Dónal – DataDog hiring engineer excercise:

Table of Contents

Overview: This document is a response to the following datadog engineer exercise: https://github.com/DataDog/hiring-engineers/blob/solutions-engineer/README.md#visualizing-data

In this document I will go through each of the questions asked and demonstrate how I attempted to answer each section. Overall, I will demonstrate how to set up a database and how Datadog can gather information/metrics. Once these metrics are gathered, we can articulate this information to gain value from it.

Question 1:
Prerequisites - Setup the environment

A. *You can utilize any OS/host that you would like to complete this exercise. However, we recommend one of the following approaches:*

For this I used my laptop as the host.

I decided to download DataDog Agent and connect it to a standalone MongoDB instance which will also be on my host machine.

As my machine is a windows machine, I followed the below document to check the requirements and install the agent manager.

https://docs.datadoghq.com/agent/guide/datadog-agent-manager-windows/

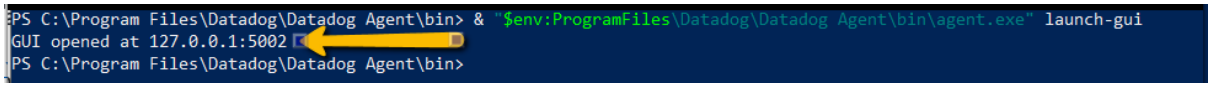
Once I ran through the typical steps of an installation, I was able to 'change directory' to the new directory that I will work from –

```
PS C:\Program Files\Datadog\Datadog Agent\bin>
```

Once directory is changed, I can run the following command to open the Datadog Agent Manager

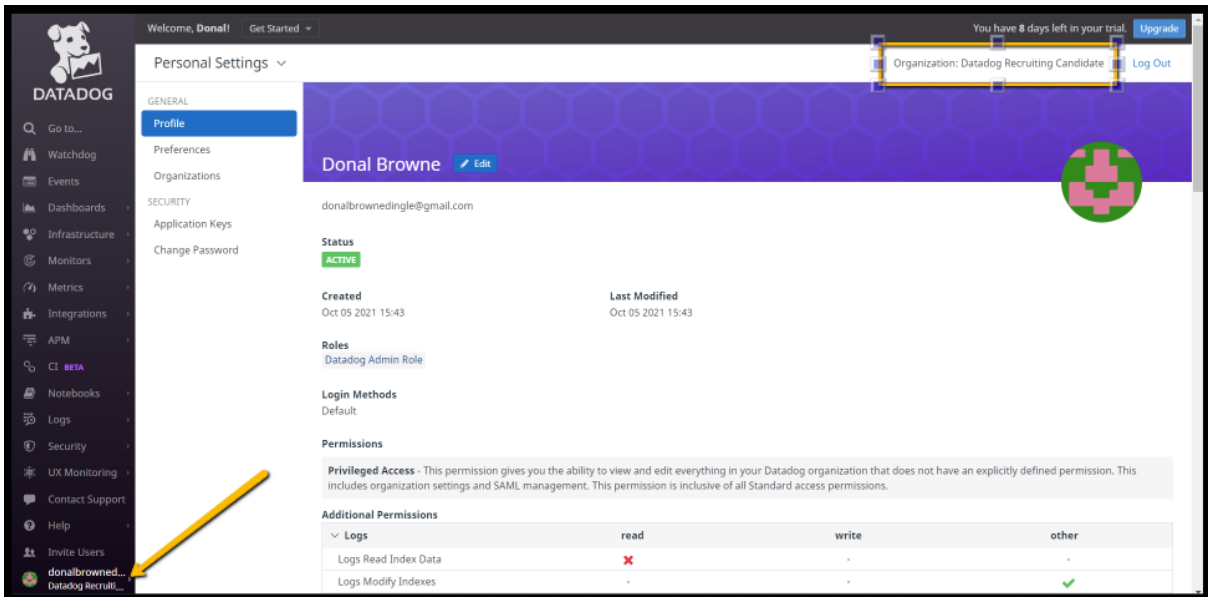
```
& "$env:ProgramFiles\Datadog\Datadog Agent\bin\agent.exe" launch-gui
```

We can see after we run the command that the GUI has opened from a local host (It specifies this is the only option in the requirements)



B. *Once this is ready, sign up for a trial Datadog at https://www.datadoghq.com/*
Please make sure to use "Datadog Recruiting Candidate" in the "Company" field

The Agent is now installed and we need to sign up for DataDog. As specified in the Hiring Engineer document you need to sign up as a 'DataDog Recruiting Candidate'
You can check this by clicking the bottom left-hand corner and checking the details. See below.



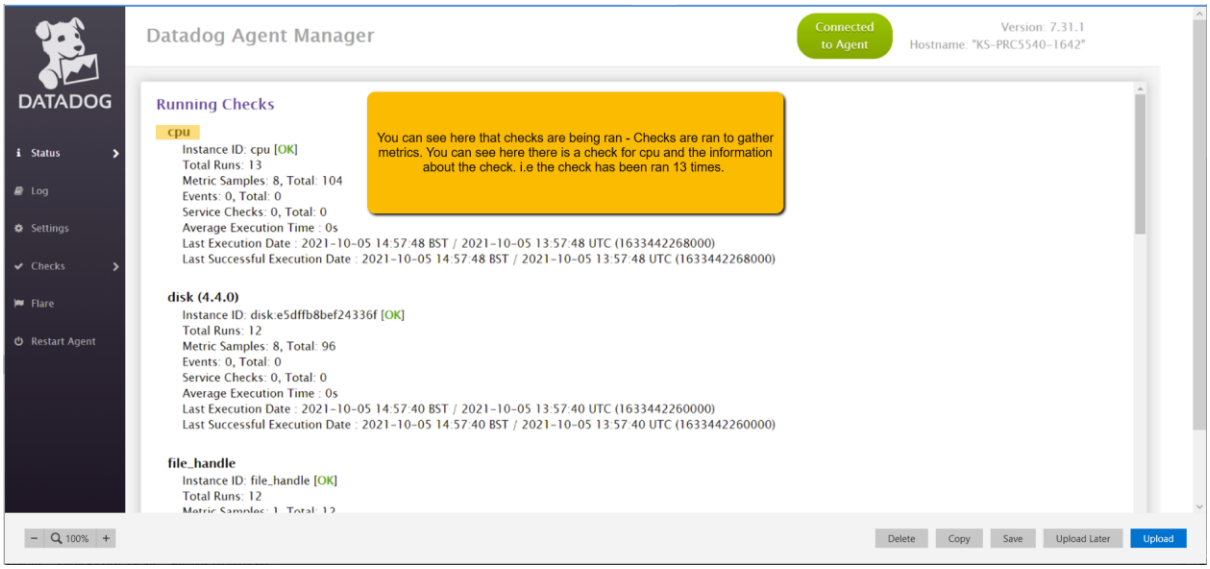
C. *Then, get the Agent reporting metrics from your local machine and move on to the next section...*

I could see that the host was showing

Hostnames
Hostname: KS-PRC5540-1642
Socket-Fqdn: KS-PRC5540-1642.kneat.org.
Socket-Hostname: KS-PRC5540-1642
Hostname Provider: os
Unused Hostname Providers:
Aws: not retrieving hostname from AWS: the host is not an ECS instance and other providers already retrieve non-default hostnames
Azure: azure_hostname_style is set to 'os'
Configuration/Environment: hostname is empty
Gce: unable to retrieve hostname from GCE: Get "http://169.254.169.254/computeMetadata/v1/instance/hostname": dial tcp 169.254.169.254:80: connectex: A socket operation was attempted to an unreachable network.

I then clicked around the Agent to familiarise myself with what exactly the agent did.

I found that when you go to the DataDog Agent Manager > Status > Collector you can see that metrics are being gathered.



Question 2:
Collecting Metrics:

a. *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.*

This document and its associated documents at the bottom of the page help us add tags.
https://docs.datadoghq.com/getting_started/tagging/

Additional youtube video on tagging - https://www.youtube.com/watch?v=7mCxL1goRDI

I need to find out what tags do exactly – Tags, essentially allow for better search functionality. If you add the tag 'role:db' to your database server's you can search for them on your hostmap and only show database server's.

Possible tags include.

- "Tag:windows" – because I am using a windows machine.
    - In terms of adding value, you may have different machines but might only want search windows server 2016 so you could tag them – "Tag:windows2016"
- "db:Mongo" – because I will be using a MongoDB.
    - Again you might have different databases but only want to search or group by MongoDB's. You may have some servers with Mongo and some with SQL Server's

I now need to find out where to apply them – Tags are added in the `datadog.yaml` file which in our case is stored at the following location – *'C:\ProgramData\Datadog\datadog.yaml'*
There are requirement in regards the adding the tags to the yaml file.
(https://docs.datadoghq.com/getting_started/tagging/assigning_tags/?tab=noncontainerizedenvironments)

The aim here was to update the yaml file to include the tags but I didn't no where to add the tags. *Before I go changing the file I saved a copy of the file with a different name and saved it to my desktop.

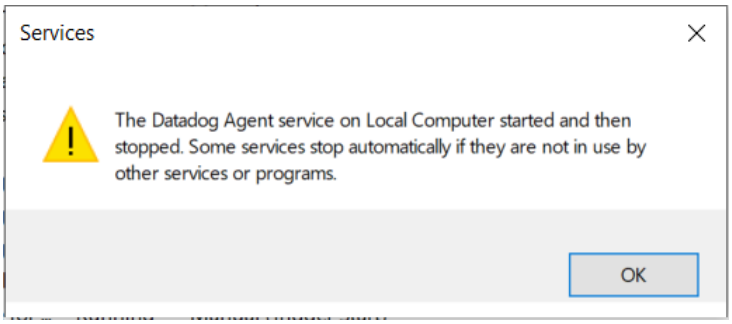
I then added the tags:

```
## Additional tags can be supplied using the  DD
##
## Learn more about tagging: https://docs.datad
#
# tags:
#   - environment:dev
#   - <TAG_KEY>:<TAG_VALUE>
#   - instance:local
#   - test:test_tag

## @param env - string - optional
## @env DD_ENV - string - optional
```

This made no change to the hostfile because it was still commented out – I found this out by googling if a '#' would comment out a line in YAML, which it does!

I then tried to remove the comment but found the Agent had gone completely down after the restart of the Agent to make the change – I wasn't sure if it was something I did at this time as I didn't realise that the YAML file was so central to the agent running, even if it was just the tags section.

My initial reaction was to check if the service was running. It wasn't but on re-start it didn't work. I got the following error which made me realise that it was me changing the YAML file that made it go down and that the YAML file needed to be perfectly parse-able for the agent to run. Any error will bring down the agent.



