oduct or feature is in a pre-release state and might change or have limited support. For more information, see the pr
stages (/products/#product-launch-stages).

StatsD (https://github.com/etsy/statsd/wiki) is a protocol for submitting metrics and a daemon for metric data aggregation. By configuring the Monitoring agent's StatsD plugin, you make the agent function as a StatsD daemon that writes metrics to Monitoring.

Using the StatsD plugin with its default configuration is the easiest way to get your custom metrics into Monitoring. The StatsD plugin is only available in the Linux Stackdriver Monitoring agent.

The Monitoring agent can also export other collectd metrics as custom metrics, but there is no simple default configuration. See Custom Metrics from the agent (/monitoring/agent/custom-metrics-agent) for details.

This functionality is only available for agents running on Linux. It is not available on Windows.

Monitoring does not automatically detect StatsD. To use StatsD metrics, configure the StatsD plugin as described in the next section.

The StatsD plugin requires version 5.5.2-356 or later of the Monitoring agent. To update the agent, see Updating the agent (/monitoring/agent/install-agent#agent-version).

Do the following on your supported VM instance running Linux:

1. Download statsd.conf (https://raw.githubusercontent.com/Stackdriver/stackdriver-agent-service-configs/master/etc/collectd.d/statsd.conf) and place it in `/opt/stackdriver/collectd/etc/collectd.d/`, using the following command:

2. The default configuration file instructs the agent to accept StatsD metrics on the default StatsD port, **8125**.

   If you want to send some metrics to your own StatsD daemon and other metrics to the agent's StatsD daemon, then change the port settings in the configuration file.

3. Restart the Monitoring agent to pick up the StatsD configuration by running the following command:

For more information on the `collectd` `statsd` plugin, see Plugin:StatsD (https://collectd.org/wiki/index.php/Plugin:StatsD).

To get you started quickly, the agent's StatsD plugin comes with a default collectd configuration that maps from StatsD metrics to Stackdriver custom metrics:

- All metrics from the StatsD plugin have `statsd` in the collectd `plugin` component.
- Each StatsD **metric type**—held in the collectd `type` component—has a corresponding custom metric type name.
- The StatsD **metric name**—held in the collectd `type_instance` component—is stored as the value of a label named `metric`.

  The value of the `metric` label is different for the metric type `Timer`: it includes both the metric name and a counter name: average, upper, lower, sum, percentile-50, and percentile-95.

For example, the following table shows how the supported StatsD metric types and the metric's name are mapped to Monitoring custom metrics:

StatsD type	StatsD name	Stackdriver metric type	Metric kind	Value type	Metric label(s)
Counter	my.counter	custom.googleapis.com/statsd/derive	Cumulative	Int64	metric:my.counter
Gauge	my.gauge	custom.googleapis.com/statsd/gauge	Gauge	Double	metric:my.gauge
Set	my.set	custom.googleapis.com/statsd/objects	Gauge	Double	metric:my.set
Timer[1]	my.timer	custom.googleapis.com/statsd/latency	Gauge	Double	metric:my.timer-average
		(same)	(same)	(same)	metric:my.timer-upper
		(same)	(same)	(same)	metric:my.timer-lower
		(same)	(same)	(same)	metric:my.timer-sum
		(same)	(same)	(same)	metric:my.timer-percentile-50
		(same)	(same)	(same)	metric:my.timer-percentile-95
		custom.googleapis.com/statsd/gauge	Gauge	(same)	metric:my.timer-count

Notes:
[1] There is an incoming sequence of statsd timer metrics with the same name. The agent aggregates StatsD timer metrics and exports summary data to 7 different time series.

For more information on StatsD types, see the StatsD specification (https://github.com/etsy/statsd/blob/master/docs/metric_types.md).

The default StatsD configuration is designed to get you started quickly. This section helps you customize the configuration to suit more complex needs.

You should be familiar with Stackdriver's custom metrics. For an introduction to Stackdriver metrics, see Metrics, Time Series, and Resources (/monitoring/api/v3/metrics). For more details about custom metrics, see Custom Metrics (/monitoring/custom-metrics/).

You can customize the following things:

- You can change the values assigned to the default `metric` label. Using more label values results in more time series in your custom metric. Using fewer label values results in fewer time series.

