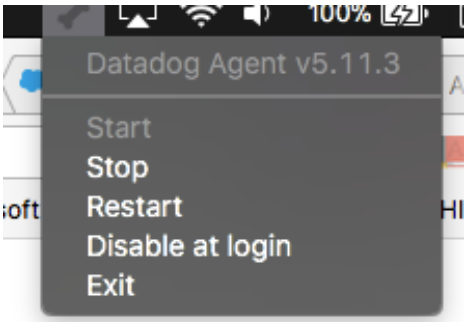
Lloyd Soldatt
DataDog, Sales Engineering, London

LSoldatt@Gmail.com

+44 7799 213030

Prerequisites – Set Up the Environment

- DataDog Agent was installed on MacOS 10.13
-



- Environment was configured and is up and running

Output of current state with DD 'info' command

SLDTTMacBookAir:~ prism$ /usr/local/bin/datadog-agent **info**
==========
Collector (v 5.11.3)
==========

Status date: 2018-01-29 10:05:11 (5s ago)
Pid: 25405
Platform: Darwin-17.4.0-x86_64-i386-64bit
Python Version: 2.7.12, 64bit
Logs: <stderr>, /var/log/datadog/collector.log, syslog:/var/run/syslog

Clocks
======

NTP offset: Unknown (No response received from 1.datadog.pool.ntp.org.)
System UTC time: 2018-01-29 10:05:17.768102

Paths
=====

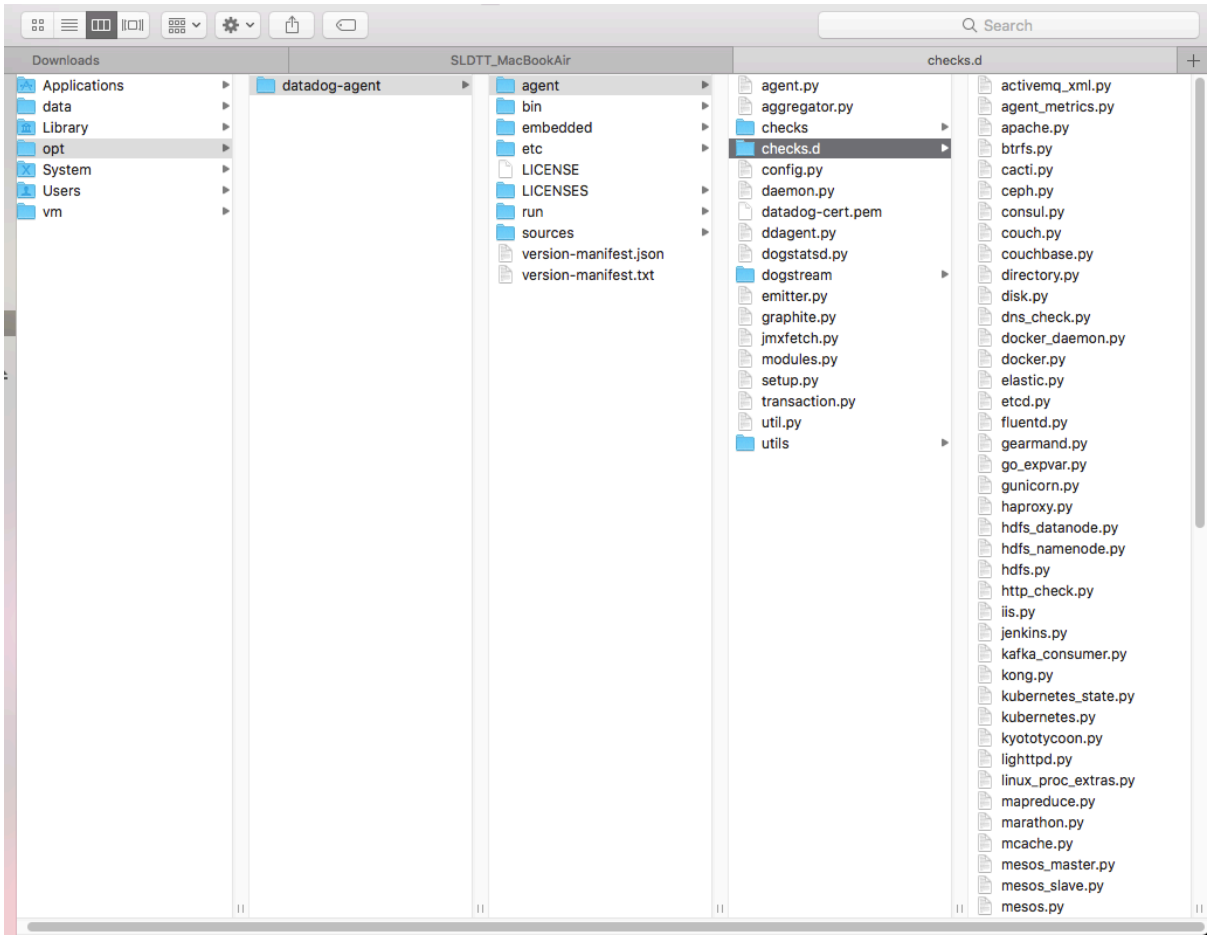
```
    conf.d: /opt/datadog-agent/etc/conf.d
    checks.d: /opt/datadog-agent/agent/checks.d

  Hostnames
  =========

    No host information available yet.

  Checks
  ======

    No checks have run yet.

  Emitters
  ========

    No emitters have run yet.

====================
Dogstatsd (v 5.11.3)
====================

  Status date: 2018-01-29 10:05:11 (6s ago)
  Pid: 25403
  Platform: Darwin-17.4.0-x86_64-i386-64bit
  Python Version: 2.7.12, 64bit
  Logs: <stderr>, /var/log/datadog/dogstatsd.log, syslog:/var/run/syslog

  Flush count: 0
  Packet Count: 0
  Packets per second: 0
  Metric count: 0
  Event count: 0
  Service check count: 0

====================
Forwarder (v 5.11.3)
====================

  Status date: 2018-01-29 10:05:16 (2s ago)
  Pid: 25404
  Platform: Darwin-17.4.0-x86_64-i386-64bit
  Python Version: 2.7.12, 64bit
  Logs: <stderr>, /var/log/datadog/forwarder.log, syslog:/var/run/syslog

  Queue Size: 0 bytes
```

```
  Queue Length: 0
  Flush Count: 1
  Transactions received: 0
  Transactions flushed: 0
  Transactions rejected: 0
  API Key Status: API Key is valid


SLDTTMacBookAir:~ prism$ /usr/local/bin/datadog-agent info
====================
Collector (v 5.11.3)
====================

  Status date: 2018-01-29 10:05:47 (19s ago)
  Pid: 25405
  Platform: Darwin-17.4.0-x86_64-i386-64bit
  Python Version: 2.7.12, 64bit
  Logs: <stderr>, /var/log/datadog/collector.log, syslog:/var/run/syslog

  Clocks
  ======

    NTP offset: 0.0021 s
    System UTC time: 2018-01-29 10:06:06.963100

  Paths
  =====

    conf.d: /opt/datadog-agent/etc/conf.d
    checks.d: /opt/datadog-agent/agent/checks.d

  Hostnames
  =========

    socket-hostname: SLDTTMacBookAir.home
    hostname: SLDTTMacBookAir.home
    socket-fqdn: sldttmacbookair.home

  Checks
  ======

    ntp
    ---
      - Collected 0 metrics, 0 events & 0 service checks

    disk
    ----
```

```
    - instance #0 [OK]
    - Collected 32 metrics, 0 events & 0 service checks

  mongo
  -----
    - instance #0 [OK]
    - Collected 334 metrics, 0 events & 1 service check
    - Dependencies:
        - pymongo: 3.2

  network
  -------
    - instance #0 [OK]
    - Collected 24 metrics, 0 events & 0 service checks


  Emitters
  ========

    - http_emitter [OK]

====================
Dogstatsd (v 5.11.3)
====================

  Status date: 2018-01-29 10:06:01 (5s ago)
  Pid: 25403
  Platform: Darwin-17.4.0-x86_64-i386-64bit
  Python Version: 2.7.12, 64bit
  Logs: <stderr>, /var/log/datadog/dogstatsd.log, syslog:/var/run/syslog

  Flush count: 5
  Packet Count: 0
  Packets per second: 0.0
  Metric count: 1
  Event count: 0
  Service check count: 0

====================
Forwarder (v 5.11.3)
====================

  Status date: 2018-01-29 10:06:06 (1s ago)
  Pid: 25404
  Platform: Darwin-17.4.0-x86_64-i386-64bit
  Python Version: 2.7.12, 64bit
  Logs: <stderr>, /var/log/datadog/forwarder.log, syslog:/var/run/syslog
```

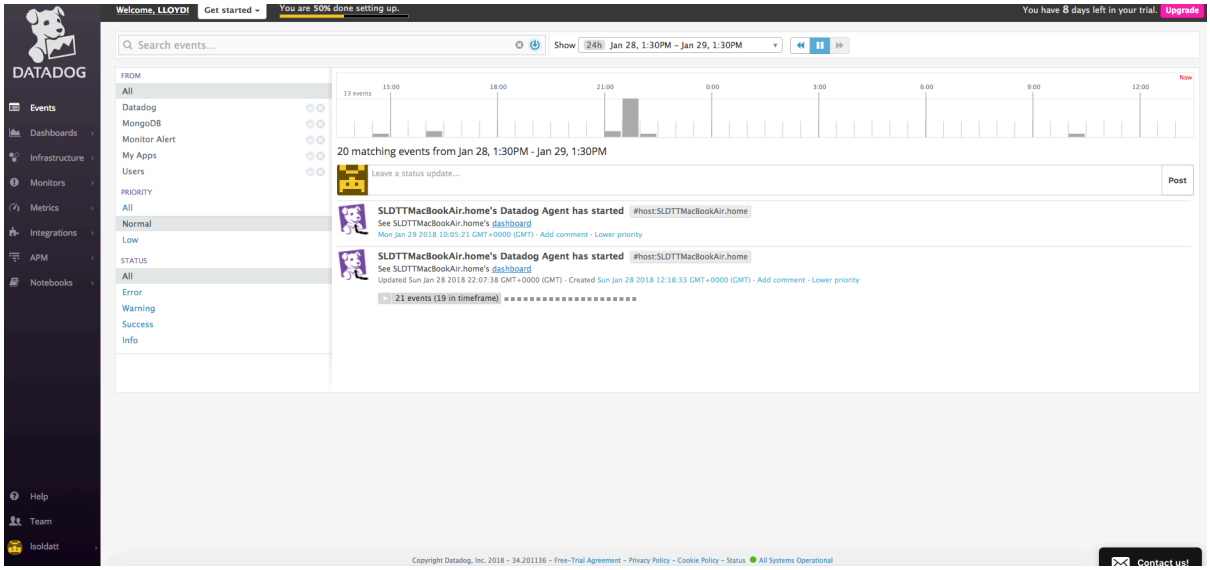
Queue Size: 0 bytes
Queue Length: 0
Flush Count: 17
Transactions received: 6
Transactions flushed: 6
Transactions rejected: 0
API Key Status: API Key is valid

- DD Agent ‘checks.d’ and ‘conf.d’ folders, essential for the configuration of **Integrations** and bespoke **Checks**

