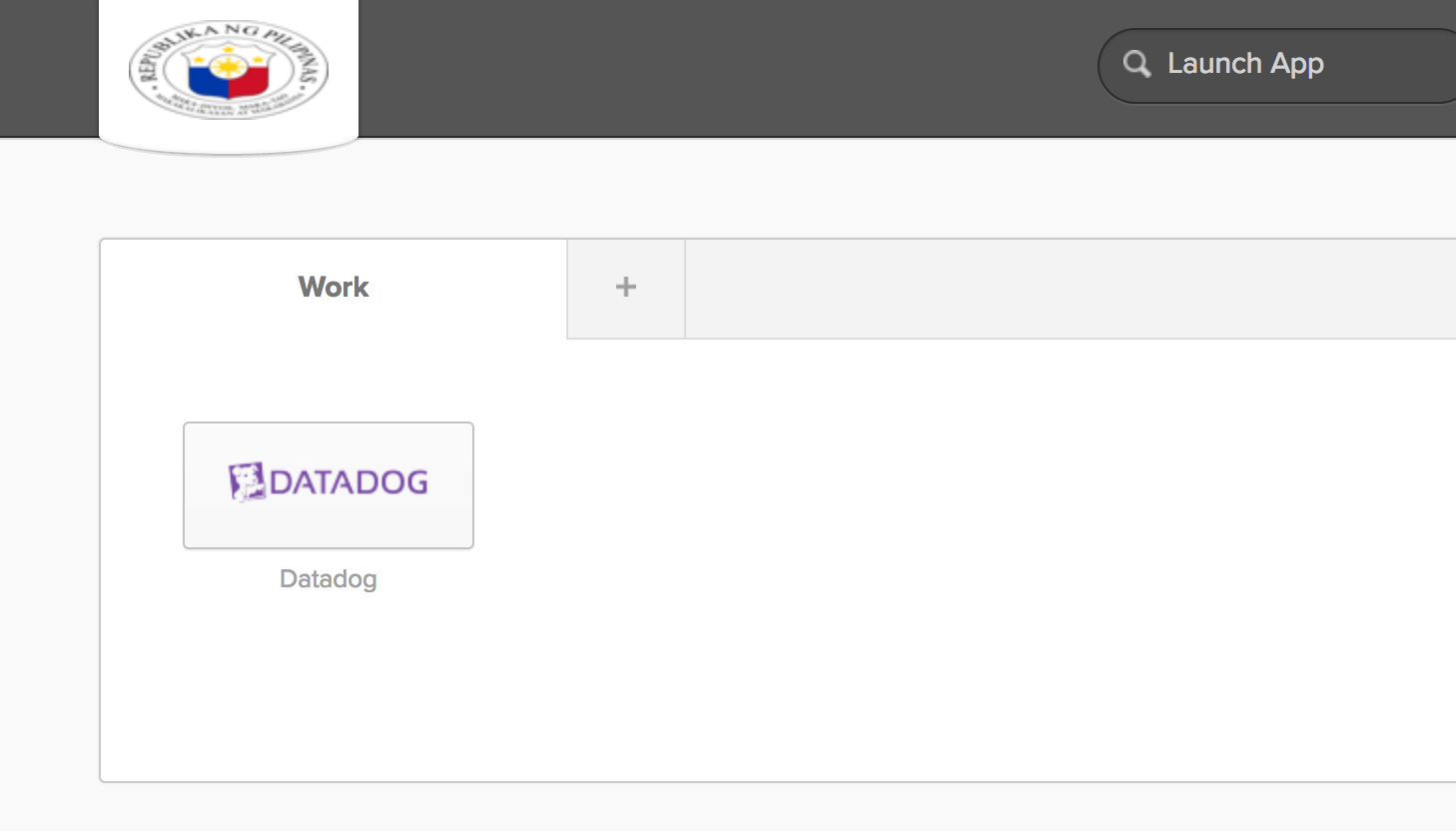
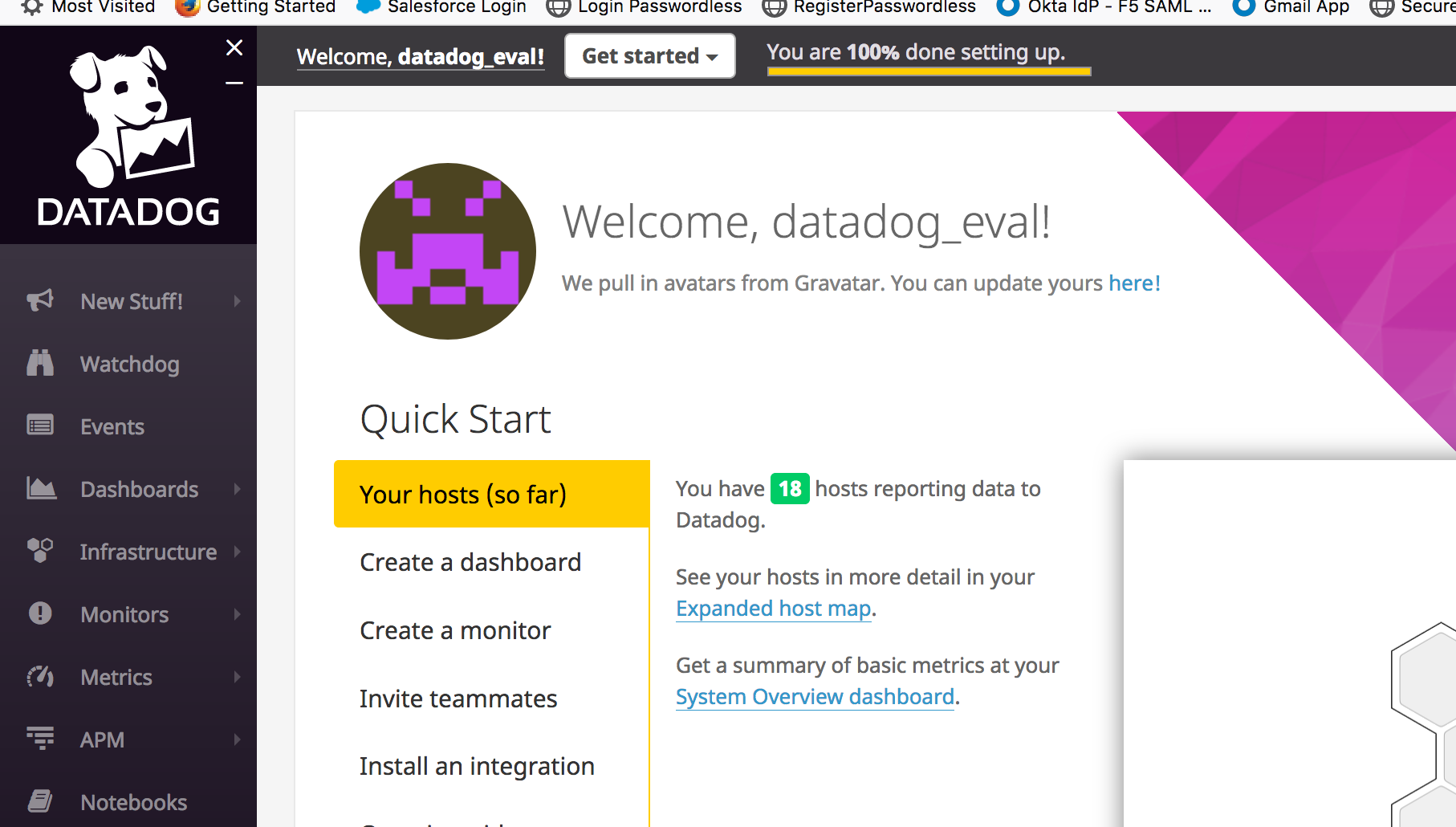
You can login to my Datadog trial instance by doing SAML IdP initiation request via:

Okta IdP page: https://dev-998003.oktapreview.com/

Username: datadog\_eval@okta.com

Password: SEChallenge123!





Add tags in the Agent config file and show us a screenshot of your host and its tags on the Host Map page in Datadog.

Jefferson Haw: Please check datadog.yaml file.

