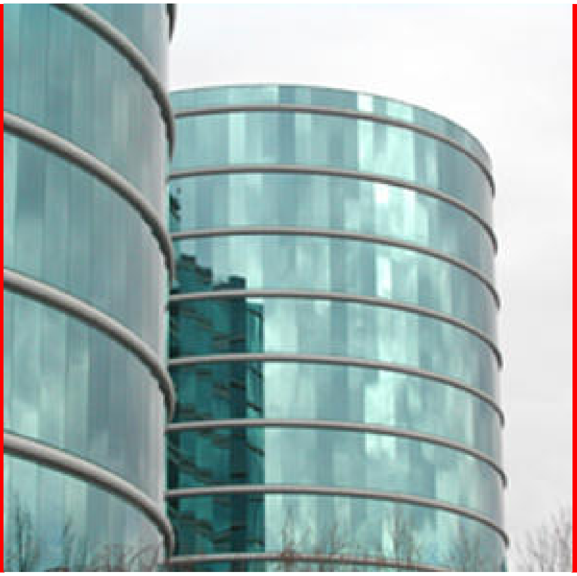


MySQL Utilities
Time Saving Scripts for the DBA

v2.0

Topics

- What is MySQL Utilities?
- Use cases for each utility
- How to get MySQL Utilities
- Architecture of MySQL Utilities
- Examples of usage
- Using the MySQL Utilities Python library
    - How to group utilities for even more usability
    - How to modify utilities for your needs
    - How to develop and contribute new utilities

What is MySQL Utilities?

- A collection of Python utilities for managing MySQL databases
- Latest version is release-1.0.5
- Part of MySQL Workbench 5.2.31+
  - Current version 5.2.39
- Available under the GPLv2 license
- Written in Python
- Easily enhanced using a growing code library
- Goal is to provide a Python library to grow solutions for common administrative problems

List of Utilities

- Database Operations
  - mysqldbcompare – compare databases
  - mysqldbcopy – copy databases between servers
  - mysqldbexport – export metadata and data
  - mysqldbimport – import metadata and data
  - mysqldiff – compare object definitions
- General Utilities
  - mysqldiskusage – show disk usage for databases
  - mysqlindexcheck – check for redundant indexes
  - mysqlmetagrep – search metadata
  - mysqlprocgrep – search process information
  - mysqluserclone – clone a user account

List of Utilities

- High Availability
  - mysqlfailover – automatic failover for MySQL 5.6.5+
  - mysqlreplicate – setup replication
  - mysqlrpladmin – general replication administration
    - switchover
    - failover for MySQL 5.6.5
  - mysqlrplcheck – check replication configuration
  - mysqlrplshow – display map of replication topology
- Server Operations
  - mysqlserverclone – start a scratch server
  - mysqlserverinfo – show server information

How to get Utilities

- Available on Launchpad
  - https://launchpad.net/mysql-utilities
  - bzr branch lp:mysql-utilities
  - Requires Connector/Python
    - https://launchpad.net/myconnpy
    - bzr branch lp:myconnpy
- Available as a plugin in MySQL Workbench
  - http://www.mysql.com/downloads/workbench/
- Documentation is here:
  - http://http://dev.mysql.com/doc/workbench/en/mysql-utilities.html

