Tutoriel sur la mise en place d'un serveur MySQL multiple-instances

par ska_rOOt

Date de publication : 28/09/2005

Dernière mise à jour : 28/09/2005

Ce tutoriel a pour objectif de vous aider à configurer MySQL avec plusieurs instances. Ainsi vous pourrez obtenir sur la même machine plusieurs serveurs pouvant être attribués à des administrateurs différents (un peu comme les hébergeurs). Bien que cela ne soit pas obligatoire mais préconisé, les répertoires de stockage des fichiers de base sont distincts. On ne consomme pas plus de ressources qu'avec un seul serveur. Cette manipulation a été testée sur Fedora Core 4 , Apache 2, MySQL 4.1.10

Télécharger l'article au format PDF (196ko)

I - Pré-requis

- Un système d'exploitation Linux
- Un serveur HTTP
- Un serveur MySQL 4.1.x

II - Information sur mysqld_multi

mysqld_multi est un programme livré avec MySQL. Il sert à gérer plusieurs serveurs utilisant différents sockets Unix et ports TCP/IP sur la même machine.

Pour fonctionner, le programme lit un fichier de configuration dans lequel sont référencés les différents groupes (serveurs de base de données) ainsi que le moyen d'y accéder.

Ce programme se substitue au démon mysqld pour démarrer un, plusieurs, ou tous les serveurs. On peut, comme mysqld configurer le système pour que le service se lance au démarrage de la machine. Cette manipulation est expliquée à la fin du tutoriel.

III - Préparation

Ce tutoriel n'étant ni destiné à vous faire découvrir le monde Unix, ni faire de vous un administrateur Web, nous partirons du fait que l'on a déjà installé Linux ainsi qu'un serveur HTTP.

Pour l'exemple, et vu que les tests ont été réalisés sur ce type de système, les commandes et les chemins des répertoires utilisés sont ceux d'une distribution Fedora Core 4 installée par défaut pour une station de travail. Je ne détaillerai ici ni la configuration de Fedora, ni celle d'Apache, ni celle de MySQL.

Toutefois, le service mysqld n'étant pas démarré par défaut, vous pouvez l'activer après vous être logué en tant que super-utilisateur (root) et en tapant :

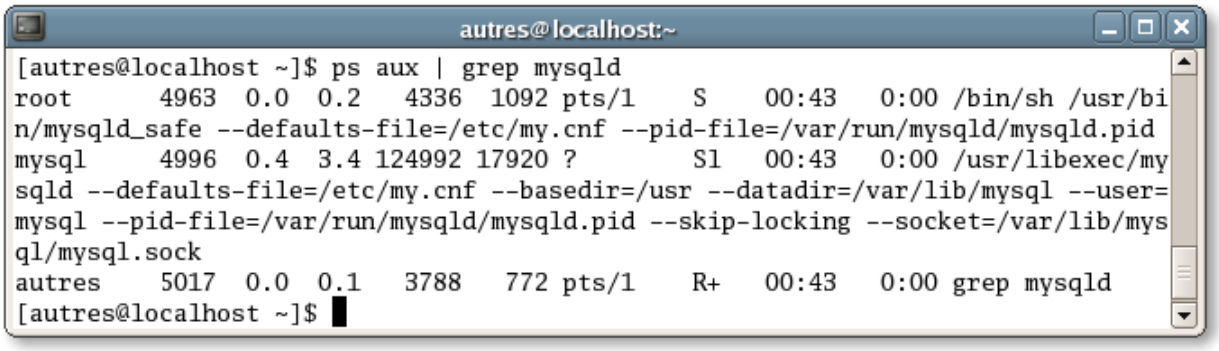
```
# /etc/init.d/mysqld start
```

Ce qui aura pour effet, par l'intermédiaire du sous-programme mysql_install_db d'instaurer une première base de données nommée "**mysql**" contenant les privilèges nécessaires à une première utilisation ainsi qu'à la suite de ce tutoriel.

