

Technical Exercise

Wednesday, May 31, 2017 10:44 AM

hashicorp\vagrant\bin\

Vagrant.exe init hashicorp/precise64
Vagrant.exe up

```
Administrator: Command Prompt
Microsoft Windows [version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd hashicorp\vagrant\bin\
The system cannot find the path specified.

C:\WINDOWS\system32>cd\

C:\>cd HashiCorp

C:\HashiCorp> vagrant\bin

C:\HashiCorp\vagrant\bin>vagrant.exe init hashicorp/precise64
A 'vagrantfile' has been placed in this directory. You are now
ready to vagrant up your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.

C:\HashiCorp\vagrant\bin>vagrant.exe up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'hashicorp/precise64' could not be found. Attempting to find and install...
default: Box Provider: virtualbox
default: Box Version: 0
==> default: Loading metadata for box 'hashicorp/precise64'
default: URL: https://atlas.hashicorp.com/hashicorp/precise64
==> default: Adding box 'hashicorp/precise64' (v1.1.0) for provider: virtualbox
default: Downloading: https://atlas.hashicorp.com/hashicorp/boxes/precise64/versions/1.1.0/providers/virtualbox.box
default: Progress: 100% (rate: 13.0M/s, estimated time remaining: --:--:--:--)
==> default: Successfully added box 'hashicorp/precise64' (v1.1.0) for 'virtualbox'!
==> default: Importing base box 'hashicorp/precise64'...
==> default: Matching VNC address for our networking...
==> default: Checking if box 'hashicorp/precise64' is up to date...
==> default: Setting the name of the VM: bin_default_1480242427904_n8n35
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
==> default: Forwarding ports...
default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
There was an error while executing 'vboxmanage', a CLI used by Vagrant
for controlling VirtualBox. The command and stderr is shown below.

Command: ["startvm", "c2ab7ff1-b0fd-4c39-b003-b3a2a5a4b295", "--type", "headless"]

Stderr: vboxmanage.exe: error: VT-x is disabled in the BIOS for all CPU modes (VERR_VMX_MSR_ALL_VMX_DISABLED)
vboxmanage.exe: error: Details: code E_FAIL (0x80000000), component ConsoleWrap, interface IConsole

C:\HashiCorp\vagrant\bin>
```

Screen clipping taken: 5/31/2017 10:55 AM

<https://stackoverflow.com/questions/33304393/vt-x-is-disabled-in-the-bios-for-both-all-cpu-modes-verr-vmx-msr-all-vmx-disabl>

Fixed Virtualization in BIOS and

```
Administrator: Command Prompt
Microsoft Windows [version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd\

C:\>cd HashiCorp

C:\HashiCorp> vagrant\bin

C:\HashiCorp\vagrant\bin>vagrant.exe up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'hashicorp/precise64' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
==> default: Forwarding ports...
default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
default: SSH address: 127.0.0.1:2222
default: SSH username: vagrant
default: SSH auth method: private key
default:
default: Vagrant insecure key detected. Vagrant will automatically replace
default: this with a newly generated keypair for better security.
default:
default: Inserting generated public key within guest...
default: Removing insecure key from the guest if it's present...
default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
default: The guest additions on this VM do not match the installed version of
default: VirtualBox! In most cases this is fine, but in rare cases it can
default: prevent things such as shared folders from working properly. If you see
default: shared folder errors, please make sure the guest additions within the
default: virtual machine match the version of VirtualBox you have installed on
default: your host and reload your VM.
default:
default: Guest Additions Version: 4.2.0
default: VirtualBox Version: 5.1
==> default: Mounting shared folders...
default: /vagrant => C:\HashiCorp\vagrant\bin

C:\HashiCorp\vagrant\bin>
```

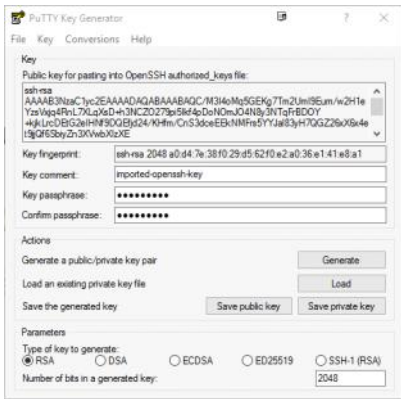
Screen clipping taken: 5/31/2017 11:04 AM

