Forensic Log Analysis with BigQuery

@thagomizer_rb

Aja Hammerly

Developer Advocate

@thagomizer_rb

github.com/thagomizer/examples

Google Cloud Platform

Licensing:

All code is copyright Google and licensed under Apache V2.

Survey

@thagomizer_rb

Me too!

@thagomizer_rb

Story

https://www.flickr.com/photos/jamidwyer/2232846153

https://www.flickr.com/photos/thienzieyung/7156187749


tps://www.flickr.com/photos/12265657@N04/4925461145

What Then?

Get Back Up

Figure Out Why

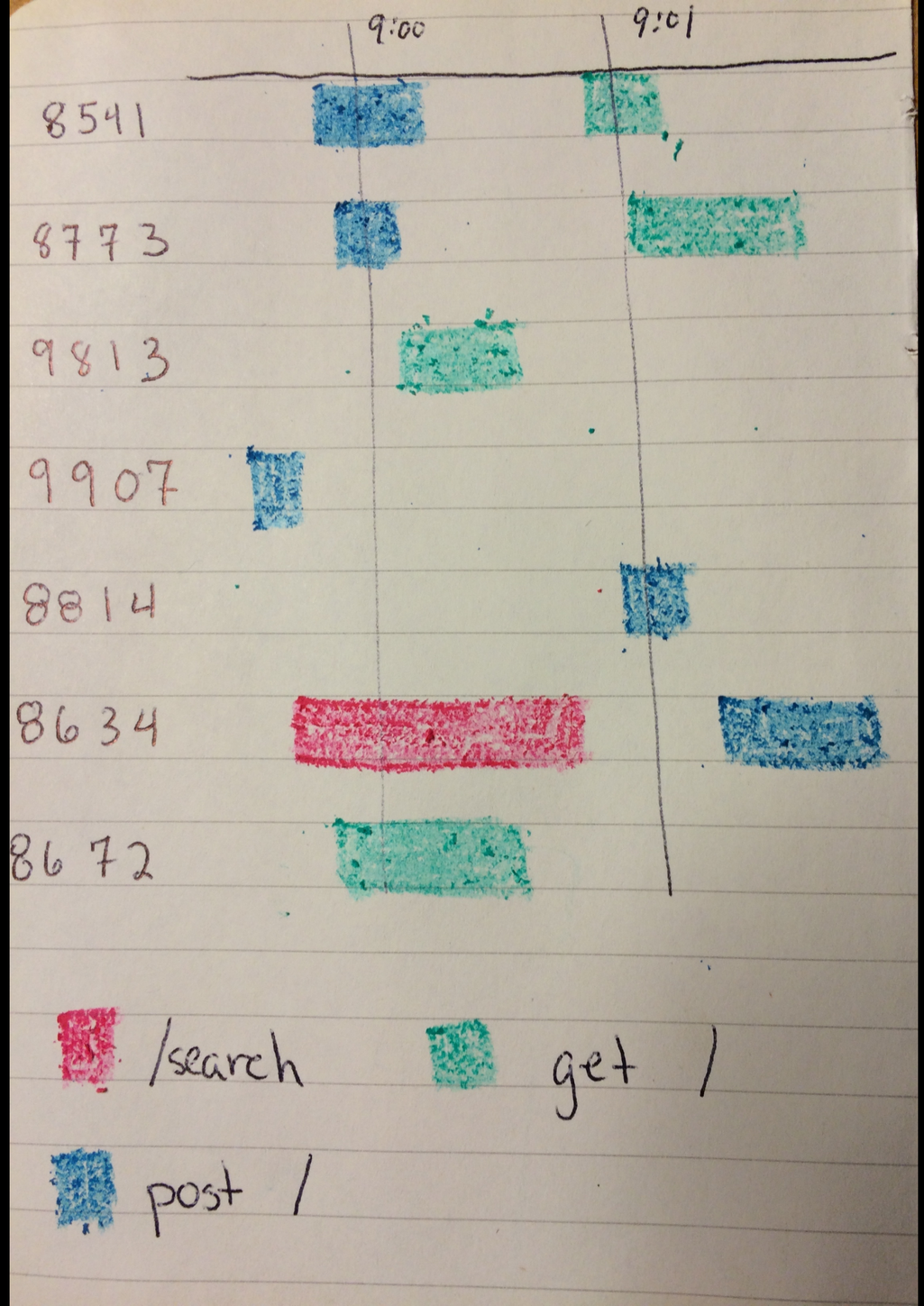
@thagomizer_rb

Easy

@thagomizer_rb

...not

Possible Tools


9:00
9:01
8541
8773
9813
9907
8814
8634
8672
/search
get /
post /

But Scale

More Tools

$$$

Tools

@thagomizer_rb

Forethought

@thagomizer_rb

Google BigQuery

@thagomizer_rb

What Is It?

