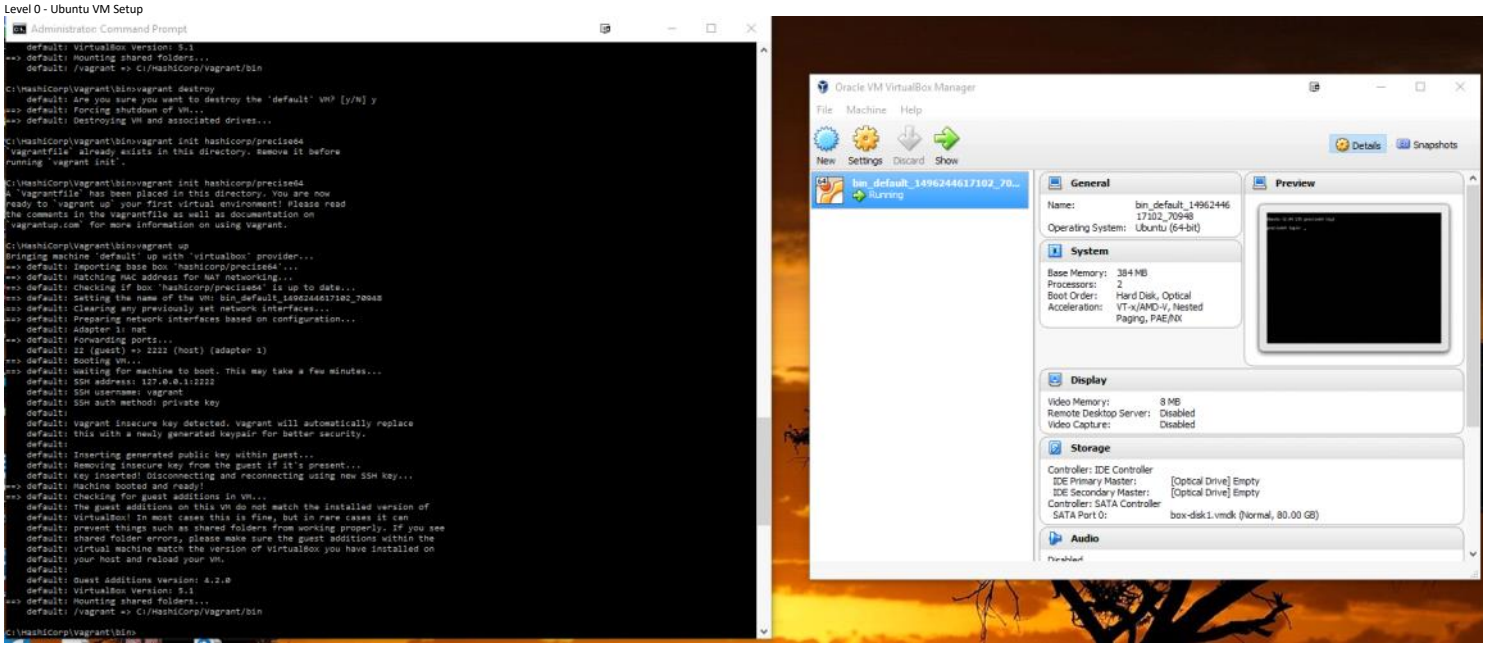
Exercise Answers

Friday, June 2, 2017 1:29 PM

Level 0 - Ubuntu VM Setup



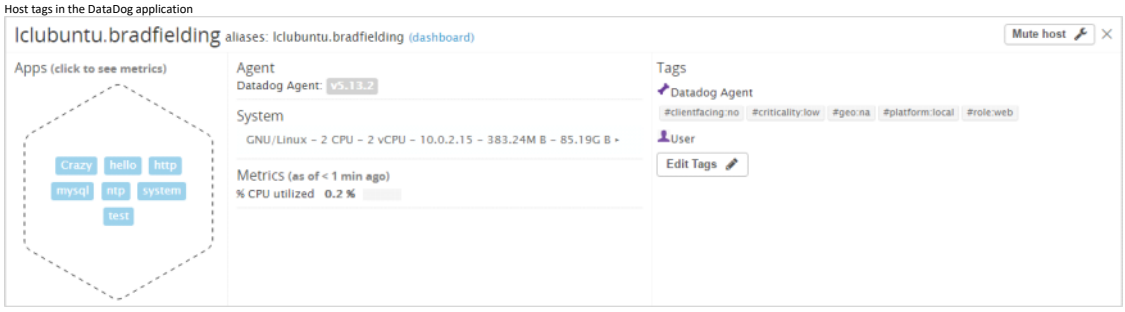
Level 1 - Collecting Data

Host Tags

Ubuntu
Tags in host config file

```
# Force the hostname to whatever you want. (default: auto-detected)
 hostname: lclubuntu.bradfielding

# Enable the trace agent.
# apm_enabled: false

# Set the host's tags (optional)
 tags: role:web, geo:na, criticality:low, clientfacing:no, platform:local
```

