

The Exercise

By Ismael Cama Moumen

1.1 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.

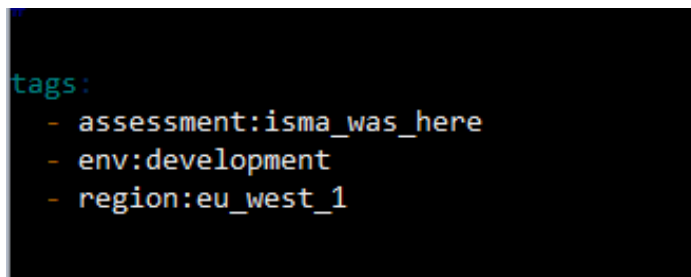
Datadog provides a rich range of possibilities when it comes to organizing and visualizing data, yet one of the simplest but most powerful tools are tags. Tags allow to clearly classify data sources intuitively, allowing to group them by any criteria that suits the user best.

Each data source can contain multiple tags, allowing aggregate metrics providing very detailed insights about the user's infrastructure. For example, using tags, we can get the metrics from hosts which are part of the production environment, in the region Neptune-34. The hosts are also executing the application "satellite radar", deployed on docker containers running on distro astro:

```
tags: ["env:prod", "region:neptune_34", "app:satellite_radar", "platform:docker", "distribution:distro_astro"]
```

This granularity level empowers users to have both a broad overview of their systems, and a thorough in-depth understanding of a very specific range of their infrastructure. In no time, and without a drop of sweat.

As shown above, tags can be defined using a single line, which is portable and simple to manipulate developing scripts. However, tags can also be provided in a more human-readable format, as shown below.



```
tags:
  - assessment:isma_was_here
  - env:development
  - region:eu_west_1
```

Tags can be defined in the datadog agent configuration file which you can find in `/etc/datadog-agent/datadog.yaml`.

In order to apply the new tags, the datadog agent process needs to be restarted. Please

make sure that the datadog agent was able to be started correctly using "systemctl status datadog-agent".

It can take a couple of minutes for tags to be visible in the host map dashboard. But if you are testing and would want to wait even less to confirm, the new tags should be available sooner in the first field when creating a new monitor.

Below we can see a host using the "host map" dashboard. The tags show that it is a vagrant host from the development environment, which is in the region eu-west-1, and that Isma has been there.



**NOTE: If the agent experiences any issues to start, you can find more details in "/var/log/datadog/agent.log".*

For more information on tags, please visit [1].

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

In this section we will explore how datadog provides solutions to integrate third party tools such as Apache, EC2 or MySQL right out of the box. For this example, we will configure a monitoring solution for MongoDB databases in a single host.

We only need to follow three steps: creating a MongoDB read-only user, defining a few fields in a datadog configuration file, and enabling (with just 2 clicks!) the datadog portal integration.

A. Configuring MongoDB

For this section, we assume that users have already installed MongoDB on their host. However, if that's not the case, you can follow the steps in [2] to install the application before following this exercise.

On the MongoDB side, we only need to create a read-only user to allow Datadog to monitor metrics from our databases. For that, we will create a user with read-only access in the admin database.

```
root@vagrant:/etc/datadog-agent/conf.d/mongo.d#
root@vagrant:/etc/datadog-agent/conf.d/mongo.d# mongosh
Current Mongosh Log ID: 61937df07ccb30031dd12484
Connecting to:      mongodb://127.0.0.1:27017/?directCo
Using MongoDB:      5.0.4
Using Mongosh:      1.1.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
  The server generated these startup warnings when booting
  2021-11-16T09:46:19.318+00:00: Using the XFS filesystem
  2021-11-16T09:46:20.140+00:00: Access control is not enabled
-----

test> use admin
switched to db admin
admin> db.auth("myUserAdmin", "abc123")
{ ok: 1 }
admin>

admin>

admin>

admin> db.createUser({
...   "user": "datadogUser",
...   "pwd": "mypassword",
...   "roles": [
...     { role: "read", db: "admin" },
...     { role: "clusterMonitor", db: "admin" },
...     { role: "read", db: "local" }
...   ]
... })
{ ok: 1 }
admin>
```

Do not forget to authenticate with your user admin to create the datadog read-only user. And that's all we need to do on mongoDB! Next, we need to configure datadog.

B. Configuring the datadog agent

No extra packages need to be installed to integrate datadog with mongodb. As promised, out of the box! We only need to let the datadog agent know that there are MongoDB databases to be monitored. For this, we will provide some details about the MongoDB process running on the host using the template stored in "/etc/mongodb/conf.d/conf.yaml.example". We will create a copy of this template and define the configuration in it:

```
cp /etc/mongodb/conf.d/conf.yaml.example /etc/mongodb/conf.d/conf.yaml
```

In this configuration file we will tell the datadog agent the following:

- * Where can it find the MongoDB process running (if you are not sure, you can run "netstat -luntp" to get this data)
- * The username and password we configured for the agent's read-only access
- * The database to monitor
- * Authorization options

Below you can see an example. All comments have been subtracted for convenience (and because we still have them in the template).

```
init_config:
instances:
  - hosts:
    - 127.0.0.1:27017
    username: datadogUser
    password: mypassword
    database: testingdb
    options:
      authSource: admin
    tags:
      - database:mongodb
```

Now, we just need to restart our agent: "systemctl restart datadog-agent".

