



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.

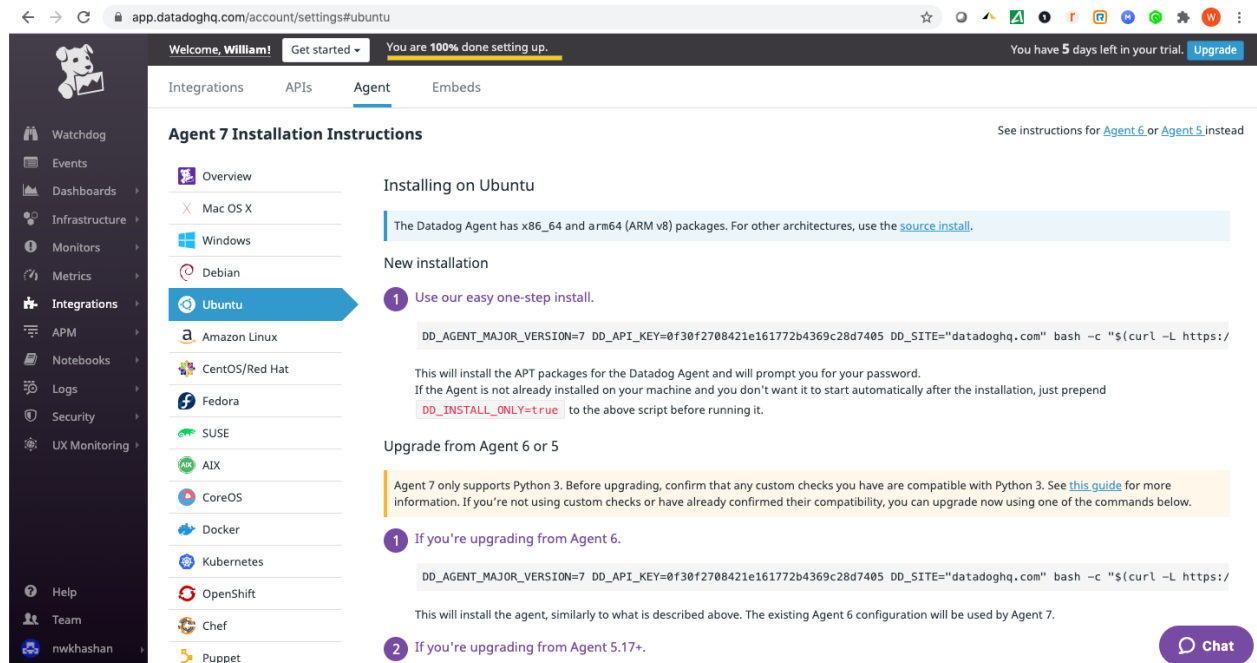
The screenshot displays the AWS Management Console interface. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and account information for 'Nabeels AWS Account' in the 'Ohio' region. The left sidebar shows the 'EC2 Dashboard' and various navigation links like 'Events', 'Tags', 'Limits', 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images', 'AMIs', 'Elastic Block Store', and 'Volumes'. The main content area shows a table with one instance: 'i-098f778d8c7272ab2', which is a 't2.micro' instance in the 'us-east-2b' availability zone, currently in a 'running' state. Below the table, the 'Description' tab is selected, showing details for the instance, including its ID, state, type, finding, private DNS, private IPs, secondary private IPs, VPC ID, public DNS (IPv4), IPv4 public IP, IPv6 IPs, Elastic IPs, availability zone, security groups, scheduled events, and AMI ID.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public
	i-098f778d8c7272ab2	t2.micro	us-east-2b	running	2/2 checks ...	None	ec2-18-220-0-234.us-e...	18.220.0.234

Instance: **i-098f778d8c7272ab2** Public DNS: ec2-18-220-0-234.us-east-2.compute.amazonaws.com

Description	Status Checks	Monitoring	Tags
Instance ID	i-098f778d8c7272ab2	Public DNS (IPv4)	ec2-18-220-0-234.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	18.220.0.234
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-172-31-31-238.us-east-2.compute.internal	Availability zone	us-east-2b
Private IPs	172.31.31.238	Security groups	launch-wizard-1 , view inbound rules , view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-23258348	AMI ID	ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-

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

```
ubuntu@ip-172-31-31-238: /etc/datadog-agent — ssh -i dd-ec2-keys.pem ubuntu@ec2-18-220-0-234.us-east-2....
#
# skip_ssl_validation: false

## @param force_tls_12 - boolean - optional - default: false
## Setting this option to "true" forces the Agent to only use TLS 1.2 when
## pushing data to the Datadog intake specified in "site" or "dd_url".
#
# force_tls_12: false

## @param hostname - string - optional - default: auto-detected
## Force the hostname name.
#
# hostname: <HOSTNAME_NAME>

## @param hostname_fqdn - boolean - optional - default: false
## When the Agent relies on the OS to determine the hostname, make it use the
## FQDN instead of the short hostname. Recommended value: true
## More information at https://dtdg.co/flag-hostname-fqdn
#
# hostname_fqdn: false