Finally!

```
Administrator Command Prompt
default: virtualbox version: 5.1
default: mounting shared folders...
default: /vagrant => C:\hashicorp\vagrant\bin
C:\hashicorp\vagrant\bin>vagrant destroy
default: Are you sure you want to destroy the 'default' vm? [y/n] y
default: Forcing shutdown of VM...
default: Destroying VM and associated drives...
C:\hashicorp\vagrant\bin>vagrant init hashicorp/precise64
vagrantfile already exists in this directory. remove it before
running 'vagrant init'
C:\hashicorp\vagrant\bin>vagrant init hashicorp/precise64
A 'vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.
C:\hashicorp\vagrant\bin>vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'hashicorp/precise64'...
==> default: Matching host address for NAT networking...
==> default: Checking if box 'hashicorp/precise64' is up to date...
==> default: Setting the name of the VM: bin_default_1496244617102_70...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
==> default: Adapter 1: nat
==> default: Forwarding ports...
==> default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
==> default: SSH address: 127.0.0.1:2222
==> default: SSH username: vagrant
==> default: SSH auth method: private key
==> default: vagrant insecure key detected. Vagrant will automatically replace
==> default: this with a newly generated keypair for better security.
==> default:
==> default: Inserting generated public key within guest...
==> default: Removing insecure key from the guest if it's present...
==> default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: The guest additions on this VM do not match the installed version of
==> default: VirtualBox! In most cases this is fine, but in rare cases it can
==> default: prevent things such as shared folders from working properly. If you see
==> default: shared folder errors, please make sure the guest additions within the
==> default: virtual machine match the version of VirtualBox you have installed on
==> default: your host and reload your VM.
==> default:
==> default: Guest additions version: 4.2.0
==> default: VirtualBox Version: 5.1
==> default: mounting shared folders...
default: /vagrant => C:\hashicorp\vagrant\bin
C:\hashicorp\vagrant\bin>
```

Screen clipping taken: 5/31/2017 11:35 AM

Putty KeyGen to support Private Key Auth on Windows host



Screen clipping taken: 5/31/2017 11:57 AM

<https://support.rackspace.com/how-to/logging-in-with-an-ssh-private-key-on-windows/>

Success!

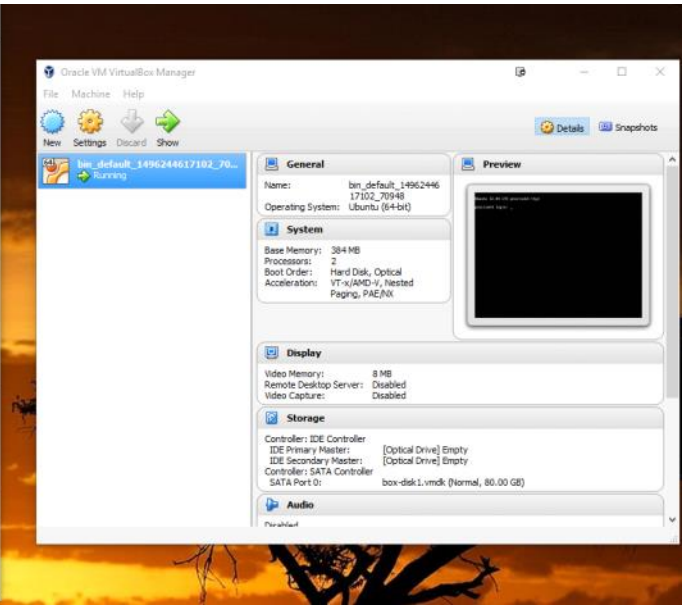
```
vagrant@precise64 ~$
Using username "vagrant".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Wrong passphrase
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)

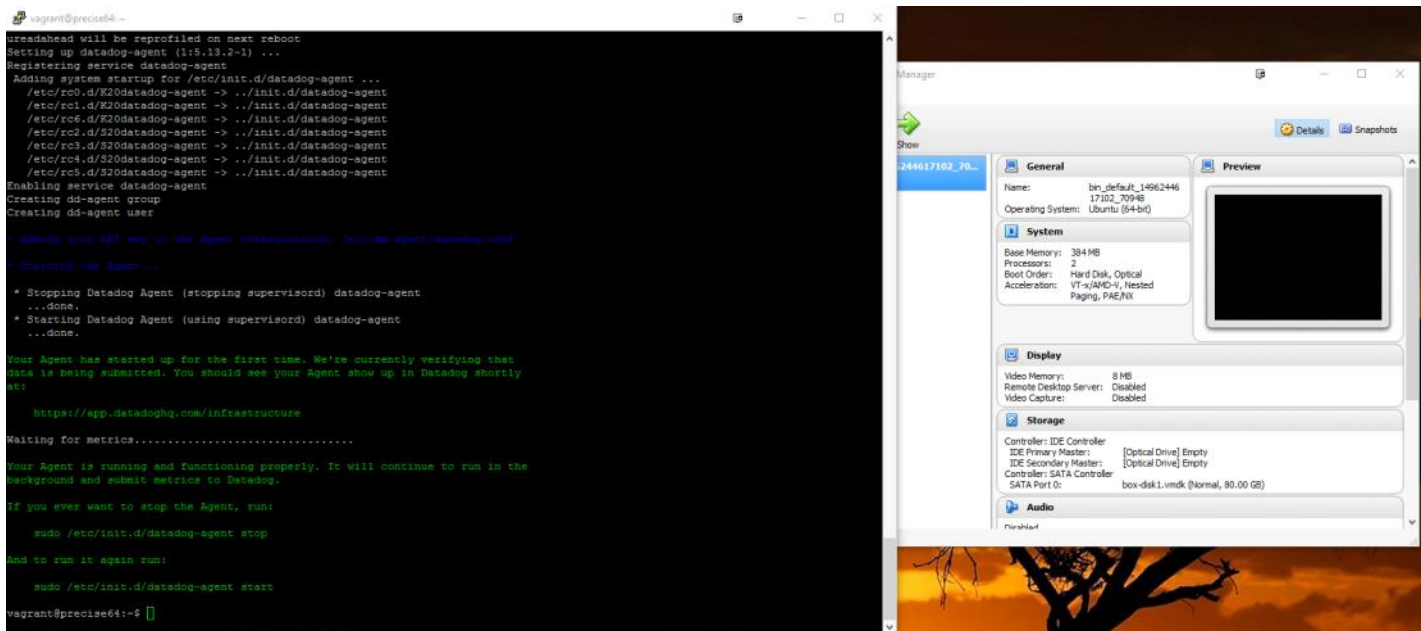
 * Documentation:  https://help.ubuntu.com/
New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 06:23:18 2012 from 10.0.2.2
vagrant@precise64:~$
```

