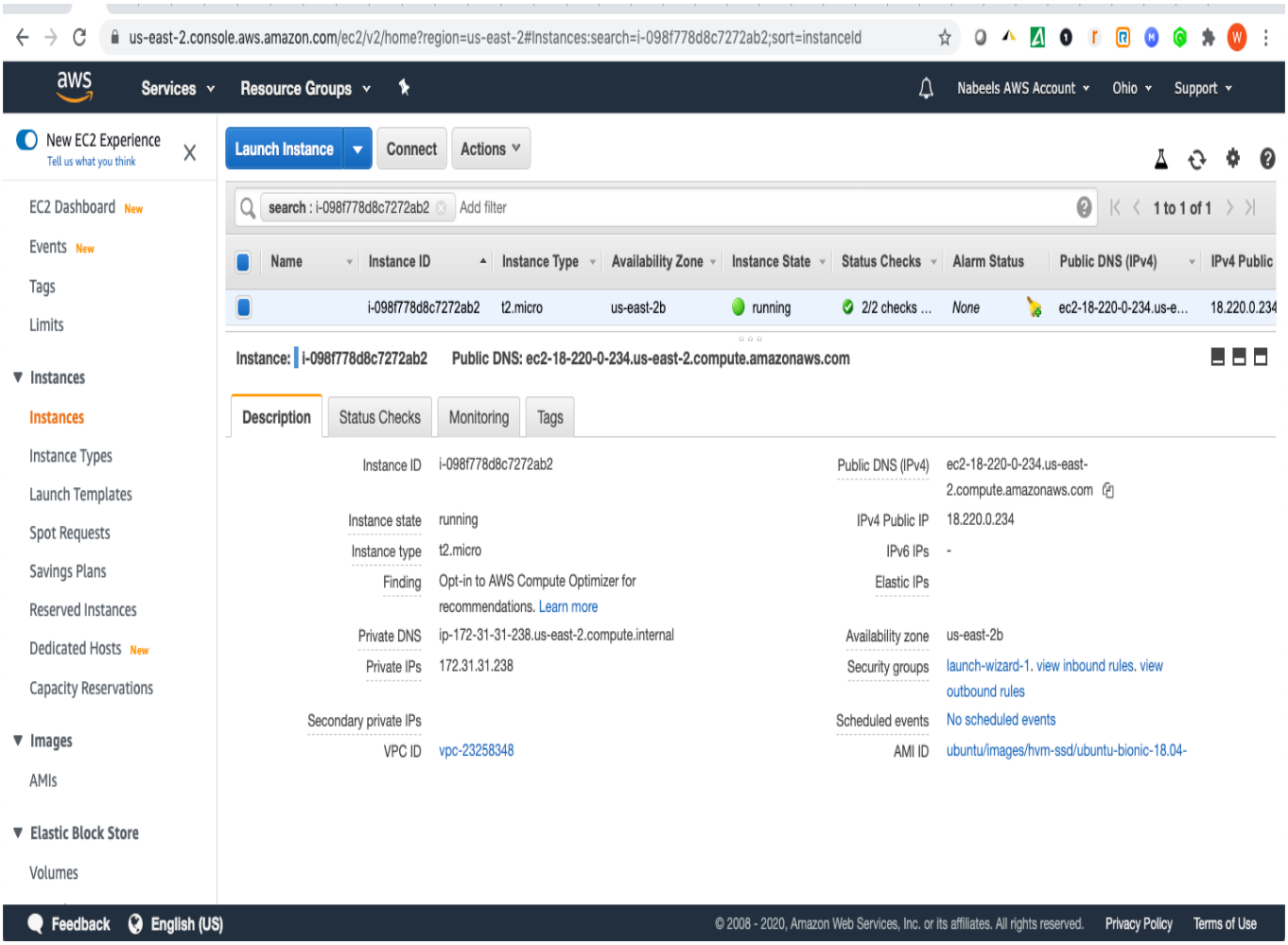

# DataDog Technical Assignment

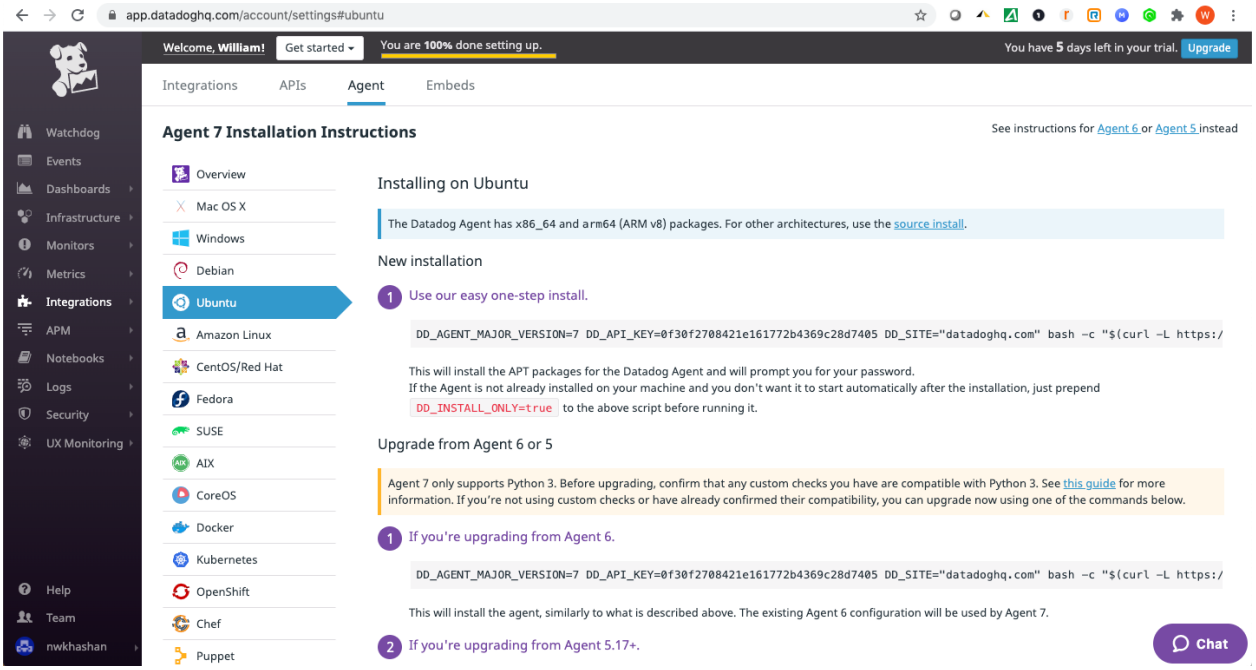
## Nabeel William Khashan

# Prerequisites – Setup the environment

I used the Parallels instance as a test bed and I decided to spin up an AWS instance using Ubuntu 18.04. The exercise was done on AWS.




The next step was to sign up for a DataDog account with the title “DataDog Recruiting Candidate”. This was quite intuitive and easy to do. Setting up Ubuntu to collect system metrics was easy to set up as well.

