

# Surveillance système d'une machine avec Monit

par Nicolas Vallée ([Home Page](#))

Date de publication : 13/05/2007

Dernière mise à jour : 16/05/2007

Surveiller de manière simple l'activité de tous les services importants sur sa machine...

## I - Introduction

## II - Mise en place

### II-1 - Installation

### II-2 - Configuration du service

### II-3 - Création des plugins

#### Les différents types de contrôle

#### II-3-1 - Comment créer un contrôle ?

#### II-3-2 - Les actions possibles

#### II-3-3 - Quelques tests disponibles

## III - Consultation des résultats

### III-1 - A distance via interface web

### III-2 - En local, par la ligne de commande

#### III-2-1 - Commandes applicables à un sous-ensemble des contrôles

#### III-2-2 - Commandes appliquées à l'ensemble des contrôles

## IV - Divers

### IV-1 - Vue de l'interface web

### IV-2 - Liens utiles

## I - Introduction

**Monit** est un outil de surveillance des services locaux installés sur une machine. Il peut vérifier la disponibilité d'un *démon* et les ressources occupées qu'il consomme, et en fonction du résultat choisir de le laisser tranquille, de le redémarrer ou de le stopper. Cette faculté de ne pas seulement surveiller, et éventuellement alerter les administrateurs, est une particularité de Monit en comparaison à d'autres logiciels comme **Munin** ou **Nagios**. Il faut toutefois signaler qu'il existe un moyen détourné de le faire dans Nagios avec les *Event Handlers*.

Il faut aussi préciser que Monit est un logiciel libre distribué sous la licence GPL.

J'ai choisi de présenter cet outil en raison de la grande facilité de prise en main, tout en conservant de grandes possibilités. C'est d'ailleurs la même raison qui m'avait poussé à rédiger un [article sur Munin](#).

Comme souvent, j'ai choisi de vous détailler l'installation sur le système Debian-like (ici Ubuntu Feisty Fawn 7.04), mais tout peut s'appliquer à un autre Unix.

## II - Mise en place

### II-1 - Installation

Pour installer le démon monit, il suffit de lancer cette commande :

```
$ aptitude install monit
$
```

### II-2 - Configuration du service

Pour la configuration du service, j'ai choisi de repartir d'un fichier entièrement vierge, et de garder celui d'origine pour mémoire ;-)

```
$ cd /etc/monit
$ cp monitrc monitr.bak
$ touch monitrc
$
```

Tout d'abord, il faut prévenir le système que nous avons configuré monit, et par conséquent, que le démon peut être démarré sans risque. Il s'agit d'une précaution prise par certains unix pour éviter le démarrage d'un démon en configuration par défaut après son installation.

Pour cela, il suffit de mettre à 1 la valeur de la directive *startup* dans le fichier **/etc/default/monit**.

```
/etc/default/monit
startup=1
```

Au passage, vous pouvez aussi régler dans ce fichier la durée entre deux vérifications effectuées par le démon monit. Il suffit de donner une valeur (en secondes) à la directive *CHECK\_INTERVALS*. Pour ma part, j'ai choisi de le faire au niveau de la configuration principale du service ;-)

```
/etc/default/monit

startup=1
# durée entre deux surveillances réglée à 3 minutes (180 secondes)
CHECK_INTERVALS=180
```

Passons maintenant à la véritable configuration du service. Il faut éditer le fichier **/etc/monit/monitrc**.

Commençons par demander que le démon surveille toutes les 2 minutes (120 secondes), ce qui va écraser l'éventuelle valeur de *CHECK\_INTERVALS* contenue dans **/etc/default/monit**. Par ailleurs, on spécifie également le mode de logs système à utiliser (ici on veut recevoir les messages dans */var/log/daemon.log*).

```
/etc/monit/monitrc

### Global Section


# durée entre deux controles : ici 2 minutes (120 secondes)
set daemon 120
# activer les logs
set logfile syslog facility log_daemon # par défaut log_user
```


```
/etc/monit/monitrc
```

