Triton Server Quick Start

Author:Kenny Zhou

Last Update: 2022-04-02 11:38:07

NVIDIA Docker Install

1. Add NVIDIA update source

```
$ curl -s -L https://nvidia.github.io/nvidia-
docker/gpgkey | \
  sudo apt-key add -
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
$ curl -s -L https://nvidia.github.io/nvidia-
docker/$distribution/nvidia-docker.list | \
  sudo tee /etc/apt/sources.list.d/nvidia-docker.list
$ sudo apt-get update
```

2.安装nvidia-docker2软件包并重新加载docker守护程序配置

```
# 安装nvidia-docker2软件包并重新加载docker守护程序配置
$ sudo apt-get install -y nvidia-docker2
$ sudo pkill -SIGHUP dockerd
```

yolov5转tensorrt模型 or 如何使用 Triton 推理服务器部署 yolov5 模型

1. 生成.wts文件

```
git clone -b v5.0
https://github.com/ultralytics/yolov5.git
git clone https://github.com/wang-xinyu/tensorrtx.git
```

下载模型`yolov5s.pt`模型：
https://github.com/ultralytics/yolov5/releases/download/v5.0/yolov5s.pt
这里也可以使用自己训练好的pt模型文件，本文我将采用官方模型做转换。

2 生成yolov5s.wts文件

```
cp {tensorrtx}/yolov5/gen_wts.py {ultralytics}/yolov5
cd {ultralytics}/yolov5
python gen_wts.py -w yolov5s.pt
```

3 Install Cuda Toolkit

```
wget
https://developer.download.nvidia.com/compute/cuda/repos/
ubuntu2004/x86_64/cuda-ubuntu2004.pinsudo mv cuda-
ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-
pin-600
wget
https://developer.download.nvidia.com/compute/cuda/11.6.1
/local_installers/cuda-repo-ubuntu2004-11-6-local_11.6.1-
510.47.03-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu2004-11-6-local_11.6.1-
510.47.03-1_amd64.deb
sudo apt-key add /var/cuda-repo-ubuntu2004-11-6-
local/7fa2af80.pub
sudo apt-get update
sudo apt-get -y install cuda
```

4 安装依赖

```
sudo apt updatesudo apt install libopencv-dev python3-
opencv
```

5 build

```
cd {tensorrtx}/yolov5/
mkdir build
cd build
cp {ultralytics}/yolov5/yolov5s.wts
{tensorrtx}/yolov5/build

cmake ..
make
```

debug

Jetson

Info

nvidia

pw:nvidia

ip address:192.168.55.1

share network

~~Install~~

~~Server~~

~~Triton Inference Server Support for Jetson and JetPack~~

~~Client~~

~~Triton Client Libraries and Examples~~

（emmc2ssd）

Jetson Install Triton-inference-server

安装之前 需要

```
 python3 -m pip install -U pip
```

```
python3 -m pip install --upgrade
clients/python/tritonclient-2.19.0-py3-none-
manylinux2014_aarch64.whl[all]
```

Jetson Jetpack 支持

JetPack 4.6.1的 Triton 版本在随附的 tar 文件中提供：tritonserver2.19.0-jetpack4.6.1.tgz。

- 此版本支持**TensorFlow** 2.7.0、**TensorFlow** 1.15.5、**TensorRT** 8.2.1.8、**Onnx Runtime** 1.10.0、**PyTorch** 1.11.0、**Python** 3.6 以及

集成。

- Onnx 运行时后端不支持 OpenVino 执行提供程序。但是支持 TensorRT 执行提供程序。
- Jetson 支持系统共享内存。不支持 CUDA 共享内存。
- 不支持 GPU 指标、GCS 存储、S3 存储和 Azure 存储。

tar 文件包含 Triton 服务器可执行和共享库以及 C++ 和 Python 客户端库和示例。有关如何在 JetPack 上安装和使用 Triton 的更多信息，请参阅jetson.md。

Python 客户端库的轮子存在于 tar 文件中，可以通过运行以下命令来安装：

TIS on Jetson Quick Start

server

1. git clone https://github.com/triton-inference-server/server
2. model-repository 即为server/docs/examples/model_repository
3. `~/server/docs/examples$ sh fetch_models.sh`
4. 解压缩tritonserver2.19.0-jetpack4.6.1.tgz

   `mkdir ~/TritonServer && tar -xzvf tritonserver2.19.0-jetpack4.6.1.tgz -C ~/TritonServer`
5. 在文件夹中

```
~/TIS/bin$ ./tritonserver --model-repository=/home/nvidia/server/docs/examples/model_repository --backend-directory=/home/nvidia/TIS/backends (--strict-model-config=false)
```

6. 自动生成的模型配置

   默认情况下，每个模型都必须提供包含所需设置的模型配置文件。但是，如果 Triton 使用 --strict-model-config=false 选项启动，那么在某些情况下，模型配置文件的所需部分可以由 Triton 自动生成。模型配置的必需部分是最小模型配置中显示的那些设置。具体来说，TensorRT、TensorFlow 保存模型和 ONNX 模型不需要模型配置文件，因为 Triton 可以自动导出所有需要的设置。所有其他模型类型必须提供模型配置文件。

   使用 --strict-model-config=false 时，您可以查看使用模型配置端点为模型生成的模型配置。最简单的方法是使用*curl*之类的实用程序：

   ```
   $ curl localhost:8000/v2/models/<模型名称>/config
   ```

   这将返回生成的模型配置的 JSON 表示。您可以从中获取 JSON 的 max_batch_size、输入和输出部分，并将其转换为 config.pbtxt 文件。Triton 只生成模型配置的最小部分。您仍然必须通过编辑 config.pbtxt 文件来提供模型配置的可选部分。

PS: if error with libb64.so.0d: `sudo apt-get install libb64-0d`

and error with libre4.so.4: `sudo apt-get install libre2-dev`

PPS:if error: creating server: Internal - failed to load all models

```
You can try using --exit-on-error=false when
launching the server.
```

Use Triton's *ready* endpoint to verify that the server and the models are ready for inference. From the host system use curl to access the HTTP endpoint that indicates server status.

```
$ curl -v localhost:8000/v2/health/ready
...
< HTTP/1.1 200 OK
< Content-Length: 0
< Content-Type: text/plain
```

client

使用 docker pull 从 NGC 获取客户端库和示例图像。

```
$ docker pull nvcr.io/nvidia/tritonserver:<xx.yy>-py3-sdk
```

其中 <xx.yy> 是您要提取的版本（22.02）。运行客户端映像。

```
$ docker run -it --rm --net=host
nvcr.io/nvidia/tritonserver:<xx.yy>-py3-sdk
```

From within the nvcr.io/nvidia/tritonserver:<xx.yy>-py3-sdk image, run the example image-client application to perform image classification using the example densenet_onnx model.