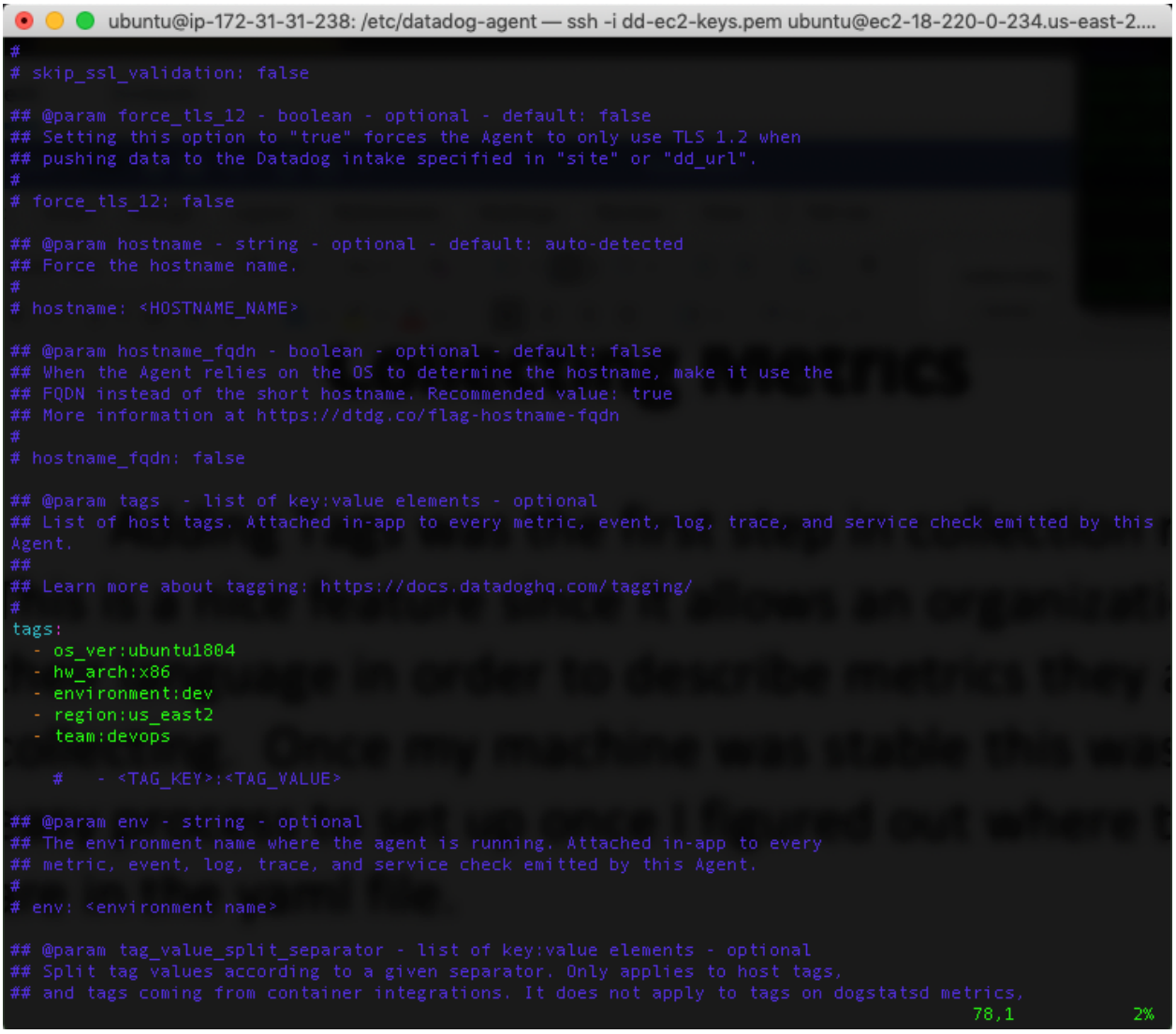



# Collecting Metrics

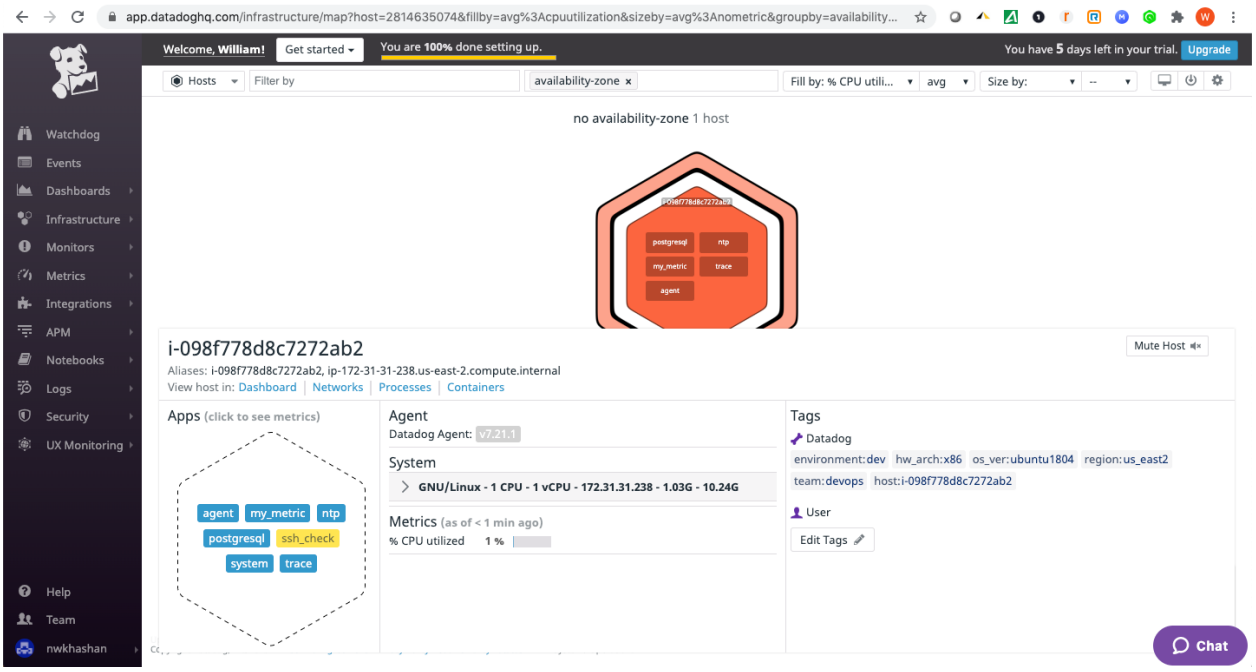
Adding Tags was the first step in collection metrics. This is a nice feature since it allows an organization to add their language in order to describe metrics they are collecting. Once my machine was stable this was quite an easy process to set up once I figured out where the tags are in the datadog.yaml file. In the datadog.yaml file I added the following tags.

Tags:

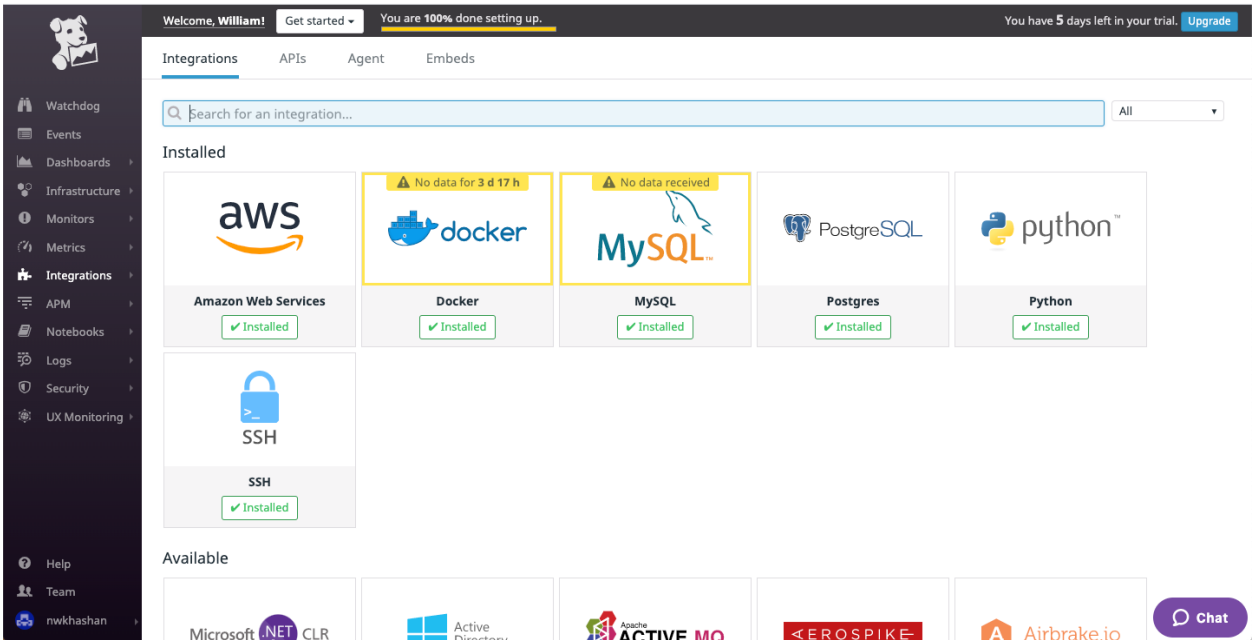
- Os_vert:ubuntu1804
- hw_arch:x86
- environment:dev
- region:us_east2
- team:devops



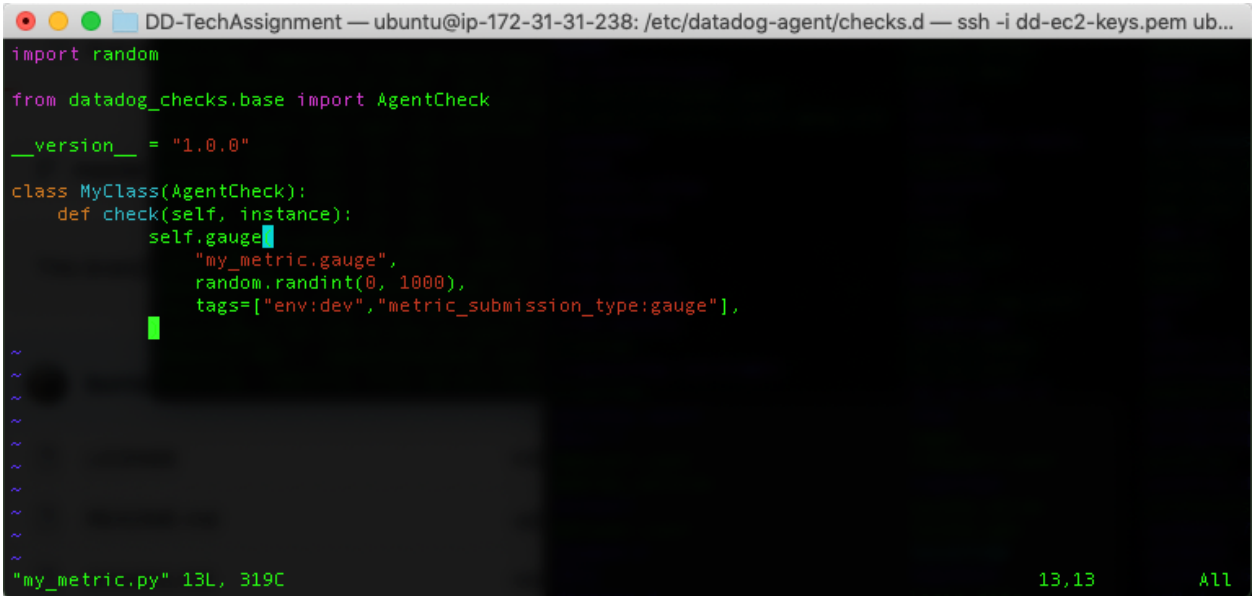

Below I am showing the Datadog console where the tags are appearing in the Host Map.



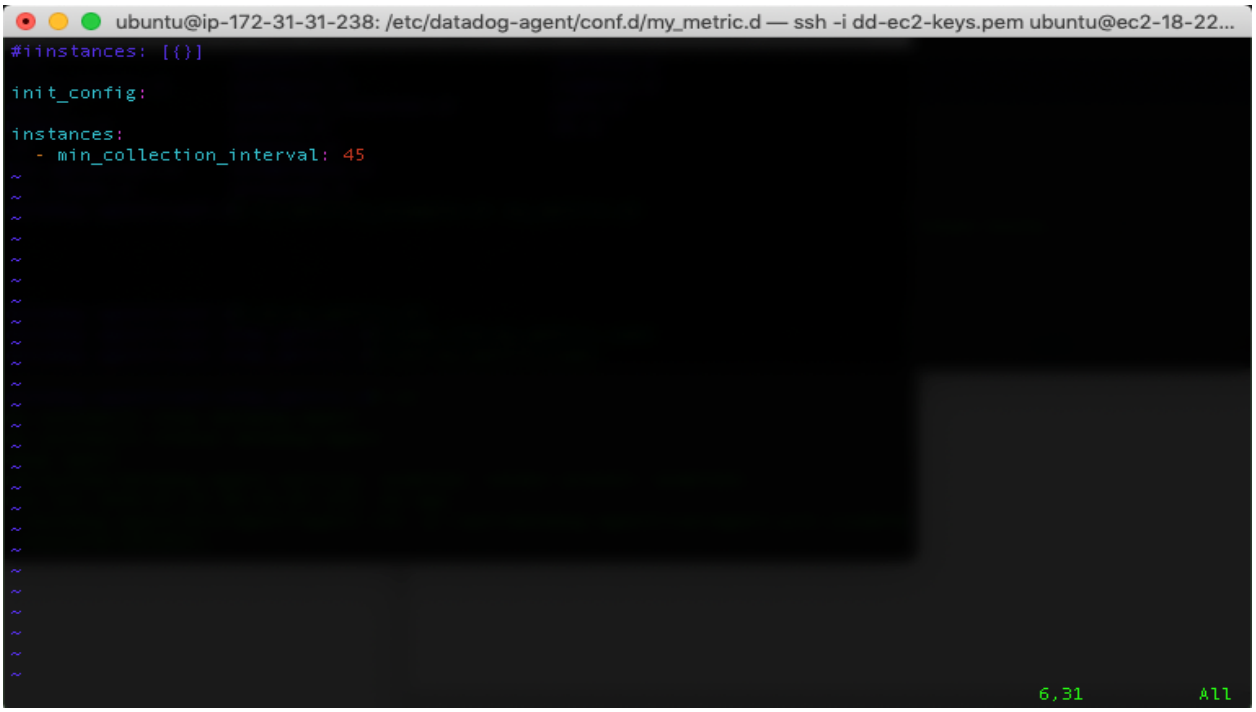

The next step was to install a database on my machine. I initially started out with MySQL using docker. I decided to use PostgreSQL on my AWS EC2 instance.