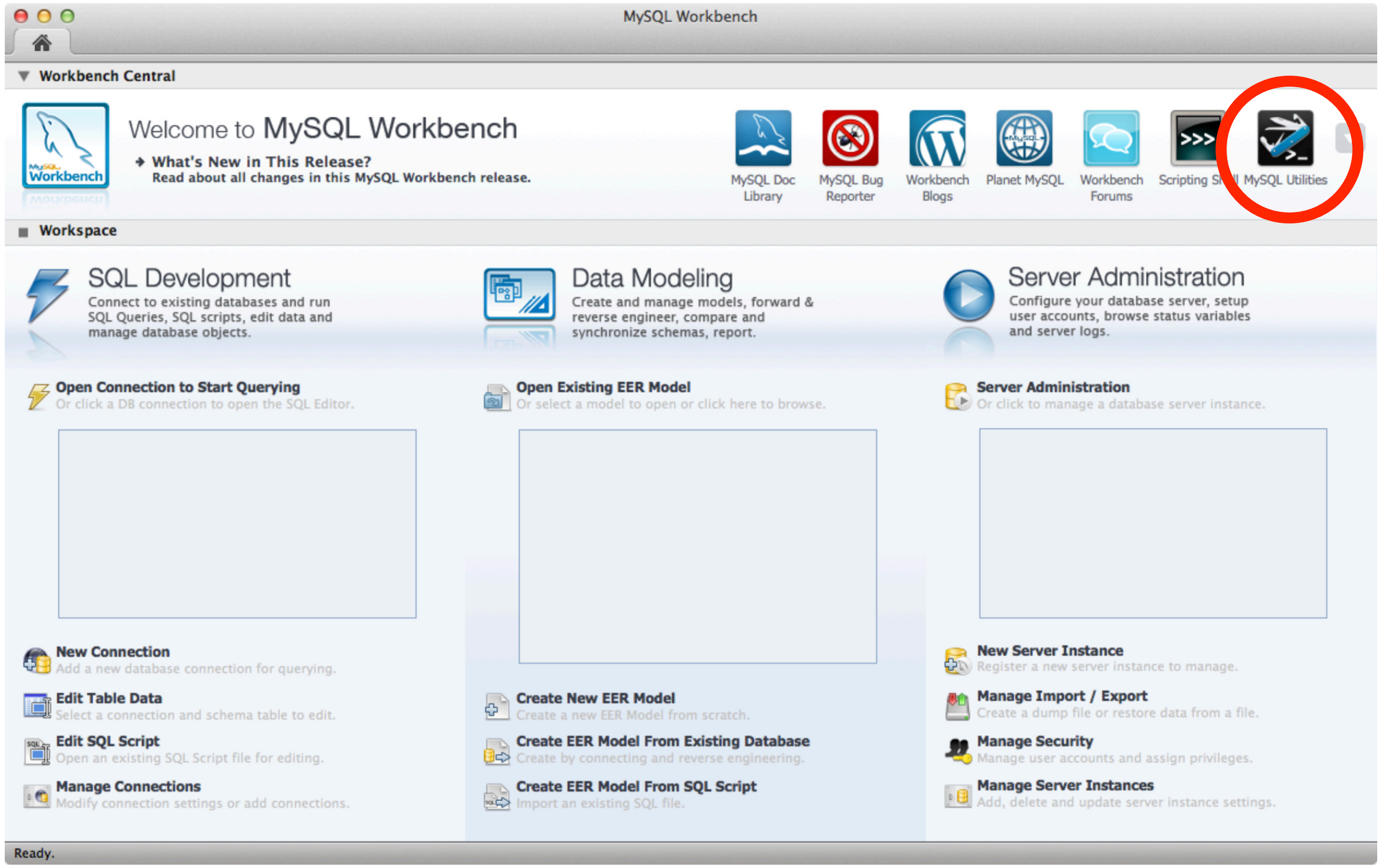
Utilities in MySQL Workbench

- Launch the MySQL Utilities command window:
  - Click on the Plugins menu item
  - Select “Start Shell for MySQL Utilities”

OR

- From the Workbench main window:
  - Click on the drop down arrow icon to the right of the main window
  - Scroll through screens to find the MySQL Utilities icon

How do I access the utilities?



MySQL Workbench

Workbench Central

Welcome to MySQL Workbench

What's New in This Release?
Read about all changes in this MySQL Workbench release.

Workspace

SQL Development
Connect to existing databases and run SQL Queries, SQL scripts, edit data and manage database objects.

Open Connection to Start Querying
Or click a DB connection to open the SQL Editor.

New Connection
Add a new database connection for querying.

Edit Table Data
Select a connection and schema table to edit.

Edit SQL Script
Open an existing SQL Script file for editing.

Manage Connections
Modify connection settings or add connections.

Data Modeling
Create and manage models, forward & reverse engineer, compare and synchronize schemas, report.

Open Existing EER Model
Or select a model to open or click here to browse.

Create New EER Model
Create a new EER Model from scratch.

Create EER Model From Existing Database
Create by connecting and reverse engineering.

Create EER Model From SQL Script
Import an existing SQL file.

Server Administration
Configure your database server, setup user accounts, browse status variables and server logs.

Server Administration
Or click to manage a database server instance.

New Server Instance
Register a new server instance to manage.

