Datadog Proof of Value (PoV)

Prepared for Wayne Enterprises

January 11, 2019

Presented by: Jon Thompson, Sr. Solutions Architect
jonmthompson@gmail.com
415.793.7778

DATADOG

Executive Summary

Introduction

Thank you for the opportunity to participate in Wayne Enterprise’s (WE) Proof Of Value with Datadog.

Datadog is the essential monitoring platform for cloud and on-premise applications. We bring together data from servers, containers, databases, and third-party services to make your stack entirely observable. These capabilities help teams avoid downtime, resolve performance issues, and ensure customers are getting the best user experience.

Detailed Requirements

The technical requirements for this POV can also be found on this page:
https://github.com/DataDog/hiring-engineers/tree/solutions-engineer

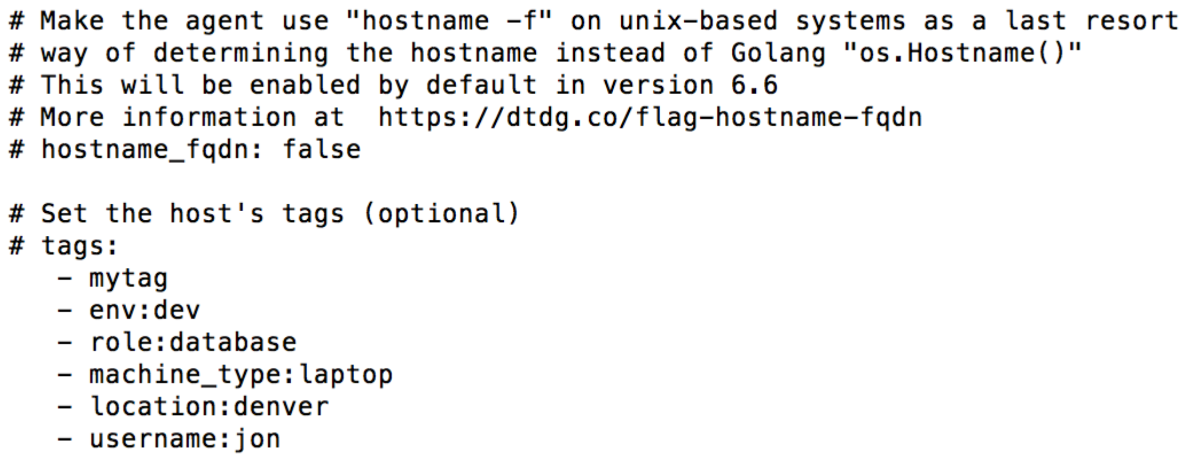
Environment

The Datadog Agent (v6) was installed on three separate machines, representative of WE’s actual environment, which consists of physical MacOS workstations, servers, and Linux Virtual Machines.

Technical Requirement: Collecting Metrics

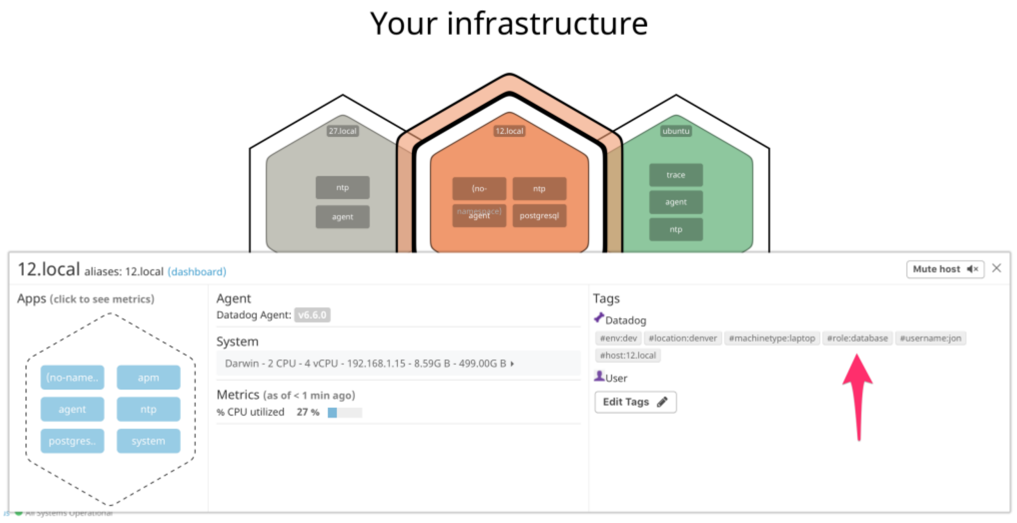
1. *Add tags in the Agent config file and show us a screenshot of your host and its tags on the Host Map page in Datadog.*

   Screenshot of /opt/datadog-agent/etc/datadog.yaml from workstation running macOS Sierra 10.12.6:



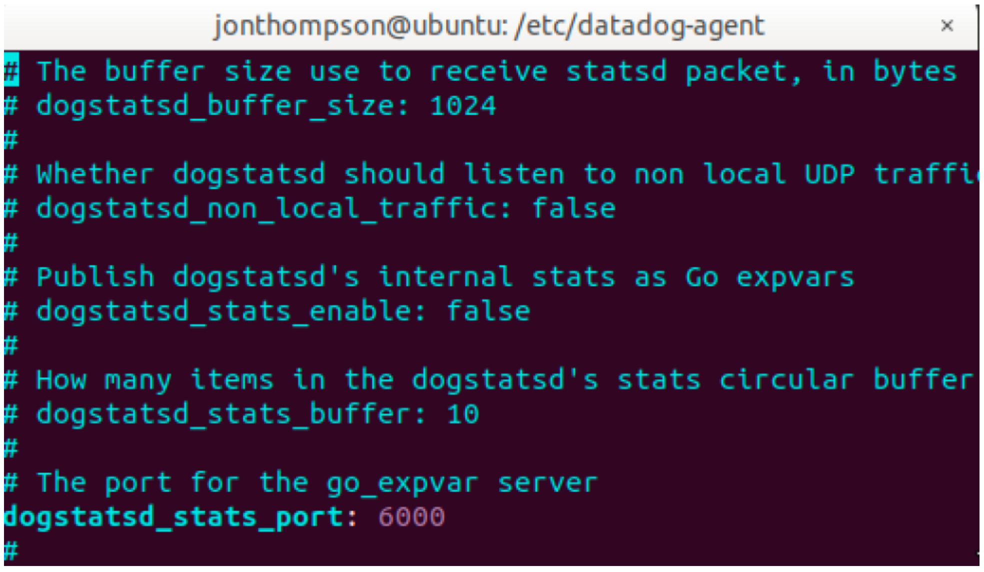
```
# Make the agent use "hostname -f" on unix-based systems as a last resort
# way of determining the hostname instead of Golang "os.Hostname()"
# This will be enabled by default in version 6.6
# More information at  https://dtdg.co/flag-hostname-fqdn
# hostname_fqdn: false

# Set the host's tags (optional)
# tags:
   - mytag
   - env:dev
   - role:database
   - machine_type:laptop
   - location:denver
   - username:jon
```

Screenshot of these new tags on the Host Map page in Datadog:



NOTE: An additional, last minute requirement was to run the Datadog Agent on a non-standard port, which is 5000 by default.

