Database Defense in Depth

Geoffrey Anderson
Percona Live 2015

Geoffrey Anderson

- DBA @ Box, Inc.
- MySQL, HBase, and more!
- #DBHangOps


box

OUR VISION:

Share, manage and access your content from any device, anywhere

275K+
BUSINESSES

32MM+
USERS

99%
FORTUNE 500

Global Enterprises Trust Box

HEALTHCARE	INDUSTRIAL	HIGH TECH	MEDIA	RETAIL	SERVICES
Beaumont Health System	Sunbelt Rentals	Rosetta Stone	food network	P&G	GREY group
Wake Forest School of Medicine	Webcor Builders	O2	Discovery Communications	Aéropostale	AARP
St. Jude Medical	AECOM	McAfee	PANDORA	Mattress Firm	Ameriprise Financial
Garden City Hospital	Hermance Machine Company	Avago Technologies	O2	Jamba Juice	Southwest Airlines

Defenses? For a Database?

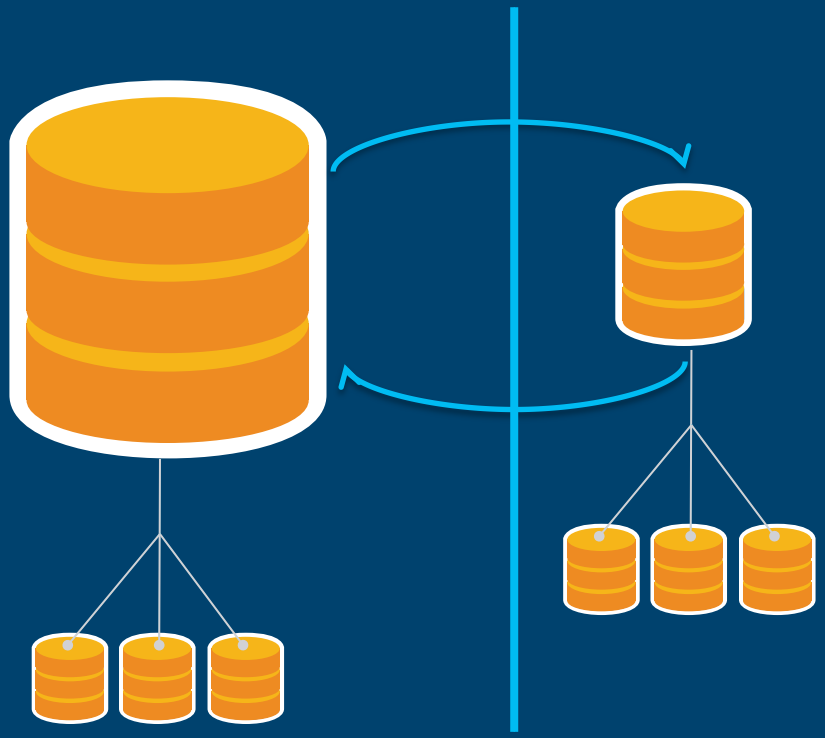
Huh?

Defenses

- Query Tuning
- Anemometer
- Raingauge
- Table Partitioning
- User Management / Least Privilege
- Graphs / Monitoring
  - OpenTSDB / Graphite / Kibana
  - statsd / tcollector
- user_statistics plugin
- Percona Server
- Proxying
- Sharding
- Vertically Scaling
- Replication
  - Multiple read replicas
- Backups
  - …and recovery!
- Binary Logs
- Percona Toolkit
  - pt-query-digest
  - pt-slave-delay
  - pt-stalk
  - pt-table-checksum
  - pt-table-sync
- Killing connections
- Killing bad queries
- Killing bad transactions
- Nagios alerts / PagerDuty
- Innotop
- Health checking

Let’s take a journey

You have your database



Queries

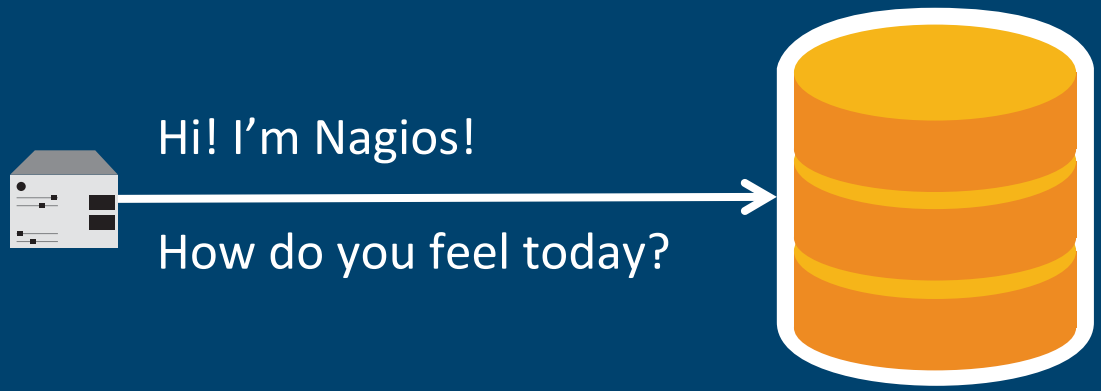
Yay business!

Queries

```
mysql [localhost] {root} (sakila) > show processlist;
+--------+----------------------+-----------+--------+---------+------+-----------------+-----------------------------------------------------------------------+
| Id     | User                 | Host      | db     | Command | Time | State           | Info                                                                  |
+--------+----------------------+-----------+--------+---------+------+-----------------+-----------------------------------------------------------------------+
|     10 | root                 | localhost | sakila | Query   |    0 | init            | show processlist                                                      |
| 426586 | unauthenticated user | localhost | NULL   | Connect | NULL | Reading from net | NULL                                                                  |
| 426596 | app                  | localhost | sakila | Query   |    0 | Writing to net  | SELECT * FROM rental WHERE customer_id=908 AND return_date IS NULL |
| 426597 | app                  | localhost | sakila | Query   |    0 | Opening tables  | SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL |
| 426598 | app                  | localhost | sakila | Query   |    0 | Opening tables  | SELECT * FROM rental WHERE customer_id=825 AND return_date IS NULL |
| 426599 | app                  | localhost | sakila | Query   |    0 | Opening tables  | SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL |
| 426600 | app                  | localhost | sakila | Query   |    0 | Opening tables  | SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL |
| 426601 | app                  | localhost | sakila | Query   |    0 | Opening tables  | SELECT * FROM rental WHERE customer_id=871 AND return_date IS NULL |
+--------+----------------------+-----------+--------+---------+------+-----------------+-----------------------------------------------------------------------+
139 rows in set (0.00 sec)
```

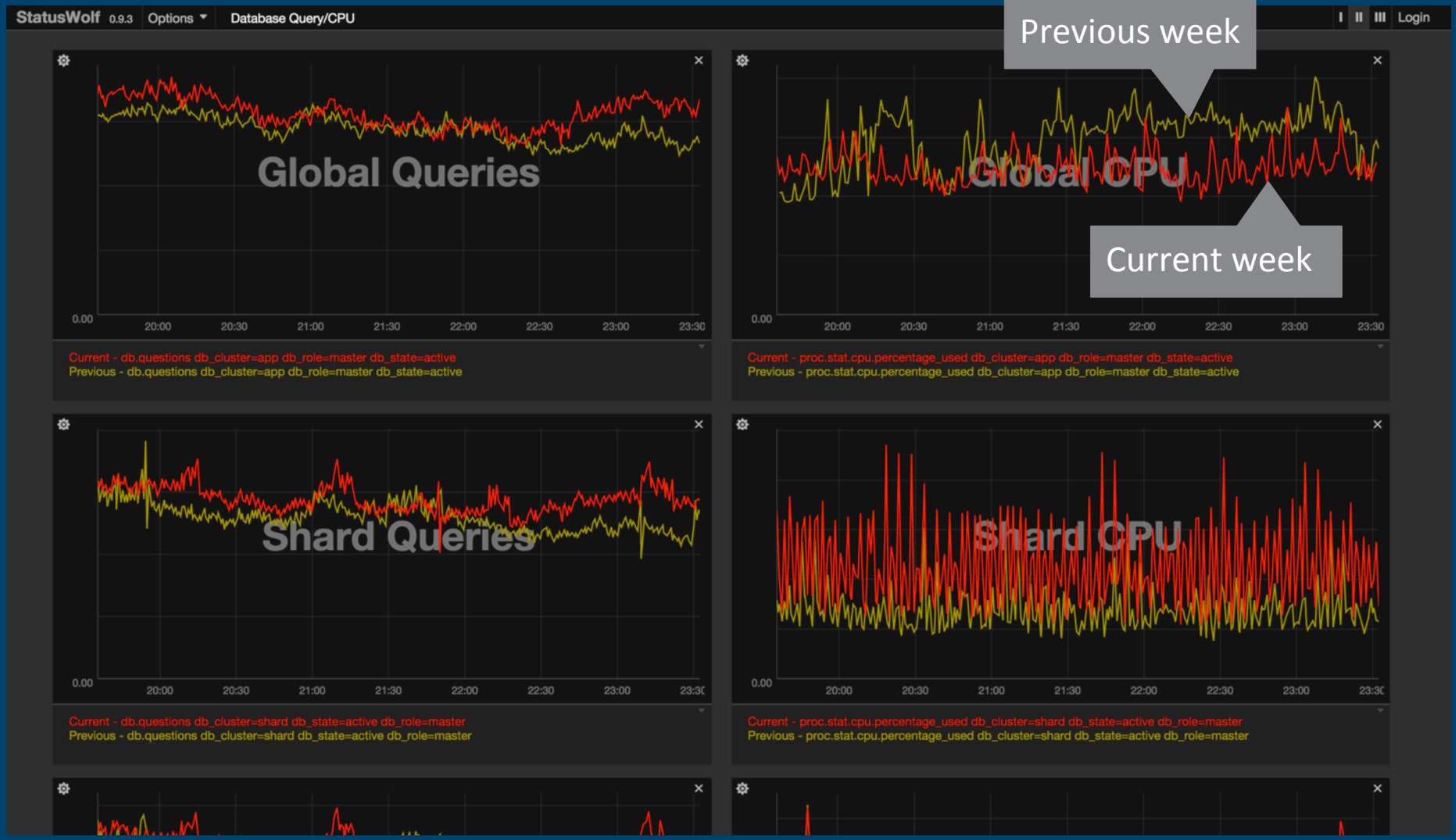
Defense #1 – Query comments!

Defense #2 – Alerting on the Database

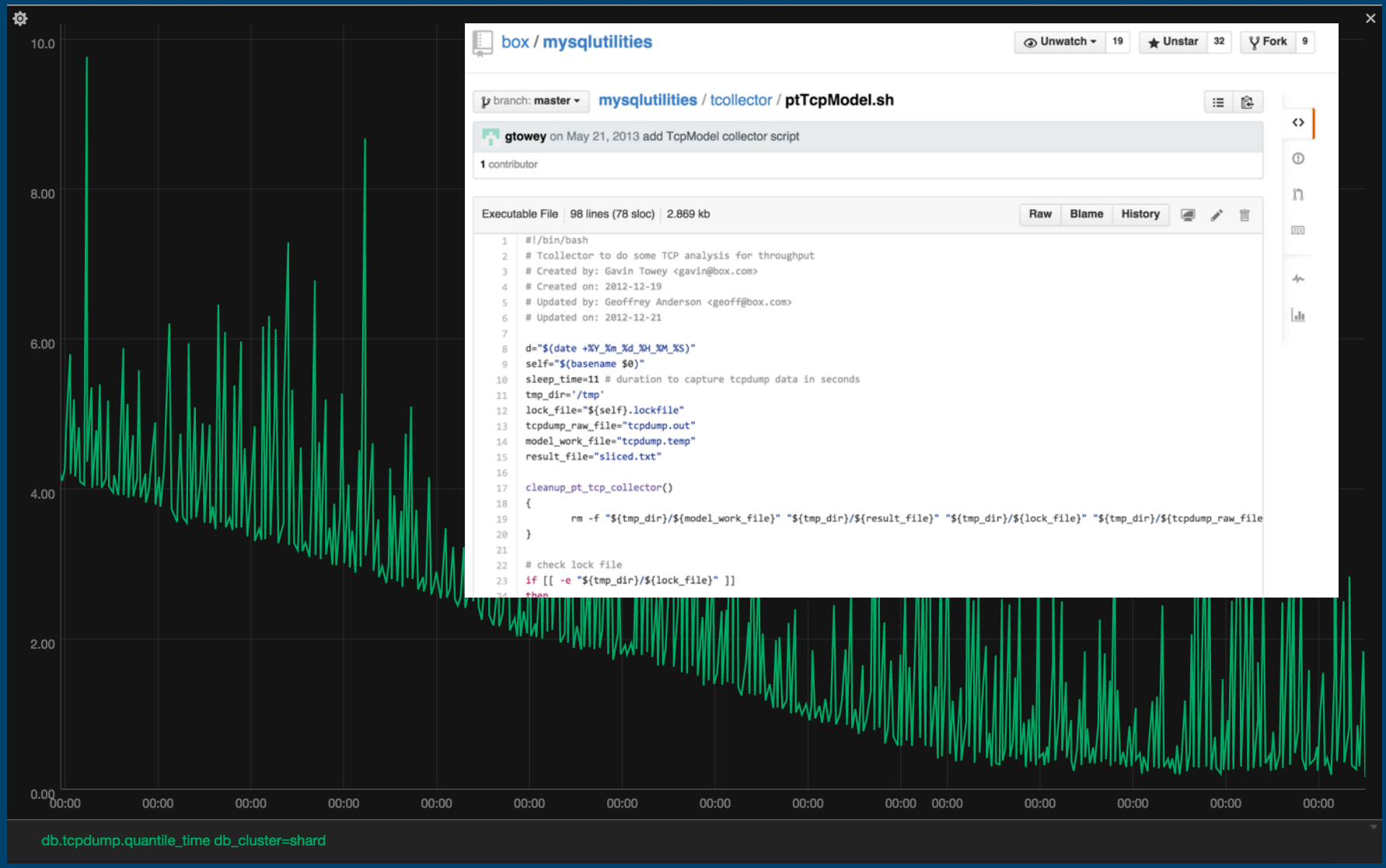


MySQL	OK	21:17:06	31d 9h 16m 24s	1/4	IO: Yes Slave SQL: Yes Seconds Behind Master: 0
MySQL - Long Queries	OK	21:16:56	31d 9h 18m 44s	1/4	OK - no long running queries
MySQL Config Changes	OK	21:14:56	26d 10h 19m 3s	1/4	OK creating temp file in /tmp/mysql_vars
MySQL Connections	OK	21:17:07	24d 7h 35m 45s	1/4	OK utilization=3% threads_connected=1043 max_connections=30000
MySQL History List	OK	21:14:56	31d 9h 16m 33s	1/4	Innodb_history_list_length = 1561
MySQL Read Only	OK	21:17:06	18d 10h 45m 54s	1/4	global.read_only = 0: db_role = master
MySQL Replication	OK	21:17:07	0d 1h 23m 6s	1/4	Replication OK.
MySQL Slow Query Log	OK	21:17:16	0d 0h 0m 6s	1/4	global.slow_query_log = 0

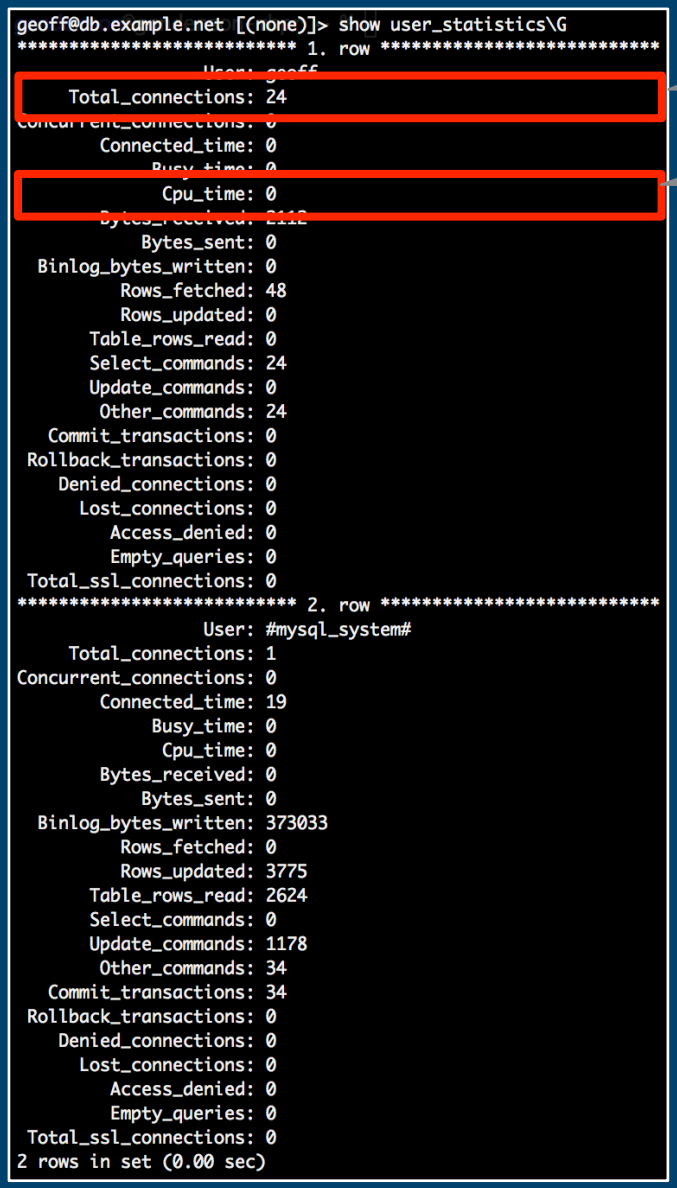
Defense #3 – Dashboards



Defense #3 – Dashboards



Defense #4 – User Statistics



```
geoff@db.example.net [(none)]> show user_statistics\G
*************************** 1. row ***************************
                  User: geoff
     Total_connections: 24
Concurrent_connections: 0
        Connected_time: 0
            Busy_time: 0
              Cpu_time: 0
         Bytes_received: 2112
             Bytes_sent: 0
   Binlog_bytes_written: 0
          Rows_fetched: 48
          Rows_updated: 0
       Table_rows_read: 0
       Select_commands: 24
       Update_commands: 0
        Other_commands: 24
    Commit_transactions: 0
  Rollback_transactions: 0
     Denied_connections: 0
       Lost_connections: 0
          Access_denied: 0
          Empty_queries: 0
  Total_ssl_connections: 0
*************************** 2. row ***************************
                  User: #mysql_system#
     Total_connections: 1
Concurrent_connections: 0
        Connected_time: 19
            Busy_time: 0
              Cpu_time: 0
         Bytes_received: 0
             Bytes_sent: 0
   Binlog_bytes_written: 373033
          Rows_fetched: 0
          Rows_updated: 3775
       Table_rows_read: 2624
       Select_commands: 0
       Update_commands: 1178
        Other_commands: 34
    Commit_transactions: 34
  Rollback_transactions: 0
     Denied_connections: 0
       Lost_connections: 0
          Access_denied: 0
          Empty_queries: 0
  Total_ssl_connections: 0
2 rows in set (0.00 sec)
```

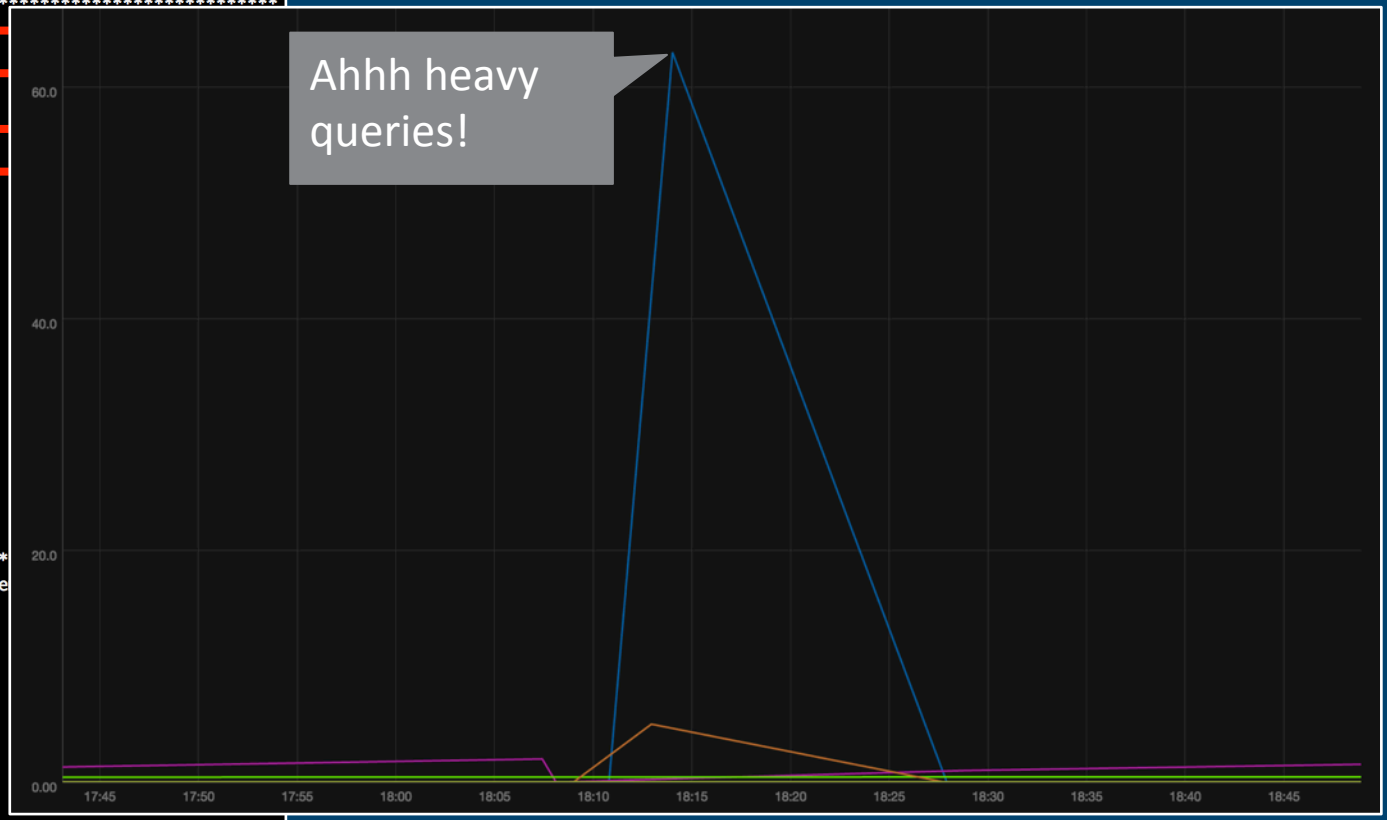
Per-user connection counts

Per-user cpu time

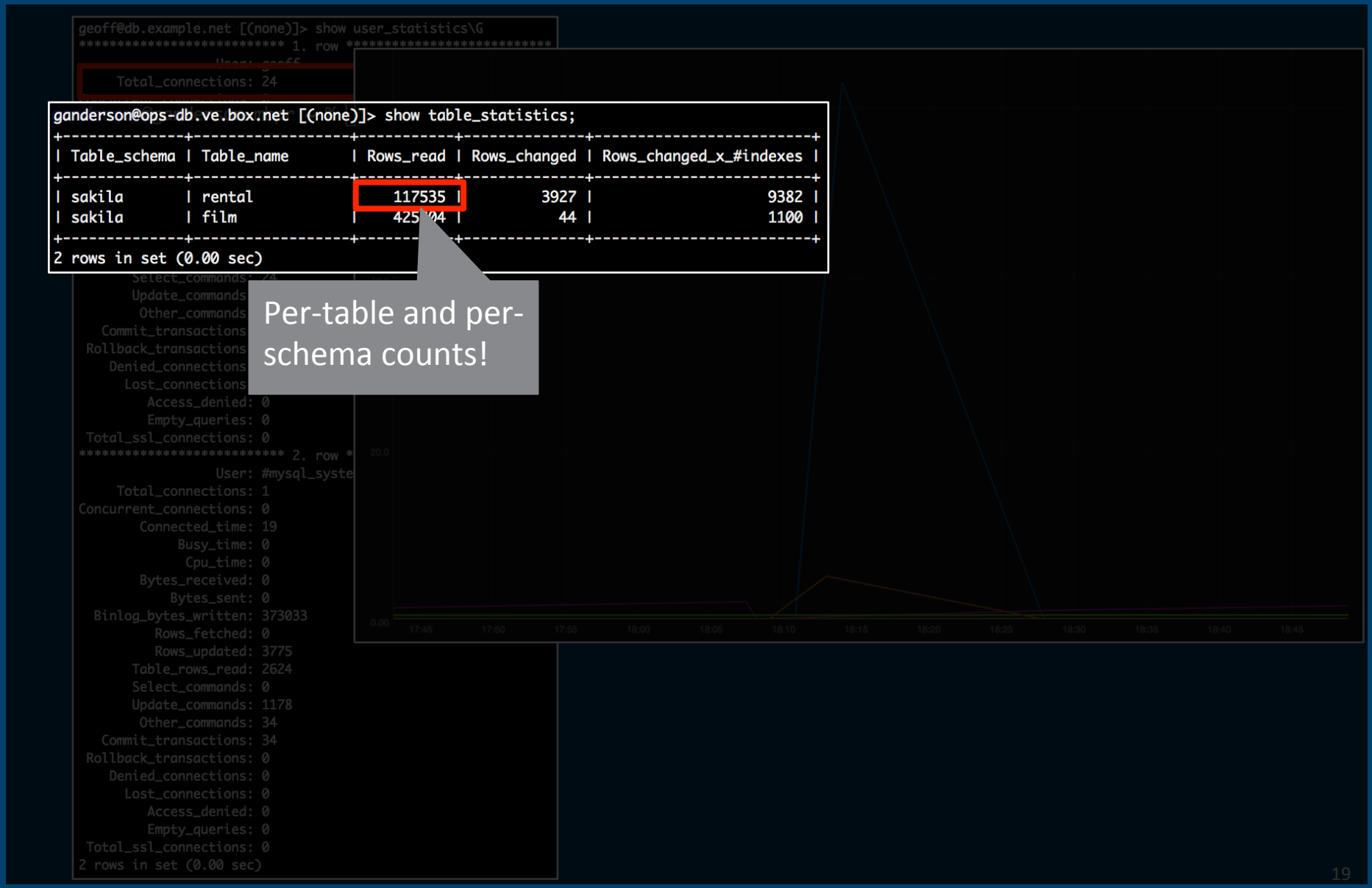
```
mysql> select * from sys.user_summary_by_file_io;
+------------+----------+-----------+
| root       |      159 | 5.11 h    |
| app        | 43777705 | 57.06 m   |
| background |     2174 | 754.67 ms |
+------------+----------+-----------+
3 rows in set (0.04 sec)
```