Screen clipping taken: 5/31/2017 12:00 PM

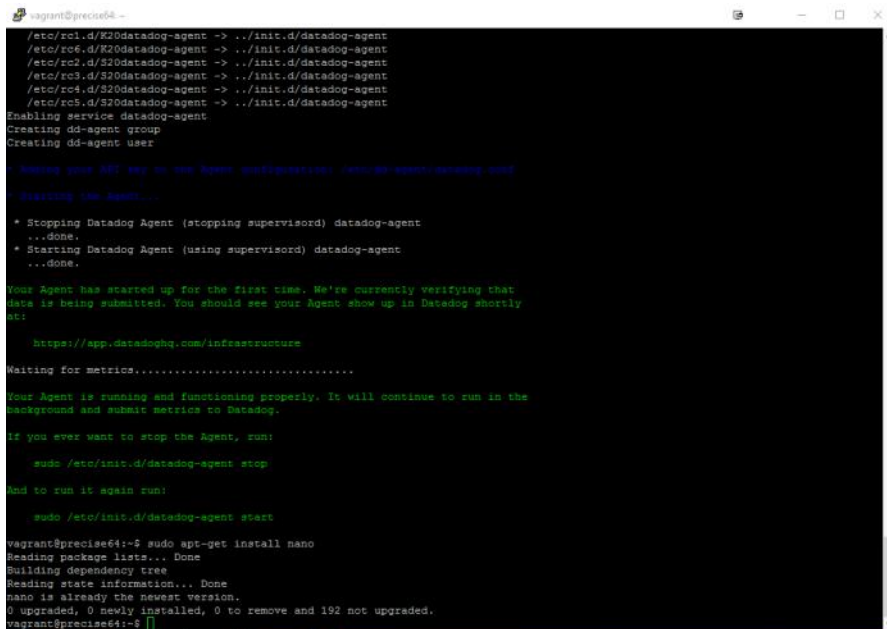
Agent installed successfully! Needed to install 'curl'





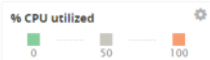
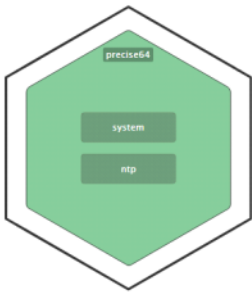
Screen clipping taken: 5/31/2017 12:08 PM

Nano terminal text editor



Screen clipping taken: 5/31/2017 12:15 PM

Your infrastructure



Screen clipping taken: 5/31/2017 12:33 PM

Changed hostname in config file prior to changing tags - which duplicated the hosts on the infrastructure list. Then saw a note saying it would take about 3 hours for them to disappear if they had been removed.



Screen clipping taken: 5/31/2017 1:28 PM

From `sudo /etc/init.d/datadog-agent info` saw - so I know the update worked

```
Hostnames
-----
agent-hostname: ubuntuweb
hostname: ubuntuweb
socket-fqdn: precise64
```

Screen clipping taken: 5/31/2017 1:29 PM

Extra machines removed by 2

Added Tags

```

GNU nano 2.2.6                                File: /etc/dd-agent/datadog.conf

[Main]

# The host of the Datadog intake server to send Agent data to
dd_url: https://app.datadoghq.com

# If you need a proxy to connect to the Internet, provide the settings here (default: disabled)
# proxy_host: my-proxy.com
# proxy_port: 8128
# proxy_user: user
# proxy_password: password
# To be used with some proxies that return a 302 which make curl switch from POST to GET
# See http://stackoverflow.com/questions/8156073/curl-violate-rfc-2616-10-3-2-and-switch-from-post-to-get
# proxy_forbid_method_switch: no

# If you run the agent behind haproxy, you might want to enable this
# skip_ssl_validation: no

# The Datadog api key to associate your Agent's data with your organization.
# Can be found here:
# https://app.datadoghq.com/account/settings
# (default: None, the agent doesn't start without it)
api_key: 2b12d61240ff877b086e15569db01abb

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

# Enable the trace agent.
# apm_enabled: false

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

# Set timeout in seconds for outgoing requests to Datadog. (default: 20)
# When a request timeout, it will be retried after some time.
# It will only be deleted if the forwarder queue becomes too big. (30 MB by default)
# forwarder_timeout: 20

# Set timeout in seconds for integrations that use HTTP to fetch metrics, since
# unbounded timeouts can potentially block the collector indefinitely and cause
# problems!
# default_integration_http_timeout: 9

# Add one "dd_check:checkname" tag per running check. It makes it possible to slice
# and dice per monitored app (= running Agent Check) on Datadog's backend.
# create_dd_check_tags: no

# Collect AWS EC2 custom tags as agent tags (requires an IAM role associated with the instance)
# collect_ec2_tags: no
# Incorporate security-groups into tags collected from AWS EC2
# collect_security_groups: no