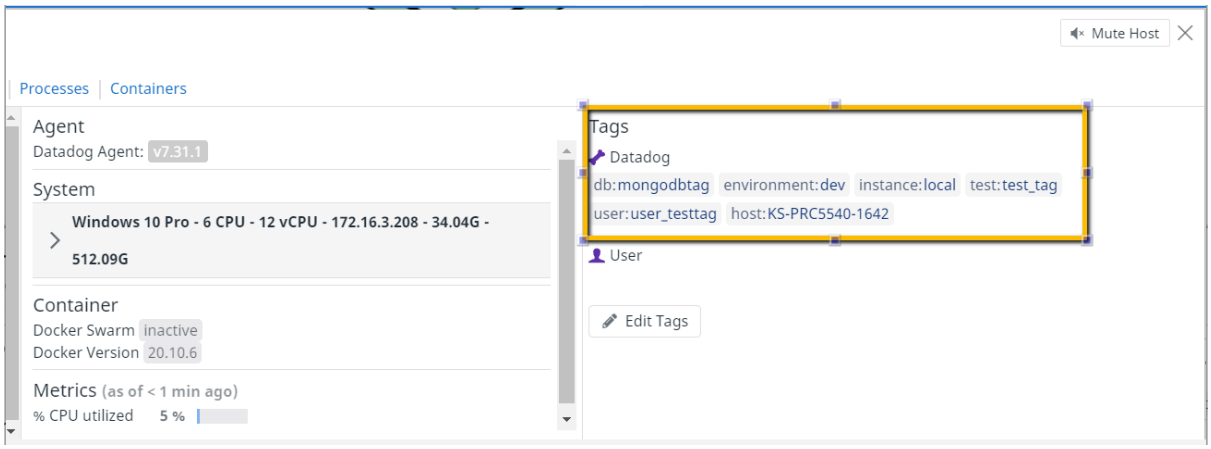
At this stage I wasn't sure how to find exactly what was wrong so I looked back on agent commands that I had run and found that the Agent status check when ran in power shell would give me the error handling I needed.

```
PS C:\Program Files\Datadog\Datadog Agent\bin> & "$env:ProgramFiles\Datadog\Datadog Agent\bin\agent.exe" status
Error: unable to set up global agent configuration: unable to load Datadog config file: While parsing config: yaml: line 104: did not find e
xpected '-' indicator
unable to set up global agent configuration: unable to load Datadog config file: While parsing config: yaml: line 104: did not find expected
 '-' indicator
```

I commented it all out again to ensure the agent would run.
I then changed the code around again and found that the following snippet would work. In changing the code I found that spaces in the code or not having spaces in the YAML code can cause issues.

```
tags:
    - "environment:dev"
    - "instance:local"
    - "test:test_tag"
    - "user:user_testtag"
    - "Db:MongoDbtag"
```

See below tags successfully added to HostMap.



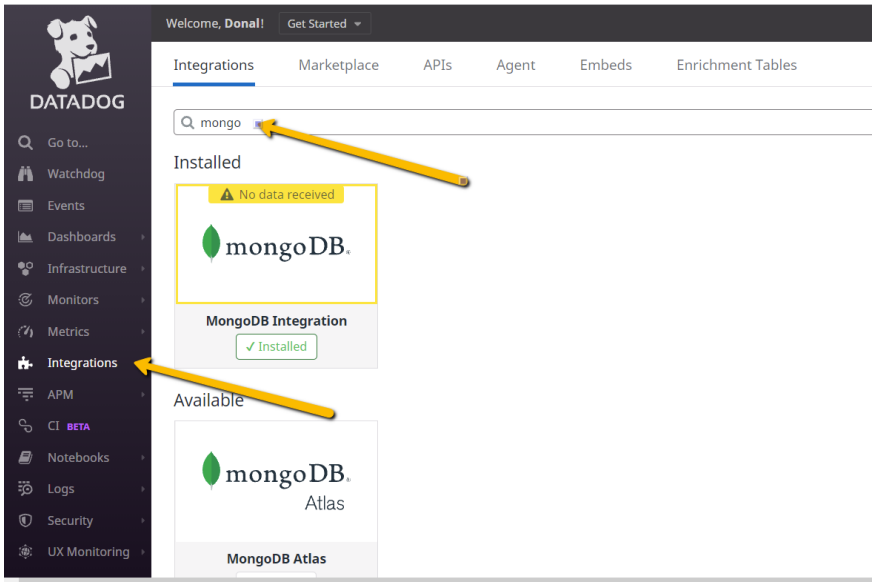
b. Install a database on your machine (MongoDB, MySQL, or PostgreSQL) and then install the respective Datadog integration for that database.

I chose to run a MongoDb on my machine. I also downloaded Mongo Compass (which is just a front-end for Mongodb)

Why would we want to gather metrics on MongoDB?
Like any other database you need to gather stats on the basics. Uptime, diskspace used, collection stats. These are just metrics on the health of the database however we can add further value – Say we have a table/collection in mongo db that's growing at a ferocious rate, Datadog can gather metrics on the disk space used and we can add this to a graph on a time-series and see how much and quickly the database is growing.
To add even further value we can apply alerts to these metrics graph to ensure we don't run out of disk space.

Installing the integration

Go to the DataDog API and click Integrations on the side bar. When you click integrations, you can see a search bar. In the example below I ran a search for Mongo.



- Once the integration shows, click on install.
- This will install the integration as you can see in the above screenshot it will say 'installed' on the integration.

Configuring the Agent for use with the integration

Although it is not a part of this task I will show how we configure MongoDb with the Agent.

When the integration is installed, click on configure. This gives us details on how to configure the integration. To configure we need to create a user in mongodb.

- Open a powershell window and change directory to the Mongo program files.

```
PS C:\Program Files> cd MongoDB\Server\5.0\bin
```

- Open Mongo Shell which is what we speak to Mongo DB with

```
PS C:\Program Files\MongoDB\Server\5.0\bin> .\mongo
MongoDB shell version v5.0.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a199b99d-d191-49bd-b4b0-e6f5b5af2332") }
MongoDB server version: 5.0.3
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
```

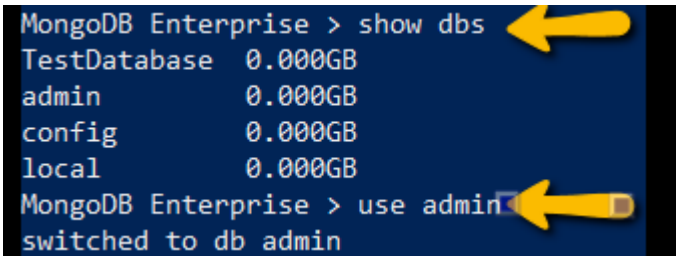
- Create a read-only user for the datadog agent in the admin database
    - Submit the following code in mongo shell.

```
# Authenticate as the admin user.
use admin
db.auth("admin", "<YOUR_MONGODB_ADMIN_PASSWORD>")
```

```
# On MongoDB 2.x, use the addUser command.
db.addUser("datadog", "<UNIQUEPASSWORD>", true)

# On MongoDB 3.x or higher, use the createUser command.
db.createUser({
  "user": "datadog",
  "pwd": "<UNIQUEPASSWORD>",
  "roles": [
    { role: "read", db: "admin" },
    { role: "clusterMonitor", db: "admin" },
    { role: "read", db: "local" }
  ]
})
```

- In mongo shell I can run the `db.getUsers()` to show that I have created the user in the admin database with the relevant roles. See below commands on how to use the admin database and how to show the users.

-



-

c. Create a custom Agent check that submits a metric named my_metric with a random value between 0 and 1000.

When creating a check, in this case called, 'my_metric' we need to update our YAML and Python file and configure both to what we need it to do. Each file relies on the other to run the check.

1. *my_metric.yaml*

Located in >> *C:\ProgramData\Datadog\conf.d\ my_metric.yaml.* See below for the code used.

Note: I was not able to open this file and it showed no errors until I opened it using Azure DataStudio

Code:

```
instances: [{}]
```

2. my_metric.py

Located in >> *C:\ProgramData\Datadog\checks.d\ my_metric.py*

Code:

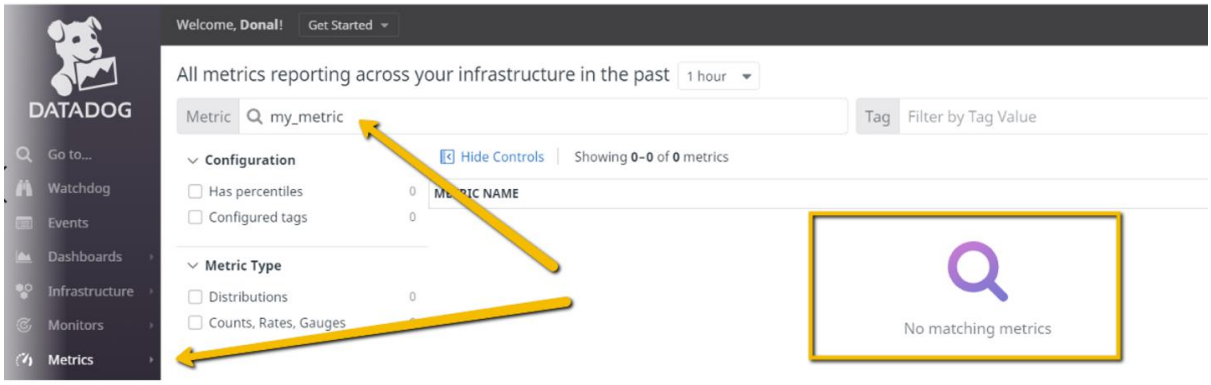
```
import random

from datadog_checks.base import AgentCheck

__version__ = "1.0.0"

class MyClass(AgentCheck):
    def check(self, instance):
        self.count(
            "example_metric.count",
            2,
            tags=["env:dev","metric_submission_type:count"],
        )
        self.count(
            "example_metric.decrement",
            -1,
            tags=["env:dev","metric_submission_type:count"],
        )
        self.count(
            "example_metric.increment",
            1,
            tags=["env:dev","metric_submission_type:count"],
        )
        self.rate(
            "example_metric.rate",
            1,
            tags=["env:dev","metric_submission_type:rate"],
        )
        self.gauge(
            "example_metric.gauge",
            random.randint(0, 10),
            tags=["env:dev","metric_submission_type:gauge"],
        )
        self.monotonic_count(
            "example_metric.monotonic_count",
            2,
            tags=["env:dev","metric_submission_type:monotonic_count"],
        )

        # Calling the functions below twice simulates
        # several metrics submissions during one Agent run.
        self.histogram(
```

```
            "example_metric.histogram",
            random.randint(0, 1000),
            tags=["env:dev","metric_submission_type:histogram"],
        )
        self.histogram(
            "example_metric.histogram",
            random.randint(0, 1000),
            tags=["env:dev","metric_submission_type:histogram"],
        )
```

When you run the agent check command it appears to be running:

```
========
Collector
========

  Running Checks
  ==============

    my_metric (1.0.0)
    -----------------
      Instance ID: my_metric:d884b5186b651429 [←[32mOK←[0m]
      Configuration Source: file:C:\ProgramData\Datadog\conf.d\my_metric.yaml
      Total Runs: 1
      Metric Samples: Last Run: 8, Total: 8
      Events: Last Run: 0, Total: 0
      Service Checks: Last Run: 0, Total: 0
      Average Execution Time : 0s
      Last Execution Date : 2021-10-12 16:21:48 BST / 2021-10-12 15:21:48 UTC (1634052108000)
      Last Successful Execution Date : 2021-10-12 16:21:48 BST / 2021-10-12 15:21:48 UTC (1634052108000)
```

However it is not showing in all metrics and it is also giving a loader error in the Agent.



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

I had to update the interval we need to change the 'rate' in the python file. The rate is the measure of time in between checks and is measured in seconds. It was previously 1 but I have changed it to 45 as requested.