To confirm whether the agent is accessing the mongoDB, you can execute "sudo datadog-agent status", and look for the mongo integration status. Any errors that require remediation will be displayed in this section. You can find what to expect if the

integration works fine below.

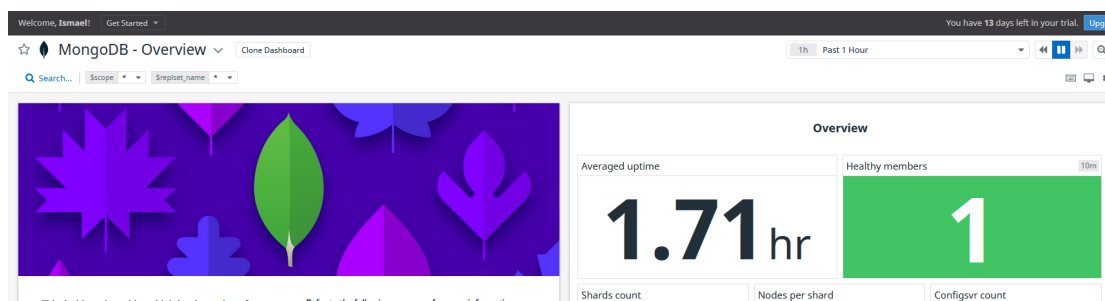
```
mongo (2.7.0)
-----
Instance ID: mongo:d7696121de6e8456 [OK]
Configuration Source: file:/etc/datadog-agent/conf.d/mongo.d/conf.yaml
Total Runs: 281
Metric Samples: Last Run: 119, Total: 33,439
Events: Last Run: 0, Total: 0
Service Checks: Last Run: 1, Total: 281
Average Execution Time : 15ms
Last Execution Date : 2021-11-16 11:31:19 UTC (1637062279000)
Last Successful Execution Date : 2021-11-16 11:31:19 UTC (1637062279000)
metadata:
  version.major: 5
  version.minor: 0
  version.patch: 4
  version.raw: 5.0.4
  version.scheme: semver
```

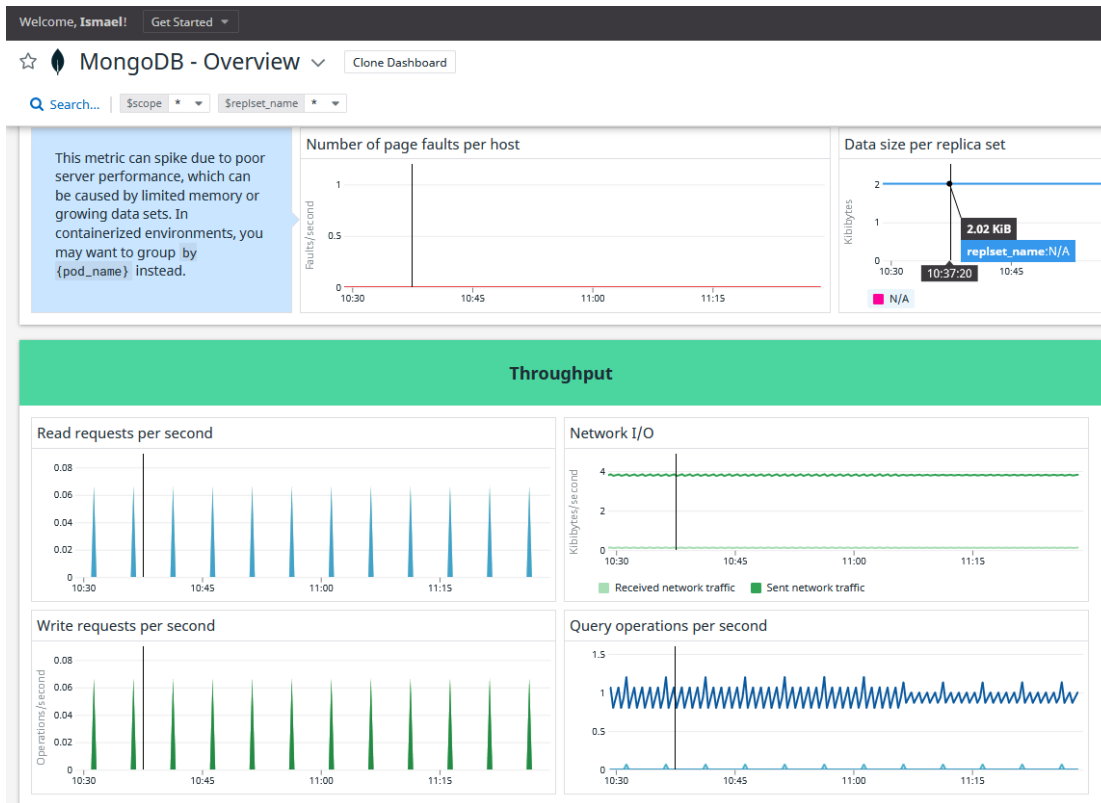
C. 2-click MongoDB integration setup for the DataDog portal

For this step, we only need to browse to the integrations Menu in the Datadog portal, select the MongoDB integration, and to press the "Install integration" button in the bottom part of the window.

And voilà! Integration finished! To see the results we just need to navigate to the dashboards section and select "MongoDB".

