Helm Chart Values

> ⚠ **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.

The TMS helm-chart contains a `values.yaml` file which contains all of the deployment configuration options available. TMS configuration is broken into three sections:

- `images`: Container image information used by TMS.
  - `server`: Name of the container image containing TMS Server. *(required)*
  - `sidecar`: Name of the container image containing TMS Triton Sidecar. *(required)*
  - `triton`: Name of the container image containing Triton Inference Server. *(required)*
  - `mongodb`: Name of the container image containing MongoDB database used by TMS Server. *(required)*
  - `rest`: Name of the container image containing TMS HTTP API Server. *(optional)*
  - `secrets`: Name(s) of Kubernetes secrets used to pull container image during pod deployment.
- `kubernetes`: Configuration options affecting how TMS deploys objects with Kubernetes. *(optional)*
  - `customAnnotations`: Custom annotations added to the metadata of pods deployed by TMS.
  - `customLabels`: Custom labels added to the metadata of pods deployed by TMS.

    - `partOf`: Name of a higher level application TMS is a part of, applied as label 'app.kubernetes.io/part-of'.

- `server`: Configuration options related to how TMS Server is deployed an operates.

    - `apiService`: Configuration options related to the network services provided by the TMS Server.

        - `port`: Port to use to connect to the gRPC API service when external to the Kubernetes cluster (default: `30345`).

          External ports must be in the range [30000, 32767].

        - `type`: Type of Kubernetes service connection used by the server to provide network API services (default: `ClusterIP`).

          Valid options are ExternalName, ClusterIP, NodePort, and LoadBalancer.

    - `resources`: Defines the computing and memory resources allocated and reserved for the pod hosting the API server and database. If many concurrent requests are expected to the API server from many different clients, it is highly recommended to change these values. A starting suggestion is to dedicate 25% of the resources to the API server, and 75% to the database.

        - `apiServer`: Defines the resources allocated and reserved for the API server's container.

            - `cpu`: The number of CPUs to be allocated and reserved for the API server container (default: 0). If 0, no CPUs will be requested, but a limit of 1 will be set. If it is any other value, that value will be used used as both the request and limit.

            - `memory`: The amount of memory to be allocated and reserved for the API server container (default: 1Gi). Must be a number with memory units (e.g. Mi, Gi).

        - `database`: Defines the resources allocated and reserved for by the database container.

        - `cpu`: The number of CPUs to be allocated and reserved for the database container (default: 1). This will be used as both the request and limit.
        - `memory`: The amount of memory to be allocated and reserved for the database container (default: 2Gi). Must be a number with memory units (e.g. Mi, Gi).
    - `lease`: Configuration options for the creation and management of leases.
        - `timeout`: Configures amount of time a lease is allowed to attempt loading before timing out.
        - `duration`: Configuration options related to lease durations.
            - `initial`: Configuration options related to initial requested duration of leases.
                - `default`: Default requested duration of a lease (default: `10m`).
                - `maximum`: Maximum requested duration of a lease (default: `30m`).
            - `renewal`: Configures options related to requested renewal duration of leases.
                - `default`: Default requested renewal duration of a lease (default: `10m`).
                - `maximum`: Maximum requested renewal duration of a lease (default: `30m`).
        - `automaticRenewal`: Configuration options related to automatic renewal of leases. *(optional)*
            - `enabled`: Determines if the service supports automatically renewed leases or not (default: `true`).

When not enabled, leases will not be allowed to request to be automatically renewed.

- `window`: Configuration options related to the time since a lease has last been active to be automatically renewed.
  - `default`: Default requested amount of time since a lease has last been active to be automatically renewed (default: `5m`).
  - `maximum`: Maximum requested amount of time since a lease has last been active to be automatically renewed (default: `5m`).

- `databaseStorage`: Configuration option to define persistent storage for the server's database.
  - `volumeClaimName` : Kubernetes persistent volume claim (pvc) attached to the volume where TMS server's database will be stored.