The Agent can easily be configured to run on a different port by editing the expvar_port entries in /opt/datadog-agent/etc/datadog.yaml:



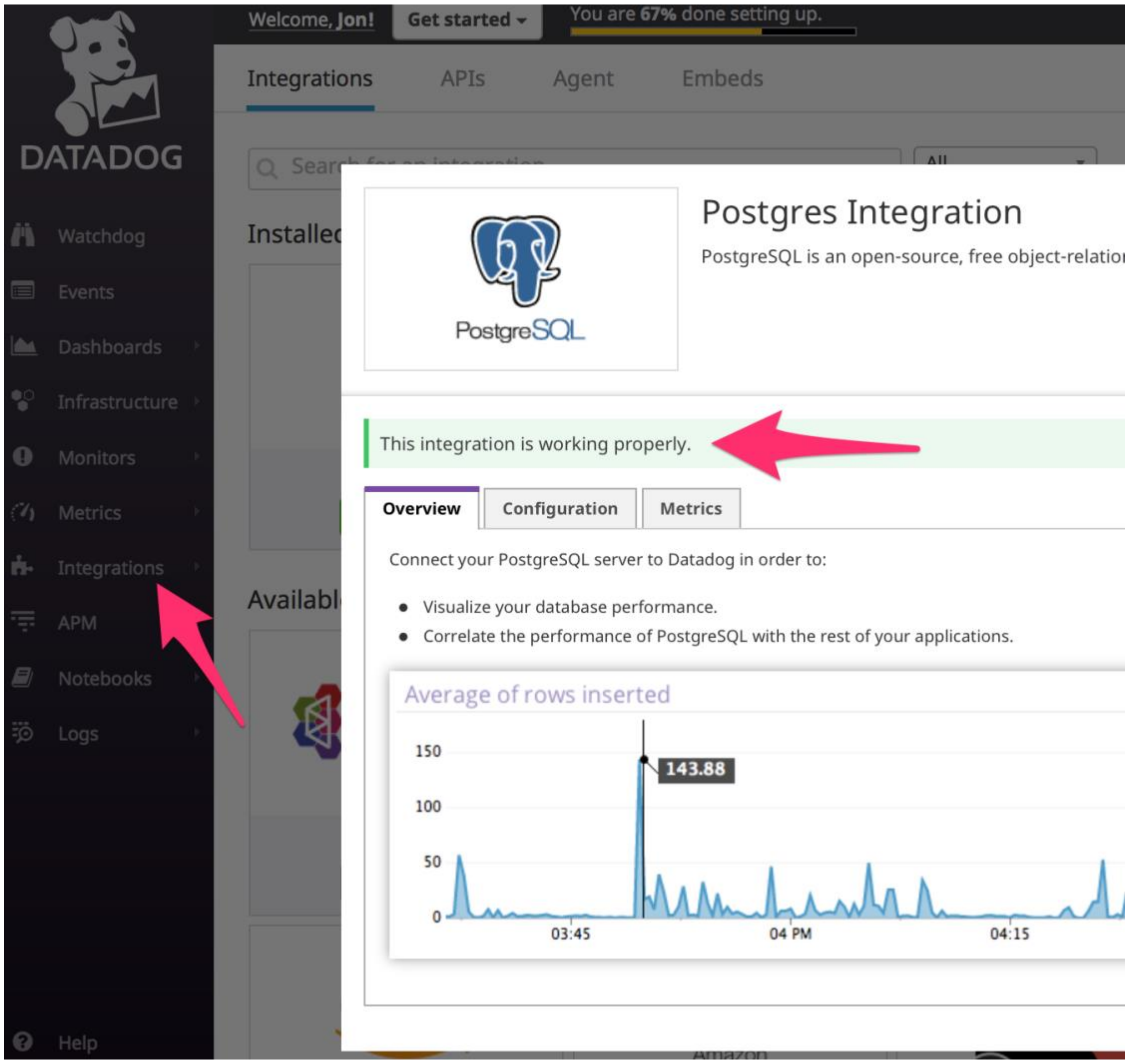
```
jonthompson@ubuntu: /etc/datadog-agent
# The buffer size use to receive statsd packet, in bytes
# dogstatsd_buffer_size: 1024
#
# Whether dogstatsd should listen to non local UDP traffi
# dogstatsd_non_local_traffic: false
#
# Publish dogstatsd's internal stats as Go expvars
# dogstatsd_stats_enable: false
#
# How many items in the dogstatsd's stats circular buffer
# dogstatsd_stats_buffer: 10
#
# The port for the go_expvar server
dogstatsd_stats_port: 6000
#
```

Additional information on the Datadog Agent (v6) can be found here:
https://docs.datadoghq.com/agent/?tab=agentv6

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

PostgreSQL is used throughout WE and was an ideal choice to demonstrate the robust database integration capabilities of Datadog.

From the Datadog Integrations menu we chose PostgreSQL, followed the steps outlined in the Configuration tab, and were quickly gathering metrics.



Postgres Integration

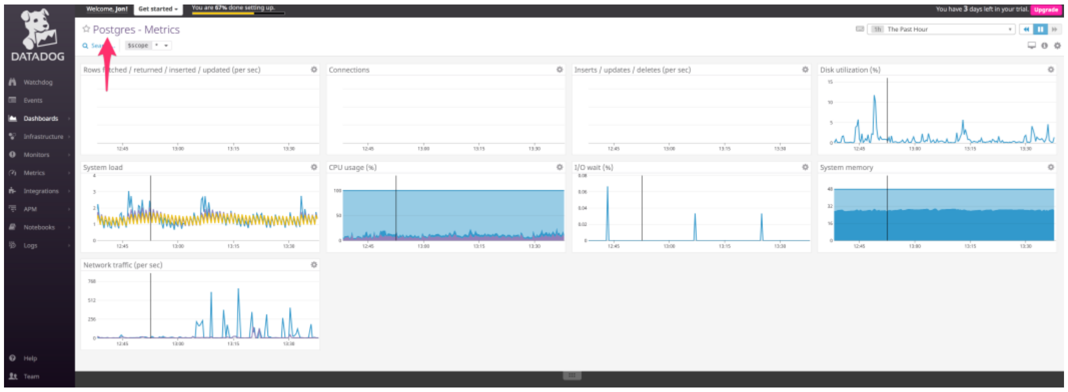
PostgreSQL is an open-source, free object-relatio

This integration is working properly.

Connect your PostgreSQL server to Datadog in order to:

- Visualize your database performance.
- Correlate the performance of PostgreSQL with the rest of your applications.

Automatic dashboard created upon PostgreSQL Integration:

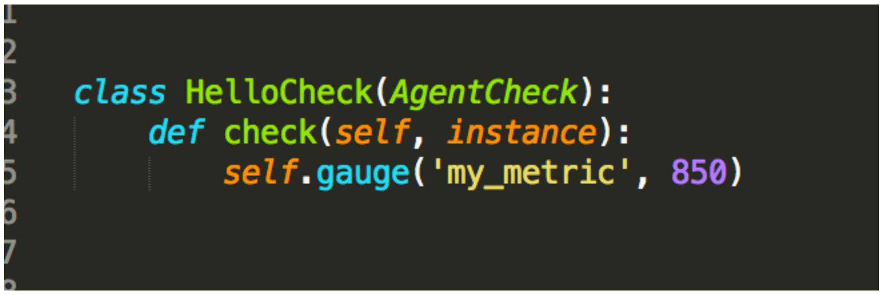


3. *Create a custom Agent check that submits a metric named my_metric with a random value between 0 and 1000.*

   Custom checks are well suited to collect metrics from custom applications or unique systems. There are many custom applications and unique use cases at WE.   When you need to monitor something that isn’t covered by a standard integration, a Custom Agent Check is recommended.

