Process Monitor Service - Ghid de Utilizare

Acest ghid oferă instrucțiuni pentru administratorii de sistem despre cum să instaleze, configureze, utilizeze și să depaneze Process Monitor Service.

Cuprins

1. Introducere
2. Instalare
3. Configurare
4. Utilizarea Serviciului
5. Testare
6. Depanare
7. Întrebări Frecvente

Introducere

Process Monitor Service este o soluție de monitorizare a sistemului care repornește automat procesele critice atunci când acestea sunt în alarmă. Funcționează prin monitorizarea unei baze de date MySQL pentru procesele în stare de alarmă și luarea măsurilor adecvate pentru a le reporni.

Caracteristici Principale

- Monitorizarea și repornirea automată a proceselor
- Strategii multiple de repornire (service, process, auto)
- Heath Check pentru a confirma repornirile reușite
- Model circuit breaker pentru a preveni încercările excesive de repornire
- Jurnalizare cuprinzătoare cu rotație automată
- Configurabil pentru diferite procese și medii

Instalare

Cerințe Preliminare

Înainte de a instala Process Monitor Service, asigurați-vă că sistemul dvs. îndeplinește următoarele cerințe:

- RedHat Linux sau o distribuție compatibilă
- MySQL Server 5.7 sau mai nou
- User MySQL cu privilegii minime de SELECT
- Bash 4.0 sau mai nou
- systemd

Pași de Instalare

1. **Creați Directorul Serviciului**

```
sudo mkdir -p /opt/monitor_service
```

2. **Copiați Fișierele Serviciului**

```
sudo cp monitor_service.sh config.ini /opt/monitor_service/
sudo cp monitor_service.service /etc/systemd/system/
sudo cp monitor_service.8 /usr/share/man/man8/
```

3. **Setați Permisiunile Corespunzătoare**

```
sudo chmod 755 /opt/monitor_service
sudo chmod 700 /opt/monitor_service/monitor_service.sh
sudo chmod 600 /opt/monitor_service/config.ini
sudo chown -R root:root /opt/monitor_service
```

4. **Configurați Baza de Date [Mediul de testare]**

```
# Porniți serviciul MySQL dacă nu rulează
sudo systemctl start mysqld

# Creați baza de date și tabelele
sudo mysql < setup.sql
```

5. **Configurați Serviciul**

   Editați fișierul de configurare pentru a seta serviciul:

```
sudo vim /opt/monitor_service/config.ini
```

6. **Activați și Porniți Serviciul**

```
sudo systemctl daemon-reload
sudo systemctl enable monitor_service
sudo systemctl start monitor_service
```

7. **Actualizați Baza de Date a Paginilor de Manual**

```
sudo mandb
```

Verificarea Instalării

Pentru a verifica dacă serviciul este instalat și rulează corect:

1. **Verificați Starea Serviciului**

   ```
   sudo systemctl status monitor_service
   ```

   Ar trebui să vedeți "active (running)" în rezultat.

2. **Verificați Fișierul de Jurnal**

   ```
   sudo tail -f /var/log/monitor_service.log
   ```

   Ar trebui să vedeți mesaje de pornire și mesaje periodice de verificare a bazei de date.

Configurare

Process Monitor Service este configurat prin fișierul `config.ini` localizat în `/opt/monitor_service/`.

1. Secțiunea `[database]`

Parametri pentru conectarea la baza de date MySQL.

Parametru	Descriere	Obligatoriu	Exemplu
host	Adresa serverului MySQL	Da	host = localhost
user	Utilizator MySQL	Da	user = root
password	Parola utilizatorului MySQL	Da	password = secret
database	Numele bazei de date	Da	database = v_process_monitor

2. Secțiunea `[monitor]`

Parametri globali pentru comportamentul monitorului.

Parametru	Descriere	Implicit	Exemplu
check_interval	Intervalul (secunde) între verificări	120	check_interval = 120
max_restart_failures	Număr maxim de eșecuri la restart înainte de a activa circuit breaker	3	max_restart_failures = 3
circuit_reset_time	Timp (secunde) până la resetarea circuit breaker-ului	600	circuit_reset_time = 600

3. Secțiunea `[logging]`

Configurare pentru loguri.