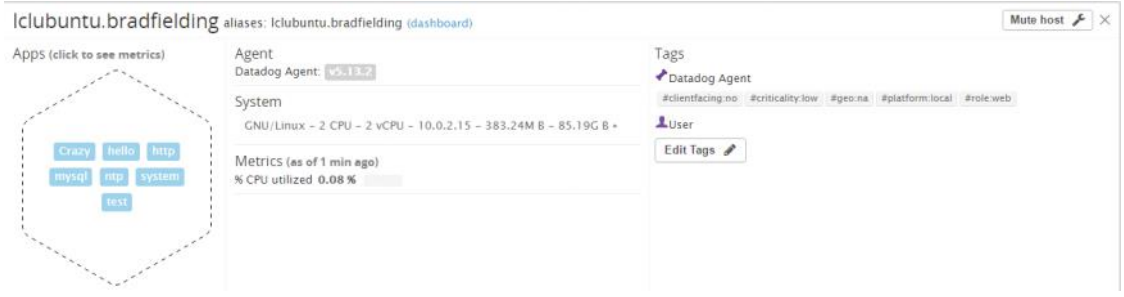
Host tags in the DataDog application



Windows
Tags in host config file

```
22
23 # Force the hostname to whatever you want. (default: auto-detected)
24  hostname: gpewinserver.bradfielding
25
26 # Set the host's tags (default: no tags)
27  tags: role:ad, geo:asiapac, criticality:high, clientfacing:no, platform:gpc
```

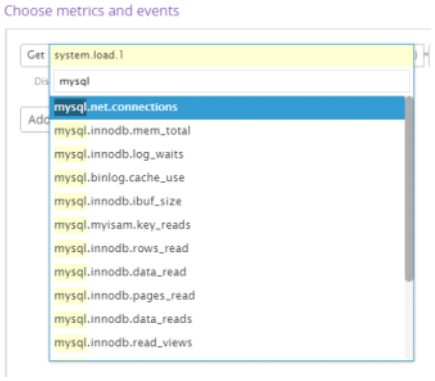
Host tags in the DataDog application



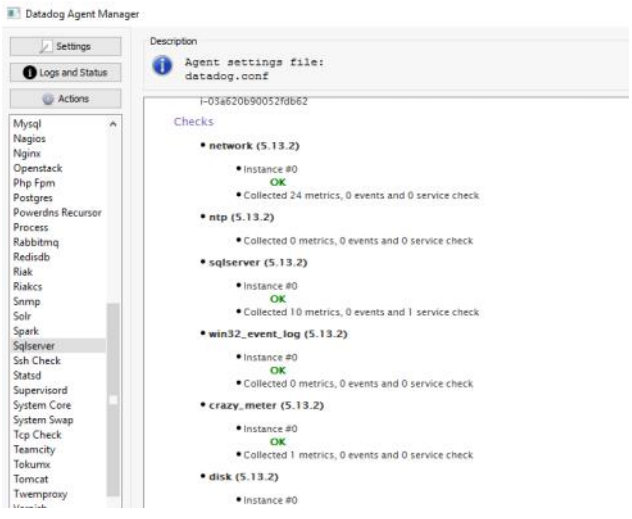
MySQL Integration
Agent Check showing MySQL success

```
Checks
======

  matt_random (5.13.2)
  --------------------
    - instance #0 [OK]
    - Collected 1 metric, 0 events & 0 service checks

  http (5.13.2)
  -------------
    - instance #0 [OK]
    - instance #1 [OK]
    - instance #2 [OK]
    - Collected 1 metric, 2 events & 0 service checks

  network (5.13.2)
  ----------------
    - instance #0 [OK]
    - Collected 15 metrics, 0 events & 0 service checks

  crazy_meter (5.13.2)
  --------------------
    - instance #0 [OK]
    - Collected 1 metric, 0 events & 0 service checks

  mysql (5.13.2)
  --------------
    - instance #0 [OK]
    - Collected 168 metrics, 0 events & 1 service check
    - Dependencies:
        - pymysql: 0.6.6.None

  ntp (5.13.2)
  ------------
    - Collected 0 metrics, 0 events & 0 service checks

  disk (5.13.2)
  -------------
    - instance #0 [OK]
    - Collected 32 metrics, 0 events & 0 service checks

  hello (5.13.2)
  --------------
    - instance #0 [OK]
    - Collected 1 metric, 0 events & 0 service checks
```

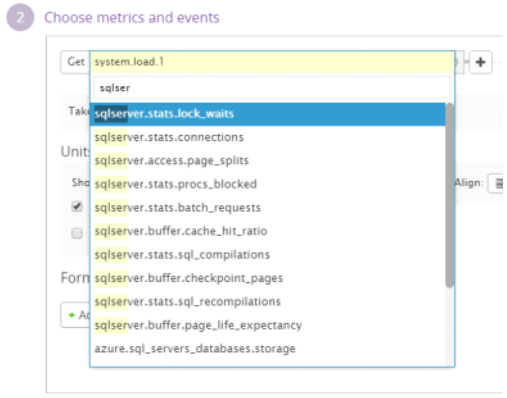
DataDog App showing available MySQL metrics