**# Set the host's tags (optional)**

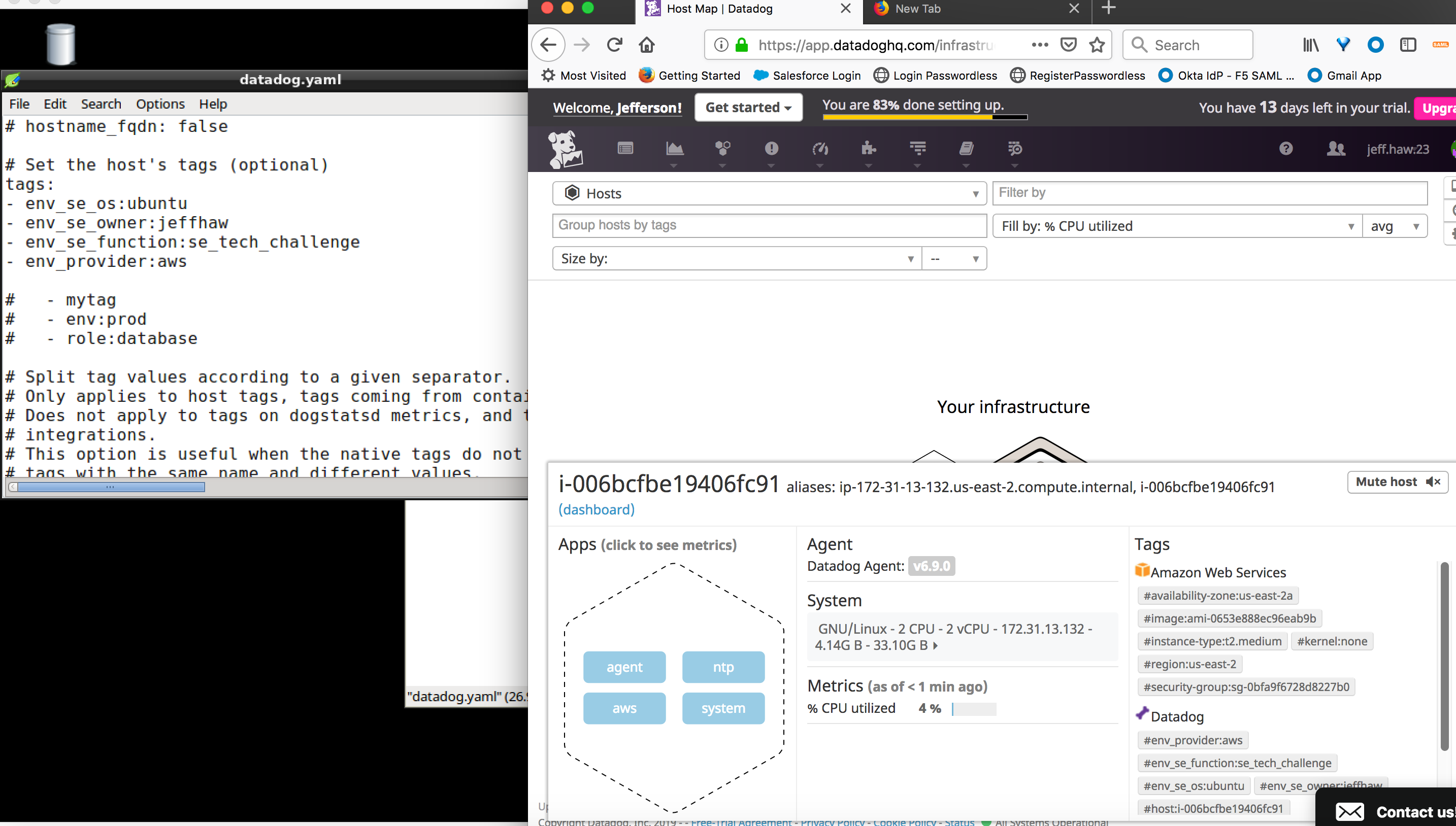
tags:

- env\_se\_os:ubuntu

- env\_se\_owner:jeffhaw

- env\_se\_function:se\_tech\_challenge

- env\_provider:aws



Install a database on your machine (MongoDB, MySQL, or PostgreSQL) and then install the respective Datadog integration for that database.

Jefferson Haw: installing MySQL

All done!

ubuntu@ip-172-31-13-132:/etc/datadog-agent$ systemctl status mysql.service

● mysql.service - MySQL Community Server

Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: en

Active: active (running) since Sun 2019-02-03 11:39:15 UTC; 1min 27s ago

Main PID: 11406 (mysqld)

CGroup: /system.slice/mysql.service

└─11406 /usr/sbin/mysqld

mysqladmin Ver 8.42 Distrib 5.7.25, for Linux on x86\_64

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Server version      5.7.25-0ubuntu0.16.04.2

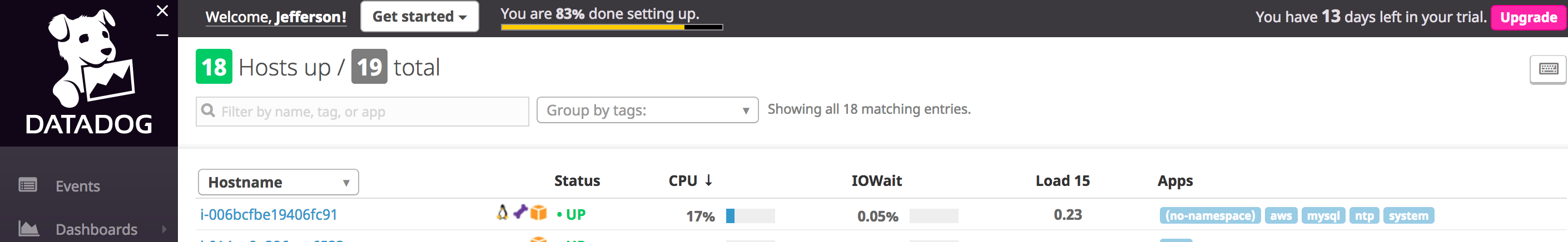
Protocol version    10

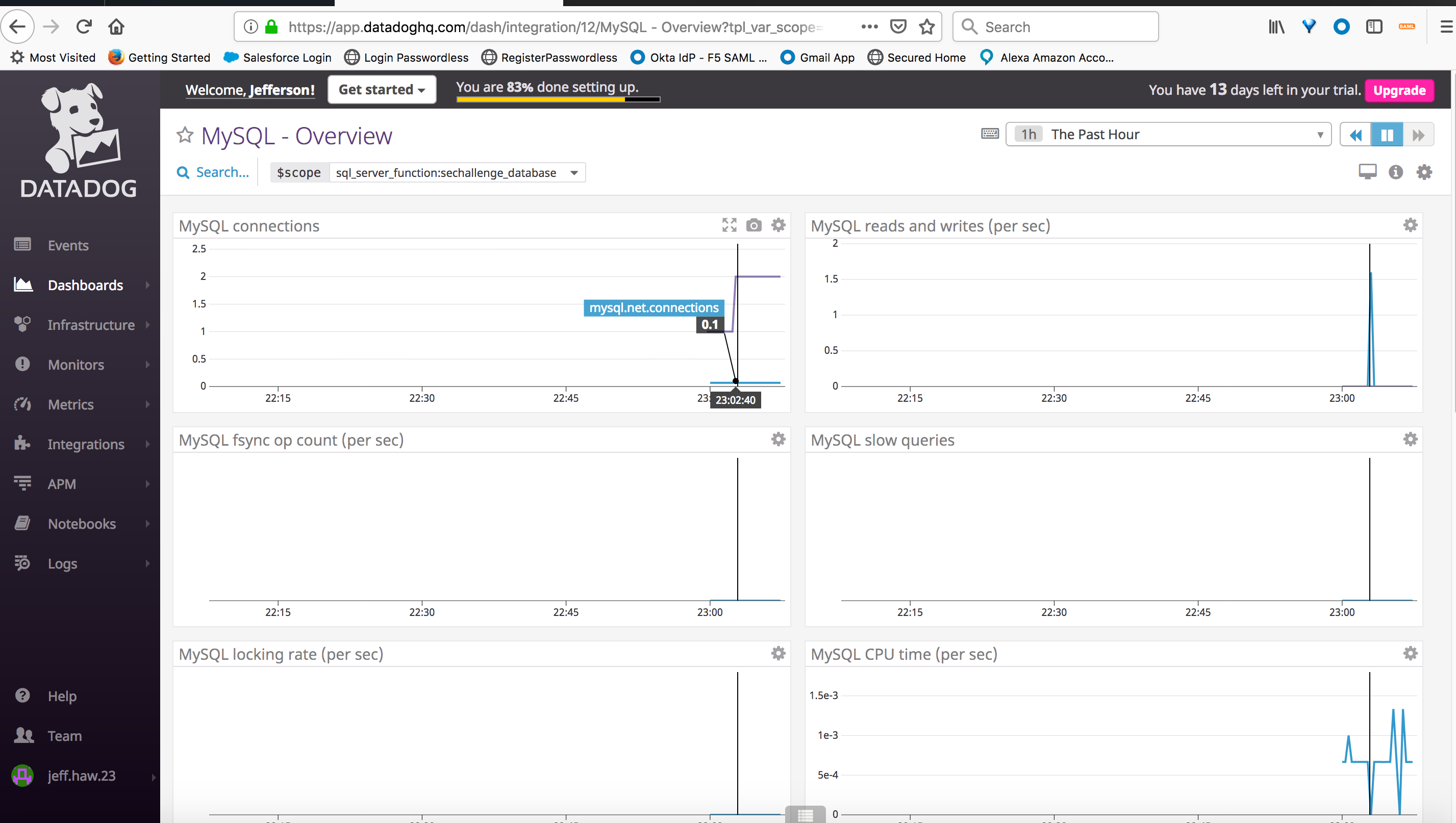
Connection      Localhost via UNIX socket