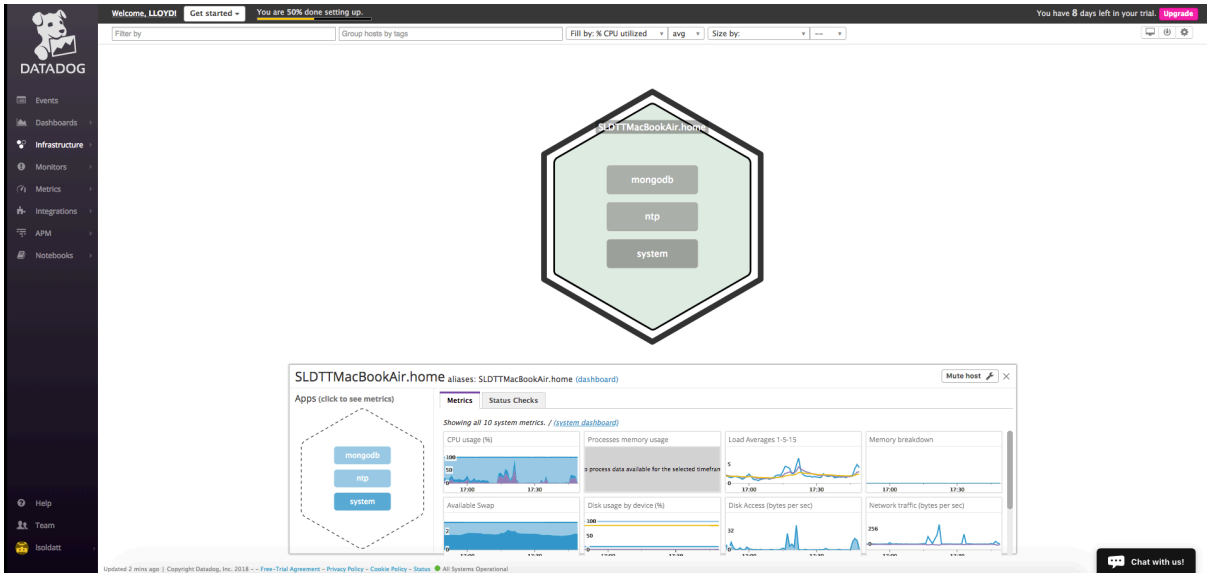


Collecting Metrics

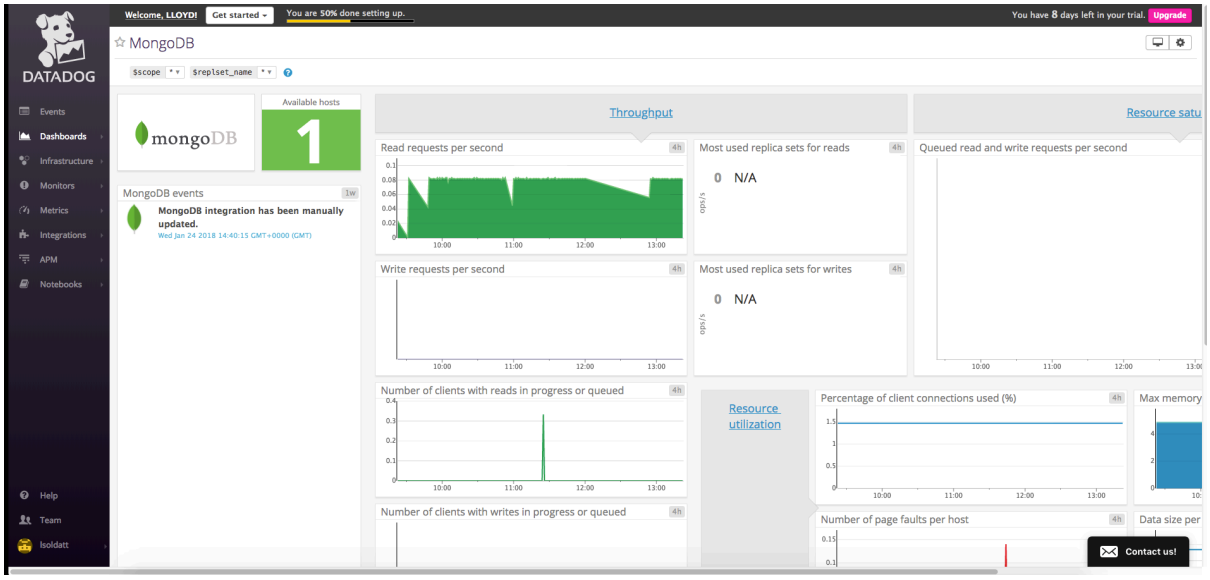
- MongoDB ‘admin’ DB instance v3.6.2 installed on the system
- MongoDB ‘Admin’ User configured within the ‘admin’ instance
- DD User configured within this Mongo DB instance
- My DD online portal – connected to LSoldatt@Gmail.com



- **Host / MongoDB / NTP map of my Infrastructure**

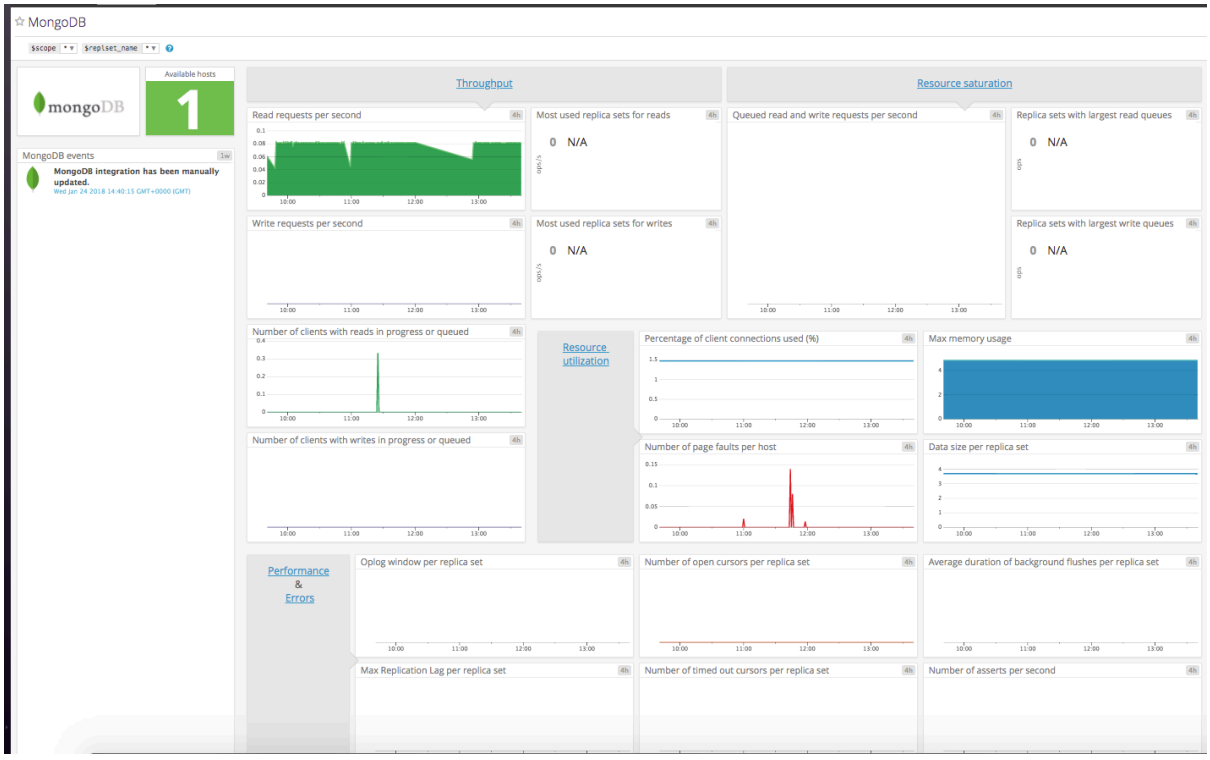


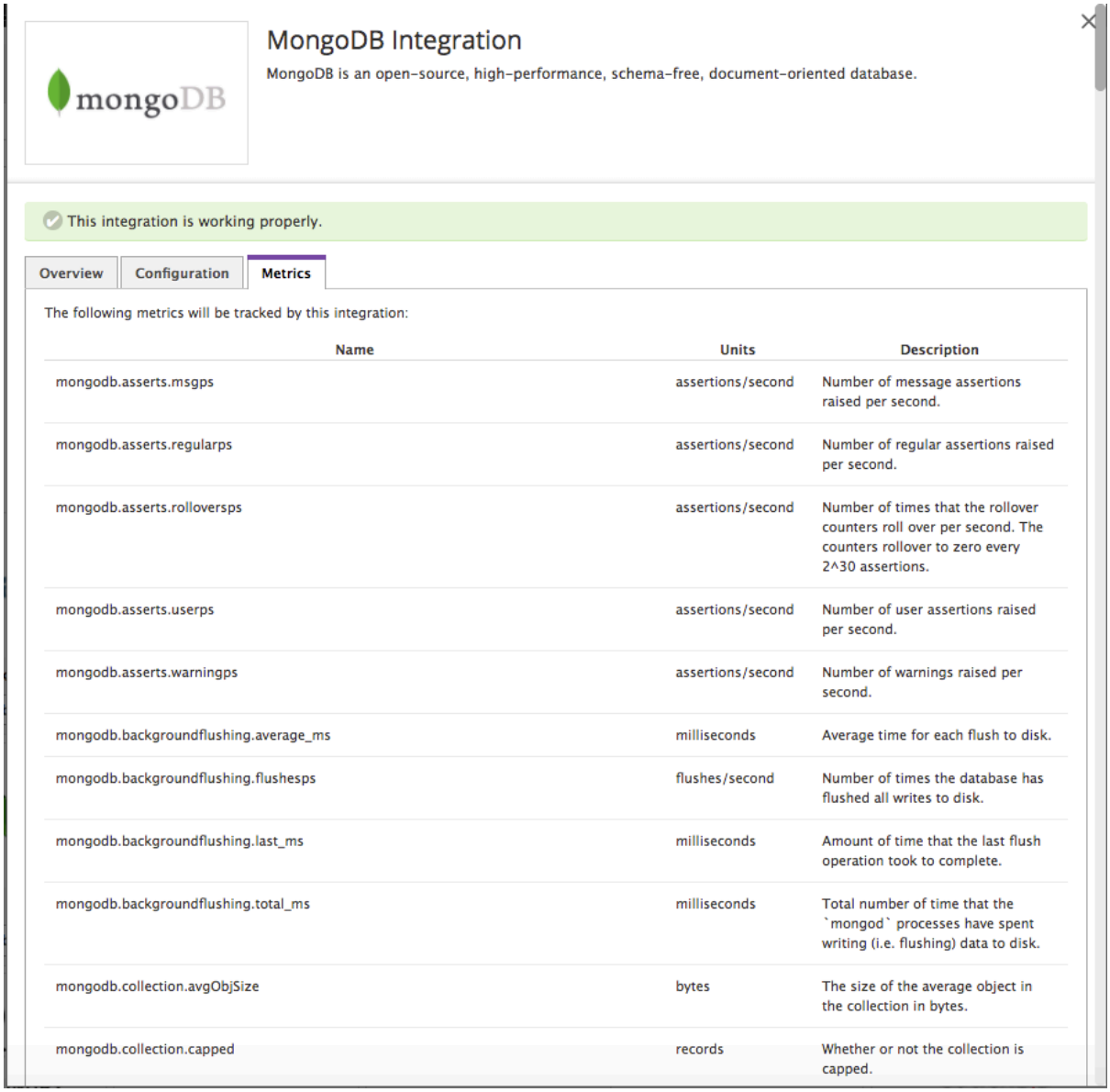
- **Mongo DB V3.62 Integration**



- **Collecting all standard and optional MongoDB metrics with 1 valid instance ('admin') connected to DD Dashboard**

-





MongoDB Integration

MongoDB is an open-source, high-performance, schema-free, document-oriented database.