MS SQL integration
Agent Info Showing success



DataDog application showing available SQL metrics



Custom Agent Check
Check named - matt_random

```
Checks
======

  matt_random (5.13.2)
  --------------------
    - instance #0 [OK]
    - Collected 1 metric, 0 events & 0 service checks

  http (5.13.2)
  -------------
    - instance #0 [OK]
    - instance #1 [OK]
    - instance #2 [OK]
    - Collected 1 metric, 2 events & 0 service checks

  network (5.13.2)
  ----------------
    - instance #0 [OK]
    - Collected 15 metrics, 0 events & 0 service checks

  mysql (5.13.2)
  --------------
    - instance #0 [OK]
    - Collected 168 metrics, 0 events & 1 service check
    - Dependencies:
        - pymysql: 0.6.6.None

  ntp (5.13.2)
  ------------
    - Collected 0 metrics, 0 events & 0 service checks

  disk (5.13.2)
  -------------
    - instance #0 [OK]
    - Collected 32 metrics, 0 events & 0 service checks

  hello (5.13.2)
  --------------
    - instance #0 [OK]
    - Collected 1 metric, 0 events & 0 service checks
```

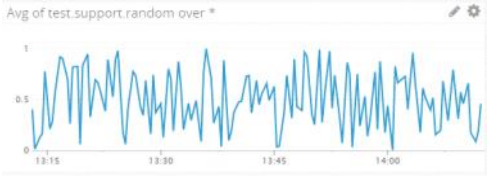
Script Source

```
import random
from checks import AgentCheck
class MattCheck(AgentCheck):
   def check(self, instance):
      self.gauge('test.support.random', random.random())
```

Script Config

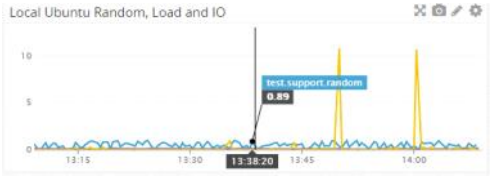
```
init_config:
 min_collection_interval: 20
instances:
   [{}]
```

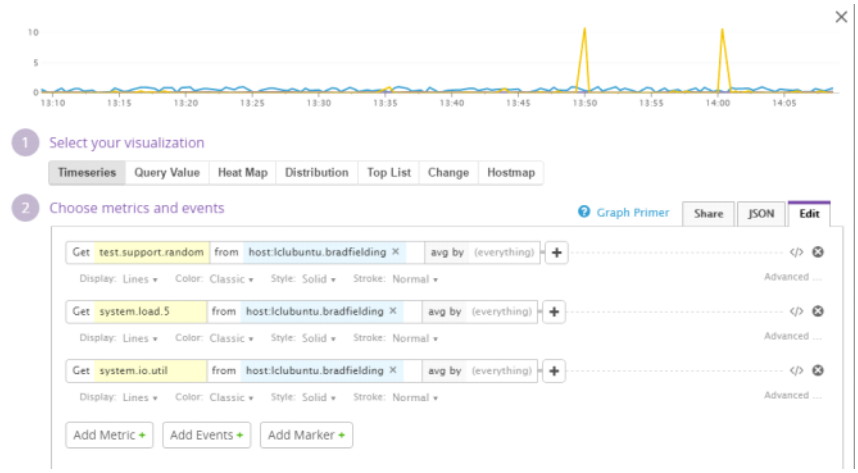
Graph of Random Value - showing results



Level 2 - Visualizing Data

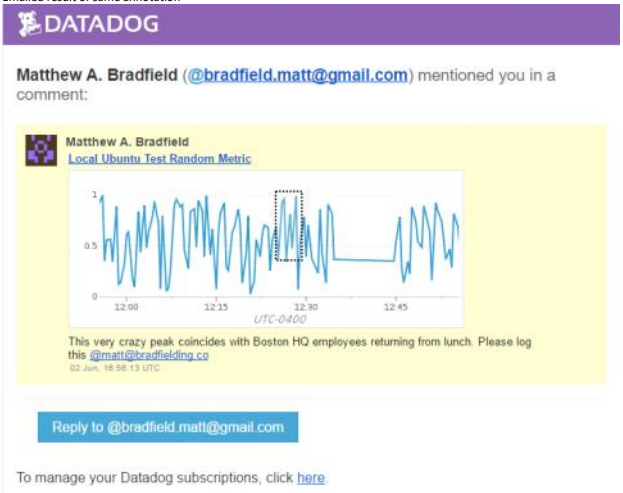
Created basic Dashboard showing 3 metrics from the local ubuntu host