```

Screen clipping taken: 5/31/2017 2:10 PM

started +

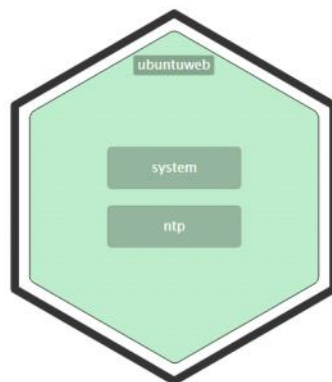
You are **33%** done setting up.

You have **14** days

Group hosts by tags

Fill by: % CPU utilized + avg +

Size by: + -- +



ubuntuweb aliases: ubuntuweb (dashboard)

Mute host

Apps (click to see metrics)

system

ntp

Agent

Datadog Agent: **v5.13.2**

System

GNU/Linux - 2 CPU - 2 vCPU - 10.0.2.15 - 383.24M B - 204.75G B

Metrics (as of 3 mins ago)

% CPU utilized **0.2 %**

Tags

Datadog Agent

#clientfacing:no #criticality:low #geo:na #role:web

User

Edit Tags

g. Inc. 2017 - Terms - Privacy - Status All Systems Operational

Screen clipping taken: 5/31/2017 2:10 PM

Deployed MySQL

Deployed Database just to make sure

```

create database datadogmysql;
grant all on datadogmysql.* to 'matt' identified by 'datadog';

```

```

mysql> create database datadogmysql;
Query OK, 1 row affected (0.01 sec)

mysql> grant all on datadogmysql.* to 'matt' identified by 'datadog';
Query OK, 0 rows affected (0.00 sec)

```

Configured MySQL Integration

```
mysql> show grants for 'datadog'@'localhost';
+-----+
-----+
--++
| Grants for datadog@localhost
|
+-----+
-----+
--++
| GRANT PROCESS, REPLICATION CLIENT ON *.* TO 'datadog'@'localhost' IDENTIFIED BY
Y PASSWORD '*2647B4EE68112D02461EE0510C66D7E8DACD918' WITH MAX_USER_CONNECTIONS
5 |
| GRANT SELECT ON 'performance_schema'.* TO 'datadog'@'localhost'
|
+-----+
-----+
--++
2 rows in set (0.00 sec)
```

Screen clipping taken: 5/31/2017 2:43 PM

Deleted users and started over - needed to understand running mysql commands from command line more.

```
vagrant@precise64:~$ sudo mysql -u root -p -e "CREATE USER 'datadog'@'localhost' IDENTIFIED BY 'LtzM0lscIJEnteaPJlUUVBJ';"
Enter password:
vagrant@precise64:~$ sudo mysql -u root -p -e "GRANT REPLICATION CLIENT ON *.* TO 'datadog'@'localhost' WITH MAX_USER_CONNECTIONS 5;"
Enter password:
vagrant@precise64:~$ sudo mysql -u root -p -e "GRANT PROCESS ON *.* TO 'datadog'@'localhost';"
Enter password:
vagrant@precise64:~$ sudo mysql -u root -p -e "GRANT SELECT ON performance_schema.* TO 'datadog'@'localhost';"
Enter password:
vagrant@precise64:~$ mysql -u datadog --password=LtzM0lscIJEnteaPJlUUVBJ -e "show status" | \
> grep Uptime && echo -e "\033[0;32mMySQL user - OK\033[0m" || \
> echo -e "\033[0;31mCannot connect to MySQL\033[0m"
Uptime 3264
Uptime since flush_status 3264
MySQL user - OK
vagrant@precise64:~$ mysql -u datadog --password=LtzM0lscIJEnteaPJlUUVBJ -e "SELECT * FROM performance_schema.threads" && \
> echo -e "\033[0;32mMySQL SELECT grant - OK\033[0m" || \
> echo -e "\033[0;31mMissing SELECT grant\033[0m"
MySQL SELECT grant - OK
vagrant@precise64:~$ mysql -u datadog --password=LtzM0lscIJEnteaPJlUUVBJ -e "SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST" && \
> echo -e "\033[0;32mMySQL PROCESS grant - OK\033[0m" || \
> echo -e "\033[0;31mMissing PROCESS grant\033[0m"
MySQL PROCESS grant - OK
vagrant@precise64:~$
vagrant@precise64:~$
```

ID	USER	HOST	DB	COMMAND	TIME	STATE	INFO
55	datadog	localhost	NULL	Query	0	executing	SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST

Screen clipping taken: 5/31/2017 3:23 PM

Checks Successful!

