

Writing Nagios plugins with Perl help

Jose Luis Martinez

JLMARTIN

jlmartinez@capside.com

www.pplusdomain.net

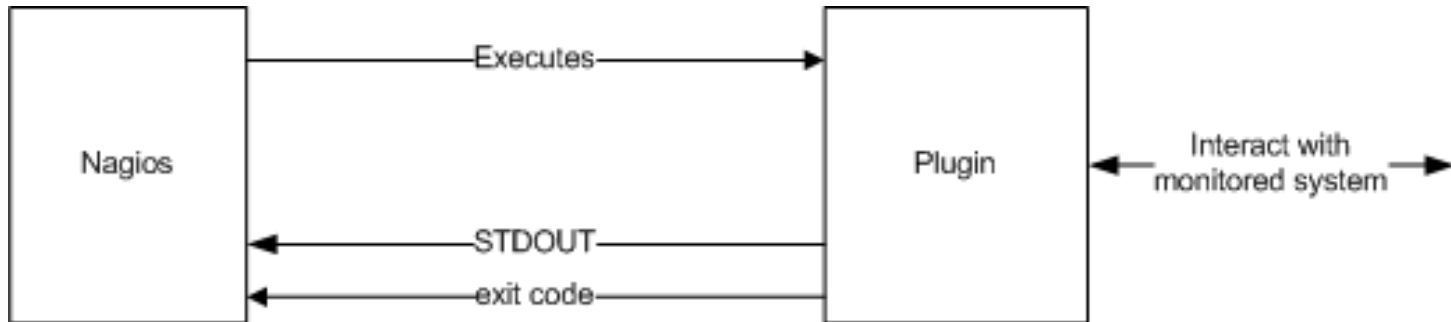
Nagios

What is Nagios?

Nagios

- Monitoring tool that doesn't know how to monitor anything!?!?
 - The real monitoring is done by plugins
 - Just external programs with a defined interface to communicate with Nagios
 - Written in any language

Plugins



```
$ ./check_mysql_performance  
MYSQL_PERFORMANCE OK - All parameters OK | Connections=0.3;; Max_used_connections=6;;
```

Service State Information

Current Status:	OK (for 0d 0h 5m 21s)
Status Information:	MYSQL_PERFORMANCE OK - All parameters OK
Performance Data:	Connections=0.437710437710438;; Max_used_connections=6;;
Current Attempt:	1/3 (HARD state)
Last Check Time:	30-07-2010 02:12:08
Check Type:	ACTIVE
Check Latency / Duration:	0.239 / 0.755 seconds
Next Scheduled Check:	30-07-2010 02:17:08
Last State Change:	30-07-2010 02:07:08
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	30-07-2010 02:12:26 (0d 0h 0m 3s ago)

Writing your plugins

First rule

- Is it already done?
 - www.nagiosplugins.org
 - Nagios official plugins
 - www.monitoringexchange.org
 - User contributed
 - exchange.nagios.org
 - User contributed
 - Google “xxx nagios”

We'll take a look at

- Nagios::Plugin
- Nagios::Plugin::DieNicely
- Nagios::Plugin::WWW::Mechanize
- Nagios::Plugin::SNMP
- *Nagios::Plugin::Differences*

Nagios::Plugin

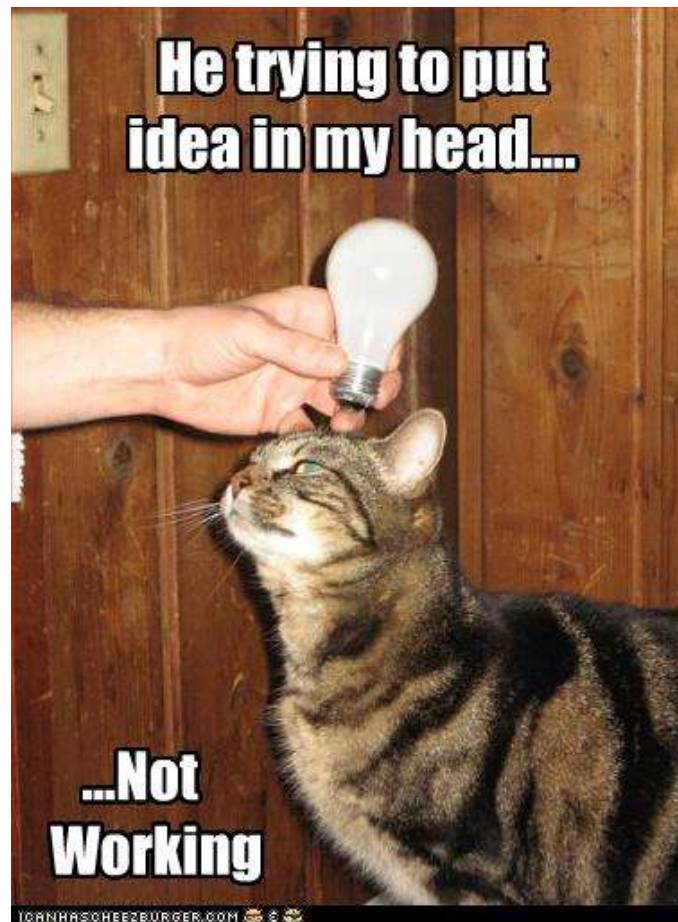
- Lots of common functionality for free!
 - When writing a plugin you have to
 - Handle arguments
 - Get and manipulate the data
 - Calculate state (OK, WARNING, CRITICAL)
 - Output performance data