UNIX socket     /var/run/mysqld/mysqld.sock

Uptime:         2 min 13 sec

Threads: 1 Questions: 5 Slow queries: 0 Opens: 115 Flush tables: 1 Open tables: 34 Queries per second avg: 0.037





Create a custom Agent check that submits a metric named my\_metric with a random value between 0 and 1000.

Jefferson Haw: Please check myMetric.py

**#import random modules to generate number between 0 to 1000**

import random

**# the following try/except block will make the custom check compatible with any Agent version**

try:

# first, try to import the base class from old versions of the Agent...

from checks import AgentCheck

except ImportError:

# ...if the above failed, the check is running in Agent version 6 or later

from datadog\_checks.checks import AgentCheck

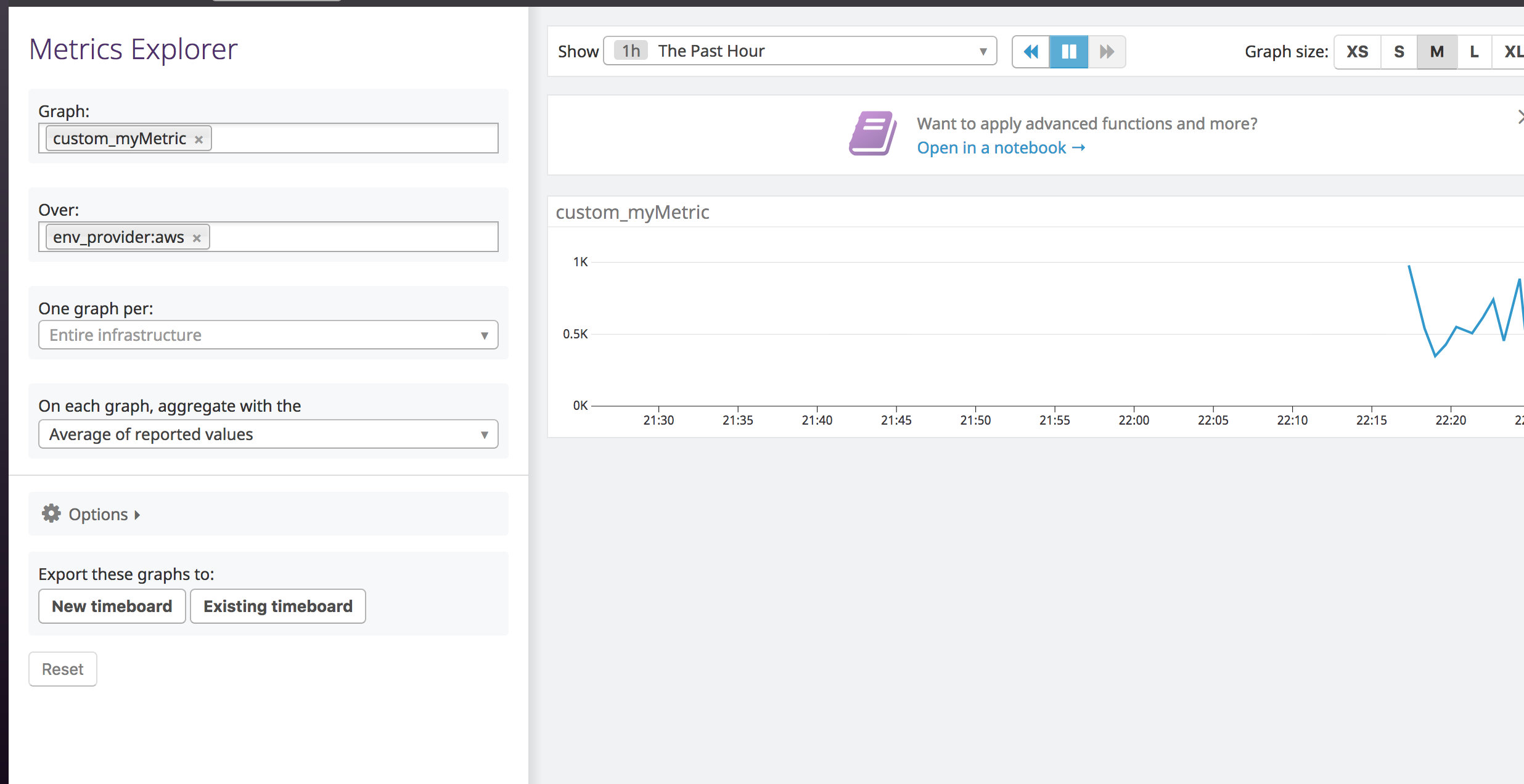
**# content of the special variable \_\_version\_\_ will be shown in the Agent status page**

**\_\_version\_\_** = "1.0.0"

class myMetricCheck(AgentCheck):

def check(self, instance):

self.gauge('custom\_myMetric',random.uniform(0, 1000))



Change your check's collection interval so that it only submits the metric once every 45 seconds.

Jefferson Haw: Please check myMetric.yaml file.

init\_config:

instances:

- min\_collection\_interval: 45

Bonus Question Can you change the collection interval without modifying the Python check file you created?

Jefferson Haw: create a myMetric.yaml file wherein you can set the interval collection frequency of the custom check. This is better rather than programmatically implementing this within the python code.

**#Visualizing Data:**

Utilize the Datadog API to create a Timeboard that contains:

Your custom metric scoped over your host.

Any metric from the Integration on your Database with the anomaly function applied.