Could not find file /etc/dd-agent/conf.d/mysql.yaml - but found mysql.yaml.example, modified the values to match the screenshot and then saved it as mysql.yaml - restarted agent, ran info

```
Checks
=====
ntp (5.13.2)
-----
- instance #0 [OK]
- Collected 1 metric, 0 events & 1 service check

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

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

mysql (unknown)
-----
- initialize check class [ERROR]: 'mapping values are not allowed in this context\n in "<byte string>"', line 5, column 10'
```

Screen clipping taken: 5/31/2017 3:53 PM

Initialization failure. - Turned on line numbers in nano

```
GNU nano 2.2.6 File: mysql.yaml

init_config:

instances:
- server: localhost
  user: datadog
  pass: LtzM0lscIJEnteaPJlUUVBJ
  # port: 3306 # Optional
  # sock: /path/to/sock # Connect via Unix Socket
  # defaults_file: my.cnf # Alternate configuration mechanism
  # connect_timeout: None # Optional integer seconds
  tags: # Optional
    - optional_tag1
    - optional_tag2
  options: # Optional
    replication: 0
    # replication_non_blocking_status: false # grab slave count in non-blocking manner (req. performance_schema)
    galera_cluster: false
    # extra_status_metrics: true
    # extra_innodb_metrics: true
    # extra_performance_metrics: true
    # schema_size_metrics: false
    # disable_innodb_metrics: false
```

Screen clipping taken: 5/31/2017 3:59 PM

Tried again - same problem, found this - <https://botbot.me/freenode/datadog/2015-07-19/?page=1>
Changed formatting, tried again

Malformed yaml, modified the formatting and used <https://yaml-online-parser.appspot.com/> and <http://www.yamlint.com/> to validate YAML

Finally - this YAML worked
init_config:

```
instances:
- server: localhost
  user: datadog
  pass: LtzM0lscIJEnteaPJlUUVBJ
```

tags:

```
- role:db
- geo:na
options:
  replication: 0
  galera_cluster: 1
  extra_status_metrics: true
  extra_innodb_metrics: true
  extra_performance_metrics: true
  schema_size_metrics: false
  disable_innodb_metrics: false
```

```
Checks
=====

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

disk (5.13.2)
-----
- instance #0 [OK]
- Collected 32 metrics, 0 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
```

Screen clipping taken: 5/31/2017 6:38 PM

Writing Custom Agent Check

Hello World Example
/etc/dd-agent/checks.d/hello.py
/etc/dd-agent/conf.d/hello.yaml

```
Checks
=====

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

mysql (5.13.2)
-----
- instance #0 [OK]
- Collected 62 metrics, 0 events & 1 service check

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

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
```

Screen clipping taken: 5/31/2017 6:52 PM

HTTP Check

YAML Config File
/etc/dd-agent/conf.d/hello.yam

init_config:
 default_timeout: 5

instances:
 - url: <https://google.com>
 - url: <http://httpbin.org/delay/10>
 timeout: 8
 - url: <http://httpbin.org/status/400>

PY Script
/etc/dd-agent/checks.d/http.py

```
import time
import requests

from checks import AgentCheck
from hashlib import md5

class HTTPCheck(AgentCheck):
    def check(self, instance):
        if 'url' not in instance:
            self.log.info("Skipping instance, no url found.")
            return

        # Load values from the instance config
        url = instance['url']
        default_timeout = self.init_config.get('default_timeout', 5)
        timeout = float(instance.get('timeout', default_timeout))

        # Use a hash of the URL as an aggregation key
        aggregation_key = md5(url).hexdigest()

        # Check the URL
        start_time = time.time()
        try:
            r = requests.get(url, timeout=timeout)
            end_time = time.time()
        except requests.exceptions.Timeout as e:
            # If there's a timeout
            self.timeout_event(url, timeout, aggregation_key)
            return

        if r.status_code != 200:
            self.status_code_event(url, r, aggregation_key)

        timing = end_time - start_time
        self.gauge('http.reponse_time', timing, tags=['http_check'])

    def timeout_event(self, url, timeout, aggregation_key):
        self.event([
            'timestamp': int(time.time()),
```

```

        'event_type': 'http_check',
        'msg_title': 'URL timeout',
        'msg_text': '%s timed out after %s seconds.' % (url, timeout),
        'aggregation_key': aggregation_key
    })

def status_code_event(self, url, r, aggregation_key):
    self.event({
        'timestamp': int(time.time()),
        'event_type': 'http_check',
        'msg_title': 'Invalid reponse code for %s' % url,
        'msg_text': '%s returned a status of %s' % (url, r.status_code),
        'aggregation_key': aggregation_key
    })

if __name__ == '__main__':
    check, instances = HTTPCheck.from_yaml('/path/to/conf.d/http.yaml')
    for instance in instances:
        print "\nRunning the check against url: %s" % (instance['url'])
        check.check(instance)
        if check.has_events():
            print 'Events: %s' % (check.get_events())
        print 'Metrics: %s' % (check.get_metrics())