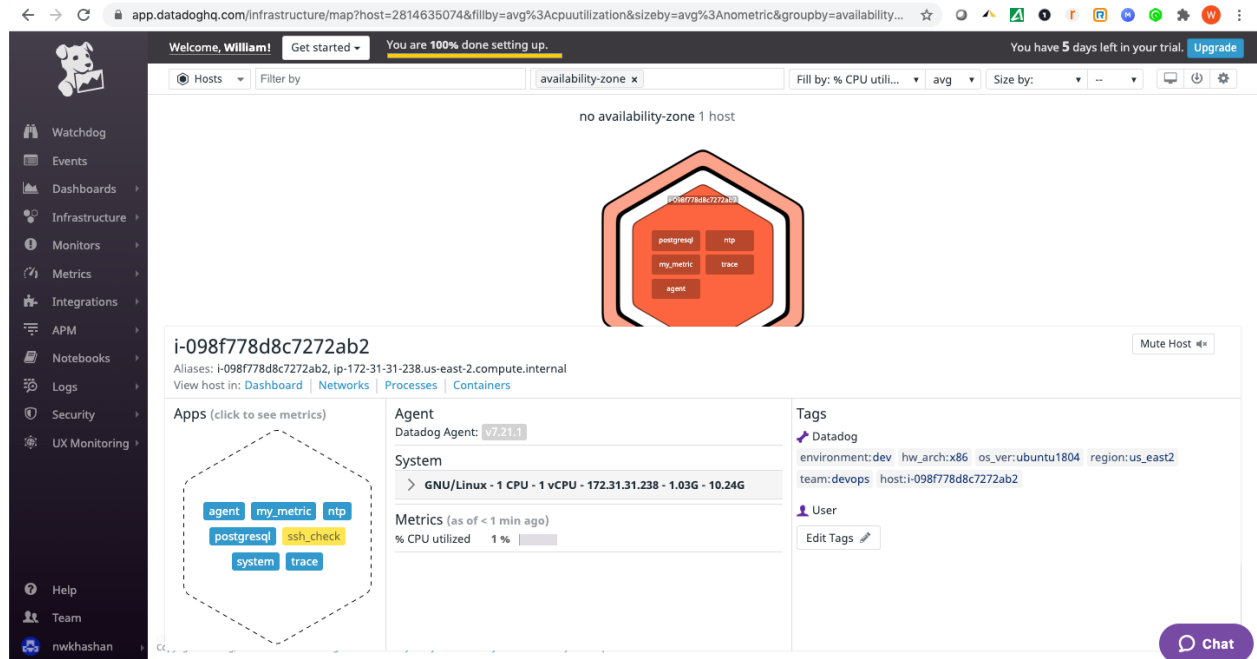
## @param tags - list of key:value elements - optional
## List of host tags. Attached in-app to every metric, event, log, trace, and service check emitted by this
## Agent.
## Learn more about tagging: https://docs.datadoghq.com/tagging/
#
tags:
- os_ver:ubuntu1804
- hw_arch:x86
- environment:dev
- region:us_east2
- team:devops

# - <TAG_KEY>:<TAG_VALUE>

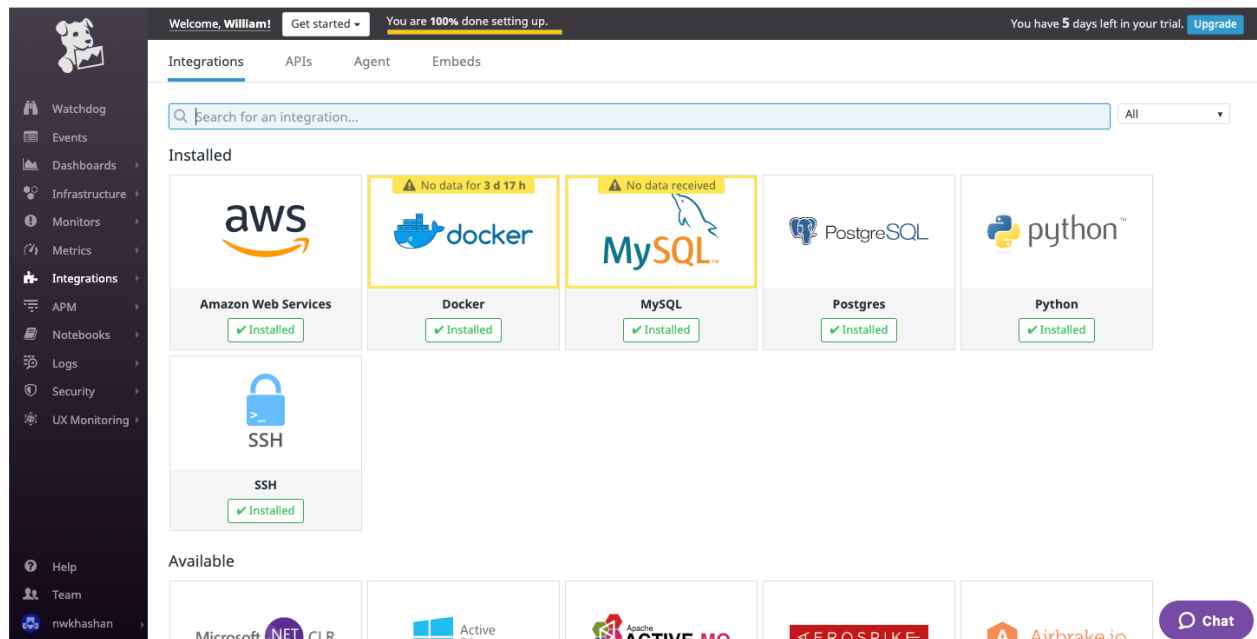
## @param env - string - optional
## The environment name where the agent is running. Attached in-app to every
## metric, event, log, trace, and service check emitted by this Agent.
#
# env: <environment name>

## @param tag_value_split_separator - list of key:value elements - optional
## Split tag values according to a given separator. Only applies to host tags,
## and tags coming from container integrations. It does not apply to tags on dogstatsd metrics,
78.1 2%
```

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.