Parametru	Descriere	Implicit	Exemplu
max_log_size	Dimensiunea maximă a unui fișier de log (KB)	5120	max_log_size = 5120
log_files_to_keep	Număr de fișiere de log păstrate (rotație)	5	log_files_to_keep = 5

4. Secțiunea `[process.default]`

Valori implicite pentru toate procesele, dacă nu sunt suprascrise la nivel de proces.

Parametru	Descriere	Implicit	Exemplu
restart_strategy	Strategia de restart: `auto`, `service`, `process`, `custom`	auto	restart_strategy = auto
health_check_command	Comanda pentru health check (ex: `pgrep %s`, `systemctl is-active %s`)	pgrep %s	health_check_command = pgrep %s
health_check_timeout	Timeout (secunde) pentru health check	5	health_check_timeout = 5
restart_delay	Pauză (secunde) între încercări de restart	2	restart_delay = 2
max_attempts	Număr maxim de încercări de restart la o alarmă	2	max_attempts = 2
pre_restart_command	Comandă executată înainte de restart	-	pre_restart_command = /usr/local/bin/check.sh
restart_command	Comandă specifică pentru restart (folosită cu strategia custom)	-	restart_command = systemctl restart %s
system_name	Numele real al serviciului/procesului pe sistem	-	system_name = mariadb

5. Secțiuni `[process.<nume>]`

Configurare specifică pentru fiecare serviciu/proces monitorizat. Suprascrie valorile din `[process.default]`.

Parametru	Descriere	Folosit când...	Exemplu
restart_strategy	`auto`, `service`, `process`, `custom`	Pentru orice proces	restart_strategy = custom
system_name	Numele real al serviciului pe server (dacă diferă de cheie)	Folosit la `service`/`process` sau `%s`	system_name = rsyslog

Parametru	Descriere	Folosit când...	Exemplu
restart_command	Comanda custom de restart (doar pentru `custom`)	Doar dacă `restart_strategy = custom`	restart_command = systemctl restart %s
health_check_command	Comanda pentru health check	Oricând	health_check_command = systemctl is-active sshd
health_check_timeout	Timeout (secunde) pentru health check	Oricând	health_check_timeout = 5
restart_delay	Pauză (secunde) între încercări de restart	Oricând	restart_delay = 5
max_attempts	Număr maxim de încercări de restart la o alarmă	Oricând	max_attempts = 3
pre_restart_command	Comandă executată înainte de restart (opțional)	Oricând, dacă e definită	pre_restart_command = /usr/bin/precheck.sh

Explicații detaliate pentru parametri cheie

- **restart_strategy**
  - `auto`: Scriptul decide automat strategia optimă bazată pe tipul procesului.
    - Pentru servicii systemd detectate, folosește strategia `service`
    - Pentru procese simple, folosește strategia `process`
    - Recomandată pentru cazurile generale
  - `service`: Strategia pentru servicii systemd
    - Folosește `systemctl restart <system_name>`
    - Verifică starea cu `systemctl is-active`
    - Potrivită pentru servicii precum nginx, sshd, mysql
    - Exemplu:

```
[process.nginx_web]
restart_strategy = service
system_name = nginx
health_check_command = systemctl is-active nginx
```

  - `process`: Strategia pentru procese simple
    - Folosește `pkill` pentru oprire și execuție directă pentru pornire
    - Verifică starea cu `pgrep`
    - Potrivită pentru daemons și procese de fundal
    - Exemplu:

```
[process.crond]
restart_strategy = process
system_name = cron
health_check_command = pgrep cron
```

- `custom`: Strategia pentru cazuri speciale
  - Execută comanda definită în `restart_command`
  - Oferă control total asupra procesului de restart
  - Necesită definirea explicită a comenzii de restart
  - Exemplu:

```
[process.custom_app]
restart_strategy = custom
restart_command = /usr/local/bin/custom_restart.sh %s
health_check_command = curl -s http://localhost:8080/health
```

- **system_name**
  - Numele real al serviciului/procesului pe sistem
  - Folosit în următoarele situații:
    1. Cu strategia `service`: Pentru numele corect al serviciului systemd
    2. Cu strategia `process`: Pentru identificarea procesului în sistem
    3. În comenzi custom: Înlocuiește %s în comenzi
  - Exemple:

```
# Serviciu cu nume diferit
[process.mysql_database]
restart_strategy = service
system_name = mariadb

# Proces cu nume specific
[process.web_app]
restart_strategy = process
system_name = myapp-server
```

