JaM - PHP Error Monitoring Extension

Jess Portnoy

Kaltura, Inc

jess.portnoy@kaltura.com

April 20, 2016

The Need

Big complex PHP based systems have a lot of moving parts.
It is very common for something to malfunction without being immediately evident to end users.

It is therefore mandatory to have monitors in place to babysit your system and alert in the event of trouble.

Ideally, you would like to catch malfunctioning code during development time so that it never reaches production.

What to monitor

There are a lot of things to monitor:

- Your servers load, memory usage, disk space, etc
- The services your app relies on - DB, web server, caching server, etc
- And finally, your actual APPLICATION...

How can you monitor your app?

Most mature platforms, Kaltura included, extend an API you can use to check your application's health.

Kaltura does this by means of offering client libs you can use to write code that will monitor the application operations.

This is, of course, very good and necessary.
However, a key monitoring component, for Kaltura and most other platforms out there, is to monitor the logs..

The problem

Monitoring the logs can be difficult.

Why? Complex applications log A LOT of data and of many different types.
They log errors, warnings, informative messages that are NOT errors and so on and so on.

They often write to many log files, not just one.

Parsing big files is slow and sometimes difficult because you need your algorithm to determine whether or not the line it is now looking at is actually an error or not.
Looking at parsed results can also be cumbersome, especially when you have gigs and gigs of daily logs.

The Solution

Don't parse log files at all!

If your application is written in PHP, you can plug into PHP's core [AKA Zend Engine] directly and make it send you the relevant data.

See, the engine already knows whether something is an error or not.. In fact, it even knows the severity of that error.

This is exactly what the JaM extension does.

Credit Where Credit Is Due, or: Why FOSS Rules

Like any successful entrepreneur will tell you, the first step when you get an idea is to check whether someone is already doing it.

So I did.. and after a moderately extensive search for "zend_error_cb" found an excellent project called php-aware.
php-aware was using the very same technique I had in mind and was also well written and extensible so I tried to compile it and... it failed.

That was only because it was written in 2009 and the PHP Engine changed quite a bit since. A few minor fixes later, it compiled... But crushed when running.

Credit Where Credit Is Due, or: Why FOSS Rules - cont'd

After a few more fixes, the basics were working and it was time to implement the ElasticSearch backend, which the original project did not have.

When it was all done and working nicely, I made a pull request to the original author - Mikko Koppanen.

Mikko said he is happy to see someone taking active interest in it but.. He himself is no longer into PHP at all.
And, so, to make a long story short, this is how JaM was born:)

Implementation

The JaM PHP extension overrides Zend's Engine zend_error_cb(), set_error_handler() and restore_error_handler() with a custom function that takes a copy of the current context, sends the error to the backends set in the aware.storage_modules directive and then calls the original error handler(s).

Advantages

- Faster than parsing log files and more reliable
- Can be enabled easily
- Can monitor any PHP app for errors without having to make code changes to the application itself
- Supports a variety of solutions for storing and looking up errors [ElasticSearch, email, SNMP traps]
- Extensible: additional storage engines can be easily added without changing JaM's core code

Target Audience

Any organisation that uses or develops or runs a big PHP application, be it Kaltura, Wordpress, Drupal, PUT-YOUR-SYSTEM-HERE.

Wait... Can JaM collect things OTHER THAN PHP engine level errors?

Yes. It can.

JaM exposes a very simple method called jam_event_trigger(error_level, message)

This can be called from any PHP code to store any event you want and the flow will be the same as for PHP errors.

Future Roadmap

- Porting to PHP 7 is in progress, currently supports all 5.n versions, up to and including 5.6
- Involve other contributors in the project
- Create additional backend storage engines

Demo Time

This demo will show what the events look like in Kibana, how to search for events of a specific type, on specific hosts and in specific time ranges.
We will also show how events happen and are immediately sent by mail.

Thank You

I'd like to thank the author of PHP Aware, Mikko Koppanen, for writting excellent code and making it open.
JaM would have been a much longer effort had he not done that:)

References

- php-aware's GitHub repo - https://github.com/mkoppanen/php-aware
- JaM's GitHub repo - https://github.com/jessp01/jam/blob/master
- ElasticSearch - https://www.elastic.co
- Kibana Queries Reference Guide - https://www.elastic.co/guide/en/kibana/3.0/queries.html

The End && questions