Visualizing Data

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

```
from datadog import initialize, api

options = {
    'api_key': '0f30f2708421e161772b4369c28d7405',
    'app_key': '2dd747741348d513eda715b981219c2407a9d907'
}

initialize(**options)

title = 'Visualizing Data Exercise - Custom Metrics Timeboard'
widgets = [
    {
        'definition': {
            'type': 'timeseries',
            'requests': [
                ('q': 'avg:my_metric.gauge(host:i-098f778d8c7272ab2)')
            ],
            'title': 'Value of my_metric over time scoped over my host'
        }
    },
    {
        'definition': {
            'type': 'timeseries',
            'requests': [
                ('q': "anomalies(avg:postgresql.percent_usage_connections(*), 'basic', 2)")
            ],
            'title': 'Metric from Integration on DB with anomaly function'
        }
    },
    {
        'definition': {
            'type': 'query_value',
            'requests': [
                ('q': 'avg:my_metric.gauge(host:i-098f778d8c7272ab2).rollup(sum, 3600)')
            ],
            'title': 'My custom metric with rollup function applied to sum up all the points'
        }
    }
]

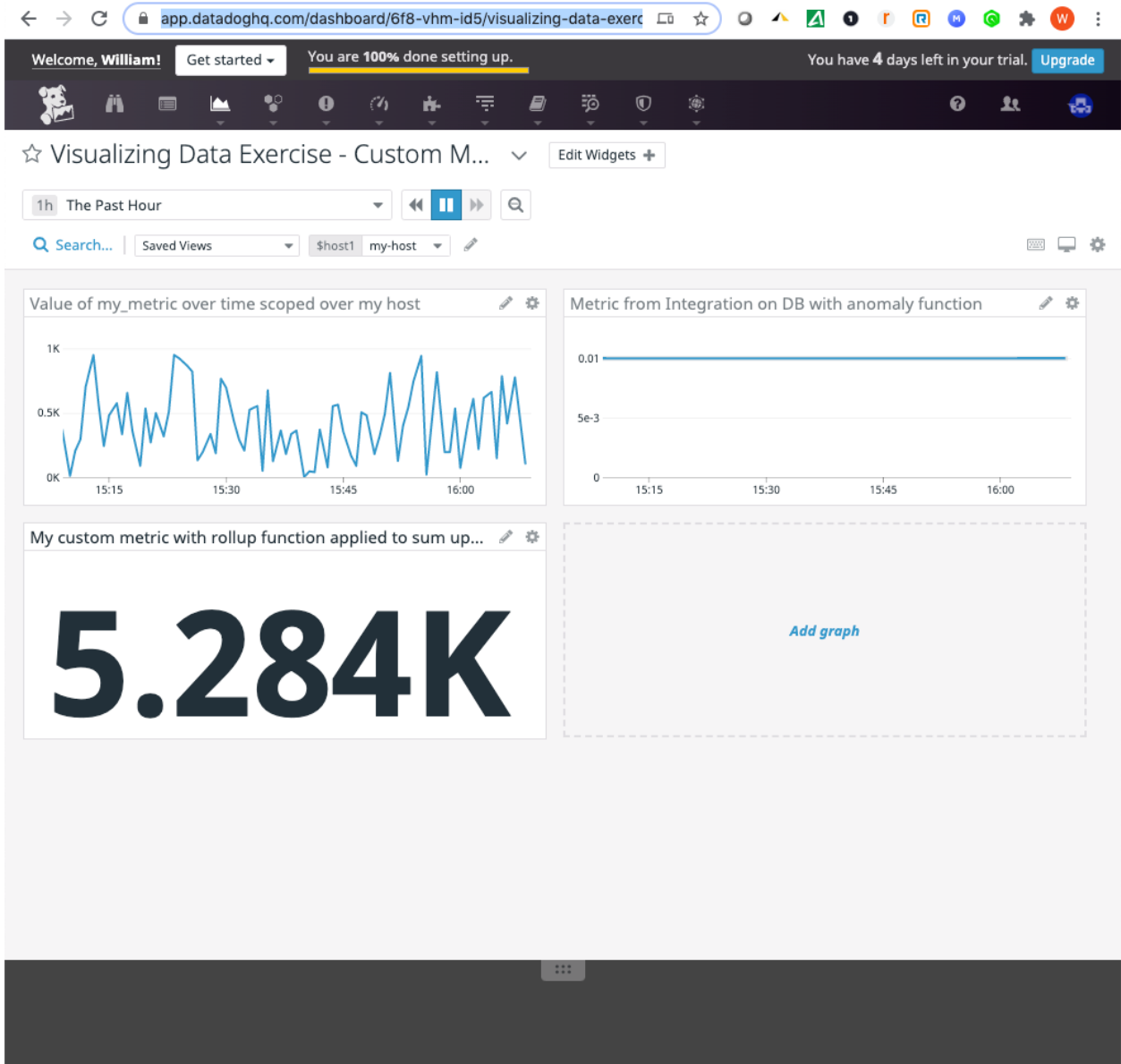
layout_type = 'ordered'
description = 'A dashboard displaying value of my_metric'
is_read_only = True
notify_list = ['nvkhashan@yandex.com']
template_variables = [
    {
        'name': 'host1',
        'prefix': 'host',
        'default': 'my-host'
    }
]

saved_view = [
    {
        'name': 'Saved views for hostname 2',
        'template_variables': [
