iVoyeur
Nagios XI (cont.)

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

Before I begin in earnest, I should point out that this article is the second in a series of articles on Nagios XI, which is the commercial version of Nagios. Herein I assume you've read the previous article [1] and/or have a working understanding of the general XI architecture, which is different from Open Source Nagios, or "Nagios Core." So now that we have that out of the way...

Quick, what do you think of when I say "wizard"?

I'll risk being a bit presumptions in my hope that the image of Milamber, Gandalf, Dallben, Merlin, or etc. is what probably occurs to the type of person who might happen to accidentally read this article. Or if you're of a certain disposition, perhaps it was Sidi, Sauron, Arawn, or etc. (I'm not judging). If that's what you thought, then I'm with you. Those guys and the ideas connected with them are certainly the first thing that pops into my head, and although a few alternatives occur to me, the absolute last wizard on my mind, a wizard worse than the absolute darkest of the wizards of lore, a wizard so utterly corrupt and vile that I hesitate to mention it much less write an entire article about it, is the configuration wizard.

Was ever there a thing less wizardly? The configuration wizard is like a wizard in the same way Facebook is like a book (or dare I say for you Colorado readers: in the same way the flower pot is... well never mind). So I admit, I'm not looking forward to writing this particular article. And I've put it off, as long as absolutely possible (as my editor may attest), but it must be done. This of course is no slight to Nagios XI, which is awesome, and although the Nagios crew have done a top-notch job implementing a feature that will help a ton of people and fling wide for them the heavy, spiked portcullis that bars the entrance to corporate America, you'll forgive me, I'm sure, for feeling a bit reluctant in the documenting of it.

As I write this in the twilight of the year two thousand and twelve, there are system administrators who, while mostly competent and sane in other respects, have managed to carry out their entire careers using nothing but graphical configuration tools. As I related in the previous article, one of the major, oft-repeated gripes these admin have with Nagios Core is its reliance on configuration files and the accompanying assumption that you will edit them when you want the configuration to change.

To address this—perhaps the largest barrier to adoption for many corporate shops who need to simplify the configuration process—Nagios XI comes complete with all of the plugins in the standard plugins package, as well as NRPE, NSCA, and NRDP pre-installed. Additionally, the XI developers have provided a plethora of

semi-automated configuration wizards, which, given the bare-minimum information about a host, take care of the initial setup as well as adding and modifying services on already-configured hosts.

Pay No Attention to the Files Behind the Curtain

If you consult the official XI documentation at http://library.nagios.com/library/products/nagiosxi/documentation, you'll quickly form the impression that the wizards are the only method for host and service configuration. The configuration files themselves are rarely if ever mentioned, as if they don't exist. With names such as "Exchange Server," "Website," and "Windows Workstation," the wizards make setting up new hosts and services easy enough that these tasks can be delegated to 1st-level support techs, or even end-users. The auto-discovery wizard is capable of bootstrapping an environment given only a CIDR net-block to start with, and in my experience does a good job of initial setup. To add NRPE-based host checks, or other services after the fact, just run the appropriate wizard on the preexisting host.

For example, if Server1 was created with the auto-discovery wizard, and you now want to add NRPE checks to get CPU, Memory, and Disk information from the host, you must first install NRPE on Server1. If Server1 doesn't already have NRPE on it, and is one of several common server types, such as a Windows server, Red Hat, or Ubuntu, the XI developers have an agent package designed to work with XI specifically at:

```
http://assets.nagios.com/downloads/nagiosxi/wizards
```

Once the agent is installed on Server1, simply run the NRPE wizard on the server from the configuration tab of the XI user interface, as shown in Figure 1, entering the IP or FQDN of the server, and choosing the type from the drop-down list. The wizard will then display a pre-configured subset of available check commands relevant to your server type, and provide text-entry fields for you to specify custom settings or additional commands if you wish.

