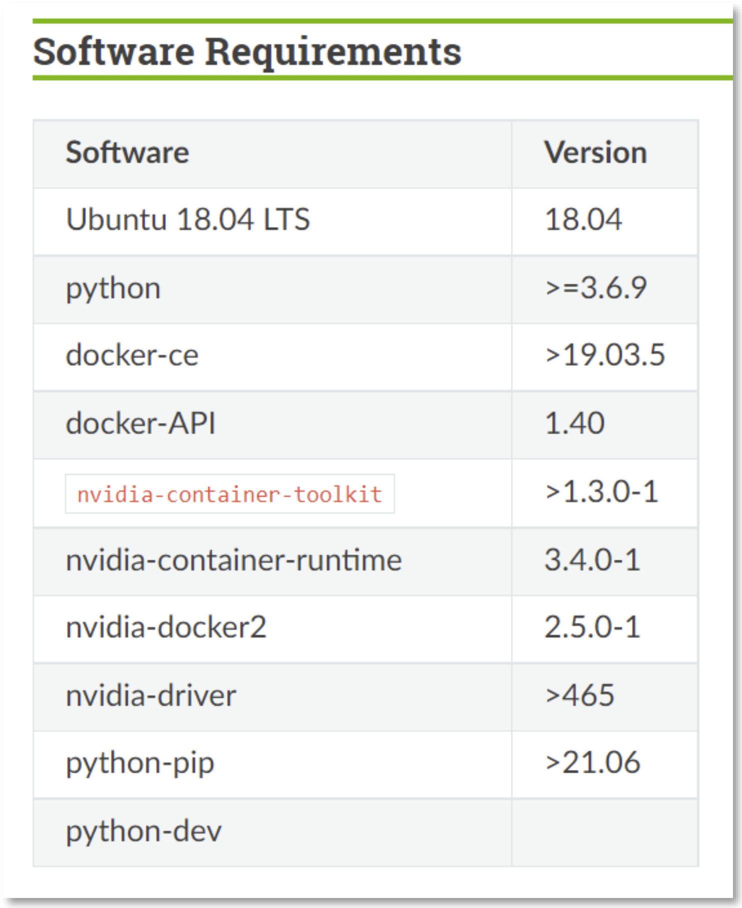
Azure ML labeling and data training with TAO toolkit for mask RCNN Detection

Mask RCNN is an object detection model based on deep convolution neural network. The model can return both the bounding box and a mask for each detected object in an image. There are two stages on mask RCNN. First it generates object dataset about the regions where there might be an object based on the input image. Second, it predicts the class of the object, refines the bounding box and generates a mask in pixel level of the object based on the first stage proposal. Here we focus on the first stage, which include object labeling, coco to TFRecord format conversion and data training using Microsoft Azure and Nvidia Tao toolkit.

1. Prerequisite

- Software requirements

Below shows the software requirement (Figure-1):



Software Requirements

Software	Version
Ubuntu 18.04 LTS	18.04
python	>=3.6.9
docker-ce	>19.03.5
docker-API	1.40
`nvidia-container-toolkit`	>1.3.0-1
nvidia-container-runtime	3.4.0-1
nvidia-docker2	2.5.0-1
nvidia-driver	>465
python-pip	>21.06
python-dev	