            {
                'name': 'host',
                'value': 'i-098f778d8c7272ab2'
            }
        ]
    }
]

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

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



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

mail.yandex.com/?uid=1119741023#message/173107110677053586

Compose Reply Forward Delete Spam! Unread Label To folder Pin To the top

Mailing lists 90/99+ Social networks Attachments Create folder Sent Trash Spam Drafts 91 Create label Add mailbox

[Datadog] Value of my_metric over time scoped over my host

William Khashan no-reply@datdg.co today at 1:11
To you: nwkhashan@yandex.com
Folder: Mailing lists

Related messages Show

DATADOG

William Khashan (@nwkhashan@gmail.com) mentioned you in a comment:

William Khashan
Value of my_metric over time scoped over my host

Time (UTC-0700)	Value (K)
01:00	0.5
01:01	0.8
01:02	0.6
01:03	0.4
01:04	0.7
01:05	0.9
01:06	0.6
01:07	0.8
01:08	0.4
01:09	0.5
01:10	0.8

Nabeel there is an anomaly here of 995. Please take a look you may need to restart the server, @nwkhashan@yandex.com
29 Jul, 08:11:37 UTC

View or reply in Datadog

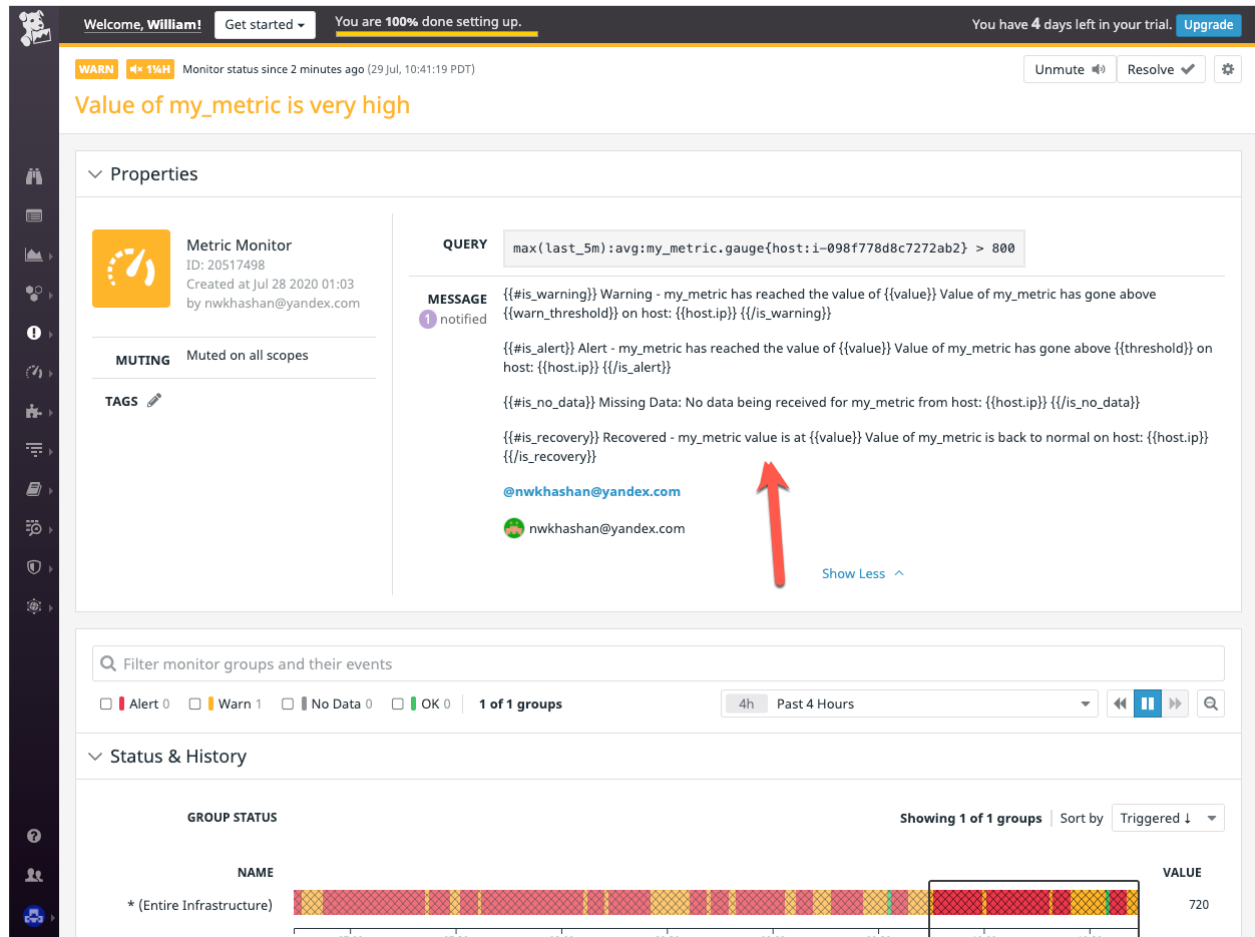
To manage your Datadog subscriptions, click [here](#).

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

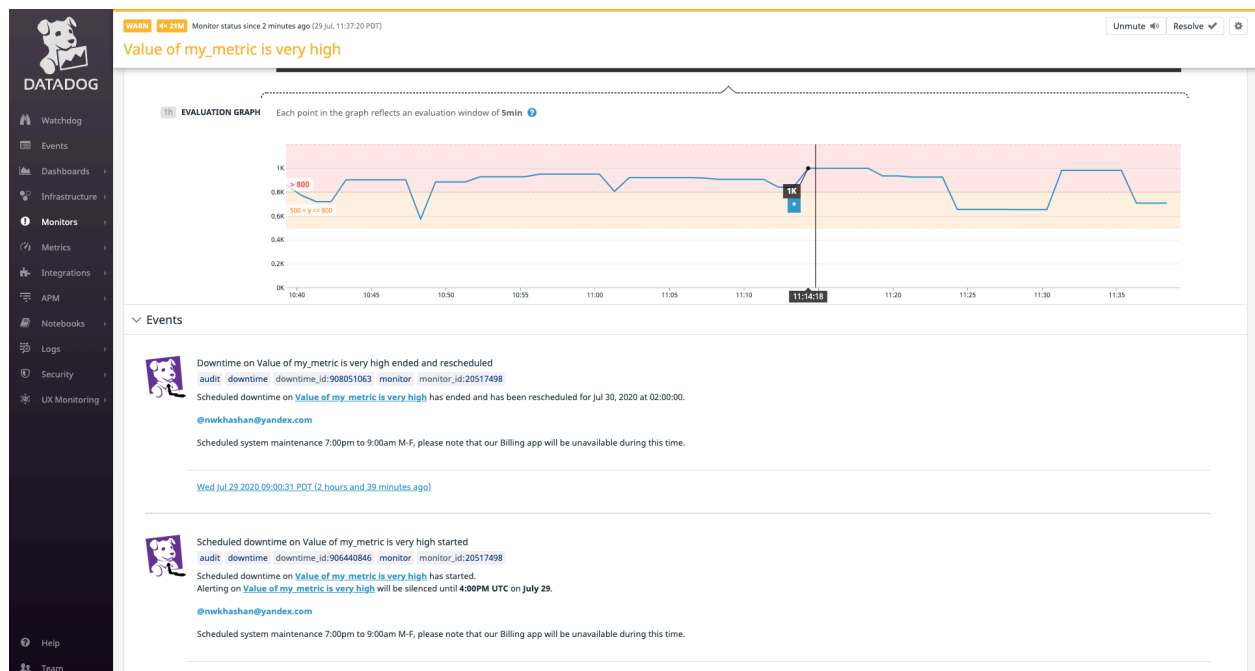
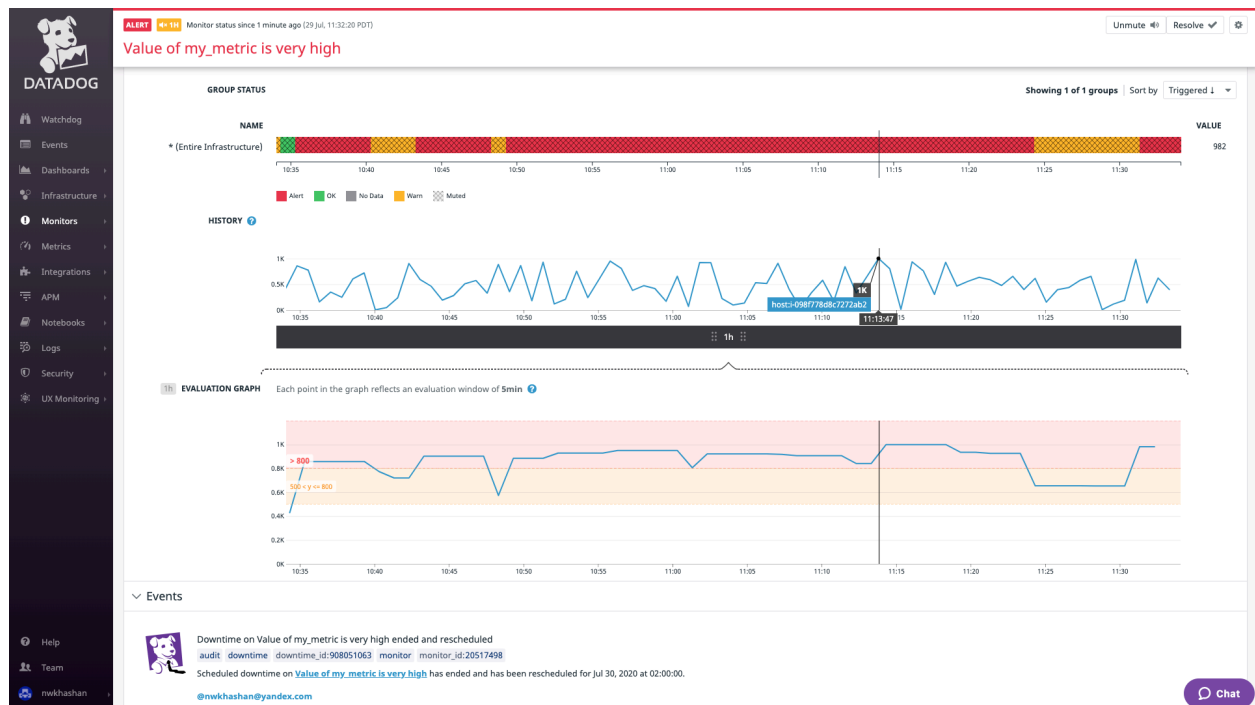
The data below highlights the monitors for this exercise which include warning thresholds for 500 and alerts for 800 or above. Missing data trigger is also set as well.