- `shareTriton`: Configuration options related to the sharing of Triton Server instances by leases. *(optional)*
  - `enabled`: Determines if the service supports the sharing of Triton Server instances by leases or not. (default: `false`)
  - `byDefault`: Default value applied to lease requests when not specified (default: `false`).

- `modelRepositories`: Configuration options related to model repositories with models available to instances of Triton.
  - `s3`: Model repositories which contain models stored in a S3 bucket.

    Access is managed by the ARN specified by `server.security.aws.role`.
    - `repositoryName`: Name used to reference this model repository as part of lease acquisition.

May contain only lowercase alphanumeric characters (without spaces, hyphens `-` are permitted).

- `bucketName`: Name of the S3 bucket used to fetch models.
- `awsRegion`: Region code of the S3 Bucket.

  Must be a valid code designating to existing AWS region (eg. “us-west-2”).

  For additional information, refer to https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html.

- `endpoint`: Service URL of the S3 bucket.

  When both ‘endpoint’ and ‘awsRegion’ fields are specified, the ‘endpoint’ value will be used instead of the `awsRegion` value.

  Must be a valid URL designating to an existing endpoint (eg. “http:/s3.us-west-2.amazonaws.com” or “http:/play.min.io:9000”).

  For additional information, refer to https://docs.aws.amazon.com/general/latest/gr/s3.html#amazon_s3

- `accessKey`: Name of the Kubernetes secret to read and provide as the access key ID to download objects from the S3 bucket.

  Optional value when IAM or default AWS environment variables are not used for authorizing TMS to read from an S3 bucket.

- `accessSecret`: Name of the Kubernetes secret containing the secret access key to read from the S3 bucket.

  Optional value when IAM or default AWS environment variables are not used for authorizing TMS to read from an S3 bucket.

- `https`: Model repositories which provide models as compressed archive downloads via web-service using HTTP GET.

        - `secretName`: Name of the Kubernetes secret to read and provide as a Authorization header for download requests.
        - `targetUri`: URL of the remote web-sever in <domain_label_or_ip_address>/<path> format, used to determine if secrets apply to a model request or not.
    - `volumes`: Model repositories which contain models stored in a file-system-like structure.
        - `repositoryName`: Name used to reference this model repository as part of lease acquisition.

          May contain only lowercase alphanumeric characters (without spaces, hyphens `-` are permitted).
        - `volumeClaimName`: Kubernetes persistent volume claim (pvc) used to fetch models.
- `autoscaling`: Configuration options related to autoscaling Triton instances. If this section is missing, autoscaling will be disabled. *(optional)*
    - `enabled`: Determines if TMS Server supports autoscaling Triton instances or not (default: `false`).

      When not enabled, requests for autoscaling leases will not be allowed.
    - `replicas`: Configuration options related to replication of autoscaling Triton instances.
        - `default`: Values used for autoscaling Triton instances when values are not provided during lease acquisition.
            - `maximum`: The maximum number of replicas.Must be within the limits specified in the “limits” section (default: 5).
            - `minimum`: The minimum number of replicas. Must be within the limits specified in the “limits” section (default: 1).

              Must be a positive integer.

        - `limits`: Defines the limits imposed on the number of replicas that may be requested for a lease.
            - `maximum`: The maximum number of replicas (default: 10). Must be a non-negative number greater than or equal to `maximum-idle`.
            - `maximum-idle`: The maximum number of idle instances that are allowed (default: 1). In other words, the maximum value for the minimum number of replicas a user may request. Must be a non-negative number less than or equal to `maximum`.
            - `minimum`: The minimum number of replicas (default: 1). Must be a non-negative number less than or equal to `maximum-idle`.
    - `metrics`: Configuration options related to how metrics are used by autoscaling Triton instances to determine availability and scale.

      At least one metric must be enabled when support for autoscaling Triton instances is enabled.