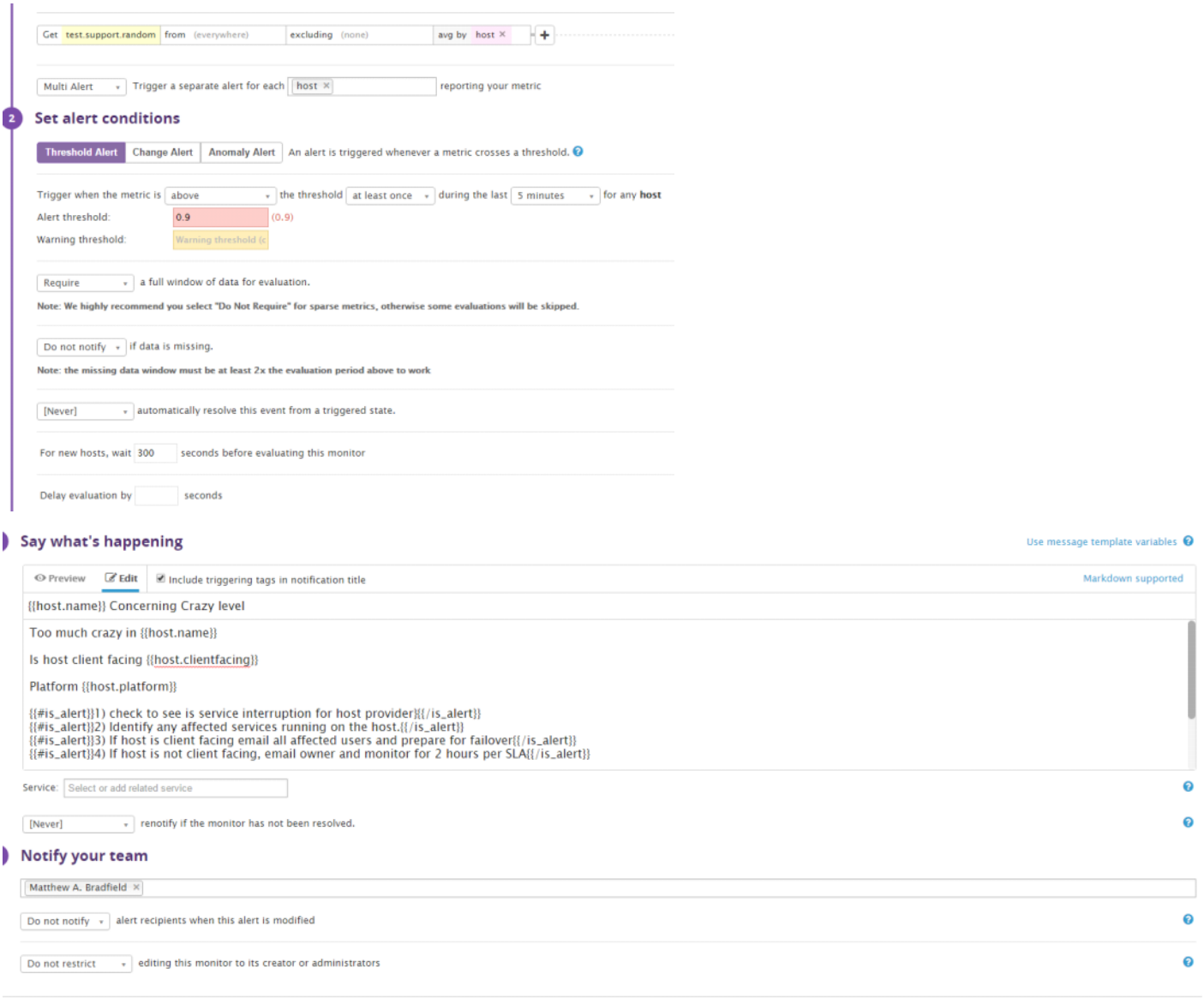
Box around graph at higher than .9

Emailed result of same annotation

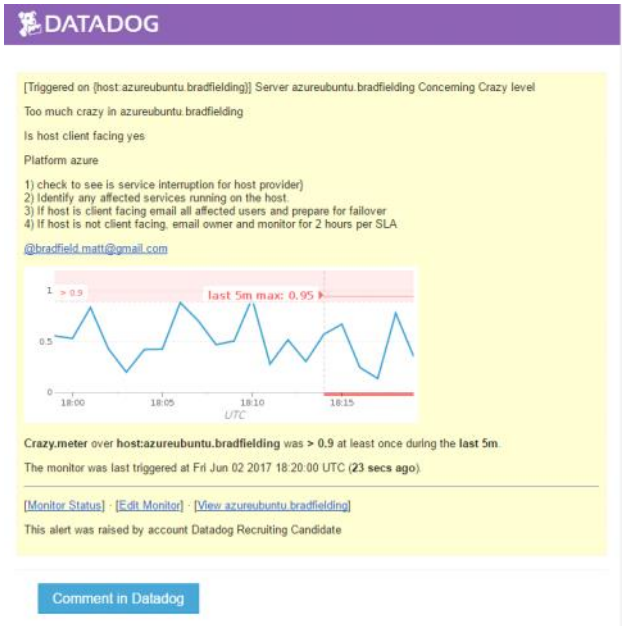


Level 3 - Alerting on your Data

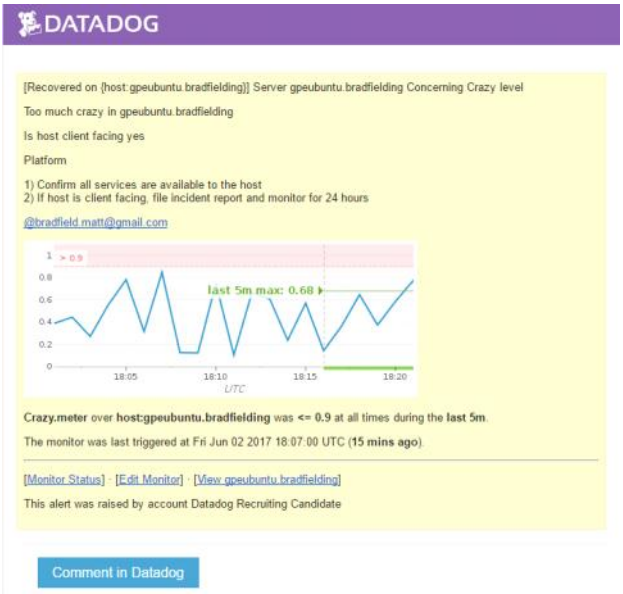
Monitor to notify of random metric going over .9. Created as multi-alert to trigger by each host.
