Script it!

Make professional DBA tools out of nothing

Giuseppe Maxia

Continuent, Inc

Why scripting?

Why not using Puppet or Chef?

- Not all companies use it
- Some tasks don't fit the pattern
- If you want to do testing on-the-fly, knowing how to script will save time

Why shell scripts?

Why not {Perl|Python|Ruby|PHP|Lua} ?

- **Shell is available everywhere**
  - (unless you run Windows, but in this case you are in the wrong room)
- **Shell scripting is powerful**
- **It is fast to write, and reasonably easy to read**

Define the objectives of your scripts

If you know where you want to go, you can easily list what you need to do to get there

- **We want:**
  - Install a master
  - Install and provision one or more slaves
  - Check if replication is working

Find implementation requirements

For each goal, find what data you need

- **Install a master**
  - Choose a host
  - Find a data directory with enough storage
  - Define user names and passwords
  - Define port, socket, and log directory
- **Install a slave**
  - Choose the hosts
  - Assume same requirements as the master
- **Check replication**
  - Define which values you need from the database and from the OS

Install clean

The installation should be reversible

- **Objectives:**
  - Do not confuse the installation scripts with their target;
  - Do not mix installation scripts and the product of the installation;
  - Allow a clean removal of what you installed;

CODE FOR THIS SESSION

The code is committed to a public repository

- https://code.google.com/p/tungsten-toolbox
- Go to "source"
- Browse the code
- Look for "deploy samples"

- You may also need MySQL Sandbox
  - (http://mysqlsandbox.net)

/trunk - tungsten-toolbox - Open source tools for database replication and clustering - Google Project Hosting

https code.google.com/p/tungsten-toolbox/source/browse/trunk#trunk%2Fdeploy_samples

Reader

My favorites ▼ | Sign in

tungsten-toolbox

Open source tools for database replication and clustering

Search projects

Project Home | Downloads | Wiki | Issues | **Source**

Checkout **Browse** Changes

Source path: svn/ trunk

‹r151 **r152**

- trunk
  - cookbook
  - deploy_samples
    - v2
  - ldg
  - simple_services
  - tungsten-progress
  - tungsten-sandbox

Filename	Size	Rev	Date	Author
CONFIG_default.sh	621 bytes	r152	Today (19 minutes ago)	g.maxia
CONFIG_sb_5_1_69.sh	604 bytes	r152	Today (19 minutes ago)	g.maxia
CONFIG_sb_5_5_31.sh	603 bytes	r152	Today (19 minutes ago)	g.maxia
CONFIG_sb_5_6_11.sh	603 bytes	r152	Today (19 minutes ago)	g.maxia
README	94 bytes	r152	Today (19 minutes ago)	g.maxia
check_replication.sh	3.1 KB	r152	Today (19 minutes ago)	g.maxia
install_replication.sh	2.9 KB	r152	Today (19 minutes ago)	g.maxia
provision_slaves.sh	679 bytes	r152	Today (19 minutes ago)	g.maxia
provision_slaves_parallel.sh	788 bytes	r152	Today (19 minutes ago)	g.maxia
remove_replication.sh	630 bytes	r152	Today (19 minutes ago)	g.maxia
stop_replication.sh	401 bytes	r152	Today (19 minutes ago)	g.maxia
test_replication.sh	930 bytes	r152	Today (19 minutes ago)	g.maxia

Terms - Privacy - Project Hosting Help

Powered by Google Project Hosting

Tips - 1 - use a config file

Make your scripts easily changeable

- Do not write host names, ports, paths, user names and passwords in scripts
- Use a configuration file
- Load that file before each script
- Make different config files for each scenario, and link to it

Tips - 2 - Use a remote messenger

Simplify remote commands

- **Do not write a series of "ssh ..." commands**
- **Getting the return values correctly and quotes is messy**
- **instead:**

1. create a script with the instructions for a given node
2. transfer the file to the node
3. run the file remotely

Tips - 3 - Write a banner

Help your memory, avoid conflicts

- **When the installation task is finished, write a file containing**
  - what you have installed
  - where dis you install from
- **Check for that file, to avoid overwriting an installation**
- **Remove the file when uninstalling**

Tips - 4 - Write customized shortcuts

Type once, type less

- **Your installation may depend on variable components (e.g. the path of your tools)**
- **As part of the installation, write a shortcut file**
- **example:**

```
$ cat client
#!/bin/bash
$HOME/opt/mysql/5.5.31/bin/mysql $@
```

Tips - 5 - Run operations in parallel

Keep them all busy

- **If your operation takes long, and you have to operate it in many servers:**

1. Start all of them at once, in background
2. Send the output to a different file for each task
3. wait for the tasks to finish
4. display the contents of the output files

