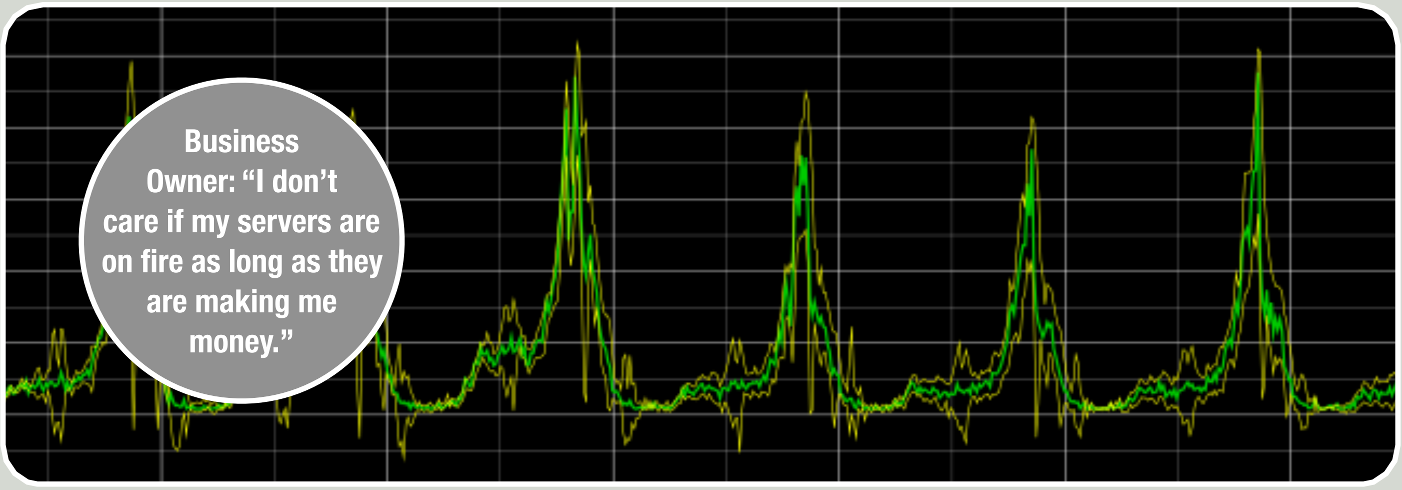
TRENDING

Measure Every Metric - September 2012



BUILDING A BUSINESS METRICS DASHBOARD

This article is the result of an adventure into the world of devops and business metric driven development. The future of monitoring is a polyglot toolbox and not a monolithic swiss army knife.

Nagios is installed everywhere. It's handy because it tells you when things break. However, just because a process on a server responds to a probe, it does not mean that it's working. Nagios does not understand trends and the daily variations in your traffic. It sends alerts when a number passes a fixed threshold.

There is something better. That's where trending comes in. Trends can tell you when something is **about** to break. That's much better than hearing it from the customer. Looking at data can also tell you when something is already broken but you didn't know about it yet.

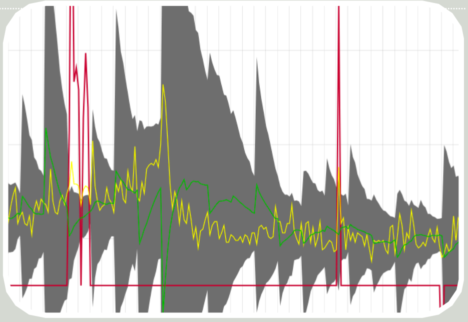
What data should you collect? You don't know what might break, so collect everything - application, machine and network. That sounds like a lot of work, but it's not - the tools are freely available. **Graphite** is a place for your business metrics to live. It provides advanced graphing, metric storage, all sorts of advanced functions and formulae and a Web interface with dashboards. Graphite doesn't just serve graphs - by changing a single parameter in a URL you can get raw data. That raw data can then be polled by Nagios. This is a fantastic capability. You can apply a function to a trend, say to transform it, make it linear or smooth it and then have Nagios send alerts based on the trend data.

In Graphite, every graph is represented by a URL. That makes it easy to build business dashboards to display around the office.

How do you collect the metrics? You have a range of tools at your disposal and you should use most of them. **StatsD** is great for application metrics. It uses UDP to broadcast counts and timings from inside your application. It won't block up your application because UDP is 'fire and forget'. StatsD aggregates metrics and passes them on to Graphite. There is client code available for a multitude of programming languages.

Logster will rip metrics from your log files and pass them on to Graphite. You configure regex to match what you want. It could be as simple as graphing your error count or HTTP response codes.

JmxTrans will get your Java JMX data into Graphite. That makes it easy to monitor


Forecasting the future

We can forecast the future based on trends by using **Holt-Winters** exponential smoothing. This function takes account of seasonality as well as trends. The chart above shows an actual metric (green), a forecast (yellow), confidence area (grey) and the aberration (red).