Maintenant, demandons lui de nous alerter par mail en cas de problèmes. Pour cela, il suffit d'ajouter ceci au fichier de configuration.

```
/etc/monit/monitrc
```

```
# configurer l'alerte par mail
set mailserver localhost
set mail-format { from: monit@mon.domaine.com }
set alert root@localhost
```

 *L'adresse mail `monit@mon.domaine.com` n'est pas obligée d'exister. Ici, il s'agit juste d'une information que l'on décide de mettre dans l'entête du message... rien n'indique que l'on puisse répondre au message ;-)*

 *Attention dans cette configuration, si aucun MTA (Mail Transfert Agent) n'est configuré en local, il est probable que les alertes finiront dans le fichier `/var/mail/root` (ou la boîte de l'utilisateur qui reçoit les mails de root).*

Enfin, il faut maintenant configurer la surveillance des services proprement dite. Normalement, il faudrait écrire toutes les commandes à la suite dans ce même fichier de configuration. Mais il existe une méthode plus élégante, qui consiste à fractionner chaque configuration de service à surveiller dans différents fichiers.


Je vais prendre le modèle de la gestion des mods sous Apache... Tout d'abord, créons les répertoires nécessaires. Nous placerons les "plugins" dans le répertoire `/etc/monit/plugins-available`, et nous ferons un lien symbolique dans `/etc/monit/plugins-enabled` pour activer ce que nous souhaitons utiliser.

```
$ mkdir /etc/monit/plugins-available
$ mkdir /etc/monit/plugins-enabled
```

Il faut maintenant dire à Monit d'aller chercher ses plugins dans le bon répertoire. Pour cela, il suffit d'ajouter ceci au fichier **`/etc/monit/monitrc`**.


```
/etc/monit/monitrc
```

```
# aller chercher les plugins
include /etc/monit/plugins-enabled/*
```

 *Si le répertoire est vide, cela créera une erreur au démarrage :(*

## II-3 - Création des plugins

Nous arrivons maintenant à la partie la plus intéressante, mais la plus complexe, à savoir la création des plugins pour surveiller un service.

 *Cette configuration est insensible à la casse. Il pourra donc m'arriver de mélanger minuscules/majuscules tout au long de mes exemples...*

## Les différents types de contrôle

Monit permet de surveiller différents types d'entités accessibles depuis notre système, tels que :

- les processus tournant sur la machine locale ;
- les fichiers accessibles depuis le système de fichiers comme un fichier local (donc y compris montage NFS & cie...);
- les répertoires accessibles depuis le système de fichiers comme un fichier local (d'ailleurs sous Unix, un répertoire n'est qu'un fichier particulier ;-);
- le matériel connecté à la machine locale ;
- les hôtes distants (utile pour les switches & cie).

J'ai choisi de ne présenter que des exemples de contrôle sur les processus et les fichiers. Si vous souhaitez faire un autre type de contrôle, sachez qu'il n'y a que la première ligne (*check...* introduisant le contrôle) qui change par rapport à mes exemples. Le reste est identique :-)

### II-3-1 - Comment créer un contrôle ?


Au minimum, il faut indiquer :


- le fichier contenant le pid du processus (souvent dans `/var/run/`) ;
- la commande pour démarrer le service ;
- la commande pour arrêter le service ;

#### `/etc/monit/plugins-available/service`

```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
    start program = "/etc/init.d/service start"
    stop  program = "/etc/init.d/service stop"
    depends on service_rc

check file service_rc with path /etc/init.d/service
    if failed checksum then unmonitor
    if failed permission 755 then unmonitor
    if failed uid root then unmonitor
    if failed gid root then unmonitor
```

 *Remarquez que cet exemple minimal implique déjà la création de la vérification d'une dépendance, à savoir l'existence du ou des exécutables permettant de lancer, et d'arrêter le service. D'ailleurs, il doit vérifier que ce fichier appartienne bien à root et que seul root puisse le modifier, afin d'être sûr qu'un petit malin ne tente pas de changer le comportement du service...*

 *Lorsque vous écrivez un plugin, veillez à ce que le nom de chaque "check" que vous créez ne soit pas déjà utilisé par un autre plugin. Un moyen simple de ne pas se tromper est de préfixer ce nom par celui du fichier contenant ce plugin. En effet, à la fin tout se passera comme si l'on avait concaténé tous ces petits fichiers, et il ne faut pas avoir d'ambiguïté sur les noms.*

Entrons un peu plus dans les détails...