The next step is to create a customer Agent check that submits a metric called my_metric with a random value of 0 to 1000. Please see below for script.



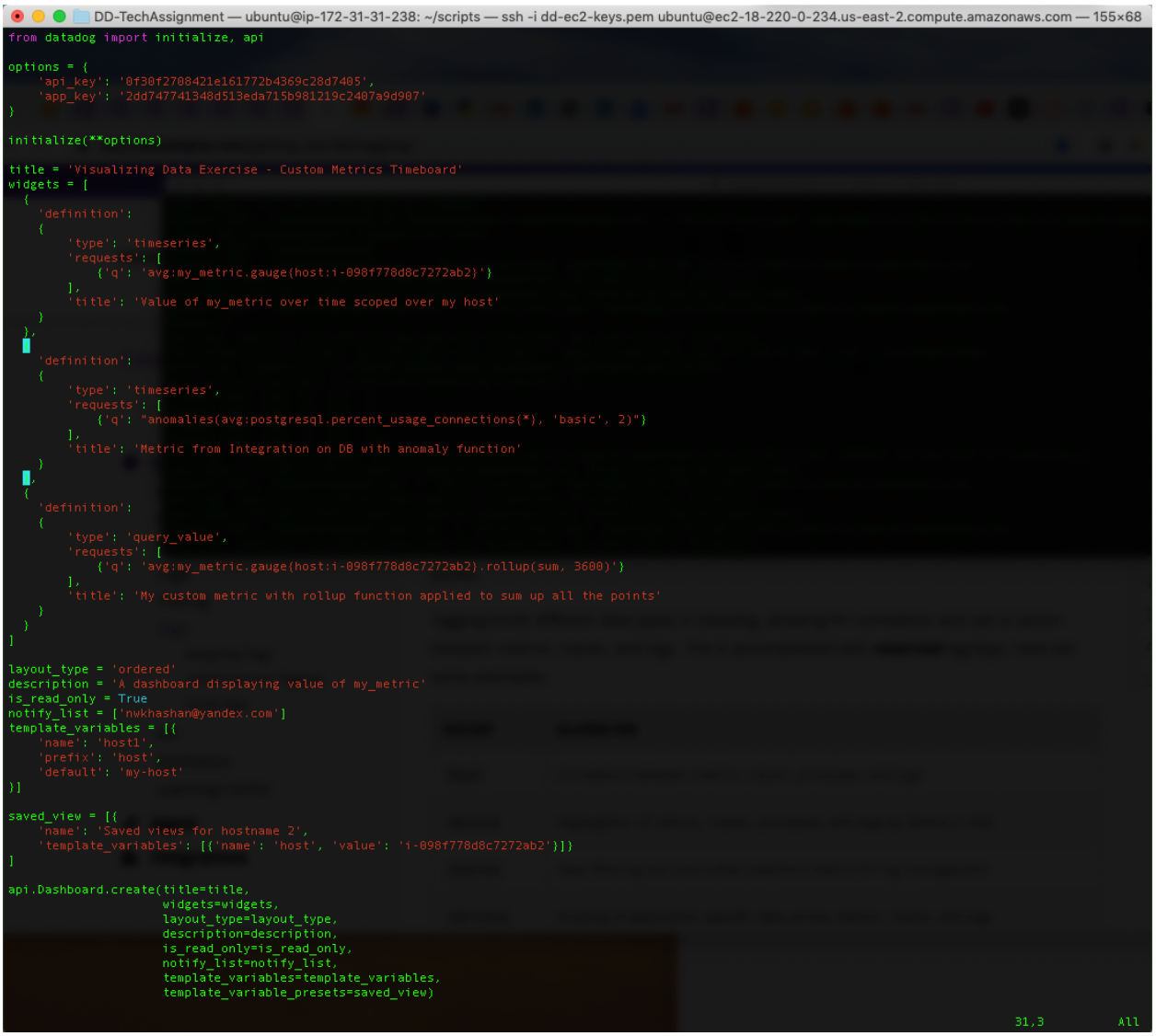

The next step is to check the collections interval every 45 seconds.




**Bonus Question:** Can you change the collection interval without modifying the Python check file you created? Answer: Yes, you can change it globally by editing the config file config file `~/.datadog-agent/conf.d/my_metric.d/mymetric.yaml` as shown above.

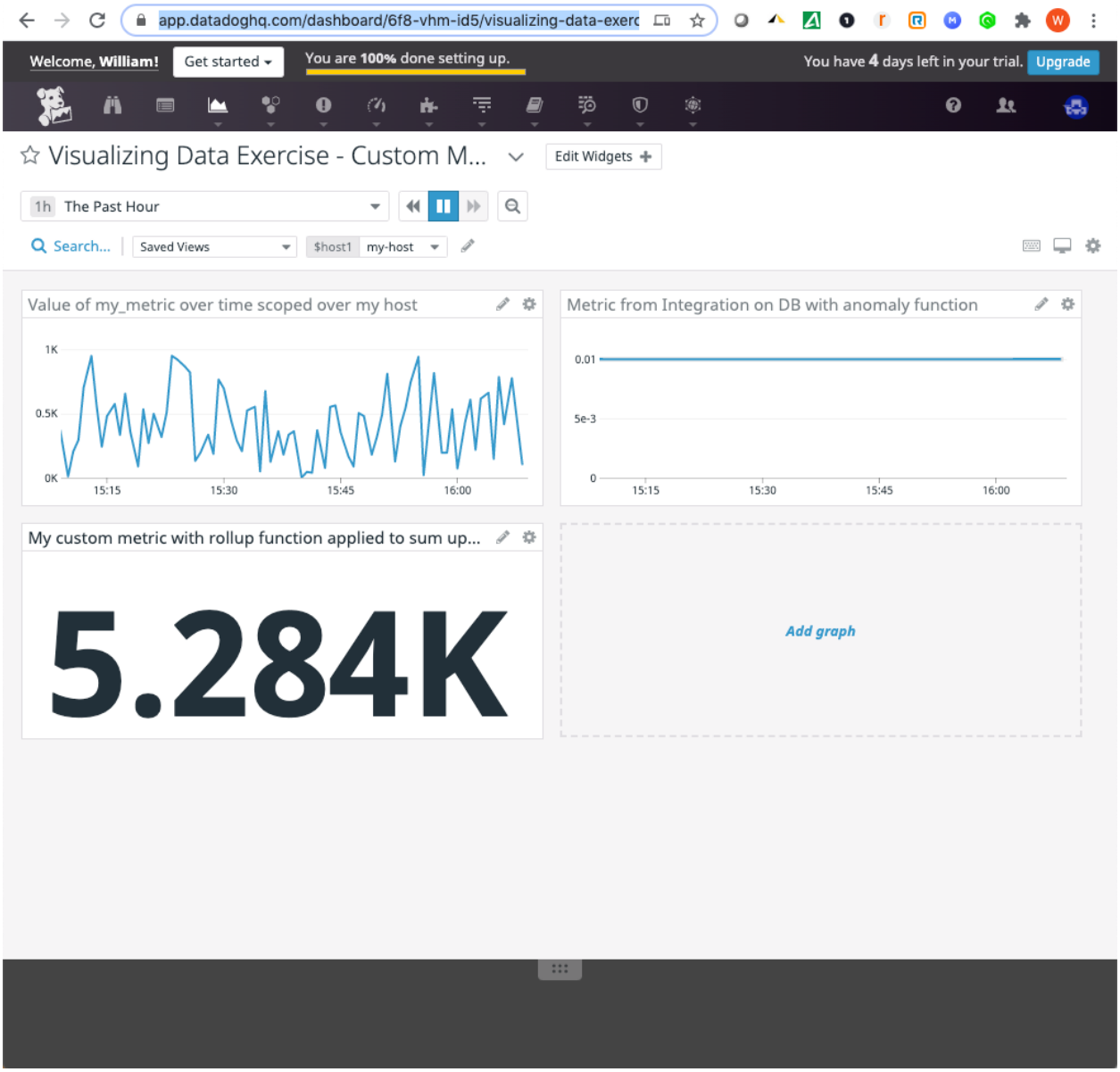
# Visualizing Data

Utilize the Datadog API to create a Timeboard that contains the customer metric created over the my host.  Below is the script.