```

Restarted Agent and ran -info - new checks show up as ok for all three instances

```

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 0 metrics, 0 events & 0 service checks

mysql (5.13.2)
-----
- instance #0 [OK]
- Collected 62 metrics, 0 events & 1 service check

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

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

```

Screen clipping taken: 6/1/2017 9:01 AM

Ran the check also by running (sudo -u dd-agent dd-agent check http) just to see the output and got the below

```

vagrant@precise64:~$ sudo -u dd-agent dd-agent check http
2017-06-01 13:04:22,945 | INFO | dd.collector | config(config.py:1139) | initialized checks.d checks: ['http', 'network', 'mysql', 'ntp', 'disk', 'hello']
2017-06-01 13:04:22,945 | INFO | dd.collector | config(config.py:1140) | initialization failed checks.d checks: []
2017-06-01 13:04:22,946 | INFO | dd.collector | checks.collector(collector.py:542) | Running check http
Metrics:
[('http.response_time',
  1496322271,
  0.04824185371398926,
  {'hostname': 'ubuntuweb', 'tags': ['http_check'], 'type': 'gauge'})]
Events:
[('aggregation_key': '03edaf1ca1b67234e00d5164a384540b',
  'event_type': 'http_check',
  'msg_text': 'http://httpbin.org/delay/10 timed out after 8.0 seconds.',
  'msg_title': 'URL timeout',
  'timestamp': 1496322271),
 ('aggregation_key': 'b6f5dcea9029b7f92151ca070ef1d367',
  'event_type': 'http_check',
  'msg_text': 'http://httpbin.org/status/400 returned a status of 400',
  'msg_title': 'Invalid reponse code for http://httpbin.org/status/400',
  'timestamp': 1496322271)]
Service Checks:
[]
Service Metadata:
[({}, {}, {})]
[({}, {}, {})]
http (5.13.2)
-----
- instance #0 [OK]
- instance #1 [OK]
- instance #2 [OK]
- Collected 1 metric, 2 events & 0 service checks

```

Screen clipping taken: 6/1/2017 9:06 AM

Creating 'Random Value Check'

Config File - /etc/dd-agent/conf.d/random.yaml
 Started with basically blank config
 init_config:

instances:
 [{}]

Script File - /etc/dd-agent/checks.d/random.py
 Started with

```

import random

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

```

Restarted - Info. Appears something blew up on the collector - 'Uncaught error running the agent'
 Looked at Collector log - /var/log/datadog/collector.log


```

2017-06-01 13:24:22 UTC | INFO | dd.collector | daemon(daemon.py:157) | Starting
2017-06-01 13:24:22 UTC | ERROR | dd.collector | config(config.py:833) | Unable to import check module random.py from checks.d
Traceback (most recent call last):
  File "/opt/datadog-agent/agent/config.py", line 829, in _get_check_class
    check_module = imp.load_source('checks.d.%s' % check_name, check_path)
  File "/etc/dd-agent/checks.d/random.py", line 1
    Import random
    ^
IndentationError: unexpected indent
2017-06-01 13:24:22 UTC | INFO | dd.collector | config(config.py:1139) | initialized checks.d checks: ['http', 'network', 'mysql', 'ntp', 'disk', 'hello']
2017-06-01 13:24:22 UTC | INFO | dd.collector | config(config.py:1140) | initialization failed checks.d checks: ['random']
2017-06-01 13:24:26 UTC | INFO | dd.collector | checks.collector(collector.py:403) | Running check http
2017-06-01 13:24:35 UTC | INFO | dd.collector | checks.collector(collector.py:403) | Running check network
2017-06-01 13:24:35 UTC | INFO | dd.collector | checks.collector(collector.py:403) | Running check mysql
2017-06-01 13:24:35 UTC | INFO | dd.collector | checks.collector(collector.py:403) | Running check ntp
2017-06-01 13:24:35 UTC | INFO | dd.collector | checks.collector(collector.py:403) | Running check disk
2017-06-01 13:24:35 UTC | INFO | dd.collector | checks.collector(collector.py:403) | Running check hello
2017-06-01 13:24:35 UTC | INFO | dd.collector | checks.collector(collector.py:685) | Hostnames: ('agent-hostname': 'ubuntuweb', 'hostname': 'ubuntuweb', 'socket-fqdn': 'precise64', 'timezones': ('UTC', 'UTC')), tags: {'system':
2017-06-01 13:24:35 UTC | INFO | dd.collector | checks.collector(collector.py:530) | Finished run #1. Collection time: 12.75s. Emit time: 0.03s
2017-06-01 13:24:36 UTC | ERROR | dd.collector | collector(agent.py:604) | Uncaught error running the Agent
Traceback (most recent call last):
  File "/opt/datadog-agent/agent/agent.py", line 600, in _xmodule>
    sys.exit(main())
  File "/opt/datadog-agent/agent/agent.py", line 526, in main
    return Agent.info(verbose=options.verbose)
  File "/opt/datadog-agent/agent/agent.py", line 218, in info
    return CollectorStatus.print_latest_status(verbose=options.verbose)
  File "/opt/datadog-agent/agent/checks/check_status.py", line 281, in print_latest_status
    message = module.status.render()
  File "/opt/datadog-agent/agent/checks/check_status.py", line 179, in render
    ] + ["", ""]
  File "/opt/datadog-agent/agent/checks/check_status.py", line 545, in body_lines