Included conditional formatting so Alert and Alert Recovery have different messages



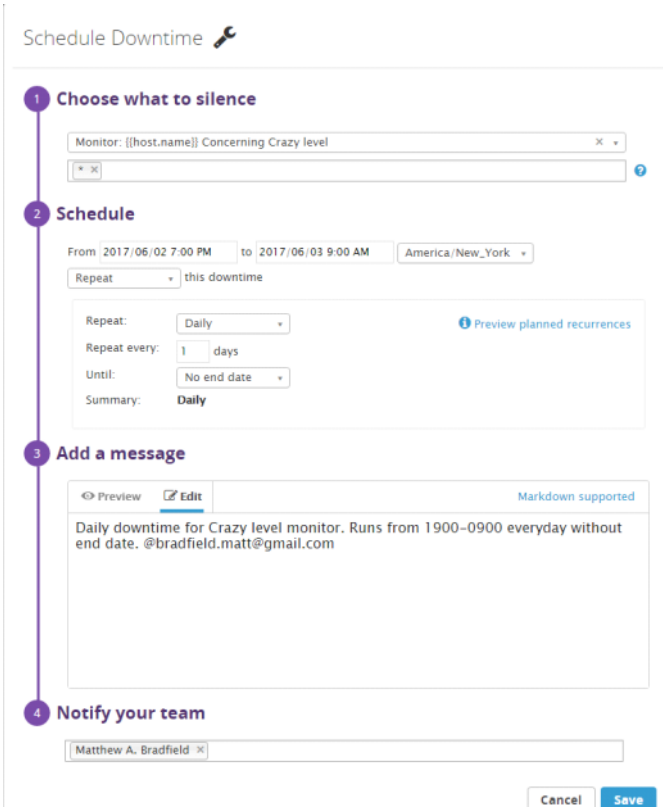
Notification Received when alert was triggered



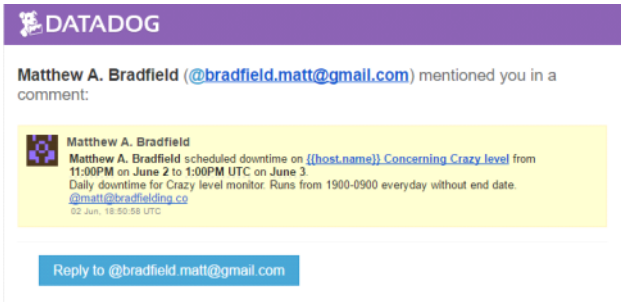
Notification received when alert was recovered



