DATADOG

Bill Shelton

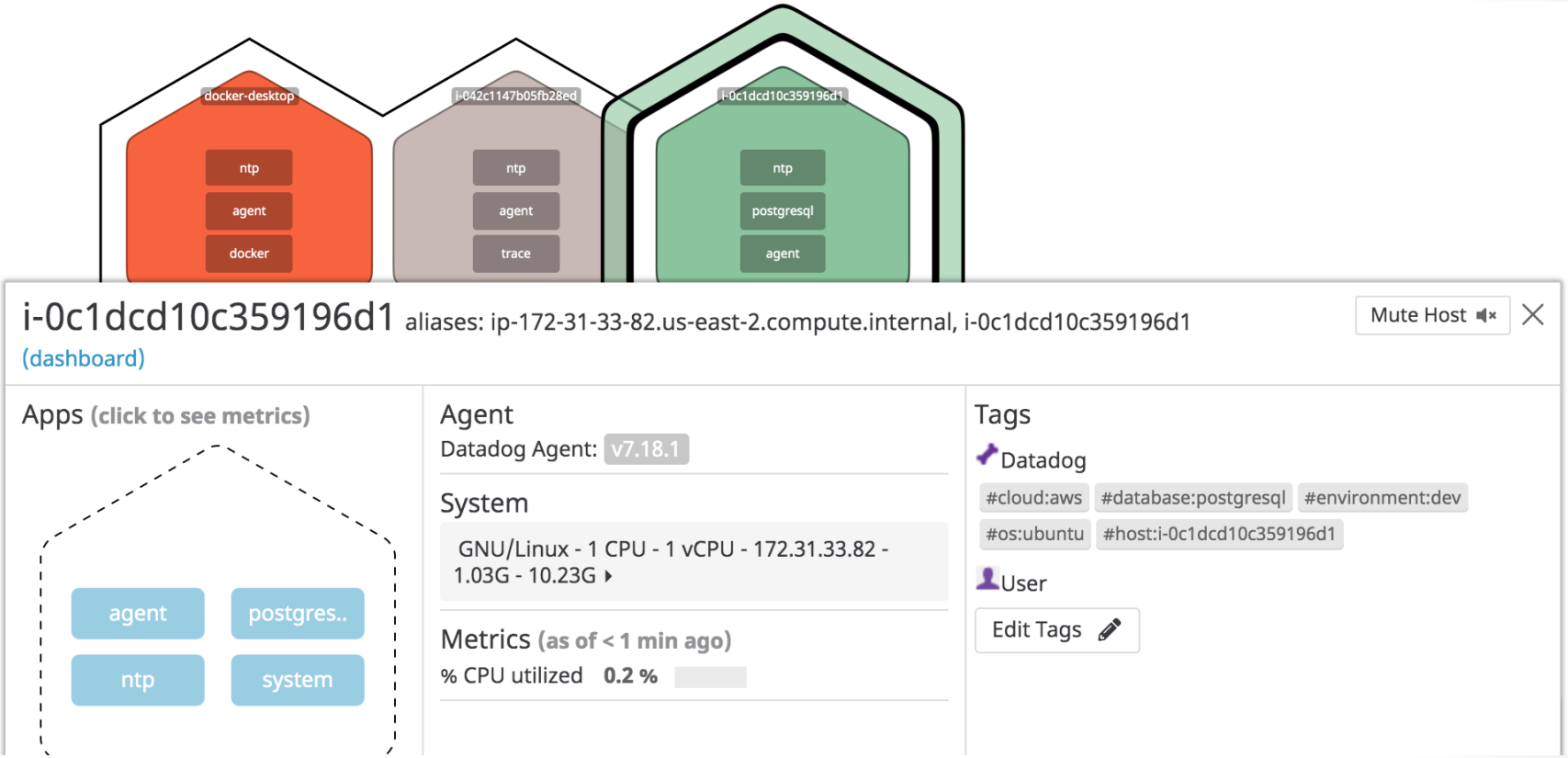
Solutions Engineering Candidate

Roadmap

1. **Collecting Metrics**
2. **Visualizing Data**
3. **Monitoring Data**
4. **Collecting APM Data**
5. **What Would You Use Datadog for?**
6. **Automatic Data Generation**