- **restart_command**
  - Obligatoriu pentru strategia `custom`
  - Poate include următoarele placeholdere:
    - %s: Înlocuit cu system_name sau numele procesului
    - %d: Înlocuit cu directorul curent al procesului
  - Exemple practice:

```
```

```
# Restart Docker container
restart_command = docker restart %s

# Restart cu script custom
restart_command = /usr/local/bin/restart_app.sh %s

# Restart complex
restart_command = cd %d && ./stop.sh && ./start.sh
```

- **health_check_command**
  - Verifică starea procesului după restart
  - Trebuie să returneze cod 0 pentru succes
  - Exemple pentru diferite tipuri de verificări:

```
# Verificare serviciu systemd
health_check_command = systemctl is-active %s

# Verificare proces
health_check_command = pgrep -f %s

# Verificare endpoint HTTP
health_check_command = curl -sf http://localhost:8080/health

# Verificare port
health_check_command = netstat -tulpn | grep %s
```

- **pre_restart_command**
  - Execută verificări sau pregătiri înainte de restart
  - Util pentru:
    1. Salvare stare sau backup
    2. Oprire servicii dependente
    3. Verificări de sistem
  - Exemple practice:

```
# Backup înainte de restart
pre_restart_command = /usr/local/bin/backup_db.sh

# Verificare dependențe
pre_restart_command = systemctl is-active dependency1 &&
systemctl is-active dependency2

# Curățare cache
pre_restart_command = rm -rf /tmp/cache/*
```

Exemple de configurare

1. Serviciu clasic systemd

```
[process.sshd]
restart_strategy = service
system_name = sshd
health_check_command = systemctl is-active sshd
```

2. Serviciu cu nume diferit pe server

```
[process.rsyslogd]
restart_strategy = service
system_name = rsyslog
health_check_command = systemctl is-active rsyslog
```

3. Restart custom cu script extern

```
[process.myapp]
restart_strategy = custom
restart_command = /usr/local/bin/restart_myapp.sh %s
system_name = myapp_service
health_check_command = pgrep myapp_service
```

4. Restart direct proces

```
[process.crond]
restart_strategy = process
system_name = cron
health_check_command = pgrep cron
```

Considerații importante pentru configurare

1. **Ierarhia parametrilor**
   - Parametrii definiți în secțiunea specifică a procesului (`[process.<nume>]`) au prioritate
   - Dacă un parametru lipsește, se folosește valoarea din `[process.default]`
   - Dacă parametrul nu există nici în `[process.default]`, se folosesc valorile implicite ale sistemului
2. **Interacțiunea dintre parametri**

- Pentru strategia `custom`: `restart_command` este obligatoriu
- Pentru strategia `service`: `health_check_command` ar trebui să folosească `systemctl is-active`
- Pentru strategia `process`: Se recomandă folosirea `pgrep` pentru health check

3. **Securitate și permisiuni**
   - Comenzile specificate în `restart_command` și `pre_restart_command` trebuie să fie executabile
   - Scripturile custom trebuie să aibă permisiunile corecte (cel puțin 700)
   - Se recomandă folosirea căilor absolute în toate comenzile

4. **Best Practices**
   - Testați configurația nouă într-un mediu de dezvoltare înainte de producție
   - Folosiți timeouts rezonabile pentru health checks (5-10 secunde recomandat)
   - Păstrați comenzile de restart cât mai simple și robuste posibil
   - Verificați jurnalele după modificarea configurației

Utilizarea Serviciului

Gestionarea de Bază a Serviciului

- **Pornirea Serviciului**

```
sudo systemctl start monitor_service
```

- **Oprirea Serviciului**

```
sudo systemctl stop monitor_service
```

- **Repornirea Serviciului**

```
sudo systemctl restart monitor_service
```

- **Verificarea Stării Serviciului**

```
sudo systemctl status monitor_service
```

Vizualizarea Jurnalelor

- **Vizualizarea Jurnalelor Serviciului în Journal**

```
```

```
sudo journalctl -u monitor_service -f
```

- **Vizualizarea Fișierului Jurnal al Serviciului**

```
sudo tail -f /var/log/monitor_service.log
```