Figure 1. software requirement

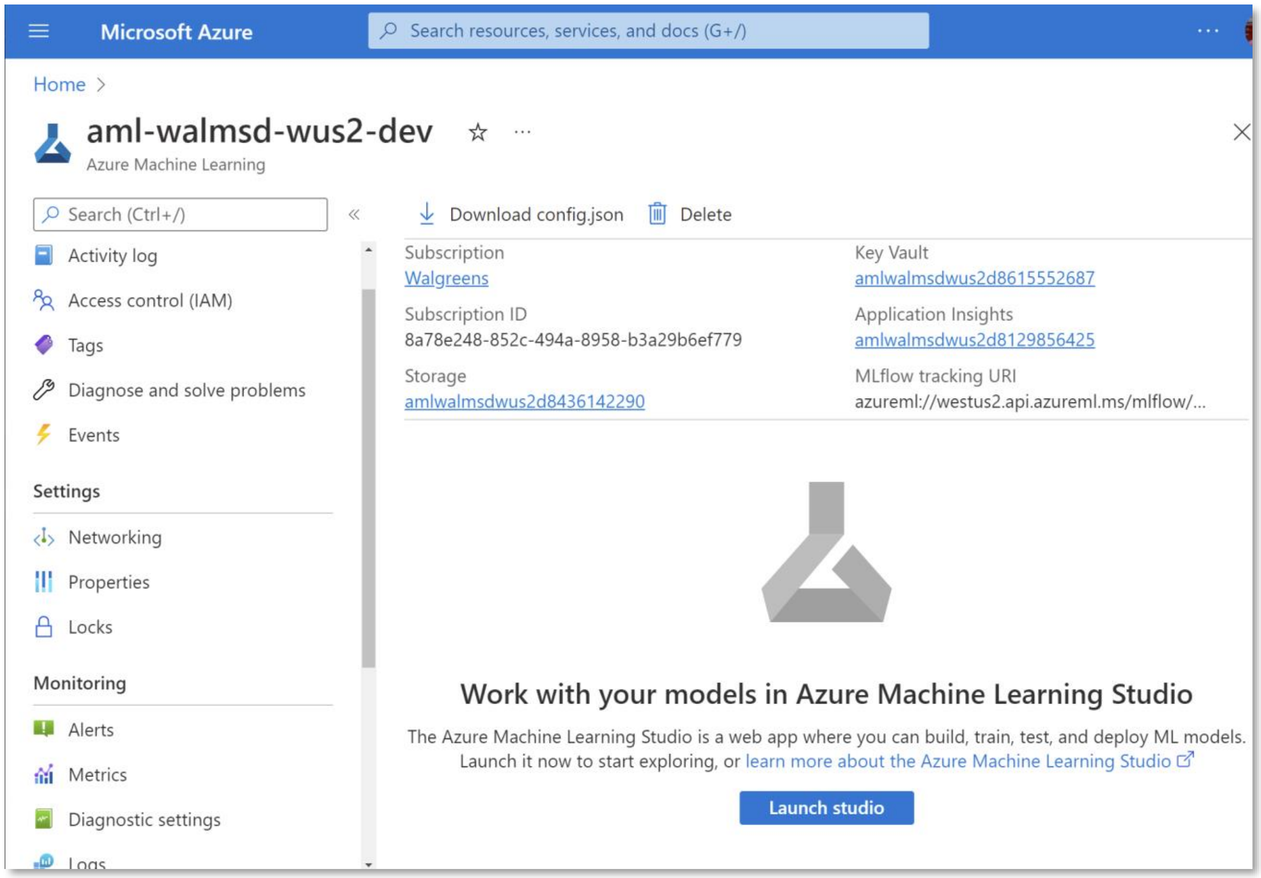
- Install Nvidia Tao Toolkit

Before installing TAO Toolkit, there are some Pre-requisites: Install Docker-ce, Nvidia-container-toolkit, and get an NGC account and API key, then log in to the NGC for using TAO Toolkit container and other package. Please see the installation details through the link: TAO Toolkit Quick Start Guide — TAO Toolkit 3.22.02 documentation (nvidia.com)

2. Azure ML Label tool

To label each item, Microsoft Azure provides ML label tool. On Azure Machine Learning section, click “Launch studio”(Figure-2), then select “data labeling” (Figure-3) if data labeling project exists. If no, create a new one by selecting “data labeling project”.

aml-walmsd-wus2-dev - Microsoft Azure



Subscription
Walgreens

Subscription ID
8a78e248-852c-494a-8958-b3a29b6ef779

Storage
amlwalmsdwus2d8436142290

Key Vault
amlwalmsdwus2d8615552687

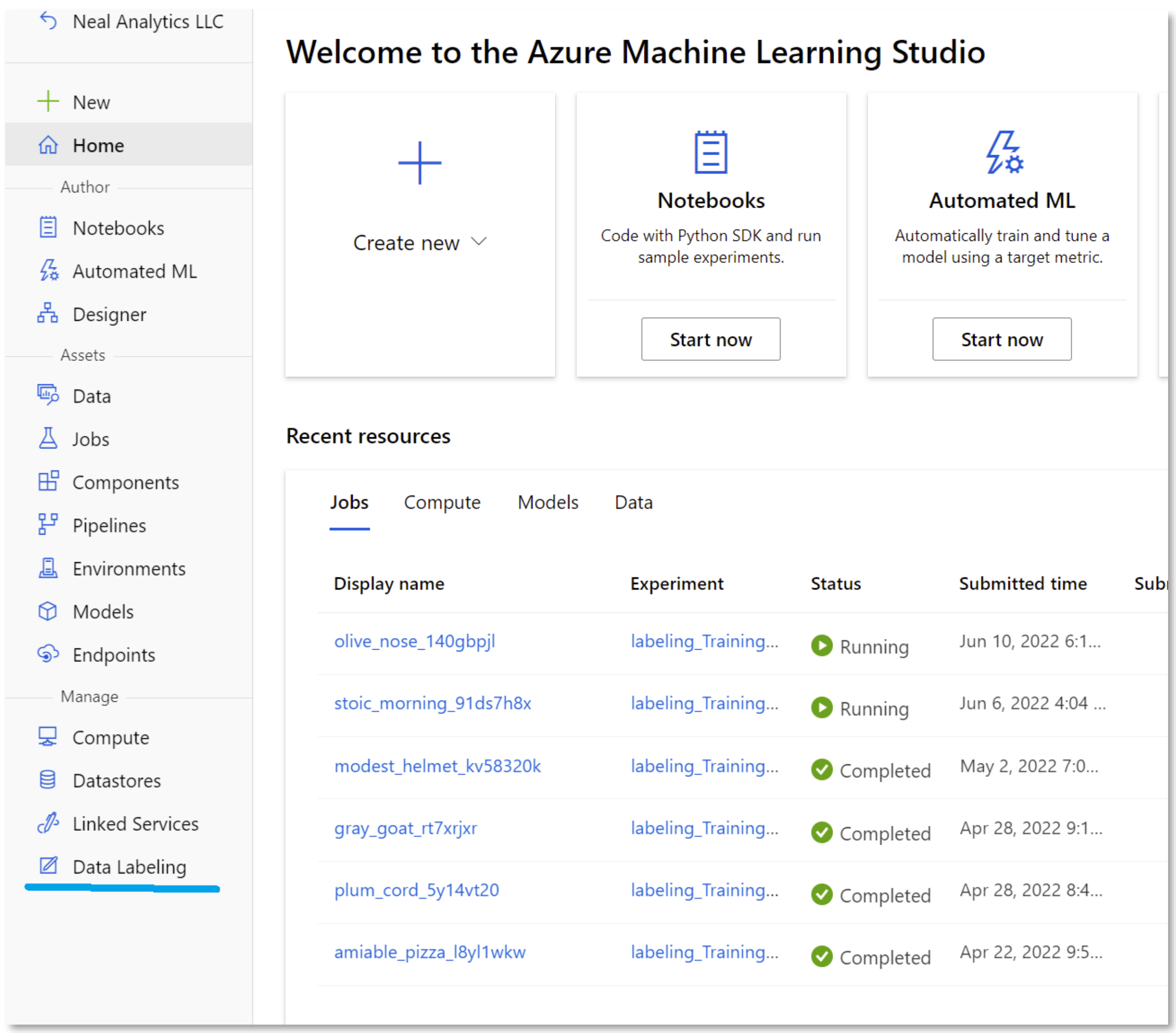
Application Insights
amlwalmsdwus2d8129856425

MLflow tracking URI
azureml://westus2.api.azureml.ms/mlflow/...