Collecting Metrics

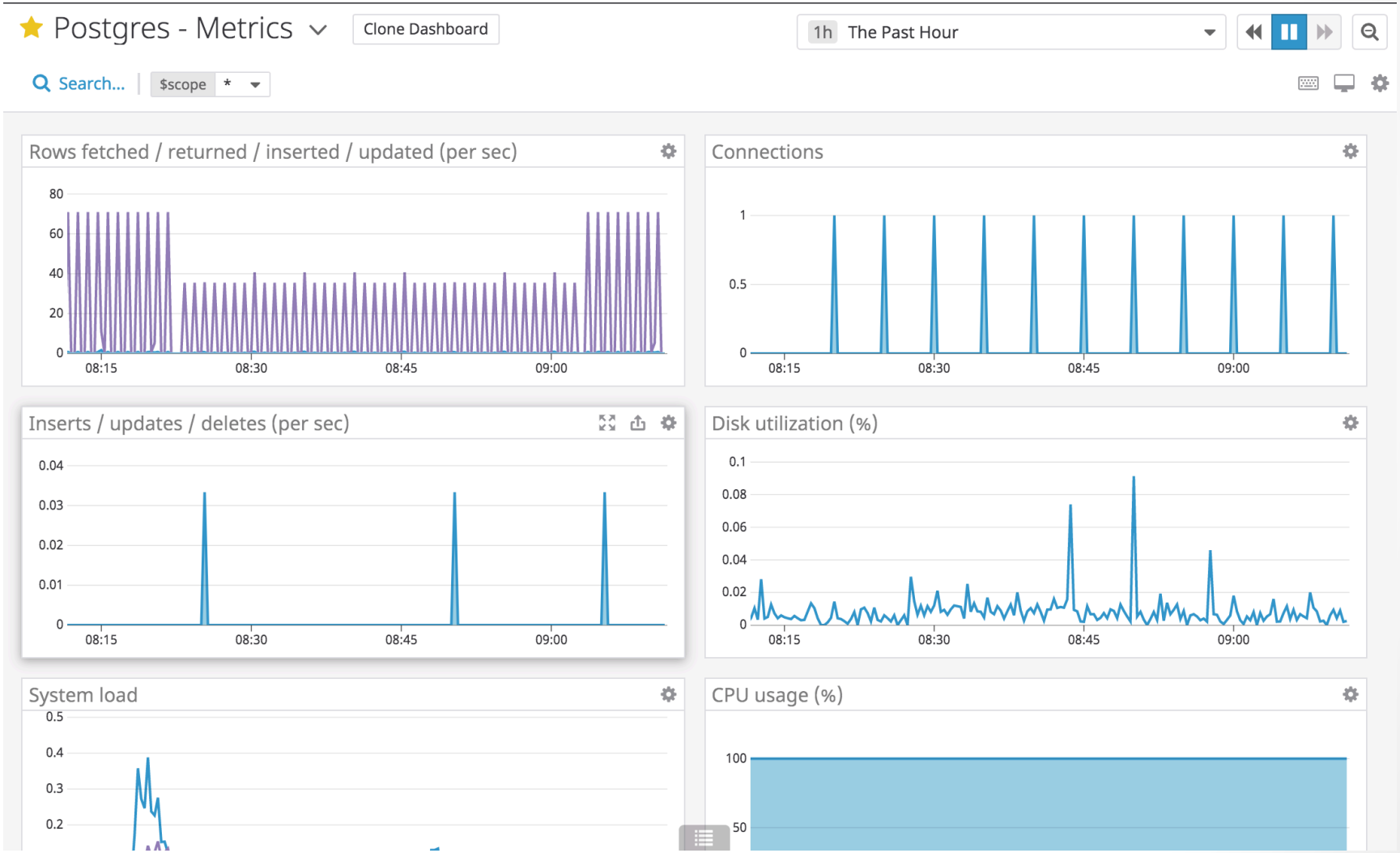
This image shows the three hosts that I installed the Datadog agent on. I installed the agent on a Docker container, and two Ubuntu AWS images. I then added relevant tags to differentiate and sort the hosts.



DATADOG

Collecting Metrics

This dashboard shows the information produced through the installation of Datadog monitoring on my Postgresql database.



DATADOG

Collecting Metrics

I implemented a python script that sent a random value between 0 and 1000 via the API to Datadog.

I also developed a script that called the data generating script every 45 seconds.

interval.py

```python
import schedule
import time
import os

def job():
  os.system('python custom_metric.py')

schedule.every(45).seconds.do(job)

while 1:
    schedule.run_pending()
    time.sleep(1)
```