Auto-Configuration Gotchas

Static configuration files may still be maintained in etc/nagios/static. So it's entirely possible to run your own scripts, or auto-generation tools such as those

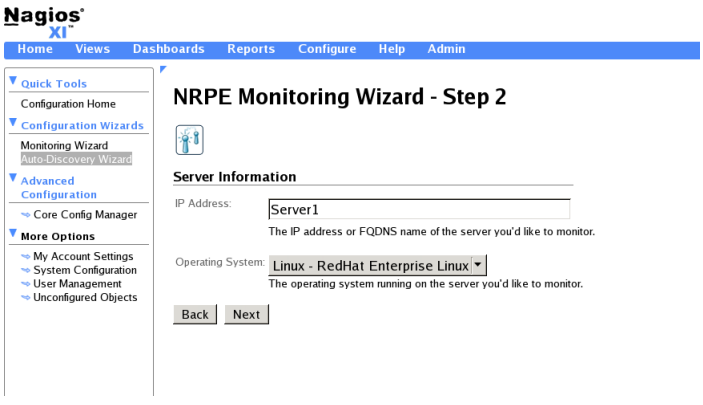

Figure 1: The Nagios XI NRPE wizard

included with Check_MK, provided you configure them to write their configuration to the static directory. I can't deny that the automated configuration features in XI have, ironically, complicated things a bit for those of us who have reason to maintain the configuration manually. While in the Nagios Core universe, there is a single way to configure Nagios (text files), there are three ways to configure Nagios in the XI universe (text files, NagiosQL, and XI wizards), and although the three co-exist as well as I think it's possible, it can become burdensome to ensure uniform parameters if the administrators mix-and-match their configuration methodologies in XI. I'll give you an example.

Larry, his brother Darryl, and his other brother Darryl all work at bloody stump lumber mill, where they recently purchased a Nagios XI server to monitor their growing sales Web-application server farm. Larry was a UNIX admin in college, so he prefers to edit the config files; Darryl likes to have fine-grained control over the config, but isn't very good in vim, so he uses the XI advanced configuration section; and other Darryl would rather be watching football (an American sport, similar to rugby but with armor), so he just runs the wizard for everything. Each of the brothers has a server running SSHD that he wants to configure in XI.

When other Darryl runs the auto-discovery wizard on his server's IP, XI scans the host and automatically configures a host check and a check_tcp service check for the SSH port. It then pushes the config to NagiosQL, which commits it to the DB, writes out the configuration, and restarts the daemon.

Darryl, meanwhile, sets up his host using the NagiosQL forms directly, but instead of choosing check_tcp, he chooses the check_ssh service, which does pretty much the same thing, but returns slightly different output. He also names the service "ssh" instead of "SSH" like the wizard does.

Larry, meanwhile, has really done his homework. He already has a service group for ssh servers in the static config files he created, so rather than doing all the typing and clicking that his brothers do, he simply adds his server to the ssh_servers service group, and the rest takes care of itself. The problem is, his service group inherits a different set of templates than NagiosQL, so although his service check uses the same name and check command as the wizard, his polling interval is different, and he has a different notification target for service warnings.

In this way the brothers end up with three different definitions for the same service, which might not be a problem immediately, but will cause all manner of headaches if and when they want to integrate Nagios with another tool, or generally try to do any sort of automation using their monitoring server.

I admit these sorts of disconnects are possible with text configuration files, but my point is the text configuration encourages administrators to use templates to normalize the configuration, as Larry did in the example above. The automated tools by comparison encourage isolating the configuration at the host level, because it's easier for the automated tools to parse them that way. Thus in Larry's configuration, we find a single services.cfg wherein every service is defined and assigned a host group, while in NagiosQL's configuration we find a services directory with a single file for each host. The former makes it pretty easy to verify that all the service checks for every host are implemented in the same way. The latter makes it much more difficult.

Further, in my experience, the disdain that people like Larry naturally feel for people like other Darryl generally discourages them from paying close attention

to what people like other Darryl are doing. In fact, merely inviting other Darryl to configure the monitoring server with wizards might trigger a tendency in Larry to go off on his own and "do it the right way" using well-written static config files, which only exacerbates the problem by more widely diverging the configuration paths.