Defense #4 – User Statistics

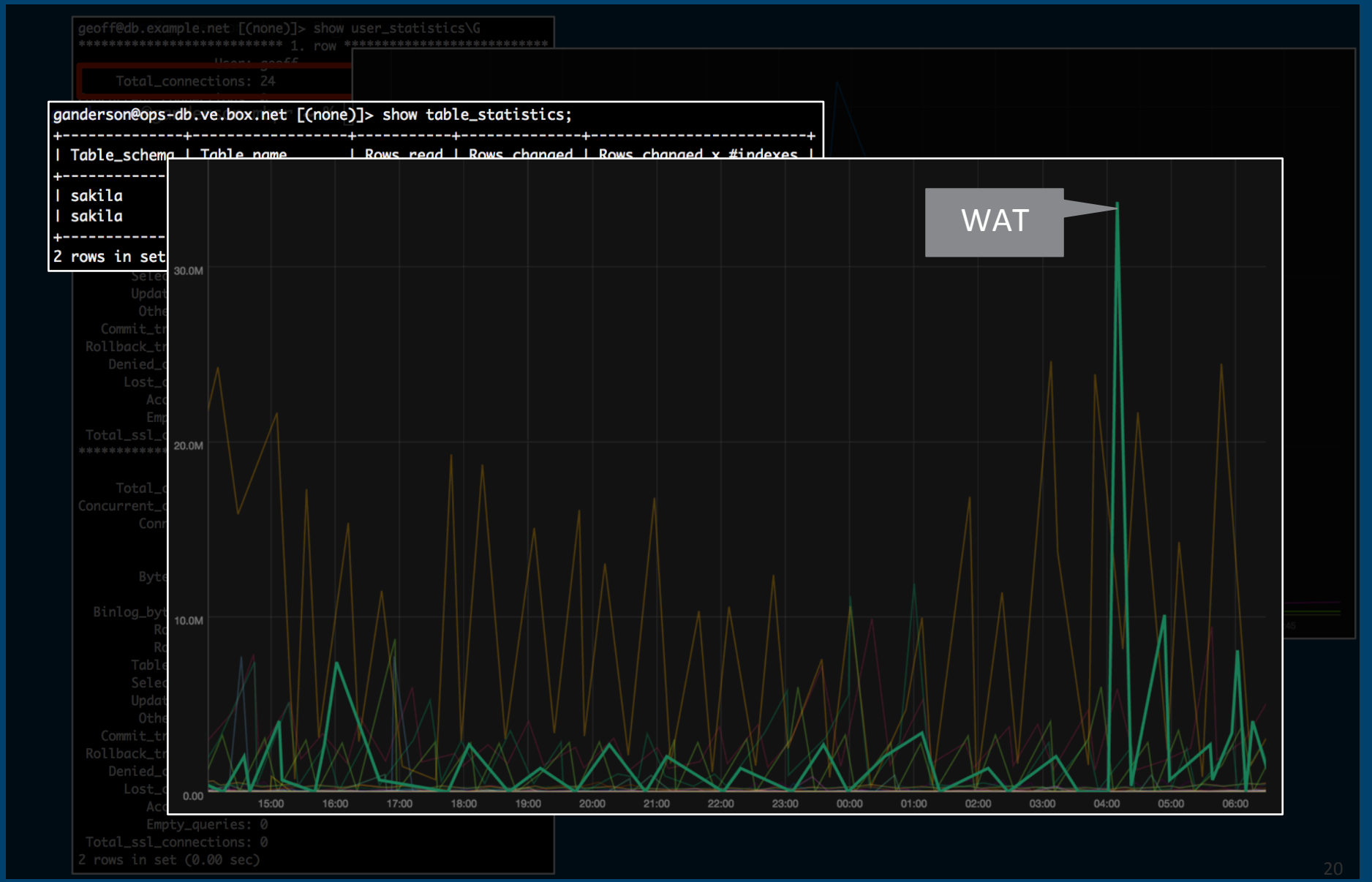
```
geoff@db.example.net [(none)]> show user_statistics\G
*************************** 1. row ***************************
                  User: geoff
     Total_connections: 24
Concurrent_connections: 0
        Connected_time: 0
             Busy_time: 0
              Cpu_time: 0
        Bytes_received: 2112
            Bytes_sent: 0
  Binlog_bytes_written: 0
          Rows_fetched: 48
          Rows_updated: 0
       Table_rows_read: 0
       Select_commands: 24
       Update_commands: 0
        Other_commands: 24
   Commit_transactions: 0
 Rollback_transactions: 0
    Denied_connections: 0
      Lost_connections: 0
         Access_denied: 0
         Empty_queries: 0
 Total_ssl_connections: 0
*************************** 2. row ***************************
                  User: #mysql_syste
     Total_connections: 1
Concurrent_connections: 0
        Connected_time: 19
             Busy_time: 0
              Cpu_time: 0
        Bytes_received: 0
            Bytes_sent: 0
  Binlog_bytes_written: 373033
          Rows_fetched: 0
          Rows_updated: 3775
       Table_rows_read: 2624
       Select_commands: 0
       Update_commands: 1178
        Other_commands: 34
   Commit_transactions: 34
 Rollback_transactions: 0
    Denied_connections: 0
      Lost_connections: 0
         Access_denied: 0
         Empty_queries: 0
 Total_ssl_connections: 0
2 rows in set (0.00 sec)
```



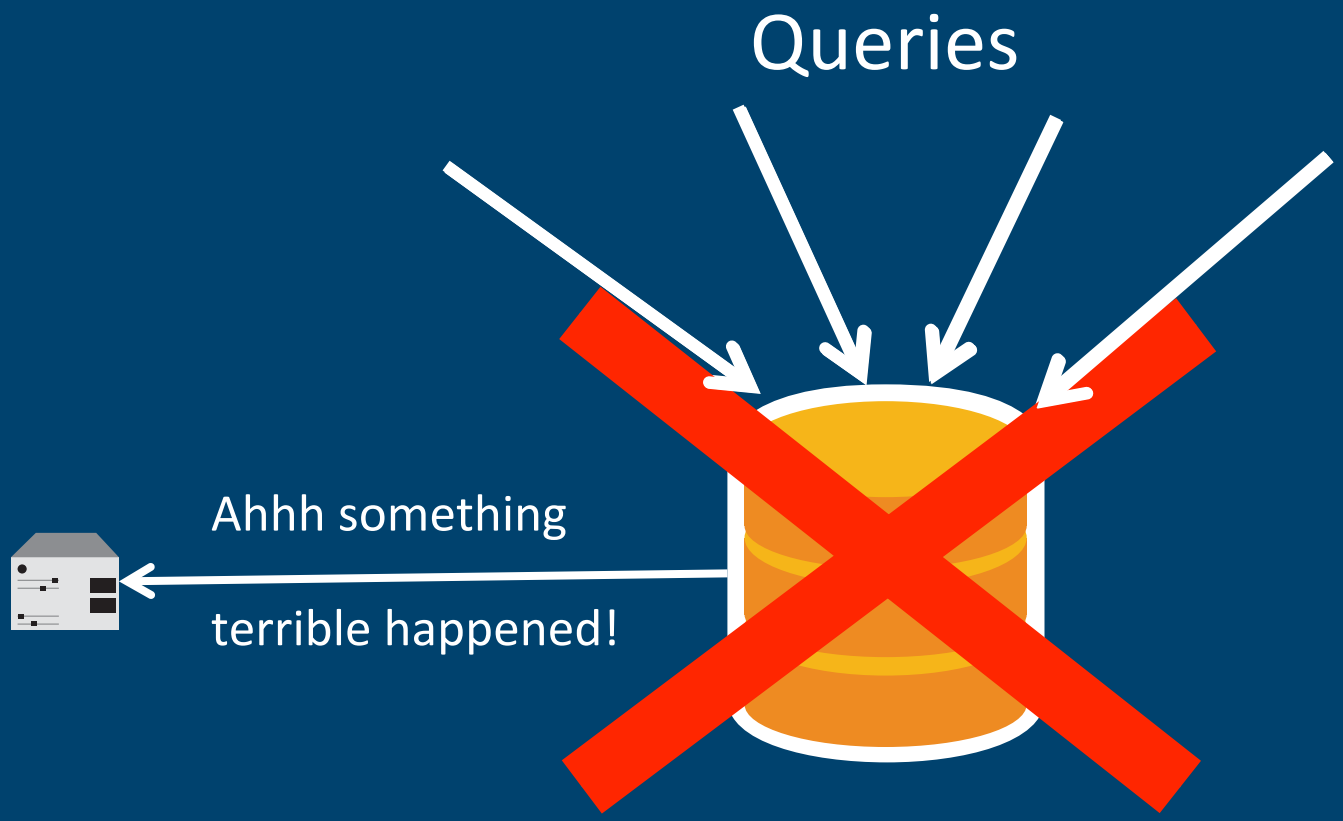
Defense #4 – User and Table statistics



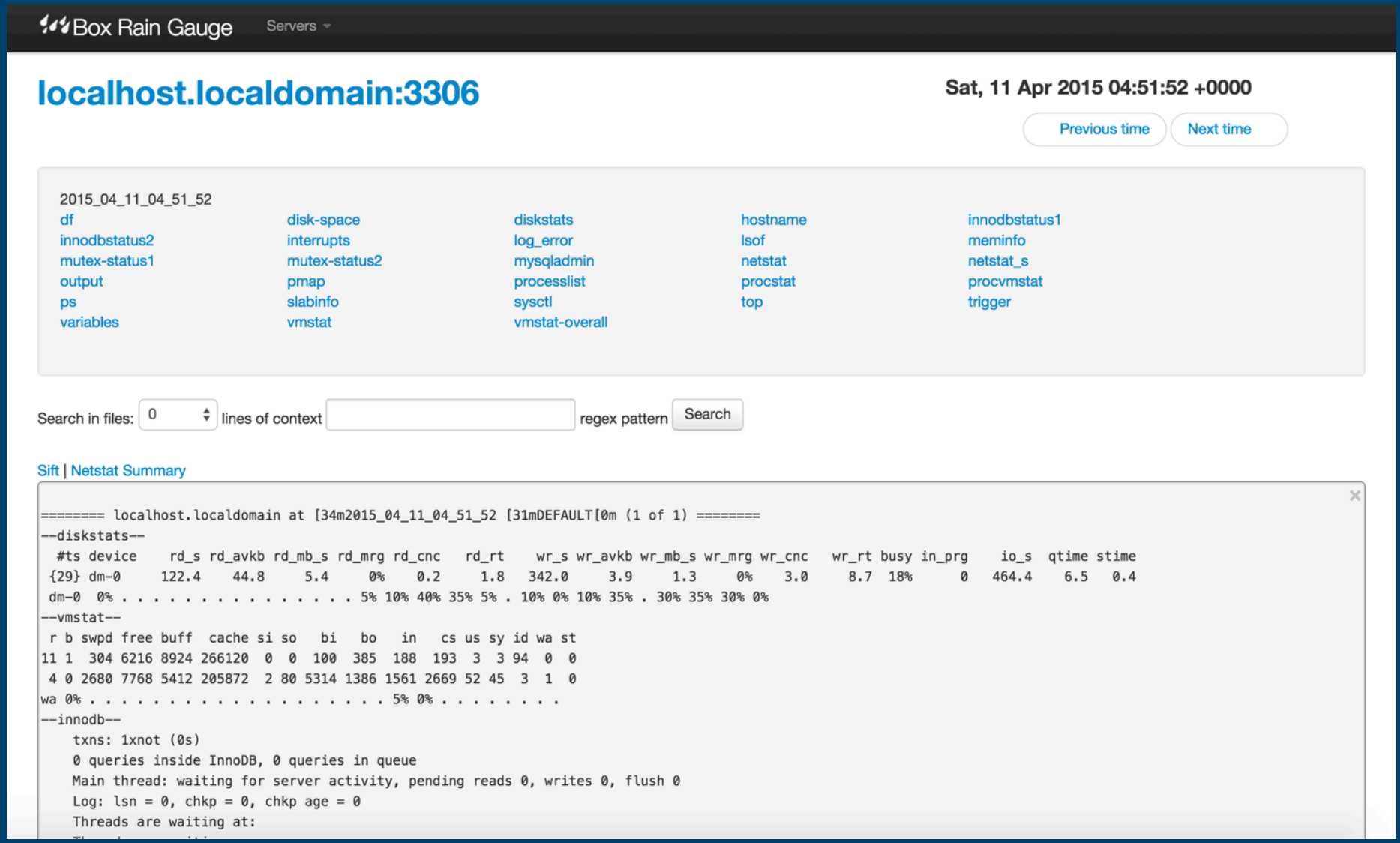
Defense #4 – User and Table statistics



Queries


Queries
Ahhh something
terrible happened!

Defense #5 – Box Raingauge



Defense #5 – Box Raingauge

Box Rain Gauge Servers

Sat, 11 Apr 2015 04:51:52 +0000

Previous time Next time

RainGauge, the new killer tool ?

By Cédric PEINTRE in Admin

09/09/2012 2 Comments

I'm sure you've heard of **Box Anemometer** from the Box (MySQL) team, an excellent UI tool based on *pt-query-digest*. Now the guys from the box team offer us another killer UI tool based on *pt-stalk*, and I'm sure you will really appreciate to use it !

What is it ?

"*RainGauge*" consists of three parts :

- A set of scripts that collect data about the health of your system and your databases (based on *pt-stalk*)
- A process that push these data on a centralised place
- A very simple interface to navigate in the collected data

But the collection begins only when specific conditions are triggered, and you choose what these conditions are. Find below the general flowsheet :

box

Anemometer & Raingauge

Gavin Towey
Senior Database Administrator @ Box
gavin@box.com

```
Main thread: waiting for server activity, pending reads 0, writes 0, flush 0
Log: lsn = 0, chkp = 0, chkp age = 0
Threads are waiting at:
```

```
mysql [localhost] {root} ((none)) > show processlist;
+-------+------+-----------+--------+---------+------+-------------+-----------------------------------------------------------------------------------------------------------------------+
| Id    | User | Host      | db     | Command | Time | State       | Info                                                                                                                  |
+-------+------+-----------+--------+---------+------+-------------+-----------------------------------------------------------------------------------------------------------------------+
|     1 | root | localhost | NULL   | Query   |    0 | init        | show processlist                                                                                                      |
|   490 | app  | localhost | NULL   | Sleep   |  162 |             | NULL                                                                                                                  |
|   491 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   492 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   493 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   494 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   495 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   574 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   575 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   576 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   577 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   578 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   579 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   606 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   607 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   608 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   609 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   610 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   611 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   664 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   665 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   666 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   667 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   668 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   669 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   670 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   671 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   672 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   673 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   674 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   675 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   806 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   807 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   808 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   809 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   810 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   811 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
```

Defense #6 – Monitor the queries

pt-query-digest

```
# 61.2s user time, 300ms system time, 30.57M rss, 2.40G vsz
# Current date: Sun Apr  5 20:42:51 2015
# Hostname: ganderson-mbpr.local
# Files: /Users/ganderson/sandboxes/msb_5_6_17/data/ganderson-mbpr-slow.log
# Overall: 381.22k total, 10 unique, 3.23k QPS, 9.89x concurrency _______
# Time range: 2015-04-05 20:39:20 to 20:41:18
# Attribute          total     min     max     avg     95%  stddev  median
# ===========        ======= ======= ======= ======= ======= ======= =======
# Exec time          1167s     1us   134ms     3ms    16ms     6ms   152us
# Lock time            23s       0     7ms    60us   138us   200us    28us
# Rows sent         23.04M       0     698   63.38  463.90  150.70       0
# Rows examine      26.50M       0     698   72.88  463.90  147.41   19.46
# Query size        50.42M      13     187  138.68  183.58   64.50  158.58

# Profile
# Rank Query ID           Response time  Calls  R/Call V/M   Item
# ==== ================== ============= ====== ====== ==== ============
#    1 0x4C25358328812E85 608.3529 52.1%  65457 0.0093  0.01 SELECT t?
#    2 0xCAEC22E79B0EFD3B 530.2798 45.4%  91606 0.0058  0.01 INSERT t?
#    3 0xCD051467F1BAE41A  25.6026  2.2% 189200 0.0001  0.00 SELECT rental
#    4 0x0C922D63DA1D917B   2.0297  0.2%    132 0.0154  0.00 CREATE TABLE t? `t1`
#    5 0x5722277F9D796D79   0.3962  0.0%    132 0.0030  0.00 DROP
#    6 0xAA353644DE4C4CB4   0.0691  0.0%  34424 0.0000  0.00 ADMIN QUIT
#    7 0xD92238E4C79EFE97   0.0263  0.0%    132 0.0002  0.00 CREATE
#    8 0x44AE35A182869033   0.0040  0.0%    132 0.0000  0.00 ADMIN INIT DB
#    9 0xE3A3649C5FAC418D   0.0001  0.0%      1 0.0001  0.00 SELECT
#   10 0x389FF9DA2DF3DF62   0.0001  0.0%      1 0.0001  0.00 SELECT

# Query 1: 1.72k QPS, 16.01x concurrency, ID 0x4C25358328812E85 at byte 105028989
# Scores: V/M = 0.01
# Time range: 2015-04-05 20:40:38 to 20:41:16
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ===========  ===  ======= ======= ======= ======= ======= ======= =======
# Count         17   65457
# Exec time     52    608s   175us   134ms     9ms    27ms     9ms     6ms
# Lock time     18      4s       0     4ms    66us   287us   156us    23us
# Rows sent     99  23.00M     100     698  368.47  621.67  158.55  346.17
# Rows examine  86  23.00M     100     698  368.47  621.67  158.55  346.17
# Query size     3   2.00M      32      32      32      32       0      32
```

mysql-sys

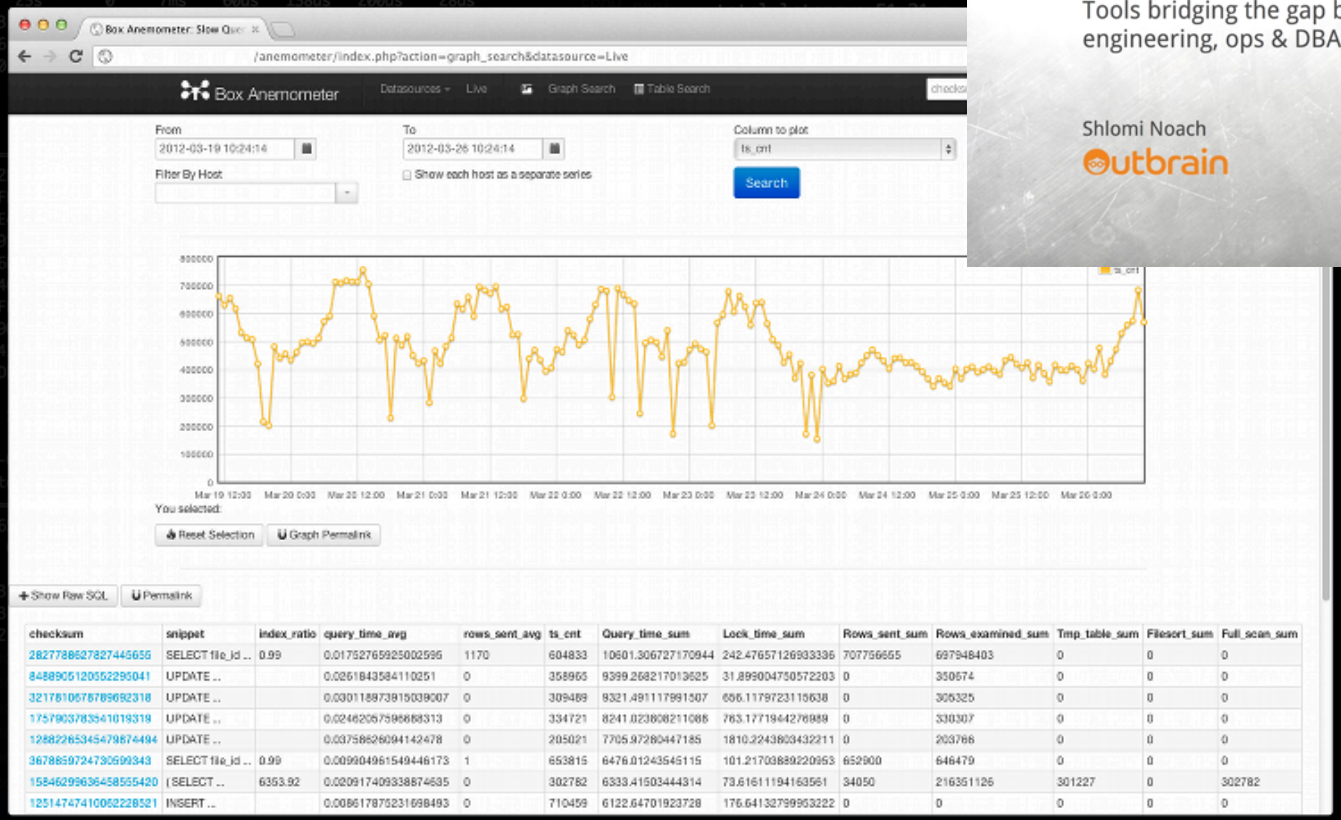
```
mysql [localhost] {root} (sys) > select * from statement_analysis LIMIT 1\G
*************************** 1. row ***************************
             query: SELECT `intcol1` , `charcol1` FROM `t1`
                db: mysqlslap
         full_scan: *
        exec_count: 361628
         err_count: 52
        warn_count: 0
     total_latency: 51.21 m
       max_latency: 239.39 ms
       avg_latency: 8.50 ms
      lock_latency: 1.52 m
         rows_sent: 134632249
     rows_sent_avg: 372
     rows_examined: 134641306
 rows_examined_avg: 372
     rows_affected: 0
 rows_affected_avg: 0
       tmp_tables: 0
  tmp_disk_tables: 0
      rows_sorted: 0
 sort_merge_passes: 0
            digest: 239eeeb1e52b81b52e3020c23988c8cc
        first_seen: 2015-04-05 16:10:47
         last_seen: 2015-04-05 20:41:17
1 row in set (0.01 sec)
```