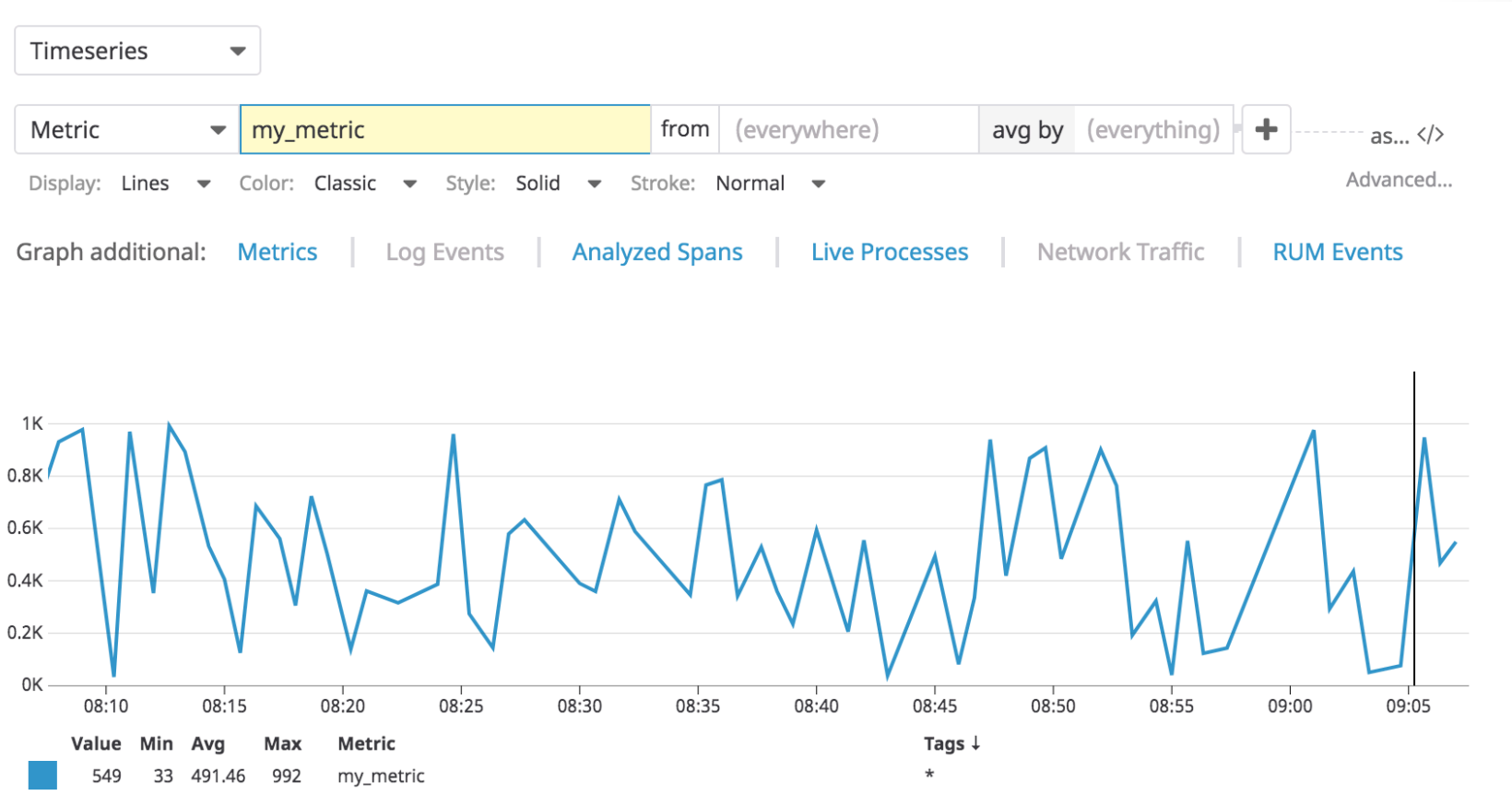
custom_metric.py

```python
from datadog import initialize, api
import random


with open("../config.json") as f:
    config = json.load(f)

options = {'api_key': config["api_key"],
           'app_key': config["app_key"],
           'api_host': 'https://api.datadoghq.com'}

initialize(**options)

random_value = random.randint(0,1000)

api.Metric.send(metric='my_metric', points=random_value)
```

DATADOG

Collecting Metrics

The graph below shows the data generated by the python script.



