Monitor the Monitors

SELENA DECKELMANN

Selena Deckelmann is a major contributor to PostgreSQL and a data architect at Mozilla. She's been involved with free and open source software since 1995 and began running conferences for PostgreSQL in 2007. In 2012, she founded PyLadiesPDX, a portland chapter of PyLadies. Deckelmann founded Open Source Bridge and Postgres Open, and she speaks internationally about open source, databases, and community. She is an advisor to the Ada Initiative, an organization dedicated to increasing the participation of women in open source and technology communities. You can find her on twitter (@selenamarie) and on her blog at chesnok.com. selena@chesnok.com

A colleague asked: What tools do people use for monitoring the monitors? His question was inspired by "Dead Man's Snitch," a web-based tool that detects and reports out when a regularly scheduled job hasn't run. This got me thinking about all the kinds of systems teams set up to detect silent failures. Most involve turning silence into noise.

When asked, friends had quite a few different names for these strategies: meta-monitoring, the "everything's ok" alarm, a "canary in the mine," Dead Man's Switch, heartbeat, watchdog, high availability response, and the entertainingly painful OOBETET (out-of-band-end-to-end test).

You could divide that list up in a few different ways:

- Automated vs. manual execution
- Automated vs. manual response
- Destructive vs. non-destructive response
- Monitoring vs. monitoring of notification services

Classifying the strategies seems fairly simple. The Dead Man's Switch is automated and contains a destructive response, at least in the movies. An out-of-band-end-to-end test is often manual both in execution and response, and non-destructive.

Definitions become a bit fuzzier, however, when we consider "monitoring vs. monitoring of notification services". For example, how do you ensure (within reason) that text message alerts actually made it to the staff who can fix a problem?

This kind of testing seems to fall under "out of band" checks. These are verification routines we don't include in our primary monitoring systems, just in case the primary systems aren't available.

Teams have lots of informal—and sometimes formal—ways of managing these issues. For example, when on-call switches over, the phone company may send an initial alert text message to the new shift's phone number.

In my first job out of college, our team sent out a test alarm at 9 a.m. every day to the on-call team member, signaling that the paging service was still working. If you were on call and didn't receive the page, you knew to contact your manager to get things fixed as soon as possible.

Critically, this kind of verification system is not fully automated. Instead, the system is managed by training people to notice a simple signal (in this case, a disruption in a well-known routine) and supplying instructions for exactly how to respond.

Much of the work involved in keeping formal out-of-band checks working and useful is in documentation and training. When systems are designed for human intervention, there's a temptation to not document the process fully because you know a person must puzzle their way through responding to the alarm anyway.

But there's significant value in documenting even our informal process. Training is much easier when you've got a document to work from, and you can have some degree of assurance that team members will run through steps to fix problems the same way every time. If something changes, you know where to document the change – in the written documentation. You may find that the check and even the fix can be automated. And, staged failures to test the procedure is a lot easier to manage when you have a fix procedure to test.

So, take a few minutes and have a look at your team's out-of-band monitoring. What are the things that might fail silently? Can you turn that silent failure into noise? How will your team detect and respond? And then write it down!