**Collecting Metrics:**

* 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.

Tags can be defined on the config file (datadog.yaml), by adding the folwing lines:

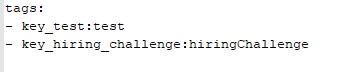
tags:

- key1:value1

- key2:value2

In our example the tags are: “test” and “hiringChallenge”.

So the tags on the configuration file looks like below.



And then the host on the hostmap will display the tags as below:

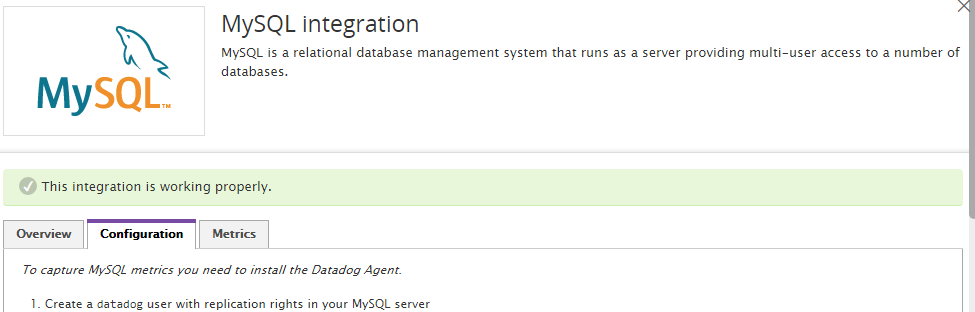


Please note that to update tags the system needs around 15minutes.

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

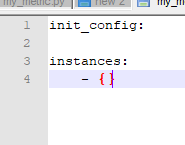
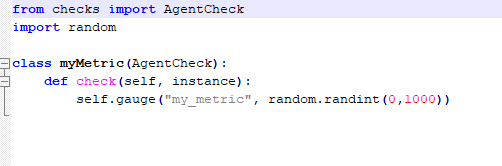
The integrations are done on the “Integrations” menu, so all we need to do is search for the needed integration and follow the installation steps as described on the “Configuration” tab.

Below a screenshot of the integration screen for MySQL

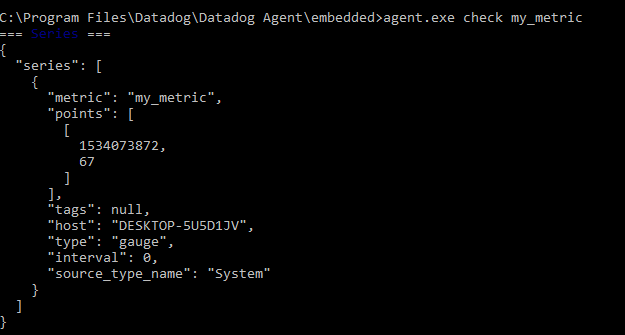


* Create a custom Agent check that submits a metric named my\_metric with a random value between 0 and 1000.

Below the python script and the YAML configuration for the custom metric (note both files must have the same name and placed respectively on the check.d and conf.d folder).

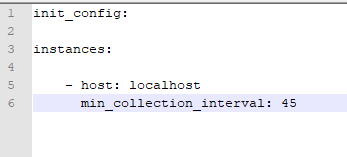


To test our custom metric, we run the check on the command line using the agent as below:



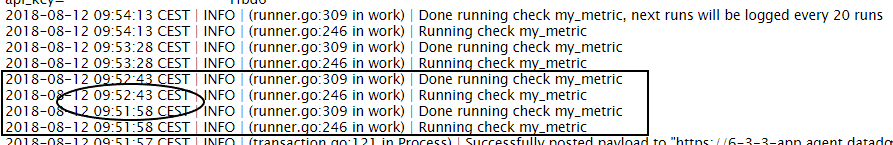
* Change your check's collection interval so that it only submits the metric once every 45 seconds.

The checks interval is configurable by instance on the YAML config file. So all we need to do is add an instance on the YAML file and define a collection interval for it.



Let’s try our check interval, by checking the logs and see if the interval is 45 second.

The screenshot below shows the interval between two metric runs.