        - `cpuUtilization`: Metric used to determine scaling based on CPU utilization.
            - `allowed`: Determines whether autoscaling based on CPU utilization is allowed (default: `false`).
            - `enabled`: Determines if scaling based on CPU utilization is enabled by default (default: `false`).
            - `threshold`: Threshold, expressed as a percentage, used to determine scaling (default: `90`).

              Must be a positive integer in the exclusive range (0, 100).

                - `default`: Default value used for the threshold, as a percentage (default: `90`).

        - `minimum`: Minimum value for the threshold, as a percentage (default: `50`).
        - `maximum`: Maximum value for the threshold, as a percentage (default: `100`).
    - `gpuUtilization`: Metric used to determine scaling based on GPU utilization.
        - `allowed`: Determines whether autoscaling based on GPU utilization is allowed (default: `false`).
        - `enabled`: Determines if scaling based on GPU utilization is enabled by default (default: `false`).
        - `threshold`: Threshold, expressed as a percentage, used to determine scaling (default: `90`).

          Must be a positive integer in the exclusive range (0, 100).
            - `default`: Default value used for the threshold, as a percentage (default: `90`).
            - `minimum`: Minimum value for the threshold, as a percentage (default: `50`).
            - `maximum`: Maximum value for the threshold, as a percentage (default: `100`).
    - `queueTime`: Metric used to determine if scaling based on Triton inference-query queue times.
        - `allowed`: Determines whether autoscaling based on Triton inference-query queue times is allowed (default: `false`).
        - `enabled`: Determines if scaling based on Triton inference-query queue times is enabled by default (default: `false`).

        - `threshold`: Threshold, in microseconds, used to determine scaling.
            - `default`: Default value for the threshold, as a time in microseconds (default: `10000`).
            - `minimum`: Minimum value for the threshold, as a time in microseconds (default: `10000`).
            - `maximum`: Maximum value for the threshold, as a time in microseconds, with 0 and negative numbers meaning no limit (default: `0`).
    - `queueTimePercentage`: Metric used to determine scaling based on the percentage of the total inference time that requests spend in the queue.
        - `allowed`: Determines whether autoscaling based on the percentage of time spent in the queue is allowed (default: false).
        - `enabled`: Determines whether autoscaling based on the percentage of time spent in the queue is enabled by default (default: false).
        - `threshold`: Threshold, as a percentage, used to determine scaling.
            - `default`: Default value for the threshold, as a percentage (default: 50).
            - `minimum`: Minimum value for the threshold, as a percentage (default: 1).
            - `maximum`: Maximum value for the threshold, as a percentage (default: 100)
- `metrics`: Configuration options for the collection and reporting of runtime metrics by TMS Server.

- `verbosity`: Verbosity (volume of total metrics) of metrics collected and reported (default: `0`). *(optional)*

  Must be in the range [0, 3].

- `reportingWindow`: Period of time from the time of request used when determining metric values reported (default: `60s`).

- `port`: Port used to connect to the metrics service when external to the Kubernetes cluster (default: `30543`).

  Must be in the range [30000, 32767].

- `models`: Configuration options controlling model deployment (fetching, loading into Triton, etc.) metrics collection.

  - `verbosity`: Verbosity (volume of total metrics) of metrics collected and reported (default: `0`).

    Must be in the range [0, 3].

  - `reportingWindow`: Frequency which model metrics are pushed from Triton sidecar to TMS Server (default: `15s`).

- `security`: Configuration options related to Transport Layer Security (TLS) connection encryption and security.

  - `aws`: Configuration options for instances deployed using Amazon EKS.

    - `role`: AWS IAM role used read models S3 buckets configured in `server.modelRepositories.awsS3`.

  - `tls`: Configuration options related to Transport Layer Security (TLS) connection encryption.

    - `enabled`: Determines if TLS is expected to be enabled or not (default: `false`).