To send a request for the densenet_onnx model use an image from the /workspace/images directory. In this case we ask for the top 3 classifications.

```
$ /workspace/install/bin ./image_client -m densenet_onnx
-c 3 -s INCEPTION /workspace/images/mug.jpg
Request 0, batch size 1
Image '/workspace/images/mug.jpg':
    15.346230 (504) = COFFEE MUG
    13.224326 (968) = CUP
    10.422965 (505) = COFFEEPOT
```

-i flag to use GRPC:

```
/workspace/install/bin# ./image_client -m densenet_onnx -
c 3 -s INCEPTION /workspace/images/mug.jpg -i grpc
```

-u flag to use specific host and port:

```
/workspace/install/bin# ./image_client -m densenet_onnx -
c 3 -s INCEPTION /workspace/images/mug.jpg -u
192.168.4.138:8000
```

so like that:

```
root@kenny-System-Product-Name:/workspace/install/bin#
./image_client -m densenet_onnx -c 3 -s INCEPTION
/workspace/images/mug.jpg -u 192.168.4.138:8001 -i grpc
Request 0, batch size 1
Image '/workspace/images/mug.jpg':
    15.349568 (504) = COFFEE MUG
    13.227467 (968) = CUP
    10.424896 (505) = COFFEEPOT
```

PS: http is port 8000 gRPC is 8001

Send images to server for inference with Node-RED

在 Node-RED 中使用 TensorFlow

Configure NVIDIA Triton Inference Server on different platforms. Deploy object detection model in Tensorflow SavedModel format to server. Send images to server for inference with Node-RED. Triton Inference Server HTTP API is used for inference.

Software	Version	Link
Jetpack	4.6	https://developer.nvidia.com/embedded/downloads
Triton	2.17.0	https://github.com/triton-inference-server/server/releases

Other Jetson´s might work too，like Jetpack == 4.6.1,Triton == 2.19.0

1. Install official Jetpack 4.6

https://developer.nvidia.com/embedded/learn/get-started-jetson-xavier-nx-devkit

2. Download and extract Triton Inference Server package

```
mkdir /home/$USER/triton_server
cd /home/$USER/triton_server
wget https://jetson-nodered-files.s3.eu.cloud-object-
storage.appdomain.cloud/tritonserver2.17.0-jetpack4.6.tgz
tar xzvf tritonserver2.17.0-jetpack4.6.tgz
```

3. Create model repository

Download example model repository with pre-installed Tensorflow savedmodel.

Example model is SSD MobileNet v2 320x320 model from TensorFlow 2 Detection Model Zoo

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Model is trained to detect usual things like person, cat, dog, teddy bear etc..

```
wget https://jetson-nodered-files.s3.eu.cloud-object-
storage.appdomain.cloud/model_repository.zip
unzip model_repository.zip
```

4. Install dependencies

```
sudo apt-get update &&
sudo apt-get install -y --no-install-recommends software-
properties-common &&
sudo apt-get install -y --no-install-recommends autoconf
&&
```

```
sudo apt-get install -y --no-install-recommends automake
&&
sudo apt-get install -y --no-install-recommends build-
essential &&
sudo apt-get install -y --no-install-recommends cmake &&
sudo apt-get install -y --no-install-recommends git &&
sudo apt-get install -y --no-install-recommends libb64-
dev &&
sudo apt-get install -y --no-install-recommends libre2-
dev &&
sudo apt-get install -y --no-install-recommends libssl-
dev &&
sudo apt-get install -y --no-install-recommends libtool
&&
sudo apt-get install -y --no-install-recommends libboost-
dev &&
sudo apt-get install -y --no-install-recommends libcurl4-
openssl-dev &&
sudo apt-get install -y --no-install-recommends
libopenblas-dev &&
sudo apt-get install -y --no-install-recommends
rapidjson-dev &&
sudo apt-get install -y --no-install-recommends patchelf
&&
sudo apt-get install -y --no-install-recommends zlib1g-
dev
```

5. Start Triton Inference Server

```
cd /home/$USER/triton_server/bin
./tritonserver --model-
repository=/home/$USER/triton_server/model_repository --
backend-directory=/home/$USER/triton_server/backends --
backend-config=tensorflow,version=2 --strict-model-
config=false
```

6. Autostart on boot

Create file 'triton.service' to folder /etc/systemd/system

```
sudo nano /etc/systemd/system/triton.service
```

Example file content, correct your paths if needed.

```
[Unit]
Description=NVIDIA Triton Inference Server

Wants=network.target
After=syslog.target network-online.target

[Service]
Type=simple
ExecStart=/home/nvidia/TritonServer/bin/tritonserver --
model-
repository=/home/nvidia/server/docs/examples/model_reposi
tory --backend-
directory=/home/nvidia/TritonServer/backends --backend-
con$
Restart=on-failure
RestartSec=10
```

```
KillMode=process

[Install]
WantedBy=multi-user.target
```

Enable service

```
sudo systemctl enable triton.service
```

Start service

```
sudo systemctl start triton.service
```

and

```
sudo npm install -g --unsafe-perm node-red
cd /.node-red
npm install node-red-contrib-image-info
npm install node-red-node-exif
npm install node-red-contrib-browser-utils
npm install node-red-contrib-image-output
npm install node-red-contrib-moment

node-red
```

and in node-red website import :