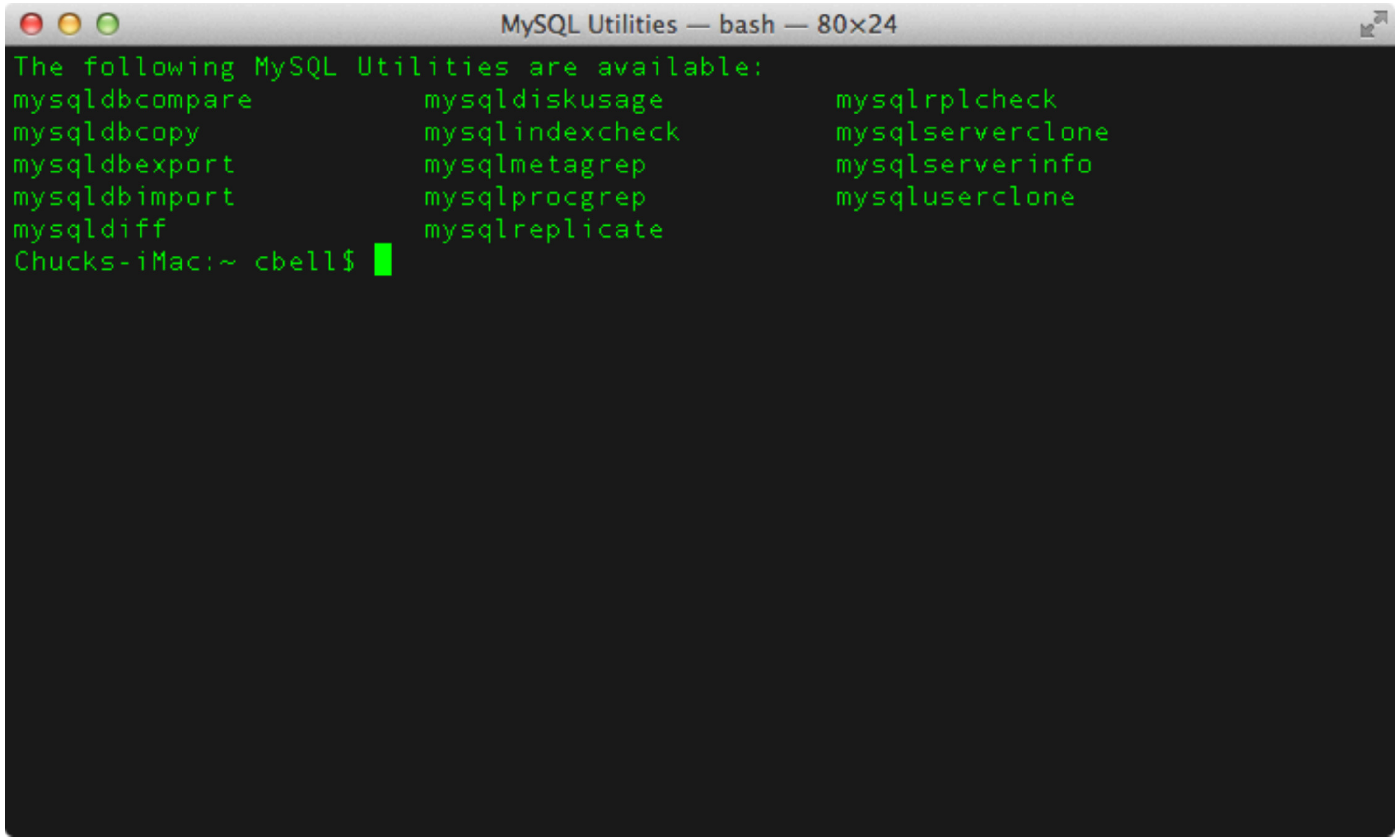
Manage Import / Export
Create a dump file or restore data from a file.

Manage Security
Manage user accounts and assign privileges.

Manage Server Instances
Add, delete and update server instance settings.