Defense #6 – Monitor the queries

pt-query-digest

mysql-sys

Anemometer

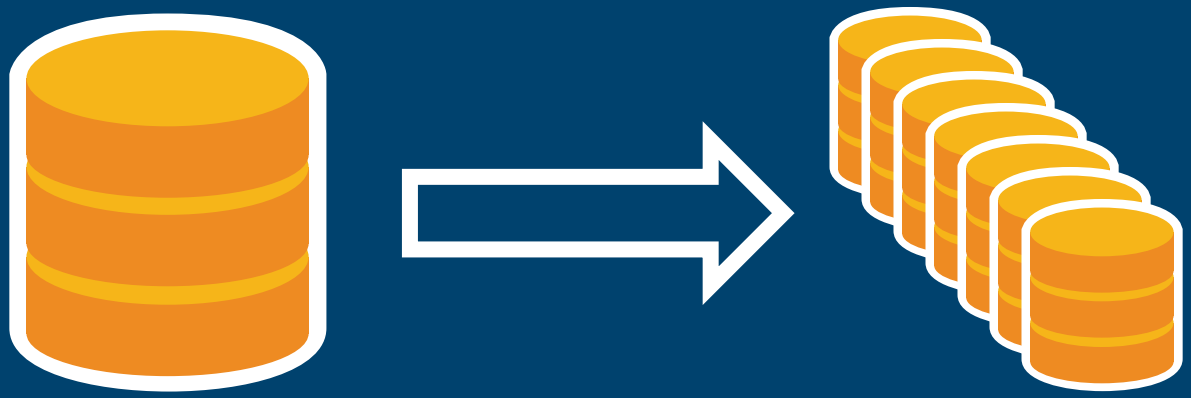


Queries

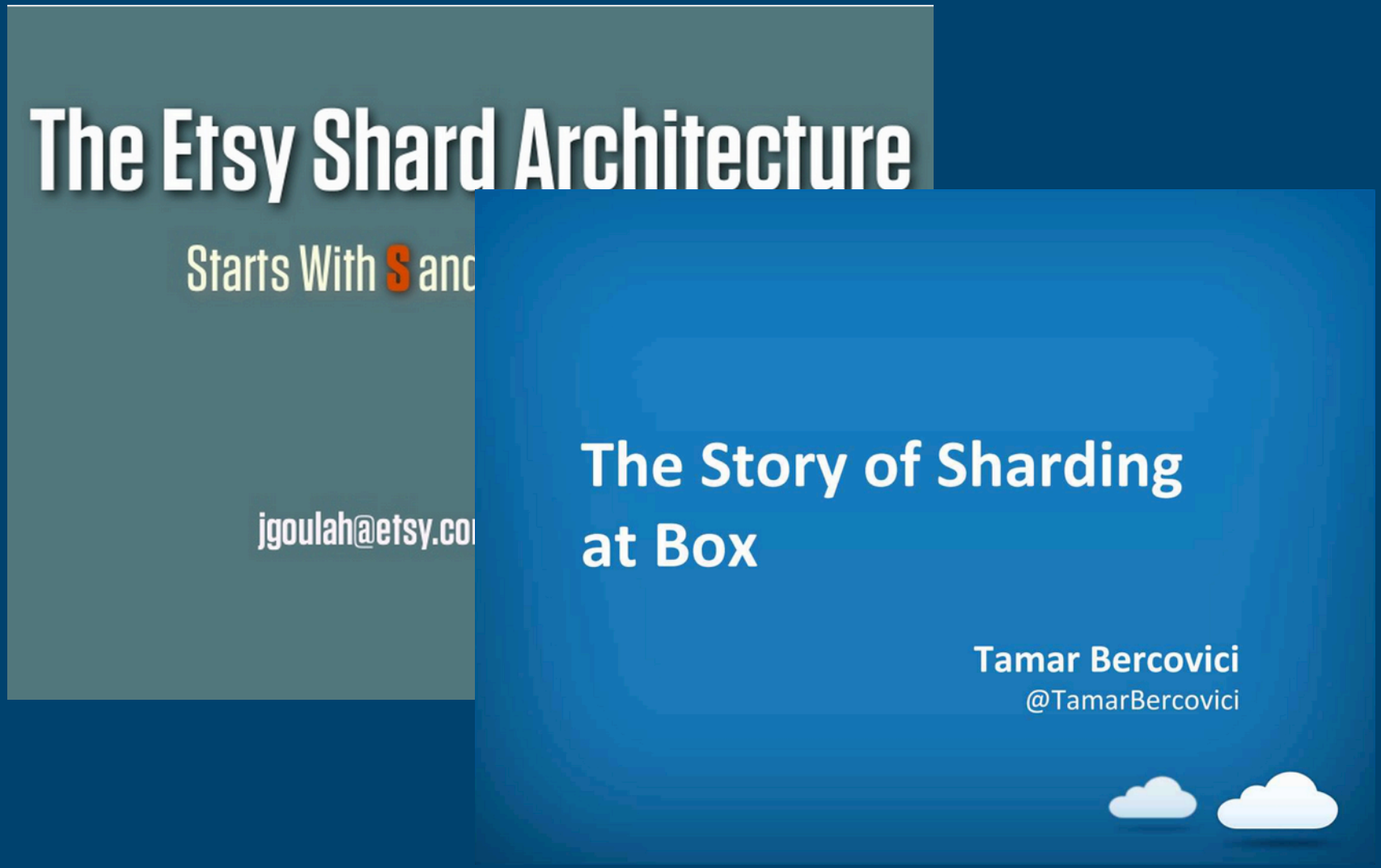
Queries

Time to
break up
the data

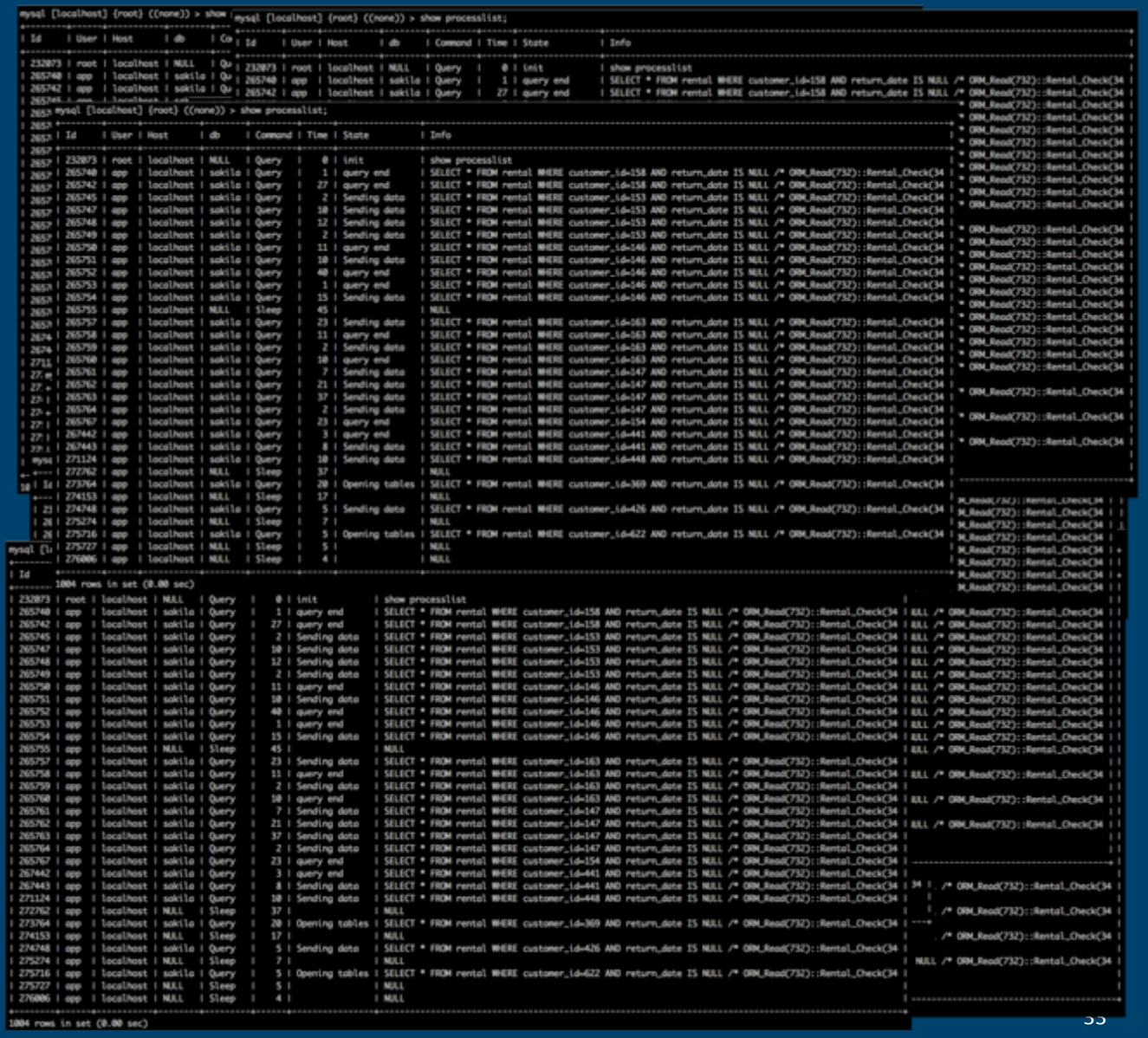
Defense #7 – Sharding



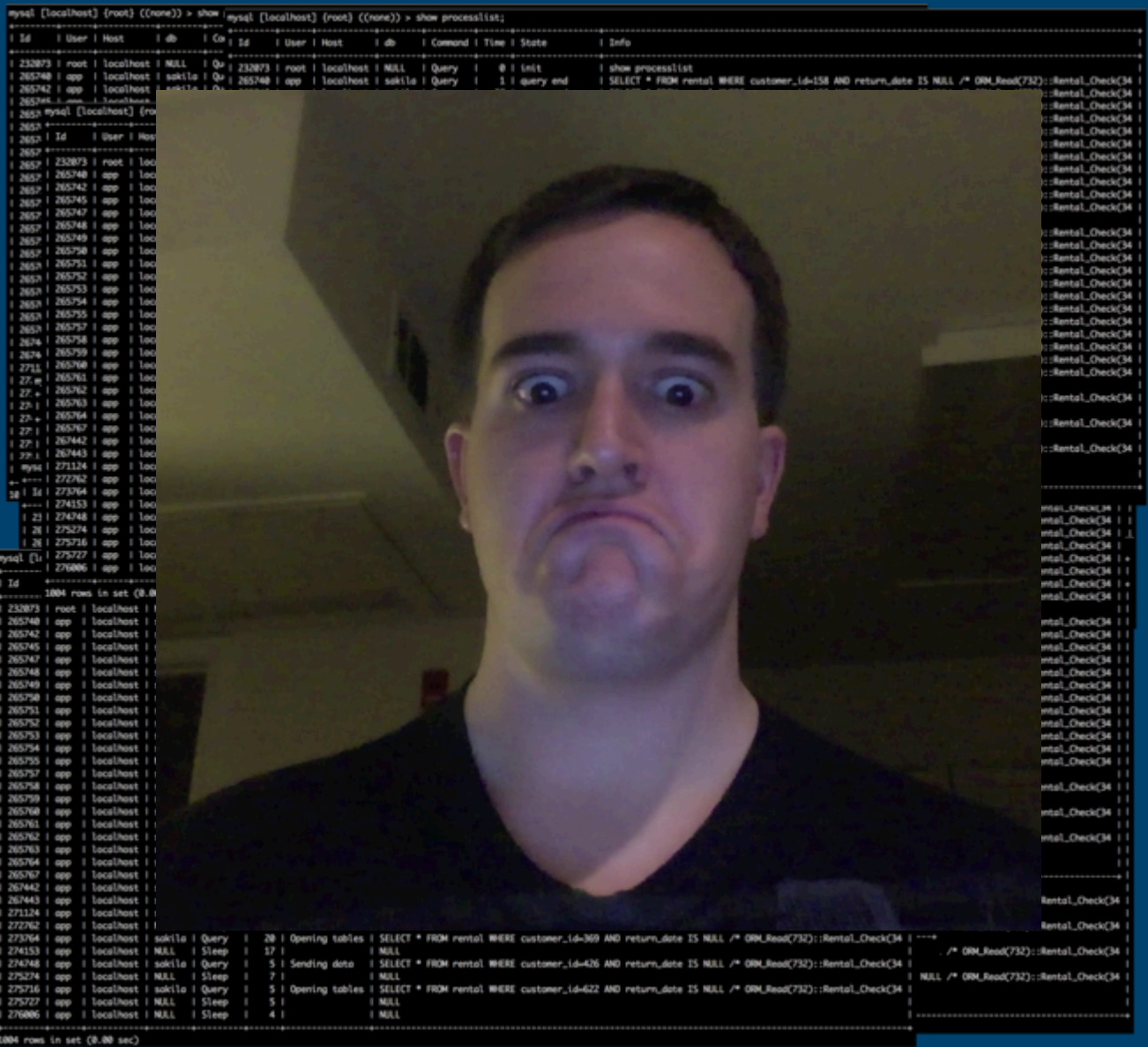
Defense #7 – Sharding



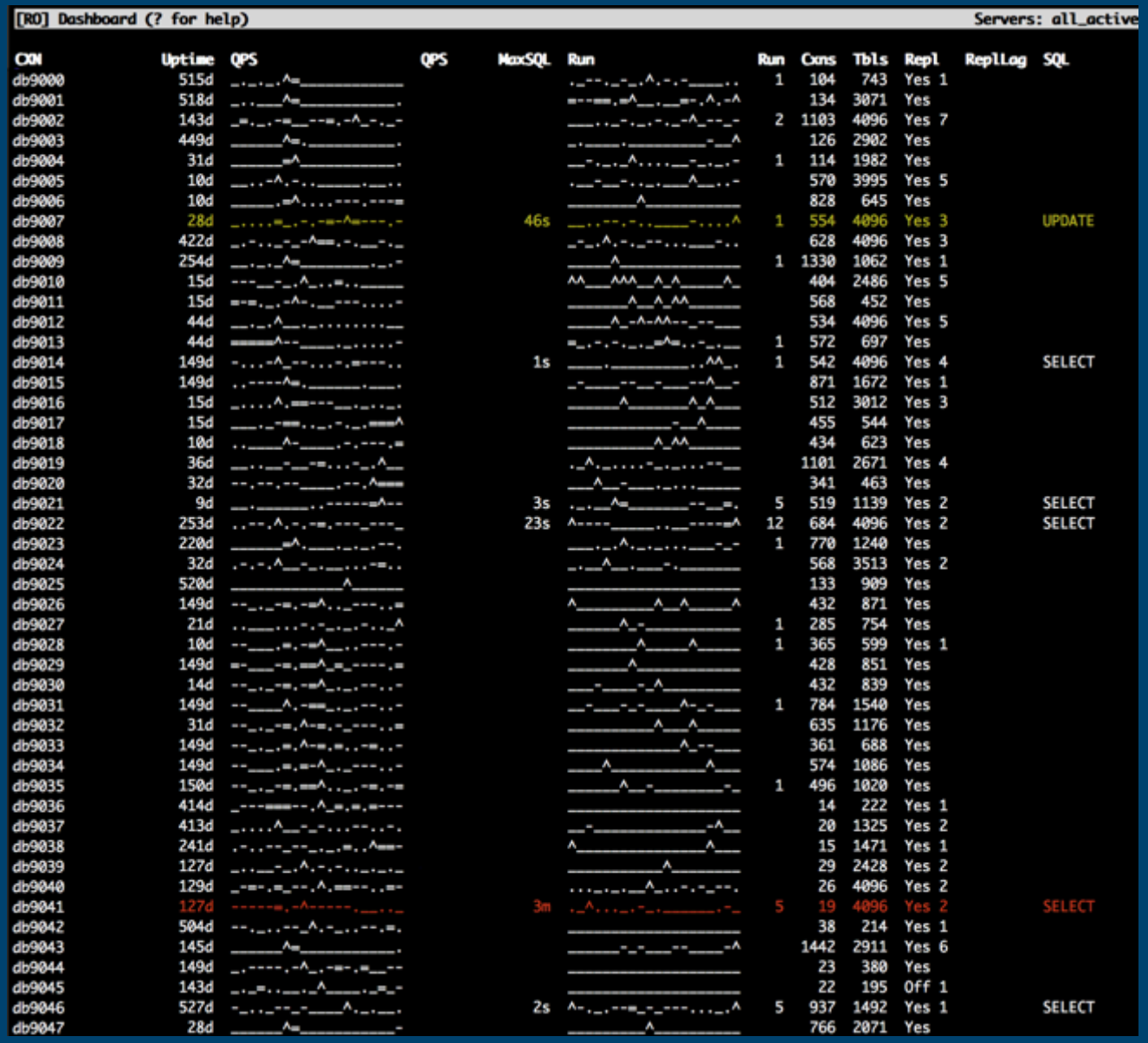
Defense #8 – innotop



Defense #8 – innotop



Defense #8 – innotop



Defense #8 – innotop

```
[general]
mode=A
[/general]

[connections]
mysql56Test=user= dsn=DBI:mysql:;host=127.0.0.1;port=3306;mysql_read_default_group=client
anotherDb=user= dsn=DBI:mysql:;host=127.0.0.1;port=3306;mysql_read_default_group=client
[/connections]

[server_groups]
all=mysql56Test anotherDb
[/server_groups]

[active_server_groups]
A=all
[/active_server_groups]

[active_columns]
health_dashboard=cxn uptime spark_qps qps max_query_time spark_run run connections open slave_running
time_behind_master longest_sql
[/active_columns]

[visible_tables]
A=health_dashboard
[/visible_tables]

[colors]
health_dashboard=col='slave_running' op='eq' arg='No' color='black on_red'
health_dashboard=col='time_behind_master' op='>' arg='30' color='cyan'
health_dashboard=col='max_query_time' op='>' arg='60' color='red'
health_dashboard=col='max_query_time' op='>' arg='30' color='yellow'
[/colors]
```

Assign connections to server groups

Set the default active group

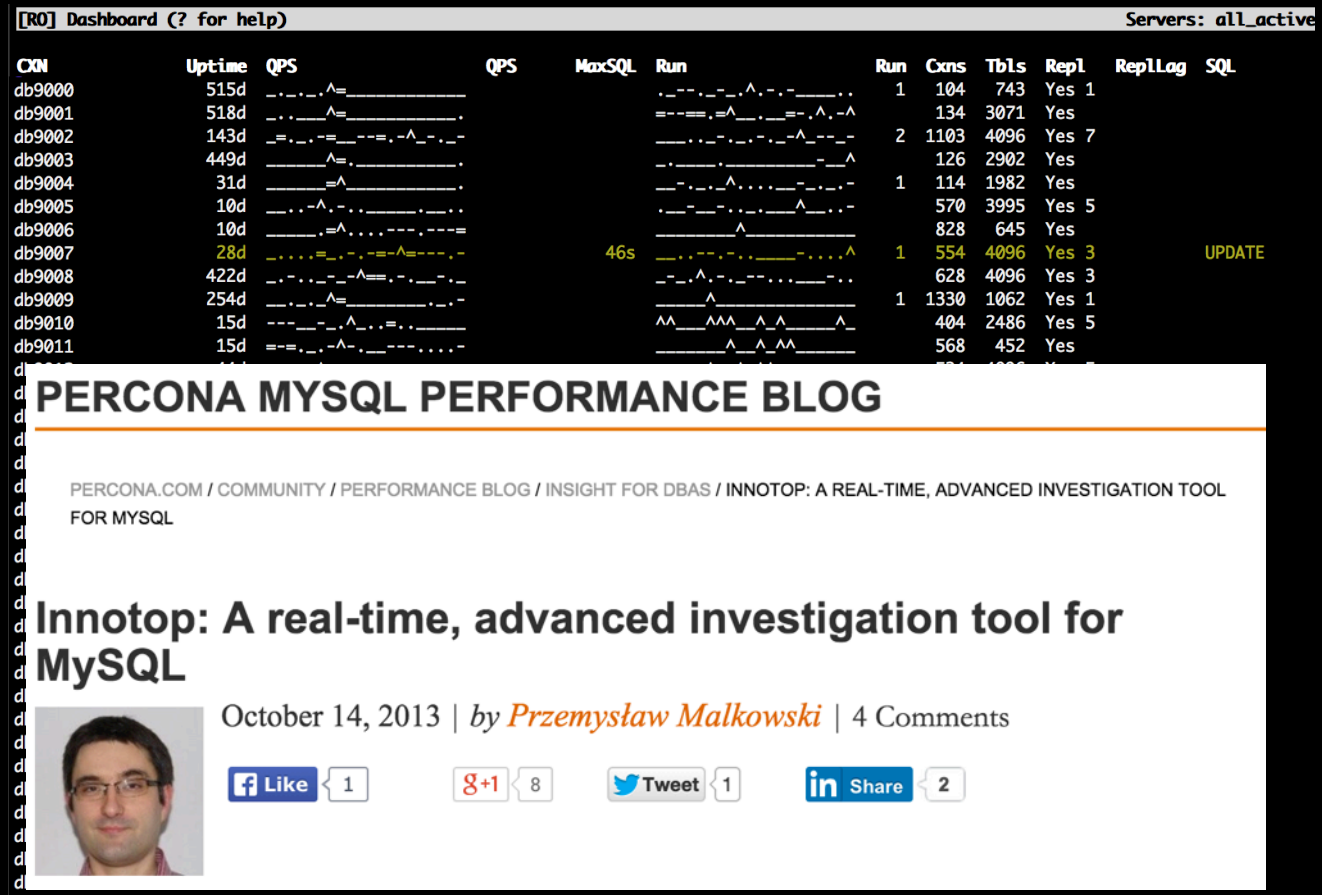
Define views

Define connections (with custom names)

Set the default view

Setup color highlighting

Defense #8 – innotop




Now for the
fun stuff

Bad queries happen

```
mysql [localhost] {root} ((none)) > show processlist;
+-------+------+-----------+--------+---------+------+-------------+-----------------------------------------------------------------------------------------------------------------------+
| Id    | User | Host      | db     | Command | Time | State       | Info                                                                                                                  |
+-------+------+-----------+--------+---------+------+-------------+-----------------------------------------------------------------------------------------------------------------------+
|     1 | root | localhost | NULL   | Query   |    0 | init        | show processlist                                                                                                      |
|   490 | app  | localhost | NULL   | Sleep   |  162 |             | NULL                                                                                                                  |
|   491 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   492 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   493 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   494 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   495 | app  | localhost | sakila | Query   |  162 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   574 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   575 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   576 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   577 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   578 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   579 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   606 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   607 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   608 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   609 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   610 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   611 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   664 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   665 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   666 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   667 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   668 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   669 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   670 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   671 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   672 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   673 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   674 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   675 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   806 | app  | localhost | NULL   | Sleep   |  161 |             | NULL                                                                                                                  |
|   807 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   808 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   809 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   810 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
|   811 | app  | localhost | sakila | Query   |  161 | Sending data | SELECT * FROM rental r JOIN payment p JOIN rental t1 JOIN rental t2 JOIN rental t3 WHERE r.customer_ |
```

Bad queries happen

I woke up at 3am?
FOR THIS!?

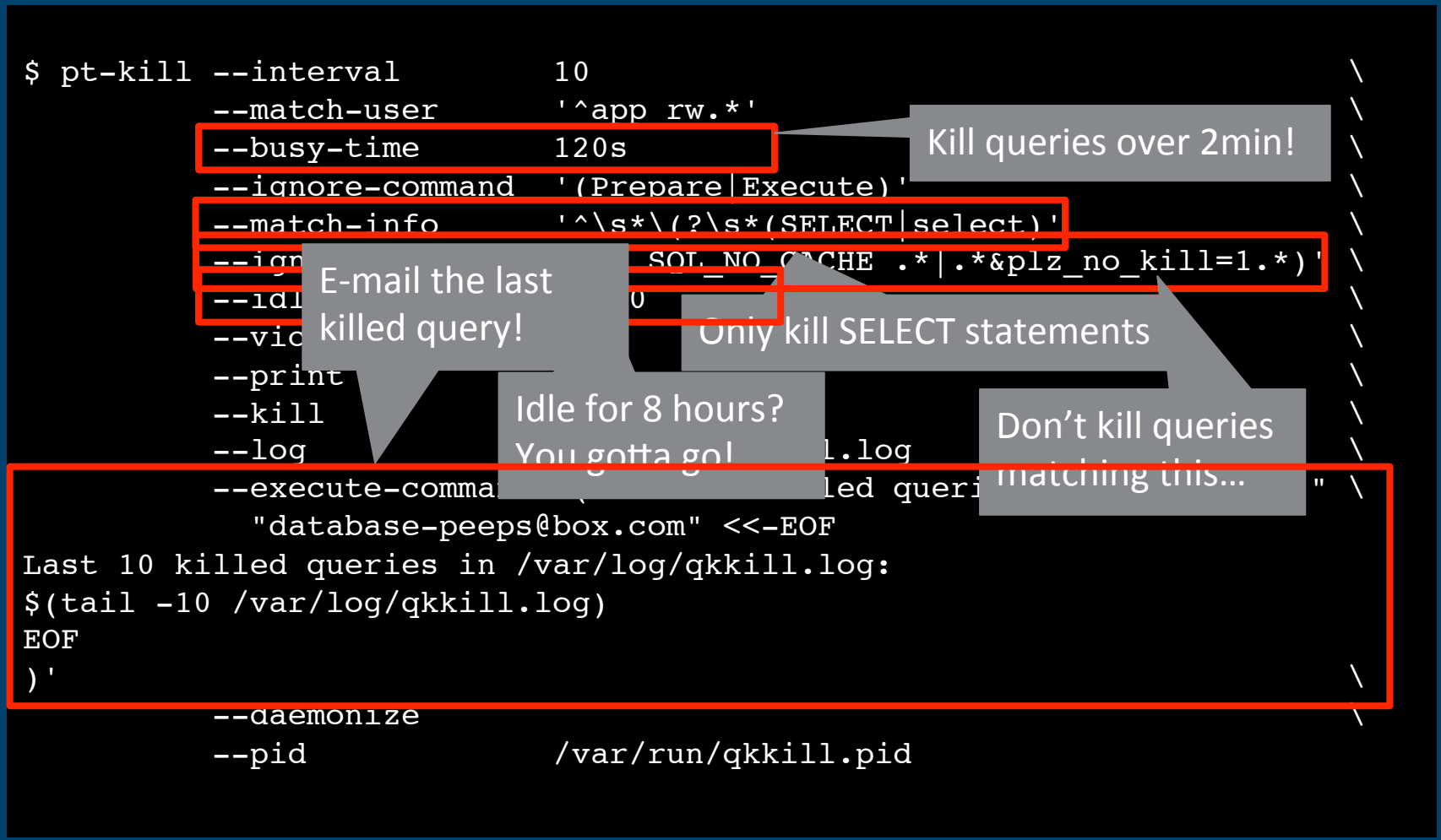
Who added that crappy query…?

This should take care of itself… >:[

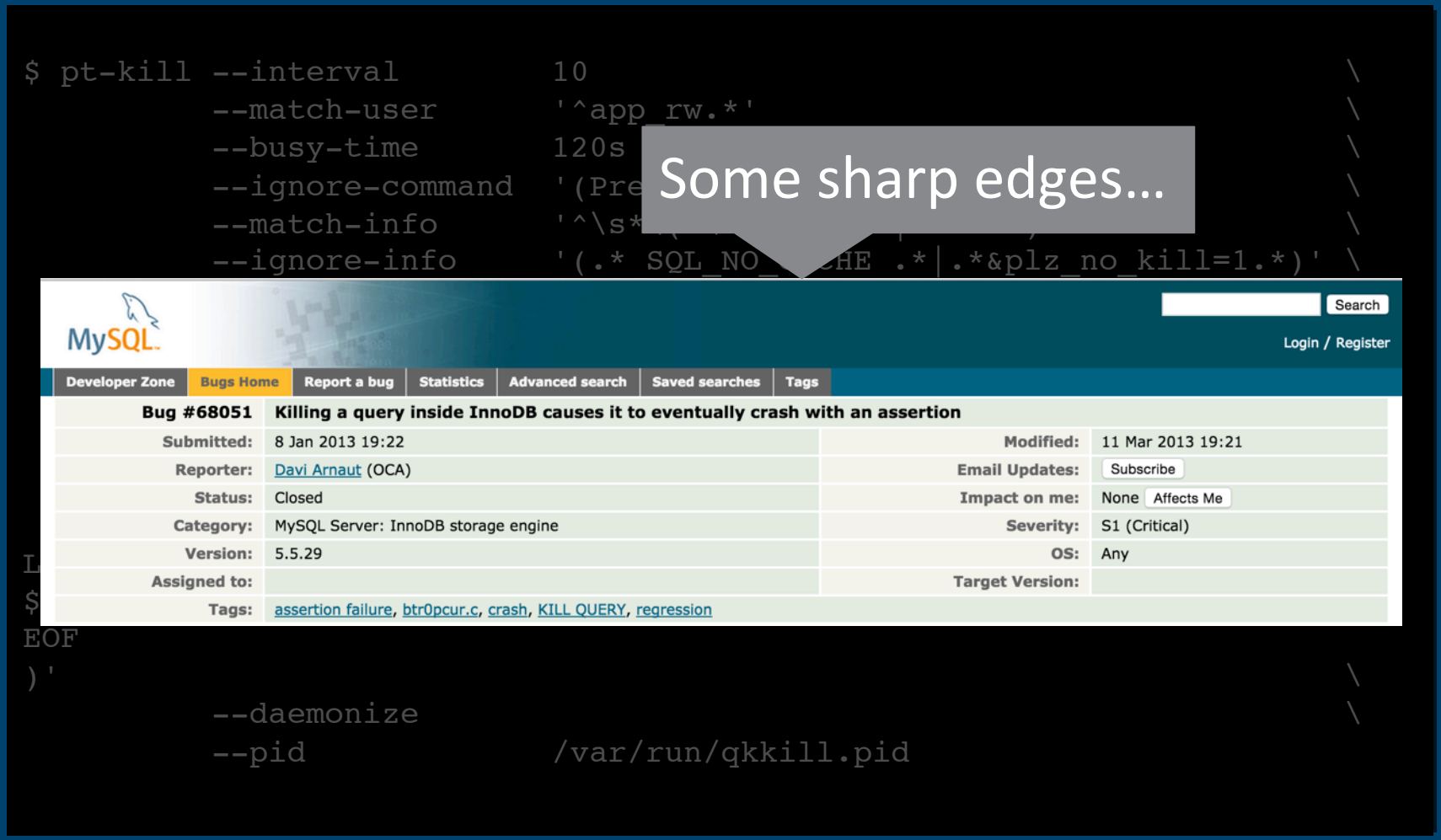
Defense #9 – Query killer

```
$ pt-kill --interval        10                                          \
          --match-user      '^app_rw.*'                                 \
          --busy-time       120s                                        \
          --ignore-command  '(Prepare|Execute)'                         \
          --match-info      '^\s*\(?\s*(SELECT|select)'                 \
          --ignore-info     '(.* SQL_NO_CACHE .*|.*&plz_no_kill=1.*)' \
          --idle-time       28800                                       \
          --victims         all                                         \
          --print                                                       \
          --kill                                                        \
          --log             /var/log/qkkill.log                         \
          --execute-command '(mail –s "Killed queries on ${HOSTNAME}" \
            "database-peeps@box.com" <<-EOF
Last 10 killed queries in /var/log/qkkill.log:
$(tail -10 /var/log/qkkill.log)
EOF
)'                                                                      \
          --daemonize                                                   \
          --pid             /var/run/qkkill.pid
```