Scheduled Downtime

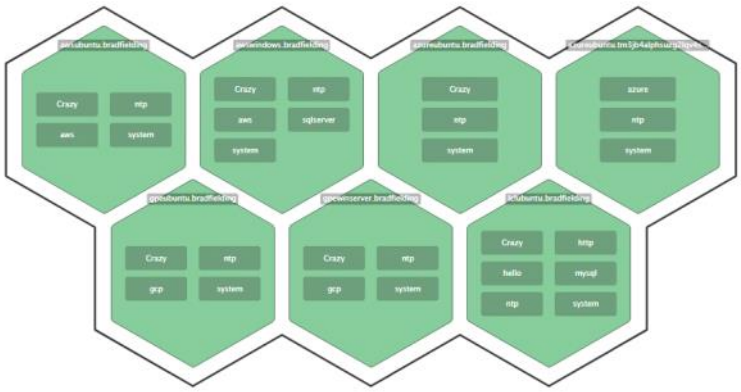


Email Notification



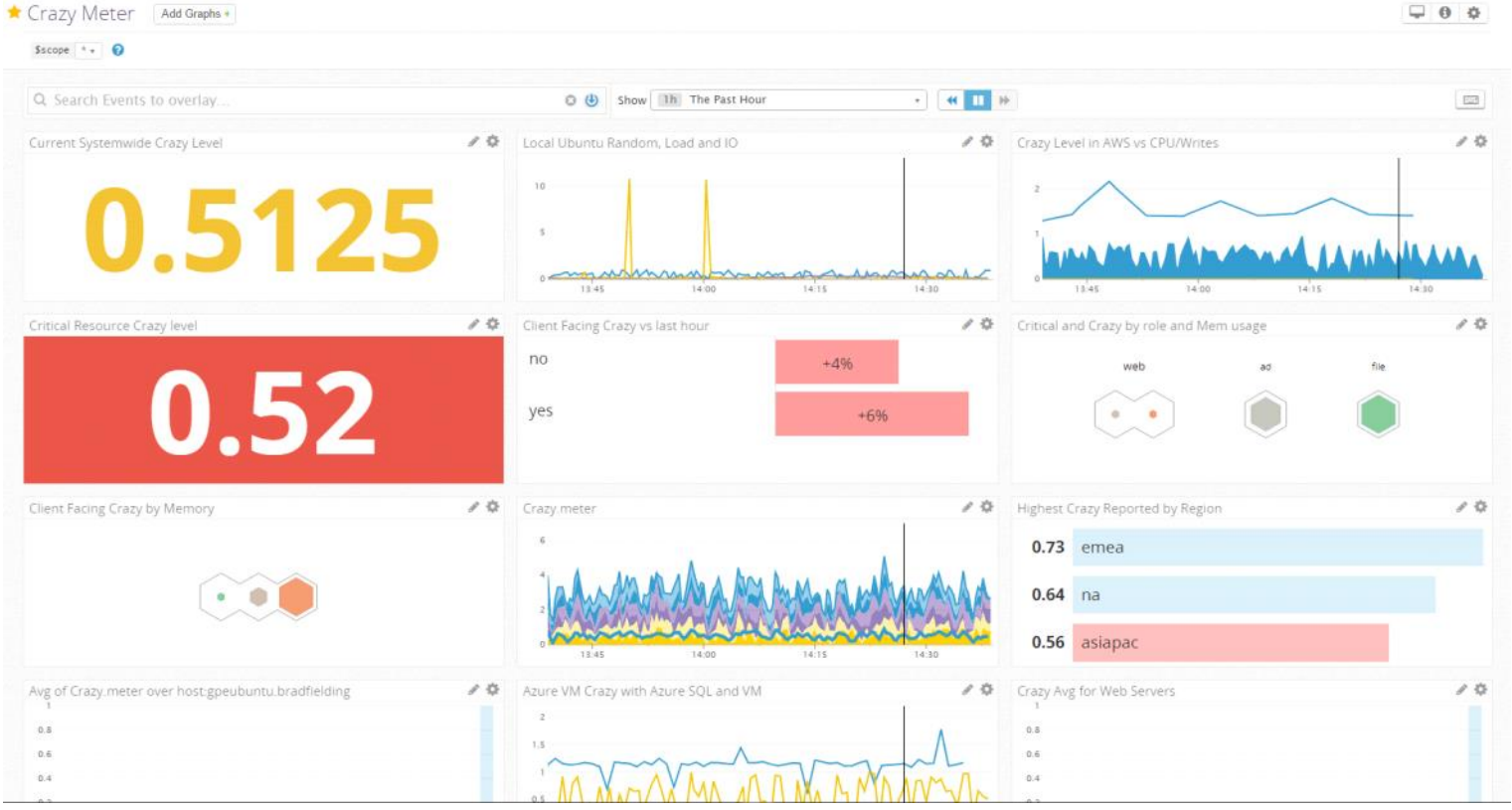
Additional Steps Complete

1) Deployed Ubuntu and Windows Server to Google Cloud Platform, AWS and Azure, deployed agents to all machines. Updated tags and hostnames appropriately for all