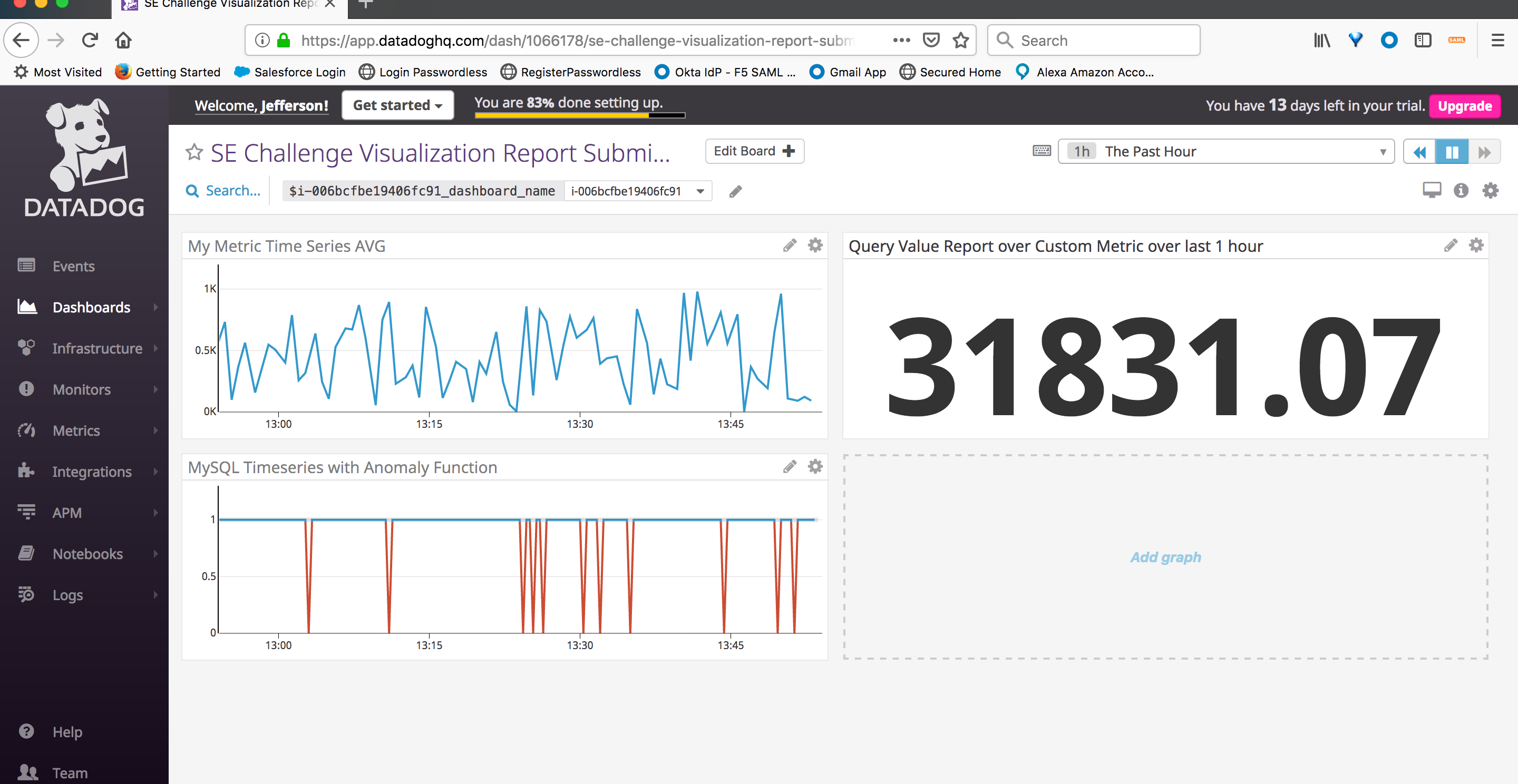
Your custom metric with the rollup function applied to sum up all the points for the past hour into one bucket

Please be sure, when submitting your hiring challenge, to include the script that you've used to create this Timeboard.

Jefferson Haw: Please see TimeboardRequestJSON.json and TimeboardResponseJSON.json for API script. I've used curl command via postman using JSON objects.

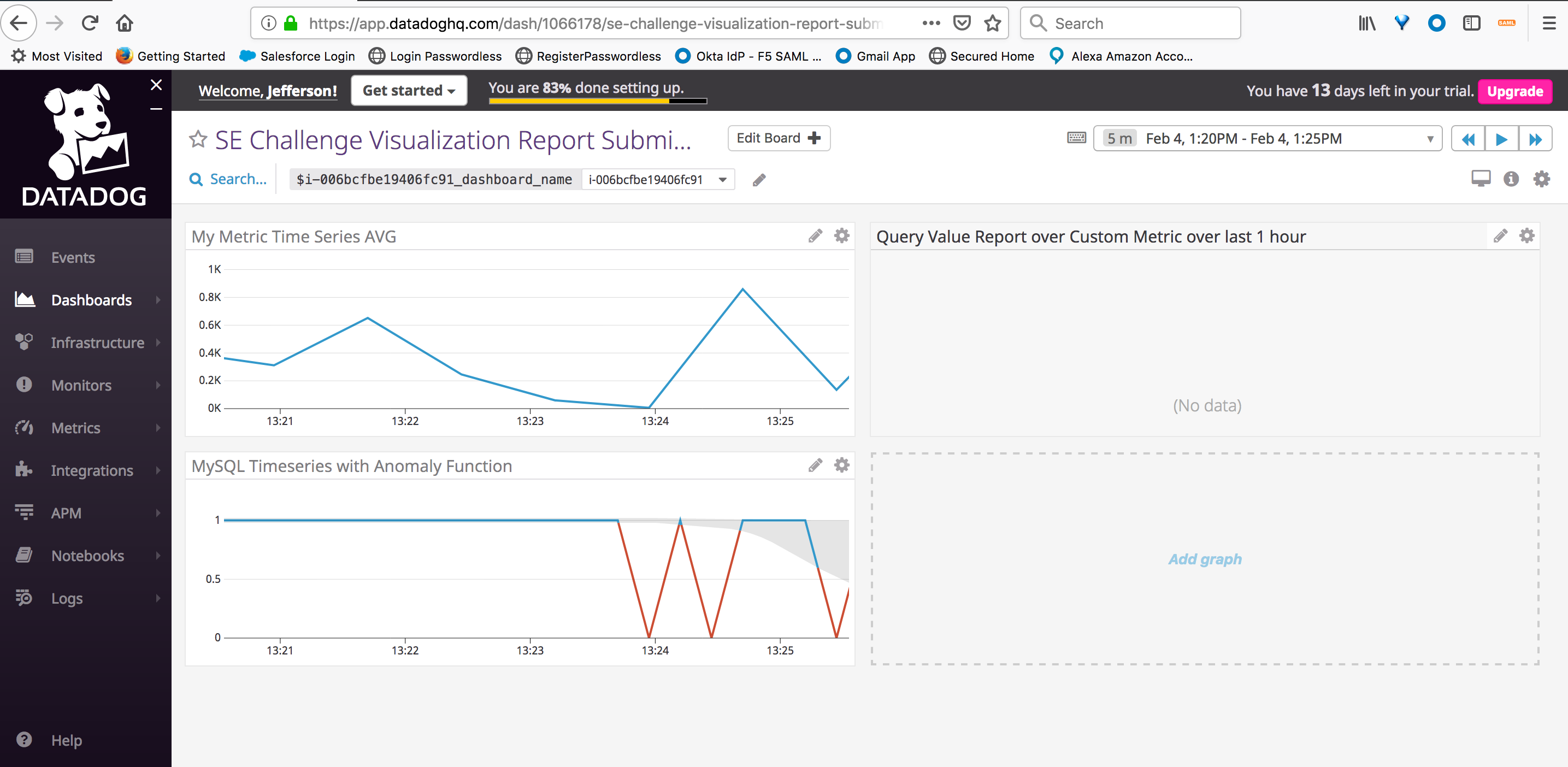
Once this is created, access the Dashboard from your Dashboard List in the UI:

Jefferson Haw: Please see Timeboard.png



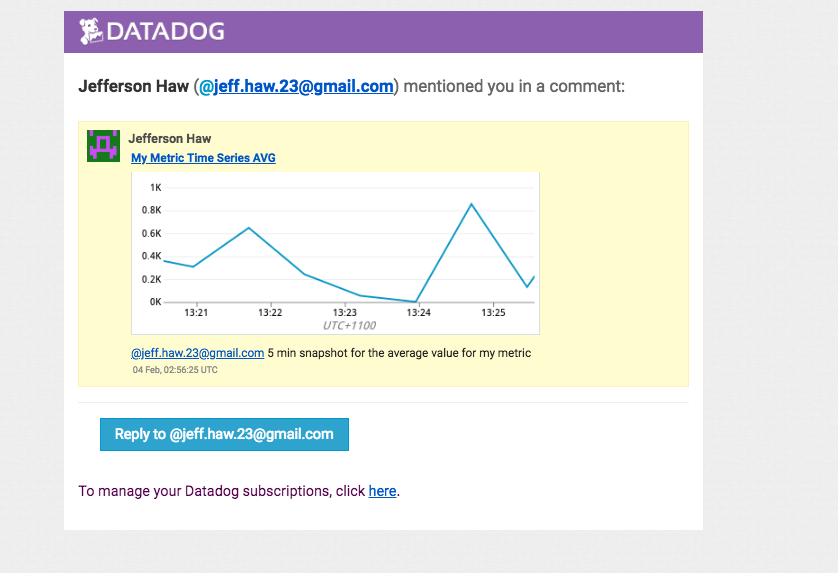
Set the Timeboard's timeframe to the past 5 minutes

Jeffeson Haw: Please see Timeboard\_5mins.png



Take a snapshot of this graph and use the @ notation to send it to yourself.