```

Unexpected indent in my python script - took out the space between import random and from checks and restarted. This time it loaded, but the check failed for the random check I created

```

vagrant@precise64:~$ sudo -u dd-agent dd-agent check
Usage: /usr/bin/dd-agent check <check_name> [check_rate]
Add check_rate as 1st argument to compute rates
vagrant@precise64:~$ sudo -u dd-agent dd-agent check random
2017-06-01 13:35:47.095 | ERROR | dd.collector | config(config.py:833) | Unable to import check module random.py from checks.d
Traceback (most recent call last):
  File "/opt/datadog-agent/agent/config.py", line 829, in _get_check_class
    check_module = imp.load_source('checks.d.%s' % check_name, check_path)
  File "/etc/dd-agent/checks.d/random.py", line 1
    Import random
    ^

```

Screen clipping taken: 6/1/2017 9:37 AM

Apparently random.py is reserved for the module I'm importing, so I need to change the name. I just switched the files to matt_random.py and matt_random.yaml and deleted my existing files, restarted the agent and ran -info

```

vagrant@precise64:~$ sudo nano /etc/dd-agent/conf.d/random.yaml
vagrant@precise64:~$ sudo nano /etc/dd-agent/checks.d/random.py
vagrant@precise64:~$ sudo nano /etc/dd-agent/checks.d/matt_random.py
vagrant@precise64:~$ sudo rm /etc/dd-agent/conf.d/random.yaml /etc/dd-agent/checks.d/random.py
vagrant@precise64:~$ ls
ddagent-install.log  postinstall.sh
vagrant@precise64:~$ sudo nano /etc/dd-agent/checks.d/random.py
vagrant@precise64:~$ sudo /etc/init.d/datadog-agent restart
* Stopping Datadog Agent (stopping supervisor) datadog-agent
* Starting Datadog Agent (using supervisor) datadog-agent
vagrant@precise64:~$ ll

```

Screen clipping taken: 6/1/2017 10:47 AM

Collector blew up again on info command - 'uncaught error running the agent' - Realized my python syntax was wrong and it keeps complaining because a null value is being returned.

Trying

```

import random
mattrandomvalue = random.random()

```

```

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

```

Same error about unexpected indent, modified the formatting to take out initial spaces, restarted Info - 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

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

```

Ran check directly against this to see results - VICTORY!

```
vagrant@precise64:~$ sudo -u dd-agent dd-agent check matt_random
2017-06-01 15:27:43,572 | INFO | dd.collector | config(config.py:1139) | initialized checks.d checks: ['matt_random', 'http', 'network', 'mysql', 'ntp', 'disk', 'hello']
2017-06-01 15:27:43,572 | INFO | dd.collector | config(config.py:1140) | initialization failed checks.d checks: []
2017-06-01 15:27:43,572 | INFO | dd.collector | checks.collector(collector.py:542) | Running check matt_random
Metrics:
[{'test.support.random',
  1496330863,
  0.6703302721766303,
  {'hostname': 'ubuntuweb', 'type': 'gauge'}}]
Events:
[]
Service Checks:
[]
Service Metadata:
[{}]
matt_random (5.13.2)
-----
- instance #0 [OK]
- Collected 1 metric, 0 events & 0 service checks
```

Screen clipping taken: 6/1/2017 11:28 AM

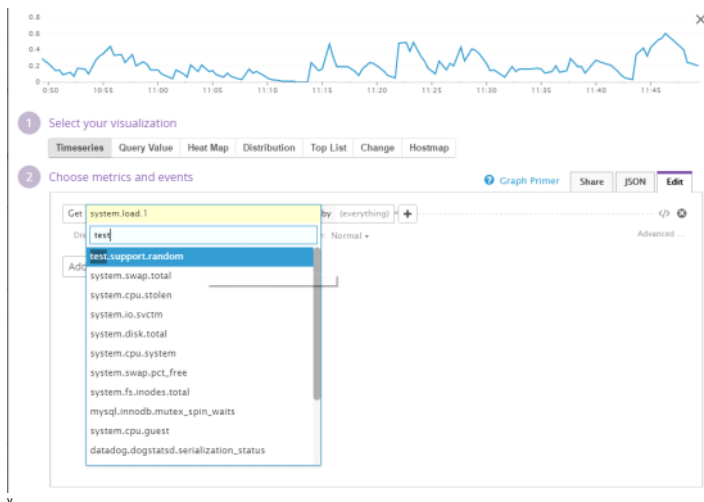
Ran again just to make sure that my script didn't inadvertently store the same random value - it did not. Just paranoid mode confirmation!