Whether this will be a problem in your shop will depend on how many hands are stirring the pot, and the extent to which the more clueful users are aware of the potential problem. The idea of delegating the configs is certainly tempting, and I'm not saying you shouldn't. If you do, my advice would be to use either the wizards or static config for service and host creation, and avoid using NagiosQL directly if you can avoid it (you could still safely use it to modify objects, just not to create them). That way, you can carefully set up the static config to ensure it references the wizard templates, or simply copy definitions from the NagiosQL files, and everything should remain pretty much uniform.

Automated Configuration for Passive Checks

One cool bit of functionality that is related to automated configuration in Nagios XI is the "Unconfigured Objects" feature. In the event that XI receives a passive check result for a host or service that it doesn't know about, it automatically generates an inert configuration for that host or service, and places it in the "Unconfigured Objects" section of the "Configure" tab. Administrators may then approve the inert objects, and they will become part of the running configuration. This is a welcome addition that I can imagine myself becoming reliant on, and it wouldn't be possible without the other wizards in place.

Auto-Discovery Is Dead, Long Live Auto-Discovery

Four or five years ago, a monitoring system's ability to perform auto-discovery seemed to be the feature that enabled forum trolls to distinguish the "cool" monitoring systems from the insipid wanna-be toys, and Nagios, being bereft in this respect, was in the latter group. At the time, it seemed like I couldn't read a monitoring-related Slashdot post without being bombarded with comments from the adherents for various commercial products who were forever chanting this strange "auto-discovery or death" rhetoric.

Why they chose that particular feature I can't guess. I've rarely in my professional career found myself in want of such a tool for Nagios, which is not to imply that options were lacking. On the contrary, the whining in the forums begat an explosion of these add-ons for Nagios in every language at every level of complexity. So numerous were they that groups of them would loiter in the parks at night, and in the morning they would flock outside the Best Buy entrance, hoping for work. As a group I think most of us found them unwieldy; they made strange assumptions and were overly enamored of XML.

Today the various auto-discovery add-ons for Nagios have either disappeared or have become abandonware. Yes, all of them, 100%. Some light googling retrieves only ancient blog posts from bygone tool-writers announcing or justifying the creation of their now-abandoned hot new auto-discovery tool for Nagios (now with extra XML!). Given the firestorm of controversy that once surrounded this topic, I find it disorienting that not only the tools, but even the trolls have utterly vanished. It's a vexing turn of events but not, I think, an unhealthy one for the Nagios community, and I suspect two things account for it.

The first is a plugin written by Mathias Kettner called Check_MK, which I covered at length in [2] and [3]. The second is Nagios XI, which has everything the trolls would expect to see in a "cool kid" monitoring system and more, especially configuration wizards. I can't prove it, but my suspicion is that real administrators with real problems to solve discovered Check_MK and never looked back, or convinced their managers to pony up and buy them XI (or both); at the same time, one look at the XI screenshots caused a massive spontaneous troll migration away from the monitoring forums and toward dpreview.com or perhaps YouTube, where they all live happily trolling it up to this day (sorry about that, YouTube).

I jest, but truly, I think my hypothesis has some merit. If you're the kind of sysadmin who likes to get hacky with Nagios Core, you're going to write a one-liner for auto-discovery and be done. (The old auto-discovery tools wouldn't have given you enough control, anyway.) If you're the type who just wants to install something without getting too involved, you'll install Check_MK and be done. And if you're in the market for an effective, established, polished commercial product with support behind it, then you'll buy Nagios XI and be done. Even if it is an untestable assertion, I think I've decided to believe it on the grounds that it's also poetic; the wizards, after all, appear to have conquered the trolls.

Take it easy.

[1] https://www.usenix.org/publications/login/december-2012-volume-37-number-6/ivoyeur-nagios-xi.

[2] https://www.usenix.org/publications/login/june-2012-volume-37-number-3/ivoyeur-changing-game-part-4.

[3] https://www.usenix.org/publications/login/august-2012-volume-37-number-4/ivoyeur-gift-fire.