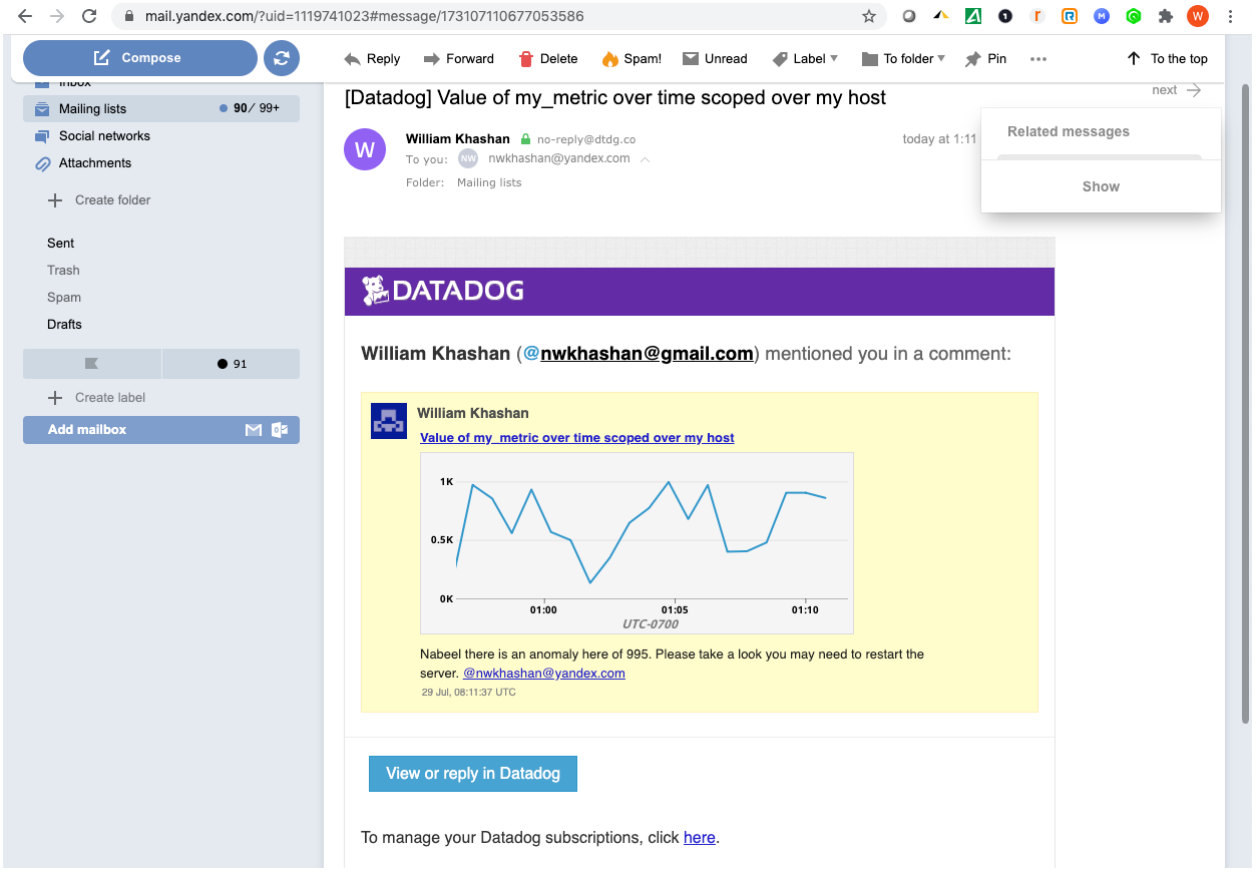



The script uses percentage usage connections as the metric for the anomaly function that is applied.  The last function is the rollup function applied to sum up all the points for the past hour in one bucket is show in the script as well.  The link below shows you the Timeboard created by utilizing the Datadog API.

https://app.datadoghq.com/dashboard/6f8-vhm-id5/visualizing-data-exercise---custom-metrics-timeboard?from_ts=1595974102921&to_ts=1595977702921&live=true


app.datadoghq.com/dashboard/6f8-vhm-id5/visualizing-data-exerc
Welcome, William!
Get started
You are 100% done setting up.
You have 4 days left in your trial.
Upgrade
Visualizing Data Exercise - Custom M...
Edit Widgets
1h
The Past Hour
Search...
Saved Views
$host1
my-host
Value of my_metric over time scoped over my host
1K
0.5K
0K
15:15
15:30
15:45
16:00
Metric from Integration on DB with anomaly function
0.01
5e-3
0
15:15
15:30
15:45
16:00
My custom metric with rollup function applied to sum up...
5.284K
Add graph

The following shows the Timeboard's timeframe to the past 5 minutes.

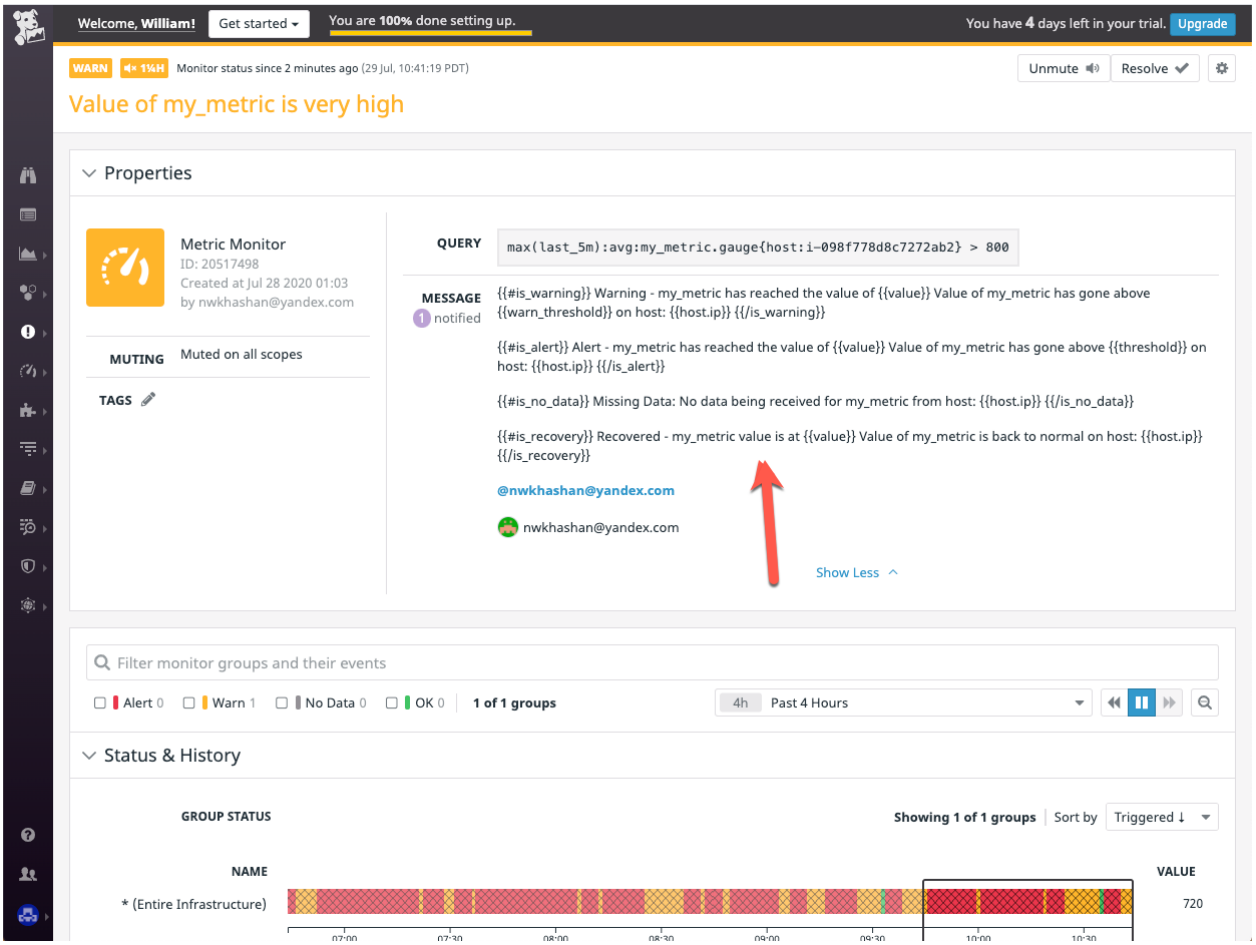



**Bonus Question:** What is the Anomaly graph displaying?

The anomaly graph is used to highlight any data points part of time series data, that are observed to be anomalies or deviations from a set of data points that are considered normal behavior.

# Monitoring Data

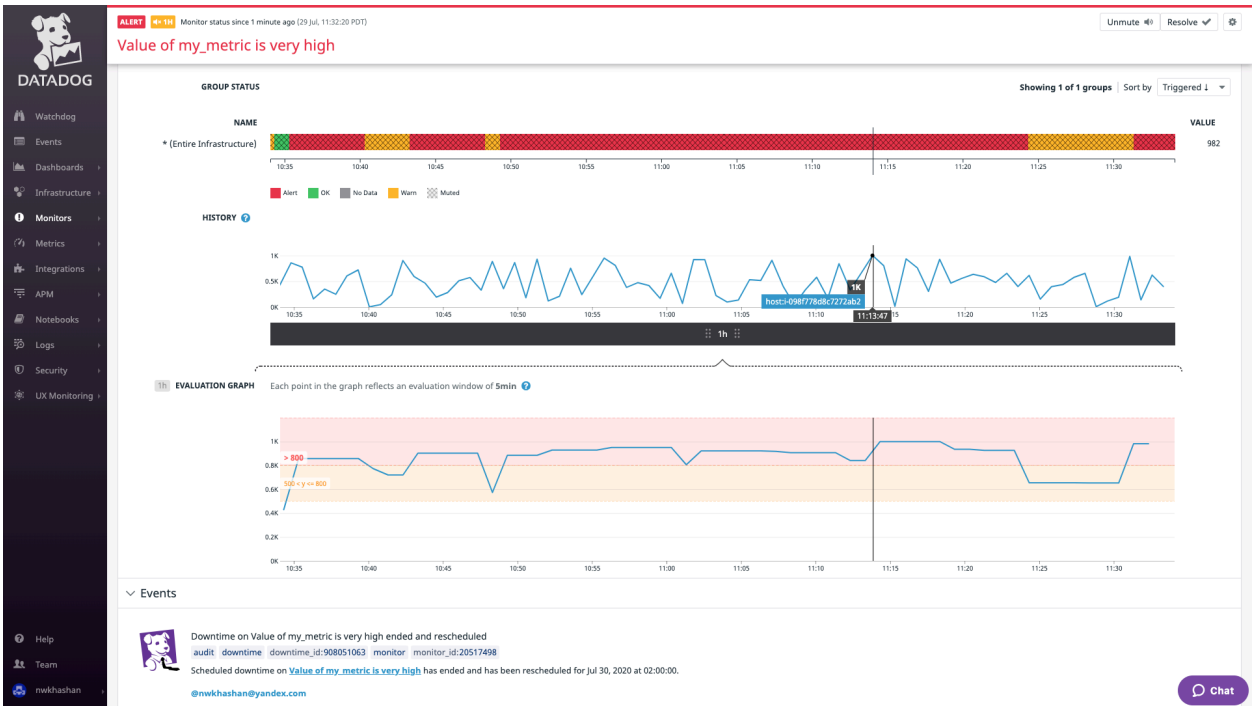
The data below highlights the monitors for this exercise which include warning thresholds for 500 and alerts for 800 or above.  Missing data trigger is also set as well.