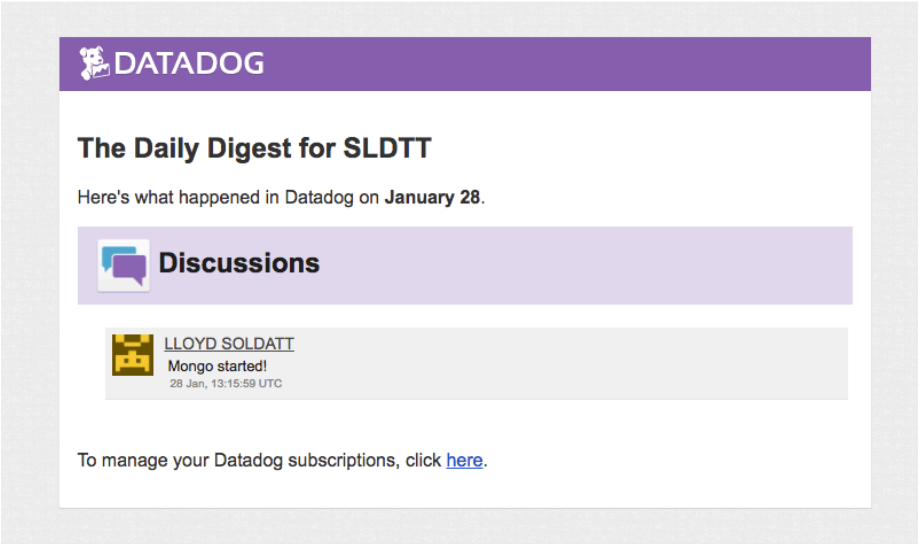
This integration is working properly.

Overview | Configuration | Metrics

The following metrics will be tracked by this integration:

Name	Units	Description
mongodb.asserts.msgps	assertions/second	Number of message assertions raised per second.
mongodb.asserts.regularps	assertions/second	Number of regular assertions raised per second.
mongodb.asserts.rolloversps	assertions/second	Number of times that the rollover counters roll over per second. The counters rollover to zero every 2^30 assertions.
mongodb.asserts.userps	assertions/second	Number of user assertions raised per second.
mongodb.asserts.warningps	assertions/second	Number of warnings raised per second.
mongodb.backgroundflushing.average_ms	milliseconds	Average time for each flush to disk.
mongodb.backgroundflushing.flushesps	flushes/second	Number of times the database has flushed all writes to disk.
mongodb.backgroundflushing.last_ms	milliseconds	Amount of time that the last flush operation took to complete.
mongodb.backgroundflushing.total_ms	milliseconds	Total number of time that the `mongod` processes have spent writing (i.e. flushing) data to disk.
mongodb.collection.avgObjSize	bytes	The size of the average object in the collection in bytes.
mongodb.collection.capped	records	Whether or not the collection is capped.

- Post within my DD online dashboard



DATADOG

The Daily Digest for SLDTT

Here's what happened in Datadog on **January 28**.

Discussions

LLOYD SOLDATT
Mongo started!
28 Jan, 13:15:59 UTC

To manage your Datadog subscriptions, click here.

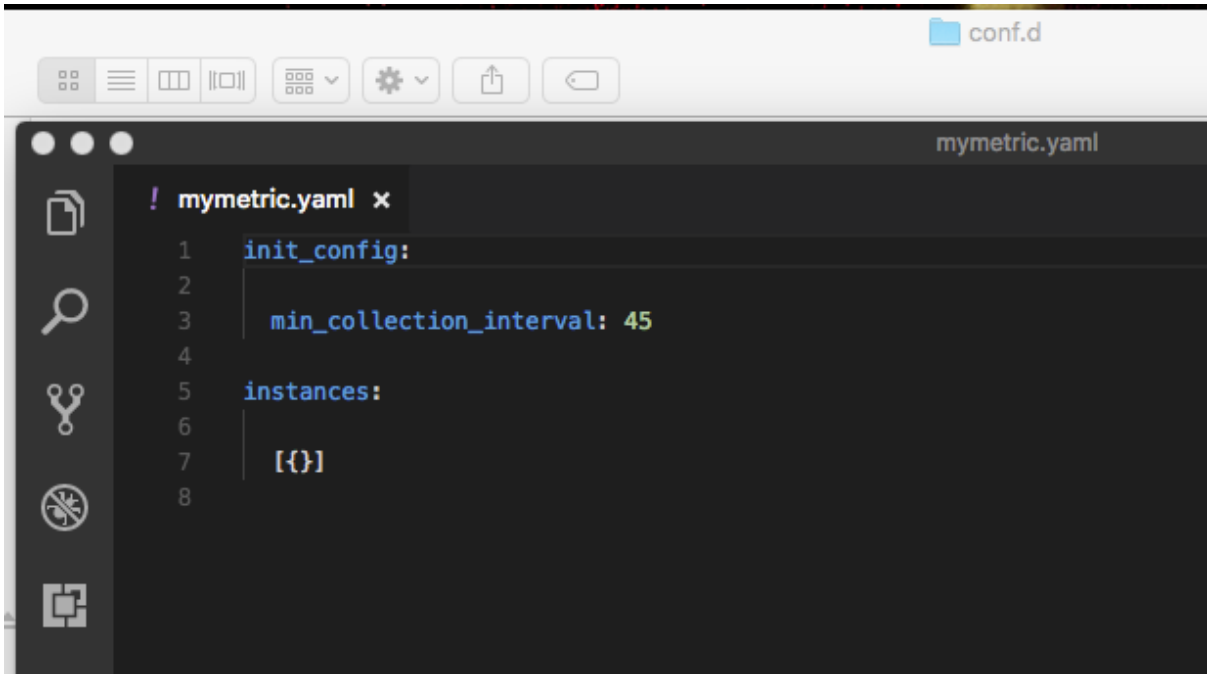
- Created the custom check ‘my_metric’ both in Python and in the corresponding YAML

mymetric.yaml config file source code

(tabs are not used in this YAML, just spaces for indentation for parsing)

mymetric.yaml

```
init_config:

  min_collection_interval: 45

instances:

  [{}]
```

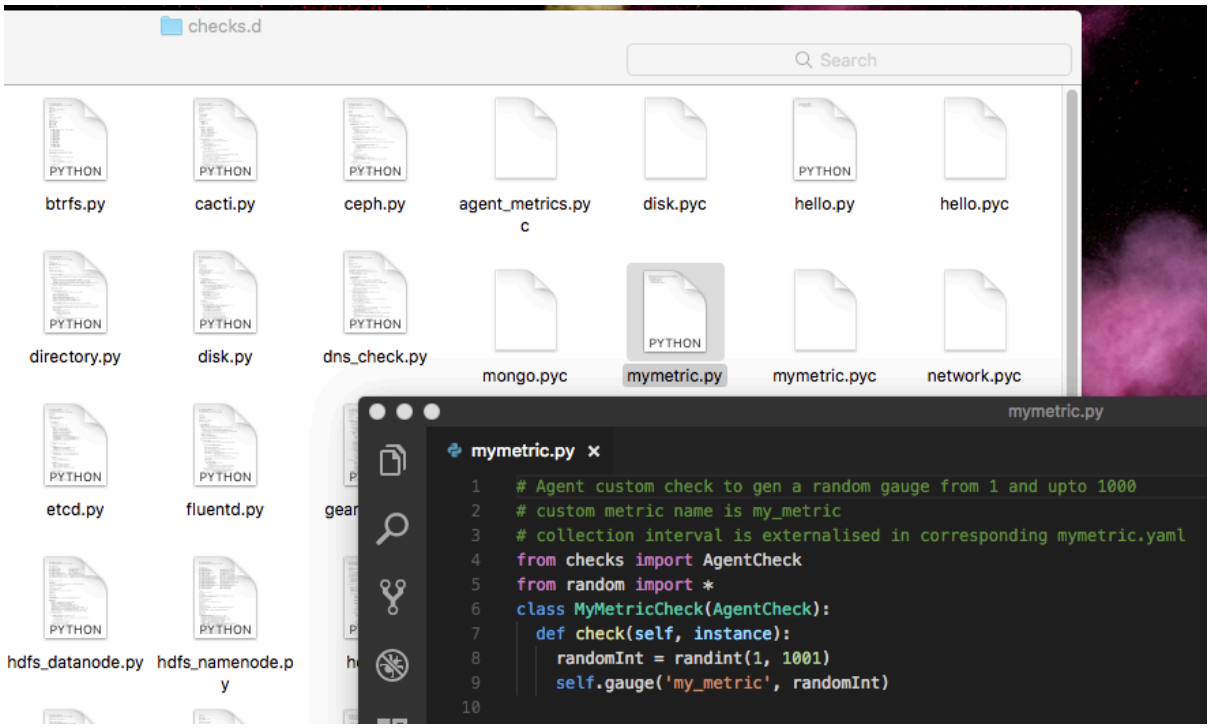


mymetric.py source code

mymetric.py

```
from checks import AgentCheck
from random import *
class MyMetricCheck(AgentCheck):
  def check(self, instance):
    randomInt = randint(1, 1001)
```

self.gauge('my_metric',randomInt)



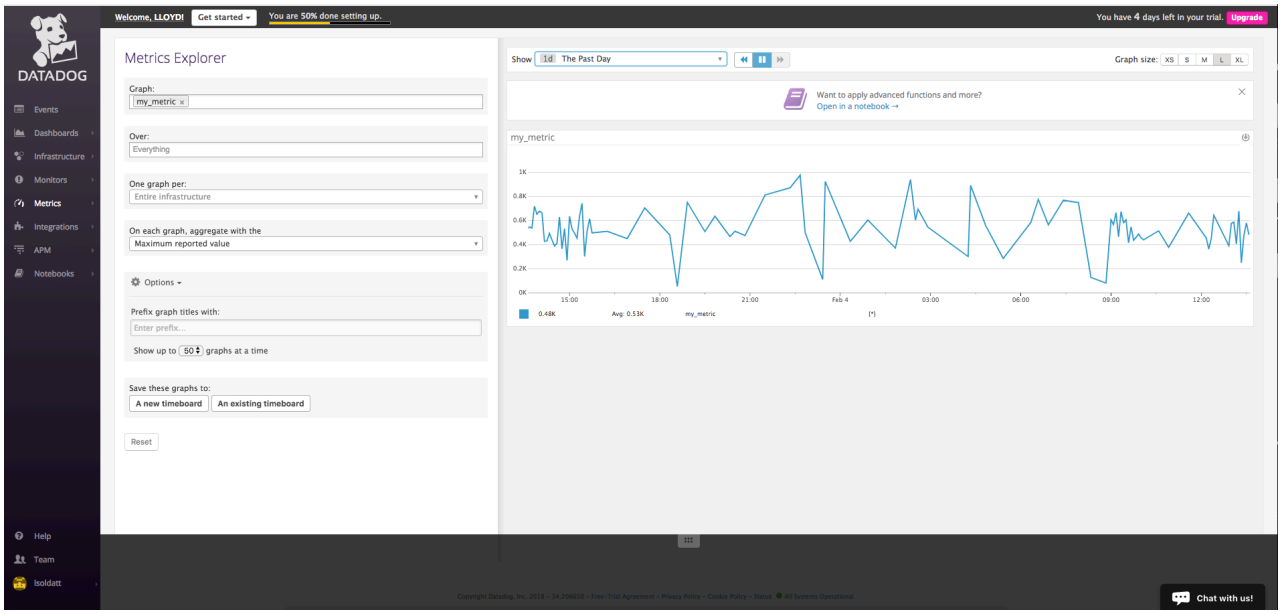
Test for mymetric.py for 'my_metric' as type gauge with random number output every 45sec

SLDTTMacBookAir:~ prism$ sudo -u prism datadog-agent check mymetric
2018-02-02 21:01:50,961 | INFO | dd.collector | config(config.py:1061) | initialized checks.d checks: ['mymetric', 'network', 'ntp', 'disk', 'hello', 'mongo']
2018-02-02 21:01:50,964 | INFO | dd.collector | config(config.py:1062) | initialization failed checks.d checks: []
2018-02-02 21:01:50,965 | INFO | dd.collector | checks.collector(collector.py:542) | Running check mymetric
Metrics:
[('my_metric',
1517605310,
831,
{'hostname': 'SLDTTMacBookAir.home', 'type': 'gauge'})]
Events:
[]
Service Checks:
[]
Service Metadata:
[{}]
mymetric