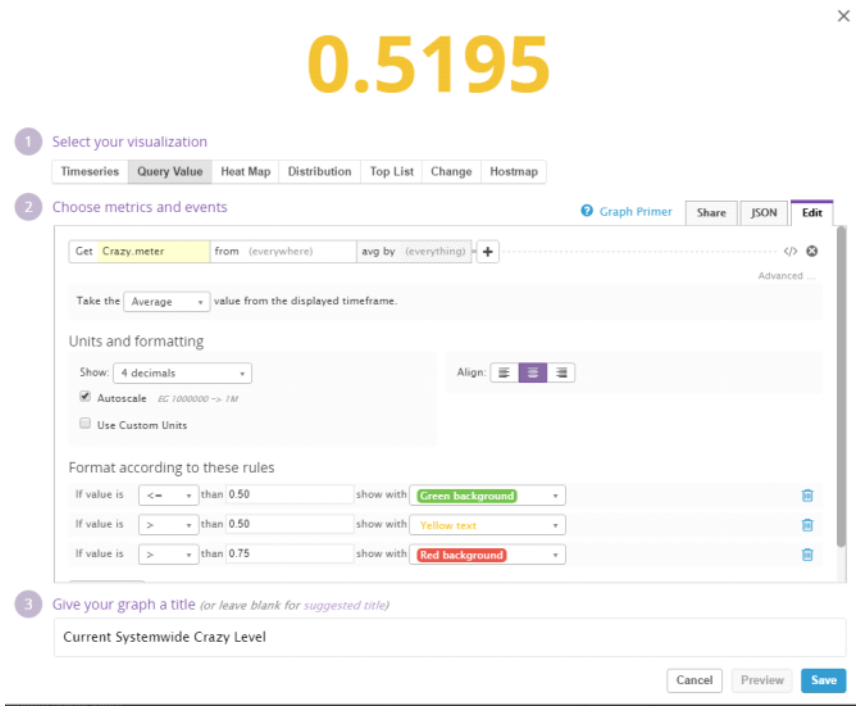


2) Created a 'Crazy_Meter' agent check and configured it on all of the hosts successfully

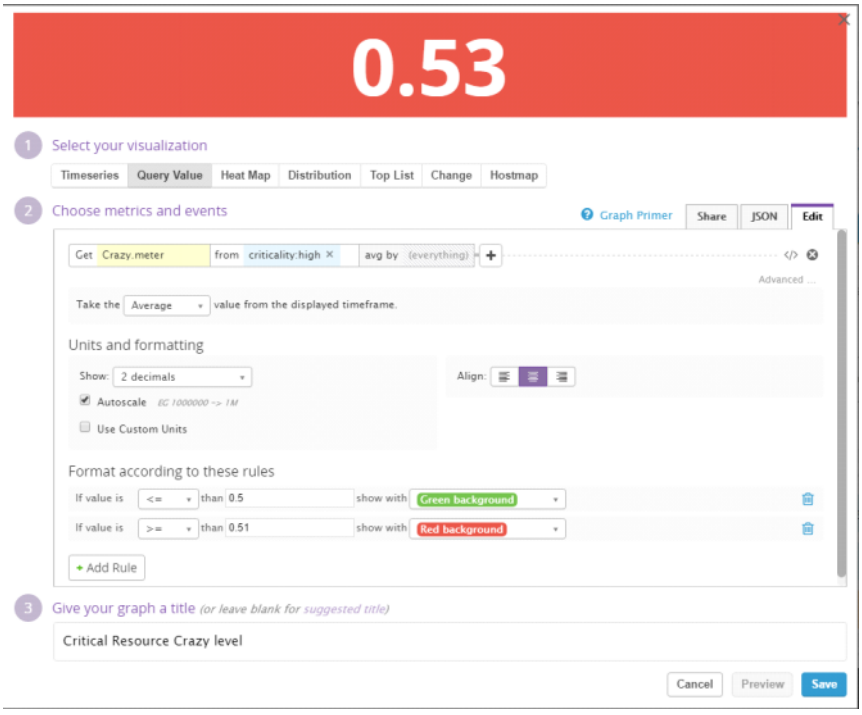
3) Configured Integrations for
   a. Azure
   b. AWS
   c. GCP
   d. SQL Server
   e. MSFT Event Viewer
   f. Azure SQL DB
4) Created a number of different graphs on a 'Crazy Meter' dashboard.
   https://app.datadoghq.com/dash/297649/crazy-meter?live=true&page=0&is_auto=false&from_ts=1496425260196&to_ts=1496428860196&tile_size=m



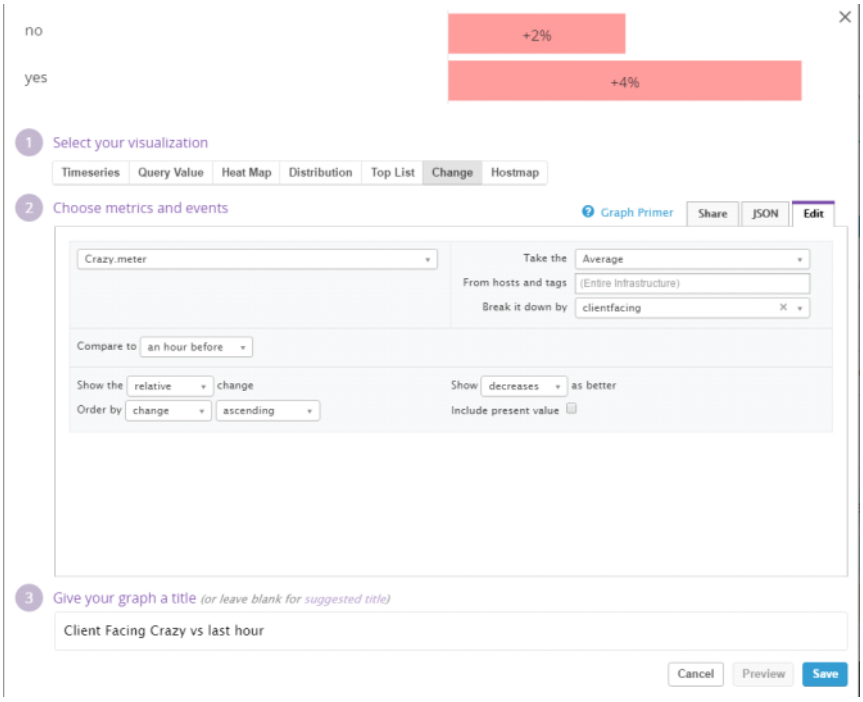
Dashboard includes a number of different data types/representations including
   a. Current average crazy level across all reporting hosts



