Construction de Redis sur Raspbian

Redis est un magasin clé-valeur utilisé par **ntopng** pour mettre en cache les données et les préférences. **Redis** s'exécute en tant que service externe. **ntopng** se connecte à **Redis** à l'aide de sockets. Ne pas se connecter à **Redis** entraînerait un dysfonctionnement de **ntopng**. **Redis** doit toujours être opérationnel et accessible pour garantir l'intégralité des fonctionnalités.

Construire les binaires à partir des sources

Télécharger les sources **Redis**.

```
wget http://download.redis.io/redis-stable.tar.gz
```

Désarchiver le package téléchargé.

```
tar xzf redis*
```

Aller dans le dossier `redis/`:

```
cd redis*
```

Construire maintenant **Redis** à partir de sources sur le Raspberry Pi avec les commandes suivantes.

```
sudo make
make test
sudo make install PREFIX=/usr
```

À ce stade, on a compilé Redis maintenant, il faut le rendre accessible à tous les utilisateurs:

```
sudo mkdir /etc/redis
```

Copier le fichier `redis.conf` là dedans

```
sudo cp redis.conf /etc/redis/
```

Il est temps de nettoyer

```
cd ..
```

Supprimer l'archive et le dossier avec cette commande.

```
```

```
sudo rm -Rf redis*
```

Installation et configuration

Créer un utilisateur sans répertoire personnel et sans possibilité de se connecter au système pour exécuter Redis en tant qu'utilisateur normal.

```
sudo adduser --system --group --disabled-login redis --no-create-home --
shell /bin/nologin --quiet
```

Vérifier le fichier shadow pour voir si l'utilisateur Redis a été créé.

```
cat /etc/passwd | grep redis
```

Modifier le fichier de configuration Redis pour configurer la mise en cache.

```
sudo nano /etc/redis/redis.conf
```

Rechercher les entrées suivantes. certains sont déjà définis. Mais, il y a encore peu de valeurs à définir. Par exemple,

- `daemonize yes` pour exécuter Redis en tant que Deamon.
- `stop-writes-on-bgsave-error no`, Redis s'arrête en cas d'échec et nécessite un redémarrage manuel. En utilisant ce paramètre, Redis ne s'arrêtera pas en cas d'échec.
- `maxmemory 70M`, la RAM est limitée sur un RaspberryPi, donc 70 Mo suffisent pour la plupart des besoins. Si on a besoin de plus de mémoire, ajuster ce paramètre ultérieurement.

```
bind 127.0.0.1
port 6379
daemonize yes
stop-writes-on-bgsave-error no
rdbcompression yes
maxmemory 50M
maxmemory-policy allkeys-lru
```

À ce stade, on peut créer un script pour exécuter le serveur **Redis**.

Créer un répertoire à partir duquel ce script sera exécuté.

```
sudo mkdir -p /var/run/redis
```

Faire de l'utilisateur redis le propriétaire de ce répertoire.

```
sudo chown -R redis /var/run/redis
```

Daemonisation

Créer le scénario.

```
sudo nano /etc/init.d/redis-server
```

Coller ce qui suit.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides: redis-server
# Required-Start: $syslog $remote_fs
# Required-Stop: $syslog $remote_fs
# Should-Start: $local_fs
# Should-Stop: $local_fs
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: redis-server - Persistent key-value db
# Description: redis-server - Persistent key-value db
### END INIT INFO

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/bin/redis-server
DAEMON_ARGS=/etc/redis/redis.conf
NAME=redis-server
DESC=redis-server

RUNDIR=/var/run/redis
PIDFILE=$RUNDIR/redis-server.pid

test -x $DAEMON || exit 0

if [ -r /etc/default/$NAME ]
then
. /etc/default/$NAME
fi

. /lib/lsb/init-functions

set -e

case "$1" in
  start)
echo -n "Starting $DESC: "
mkdir -p $RUNDIR
touch $PIDFILE
chown redis:redis $RUNDIR $PIDFILE
chmod 755 $RUNDIR
```

```
if [ -n "$ULIMIT" ]
then
ulimit -n $ULIMIT
fi

if start-stop-daemon --start --quiet --umask 007 --pidfile $PIDFILE --chuid
redis:redis --exec $DAEMON -- $DAEMON_ARGS
then
echo "$NAME."
else
echo "failed"
fi
;;
  stop)
echo -n "Stopping $DESC: "
if start-stop-daemon --stop --retry forever/TERM/1 --quiet --oknodo --
pidfile $PIDFILE --exec $DAEMON
then
echo "$NAME."
else
echo "failed"
fi
rm -f $PIDFILE
sleep 1
;;

  restart|force-reload)
${0} stop
${0} start
;;

  status)
status_of_proc -p ${PIDFILE} ${DAEMON} ${NAME}
;;

  *)
echo "Usage: /etc/init.d/$NAME {start|stop|restart|force-reload|status}" >&2
exit 1
;;
esac

exit 0
```

Rendre ce script exécutable.

```
sudo chmod +x /etc/init.d/redis-server
```

Ajouter ce script à exécuter en tant que commande

```
```

```
sudo update-rc.d par défaut du serveur redis
```

Si tout est correctement configuré, le serveur Redis peut désormais être manipulé via cette commande

```
sudo service redis-server restart
```

Maintenant, vérifier l'état du serveur

```
redis-server -v
```

S'assurer qu'il écoute sur 127.0.0.1

```
netstat -antp
```

S'assurer que le processus Redis s'exécute en tant qu'utilisateur redis.

```
ps aux | grep redis
```

On peut maintenant profiter de l'installation du serveur de mise en cache Redis sur le Raspberry Pi.