superlance Documentation

Release 1.0.0

Chris McDonough, Agendaless Consulting, Inc.

Jun 18, 2017

Contents

Superlance is a package of plugin utilities for monitoring and controlling processes that run under Supervisor. It provides these plugins:

httpok This plugin is meant to be used as a supervisor event listener, subscribed to `TICK_*` events. It tests that a given child process which must in the `RUNNING` state, is viable via an HTTP `GET` request to a configured URL. If the request fails or times out, **httpok** will restart the "hung" child process.

crashmail This plugin is meant to be used as a supervisor event listener, subscribed to `PROCESS_STATE_EXITED` events. It email a user when a process enters the `EXITED` state unexpectedly.

memmon This plugin is meant to be used as a supervisor event listener, subscribed to `TICK_*` events. It monitors memory usage for configured child processes, and restarts them when they exceed a configured maximum size.

crashmailbatch Similar to **crashmail**, **crashmailbatch** sends email alerts when processes die unexpectedly. The difference is that all alerts generated within the configured time interval are batched together to avoid sending too many emails.

fatalmailbatch This plugin sends email alerts when processes fail to start too many times such that supervisord gives up retrying. All of the fatal start events generated within the configured time interval are batched together to avoid sending too many emails.

crashsms Similar to **crashmailbatch** except it sends SMS alerts through an email gateway. Messages are formatted to fit in SMS.

Contents:

CHAPTER 1

httpok Documentation

httpok is a supervisor "event listener" which may be subscribed to a concrete `TICK_5`, `TICK_60` or `TICK_3600` event. When **httpok** receives a `TICK_x` event (`TICK_60` is recommended, indicating activity every 60 seconds), **httpok** makes an HTTP GET request to a confgured URL. If the request fails or times out, **httpok** will restart the "hung" child process(es). **httpok** can be configured to send an email notification when it restarts a process.

httpok can only monitor the process status of processes which are **supervisord** child processes.

httpok is a "console script" installed when you install `superlance`. Although **httpok** is an executable program, it isn't useful as a general-purpose script: it must be run as a **supervisor** event listener to do anything useful.

Command-Line Syntax

```
$ httpok [-p processname] [-a] [-g] [-t timeout] [-c status_code] \
         [-b inbody] [-m mail_address] [-s sendmail] URL
```

-p <process_name>, **--program**=<process_name>
: Restart the **supervisord** child process named `process_name` if it is in the `RUNNING` state when the URL returns an unexpected result or times out.

  This option can be provided more than once to have **httpok** monitor more than one process.

  To monitor a process which is part of a **supervisord** group, specify its name as `group_name:process_name`.

-a, --any
: Restart any child of **supervisord** in the `RUNNING` state if the URL returns an unexpected result or times out.

  Overrides any `-p` parameters passed in the same **httpok** process invocation.

-g <gcore_program>, **--gcore**=<gcore_program>
: Use the specifed program to `gcore` the **supervisord** child process. The program should accept two arguments on the command line: a filename and a pid. Defaults to `/usr/bin/gcore -o`.

-d <core_directory>, **--coredir**=<core_directory>
If a core directory is specified, **httpok** will try to use the gcore program (see -g) to write a core file into this directory for each hung process before restarting it. It will then append any gcore stdout output to the email message, if mail is configured (see the -m option below).

-t <timeout>, **--timeout**=<timeout>
The number of seconds that **httpok** should wait for a response to the HTTP request before timing out.

If this timeout is exceeded, **httpok** will attempt to restart child processes which are in the RUNNING state, and specified by -p or -a.

Defaults to 10 seconds.

-c <http_status_code>, **--code**=<http_status_code>
Specify the expected HTTP status code for the configured URL.

If this status code is not the status code provided by the response, **httpok** will attempt to restart child processes which are in the RUNNING state, and specified by -p or -a.

Defaults to 200.

-b <body_string>, **--body**=<body_string>
Specify a string which should be present in the body resulting from the GET request.