New tactic: start again and go over every bit of documentation:

- I began by installing the latest version of python
- Double checked the python and YAML file names

- It is not the YAML file as there is no error anymore –
- The check is running. This means its not an issue declaring the host.
- I then ran it with idle – So I'm definitely missing the module but I don't know where that is stored.

```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: C:\ProgramData\Datadog\checks.d\metrics_example.py =========
Traceback (most recent call last):
  File "C:\ProgramData\Datadog\checks.d\metrics_example.py", line 3, in <module>
    from datadog_checks.base import AgentCheck
ModuleNotFoundError: No module named 'datadog_checks'
```

- After troubleshooting it appears I don't have *datadog-checks-base* installed.
- To do this I need to use the command
    - `pip install datadog-checks-base`
- I can use pip as I don't have the python path in my environment variables. I followed the below article to add this
    - https://datatofish.com/add-python-to-windows-path/
- Close and reopen powershell

```
PS C:\Users\donal.browne> pip install datadog-checks-base
Collecting datadog-checks-base
  Downloading datadog_checks_base-23.1.2-py2.py3-none-any.whl (208 kB)
     |████████████████████████████████| 208 kB 3.3 MB/s
Installing collected packages: datadog-checks-base
Successfully installed datadog-checks-base-23.1.2
```

- Now there is no module named YAML - ModuleNotFoundError: No module named 'yaml'

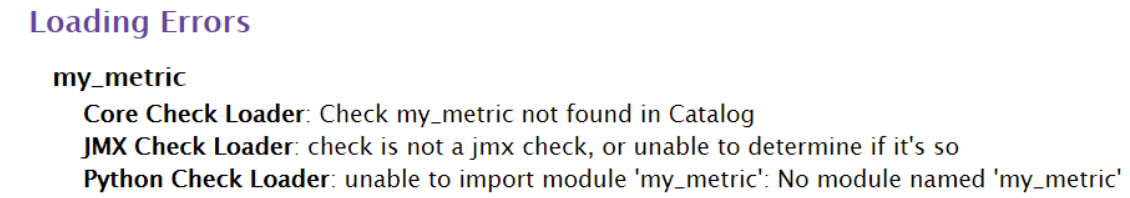
```
PS C:\Users\donal.browne> pip install pyyaml
Collecting pyyaml
  Downloading PyYAML-5.4.1.tar.gz (175 kB)
     |████████████████████████████████| 175 kB 3.2 MB/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Building wheels for collected packages: pyyaml
  Building wheel for pyyaml (PEP 517) ... done
  Created wheel for pyyaml: filename=PyYAML-5.4.1-cp310-cp310-win_amd64.whl size=45654 sha256=c5e3da092be9609b29051a0c62917183bec8ccdd0ff3b1cdd3d8a7ddedfe9ef1
  Stored in directory: c:\users\donal.browne\appdata\local\pip\cache\wheels\c7\0d\22\696ee92245ad710f506eee79bb05c740d8abccd3ecdb778683
Successfully built pyyaml
Installing collected packages: pyyaml
Successfully installed pyyaml-5.4.1
```

- Now there is no module called six

```
Traceback (most recent call last):
  File "C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\Lib\site
ackages\datadog_checks\base\checks\base.py", line 18, in <module>
    from six import PY2, binary_type, iteritems, raise_from, text_type
ModuleNotFoundError: No module named 'six'
```

```
PS C:\Users\donal.browne> pip install six
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six
Successfully installed six-1.16.0
```

- I was restarting the Agent after each install but I'm still getting the same error. See loading error below



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

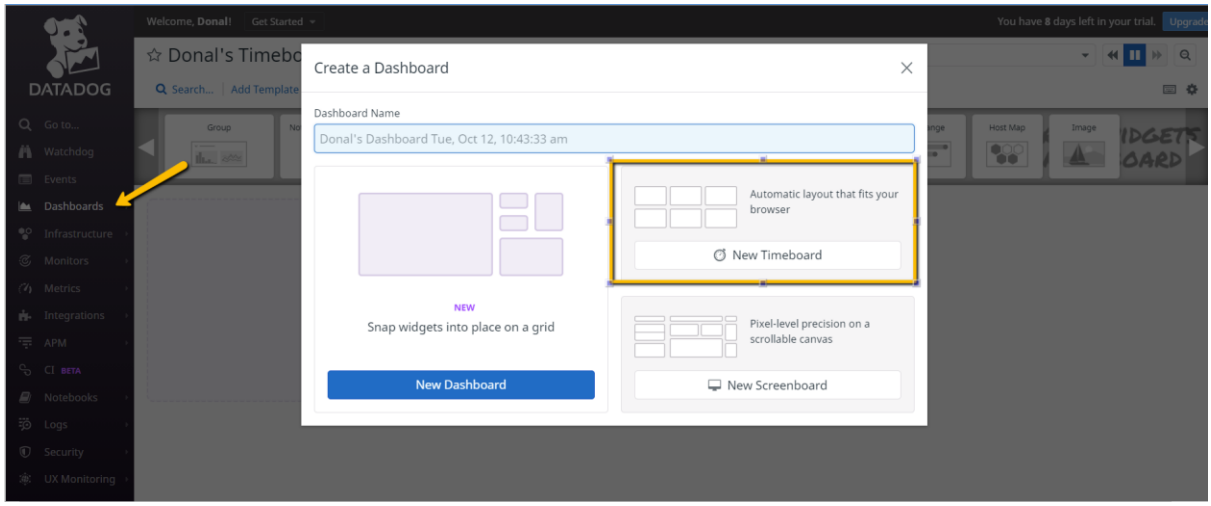
You can also change the yaml file in the from the agent or the root file

```
1  instances: [{}]
2
3  instances:
4    - min_collection_interval: 45
```

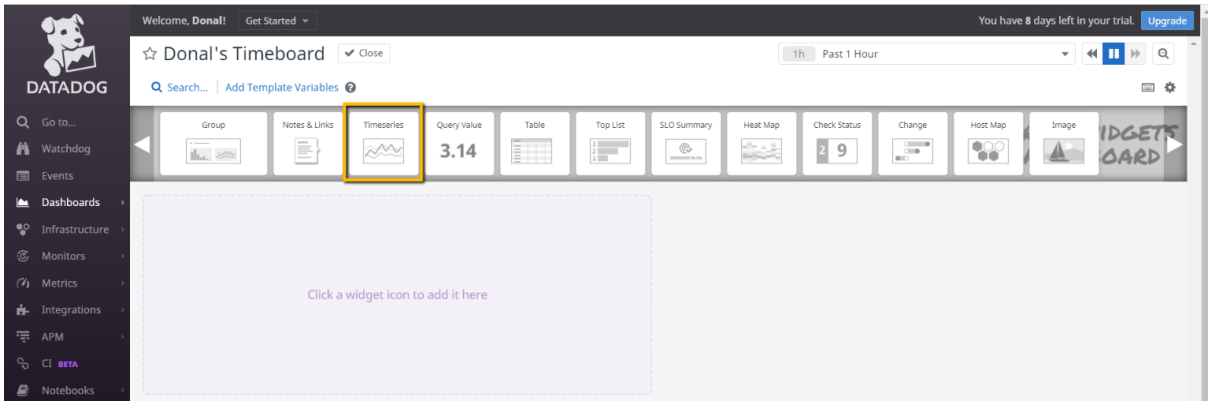
Question 3:
Visualizing Data:

Utilize the Datadog API to create a Timeboard that contains:

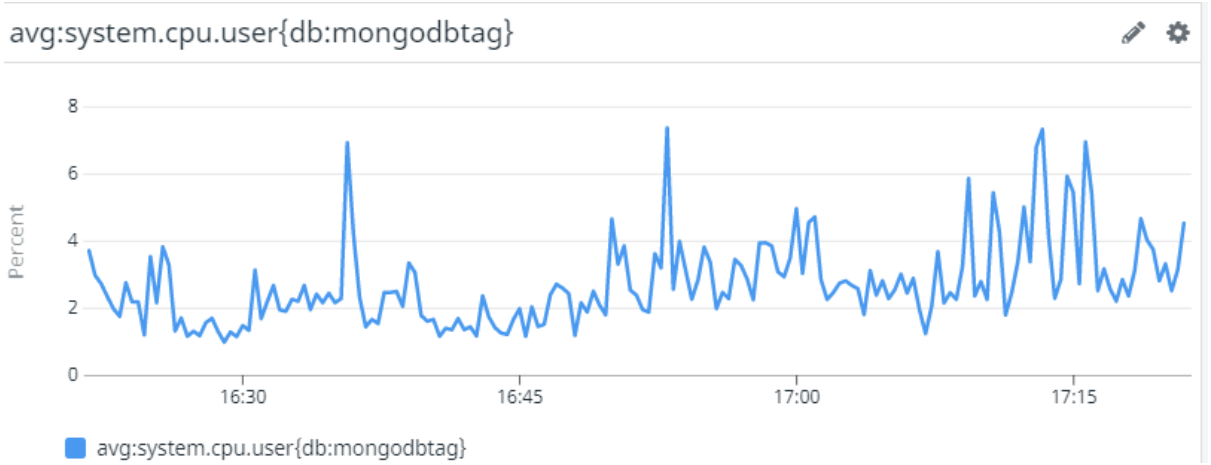
a. Your custom metric scoped over your host.
Click on 'Dashboard' on the sidebar in API. The widget below will show – From here click on 'New Timeboard'



Click and drag time series

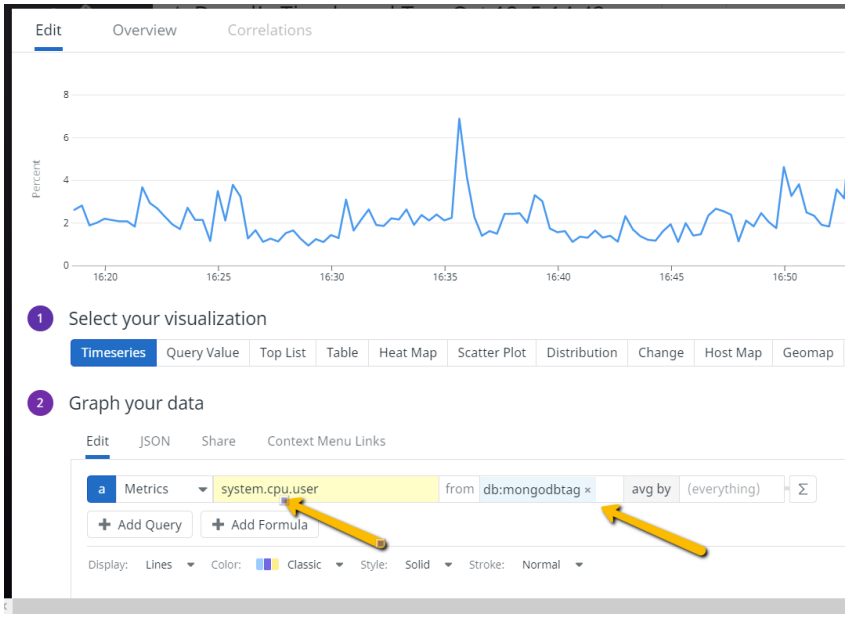


Below is a timeseries of CPU utilization on my new Timeboard.



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

I don't have the integration set up properly however if I did, I could look at CPU consumption on any server with MongoDB on it due to my tags.



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

The code that needs to be executed for this to happen.

```
avg:system.cpu.user{db:mongodbtag}.rollup(sum, 3600)
```

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

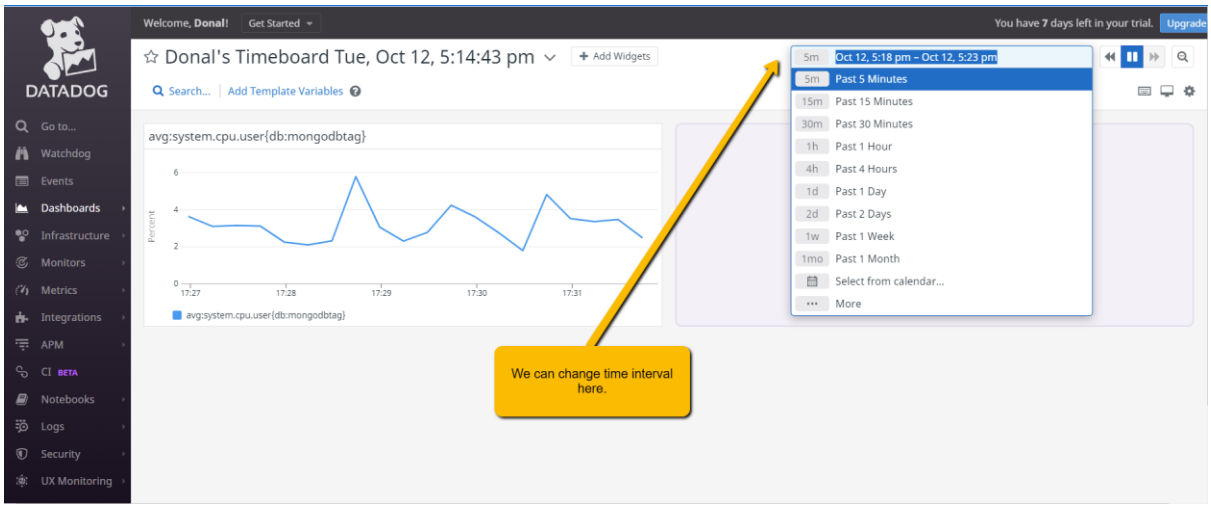
I don’t have the metric working however this is the code that will develop the Timeboard

```
{
    "viz": "timeseries",
    "requests": [
        {
            "formulas": [
                {
                    "formula": "query1"
                }
            ],
            "queries": [
                {
                    "data_source": "metrics",
                    "name": "query1",
                    "query": "avg:system.cpu.user{db:mongodbtag}.rollup(sum, 3600)"
                }
            ],
            "response_format": "timeseries",
            "type": "line",
            "style": {
                "palette": "dog_classic",
                "type": "solid",
                "width": "normal"
            }
        }
    ],
    "yaxis": {
```

```
        "scale": "linear",
        "min": "auto",
        "max": "auto",
        "include_zero": true,
        "label": ""
    },
    "markers": []
}
```