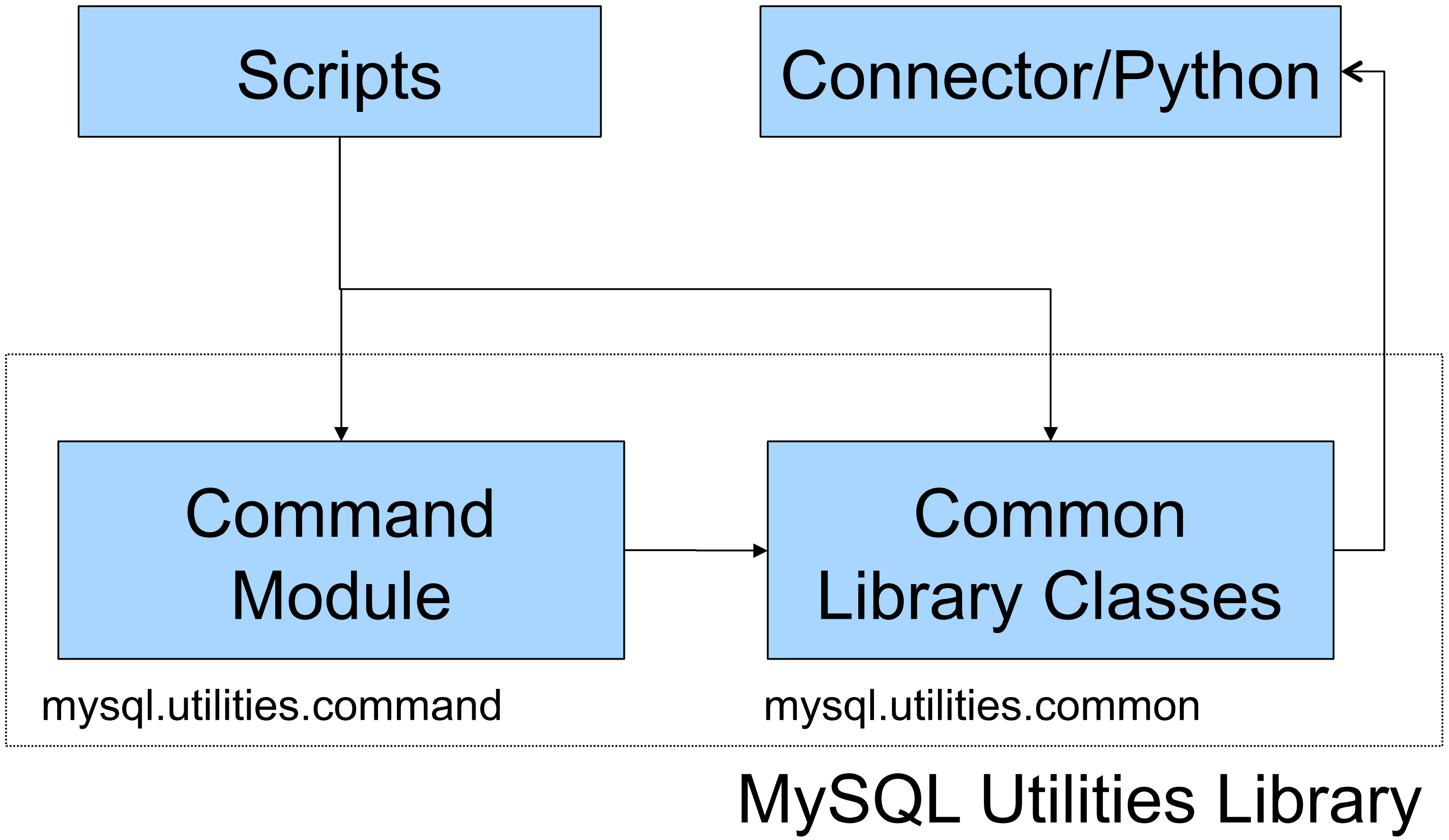
Ready.

How do I access the utilities?



```
The following MySQL Utilities are available:
mysqldbcompare          mysqldiskusage          mysqlrplcheck
mysqldbcopy             mysqlindexcheck         mysqlserverclone
mysqldbexport           mysqlmetagrep           mysqlserverinfo
mysqldbimport           mysqlprocgrep           mysqluserclone
mysqldiff               mysqlreplicate
Chucks-iMac:~ cbell$ 
```

Architecture of the MySQL Utilities



mysqldbcompare – compare databases

- Find missing objects from either database
- Find objects that differ in definition
- Find differences in data among tables
- Print difference in formats differ, ndiff, or context
- Print output rows in SQL, GRID, TAB, CSV, or VERTICAL formats
- SQL output produces transformation statements for synchronizing objects and data
- Scenarios
  - checking master and slave for consistency
  - checking production and development databases for consistency
  - generating a difference report for expected differences among new and old data
  - comparing backups for differences

Sample execution

```
$ mysqldbcompare --server1=root@localhost --server2=root@backup_host:3310 \
> inventory1:inventory2 --run-all-tests
# server1 on localhost: ... connected.
# server2 on localhost: ... connected.
# Checking databases inventory1 on server1 and inventory2 on server2

WARNING: Objects in server1:inventory but not in server2:inventory:
        VIEW: finishing_up
        VIEW: cleaning
                                                    Defn    Row     Data
Type      Object Name                               Diff    Count   Check
---------------------------------------------------------------------------
TABLE     supplier                                  pass    FAIL    FAIL

Row counts are not the same among inventory1.supplier and inventory2.supplier.

Data differences found among rows:
--- inventory1.supplier
+++ inventory1.supplier
@@ -1,2 +1,2 @@
 code,name
-2,Never Enough Inc.
+2,Wesayso Corporation
```

