Assignment 8: Datadog Dashboard

Project Description

In this project, I integrated Datadog monitoring with a MySQL database hosted on my Ubuntu virtual machine. I installed and configured the Datadog Agent, created a MySQL database that logs different types of queries, and visualized system metrics (CPU, memory, disk usage) alongside MySQL query metrics using a custom Datadog dashboard. The goal was to observe how database load affects system performance and to practice real-time monitoring.

Step 1: Install Datadog

1. **Create a Datadog Account (free trial):**

   https://www.datadoghq.com

   (https://www.datadoghq.com/lpg/?utm_source=Advertisement&utm_medium=GoogleAdsNon1stTier&utm_campaign=GoogleAdsNon1stTier-BrandCV&utm_content=Brand&utm_keyword=%2Bdatadog&utm_matchtype=b&utm_campaignid=9551169254&utm_adg

2. **In your Ubuntu terminal, install the Datadog Agent:**

   - `sudo apt install datadog-agent`
   - `DD_API_KEY=your_datadog_api_key DD_SITE="datadoghq.com" bash -c "$(curl -L https://install.datadoghq.com/scripts/install_script_agent7.sh)"`

   ⚠️ Replace `your_datadog_api_key` with your real API key found in your Datadog account under **Integrations > APIs**.

3. **Start/Stop the Datadog Agent:**

```
sudo systemctl start datadog-agent
sudo systemctl stop datadog-agent
```

Step 2: Install MySQL

1. **Search and install MySQL:**

```
sudo apt-cache search mysql-server
sudo apt install mysql-server
```

2. **Run the secure installation:** `sudo mysql_secure_installation`
3. **Access the DB:** `sudo mysql`

Step 3: MySQL User Creation

1. **Create the user:** `CREATE USER 'yourname'@'host' INDENTIFIED BY 'password';`
2. **Grant privileges to user:** `GRANT ALL PRIVILEGES ON` `.` `TO 'eduardo'@'localhost' IDENTIFIED BY 'password';`
3. **Flush other privileges and close MySQL:**

```
FLUSH PRIVILEGES;
exit
```

Step 4: MySQL Database Creation

1. **Log in as the created user:** `mysql -u yourname -p`
2. **Create the `queries` database:**

```
CREATE DATABASE queries;
SHOW DATABASES;
```

3. **Confirm the database exists:**

```
SHOW DATABASES;
```

Expected output:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| queries            |
| sys                |
| ts                 |
+--------------------+
6 rows in set (0.01 sec)
```

4. **Insert data into tables within the database:**

```

USE queries;

Create Table: CREATE TABLE query_log (
id int NOT NULL AUTO_INCREMENT,
query_timestamp timestamp NULL DEFAULT CURRENT_TIMESTAMP,
query_type varchar(50) NOT NULL,
PRIMARY KEY (id)
)
```

5. **Confirm the table created:**

```
SHOW TABLES;
```

Expected output:

```
+-------------------+
| Tables_in_queries |
+-------------------+
| query_log         |
+-------------------+
1 row in set (0.00 sec)
```

6. **Exit:** `exit`

Step 5: Insert Initial Data

1. **Run this in terminal to insert sample rows:**

```
mysql -u yourname -p'password' --database=queries -e "
INSERT INTO query_log (query_timestamp, query_type) VALUES
(NOW(), 'SELECT'),
(NOW(), 'INSERT'),
(NOW(), 'UPDATE'),
(NOW(), 'DELETE'),
(NOW(), 'JOIN'),
(NOW(), 'TRANSACTION');"
```

2. **View data within the database:**
`mysql -u yourname -p'password' --database=queries -e "SELECT * FROM query_log;"`

   Example Output:

```
+----+---------------------+-------------+
| id | query_timestamp     | query_type  |
+----+---------------------+-------------+
|  1 | 2024-11-06 12:16:11 | SELECT      |
|  2 | 2024-11-06 12:16:11 | INSERT      |
|  3 | 2024-11-06 12:16:11 | UPDATE      |
|  4 | 2024-11-06 12:16:11 | DELETE      |
|  5 | 2024-11-06 12:16:11 | JOIN        |
|  6 | 2024-11-06 12:16:13 | TRANSACTION |
```

Step 6: Configuring Datadog

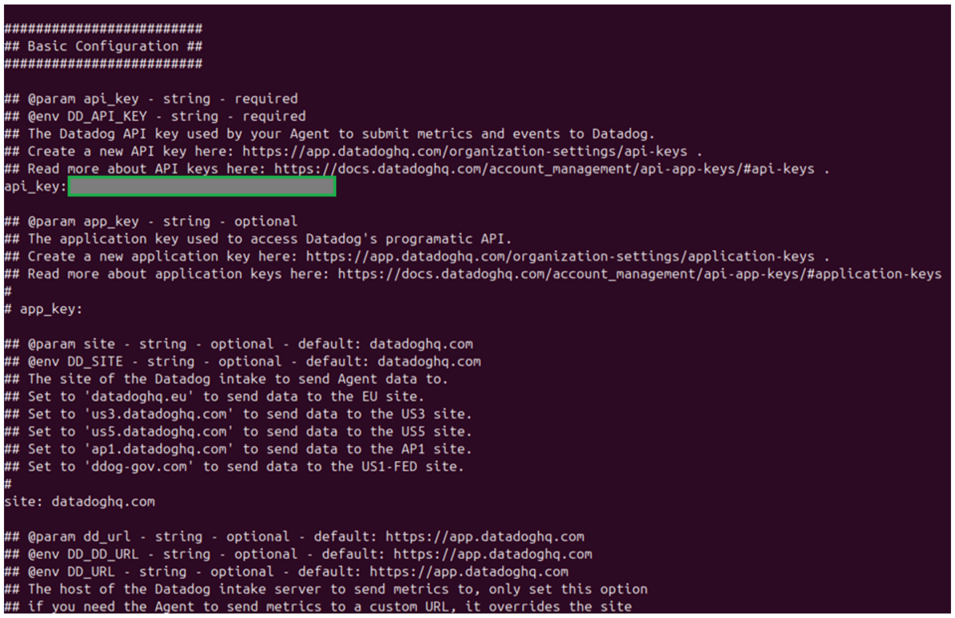
1. **Start Datadog Agent:**

```
sudo systemctl start datadog-agent
```

2. **Edit config file:**

```
cd /etc/datadog-agent/
sudo vim datadog.yaml
```

> Make sure your API key is present and valid.



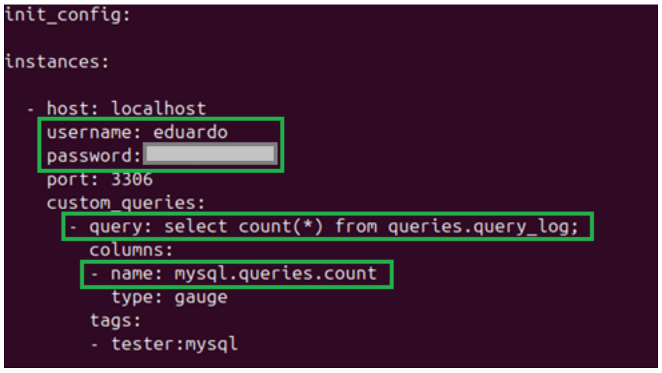
```
#########################
## Basic Configuration ##
#########################

## @param api_key - string - required
## @env DD_API_KEY - string - required
## The Datadog API key used by your Agent to submit metrics and events to Datadog.
## Create a new API key here: https://app.datadoghq.com/organization-settings/api-keys .
## Read more about API keys here: https://docs.datadoghq.com/account_management/api-app-keys/#api-keys .
api_key:[redacted]

## @param app_key - string - optional
## The application key used to access Datadog's programatic API.
## Create a new application key here: https://app.datadoghq.com/organization-settings/application-keys .
## Read more about application keys here: https://docs.datadoghq.com/account_management/api-app-keys/#application-keys
#
# app_key:

## @param site - string - optional - default: datadoghq.com
## @env DD_SITE - string - optional - default: datadoghq.com
## The site of the Datadog intake to send Agent data to.
## Set to 'datadoghq.eu' to send data to the EU site.
## Set to 'us3.datadoghq.com' to send data to the US3 site.
## Set to 'us5.datadoghq.com' to send data to the US5 site.
## Set to 'ap1.datadoghq.com' to send data to the AP1 site.
## Set to 'ddog-gov.com' to send data to the US1-FED site.
#
site: datadoghq.com

