Cloud AutoML Vision

Edge containers tutorial

Terminology: See the AutoML Vision Edge terminology (https://cloud.google.com/vision/automl/docs/terminology) page for a list of terms used in this tutorial.

After creating an AutoML Vision Edge model and exporting it to a Google Cloud Storage bucket you can use RESTful services with your **AutoML Vision Edge models** and **TF Serving Docker images**.

What you will build

Docker containers can help you deploy edge models easily on different devices. You can run edge models by calling REST APIs from containers with any language you prefer, with the added benefit of not having to install dependencies or find proper TensorFlow versions.

In this tutorial, you will have a step-by-step experience of running edge models on devices using Docker containers.

Specifically, this tutorial will walk you through three steps:

1. Getting pre-built containers.
2. Running containers with Edge models to start REST APIs.
3. Making predictions.

Many devices only have CPUs, while some might have GPUs to get faster predictions. So, we provide tutorials with both pre-built CPU and GPU containers.

Objectives

In this introductory, end-to-end walkthrough you will use code samples to:

1. Get the Docker container.
2. Start REST APIs using Docker containers with edge models.

3. Make predictions to get analyzed results.

Before you begin

To complete this tutorial, you must:

1. **Train an exportable Edge model.** Follow the Edge device model quickstart (https://cloud.google.com/vision/automl/docs/edge-quickstart) to train an Edge model.
2. **Export** (#export-model) **an AutoML Vision Edge model.** This model will be served with containers as REST APIs.
3. **Install** (#install-docker) **Docker.** This is the required software to run Docker containers.
4. (Optional) **Install** (#install-nvidia) **NVIDIA docker and driver.** This is an optional step if you have devices with GPUs and would like to get faster predictions.
5. **Prepare test images.** These images will be sent in requests to get analyzed results.

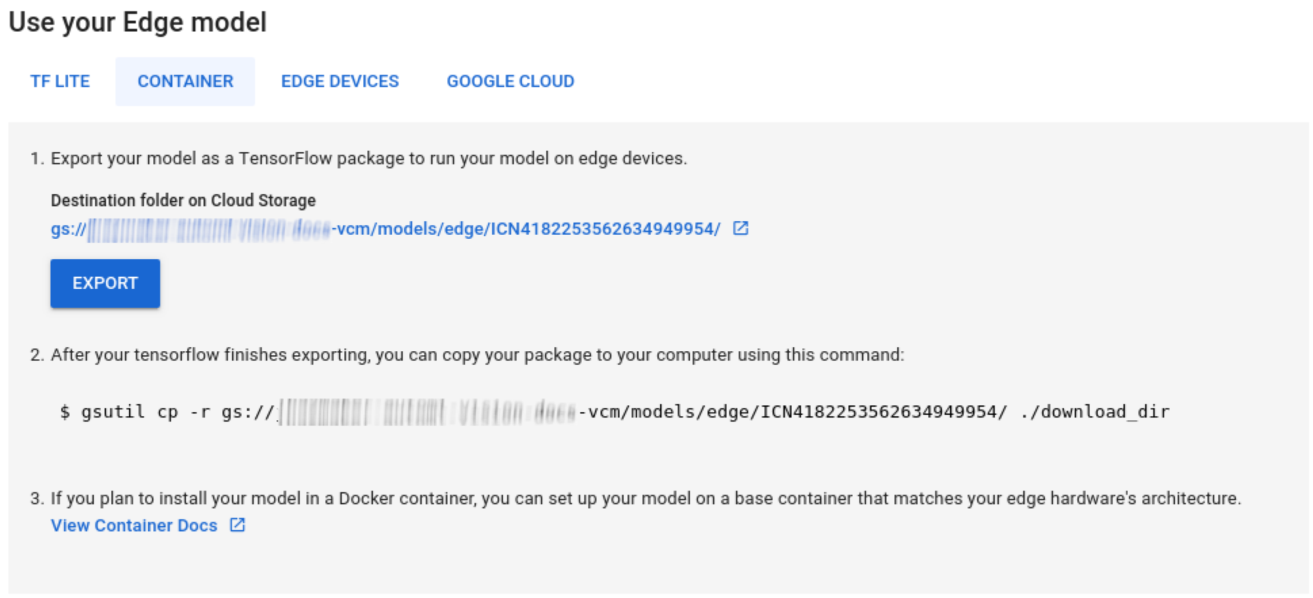
Details for exporting models and installing necessary software are in the following section.