Work with your models in Azure Machine Learning Studio

The Azure Machine Learning Studio is a web app where you can build, train, test, and deploy ML models. Launch it now to start exploring, or learn more about the Azure Machine Learning Studio

Figure-2. Using Azure ML studio to label object data



Neal Analytics LLC

New

Home

Author

Notebooks

Automated ML

Designer

Assets

Data

Jobs

Components

Pipelines

Environments

Models

Endpoints

Manage

Compute

Datastores

Linked Services

Data Labeling

Welcome to the Azure Machine Learning Studio

Create new

Notebooks

Code with Python SDK and run sample experiments.

Start now

Automated ML

Automatically train and tune a model using a target metric.

Start now

Recent resources

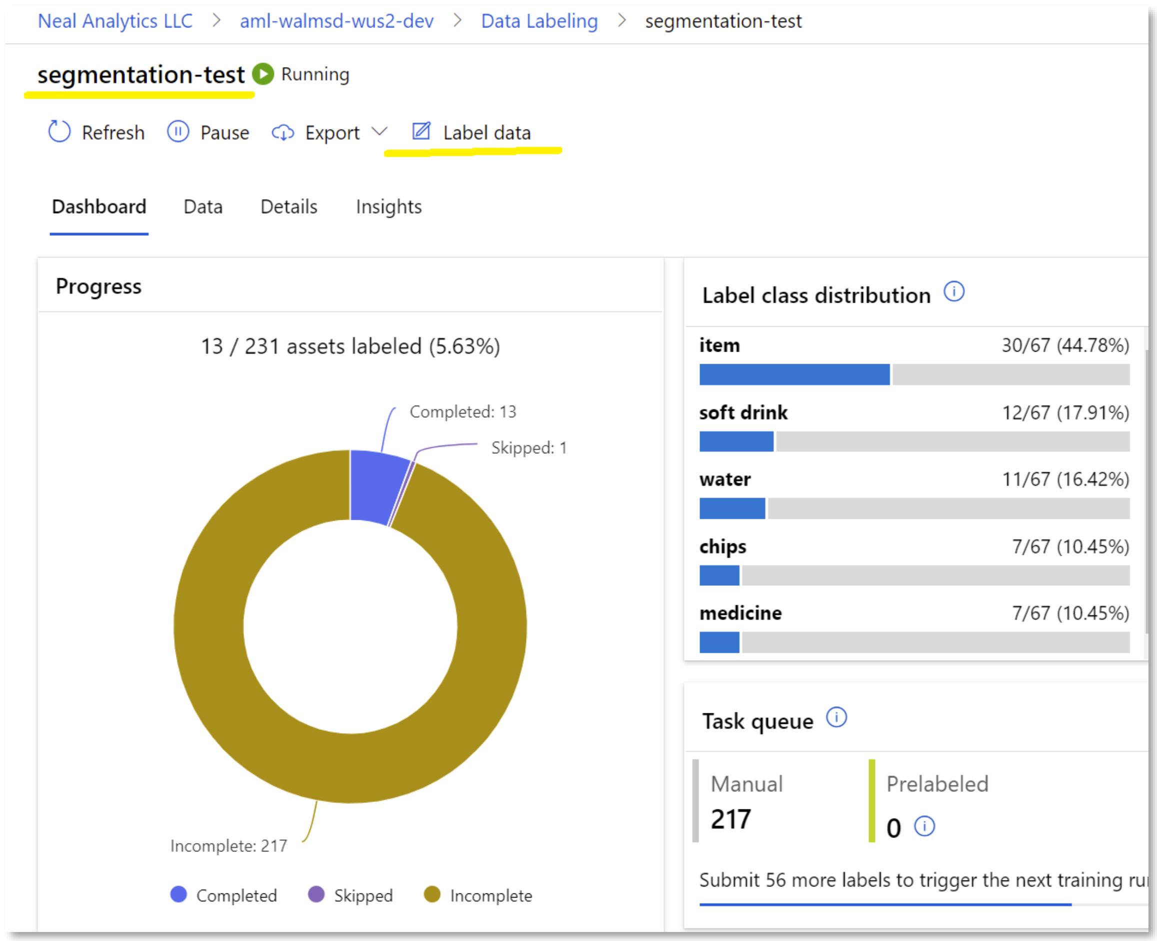
Jobs Compute Models Data

Display name	Experiment	Status	Submitted time	Sub
olive_nose_140gbpjl	labeling_Training...	Running	Jun 10, 2022 6:1...	
stoic_morning_91ds7h8x	labeling_Training...	Running	Jun 6, 2022 4:04 ...	
modest_helmet_kv58320k	labeling_Training...	Completed	May 2, 2022 7:0...	
gray_goat_rt7xrjxr	labeling_Training...	Completed	Apr 28, 2022 9:1...	
plum_cord_5y14vt20	labeling_Training...	Completed	Apr 28, 2022 8:4...	
amiable_pizza_l8yl1wkw	labeling_Training...	Completed	Apr 22, 2022 9:5...	

Figure-3. Data labeling.

On data labeling section, if you want to add a new project ➔ choose the details: media type (image or text), labeling task type (instance segmentation, classification, object identification).

If you have already had the existing data labeling project, select project name on the list➔ for example, segmentation-test ➔ click “label data”(Figure-4). Now an image will show up, you can start labeling the objects.


Neal Analytics LLC > aml-walmsd-wus2-dev > Data Labeling > segmentation-test

segmentation-test Running

Progress

13 / 231 assets labeled (5.63%)

Label class distribution

item	30/67 (44.78%)
soft drink	12/67 (17.91%)
water	11/67 (16.42%)
chips	7/67 (10.45%)
medicine	7/67 (10.45%)

Task queue

Manual	Prelabeled
217	0

Submit 56 more labels to trigger the next training ru

Figure-4. Select label data button on segmentation-test project.