Si, comme prévu, le service mysqld fonctionne, il est maintenant temps de l'arrêter. Pour vérifier s'il est actif :

```
# ps aux | grep mysqld
```

vous donnera ceci :



vérification process

Pour arrêter le service mysqld :

```
# /etc/init.d/mysqld stop
```

Nous allons à présent créer les différents répertoires qui serviront aux serveurs à stocker leurs fichiers. Il est important d'en avoir un par serveur, ceci afin d'éviter des erreurs irrécupérables.

Mettons nous comme objectif de créer deux serveurs (vous pourrez en créer autant que vous voulez). Une fois reconnu comme utilisateur root, tapez :

```
# mkdir /var/lib/mysql1 /var/lib/mysql2
```

Sur Fedora Core 4, avec installation de mysql natif, le système crée un utilisateur et un groupe nommés "**mysql**". Il

nous faut maintenant donner à cet utilisateur le droit de travailler avec ces répertoires :

```
# chown -R mysql:mysql /var/lib/mysql1 /var/lib/mysql2
```

Le programme mysqld_install_db est utilisé pour initialiser un serveur proprement avec un répertoire de stockage des fichiers de base. Mais comme nous en avons la possibilité, étant donné qu'il s'agit de la même version de moteur, dupliquons simplement le répertoire contenant les privilèges primaires de la base mysql installée par défaut (dans ma configuration, /var/lib/mysql/mysql) dans les répertoires respectifs des nouveaux serveurs.

Si vous n'êtes pas sûr de la localisation du répertoire, ouvrez le fichier my.cnf (/etc/my.cnf) et cherchez la variable **datadir** qui devrait contenir le chemin d'accès recherché.

```
# cp -Rp /var/lib/mysql/mysql /var/lib/mysql1/
# cp -Rp /var/lib/mysql/mysql /var/lib/mysql2/
```

IV - Fichier de configuration

Le fichier de configuration de mysqld se trouve (suivant les distributions) dans /etc/ et se nomme **my.cnf**

On pourrait utiliser ce fichier pour la configuration de mysqld_multi, mais par sécurité (afin de laisser la possibilité d'un éventuel retour), nous allons en créer un autre.

Comme dit précédemment, nous avons comme objectif dans ce tutoriel d'installer deux serveurs distincts, mais rien ne vous empêche d'en créer autant que vous voulez, il suffira d'adapter les répertoires nécessaires et les paramètres de groupes (**[mysqldX]** dans le fichier de configuration, **X** représentant l'identifiant du serveur) pour chacun d'entre eux.

Avant de remplir le fichier de configuration, il convient de vérifier que les répertoires suivants existent. Si ce n'est pas le cas, veuillez adapter les chemins à votre configuration système.

- Répertoire de lancement du programme (par défaut : **/usr/bin**)
- Répertoire de stockage des données et des sockets (par défaut : **/var/lib/mysql**)
- Répertoire de stockage des logs (par défaut : **/var/log**)
- Répertoire de stockage des processus (par défaut : **/var/run/mysqld**)
- Répertoire d'installation du serveur (par défaut : **/var/lib**)

Entrons dans le vif du sujet, ouvrez votre éditeur de texte préféré et créeez le fichier **/etc/multi_my.cnf** (ou le nom de votre choix) comme indiqué ci-dessous :

```
# configuration principale de mysqld_multi
# l'utilisateur multi_user est necessaire pour que
# le programme puisse arreter les moteurs.
[mysqld_multi]
mysqld = /usr/bin/mysqld_safe
mysqladmin = /usr/bin/mysqladmin
user = multi_user
password = mypass

# configuration d'un premier serveur
[mysqld1]
# repertoire de stockage des fichiers de base
datadir=/var/lib/mysql1
# chemin du socket ( imperativement different pour chaque serveur )
socket=/var/lib/mysql1/mysql.sock1
# port ( imperativement different pour chaque serveur )
port=3306
# pour compatibilite avec anciens serveurs
old_passwords=1
# chemin des fichiers de log
err-log=/var/log/mysqld1.log
# chemin des processus
pid-file=/var/run/mysqld/mysqld.pid1
# utilisateur proprietaire du processus
user = mysql

# configuration d'un second serveur
[mysqld2]
datadir=/var/lib/mysql2
socket=/var/lib/mysql2/mysql.sock2
port=3307
old_passwords=1
err-log=/var/log/mysqld2.log
pid-file=/var/run/mysqld/mysqld.pid2
user = mysql

# configuration serveur maître
[mysql.server]
user=mysql
basedir=/var/lib
```