   For a demonstration of this capability, we employed a simple python script placed in /opt/datadog-agent/etc/checks.d, called hello.py.

   The code to create “my_metric” within hello.py:

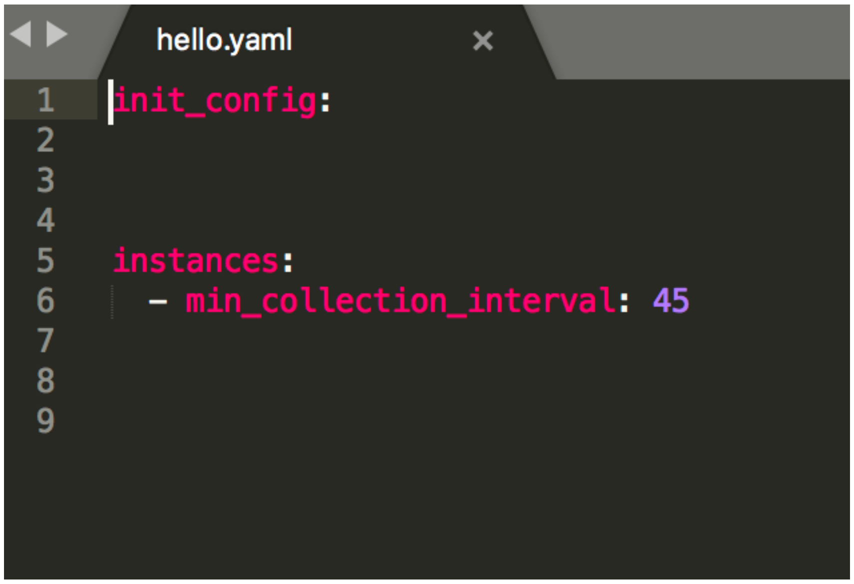


```python

class HelloCheck(AgentCheck):
    def check(self, instance):
        self.gauge('my_metric', 850)


```

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

   The hello.py script referened above requires an accompanying configuration file - /opt/datadog-agent/etc/conf.d/hello.yaml.

   “min_collection_interval” can be set to enforce the 45 second submission requirement.



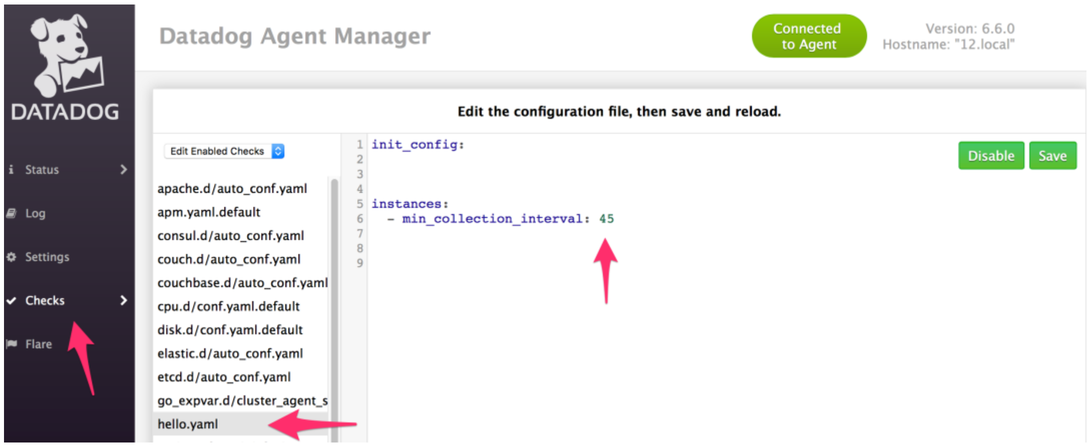
```
init_config:



instances:
  - min_collection_interval: 45
```

5. *Bonus Requirement: Can you change the collection interval without modifying the Python check file you created?*

   There may be circumstances where certain users don’t have file-level access to agent configuration files, but still need to make changes.

   The Datadog agent also comes equipped with a web-based UI – the Datadog Agent Manager.

   By selecting Checks -> hello.yaml, a user can make necessary changes, such as the collection interval, without having to modify the python check file directly.



```
init_config:



instances:
  - min_collection_interval: 45
```

More information on the Datadog Agent and Agent Manager are available here: https://docs.datadoghq.com/agent/basic_agent_usage/?tab=chefcookbook

Technical Requirement: Visualizing Data

While Datadog provides a wealth of ways to get data in, it's just as important to visualize and get value from this data.

Datadog allows users to mix and match metrics and events from all your apps, hosts, containers, and services, all from a clean, web-based UI.

WE has hundreds of users that will be consuming dashboards and visualizations, all of varying degrees of technical prowess.

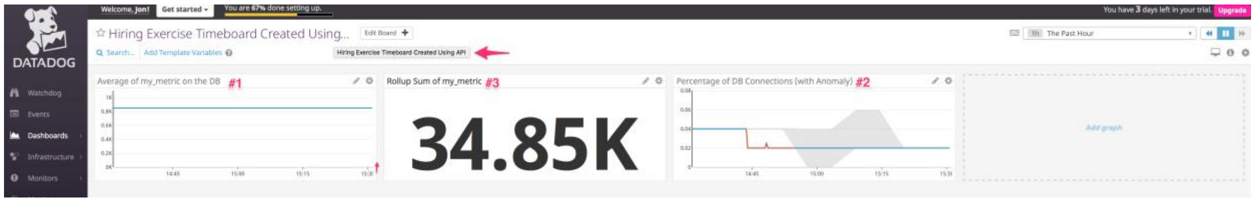
In addition, WE has a rich developer community that wishes to create rich visualizations at the API level. In other words, create dashboards programmatically, without having to use the standard Datadog web UI to do so.

As such, there are some technical requirements to meet.