Fast

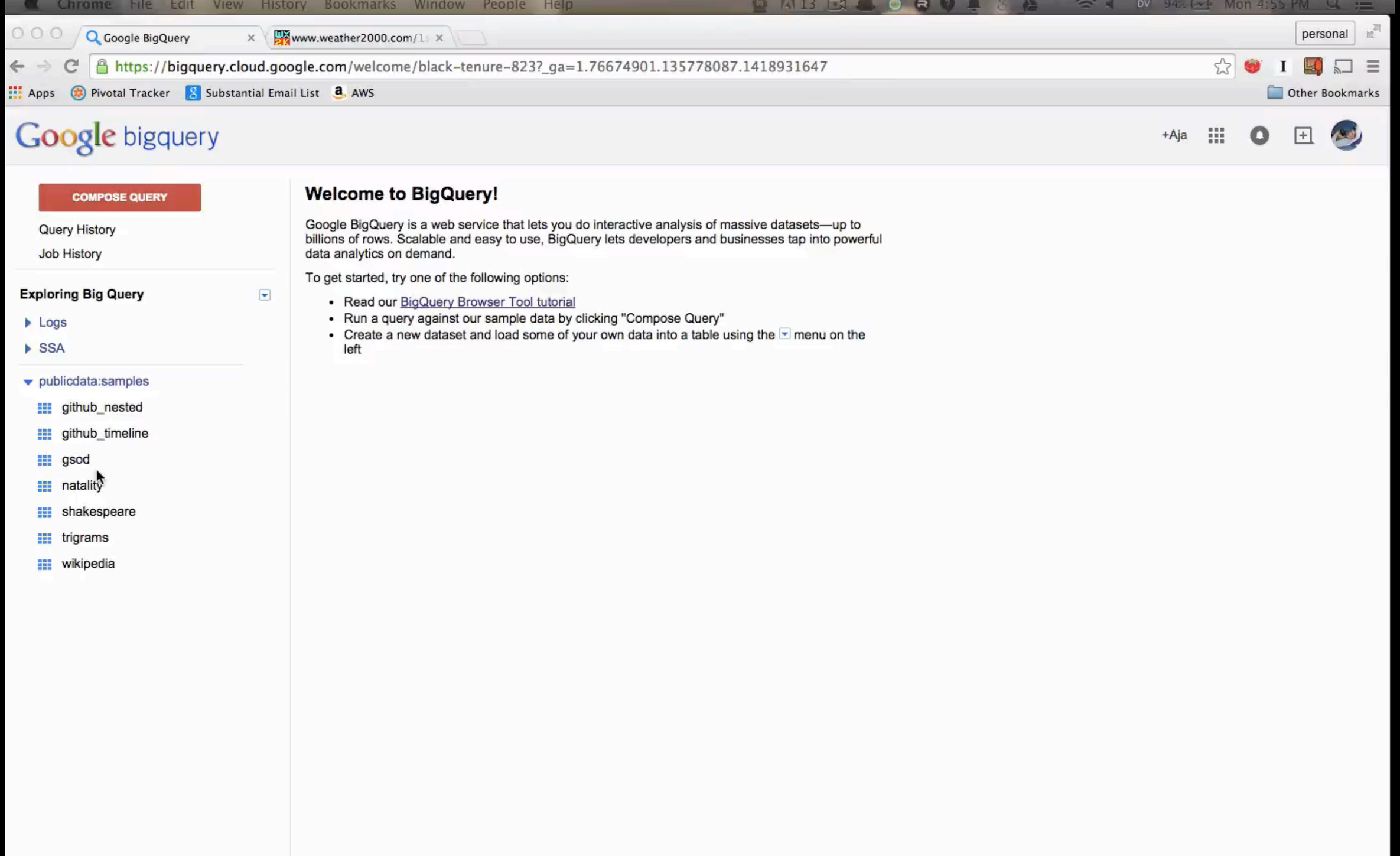
@thagomizer_rb

Easy

@thagomizer_rb

Purpose Built


The Rain Off!!!!!
https://www.flickr.com/photos/hansel5569/7856258411


Chrome File Edit View History Bookmarks Window People Help
Google BigQuery
www.weather2000.com/1
personal
https://bigquery.cloud.google.com/welcome/black-tenure-823?_ga=1.76674901.135778087.1418931647
Apps
Pivotal Tracker
Substantial Email List
AWS
Other Bookmarks
Google bigquery
+Aja
COMPOSE QUERY
Query History
Job History
Exploring Big Query
Logs
SSA
publicdata:samples
github_nested
github_timeline
gsod
natality
shakespeare
trigrams
wikipedia
Welcome to BigQuery!
Google BigQuery is a web service that lets you do interactive analysis of massive datasets—up to billions of rows. Scalable and easy to use, BigQuery lets developers and businesses tap into powerful data analytics on demand.
To get started, try one of the following options:
Read our BigQuery Browser Tool tutorial
Run a query against our sample data by clicking "Compose Query"
Create a new dataset and load some of your own data into a table using the menu on the left

How To

@thagomizer_rb

Step 1:
Extract

Step 2: Upload

Step 3:
Query

Step 4:
Understand

Step 4:

~~Understand~~

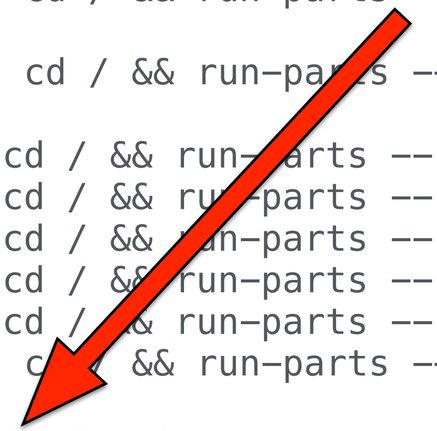
Hopefully Understand

Context

Extract

```
Apr  4 06:25:01 frontend-rails CRON[13907]: (root) CMD (test -x /usr/sbin/anacron || ( cd / && run-
parts --report /etc/cron.daily ))
Apr  2 07:17:01 frontend-rails CRON[18353]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 08:17:01 frontend-rails CRON[20319]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 09:17:01 frontend-rails CRON[22285]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 10:17:01 frontend-rails CRON[24241]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 11:17:01 frontend-rails CRON[26202]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 12:17:01 frontend-rails CRON[28162]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 13:17:01 frontend-rails CRON[30126]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 14:17:01 frontend-rails CRON[32093]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 15:17:01 frontend-rails CRON[1654]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 16:17:01 frontend-rails CRON[3624]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 17:17:01 frontend-rails CRON[5620]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 18:17:01 frontend-rails CRON[7591]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 19:17:01 frontend-rails CRON[9558]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 20:17:01 frontend-rails CRON[11513]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 21:09:22 frontend-rails google-address-manager: ERROR SyncAddresses exception: HTTP Error 503:
Service Unavailable
Apr  2 21:09:23 frontend-rails kernel: [182043.624965] Clocksource tsc unstable (delta = 241092263 ns)
```

```
Apr  4 06:25:01 frontend-rails CRON[13907]: (root) CMD (test -x /usr/sbin/anacron || ( cd / && run-
parts --report /etc/cron.daily ))
Apr  2 07:17:01 frontend-rails CRON[18353]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 08:17:01 frontend-rails CRON[20319]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 09:17:01 frontend-rails CRON[22285]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 10:17:01 frontend-rails CRON[24241]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 11:17:01 frontend-rails CRON[26202]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 12:17:01 frontend-rails CRON[28162]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 13:17:01 frontend-rails CRON[30126]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 14:17:01 frontend-rails CRON[32093]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 15:17:01 frontend-rails CRON[1654]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 16:17:01 frontend-rails CRON[3624]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 17:17:01 frontend-rails CRON[5620]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 18:17:01 frontend-rails CRON[7591]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 19:17:01 frontend-rails CRON[9558]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr  2 20:17:01 frontend-rails CRON[11513]: (root) CMD (   cd / && run-parts --report /etc/
cron.hourly)
Apr  2 21:09:22 frontend-rails google-address-manager: ERROR SyncAddresses exception: HTTP Error 503:
Service Unavailable
Apr  2 21:09:23 frontend-rails kernel: [182043.624965] Clocksource tsc unstable (delta = 241092263 ns)
```



@thagomizer_rb

```
Mar  9 07:17:01 frontend-rails CRON[21078]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 08:17:01 frontend-rails CRON[23112]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 09:17:01 frontend-rails CRON[25099]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 10:17:01 frontend-rails CRON[27096]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 11:17:01 frontend-rails CRON[29140]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 12:17:01 frontend-rails CRON[31125]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 13:17:01 frontend-rails CRON[642]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 14:17:01 frontend-rails CRON[2720]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 15:17:01 frontend-rails CRON[4805]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 16:17:01 frontend-rails CRON[6839]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 17:17:01 frontend-rails CRON[8824]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 18:17:01 frontend-rails CRON[10858]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 19:17:01 frontend-rails CRON[12897]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 20:17:01 frontend-rails CRON[14935]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 21:17:01 frontend-rails CRON[16915]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 22:17:01 frontend-rails CRON[18964]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar  9 23:17:01 frontend-rails CRON[20965]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 00:17:01 frontend-rails CRON[22941]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 01:17:01 frontend-rails CRON[24920]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 02:17:01 frontend-rails CRON[26980]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 03:17:01 frontend-rails CRON[28952]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 04:17:01 frontend-rails CRON[30904]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 05:17:01 frontend-rails CRON[354]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 06:17:01 frontend-rails CRON[2949]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 06:25:01 frontend-rails CRON[3204]: (root) CMD (test -x /usr/sbin/anacron || ( cd / && run-parts --report /
etc/cron.daily ))
Mar 10 07:17:01 frontend-rails CRON[5017]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 08:17:01 frontend-rails CRON[6954]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 09:17:01 frontend-rails CRON[8899]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 10:17:01 frontend-rails CRON[10840]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 11:17:01 frontend-rails CRON[12775]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Mar 10 12:17:01 frontend-rails CRON[14736]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
```

