Prerequisites - Setup the environment

- Utilize macOS Sierra version 10.12.6 OS to complete this exercise.
- Sign up for Datadog and Agent (v6.2.0) has been installed.
- Login on https://app.datadoghq.com/account/login
- On Integration -> Agent -> Mac OS X, follow the instruction and successfully install the Agent (v6.2.0).

Recommendation: The exercise states that any OS can be utilized although it recommends using Vagrant Ubuntu Datadog to avoid dependency issues.

However, as long as OS list on Integration -> Agent, this means that Datadog will be able to support them and there should not be any OS dependency issue. Otherwise it should clearly state that Datadog does not support OS with dependency issues, and list down those OSs with no dependency issues.

Collecting Metrics:

- Add tags in the Agent config file and kindly refer to a screenshot of my host and its tags for on the Host Map page in Datadog SaaS.

  The agent configuration file datadog.yaml on /opt/datadog-agent/etc was modified as such to include tags:

  tags:

  - mytag

  - env:prod

  - role:database

  OR

  tags: region:australia, application:database, database:primary, role:admin, mytag:stephen

  I have tried the above two tags formats. Both are working well on Host Map page. (please refer to DD1, DD2, DD3, DD4 and DD5) after restarting the agent.

  It indicates that the agent has been restarted on DataDog event stream within a few seconds. However, it took several minutes before the tags on the host map were displaced.

- Restart the agent by executing:

```
launchctl start com.datadoghq.agent
launchctl stop com.datadoghq.agent
```

OR

```
/usr/local/bin/datadog-agent stop
/usr/local/bin/datadog-agent start
```

Both are working.

Refer to https://docs.datadoghq.com/agent/faq/agent-commands/

- Install a MySQL (8.0.12) database on my machine. (please refer to DD6)
- install the respective Datadog integration for that database without showing the matrics (please refer to DD7, DD7a, DD7b and DD7c)

After the installation MySQL 8.0.12, I was going to install Datadog integration with MySQL through Integrations -> Integrations and followed the steps on Configuration tab but it returned with an error:

```
mysql> GRANT REPLICATION CLIENT ON *.* TO 'datadog'@'localhost' WITH
MAX_USER_CONNECTIONS 5;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near
'MAX_USER_CONNECTIONS 5' at line 1
```

This works after removing `MAX_USER_CONNECTIONS`.

```
mysql> GRANT REPLICATION CLIENT ON *.* TO 'datadog'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

- The YAML file called conf.yaml on /opt/datadog-agent/etc/conf.d/mysql.d has been created:

```
init_config:

instances:
       - server: localhost
       user: datadog
       pass: pfRM71RE2N3o?w58MifV3kOQt
       tags:
       - optional_tag1
       - optional_tag2
       options:
       replication: 0
       galera_cluster: 1
```

- Execute `datadog-agent status with an Error:` the datadog credential on the conf.yaml is unable to login MySQL through the Agent written by Python

```
mysql
    -----
```

```
Total Runs: 1
Metrics: 0, Total Metrics: 0
Events: 0, Total Events: 0
Service Checks: 1, Total Service Checks: 1
Average Execution Time : 9ms
Error: (1045, u"Access denied for user 'root'@'localhost' (using password: NO)")
```

But with the datadog credential, I was able to login MySQL directly:

```
Evelyns-MacBook-Air:~ evelyn$ mysql -udatadog -pfRM71RE2N3o?w58MifV3kOQt
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 119
Server version: 8.0.12 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

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

  The Python file called my_metric.py on /opt/datadog-agent/etc/checks.d has been created:

  from checks import AgentCheck

  from random import randint

  class MyCheck(AgentCheck):

  def check(self, instance):

  self.gauge('my_metric', randint(0, 1000))

  which refer to class HelloCheck on https://docs.datadoghq.com/developers/agent_checks/

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

  The YAML file called my_metric.yaml on /opt/datadog-agent/etc/conf.d has been created
  init_config:

  instances:
  - min_collection_interval: 45

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

  It is obvious that the collection interval can be changed by modifying the min_collection_interval on the YAML configuration file on /opt/datadog-agent/etc/conf.d not Python check file.