Export AutoML Vision Edge Model

After training an Edge model, you can export it to different devices.

The containers support TensorFlow models (https://www.tensorflow.org/guide/extend/model_files), which are named `saved_model.pb` on export.

To export a AutoML Vision Edge model for containers, select the **Container** tab in the UI and then export the model to ${***YOUR_MODEL_PATH***} on Google Cloud Storage. This exported model will be served with containers as REST APIs later.



1. Export your model as a TensorFlow package to run your model on edge devices.

   Destination folder on Cloud Storage
   gs://[redacted]-vcm/models/edge/ICN4182253562634949954/

2. After your tensorflow finishes exporting, you can copy your package to your computer using this command:

```
$ gsutil cp -r gs://[redacted]-vcm/models/edge/ICN4182253562634949954/ ./download_dir
```

3. If you plan to install your model in a Docker container, you can set up your model on a base container that matches your edge hardware's architecture.

To download the exported model locally, run the following command.

Where:

- ${***YOUR_MODEL_PATH***} - The model location on Google Cloud Storage (for example, `gs://my-bucket-vcm/models/edge/ICN4245971651915048908/2020-01-20_01-27-14-064_tf-saved-model/`)
- ${***YOUR_LOCAL_MODEL_PATH***} - Your local path where you want to download your model (for example, `/tmp`).

```
gsutil cp ${YOUR_MODEL_PATH} ${YOUR_LOCAL_MODEL_PATH}/saved_model.pb
```

Install Docker

Docker (https://www.docker.com/) is software used for deploying and running applications inside containers.

Install Docker Community Edition (CE) (https://docs.docker.com/install/) on your system. You will use this to serve Edge models as REST APIs.

Install NVIDIA Driver And NVIDIA DOCKER (optional - for GPU only)

Some devices have GPUs to provide faster predictions. The GPU docker container is provided supporting NVIDIA GPUs.

In order to run GPU containers, you must install the NVIDIA driver (https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#ubuntu-installation) and NVIDIA Docker (https://github.com/NVIDIA/nvidia-docker) on your system.

Running model inference using CPU

This section gives step-by-step instructions to run model inferences using CPU containers. You will use the installed Docker to get and run the CPU container to serve the exported Edge models as REST APIs, and then send requests of a test image to the REST APIs to get analyzed results.

Pull the Docker image

First, you will use Docker to get a pre-built CPU container. The pre-built CPU container already has the whole environment to serve exported Edge models, which does *not* yet contain any Edge models.

The pre-built CPU container is stored in Google Container Registry. Before requesting the container, set an environment variable for the container's location in Google Container Registry:

```
export CPU_DOCKER_GCS_PATH=gcr.io/automl-vision-ondevice/gcloud-container-1.12.0:lat
```

After setting the environment variable for the Container Registry path, run the following command line to get the CPU container:

```
sudo docker pull ${CPU_DOCKER_GCS_PATH}
```

Run the Docker container

After getting the existing container you will run this CPU container to serve Edge model inferences with REST APIs.

Before starting the CPU container you must set system variables:

- ${***CONTAINER_NAME***} - A string indicating the container name when it runs, for example `CONTAINER_NAME=automl_high_accuracy_model_cpu`.
- ${***PORT***} - A number indicating the port in your device to accept REST API calls later, such as `PORT=8501`.

Note: Neither ${***CONTAINER_NAME***} nor ${***PORT***} should be used or occupied.

After setting the variables, run Docker in command line to serve Edge model inferences with REST APIs:

```
sudo docker run --rm --name ${CONTAINER_NAME} -p ${PORT}:8501 -v ${YOUR_MODEL_PATH}:
```

After the container is running successfully, the REST APIs are ready for serving at `http://localhost:${PORT}/v1/models/default:predict`. The following section details how to send requests for prediction to this location.

Send a prediction request

Now that the container is running successfully, you can send a prediction request on a test image to the REST APIs.