Quick and simple, right? Datadog provides more than 450 integrations with third party tools which require the same level of configuration as the above. Handy, right? Well, that's not all: datadog allows to also store logs from the application which is monitoring[3], and to have an even more enhanced level of granularity monitoring by being able to monitor individual executions running in your databases (trace monitoring)[4].





To see if datadog provides integrations with your own stack you can visit [5].

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

If a user needs a solution not covered by the hundreds of available integrations, it is very simple to create a custom solution. Any datapoint that the user can access is a datapoint within the reach of datadog's monitoring capabilities.

For example, the little python snippet below ingests to Datadog a random number between 0 and 1000. This snippet must be in the datadog's "checks.d" directory, and a configuration file with the same name must exist inside of datadog's "conf.d" directory. For example, the name of the python file program is "my_random_value.py", and the configuration file is "my_random_value.yaml".

NOTE: The metric's name is "ismas.random.number.generator" because "my_metric" was too boring.

```

from checks import AgentCheck
from random import randint

class HelloCheck(AgentCheck):
    def check(self, instance):
        self.gauge("ismas.random.number.generator", randint(0, 1000))

```

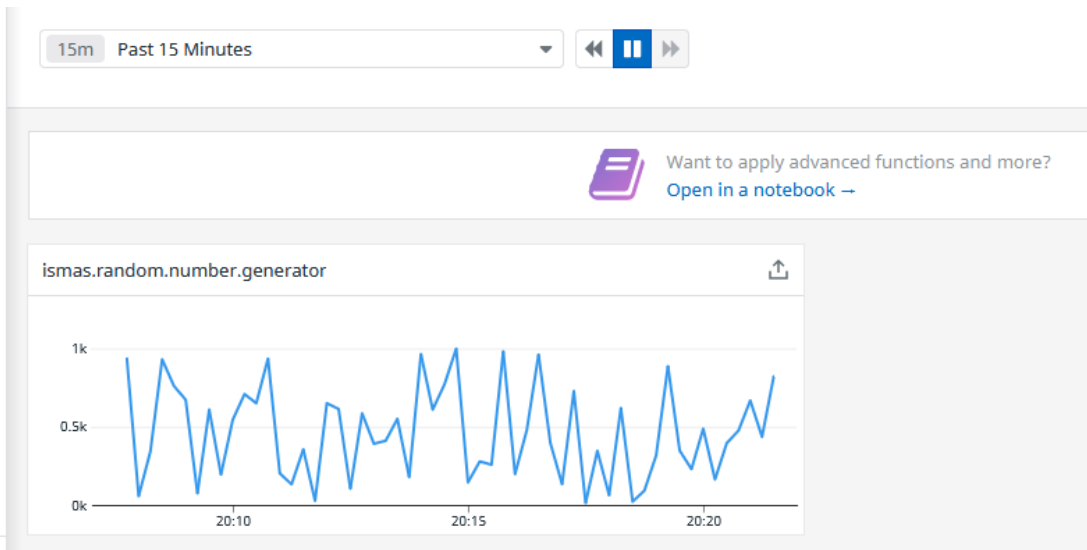
This process requires minimal configuration to get to work. Below you can find the contents of my_random_value.yaml

```

init_config:

instances:
  [{}]
```

Then, we restart the agent to let datadog know about the new metric. Within a few seconds we can see the published data in the datadog platform. After navigating over "Metrics > Summary" and selecting the new metric we created, we can see datadog in action:



Quick, clean, and simple.

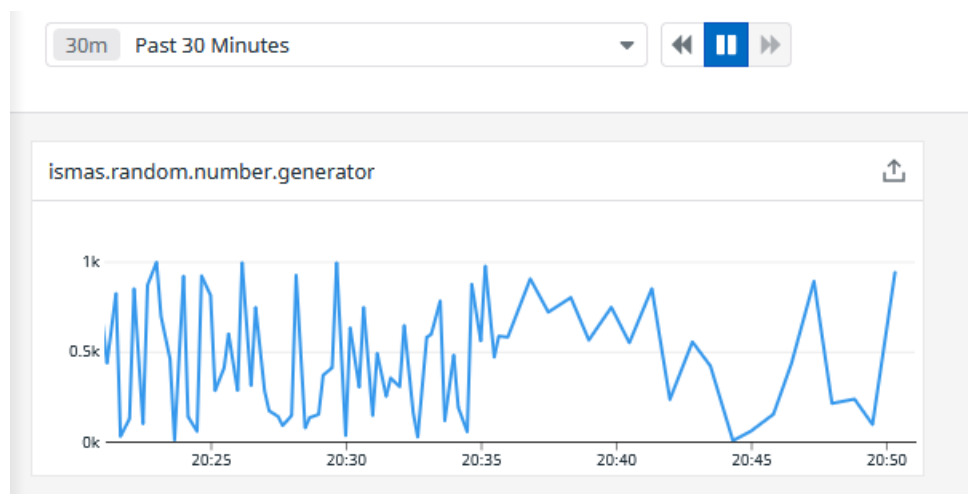
1.4 Change your check's collection interval so that it only submits the metric once every 45 seconds.