Utilize the Datadog API to create a Dashboard Timeboard that contains:

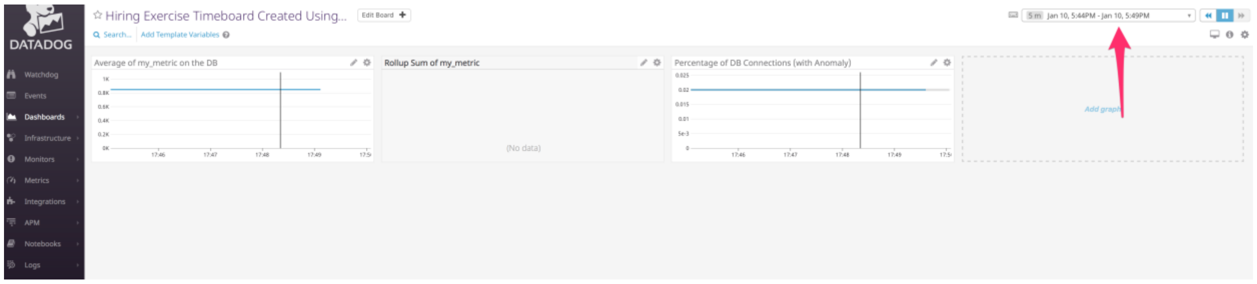
1. Your custom metric scoped over your host.
2. Any metric from the Integration on your Database with the anomaly function applied.
3. Your custom metric with the rollup function applied to sum up all the points for the past hour into one bucket

   This dashboard shows those very graphs, yet it was not created using the web UI.   It was created hitting the Datadog APIs directly:



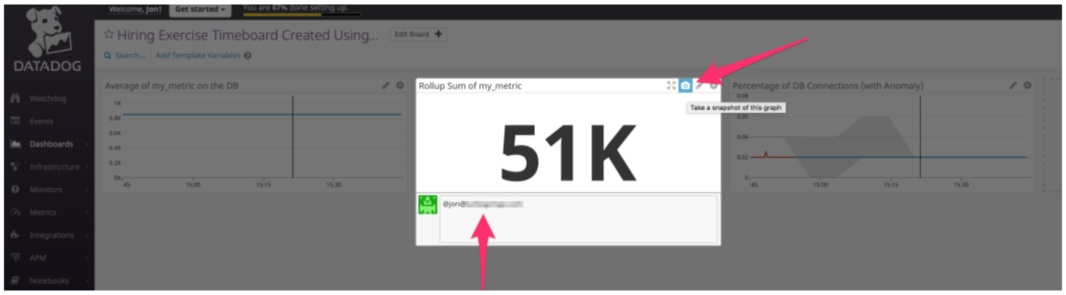
Welcome, Jon!
Get started
You are 67% done setting up.
You have 3 days left in your trial.
Upgrade
DATADOG
Hiring Exercise Timeboard Created Using...
Edit Board
Search...
Add Template Variables
Hiring Exercise Timeboard Created Using API
1h The Past Hour
Watchdog
Events
Dashboards
Infrastructure
Monitors
Average of my_metric on the DB #1
Rollup Sum of my_metric #3
34.85K
Percentage of DB Connections (with Anomaly) #2
Add graph

4. To illustrate even more flexibility, here's that same dashboard showing the last 5 minutes of data, rather than the last hour:

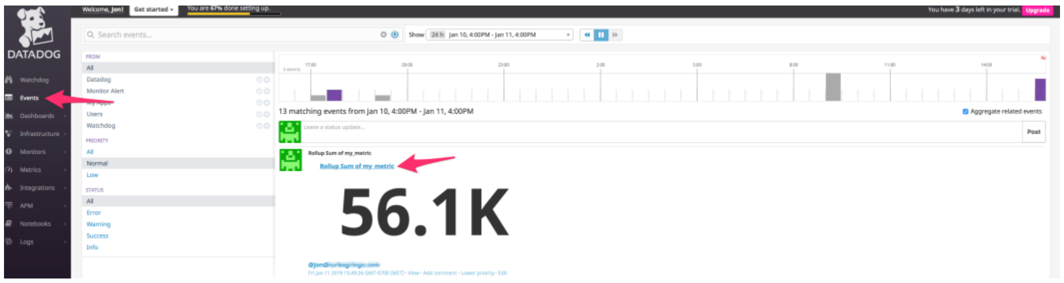


5. Take a snapshot of this graph and use the @ notation to send it to yourself.

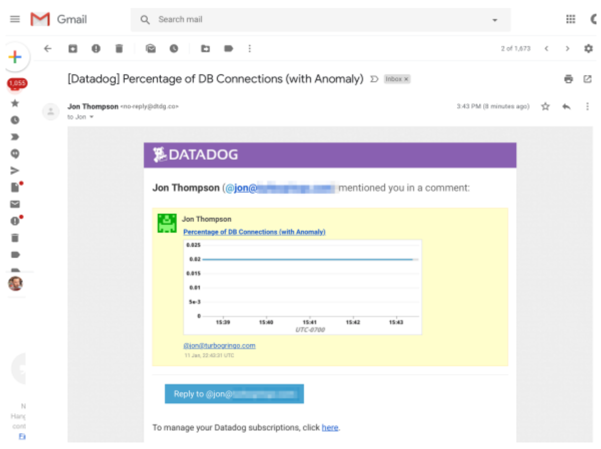
   By clicking on the small camera icon above any graph, users can annotate these moments in time and send them to themselves or other users using the @<user> feature.


Hiring Exercise Timeboard Created Using...
Average of my_metric on the DB
Rollup Sum of my_metric
51K
Take a snapshot of this graph
Percentage of DB Connections (with Anomaly)
@jon@

   These snapshots will then show up in the Events screen…



   And in the user's email, if desired:



Gmail

Search mail

2 of 1,673

[Datadog] Percentage of DB Connections (with Anomaly) Inbox ×

Jon Thompson <no-reply@dtdg.co>
to Jon

3:43 PM (8 minutes ago)

DATADOG

Jon Thompson (@jon@[redacted]) mentioned you in a comment:

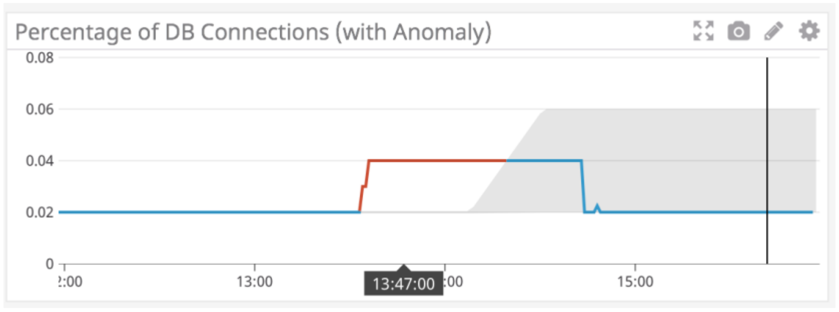
Jon Thompson
Percentage of DB Connections (with Anomaly)

@jon@turbogringo.com
11 Jan, 22:40:31 UTC

Reply to @jon@[redacted]