If this string is not present in the response, **httpok** will attempt to restart child processes which are in the RUNNING state, and specified by -p or -a.

The default is to ignore the body.

-s <sendmail_command>, **--sendmail_program**=<sendmail_command>
Specify the sendmail command to use to send email.

Must be a command which accepts header and message data on stdin and sends mail. Default is /usr/sbin/sendmail -t -i.

-m <email_address>, **--email**=<email_address>
Specify an email address to which notification messages are sent. If no email address is specified, email will not be sent.

-e, --eager
Enable "eager" monitoring: check the URL and emit mail even if no monitored child process is in the RUNNING state.

Enabled by default.

-E, --not-eager
Disable "eager" monitoring: do not check the URL or emit mail if no monitored process is in the RUNNING state.

URL
The URL to which to issue a GET request.

-n <httpok name>, **--name**=<httpok name>
An optional name that identifies this httpok process. If given, the email subject will start with httpok [<httpok name>]: instead of httpok: In case you run multiple supervisors on a single host that control different processes with the same name (eg *zopeinstance1*) you can use this option to indicate which project the restarted instance belongs to.

Configuring `httpok` Into the Supervisor Config

An `[eventlistener:x]` section must be placed in `supervisord.conf` in order for **`httpok`** to do its work. See the "Events" chapter in the Supervisor manual for more information about event listeners.

The following example assumes that **`httpok`** is on your system `PATH`.

```
[eventlistener:httpok]
command=httpok -p program1 -p group1:program2 http://localhost:8080/tasty
events=TICK_60
```

CHAPTER 2

crashmail Documentation

crashmail is a supervisor "event listener", intended to be subscribed to PROCESS_STATE_EXITED events. When **crashmail** receives that event, and the transition is "unexpected", **crashmail** sends an email notification to a configured address.

crashmail is incapable of monitoring the process status of processes which are not **supervisord** child processes.

crashmail is a "console script" installed when you install superlance. Although **crashmail** is an executable program, it isn't useful as a general-purpose script: it must be run as a **supervisor** event listener to do anything useful.

Command-Line Syntax

```
$ crashmail [-p processname] [-a] [-o string] [-m mail_address] \
            [-s sendmail]
```

-p <process_name>, **--program**=<process_name>
Send mail when the specified **supervisord** child process transitions unexpectedly to the EXITED state.

This option can be provided more than once to have **crashmail** monitor more than one program.

To monitor a process which is part of a **supervisord** group, specify its name as group_name:process_name.

-a, **--any**
Send mail when any **supervisord** child process transitions unexpectedly to the EXITED state.

Overrides any -p parameters passed in the same **crashmail** process invocation.

-o <prefix>, **--optionalheader**=<prefix>
Specify a parameter used as a prefix in the mail *Subject* header.

-s <sendmail_command>, **--sendmail_program**=<sendmail_command>
Specify the sendmail command to use to send email.

Must be a command which accepts header and message data on stdin and sends mail. Default is `/usr/sbin/sendmail -t -i`.

-m <email_address>, **--email**=<email_address>
Specify an email address to which crash notification messages are sent. If no email address is specified, email will not be sent.

Configuring `crashmail` Into the Supervisor Config

An `[eventlistener:x]` section must be placed in `supervisord.conf` in order for **`crashmail`** to do its work. See the "Events" chapter in the Supervisor manual for more information about event listeners.

The following example assumes that **`crashmail`** is on your system `PATH`.

```
[eventlistener:crashmail]
command=crashmail -p program1 -p group1:program2 -m dev@example.com
events=PROCESS_STATE_EXITED
```

CHAPTER 3

memmon Overview

memmon is a supervisor "event listener" which may be subscribed to a concrete `TICK_x` event. When **memmon** receives a `TICK_x` event (`TICK_60` is recommended, indicating activity every 60 seconds), **memmon** checks that a configurable list of programs (or all programs running under supervisor) are not exceeding a configurable amount of memory (resident segment size, or RSS). If one or more of these processes is consuming more than the amount of memory that **memmon** believes it should, **memmon** will restart the process. **memmon** can be configured to send an email notification when it restarts a process.