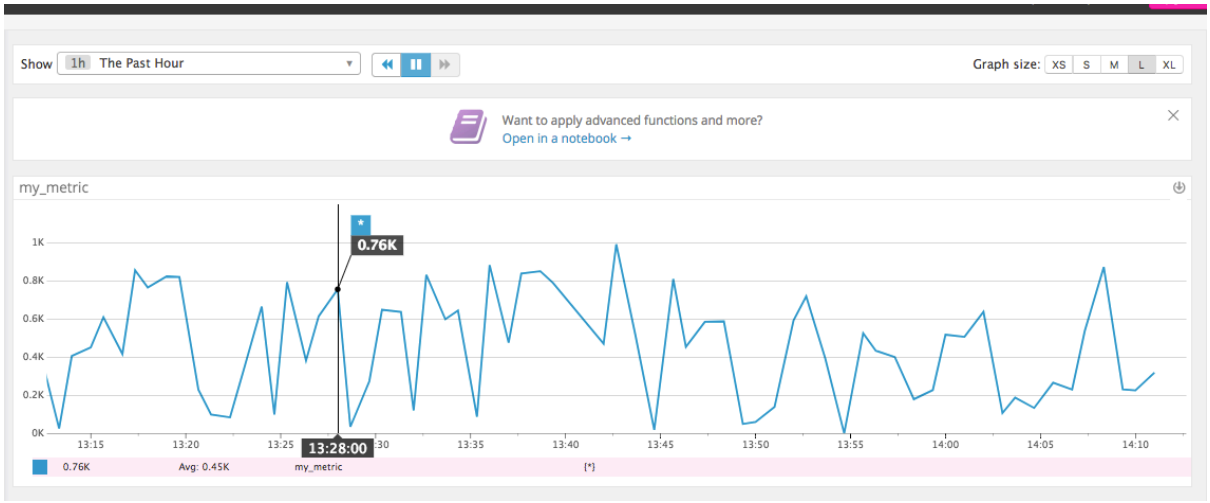
- instance #0 [OK]

- Collected 1 metric, 0 events & 0 service checks

my_metric view on my DD Cloud account



For the past hour:

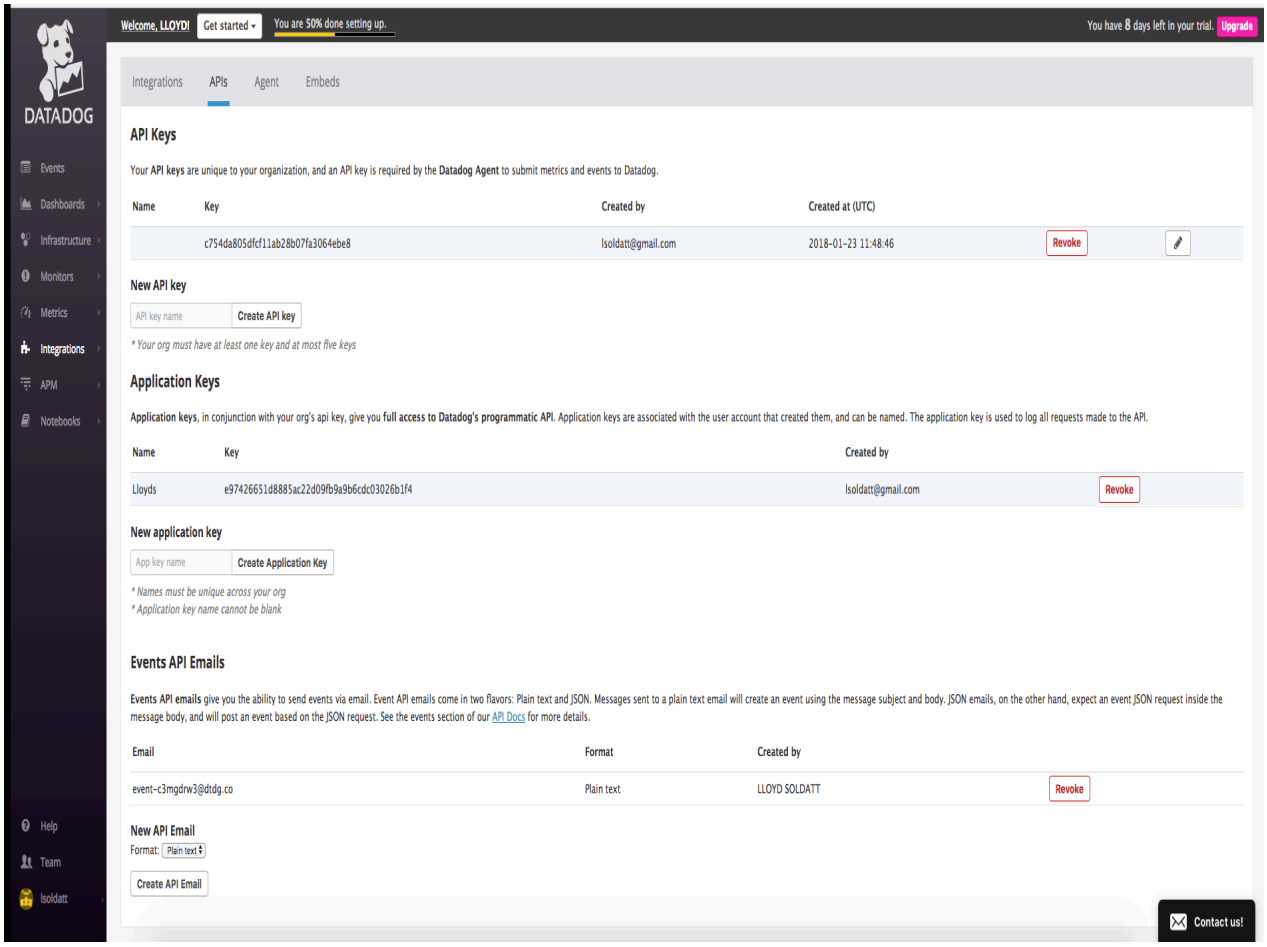


Bonus Question:

- **Yes, externalising the global parameter 'min collection interval', which defaults to 0, into the corresponding YAML file, will allow to update the YAML file without touching the PY class definition Agent Check, itself. Thus, the interval can be set and updated in the corresponding YAML, as it must be. In fact, this is the best practice way of creating a controllable class in PY/Java, all key parameters need to be read in from an external YAML/interface/config files.**

Visualizing Data

- Allocated both API and APP keys for my Timeboard script



- Created the following Python script to create the Timeboard via DD API

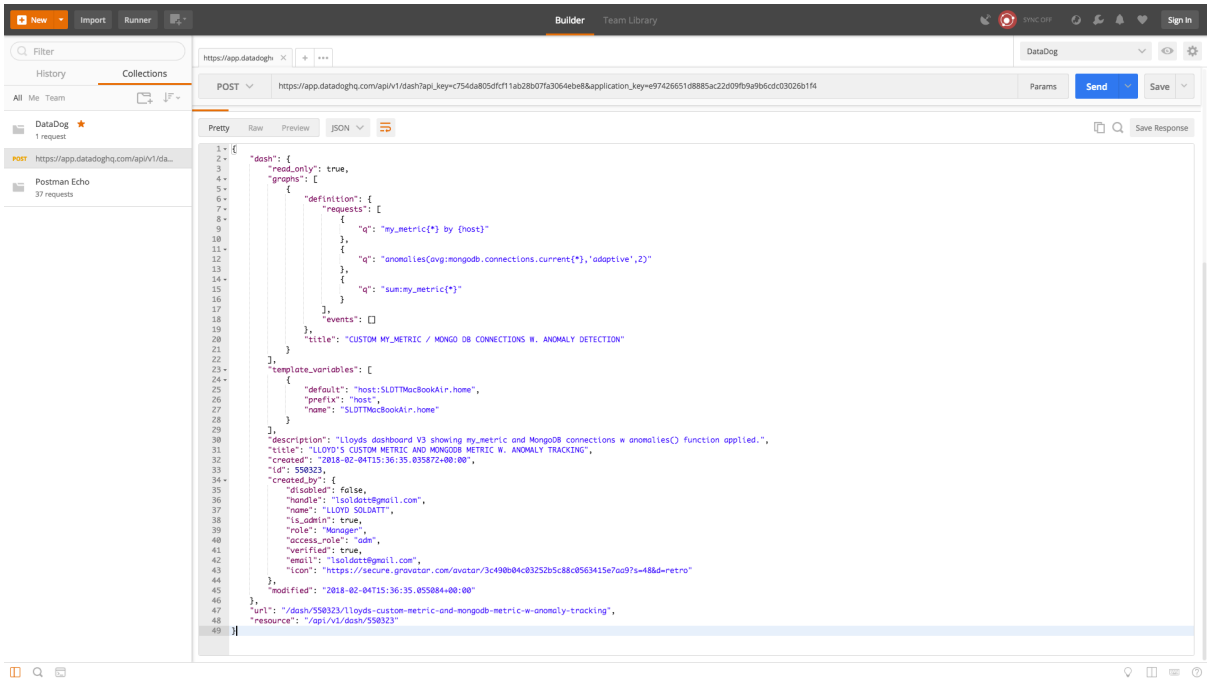
```
from datadog import initialize, api

options = {
   'api_key': 'c754da805dfcf11ab28b07fa3064ebe8',
   'app_key': 'e97426651d8885ac22d09fb9a9b6cdc03026b1f4'
}

initialize(**options)

title = "Lloyd's Timeboard"
```

```
description = "Timeboard for MongoDB / MacOS."
graphs = [{
   "definition": {
      "events": [],
      "requests": [
            {"q": "my_metric{*} by {host}"},
            {"q": "anomalies(avg:mongodb.connections.current{*},'adaptive',2)"},
            {"q": "sum:my_metric{*}"}
      ],
      "viz": "timeseries"
   },
   "title": "My Custom Metric & MongoDB Connections View"
}]

template_variables = [{
   "name": "SLDTTMacBookAir.home",
   "prefix": "host",
   "default": "host:my-host"
}]

read_only = True
api.Timeboard.create(title=title,
                     description=description,
                     graphs=graphs,
                     template_variables=template_variables,
                     read_only=read_only)
```

Timeboard with Postman

Postman is a technology partner with DD and as such they integrate natively with DD web portal, also providing an ability for me to bypass the environment constraints and execute JSON scripts with DD POST API requests.

Postman Set-Up



Note: request type set to POST, pointing to the DD Dash API endpoint. Both my account's allocated API Key and APP Key are set as DD environment pre-sets and are constructed into the post API call/URL.