## @param dd_url - string - optional - default: https://app.datadoghq.com
## @env DD_DD_URL - string - optional - default: https://app.datadoghq.com
## @env DD_URL - string - optional - default: https://app.datadoghq.com
## The host of the Datadog intake server to send metrics to, only set this option
## if you need the Agent to send metrics to a custom URL, it overrides the site
```

3. **Create MySQL integration file:**

   This file configures access to the MySQL database. Therefore, it must be modified to fit your MySQL account, database, and query.

   ```
   cd conf.d
   vim mysql.d/conf.yaml
   ```

   Example
   `conf.yaml` content:



```
init_config:

instances:

  - host: localhost
    username: eduardo
    password:[redacted]
    port: 3306
    custom_queries:
      - query: select count(*) from queries.query_log;
        columns:
          - name: mysql.queries.count
            type: gauge
        tags:
          - tester:mysql
```

4. **Restart agent:**

```
sudo systemctl stop datadog-agent
```

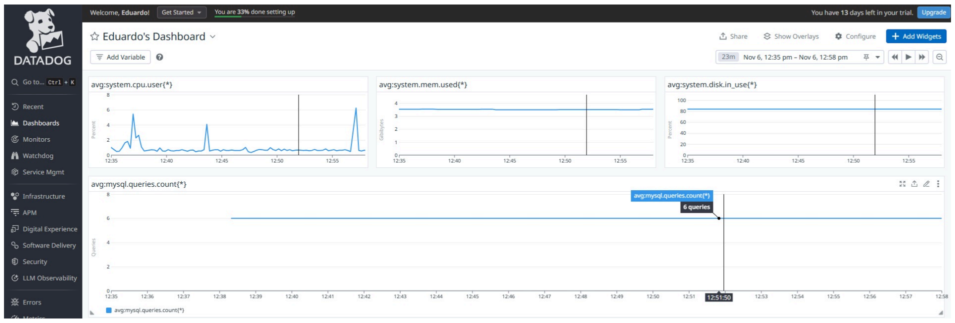
```
sudo systemctl start datadog-agent
```

5. **Check agent status:** `sudo datagog-agent status`

Step 7: Create Datadog Dashboard

1. **Go to Datadog > Dashboards > New Dashboard**
2. **Name the dashboard**
3. **Add "Timeseries" widgets for:**
    - `system.cpu.user`
    - `system.mem.used`
    - `system.disk.in_use`
    - `mysql.queries.count`

Example Datadog dashboard:

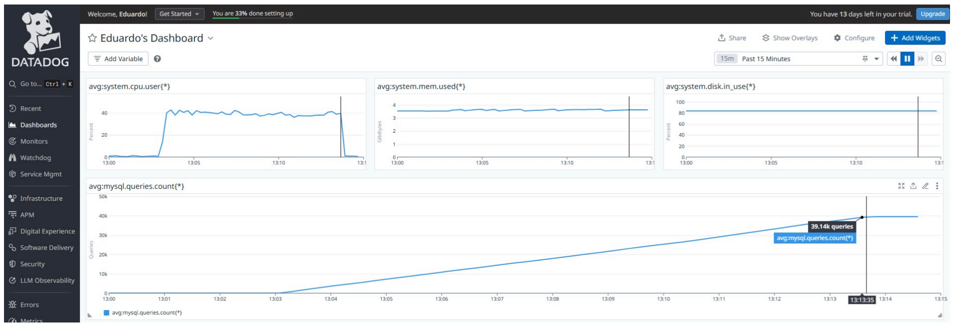


Step 8: Load Testing with a Loop

To simulate load, go back to the terminal, and run the loop command to write more data to the database, This time 16,000.

```
for i in {1..16000}; do
  mysql -u yourname -p'password' --database=queries -e "
  INSERT INTO query_log (query_timestamp, query_type) VALUES
  (NOW(), 'SELECT'),
  (NOW(), 'INSERT'),
  (NOW(), 'UPDATE'),
  (NOW(), 'DELETE'),
  (NOW(), 'JOIN'),
  (NOW(), 'TRANSACTION');"
done
```

Note: After a certain moment, I broke the loop since we can already correlate which other operating system metrics (CPU, Memory, Disk) are impacted by this load. In this case is the CPU.



Step 9: Verify Total Queries

- `mysql -u yourname -p'password' --database=queries -e "SELECT COUNT(*) FROM query_log;"`

  Example output:

```
mysql> SELECT COUNT(*) FROM query_log;
+----------+
| COUNT(*) |
+----------+
|    39461 |
+----------+
1 row in set (0.00 sec)
```