memmon is known to work on Linux and Mac OS X, but has not been tested on other operating systems (it relies on **ps** output and command-line switches).

memmon is incapable of monitoring the process status of processes which are not **supervisord** child processes. Without the *–cumulative* option, only the RSS of immediate children of the **supervisord** process will be considered.

memmon is a "console script" installed when you install `superlance`. Although **memmon** is an executable program, it isn't useful as a general-purpose script: it must be run as a **supervisor** event listener to do anything useful.

memmon uses Supervisor's XML-RPC interface. Your `supervisord.conf` file must have a valid [unix_http_server] or [inet_http_server] section, and must have an [rpcinterface:supervisor] section. If you are able to control your `supervisord` instance with `supervisorctl`, you have already met these requirements.

Command-Line Syntax

```
$ memmon [-c] [-p processname=byte_size] [-g groupname=byte_size] \
         [-a byte_size] [-s sendmail] [-m email_address] \
         [-u email_uptime_limit] [-n memmon_name]
```

-h, --help
Show program help.

-c, --cumulative
Check against cumulative RSS. When calculating a process' RSS, also consider its child processes. With this option *memmon* will sum up the RSS of the process to be monitored and all its children.

-p <name/size pair>, **--program**=<name/size pair>
A name/size pair, e.g. "foo=1MB". The name represents the supervisor program name that you would like **memmon** to monitor; the size represents the number of bytes (suffix-multiplied using "KB", "MB" or "GB") that should be considered "too much".

This option can be provided more than once to have **memmon** monitor more than one program.

Programs can be specified using a "namespec", to disambiguate same-named programs in different groups, e.g. `foo:bar` represents the program `bar` in the `foo` group.

-g <name/size pair>, **--groupname**=<name/size pair>
A groupname/size pair, e.g. "group=1MB". The name represents the supervisor group name that you would like **memmon** to monitor; the size represents the number of bytes (suffix-multiplied using "KB", "MB" or "GB") that should be considered "too much".

Multiple `-g` options can be provided to have **memmon** monitor more than one group. If any process in this group exceeds the maximum, it will be restarted.

-a <size>, **--any**=<size>
A size (suffix-multiplied using "KB", "MB" or "GB") that should be considered "too much". If any program running as a child of supervisor exceeds this maximum, it will be restarted. E.g. 100MB.

-s <command>, **--sendmail**=<command>
A command that will send mail if passed the email body (including the headers). Defaults to `/usr/sbin/sendmail -t -i`.

Note: Specifying this option doesn't cause memmon to send mail by itself: see the `-m` / `--email` option.

-m <email address>, **--email**=<email address>
An email address to which to send email when a process is restarted. By default, memmon will not send any mail unless an email address is specified.

-u <email uptime limit>, **--uptime**=<email uptime limit>
Only send an email in case the restarted process' uptime (in seconds) is below this limit. (Useful to only get notified if a processes gets restarted too frequently)

Uptime is given in seconds (suffix-multiplied using "m" for minutes, "h" for hours or "d" for days)

-n <memmon name>, **--name**=<memmon name>
An optional name that identifies this memmon process. If given, the email subject will start with `memmon [<memmon name>]:` instead of `memmon:` In case you run multiple supervisors on a single host that control different processes with the same name (eg *zopeinstance1*) you can use this option to indicate which project the restarted instance belongs to.

Configuring `memmon` Into the Supervisor Config

An `[eventlistener:x]` section must be placed in `supervisord.conf` in order for **memmon** to do its work. See the "Events" chapter in the Supervisor manual for more information about event listeners.

The following examples assume that **memmon** is on your system `PATH`.