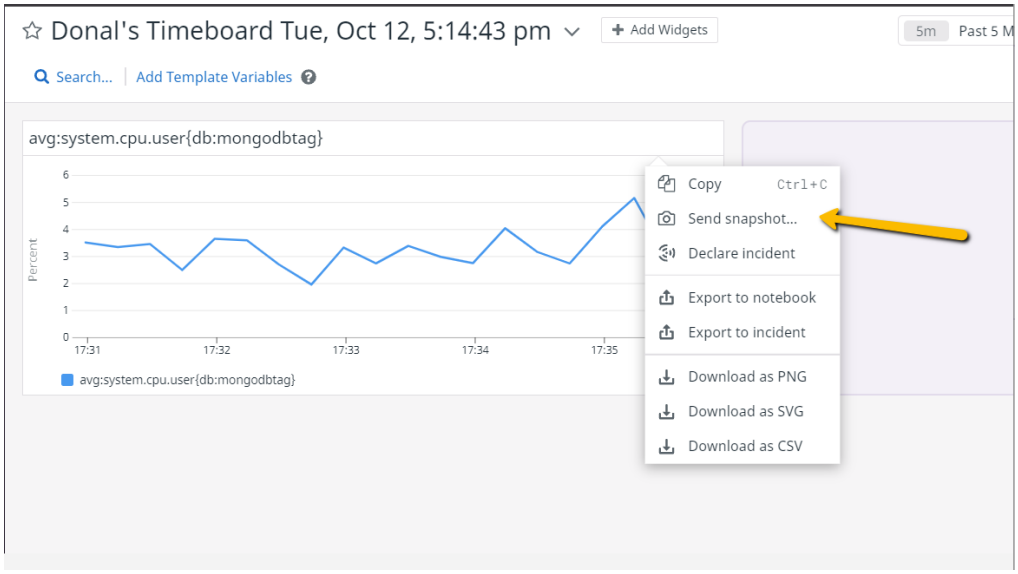
e. Once this is created, access the Dashboard from your Dashboard List in the UI: Set the Timeboard's timeframe to the past 5 minutes

We can change the time interval as per below –

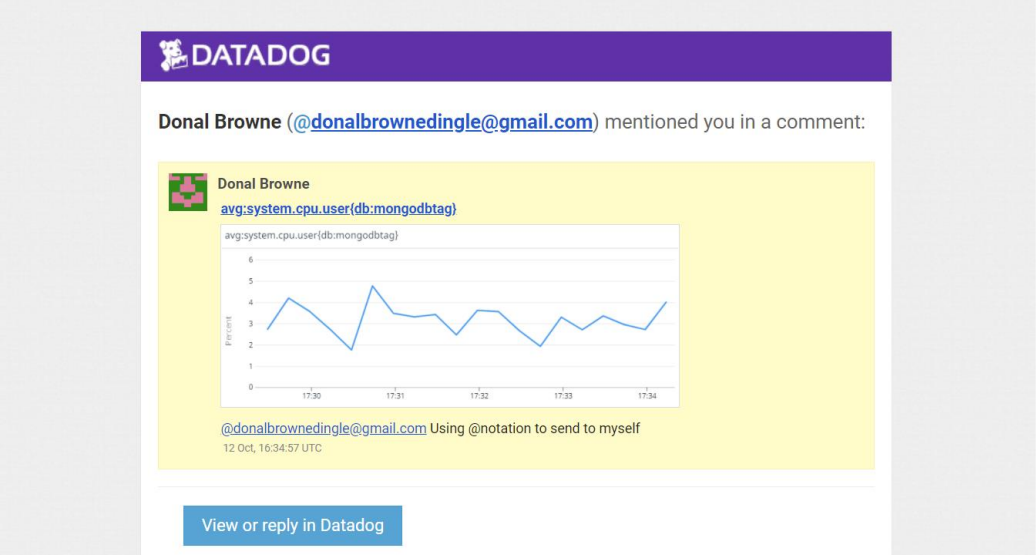


f. Take a snapshot of this graph and use the @ notation to send it to yourself.

We can send a snapshot of a dashboard to anyone using the @notation – See below how I have sent myself an email



Here is a copy of the email to show that it is working.



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

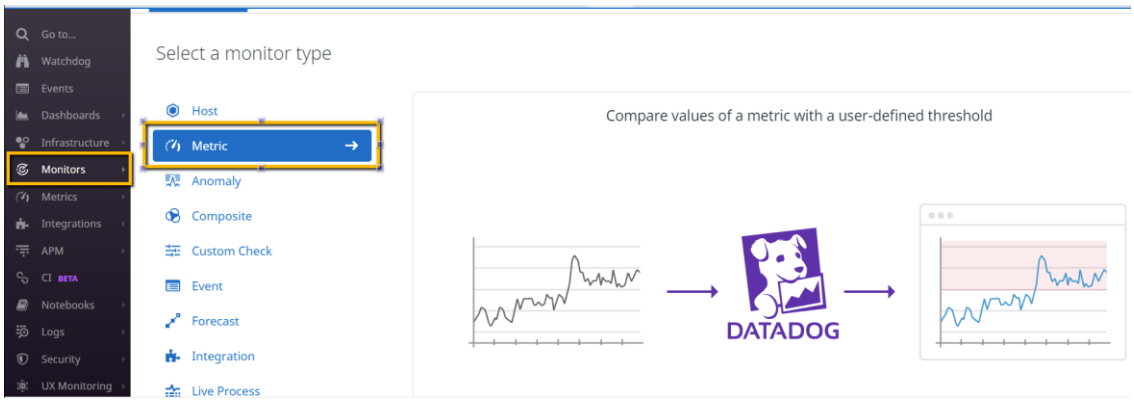
Question 4: Monitoring Data

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.

a. 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:

I don't have my_metric set up but I can show you how to set up the monitor on a metric on my host.

- Click on 'Monitors' on the side bar > Select 'New Monitor'.
- This shows a page of monitor types. In this case we want to select 'Metric' as we want to monitor a metric.



- From this page we can select
  - Our detection method – (we will want a warning threshold)

- You can select tags to apply where you want to monitor – For example, mongo databases only and how much cpu they are using?
- This would be of value where you want to compare resource usage between hosts, databases or applications.

```
avg:system.cpu.system{host:KS-PRC5540-1642}
```

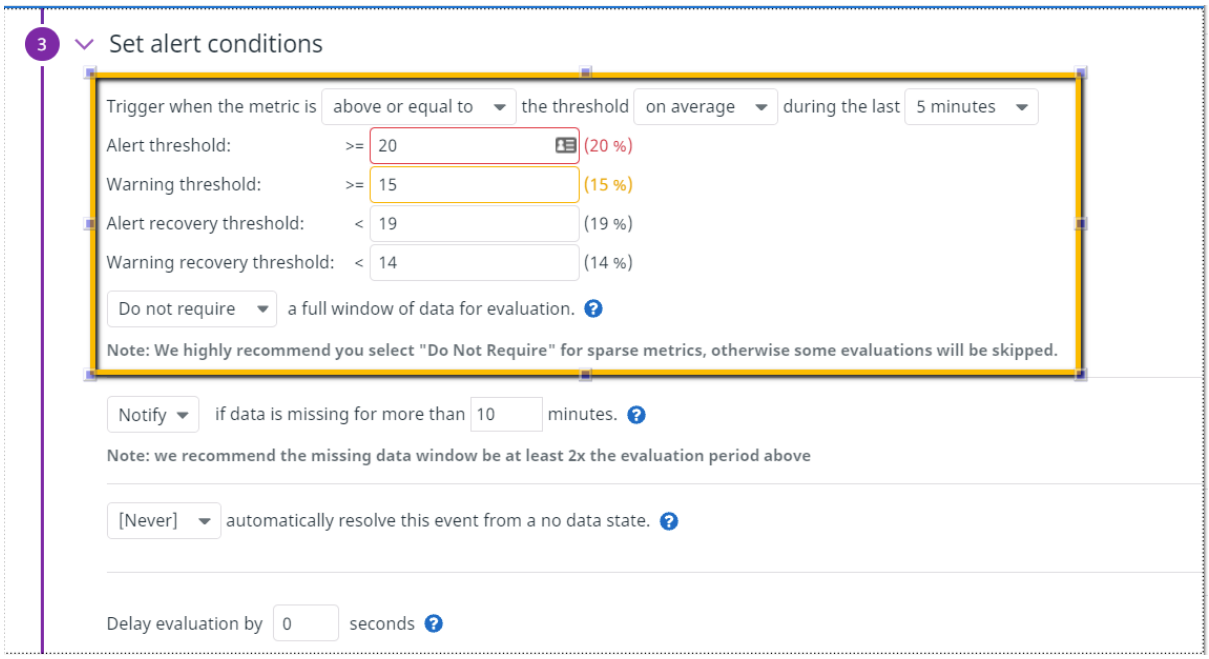
- We can select what metric we want to monitor (It should be my_metric but I will use another example)
- We can set the values of when we want to alert (This should a warning of 500 and alert at 800 however I will change this)
- Who we want to Notify
- What do we want to say?

-

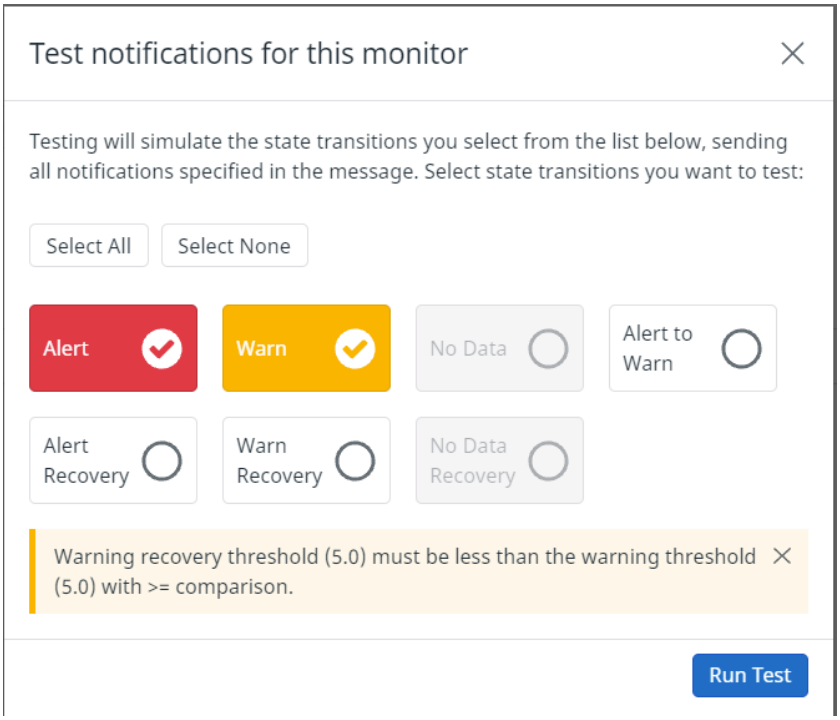
b. Warning threshold of 500

c. Alerting threshold of 800 (b and c are together below)

As I am using cpu as a metric the values are different – see screenshots below of how to set these. This is in the 'Set Alert Conditions' section.