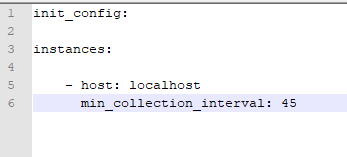


Note that using the version6 the intervals are defined at instance level.

The collector runs every 15-20 seconds depending on how much integration are enabled. If the interval on this Agent happens to be every 20 seconds, then the Agent collects and includes the Agent check. The next time it collects 20 seconds later, it sees that 20 is less than 45 and doesn’t collect the custom Agent check. The second time it sees that the time since last run was 40 which is still less than 45 and doesn’t collect the custom agent check again. But at the next one it sees that the time since last run was 60 which is greater than 45 and therefore the Agent check is collected.

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

Yes, the change is done on the configuration YAML file as shown on the screen below:



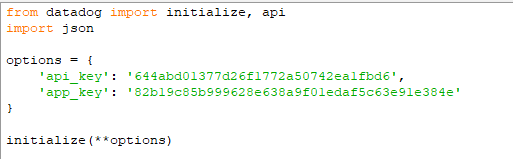
**Visualizing Data:**

Utilize the Datadog API to create a Timeboard that contains:

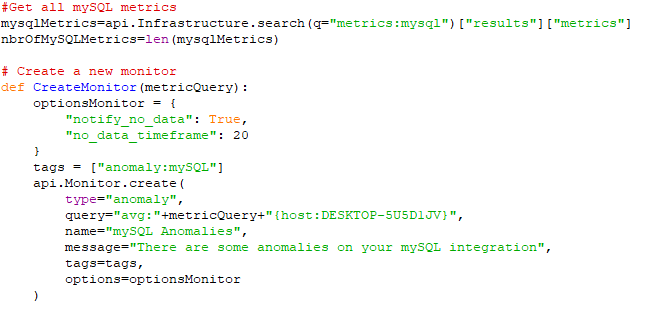
* Your custom metric scoped over your host.
* Any metric from the Integration on your Database with the anomaly function applied.
* Your custom metric with the rollup function applied to sum up all the points for the past hour into one bucket

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

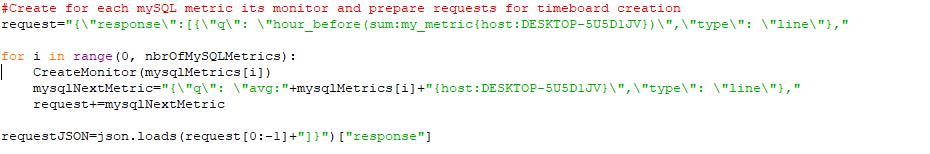
First of all we initialize the API settings



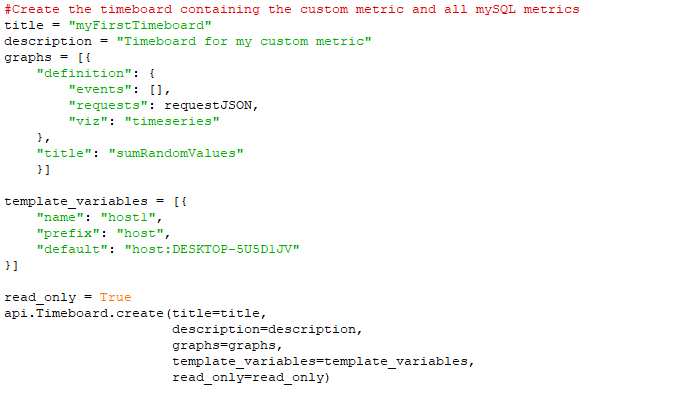
Then we get the integration (MySQL) metrics and we create the anomalies monitor creator using the API calls.



After that, we prepare the request for the API call to create the time board, which is a JSON structure.