Example Configuration 1

This configuration causes **memmon** to restart any process which is a child of **supervisord** consuming more than 200MB of RSS, and will send mail to `bob@example.com` when it restarts a process using the default **sendmail**

command.

```
[eventlistener:memmon]
command=memmon -a 200MB -m bob@example.com
events=TICK_60
```

Example Configuration 2

This configuration causes **memmon** to restart any process with the supervisor program name `foo` consuming more than 200MB of RSS, and will send mail to `bob@example.com` when it restarts a process using the default sendmail command.

```
[eventlistener:memmon]
command=memmon -p foo=200MB -m bob@example.com
events=TICK_60
```

Example Configuration 3

This configuration causes **memmon** to restart any process in the process group "bar" consuming more than 200MB of RSS, and will send mail to `bob@example.com` when it restarts a process using the default **sendmail** command.

```
[eventlistener:memmon]
command=memmon -g bar=200MB -m bob@example.com
events=TICK_60
```

Example Configuration 4

This configuration causes **memmon** to restart any process meeting the same requirements as in *Example Configuration 2* with one difference:

The email will only be sent if the process' uptime is less or equal than 2 days (172800 seconds)

```
[eventlistener:memmon]
command=memmon -p foo=200MB -m bob@example.com -u 2d
events=TICK_60
```

CHAPTER 4

crashmailbatch Documentation

crashmailbatch is a supervisor "event listener", intended to be subscribed to `PROCESS_STATE` and `TICK_60` events. It monitors all processes running under a given supervisord instance.

Similar to **crashmail**, **crashmailbatch** sends email alerts when processes die unexpectedly. The difference is that all alerts generated within the configured time interval are batched together to avoid sending too many emails.

crashmailbatch is a "console script" installed when you install `superlance`. Although **crashmailbatch** is an executable program, it isn't useful as a general-purpose script: it must be run as a **supervisor** event listener to do anything useful.

Command-Line Syntax

```
$ crashmailbatch --toEmail=<email address> --fromEmail=<email address> \
        [--interval=<batch interval in minutes>] [--subject=<email subject>] \
                [--tickEvent=<event name>]
```

-t <destination email>, **--toEmail**=<destination email>
Specify an email address to which crash notification messages are sent.

-f <source email>, **--fromEmail**=<source email>
Specify an email address from which crash notification messages are sent.

-i <interval>, **--interval**=<interval>
Specify the time interval in minutes to use for batching notifcations. Defaults to 1.0 minute.

-s <email subject>, **--subject**=<email subject>
Override the email subject line. Defaults to "Crash alert from supervisord"

-e <event name>, **--tickEvent**=<event name>
Override the TICK event name. Defaults to "TICK_60"

Configuring `crashmailbatch` Into the Supervisor Config

An `[eventlistener:x]` section must be placed in `supervisord.conf` in order for **`crashmailbatch`** to do its work. See the "Events" chapter in the Supervisor manual for more information about event listeners.

The following example assumes that **`crashmailbatch`** is on your system `PATH`.

```
[eventlistener:crashmailbatch]
command=crashmailbatch --toEmail="alertme@fubar.com" --fromEmail="supervisord@fubar.
↪com"
events=PROCESS_STATE,TICK_60
```

CHAPTER 5

fatalmailbatch Documentation

fatalmailbatch is a supervisor "event listener", intended to be subscribed to `PROCESS_STATE` and `TICK_60` events. It monitors all processes running under a given supervisord instance.

fatalmailbatch sends email alerts when processes fail to start too many times such that supervisord gives up retrying. All of the fatal start events generated within the configured time interval are batched together to avoid sending too many emails.

fatalmailbatch is a "console script" installed when you install `superlance`. Although **fatalmailbatch** is an executable program, it isn't useful as a general-purpose script: it must be run as a **supervisor** event listener to do anything useful.

Command-Line Syntax

```
$ fatalmailbatch --toEmail=<email address> --fromEmail=<email address> \
        [--interval=<batch interval in minutes>] [--subject=<email subject>]
```

-t <destination email>, **--toEmail**=<destination email>
Specify an email address to which fatal start notification messages are sent.

-f <source email>, **--fromEmail**=<source email>
Specify an email address from which fatal start notification messages are sent.