@thagomizer_rb

```
Mar  9 07:17:01 frontend-rails CRON[21078]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 08:17:01 frontend-rails CRON[23112]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 09:17:01 frontend-rails CRON[25099]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 10:17:01 frontend-rails CRON[27096]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 11:17:01 frontend-rails CRON[29140]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 12:17:01 frontend-rails CRON[31125]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 13:17:01 frontend-rails CRON[642]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar  9 14:17:01 frontend-rails CRON[2720]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar  9 15:17:01 frontend-rails CRON[4805]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar  9 16:17:01 frontend-rails CRON[6839]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar  9 17:17:01 frontend-rails CRON[8824]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar  9 18:17:01 frontend-rails CRON[10858]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 19:17:01 frontend-rails CRON[12897]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 20:17:01 frontend-rails CRON[14935]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 21:17:01 frontend-rails CRON[16915]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar  9 22:17:01 frontend-rails CRON[18964]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
```

```
Mar  9 23:17:01 frontend-rails CRON[20965]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 00:17:01 frontend-rails CRON[22941]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 01:17:01 frontend-rails CRON[24920]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 02:17:01 frontend-rails CRON[26980]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 03:17:01 frontend-rails CRON[28952]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 04:17:01 frontend-rails CRON[30904]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 05:17:01 frontend-rails CRON[354]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar 10 06:17:01 frontend-rails CRON[2949]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar 10 06:25:01 frontend-rails CRON[3204]: (root) CMD
(test -x /usr/sbin/anacron || ( cd / && run-parts --
report /etc/cron.daily ))
Mar 10 07:17:01 frontend-rails CRON[5017]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar 10 08:17:01 frontend-rails CRON[6954]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar 10 09:17:01 frontend-rails CRON[8899]: (root) CMD (
cd / && run-parts --report /etc/cron.hourly)
Mar 10 10:17:01 frontend-rails CRON[10840]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 11:17:01 frontend-rails CRON[12775]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 12:17:01 frontend-rails CRON[14736]: (root) CMD
(   cd / && run-parts --report /etc/cron.hourly)
Mar 10 13:17:01 frontend-rails CRON[16652]: (root) CMD
```

@thagomizer_rb

```
Mar  9 07:17:01 frontend-
rails CRON[21078]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 08:17:01 frontend-
rails CRON[23112]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 09:17:01 frontend-
rails CRON[25099]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 10:17:01 frontend-
rails CRON[27096]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 11:17:01 frontend-
rails CRON[29140]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 12:17:01 frontend-
rails CRON[31125]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 13:17:01 frontend-
rails CRON[642]: (root) CMD (
cd / && run-parts --report /
etc/cron.hourly)
Mar  9 14:17:01 frontend-
rails CRON[2720]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 15:17:01 frontend-
rails CRON[4805]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar  9 16:17:01 frontend-
rails CRON[6839]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 05:17:01 frontend-
rails CRON[354]: (root) CMD (
cd / && run-parts --report /
etc/cron.hourly)
Mar 10 06:17:01 frontend-
rails CRON[2949]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 06:25:01 frontend-
rails CRON[3204]: (root) CMD
(test -x /usr/sbin/anacron ||
( cd / && run-parts --
report /etc/cron.daily ))
Mar 10 07:17:01 frontend-
rails CRON[5017]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 08:17:01 frontend-
rails CRON[6954]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 09:17:01 frontend-
rails CRON[8899]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 10:17:01 frontend-
rails CRON[10840]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 11:17:01 frontend-
rails CRON[12775]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 12:17:01 frontend-
rails CRON[14736]: (root) CMD
(   cd / && run-parts --
report /etc/cron.hourly)
Mar 10 13:17:01 frontend-
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
Linux version 3.16.0-31-
generic (buildd@tipua) (gcc
version 4.8.2 (Ubuntu
4.8.2-19ubuntu1) )
#41~14.04.1-Ubuntu SMP Wed
Feb 11 19:30:13 UTC 2015
(Ubuntu 3.16.0-31.41~14.04.1-
generic 3.16.7-ckt5)
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
Command line: BOOT_IMAGE=/
boot/vmlinuz-3.16.0-31-
generic
root=UUID=0e501679-6d73-47fa-
9cf9-558872950040 ro
console=ttyS0
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
KERNEL supported cpus:
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
  Intel GenuineIntel
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
  AMD AuthenticAMD
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
  Centaur CentaurHauls
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
Disabled fast string
operations
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
e820: BIOS-provided physical
RAM map:
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
e820: update [mem
0x00000000-0x00000fff] usable
==> reserved
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
e820: remove [mem
0x000a0000-0x000fffff] usable
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
AGP: No AGP bridge found
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
e820: last_pfn = 0x220000
max_arch_pfn = 0x400000000
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
MTRR default type: write-back
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
MTRR fixed ranges enabled:
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
00000-9FFFF write-back
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
A0000-BFFFF uncachable
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
C0000-FFFFF write-protect
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
MTRR variable ranges enabled:
Mar 10 17:26:42 frontend-
rails kernel: [    0.000000]
0 base 0000C0000000 mask
3FFFC0000000 uncachable
```

```
import re
import csv
import sys
from datetime import datetime

script, logfile, outfile = sys.argv
records = []

SYSLOG_RE = re.compile(r"""
    (\w{3}\ {1,2}\d{1,2}\ \d{2}:\d{2}:\d{2}) # Timestamp
    \s([\w-]*)                               # Application
    \s([\w-]*)                               # Source
    \[(\w+)\]?                               # Optional pid
    :
    (.*)                                     # Details
    """, re.VERBOSE)

with open(logfile, 'r') as f:
    for line in f:
        line = line.strip()
        if not line:
            continue
        d = {}

        match = SYSLOG_RE.search(line)
        if match:
            groups = match.groups()
            d['timestamp'] = groups[0]
            d['app']       = groups[1]
            d['source']    = groups[2]
            d['pid']       = groups[3]
            d['details']   = groups[4]
        else:
            d = {'timestamp':None, 'app':None, 'source':None,
                 'pid':None, 'details':None}

        d['raw'] = line
        records.append(d)

for record in records:
    temp = record.get('timestamp')
    if not temp:
        continue

    temp = datetime.strptime(temp, '%b %d %H:%M:%S')
    temp = temp.isoformat()
    record['timestamp'] = temp

with open(outfile, 'wb') as csvfile:
    logwriter = csv.writer(csvfile)
    logwriter.writerow(records[0].keys())
    for record in records:
        logwriter.writerow(record.values())
```