Now that we have all settings ready, we call the API to create the time board

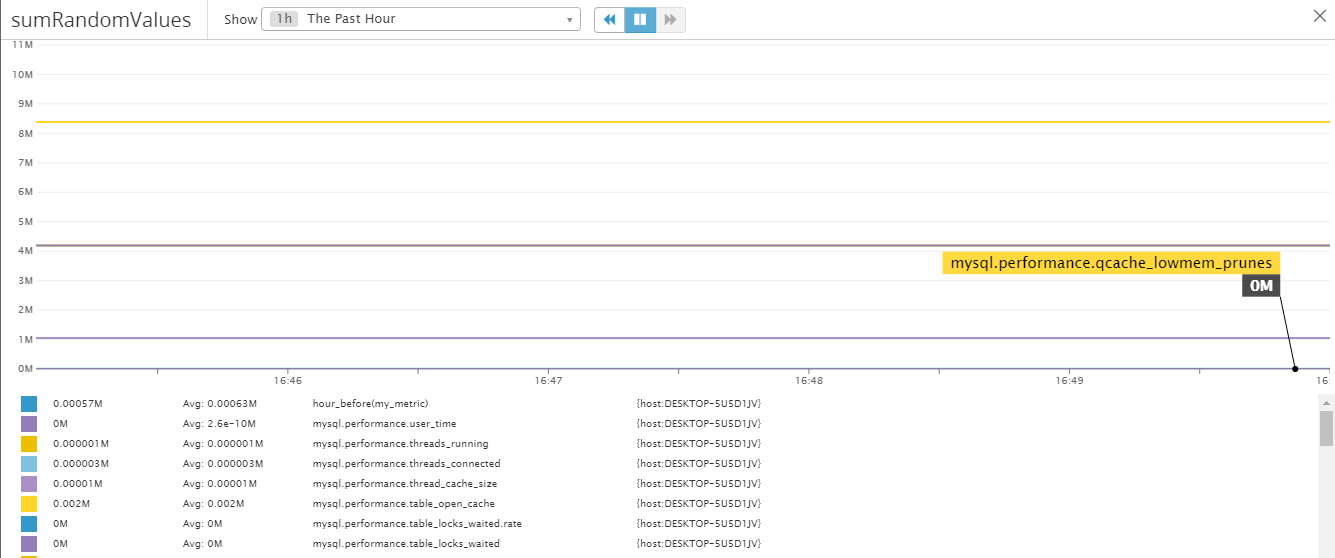


Once this is created, access the Dashboard from your Dashboard List in the UI:

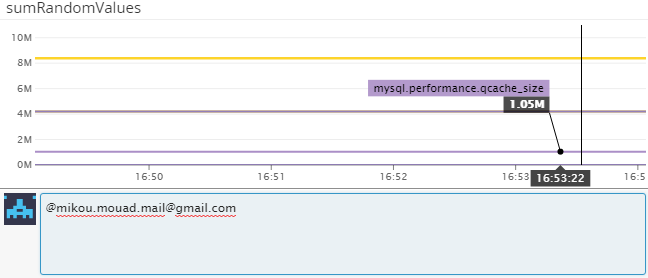
* Set the Timeboard's timeframe to the past 5 minutes
* Take a snapshot of this graph and use the @ notation to send it to yourself.

The graph contains around 62 metric so it could be little confusing, but the graph gives the ability to focus on a specific timeline interval which makes it much easier.

The graph for the 5 minutes was as below:



To send it as an email, just click on the graph and then the options will be displayed with the “Annotation” option. Once clicked, the blue textarea will be displayed and we will be able to type “@” and select one of the users.

****

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

In this example the anomaly graph doesn’t display an interesting thing as there is no activity in MySQL. But basically the anomaly graph should compare the new entries with the average of the old ones, and then if they are different, the new entry should be highlighted.

**Monitoring Data**

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

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

* Warning threshold of 500
* Alerting threshold of 800
* And also ensure that it will notify you if there is No Data for this query over the past 10m.

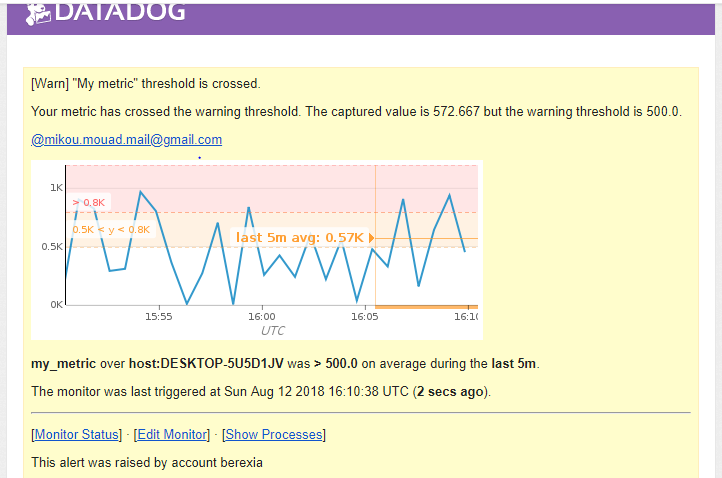
Please configure the monitor’s message so that it will:

* Send you an email whenever the monitor triggers.
* Create different messages based on whether the monitor is in an Alert, Warning, or No Data state.
* Include the metric value that caused the monitor to trigger and host ip when the Monitor triggers an Alert state.
* When this monitor sends you an email notification, take a screenshot of the email that it sends you.

The monitor configurations are done on the monitor creation screen, accessible from the “New Monitor” menu.

After defining the configurations on the screen, we should wait for a while, as our monitor will alert us when the average of values for the last 5 minutes crosses a threshold.

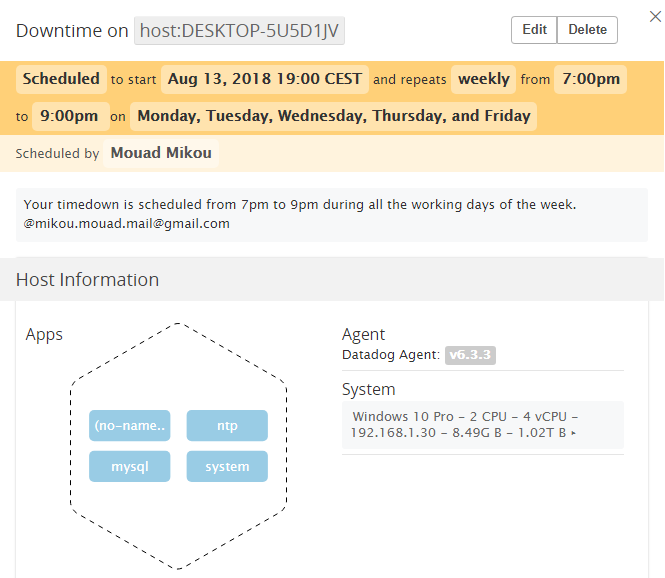
The email received was as below:



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

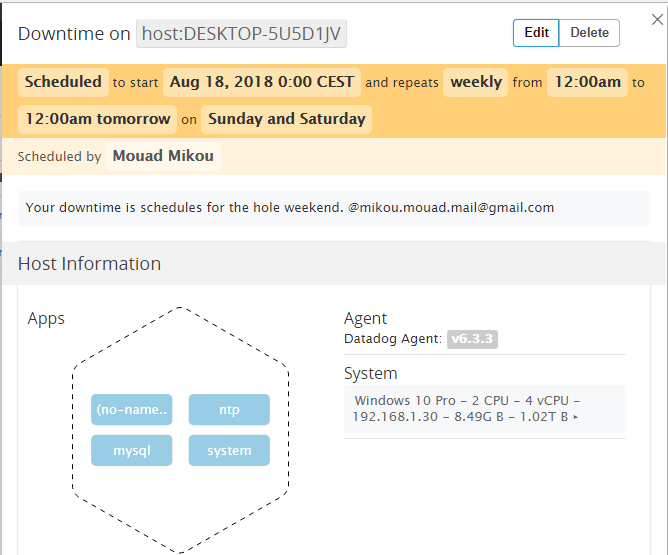
To set a downtime, we get to the “Manage downtime” menu and we define the downtime configurations.