After the quick setup for the custom metric publishing, some users might be curious about how to use the "my_random_value.yaml" configuration file. In this file, users can define different aspects of the metric collection process. For example, to specify that the metric "ismas.random.number.generator" should be collected as often as every 45 seconds, the file should include the line "- min_collection_interval: 45" as follows.

```
init_config:


instances:
  - min_collection_interval: 45
```

And below we can observe how the data collection interval changed from 15 seconds to 45s.



(BONUS QUESTION) Can you change the collection interval without modifying the Python check file you created?


The collection interval can also be set using the datadog platform interface. This can be handy if the user has no access to configuration files. In order to perform this change, the user should navigate to "Metrics" > "Summary" > Search for the metric > "Edit" (In metadata section) and introduce the value in seconds.

ismas.random.number.generator 

INGESTED CUSTOM METRICS	INDEXED CUSTOM METRICS	HOSTS	TAG VALUES
1	1	1	3

▼ Metadata

Unit: per

Metric Type: 

Interval:

Description:

This can be also done using Datadog's API, which supports Go, Python, Typescript, Java and Ruby (legacy version). Of course, it can be performed using curl as well, as shown below.

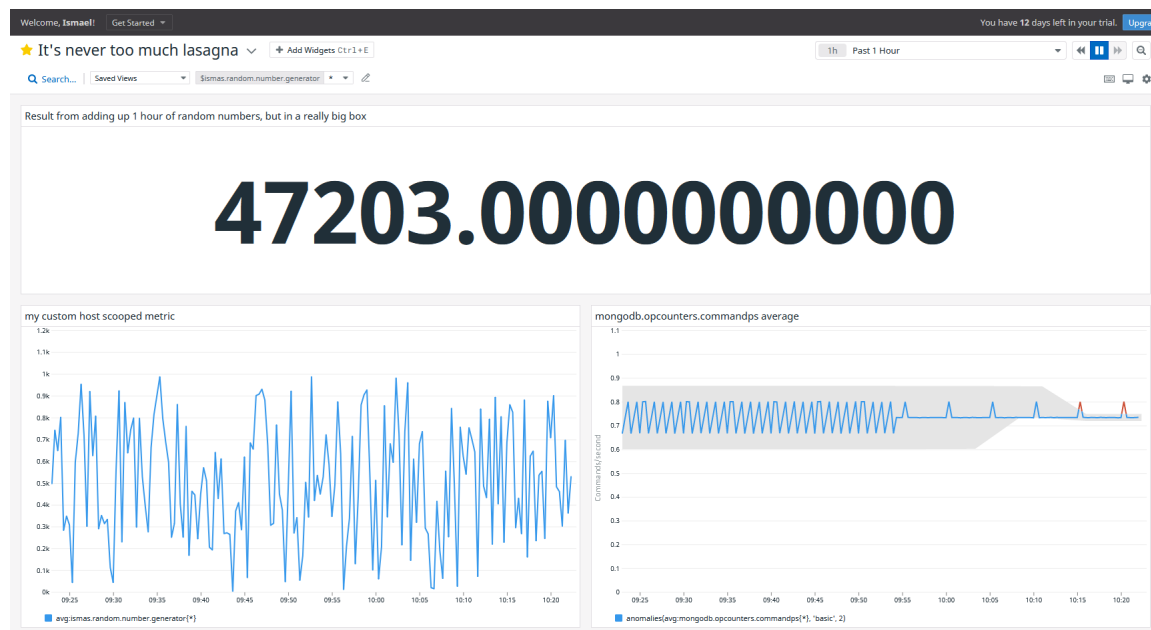
```
root@vagrant:/etc/datadog-agent/conf.d# curl -X PUT "https://api.datadoghq.eu/api/v1/metrics/ismas.random
.number.generator" -H "Content-Type: application/json" -H "DD-API-KEY: [REDACTED]55"
-H "DD-APPLICATION-KEY: [REDACTED]ba5" -d @- << EOF
{
  "statsd_interval": 58
}
EOF
{"description":null,"short_name":null,"integration":null,"statsd_interval":58,"per_unit":null,"type":"gau
ge","unit":null}root@vagrant:/etc/datadog-agent/conf.d#
```

2.1 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***

Dashboards can be created both manually and through APIs. The manual approach is a great way to get familiarized with the options that datadog offers, whereas APIs are really handy to back up pre-existing dashboards and deploy them in no time whenever it is required.

The dashboard below has been created through an API execution:



In this dashboard we have three widgets that display data using different formulas and features.

The simplest widget is "my custom host scoped metric", which shows a timeseries graph showing the evolution of our random number generator.

In the right-hand side, there is another timeseries widget, but this time it has an anomaly monitor applied over it. Anomaly monitors learn from the past data behaviour to detect any trends change, notifying the appropriate users when required.

Finally, on the top, we can observe the result of adding the last 1 hour of random numbers. Both widgets (bottom left, and top) use the exact same metric, but provide different insights from it using completely different formats. This widget, "query_value", is using the function "rollup" to provide the final result.