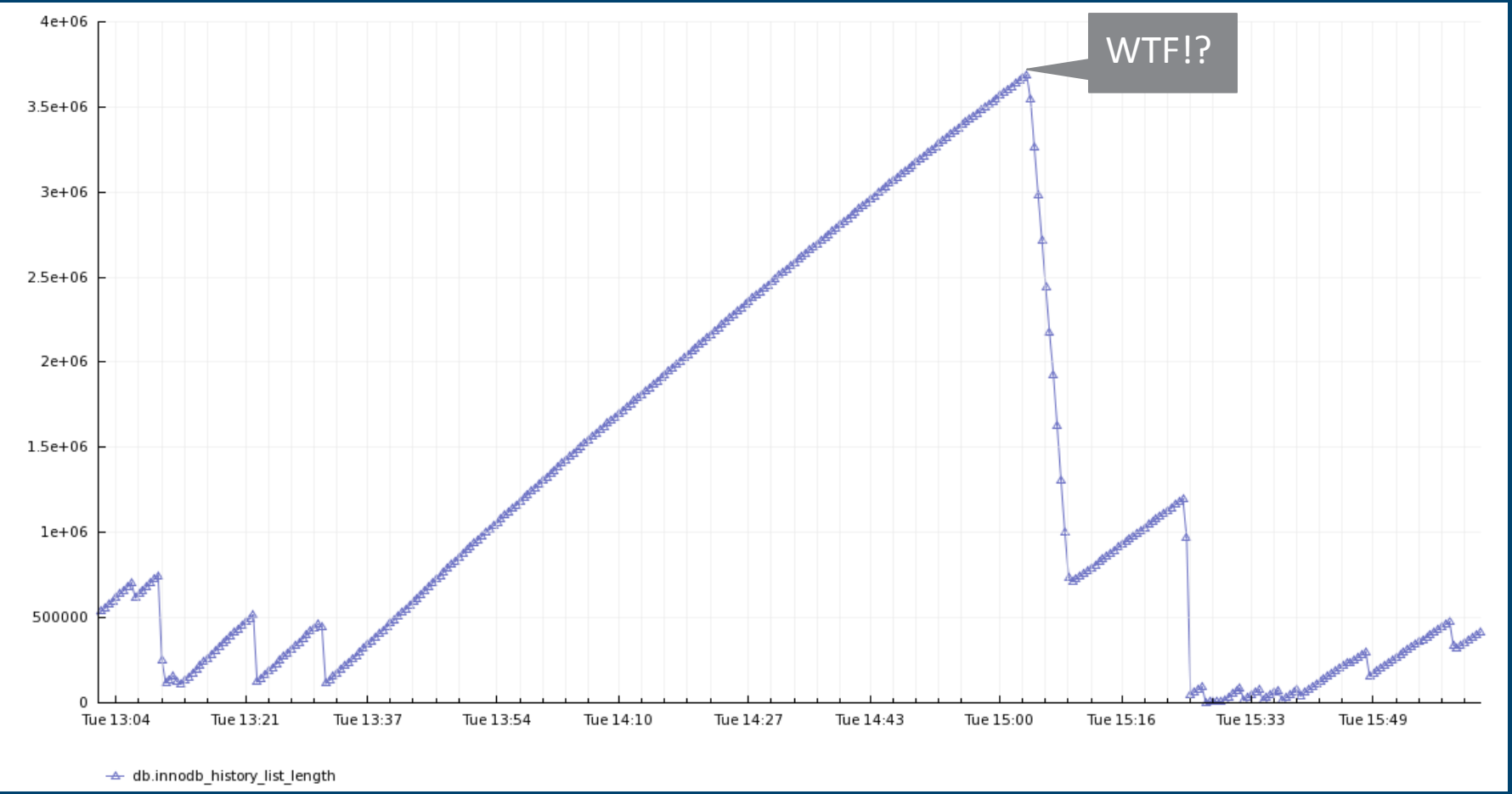
Defense #9 – Query killer



Defense #9 – Query killer



Long transactions happen…



Defense #10 – Transaction Monitor (and killer)

```
my $query = "SELECT *, unix_timestamp()-unix_timestamp(trx_started) as len FROM INFORMATION_SCHEMA.INNODB_TRX
ORDER BY trx_started LIMIT 1";
print $query ."\n";
my $sth = $dbh->prepare($query);
$sth->execute() or die $dbh->errstr;
my $data1 = get_query_id($sth);
my $query_id = $data1->{trx_mysql_thread_id};
```

```
$query = "SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST WHERE id=$query_id";
$sth = $dbh->prepare($query);
$sth->execute() or die $dbh->errstr;

my $data2 = get_source($sth);
my $source = $data2->{HOST};
my ($host,$port) = split(/:/, $source);
```

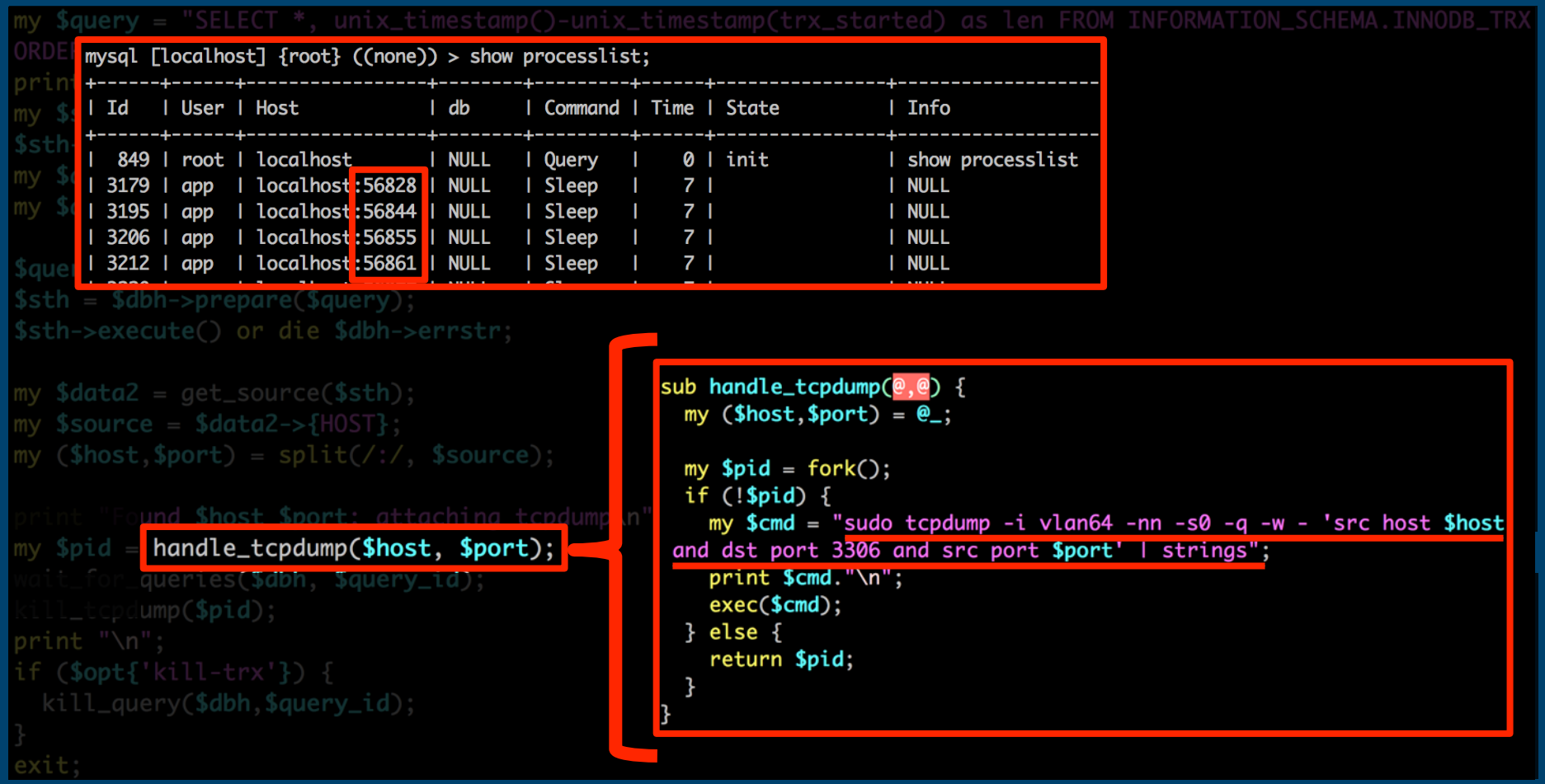
Find the connection for the longest transaction

Get the source host for that connection

```
print "Found $host $port; attaching tcpdump\n";
my $pid = handle_tcpdump($host, $port);
wait_for_queries($dbh, $query_id);
kill_tcpdump($pid);
print "\n";
if ($opt{'kill-trx'}) {
  kill_query($dbh,$query_id);
}
exit;
```

Try to catch a query from the connection before killing it

Defense #10 – Transaction Monitor (and killer)



Too many connections!

```
mysql [localhost] {root} ((none)) > show processlist;
+--------+------+-----------+--------+---------+------+---------------+-----------------------------------------------------------------------------------------------------------------------+
| Id     | User | Host      | db     | Command | Time | State         | Info                                                                                                                  |
+--------+------+-----------+--------+---------+------+---------------+-----------------------------------------------------------------------------------------------------------------------+
| 232073 | root | localhost | NULL   | Query   |    0 | init          | show processlist                                                                                                      |
| 265740 | app  | localhost | sakila | Query   |    1 | query end     | SELECT * FROM rental WHERE customer_id=158 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265742 | app  | localhost | sakila | Query   |   27 | query end     | SELECT * FROM rental WHERE customer_id=158 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265745 | app  | localhost | sakila | Query   |    2 | Sending data  | SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265747 | app  | localhost | sakila | Query   |   10 | Sending data  | SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265748 | app  | localhost | sakila | Query   |   12 | Sending data  | SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265749 | app  | localhost | sakila | Query   |    2 | Sending data  | SELECT * FROM rental WHERE customer_id=153 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265750 | app  | localhost | sakila | Query   |   11 | query end     | SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265751 | app  | localhost | sakila | Query   |   10 | Sending data  | SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265752 | app  | localhost | sakila | Query   |   40 | query end     | SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265753 | app  | localhost | sakila | Query   |    1 | query end     | SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265754 | app  | localhost | sakila | Query   |   15 | Sending data  | SELECT * FROM rental WHERE customer_id=146 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265755 | app  | localhost | NULL   | Sleep   |   45 |               | NULL                                                                                                                  |
| 265757 | app  | localhost | sakila | Query   |   23 | Sending data  | SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265758 | app  | localhost | sakila | Query   |   11 | query end     | SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265759 | app  | localhost | sakila | Query   |    2 | Sending data  | SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265760 | app  | localhost | sakila | Query   |   10 | query end     | SELECT * FROM rental WHERE customer_id=163 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265761 | app  | localhost | sakila | Query   |    7 | Sending data  | SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265762 | app  | localhost | sakila | Query   |   21 | Sending data  | SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265763 | app  | localhost | sakila | Query   |   37 | Sending data  | SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265764 | app  | localhost | sakila | Query   |    2 | Sending data  | SELECT * FROM rental WHERE customer_id=147 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 265767 | app  | localhost | sakila | Query   |   23 | query end     | SELECT * FROM rental WHERE customer_id=154 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 267442 | app  | localhost | sakila | Query   |    3 | query end     | SELECT * FROM rental WHERE customer_id=441 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 267443 | app  | localhost | sakila | Query   |    8 | Sending data  | SELECT * FROM rental WHERE customer_id=441 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 271124 | app  | localhost | sakila | Query   |   10 | Sending data  | SELECT * FROM rental WHERE customer_id=448 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 272762 | app  | localhost | NULL   | Sleep   |   37 |               | NULL                                                                                                                  |
| 273764 | app  | localhost | sakila | Query   |   20 | Opening tables | SELECT * FROM rental WHERE customer_id=369 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 274153 | app  | localhost | NULL   | Sleep   |   17 |               | NULL                                                                                                                  |
| 274748 | app  | localhost | sakila | Query   |    5 | Sending data  | SELECT * FROM rental WHERE customer_id=426 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 275274 | app  | localhost | NULL   | Sleep   |    7 |               | NULL                                                                                                                  |
| 275716 | app  | localhost | sakila | Query   |    5 | Opening tables | SELECT * FROM rental WHERE customer_id=622 AND return_date IS NULL /* ORM_Read(732)::Rental_Check(34 |
| 275727 | app  | localhost | NULL   | Sleep   |    5 |               | NULL                                                                                                                  |
| 276006 | app  | localhost | NULL   | Sleep   |    4 |               | NULL                                                                                                                  |
+--------+------+-----------+--------+---------+------+---------------+-----------------------------------------------------------------------------------------------------------------------+
1004 rows in set (0.00 sec)
```

Too many connections!

```
mysql [localhost] {root} ((none)) > show global status like 'threads_connected';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Threads_connected | 5000  |
+-------------------+-------+
1 row in set (0.00 sec)

mysql [localhost] {root} ((none)) > select @@max_connections, @@max_user_connections;
+-------------------+------------------------+
| @@max_connections | @@max_user_connections |
+-------------------+------------------------+
|              5000 |                   5000 |
+-------------------+------------------------+
1 row in set (0.00 sec)
```

Defense #11 – Dynamic Query Killer Thresholds

```
$ pt-kill --interval        10                                          \
          --match-user      '^app_rw.*'                                 \
          --busy-time       120s                                        \
          --ignore-command  '(Prepare|Execute)'                         \
          --match-info      '^\s*\(?\s*(SELECT|select)'                 \
          --ignore-info     '(.* SQL_NO_CACHE .*|.*&plz_no_kill=1.*)' \
          --idle-time       28800                                       \
          --victims         all                                         \
          --print                                                       \
          --kill                                                        \
          --log             /var/log/qkkill.log                         \
          --execute-command '(mail –s "Killed queries on ${HOSTNAME}" \
            "database-peeps@box.com" <<-EOF
Last 10 killed queries in /var/log/qkkill.log:
$(tail -10 /var/log/qkkill.log)
EOF
)'                                                                      \
          --daemonize                                                   \
          --pid             /var/run/qkkill.pid                         \
          --dynamic-time    60                                          \
          --min-busy-time   10                                          \
          --min-idle-time   30
```

Defense #11 – Dynamic Query Killer Thresholds

JAN
31

Enhancing pt-kill to Better Protect your Servers

I believe in automation as much as possible, and I'm always working to make the day to day tasks of operations as smooth as possible. Also I try not to be afraid to take good tools and make them better.

Here in Database Ops at Box, we use pt-kill running as a service to constantly monitor our servers and help protect against long running queries. But our thresholds are pretty generous, and in some cases it's possible for unforeseen circumstances to cause enough queries to storm the database such that we can have problems before any of them hit the threshold for "busy time." Ditto for idle connections.

The response is that someone has to be available to manually run another copy of pt-kill with much lower thresholds to clear out these thundering herds. But what if we could let pt-kill handle both the "normal" mode and still protect us from herds?

That's what we've done by adding a couple new variables and small code changes. Here's an example:

```
pt-kill --busy-time=60 --dynamic-time 50
```

The extra option --dynamic-time causes pt-kill to monitor the ratio of (max_connections-threads_connected)/ max_connections. This gives us a percentage of connections available, and if that percentage drops **below** the argument to --dynamic-time, then pt-kill will modify the value of busy-time so it can kill queries more aggressively.

For example, if max_connections was 1000 and threads_connected was 500, then that would trigger the new behavior.

```
connections_free = 0.5
new_busy_time  = busy_time * connections_free
```

The new busy time is the percent of connections free times the original value. In this case it would reduce the value by half and our new busy time is 30.

Similarly, if there were 900/1000 threads connected then our busy-time would go all the way down to 6 seconds.

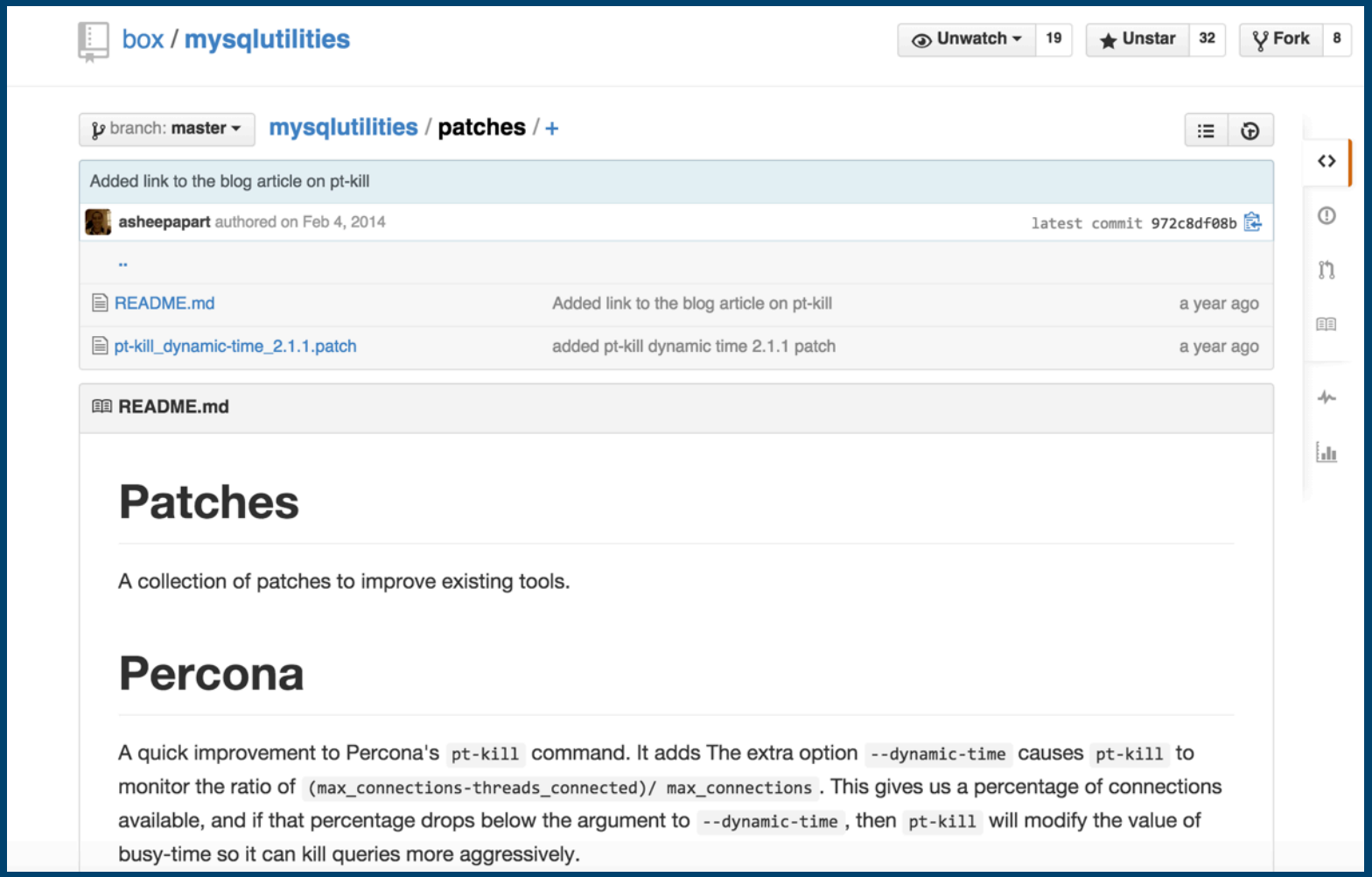
You may want to set a floor for your busy-time so it doesn't drop below some minimum value. Maybe we want to never go below 10 seconds. There's an option for that:

```
pt-kill --busy-time=60 --dynamic-time 50 --min-busy-time=10
```

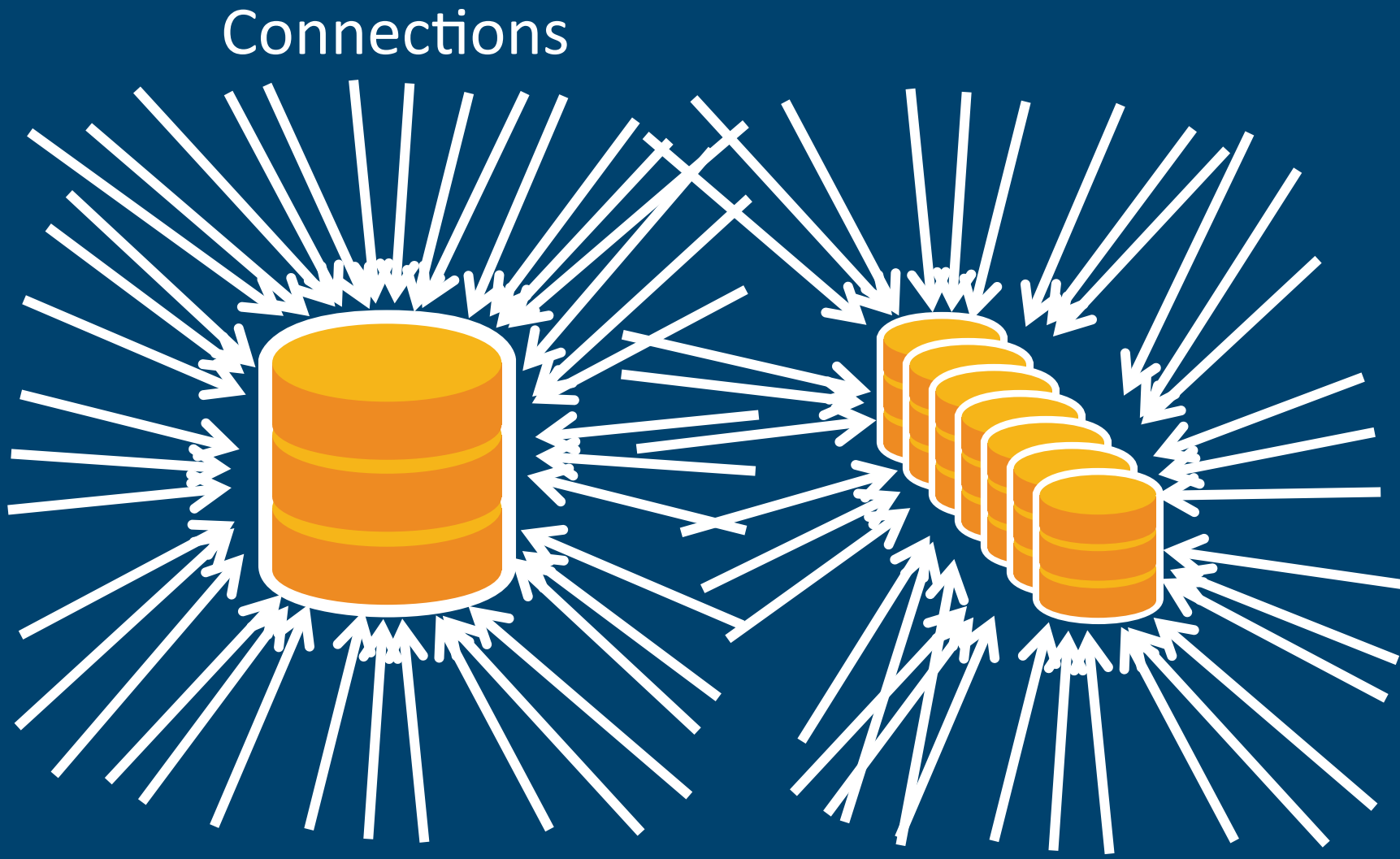
And there are similar options for idle-time. Thats it! I've provided a patch for percona toolkit 2.1.1. When I get a chance I'll try to update it for the current version (or someone feel free to do that and submit one back to me!)

Here's a link to the patch: https://github.com/box/mysqlutilities/blob/master/patches/pt-kill_dynamic-time_2.1.1.patch

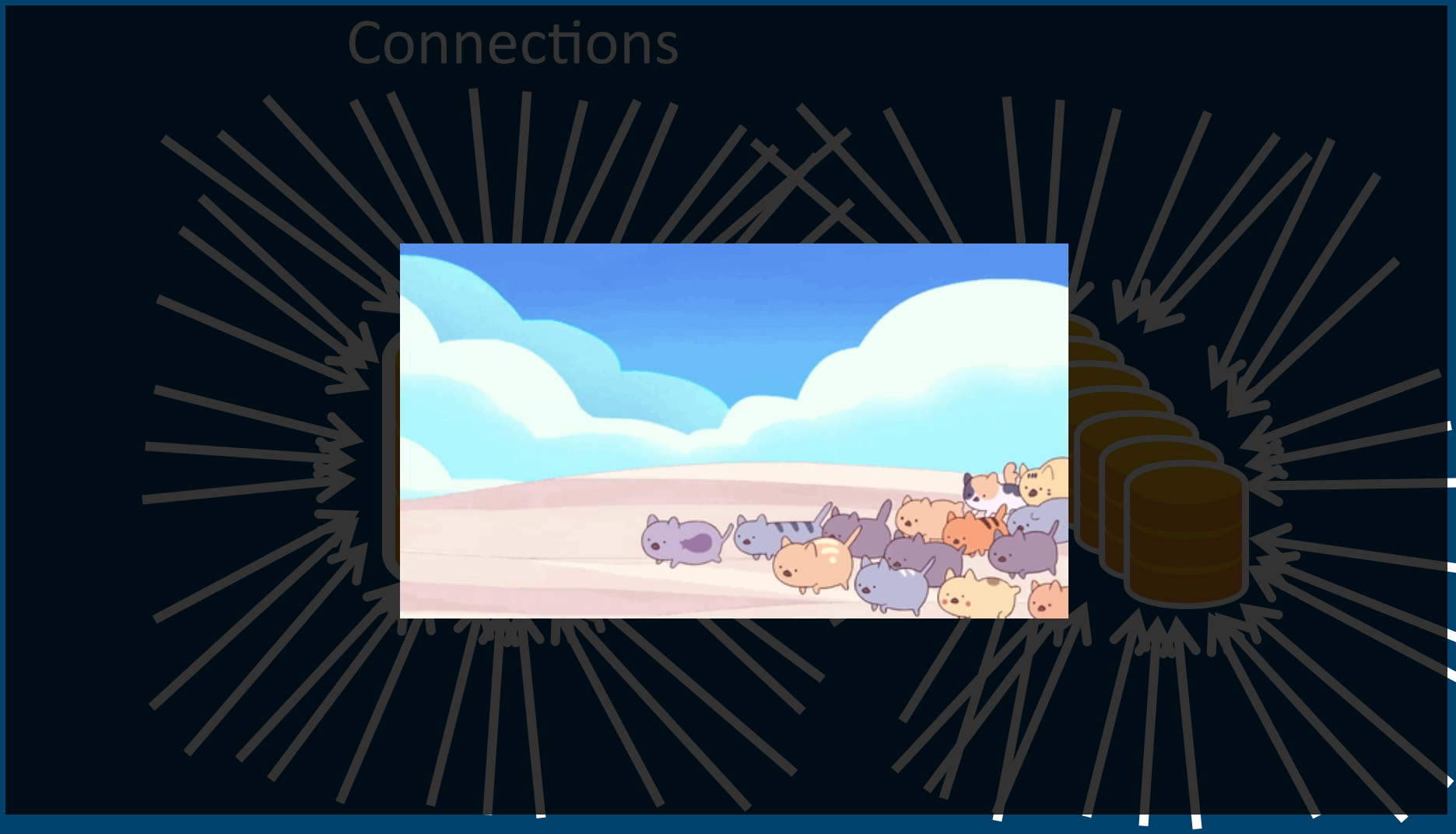
Defense #11 – Dynamic Query Killer Thresholds



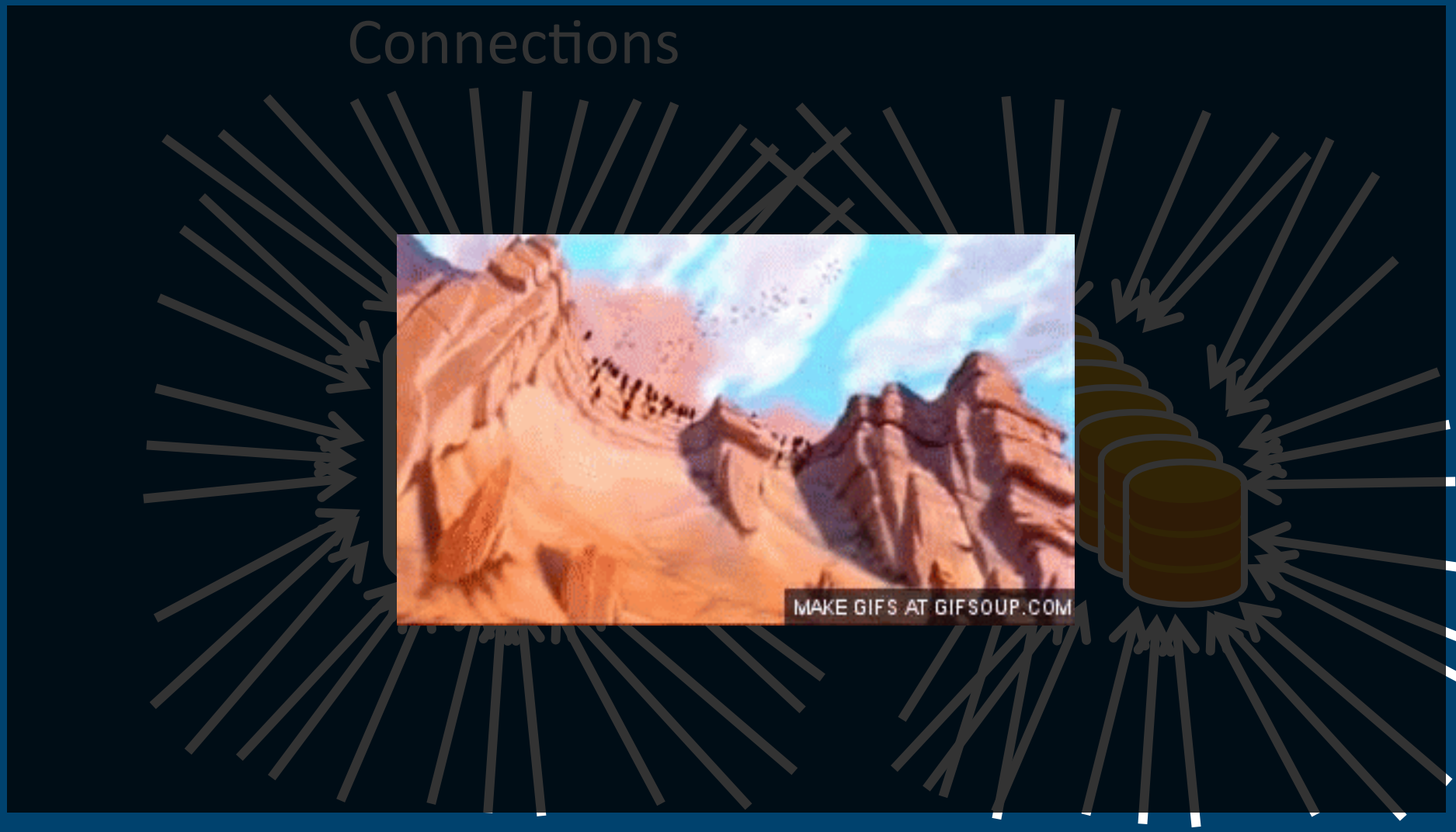
The thundering herd..