DATADOG

Visualizing Data

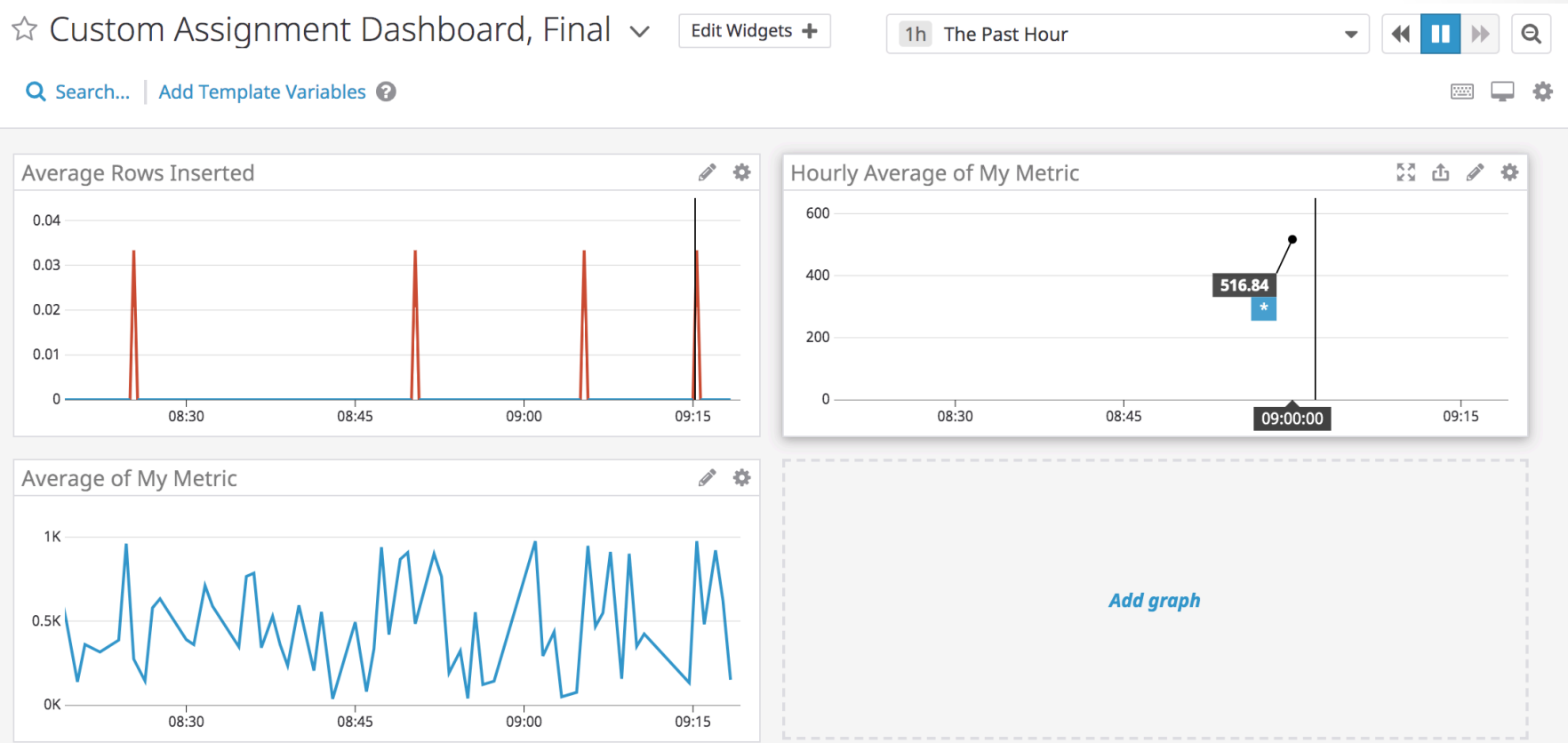
This python script generate a dashboard with a graph for the postgresql database with the anomaly function applied, my custom metric, and my custom metric with a one hour roll up.

```
import json
from datadog import initialize, api

with open("../../config.json") as f:
    config = json.load(f)

options = {
    'api_key': config["api_key"],
    'app_key': config["app_key"]
}

initialize(**options)

title = 'Custom Assignment Dashboard, Final'
widgets = [{'definition': {
        'type': 'timeseries',
        'requests': [
            {'q': "anomalies(avg:postgresql.rows_inserted{*}, 'basic', 2)"}
        ],
        'title': 'Average Rows Inserted'}},
        {'definition': {
        'type': 'timeseries',
        'requests': [
            {'q': 'avg:my_metric{*}.rollup(3600)'}
        ],
        'title': 'Hourly Average of My Metric'}},
        {'definition': {
        'type': 'timeseries',
        'requests': [
            {'q': 'avg:my_metric{*}'}
        ],
        'title': 'Average of My Metric'}}]
layout_type = 'ordered'
description = 'Final Assignment Dashboard'
is_read_only = False
notify_list = ['sheltowt@domain.com']

api.Dashboard.create(title=title,
                     widgets=widgets,
                     layout_type=layout_type,
                     description=description,
                     is_read_only=is_read_only,
                     notify_list=notify_list)
```

DATADOG

Visualizing Data

API generated custom dashboard



The anomaly graph is displaying rows inserted into the database with the basic anomaly algorithm applied and a set of two bounds. The basic algorithm uses a lagging rolling quantile computation to determine the range of expected values.