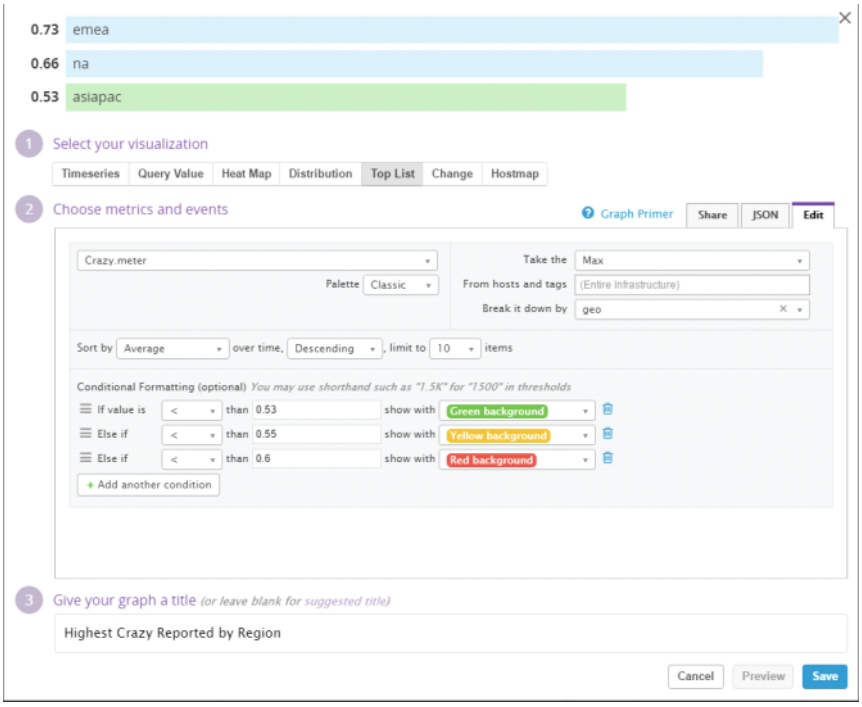
b. Current Crazy level on all 'critical' hosts



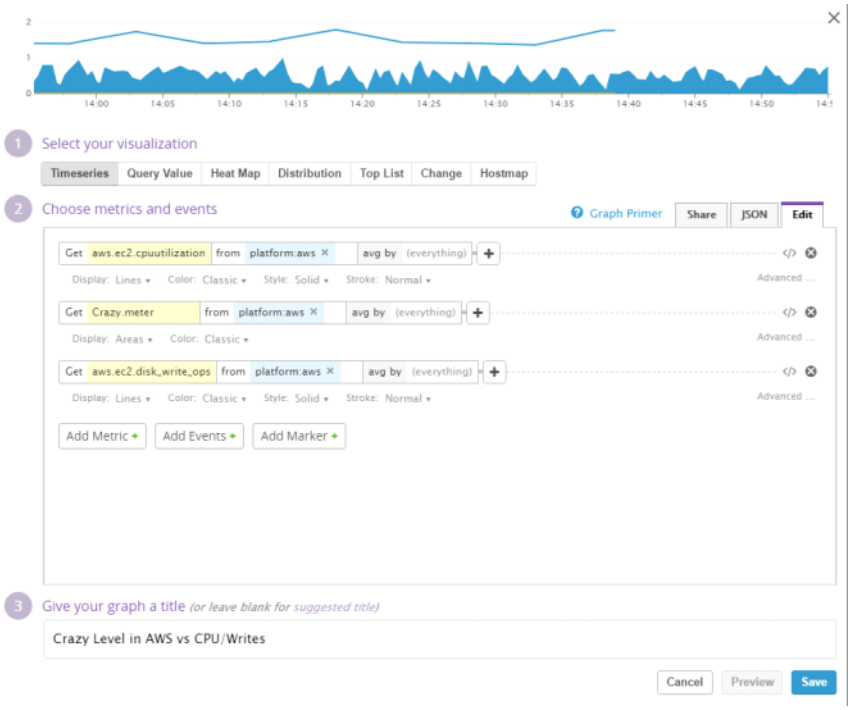
c. Change in crazy value over last hour broken down by client facing vs. non-client facing hosts



d. Highest crazy value reported per region



e. Crazy level in AWS hosts vs. CPU and number of writes for AWS hosts



f. Creating graphs and using appropriate tagging to automatically ensure new hosts are added to the appropriate visualizations