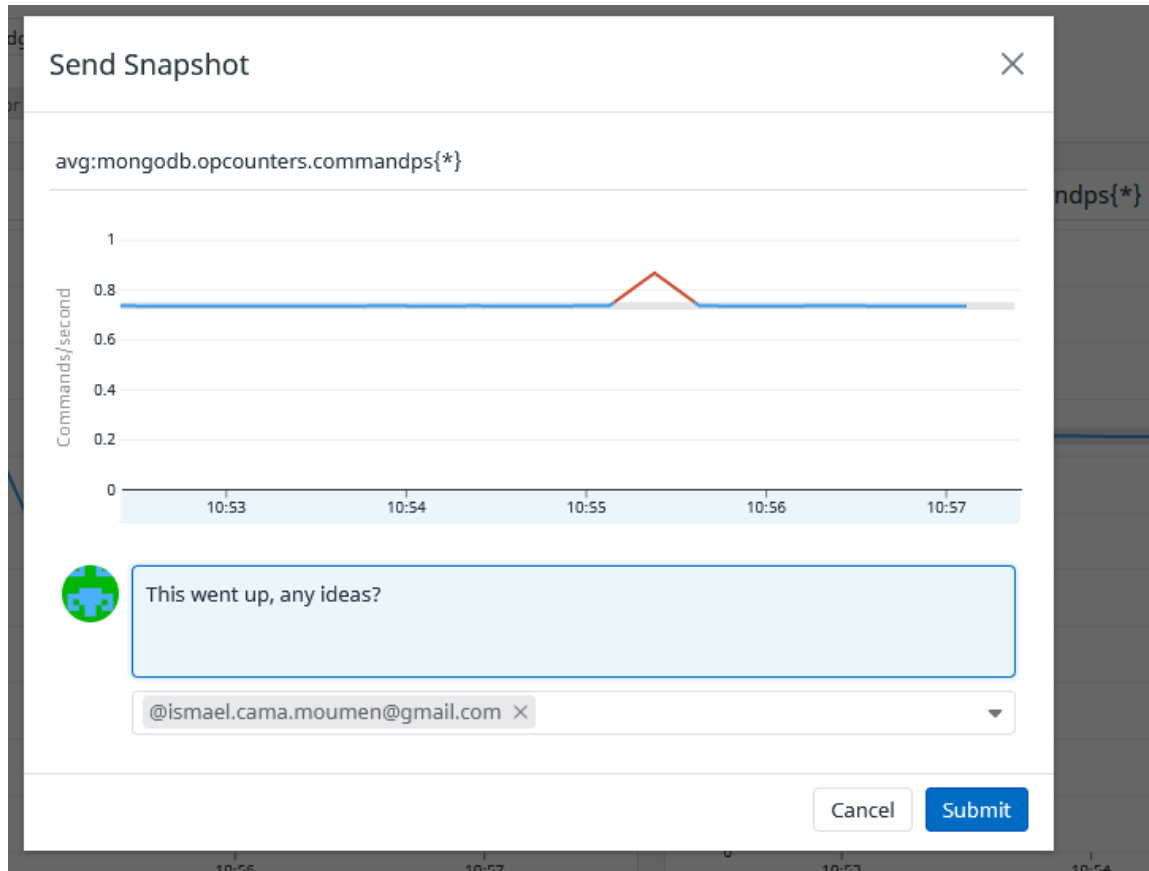
Thanks to datadog's API capabilities, I created this dashboard instantly. This allows for great portability and immediate data insights.

2.2 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.**

Any graph can also be instantly snapshotted and submitted to team members. In just two clicks, a user can notify others via mail about any trends or specific behaviours

observed. Below you can find an example of a user sharing a snapshot while asking Ismael for advice.



Below you can see what Ismael receives and sees.

[Datadog] avg:mongodb.opcounters.commandps{*} [Safata d'entrada x](#)

Ismael Cama Moumen <no-reply@datadg.eu>
per a mi

DATADOG

Ismael Cama Moumen (@ismael.cama.moumen@gmail.com) mentioned you in a comment:

Ismael Cama Moumen
avg:mongodb.opcounters.commandps{*}

avg:mongodb.opcounters.commandps{*}

Commands/second

10:53 10:54 10:55 10:56 10:57

@ismael.cama.moumen@gmail.com This went up, any ideas?

17 Nov, 10:57:52 UTC

View or reply in Datadog

[Download the Datadog Mobile App](#) to triage alerts from anywhere.

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

The anomaly graph displays the expected behaviour of this metric compared with the actual behaviour. The anomaly graph is learned using the specified algorithm from the past metric history to make predictions, and will also detect any data points that do not align with the expected trends.

3.1 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.**

Thanks to datadog monitors, users can define alerting policies to be notified every time certain conditions are met. There are currently two alerting severities: "Warning" for metrics that are at uncomfortable levels but not urgent yet, and "Alerting" for metrics that require immediate attention.

For example, let's imagine that we are managing a supermarket chain, and depending on each supermarket size, local regulations set specific limits for the amount of people that can be inside our premises at once. Datadog could be used to monitor the amount of people inside of each supermarket, and be integrated with a people access control system solution. If a specific supermarket can hold a maximum of 100 people at once...

- If below 75, the monitor does not trigger
- If above 75 (or equal), the monitor triggers a warning, which is sent to the access displays to ask customers to wait until given green light
- If above 85, the monitor triggers an alarm, which will page the supermarket personnel to personally control the access
- If no data, the monitor triggers alarm to page the supermarket personnel to personally control the access

To illustrate this, I will use our old friend "ismas.random.number.generator". To create a monitor to keep an eye on this metric, I have navigated "Monitors" > "New monitor" > "Metric".