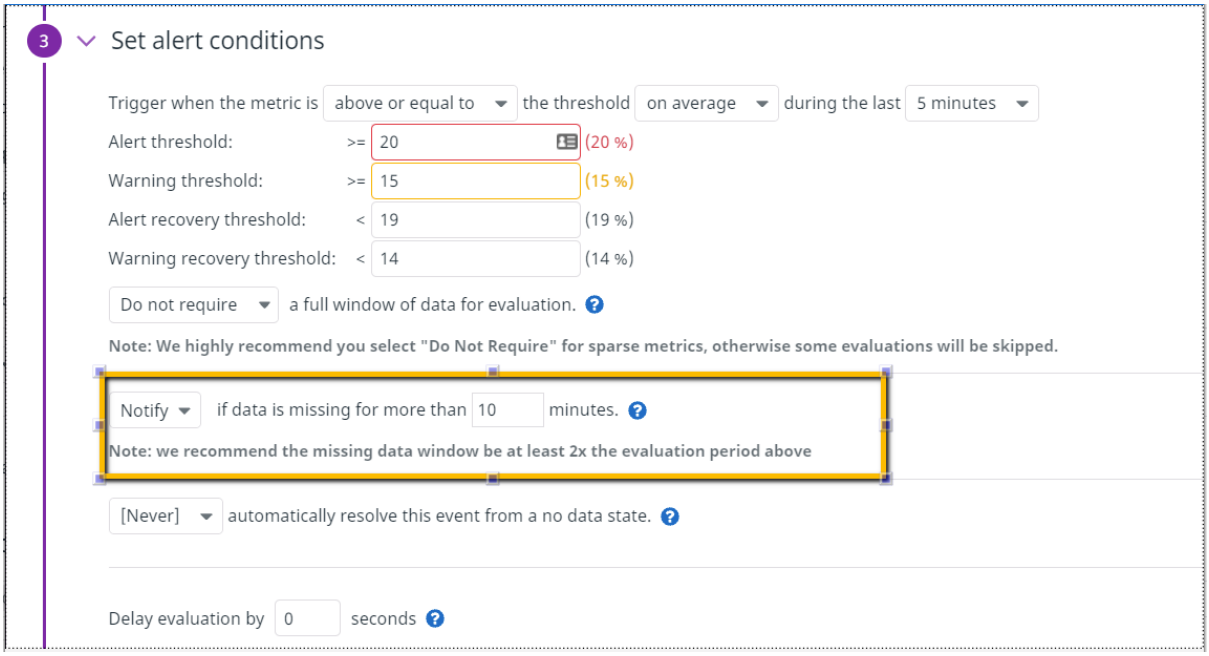


- When running a test an error may occur, as you can see, the recovery threshold must be less than the warning threshold.



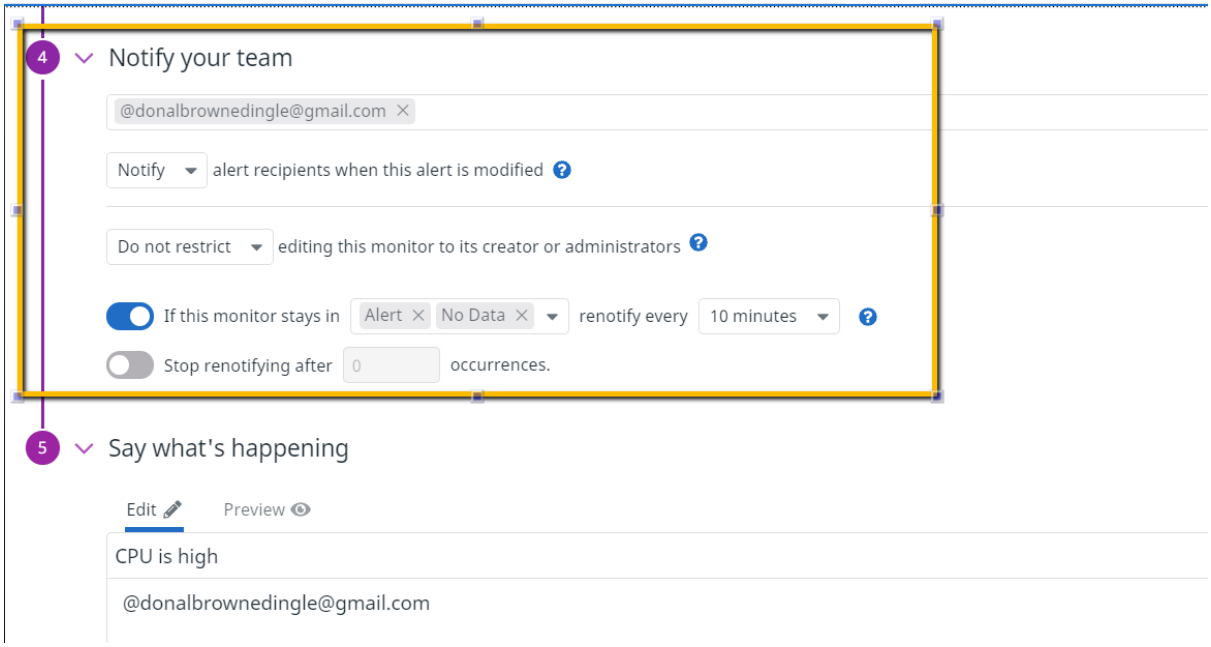
d. And also ensure that it will notify you if there is No Data for this query over the past 10m.

In the 'Set alert conditions' you can notify if data is missing for more than 10 minutes.



e. Please configure the monitor's message so that it will:
Send you an email whenever the monitor triggers.

This is set in the notify your team section. Here you can choose who you want to alert.



You can also choose to re-notify if the alert is still over it's threshold – I would advise this for a situation where CPU or memory is over 95% on a server. Personally, I would have an alert on this every minute.

I would also have this alerting in a Slack notification channel. We can use the notation @Slack if the integration is in place.

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

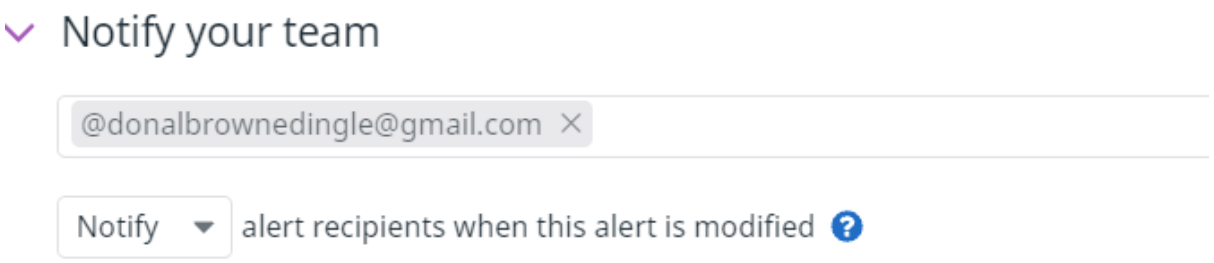
We can use different variables in the message box for this. The message box is quite friendly in error handling. See code below.

```
{{#is_alert}}
CPU is above 10%....do something! <@donalbrownedingle@gmail.com >
{{/is_alert}}

{{#is_warning}}
CPU is above 5%...be careful <@donalbrownedingle@gmail.com>
{{/is_warning}}

{{#is_recovery}}
   CPU is recovering <@donalbrownedingle@gmail.com>
{{/is_recovery}}

{{#is_no_data}}
 There has been no data for more than 10 minutes. Check it out!!
{{/is_no_data}}
```

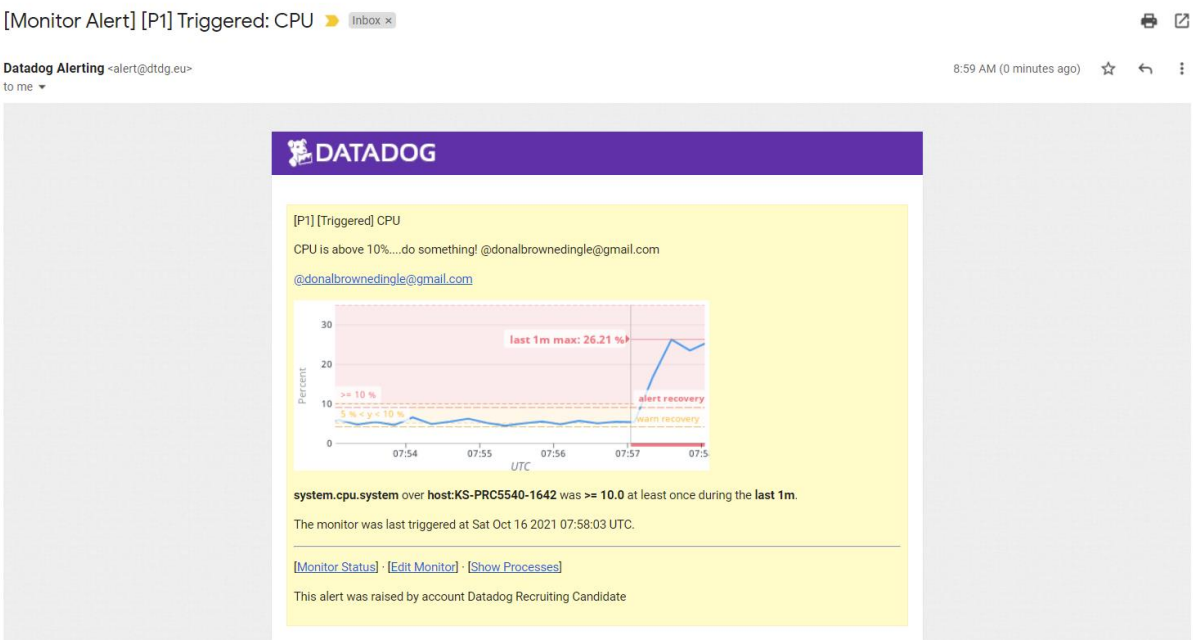
g. Include the metric value that caused the monitor to trigger and host ip when the Monitor triggers an Alert state.

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

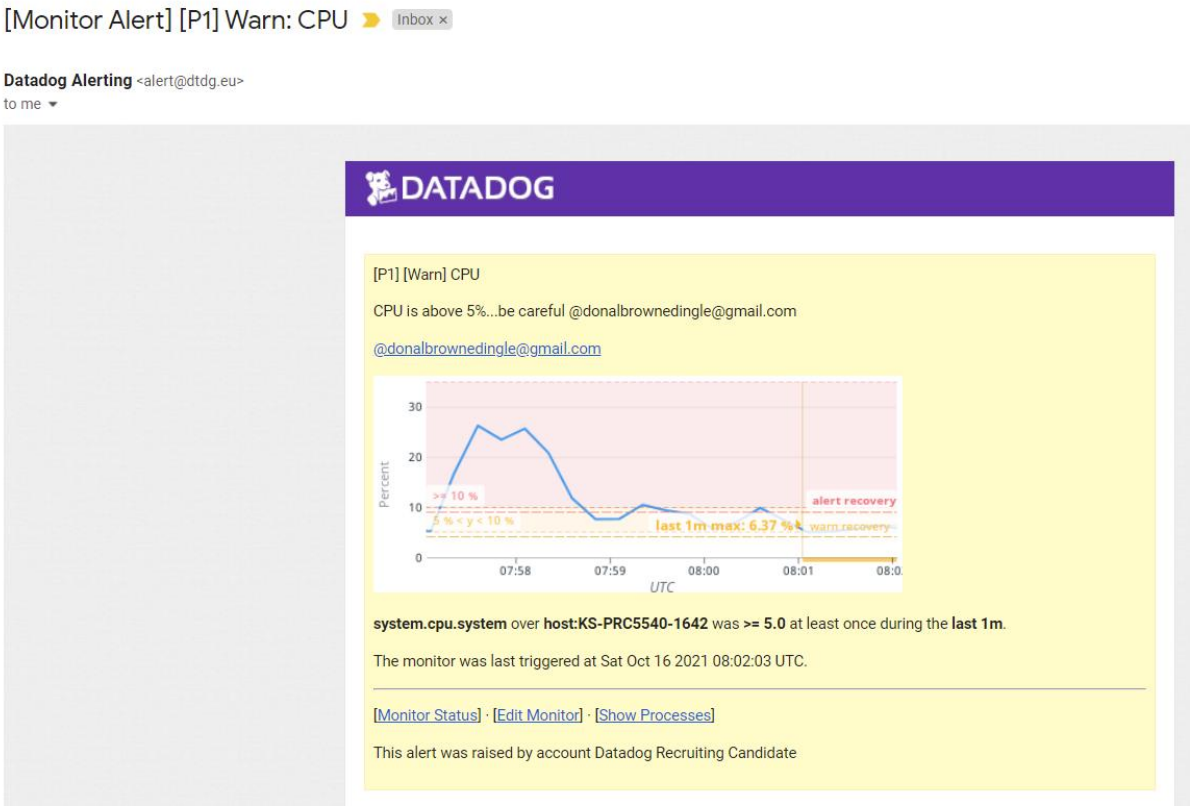
Alerts are firing but not sending me emails – I initially thought it was the code that I got wrong but it turns out, I had not filled in my own email!!