Enregistrez votre fichier, c'est fini pour la configuration mysqld_multi.

V - Première utilisation

Passons aux choses sérieuses, nous allons mettre en route mysqld_multi.

Par défaut, le programme utilise (comme mysqld) le fichier de configuration my.cnf, nous devons donc lui dire qu'il faut utiliser notre fichier fraîchement créé. Pour ce faire, nous nous servirons du paramètre **--config-file** (pour plus d'information -> *man mysqld_multi*)

Syntaxe d'utilisation :

```
mysqld_multi [options] {start|stop|report} [GNR[,GNR]...]
```

mysqld_multi prend comme paramètre : **start**, **stop**, ou **report** suivi des identifiants de serveurs à démarrer. Pour rappel, ils sont repérés par le numéro placé derrière chaque intitulé [mysqld**X**] dans la configuration. Mais on peut aussi mettre un panel de serveurs (exemple **2-6** démarrera les serveurs 2 à 6) ou ne rien mettre, au quel cas il démarrera tous les serveurs présents dans le fichier de configuration.

Nous démarrerons les serveurs 1 et 2 (pour l'exemple)

```
# /usr/bin/mysqld_multi --config-file=/etc/multi_my.cnf start 1,2
```

Une des nombreuses particularités du système Linux est de pouvoir vérifier en temps réel le contenu d'un fichier log. Ceci est réalisé à l'aide de la commande "**tail**".

Ouvrons deux terminaux, et lançons dans chacun d'entre eux une commande tail :

```
# tail -f /var/log/mysqld1.log
# tail -f /var/log/mysqld2.log
```

Si tout va bien, on doit voir ceci :

mysql1.log

mysql2.log

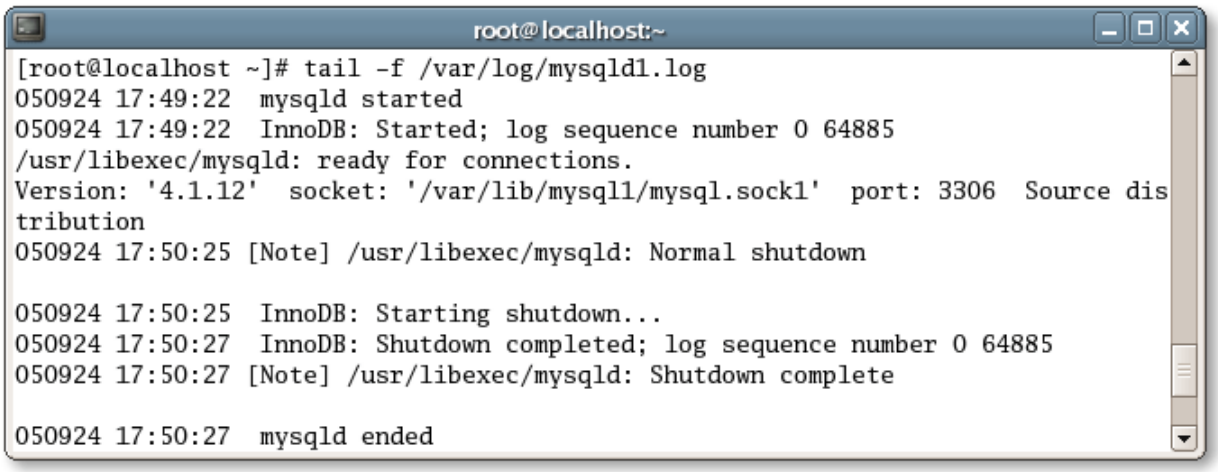
Il faut à présent créer les droits dans les bases mysql pour que l'utilisateur du programme mysqld_multi (*multi_user* dans notre fichier de configuration) puisse fermer les instances à son gré.

```
# mysql -u root -S /var/lib/mysql1/mysql.sock1 -p -e "GRANT SHUTDOWN ON *.* TO multi_user@localhost
IDENTIFIED BY 'mypass'"
# mysql -u root -S /var/lib/mysql2/mysql.sock2 -p -e "GRANT SHUTDOWN ON *.* TO multi_user@localhost
IDENTIFIED BY 'mypass'"
```

