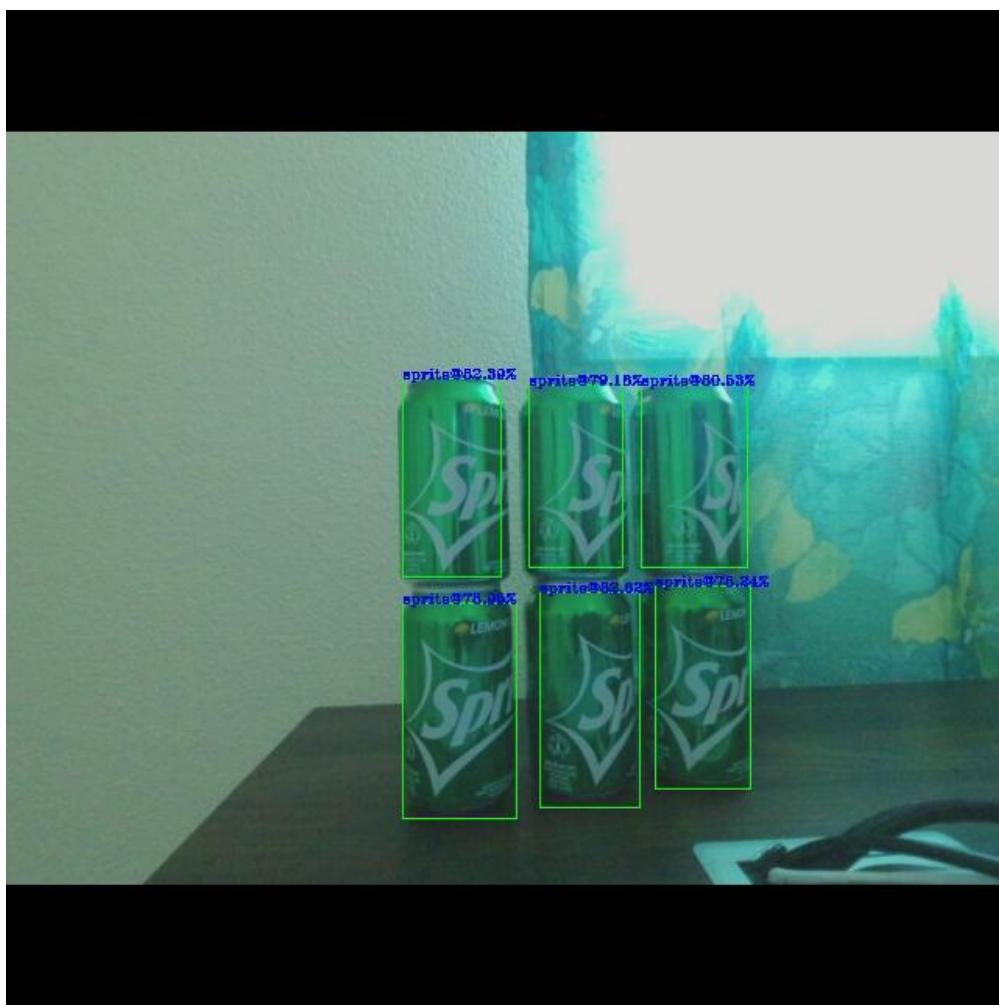
```
ls demo/annotated_images
```

- 19.** To download the images to your local machine, we'll first want to create a folder to receive the images. In a command line window, 'cd' to the directory you to place the new folder in, and run:

```
mkdir annotated_img_download
```

```
scp <uname>@x.x.x.x:/home/<uname>/demo/images_annotated/*  
annotated_img_download/
```

This will copy all of the files from the Ubuntu VM to your local device for viewing.



**Thank you for
joining Microsoft
at GTC! We
sincerely hope you
enjoyed this
workshop!**

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. Microsoft makes no warranties, express or implied, in this document.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2020 Microsoft Corporation. All rights reserved.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Microsoft, list Microsoft trademarks used in your white paper alphabetically are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.