The working days downtime then look like below:



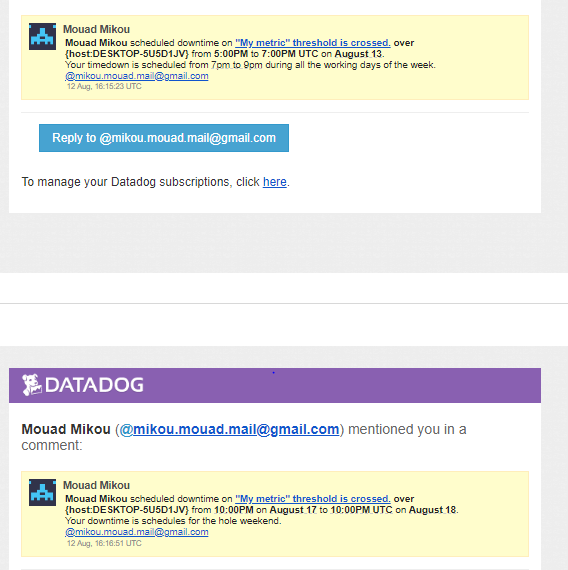
* + And one that silences it all day on Sat-Sun.

And the week-end downtime looks like below:



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

The same way we define the annotation while configuring the downtime and the email received will be as below:



**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')

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

The manual middleware insertion can be useful for customization and when the number of apps is small, but when managing a high number of apps, the best way is to use the ddtrace-run so it could be easily done automatically using automatic scripts.

In our case we use the manual solution, and the python code will be as below:



Then we will need to run the app and navigate to <http://localhost:5050/> (or any other link from the routing list) to generate the first flow to trace.

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

A service is the process or the set of processes that aims to provide some functionality to create a feature or an application. But the resource is the query that gets the service work well.

We can suppose a service that get as input two integers and return as output the sum.

In this case the service id the component that runs the process of summing the two inputs, and the resource is the query that provides the input and triggers the service process.

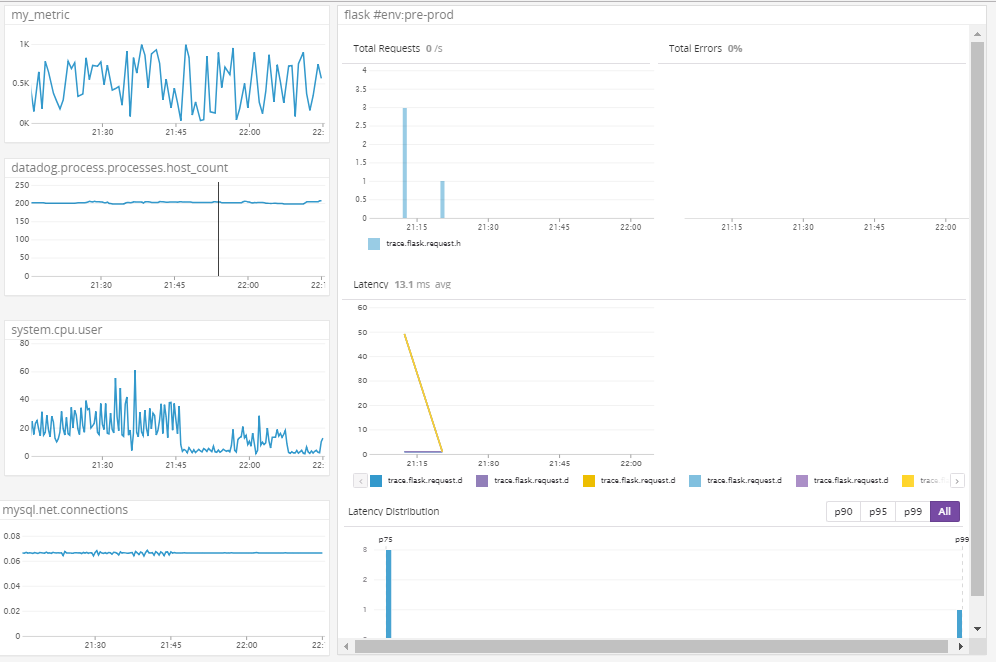
Provide a link and a screenshot of a Dashboard with both APM and Infrastructure Metrics.

After creating the APM tracer, we can now include it’s graphs into a dashboard.

Below the link to a dashboard, containing APM graphs and some other infrastructure and integration metrics.

https://p.datadoghq.com/sb/20530818125a55caf25ebcac366903d70ee5f99e7

The dashboard will be as below:



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

**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?

During my last internship I worked on the Software Defined Networks which is a technology that helps networks orchestration and centralization. I would like to find a way to merge these two technologies to get a more centralized management for the SDN concept.