When enabled, TMS will provision a certificate issuer as part of its deployment. The issuer will be used to issue TLS certificates for each Triton Inference Server instance deployed by TMS.

- `certManager`: Configuration options related to cert-manager supplied TLS certificate(s) used to encrypt network traffic.

  TMS manages and applies certificates for TLS based secure communications using cert-manager.

  - `group`: Kubernetes resource group of the CA issuer to use when creating service certificates (default: `cert-manager.io`).
  - `kind`: Kubernetes resource kind of the CA issuer to use when creating service certificates (default: `ClusterIssuer`).
  - `name`: Name of the issuer to use when creating service certificates.
  - `privateKey`: Configuration options related to the creation of certificate private keys.
    - `algorithm`: Algorithm of the private key for the certificate (default: `RSA`).

      Supported values are `RSA`, `ECDSA`, or `Ed25519`.
    - `size`: Size, in bits, of the corresponding private key for the certificate (default: `4096`).

      Supported values depend on the value of `algorithm`:
      - RSA: `2048`, `4096` or `8192`
      - ECDSA: `256`, `384` or `521`
      - Ed25519: *(property is ignored)*

    - `traceLevel`: Configures the verbosity of the logging produced by the server. *(optional)*

      TMS will produce logs for Kubernetes to collect via standard output and standard error normally when this value is not provided.

- `triton`: Configuration options related to the deployment of Triton Inference Server.

  Values can be customized based on capacity of your cluster's hardware and expected workload characteristics.

    - `enableRestrictedAccess`: Determines if the Triton API and protocols have restricted access enabled (default: true). This feature relies on Limited Endpoint Access feature in Triton.Since it is a BETA feature, it may result in compatability issues in the future.

    - `resources`: Configuration options related to default and maximum resource requests per Triton instance.

        - `default`: Values used to determine the resources assigned to a Triton instance when not provided during lease acquisition.

            - `cpu`: Number of logical CPU cores to assign to a Triton instance (default: `2`).

              Must be a positive integer.

            - `gpu`: Number of logical GPU devices to assign to a Triton instance (default: `1`).

              Must be a positive integer.

            - `sharedMemory`: Amount of a Triton instance's memory to reserve for shared-memory (default: `256Mi`).

              Must be a positive integer, followed by a scale suffix of Ki, Mi, or Gi.

            - `systemMemory`: Amount of main memory to assign to a Triton instance (default: `4Gi`).

Must be a positive integer, followed by a scale suffix of Ki, Mi, or Gi.

- `limits`: Range restrictions on resources allowed to be assigned to a Triton instance.
  - `minimum`: Minimum resources allowed to be assigned to a Triton instance.
    - `cpu`: Number of logical CPU cores to assign to a Triton instance (default: `2`).

      Must be a positive integer.
    - `gpu`: Number of logical GPU devices to assign to a Triton instance (default: `1`).

      Must be a positive integer.
    - `sharedMemory`: Amount of a Triton instance's memory to reserve for shared-memory (default: `128Mi`).

      Must be a positive integer, followed by a scale suffix of Ki, Mi, or Gi.
    - `systemMemory`: Amount of main memory to assign to a Triton instance (default: `1Gi`).

      Must be a positive integer, followed by a scale suffix of Ki, Mi, or Gi.
  - `maximum`: Maximum resources allowed to be assigned to a Triton instance.
    - `cpu`: Number of logical CPU cores to assign to a Triton instance (default: `16`).

      Must be a positive integer.

- `gpu`: Number of logical GPU devices to assign to a Triton instance (default: `4`).

  Must be a positive integer.

- `sharedMemory`: Amount of a Triton instance's memory to reserve for shared-memory (default: `2Gi`).

  Must be a positive integer, followed by a scale suffix of Ki, Mi, or Gi.

- `systemMemory`: Amount of main memory to assign to a Triton instance (default: `32Gi`).

  Must be a positive integer, followed by a scale suffix of Ki, Mi, or Gi.