After I got this working – you can see the email here for CPU above 10%

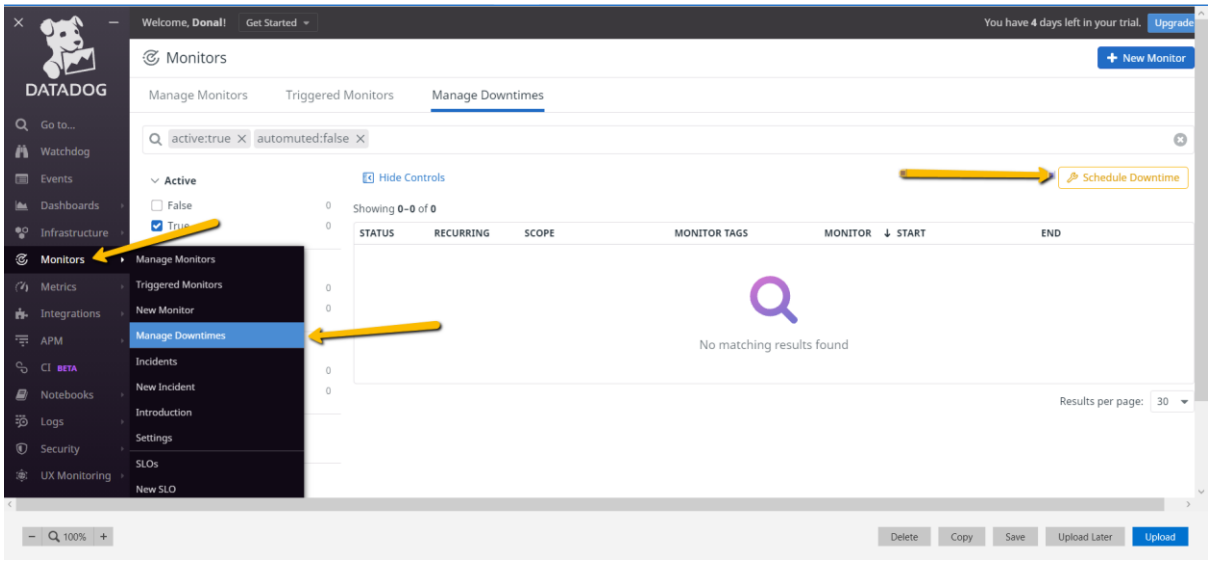


We can also see separate emails for different alerts/warning.



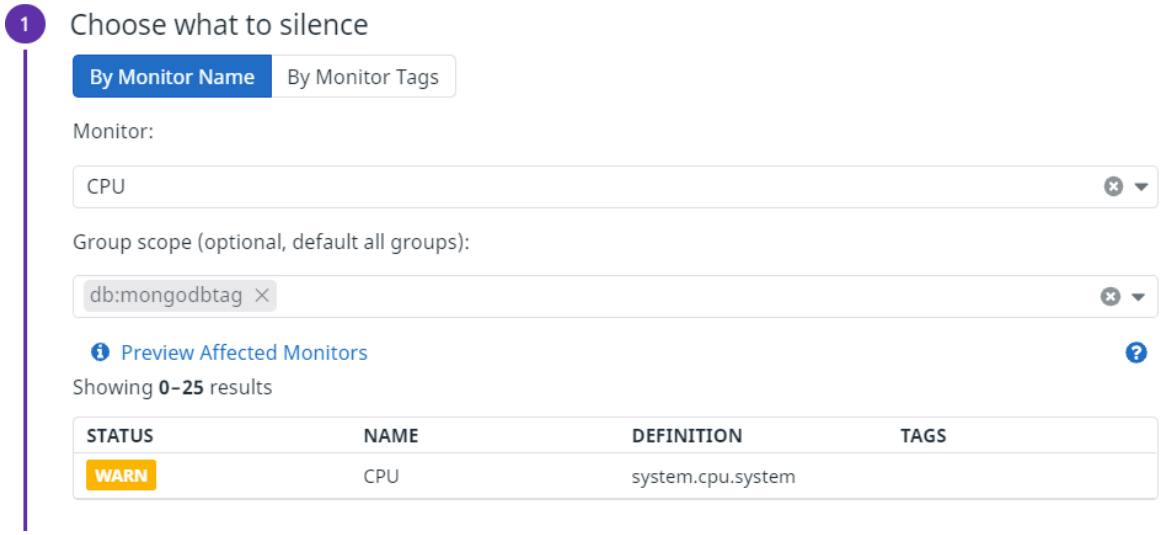
i. **Bonus Question**: Since this monitor is going to alert pretty often, you don't want to be alerted when you are out of the office. Set up two scheduled downtimes for this monitor:

To set up scheduled downtime for the monitor, go to the side bar and select Monitors > Manage downtimes. From that page select 'Schedule Downtimes'

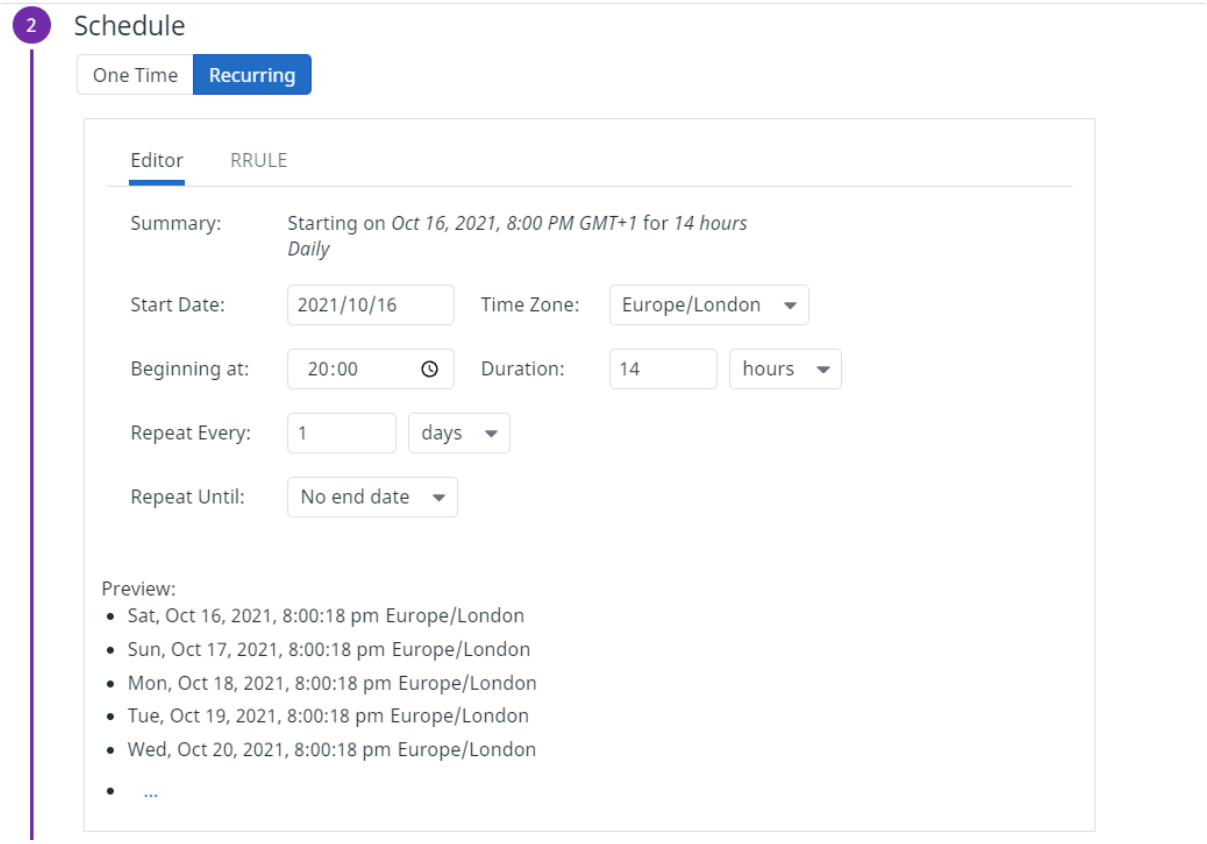


j. One that silences it from 7pm to 9am daily on M-F

- Select the monitor that you want to schedule downtime on.
- In the below screenshot I have selected that we want to monitor CPU on all hosts that have a mongoDB on it.



We schedule the downtime Daily, for 14 hours – this covers between 7 P.M and 9 A.M the following morning. We can this is working in the preview section at the bottom of the screenshot.



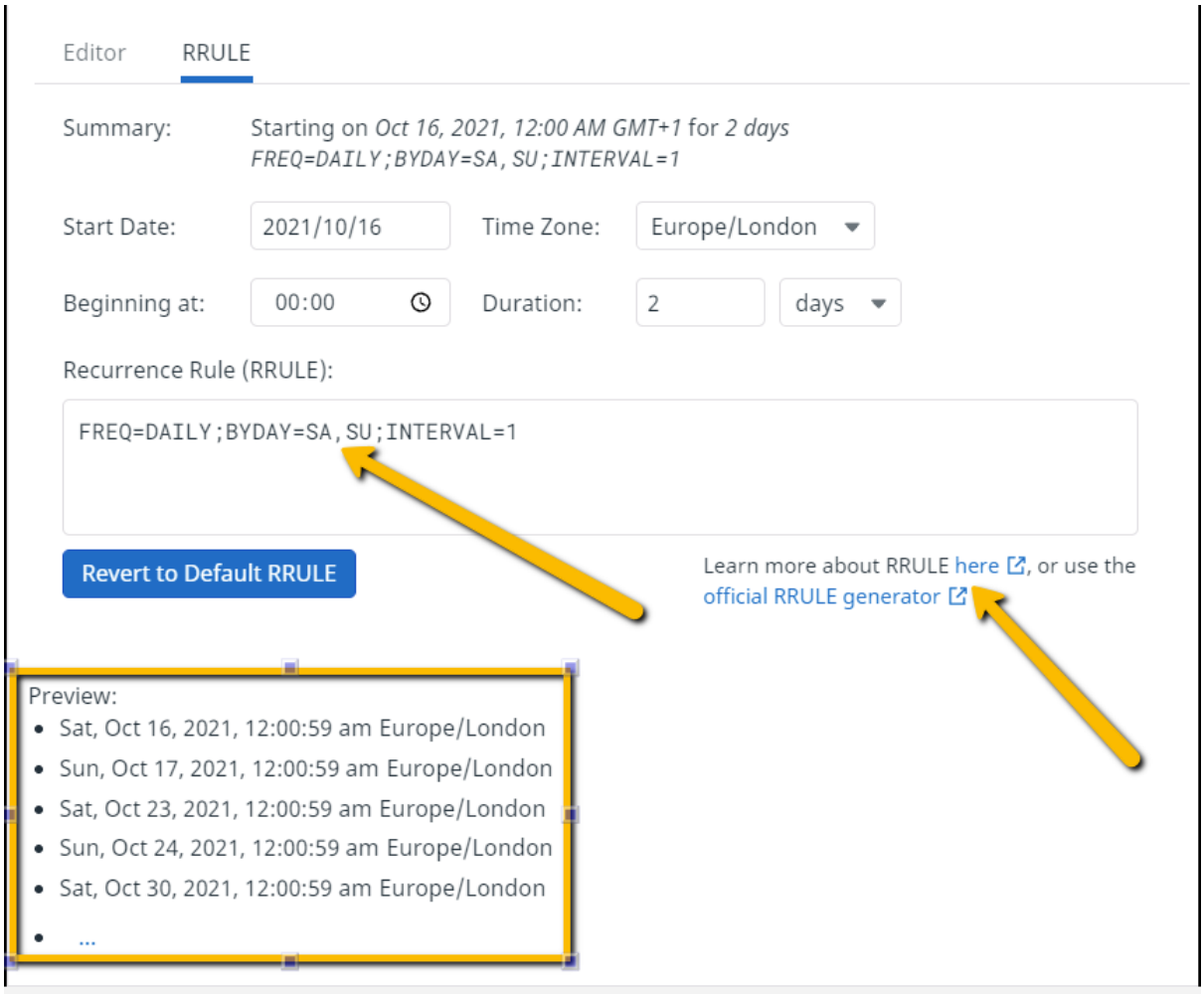
-

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

Use the RRule to define what days you want to set the alert for. You can get more help below.