My script sent in the body of the Timeboard POST request:

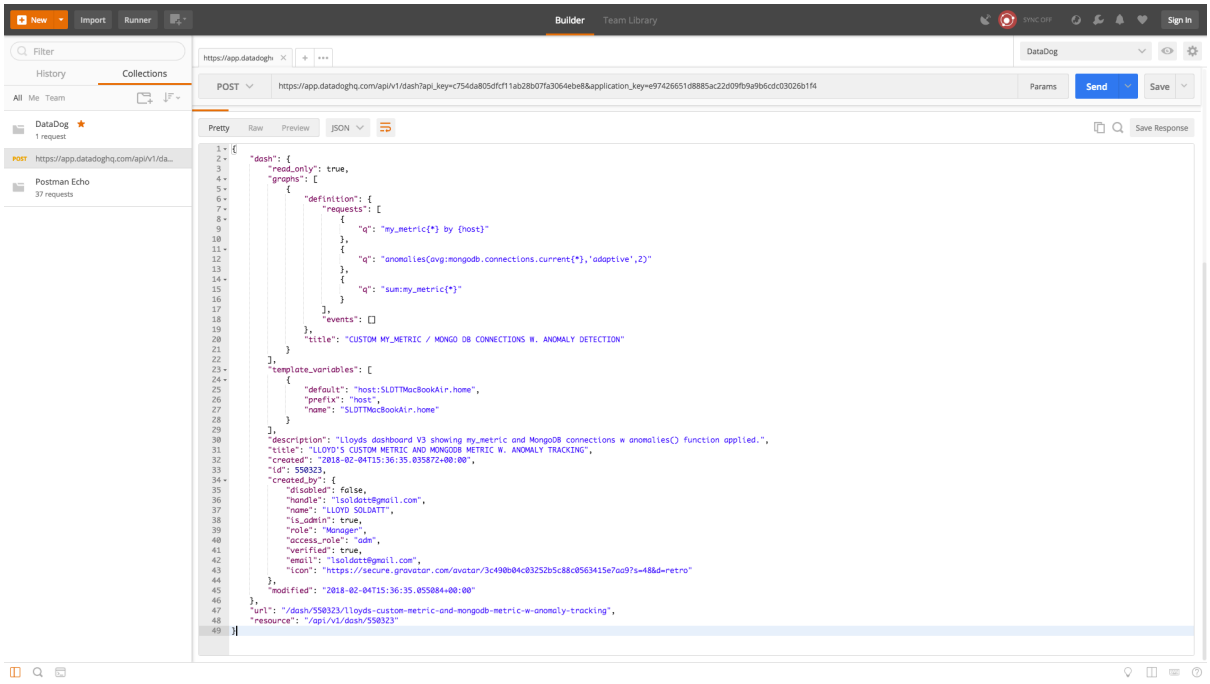
```
{
    "graphs" : [{
      "title": "CUSTOM MY_METRIC / MONGO DB CONNECTIONS W. ANOMALY
DETECTION",
        "definition": {
          "events": [],
          "requests": [
                    {"q": "my_metric{*} by {host}"},
                    {"q": "anomalies(avg:mongodb.connections.current{*},'adaptive',2)"},
                    {"q": "sum:my_metric{*}"}
          ]
        },
        "viz": "timeseries"
    }],
    "title" : "LLOYD'S CUSTOM METRIC AND MONGODB METRIC W. ANOMALY TRACKING",
    "description" : "Lloyds dashboard V3 showing my_metric and MongoDB connections w
anomalies() function applied.",
    "template_variables": [{
      "name": "SLDTTMacBookAir.home",
      "prefix": "host",
      "default": "host:SLDTTMacBookAir.home"
```

```
    }],
    "read_only": "True"
}
…
```

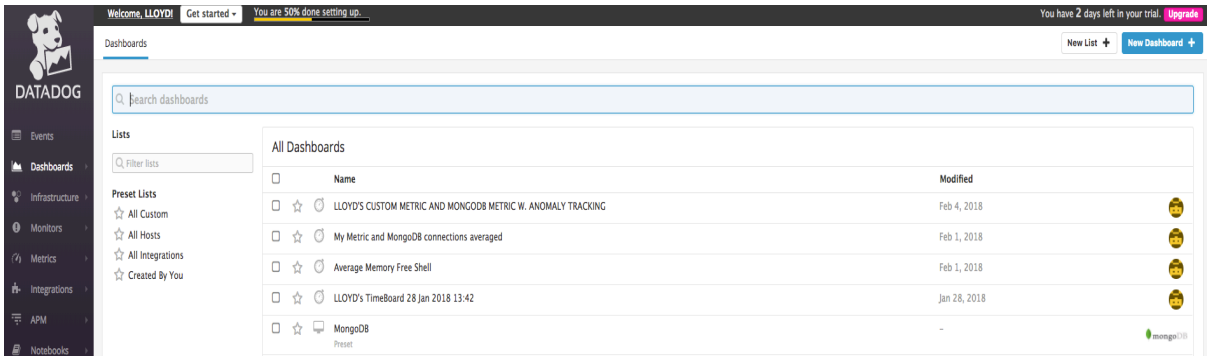
My Timeboard response JSON after its creation via the DD API POST call:


```
{
   "dash": {
      "read_only": true,
      "graphs": [
         {
            "definition": {
               "requests": [
                  {
                     "q": "my_metric{*} by {host}"
                  },
                  {
                     "q": "anomalies(avg:mongodb.connections.current{*},'adaptive',2)"
                  },
                  {
                     "q": "sum:my_metric{*}"
                  }
               ],
               "events": []
            },
            "title": "CUSTOM MY_METRIC / MONGO DB CONNECTIONS W. ANOMALY
DETECTION"
         }
      ],
      "template_variables": [
         {
            "default": "host:SLDTTMacBookAir.home",
            "prefix": "host",
            "name": "SLDTTMacBookAir.home"
         }
      ],
      "description": "Lloyds dashboard V3 showing my_metric and MongoDB connections w
anomalies() function applied.",
      "title": "LLOYD'S CUSTOM METRIC AND MONGODB METRIC W. ANOMALY TRACKING",
      "created": "2018-02-04T15:36:35.035872+00:00",
      "id": 550323,
      "created_by": {
         "disabled": false,
         "handle": "lsoldatt@gmail.com",
         "name": "LLOYD SOLDATT",
```

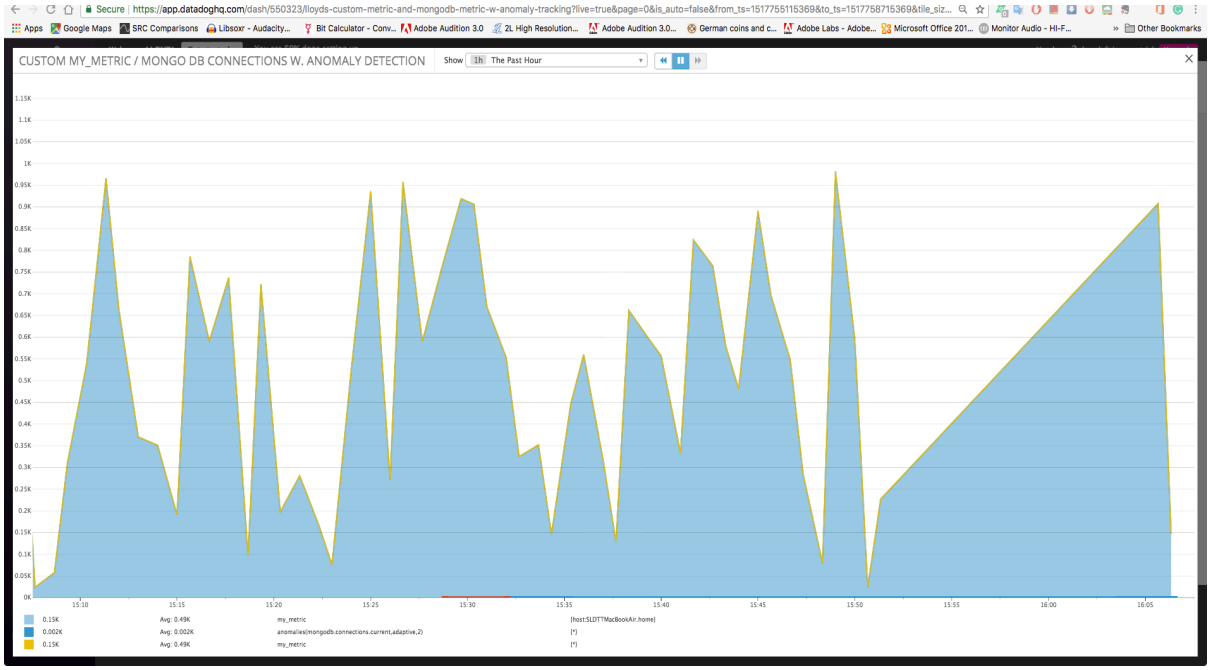
```
        "is_admin": true,
        "role": "Manager",
        "access_role": "adm",
        "verified": true,
        "email": "lsoldatt@gmail.com",
        "icon":
"https://secure.gravatar.com/avatar/3c490b04c03252b5c88c0563415e7aa9?s=48&d=retro
"
      },
      "modified": "2018-02-04T15:36:35.055084+00:00"
   },
   "url": "/dash/550323/lloyds-custom-metric-and-mongodb-metric-w-anomaly-tracking",
   "resource": "/api/v1/dash/550323"
}
```