Below you can find the metric configurations:

2 Define the metric

Source Edit

a ismas.random.number.generator from (everywhere) avg by (everything) Σ

+ Add Query + Add Formula

Simple Alert Trigger a single alert when your metric satisfies your alert conditions. ?

3 Set alert conditions

Trigger when the metric is above the threshold on average during the last 5 minutes

Alert threshold: > 800

Warning threshold: > 500

Alert recovery threshold: <= Optional

Warning recovery threshold: <= Optional

Do not require a full window of data for evaluation. ?

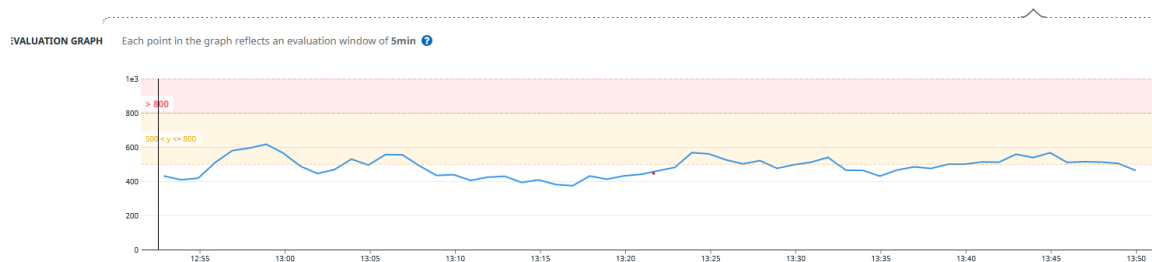
Note: We highly recommend you select "Do Not Require" for sparse metrics, otherwise some evaluations will be skipped.

Notify if data is missing for more than 10 minutes. ?

Note: we recommend the missing data window be at least 2x the evaluation period above

- i. Warning if average value of this metric is above 500 for more than 5 minutes
- ii. Alert if average value for this metric is above 800 for more than 5 minutes
- iii. Notify if data missing for more than 10 minutes

As shown below, datadog provides a graphic summary of alerts and warnings:



3.2 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.**

For this alarm, a mail will be sent every time that the monitor triggers. However, the messages will be different depending on whether it triggered a warning, an alert, or a no data notification. Below you can find an example:

{{#is_alert}}

ALERT!!! Random number in host {{host.ip}} is over 800!!! Quick, buy bitcoin!!!

Current metric value: {{value}}

{{/is_alert}}

{{#is_warning}}

Warning! Random number in host {{host.ip}} is over 500! Careful!

Current metric value: {{value}}

{{/is_warning}}

{{#is_no_data}}

Ooopssss!!! No more random numbers. No random number generator detected for host {{host.ip}}

{{/is_no_data}}

Notify: @ismael.cama.moumen@gmail.com

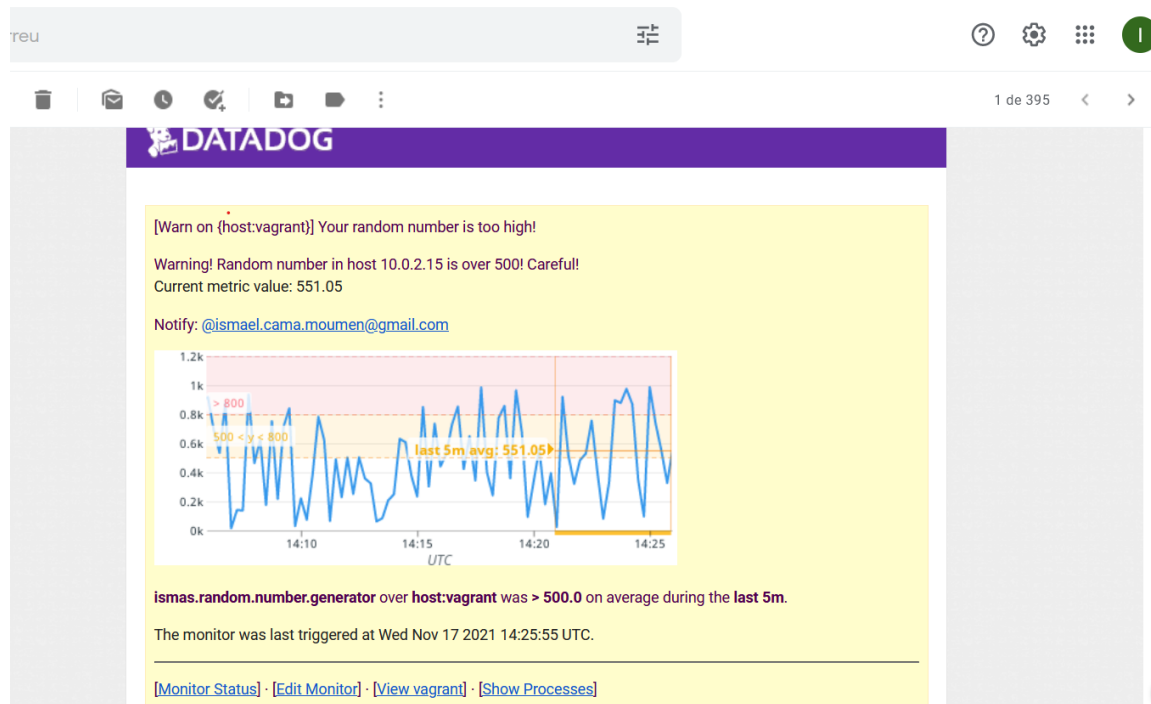
The wordings for all three events are in the same place. It can't get simpler, right?