Recommendation:

After modifying the agent configuration file datadog.yaml and restarted the agent, I expect that the system will be updated. However, I am unable to determine why my modification was not working and it took several minutes to display the tags on the host map. The documents should state that after the modified agent, this process would require a considerable period of time to refresh the tags and the system needs to indicate that 'tags updating' is in progress.

After the agent restarted, it will show the agent start on Event Stream but it would make more sense to display both the agent starting and stopping especially when the agent has been stalled for a while.

Furthermore, when the agent started again after stopping for a while, on the dashboards, it will show values by connecting two starting points which is misinterpreted, since those values was unknown (please refer to DD8).

It is preferable for DataDog to provide an UI tool that will execute configuration steps to perform the integration, and automatically generate Agent check files and configuration YAML file automatically, rather than performing manual steps as above.

With security connections, it could complete adding Tags and create an Agent check directly through DataDog SaaS service.

Visualizing Data:

(Not completed)

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

The anomaly graph applies algorithms, artificial intelligent or machine learning that compares its past and present behaviour of performance metric, for instance if there a CPU usage showing a regular period cycle in contain upper and lower values with similar and or frequent patterns, when values go outside of a certain boundary that is different from most approaches and look like unusual pattern, there is an anomaly detection will detect those unexpected values and identify abnormal behaviour.

Monitoring Data

(Not completed) Monitors -> Triggered monitors

- Send you an email whenever the monitor triggers.

Since the perilous step not complete, it cannot send an email with the monitor triggers, rather than practice sending an email triggered by the event when the agent started (Please refer to DD9 and DD10).

Collecting APM Data:

(Not completed)

APM

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

A Service can be defined by a verb, e.g. it can perform to update a user name or validate customer credit history. In a web application, it can provide a number of service in a single application for Web, Administration, Database etc. And these services can be defined by users when instrumenting their application.

A Resource is can be defined by a noun e.g. it can represent the user records or transaction history, it can describe the data provided by the resource.

From a database perspective, a Resource could be as a SQL query like “select * from users where id = ‘ID0001’”; and in REST API, a Resource can be a singleton or a collection**,** their resource URIs can be designed as below:

http://api.com/resource/users/
http://api.com/resource/users/{id}

Because of a Java software devolvement background, I will use another different analogy based on Enterprise Java Bean (EJB):

We can think of services as stateless session beans and resources as entity beans.

A Services can be viewed as session beans – serve as controllers allowing execution of a required operation, regardless of the resource.

For example, a human resource service may take the user id and the user name, this service can update the user name for any of the existing users.

A Resources is like entity beans – serve as a data access mechanism for a given instance of given data type.

For example, if we want to update a user name, it is necessary to find a resource representing this user and then update its name field accordingly.

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?

We normally perform single integration or technology with Datadog, for instance Datadog can apply AI (Artificial Intelligent) and Machine Learning technology with integrating with signifai, we can monitor IoT (Internet of Thing) devices with Datadog (e,g, AWS IoT), and we can also apply a virtual assistant (e.g. Google assistance and Amazon Alexa) to Datadog.

It would be interesting and could be a creative idea with a combination of more integrations and technology in one go into Datadog to produce a multiple-effect.

Recommendation:

For Datadog, there are a certain number of manual steps required for integrations and configurations. But organizations need to get integrations up and running quickly, and speed of implementation with those solutions add value to the business,

I have evaluated Application Performance Management called Dynatrace, after installing its OneAgent on my local host, it detects Oracle database that has been installed, and be able to display relevant performance matrix directly on Dynatrace SaaS service without any configuration and integration steps required.

Furthermore, with Dell Boomi (as cloud-based integration service), it is easy for integrations and configurations through UI without programming. It is code-less and simplified, and may also give a direction how Datadog could be improved in the future.