Pour commencer, il se peut que de nombreux services soient à surveiller sur une machine, auquel cas il peut être utile de les regrouper en groupes, afin de pouvoir par la suite effectuer certaines opérations sur tous les éléments d'un groupe en une seule commande. Pour cela, il suffit d'ajouter l'option `group`.

```
/etc/monit/plugins-available/service
```

```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
  group mongroupe
  start program = "/etc/init.d/service start"
  stop program = "/etc/init.d/service stop"
  depends on service_rc

check file service_rc with path /etc/init.d/service
  group mongroupe
  if failed checksum then unmonitor
  if failed permission 755 then unmonitor
  if failed uid root then unmonitor
  if failed gid root then unmonitor
```

Toutes les autres commandes à indiquer seront du type suivant :

```
if <test> then <action>
```

## II-3-2 - Les actions possibles

Commençons par décrire les actions qu'il est possible d'effectuer dans chaque plugin.

### alert

- envoyer un message d'alarme, en utilisant l'action **alert**

```
/etc/monit/plugins-available/service
```

```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
  group mongroupe
  start program = "/etc/init.d/service start"
  stop program = "/etc/init.d/service stop"
  if failed host 127.0.0.1 port 8888 protocol tcp then alert
  depends on service_rc

check file service_rc with path /etc/init.d/service
  group mongroupe
  if failed checksum then unmonitor
  if failed permission 755 then unmonitor
  if failed uid root then unmonitor
  if failed gid root then unmonitor
```

S'il produit une alerte, monit vous le notifiera par mail et dans le log `/var/log/daemon.log`

```
/var/log/daemon.log
```

```
May 12 23:32:49 localhost monit[19178]: 'service' process is not running
May 12 23:32:50 localhost monit[19178]: 'service' trying to restart
May 12 23:32:50 localhost monit[19178]: 'service' start: /etc/init.d/service
```

**/var/log/daemon.log**

```
May 12 23:34:50 localhost monit[19178]: 'service' process is running with pid 19678
```

**mail d'alerte**


```
From: monit@GorgonMobile Sat May 12 23:34:50 2007
Return-Path: <monit@GorgonMobile>
X-Original-To: root@localhost
Delivered-To: root@localhost
Received: from GorgonMobile (localhost [127.0.0.1])
        by localhost (Postfix) with SMTP id 207FD1174E0
        for <root@localhost>; Sat, 12 May 2007 23:34:50 +0200 (CEST)
```

```
From: monit@GorgonMobile
To: root@localhost
Subject: monit alert -- Exists service
Date: Sat, 12 May 2007 23:34:50 +0200
X-Mailer: monit 4.8.1
Mime-Version: 1.0
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable
Message-Id: <20070512213450.207FD1174E0@localhost>
```

Exists Service service

```
Date: Sat, 12 May 2007 23:34:50 +0200
Action: alert
Host: GorgonMobile
Description: 'service' process is running with pid 19678
```

Your faithful employee,  
monit

 Vous pouvez constater que si le service est inaccessible, et que rien n'est spécifié, par défaut Monit essaiera de le redémarrer.


**restart**

- redémarre le service, en utilisant l'action **restart**

**/etc/monit/plugins-available/service**

```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
    group mongroupe
    start program = "/etc/init.d/service start"
    stop program = "/etc/init.d/service stop"
    if failed host 127.0.0.1 port 8888 protocol tcp then restart
    depends on service_rc

check file service_rc with path /etc/init.d/service
    group mongroupe
    if failed checksum then unmonitor
    if failed permission 755 then unmonitor
    if failed uid root then unmonitor
    if failed gid root then unmonitor
```

 Dans cet état, cela ne fera rien... en effet, cela va juste incrémenter un "compteur" d'ordre de redémarrage. Ainsi, cela nous permet de régler plus finement quand le redémarrage