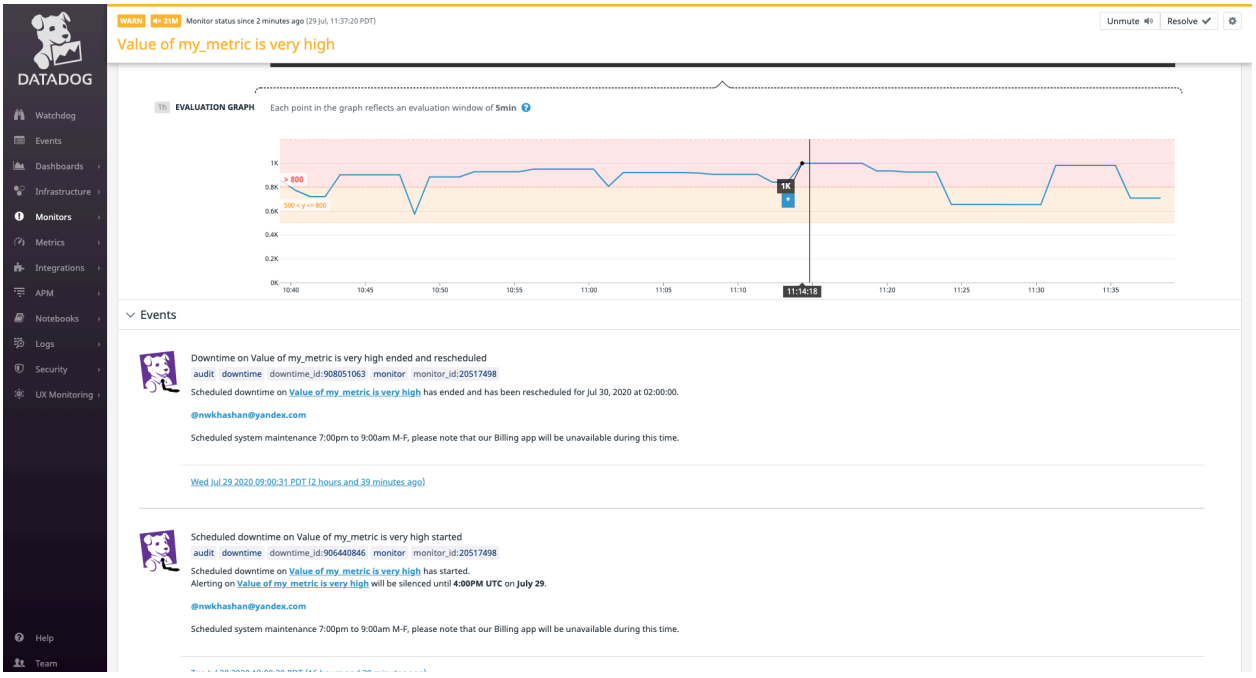



Please reference the link below for the monitors for the monitoring exercise.

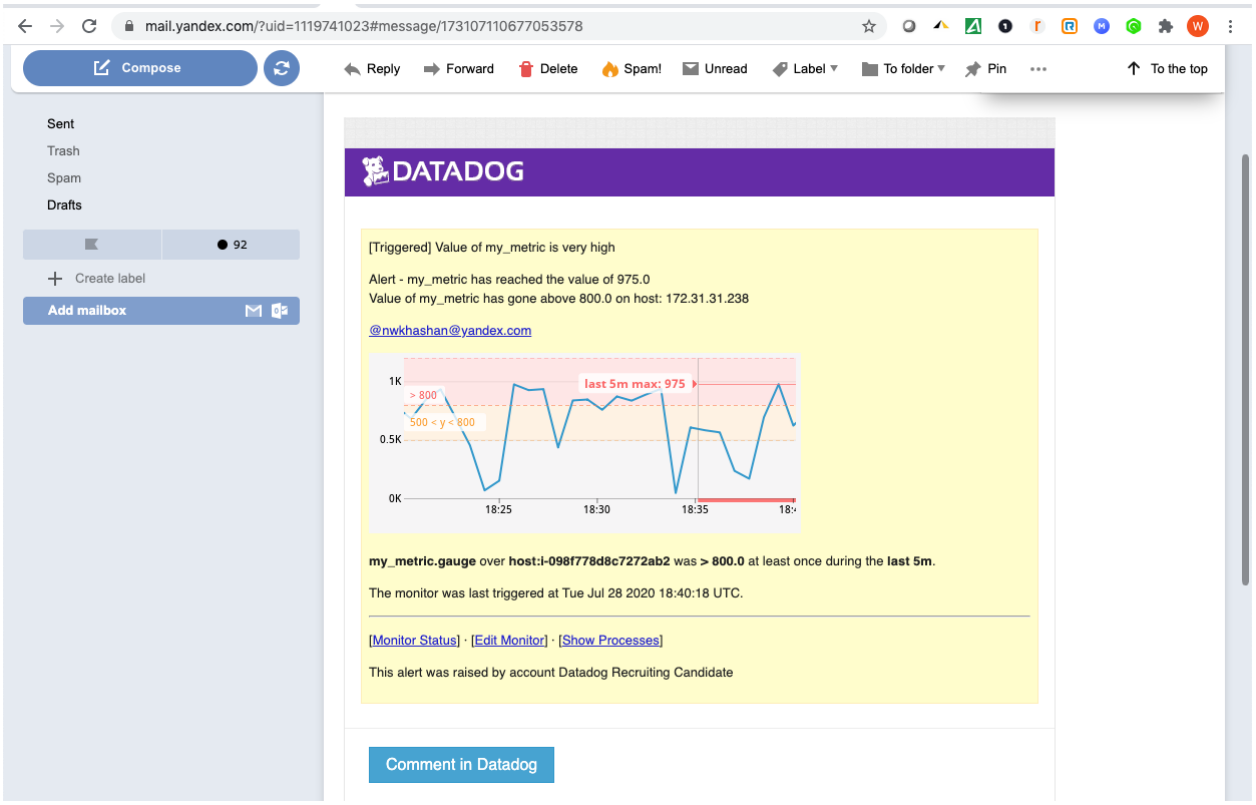
https://app.datadoghq.com/monitors/20517498

The screen shot below shows the monitor for the value of my_metric is very high alert. The alert is showing 1K in this graph and the one below.



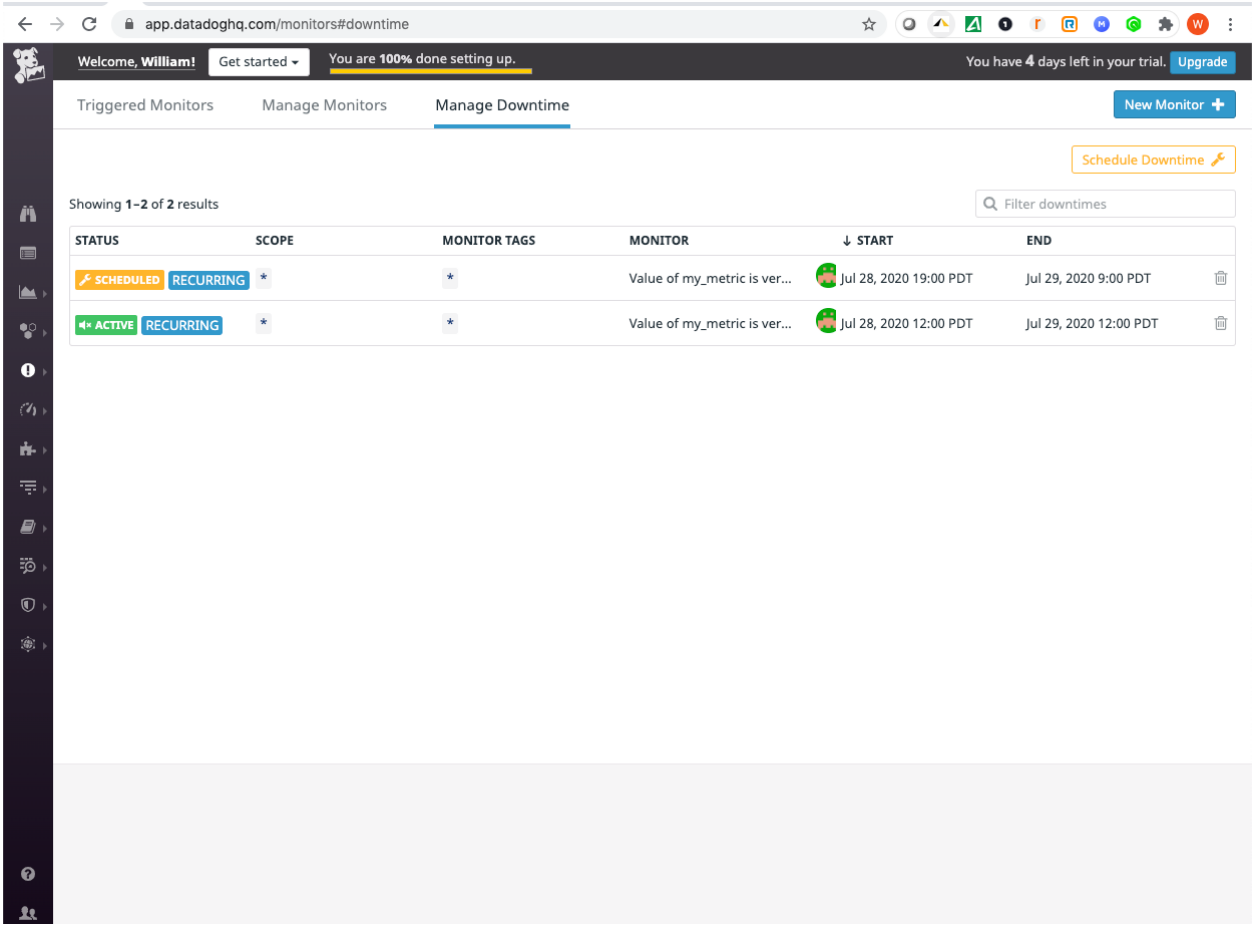



Email notification indicating a high alert of 975 which is above the 800 threshold set in the monitor.

