
Triton Inference Server

License **BSD3**

[!WARNING] ##### LATEST RELEASE You are currently on the `main` branch which tracks under-development progress towards the next release. The current release is version 2.45.0 and corresponds to the 24.04 container release on NVIDIA GPU Cloud (NGC).

Triton Inference Server is an open source inference serving software that streamlines AI inferencing. Triton enables teams to deploy any AI model from multiple deep learning and machine learning frameworks, including TensorRT, TensorFlow, PyTorch, ONNX, OpenVINO, Python, RAPIDS FIL, and more. Triton Inference Server supports inference across cloud, data center, edge and embedded devices on NVIDIA GPUs, x86 and ARM CPU, or AWS Inferentia. Triton Inference Server delivers optimized performance for many query types, including real time, batched, ensembles and audio/video streaming. Triton inference Server is part of NVIDIA AI Enterprise, a software platform that accelerates the data science pipeline and streamlines the development and deployment of production AI.

Major features include:

- Supports multiple deep learning frameworks
- Supports multiple machine learning frameworks
- Concurrent model execution
- Dynamic batching
- Sequence batching and implicit state management for stateful models
- Provides Backend API that allows adding custom backends and pre/post processing operations
- Supports writing custom backends in python, a.k.a. Python-based backends.
- Model pipelines using Ensembling or Business Logic Scripting (BLS)
- HTTP/REST and GRPC inference protocols based on the community developed KServe protocol
- A C API and Java API allow Triton to link directly into your application for edge and other in-process use cases
- Metrics indicating GPU utilization, server throughput, server latency, and more

New to Triton Inference Server? Make use of these tutorials to begin your Triton journey!

Join the Triton and TensorRT community and stay current on the latest product updates, bug fixes, content, best practices, and more. Need enterprise support? NVIDIA global support is available for Triton Inference Server with the NVIDIA AI Enterprise software suite.

Serve a Model in 3 Easy Steps

```
1 # Step 1: Create the example model repository
2 git clone -b r24.04 https://github.com/triton-inference-server/server.
  git
3 cd server/docs/examples
4 ./fetch_models.sh
5
6 # Step 2: Launch triton from the NGC Triton container
7 docker run --gpus=1 --rm --net=host -v ${PWD}/model_repository:/models
  nvcr.io/nvidia/tritonserver:24.04-py3 tritonserver --model-
  repository=/models
8
9 # Step 3: Sending an Inference Request
10 # In a separate console, launch the image_client example from the NGC
   Triton SDK container
11 docker run -it --rm --net=host nvcr.io/nvidia/tritonserver:24.04-py3-
   sdk
12 /workspace/install/bin/image_client -m densenet_onnx -c 3 -s INCEPTION
   /workspace/images/mug.jpg
13
14 # Inference should return the following
15 Image '/workspace/images/mug.jpg':
16     15.346230 (504) = COFFEE MUG
17     13.224326 (968) = CUP
18     10.422965 (505) = COFFEEPOT
```

Please read the QuickStart guide for additional information regarding this example. The quickstart guide also contains an example of how to launch Triton on CPU-only systems. New to Triton and wondering where to get started? Watch the Getting Started video.

Examples and Tutorials

Check out NVIDIA LaunchPad for free access to a set of hands-on labs with Triton Inference Server hosted on NVIDIA infrastructure.

Specific end-to-end examples for popular models, such as ResNet, BERT, and DLRM are located in the NVIDIA Deep Learning Examples page on GitHub. The NVIDIA Developer Zone contains additional documentation, presentations, and examples.

Documentation

Build and Deploy

The recommended way to build and use Triton Inference Server is with Docker images.

- Install Triton Inference Server with Docker containers (*Recommended*)

-
- Install Triton Inference Server without Docker containers
 - Build a custom Triton Inference Server Docker container
 - Build Triton Inference Server from source
 - Build Triton Inference Server for Windows 10
 - Examples for deploying Triton Inference Server with Kubernetes and Helm on GCP, AWS, and NVIDIA FleetCommand
 - Secure Deployment Considerations

Using Triton

Preparing Models for Triton Inference Server The first step in using Triton to serve your models is to place one or more models into a model repository. Depending on the type of the model and on what Triton capabilities you want to enable for the model, you may need to create a model configuration for the model.

- Add custom operations to Triton if needed by your model
- Enable model pipelining with Model Ensemble and Business Logic Scripting (BLS)
- Optimize your models setting scheduling and batching parameters and model instances.
- Use the Model Analyzer tool to help optimize your model configuration with profiling
- Learn how to explicitly manage what models are available by loading and unloading models

Configure and Use Triton Inference Server

- Read the Quick Start Guide to run Triton Inference Server on both GPU and CPU
- Triton supports multiple execution engines, called backends, including TensorRT, TensorFlow, PyTorch, ONNX, OpenVINO, Python, and more
- Not all the above backends are supported on every platform supported by Triton. Look at the Backend-Platform Support Matrix to learn which backends are supported on your target platform.
- Learn how to optimize performance using the Performance Analyzer and Model Analyzer
- Learn how to manage loading and unloading models in Triton
- Send requests directly to Triton with the HTTP/REST JSON-based or gRPC protocols

Client Support and Examples A Triton *client* application sends inference and other requests to Triton. The Python and C++ client libraries provide APIs to simplify this communication.

- Review client examples for C++, Python, and Java
- Configure HTTP and gRPC client options

-
- Send input data (e.g. a jpeg image) directly to Triton in the body of an HTTP request without any additional metadata

Extend Triton

Triton Inference Server's architecture is specifically designed for modularity and flexibility

- Customize Triton Inference Server container for your use case
- Create custom backends in either C/C++ or Python
- Create decoupled backends and models that can send multiple responses for a request or not send any responses for a request
- Use a Triton repository agent to add functionality that operates when a model is loaded and unloaded, such as authentication, decryption, or conversion
- Deploy Triton on Jetson and JetPack
- Use Triton on AWS Inferentia

Additional Documentation

- [FAQ](#)
- [User Guide](#)
- [Customization Guide](#)
- [Release Notes](#)
- [GPU, Driver, and CUDA Support Matrix](#)

Contributing

Contributions to Triton Inference Server are more than welcome. To contribute please review the contribution guidelines. If you have a backend, client, example or similar contribution that is not modifying the core of Triton, then you should file a PR in the contrib repo.

Reporting problems, asking questions

We appreciate any feedback, questions or bug reporting regarding this project. When posting issues in GitHub, follow the process outlined in the Stack Overflow document. Ensure posted examples are:

- minimal – use as little code as possible that still produces the same problem
- complete – provide all parts needed to reproduce the problem. Check if you can strip external dependencies and still show the problem. The less time we spend on reproducing problems the more time we have to fix it -

verifiable – test the code you’re about to provide to make sure it reproduces the problem. Remove all other problems that are not related to your request/question.

For issues, please use the provided bug report and feature request templates.

For questions, we recommend posting in our community GitHub Discussions.

For more information

Please refer to the NVIDIA Developer Triton page for more information.