DATADOG

Monitoring Data

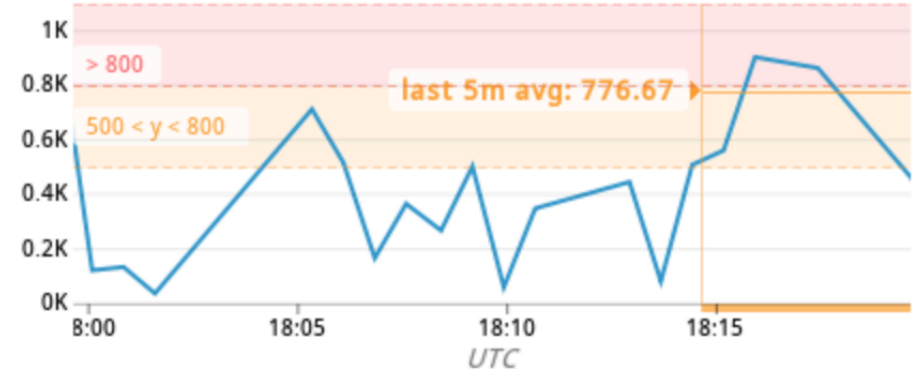
The email below was triggered because my_metric had a value over 500 for a period greater than five minutes.

[Warn] Custom Metric, Random Values Notification

@sheltowt@gmail.com

This message is a WARNING for my custom metric

The value that triggered this email is 776.667 and my host is ip-172-31-25-253




```
avg(last_5m):avg:my_metric{host:ip-172-31-25-253} > 800
```

The monitor was last triggered at Wed Apr 01 2020 18:19:48 UTC.

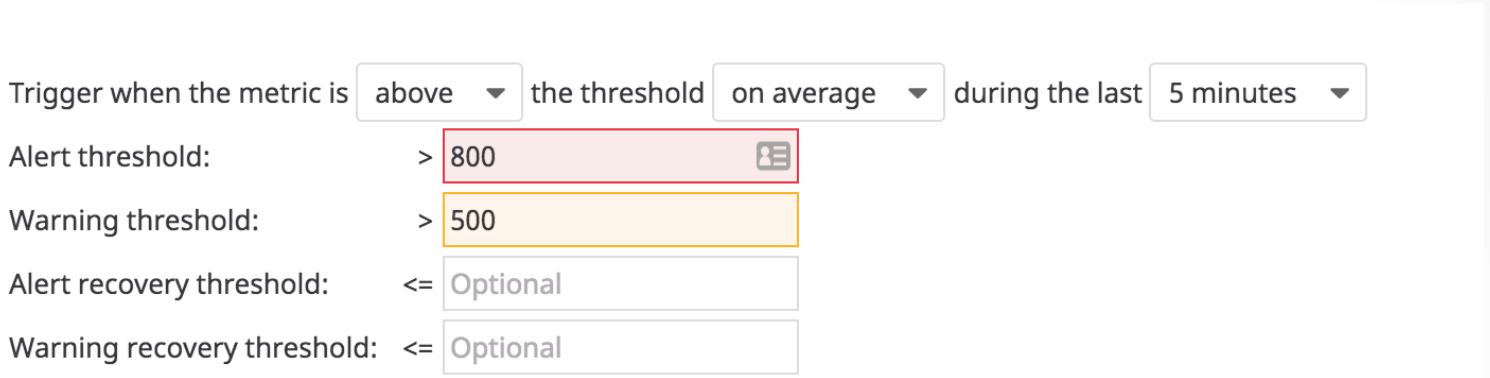
[Monitor Status] · [Edit Monitor] · [Show Processes]