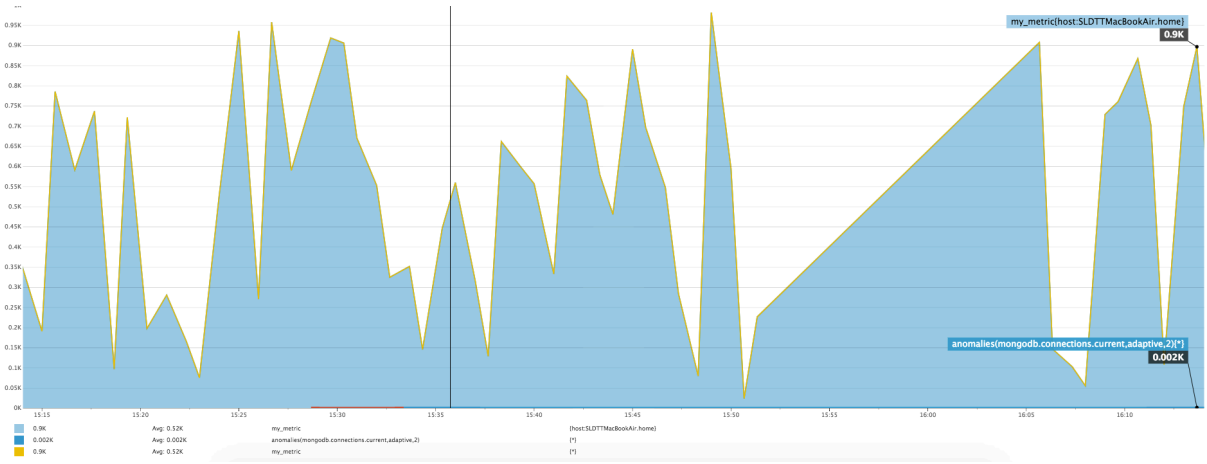
DataDog Dashboard (Timeboards) View



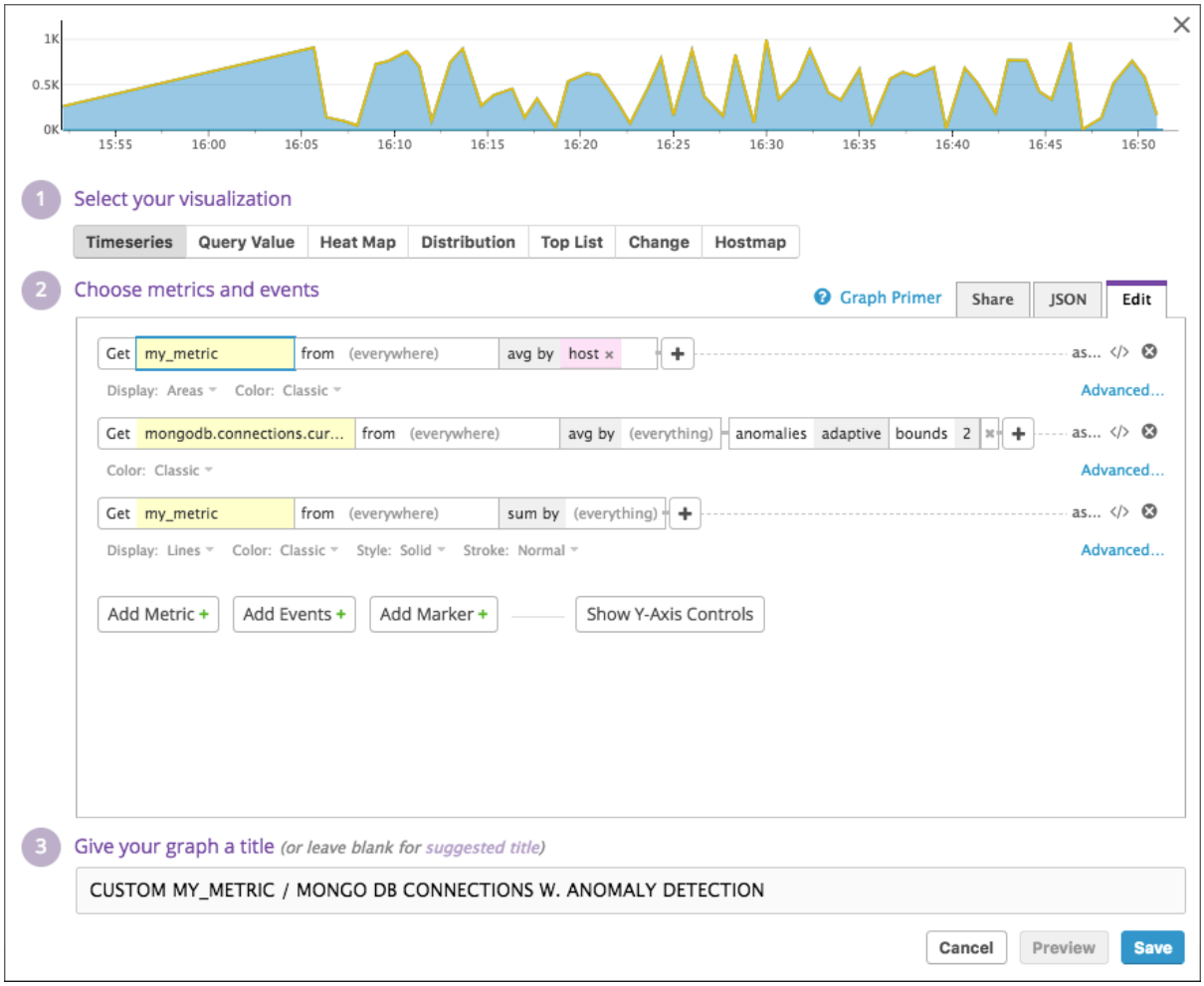
Zoom-in on the 3 Timeboard metrics created via the DD API post



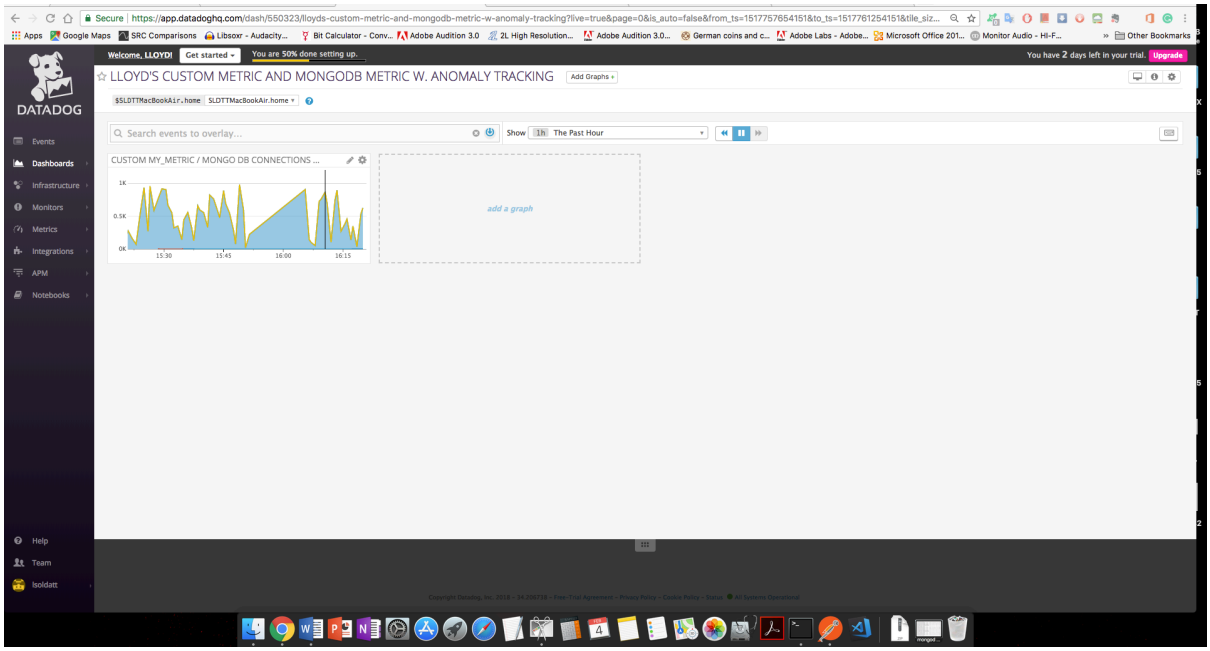
The 3 metrics positioned on one graph associated with this Timeboard:



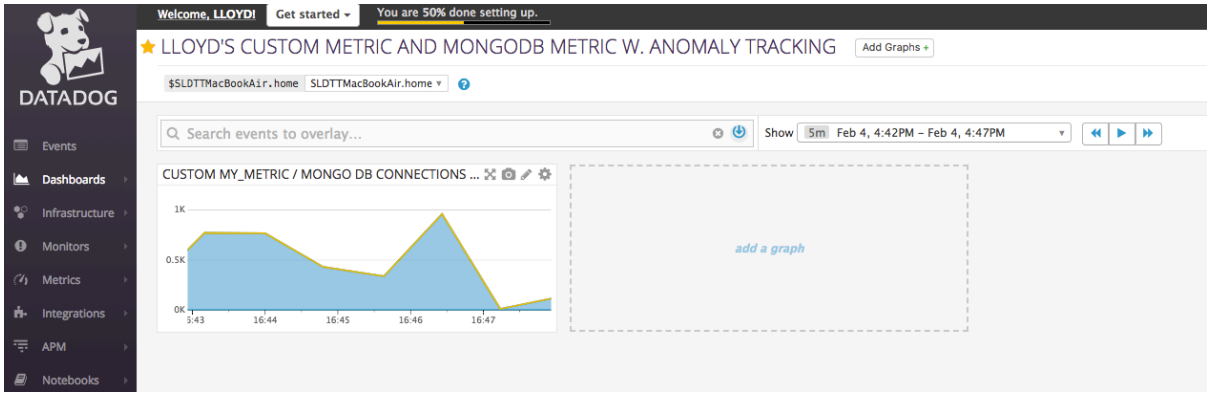
My Timeboard Settings View:

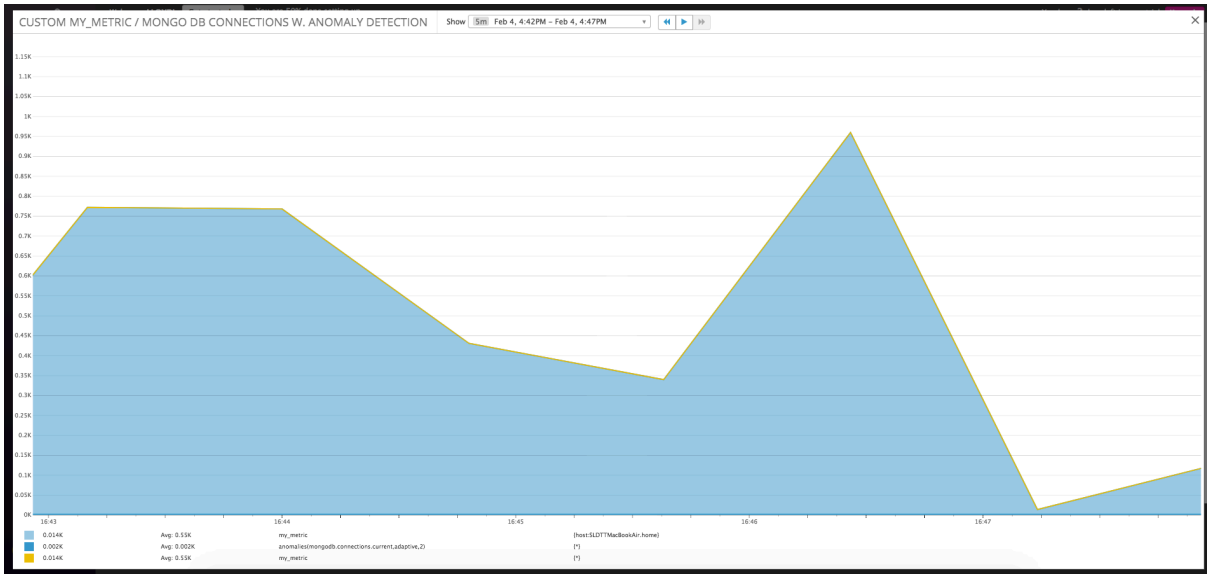


Overall view over 1 hour of the 2 metrics (my_metric, mongodb.connections.current) being monitored

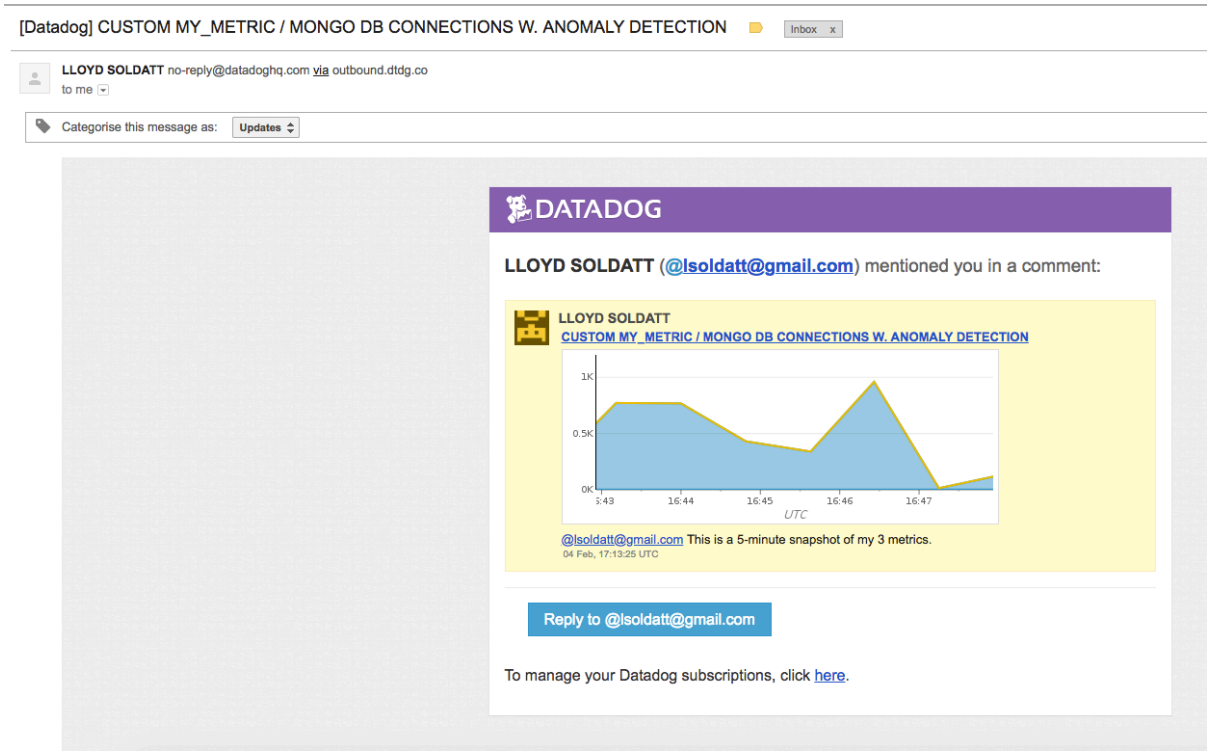


My metrics (my_metric and mongodb.connections.current) zoomed to 5 minutes on the Timeboard UI