@thagomizer_rb

```
import re
import csv
import sys
from datetime import datetime
```

```
script, logfile, outfile = sys.argv

records = []
```

```
SYSLOG_RE = re.compile(r"""
    (\w{3}\ {1,2}\d{1,2}\ \d{2}:\d{2}:\d{2}) # Timestamp
    \s([\w-]*)                               # Application
    \s([\w-]*)                               # Source
    \[(\w+)\]?                               # Optional pid
    :
    (.*)                                     # Details
    """, re.VERBOSE)
```

```
with open(logfile, 'r') as f:
    for line in f:
        line = line.strip()
        if not line:
            continue

        d = {}

        match = SYSLOG_RE.search(line)
        if match:
            groups = match.groups()
            d['timestamp'] = groups[0]
            d['app']       = groups[1]
            d['source']    = groups[2]
            d['pid']       = groups[3]
            d['details']   = groups[4]
        else:
            d = {'timestamp':None, 'app':None, 'source':None,
                 'pid':None, 'details':None}

        d['raw'] = line
        records.append(d)
```

```
with open(logfile, 'r') as f:
    for line in f:
        line = line.strip()
        if not line:
            continue
```

```
with open(logfile, 'r') as f:
    for line in f:
        line = line.strip()
        if not line:
            continue
```

```
with open(logfile, 'r') as f:
    for line in f:
        line = line.strip()
        if not line:
            continue

        d = {}

        match = SYSLOG_RE.search(line)
        if match:
            groups = match.groups()
            d['timestamp'] = groups[0]
            d['app']       = groups[1]
            d['source']    = groups[2]
            d['pid']       = groups[3]
            d['details']   = groups[4]
        else:
            d = {'timestamp':None, 'app':None, 'source':None,
                 'pid':None, 'details':None}

        d['raw'] = line
        records.append(d)
```

@thagomizer_rb

```
with open(logfile, 'r') as f:
    for line in f:
        line = line.strip()
        if not line:
            continue

        d = {}

        match = SYSLOG_RE.search(line)
        if match:
            groups = match.groups()
            d['timestamp'] = groups[0]
            d['app']       = groups[1]
            d['source']    = groups[2]
            d['pid']       = groups[3]
            d['details']   = groups[4]
        else:
            d = {'timestamp':None, 'app':None, 'source':None,
                 'pid':None, 'details':None}

        d['raw'] = line
        records.append(d)
```

```
d = {}

match = SYSLOG_RE.search(line)
if match:
    groups = match.groups()
    d['timestamp'] = groups[0]
    d['app']       = groups[1]
    d['source']    = groups[2]
    d['pid']       = groups[3]
    d['details']   = groups[4]
else:
    d = {'timestamp':None, 'app':None, 'source':None,
         'pid':None, 'details':None}

d['raw'] = line
records.append(d)
```

```
d = {}

match = SYSLOG_RE.search(line)
if match:
    groups = match.groups()
    d['timestamp'] = groups[0]
    d['app']       = groups[1]
    d['source']    = groups[2]
    d['pid']       = groups[3]
    d['details']   = groups[4]
else:
    d = {'timestamp':None, 'app':None, 'source':None,
         'pid':None, 'details':None}

d['raw'] = line
records.append(d)
```

```
d = {}

match = SYSLOG_RE.search(line)
if match:
    groups = match.groups()
    d['timestamp'] = groups[0]
    d['app']       = groups[1]
    d['source']    = groups[2]
    d['pid']       = groups[3]
    d['details']   = groups[4]
else:
    d = {'timestamp':None, 'app':None, 'source':None,
         'pid':None, 'details':None}

d['raw'] = line
records.append(d)
```

```
d = {}

match = SYSLOG_RE.search(line)
if match:
    groups = match.groups()
    d['timestamp'] = groups[0]
    d['app']       = groups[1]
    d['source']    = groups[2]
    d['pid']       = groups[3]
    d['details']   = groups[4]
else:
    d = {'timestamp':None, 'app':None, 'source':None,
         'pid':None, 'details':None}

d['raw'] = line
records.append(d)
```