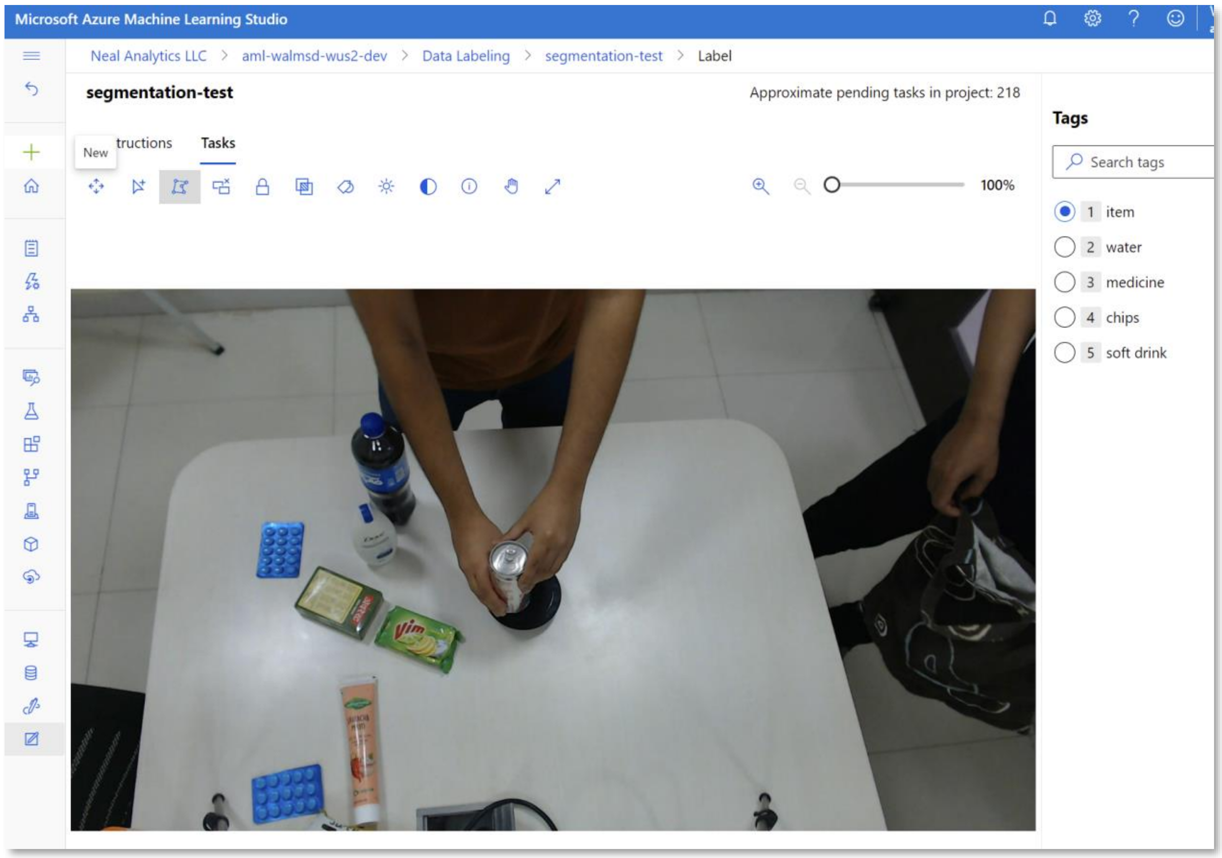

Microsoft Azure Machine Learning Studio
Neal Analytics LLC > aml-walmsd-wus2-dev > Data Labeling > segmentation-test > Label
segmentation-test
Approximate pending tasks in project: 218
New
tructions
Tasks
100%
Tags
Search tags
1 item
2 water
3 medicine
4 chips
5 soft drink

Figure-5. An image is loaded for labeling.

After clicking a few points along the boundary of an item, then double-click. A polygon is generated and shown in red (Figure-6). Moreover, you can label the object type on up-right side. Once it's done, click submit button. In addition, you make any changes for each region.