```
vagrant@precise64:~$ sudo -u dd-agent dd-agent check matt_random
2017-06-01 15:36:06,847 | INFO | dd.collector | config(config.py:1139) | initialized checks.d checks: ['matt_random', 'http', 'network', 'mysql', 'ntp', 'disk', 'hello']
2017-06-01 15:36:06,848 | INFO | dd.collector | config(config.py:1140) | initialization failed checks.d checks: []
2017-06-01 15:36:06,848 | INFO | dd.collector | checks.collector(collector.py:542) | Running check matt_random
Metrics:
[{'test.support.random',
  1496331366,
  0.747074070988056,
  {'hostname': 'ubuntuweb', 'type': 'gauge'}}]
Events:
[]
Service Checks:
[]
Service Metadata:
[{}]
matt_random (5.13.2)
-----
- instance #0 [OK]
- Collected 1 metric, 0 events & 0 service checks
```

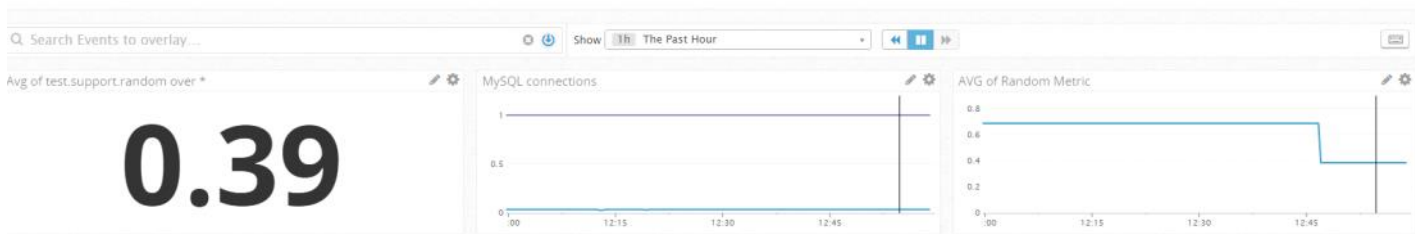
Screen clipping taken: 6/1/2017 11:37 AM

Visualizing Data

Cloned Dashboard - test.support.random showing up as an available metric.



Hmmm - flatline graph. I ran the check again directly to see if the result was different than previous and it was.



Screen clipping taken: 6/1/2017 1:00 PM

```
vagrant@precise64:~$ sudo nano /etc/dd-agent/checks.d/matt_random.py
vagrant@precise64:~$ sudo -u dd-agent dd-agent check matt_random
2017-06-01 15:57:48,833 | INFO | dd.collector | config(config.py:1139) | initialized checks.d checks: ['matt_random', 'http', 'network', 'mysql', 'ntp', 'disk', 'hello']
2017-06-01 15:57:48,833 | INFO | dd.collector | config(config.py:1140) | initialization failed checks.d checks: []
2017-06-01 15:57:48,834 | INFO | dd.collector | checks.collector(collector.py:542) | Running check matt_random
Metrics:
[{'test.support.random',
  149632668,
  0.3501909963152954,
  ('hostname': 'ubuntuweb', 'type': 'gauge')}]
Events:
[]
Service Checks:
[]
Service Metadata:
[{}]
matt_random (5.13.2)
-----
- instance #0 [OK]
- Collected 1 metric, 0 events & 0 service checks
```

Screen clipping taken: 6/1/2017 12:04 PM

Every time the service is restarted the value is changed because I set this as a global variable. Changed to below.

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

Restarted, no errors, ran info

```
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 0 metrics, 0 events & 0 service checks

mysql (5.13.2)
-----
- instance #0 [OK]
- Collected 62 metrics, 0 events & 1 service check

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

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

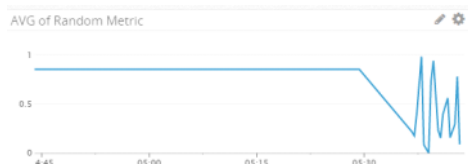
hello (5.13.2)
-----
- instance #0 [OK]
- Collected 1 metric, 0 events & 0 service checks
```

Screen clipping taken: 6/2/2017 5:42 AM
Ran check itself

```
vagrant@precise64:~$ sudo -u dd-agent dd-agent check matt_random
2017-06-02 09:40:59,683 | INFO | dd.collector | config(config.py:1139) | initialized checks.d checks: ['matt_random', 'http', 'network', 'mysql', 'ntp', 'disk', 'hello']
2017-06-02 09:40:59,684 | INFO | dd.collector | config(config.py:1140) | initialization failed checks.d checks: []
2017-06-02 09:40:59,684 | INFO | dd.collector | checks.collector(collector.py:542) | Running check matt_random
Metrics:
[{'test.support.random',
  1496396459,
  0.5565647526784846,
  ('hostname': 'ubuntuweb', 'type': 'gauge')}]
Events:
[]
Service Checks:
[]
Service Metadata:
[{}]
matt_random (5.13.2)
-----
- instance #0 [OK]
- Collected 1 metric, 0 events & 0 service checks
```

Screen clipping taken: 6/2/2017 5:43 AM

Confirmed in DataDog HQ and am now showing varied results!



Screen clipping taken: 6/2/2017 5:44 AM

Other Agents

Custom Metric - crazyscale