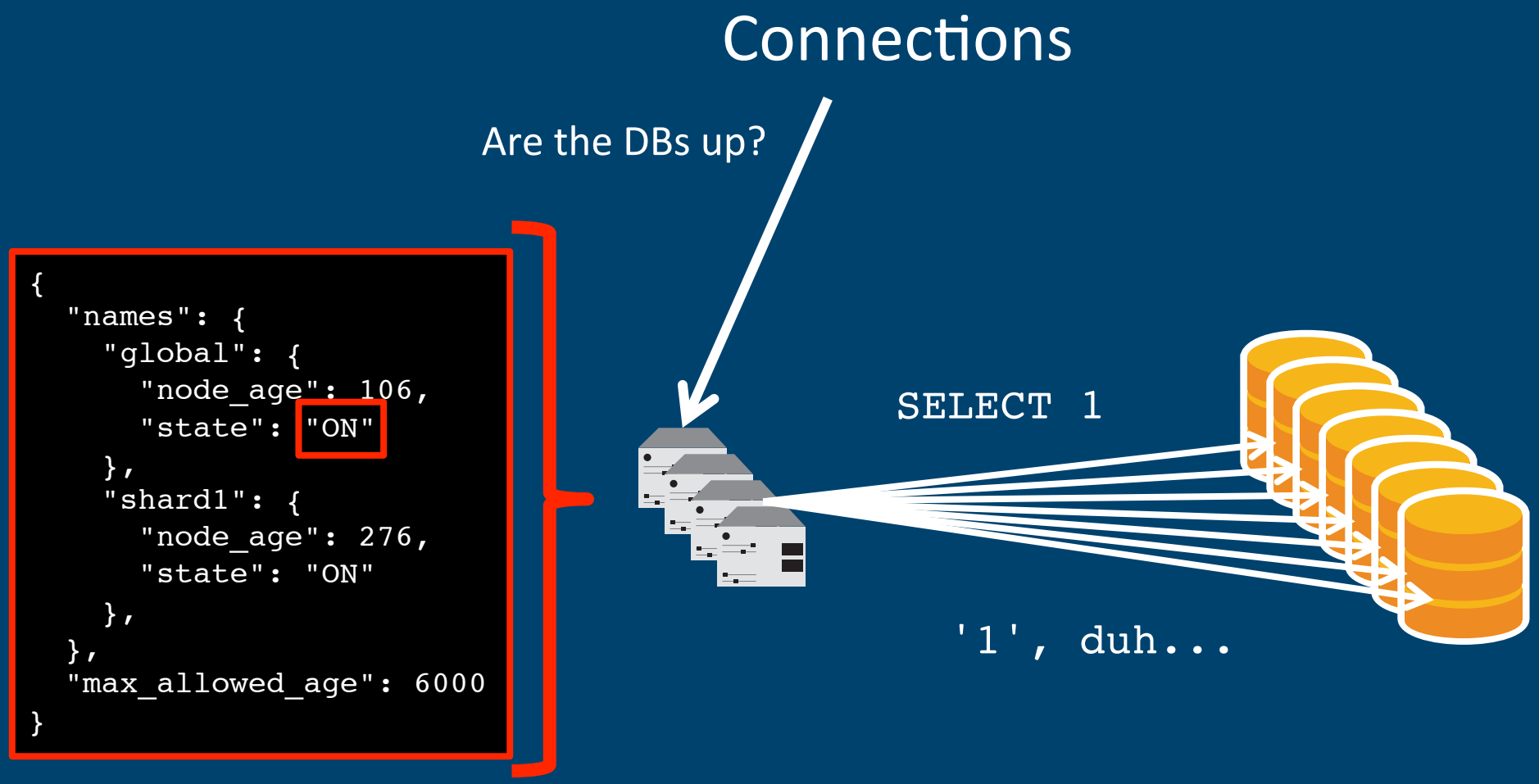
The thundering herd..



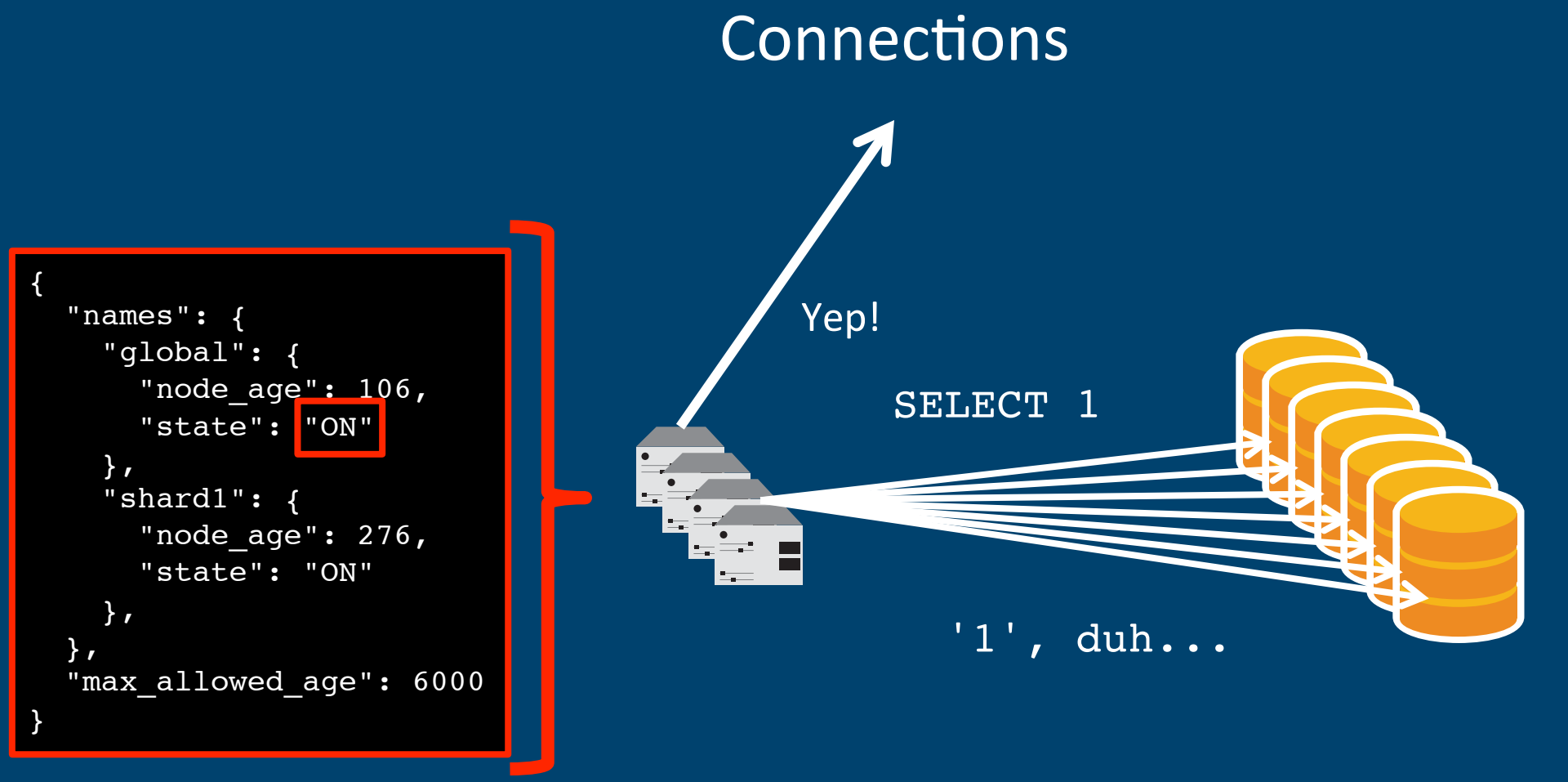
The thundering herd..



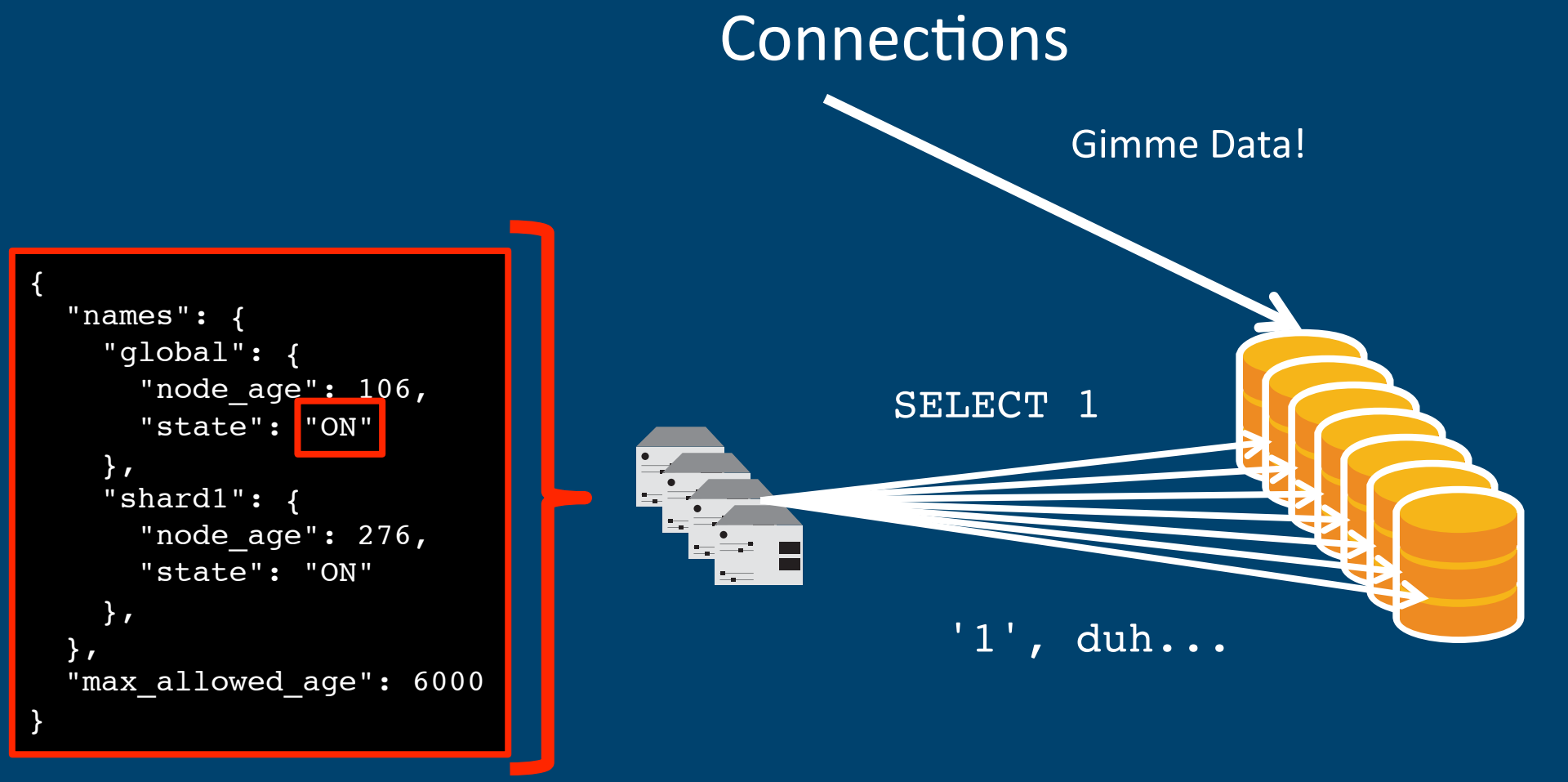
Defense #12 – Healthchecking



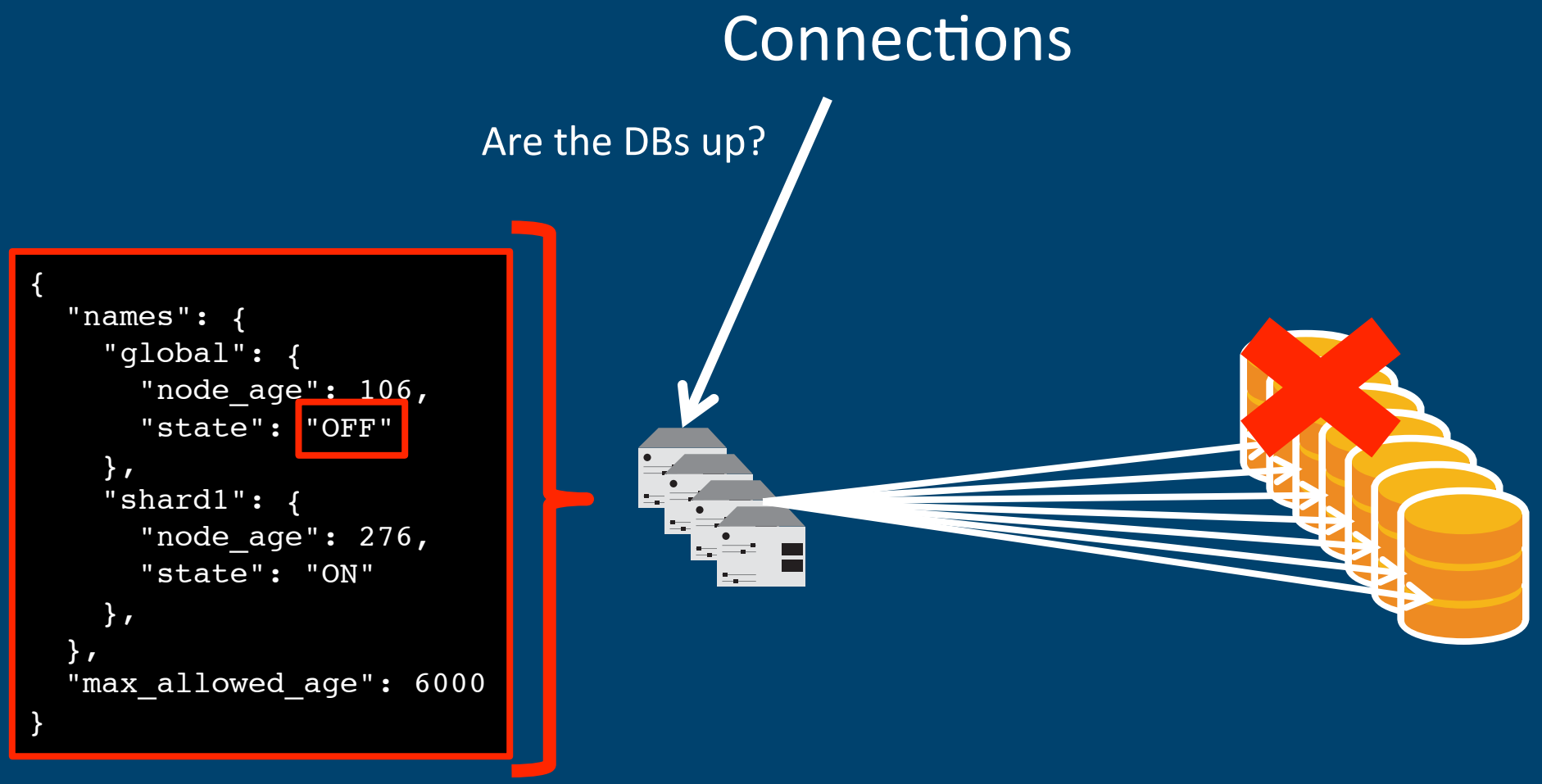
Defense #12 – Healthchecking



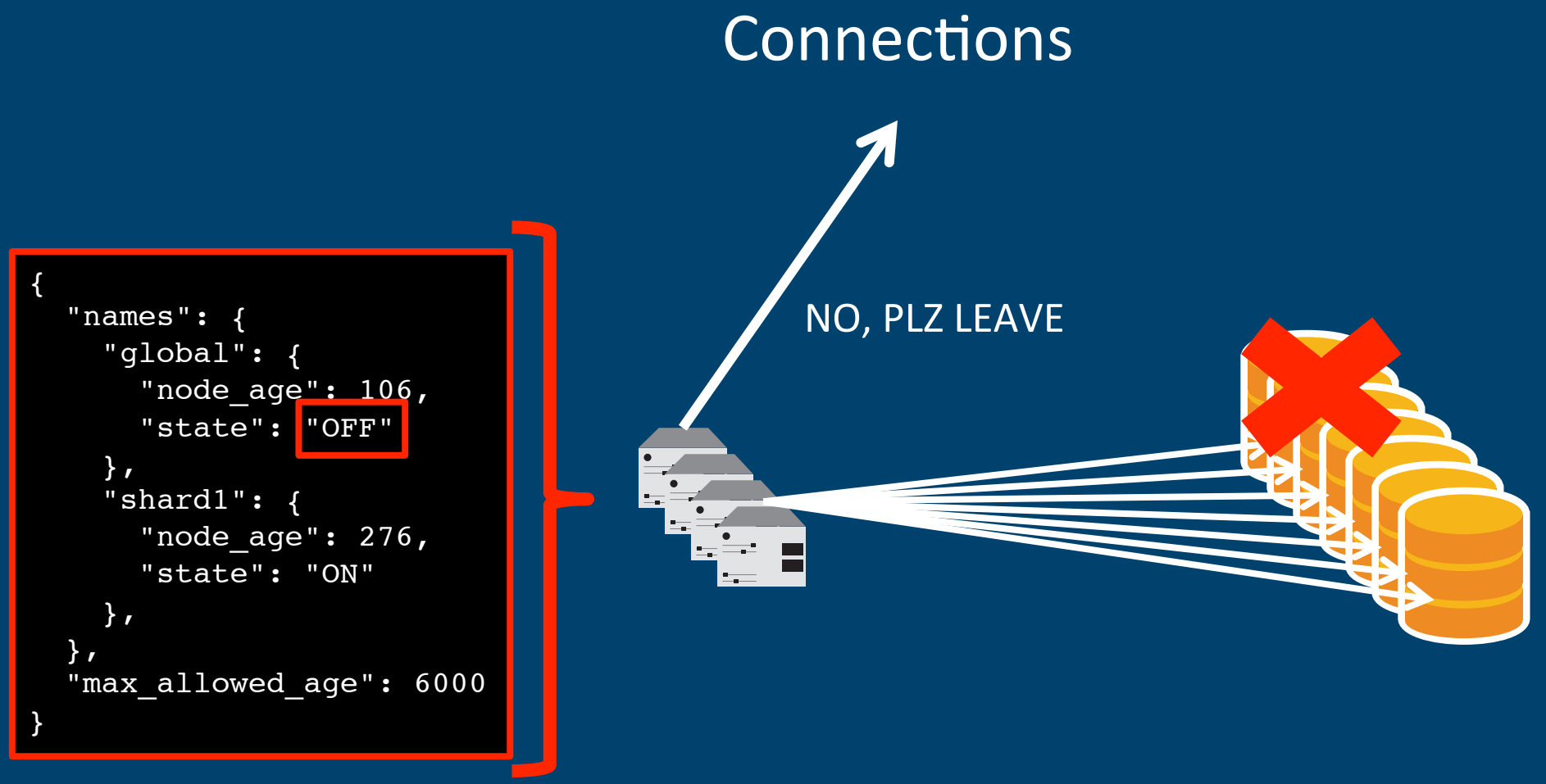
Defense #12 – Healthchecking



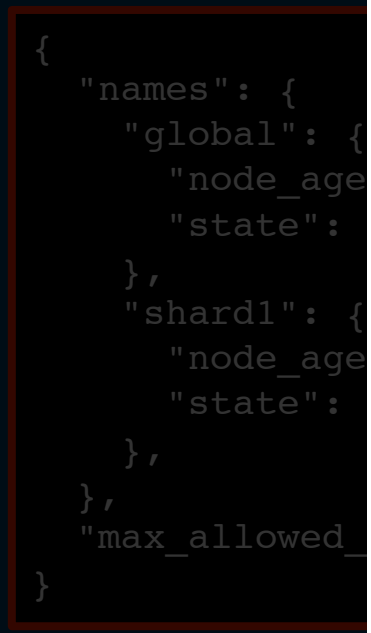
Defense #12 – Healthchecking



Defense #12 – Healthchecking



Defense #12 – Healthchecking



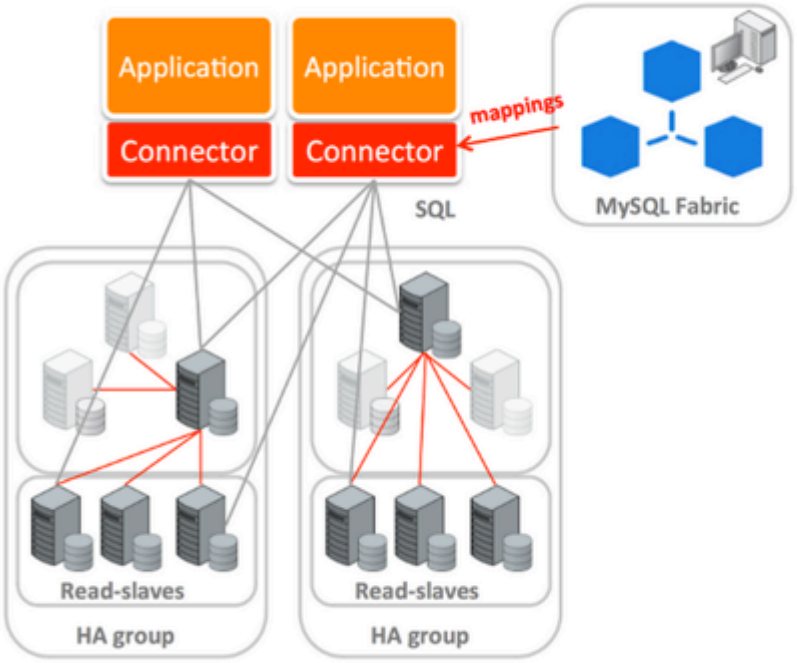
MySQL Fabric

MySQL Fabric is an extensible framework for managing farms of MySQL Servers. Two primary features have been implemented - High Availability (HA) and scaling out using data sharding. These features can be used in isolation or in combination. MySQL Fabric also provides server provisioning using cloud frameworks such as OpenStack.