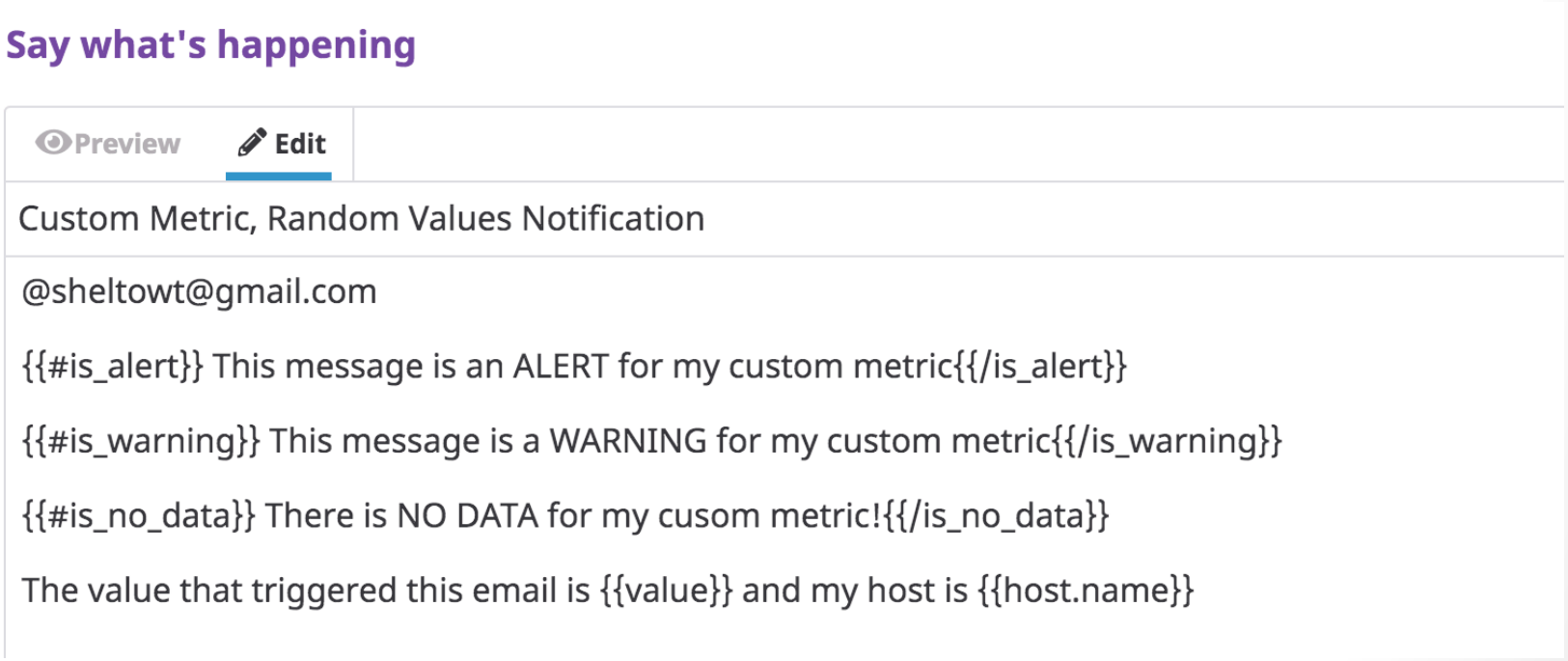
This alert was raised by account Datadog Recruiting Candidate

DATADOG

Monitoring Data

The images below show the configuration of the thresholds and email templates.