The aberration shows us where the metric deviates from the forecast. By monitoring the raw aberration data in Nagios we can easily alert when the aberration exceeds our thresholds.

JVM health and other Java application metrics that are commonly exposed via JMX. Using a local agent to query JMX and forward the data also means that you avoid the usual JMX firewall configuration hassle.

Codehale **Metrics** is a powerful library for getting your metrics out of a Java application. It provides counters, timers, gauges, histograms and other useful tools. Once again it can forward data to Graphite.

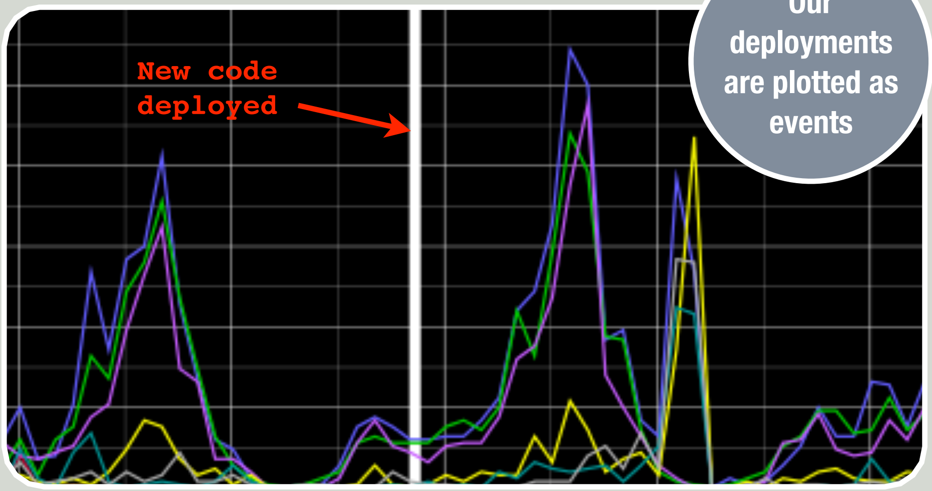
System metrics are also easy to get into Graphite. **CollectD** can gather every system metric you can imagine and forward it to Graphite. That means you can have your CPU, disk, memory and network data right alongside all of your other metrics.

Putting it all together

At first you will start with just a few metrics. It won't take long before you become hooked and start measuring everything. That means a lot of data. One great feature of Graphite's storage engine is that it can aggregate data for different retention periods. You can configure it too keep all of the data for 6 hours but then to aggregate data from 6 hours to one week and so on. Eventually you will need to scale. Luckily, Graphite supports federated storage. You can just add more servers, each of which knows how to retrieve data from the other servers.

To make this easy we use a tool called Chef to provision the servers. It's easy to get a new reporting server up and running on Amazon EC2 within 5 minutes. And, of course, we monitor the health of our Graphite cluster by collecting metrics. We can see how many metrics are recorded and how long each update takes.

Recently we found that Graphite is an excellent tool for gaining insight into an application during performance testing.



Our deployments are plotted as events

Deployments are metrics too

Software deployments can obviously have a significant effect on a system if something goes wrong. That's why we automate deployments and rollbacks. Our deployment script sends information about the event to Graphite - it's just another metric to keep track of. Verification is easy because our monitoring will let us know if something stops fitting a trend. In the case that something does go wrong it is easy to look at the graphs and correlate the deployment event with unusual data and get to the root cause quickly.

James Brook - 6 September 2012

NEXT STEPS

Articles, Books & Screencasts

- Measure Anything, Measure Everything - Etsy
  http://goo.gl/zlUK8
- The Lean Startup - Eric Ries
  http://goo.gl/brejD
- Obfuscurity Blog
  http://obfuscurity.com
- Monitoring Sucks
  http://monitoring.no.de
- Holt-Winters Exponential Smoothing
  http://goo.gl/sTnBf
- Metrics, Metrics Everywhere - Screencast
  http://goo.gl/oMFi6

Tools

- StatsD
  http://git.io/1SN46Q
- Graphite
  http://goo.gl/C7ffj
- Logster
  http://git.io/ZzWJ3g
- check_graphite_data (Nagios)
  http://git.io/yD3_sw
- Jmxtrans
  http://git.io/ptQsTw
- CollectD
  http://collectd.org
- Codehale/metrics
  http://goo.gl/PR4fX
- Chef
  http://goo.gl/raqhA

SOME THINGS WE MEASURE	CREDIT TO	FEEDBACK
• Remote recordings sent, successful, failed, pending • Timings and counts of calls to services we depend on (e.g. PEAL and SSO). • Deployment events. • Log events (info/warn/error) • System and JVM health • Login and registration app metrics • Application insights during perf testing	DevOps community, especially @obfuscurity Sergey Belyakov, Natalia Pavlova & Theofilos Papapanagiotou (LGI)	If you have questions or comments please contact James Brook We'd be interested in what you think of the format (news letter vs blog) and potential future topics.