Nagios::Plugin

- Lots of common functionality for free!
 - When writing a good plugin you have to
 - Handle arguments
 - Standard `-v -h`
 - Help has to contain documentation
 - Get and manipulate the data
 - Calculate state (OK, WARNING, CRITICAL)
 - In `f(x)` of arguments `->` STATE in a FLEXIBLE way
 - Output performance data
 - And worry about the format

That's a lot of work

I just wanted to monitor my app!



Nagios::Plugin

- Lots of common functionality for free!
 - When writing a good plugin you have to
 - ~~Handle arguments~~
 - ~~Standard -v -h~~
 - **Get and manipulate the data**
 - ~~Calculate state (OK, WARNING, CRITICAL)~~
 - ~~In f(x) of arguments → STATE~~
 - ~~Output performance data~~
 - ~~And worry about the format~~

3 Simple Steps

- Setup
- Data collection
- State calculation

Setup

- Just make an instance of N::P

```
#!/usr/bin/perl
use Nagios::Plugin;
my $np= Nagios::Plugin->new(
    'usage' => 'Usage: %s'
);
$np->getopts;
```

- You've just obtained
 - -t option (timeout)
 - -v option (verbose)
 - --help option

Setup (II) Constructor options

- usage ("Usage: %s --foo --bar")
- version <- Version string
- url <- Help and Version
- blurb <- Help description
- license <- Help
- extra <- Help
- plugin <- overrides auto detected name

Setup: GetOpt magic

```
$np->add_arg(  
  spec=> 'warning|w=s',  
  help=> "-w, --warning=RANGE",  
  required=> 1  
);  
$np->add_arg(  
  spec => 'user|u=s',  
  help => 'Username',  
  default => 'www-data'  
);  
$np->getopts;  
if($np->opts->user) { ... }
```

