DrDroid

Triton Inference Server Cheatsheet

Connecting to Triton Inference Server

curl -v localhost:8000/v2/health/ready
Check if server is ready to accept requests

curl -v localhost:8000/v2/models
List available models

curl -v localhost:8000/metrics
Get Prometheus metrics

curl -v localhost:8000/v2
Get server metadata

Client SDK Operations

python -m tritonclient.grpc.model_repository_pb2 Model
Python client: Get model details

triton_client = tritonclient.grpc.InferenceServerClient('localhost:8001')
Initialize gRPC client

triton_client = tritonclient.http.InferenceServerClient('localhost:8000')
Initialize HTTP client

triton_client.is_server_live()
Check server health

triton_client.is_model_ready(model_name)
Check if model is ready

Model Management

curl -X POST localhost:8000/v2/repository/models/{model_name}/load
Load a model

curl -X POST localhost:8000/v2/repository/models/{model_name}/unload
Unload a model

curl -X POST localhost:8000/v2/repository/index
Reload model repository

curl localhost:8000/v2/models/{model_name}/config
Get model configuration

Inference

curl -X POST localhost:8000/v2/models/{model_name}/infer -d @request.json
Make inference request (HTTP)

triton_client.infer(model_name, inputs, outputs=outputs)
Make inference request (Python client)

perf_analyzer -m {model_name} -u localhost:8000
Benchmark model performance

Monitoring

curl localhost:8000/metrics
Get Prometheus metrics

curl localhost:8000/v2/models/{model_name}/stats
Get model statistics

prometheus --config.file=prometheus.yml
Start Prometheus to collect Triton metrics

grafana-server
Start Grafana to visualize Triton metrics

Server Management

docker run -p 8000:8000 -p 8001:8001 -p 8002:8002 nvcr.io/nvidia/tritonserver:xx.xx-py3
Start Triton server in Docker

tritonserver --model-repository=/path/to/model_repository
Start Triton server from CLI

tritonserver --allow-metrics=true --metrics-port=8002
Enable metrics collection

tritonserver --log-verbose=1
Enable verbose logging