▶ need import code

```
[{"id":"83a7a965.1808a8","type":"subflow","name":"[IMG]
Annotate","info":"","category":"Tequ-API Client","in":
[{"x":120,"y":140,"wires":[{"id":"d05bfd8e.a02e"}]}],"out":
[{"x":1080,"y":140,"wires":
[{"id":"4e5f5c6c.bcf214","port":0}]}],"env":
[{"name":"box_colors","type":"json","value":"
{\"fish\":\"#FFFFFF\",\"pike\":\"#006400\",\"perch\":\"#0080
00\",\"smolt\":\"#ADD8E6\",\"salmon\":\"#0000FF\",\"trout\":
\"#0000FF\",\"cyprinidae\":\"#808080\",\"zander\":\"#009000\
",\"bream\":\"#008800\"}","ui":{"type":"input","opts":
{"types":["json"]}}},
{"name":"image_settings","type":"json","value":"
{\"quality\":0.8}","ui":{"type":"input","opts":{"types":
["json"]}}},
{"name":"image_type","type":"str","value":"image/jpeg","ui":
{"type":"select","opts":{"opts":[{"l":{"en-
US":"JPG"},"v":"image/jpeg"},{"l":{"en-
US":"PNG"},"v":"image/png"}]}}},
{"name":"bbox_lineWidth","type":"num","value":"5","ui":
{"type":"spinner","opts":{"min":0,"max":10}}},
{"name":"bbox_text_color","type":"str","value":"white","ui":
{"type":"select","opts":{"opts":[{"l":{"en-
US":"white"},"v":"white"},{"l":{"en-
US":"black"},"v":"black"},{"l":{"en-US":"blue"},"v":"blue"},
{"l":{"en-US":"green"},"v":"green"},{"l":{"en-
US":"yellow"},"v":"yellow"},{"l":{"en-US":"red"},"v":"red"},
{"l":{"en-US":"orange"},"v":"orange"}]}}},
{"name":"bbox_font","type":"str","value":"30px Arial","ui":
{"type":"select","opts":{"opts":[{"l":{"en-US":"5px
Arial"},"v":"5 px Arial"},{"l":{"en-US":"10px
```

```
Arial"},"v":"10px Arial"},{"l":{"en-US":"15px
Arial"},"v":"15px Arial"},{"l":{"en-US":"20px
Arial"},"v":"20px Arial"},{"l":{"en-US":"25px
Arial"},"v":"25px Arial"},{"l":{"en-US":"30px
Arial"},"v":"30px Arial"},{"l":{"en-US":"35px
Arial"},"v":"35px Arial"},{"l":{"en-US":"40px
Arial"},"v":"40px Arial"},{"l":{"en-US":"45px
Arial"},"v":"45px Arial"},{"l":{"en-US":"50px
Arial"},"v":"50px Arial"}]}}},
{"name":"label_offset_x","type":"num","value":"0","ui":
{"type":"input","opts":{"types":["num"]}}},
{"name":"label_offset_y","type":"num","value":"30","ui":
{"type":"input","opts":{"types":["num"]}}},
{"name":"threshold","type":"num","value":"0.75","ui":
{"type":"spinner","opts":{"min":0,"max":1}}},
{"name":"labels","type":"json","value":"[\"fish\",\"perch\",
\"pike\", \"rainbow trout\", \"salmon\", \"trout\",
\"cyprinidae\", \"zander\", \"smolt\", \"bream\"]","ui":
{"type":"input","opts":{"types":["json"]}}}],"meta":
{"module":"[IMG]
Annotate","version":"0.0.1","author":"juha.autioniemi@lapina
mk.fi","desc":"Annotates prediction results from [AI] Detect
subflows.","license":"MIT"},"color":"#87A980","icon":"font-
awesome/fa-pencil-square-o","status":
{"x":1080,"y":280,"wires":
[{"id":"7fd4f6bf24348b12","port":0}]}},
{"id":"c19ac6bd.2a9d08","type":"function","z":"83a7a965.1808
a8","name":"Annotate with  canvas","func":"const img =
msg.payload.image.buffer;\nconst image_type =
env.get(\"image_type\");\nconst image_settings =
```

```
env.get(\"image_settings\");\nconst bbox_lineWidth =
env.get(\"bbox_lineWidth\");\nconst bbox_text_color =
env.get(\"bbox_text_color\");\nconst label_offset_x =
env.get(\"label_offset_x\");\nconst label_offset_y =
env.get(\"label_offset_y\");\nconst bbox_font =
env.get(\"bbox_font\");\nconst COLORS =
env.get(\"box_colors\");\nconst objects =
msg.payload.inference.result\nconst labels =
env.get(\"labels\")\n\n//Define threshold\nlet threshold =
0;\n\nconst global_settings = global.get(\"settings\") ||
undefined\nlet thresholdType = \"\"\n\nif(global_settings
!== undefined){\n    if(\"threshold\" in global_settings){\n
threshold = global_settings[\"threshold\"]\n
thresholdType = \"global\";\n    }\n}\n\nelse
if(\"threshold\" in msg){\n    threshold = msg.threshold;\n
thresholdType = \"msg\";\n    if(threshold < 0){\n
threshold = 0\n    }\n    else if(threshold > 1){\n
threshold = 1\n    }\n}\n\nelse{\n    threshold =
env.get(\"threshold\");\n    thresholdType =
\"env\";\n}\n\nmsg.thresholdUsed =
threshold;\nmsg.thresholdTypeUsed = thresholdType;\n\nasync
function annotateImage(image) {\n  const localImage = await
canvas.loadImage(image);  \n  const cvs =
canvas.createCanvas(localImage.width, localImage.height);\n
const ctx = cvs.getContext('2d');  \n
ctx.drawImage(localImage, 0, 0); \n  \n
objects.forEach((obj) => {\n
if(labels.includes(obj.class) && obj.score >= threshold){\n
let [x, y, w, h] = obj.bbox;\n            ctx.lineWidth =
bbox_lineWidth;\n            ctx.strokeStyle =
```

```
COLORS[obj.class];\n            ctx.strokeRect(x, y, w,
h);\n            ctx.fillStyle = bbox_text_color;\n
ctx.font = bbox_font;\n            ctx.fillText(obj.class+\"
\"+Math.round(obj.score*100)+\"
%\",x+label_offset_x,y+label_offset_y);\n        }\n
});\n  \n  return cvs.toBuffer(image_type,
image_settings);\n}\n\nif(objects.length > 0){\n
msg.annotated_image = await annotateImage(img)\n
//node.done()\n    msg.objects_found = true\n}\nelse{\n
msg.objects_found = false\n}\n\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","l
ibs":
[{"var":"canvas","module":"canvas"}],"x":440,"y":140,"wires"
:[["a801355d.9f7ac8"]]},
{"id":"d05bfd8e.a02e","type":"change","z":"83a7a965.1808a8",
"name":"timer","rules":
[{"t":"set","p":"start","pt":"msg","to":"","tot":"date"}],"a
ction":"","property":"","from":"","to":"","reg":false,"x":23
0,"y":140,"wires":[["c19ac6bd.2a9d08"]]},
{"id":"a801355d.9f7ac8","type":"change","z":"83a7a965.1808a8
","name":"end timer","rules":
[{"t":"set","p":"payload.annotation.time_ms","pt":"msg","to"
:"$millis() - msg.start","tot":"jsonata"},
{"t":"set","p":"payload.annotation.buffer","pt":"msg","to":"
annotated_image","tot":"msg"},
{"t":"set","p":"payload.annotation.objects_found","pt":"msg"
,"to":"objects_found","tot":"msg"},
{"t":"delete","p":"annotated_image","pt":"msg"},
{"t":"delete","p":"start","pt":"msg"}],"action":"","property
":"","from":"","to":"","reg":false,"x":640,"y":140,"wires":
```

```
[["4e5f5c6c.bcf214","c20a6448.e6f218"]]},
{"id":"4e5f5c6c.bcf214","type":"change","z":"83a7a965.1808a8
","name":"delete useless","rules":
[{"t":"delete","p":"annotated_image","pt":"msg"},
{"t":"delete","p":"start","pt":"msg"},
{"t":"delete","p":"objects_found","pt":"msg"}],"action":"","
property":"","from":"","to":"","reg":false,"x":880,"y":140,"
wires":[[]]},
{"id":"c20a6448.e6f218","type":"switch","z":"83a7a965.1808a8
","name":"objects
found?","property":"objects_found","propertyType":"msg","rul
es":[{"t":"true"},
{"t":"false"}],"checkall":"true","repair":false,"outputs":2,
"x":660,"y":200,"wires":[["a9379cd1321a02da"],
["0ec56ca8f000a540"]]},
{"id":"a9379cd1321a02da","type":"function","z":"83a7a965.180
8a8","name":"","func":"node.status({fill:\"green\",shape:\"d
ot\",text:msg.thresholdTypeUsed+\" \"+msg.thresholdUsed+\"
in \"+msg.payload.annotation.time_ms+\"
ms\"})","outputs":0,"noerr":0,"initialize":"","finalize":"",
"libs":[],"x":860,"y":180,"wires":[]},
{"id":"0ec56ca8f000a540","type":"function","z":"83a7a965.180
8a8","name":"","func":"node.status({fill:\"green\",shape:\"d
ot\",text:msg.thresholdTypeUsed+\" \"+msg.thresholdUsed+\"
No objects to
annotate\"})","outputs":0,"noerr":0,"initialize":"","finaliz
e":"","libs":[],"x":860,"y":220,"wires":[]},
{"id":"7fd4f6bf24348b12","type":"status","z":"83a7a965.1808a
8","name":"","scope":null,"x":860,"y":280,"wires":[[]]},
{"id":"9d7441713e5a169b","type":"subflow","name":"[AI]
```

```
Detect - Triton","info":"Send image to NVIDIA Triton
Inference server using HTTP API.\n\nModel should be
configured and loaded to server before using.\n\nSupported
models:\n - tequ-fish-species-detection\n","category":"Tequ-
API Client","in":[{"x":80,"y":140,"wires":
[{"id":"a27070060363f2df"}]}],"out":
[{"x":1120,"y":300,"wires":
[{"id":"ec5e76112f20b1cd","port":0},
{"id":"2e4abd3abccee2f6","port":1}]},
{"x":980,"y":420,"wires":
[{"id":"df34ddea04c5826b","port":0}]}],"env":
[{"name":"threshold","type":"num","value":"0.75","ui":
{"type":"spinner","opts":{"min":0,"max":1}}},
{"name":"model_name","type":"str","value":"ssd-example-
model","ui":{"type":"input","opts":{"types":
["str","env"]}}},
{"name":"triton_server_url","type":"str","value":"tequ-
jetson-nx-1:8000","ui":{"type":"input","opts":{"types":
["str"]}}},
{"name":"image_width_cm","type":"env","value":"image_width_c
m","ui":{"type":"input","opts":{"types":
["json","env"]}}}],"meta":{"module":"node-red-contrib-tequ-
ai-
triton","version":"0.0.1","author":"juha.autioniemi@lapinamk
.fi","desc":"Inference image using NVIDIA Triton Inference
server.","license":"MIT"},"color":"#FFCC66","inputLabels":
["Image buffer in"],"outputLabels":["Inference
result","Model configuration"],"icon":"node-
red/status.svg","status":{"x":980,"y":620,"wires":
[{"id":"5ac4fb48cdf1f282","port":0}]}},
```