Pagina de Manual

Serviciul include o pagină de manual care poate fi accesată folosind:

```
man monitor_service
```

```
MONITOR_SERVICE(8)      System Administration Utilities
MONITOR_SERVICE(8)

NAME
       monitor_service - Monitorizare și restart automat pentru procese
Linux

SYNOPSIS
       monitor_service.sh [ --status ] [ --restart process ] [ --help ]

DESCRIPTION
       monitor_service.sh este un serviciu care monitorizează procesele
critice
       definite într-o bază de date MySQL și le repornește automat dacă
intră
       în stare de alarmă. Suportă strategii flexibile de restart, health
check,
       rotație loguri și circuit breaker.

OPTIONS
       --status
              Afișează procesele aflate în stare de alarmă (alarma=1,
sound=0).

       --restart process
              Forțează restart pentru procesul specificat.

       --help
              Afișează acest mesaj de ajutor.

CONFIGURATION
       Configurația se face în fișierul INI
/opt/monitor_service/config.ini, cu
       secțiuni pentru:
```

```
          - [database]: conexiune MySQL
          - [monitor]: intervale și circuit breaker
          - [logging]: rotație loguri
          - [process.<nume>]: configurare per proces

        Exemplu minimal:
          [database]
          host = localhost
          user = root
          password = ...
          database = v_process_monitor

          [monitor]
          check_interval = 120
          max_restart_failures = 3
          circuit_reset_time = 600

          [logging]
          max_log_size = 5120
          log_files_to_keep = 5

          [process.sshd]
          restart_strategy = service
          health_check_command = systemctl is-active sshd

EXAMPLES
        Pornește serviciul ca daemon:
          sudo systemctl start monitor_service

        Afișează procesele în alarmă:
          ./monitor_service.sh --status

        Forțează restart pentru sshd:
          ./monitor_service.sh --restart sshd

        Afișează ajutorul:
          ./monitor_service.sh --help

FILES
        /opt/monitor_service/config.ini
               Fișierul principal de configurare.

        /var/log/monitor_service.log
               Fișierul de log.

        /etc/systemd/system/monitor_service.service
               Unitatea systemd pentru pornire automată.

AUTHOR
        AD

SEE ALSO
        systemctl(1), mysql(1), pkill(1), journalctl(1)
```

Utilizare din linia de comandă (CLI)

Process Monitor Service poate fi folosit și pentru acțiuni punctuale direct din linia de comandă, fără a porni bucla de monitorizare. Sunt disponibile următoarele opțiuni:

Opțiune	Descriere
--status	Afișează procesele aflate în stare de alarmă (alarma=1, sound=0).
--restart <nume_proces>	Forțează restart pentru procesul specificat.
--help	Afișează acest mesaj de ajutor.

Exemple de utilizare:

- Afișează procesele în alarmă:

```
./monitor_service.sh --status
```

Output exemplu:

```
Procese în stare de alarmă (alarma=1, sound=0):
ID: 2 | Nume: sshd | Notes: restart failed
ID: 5 | Nume: apache2 | Notes: config error
```

- Forțează restart pentru un proces (ex: sshd):

```
./monitor_service.sh --restart sshd
```

Output exemplu:

```
Forțez restart pentru procesul: sshd
Restart reușit pentru sshd.
```

Sau, în caz de eroare:

```
Forțez restart pentru procesul: sshd
Eroare la restart pentru sshd.
```

- Afișează ajutorul pentru CLI:

```
./monitor_service.sh --help
```

Output exemplu:

```
Usage: ./monitor_service.sh [--status] [--restart <process_name>] [--
help]
  --status               Afișează procesele aflate în stare de
alarmă.
  --restart <process>    Forțează restart pentru procesul specificat.
  --help                 Afișează acest mesaj de ajutor.
```

Dacă nu se specifică niciun argument, scriptul rulează în modul de monitorizare continuă (comportamentul implicit).

Testare

Serviciul include scripturi de testare pentru a vă ajuta să verificați funcționalitatea sa.