@thagomizer_rb

```
for record in records:
    temp = record.get('timestamp')
    if not temp:
        continue

    temp = datetime.strptime(temp, '%b %d %H:%M:%S')
    temp = temp.isoformat()
    record['timestamp'] = temp
```

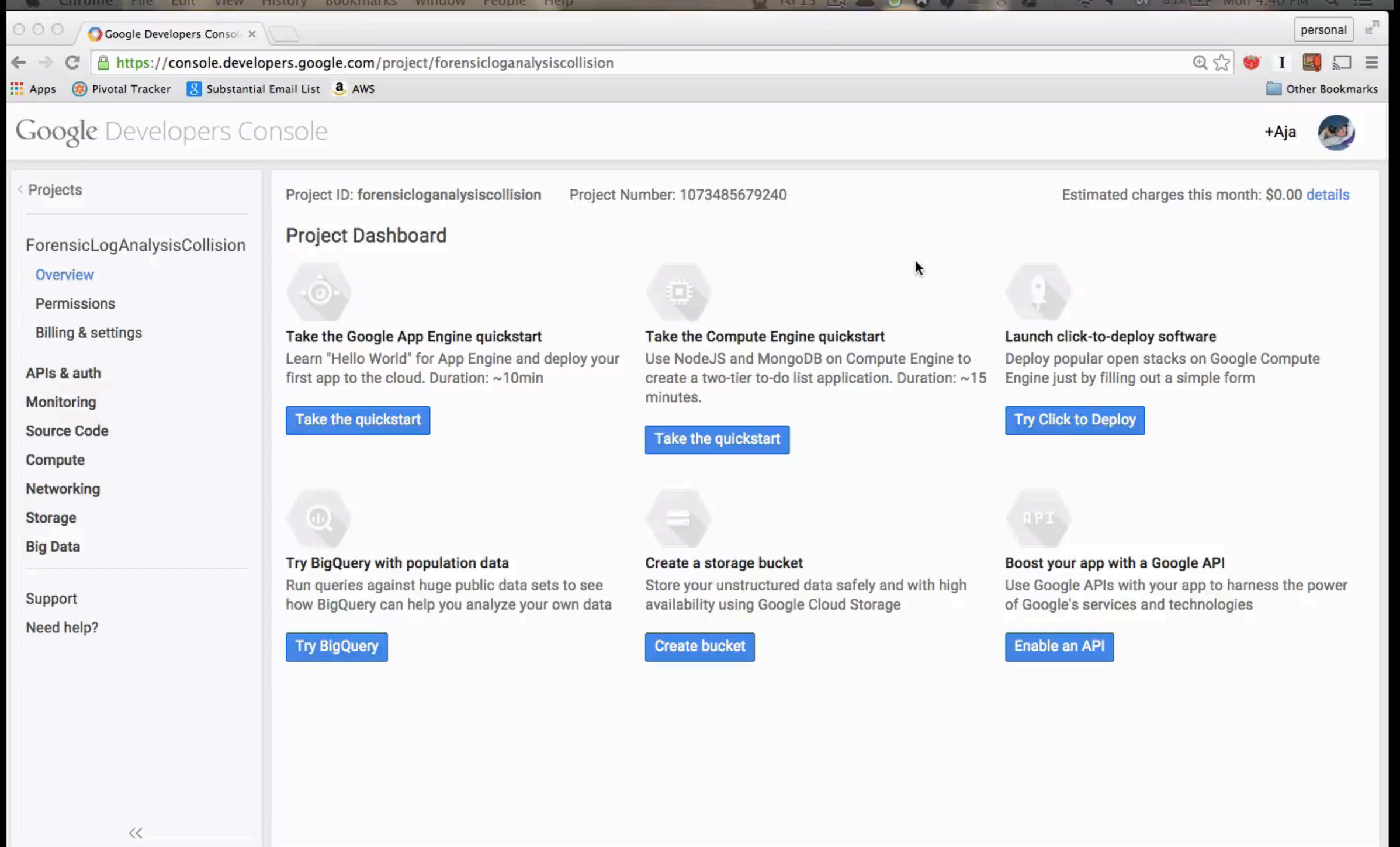
```
with open(outfile, 'wb') as csvfile:
    logwriter = csv.writer(csvfile)
    logwriter.writerow(records[0].keys())
    for record in records:
        logwriter.writerow(record.values())
```

```
with open(outfile, 'wb') as csvfile:
    logwriter = csv.writer(csvfile)
    logwriter.writerow(records[0].keys())
    for record in records:
        logwriter.writerow(record.values())
```

```
source,timestamp,app,pid,raw,details
kernel,Mar 10 17:26:42,frontend-rails,,Mar 10 17:26:42 frontend-rails kernel:
[    0.000000] Linux version 3.16.0-31-generic (buildd@tipua) (gcc version
4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #41~14.04.1-Ubuntu SMP Wed Feb 11 19:30:13
UTC 2015 (Ubuntu 3.16.0-31.41~14.04.1-generic 3.16.7-ckt5), [    0.000000]
Linux version 3.16.0-31-generic (buildd@tipua) (gcc version 4.8.2 (Ubuntu
4.8.2-19ubuntu1) ) #41~14.04.1-Ubuntu SMP Wed Feb 11 19:30:13 UTC 2015
(Ubuntu 3.16.0-31.41~14.04.1-generic 3.16.7-ckt5)
kernel,Mar 10 17:26:42,frontend-rails,,Mar 10 17:26:42 frontend-rails kernel:
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-3.16.0-31-generic
root=UUID=0e501679-6d73-47fa-9cf9-558872950040 ro console=ttyS0,
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-3.16.0-31-generic
root=UUID=0e501679-6d73-47fa-9cf9-558872950040 ro console=ttyS0
kernel,Mar 10 17:26:42,frontend-rails,,Mar 10 17:26:42 frontend-rails kernel:
[    0.000000] KERNEL supported cpus:, [    0.000000] KERNEL supported cpus:
kernel,Mar 10 17:26:42,frontend-rails,,Mar 10 17:26:42 frontend-rails kernel:
[    0.000000]   Intel GenuineIntel, [    0.000000]   Intel GenuineIntel
kernel,Mar 10 17:26:42,frontend-rails,,Mar 10 17:26:42 frontend-rails kernel:
[    0.000000]   AMD AuthenticAMD, [    0.000000]   AMD AuthenticAMD
```