Jefferson Haw: Please see Graph Notification.png



Bonus Question: What is the Anomaly graph displaying?

Jefferson Haw: It tries to identify when a metric is behaving differently than it has in the past, taking into account trends, seasonal day-of-week, and time-of-day patterns. It is well-suited for metrics with strong trends and recurring patterns that are hard or impossible to monitor with threshold-based alerting.

**#Monitoring Data**

Since you’ve already caught your test metric going above 800 once, you don’t want to have to continually watch this dashboard to be alerted when it goes above 800 again. So let’s make life easier by creating a monitor.

Create a new Metric Monitor that watches the average of your custom metric (my\_metric) and will alert if it’s above the following values over the past 5 minutes:

Warning threshold of 500

Alerting threshold of 800

And also ensure that it will notify you if there is No Data for this query over the past 10m.

Please configure the monitor’s message so that it will:

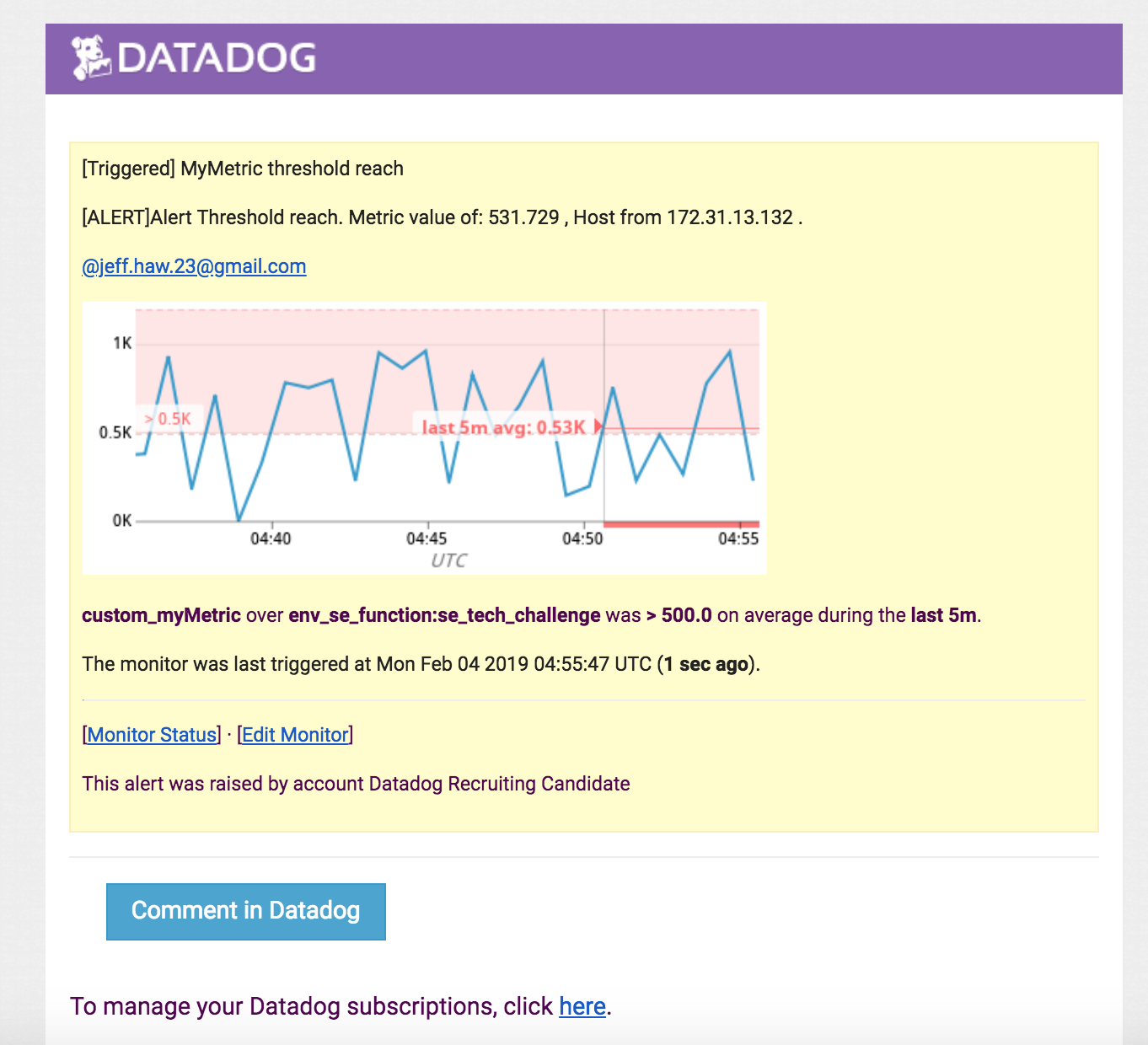
Jefferson Haw: Please see custom MetricMonitor.json

Send you an email whenever the monitor triggers.

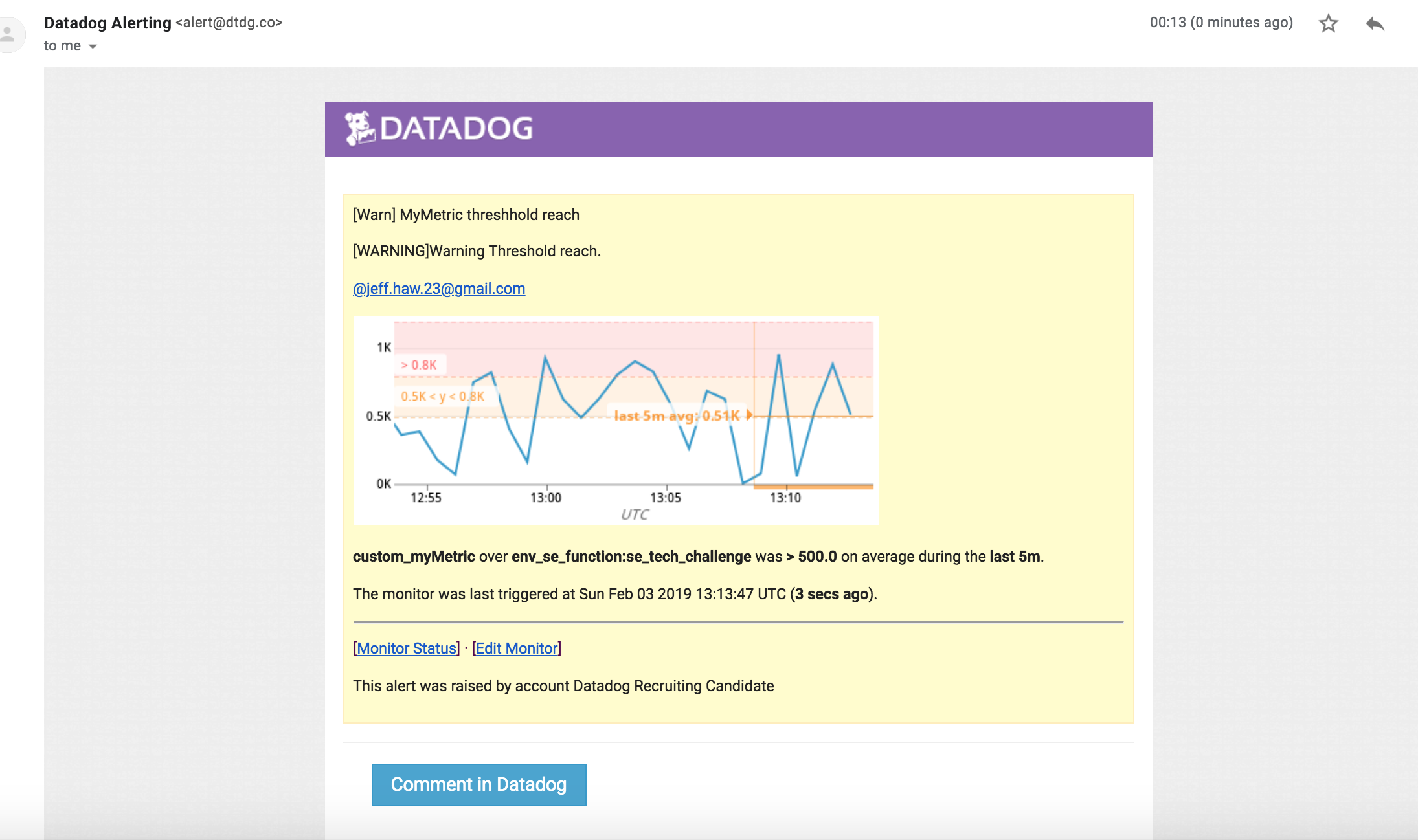
Create different messages based on whether the monitor is in an Alert, Warning, or No Data state.