Also, the notifications allow a certain level of customization using variables. Hence, users can know the exact current status of the systems without the need of accessing

the datadog portal.

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

Below you can find an example of what a user gets when the alarm above triggers with a warning



Datadog provides both host address and latest value, but many other variables can be used to provide further context and richer notifications.

- * ***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:***

A) One that silences it from 7pm to 9am daily on M-F,

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

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

Unlike Datadog, users are not available 24/7. For this reason, most might want to disable alarms/warnings when being out of the office. This can be accomplished by scheduling downtimes. To silence the monitor we just created, we will need to select

"Monitors" > "Manage Downtimes". Then, we will create a downtime.

Downtime events can be programmed using the GUI, or using RRULEs (In a nutshell, an RRULE defines the repeat pattern for events). To show both methods, for the weekends we will use the GUI, and for MON-FRI rules, we will use RRULEs.

Programming weekends:

1 Choose what to silence

By Monitor Name

By Monitor Tags

Monitor:

Your random number is too high!

Group scope (optional, default all groups):

* ×

?

Preview Affected Monitors

?

2 Schedule

One Time

Recurring

Editor

RRULE

Summary:

Starting on Nov 17, 2021, 12:00 AM GMT for 1 day
Weekly on Sunday and Saturday

Start Date:

2021/11/17

Time Zone:

Atlantic/Canary (UTC+00:00)

Beginning at:

12 : 00 AM ×

Duration:

1

days

Repeat Every:

1

weeks

Sun ×

Sat ×

Repeat Until:

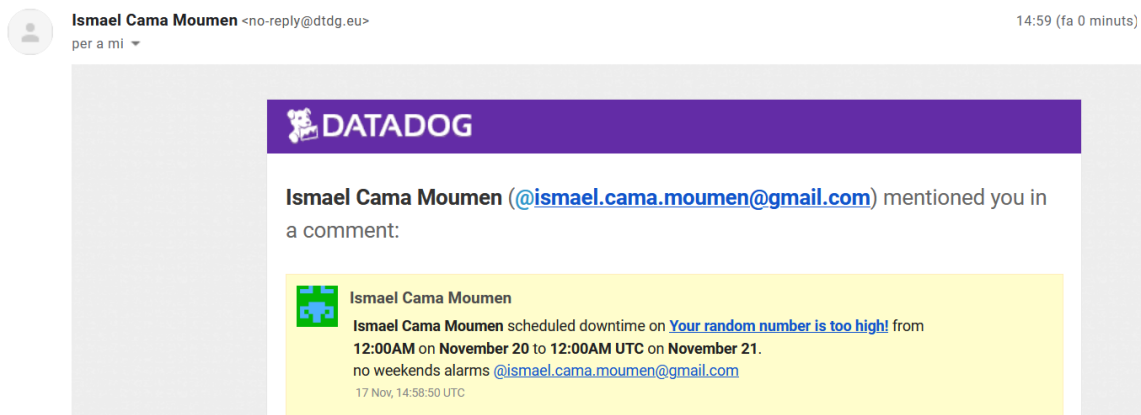
No end date

Programming for weekdays. RRULE:

"FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR;INTERVAL=1"

The screenshot shows the Datadog interface for scheduling a silence. On the left is a dark sidebar with the Datadog logo and navigation links: Go to..., Watchdog, Events, Dashboards, Infrastructure, Monitors, Metrics, Integrations, APM, CI, Notebooks, Logs, Security, UX Monitoring, Contact Support, Help, and Invite Users. The main content area is divided into two sections. Section 1, 'Choose what to silence', has two tabs: 'By Monitor Name' (selected) and 'By Monitor Tags'. It shows a monitor named 'Your random number is too high!' and a group scope dropdown set to '*'. A link 'Preview Affected Monitors' is at the bottom. Section 2, 'Schedule', has two tabs: 'One Time' and 'Recurring' (selected). It shows a summary of the schedule starting on Nov 17, 2021, at 7:00 PM GMT for 10 hours with the RRULE 'FREQ=WEEKLY;BYDAY=MO, TU, WE, TH, FR; INTERVAL=1'. The start date is set to 2021/11/17, the time zone is Atlantic/Canary (UTC+00:00), and the beginning time is 07:00 PM. The recurrence rule (RRULE) is displayed in a text box as 'FREQ=WEEKLY;BYDAY=MO, TU, WE, TH, FR; INTERVAL=1'.

And below you can find mails confirming the changes.





Ismael Cama Moumen (@ismael.cama.moumen@gmail.com) mentioned you in a comment:



Ismael Cama Moumen

Ismael Cama Moumen scheduled downtime on [Your random number is too high!](#) from 7:00PM on November 17 to 5:00AM UTC on November 18.