- You can change the custom metric types. You don't have to use the predefined types provided in the default configuration. For example, you could identify metrics with a certain name and use a different custom metric type for them.
- If you change the custom metric types, then you can can also change the labels associated with each type. The default configuration has a single label, but you can add more labels or change the label key.

If you change the metric types, you should define your new custom metric types in the Monitoring API. For details, see the following section, Designing a custom metric type (#design-metric).

Assume that you are using StatsD to monitor an application consisting of two services, `my_service_a` and `my_service_b`. For each service, you want to export to Monitoring a counter metric that represents the number of failed requests. You don't want to use the default StatsD metric types.

Before defining your own custom metric types, it is important to understand the structure of a collectd metric and how a StatsD metric by default maps into custom metrics.

Collectd metrics, including StatsD metrics, include the following components:

In this example, the StatsD metrics that you want to export have the following identifiers in collectd:

Component	Expected value(s)
Host	*any*
Plugin	`statsd`
Plugin instance	*unset*[1]
Type	`derive`[2]
Type instance	`[SERVICE_NAME].GET.[CODE]`[3]
`[VALUE]`	*any value*[4]

Notes:
[1] The StatsD plugin currently leaves this component empty.
[2] A StatsD Counter metric is mapped to the collectd `derive` type. [3] For example, a type instance might be `my_service_a.GET.500`. [4] [VALUE] is typically a timestamp and a double-precision number.

The following table shows how this metric would be mapped by default:

StatsD type	StatsD name	Stackdriver metric type	Metric kind	Value type	Metric labels
Counter	my_service_a.GET.500	custom.googleapis.com/statsd/deriveCumulativeInt64			metric:my_servce_a.GET.500
Counter	my_service_b.GET.403	custom.googleapis.com/statsd/deriveCumulativeInt64			metric:my_servce_b.GET.403

The default mapping might present some difficulties for you:

- This particular counter metric (`[SERVICE_NAME].GET.[CODE]`) is in the same custom metric type as all other counter metrics. You cannot easily get just this metric's data, because Stackdriver does not presently support regular expression searches on labels.
- You cannot easily get data for individual services or individual response codes in your data. For example, you cannot easily get the total number of errors (of all kinds) that occurred in `my_service_a`.
- The default configuration exports all StatsD metrics to Stackdriver, which could be costly if you are only interested in certain metrics.

For a full discussion of creating custom metric types, see Creating Custom Metrics (/monitoring/custom-metrics/creating-metrics#defining_your_metric).

The following custom metric type is a reasonable choice for the data in this example, because it holds only the StatsD metric you are interested in and because its choice of labels helps to better organize the data:

- Name: `custom.googleapis.com/http/request_errors`
- Labels:
  - `service_name` (STRING): The name of the service.
  - `response_code` (INT64): The HTTP response code.
- Kind: CUMULATIVE

- Type: INT64

The following table shows the desired mapping from StatsD to Stackdriver:

StatsD type	StatsD name	Stackdriver metric type	Metric kind	Value type	Metric labels
Count	my_service_a.GET.500	custom.googleapis.com/http/request_errors	Cumulative	Int64	service_name:my_serv response_code:500
Count	my_service_b.GET.403	custom.googleapis.com/http/request_errors	Cumulative	Int64	service_name:my_serv response_code:403

Once you've designed the metric type, create it using metricDescriptors.create (/monitoring/api/ref_v3/rest/v3/projects.metricDescriptors/create). For information about letting Monitoring create the metric type for you, see Auto-creation of custom metrics (/monitoring/custom-metrics/creating-metrics#auto-creation).

To export the StatsD metric to the new custom metric type, replace the contents of the default StatsD plugin configuration, `/opt/stackdriver/collectd/etc/collectd.d/statsd.conf`, with the following code:

nstalling this configuration file removes the default StatsD mapping. If you want to retain the default mapping for ot
metrics, you must add the **`Rule`** from the default StatsD configuration after the new **`rewrite_request_errors`** ru

Restart your agent to pick up the new configuration by executing the following command on your VM instance:

Your custom metric information begins to flow into Monitoring immediately.

Customizing the StatsD plugin is a special case of customizing collectd metrics for Monitoring. For more information, see the Reference and Troubleshooting sections of Custom Metrics from the Agent (/monitoring/agent/custom-metrics-agent).