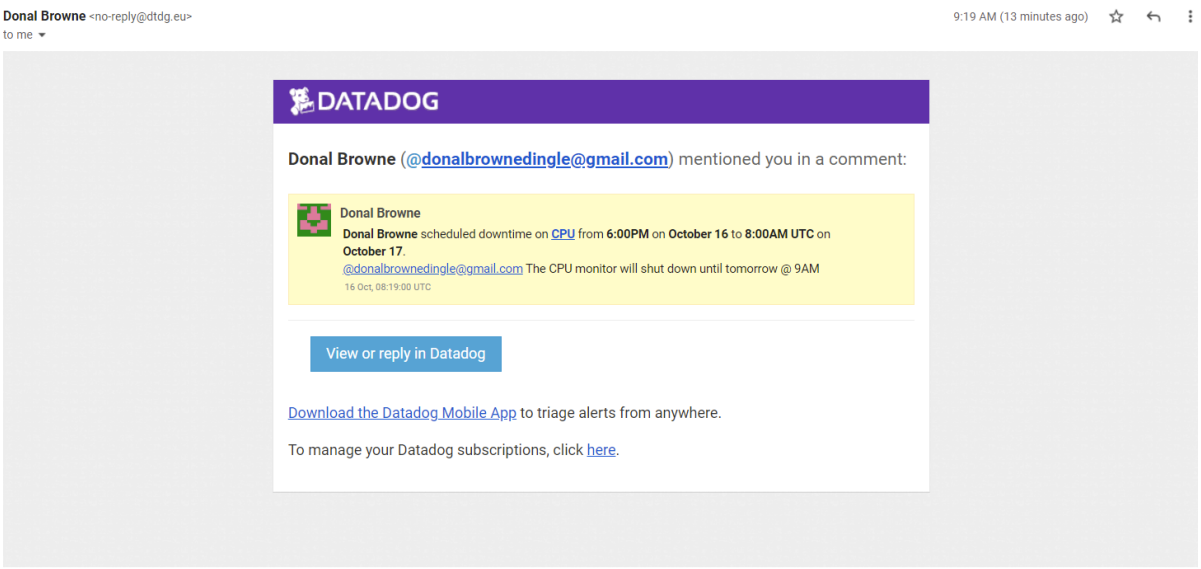
Code used:

```
FREQ=DAILY;BYDAY=SA,SU;INTERVAL=1
```



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

See notification of scheduled downtime for the monitor called ‘CPU’



Question 5: Collecting APM Data:

Given the following Flask app (or any Python/Ruby/Go app of your choice) instrument this using Datadog's APM solution:

```
from flask import Flask
import logging
import sys

# 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')
```

Steps to get set up our app

- Install pip if you have not already installed. This is a method of easy_install.
    - https://flask.palletsprojects.com/en/0.12.x/installation/#windows-easy-install

- Install 'virtualenv'. By running the below code

- https://flask.palletsprojects.com/en/0.12.x/installation/#installation
- `> pip install virtualenv`

See result below:

```
PS C:\Users\donal.browne> pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.8.1-py2.py3-none-any.whl (5.3 MB)
     |████████████████████████████████| 5.3 MB 3.3 MB/s
Requirement already satisfied: six<2,>=1.9.0 in c:\users\donal.browne\appdata\local\programs\python\python310\lib\site-packages (from virtualenv) (1.16.0)
Collecting filelock<4,>=3.0.0
  Downloading filelock-3.3.1-py3-none-any.whl (9.7 kB)
Collecting distlib<1,>=0.3.1
  Downloading distlib-0.3.3-py2.py3-none-any.whl (496 kB)
     |████████████████████████████████| 496 kB 3.2 MB/s
Collecting platformdirs<3,>=2
  Downloading platformdirs-2.4.0-py3-none-any.whl (14 kB)
Collecting backports.entry-points-selectable>=1.0.4
  Downloading backports.entry_points_selectable-1.1.0-py2.py3-none-any.whl (6.2 kB)
Installing collected packages: platformdirs, filelock, distlib, backports.entry-points-selectable, virtualenv
Successfully installed backports.entry-points-selectable-1.1.0 distlib-0.3.3 filelock-3.3.1 platformdirs-2.4.0 virtualenv-20.8.1
```

- Now to install flask > https://flask.palletsprojects.com/en/0.12.x/installation/#installation
- I now need to create my own environment.
- Install the latest version of 'venv'

```
py -3 -m venv venv
```

```
PS C:\Users\donal.browne> py -3 -m venv venv
```

- We now run the below command to activate and ensure that we are running the correct environment.

```
Venv\Scripts\activate
```

```
PS C:\Users\donal.browne> venv\Scripts\activate
(venv) PS C:\Users\donal.browne>
```

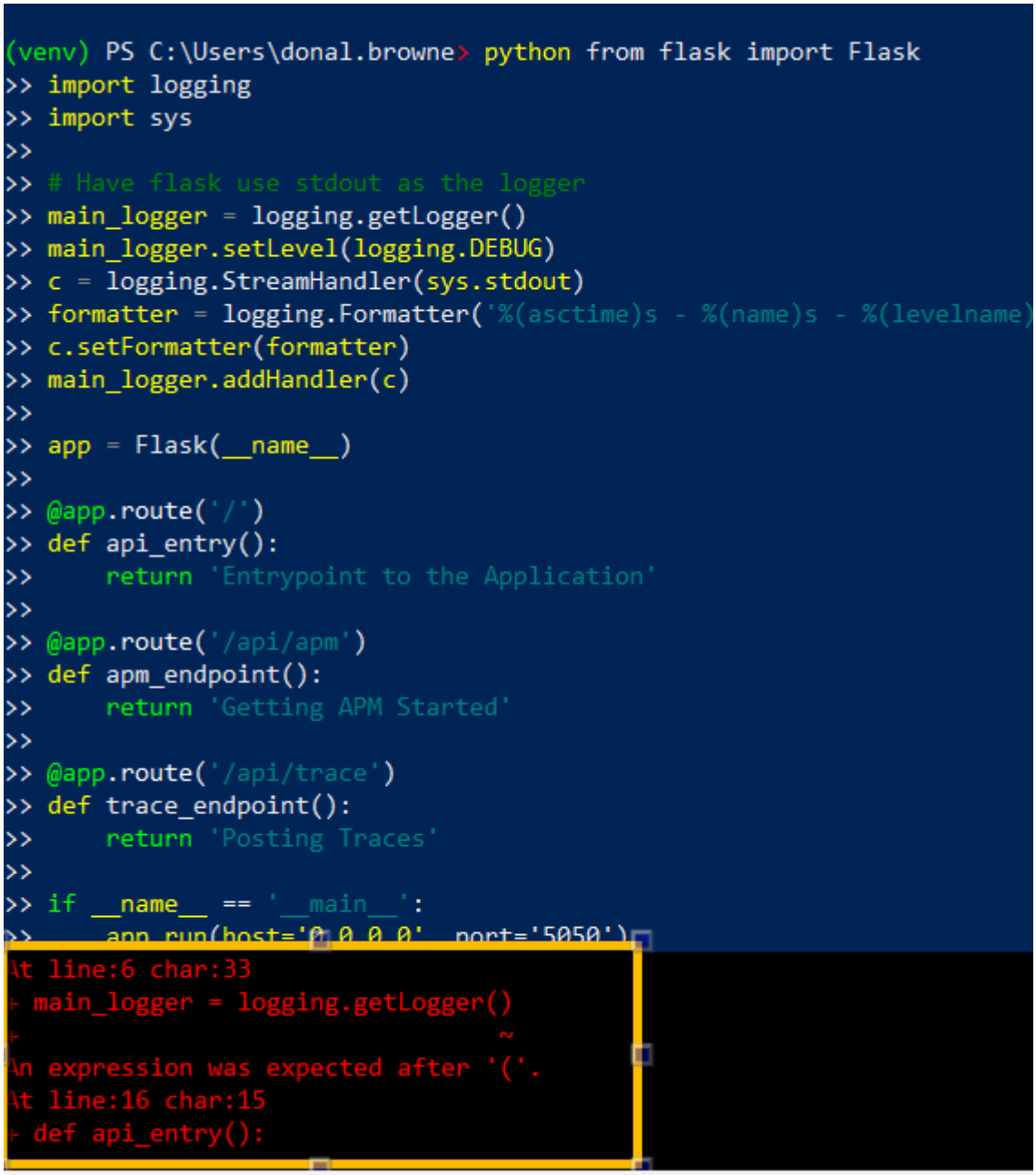
- Once we are in the environment, we install Flask.

```
pip install Flask
```

```
PS C:\Users\donal.browne> venv\Scripts\activate
(venv) PS C:\Users\donal.browne> pip install Flask
Collecting Flask
  Downloading Flask-2.0.2-py3-none-any.whl (95 kB)
     |████████████████████████████████| 95 kB 780 kB/s
Collecting Jinja2>=3.0
  Downloading Jinja2-3.0.2-py3-none-any.whl (133 kB)
     |████████████████████████████████| 133 kB 2.2 MB/s
Collecting click>=7.1.2
  Downloading click-8.0.3-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 1.7 MB/s
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
Collecting Werkzeug>=2.0
  Downloading Werkzeug-2.0.2-py3-none-any.whl (288 kB)
     |████████████████████████████████| 288 kB 3.3 MB/s
Collecting colorama
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.0.1-cp310-cp310-win_amd64.whl (15 kB)
Installing collected packages: MarkupSafe, colorama, Werkzeug, Jinja2, itsdangerous, click, Flask
Successfully installed Flask-2.0.2 Jinja2-3.0.2 MarkupSafe-2.0.1 Werkzeug-2.0.2 click-8.0.3 colorama-0.4.4 itsdangerous-2.0.1
```

Tried to run the code for the app but got an error advising that the language was not being recognised. I googled it and found out that you needed to use the word 'python' before running the code.

I then got the below error that is an error with the code on line 6 at character 33.



