iVoyeur
Nagios XI (fin)

DAVE JOSEPHSEN

Dave Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and is Senior Systems Engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project. dave-usenix@skeptech.org

We have in our community certain long-running esthetic and philosophical disagreements to which we sometimes refer as "religious battles." The ancient vi vs Emacs battle probably exemplifies this practice, there being both a "Cult of vi" [1] as well as a "Church of Emacs" [2] (and even an alt.religion.emacs mailing list).

Being a pedantic sort of fellow, I sometimes wonder how we, as a community, struck upon that particular adjective and how the theists among us feel about our use of it in a pejorative sense. It may just be me, but when we describe these battles as "religious" it seems to imply some arbitrary and thoughtless or perhaps involuntary choosing of sides. A heritage pushed on us, or the side of the river where we are born; an idea we don't understand but for which we are forever doomed to be an apologist, or an assertion we doubt and yet must defend with our lives.

But in truth this is rarely the case with the actual battles. Everyone I know who has chosen either vi or Emacs has done so deliberately, not only fully aware of the specific reasons why, but unmindful of this global struggle for text editor domination (except for the MIT graduates among us). There are, after all, real and practical reasons for choosing one or the other.

I suppose those things are sometimes fair to say about religion among the general public, but in my experience it's not true of the theists in our community, who are likely to have chosen their beliefs with the same unbiased thoughtfulness they used in choosing their text editor. I don't say that out of ignorance or political correctness. Some of the best conversations I've had at LISA have been late night semi-drunken arguments about the nature of the universe with atheists, agnostics, and theists of various description. These usually start in the deep-end and continue into vast and unknown waters where we usually encounter Descartes, Rutherford, Aquinas, Darwin, Locke, and many other gentlemen of their ilk.

It might be that, having chosen a side, it's difficult for us to understand why other people would choose something different, and therefore those people over there who chose differently look strange and silly. And by strange I mean dancing around with snakes, healing each other in big white tents, holding literary critiques with questionable conclusions at bible camp [3] strange. *Do those people have ANY idea what they're doing?* That seems a more apt description of the Emacs vs vi battle, but it's not fair to say about religion.

The problem is: all of that strange stuff I mentioned—that bible camp snake healing—none of that really has anything to do with theism. It's just people being weird and silly in a god-themed sort of way. If I surgically removed religion from their minds tomorrow, they'd just channel all of that manic, scatterbrained silliness into weird practices in the name of science, and find entirely new reasons to knock on your door, lobby for weird laws, and write questionable things in textbooks.

Insomuch as this is what we mean when we point at something and call it a religious debate, I'm beginning to think we've made an unfortunate choice in that particular adjective. At least we've done a poor job of assessing root cause. Beliefs don't hold people, it's the other way around. If people are acting silly in the name of religion, it's unfair of us to blame religion.

Isn't the fact that we have these sorts of debates in the nerdosphere—to the extent that we've literally formed tongue-in-cheek churches around them—proof of something decidedly not religious? Some agnostic quirk of human nature that inclines us to irrationally take sides against each other over trivialities? Isn't "The Church of Emacs" just nerds being silly in a god-themed way?

We do this an awful lot actually, from Maxwell's demon [4] to "magic smoke" [5] to "The Cathedral and the Bazaar" [6], we nerds have surrounded ourselves with supernatural imagery since the beginning. That's quite natural I think, given the quantity of time we spend with one foot in another world, designing, building, and planning in non-physical, supernatural space.

Nagios, which happens to be a recursive acronym for "Nagios Ain't Gonna Insist On Sainthood," is no stranger to the phenomenon we describe as a religious debate. In my first article on Nagios XI I alluded to some of the complaints voiced by adherents of, or dare I say converts to, commercial monitoring systems. In that article I introduced Nagios XI and described its architecture—all the ways that it was and was not Nagios. In my second article on XI, I covered the new auto-discovery functionality and configuration wizards, both of which put an end to manual Nagios configuration, a favorite topic of the anti-Nagios … heathens. In this, my third and final Nagios XI article, I'm going to quell the flames of that debate a little more by directly addressing the rest of the complaints that seem to be driving admins away from Nagios and toward commercial monitoring solutions.

Nagios Is Ugly Because It Doesn't Have Pretty Graphs

With Nagios core, the admin has to enable performance data processing in the nagios.cfg and then grab some external software to parse the performance data output from Nagios, and shove it all into RRDs. The two most popular choices are NagiosGraph [7] and PNP4Nagios [8], both of which I've written about in the past.

In XI, however, PNP4Nagios is integrated out of the box, and definitions exist for all included plugins. This means that without any additional configuration whatsoever you get time series data for every service you configure. The RRDtool graphs are so well integrated into the new XI user interface that the uninitiated user would never guess PNP or RRDtool were community-sourced add-ons, so you get a snazzy UI without losing any of the power and flexibility that these community-driven development efforts provide.

In addition to the RRDtool graphs, small bar-graph visualizations for metrics collected by the Nagios Core daemon, as well as remote execution tools such as NRPE, are sprinkled throughout the interface. These do a great job of conveying capacity planning info at a glance, as well as giving the UI a very polished look.

NagVis, a tool for overlaying status data from Nagios over graphics (e.g., network diagrams or geographical maps), is installed and available in the "Maps" section of the "Home" view, and setting up your own NagVis diagrams couldn't be easier.