MySQL Fabric Resource Kit »

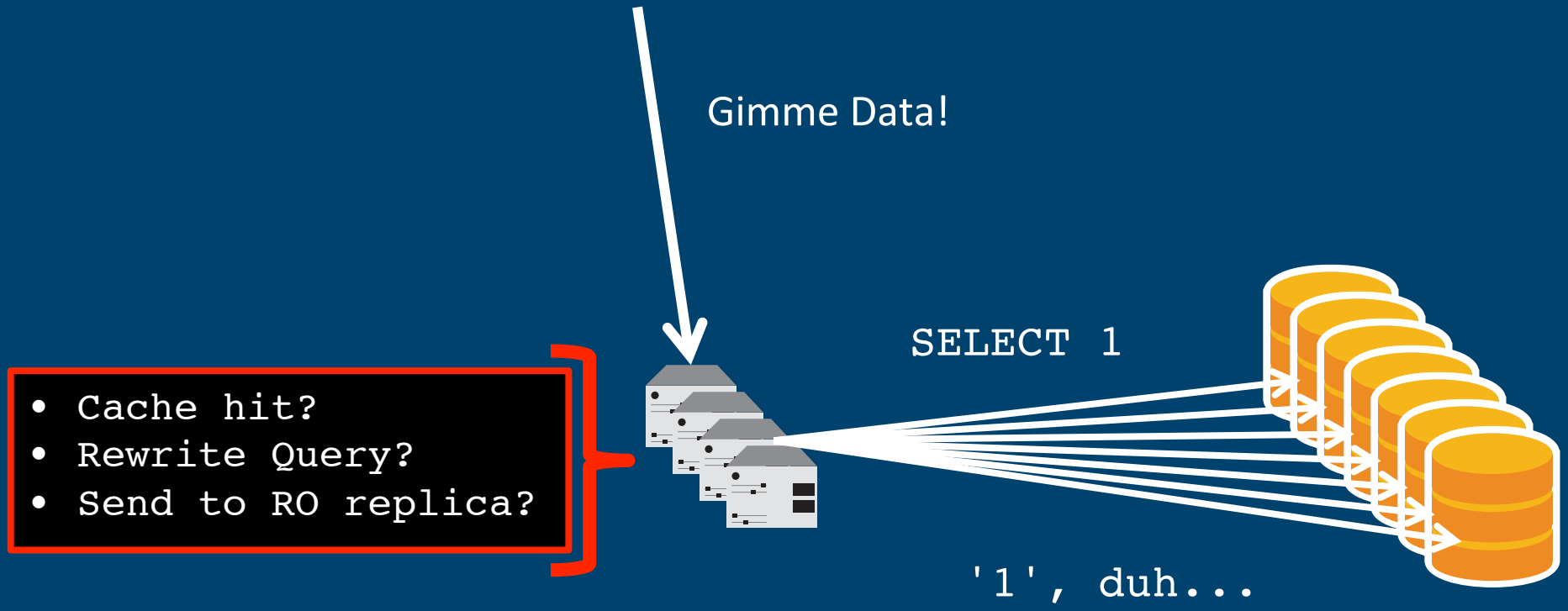
Both features are implemented in two layers:

- The mysqlfabric process which processes any management requests. When using the HA feature, this process can also be made responsible for monitoring the master server and initiating failover to promote a slave to be the new master should it fail.
- MySQL Fabric-aware connectors store a cache of the routing information that it has fetched from MySQL Fabric and then uses that information to send transactions or queries to the correct MySQL Server.



MySQL Fabric provides high availability and database sharding for MySQL Servers

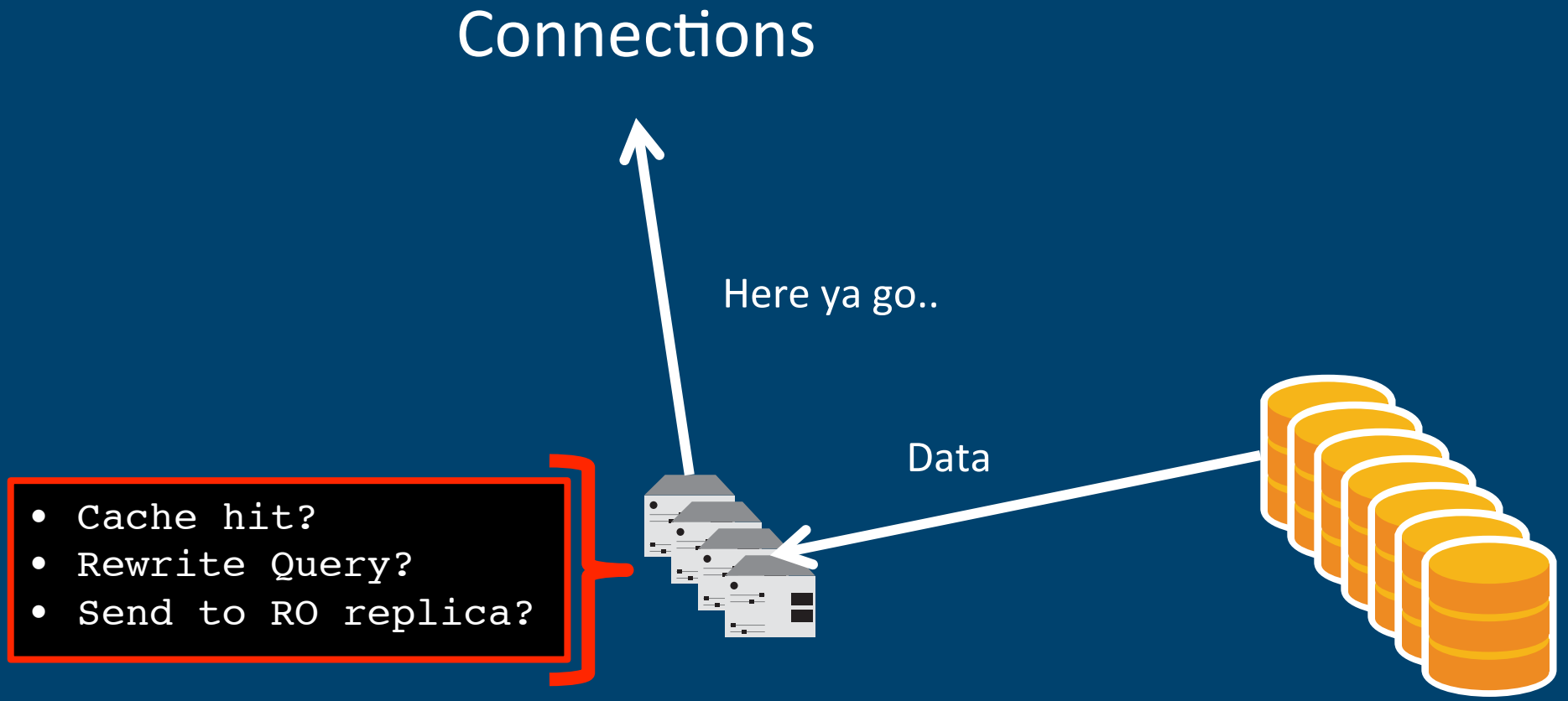
Connections



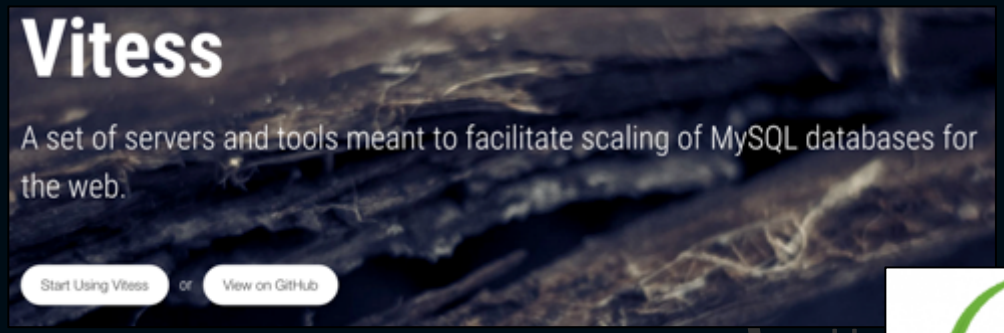
Defense #13 – Data Layer as a Service

Connections

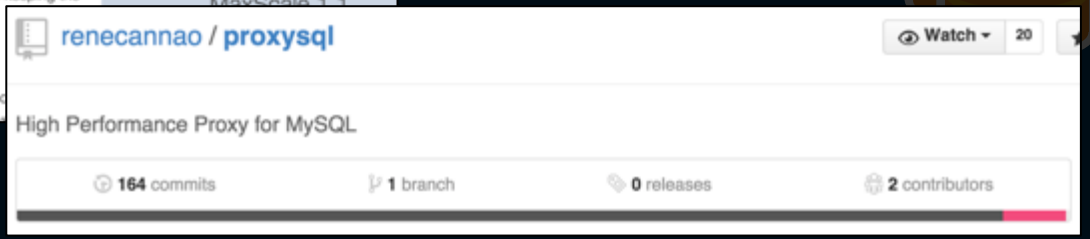
Defense #13 – Data Layer as a Service



Defense #13 – Data Layer as a Service

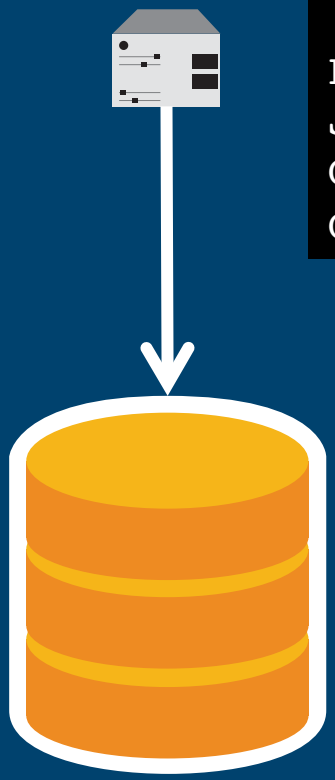


ScaleArc



Locks happen…

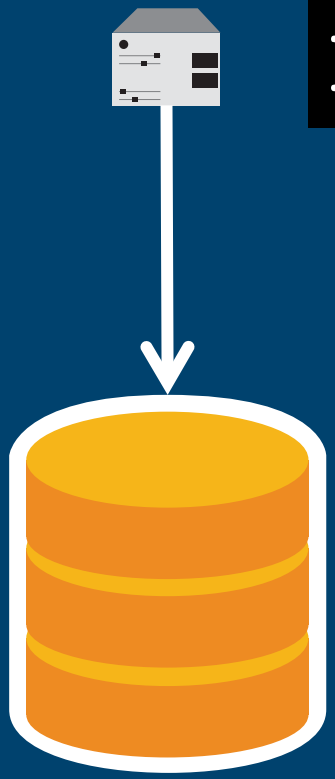
Defense #14 – Lock Monitor



```
SELECT *,
  UNIX_TIMESTAMP() - UNIX_TIMESTAMP(t.TRX_STARTED) as age
FROM INFORMATION_SCHEMA.INNODB_LOCKS l
JOIN INFORMATION_SCHEMA.INNODB_TRX t
ON t.trx_id=l.lock_trx_id
ORDER BY age DESC
```

```
+--------------------+-----------+-------------------+-----+-----+
| trx_mysql_thread_id | trx_state | lock_table        | ... | age |
+--------------------+-----------+-------------------+-----+-----+
|                419 | RUNNING   | `sakila`.`rental` | ... | 544 |
|                581 | LOCK WAIT | `sakila`.`rental` | ... |  49 |
|                631 | LOCK WAIT | `sakila`.`rental` | ... |  49 |
|                573 | LOCK WAIT | `sakila`.`rental` | ... |  49 |
...
500 rows in set (0.00 sec)
```

Defense #14 – Lock Monitor (and killer)



```
KILL 419 /* WITH FIRE */;
```

```
+--------------------+-----------+-------------------+-----+-----+
| trx_mysql_thread_id | trx_state | lock_table        | ... | age |
+--------------------+-----------+-------------------+-----+-----+
|                419 | RUNNING   | `sakila`.`rental` | ... | 544 |
|                581 | LOCK WAIT | `sakila`.`rental` | ... |  49 |
|                631 | LOCK WAIT | `sakila`.`rental` | ... |  49 |
|                573 | LOCK WAIT | `sakila`.`rental` | ... |  49 |
...
500 rows in set (0.00 sec)
```

Idle connections…

```
mysql [localhost] {root} ((none)) > show processlist;
+------+------+----------------+--------+---------+-------+---------------+-----------------------------------------
| Id   | User | Host           | db     | Command | Time  | State         | Info
+------+------+----------------+--------+---------+-------+---------------+-----------------------------------------
|    1 | root | localhost      | NULL   | Query   |     0 | init          | show processlist
| 7231 | app  | localhost:63146 | NULL   | Sleep   |     4 |               | NULL
| 7242 | app  | localhost:63157 | NULL   | Sleep   | 28800 |               | NULL
| 7450 | app  | localhost:63367 | sakila | Query   |     0 | Sending data  | SELECT * FROM rental WHERE customer_id=9 AN
| 7451 | app  | localhost:63366 | sakila | Query   |     0 | Sending data  | SELECT * FROM rental WHERE customer_id=9 AN
| 7466 | app  | localhost:63382 | sakila | Query   |     0 | Sending data  | SELECT * FROM rental WHERE customer_id=11 A
| 7468 | app  | localhost:63384 | sakila | Query   |     0 | Sending data  | SELECT * FROM rental WHERE customer_id=11 A
| 7469 | app  | localhost:63385 | sakila | Query   |     0 | Writing to net | SELECT * FROM rental WHERE customer_id=11 A
| 7470 | app  | localhost:63387 | sakila | Query   |     0 | Sending data  | SELECT * FROM rental WHERE customer_id=5 AN
| 7471 | app  | localhost:63386 | sakila | Query   |     0 | Sending data  | SELECT * FROM rental WHERE customer_id=5 AN
| 7472 | app  | localhost:63388 | sakila | Query   |     0 | Sending data  | SELECT * FROM rental WHERE customer_id=5 AN
| 7473 | app  | localhost:63389 | sakila | Sleep   |     0 |               | NULL
```

Idle connections…

```
mysql [localhost] {root} ((none)) > show processlist;
+------+------+----------------+--------+---------+------+---------------+------------------+
| Id   | User | Host           | db     | Command | Time | State         | Info             |
+------+------+----------------+--------+---------+------+---------------+------------------+
|    1 | root | localhost      | NULL   | Query   |    0 | init          | show processlist |
| 7231 | app  | localhost:63146 | NULL   | Sleep   |    4 |               | NULL             |
| 7242 | app  | localhost:63157 | NULL   | Sleep   | 28800 |              | NULL             |
| 7450 | app  | localhost:63367 | sakila | Sleep   | 28800 |              | NULL             |
| 7451 | app  | localhost:63366 | sakila | Sleep   | 28800 |              | NULL             |
| 7466 | app  | localhost:63382 | sakila | Sleep   | 28800 |              | NULL             |
| 7468 | app  | localhost:63384 | sakila | Sleep   | 28800 |              | NULL             |
| 7469 | app  | localhost:63385 | sakila | Sleep   | 28800 |              | NULL             |
| 7470 | app  | localhost:63387 | sakila | Sleep   | 28800 |              | NULL             |
| 7471 | app  | localhost:63386 | sakila | Sleep   | 28800 |              | NULL             |
| 7472 | app  | localhost:63388 | sakila | Sleep   | 28800 |              | NULL             |
| 7473 | app  | localhost:63389 | sakila | Sleep   | 28800 |              | NULL             |
```

Idle connections…

How to find an errant MySQL client

A common story: You've got some connection, either it's busy running something it shouldn't be, it's in `Sleep` but holding some important lock, or you just don't know why it's connected to your database server in the first place. You see it in your `SHOW PROCESSLIST` like so:

```
mysql> show processlist G
*************************** 1. row ***************************
     Id: 5979887
   User: root
   Host: localhost:55997
     db: NULL
Command: Sleep
   Time: 475
  State:
   Info: NULL
```

How do you find that client, especially if it's on another host? MySQL is providing you all the information you need above: `localhost:55997`. Of course `localhost` is the host or IP address, and `55997` is the source port of the socket; the port number (usually randomly assigned) on the far end of the socket, from the MySQL server's perspective. You can turn that number into something useful—the PID and user—, by running the following command on the host that made the connection:

```
# lsof -nPi :55997
COMMAND  PID  USER   FD   TYPE    DEVICE SIZE NODE NAME
mysqld  5026 mysql 1654u  IPv4 303329996       TCP 127.0.0.1:3306->127.0.0.1:559
97 (ESTABLISHED)
mysql   9146 jcole    3u  IPv4 303329995       TCP 127.0.0.1:55997->127.0.0.1:33
06 (ESTABLISHED)
```

(Note that here you can see both sides of the socket, since I'm running these commands on localhost. Disregard the first entry, as it's the half of the connection owned by the MySQL server.)

You can then find out the full command-line of the process with `ps`:

```
# ps -fp 9146
UID        PID  PPID  C STIME TTY          TIME CMD
jcole     9146  8740  0 12:53 pts/3    00:00:00 mysql -h 127.0.0.1 -u root -p
```

And see what it's doing with `strace` (waiting on a read of `stdin` in this dumb test):

```
# strace -p 9146
Process 9146 attached - interrupt to quit
read(0,  <unfinished ...>
Process 9146 detached
```

I hope that's helpful! It's a pretty common debugging trick for me.

Defense #15 – strace errant connections

```
$ db-strace-longest-trx mydb.example.com

Longest transaction on mydb.example.com:3306
MySQL thread id is 15, client is 10.0.0.5:51540, running for 08:00:07
Warning: Permanently added '10.0.0.5' (RSA) to the list of known hosts.
Process 19820 attached - interrupt to quit
write(3, "...SELECT * FROM demo_test /* ORM_READ(732):...", 28) = 28
read(3, "\1\0\0\1\0050\0\0\2\3def\4test\tdemo_test\t... ", 16384) = 309
pipe([4, 5])                                = 0
[pid 19820] write(5, "+----+-------------+------+------+------+\n| c1
 | c2          | c3   | c4   | c5   |\n+----+-------------+------+--
----+------+\n| AA | HELLO, HELLO |    8 |    0 |    0 |\n+----+------
-------+------+------+------+\n", 215 <unfinished ...>
pid [19820] close(5)                        = 0
```

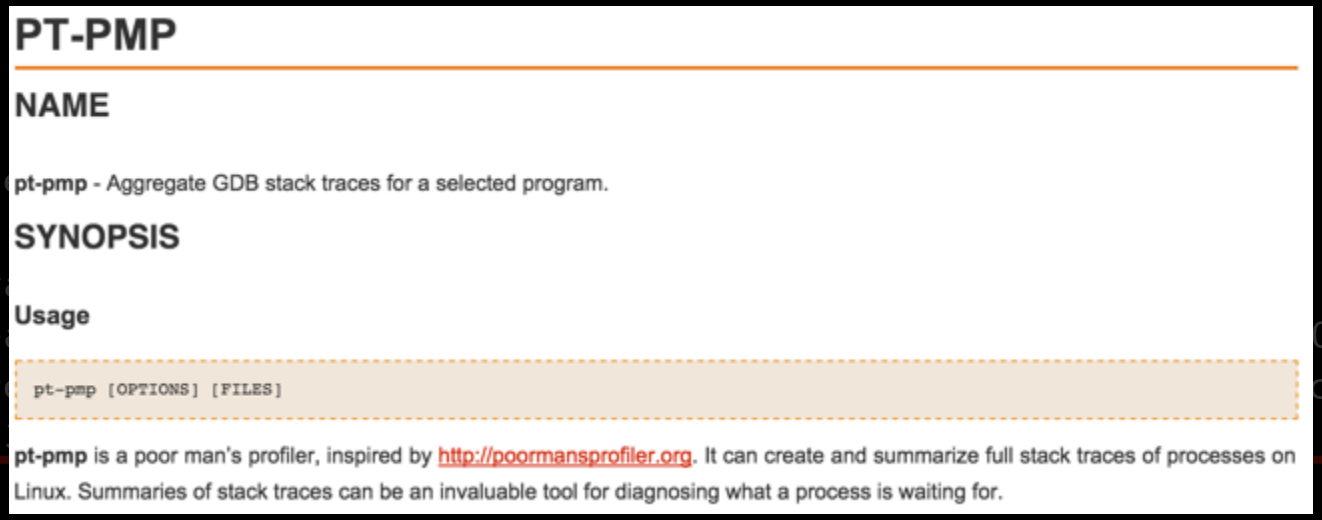
Even idler connections!

```
$ db-strace-longest-trx mydb.example.com

Longest transaction on mydb.example.com:3306
MySQL thread id is 19, client is 10.0.0.5:51740, running for 08:00:12
Warning: Permanently added '10.0.0.5' (RSA) to the list of known hosts.
Process 1833 attached - interrupt to quit
```

Where's the thing!?

Defense #16 – pt-pmp



PT-PMP

NAME

pt-pmp - Aggregate GDB stack traces for a selected program.

SYNOPSIS

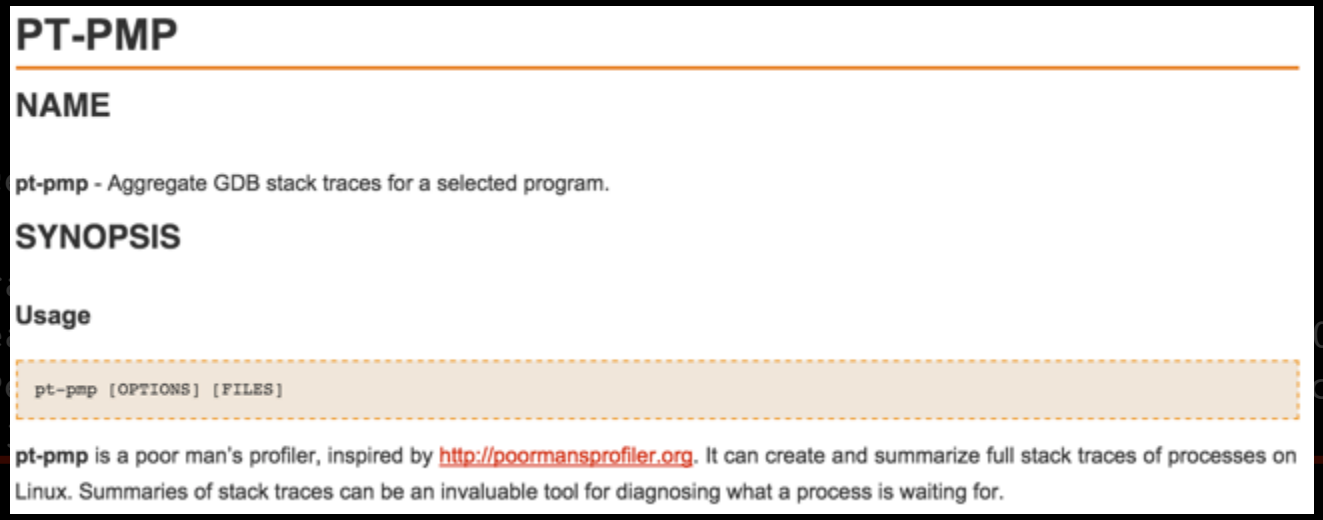
Usage

```
pt-pmp [OPTIONS] [FILES]
```

pt-pmp is a poor man's profiler, inspired by http://poormansprofiler.org. It can create and summarize full stack traces of processes on Linux. Summaries of stack traces can be an invaluable tool for diagnosing what a process is waiting for.

```
gdb -ex "set pagination 0"                                    \
    -ex "bt"                                                  \
    -ex "source /path/to/php-src/.gdbinit"                    \
    -ex "dump_bt executor_globals.current_execute_data" \
    -batch                                                    \
    -p "$pid"                                                 \
    >> "$output_file"
```

Defense #16 – ~~pt-pmp~~ php-pmp



PT-PMP

NAME

pt-pmp - Aggregate GDB stack traces for a selected program.

SYNOPSIS

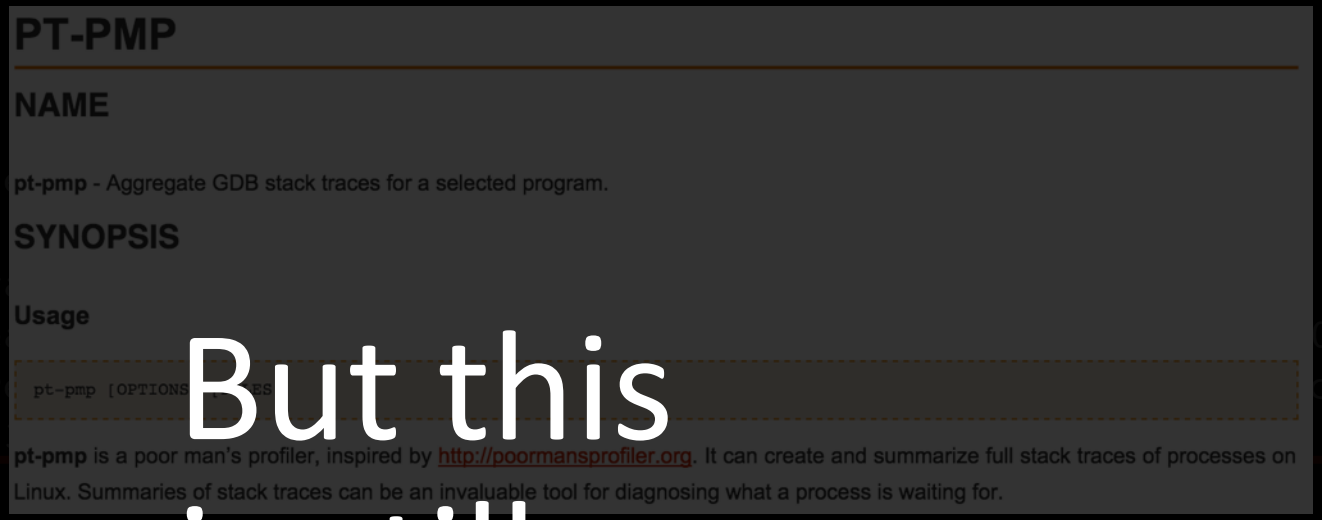
Usage

```
pt-pmp [OPTIONS] [FILES]
```

pt-pmp is a poor man's profiler, inspired by http://poormansprofiler.org. It can create and summarize full stack traces of processes on Linux. Summaries of stack traces can be an invaluable tool for diagnosing what a process is waiting for.

```
gdb -ex "set pagination 0"                                    \
    -ex "bt"                                                  \
    -ex "source /path/to/php-src/.gdbinit"                    \
    -ex "dump_bt executor_globals.current_execute_data" \
    -batch                                                    \
    -p "$pid"                                                 \
    >> "$output_file"
```

Defense #16 – ~~pt-pmp~~ php-pmp



PT-PMP

NAME

pt-pmp - Aggregate GDB stack traces for a selected program.

SYNOPSIS

Usage

```
pt-pmp [OPTIONS
```

pt-pmp is a poor man's profiler, inspired by http://poormansprofiler.org. It can create and summarize full stack traces of processes on Linux. Summaries of stack traces can be an invaluable tool for diagnosing what a process is waiting for.