Upload

Cloud Storage


Google Developers Consol
personal
https://console.developers.google.com/project/forensicloganalysiscollision
Apps
Pivotal Tracker
Substantial Email List
AWS
Other Bookmarks
Google Developers Console
+Aja
Projects
ForensicLogAnalysisCollision
Overview
Permissions
Billing & settings
APIs & auth
Monitoring
Source Code
Compute
Networking
Storage
Big Data
Support
Need help?
Project ID: forensicloganalysiscollision
Project Number: 1073485679240
Estimated charges this month: $0.00 details
Project Dashboard
Take the Google App Engine quickstart
Learn "Hello World" for App Engine and deploy your first app to the cloud. Duration: ~10min
Take the quickstart
Take the Compute Engine quickstart
Use NodeJS and MongoDB on Compute Engine to create a two-tier to-do list application. Duration: ~15 minutes.
Take the quickstart
Launch click-to-deploy software
Deploy popular open stacks on Google Compute Engine just by filling out a simple form
Try Click to Deploy
Try BigQuery with population data
Run queries against huge public data sets to see how BigQuery can help you analyze your own data
Try BigQuery
Create a storage bucket
Store your unstructured data safely and with high availability using Google Cloud Storage
Create bucket
Boost your app with a Google API
Use Google APIs with your app to harness the power of Google's services and technologies
Enable an API

Schema

```
string
integer
float
boolean
record
timestamp
```

```
source:string,
timestamp:timestamp,
app:string,
pid:integer,
raw:string,
details:string
```

TODO: No indexes

Chrome File Edit View History Bookmarks Window People Help

Google Developers Consol | Google BigQuery

https://bigquery.cloud.google.com/welcome/forensicloganalysiscollision?_ga=1.86085720.135778087.1418931647

Apps | Pivotal Tracker | Substantial Email List | AWS | Other Bookmarks

Google bigquery

+Aja

Dataset deleted.

COMPOSE QUERY

Query History

Job History

ForensicLogAnalysisCollision

No datasets found in this project.

Please create a dataset or select a new project from the menu above.