*doit avoir lieu. En effet, il est possible qu'un serveur ne réponde pas une fois pour diverses raisons, mais qu'il soit quand même en état de fonctionner... on choisit par exemple de redémarrer s'il y a eu 5 demandes de redémarrage lors des 5 derniers cycles.*

```
/etc/monit/plugins-available/service
```

```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
  group mongroupe
  start program = "/etc/init.d/service start"
  stop program = "/etc/init.d/service stop"
  if failed host 127.0.0.1 port 8888 protocol tcp then restart
  if 5 restarts within 5 cycles then timeout
  depends on service_rc

check file service_rc with path /etc/init.d/service
  group mongroupe
  if failed checksum then unmonitor
  if failed permission 755 then unmonitor
  if failed uid root then unmonitor
  if failed gid root then unmonitor
```

## start

- démarre le service, en utilisant l'action **start**, qui fonctionne sur le modèle de restart

```
/etc/monit/plugins-available/service
```

```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
  group mongroupe
  start program = "/etc/init.d/service start"
  stop program = "/etc/init.d/service stop"
  if failed host 127.0.0.1 port 8888 protocol tcp then start
  if 5 starts within 5 cycles then timeout
  depends on service_rc

check file service_rc with path /etc/init.d/service
  group mongroupe
  if failed checksum then unmonitor
  if failed permission 755 then unmonitor
  if failed uid root then unmonitor
  if failed gid root then unmonitor
```

## stop

- arrête le service, en utilisant l'action **stop**

```
/etc/monit/plugins-available/service
```

```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
  group mongroupe
  start program = "/etc/init.d/service start"
  stop program = "/etc/init.d/service stop"
```

#### /etc/monit/plugins-available/service

```
if cpu usage > 90% then stop
depends on service_rc

check file service_rc with path /etc/init.d/service
group mongroupe
if failed checksum then unmonitor
if failed permission 755 then unmonitor
if failed uid root then unmonitor
if failed gid root then unmonitor
```

Si l'action est déclenchée, vous obtiendrez un joli **Resource limit matched** dans l'interface de visualisation (j'en parlerais plus loin).


### exec

- exécute une action quelconque et le signale via une alerte, en utilisant l'action **exec**

#### /etc/monit/plugins-available/service


```
# vim.syntax=conf
### Mon Service
check process service with pidfile /var/run/service.pid
group mongroupe
start program = "/etc/init.d/service start"
stop program = "/etc/init.d/service stop"
if cpu usage > 90% then exec "/bin/echo Monit service en surcharge > /root/monfichier.log"
depends on service_rc


check file service_rc with path /etc/init.d/service
group mongroupe
if failed checksum then unmonitor
if failed permission 755 then unmonitor
if failed uid root then unmonitor
if failed gid root then unmonitor
```

 *Je n'ai pas réussi à m'en servir... si vous y parvenez, je veux bien que vous me décriviez comment vous vous y êtes pris ;-)*

### unmonitor

- désactive le contrôle concerné, en utilisant l'action **unmonitor**

 *Je ne mets pas d'exemples, il est dans chaque exemple fourni jusqu'alors... :-)*

 *Ce contrôle ne sera plus effectué, ni redémarré plus tard.*

*Si vous souhaitez le réactiver, il faudra en donner l'ordre explicitement (on verra plus tard comment faire), ou redémarrer le serveur.*

## II-3-3 - Quelques tests disponibles

Je ne présenterai pas tous les tests possibles... en raison de leur grand nombre. C'est d'ailleurs ce qui fait la force de Monit. Pour de plus amples renseignements à ce sujet, il faudra consulter la **documentation officielle**.