Vous pouvez maintenant fermer n'importe quelle instance de serveur en tapant la commande suivante (par exemple pour arrêter le serveur 1) :

```
# mysqld_multi --config-file=/etc/multi_my.cnf stop 1
```

Vérifiez dans le terminal surveillant le fichier log :



mysqld_close

VI - phpMyAdmin

Afin d'offrir aux administrateurs la possibilité de gérer agréablement leur(s) serveur(s), nous installerons également *phpMyAdmin*, une interface réalisée en PHP qui permet de gérer toutes les actions courantes (privilèges, projections, insertions, paramétrages, etc...) sur MySQL à travers un visuel convivial.

Vous trouverez les sources de ce programme à cette adresse : http://www.phpmyadmin.net/home_page/

La version disponible à l'heure ou j'écris cet article est : **phpMyAdmin-2.6.4-pl1.tar.bz2**

Téléchargez l'archive bzip2 dans le répertoire de votre choix (exemple : /home/toto/). Placez vous dans le répertoire web d'Apache (par défaut : /var/www/html) et décompressez l'archive.

```
# cd /var/www/html
# bzip2 -d /home/toto/phpMyAdmin-2.6.4-pl1.tar.bz2
# tar -xvvf /home/toto/phpMyAdmin-2.6.4-pl1.tar
```

Nous allons créer deux repertoires phpMyAdmin nommés **sqladmin1** et **sqladmin2** :

```
# mv phpMyAdmin-2.6.4-pl1 sqladmin1
# cp -Rp sqladmin1 sqladmin2
```

Modifions le fichier **sqladmin1/config.inc.php** ainsi :

```
$cfg['PmaAbsoluteUri']              = 'http://domaine_serveur_http/sqladmin1/';
$cfg['Servers'][$i]['port']         = '3306';
$cfg['Servers'][$i]['socket']       = '/var/lib/mysql1/mysql.sock1';
$cfg['Servers'][$i]['connect_type'] = 'socket';
$cfg['Servers'][$i]['auth_type']    = 'http';
```

Modifions le fichier **sqladmin2/config.inc.php** à présent :

```
$cfg['PmaAbsoluteUri']              = 'http://domaine_serveur_http/sqladmin2/';
$cfg['Servers'][$i]['port']         = '3307';
$cfg['Servers'][$i]['socket']       = '/var/lib/mysql2/mysql.sock2';
$cfg['Servers'][$i]['connect_type'] = 'socket';
$cfg['Servers'][$i]['auth_type']    = 'http';
```

Vous pouvez maintenant vous connecter à phpMyAdmin à l'adresse suivante

serveur 1 = http://localhost/sqladmin1/ (root sans mot de passe)

serveur 2 = http://localhost/sqladmin2/ (root sans mot de passe)

> *Profitez de l'interface pour créer un nouvel utilisateur avec mot de passe et tous les privilèges (GRANT ALL PRIVILEGES). Puis (je dirais surtout) supprimez l'utilisateur **root**.*
>
> *Faites attention à ne pas supprimer l'utilisateur multi_user, cela n'a pas de graves conséquences, mais vous ne pourriez plus arrêter l'instance de ce serveur avec la commande mysqld_multi.*