The notated 5-minute metric snapshot is e-mailed to myself from my DD Cloud account



Note: *My DD anomaly function watches the MongoDB connections being open over a period of time. The number of connections is 2 most of the time, which is on average 500 times less than my random number gauge, that powers my_metric range. As such, the way the anomaly function shows itself on the graph (as above) is at the very bottom. It is, unfortunately, disproportionately smaller than the custom metric's output of random numbers between 1 and 1000.*

…

Bonus Question: My anomaly function for Mongo DB connections is set to be adaptive and that is why this type of algorithm application will work best when my open MongoDB connections will be out of their 'normal' range – currently 2 connections at a time. It shows if my Mongo DB connections will exceed their normal observed threshold. The anomaly function by DD is a very useful capability, which makes the solution extremely powerful in scenarios where hosted apps can produce sudden spikes of data/fluctuations, but which is perfectly normal for these applications, ex. MongoDB within an enterprise where query throughput grows exponentially at a certain hour of the day. A MongoDB instance can log a sudden spike of queries during the day and when we apply DD anomalies() function, it will look at this app performance from the historical perspective. Thus, although, this spike may look as a problem in the context of now and today, it is predictable and was historically observed time and time before. DD anomalies() function, when applied to a metric, will look at its logging performance in the context of its behaviour historically and will analyse whether this spike is, in fact, within this metric historically documented performance 'range'. Practically, it is also useful to apply this function over metrics known to fluctuate to avoid false alerts when, indeed, their behaviour is statistically within limits.

Monitoring Data

Monitoring is set up on my_metric as requested with alerting, warning and no data email notifications demonstrated below.

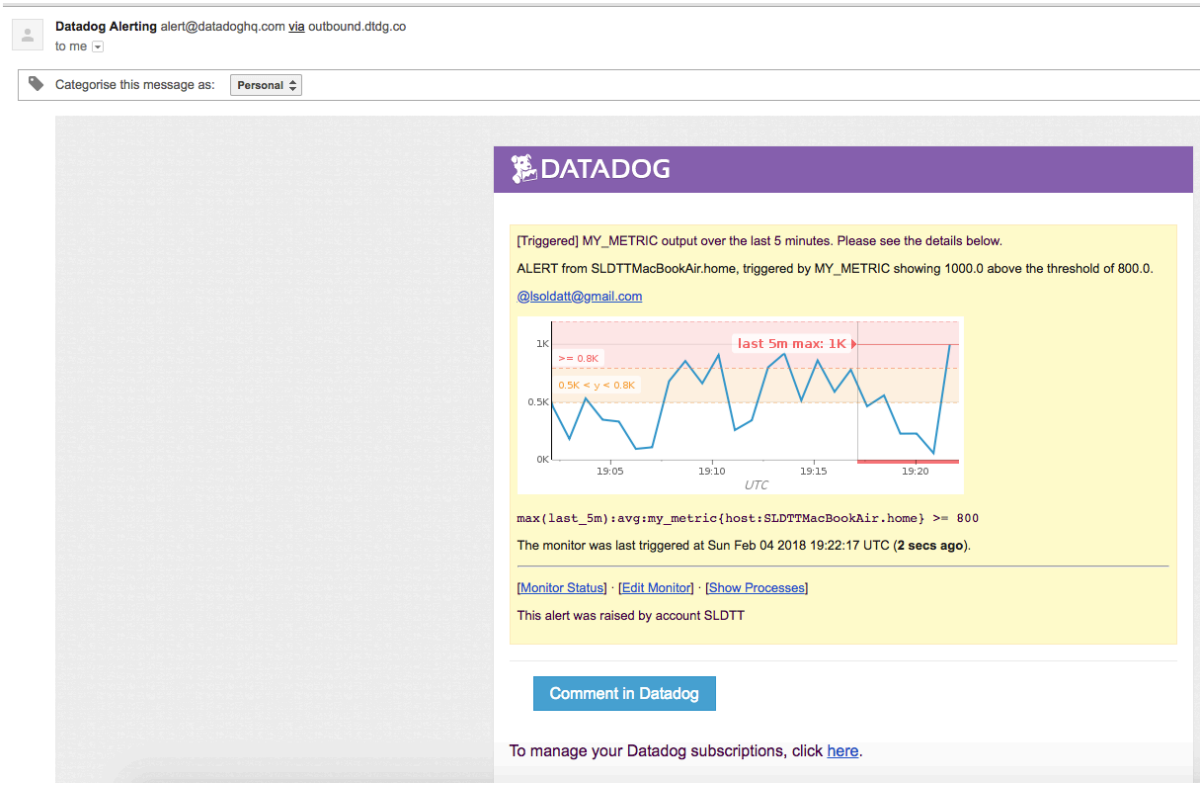
Monitor JSON export


```
{
        "name": "MY_METRIC output over the last 5 minutes. Please see the details below.",
        "type": "query alert",
        "query": "max(last_5m):avg:my_metric{host:SLDTTMacBookAir.home} >= 800",
        "message": "{{#is_alert}} ALERT from  {{host}}, triggered by MY_METRIC showing
{{value}} above the threshold of {{threshold}}. {{/is_alert}}\n{{#is_warning}} WARNING,
triggered by MY_METRIC showing {{value}} compared to the set threshold of
{{warn_threshold}}.{{/is_warning}}\n{{#is_no_data}} NO DATA, triggered by MY_METRIC
showing {{value}} .{{/is_no_data}}\n\n@lsoldatt@gmail.com",
        "tags": [
                "*"
        ],
        "options": {
                "timeout_h": 0,
                "notify_no_data": true,
                "no_data_timeframe": 10,
                "notify_audit": false,
                "require_full_window": false,
                "new_host_delay": 300,
```

```
                "include_tags": false,
                "escalation_message": "",
                "locked": true,
                "renotify_interval": "0",
                "evaluation_delay": "",
                "thresholds": {
                        "critical": 800,
                        "warning": 500
                }
        }
}
```

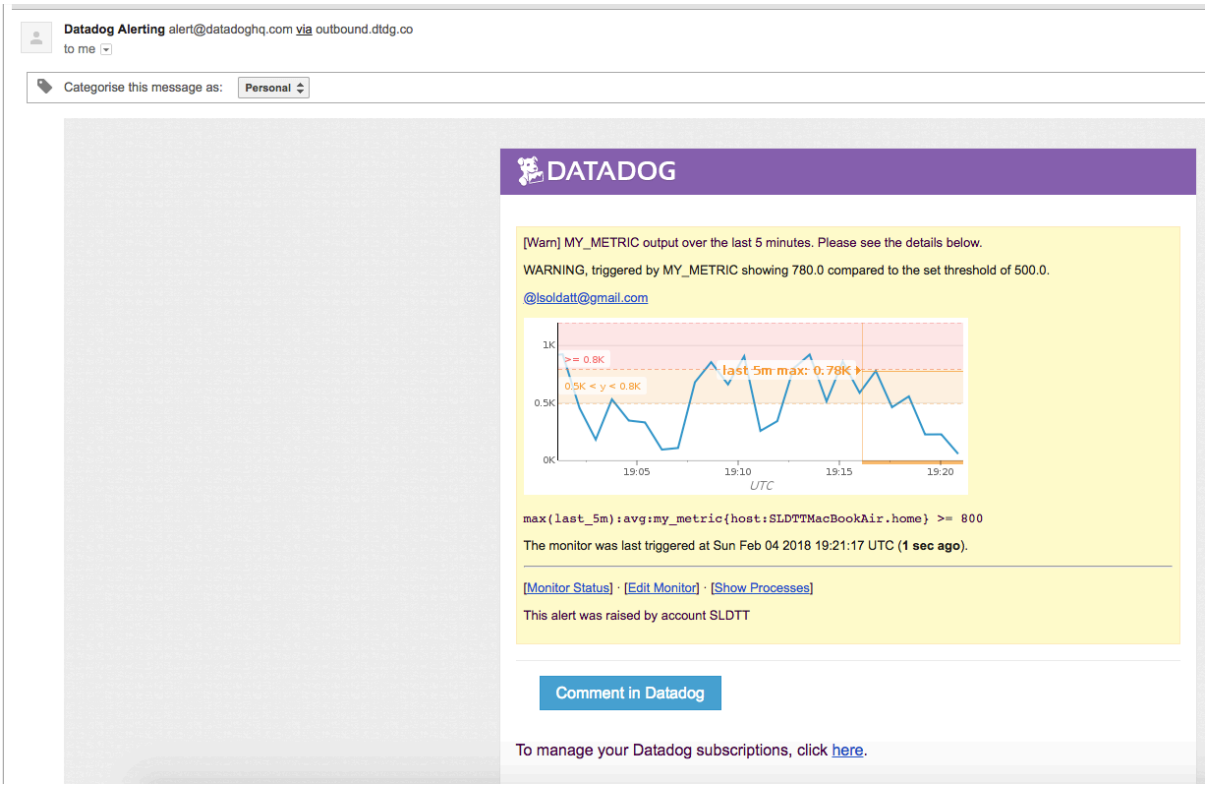
Alert monitoring message in email

(please note the alert type messaging within the email)



Warning monitoring message in email

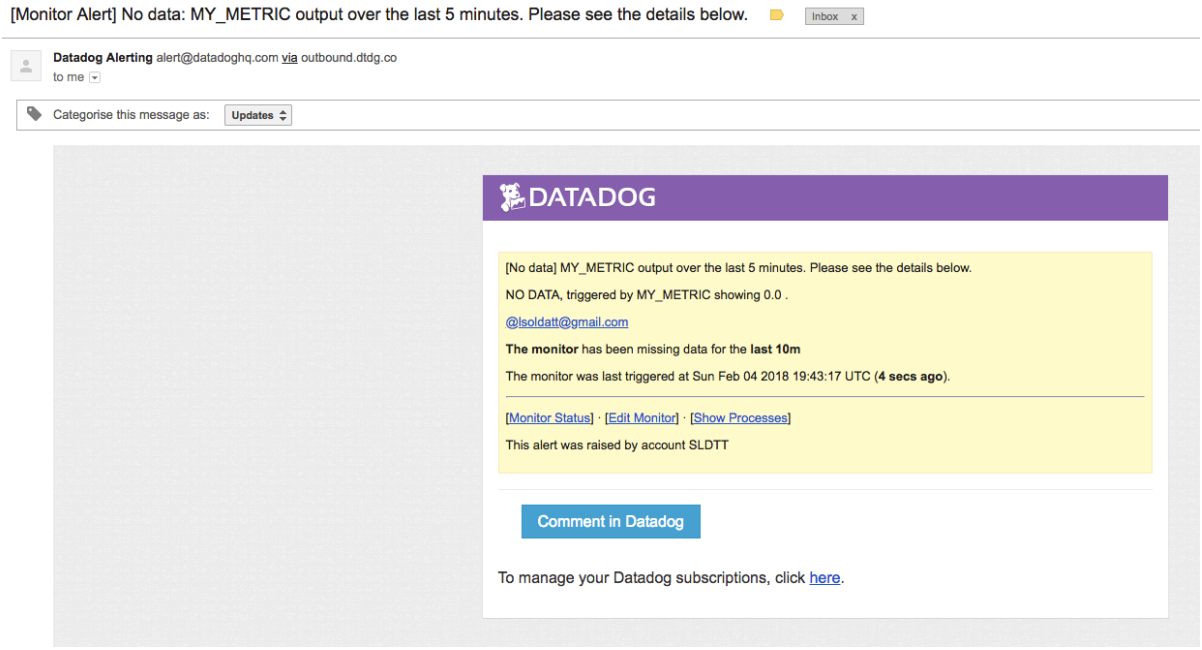
(please note the warning type messaging within the email)