No notifications while out of office @ismael.cama.moumen@gmail.com

17 Nov, 14:45:42 UTC

[View or reply in Datadog](#)

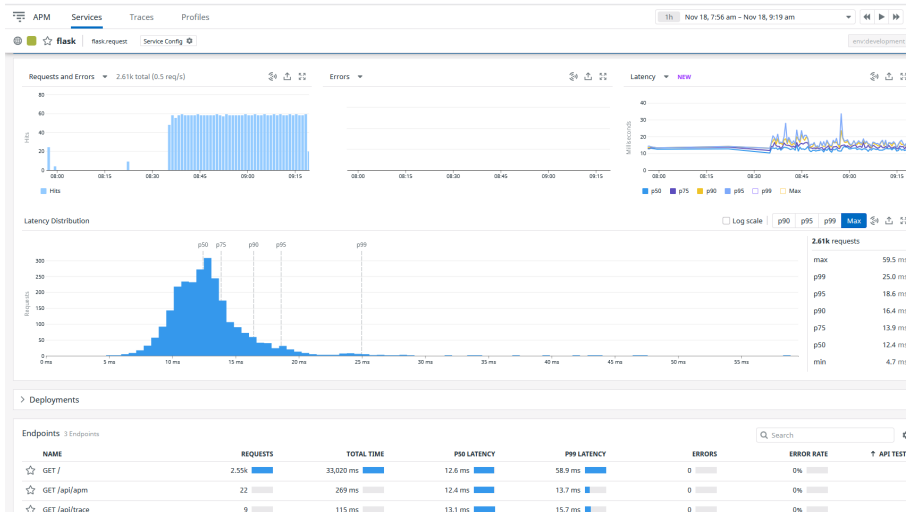
[Download the Datadog Mobile App](#) to triage alerts from anywhere.

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

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

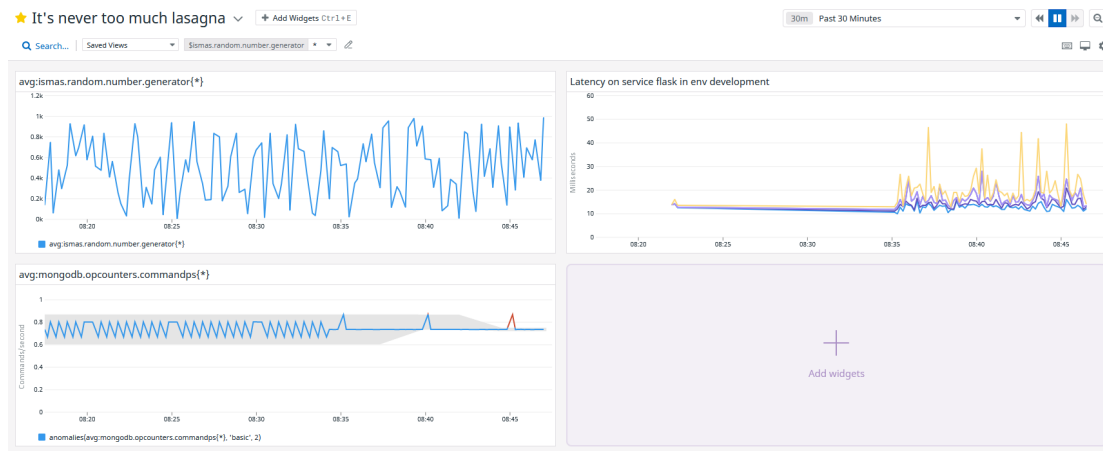
Datadog provides thorough in-depth execution flow possibilities for user's applications. Thanks to its APM (Application Performance Monitor) feature, execution flows can be analyzed to the line of code.

For example, the chart below shows metrics from a simple python application being executed. In a single view, we can see execution times, counts, latencies and errors. Specific requests can also be explored, and their execution flows analyzed.



In order to get all this insight, it was only required to use ddtrace, and set linux environment variables.

Below we can see a dashboard that combines data from both the host (isma's random number generator), mongodb and the program latency that it is executing, making it really easy to understand the whole application state, from host resource utilization to application execution flow.



Url to this dashboard:

<https://p.datadoghq.eu/sb/4dddae66-45f0-11ec-b882-da7ad0900005-26ade5f2d4b8adf3ab5707043ecd7a21>

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

A service is a compound of processes working together to provide a set of functionalities. A resource is a component used by a service in order to be executed. For example, Datadog is a service that requires CPU and Memory resources in order to be executed.

5. 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?

For a restaurant chain, Datadog could be used to monitor the amount of purchased, used and discarded ingredients on a daily/weekly basis, as well as how many times

each dish was ordered per day. After enough time of monitoring, prediction algorithms could be used to look for patterns, so ingredients are purchased in a more efficient way on each specific site, reducing costs.

This could also be used in order to measure the popularity of newly introduced dishes, and customer habit shifts over time. If the chain has a Fidelization programme, it could even track a per-customer activity, providing personalized rewards to regular customers, and some special offers to customers who stop buying in the store (so they can be "recovered").

If the restaurant chain management is all for data insights, the software could even be used to see if specific employees/site managers perform better than others (when a new manager/waiters start, revenue increases... etc), so the most successful sites can share their experience with others in order to improve overall revenue.

Resources

=====

[1] https://docs.datadoghq.com/getting_started/tagging/

[2] <https://docs.mongodb.com/manual/tutorial/>

[3] <https://docs.datadoghq.com/integrations/mongo/?tab=standalone#log-collection>

[4] https://docs.datadoghq.com/tracing/setup_overview/

[5] <https://docs.datadoghq.com/integrations/>