COMMAND-LINE PYTHON

The command line request body contains base64-encoded `image_bytes` and a string `key` to identify the given image. See the Base64 encoding (https://cloud.google.com/vision/automl/docs/base64) topic for more information about image encoding. The format of the request JSON file is as follows:

`/tmp/request.json`

```
{
  "instances":
  [
    {
      "image_bytes":
      {
        "b64": "/9j/7QBEUGhvdG9zaG9...base64-encoded-image-content...fXNWzvDEeYxxxz
      },
      "key": "your-chosen-image-key"
    }
  ]
}
```

After you have created a local JSON request file you can send your prediction request.

Use the following command to send the prediction request:

```
curl -X POST -d  @/tmp/request.json http://localhost:${PORT}/v1/models/default:pred
```

Response

You should see output similar to the following:

```
{
    "predictions": [
        {
            "labels": ["Good", "Bad"],
            "scores": [0.665018, 0.334982]
        }
    ]
}
```

Run Model Inference Using GPU Containers (optional)

This section shows how to run model inferences using GPU containers. This process is very similar to running model inference using a CPU. The key differences are the GPU container path and how you start GPU containers.

Pull the Docker image

First, you will use Docker to get a pre-built GPU container. The pre-built GPU container already has the environment to serve exported Edge models with GPUs, which does not yet contain any Edge models, or the drivers.

The pre-built CPU container is stored in Google Container Registry. Before requesting the container, set an environment variable for the container's location in Google Container Registry:

```
export GPU_DOCKER_GCS_PATH=gcr.io/automl-vision-ondevice/gcloud-container-1.12.0-gpu
```

Run the following command line to get the GPU container:

```
sudo docker pull ${GPU_DOCKER_GCS_PATH}
```

Run the Docker container

This step will run the GPU container to serve Edge model inferences with REST APIs. You must install NVIDIA driver and docker as mentioned above. You also must must set the following system variables:

- ${***CONTAINER_NAME***} - A string indicating the container name when it runs, for example `CONTAINER_NAME=automl_high_accuracy_model_gpu`.
- ${***PORT***} - A number indicating the port in your device to accept REST API calls later, such as `PORT=8502`.

Note: Neither ${***CONTAINER_NAME***} nor ${***PORT***} should be used or occupied.

After setting the variables, run Docker in command line to serve Edge model inferences with REST APIs:

```
sudo docker run --runtime=nvidia --rm --name "${CONTAINER_NAME}" -v \
${YOUR_MODEL_PATH}:/tmp/mounted_model/0001 -p \
${PORT}:8501 -t ${GPU_DOCKER_GCS_PATH}
```

After the container is running successfully, the REST APIs are ready for serving in `http://localhost:${PORT}/v1/models/default:predict`. The following section details how to send requests for prediction to this location.

Send a prediction request

Now that the container is running successfully, you can send a prediction request on a test image to the REST APIs.

COMMAND-LINE PYTHON

The command line request body contains base64-encoded `image_bytes` and a string `key` to identify the given image. See the Base64 encoding (https://cloud.google.com/vision/automl/docs/base64) topic for more information about image encoding. The format of the request JSON file is as follows:

`/tmp/request.json`

```
{
  "instances":
  [
    {
      "image_bytes":
      {
        "b64": "/9j/7QBEUGhvdG9zaG9...base64-encoded-image-content...fXNWzvDEeYxxxz
      },
      "key": "your-chosen-image-key"
    }
  ]
}
```

After you have created a local JSON request file you can send your prediction request.

Use the following command to send the prediction request:

```
curl -X POST -d  @/tmp/request.json http://localhost:${PORT}/v1/models/default:pred
```

Response

You should see output similar to the following:

```
{
    "predictions": [
        {
            "labels": ["Good", "Bad"],
            "scores": [0.665018, 0.334982]
        }
    ]
}
```

Summary

In this tutorial, you have walked through running Edge models using CPU or GPU Docker containers. You can now deploy this container based solution on more devices.

What Next

- Learn more about TensorFlow generally with TensorFlow's Getting Started (https://www.tensorflow.org/tutorials) documentation.
- Learn more about Tensorflow Serving (https://www.tensorflow.org/tfx/serving/docker).
- Learn how to use TensorFlow Serving with Kubernetes (https://www.tensorflow.org/tfx/serving/serving_kubernetes).

Last updated November 20, 2019.