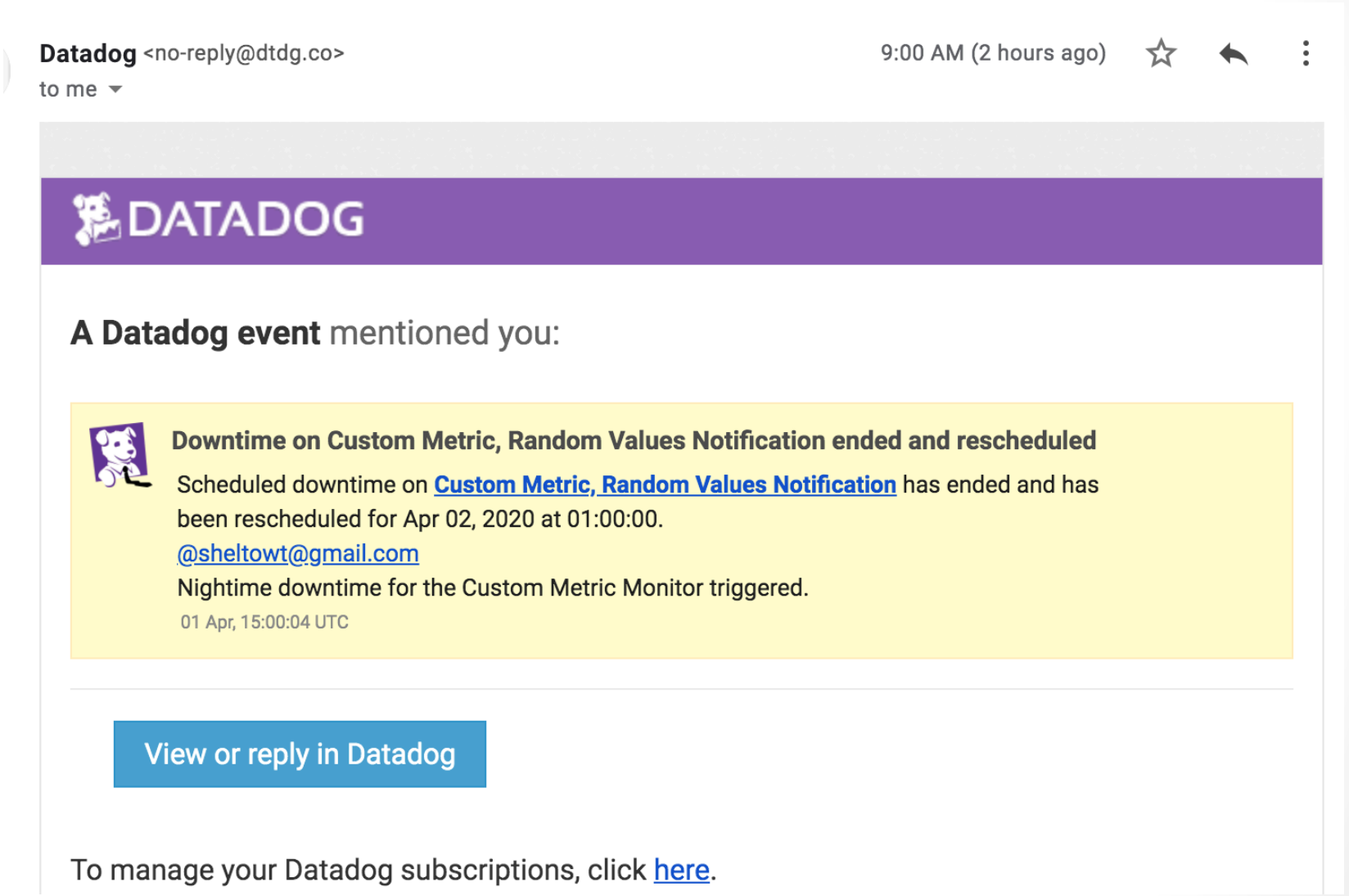




DATADOG

Monitoring Data

The image below shows the email triggered by scheduled downtime ending.