Collect data

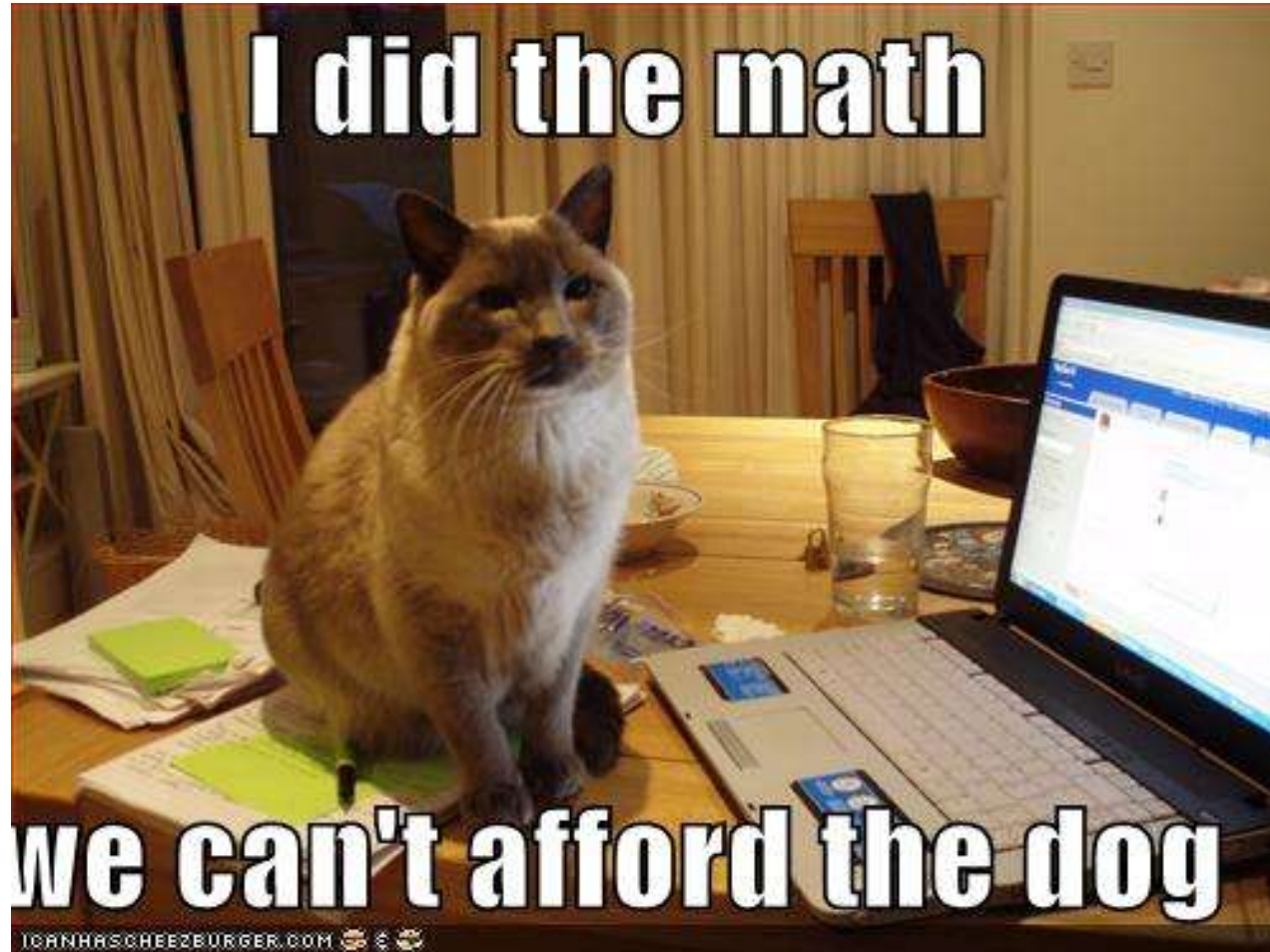
Now you have to work



HARD WORK

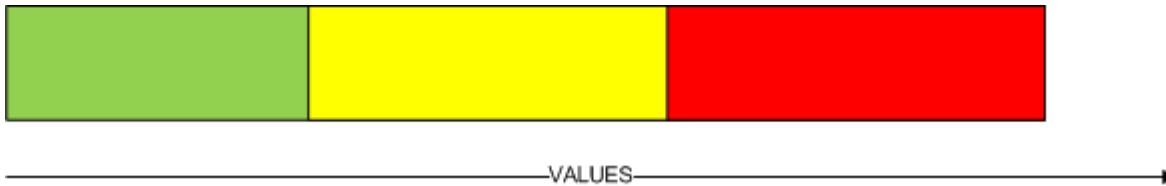
and they said it never killed anyone

State calculation



Ranges

- Most plugins suppose that
 - OK<WARNING<CRITICAL



- But... what if...



Ranges

Range definition	Generate alert if x...
10	Is not between 0 and 10
10:	Is not between 10 and infinity
~:10	Is not between -Inf and 10
10:20	Is not between 10 and 20
@10:20	Is between 10 and 20

```
$code= $np->check_threshold(  
    check => $value,  
    warning => $warning_threshold,  
    critical => $critical_threshold  
);  
$np->nagios_exit( $code, "Thresholdcheckfailed" ) if ($code!= OK);
```

Output status

- `$np->nagios_exit(CRITICAL, "Too many connections");`
- `$np->nagios_exit(OK, "OK");`
- `$np->nagios_exit(WARNING, "Too few connections");`
- `$np->nagios_exit(UNKNOWN, "Bad options");`

Performance Data

```
$np->add_perfdata(  
  label      => "size",  
  value      => $value,  
  uom        => "kB",  
  warning    => $warning,  
  critical   => $critical  
);
```

UOM Unit of measurement	Is for
No unit specified	Assume a number of things
s,ms,us	econds, milliseconds, nanoseconds
%	Percentage
B,KB,MB,TB	Bytes
c	A continuous counter

Nagios::Plugin::DieNicely



PLAYING DEAD

A perfect tactic when caught stealing jelly beans

Nagios::Plugin::DieNicely

- Using CPAN modules to monitor will make this happen (eventually)

Service State Information

Current Status:	CRITICAL (for 0d 0h 0m 21s)
Status Information:	(Return code of 111 is out of bounds)
Performance Data:	
Current Attempt:	2/3 (SOFT state)
Last Check Time:	01-08-2010 22:12:52
Check Type:	ACTIVE
Check Latency / Duration:	1.251 / 0.400 seconds
Next Scheduled Check:	01-08-2010 22:13:52
Last State Change:	01-08-2010 22:12:45
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	01-08-2010 22:13:00 (0d 0h 0m 6s ago)

Nagios::Plugin::DieNicely

- What happened?