## test de dépendance


- on vérifie qu'un autre contrôle est correct ;

```
depends on service_rc
```

## test de fichiers

- on effectue un contrôle sur les propriétés d'un fichier ;


```
if failed checksum then unmonitor
if failed checksum and expect the sum 8f7f419955cefa0b33a2ba316cba3659 then alert
if failed permission 755 then unmonitor
if failed uid root then unmonitor
if failed gid root then unmonitor
if timestamp > 10 minutes then alert
```

 *En ce qui concerne checksum, il faut avoir conscience de deux subtilités. D'une part, si l'on ne spécifie pas la somme de fichier à tester, Monit la calculera lors du premier chargement du plugin, et l'enregistrera pour les futures comparaisons. D'autre part, la checksum est calculée en md5 (par défaut) ou en sha1 (mot-clé à ajouter devant checksum). Si l'on ne précise pas l'algorithme de hashage, Monit le déterminera avec la chaîne passée en paramètre.*

## test de connexion

- on teste si l'on peut se connecter à un hôte sur un certain port ;

```
if failed host 127.0.0.1 port 4949 type tcp then restart
if failed host 127.0.0.1 port 53 type udp then restart
if failed host 127.0.0.1 port 443 type tcpssl then restart
```

 *Notez que l'on peut également spécifier le protocole de chiffage utilisé par la connexion tcpssl (SSLAUTO,SSLV2,SSLV3,TLSV1), et vérifier la checksum du certificat (CERTMD5 md5sum)*

## test de protocole

- on effectue une connexion sur un serveur suivant un protocole spécifié ;

```
if failed host 127.0.0.1 port 80 protocol http then ...
if failed host 127.0.0.1 port 25 protocol smtp then ...
if failed host 127.0.0.1 port 3306 protocol mysql then ...
if failed host 127.0.0.1 port 5432 protocol pgsq then ...
if failed host 127.0.0.1 port 22 protocol ssh then ...
```

## test de requêtes

- on effectue une requête sur un serveur suivant un protocole spécifié ;

```
if failed host 127.0.0.1 port 80 protocol http request /monit/token then ...
if failed host 127.0.0.1 port 80 protocol http request /monit/token
```

```
with checksum f9d26b8393736b5dfad837bb13780786 then ...
```

### test de ressources

- on teste les ressources occupées par un service. ici, on teste respectivement (dans l'ordre d'apparition) l'utilisation du processeur, l'utilisation de la ram en pourcentage et en quantitatif, le nombre fils du processus, la charge moyenne sur les 5 dernières minutes, l'espace occupé et le nombre d'inodes utilisés dans un device ;

```
if cpu usage > 90% then stop
if mem usage > 90% for 2 cycles then alert
if totalmem > 200.0 MB for 5 cycles then restart
if children > 50 then restart
if loadavg(5min) greater than 10 for 8 cycles then stop
if space usage > 90% then stop
if inode usage > 30000 then alert
```

## III - Consultation des résultats

### III-1 - A distance via interface web

Il faut commencer par activer le serveur http embarqué dans Monit.


/etc/monit/monitrc

```
set httpd port 2812 and
  use address localhost
# notes autorisés à se connecter
allow localhost
# authentication http
allow admin:monit
# certains utilisateurs contenus dans un fichier htpasswd
# avec des mots de passe chiffrés en md5
# accès en lecture seule
allow md5 /etc/httpd/htpasswd admins read-only
```

 Vous pouvez aussi préférer le HTTPS, auquel cas il faudra ajouter ces lignes :

/etc/monit/monitrc


```
# pour serveur web en ssl
SSL ENABLE
PEMFILE /path/to/my/cert/monit.pem
```

 Mais c'est quand même plus pratique avec une adresse web "classique" sur le port 80...  
je suis sympa, voilà comment faire :


/etc/apache2/sites-available/monit

```
<VirtualHost "monit:80">
  RewriteEngine On
  ProxyRequests Off
  ProxyPreserveHost on
  ProxyPass / http://localhost:2812/
  ProxyPassReverse / http://localhost:2812/

  <Proxy *>
    Order deny,allow
    Deny from all
    Allow from localhost
  </Proxy>
</VirtualHost>
```


 Attention, il vous faudra activer les mods d'Apache suivants : proxy.conf proxy\_http.load  
proxy.load rewrite.load

(fichiers à lier symboliquement dans ma configuration)

 Si vous souhaitez utiliser un autre répertoire que /, vous aurez des problèmes car les urls de navigation sont faites à base de "/module" dans la page générée par Monit... Personnellement, j'ai eu recours à une frame pour régler le problème.

## III-2 - En local, par la ligne de commande


Il faut utiliser la commande `/usr/sbin/monit`.