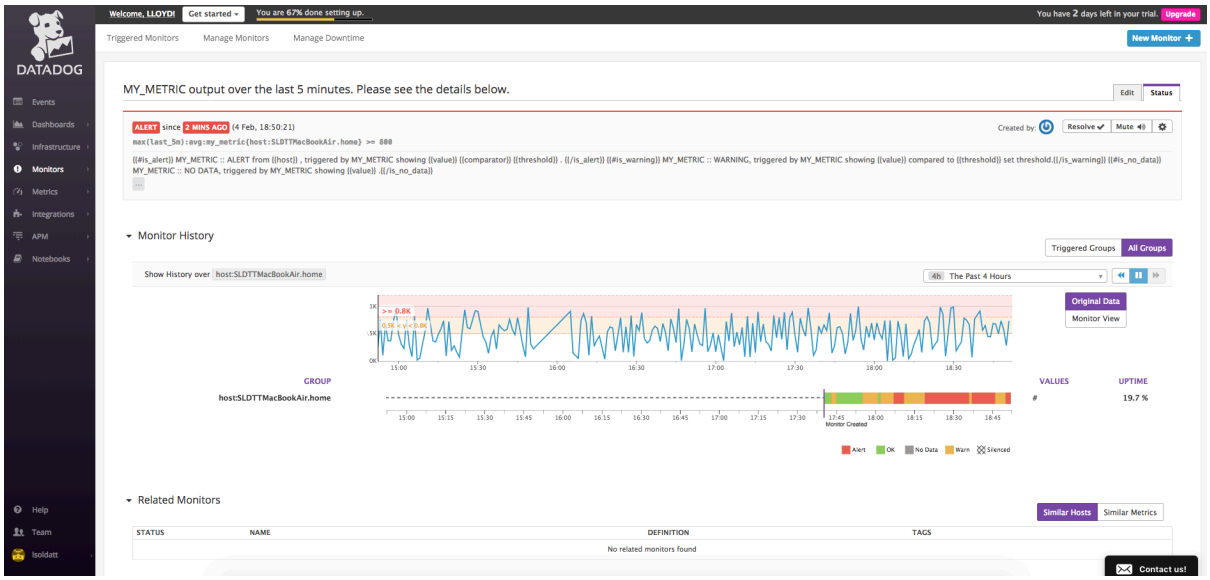
No data monitoring message in email

(please note the no data type messaging within the email)

In order to simulate the no data scenario, I disabled my DD agent running on my host for more than 10 minutes and this produced the desired no data type email messaging as below.

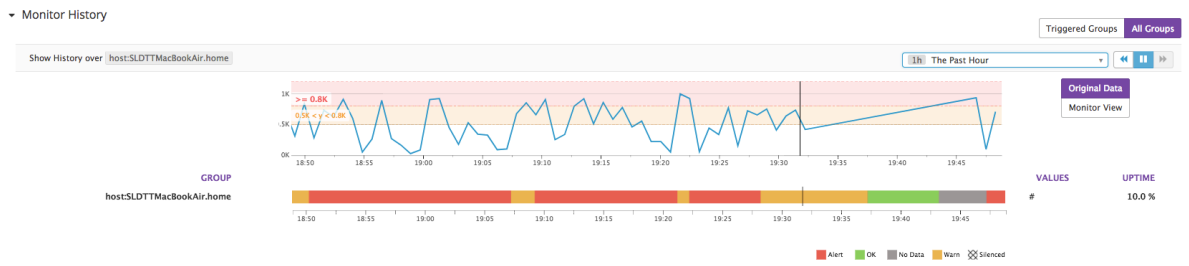


Alert, Warning, and No data monitors set up on my DD Cloud Account.

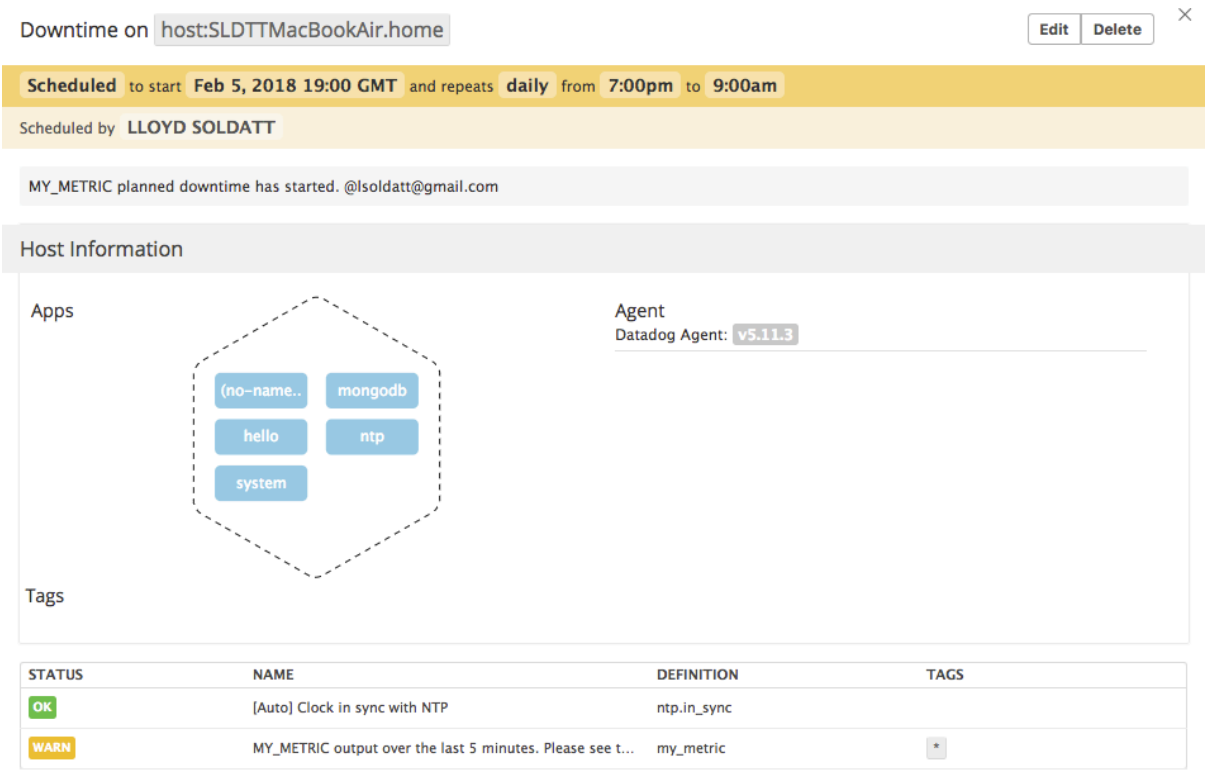


My monitor showing all the 3 states of Alert, Warning and No data taken place in the last hour

(the history legend shows all of the 3 states in red, orange and grey)



Bonus Question: I created the 2 scheduled downtimes as planned. One to cover a daily period from 7PM until 9 AM in the morning of the next business day, and the other – to mute the Alert/Warning/No data type messages for the entire weekend (Saturday/Sunday).



Downtime on host:SLDTTMacBookAir.home

Edit | Delete

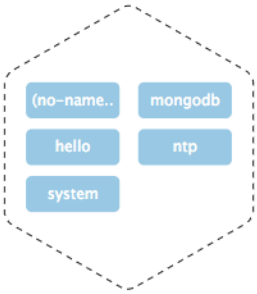
Scheduled to start **Feb 10, 2018 0:00 GMT** and repeats **weekly** from **12:00am** to **12:00am 2 days later** on **Sunday and Saturday**

Scheduled by **LLOYD SOLDATT**

MY_METRIC weekend planned downtime started. @lsoldatt@gmail.com

Host Information

Apps



Agent

Datadog Agent: v5.11.3

Tags

STATUS	NAME	DEFINITION	TAGS
OK	[Auto] Clock in sync with NTP	ntp.in_sync	
WARN	MY_METRIC output over the last 5 minutes. Please see t...	my_metric	*

[Datadog] LLOYD SOLDATT scheduled downtime on {host:SLDTTMacBookAir.home} Inbox x

LLOYD SOLDATT no-reply@datadoghq.com via outbound.dtdg.co
to me

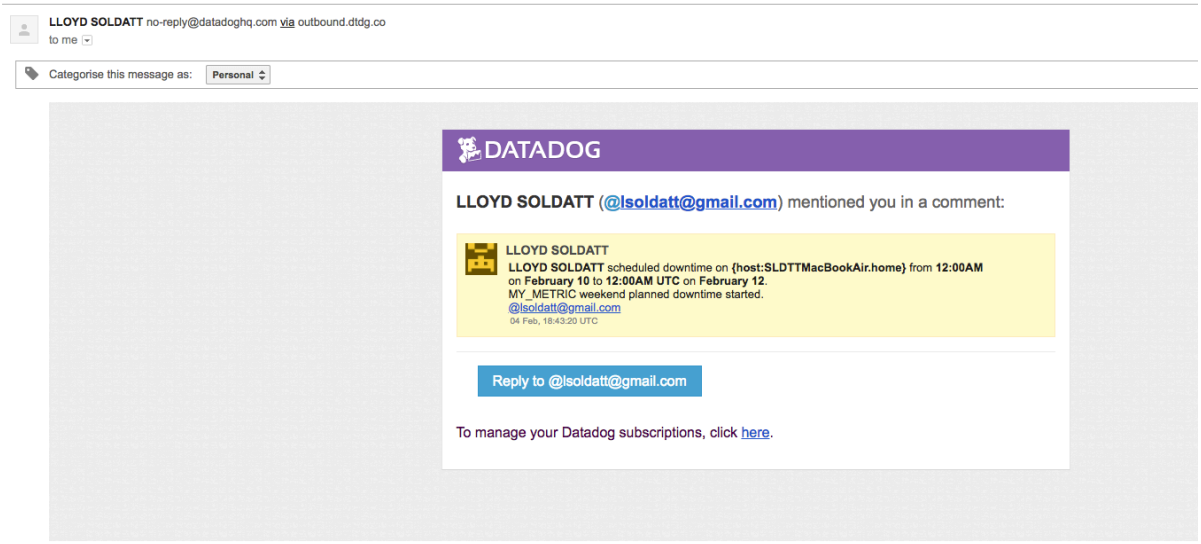
DATADOG

LLOYD SOLDATT (@lsoldatt@gmail.com) mentioned you in a comment:

LLOYD SOLDATT
LLOYD SOLDATT scheduled downtime on **{host:SLDTTMacBookAir.home}** from **7:00PM** on **February 5** to **9:00AM UTC** on **February 6**.
MY_METRIC planned downtime has started.
@lsoldatt@gmail.com
04 Feb, 18:39:49 UTC

Reply to @lsoldatt@gmail.com

To manage your Datadog subscriptions, click here.



Final Bonus Question:

My creative way of using DataDog Cloud Monitoring: At Prism Skylabs (www.prism.com) we monitor IoT devices (IP cameras and others) and sensors located in our customer sites around the world, as such, we are in 100+ countries now. It is humanly impossible to know what is happening to every sensor at any given point in time. We rely on some basics internal dashboard to email our Support Team when any sensor was unresponsive for more than 20min. DataDog will be ideal solution for that, we would have our own Support-run DD cloud account, an API channel to send us/and our customers directly, alerts when any particular IoT sensor is down or has not been responsive for a few minutes. This capability is essential to our customers, as, historically, we ran into lots of issues for the last 2 years when we could not guarantee our SLA, vital to our contract extension and upsell potential on the accounts. The losses we incurred would justify the subscription fees already, or these fees could be factored into our pricing.

_end