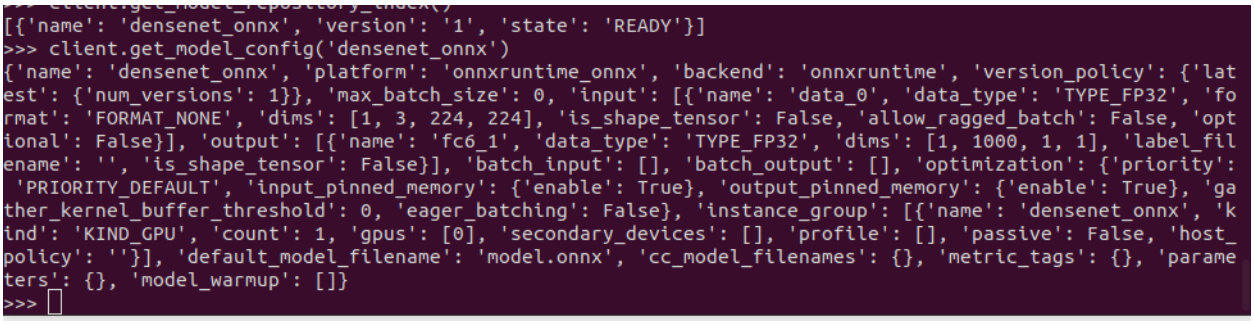
Triton POC Testing

```
# Using 23.02 (date:03-06-23)
docker run --gpus all --rm -p 8000:8000 -p 8001:8001 -p 8002:8002 -v ${PWD}/model_reposito
ry:/models nvcr.io/nvidia/tritonserver:23.02-py3 tritonserver --model-repository=/models


# Running the client server
docker run -it --rm --net=host -v ${PWD}:/workspace/ nvcr.io/nvidia/tritonserver:23.02-py3
-sdk bash
```

Triton Client Code

```
import tritonclient.http as httpclient

# Read in the image with pytorch to get the shape
https://pytorch.org/vision/main/generated/torchvision.io.read_image.html



# Check if the server is alive
client.is_server_live()

transformed_img_shape = list(transformed_img.shape())

# Prep the data shape for inference
inputs = httpclient.InferInput("data_0", transformed_img_shape, datatype="FP32")

# Convert the data to numpy to send to inference server
import numpy as np
>>> inputs.set_data_from_numpy(np.float32(transformed_img.numpy()), binary_data=True)

inputs.set_data_from_numpy(transformed_img, binary_data=True)

outputs = httpclient.InferRequestedOutput("fc6_1", binary_data=True, class_count=1000)typ
```

```
# Use this to get the models on the server
client.get_model_repository_index()
[{'name': 'densenet_onnx', 'version': '1', 'state': 'READY'}]
```




```
perf_analyzer -m densenet_onnx -b 1 --shape input.1:3,224,224 --concurrency-range 2:8:2 --
percentile=95
```