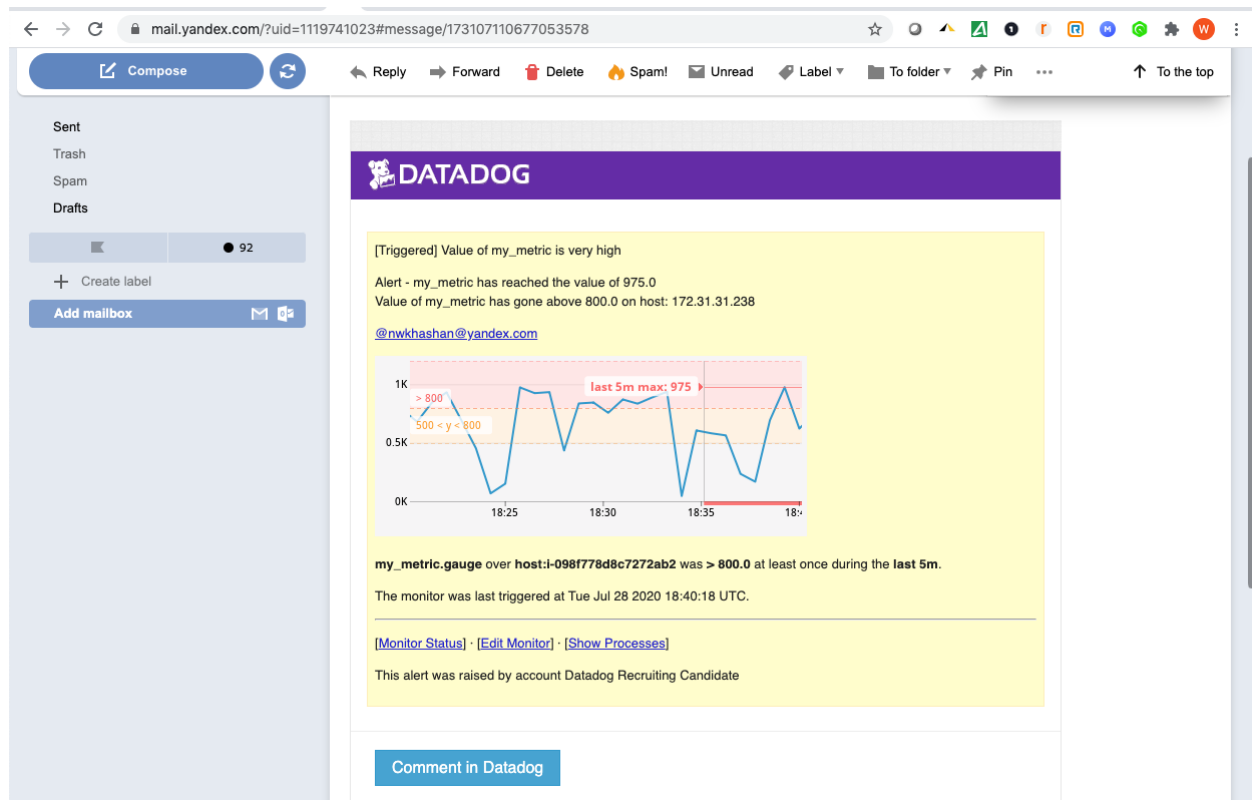
Please reference the link below for the monitors for the monitoring exercise.

<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.

The screenshot shows the Datadog Monitors Downtime management interface. The browser address bar displays `app.datadoghq.com/monitors#downtime`. The top navigation bar includes a welcome message for 'William!', a 'Get started' dropdown, a status message 'You are 100% done setting up.', and a trial notice 'You have 4 days left in your trial.' with an 'Upgrade' button. The main navigation tabs are 'Triggered Monitors', 'Manage Monitors', and 'Manage Downtime', with 'Manage Downtime' being the active tab. A 'New Monitor +' button is located in the top right corner. Below the tabs, there is a 'Schedule Downtime' button with a calendar icon. The main content area shows 'Showing 1-2 of 2 results' and a search bar labeled 'Filter downtimes'. A table lists two scheduled downtime events:

STATUS	SCOPE	MONITOR TAGS	MONITOR	↓ START	END	
SCHEDULED RECURRING *	*	*	Value of my_metric is ver...	Jul 28, 2020 19:00 PDT	Jul 29, 2020 9:00 PDT	
ACTIVE RECURRING *	*	*	Value of my_metric is ver...	Jul 28, 2020 12:00 PDT	Jul 29, 2020 12:00 PDT	

Email notification for scheduled down time shown below.