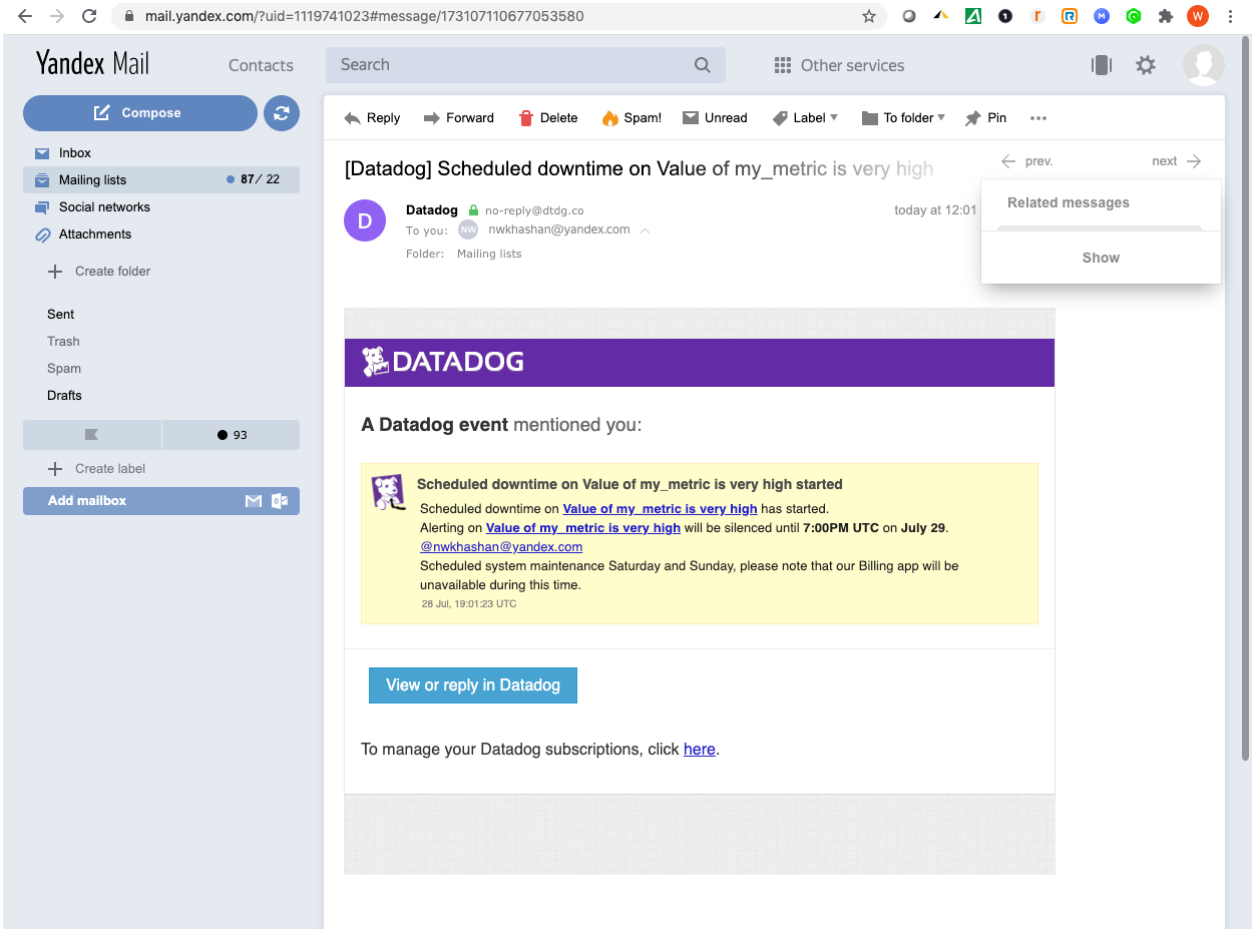



**Bonus Question:** Monitors set up to be turned off from 7pm to 9am daily and all day on Sat-Sun. Please see the screenshots below.

The monitors are set up for managed downtime. The first monitor is scheduled from 7pm to 9am on Monday to Friday. The second monitor is scheduled for downtime all day on Saturday and Sunday.

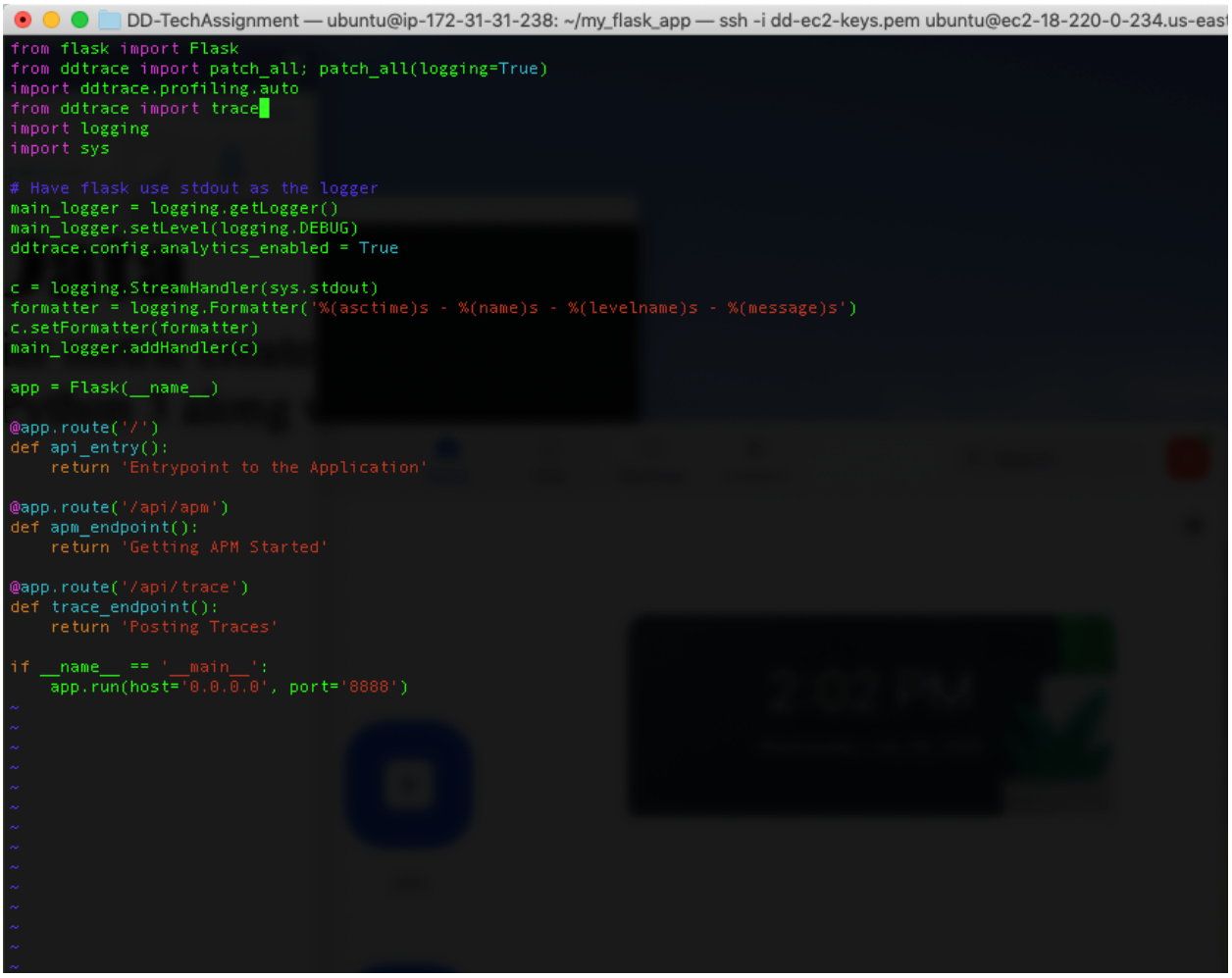


Email notification for scheduled down time shown below.

