TMS Basics Tutorial

Table of contents

> ⚠ **Attention**
>
> NVIDIA Triton Management Service (TMS) will reach the end of life on July 31, 2024. The version 1.4.0 is the last release.

This guide will walk you through the basics of creating a lease, running inference against it, and releasing the lease. This guide assumes the following:

- You are familiar with the basics of leases.
- You already have a TMS cluster up and running, with the appropriate secrets configured to get containers from NGC. If you do not, please see the deployment guide to learn how to configure and install TMS.
- You have `tmsctl`, the TMS CLI tool, already installed. This can be downloaded from NGC.
- You have a model repository configured that is hosting your models.
- You can run `kubectl` commands to communicate with your cluster. This is needed to run `kubectl port-forward` commands to open up ports to the Triton server which will host your models. Another option is to run the tutorial in a pod inside the same cluster running Kubernetes. In this case, you can skip the `kubectl port-forward` commands (you will also need to make some slight modifications to refer to the correct service rather than `localhost`).

Connecting to TMS

Before getting to the more interesting steps, you need to ensure you can communicate with the TMS server. This tutorials assumes you are running the steps outlined here outside the Kubernetes cluster hosting TMS and need to open a port to connect to it. The way to do this is to use the `kubectl port-forward`.

First, you need to locate the TMS service. By default, it should be named `tms` and be running on port 30345. The rest of the tutorials assumes this is the case for your

installation. If not, you'll have to modify some commands. To check where TMS is running, run `kubectl get svc`:

```
% kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
tms ClusterIP 10.98.225.223 <none> 30345/TCP 7s
```

With the name of the service and port number on which it is listening, you can run a `kubectl port-forward` command so you can communicate with the service from your local machine. You need to leave this command running, so you will need to do it in separate terminal form the one one which you will be running the rest of the commands.

```
% kubectl port-forward svc/tms 30345:30345
Forwarding from 127.0.0.1:30345 -> 9345
Forwarding from [::1]:30345 -> 9345
```

You should now be able to communicate with TMS. To test it, run a `tmsctl lease list` command:

```
% tmsctl lease list -t http://localhost:30345
Lease State Expires Triton
Count: 0
```

Notice that above, you had to specify the address of the TMS via the `-t` flag. To avoid having to do that each time, run the `tmsctl target add` command to set a default target.

```
% tmsctl target add --set-default test-target http://localhost:30345
```

To inspect your set of named targets and see the default, you can run `tmsctl target list`

```
% tmsctl target list
* test-target -> http://localhost:30345
```

Now you can run `tmsctl` without specifying the `-t` option.

```
% tmsctl lease list
Lease State Expires Triton
Count: 0
```

The rest of this tutorial will assume you have set this and will not specify the `-t` option on each command.

Creating Your First Lease

Creating a lease requires two things:

1. The URI from which to fetch the model.
2. The name of the model which will be used by Triton.

Your TMS installation should be configured model repositories, such as one hosted in an S3 bucket, or in a Kubernetes persistent volume. If this has not already been done, please refer to the model repository documentation. You can also pull a model from an HTTP server.

The below assumes that you have set `$MODEL_URI` to the URI from which to fetch the model. For example, if the model is in a repository named `my_repo`, and is in a folder named `my_model`, you would set `MODEL_URI=model://my_repo/my_model`. If instead your model is hosted on an HTTP server, you would set `MODEL_URI=https://www.example.com/my_model.zip`.

In addition to `$MODEL_URI`, the below assumes you have set `$MODEL_NAME` to the name that your model should have in Triton. This will be used as part of the path for inference requests.

To create a lease, run the `tmsctl lease create` command. Specify a duration of at least 30 minutes (e.g. `--duration 30m`, `--duration 1h`) to have enough time to run the tutorial, but not so much that if you forget to release the lease you hold the resources for too long.

```
% tmsctl lease create -t $TMS_ADDRESS -m name=$MODEL_NAME,uri=$MODEL_URI
--duration 30m
```

```
Lease 9fd209b1f45f424c914ebc2967a3b591
State: Valid
Expires: 2023-10-12T23:27:19Z
Triton: triton-de245ce9.yournamespace.svc.cluster.local
<nvcr.io/nvidia/tritonserver:23.09-py3>
Models:
Name Url Status
$MODEL_NAME $MODEL_URI Ready
```

Assuming everything went well, you should see output similar to the above. A few things to note of importance:

- The lease ID is listed first. This is how you will refer to the lease for operations like getting its status, renewing it, or releasing it. In the example above, this is `9fd209b1f45f424c914ebc2967a3b591`. The rest of this tutorial will refer to this as `$LEASE_ID`.