{"id":"18efd978252e44b4","type":"function","z":"9d7441713e5a169b","name":"Pre-process","func":"const imageBuffer = msg.payload;\nconst modelName = env.get(\"model_name\")\nconst server_url = env.get(\"triton_server_url\")\n\nasync function createImageTensor(input){\n   return tf.tidy(() => {\n const tensor = tf.node.decodeJpeg(input, 3).expandDims(0);\n const shape = tensor.shape;\n   return {\"tensor\":tensor,\"shape\":shape}\n   });\n}\n\nconst image_tensor = await createImageTensor(imageBuffer)\nconst image_tensor_buffer = Buffer.from(image_tensor[\"tensor\"].dataSync());\nconst image_tensor_buffer_length = image_tensor_buffer.length;\nimage_tensor[\"tensor\"].dispose()\n\nlet inference_header = {\n   \"model_name\" : \"test\",\n   \"inputs\" : [\n   {\n \"name\" : \"input_tensor\",\n   \"shape\" : image_tensor[\"shape\"],\n   \"datatype\" : \"UINT8\",\n   \"parameters\" : {\n \"binary_data_size\":image_tensor_buffer_length\n }\n   }\n   ],\n   \"outputs\" : [\n   {\n \"name\" : \"detection_scores\",\n   \"parameters\" : {\n   \"binary_data\" : false\n   }\n },\n   {\n   \"name\" : \"detection_boxes\",\n \"parameters\" : {\n   \"binary_data\" : false\n }\n   },\n   {\n   \"name\" : \"detection_classes\",\n   \"parameters\" : {\n \"binary_data\" : false\n   }\n   }\n   ]\n }\n\nlet inference_header_buffer = Buffer.from(JSON.stringify(inference_header))\nlet

```
inference_header_length =
inference_header_buffer.length\n\nmsg.method =
\"POST\";\nmsg.payload =
Buffer.concat([inference_header_buffer,image_tensor_buffer])
\nmsg.url =
server_url+\"/v2/models/\"+modelName+\"/infer\";\nmsg.header
s = {\n    \"Content-Type\":\"application/octet-stream\",\n
\"Inference-Header-Content-
Length\":inference_header_length,\n    \"Content-
Length\":inference_header_length+image_tensor_buffer_length\
n};\n    \ntime_ms =  new Date().getTime() -
msg.start;\nmsg.pre_process_time =
time_ms;\nnode.status({fill:\"green\",shape:\"dot\",text:tim
e_ms+\" ms\"});    \nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"//
Code added here will be run when the\n// node is being
stopped or re-
deployed.\nnode.status({fill:\"blue\",shape:\"dot\",text:\"S
topping node...\"});
\ncontext.set(\"model\",undefined)\ntf.disposeVariables()","
libs":[{"var":"fs","module":"fs"},
{"var":"os","module":"os"},{"var":"zlib","module":"zlib"},
{"var":"tf","module":"@tensorflow/tfjs-
node"}],"x":210,"y":280,"wires":[["d0fddd20c79b5628"]]},
{"id":"d0fddd20c79b5628","type":"http
request","z":"9d7441713e5a169b","name":"","method":"use","re
t":"obj","paytoqs":"ignore","url":"","tls":"","persist":true
,"proxy":"","authType":"","senderr":false,"x":570,"y":280,"w
ires":[["2e4abd3abccee2f6"]]},
{"id":"ec5e76112f20b1cd","type":"function","z":"9d7441713e5a
```

```
169b","name":"Post-process","func":"let inference_result =
msg.payload;\nlet results = [];\nconst modelName =
env.get(\"model_name\")\nconst model =
flow.get(modelName);\nconst labels =
model.parameters.labels;\nconst metadata =
model.parameters.metadata;\nconst ts = msg.ts;\n\nconst
threshold = msg.threshold;\nconst image_width =
msg.width;\nconst image_height = msg.height;\nconst
originalImage = msg.image;\nlet scores;\nlet boxes;\nlet
names;\n\nfunction chunk(arr, chunkSize) {\n  if (chunkSize
<= 0) throw \"Invalid chunk size\";\n  var R = [];\n  for
(var i=0,len=arr.length; i<len; i+=chunkSize)\n
R.push(arr.slice(i,i+chunkSize));\n  return R;\n}\n\nfor(let
i=0;i<(inference_result.outputs).length;i++){\n
//node.warn(inference_result.outputs[i])\n    \n
if(inference_result.outputs[i][\"name\"] ==
\"detection_scores\"){\n        scores =
inference_result.outputs[i].data\n    }\n    else
if(inference_result.outputs[i][\"name\"] ==
\"detection_boxes\"){\n        boxes =
inference_result.outputs[i].data\n        boxes =
chunk(boxes,4)\n        \n    }\n    else
if(inference_result.outputs[i][\"name\"] ==
\"detection_classes\"){\n        names =
inference_result.outputs[i].data \n    }\n} \n\nfor (let i =
0; i < scores.length; i++) {\n    if (scores[i] > threshold)
{\n        newObject = {\n            \"bbox\":[\n
boxes[i][1] * image_width,\n                    boxes[i][0]
* image_height,\n                    (boxes[i][3] - boxes[i]
[1]) * image_width,\n                    (boxes[i][2] -
```

```
boxes[i][0]) * image_height\n            ],\n
\"class\":labels[names[i]-1],\n
\"label\":labels[names[i]-1],\n
\"score\":scores[i],\n            \"length_cm\":NaN\n
}\n        results.push(newObject)\n    }\n}\n
\n//Calculate object width if image_width_cm is given input
message\n if(\"image_width_cm\" in msg){\n    const
image_width_cm = msg.image_width_cm;    \n    for(let
j=0;j<results.length;j++){\n        px_in_cm =
image_width_cm / msg.width\n        object_size_cm =
px_in_cm * results[j].bbox[2]\n        results[j].length_cm
= Math.round(object_size_cm)\n    }\n}\n        \n// Create
output message\nlet result_message = {\n
\"labels\":labels,\n
\"thresholdType\":msg.thresholdType,\n    \"threshold\":
msg.threshold,\n    \"image_width_cm\":msg.image_width_cm,\n
\"image_width_cm_type\":msg.image_width_cm_type,\n
\"topic\":msg.topic,\n    \"payload\":{\n
\"inference\":{\n            \"metadata\":metadata,\n
\"time_ms\": new Date().getTime() - msg.start,\n
\"validated\":false,\n            \"result\":results,\n
\"type\":\"object detection\"\n        },\n
\"image\":{\n            \"buffer\":originalImage,\n
\"width\": msg.width,\n            \"height\": msg.height,\n
\"type\": msg.type,\n            \"size\":
(originalImage).length,\n            \"exif\":{}\n
}\n    }\n}\n\n// Add exif information\nif(msg.exif){\n
result_message.payload.image.exif = msg.exif\n}\n
\ntotal_ms =  new Date().getTime() - msg.start;\n
\nnode.status({fill:\"blue\",shape:\"dot\",text:
```

```
msg.pre_process_time+\" ms | \"+
(result_message.payload.inference.time_ms-
msg.pre_process_time)+\" ms |
\"+result_message.payload.inference.time_ms+\" ms\"});
\n    \nreturn
result_message;","outputs":1,"noerr":0,"initialize":"","fina
lize":"","libs":[],"x":910,"y":260,"wires":[[]]},
{"id":"bdf244aaa74449f8","type":"function","z":"9d7441713e5a
169b","name":"Set threshold &
image_width_cm","func":"//Define threshold\nlet threshold =
0;\nconst global_settings = global.get(\"settings\") ||
undefined\nlet thresholdType = \"\"\n\nif(global_settings
!== undefined){\n    if(\"threshold\" in global_settings){\n
threshold = global_settings[\"threshold\"]\n
thresholdType = \"global\";\n    }\n}\n\nelse
if(\"threshold\" in msg){\n    threshold = msg.threshold;\n
thresholdType = \"msg\";\n    if(threshold < 0){\n
threshold = 0\n    }\n    else if(threshold > 1){\n
threshold = 1\n    }\n}\n\nelse{\n    try{\n
threshold = env.get(\"threshold\");\n        thresholdType =
\"env\";\n    }\n    catch(err){\n        threshold = 0.5\n
thresholdType = \"default\";\n    }\n}\n\n\ntry{\n
image_width_cm_type = \"env\";\n    image_width_cm =
JSON.parse(env.get(\"image_width_cm\"))[msg.topic];\n
\n}\ncatch(err){\n    image_width_cm = 130\n
image_width_cm_type = \"default\";\n}\n\nif(image_width_cm
== undefined){\n    image_width_cm =
130\n}\n\n\nif(threshold == undefined){\n    threshold =
0\n}\n\nmsg.thresholdType = thresholdType;\nmsg.threshold =
threshold;\nmsg.image_width_cm =
```

```
image_width_cm;\nmsg.image_width_cm_type =
image_width_cm_type;\n//node.status({fill:\"green\",shape:\"
dot\",text:\"threshold: \"+threshold+\" | Image width:
\"+image_width_cm});\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","l
ibs":[],"x":980,"y":200,"wires":[["18efd978252e44b4"]]},
{"id":"aac6b7c6db9f9a61","type":"function","z":"9d7441713e5a
169b","name":"isBuffer?","func":"let timestamp = new
Date().toISOString();\nmsg.start = new
Date().getTime()\n\nif(Buffer.isBuffer(msg.payload)){\n
node.status({fill:\"green\",shape:\"dot\",text:timestamp +
\" OK\"});  \n    msg.image = msg.payload;\n    return
msg;\n}\nelse{\n    node.error(\"msg.payload is not an image
buffer\",msg)\n
node.status({fill:\"red\",shape:\"dot\",text:timestamp + \"
msg.payload is not an image buffer\"});  \n    return
null;\n}","outputs":1,"noerr":0,"initialize":"","finalize":"
","libs":[],"x":200,"y":200,"wires":[["a9c87fd163b03fe8"]]},
{"id":"def713aaa3aa2726","type":"exif","z":"9d7441713e5a169b
","name":"","mode":"normal","property":"payload","x":750,"y"
:200,"wires":[["bdf244aaa74449f8"]]},
{"id":"6f0b51afb2815ede","type":"image-
info","z":"9d7441713e5a169b","name":"","x":570,"y":200,"wire
s":[["def713aaa3aa2726"]]},
{"id":"71ce551ba3faaef2","type":"http
request","z":"9d7441713e5a169b","name":"http
request","method":"use","ret":"obj","paytoqs":"ignore","url"
:"","tls":"","persist":false,"proxy":"","authType":"","x":59
0,"y":440,"wires":[["df34ddea04c5826b"]]},
{"id":"e2b2f167d1e53859","type":"function","z":"9d7441713e5a
```