The screenshot displays the Yandex Mail web interface. The left sidebar contains navigation links for 'Compose', 'Inbox', 'Mailing lists' (87/22), 'Social networks', 'Attachments', 'Create folder', 'Sent', 'Trash', 'Spam', 'Drafts', '93', 'Create label', and 'Add mailbox'. The main content area shows an email from 'Datadog' (no-reply@datadg.co) to 'nwkhassan@yandex.com' with the subject '[Datadog] Scheduled downtime on Value of my_metric is very high'. The email body features the Datadog logo, a section titled 'A Datadog event mentioned you:', and a yellow alert box stating: 'Scheduled downtime on Value of my_metric is very high started. Alerting on Value of my_metric is very high will be silenced until 7:00PM UTC on July 29. @nwkhassan@yandex.com. Scheduled system maintenance Saturday and Sunday, please note that our Billing app will be unavailable during this time. 28 Jul, 19:01:23 UTC'. Below the alert box is a blue button 'View or reply in Datadog' and a link 'To manage your Datadog subscriptions, click here.' A 'Related messages' dropdown menu is open, showing a 'Show' button.

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.

```
DD-TechAssignment — ubuntu@ip-172-31-31-238: ~/my_flask_app — ssh -i dd-ec2-keys.pem ubuntu@ec2-18-220-0-234.us-east-1.amazonaws.com
from flask import Flask
from ddtrace import patch_all; patch_all(logging=True)
import ddtrace.profiling.auto
from ddtrace import trace
import logging
import sys

# Have flask use stdout as the logger
main_logger = logging.getLogger()
main_logger.setLevel(logging.DEBUG)
ddtrace.config.analytics_enabled = True

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__)

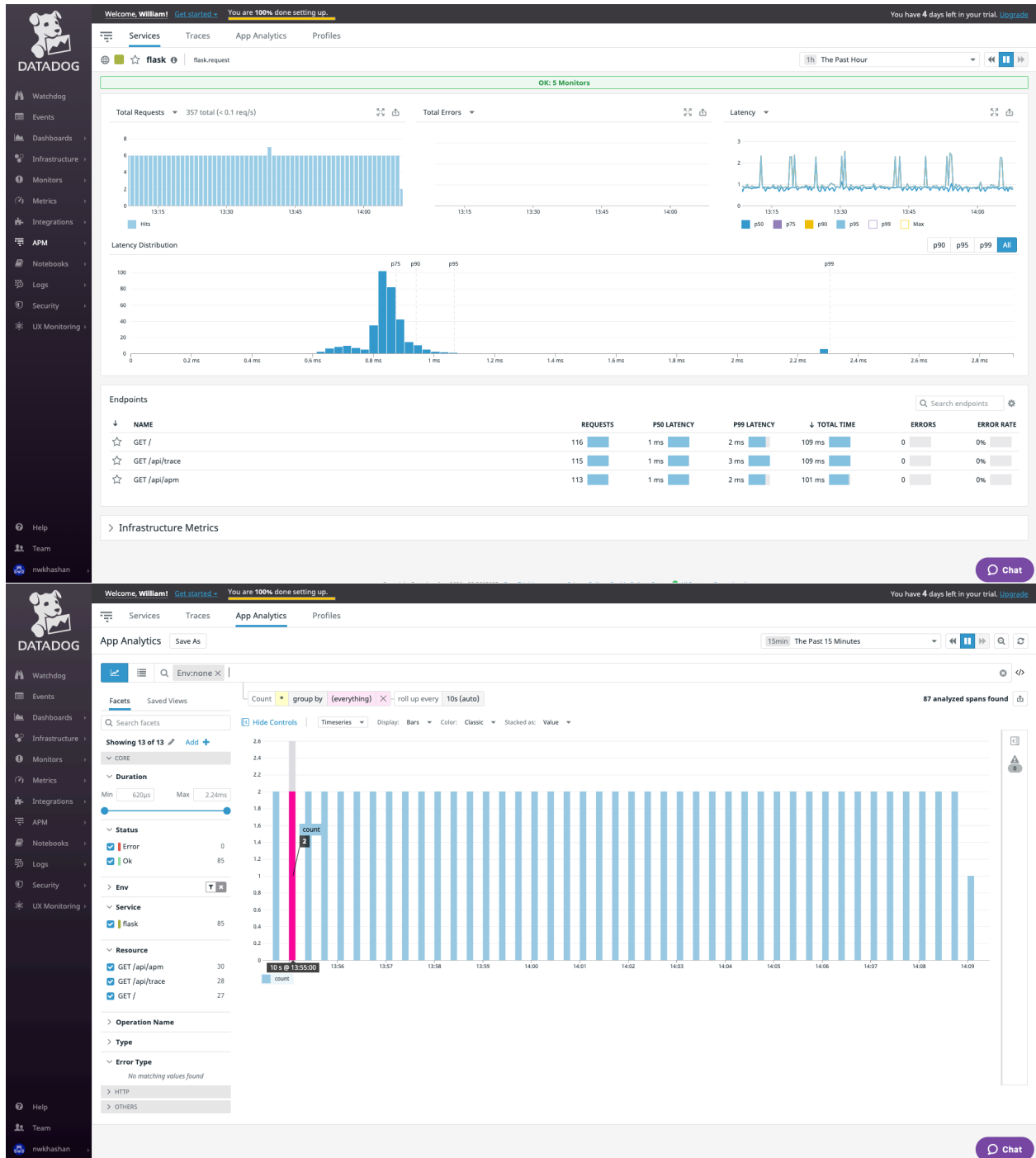
@app.route('/')
def api_entry():