 Cet exécutable sert aussi à démarrer le démon avec différents paramètres, mais je ne l'expliquerai pas ici...

### III-2-1 - Commandes applicables à un sous-ensemble des contrôles

La syntaxe est du type : `monit <commande> <service|all>`

commande	action effectuée
start	démarrer un contrôle (charge les réglages depuis le fichier de configuration)
stop	arrêter un contrôle
restart	redémarrer un contrôle (recharge les réglages depuis le fichier de configuration)
monitor	activer un contrôle (charge les réglages depuis ce qui était en mémoire depuis le lancement du démon)
unmonitor	désactiver un contrôle

 Par ailleurs, l'option `-g` permet de spécifier que l'application de la commande ne doit se faire qu'aux contrôles appartenant à un certain groupe.

### III-2-2 - Commandes appliquées à l'ensemble des contrôles

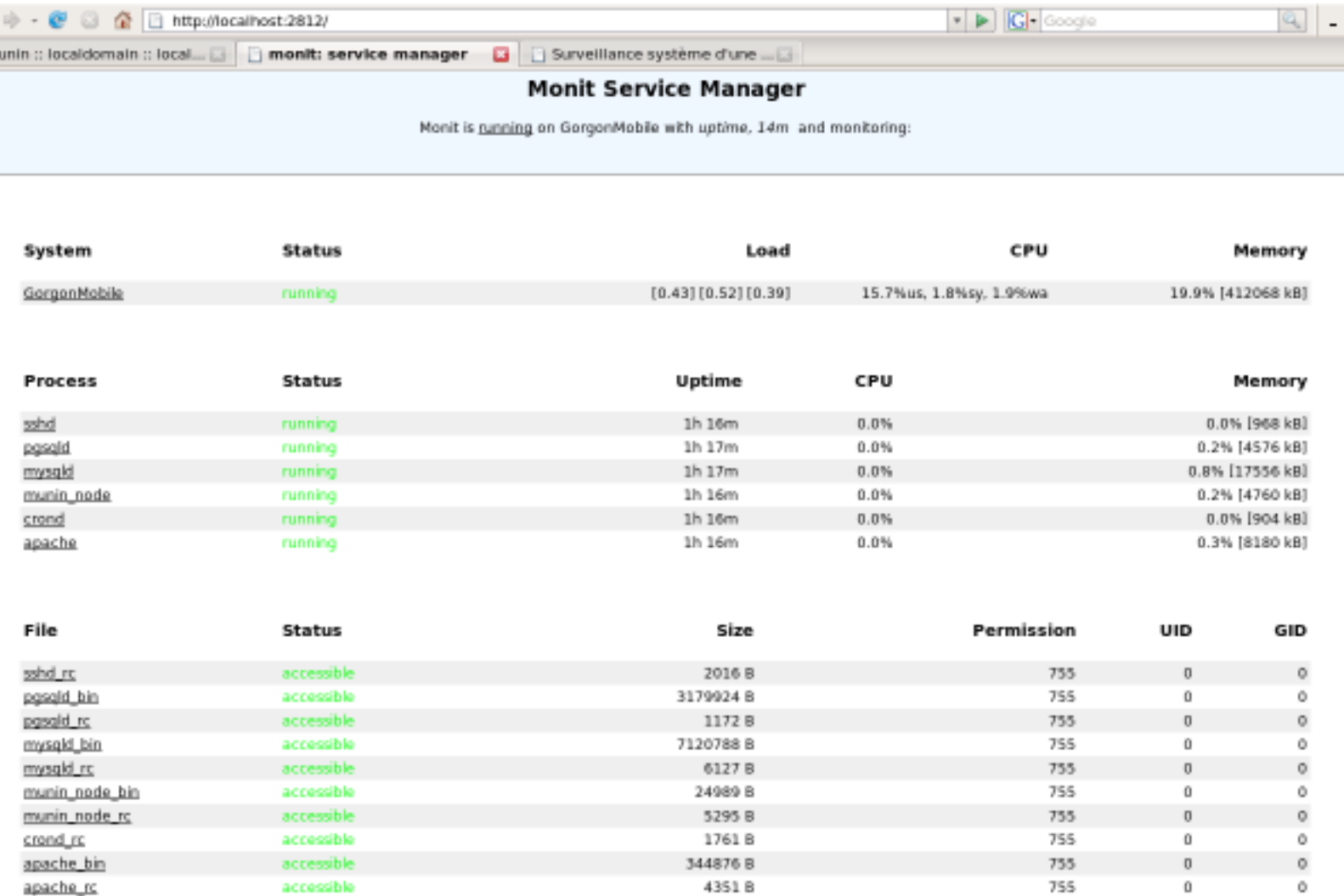
La syntaxe est du type : `monit <commande>`