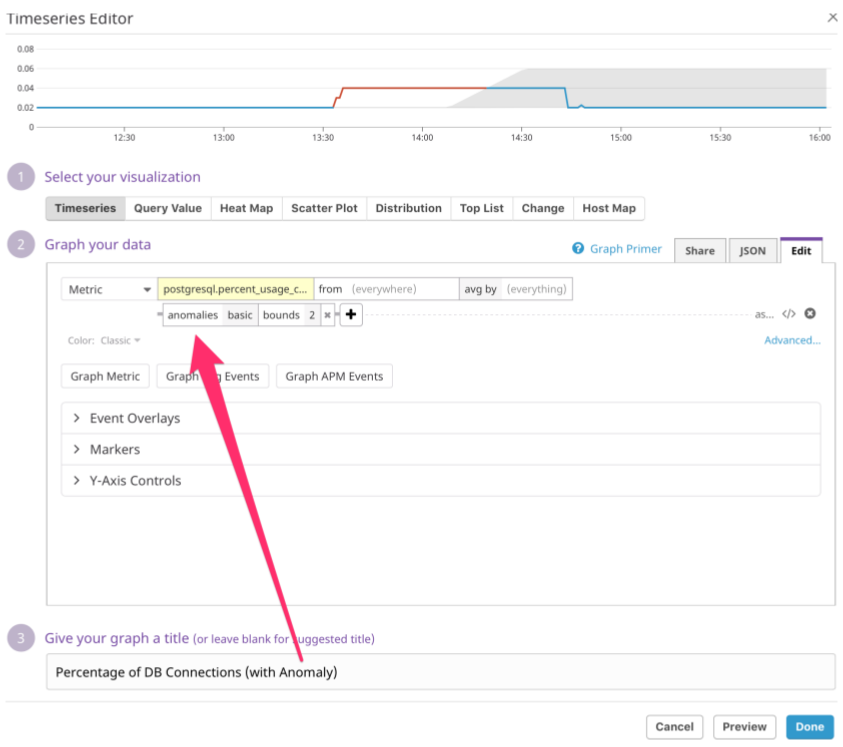
To manage your Datadog subscriptions, click here.

6. Bonus Requirement**:** What is the Anomaly graph displaying?



There are a number of configuration options when creating graphs, either in the web UI or via the API.

Looking at the configuration of this graph using the web UI, you'll see how it was created, including the options for anomaly detection:


Timeseries Editor

1 Select your visualization

Timeseries | Query Value | Heat Map | Scatter Plot | Distribution | Top List | Change | Host Map

2 Graph your data

Metric: postgresql.percent_usage_c... from (everywhere) avg by (everything)

anomalies basic bounds 2

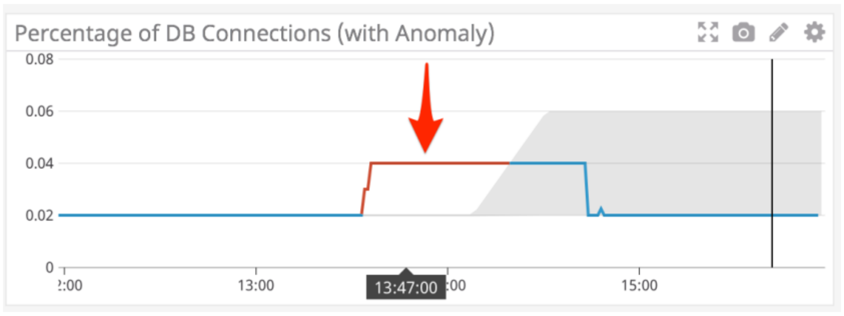
Color: Classic

3 Give your graph a title (or leave blank for suggested title)

Percentage of DB Connections (with Anomaly)

The graph is showing the percentage of database connections used (as a fraction of the overall connections allowed). In this specific case, this is showing the last 4 hours of data. The more historical data you have, the more accurate and reliable the anomaly detection will be.

Anomaly detection is not just about showing you what's normal—it's also about surfacing what's not. Here we see an unexpected increase of DB Connections, which is quickly flagged as an anomaly in red.


Percentage of DB Connections (with Anomaly)

Anomaly detection in Datadog takes two parameters:

- The algorithm (*basic, agile,* or *robust*)
- The bounds for that algorithm

Datadog automatically sets the appropriate algorithm for you after analyzing your chosen metric. However, you can still change these parameters manually.

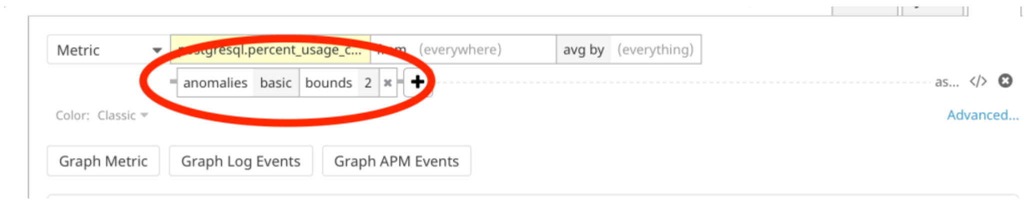
In our case, this wasn't a long-term POV – we only had about a week's worth of data. As a result, only the "Basic" algorithm could be used. The other two – "Agile" and "Robust" require more historical data.

What does "Basic" mean? From our excellent blog on Datadog anomaly detection:

Basic uses a simple lagging rolling quantile computation to determine the range of expected values. It adjusts quickly to changing conditions but has no knowledge of seasonality or long-term trends.

What about the bounds setting?

The bounds parameter determines the tolerance of the anomaly detection algorithm. You can think of these bounds as deviations from the predicted timeseries value. For most timeseries, setting the bounds to 2 (like ours) or 3 will capture most "normal" points.


Metric postgresql.percent_usage_c... from (everywhere) avg by (everything)

anomalies basic bounds 2

We were happy to demonstrate our anomaly detection capabilities as part of this POV.

Feel free to read more about it at our blogs site: https://www.datadoghq.com/blog/introducing-anomaly-detection-datadog/

Technical Requirement: Monitoring Data

It’s one thing to get data in, it’s another to visualize that data, but how about using Datadog to monitor everything on your behalf?

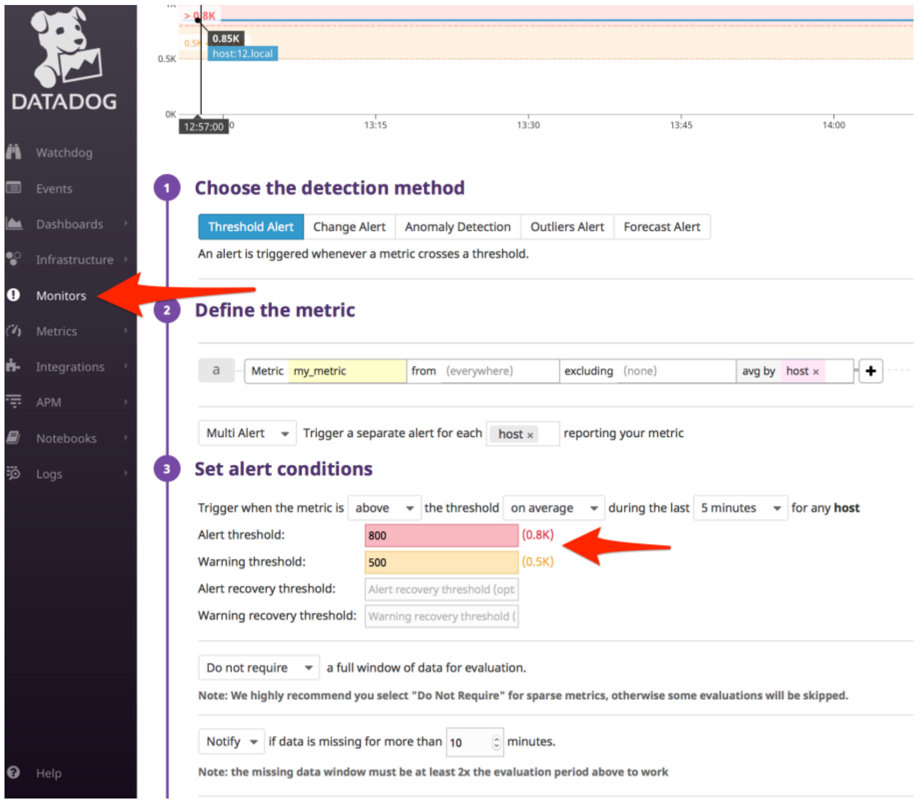
Rather than having to log in to Datadog to see how things are going, Datadog can monitor all of your metrics, notifying you when certain conditions are met.