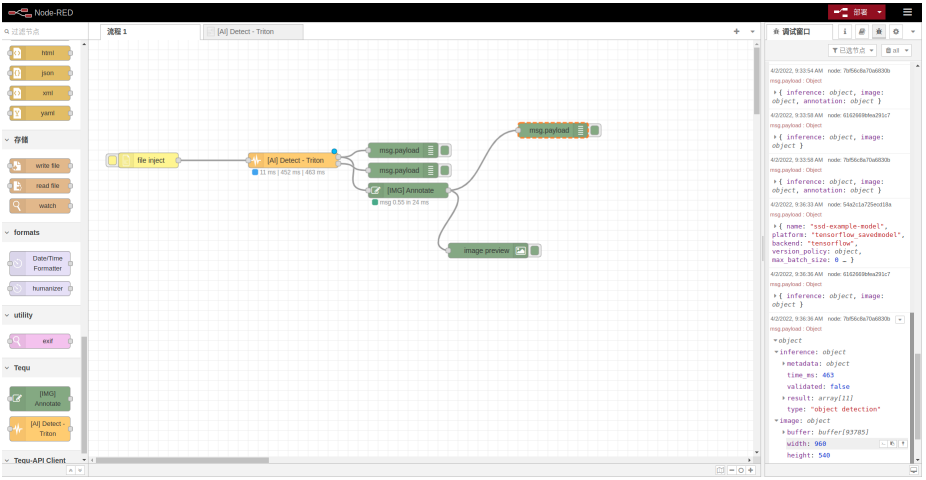
```
169b","name":"request","func":"let modelName =
env.get(\"model_name\")\nlet server_url =
env.get(\"triton_server_url\")\n\nmsg.method =
\"GET\";\nmsg.url =
server_url+\"/v2/models/\"+modelName+\"/config\";\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","l
ibs":[],"x":420,"y":440,"wires":[["71ce551ba3faaef2"]]},
{"id":"df34ddea04c5826b","type":"function","z":"9d7441713e5a
169b","name":"status","func":"let statusCode =
msg.statusCode;\n\nif(statusCode == 200){\n    let
model_data = msg.payload;\n    let modelName =
env.get(\"model_name\")\n    parsed_value =
JSON.parse(model_data.parameters.metadata.string_value)\n
model_data.parameters.labels = parsed_value.labels\n
model_data.parameters.metadata = parsed_value.metadata\n
flow.set(modelName,model_data)  \n
node.status({fill:\"green\",shape:\"dot\",text:\" Model
configuration loaded.\"});   \n
flow.set(\"ready\",true)\n    return [msg,null]\n}\nelse{\n
node.status({fill:\"red\",shape:\"dot\",text:msg.statusCode+
\":\"+msg.payload.error});   \n
node.error(msg.statusCode,msg.payload)\n    return
[null,msg]\n}\n\nreturn
msg;","outputs":2,"noerr":0,"initialize":"","finalize":"","l
ibs":[],"x":750,"y":440,"wires":[[],["57c77c63c706ef2b"]]},
{"id":"5ac4fb48cdf1f282","type":"status","z":"9d7441713e5a16
9b","name":"","scope":
["18efd978252e44b4","d0fddd20c79b5628","ec5e76112f20b1cd","6
f0b51afb2815ede","71ce551ba3faaef2","df34ddea04c5826b","2e4a
bd3abccee2f6"],"x":700,"y":620,"wires":[[]]},
```