    return 'Entrypoint to the Application'

@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='8888')
~
~
~
~
~
~
~
~
~
~
~
~
```

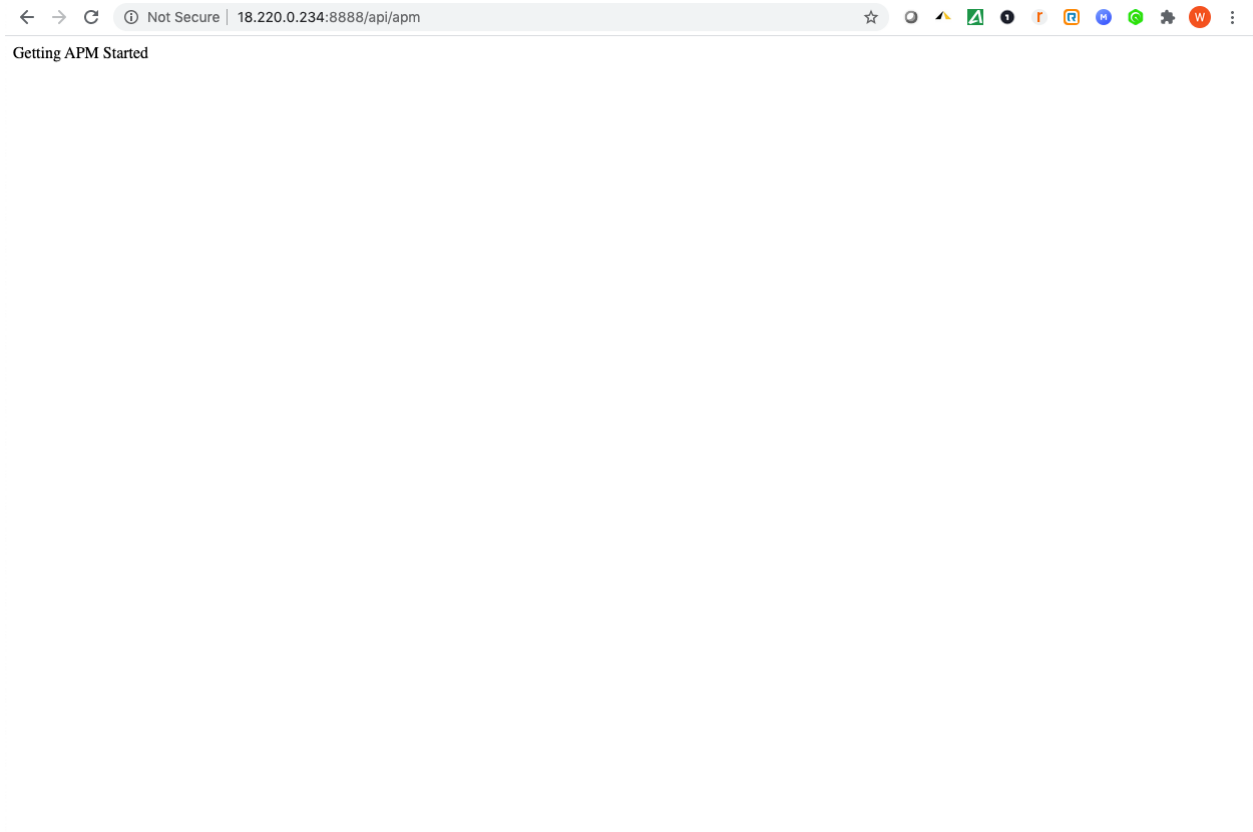
The results on the console are Flask script are shown below.



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

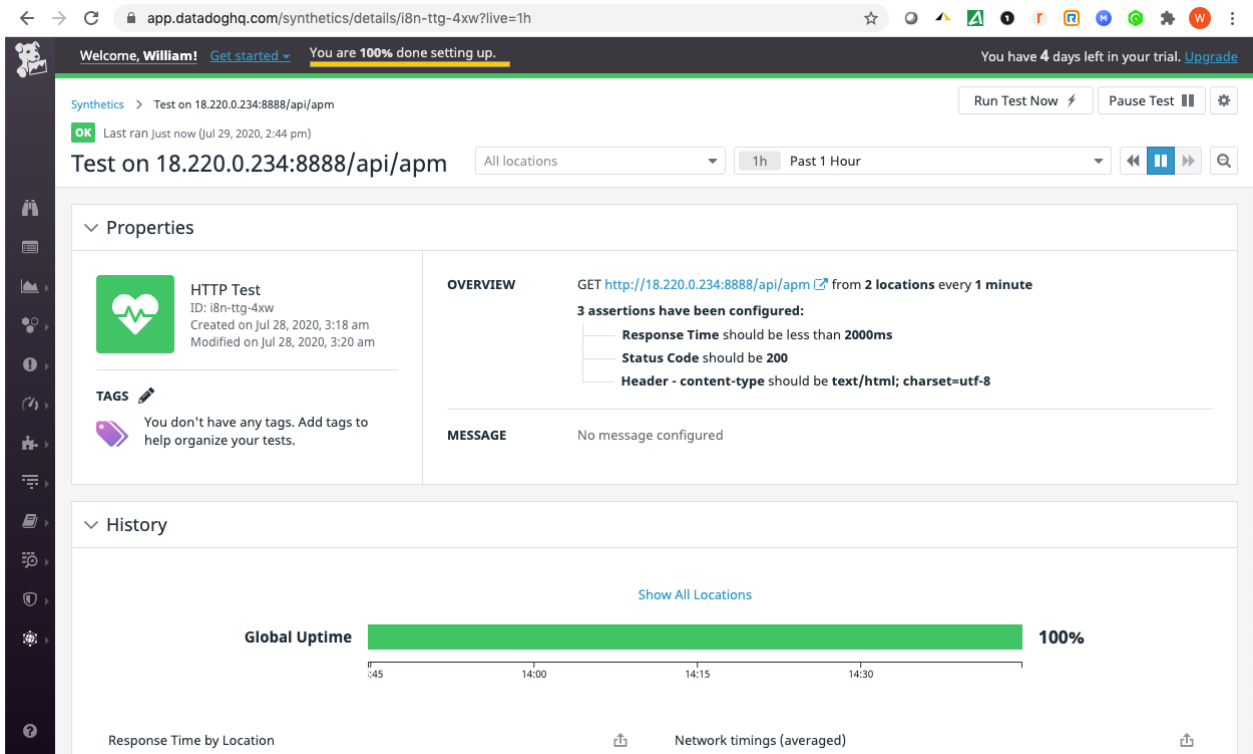
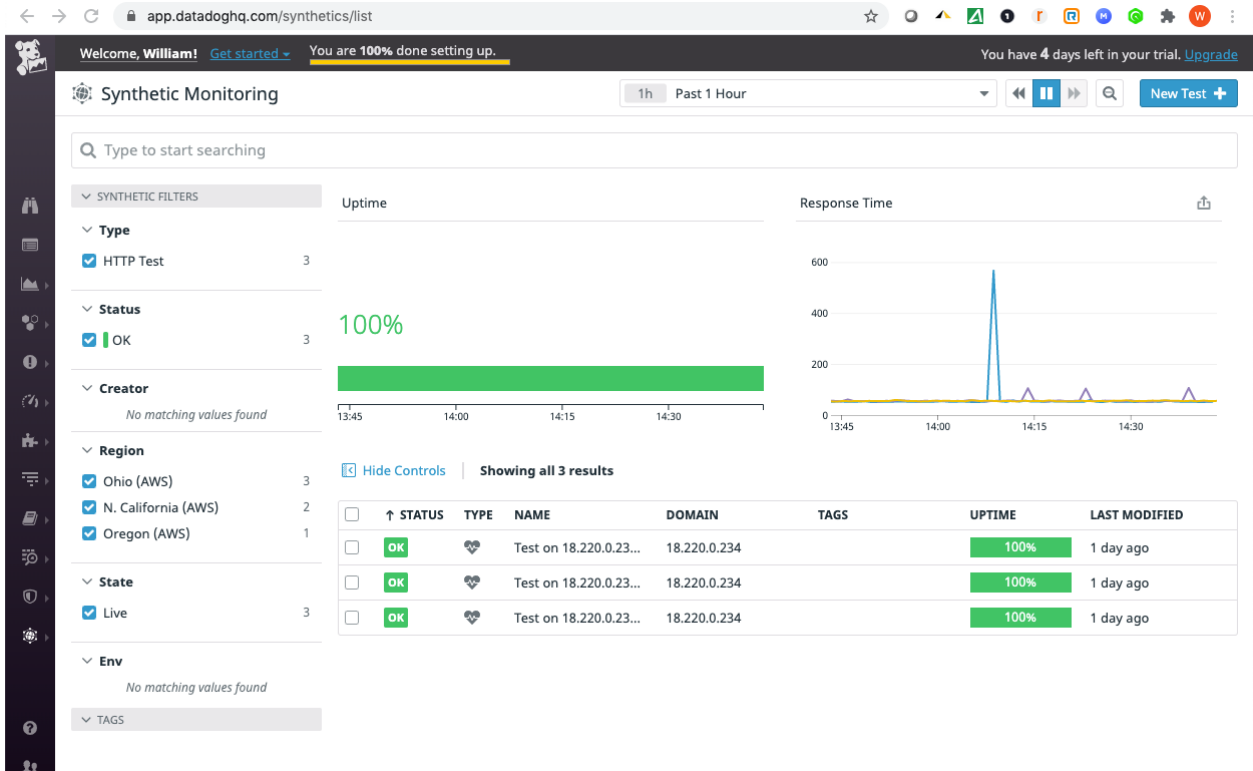
Entrypoint to the Application

Posting Traces

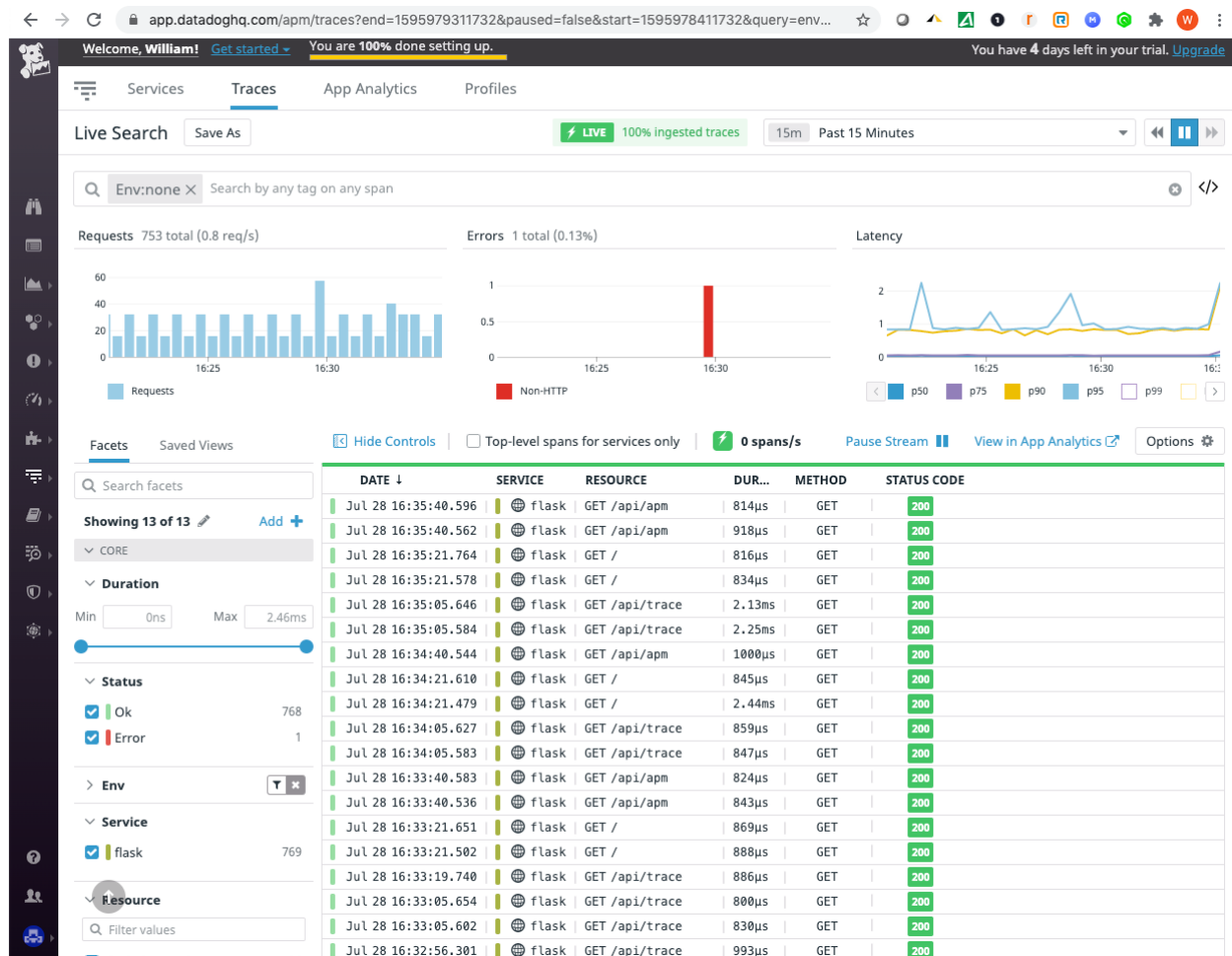


One thing to note about using the flask script. The port listed in the exercise was 5050. The port for whatever reason was not working at all. I finally figured this out through trial and error and reading through the error messages. The port used for the Flask script is 8888. The other thing of note 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.



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.