But this is still too manual…

```
gdb -ex "set pagination 0"                                          \
    -ex "bt"                                                        \
    -ex "source /path/to/php-src/.gdbinit"                          \
    -ex "dump_bt executor_globals.current_execute_data"             \
    -batch                                                          \
    -p "$pid"                                                       \
    >> "$output_file"
```

Defense #16.1 – Autorun php-pmp

```
$ pt-kill --interval        10                                          \
          --match-user      '^app_rw.*'                                 \
          --busy-time       120s                                        \
          --ignore-command  '(Prepare|Execute)'                         \
          --match-info      '^\s*\(?\s*(SELECT|select)'                 \
          --ignore-info     '(.* SQL_NO_CACHE .*|.*&plz_no_kill=1.*)' \
          --idle                                                        \
          --vict                                                        \
          --prin                                                        \
          --kill                                                        \
          --log             /var/log/qkkill.log                         \
          --execute-command '(mail –s "Killed queries on ${HOSTNAME}" \
            "database-peeps@box.com" <<-EOF
Last 10 killed queries in /var/log/qkkill.log:
$(tail -10 /var/log/qkkill.log)
EOF
)'                                                                      \
          --daemonize                                                   \
          --pid             /var/run/qkkill.pid
```

Change this to…

Defense #16.1 – Autorun php-pmp

```
#!/usr/bin/env bash
SUDOUSER="geoff"
QK_FILE="/var/log/qkkill.log"
date="$(date +%Y_%m_%d_%H_%M_%S)"
TMP_FILE="/tmp/${date}-qkkill-trace"
trace_file="/tmp/${HOSTNAME}-${date}.tar.gz"
trace_url="http://raingauge.example.com"
PORT='3306'

cleanup() {
        rm -f "$TMP_FILE" "$trace_file"
}

query="$(tac "${QK_FILE}" | sed -n '0,/^#/ p' | tac )"
mysql_client="$(tac "${QK_FILE}" | sed -n '0,/^#/ p' | tail -n1 | awk '{ print $8 }' | cut -d"'" -f4)"
echo "mysql client host: ${mysql_client}" | tee "$TMP_FILE"
if [[ -z "${mysql_client}" ]]
then
        echo "Can't extract cilent host and port from pt-kill log ${mysql_client}"
        cleanup
        exit 1;
fi
IFS=':' read -ra mysql_host <<< "${mysql_client}"

pid="$( sudo -u "${SUDOUSER}" ssh -o StrictHostKeyChecking=no "${mysql_host[0]}" 'sudo /usr/sbin/lsof | grep '"${mysql_host[1]}"' | awk '"'"'{ print $2 }'"'"')"
echo "remote pid: ${pid}" | tee -a "$TMP_FILE"
echo "${query}" | tee -a "$TMP_FILE"
if [[ -z "${pid}" ]]
then
        echo "Can't find remote pid: ${mysql_host[0]}, ${mysql_host[1]}"
        cleanup
        exit 1
fi

sudo -u "${SUDOUSER}" ssh -n -o StrictHostKeyChecking=no ${mysql_host[0]} 'sudo /bin/php-pmp '"${pid}" | tee -a "$TMP_FILE"

# Package and ship this recent stack dump to the qkviewer interface
tar -zcf "$trace_file" -C "${TMP_FILE%/*}" "${TMP_FILE##*/}"
curl -F "file=@${trace_file}" "${trace_url}/index.php?action=upload&hostname=${HOSTNAME}&port=${PORT}"
cleanup
```

Defense #16.1 – Autorun php-pmp

```
#!/usr/bin/env bash
SUDOUSER="geoff"
QK_FILE="/var/log/qkkill.log"
date="$(date +%Y_%m_%d_%H_%M_%S)"
TMP_FILE="/tmp/${date}-qkkill-trace"
trace_file="/tmp/${HOSTNAME}-${date}.tar.gz"
trace_url="http://raingauge.example.com"
PORT='3306'

cleanup() {
        rm -f "$TMP_FILE" "$trace_file"
}

query="$(tac "${QK_FILE}" | sed -n '0,/^#/ p' | tac )"
mysql_client="$(tac "${QK_FILE}" | sed -n '0,/^#/ p' | tail -n1 | awk '{ print $8 }' | cut -d"'" -f4)"
echo "mysql client host: ${mysql_client}" | tee "$TMP_FILE"
if [[ -z "${mysql_client}" ]]
then
        echo "Can't extract cilent host and port from pt-kill log ${mysql_client}"
        cleanup
        exit 1;
fi
IFS=':' read -ra mysql_host <<< "${mysql_client}"

pid="$( sudo -u "${SUDOUSER}" ssh -o StrictHostKeyChecking=no "${mysql_host[0]}" '                    ql_host[1]}"' | awk '"'"'{ print $2 }'"'")"
echo "remote pid: ${pid}" | tee -a "$TMP_FILE"
echo "${query}" | tee -a "$TMP_FILE"
if [[ -z "${pid}" ]]
then
        echo "Can't find remote pid: ${mysql_hos
        cleanup
        exit 1
fi

sudo -u "${SUDOUSER}" ssh -n -o StrictHostKeyChecking=n    mysql_host[0]} 'sudo /bin/php-pmp '"${pid}" | tee -a "$TMP_FIL

# Package and ship this recent stack dump to the qkviewer interface
tar -zcf "$trace_file" -C "${TMP_FILE%/*}" "${TMP_FILE##*/}"
curl -F "file=@${trace_file}" "${trace_url}/index.php?action=upload&hostname=${HOSTNAME}&port=${PORT}"
cleanup
```

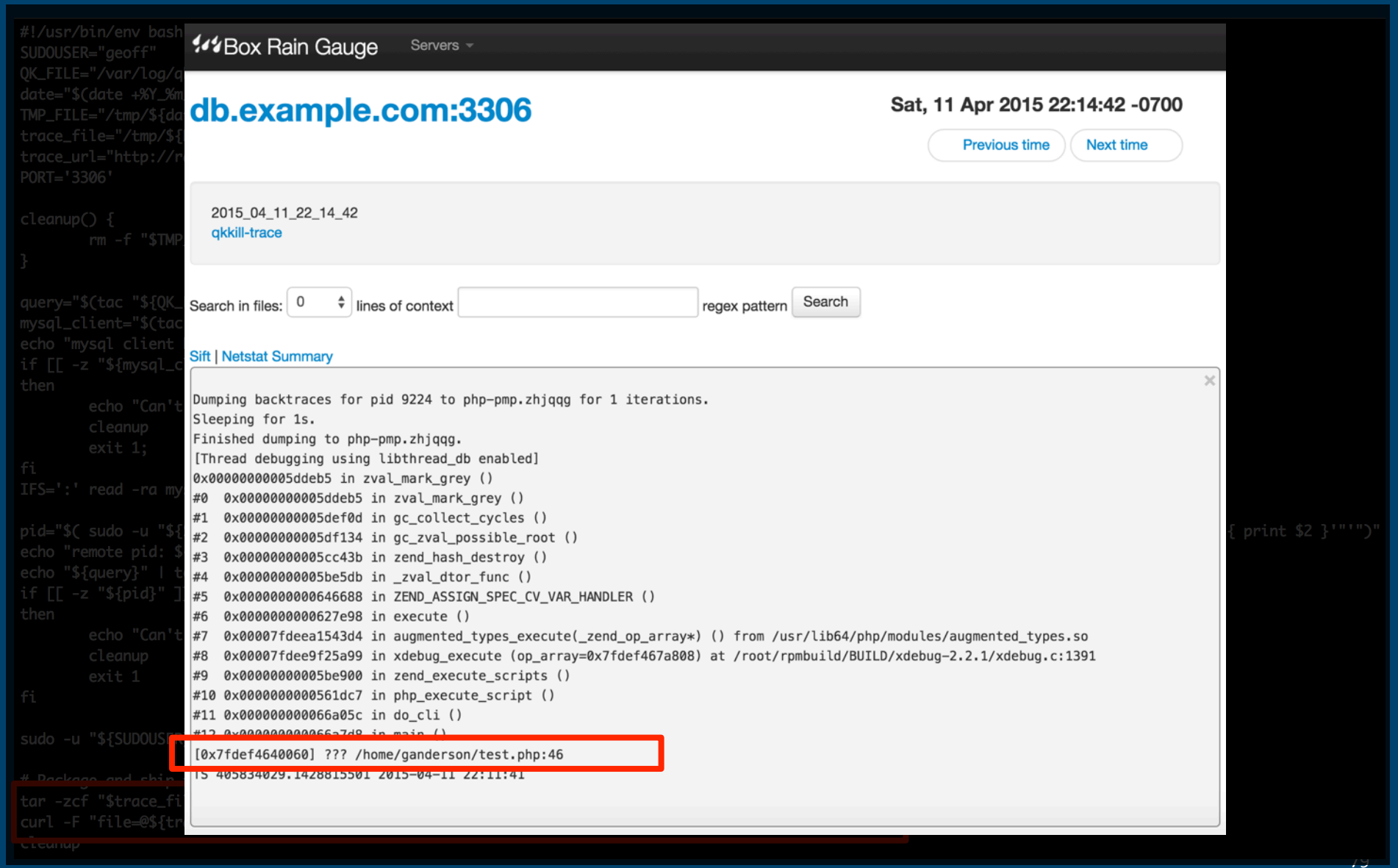
Extract source client info

Run php-pmp on client pid

Discover remote pid

Send to raingauge!

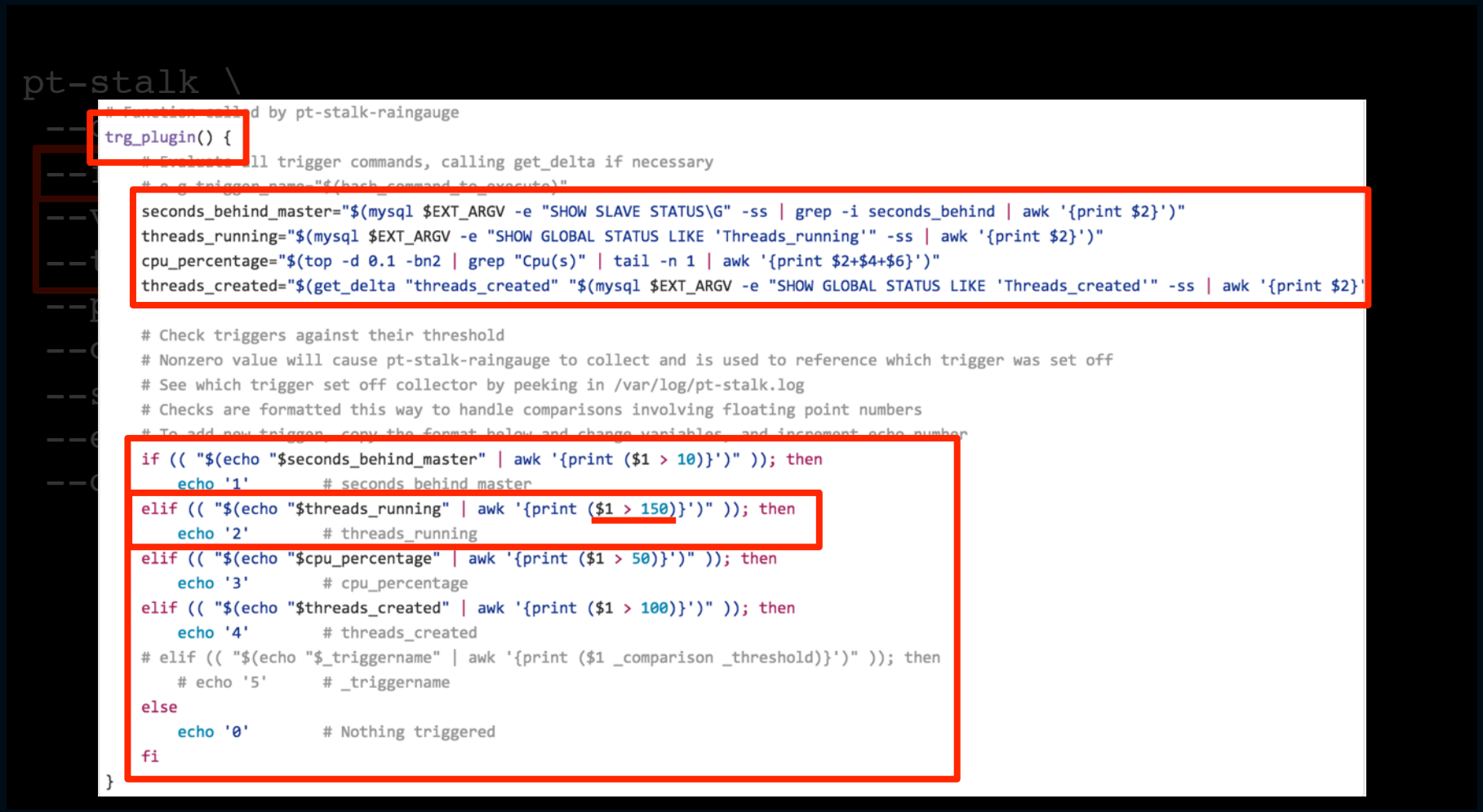
Defense #16.1 – Autorun php-pmp



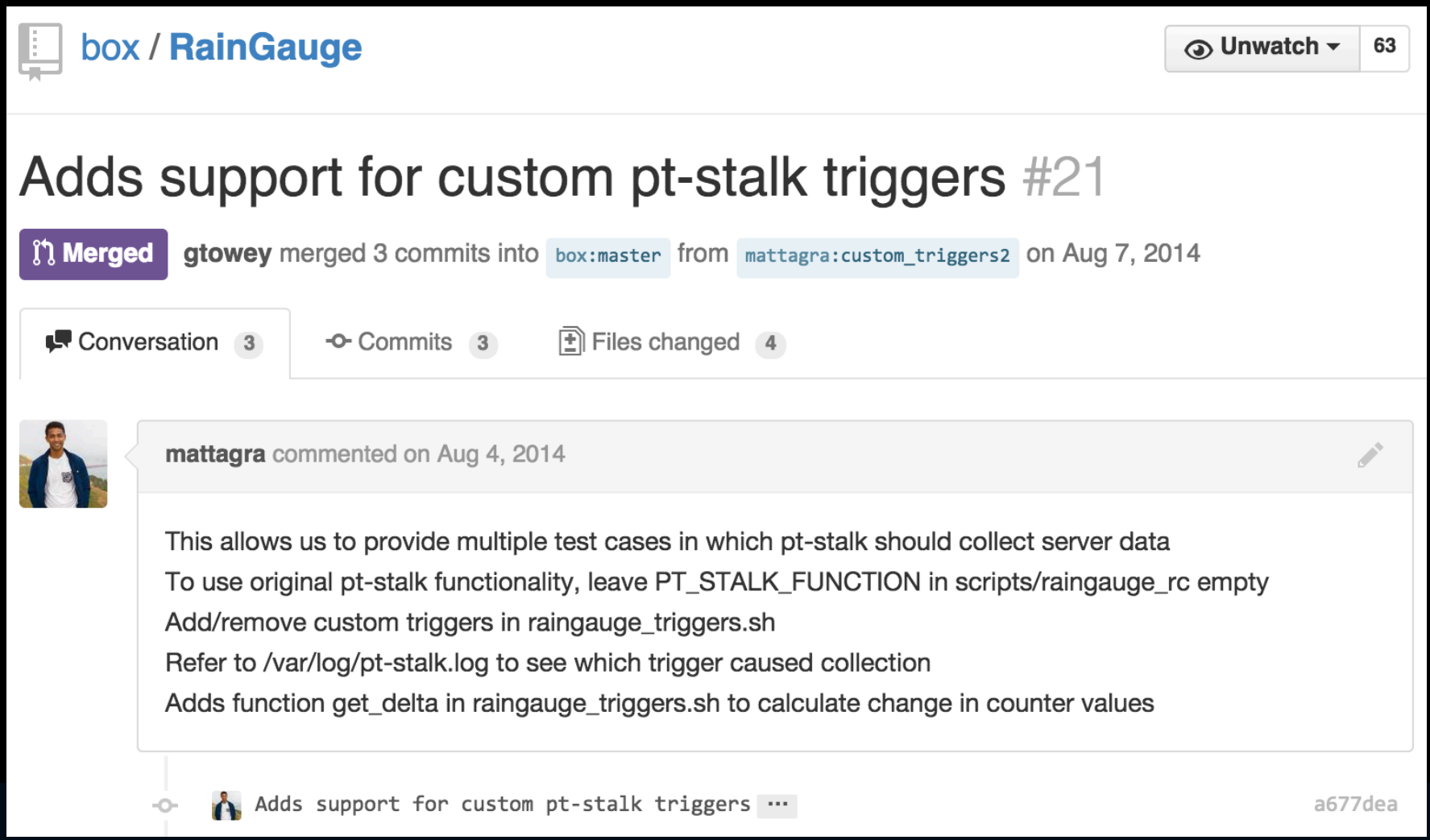
```
pt-stalk \
 --daemonize                                                    \
 --variable=Threads_running                                     \
 --threshold=150                                                \
 --pid=/usr/var/run/ptstalk.pid                                 \
 --cycles=2                                                     \
 --sleep=60                                                     \
 --exec-after-sleep=/usr/bin/raingauge_package_and_send.sh \
 --dest=/tmp/raingauge
```

What about Threads_Created?
What about CPU?

Defense #17 – Multivariable pt-stalk



Defense #17 – Multivariable pt-stalk



box / RainGauge

Unwatch 63

Adds support for custom pt-stalk triggers #21

Merged **gtowey** merged 3 commits into `box:master` from `mattagra:custom_triggers2` on Aug 7, 2014

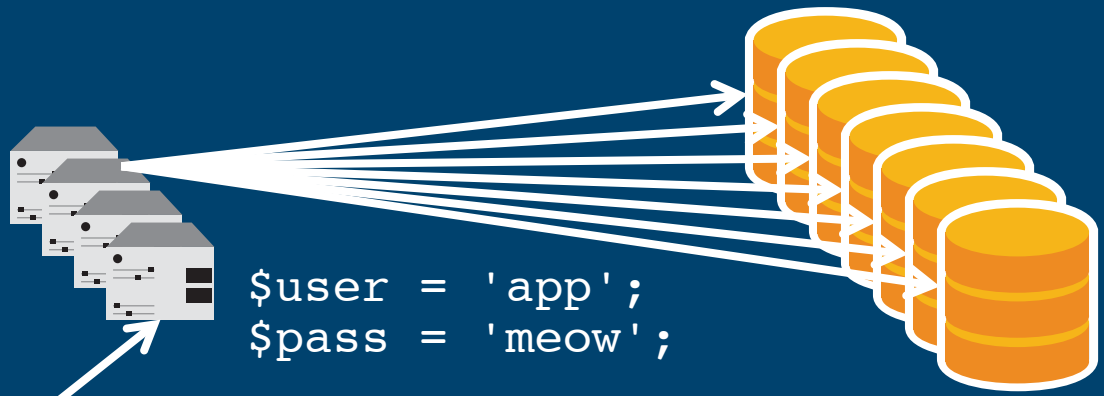
Conversation 3 | Commits 3 | Files changed 4

mattagra commented on Aug 4, 2014

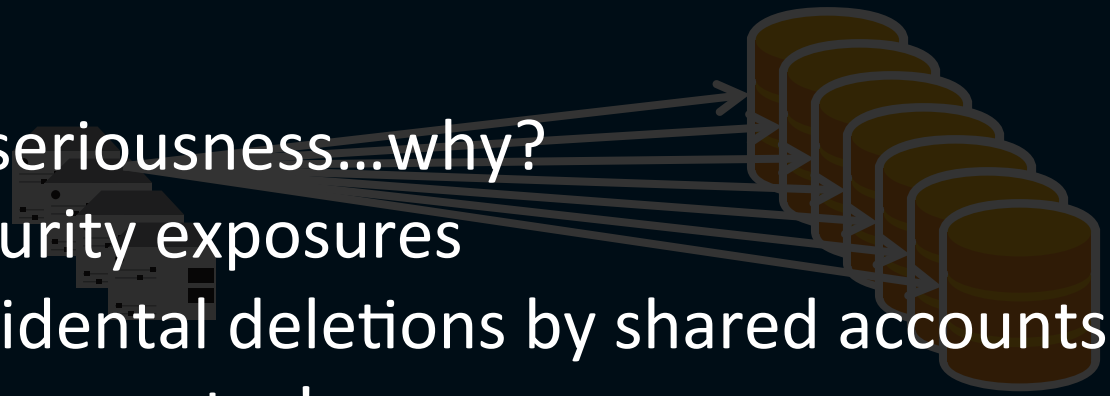
This allows us to provide multiple test cases in which pt-stalk should collect server data
To use original pt-stalk functionality, leave PT_STALK_FUNCTION in scripts/raingauge_rc empty
Add/remove custom triggers in raingauge_triggers.sh
Refer to /var/log/pt-stalk.log to see which trigger caused collection
Adds function get_delta in raingauge_triggers.sh to calculate change in counter values

`Adds support for custom pt-stalk triggers ···` a677dea

Time to rotate passwords!

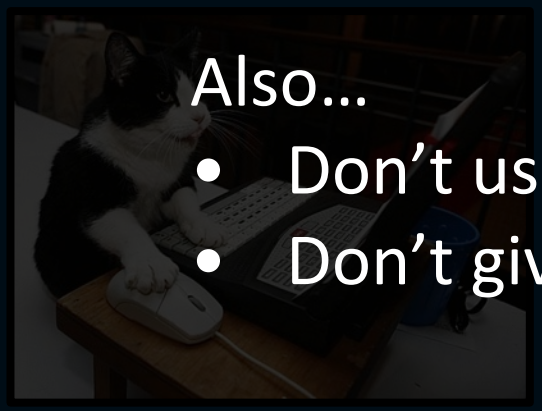


Defense #18 – Automated User Management



In all seriousness…why?

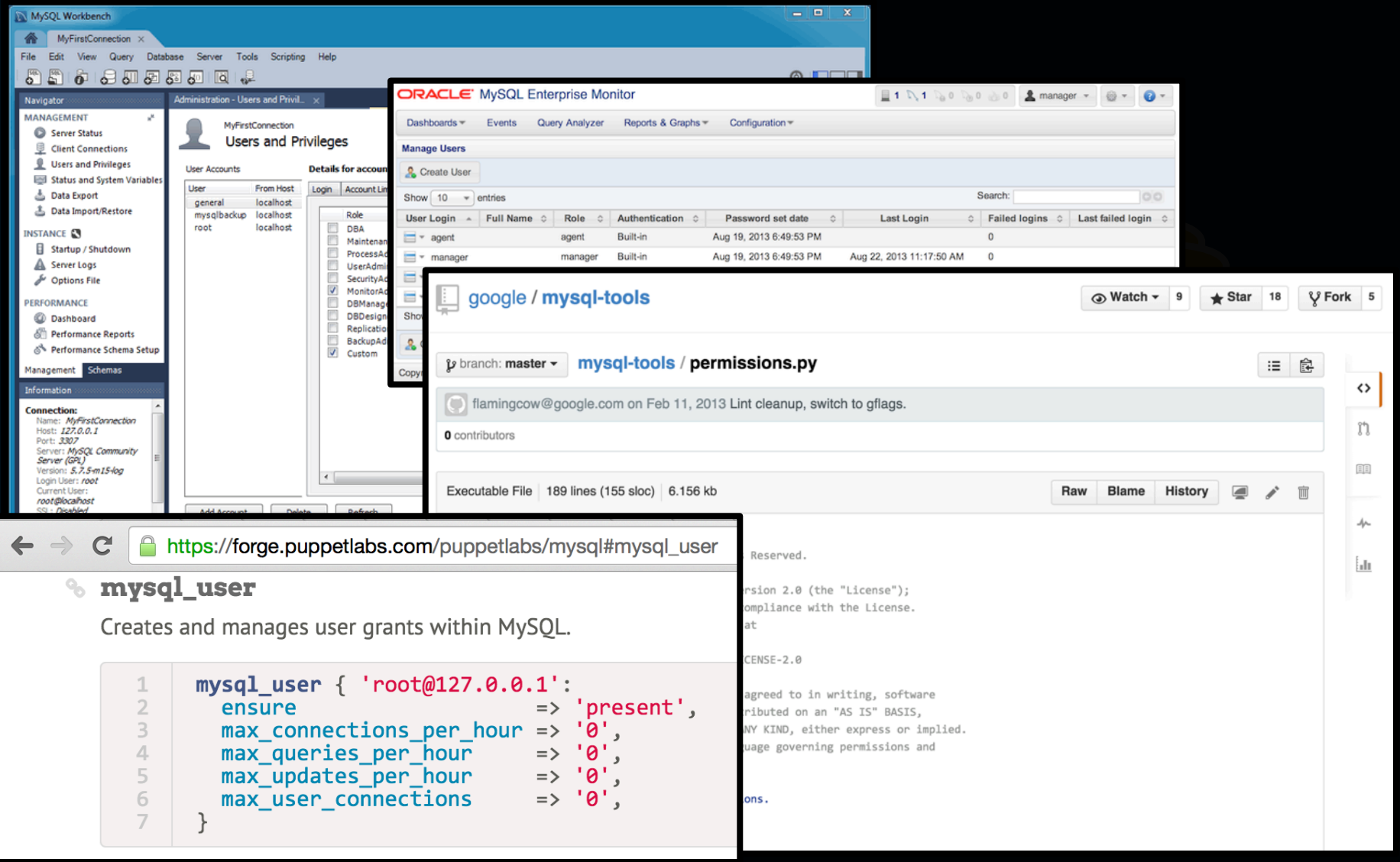
- Security exposures
- Accidental deletions by shared accounts
- Setup quotas!