- The line starting with `Triton:` gives the URL of the Triton server hosting your lease (with all it models). Above, it is `triton-de245ce9.yournamespace.svc.cluster.local`. The first component of this (`triton-de245ce9`), will be referred to as `$TRITON_SERVER` in the rest of the tutorial.

Running Inference

With your lease ready, you can run inference against any of its models (just one in this example). The details of the parameters will vary widely depending on your particular model. The below shows the overall idea, but you will have to adjust it for your model. In a common deployment scenario, you would likely have an application that is making inference requests rather than doing it manually. This is meant to simply demonstrate how to go from creating a lease to running inference.

An important thing to note is that the Kubernetes services associated with the leases are only available inside the cluster. To reach it externally, you need to run a `kubectl port-forward` command like you did for TMS.

```
% kubectl port-forward svc/$TRITON_SERVER 8000:8000
```

You can now run inference against the server. Again, the particulars of the parameters to your model will vary.

```
% curl -X POST -H "Content-Type: application/json"
http://localhost:8000/v2/models/$MODEL_NAME/infer \
--data '{ "inputs": [ {"name": "INPUT", "shape": [1], "datatype": "FP32", "data": [10] }]}'
```

You should see output like the below:

```
{"model_name":"$MODEL_NAME","model_version":"1","outputs":
[{"name":"OUTPUT","datatype":"FP32","shape":[1],"data":[10.0]}]}
```

Other Lease Operations

Now that you have a lease, you can perform many different operations on it. Below are a few basic ones.

List Leases

You can always run `tmsctl lease list` to see the state of leases in your TMS installation.

```
% tmsctl lease list
Lease State Expires Triton
9fd209b1f45f424c914ebc2967a3b591 Valid 2023-10-12T23:39:28 triton-
de245ce9.epauli.svc.cluster.local
Count: 1
```

Lease Status

To get detailed information about a lease, run `tmsctl lease status`.

```
% tmsctl lease status $LEASE_ID
Lease 9fd209b1f45f424c914ebc2967a3b591
State: Valid
Expires: 2023-10-12T23:39:28Z
```

```
Triton: triton-de245ce9.yournamespace.svc.cluster.local
<nvcr.io/nvidia/tritonserver:23.09-py3>
Readied: 2023-10-12T23:17:19Z
Models:
Name Url Status
$MODEL_NAME $MODEL_URI Ready
Events:
Type Source Age Message
Status Triton Manager 0s Creating Triton deployment.
Status Triton Manager 4s Triton deployment ready.
Status Triton Sidecar 5s identity cached; model size: 1930.
Status Triton Sidecar 7s identity is ready.
Status Lease Provider 9s Lease ready.
Status Lease Service 8m Lease renewed by request.
Status Lease Service 12m Lease renewed by request.
```

Renew

Your TMS installation will be configured with a default duration for leases. After that time elapses, TMS will automatically release the lease. If you still need it, you can run `tmsctl lease renew` to renew the lease.

```
% tmsctl lease renew $LEASE_ID
Renewed lease 9fd209b1f45f424c914ebc2967a3b591 [Valid]
Expires: 2023-10-12T23:34:44
```

Create a Custom Lease Name

In the section above where you ran inference, you had to use the name of the Triton instance in the URL. You can create additional names for a lease, which you can use to run inference. You can use this feature to provide more meaningful names to your Triton instances, as well as move the name from one lease to another so you can update your models without changing the URL your application uses.

To be able to use the name `myname` to refer to your lease, run the below.

```
% tmsctl lease name create myname $LEASE_ID
Lease name "myname".
Target lease: $LEASE_ID
```

You can now run inference using the `myname` hostname. You can still use the name of the Triton server as well.

To test the new name, kill your previous `kubectl port-forward` command, and run a new one. This time, use `myname` instead of the name previously provided by TMS.

```
% kubectl port-forward svc/myname 8000:8000
Forwarding from 127.0.0.1:8000 -> 8000
Forwarding from [::1]:8000 -> 8000
```

```
% curl -X POST -H "Content-Type: application/json"
http://localhost:8000/v2/models/$MODEL_NAME/infer \
--data '{ "inputs": [ {"name": "INPUT", "shape": [1], "datatype": "FP32", "data": [10] }]}'
```

Releasing the Lease

When you are done using a lease, you can release all resources associated with it by running `tmsctl lease release`.

```
% tmsctl lease release $LEASE_ID
Lease $LEASE_ID
State: Released
```