publicdata:samples

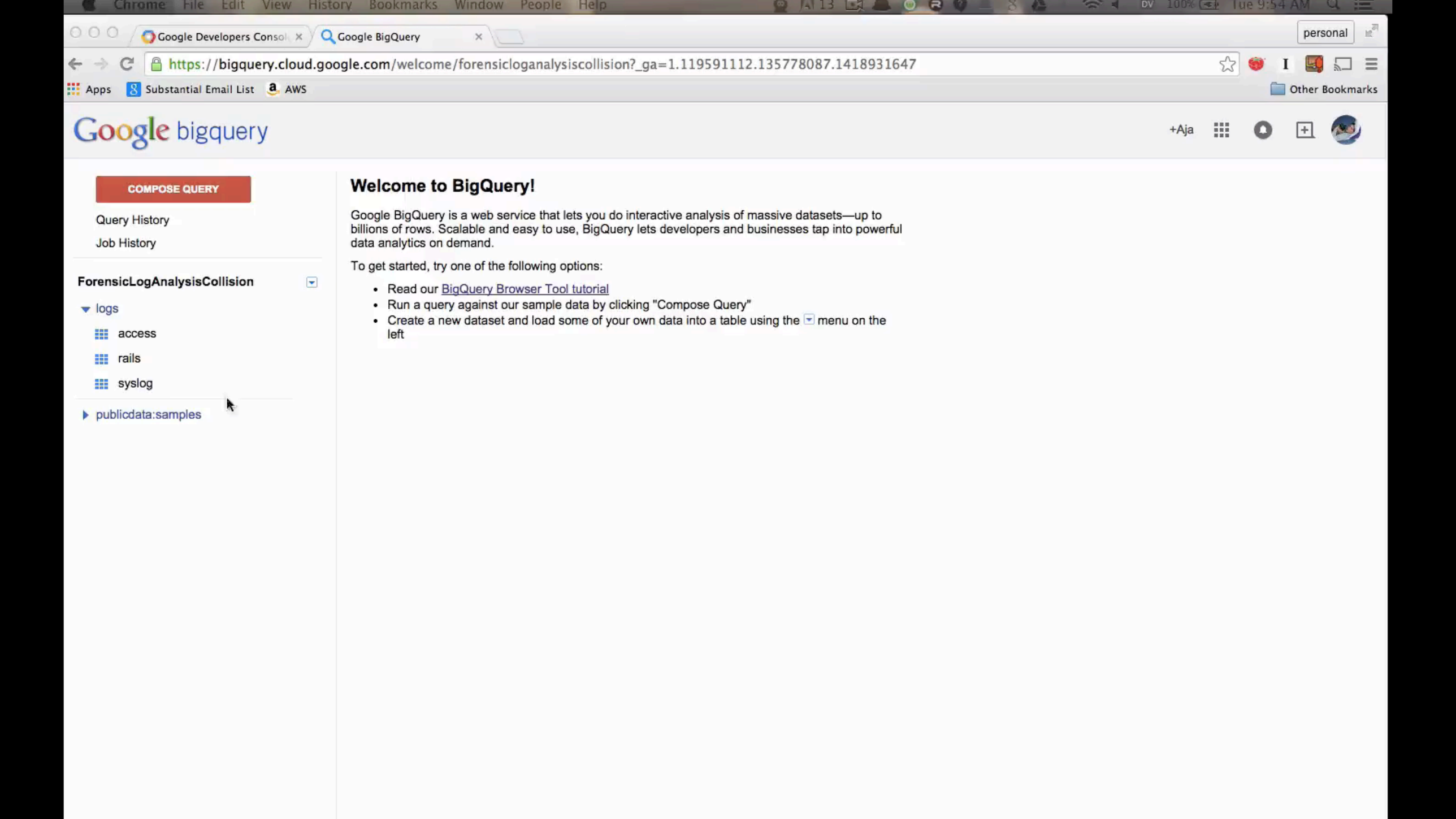
Welcome to BigQuery!

Google BigQuery is a web service that lets you do interactive analysis of massive datasets—up to billions of rows. Scalable and easy to use, BigQuery lets developers and businesses tap into powerful data analytics on demand.

To get started, try one of the following options:

- Read our BigQuery Browser Tool tutorial
- Run a query against our sample data by clicking "Compose Query"
- Create a new dataset and load some of your own data into a table using the menu on the left

Query


Chrome File Edit View History Bookmarks Window People Help
Google Developers Consol
Google BigQuery
personal
https://bigquery.cloud.google.com/welcome/forensicloganalysiscollision?_ga=1.119591112.135778087.1418931647
Apps
Substantial Email List
AWS
Other Bookmarks
Google bigquery
+Aja
COMPOSE QUERY
Query History
Job History
ForensicLogAnalysisCollision
logs
access
rails
syslog
publicdata:samples
Welcome to BigQuery!
Google BigQuery is a web service that lets you do interactive analysis of massive datasets—up to billions of rows. Scalable and easy to use, BigQuery lets developers and businesses tap into powerful data analytics on demand.
To get started, try one of the following options:
Read our BigQuery Browser Tool tutorial
Run a query against our sample data by clicking "Compose Query"
Create a new dataset and load some of your own data into a table using the menu on the left

Understand

Rendering

Redirects

Why

Retroactive

Simple

@thagomizer_rb

Fast

@thagomizer_rb

Share

@thagomizer_rb

Cost

Learn More

@thagomizer_rb

Real-time logs analysis using Fluentd and BigQuery

@thagomizer_rb

BigQuery Web UI Quickstart

@thagomizer_rb

r/bigquery

TODO: StackOverflow

Google Cloud Logging

@thagomizer_rb

Thank You

@thagomizer_rb

Questions?

@thagomizer_rb