Tips - 6 - Use convenience arrays

If they look alike, put them together

- **Use arrays to group together items that you want to use:**
- **e.g.:**
  - slaves=($NODE2 $NODE3)
  - all_nodes=($NODE1 $NODE2 $NODE3)
  - tools=(mysql mysqldump mysqladmin)
- **use loops**
  - for NODE in ${all_nodes[*]} ; do .... ; done

Getting started : the config file

Remember your goals, and put the data together

- **Should contain:**
  - list of your nodes
  - database path
  - software path
  - user names
  - convenience arrays (see tip #6)

Default Config file (1)

```
$ cat CONFIG_default.sh
#!/bin/bash
export NODE1=host1
export NODE2=host2
export NODE3=host3
#export NODE4=host4

## ====================
export BANNER=$HOME/current_replication.txt
export DB_PORT=3306
export DATADIR=/var/lib/mysql
export BASEDIR=/usr
...
```

Default Config file (2)

```
...
export DB_USER=powerful
export DB_PASSWORD=can_do_all
export REPL_USER=slave_user
export REPL_PASSWORD=can_do_little

## ====================
export ALL_NODES=($NODE1 $NODE2 $NODE3 $NODE4)
export MASTERS=($NODE1)
export SLAVES=($NODE2 $NODE3 $NODE4)
export
DASH_LINE='------------------------------------------------
----------------'
```

Customized Config file (1)

```
$ cat CONFIG_sb_5_5_31.sh
#!/bin/bash
...
export DB_PORT=15531
export SBDIR=$HOME/sandboxes/mysql_5_5_31
export DATADIR=$SBDIR/data
export BASEDIR=$HOME/opt/mysql/5.5.31
export DB_USER=msandbox
export DB_PASSWORD=msandbox
export REPL_USER=rsandbox
export REPL_PASSWORD=rsandbox
...
```

Use only one:

```
$ ln -s CONFIG_default.sh CONFIG.sh

$ ls -l CONFIG*
-rwxr-xr-x 1 x.x CONFIG_default.sh
-rwxr-xr-x 1 x.x CONFIG_sb_5_1_69.sh
-rwxr-xr-x 1 x.x CONFIG_sb_5_5_31.sh
-rwxr-xr-x 1 x.x CONFIG_sb_5_6_11.sh
lrwxrwxrwx 1 x.x CONFIG.sh -> CONFIG_sb_5_5_31.sh
```

Check and load for your config in every script

This should always be the first operation

```
$ head install_replication.sh
#!/bin/bash

if [ ! -f ./CONFIG.sh ]
then
    echo "Configuration file CONFIG.sh not found"
    exit 1
fi

. ./CONFIG.sh
```

Code defensively

Make sure you can do what you want to do

```
for NODE in ${MASTERS[*]}
do
RUNNING=$($MYSQL_SLAVE -BN --host=$NODE -e 'select 1')
    if [ -z "$RUNNING" ]
    then
        echo "# WARNING:"
        echo " mysql not reachable "
        echo " by user $REPL_USER in node $NODE"
        exit 1
    fi
done
```

Get precise values (1)

Filter and cut

```
$ mysql -e 'show master status\G'
*************** 1. row ***************
             File: mysql-bin.000001
         Position: 107
     Binlog_Do_DB:
Binlog_Ignore_DB:

$ mysql -e 'show master status\G' | grep File
             File: mysql-bin.000001

$ mysql -e 'show master status\G' | grep File |awk
'{print $2}'
mysql-bin.000001
```

Get precise values (2)

Filter and cut

```
$ mysql -e 'show master status\G'
*************************** 1. row ***************************
             File: mysql-bin.000001
         Position: 107
     Binlog_Do_DB:
Binlog_Ignore_DB:

$ mysql -e 'show master status\G' | grep Position
         Position: 107

$ mysql -e 'show master status\G' | grep Position |awk
'{print $2}'
107
```

Get precise values (3)

Filter and cut

```
$ BINLOG_POS=$(mysql -e 'show master status\G' | grep
Position |awk '{print $2}')

$ BINLOG_FILE=$(mysql -e 'show master status\G' | grep
File |awk '{print $2}')

$ echo $BINLOG_FILE
mysql-bin.000001

$ echo $BINLOG_POS
107
```

Set replication

Using the precise values, set the slave status

```
    for NODE in ${SLAVES[*]}
    do
        $MYSQL --host=$NODE -e 'stop slave'

        $MYSQL --host=$NODE -e "CHANGE MASTER TO
master_host='$MASTER', master_port=$DB_PORT,
master_user='$REPL_USER',
master_password='$REPL_PASSWORD',
master_log_file='$BINLOG_FILE', master_log_pos=
$BINLOG_POS "

        $MYSQL --host=$NODE -e 'start slave'
    done
```