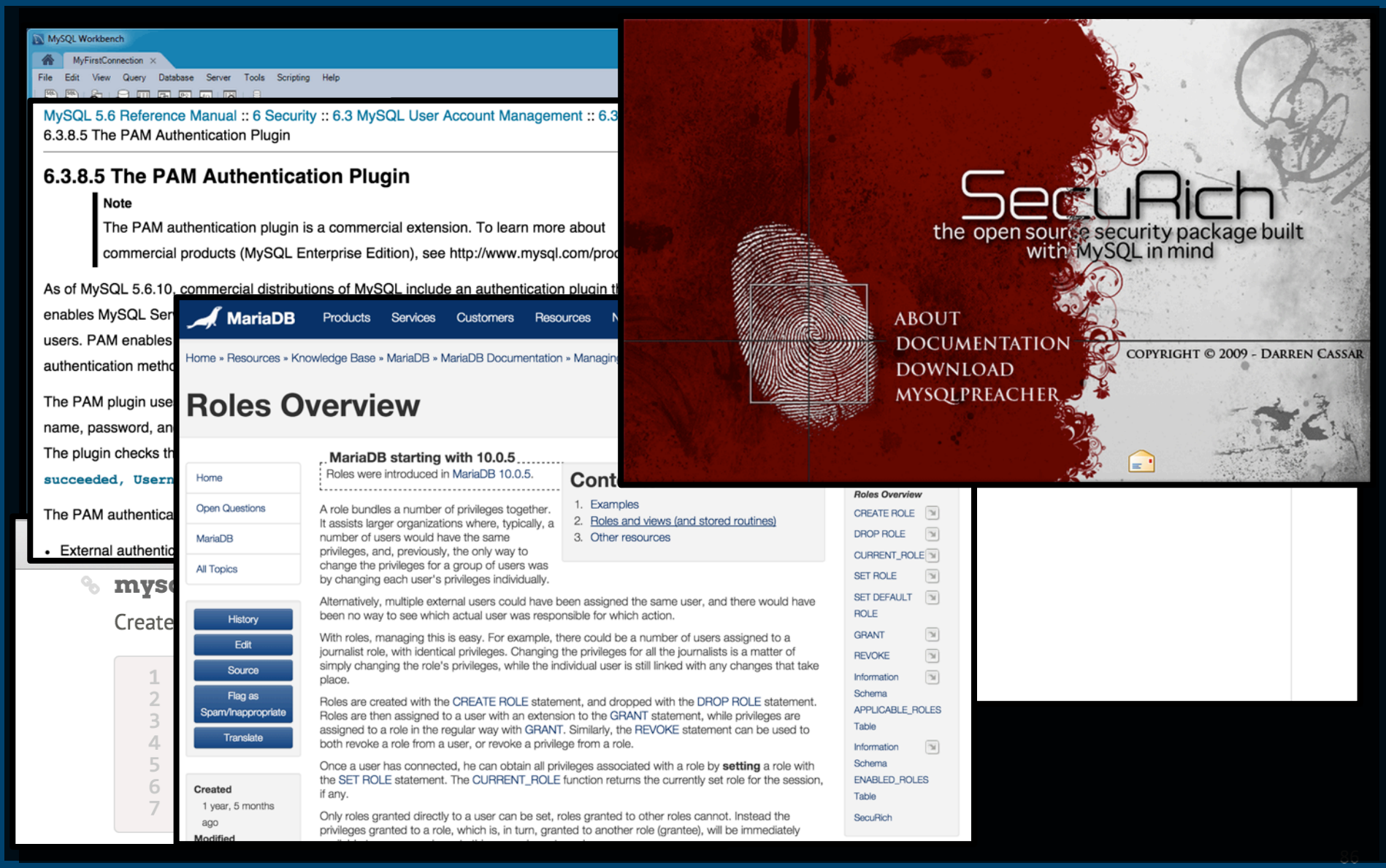
Also…

- Don’t use your root account
- Don’t give nagios the SUPER privilege…

Defense #18 – Automated User Management



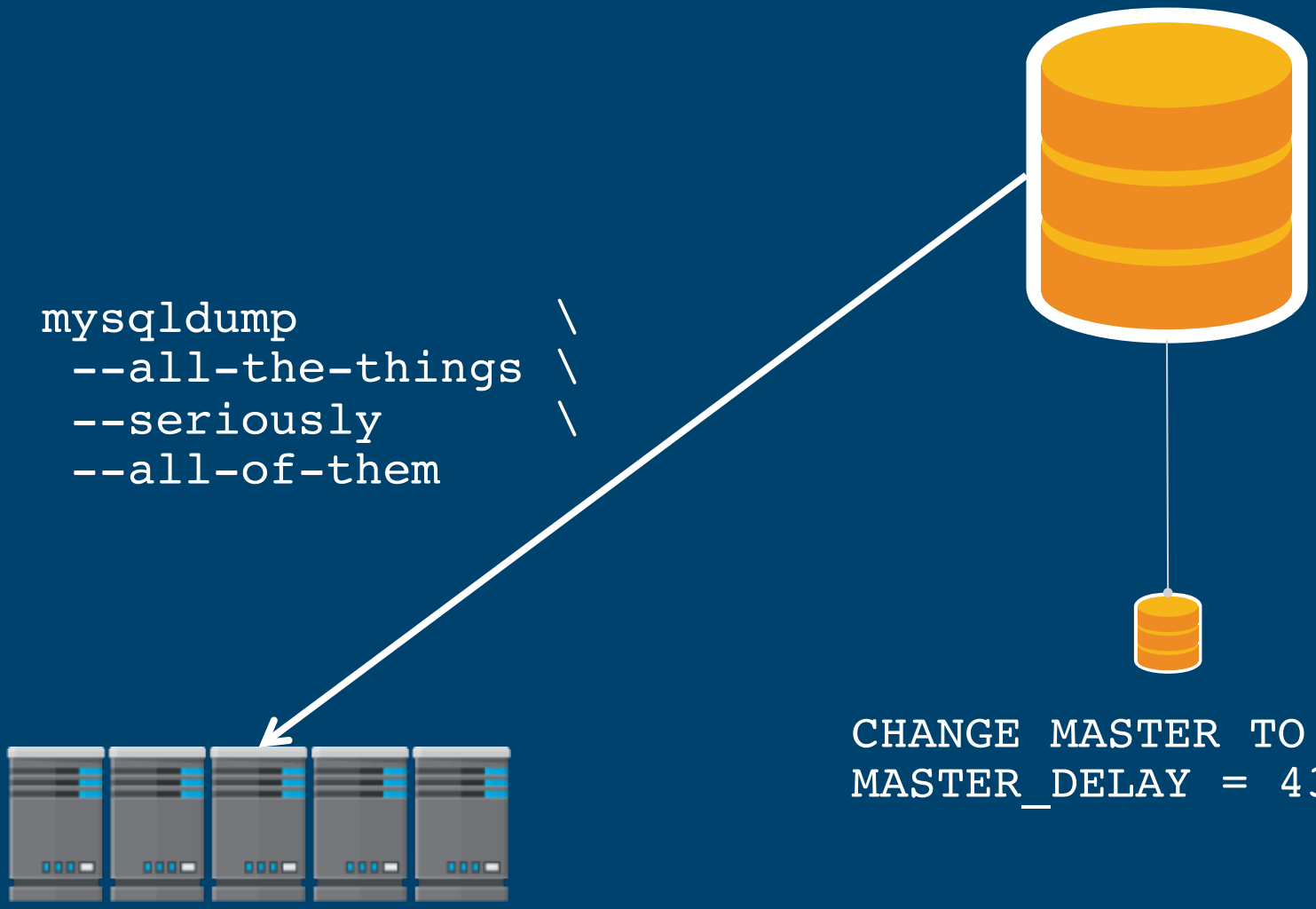
Defense #18 – Automated User Management



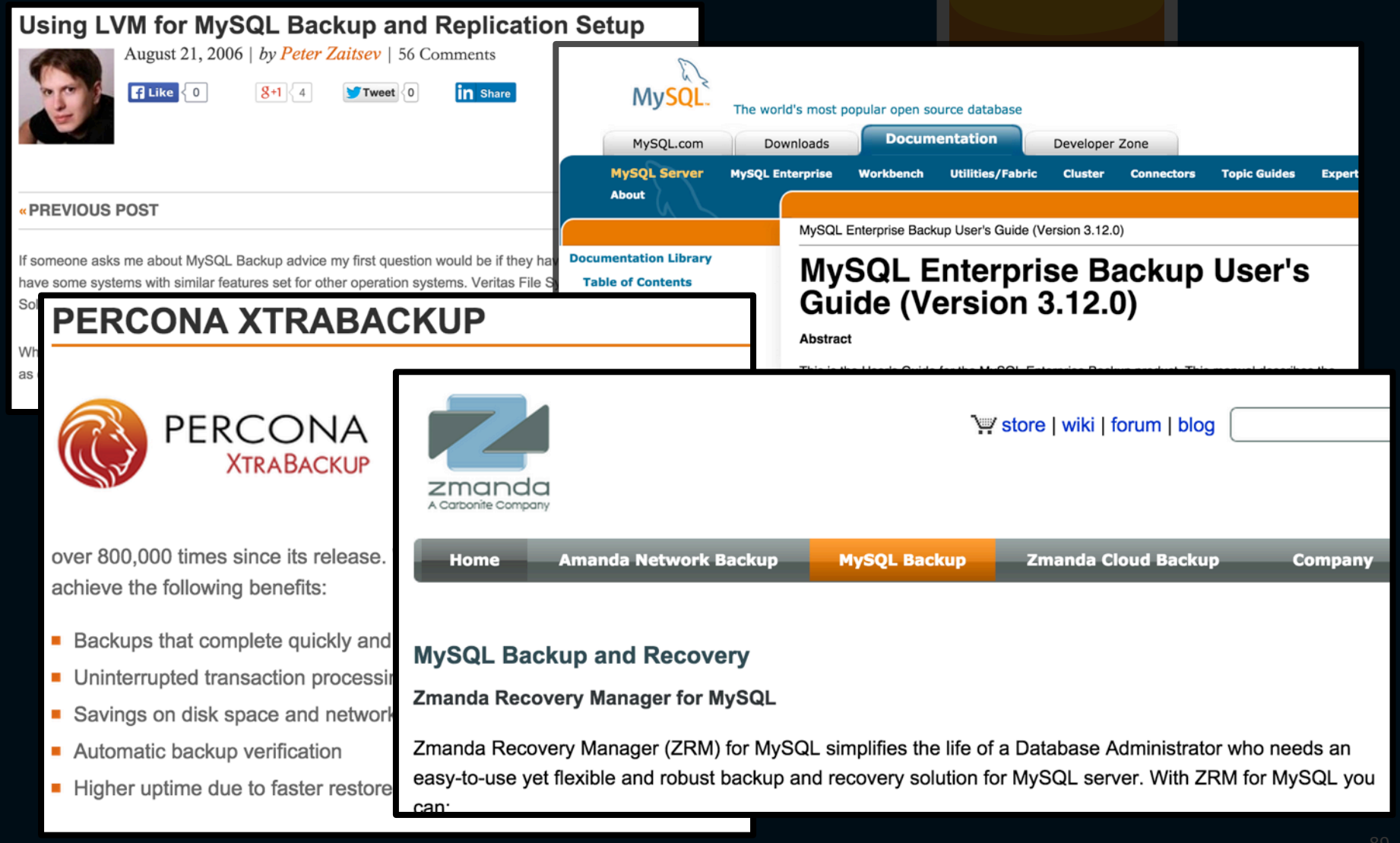
Backups

How many of you will take backups?
How many of you **actually** take backups?

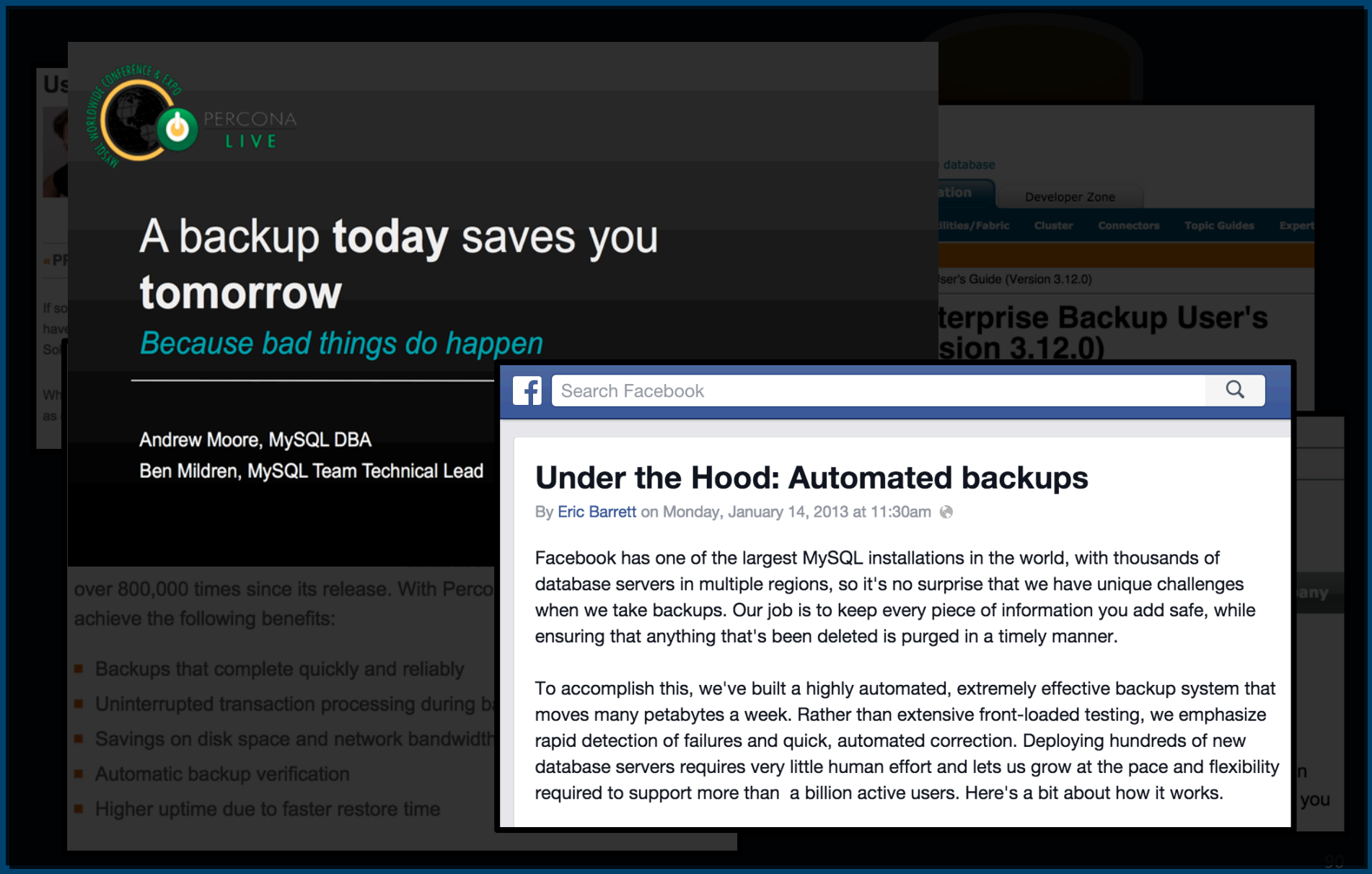
Defense #19 – Backups



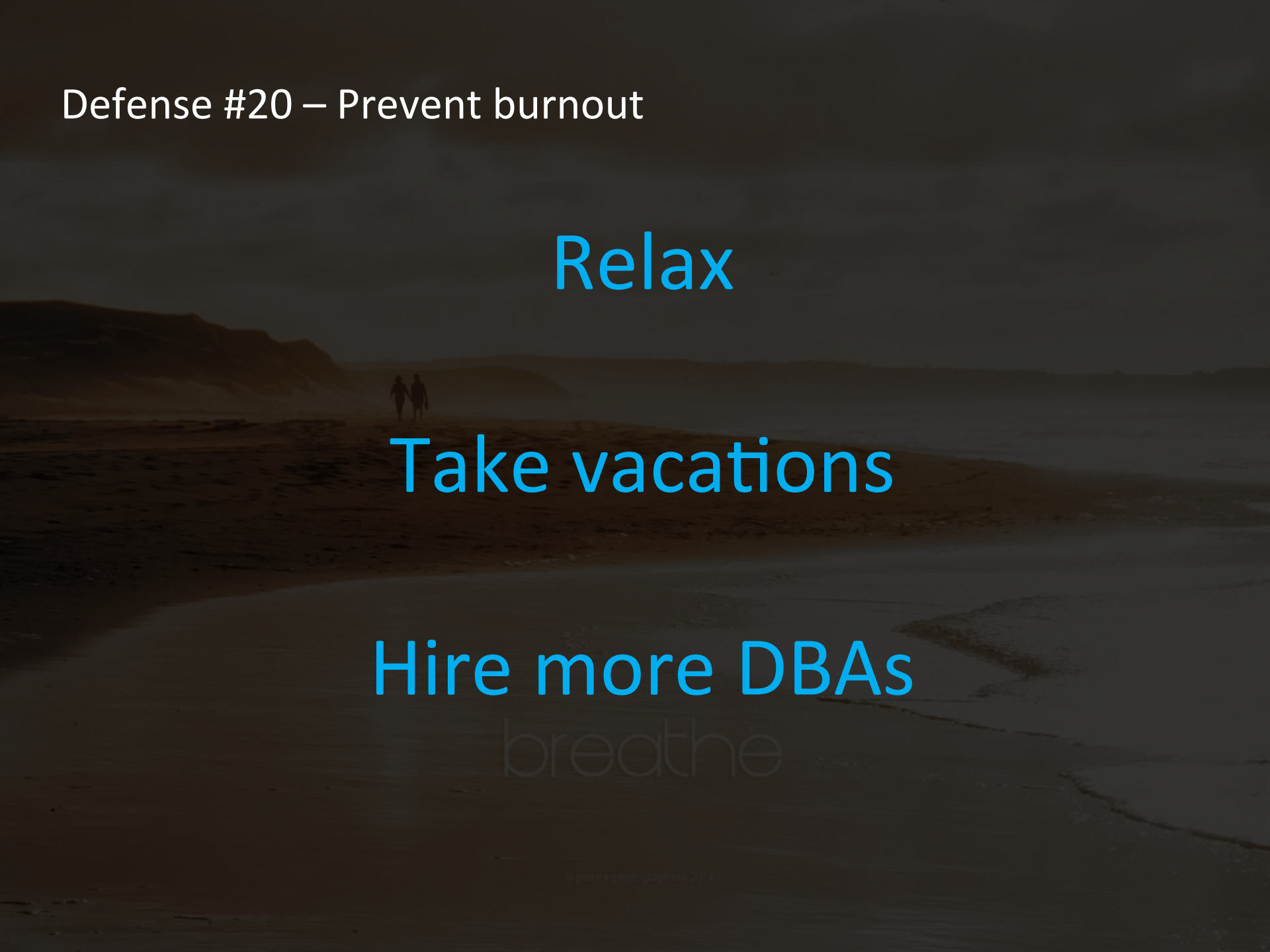
Defense #19 – Backups



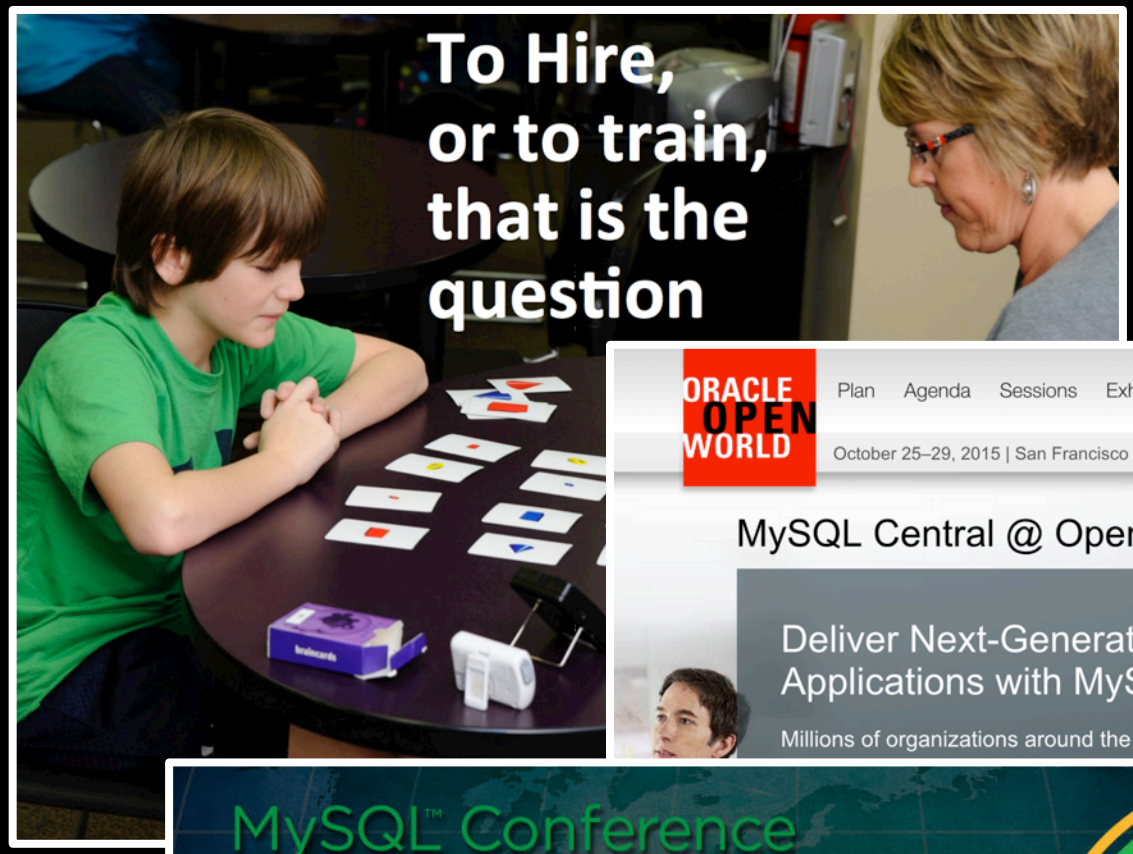
Defense #19 – Backups

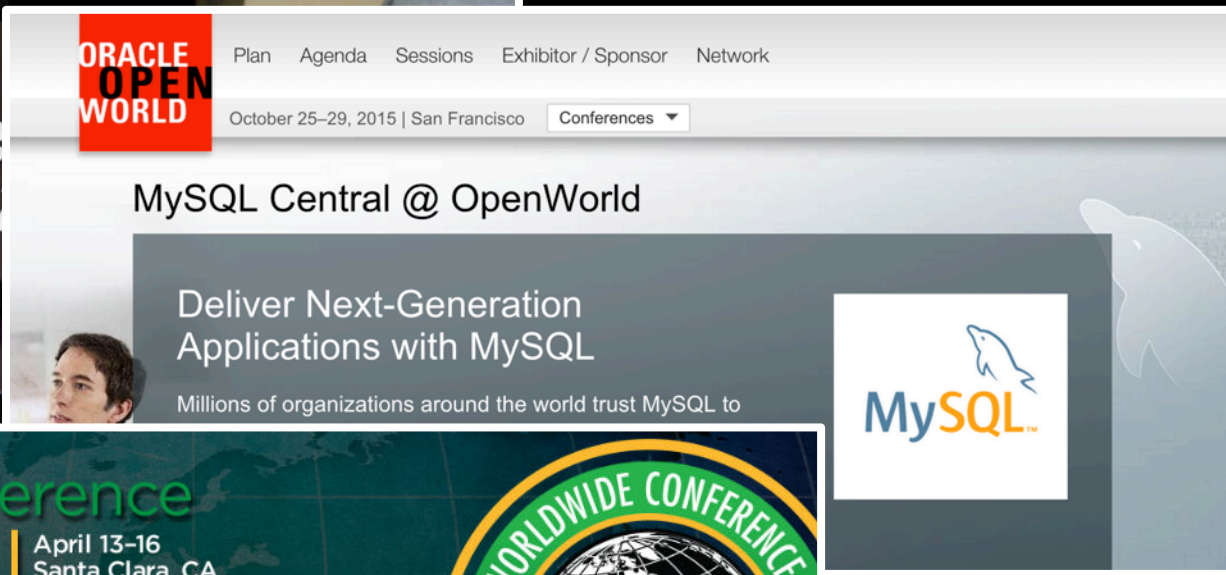



breathe
© patrice-photographiste 2014


Defense #20 – Prevent burnout
Relax
Take vacations
Hire more DBAs
breathe

Defense #20 – Prevent burnout





Defense #20 – Prevent burnout

KISSmetrics
A BLOG ABOUT ANALYTICS, MARKETING AND TESTING

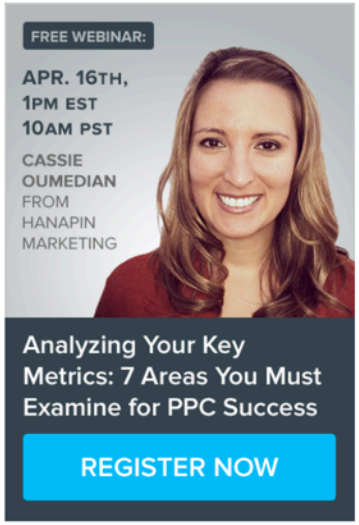
Infographics | Marketing Guides | Webinars

FREE EMAIL UPDATES
Get the latest content first.

First Name

Your Email

Join Us



How to Prevent Employee Burnout

66 Like | 555 Tweet | 40 +1 | 173 Share

We all know what burnout looks like.

It's the stressed-out manager who proclaims they've **"had it"** and books the next flight to Cancun.

The start-up co-founder **working 130 hour weeks** who claims he **"can't take all this bullsh*t anymore"** and goes on a week-long bender in Miami.

The boss who **yells with such intensity** you fear he might have a heart attack.

It is so common that everyone in any career – engineers, sales managers, tech support personnel, etc. – is susceptible to burnout. Despite being so common, many managers aren't aware of why burnout happens or how to keep it from happening. But they need to know.

Being able to understand burnout, its causes, and how to prevent it is essential in order to maintain a positive environment and keep the best talent on the team.

Burnout Defined

> *Burnout is an individual's response to chronic emotional and interpersonal stressors within the workplace (Maslach et al., 2001).*

Defenses

- Defense #1 – Query Comments
- Defense #2 – Alerting on the Database
- Defense #3 – Dashboards
- Defense #4 – User and Table Statistics
- Defense #5 – Box Raingauge
- Defense #6 – Monitor the Queries
- Defense #7 – Sharding
- Defense #8 – Innotop
- Defense #9 – Query Killer
- Defense #10 – Transaction Monitor (and killer)
- Defense #11 – Dynamic Query Killer Thresholds
- Defense #12 – Healthchecking
- Defense #13 – Data Layer as a Service
- Defense #14 – Lock Monitor (and killer)
- Defense #15 – strace Errant Connections
- Defense #16 – ~~pt-pmp~~ php-pmp
- Defense #17 – Multivariable pt-stalk
- Defense #18 – Automated User Management
- Defense #19 - Backups
- Defense #20 – Prevent Burnout

COMMUNITY NETWORKING RECEPTION (IN EXPO HALL), CO-SPONSORED BY: YELP, BOX AND GITHUB - MYSQL COMMUNITY AWARDS & LIGHTNING TALKS

▾ Schedule info

Status: Accepted
Time Slot:
15 April 5:30PM - 7:00PM
Room:
None

#DBHANGOPS IN REAL LIFE

14 April 6:00PM - 7:00PM @ Room 203

Experience level: -
Duration: Birds of a Feather

A live, in-person meeting with some of the participants of bi-weekly virtual meetup to simply talk about anything database-related and solutions to common problems. Also be sure to check out http://dbhangops.github.io for previous recordings and discussions!

box

Come to the **#DBHangOps BoF** @ **6pm** in **room 203**!

Email geoff@box.com

Twitter @geodbz

Tech Blog www.box.com/blog/engineering

Github github.com/box

Links of Supreme Interest!

- https://www.percona.com/live/mysql-conference-2014/sessions/box-weather-station-open-source-tools-performance-and-forensic-monitoring
- https://www.percona.com/live/mysql-conference-2014/sessions/mysql-devops-outbrain
- http://www.percona.com/live/mysql-conference-2012/sessions/etsy-shard-architecture-starts-s-and-ends-hard
- http://www.infoq.com/presentations/box-mysql-sharding

Image Credit / Source

- http://ak-hdl.buzzfed.com/static/2014-06/19/12/enhanced/webdr06/anigif_enhanced-26665-1403194377-3.gif
- http://31.media.tumblr.com/5f6ffc7bf7f456ba0442596d07faa925/tumblr_n4zmymPU191s0teago2_r1_400.gif
- https://www.flickr.com/photos/patrick_verstappen/8421238929
- https://www.flickr.com/photos/apocalust/5092520561
- http://38.media.tumblr.com/e1a7a9881573597bbe97ade824f578a3/tumblr_ngcsl73nLt1u4i1dio4_1280.gif
- http://stream1.gifsoup.com/view3/4891599/lion-king-stampede-o.gif
- http://24.media.tumblr.com/tumblr_lv5jvwWTyT1r4hwmoo1_500.gif
- http://cl.jroo.me/z3/2/R/v/e/a.baa-the-hacker-cat.jpg
- https://www.flickr.com/photos/daryl_mitchell/1199598508
- https://www.flickr.com/photos/patrice-photographiste/14909768574