-i <interval>, **--interval**=<interval>
Specify the time interval in minutes to use for batching notifcations. Defaults to 1 minute.

-s <email subject>, **--subject**=<email subject>
Override the email subject line. Defaults to "Fatal start alert from supervisord"

Configuring `fatalmailbatch` Into the Supervisor Config

An `[eventlistener:x]` section must be placed in `supervisord.conf` in order for **`fatalmailbatch`** to do its work. See the "Events" chapter in the Supervisor manual for more information about event listeners.

The following example assumes that **`fatalmailbatch`** is on your system `PATH`.

```
[eventlistener:fatalmailbatch]
command=fatalmailbatch --toEmail="alertme@fubar.com" --fromEmail="supervisord@fubar.
↪com"
events=PROCESS_STATE,TICK_60
```

CHAPTER 6

crashsms Documentation

crashsms is a supervisor "event listener", intended to be subscribed to PROCESS_STATE events and TICK events such as TICK_60. It monitors all processes running under a given supervisord instance.

Similar to **crashmailbatch**, **crashsms** sends SMS alerts through an email gateway. Messages are formatted to fit in SMS

crashsms is a "console script" installed when you install superlance. Although **crashsms** is an executable program, it isn't useful as a general-purpose script: it must be run as a **supervisor** event listener to do anything useful.

Command-Line Syntax

```
$ crashsms --toEmail=<email address> --fromEmail=<email address> \
        [--interval=<batch interval in minutes>] [--subject=<email subject>] \
                [--tickEvent=<event name>]
```

-t <destination email>, **--toEmail**=<destination email>
Specify an email address to which crash notification messages are sent.

-f <source email>, **--fromEmail**=<source email>
Specify an email address from which crash notification messages are sent.

-i <interval>, **--interval**=<interval>
Specify the time interval in minutes to use for batching notifcations. Defaults to 1.0 minute.

-s <email subject>, **--subject**=<email subject>
Set the email subject line. Default is None

-e <event name>, **--tickEvent**=<event name>
Override the TICK event name. Defaults to "TICK_60"

Configuring `crashsms` Into the Supervisor Config

An `[eventlistener:x]` section must be placed in `supervisord.conf` in order for **`crashsms`** to do its work. See the "Events" chapter in the Supervisor manual for more information about event listeners.

The following example assumes that **`crashsms`** is on your system `PATH`.

```
[eventlistener:crashsms]
command=crashsms --toEmail="<mobile number>@<sms email gateway>" --fromEmail=
↪"supervisord@fubar.com"
events=PROCESS_STATE,TICK_60
```

CHAPTER 7

Resources and Development

Mailing Lists

Superlance does not have a dedicated mailing list but the main Supervisor-users mailing list can be used for discussing Superlance. This is a good place to ask questions about installing or using Superlance.

Bug Tracker

Superlance has a bug tracker where you may report any bugs or other errors you find. Please report bugs to the GitHub issues page.

Version Control Repository

You can also view the Superlance version control repository.

Contributing

Pull requests can be submitted to the Superlance repository on GitHub.

In the time since Superlance was created, there are now many third party plugins for Supervisor. Most new plugins should be in their own package rather than added to Superlance.

Author Information

The following people are responsible for creating Superlance.

Original Author

Chris McDonough is the original author of Superlance.

Contributors

Contributors are tracked on the GitHub contributions page.

The list below is included for historical reasons. It records contributors who signed a legal agreement. The legal agreement was introduced in January 2014 but later withdrawn in April 2014. This list is being preserved in case it is useful later (e.g. if at some point there was a desire to donate the project to a foundation that required such agreements).

- Chris McDonough, 2008-09-18
- Tres Seaver, 2009-02-11
- Roger Hoover, 2010-07-30
- Joaquín Cuenca Abela, 2011-06-23
- Harald Friessnegger, 2012-11-01
- Mikhail Lukyanchenko, 2013-12-23
- Patrick Gerken, 2014-01-27

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index

Symbols

C

E

F

H

M

P

U