Create and use shortcuts

Save yourself some typing

```
# write utility scripts
for script in mysql mysqldump mysqladmin
do
    echo '#!/bin/bash' > $script
    echo "$BASEDIR/bin/$script --user=$DB_USER --port=
$DB_PORT \$@" >> $script
    chmod +x $script
done

######
$ ./mysql -p
```

sample installation run

```
$ ./install_replication.sh
# Making sure MySQL is running
# MASTER host1: ok
# Node host1: ok
# Node host2: ok
# Node host3: ok
# node host1 - server_id: 10
# node host1 - log_bin: 1
# node host2 - server_id: 20
# node host2 - log_bin: 1
# node host3 - server_id: 30
# node host3 - log_bin: 1
```

Check replication

Improved from the one published in
http://datacharmer/blogspot.com

- read the master status
- read the slave status
- compare values
- scream if they don't match

sample check run

```
SLAVE host2 : Replication OK
file: mysql-bin.000001 at 106
SLAVE host3 : Replication OK
file: mysql-bin.000001 at 106
SLAVE host4 : Replication OK
file: mysql-bin.000001 at 106
```

Run a test

- Create a table in the master
- insert some records
- Wait 1 second
- Count the records in all servers

Sample test run

```
$ ./test_replication.sh
node host1 - 0
node host2 - 0
node host3 - 0
node host4 - 0
node host1 - 3
node host2 - 3
node host3 - 3
node host4 - 3
```

Remove replication

```
MYSQL="$MYSQL --user=$DB_USER --password=$DB_PASSWORD
--port=$DB_PORT"

for NODE in ${SLAVES[*]}
do
    $MYSQL -BN --host=$NODE -e 'stop slave'
    $MYSQL -BN --host=$NODE -e 'reset slave'
done

### Remove banners and shortcuts
```

Install using a different version

This method works fine from MySQL 5.0 up to 5.5

```
$ rm CONFIG.sh

$ ln -s CONFIG_sb_5_5_31.sh CONFIG.sh

$ ./install_replication.sh
```

Install using a MySQL 5.6

Looks like it's the same, but ...

```
./install_replication.sh
# Making sure MySQL is running
Warning: Using a password on the command line
interface can be insecure.
# MASTER host1: ok
Warning: Using a password on the command line
interface can be insecure.
# Node host1: ok
Warning: Using a password on the command line
interface can be insecure.
# Node host2: ok
Warning: Using a password on the command line
interface can be insecure.
# Node host3: ok
```

Changing approach

Instead of a password in the command line, we generate an options file dynamically

```
export user_cnf=user$$.cnf

echo "[client]" > $user_cnf
echo "user=$DB_USER" >> $user_cnf
echo "password=$DB_PASSWORD" >> $user_cnf

export MYSQL="$MYSQL --defaults-file=$PWD/user$$.cnf
--port=$DB_PORT"
```

sample run with new installer and MySQL 5.6

No nasty messages

```
$ ./install_replication.sh
# Making sure MySQL is running
# MASTER host1: ok
# Node host1: ok
# Node host2: ok
# Node host3: ok
# node host1 - server_id: 10
# node host1 - log_bin: 1
# node host2 - server_id: 20
# node host2 - log_bin: 1
# node host3 - server_id: 30
# node host3 - log_bin: 1
```

Serial restore operations

```
$ time ./provision_slaves.sh
taking backup
restoring on slaves
Wed Apr 24 09:10:55 EDT 2013
restore started in node host2
restore done
restore started in node host3
restore done
restore started in node host4
restore done
Wed Apr 24 09:11:03 EDT 2013
real 0m8.539s
user 0m0.323s
sys  0m0.442s
```

parallel restore operations

```
$ time ./provision_slaves_parallel.sh
taking backup
restoring on slaves
Wed Apr 24 09:12:59 EDT 2013
restore started in node host2
restore started in node host3
restore started in node host4
restore done
restore done
restore done
Wed Apr 24 09:13:03 EDT 2013
real 0m4.330s
user 0m0.404s
sys  0m0.542s
```

Serial restore operations (bigger database)

```
$ time ./provision_slaves.sh
taking backup
restoring on slaves
Wed Apr 24 09:20:11 EDT 2013
restore started in node host2
restore done
restore started in node host3
restore done
restore started in node host4
restore done
Wed Apr 24 09:23:53 EDT 2013
real 3m53.504s
user 0m11.401s
sys  0m4.698s
```

Parallel restore operations (bigger database)

```
$ time ./provision_slaves_parallel.sh
taking backup
restoring on slaves
Wed Apr 24 09:26:11 EDT 2013
restore started in node host2
restore started in node host3
restore started in node host4
restore done
restore done
restore done
Wed Apr 24 09:28:00 EDT 2013
real 1m59.583s
user 0m16.505s
sys  0m6.983s
```

Thanks for your attention

- Blog: http://datacharmer.blogspot.com
- Twitter: @datacharmer