<user>:<password>@<host>:<port>:<socket>

mysqlmetagrep – search objects

- Search for objects with names matching a pattern
- Match using SQL patterns or POSIX regular expressions
- Search bodies of routines (procedures, events, triggers)
- Generate SQL for executing the query
  - Can be used in applications
  - Can be stored in events or views

Searching for objects by name

Searching a database for objects starting with “t”

mysqlmetagrep --pattern="t_" --server=mats@localhost

```
$ mysqlmetagrep --pattern="t_" --server=mats@localhost
+------------------------+-------------+-------------+----------+-------------+---------+
| Connection             | Object Type | Object Name | Database | Field Type  | Matches |
+------------------------+-------------+-------------+----------+-------------+---------+
| mats:*@localhost:3307  | TABLE       | t1          | test     | COLUMN      | th      |
| mats:*@localhost:3307  | TABLE       | t1          | test     | TABLE       | t1      |
| mats:*@localhost:3307  | TABLE       | t2          | test     | TABLE       | t2      |
| mats:*@localhost:3307  | TABLE       | tt5         | test     | COLUMN      | t2,t1   |
+------------------------+-------------+-------------+----------+-------------+---------+
```

Searching routine bodies

Searching a database for objects containing “l_host”

mysqlmetagrep --body --pattern="%l_host%" \ --server=mats@localhost

```
$ mysqlmetagrep --body --pattern="%l_host%" --server root@localhost:3307
+------------------------+--------------+----------------+-----------+-------------+----------------+
| Connection             | Object Type  | Object Name    | Database  | Field Type  | Matches        |
+------------------------+--------------+----------------+-----------+-------------+----------------+
| root:*@localhost:3307  | PROCEDURE    | switch_master  | test      | ROUTINE     | switch_master  |
+------------------------+--------------+----------------+-----------+-------------+----------------+
```

mysqlprocgrep – search processes

- Search processes on multiple machines
- Match by PROCESSLIST fields
  - Id, State, User, Host, Database, Command, State, Info
- Match by age
  - Find long-running queries, or idle connections
- Get SQL for performing the query or action
  - Put in application
  - Put in events
- Kill queries or connections
  - Option: --kill-query
  - Option: --kill-connection

Sample usage

- Find queries by 'www-data' that have been executing for more than 20 minutes

```
mysqlprocgrep --server=mats@example.com
    --match-user=www-data
    --match-state=executing
    --age=20m
```

```
+-------------------------+-----+-------+-------------+-------+---------+-------+-----------+--------------------+
| Connection              | Id  | User  | Host        | Db    | Command | Time  | State     | Info               |
+-------------------------+-----+-------+-------------+-------+---------+-------+-----------+--------------------+
| mats:*@example.com:3306 | 53  | mats  | example.com | user  | Query   | 2040  | executing | select ...         |
+-------------------------+-----+-------+-------------+-------+---------+-------+-----------+--------------------+
```

Sample usage

- ~~Find queries by 'www-data' that have been executing for more than 20 minutes~~