Include the metric value that caused the monitor to trigger and host ip when the Monitor triggers an Alert state.

ALERT:



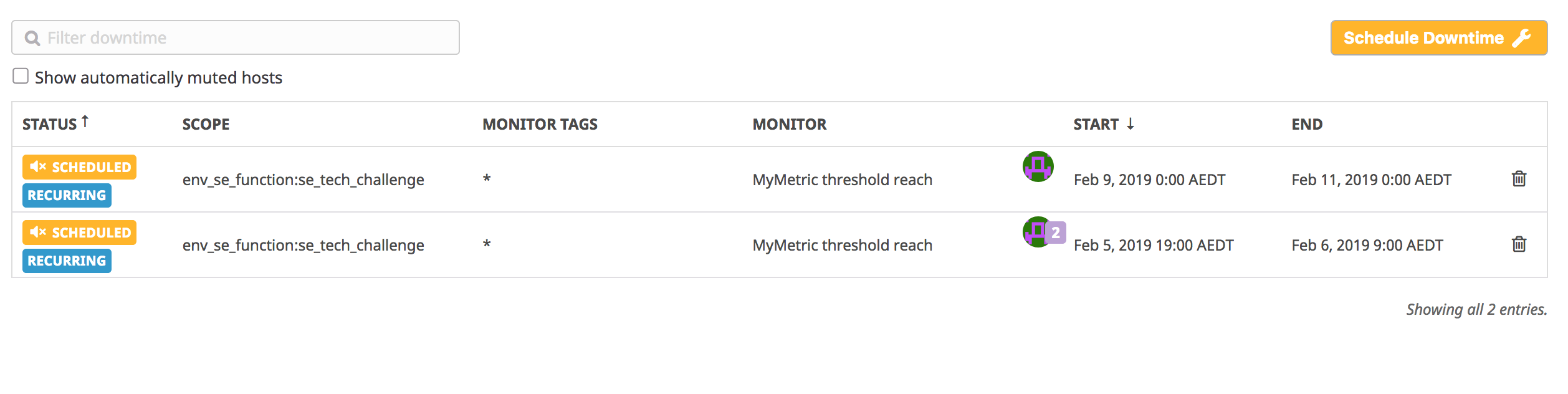
WARNING:



When this monitor sends you an email notification, take a screenshot of the email that it sends you.

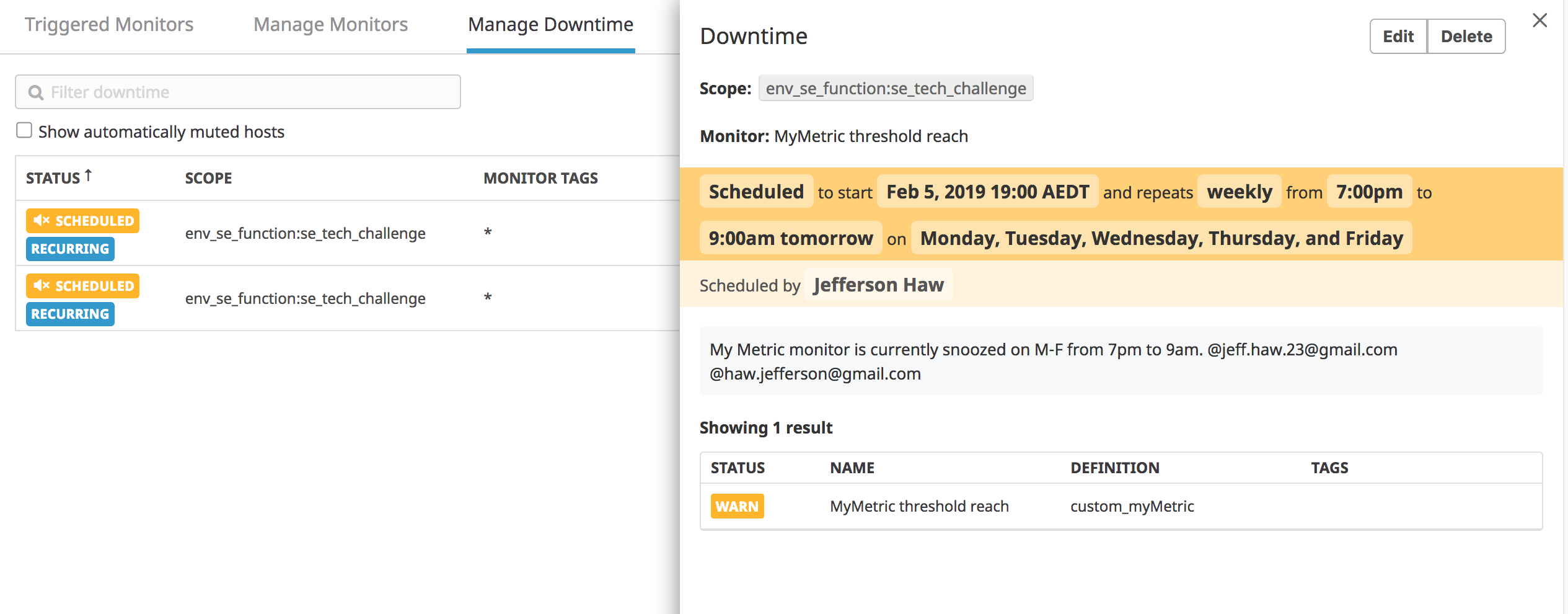
Bonus Question: Since this monitor is going to alert pretty often, you don’t want to be alerted when you are out of the office. Set up two scheduled downtimes for this monitor:

Jefferson Haw: Please see Downtime.png



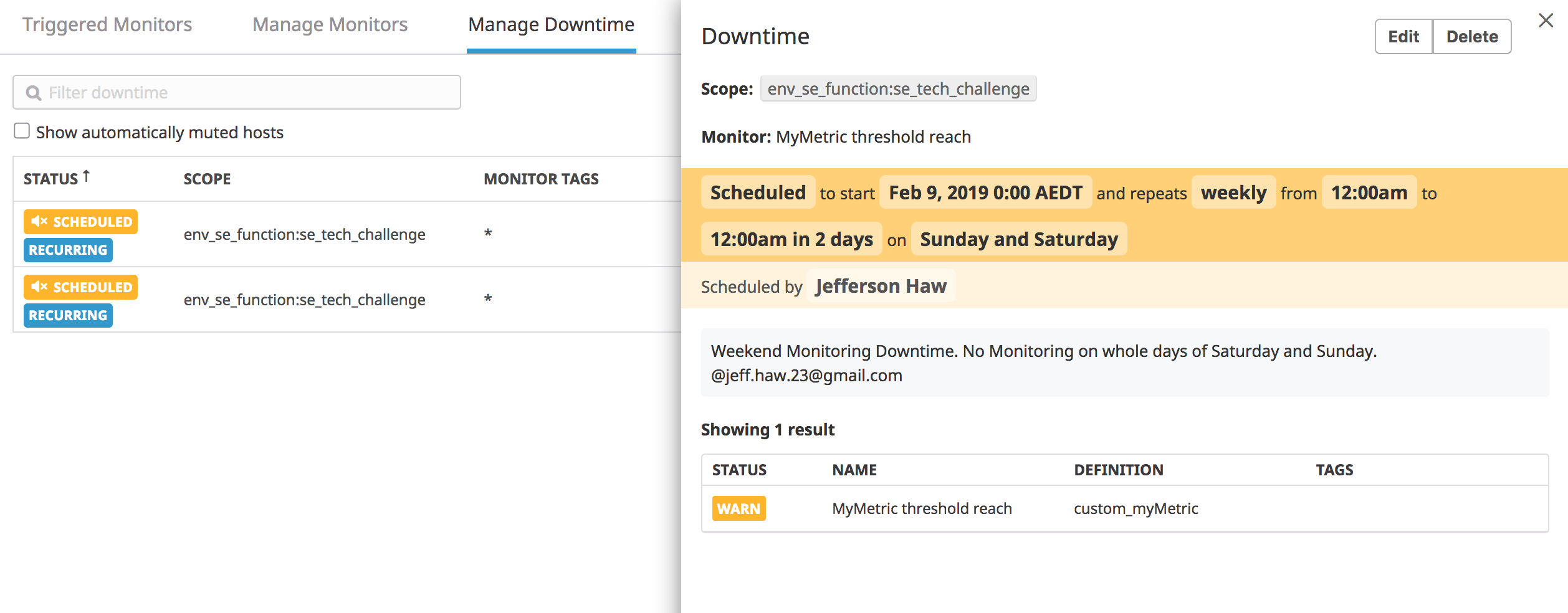
One that silences it from 7pm to 9am daily on M-F,

Jefferson Haw: Please see WeekdayDowntimeConfig.png



And one that silences it all day on Sat-Sun.