Une fois les modifications de privilèges réalisées, modifiez les deux fichiers **config.inc.php** afin qu'ils correspondent avec le nom d'utilisateur ayant tous les privilèges.

```
$cfg['Servers'][$i]['user']          = 'utilisateur_admin';
```

Modifiez les droits sur les répertoires sqladmin pour empêcher tout accès direct aux fichiers. Seul l'utilisateur d'apache doit pouvoir y avoir accès (dans ma config : *apache*) :

```
# chown -R apache:apache /var/www/html/sqladmin1/ /var/www/html/sqladmin2/
# chmod -R go-wrx /var/www/html/sqladmin1/ /var/www/html/sqladmin2/
```

Puis créez un fichier **.htaccess** dans chaque répertoire "**sqladmin**" :

```
<Files config.inc.php>
   Deny from all
</Files>
```

C'est fini pour **phpMyAdmin** et sa sécurité, vous pouvez dormir tranquille...

VI - Démarrage automatique au boot

Il ne reste plus qu'à créer un script pour automatiser le démarrage des serveurs voulus au boot de la machine. Ouvrez à nouveau votre éditeur de texte et copier/coller le script ci-dessous. Enregistrez-le sous **/etc/init.d/multi_mysqld**

> *N'oubliez pas d'ajouter les références de serveurs à démarrer derrière la commande mysqld_multi si vous ne voulez en démarrer qu'un partie.*

```
#!/bin/bash
#
# multi_mysqld       Ce script permet de demarrer ou d'arreter
#                    le programme MySQL (mysqld_multi).
#
# chkconfig: - 64 36
# description:    MySQL database servers.
# processname: mysqld_multi
# config: /etc/multi_my.cnf

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Nom du programme
prog="MySQL multi-serveurs"

# Chemin du fichier de configuration
conf="/etc/multi_my.cnf"

# Chemin du fichier de log
err_m="/var/log/multi_mysqld.log"

# deco
sep="*--------------------------------*"


test_var(){

    variable="$1"
    type_rech="$2"
    msg=""
    if [ $type_rech = "x" ];then
       msg="exécutable"
    else
       msg="lisible"
    fi

    # extraction des donnees
    awk 'BEGIN{FS="="}($1 ~ /^'$variable'/){print $2;}' "$conf" | sed 's/[ \t]*//' 2>/dev/null >
/tmp/.tmpfic
    # test d'existence des fichiers en lecture et repertoires
    retour=`awk 'function TestDir(Rep)
         {
            Cmd = "test -'$type_rech' " Rep ";echo $?"
            Cmd|getline Result
            close(Cmd)
            if (Result==1) return 0
            if (Result==0) return 1
            print Result
         }
         {
            if (TestDir($1)) printf "%-40s %s\n","'$variable'="$1,"OK"
            else printf "%-40s %s %s\n","'$variable'="$1,"non trouvé ou non","'$msg'"
         }' /tmp/.tmpfic 2>/dev/null`
    echo "$retour"
}

# fonction qui verifie l'existence d'une variable de configuration
get_mysql_option(){
    result=`sed -n "s/^[ \t]*$2[ \t]*=[ \t]*//p" "$1" 2>/dev/null | tail -n 1`
    if [ -z "$result" ]; then
```

```
        # non trouve retourne KO
        result="KO"
    else

        # retire les quotes et commentaires
        dequoted=`echo "$result" | sed "s/^'\([^']*\)'.*$/\1/"`
        if [ x"$dequoted" != x"$result" ]; then
            result="$dequoted"
        else
            dequoted=`echo "$result" | sed 's/^"\([^"]*\)".*$/\1/'`
            if [ x"$dequoted" != x"$result" ]; then
                result="$dequoted"
            else
                result=`echo "$result" | sed 's/^\([^ \t#]*\).*$/\1/'`
            fi
        fi

    fi
}

# teste des variables de configuration
get_mysql_option $conf datadir
datadir="$result"
get_mysql_option $conf socket
socketfile="$result"
get_mysql_option $conf err-log
errlogfile="$result"
get_mysql_option $conf pid-file
mypidfile="$result"

# fonction qui etablit le message d'erreur
get_params_ok(){
    retour=""
    if [ ! -r "$1" ]; then
       retour="Fichier de configuration introuvable : $1"
       return -1