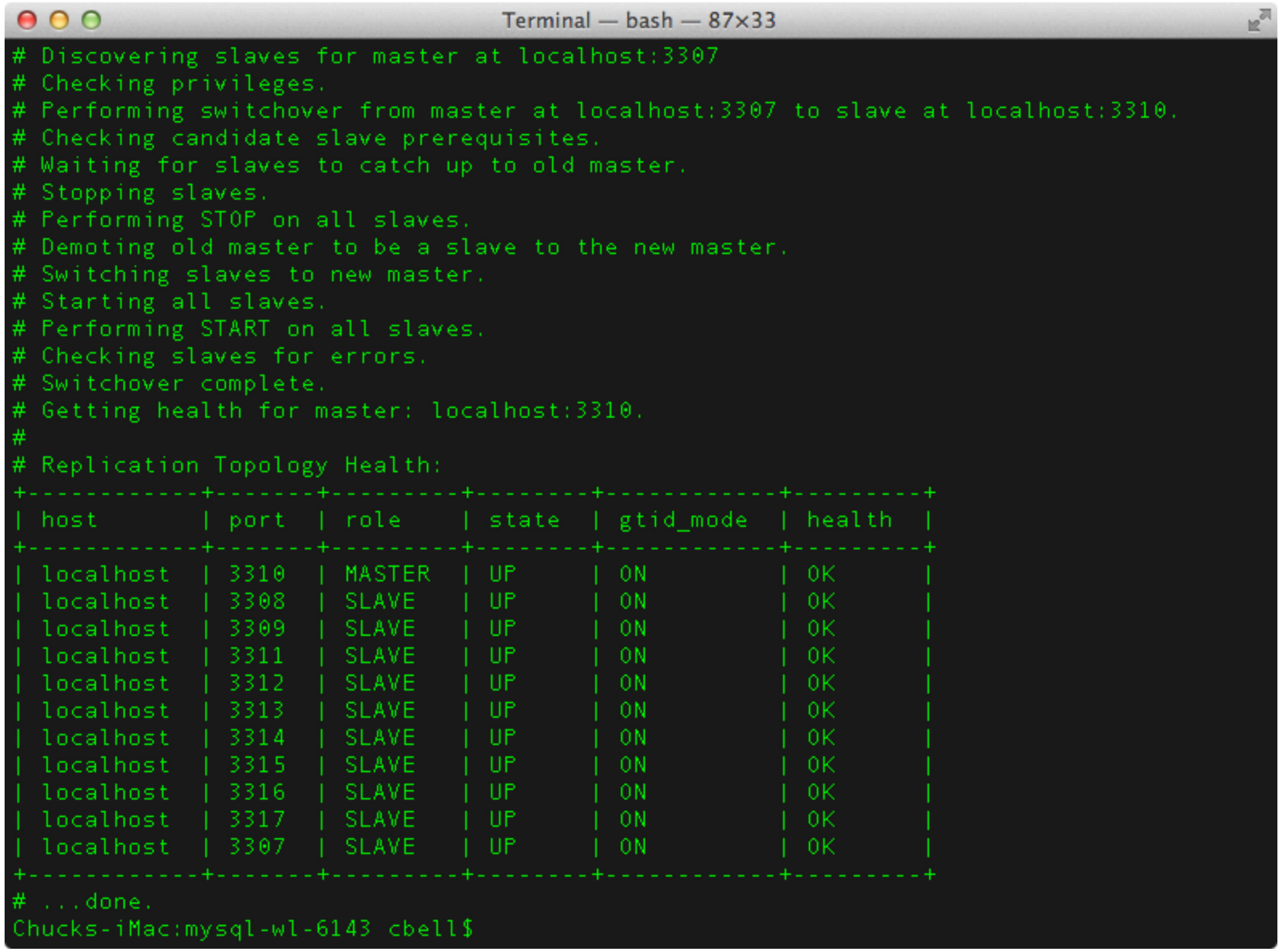
```
mysqlprocgrep --server=mats@example.com
    --match-user=www-data
    --match-state=executing
    --age=20m --kill-query
```

Kill Them!

mysqlrpladmin – replication administration

- **elect** - (GTIDs) Perform best slave election and report best slave to use in the event a switchover or failover is required.
- **failover** - (GTIDs) Conduct failover to the ‘best’ slave. The command will test each candidate slave listed for the prerequisites. Once a candidate slave is elected, it is made a slave of each of the other slaves thereby collecting any transactions executed on other slaves but not the candidate. In this way, the candidate becomes the most up-to-date slave and therefore a replacement for the master.
- **gtid** - (GTIDs) Displays the contents of the GTID variables used to report GTIDs in replication.
- **health** - Display the replication health of the topology.
- **reset** - Execute the STOP SLAVE and RESET SLAVE commands on slaves.
- **start** - Execute the START SLAVE command on all slaves.
- **stop** - Execute the STOP SLAVE command on all slaves.
- **switchover** - Perform slave promotion to a specified candidate slave as designated by the --new-master option. This command supports both gtid-enabled servers and non-gtid-enabled scenarios.