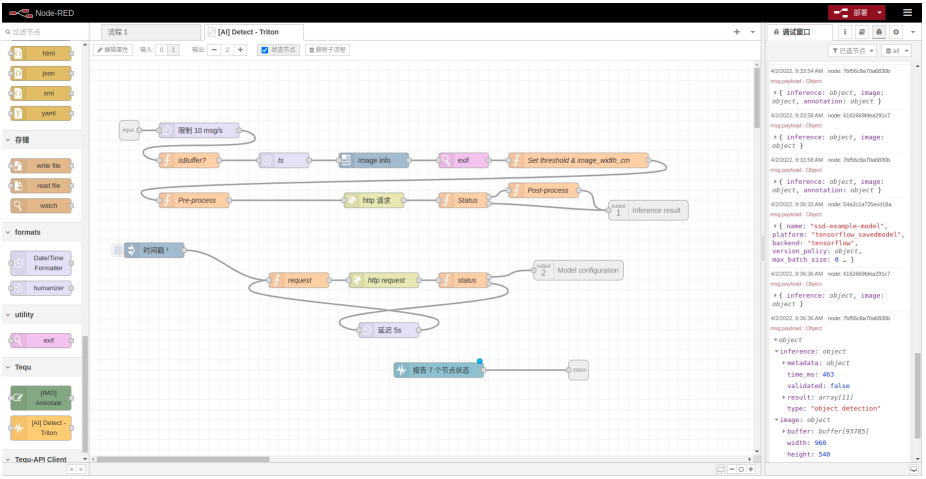
```
{"id":"2e4abd3abccee2f6","type":"function","z":"9d7441713e5a
169b","name":"Status","func":"let statusCode =
msg.statusCode;\nlet timestamp = new
Date().toISOString();\n\nif(statusCode == 200){\n    return
[msg,null]\n}\nelse{\n
node.status({fill:\"yellow\",shape:\"dot\",text:\"Request
failed... checking server...\"});  \n
flow.set(\"ready\",false)\n    return
[null,msg]\n}","outputs":2,"noerr":0,"initialize":"","finali
ze":"","libs":[],"x":750,"y":280,"wires":
[["ec5e76112f20b1cd"],[]]},
{"id":"a27070060363f2df","type":"delay","z":"9d7441713e5a169
b","name":"","pauseType":"rate","timeout":"5","timeoutUnits"
:"seconds","rate":"10","nbRateUnits":"1","rateUnits":"second
","randomFirst":"1","randomLast":"5","randomUnits":"seconds"
,"drop":true,"allowrate":false,"outputs":1,"x":220,"y":140,"
wires":[["aac6b7c6db9f9a61"]]},
{"id":"48add00aafb4b737","type":"inject","z":"9d7441713e5a16
9b","name":"","props":[{"p":"payload"},
{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":tr
ue,"onceDelay":"1","topic":"","payload":"","payloadType":"da
te","x":130,"y":380,"wires":[["e2b2f167d1e53859"]]},
{"id":"57c77c63c706ef2b","type":"delay","z":"9d7441713e5a169
b","name":"","pauseType":"delay","timeout":"5","timeoutUnits
":"seconds","rate":"1","nbRateUnits":"1","rateUnits":"second
","randomFirst":"1","randomLast":"5","randomUnits":"seconds"
,"drop":false,"allowrate":false,"outputs":1,"x":600,"y":560,
"wires":[["e2b2f167d1e53859"]]},
{"id":"a9c87fd163b03fe8","type":"moment","z":"9d7441713e5a16
9b","name":"ts","topic":"","input":"","inputType":"date","in
```

```
Tz":"Europe/Helsinki","adjAmount":0,"adjType":"days","adjDir
":"add","format":"HH:mm:ss.SSS","locale":"fi-
FI","output":"ts","outputType":"msg","outTz":"Europe/Helsink
i","x":390,"y":200,"wires":[["6f0b51afb2815ede"]]},
{"id":"3d0cd6d906f70633","type":"tab","label":"流程
1","disabled":false,"info":"","env":[]},
{"id":"3ff6e11b151051a1","type":"subflow:9d7441713e5a169b","
z":"3d0cd6d906f70633","name":"","env":
[{"name":"threshold","value":"0.5","type":"num"},
{"name":"triton_server_url","value":"http://192.168.4.154:80
00","type":"str"}],"x":410,"y":240,"wires":
[["6162669bfea291c7","6e3e72cd784b2471"],
["54a2c1a725ecd18a"]]},
{"id":"6162669bfea291c7","type":"debug","z":"3d0cd6d906f7063
3","name":"","active":true,"tosidebar":true,"console":false,
"tostatus":false,"complete":"payload","targetType":"msg","st
atusVal":"","statusType":"auto","x":630,"y":220,"wires":[]},
{"id":"54a2c1a725ecd18a","type":"debug","z":"3d0cd6d906f7063
3","name":"","active":true,"tosidebar":true,"console":false,
"tostatus":false,"complete":"false","statusVal":"","statusTy
pe":"auto","x":630,"y":260,"wires":[]},
{"id":"a862fae855735785","type":"image","z":"3d0cd6d906f7063
3","name":"","width":"480","data":"payload.annotation.buffer
","dataType":"msg","thumbnail":false,"active":true,"pass":fa
lse,"outputs":0,"x":800,"y":420,"wires":[]},
{"id":"6e3e72cd784b2471","type":"subflow:83a7a965.1808a8","z
":"3d0cd6d906f70633","name":"","env":
[{"name":"box_colors","value":"
{\"dog\":\"#FFFFFF\",\"car\":\"#52FF47\"}","type":"json"},
{"name":"bbox_text_color","value":"yellow","type":"str"},
```