    fi
    if [ "$2" = "KO" ]; then
       retour="Aucun paramètre datadir trouvé !"
       return -1
    fi
    if [ "$3" = "KO" ]; then
       retour="Aucun paramètre socket trouvé !"
       return -1
    fi
    if [ "$4" = "KO" ]; then
       retour="Aucun paramètre err-log trouvé !"
       return -1
    fi
    if [ "$5" = "KO" ]; then
       retour="Aucun paramètre pid-file trouvé !"
       return -1
    fi
    if [ ! -d "$2" ]; then
       retour="Répertoire fichiers MySQL ( datadir ) introuvable !"
       return -1
    fi
}

# demarrage
start(){

    # teste si au moins un serveur peut demarrer
    get_params_ok $conf $datadir $socketfile $errlogfile $mypidfile

    # si test ko
    if [ ! -z "$retour" ]; then

        # log multi_mysqld.log
        touch "$err_m"
        echo `LC_ALL=fr_FR date '+%A %d %B %Y %H:%M:%S'` " : $retour">>"$err_m"
        echo $"Démarrage $prog: "
        action $"$retour" /bin/false
        return -1
    fi

    # execution de mysqld_multi
    /usr/bin/mysqld_multi --config-file="$conf" start >/dev/null 2>&1
```

```
    ret=$?
    if [ $ret -eq 0 ]; then

        # si tout ok
        sleep 2
        action $"Démarrage $prog: " /bin/true
    else

        # si non
        action $"Démarrage $prog: " /bin/false
    fi
    return $ret
}

# arret
stop(){

    # teste les parametres de configuration
    get_params_ok $conf $datadir $socketfile $errlogfile $mypidfile

    # si test ko
    if [ ! -z "$retour" ]; then

        # log multi_mysqld.log
        touch "$err_m"
        echo `LC_ALL=fr_FR date '+%A %d %B %Y %H:%M:%S'` " : $retour">>"$err_m"
        echo $"Démarrage $prog: "
        action $"$retour" /bin/false
        return -1
    fi

    # execution de mysqld_multi
    /usr/bin/mysqld_multi --config-file="$conf" stop >/dev/null 2>&1
    ret=$?
    if [ $ret -eq 0 ]; then

        # si tout ok
        sleep 1
        action $"Arrêt $prog: " /bin/true
    else

        # si non
        action $"Arrêt $prog: " /bin/false
    fi
    return $ret
}

restart(){
    stop
    start
}


config(){
    # si fichier de configuration existe
    if [ ! -r "$conf" ]; then
        echo "fichier de configuration introuvable : $conf"
        echo $sep
        action $"Lecture fichier de configuration $prog" /bin/false
    else
   # affichage sans les commentaires et sans espaces ni tabulations
        echo $sep
        echo "| Lecture fichier configuration  |"
        echo $sep
        awk '! (/^ *#/ || /^$/) { print $0 }' "$conf" 2>/dev/null | sed 's/[ \t]*=[ \t]*/=/'
        echo $sep
        action $"Lecture fichier de configuration $prog" /bin/true
    fi
}


config_test(){
    # si fichier de configuration existe
    if [ ! -r "$conf" ]; then
       echo "fichier de configuration introuvable : $conf"
       echo $sep
       action $"Test fichier de configuration $prog" /bin/false
    else
   # verification de l'existence des valeurs de chaque variable
```

```
            echo $sep
            echo "| Test des variables nécessaires |"
            echo $sep
            test_var "err-log" "r"
            test_var "datadir" "d"
            test_var "mysqld" "x"
            test_var "mysqladmin" "x"
            test_var "basedir" "d"
            rm -f /tmp/.tmpfic 2>/dev/null
            echo $sep
            action $"Test du fichier de configuration $prog" /bin/true
        fi
    }

    case "$1" in
      start)
        start
        ;;
      stop)
        stop
        ;;
      status)
        status mysqld
        ;;
      restart)
        restart
        ;;
      config)
        config
        ;;
      config_test)
        config_test
        ;;
      *)
        echo $"Usage: $0 {start|stop|status|restart|config|config_test}"
        exit 1
    esac

    exit $?
```