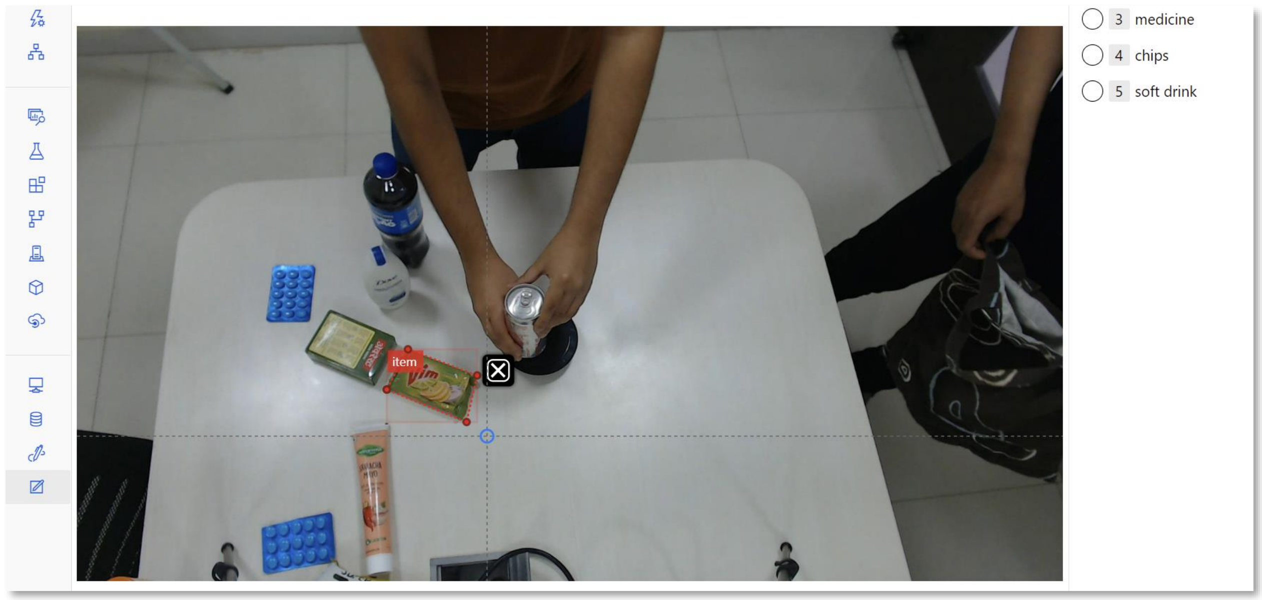


Figure-6. A polygon region is created

Once finishing the labeling, you export labeled data to coco file or Azure ML dataset for further processing.

3. Downloading Azure ML dataset with python code

To download all labeled dataset, the Microsoft Azure provides blobs management python SDK for the developers. If you have an Azure account with an active subscription and an Azure storage account, you can download the data set with Azure storage SDK. Below are the packages to be installed:

- pip install azure-storage-blob
- pip install azure-identity

Here **download-aml_blobs.py** (~/repos/manufacturing-demo/) is used to download the labeled image dataset by running this command:

Python3 download-aml_blobs.py ./aml_coco_config.ini

Below is the configuration file of aml_coco_config.ini (figure-7):

```
[section]
connection_string =
DefaultEndpointsProtocol=https;AccountName=amlwalmsdwus2d8436142290;AccountKey=Ir
```

```
D/MOSsO342xQ0COSLc/7tZAgTHsMSx2TzC9XwKugvSisCZqiQwNB45Ok1Kp50WgcnzXi6jvATomwmEfcP
m4Q==;EndpointSuffix=core.windows.net
Blob_Container_id = azureml-blobstore-f62d7dac-3360-48d1-9530-e20a65fb87a7
[Path]
aml_exported_coco_json: ./workspace/data/inputs/d02242f7-038a-4090-b39c-
922e1542d0aa.json
```

Figure-7. aml_coco_config.ini

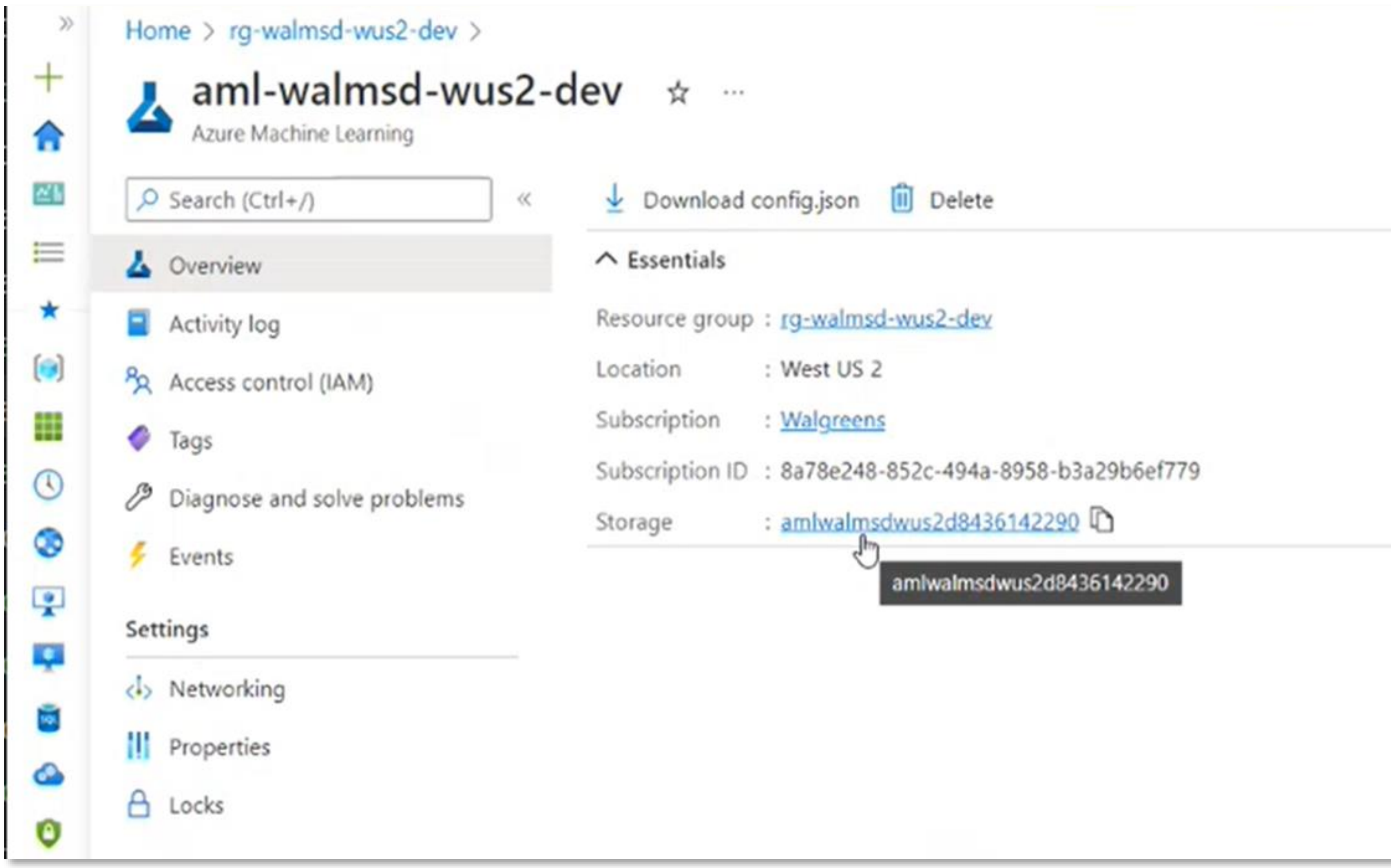
There are three settings:

(a). connection_string: it contains account name, account key. Below is a snapshot (figure-8, Figure-9)

(b). Blob container ID. You can get this information from exported Azure coco json file (figure-10).