Our technical requirements for monitoring data are as follows:

1. *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:*

   A Warning Threshold of 500, an Alerting Threshold of 800, and also a notification if there is No Data for this query over the past 10 minutes.

   Within the Datadog web UI, the Monitors section allows you to set up the monitoring and configure all notification details.



1 Choose the detection method

Threshold Alert | Change Alert | Anomaly Detection | Outliers Alert | Forecast Alert

An alert is triggered whenever a metric crosses a threshold.

2 Define the metric

a Metric my_metric from (everywhere) excluding (none) avg by host ×

Multi Alert Trigger a separate alert for each host × reporting your metric

3 Set alert conditions

Trigger when the metric is above the threshold on average during the last 5 minutes for any **host**

Alert threshold:	800 (0.8K)
Warning threshold:	500 (0.5K)
Alert recovery threshold:	Alert recovery threshold (opt
Warning recovery threshold:	Warning recovery threshold (

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.

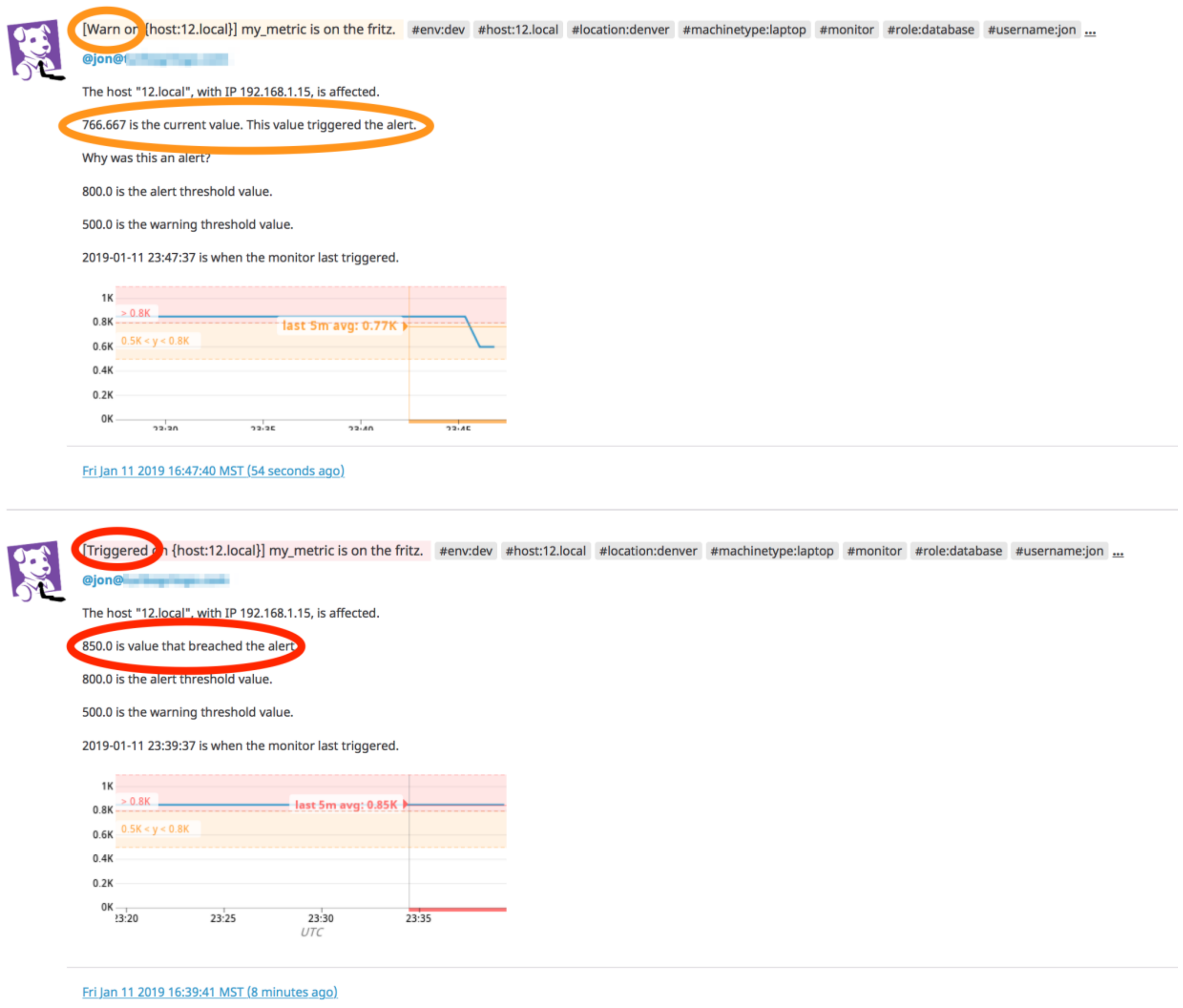
Notify if data is missing for more than 10 minutes.

Note: the missing data window must be at least 2x the evaluation period above to work

2. *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.*

Below you’ll see a screenshot of these web UI notifications, with the relevant items highlighted:



[Warn on {host:12.local}] my_metric is on the fritz. #env:dev #host:12.local #location:denver #machinetype:laptop #monitor #role:database #username:jon ...

@jon@[redacted]

The host "12.local", with IP 192.168.1.15, is affected.

766.667 is the current value. This value triggered the alert.

Why was this an alert?

800.0 is the alert threshold value.

500.0 is the warning threshold value.

2019-01-11 23:47:37 is when the monitor last triggered.

Fri Jan 11 2019 16:47:40 MST (54 seconds ago)

[Triggered on {host:12.local}] my_metric is on the fritz. #env:dev #host:12.local #location:denver #machinetype:laptop #monitor #role:database #username:jon ...

@jon@[redacted]

The host "12.local", with IP 192.168.1.15, is affected.

850.0 is value that breached the alert.

800.0 is the alert threshold value.

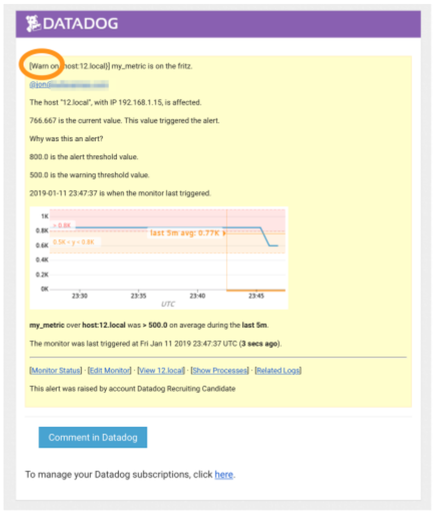
500.0 is the warning threshold value.

2019-01-11 23:39:37 is when the monitor last triggered.

Fri Jan 11 2019 16:39:41 MST (8 minutes ago)

And email notifications…

Warning:



DATADOG

[Warn on {host:12.local}] my_metric is on the fritz.