DATADOG

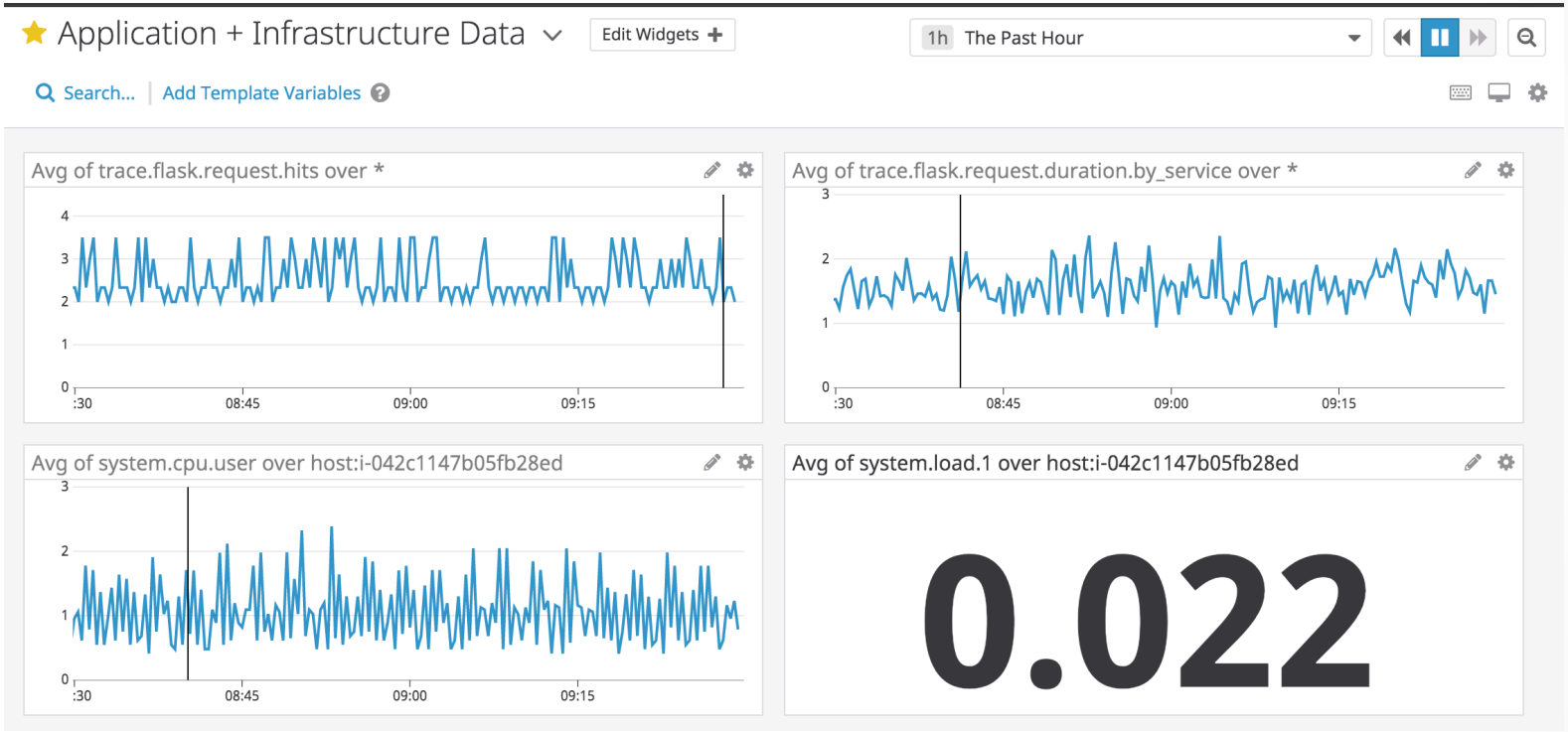
Collecting APM Data

This script instruments the APM functionality within a Flask API. I installed an Nginx server that pointed to the Flask API so I could generate dummy data both locally and automatically from another cloud server.

```
from flask import Flask
import logging
import sys
import ddtrace.profile.auto
ddtrace.config.analytics_enabled = True

# Have flask use stdout as the logger
main_logger = logging.getLogger()
main_logger.setLevel(logging.DEBUG)
c = logging.StreamHandler(sys.stdout)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
c.setFormatter(formatter)
main_logger.addHandler(c)

app = Flask(__name__)

@app.route('/')
def api_entry():
    return 'Entrypoint to the Application'

@app.route('/api/apm')
def apm_endpoint():
    return 'Getting APM Started'

@app.route('/api/trace')
def trace_endpoint():
    return 'Posting Traces'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port='5050')
```

Collecting APM Data

These graphs show data generated from the Flask app in terms of requests and request duration, as well as CPU and system load information from the server running the application.



A resource supports a specific piece of data such as a user of an application. REST semantics structure a set of operations that can take place on the piece of data such as updating, creating, deleting and retrieving.

A service is a software implementation that supports a business relevant functionality, such as an authentication service.

DATADOG

What Would You Use Datadog For?

I would place an AWS IoT button next to the sink and track the frequency with which I am washing my hands.

I would set an alert if the button was not triggered within a 3 hour time frame during the hours of 7am to 10pm. ☺

DATADOG

Automatic Data Generation

This script executes a recursive function that either queries or inserts data into the database.

database_activity.js

```
const { Pool, Client } = require('pg')
config = require('../../config.json')

const pool = new Pool({
  user: config.database_user,
  host: 'localhost',
  database: 'billdb',
  password: config.database_password,
  port: 5432,
})

pool.query('CREATE TABLE IF NOT EXISTS dummy_data(user_info text, action text, reason text);', (err, res) => {
  if (err) {
    console.log(err.stack)
  } else {
    console.log(res.rows[0])
  }
})

function insertData(){
  pool.query("INSERT INTO dummy_data(user_info, action, reason)VALUES('billsinfo', 'wrotecode', 'solvedproblem');", (err, res) => {
    if (err) {
        console.log(err.stack)
      } else {
        console.log(res.rows[0])
      }
  })
}

function queryData(){
  pool.query('SELECT * FROM dummy_data', (err, res) => {
    if (err) {
        console.log(err.stack)
      } else {
        console.log(res.rows[0])
      }
  })
}

function getRandomInt(max) {
  return Math.floor(Math.random() * Math.floor(max));
}

function recursiveLoop() {
  random_int = getRandomInt(2)
  if (random_int == 0) {
    insertData()
  } else {
    queryData()
  }
  setTimeout(recursiveLoop, 30000);
}

recursiveLoop()
```

DATADOG

Automatic Data Generation

This script executes a recursive function that hits one of the endpoints on the application monitored with APM.

webserver_activity.js

```
const request = require('request');

function hitIndex(){
    request('http://3.21.236.69/', (err, res, body) => {
      if (err) { return console.log(err); }
      console.log("query to '/'")
    });
}

function hitApiApm(){
    request('http://3.21.236.69/api/apm', (err, res, body) => {
      if (err) { return console.log(err); }
      console.log("query to '/api/apm'")
    });
}

function hitApiTrace(){
    request('http://3.21.236.69/api/trace', (err, res, body) => {
      if (err) { return console.log(err); }
      console.log("query to '/api/trace'")
    });
}

function getRandomInt(max) {
  return Math.floor(Math.random() * Math.floor(max));
}

function recursiveLoop() {
    random_int = getRandomInt(3)
    if (random_int == 0) {
        hitIndex()
    } else if (random_int == 1) {
        hitApiApm()
    } else {
        hitApiTrace()
    }
    setTimeout(recursiveLoop, 3000);
}

recursiveLoop()
```

DATADOG

GitHub Repos

1. https://github.com/sheltowt/hiring-engineers/tree/bill_shelton