{"name":"threshold","value":"0.50","type":"num"},
{"name":"labels","value":"
[\"person\",\"bicycle\",\"car\",\"motorcycle\",\"airplane\",
\"bus\",\"train\",\"truck\",\"boat\",\"traffic
light\",\"fire hydrant\",\"street sign\",\"stop
sign\",\"parking
meter\",\"bench\",\"bird\",\"cat\",\"dog\",\"horse\",\"sheep
\",\"cow\",\"elephant\",\"bear\",\"zebra\",\"giraffe\",\"hat
\",\"backpack\",\"umbrella\",\"shoe\",\"eye
glasses\",\"handbag\",\"tie\",\"suitcase\",\"frisbee\",\"ski
s\",\"snowboard\",\"sports ball\",\"kite\",\"baseball
bat\",\"baseball
glove\",\"skateboard\",\"surfboard\",\"tennis
racket\",\"bottle\",\"plate\",\"wine
glass\",\"cup\",\"fork\",\"knife\",\"spoon\",\"bowl\",\"bana
na\",\"apple\",\"sandwich\",\"orange\",\"broccoli\",\"carrot
\",\"hot
dog\",\"pizza\",\"donut\",\"cake\",\"chair\",\"couch\",\"pot
ted plant\",\"bed\",\"mirror\",\"dining
table\",\"window\",\"desk\",\"toilet\",\"door\",\"tv\",\"lap
top\",\"mouse\",\"remote\",\"keyboard\",\"cell
phone\",\"microwave\",\"oven\",\"toaster\",\"sink\",\"refrig
erator\",\"blender\",\"book\",\"clock\",\"vase\",\"scissors\
",\"teddy bear\",\"hair drier\",\"toothbrush\",\"hair
brush\"]","type":"json"}],"x":640,"y":300,"wires":
[["a862fae855735785","7bf56c8a70a6830b"]]},
{"id":"1f64ebf42699f276","type":"fileinject","z":"3d0cd6d906
f70633","name":"","x":120,"y":240,"wires":
[["3ff6e11b151051a1"]]},
{"id":"7bf56c8a70a6830b","type":"debug","z":"3d0cd6d906f7063