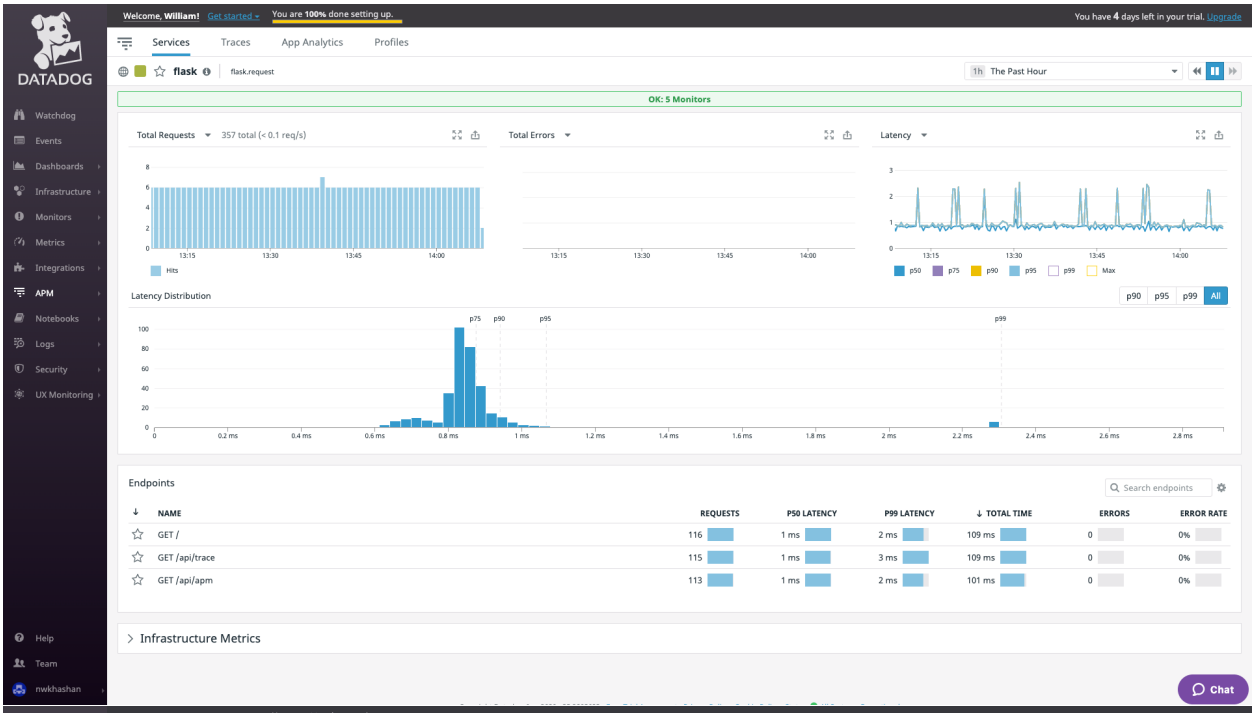


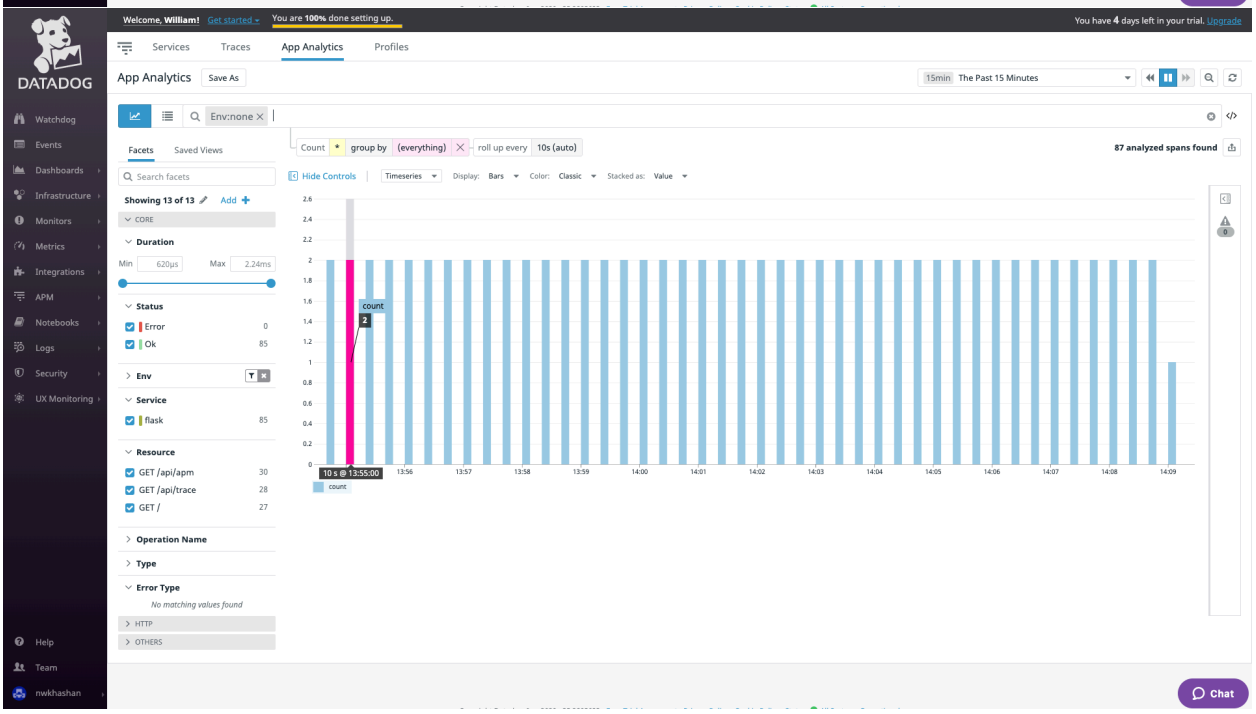
# Collecting APM Data

The APM piece of the project was done in Python since the earlier metric script created for the Timeboard was done in Python. I needed to install Python 3 along with Flask on Ubuntu to create the script for this part of the exercise.

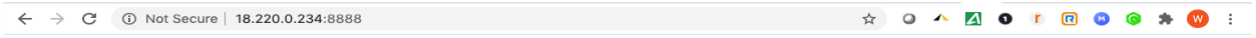



The results on the console are Flask script are shown below.

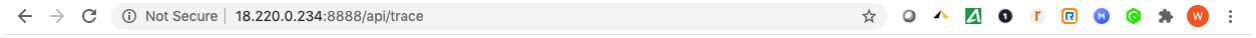


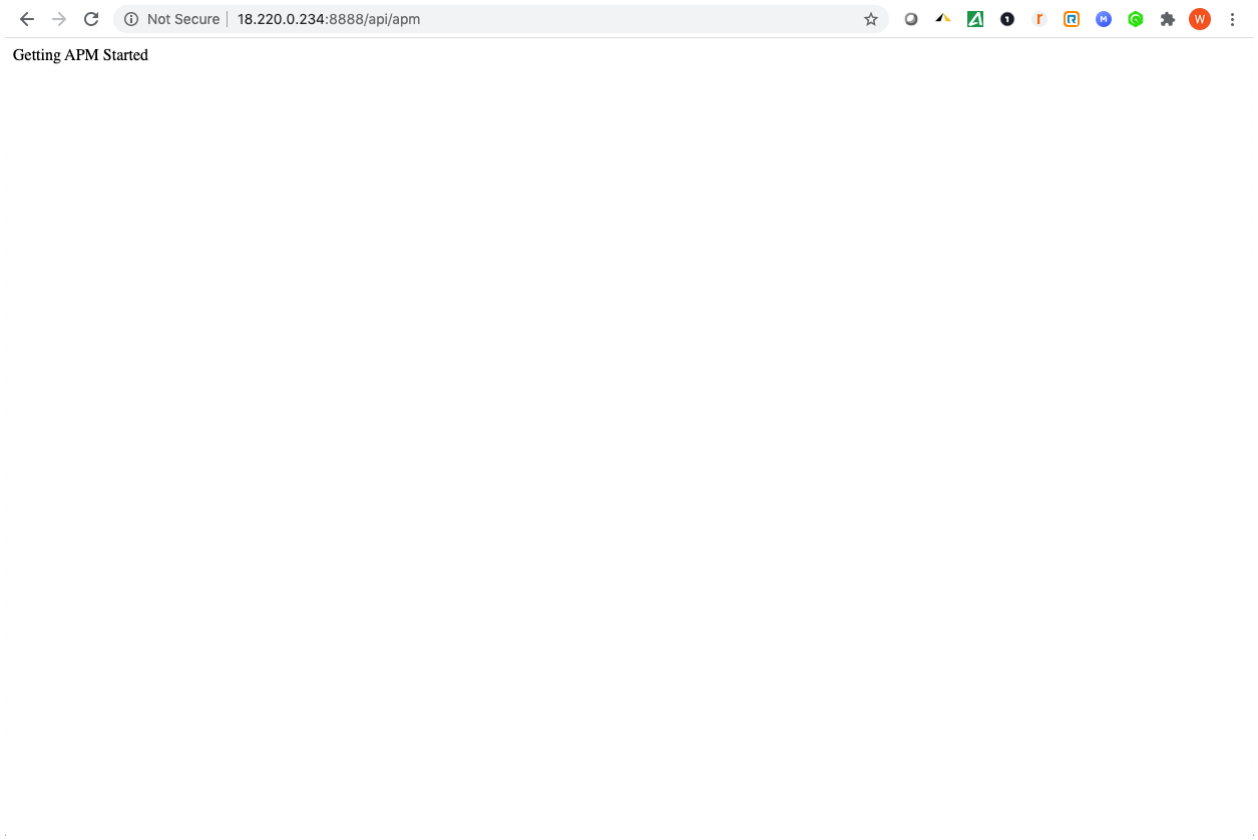

The following screen shots show the results of the Flask script in the browser.