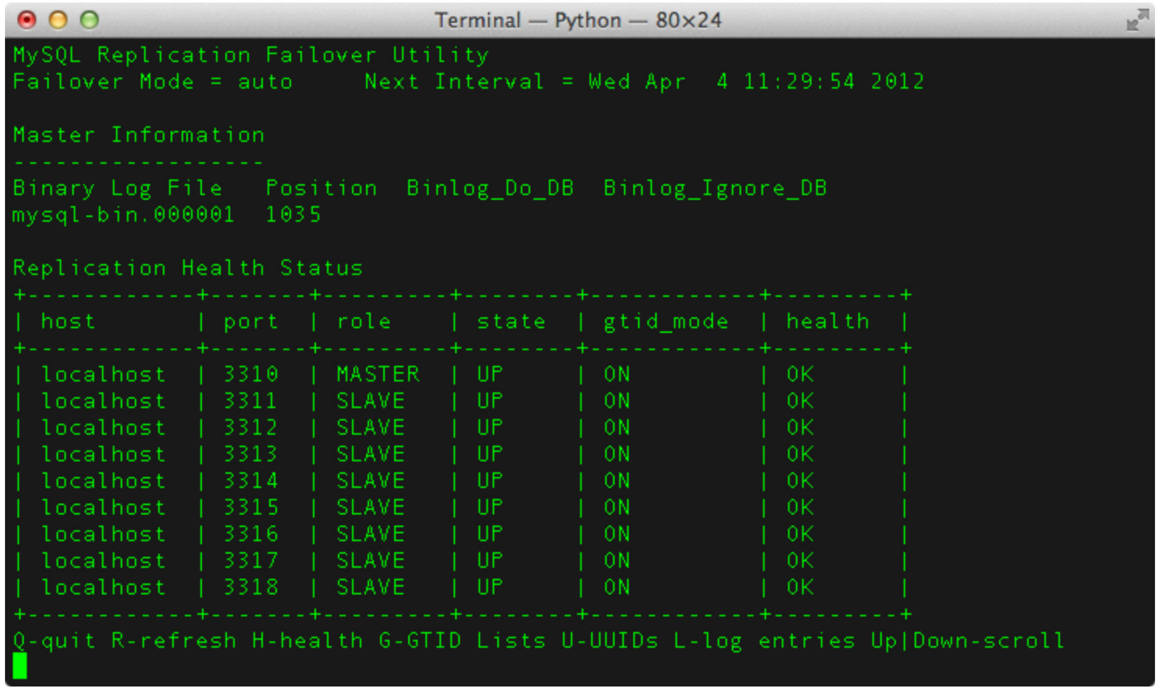
Sample output - switchover



```
# Discovering slaves for master at localhost:3307
# Checking privileges.
# Performing switchover from master at localhost:3307 to slave at localhost:3310.
# Checking candidate slave prerequisites.
# Waiting for slaves to catch up to old master.
# Stopping slaves.
# Performing STOP on all slaves.
# Demoting old master to be a slave to the new master.
# Switching slaves to new master.
# Starting all slaves.
# Performing START on all slaves.
# Checking slaves for errors.
# Switchover complete.
# Getting health for master: localhost:3310.
#
# Replication Topology Health:
+------------+-------+---------+--------+------------+---------+
| host       | port  | role    | state  | gtid_mode  | health  |
+------------+-------+---------+--------+------------+---------+
| localhost  | 3310  | MASTER  | UP     | ON         | OK      |
| localhost  | 3308  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3309  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3311  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3312  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3313  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3314  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3315  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3316  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3317  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3307  | SLAVE   | UP     | ON         | OK      |
+------------+-------+---------+--------+------------+---------+
# ...done.
Chucks-iMac:mysql-wl-6143 cbell$
```

mysqlfailover

- Automatic failover for global transaction identifier-enabled servers. MySQL v5.6.5+
- Replication health reporting
- New in release-1.0.5 (WB 5.2.39)