Jefferson Haw: Please see WeekendDowntimeConfig.png



Make sure that your email is notified when you schedule the downtime and take a screenshot of that notification.

Jefferson Haw: Please see SnoozeNotificationSample.png. Instead of waiting for the weekend/time for the weekday. I’ve just created a third one-time off downtime configuration to show how the notification should work.



**#Collecting APM Data:**

Note: Using both ddtrace-run and manually inserting the Middleware has been known to cause issues. Please only use one or the other.

Note: Using both ddtrace-run and manually inserting the Middleware has been known to cause issues. Please only use one or the other.

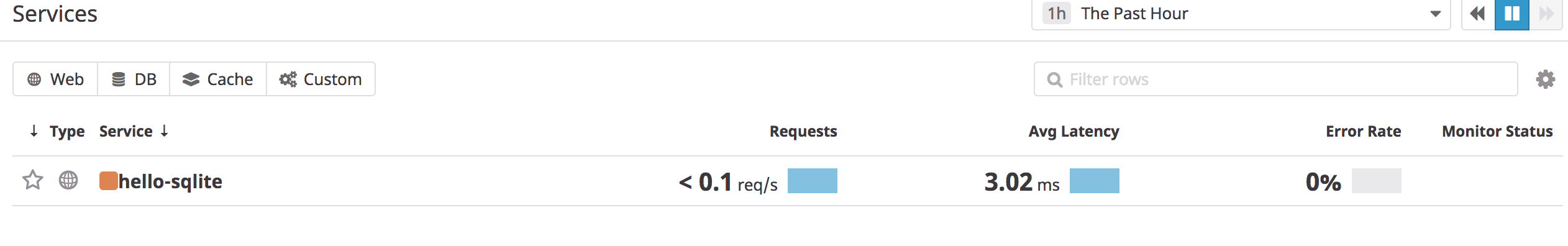
Jefferson Haw: Please see datadog.yaml where I’ve enabled APM.

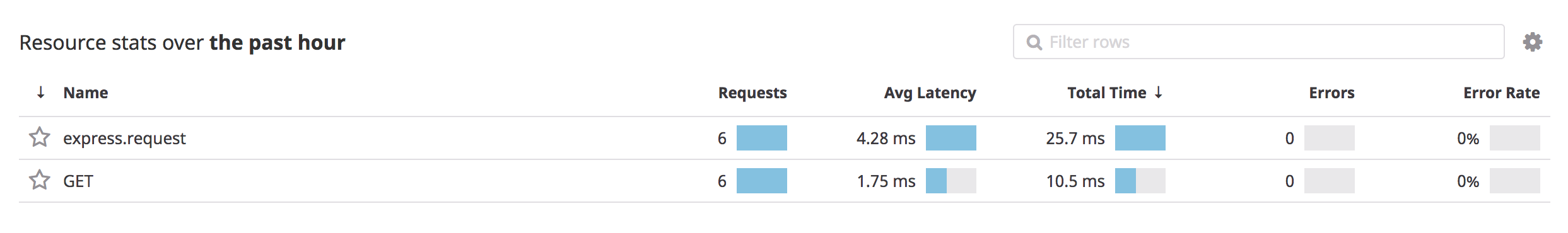
Bonus Question: What is the difference between a Service and a Resource?

Jefferson Haw: Service is the application or component itself. e.g. Web App, SQL Database

Resource is the operations managed within the Service. e.g. GET/POST/Servlet method on the Web App Service. SELECT SQL or INSERT SQL on the Database service.

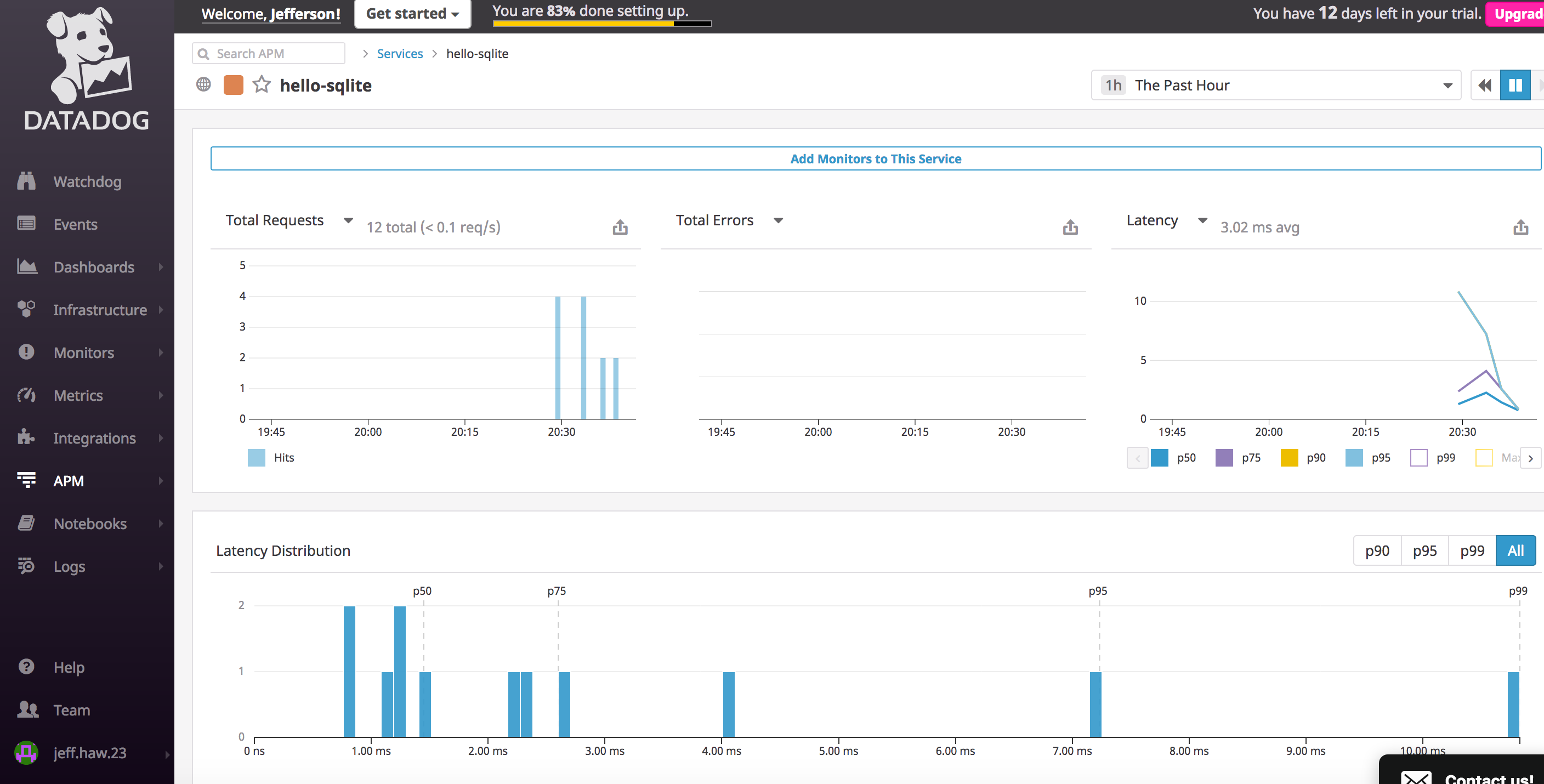
Service:



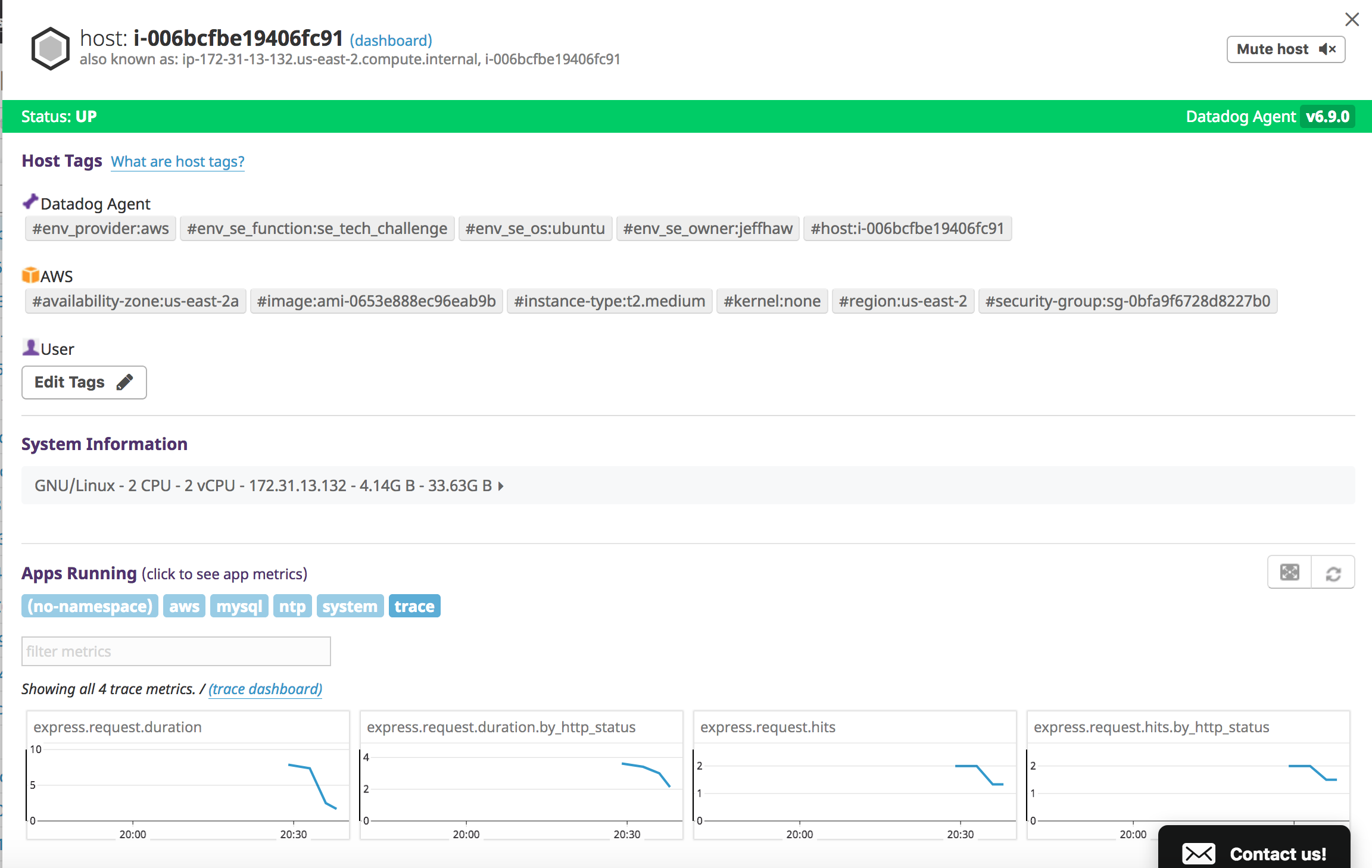
Resource:

Provide a link and a screenshot of a Dashboard with both APM and Infrastructure Metrics.

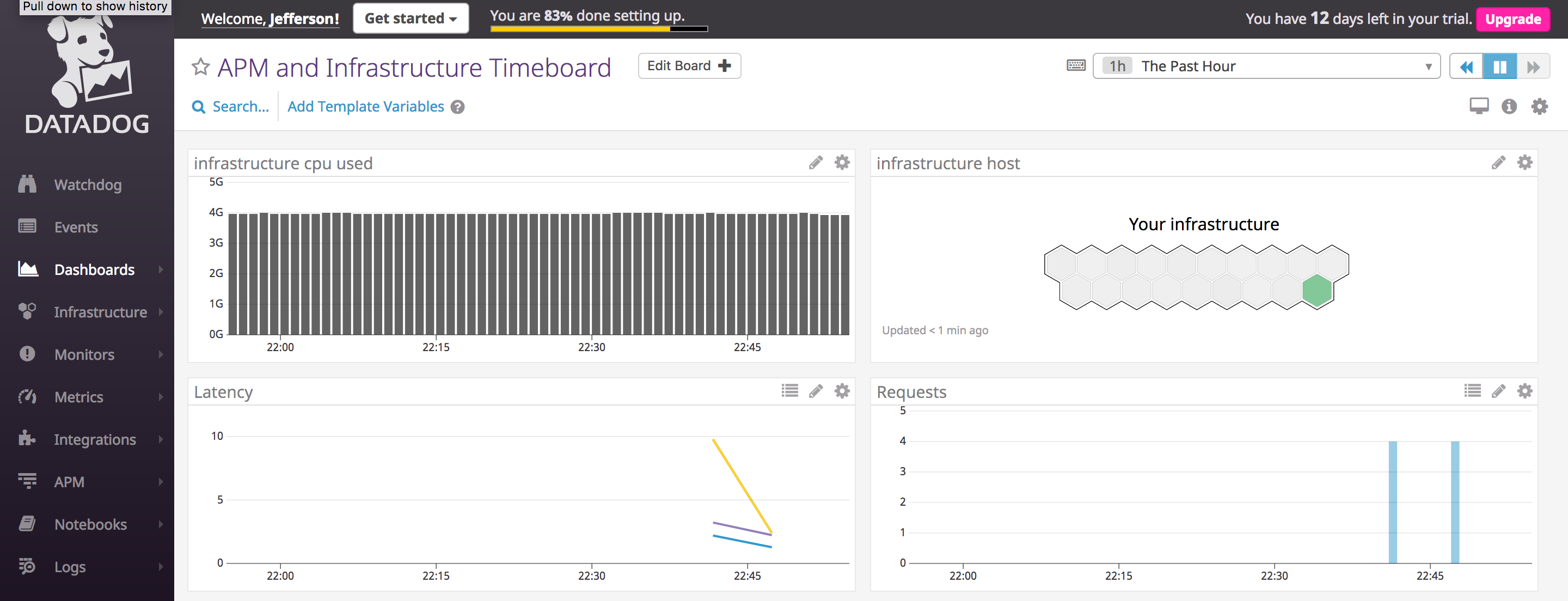
APM only:



Host Infrastructure:



Dashboard with APM and Infrastructure:



Please include your fully instrumented app in your submission, as well.

Jefferson Haw:

Please see nodejs\_datadog folder.

I've re-used a nodejs web application that allows you to submit information or data to the web application. By submitting the data/information you provided, the data gets stored on a SQLite backend.

You can access the web application via: <http://18.221.245.80:34936/>

**#Final Question:**

Datadog has been used in a lot of creative ways in the past. We’ve written some blog posts about using Datadog to monitor the NYC Subway System, Pokemon Go, and even office restroom availability!

Is there anything creative you would use Datadog for?

Jefferson Haw: Rather than answering this question in a traditional Monitoring way, I would attempt to use Datadog's engine to provide unique customer experience by allowing datadog to sift through http server, web appliation session logs that can monitor and analyze customer behavior or activity. Once a customer or end user fits on a certain metric or behavior defined within Datadog, Datadog can call or initiate webhooks that can provide a tailorized experience to the customer/end user. This could be as simple as sending a One Time offer in the form of SMS or email. This can also be triggering the current web/mobile application the customer is using for promotional offers or even as simple as providing relevant information base on what he/she has browse through during his activity period over time.