But wait, that's not all; rounding out the time series visualization is a Graph Explorer tool, which allows you to draw among other things, ad hoc time series and "stacked" time series graphs. The Graph Explorer uses a commercial JavaScript library from Highcharts.com and looks quite elegant. The data comes from the RRDs resident on the Nagios server via rrdtool fetch, and is provided to the end-users' browser to compute the graph locally. This saves the server's CPU and provides a snappy, feature-rich data visualization, allowing you to scale the graph by dragging to select a range, and providing pop-up numerical values when you mouse over any data areas. The "stacked" time series graphs include time-shifted historical data, so you can easily compare today's data to that of yesterday etc.

The Nagios UI Sucks Because You Can't Extend It

The Nagios Core UI is implemented as a series of CGIs written in C, and hasn't had a major overhaul in years. It is often maligned for its "outdated" design, and was one of the central points of contention in the Nagios community that gave birth to the fork that became Ichinga. Several free replacement UIs are available from the community, the most popular of which are MK_Multisite [9] and thruk [10].

XI, by comparison, comes with a wholly new and redesigned PHP-based user interface. The UI as a whole is highly modular, incorporating add-on components to implement extra features. This enables the XI developers to react quickly to the needs of the user community by adding features to the UI as needed or even custom developing features for larger end-users with special needs. A notable example is the "Operations Screen," which is intended to be displayed on a dedicated monitor in a network operations center. In addition to this and other single-page summaries, custom views can be configured to rotate between pages with more detailed information on timed intervals.

Another component that implements a feature for which the core community has been begging for years is the "Mass Acknowledgment Component." This allows an admin to schedule downtime and acknowledge problems for groups of hosts and services. I know more than one sysadmin who would purchase XI for this feature alone.

Nagios-Generated Reports Look Boring

Nagios Core is capable of generating usage and availability reports by way of the "Reporting" section off the left nav. These reports are sort of difficult to build, export, and link to, and they're basically just spreadsheets. Fine for you and me, but not something that would impress the suits.

The "Reporting" tab in XI by comparison shows some interesting data visualization techniques from the Neoformix data-visualization field. Components that implement heatmaps, force directed graphs, and stream graphs have been added to the classic reporting options. Several shiny new implementations of the core reports are also provided, each of which I find generally cleaner than their legacy counterparts and more likely to impress the wearers of neckties and high-heels in our lives. The new reports may be exported in CSV and PDF formats with the click of a button. The button, which links to a predictable URL, makes it possible for us shorts and t-shirt wearers to grab the reports automatically with tools like curl and wget.

Nagios Can Only Model Hosts and Services

Nagios Core tracks hosts and the services running on those hosts. Any larger sort of entity composed of groups of hosts, or services from groups of hosts is more difficult to model, though there are add-ons like check_mk that make this sort of thing possible.

Nagios XI, on the other hand, contains wrapper logic for grouping individual services together into higher level entities called business processes. The intent here is to implement what The Gardiner Group calls "BAM," or "Business Application Monitoring." BAM attempts to provide real-time status for critical business entities like a sales catalog Web site, or corporate email. Nagios XI implements BAM by breaking a high-level concept like "corporate email" into its requisite pieces— mail transfer agents (MTAs), mail exchangers (MXs), groupware systems, and databases—and then quantifying the relative importance of each of the services that make up those pieces as well as describing dependency relationships between them.

XI business process groups contain services that are said to be "essential" or "non-essential." A database service in our example might be considered essential, while the SMTP port on a single mail exchanger might be "non-essential" (because they are usually redundant, and even if they go down, the mail will queue somewhere else). When any essential service or the combination of all non-essential services goes critical, the XI business process logic registers this as a "problem."

Each business process group contains critical and warning thresholds that depend on the number of problems that are occurring in the group. In our example, we might imagine two business process groups, one for SMTP-speakers (MXs and MTAs) and one for SQL-speakers (groupware systems and DBs). If the latter group registers a single problem, because a database is down, that might throw the whole group into a warning state.

Business process groups can contain other nested business process groups and so on. Our top-level entity, "corporate email," is therefore just a business process group that contains the two groups described above. It is configured just like the other two groups such that a single "problem" in any of the nested groups causes it to go into a warning state. Finally, notification commands can be assigned on each business process group in the same way they are assigned to individual host and service events. Additionally, visualization widgets exist for the top-level groups. These can be added to any dashboard or view and allow the user to drill down into the groups to see what services or subgroups they contain.

I hope you've enjoyed this series on Nagios XI as much as I've enjoyed writing it. If you're in a corporate environment, or find yourself in want of a turn-key commercial systems monitoring solution, I can highly recommend Nagios XI.

Take it easy.

References

[1] The Cult of vi: http://6thstreetradio.org/~davek/vi.html.

[2] The Church of Emacs: http://www.emacswiki.org/emacs/ChurchOfEmacs.

[3] Harry Potter is the Devil: http://www.youtube.com/watch?v=RSwZJ55g80Q.

[4] Maxwell's demon of thermodynamics: http://en.wikipedia.org/wiki/Maxwell%27s_demon.

[5] Magic smoke: http://www.outpost9.com/reference/jargon/jargon_28.html#TAG1095.

[6] ESR's "The Cathedral and the Bazaar": http://www.catb.org/esr/writings/homesteading/.

[7] NagiosGraph: http://nagiosgraph.sourceforge.net.

[8] PNP4Nagios: http://www.pnp4nagios.org.

[9] MK_Multisite: http://mathias-kettner.de/checkmk_multisite.html.

[10] Thruk: http://www.thruk.org.