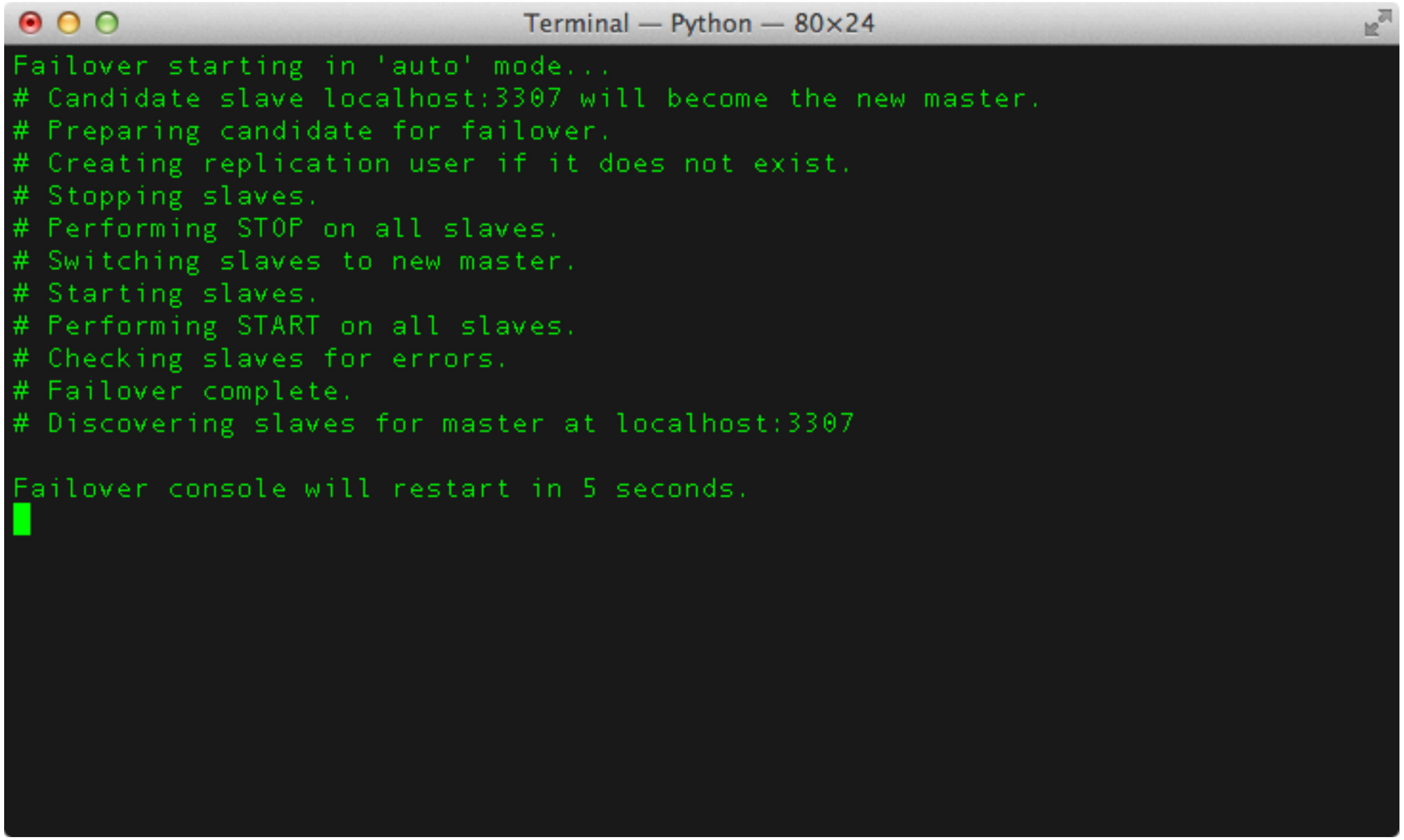


```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Wed Apr  4 11:29:54 2012

Master Information
------------------
Binary Log File   Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql-bin.000001  1035

Replication Health Status
+------------+-------+---------+--------+------------+---------+
| host       | port  | role    | state  | gtid_mode  | health  |
+------------+-------+---------+--------+------------+---------+
| localhost  | 3310  | MASTER  | UP     | ON         | OK      |
| localhost  | 3311  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3312  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3313  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3314  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3315  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3316  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3317  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3318  | SLAVE   | UP     | ON         | OK      |
+------------+-------+---------+--------+------------+---------+
Q-quit R-refresh H-health G-GTID Lists U-UUIDs L-log entries Up|Down-scroll
```

mysqlfailover

- Failover Modes
  - **Auto** – automatic failover to candidate list first then available slaves
  - **Elect** – failover to candidate list only
  - **Fail** – stop if failover event detected
- Extension Points
  - **exec-fail-check** - execute a script to determine if failover is needed for application-specific detection for failover.
  - **exec-before** - execute a script before failover is performed.
  - **exec-after** - execute a script immediately after failover to a new master.
  - **exec-post-fail** - execute a script after failover is complete and all slaves have been attached to the new master.

Sample usage - failover



```
Failover starting in 'auto' mode...
# Candidate slave localhost:3307 will become the new master.
# Preparing candidate for failover.
# Creating replication user if it does not exist.
# Stopping slaves.
# Performing STOP on all slaves.
# Switching slaves to new master.
# Starting slaves.
# Performing START on all slaves.
# Checking slaves for errors.
# Failover complete.
# Discovering slaves for master at localhost:3307

Failover console will restart in 5 seconds.
```