@jon@[redacted]

The host "12.local", with IP 192.168.1.15, is affected.

766.667 is the current value. This value triggered the alert.

Why was this an alert?

800.0 is the alert threshold value.

500.0 is the warning threshold value.

2019-01-11 23:47:37 is when the monitor last triggered.

my_metric over **host:12.local** was **> 500.0** on average during the **last 5m**.

The monitor was last triggered at Fri Jan 11 2019 23:47:37 UTC (**3 secs ago**).

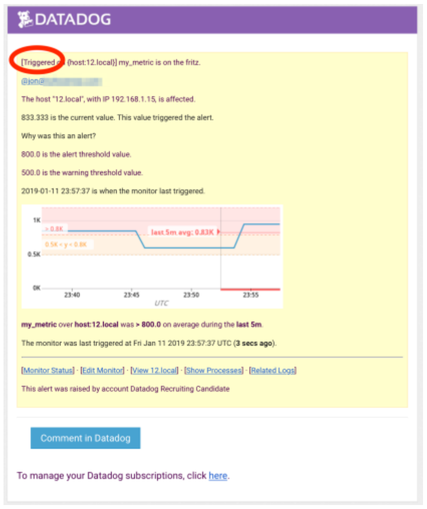
[Monitor Status] · [Edit Monitor] · [View 12.local] · [Show Processes] · [Related Logs]

This alert was raised by account Datadog Recruiting Candidate

Comment in Datadog

To manage your Datadog subscriptions, click here.

Triggered Alert:



DATADOG

[Triggered on {host:12.local}] my_metric is on the fritz.

@jon@[redacted]

The host "12.local", with IP 192.168.1.15, is affected.

833.333 is the current value. This value triggered the alert.

Why was this an alert?

800.0 is the alert threshold value.

500.0 is the warning threshold value.

2019-01-11 23:57:37 is when the monitor last triggered.

my_metric over **host:12.local** was **> 800.0** on average during the **last 5m**.

The monitor was last triggered at Fri Jan 11 2019 23:57:37 UTC (**3 secs ago**).

[Monitor Status] · [Edit Monitor] · [View 12.local] · [Show Processes] · [Related Logs]

This alert was raised by account Datadog Recruiting Candidate

Comment in Datadog

To manage your Datadog subscriptions, click here.

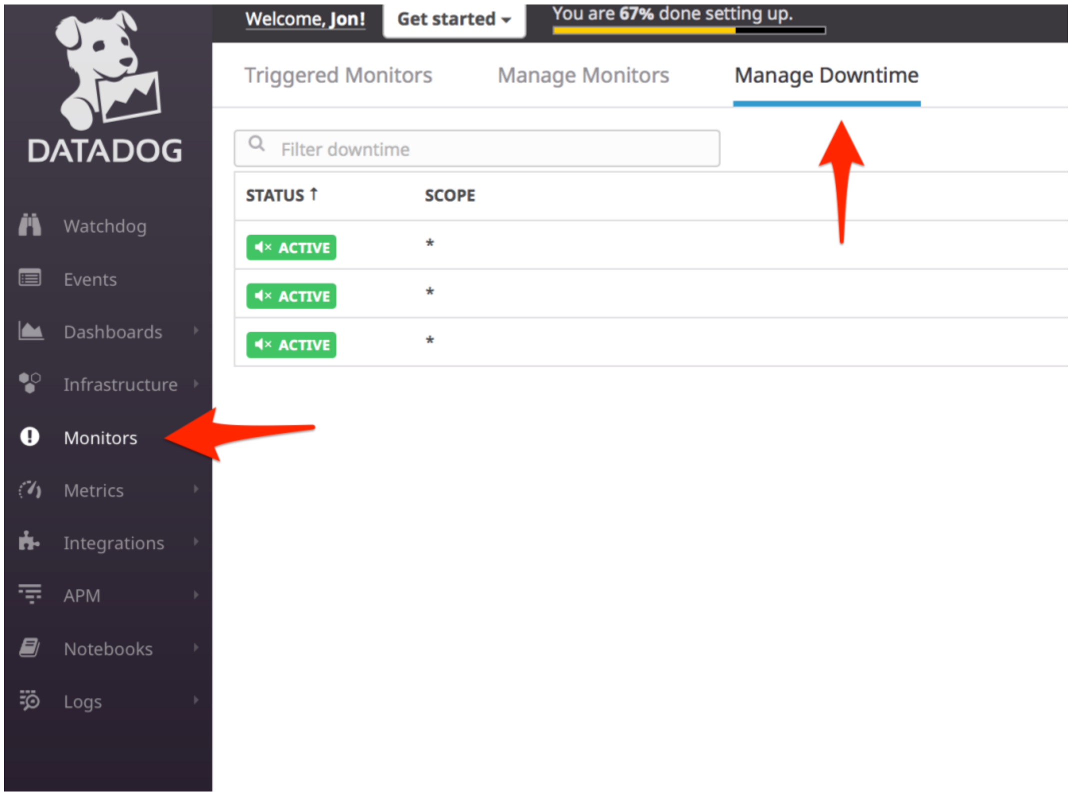
3. *Bonus Requirement: Since this monitor is going to alert pretty often, you don't want to be alerted when you are out of the office. Set up two scheduled downtimes for this monitor:*

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

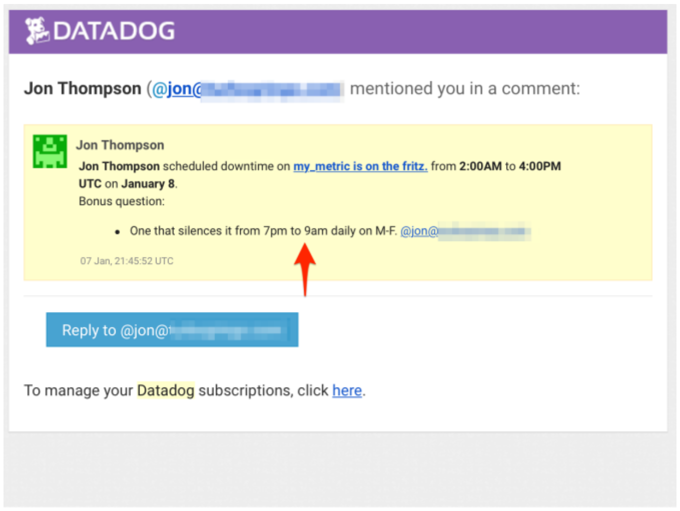
   One that silences it all day on Sat-Sun

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

   Datadog makes it easily to quickly account for scheduled downtimes. These prevent you from being flooded with alerts and notifications during times of expected outages, maintenance, etc.