CONFUSION

looks better sideways

Nagios::Plugin::DieNicely

- What happened?
 - Output went to STDERR
 - Nagios doesn't care
 - Exit code follows Perls rules
 - Nagios understands 0-3

Nagios::Plugin::DieNicely

- use Nagios::Plugin::DieNicely

Service State Information

Current Status:	CRITICAL (for 0d 0h 6m 19s)
Status Information:	CRITICAL - Can't call method "syswrite" on an undefined value at /usr/share/perl5/Gearman/Taskset.pm line 201.
Performance Data:	
Current Attempt:	3/3 (HARD state)
Last Check Time:	01-08-2010 22:18:52
Check Type:	ACTIVE
Check Latency / Duration:	0.186 / 0.125 seconds
Next Scheduled Check:	01-08-2010 22:18:55
Last State Change:	01-08-2010 22:12:45
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (6.12% state change)
In Scheduled Downtime?	NO
Last Update:	01-08-2010 22:19:00 (0d 0h 0m 4s ago)

Nagios::Plugin::WWW::Mechanize



Nagios::Plugin::WWW::Mechanize

Nagios::Plugin

+

WWW::Mechanize

You where going to do it anyway!

Nagios::Plugin::WWW::Mechanize

- Again... Just create an instance. Use it as a Nagios::Plugin object
- Automatically tracks response time for you
- `$np->mech`
- `$np->content`
- `$np->get`, `$np->submit_form`

Nagios::Plugin::WWW::Mechanize

- A couple of tricks
 - Gzipped content

```
my $np = Nagios::Plugin::WWW::Mechanize->new(  
    'mech' => WWW::Mechanize::GZip->new(autocheck => 0)  
);
```

- Proxy

```
my $proxy = $np->opts->proxy;  
if (defined $proxy) {  
    $np->mech->proxy(['http', 'https'], $proxy);  
}
```

Nagios::Plugin::SNMP

Nagios::Plugin

+

Net::SNMP

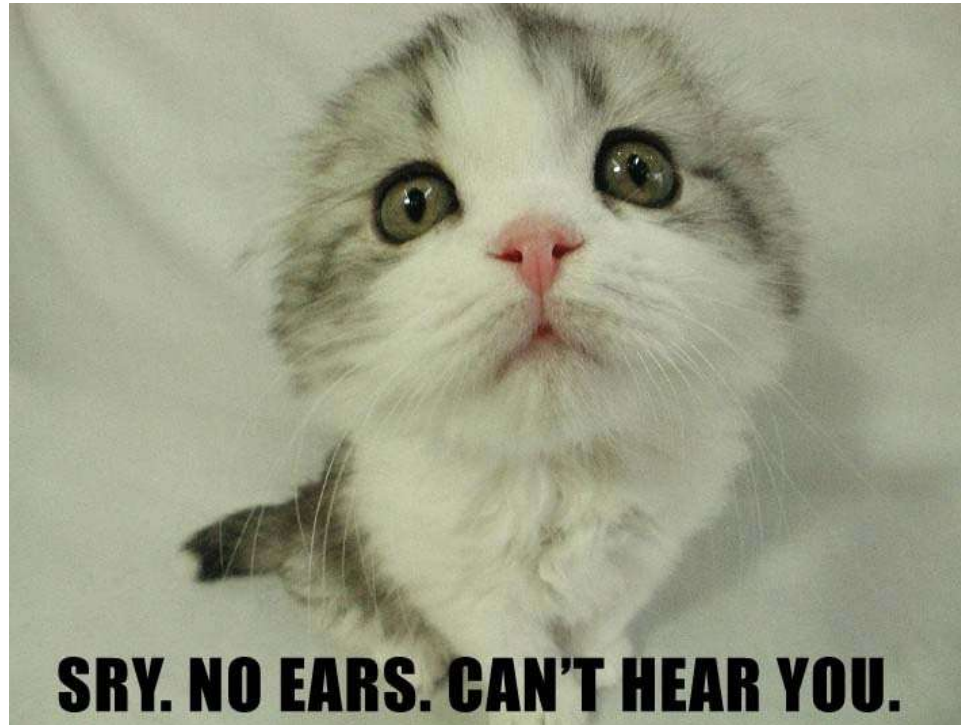
Nagios::Plugin::SNMP

- Again... Just create an instance. Use it as a Nagios::Plugin object
- Sets up a lot of arguments
- Does delta between values of counters

A module in the works

- Nagios::Plugin::Differences
 - In the works. Help out with continuously growing counters

Any Questions?



Thanks to:
Ton Voon for his hard work on Nagios::Plugin
Icanhazcheezburger for the cats