commande	action effectuée
status	affiche un rapport détaillé sur chaque contrôle
summary	affiche un résumé de l'état de chaque service (actif, inactif, éteint, en marche)
validate	vérifie la configuration du démon (utile pour debugger les plugins)

## IV - Divers

### IV-1 - Vue de l'interface web

Voilà ce que devrait vous donner l'interface web de Monit...



System	Status	Load	CPU	Memory
GorgonMobile	running	[0.43] [0.52] [0.39]	15.7%us, 1.8%sy, 1.9%wa	19.9% [412068 kB]

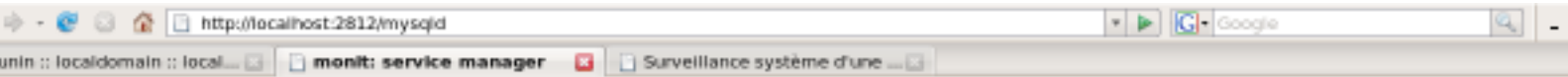
  

Process	Status	Uptime	CPU	Memory
sshd	running	1h 16m	0.0%	0.0% [968 kB]
pgsql	running	1h 17m	0.0%	0.2% [4576 kB]
mysqld	running	1h 17m	0.0%	0.8% [17556 kB]
munin_node	running	1h 16m	0.0%	0.2% [4760 kB]
crond	running	1h 16m	0.0%	0.0% [904 kB]
apache	running	1h 16m	0.0%	0.3% [8180 kB]

File	Status	Size	Permission	UID	GID
sshd_rc	accessible	2016 B	755	0	0
pgsql_bin	accessible	3179924 B	755	0	0
pgsql_rc	accessible	1172 B	755	0	0
mysqld_bin	accessible	7120788 B	755	0	0
mysqld_rc	accessible	6127 B	755	0	0
munin_node_bin	accessible	24989 B	755	0	0
munin_node_rc	accessible	5295 B	755	0	0
crond_rc	accessible	1761 B	755	0	0
apache_bin	accessible	344876 B	755	0	0
apache_rc	accessible	4351 B	755	0	0

*vue d'ensemble de l'interface*



**Process status**

Parameter	Value
Name	mysqlid
Pid file	/var/run/mysqlid/mysqlid.pid
Status	running
Group	database
Monitoring mode	active
Monitoring status	monitored
Depends on service	mysqlid_rc
Depends on service	mysqlid_bin
Start program	/etc/init.d/mysqlid start
Stop program	/etc/init.d/mysqlid stop
Check service	every 1 cycle
Timeout	If 5 restart within 5 cycles then unmonitor else if passed then alert
Data collected	Sat May 12 22:31:55 2007
Port Response time	0.002s to 127.0.0.1:3306 [MYSQL via TCP]
Process id	5404
Parent process id	5362
Process uptime	1h 23m
CPU usage	0.0%
Memory usage	0.8% [17556kB]
Children	0
Total CPU usage (incl. children)	0.0%
Total memory usage (incl. children)	0.8% [17556kB]
Port	If failed 127.0.0.1:3306 [MYSQL via TCP] with timeout 5 seconds 1 times within 1 cycle(s) then restart else if passed 1 times within 1 cycle(s) then alert
Pid	If changed 1 times within 1 cycle(s) then alert
Ppid	If changed 1 times within 1 cycle(s) then alert

*vue d'un service dans l'interface*

## IV-2 - Liens utiles

- [Site officiel de l'éditeur \(en anglais\)](#)