(c). Azure ML exported coco file.

Output blob images are saved as the same folder.


Home > rg-walmsd-wus2-dev >

aml-walmsd-wus2-dev

Azure Machine Learning

Resource group : rg-walmsd-wus2-dev

Location : West US 2

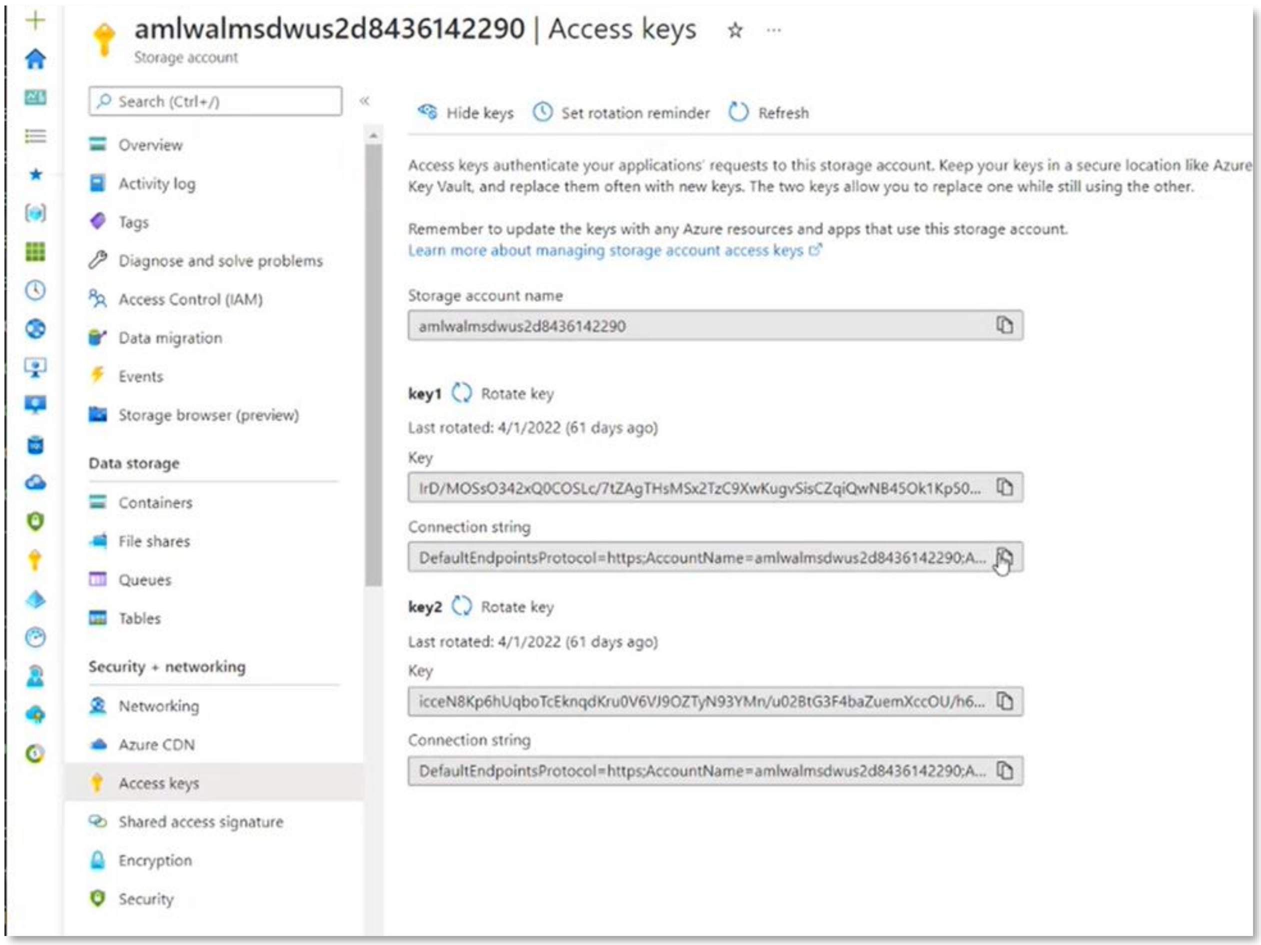
Subscription : Walgreens

Subscription ID : 8a78e248-852c-494a-8958-b3a29b6ef779

Storage : amlwalmsdwus2d8436142290

amlwalmsdwus2d8436142290

Figure-8. AccountName = amlwalmsdwus2d8436142290



Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

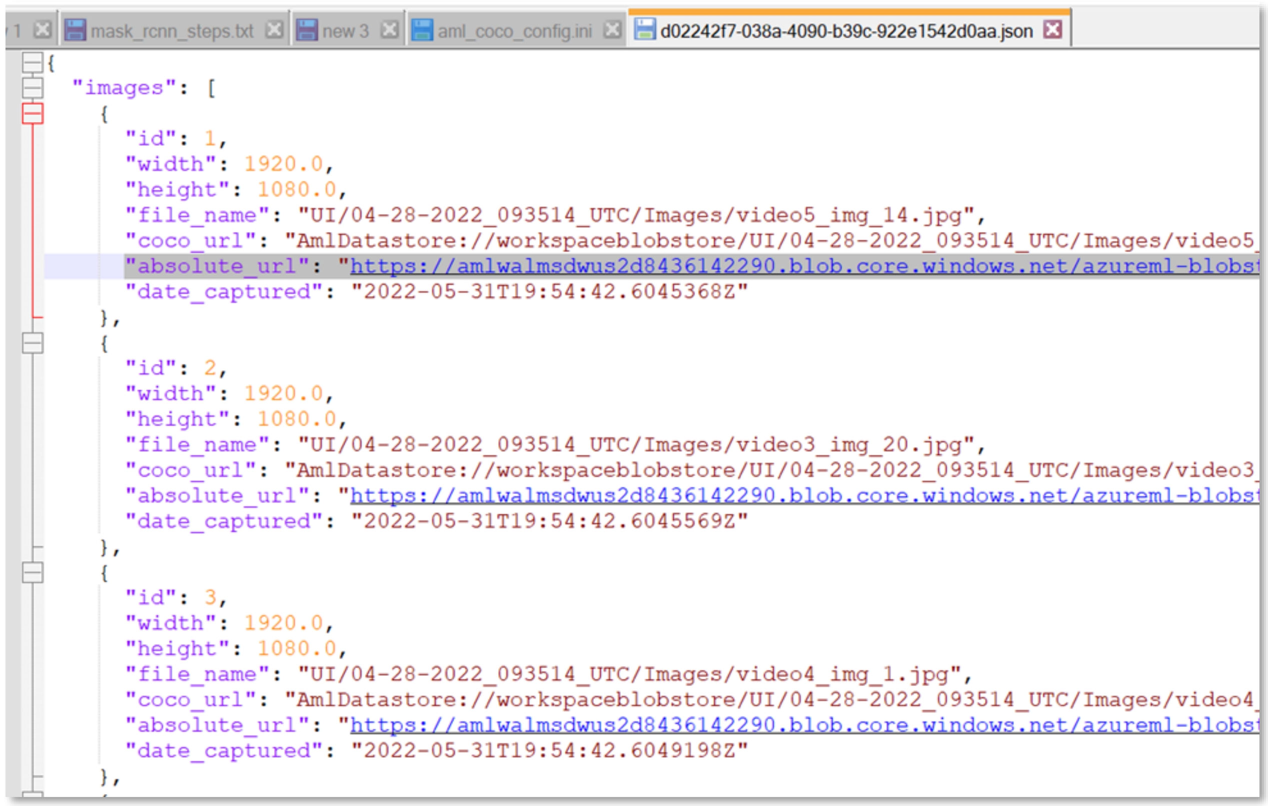
Remember to update the keys with any Azure resources and apps that use this storage account.
Learn more about managing storage account access keys

Storage account name
amlwalmsdwus2d8436142290

key1 Rotate key
Last rotated: 4/1/2022 (61 days ago)
Key
IrD/MOSsO342xQ0COSLc/7tZAgTHsMSx2TzC9XwKugvSisCZqiQwNB45Ok1Kp50...
Connection string
DefaultEndpointsProtocol=https;AccountName=amlwalmsdwus2d8436142290;A...

key2 Rotate key
Last rotated: 4/1/2022 (61 days ago)
Key
icceN8Kp6hUqboTcEknqdKru0V6VJ9OZTyN93YMn/u02BtG3F4baZuemXccOU/h6...
Connection string
DefaultEndpointsProtocol=https;AccountName=amlwalmsdwus2d8436142290;A...

Figure-9. Access key



```
{
  "images": [
    {
      "id": 1,
      "width": 1920.0,
      "height": 1080.0,
      "file_name": "UI/04-28-2022_093514_UTC/Images/video5_img_14.jpg",