It then clicked with me that this was python code and needed to be ran from a python file

```
(Venv) PS C:\Users\donal.browne>  set FLASK_APP=hello.py
>>  flask run
 * Serving Flask app 'hello.py' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
Traceback (most recent call last):
  File "C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\lib\runpy.py", line 196, in _run_module_as_main
    return _run_code(code, main_globals, None,
  File "C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\lib\runpy.py", line 86, in _run_code
    exec(code, run_globals)
  File "C:\Users\donal.browne\Venv\Scripts\flask.exe\__main__.py", line 7, in <module>
  File "C:\Users\donal.browne\venv\lib\site-packages\flask\cli.py", line 994, in main
    cli.main(args=sys.argv[1:])
  File "C:\Users\donal.browne\venv\lib\site-packages\flask\cli.py", line 600, in main
    return super().main(*args, **kwargs)
  File "C:\Users\donal.browne\venv\lib\site-packages\click\core.py", line 1053, in main
    rv = self.invoke(ctx)
  File "C:\Users\donal.browne\venv\lib\site-packages\click\core.py", line 1659, in invoke
    return _process_result(sub_ctx.command.invoke(sub_ctx))
  File "C:\Users\donal.browne\venv\lib\site-packages\click\core.py", line 1395, in invoke
    return ctx.invoke(self.callback, **ctx.params)
  File "C:\Users\donal.browne\venv\lib\site-packages\click\core.py", line 754, in invoke
    return __callback(*args, **kwargs)
  File "C:\Users\donal.browne\venv\lib\site-packages\click\decorators.py", line 84, in new_func
    return ctx.invoke(f, obj, *args, **kwargs)
  File "C:\Users\donal.browne\venv\lib\site-packages\click\core.py", line 754, in invoke
    return __callback(*args, **kwargs)
  File "C:\Users\donal.browne\venv\lib\site-packages\flask\cli.py", line 853, in run_command
    run_simple(
  File "C:\Users\donal.browne\venv\lib\site-packages\werkzeug\serving.py", line 1010, in run_simple
    inner()
  File "C:\Users\donal.browne\venv\lib\site-packages\werkzeug\serving.py", line 950, in inner
    srv = make_server(
  File "C:\Users\donal.browne\venv\lib\site-packages\werkzeug\serving.py", line 782, in make_server
    return ThreadedWSGIServer(
  File "C:\Users\donal.browne\venv\lib\site-packages\werkzeug\serving.py", line 688, in __init__
    super().__init__(server_address, handler)  # type: ignore
  File "C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\lib\socketserver.py", line 452, in __init__
    self.server_bind()
  File "C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\lib\http\server.py", line 138, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\lib\socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissions
```

Same error after running the below

```
(Venv) PS C:\Users\donal.browne> deactivate
>> set FLASK_APP=hello.py
>> venv\scripts\activate
>> flask run
```

`python -m flask run`

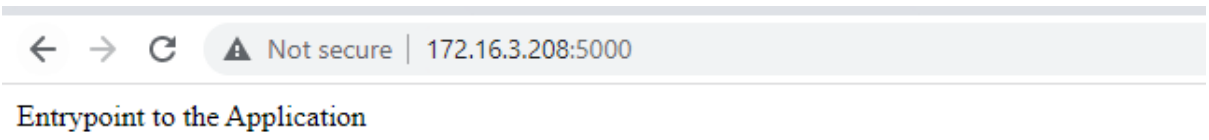
I tried changing to 5050, 8090 and 7000 – none of them worked.

I ran `Get-NetTcpConnection` to see if the ports were in use but they weren't.

I then switched directory and ran the following command and got the app working with the following command

```
(venv) PS C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\Lib> python hello.py
runserver 127.0.0.1:5050
```

Screenshot of app running on the following URL:

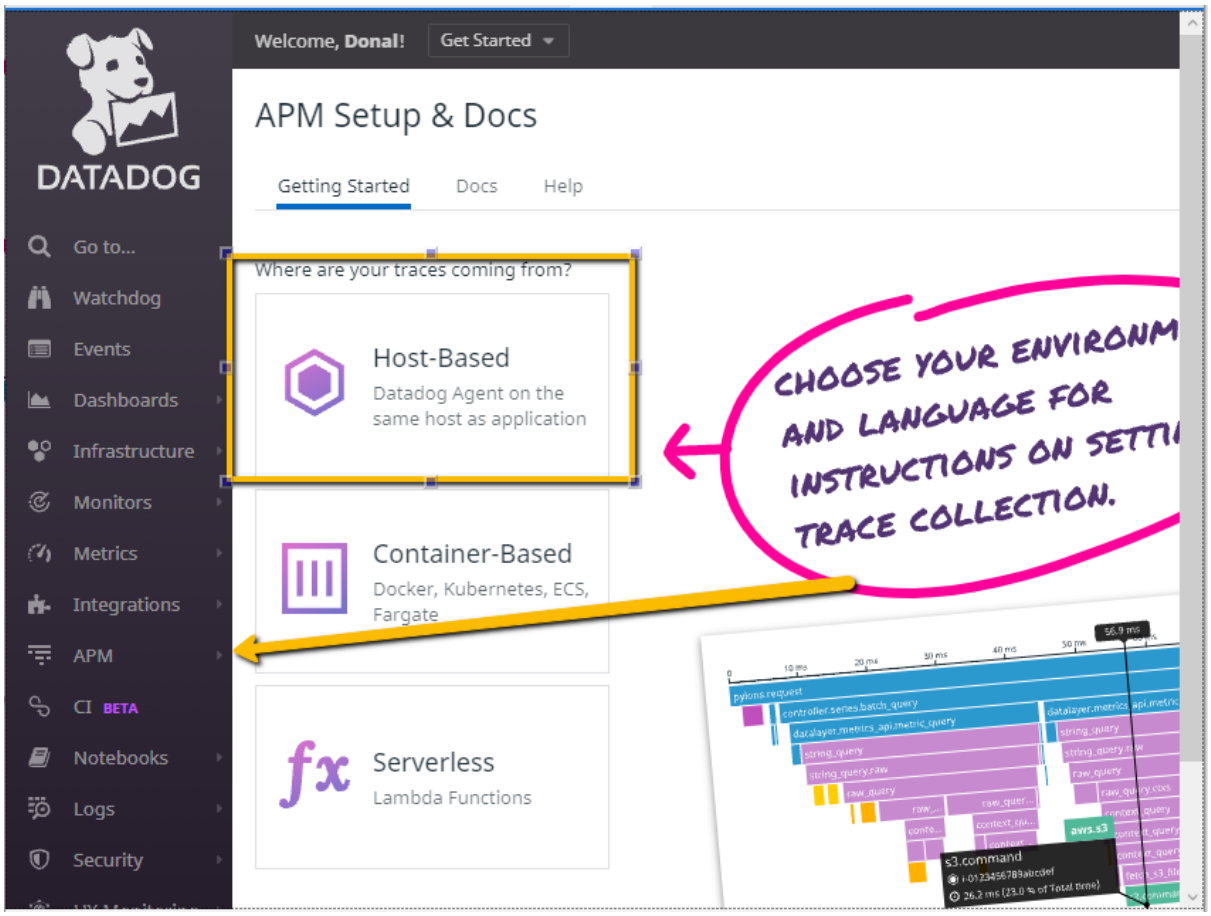


Entrypoint to the Application

Now that the app is up and running we can work on the APM.

Go to APM > Introduction > Select ‘Get Started’

From here I will select ‘Host Based’



When we select that we can choose our language, which will be python.

Run

```
pip install ddtrace
```

From an admin powershell window.

```
  ERROR: Failed building wheel for ddtrace
Failed to build ddtrace
ERROR: Could not build wheels for ddtrace, which is required to install pyproject.toml-based projects
```

Ran

```
pip install --upgrade pip setuptools wheel
```

I upgraded pip to be sure:

```
pip install --upgrade pip
```

```
PS C:\Windows\system32> pip install --upgrade pip setuptools wheel
Requirement already satisfied: pip in c:\users\donal.browne\appdata\local\programs\python\python310\lib\site-packages (21.3)
Requirement already satisfied: setuptools in c:\users\donal.browne\appdata\local\programs\python\python310\lib\site-packages (57.4.0)
Collecting setuptools
  Using cached setuptools-58.2.0-py3-none-any.whl (946 kB)
Collecting wheel
  Using cached wheel-0.37.0-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel, setuptools
  Attempting uninstall: setuptools
    Found existing installation: setuptools 57.4.0
    Uninstalling setuptools-57.4.0:
      Successfully uninstalled setuptools-57.4.0
Successfully installed setuptools-58.2.0 wheel-0.37.0
PS C:\Windows\system32>
```

I the requirements which are okay – I'm on python 3.5+

APM is enabled.

Read back over the error

```
building 'ddtrace.profiling.collector._memalloc' extension
  error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build
Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
```

I downloaded the relevant file from this link and ran

https://docs.microsoft.com/en-us/answers/questions/136595/error-microsoft-visual-c-140-or-greater-is-require.html

```
PS C:\Users\donal.browne> pip install --upgrade setuptools
Requirement already satisfied: setuptools in c:\users\donal.browne\appdata\local\programs\python\python310\lib\site-packages (58.2.0)
PS C:\Users\donal.browne> pip install ddtrace
Collecting ddtrace
  Using cached ddtrace-0.54.1.tar.gz (1.4 MB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing wheel metadata (pyproject.toml) ... done
Collecting tenacity>=5
  Using cached tenacity-8.0.1-py3-none-any.whl (24 kB)
Requirement already satisfied: six>=1.12.0 in c:\users\donal.browne\appdata\local\programs\python\python310\lib\site-packages (from ddtrace) (1.16.0)
Collecting attrs>=19.2.0
  Using cached attrs-21.2.0-py2.py3-none-any.whl (53 kB)
Collecting protobuf>=3
  Using cached protobuf-3.18.1-py2.py3-none-any.whl (174 kB)
Collecting packaging>=17.1
  Using cached packaging-21.0-py3-none-any.whl (40 kB)
Collecting pyparsing>=2.0.2
  Using cached pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
Building wheels for collected packages: ddtrace
  Building wheel for ddtrace (pyproject.toml) ... done
  Created wheel for ddtrace: filename=ddtrace-0.54.1-cp310-cp310-win_amd64.whl size=855962 sha256=0b6a7a6f671db1f37e1be3800173bb1761319bccea123dc674704b62aa4e8481
  Stored in directory: c:\users\donal.browne\appdata\local\pip\cache\wheels\2e\09\01\c97da5a741711d0e74f50db670c172e1baf43eda43cca5644a
Successfully built ddtrace
Installing collected packages: pyparsing, tenacity, protobuf, packaging, attrs, ddtrace
Successfully installed attrs-21.2.0 ddtrace-0.54.1 packaging-21.0 protobuf-3.18.1 pyparsing-2.4.7 tenacity-8.0.1
```

Ran ddtrace with added variables and configuration and it is now up and running

```
usage: ddtrace-run <your usual python command>
PS C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\Lib> ddtrace-run python hello.py DD_SERVICE="hello" DD_ENV="hellostage" DD_LOGS_INJECTION=true DD_RUNTIME_METRICS_ENABLED=true
PS C:\Users\donal.browne\AppData\Local\Programs\Python\Python310\Lib>  * Serving Flask app 'hello' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
WARNING:werkzeug: * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
2021-10-19 19:41:59,129 - werkzeug - WARNING -  * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
INFO:werkzeug: * Running on http://192.168.1.21:5000/ (Press CTRL+C to quit)
2021-10-19 19:41:59,130 - werkzeug - INFO -  * Running on http://192.168.1.21:5000/ (Press CTRL+C to quit)
```

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

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

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

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

Question 6: 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?

https://github.com/DataDog/trace-examples/blob/master/python/pymongo/demo.py

An Iothon in Dingle – Big tourist industry.
Sensors on the road, just counts of people coming in. As a proud dingle man you could use it to built up the industry. Make a case for the government to spend money in Dingle because we would have stats of people coming in and out.
A network could be set up?

Electric Cars – Work with tesla Ireland to decipher where to put the next electric chargers.

Diabetes: Weekly/Monthly stats.

Magic seaweed – Picking the correct waves for you with metrics of wind, swell, tide etc. The biggest waves may not be the best so you could get notified.
Also Magic seaweed would use this because the different beaches have better waves depending on different metrics. i.e wine strand has the best waves 2 hours before high tide.