3","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":930,"y":180,"wires":[]}]





~~Triton-inference-server python backend quick-start~~

How to Auto-Generated Model Configuration

Use camera on Jetson

```
python3 jetson_cam.py --vid 0 --width 1280 --height 720
```

*PS: if have problem,please reinstall libopencv and libopencv-python

Install the DeepStream SDK

- **Method 1**: Using SDK Manager

  Select `DeepStreamSDK` from the `Additional SDKs` section along with JP 4.6.1 software components for installation.

- **Method 2**: Using the DeepStream tar package: https://developer.nvidia.com/deepstream_sdk_v6.0.1_jetsontbz2

  1. Download the DeepStream 6.0.1 Jetson tar package `deepstream_sdk_v6.0.1_jetson.tbz2` to the Jetson device.
  2. Enter the following commands to extract and install the DeepStream SDK:

```
$  sudo tar -xvf
deepstream_sdk_v6.0.1_jetson.tbz2 -C /
$ cd /opt/nvidia/deepstream/deepstream-6.0
$ sudo ./install.sh
$ sudo ldconfig
```

- **Method 3**: Using the DeepStream Debian package: https://developer.nvidia.com/deepstream-6.0_6.0.1-1_arm64deb

  Download the DeepStream 6.0.1 Jetson Debian package `deepstream-6.0_6.0.1-1_arm64.deb` to the Jetson device. Enter the following command:

  ```
  $ sudo apt-get install ./deepstream-6.0_6.0.1-
  1_arm64.deb
  ```

  Note

  If you install the DeepStream SDK Debian package using the `dpkg` command, you must install the following packages before installing the Debian package:

  - `libgstrtspserver-1.0-0`
  - `libgstreamer-plugins-base1.0-dev`

- **Method 4**: Use Docker container DeepStream docker containers are available on NGC. See the Docker Containers section to learn about developing and deploying DeepStream using docker containers.

K8S

换源

解决的办法是换国内的源。

```
root@ecs-b769:~# curl
https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg
| apt-key add -
  % Total    % Received % Xferd  Average Speed   Time
Time     Time  Current
                                 Dload  Upload   Total
Spent    Left  Speed
100  1974  100  1974    0     0   6207      0 --:--:-- --
:--:-- --:--:--  6227
OK
root@ecs-b769:~# cat <<EOF
>/etc/apt/sources.list.d/kubernetes.list
> deb https://mirrors.aliyun.com/kubernetes/apt/
kubernetes-xenial main
> EOF
root@ecs-b769:~# sudo apt-get update
root@ecs-b769:~# apt-get install -y kubelet kubeadm
kubectl
```

在 Linux 系统中安装 kubectl

安装minikube（JUST single Node）

```
curl -LO
https://storage.googleapis.com/minikube/releases/latest/m
inikube_latest_arm64.deb
sudo dpkg -i minikube_latest_arm64.deb
```

给docker添加权限

```
sudo usermod -aG docker $USER && newgrp docker
```

用docker当做driver

Usage

Start a cluster using the docker driver:

```
minikube start --driver=docker
```

Copy

To make docker the default driver:

```
minikube config set driver docker
```

安装 kubeadm

使用 kubeadm 创建集群

```
nvidia@ubuntu:~$ sudo kubeadm init --control-plane-
endpoint=cluster-endpoint
```

```
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a
Kubernetes cluster
[preflight] This might take a minute or two, depending on
the speed of your internet connection
[preflight] You can also perform this action in
beforehand using 'kubeadm config images pull'
```

如果遇见墙...

`sudo kubeadm config images pull --image-repository=registry.aliyuncs.com/google_containers`

or `sudo kubeadm init --image-repository=registry.aliyuncs.com/google_containers --pod-network-cidr=10.244.0.0/16 --kubernetes-version=v1.18.5`

If dial tcp 127.0.0.1:10248: connect: connection refused（P53）

```
sudo vim /etc/docker/daemon.json
```

Add

```
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
```

openfaas

faasd

When should you use faasd over OpenFaaS on Kubernetes?

- To deploy microservices and functions that you can update and monitor remotely
- When you don't have the bandwidth to learn or manage Kubernetes
- To deploy embedded apps in IoT and edge use-cases
- To distribute applications to a customer or client
- You have a cost sensitive project - run faasd on a 1GB VM for 5-10 USD / mo or on your Raspberry Pi
- When you just need a few functions or microservices, without the cost of a cluster

faasd does not create the same maintenance burden you'll find with maintaining, upgrading, and securing a Kubernetes cluster. You can deploy it and walk away, in the worst case, just deploy a new VM and deploy your functions again.

arkade

torch2trt