      "coco_url": "AmlDatastore://workspaceblobstore/UI/04-28-2022_093514_UTC/Images/video5
      "absolute_url": "https://amlwalmsdwus2d8436142290.blob.core.windows.net/azureml-blobst
      "date_captured": "2022-05-31T19:54:42.6045368Z"
    },
    {
      "id": 2,
      "width": 1920.0,
      "height": 1080.0,
      "file_name": "UI/04-28-2022_093514_UTC/Images/video3_img_20.jpg",
      "coco_url": "AmlDatastore://workspaceblobstore/UI/04-28-2022_093514_UTC/Images/video3
      "absolute_url": "https://amlwalmsdwus2d8436142290.blob.core.windows.net/azureml-blobst
      "date_captured": "2022-05-31T19:54:42.6045569Z"
    },
    {
      "id": 3,
      "width": 1920.0,
      "height": 1080.0,
      "file_name": "UI/04-28-2022_093514_UTC/Images/video4_img_1.jpg",
      "coco_url": "AmlDatastore://workspaceblobstore/UI/04-28-2022_093514_UTC/Images/video4
      "absolute_url": "https://amlwalmsdwus2d8436142290.blob.core.windows.net/azureml-blobst
      "date_captured": "2022-05-31T19:54:42.6049198Z"
    },
```

Figure-10. Blob container id on Azure exported coco file.

4. Split all input data set into test, train and val subsets

split_aml_exported_coco.py implements all input data into three subsets such as test, train and val.

If you set train percentage is 0.80, then train set = 80%, validation set =10% and test set = 10% (Figure-11).

