Heroku Datadog Drain Library (Go)

Datadog Community Spotlight

Every week in the *Datadog Community Spotlight* we showcase an integration, library, or other contributions generously created by members of the Datadog community. In the past we have generally showcased libraries and integrations for various programming languages, but this week we'll take a look at an integration for one of the worlds largest cloud platforms, Heroku.

Heroku is a platform as a service (*PaaS*) that enables developers to build, run, and operate applications entirely in the cloud. It's no doubt that its booming popularity led to the development of our featured library this week, Heroku Datadog Drain Library. This library in particular is written in, and for the Go language which is also a quickly growing community.

Why use this library?

By default, Heroku only emits its metrics via logs. Although Datadog has a wonderful product for log analysis, it does not, at this moment, generate metrics using these logs. This library will allow us to do just that and then ingest those new metrics into Datadog for enhanced monitoring. There'll be no more wondering what's going on in Heroku-land as we will have a plethora of metrics coming in to Datadog.

The *heroku-datadog-drain* library makes use of the Dogstatsd metrics aggregation service that's bundled with the Datadog Agent by default. By levereging *Dogstatsd* we can send just about any metric you could imagine to Datadog. Luckily setup is quick and painless, lets get started!

Installation & Configuration:

1. First we need to clone the GitHub repository for Heroku Datadog Drain Library:

```
git clone git@github.com:apiaryio/heroku-datadog-drain-golang.git
cd heroku-datadog-drain-golang
```

2. Next we need to setup Heroku and specify the app you want to monitor, as well as create a password for that app.

```
heroku create
heroku config:set ALLOWED_APPS=<appname> <APPNAME>_PASSWORD=<password>
```

Ensure that you [set right Go version](https://devcenter.heroku.com/articles/go-support#go-versions).

```
heroku config:set GOVERSION=go1.12
```

(**Note:**You can use specific settings for Go modules)

Deploy to Heroku and add the Heroku log drain:

1. From your local project repository, simply run the following to deploy your Heroku app.

```
git push heroku master
heroku ps:scale web=1
```

2. Lastly, add the Heroku log drain.

```
heroku drains:add https://<appname>:<password>@<host>.herokuapp.com/ -- app <appname>
```

Usage:

Before you begin configuring your Heroku application to use *heroku-datadog-drain*, you're strongly advised to enable *log-runtime-metrics* for every Heroku application you wish to monitor. This will add basic metrics such as CPU, memory, and other low-level system metrics into logs.

To do this simply run:

```
`heroku labs:enable log-runtime-metrics -a APP_NAME`
```

Now you are able to start using the below configuration keys in your Heroku app.

Available Configuration Keys

Key	Required	Description
STATSD_URL	Required	Set to: localhost:8125
DD_API_KEY	Required	Datadog API Key
ALLOWED_APPS	Required	Comma separated list of app names
APPNAME_PASSWORD	Required	One per allowed app where <APP-NAME> corresponds to an app name from ALLOWED_APPS
APPNAME_TAGS	Optional	Comma separated list of default tags for each app
APPNAME_PREFIX	Optional	String prepended to all metrics from a given app
DATADOG_DRAIN_DEBUG	Optional	Generates A LOT of logs
EXCLUDED_TAGS:	Optional	Useful for solving problems with tags limit (1000)

Note: When you see ALL CAPS used in the application name it should also be in full caps. For example, to set the password for an application named *my-app*, you would need to specify the following.

```
heroku config:set ALLOWED_APPS=my-app MY-APP_PASSWORD=example_password
```

Why Excluded Tags?

The rationale for EXCLUDED_TAGS is that *path=tag* in the Heroku logs includes the full HTTP path - including, for instance, query parameters. This makes very easy to flood Datadog with numerous distinct tag/value pairs; and Datadog has a hard limit of 1000 such distinct pairs. When the limit is breached the entire metric is shut down, and nobody wants that.

Custom Metrics:

If you want to log custom metrics, simply format the log line like following:

```
app web.1 - info: responseLogger: metric#tag#route=/parser metric#request_id=11747467-f4c
```

Supported tags for custom metrics:

- `metric#` and `sample#` for gauges
- `metric#tag` for tags.

- `count#` for counter increments
- `measure#` for histograms

(**Note:** More information about accepted *data-types* can be found here.)

Wrapping up

Hopefully by now your custom Heroku app is sending its metrics into Datadog and you're making good use of them. I highly recommend reading through the Metrics Introduction on our website to get you started, or if you're already a metrics wizard take a look at some more advanced monitoring solutions like APM & Distributed Tracing, or Security Monitoring. You'll be suprised at what you can do with even the most basic of metrics.

Last, but not least, we would like to thank Apiary for the development, and contribution this library to the community! Your efforts make it possible to bring the power of Datadog metrics to the Heroku community.