When you schedule downtimes you are automatically sent an email with details of the specified downtimes. Here are two emails, one for each downtime:



Jon Thompson (@jon@[redacted] mentioned you in a comment:

Jon Thompson

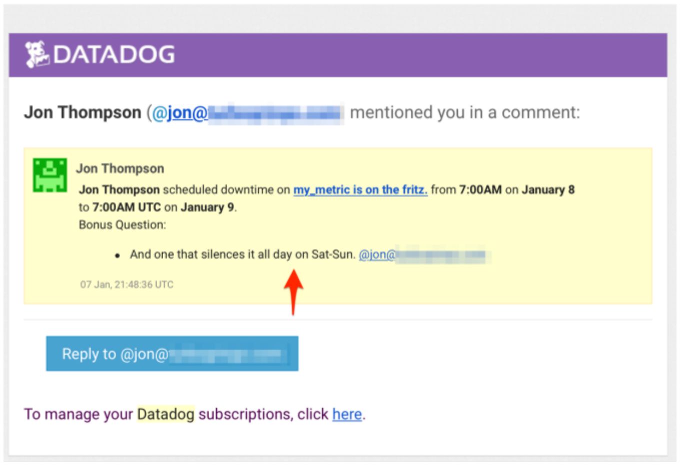
Jon Thompson scheduled downtime on **my_metric is on the fritz.** from **2:00AM** to **4:00PM UTC** on **January 8**.
Bonus question:

- One that silences it from 7pm to 9am daily on M-F. @jon@[redacted]

07 Jan, 21:45:52 UTC

Reply to @jon@[redacted]

To manage your Datadog subscriptions, click here.



Jon Thompson (@jon@[redacted] mentioned you in a comment:

Jon Thompson

Jon Thompson scheduled downtime on **my_metric is on the fritz.** from **7:00AM** on **January 8** to **7:00AM UTC** on **January 9**.
Bonus Question:

- And one that silences it all day on Sat-Sun. @jon@[redacted]

07 Jan, 21:48:36 UTC

Reply to @jon@[redacted]

To manage your Datadog subscriptions, click here.

For more information on Datadog Monitors, please see the following link: https://docs.datadoghq.com/monitors/

Wayne Enterprises employs dozens of custom applications, both internally and customer-facing.

These applications are mission-critical and need to be monitored – both from an infrastructure (CPU, disk, memory, etc) perspective as well as the latency experienced by end users of these applications.

A technical requirement of this POV was to monitor one of these applications and the server from which it runs, in this case – a python application (“ddflask_jon.py”) being served from a Linux VM.

Note: There were a number of technical challenges we faced during this Requirement. The macOS workstations and server boxes both had networking issues of some sort, still unresolved.

When running both ddtrace or manually instrumenting the application, we received the following error, on both macOS machines:

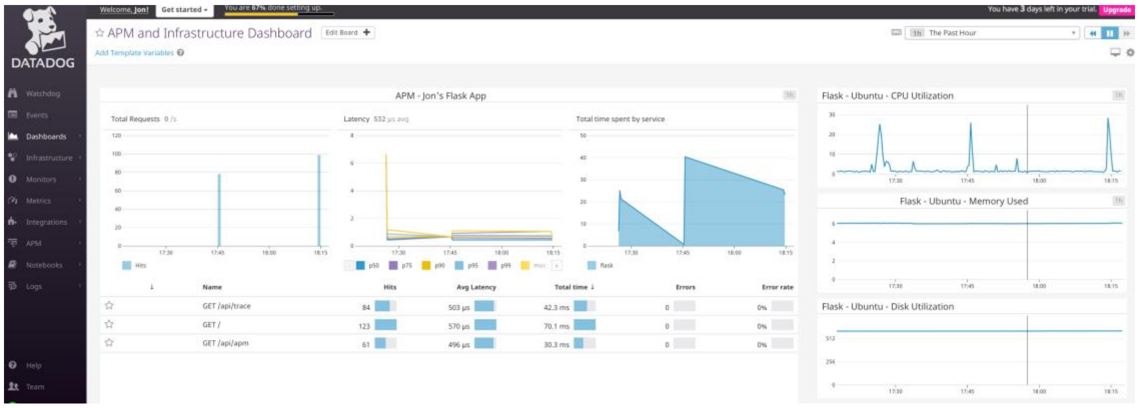
```
2019-01-07 13:25:39,700 - ddtrace.writer - ERROR - cannot send spans to localhost:8126: [Errno 61] Connection refused
```

We tried what seemed obvious – check for local firewalls, port conflicts, changing the APM port in the Agent configuration file, though nothing resolved that error.

Rather than spend too many cycles of the POV diagnosing this issue, when most applications at WE *run on Linux*, we:

- Installed the Datadog agent on a fresh Ubuntu Linux VM
- Created a simple python application
- Instrumented it with Datadog (see ddflask_jon.py)
- Instantly saw APM metrics flooding the Datadog APM Services screen!

Here’s a screenshot of a single Datadog Dashboard displaying both APM and Infrastructure Metrics:


APM and Infrastructure Dashboard
APM - Jon's Flask App
Total Requests
Latency
Total time spent by service
Flask - Ubuntu - CPU Utilization
Flask - Ubuntu - Memory Used
Flask - Ubuntu - Disk Utilization
The Past Hour

Name	Hits	Avg Latency	Total time	Errors	Error rate
GET /api/trace	84	503 μs	42.3 ms	0	0%
GET /	123	570 μs	70.1 ms	0	0%
GET /api/apm	61	496 μs	30.3 ms	0	0%

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

In this case, “Flask” is our Service. A service is a set of processes that do the same job. It could also be something like “Database” or “Webapp”.

The Resources – “/”, “/api/trace”, and “/api/apm”, are particular actions for a service.

This is a simple python application with really only three pages to serve up, but they are each considered individual resources and are reflected as such in the Datadog APM UI.

For more information on Datadog APM, please consult this link: https://docs.datadoghq.com/tracing/visualization/

Final Requirement: Get Creative!

It’s been a pleasure working with the technical team at Wayne Enterprises. They’re just getting their feet wet and beginning to grasp all the possibilities of what is possible with Datadog.

What else can you do? There are all sorts of different (think non-IT) types of data out there, and more than one way to get it into Datadog.

1) The Datadog Agent ships with DogStatsD so you can submit custom metrics via the Agent’s non-blocking API functions for many programming languages.

2) The Datadog Agent allows for the creation of custom integrations via plugins to the Agent. This plugin system allows the Agent to collect custom metrics on your behalf. Integrations which are contributed back to the Datadog Agent convert to standard metrics.

3) Datadog also has a full-featured API that you can send your metrics to—either directly over HTTP or with a language-specific library.

So… how about Datadog @ Home?

Stop and think about all the potential valuable and insightful data sources in your home!

IOT Household Items: Nest/Dropcams, Garage Doors, Alarm Systems, Amazon Alexa…

Many of these devices expose performance and/or other metrics via network wire data, http, udp, or tcp. Remember the above, so many ways to get data into Datadog, we just need to know where to get it!

Household Media: Netflix, Pandora, Spotify…

Many of these services also expose usage data via a number of methods, similar to the above. Who’s watching what? When? How much?

Personal Health: Google Fit, Withings, FitBit, Garmin, Strava…

DATADOG

How about correlating personal fitness goals with Netflix consumption? My weight goes up… is my TV watching possibly contributing?

<u>Your Car</u>: Automatic dongle

How about collecting your driving metrics directly from your car? Datadog can help improve your driving!   Automatic makes many data points available via their REST API, so let's get it into Datadog!

<u>That's just scratching the surface</u>. We'd love to share more information once Wayne Enterprises embraces Datadog at scale and begins to deploy. We're looking forward to a lasting partnership in 2019 and beyond. Thank you for your consideration.