Entrypoint to the Application

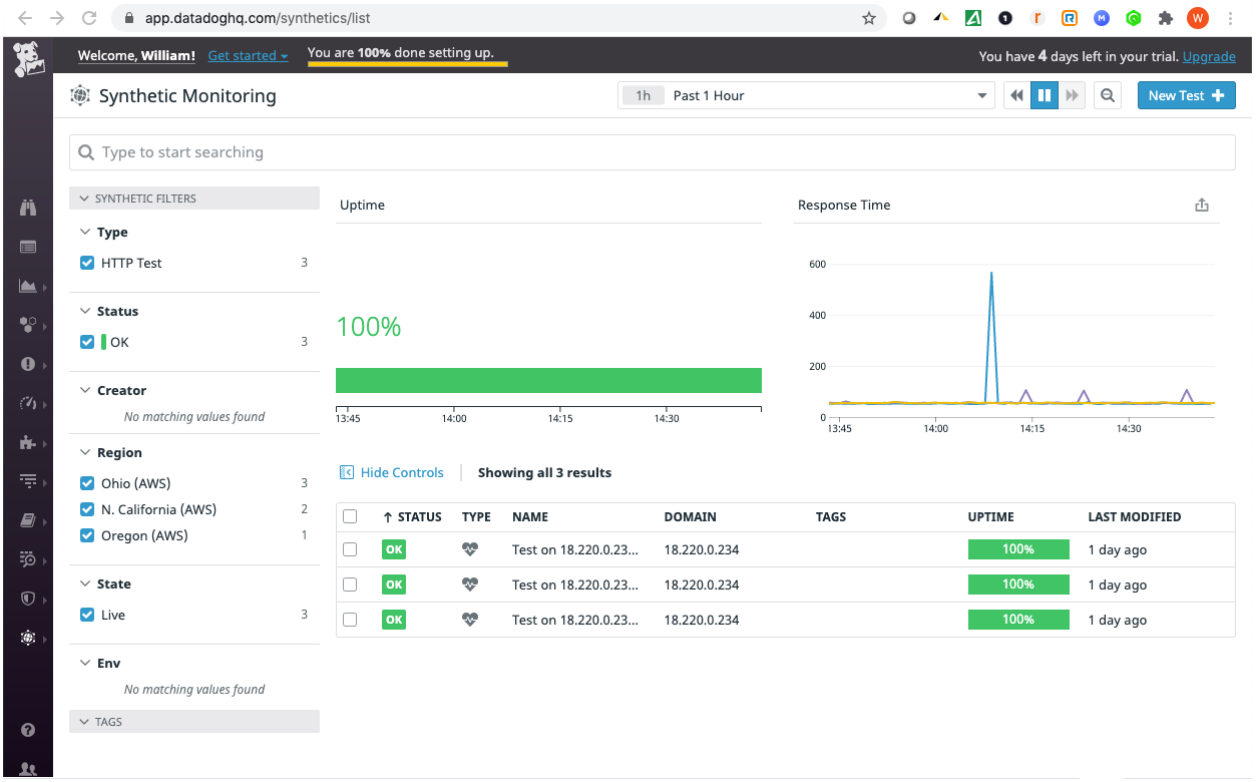


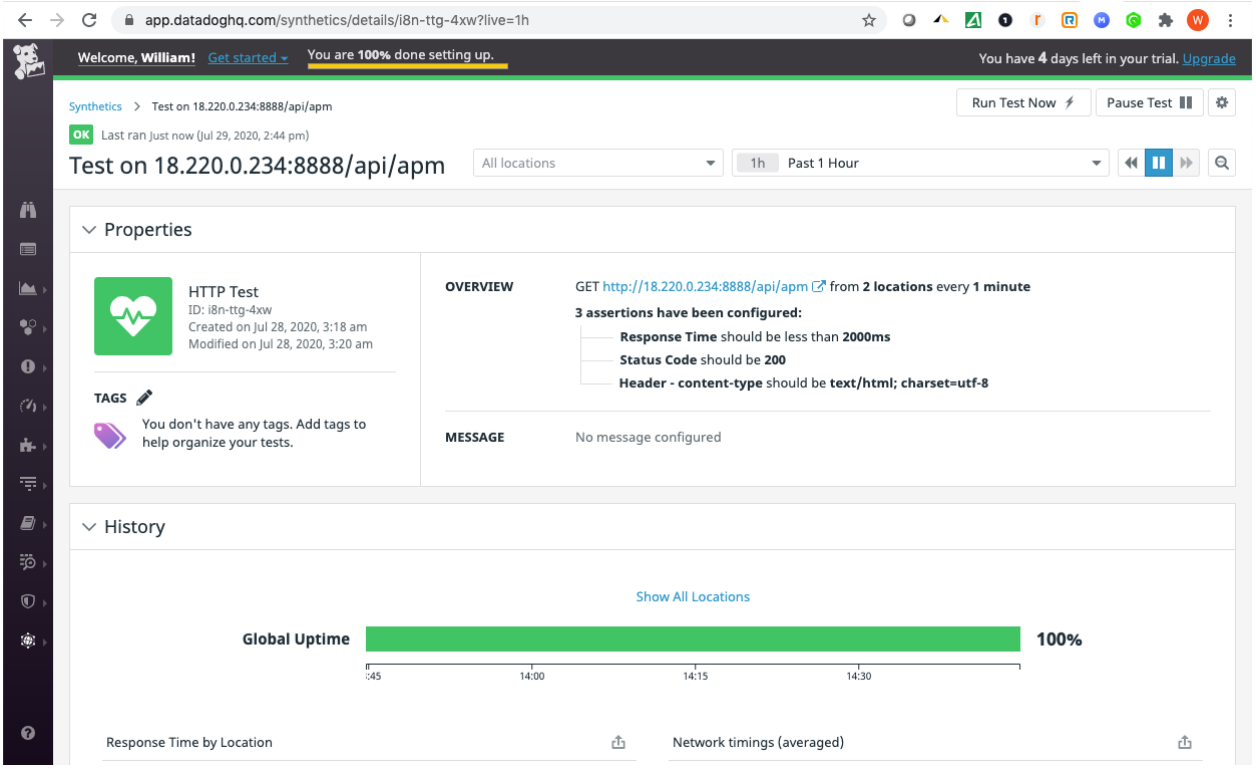

Posting Traces



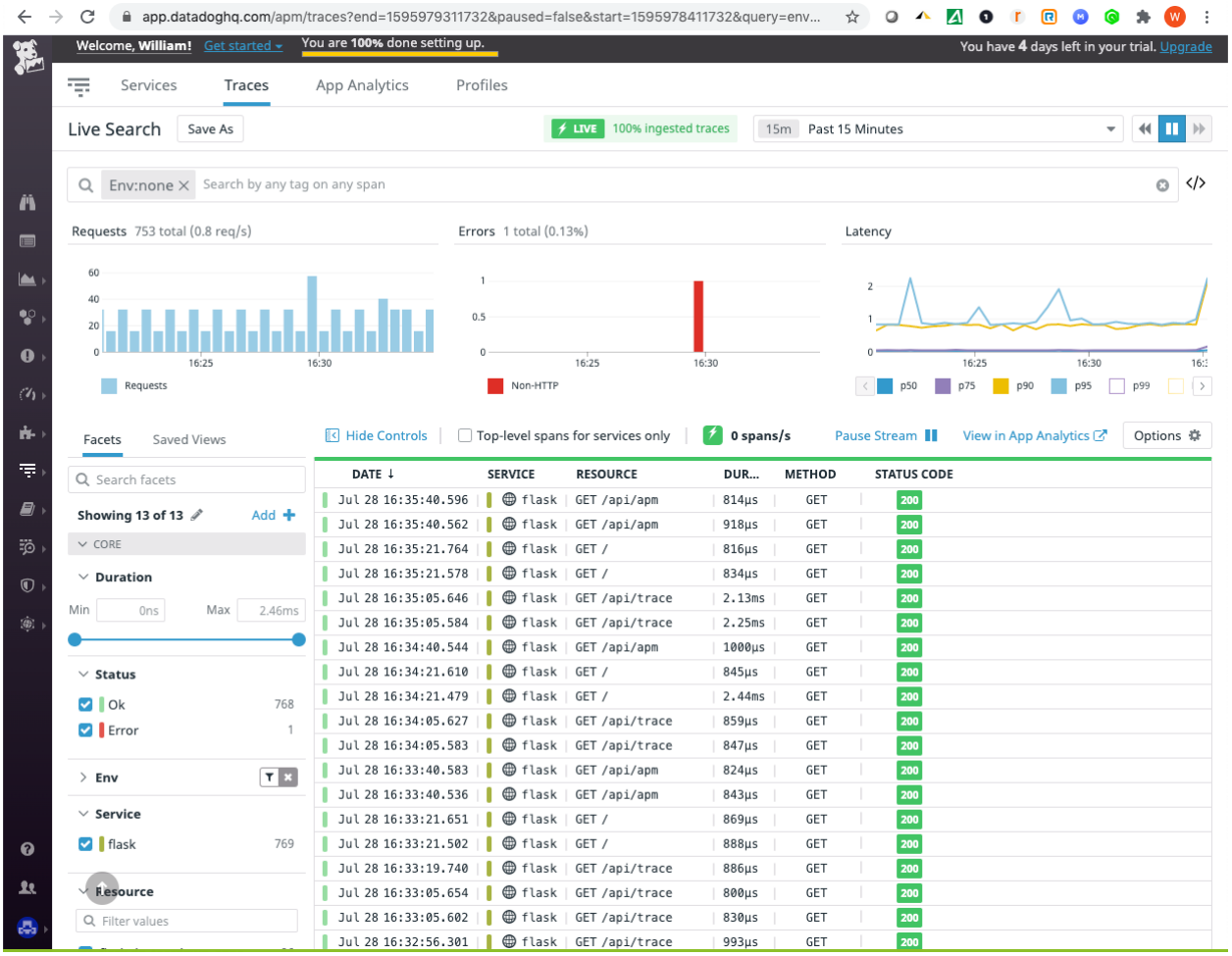

One thing to note about using the flask script. The port listed in the exercise was 5050. The port for whatever reason was not working at all. I finally figured this out through trial and error and reading through the error messages. The port used for the Flask script is 8888. The other thing of note was since activity was needed to see the results seen in the console I created 3 synthetic scripts for all 3 API's called from the Flask script in the browser.

I will be adding screen shots for the synthetic scripts below.


app.datadoghq.com/synthetics/list
Welcome, William! Get started
You are 100% done setting up.
You have 4 days left in your trial. Upgrade
Synthetic Monitoring
1h Past 1 Hour
New Test
Type to start searching
SYNTHETIC FILTERS
Type
HTTP Test 3
Status
OK 3
Creator
No matching values found
Region
Ohio (AWS) 3
N. California (AWS) 2
Oregon (AWS) 1
State
Live 3
Env
No matching values found
TAGS
Uptime
100%
13:45
14:00
14:15
14:30
Response Time
600
400
200
0
Hide Controls
Showing all 3 results
STATUS
TYPE
NAME
DOMAIN
TAGS
UPTIME
LAST MODIFIED
OK
Test on 18.220.0.23...
18.220.0.234
100%
1 day ago


app.datadoghq.com/synthetics/details/i8n-ttg-4xw?live=1h
Welcome, William! Get started
You are 100% done setting up.
You have 4 days left in your trial. Upgrade
Synthetics > Test on 18.220.0.234:8888/api/apm
Run Test Now
Pause Test
OK Last ran just now (Jul 29, 2020, 2:44 pm)
Test on 18.220.0.234:8888/api/apm
All locations
1h Past 1 Hour
Properties
HTTP Test
ID: i8n-ttg-4xw
Created on Jul 28, 2020, 3:18 am
Modified on Jul 28, 2020, 3:20 am
TAGS
You don't have any tags. Add tags to help organize your tests.
OVERVIEW
GET http://18.220.0.234:8888/api/apm from 2 locations every 1 minute
3 assertions have been configured:
Response Time should be less than 2000ms
Status Code should be 200
Header - content-type should be text/html; charset=utf-8
MESSAGE
No message configured
History
Show All Locations
Global Uptime
100%
14:00
14:15
14:30
Response Time by Location
Network timings (averaged)

The activity below is created by the synthetic scripts created for the API calls in the Flask script.




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

A service is a set of calls or processes similar to the API calls made in the Flask script that conduct the same task. A resource is a particular function or action for a service similar to the URL in the web Flask application.

**Final Question:** Datadog has quite an extensive amount of functionality within its scope. Datadog does logs, security, networking monitoring, infrastructure monitoring, APM, and Synthetic testing just to name a few things. The defining issue in our life right now is Covid-19. The Covid-19 issue is not only impacting people in the USA, but all over the world. The data used to track, trace, or even monitor what is going on with the virus is vital and is life or death across the world. Datadog can be used to understand if the applications used for Covid-19 are performing well, are secure, and are staying up at all times. The Covid-19 issue is impacting almost every country in the world and the data shared is vital to understanding trends and potentially saving lives. The exercise shows the flexibility of Datadog and this would allow developers to create scripts or API calls specific to the language and applications used for Covid-19. The alerts can be set up so the Operations folks, the scientist, and medical staff who monitor

cases, create vaccines, or even help determine who needs ventilators are provided the information they need to determine if these critical life or death applications are working at an optimal level.