Enregistrons le et rendons-le exécutable

```
# chmod +x /etc/init.d/multi_mysqld
```

A ce stade, vous n'avez plus besoin de passer par la commande /usr/bin/mysqld_multi pour gérer l'ensemble de vos serveurs. Par exemple, je veux démarrer tous mes serveurs :

```
# /etc/init.d/multi_mysqld start
```

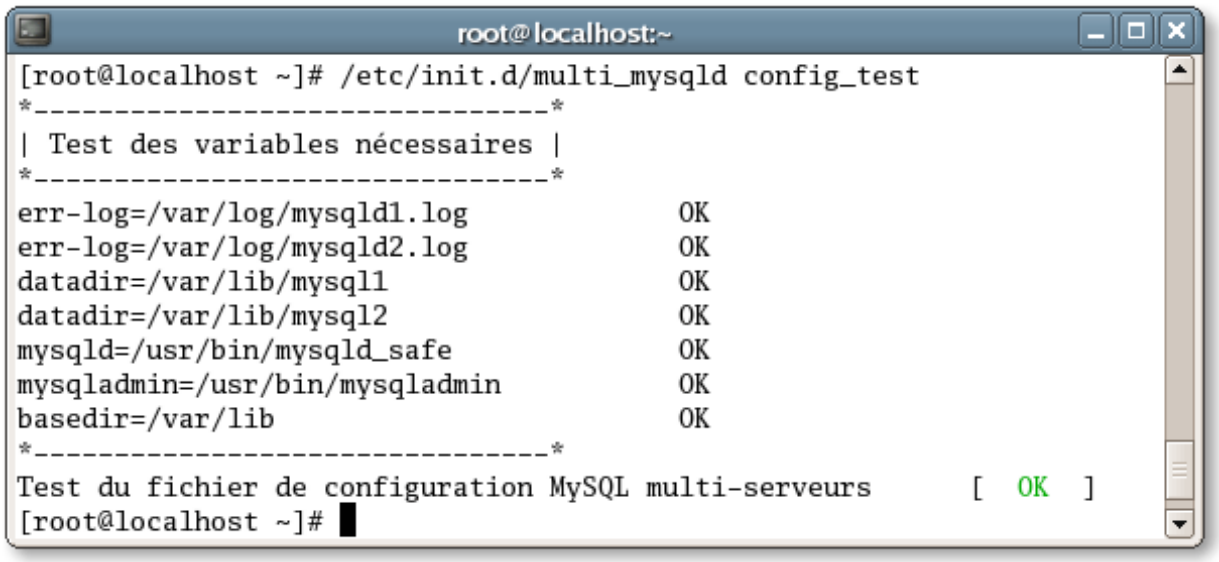
démarrage

Le script permet aussi de réaliser deux autres tâches intéressantes, le paramètre "**config**" affichera le contenu du fichier de configuration épuré, le paramètre "**config_test**" affichera le résultat du test d'existence des valeurs de variables déclarées.

J'ai trouvé pratique de pouvoir vérifier l'intégrité du fichier de configuration avant de lancer le démon notamment lors de la création d'une nouvelle instance.

```
# /etc/init.d/multi_mysqld config_test
```

et voici le résultat du test d'intégrité de notre fichier de configuration :



test de la configuration

Il ne nous reste plus qu'à configurer le system V :

```
# chkconfig --add multi_mysqld
# chkconfig --level 345 multi_mysqld on
# chkconfig --level 345 mysqld off
```

C'est terminé, bravo, vous avez maintenant 2 instances de mysql sur votre machine dès le boot de la machine !

Merci à Maximilian pour la relecture de cet article.