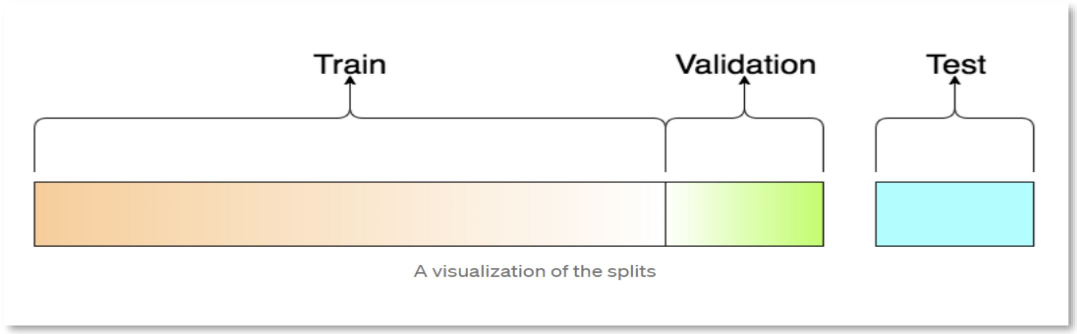


Figure-11. Train, validation and test sets

```
    rm -rf ./workspace/data/maskrcnn_images
    source ~/deepstream/deepstream_env/bin/activate
    sudo python3 split_aml_exported_coco.py ./workspace/data/inputs/d02242f7-
038a-4090-b39c-922e1542d0aa.json
./workspace/data/inputs  ./workspace/models/mask_rcnn/workspace/data/outputs --
train_pct 0.8
```

Inputs/Outputs parameters:

- input directory with all image files and json file annotations
- output directory to store your train, test, and validation sets
- percentage of data to use for training set. Default is .9

Below are the input/output file structures on train, val and test subset:

- Input file structure
  ./workspace/data/inputs/aml-exported-coco.json

- Output file structure
  ./workspace/models/mask_rcnn/workspace/data/outputs

  - for test
    ./workspace/models/mask_rcnn/workspace/data/outputs/test/annotation.json

  - test->images:
    ./workspace/models/mask_rcnn/workspace/data/outputs/test/images

  - for train
    ./workspace/models/mask_rcnn/workspace/data/outputs/train/annotation.json

  - train->images:
    ./workspace/models/mask_rcnn/workspace/data/outputs/train/images

  - for val
    workspace/models/mask_rcnn/workspace/data/outputs/val/annotation.json

  - val->images:
    ./workspace/models/mask_rcnn/workspace/data/outputs/val/images

5. Convert labels into TFRecords format for TAO Toolkit

Run the "./workspace/models/mask_rcnn/**process-labels-aml.sh**" script (see below) found in the workspace/models/mask_rcnn folder.

This script runs create_coco_tf_record.py to convert the annotation COCO format to TFRecords format.

```
PWD="$(pwd)"
BASE_DIR="${PWD}/workspace/data/inputs"

OUT_DIR="${PWD}/workspace/data/outputs"
TRAIN_DIR="${OUT_DIR}/train"
VAL_DIR="${OUT_DIR}/val"
TEST_DIR="${OUT_DIR}/test"

echo "Cloning Tensorflow models directory (for conversion utilities)"
if [ ! -e tf-models ]; then
  git clone http://github.com/tensorflow/models tf-models
fi

(cd tf-models/research && protoc object_detection/protos/*.proto --python_out=.)

# Setup packages
touch tf-models/__init__.py
touch tf-models/research/__init__.py
source ~/deepstream/deepstream_env/bin/activate

TRAIN_IMAGE_DIR="${TRAIN_DIR}/images"
VAL_IMAGE_DIR="${VAL_DIR}/images"
TEST_IMAGE_DIR="${TEST_DIR}/images"
TRAIN_COCO_ANNOTATION_FILE="${TRAIN_DIR}/annotations.json"
VAL_ANNOTATION_FILE="${VAL_DIR}/annotations.json"
TESTDEV_ANNOTATION_FILE="${TEST_DIR}/annotations.json"

OUTPUT_DIR="${OUT_DIR}/tfrecords"

echo "output folder:"
echo $OUTPUT_DIR

SCRIPT_DIR=$(dirname "$(readlink -f "$0")")
```

```
PYTHONPATH="tf-models:tf-models/research" python
$SCRIPT_DIR/create_coco_tf_record.py \
  --logtostderr \
  --include_masks \
  --train_image_dir="${TRAIN_IMAGE_DIR}" \
  --val_image_dir="$VAL_IMAGE_DIR" \
  --test_image_dir="${TEST_IMAGE_DIR}" \
  --train_object_annotations_file="${TRAIN_COCO_ANNOTATION_FILE}" \
  --val_object_annotations_file="${VAL_ANNOTATION_FILE}" \
  --testdev_annotations_file="${TESTDEV_ANNOTATION_FILE}" \
  --output_dir="${OUTPUT_DIR}" \
```

6. Train model

After TFRecords data set is ready, you can start to train model using TAO toolkit container. You can find the detailed description from this link: https://docs.nvidia.com/tao/tao-toolkit/text/instance_segmentation/mask_rcnn.html.

```
train-mrcnn-adhoc:
    sudo rm -rf ~/repos/manufacturing-
demo/workspace/models/mask_rcnn/experiment_unpruned
    sudo mkdir ~/repos/manufacturing-
demo/workspace/models/mask_rcnn/experiment_unpruned
    source ~/deepstream/deepstream_env/bin/activate
    tao mask_rcnn train -e ~/repos/manufacturing-
demo/workspace/models/mask_rcnn/maskrcnn_train_resnet50.txt \
                        -d ~/repos/manufacturing-
demo/workspace/models/mask_rcnn2/experiment_unpruned \
                        -k nvidia_tlt \
                        --gpus 1
```

Training Configuration:

(a). "./workspace/models/mask_rcnn/**maskrcnn_train_resnet50.txt"**

(b). input data set:

(i). training_file_pattern: "./workspace/models/mask_rcnn/workspace/data/outputs/tfrecords/**train*.tfrecord**"

(ii). validation_file_pattern:
"./workspace/models/mask_rcnn/workspace/data/outputs/tfrecords/**val*.tfrecord**"

(iii). val_json_file:
"./workspace/models/mask_rcnn/workspace/data/outputs/val/**annotations.json**"

(c). Output directory:
“./workspace/models/mask_rcnn**/experiment_unpruned”**