Testare Interactivă cu test_alarm.sh

Scriptul `test_alarm.sh` oferă o modalitate interactivă de a testa serviciul de monitorizare:

```
chmod +x test_alarm.sh
./test_alarm.sh
```

Acest script vă permite să:

- Vizualizați toate serviciile monitorizate
- Setați servicii specifice în stare de alarmă
- Setați toate serviciile în stare de alarmă
- Resetați toate alarmele

Flux de Testare

1. Utilizați unul dintre scripturile de testare pentru a seta un serviciu în stare de alarmă
2. Verificați dacă serviciul de monitorizare detectează alarma și încearcă să repornească serviciul
3. Verificați jurnalele pentru a confirma încercarea de repornire și rezultatul acesteia
4. Verificați dacă starea de alarmă este ștearsă din baza de date după repornirea cu succes

Depanare

Probleme Comune și Soluții

Serviciul Nu Pornește

- **Verificați Fișierele de Jurnal**

```
sudo journalctl -u monitor_service
sudo cat /var/log/monitor_service.log
```

Căutați mesaje de eroare care ar putea indica cauza problemei.

- **Verificați Permisiunile Fișierelor**

```
sudo ls -la /opt/monitor_service/
```

Asigurați-vă că scriptul este executabil și fișierul de configurare are permisiunile corecte.

- **Verificați Configurația**

```
sudo cat /opt/monitor_service/config.ini
```

Verificați dacă fișierul de configurare este formatat corect și conține setări valide.

Probleme de Conexiune la Baza de Date

- **Verificați Serviciul MySQL**

```
sudo systemctl status mysqld
```

Asigurați-vă că MySQL rulează.

Înțelegerea Mesajelor de Jurnal

Fișierul de jurnal (**/var/log/monitor_service.log**) conține informații detaliate despre operațiunea serviciului:

- Mesajele de nivel **INFO** indică operarea normală
- Mesajele de nivel **WARNING** indică probleme potențiale
- Mesajele de nivel **ERROR** indică eșecuri care necesită atenție
- Mesajele de nivel **DEBUG** oferă informații detaliate pentru depanare

Exemple de Mesaje de Jurnal

- **Pornirea Serviciului**

```
2023-06-15 10:00:00 - INFO - Starting Process Monitor Service
```

- **Verificarea Bazei de Date**

```
2023-06-15 10:05:00 - INFO - Found process in alarm: apache2 (ID: 1)
```

- **Încercare de Repornire**

```
2023-06-15 10:05:01 - INFO - RESTART LOG: Beginning restart procedure
for apache2 (strategy: service)
```

- **Repornire cu Succes**

```
2023-06-15 10:05:10 - INFO - RESTART LOG: Successfully restarted and
verified service: apache2
```

- **Eșec de Repornire**

```
2023-06-15 10:05:10 - ERROR - RESTART LOG: Service restart command
succeeded but health check failed for: apache2
```

- **Circuit Breaker**

```
2023-06-15 10:15:10 - WARNING - Circuit breaker opened for apache2
after 3 failures
```

Întrebări Frecvente

Cum știe serviciul ce procese să monitorizeze?

Serviciul monitorizează procesele listate în tabelul **PROCESE** din baza de date MySQL. Fiecare proces are o intrare corespunzătoare în tabelul **STATUS_PROCESS** care îi urmărește starea de alarmă.

Cum decide serviciul când să repornească un proces?

Serviciul verifică tabelul **STATUS_PROCESS** pentru procesele cu **alarma=1** şi **sound=0**. Când găsește un proces în stare de alarmă care nu este bifat, încearcă să-l repornească folosind strategia configurată.

Ce este modelul circuit breaker?

Modelul **circuit breaker** previne încercările excesive de repornire pentru procesele care eșuează. După un număr configurabil de eșecuri, circuit breaker-ul "se deschide" și blochează încercările ulterioare de repornire pentru o perioadă de timp. Acest lucru previne epuizarea resurselor și eșecurile în cascadă.

Cum pot adăuga health checkuri personalizate?

Puteți adăuga health check personalizat configurând parametrul health_check_command pentru un proces. Această comandă ar trebui să returneze codul de ieșire 0 dacă procesul este ON, sau non-zero dacă este OFF.

Ce se întâmplă dacă baza de date MySQL este oprită?

Dacă baza de date MySQL este oprită, serviciul va înregistra o eroare și va continua să încerce să se conecteze la intervalul de verificare configurat. Nu va putea detecta sau reporni procese până când conexiunea la baza de date nu este restabilită.

Cum pot schimba intervalul de verificare?

Editați fișierul config.ini și actualizați parametrul check_interval din secțiunea [monitor]. Valoarea este în secunde.

```
[monitor]
check_interval = 300  # 5 minutes
```