

SCF/FEF Evaluation of Nagios and Zabbix Monitoring Systems

Ed Simmonds and Jason Harrington
7/20/2009

Introduction

For FEF, a monitoring system must be capable of monitoring thousands of servers and tens of thousands of services. Further, it should be easy to deploy with as much automation as possible, such that new servers can be provisioned and added to monitoring with little administrative overhead and the capability of scripting changes and commands to the server. With this in mind, the FEF department has examined various open source monitoring tools to select the one that most meets our needs.

After an initial survey of existing open source monitoring tools, we chose to evaluate two of the most prominent, Zabbix and Nagios. We chose a set of criteria upon which to rate the two systems that is specific to the needs of our department, and which will integrate with our other administrative tools and procedures.

Evaluation Criteria

We evaluated the two systems based on the following criteria:

- Ability to scale to thousands of host checks and tens of thousands of service checks
- Ease of importing host and service checks from a script
- Ease of modifying a configuration or scheduled checks, particularly the ability to do mass changes to many nodes at once.
- API which can be accessed via a command line
- Ability to do distributed checks from multiple monitoring servers
- Reporting capabilities
- Ability to create Remedy tickets for critical alerts
- Industry acceptance, availability of expertise, and community support
- Potential to replace Faultlog, FEF's fault tracking application
- Ability to schedule planned maintenance periods and store work comments for hosts
- Ease of maintenance- is it easy to add new hosts and new service checks, particularly new ad hoc/custom checks, upgrade and patch?

There are numerous other features common to monitoring systems which we took note of, but did not identify as significant for our needs. Both Nagios and Zabbix have the ability to monitor hosts and services, to generate alerts, create Remedy tickets (with the assistance of the Remedy developers), store alert histories, and share many other capabilities. Therefore, we are interested in how the two differ based on how we intend to use them to complement our other procedures and tools.

Evaluation of Nagios

Configuration Data Store

Nagios stores configuration in flat files with a very simple format. This is an advantage for several of our evaluation criteria because it allows us to script the addition of new service checks and make mass changes using various scripting languages or basic shell scripts. For example, with little effort, we were able to generate a simple Nagios configuration for 3200 hosts and numerous clusters directly from the MISCOMP database with a simple Python script.

Scalability

Nagios has numerous data collection options. The default method does not scale well at all because it requires fork/exec's on the monitoring server for each check. Several other methods are agent-based, in which an agent runs on each monitored host and either 1. forwards updates to the server automatically or 2. waits for queries from the monitoring server and runs checks as requested. Nagios also has the ability, via an add-on, to set up proxy servers which run host and service checks on behalf of the central server. This allows the system to be scaled horizontally by adding more proxy servers as needed.

Nagios also has the ability to run multiple monitoring servers and forward results to a central server that aggregates check results. There are some capabilities lost in this configuration, the most significant of which is the ability to run ad hoc service checks from the central server's GUI. (To schedule an immediate check of a service outside of a normal scheduled check, an admin would have to log into the server actually running the check). The other disadvantage to this configuration is that the configurations on all servers must be maintained separately and do not replicate.

In tests on two modest servers, a central server and a proxy, Nagios monitored approximately one thousand servers with no performance issues noted.

Reporting Capabilities

Nagios keeps a history of alerts and downtimes for all hosts and service checks by default. Various reports of availability can be created for individual servers, host groups, specific service checks, etc. Nagios does not, by default, track performance metrics or provide graphs showing trends using this kind of data. It does, however, have numerous add-ons that allow this capability, as well as the ability to export this data to RRD format for external tools such as Cacti to use.

Nagios allows administrators to store comments with time stamps, so that administrators can store work history or other information and leave a log of notes for particular hosts. Each of these comments has a time stamp.

Application Interfaces

Nagios has a default web-based GUI, which is installed with the standard package. This interface is used to monitor hosts, acknowledge alerts, schedule maintenance, and run reports. The interface does not provide the capability to add or modify hosts or service checks. All host and service check changes are done via config files on the filesystem using an editor or external tool. There are numerous third-party tools that do provide the ability to modify configuration data via a GUI, but we have not evaluated any of them at this time.

Nagios also has the ability to accept commands via a named pipe, which is how all commands are sent to it from the GUI and the numerous third-party add-ons available. Nagios can be fully controlled from a command line via this method, so the server can be controlled entirely from the command line. This is useful for scheduling planned maintenance times in batches, for example, rather than clicking through an interface. Nagios has the ability to accept passive check results as well via the pipe. This allows arbitrarily complex or demanding external processes to submit service updates to the nagios server.

Evaluation of Zabbix

Configuration Data Store

Zabbix uses a database to store configuration definitions, but does not encourage directly modifying it. Therefore, all configuration changes must be made via the GUI, or by XML imports. Direct manipulation of the database does not appear to be a common practice by Zabbix users. Having the configuration in a relational database could be used for ad hoc reporting as well as updates.

Scalability

Zabbix works primarily via an agent that runs on each monitored host. That client collects a large set of information, which is then sent to the server on a regular basis. Our tests with Zabbix indicate that it would scale to the number of servers we require. Zabbix scales to this size without any third-party additions, whereas Nagios does require some changes and third-party software to service the volume of checks we anticipate.

Like Nagios, Zabbix has the ability to run multiple servers and aggregate the results centrally, but we do not anticipate centralizing Zabbix in this manner, so we did not test it. Zabbix also synchronizes distributed servers, so manual maintenance of multiple configurations is not necessary if this feature is used.

Also, like Nagios, Zabbix has a proxy configuration where secondary servers can act as a proxy to run checks from secondary locations on behalf of the central server. These proxies have some advantages and disadvantages over their Nagios counterparts. In Nagios, all proxies will randomly run service checks for all hosts. In other words, specific hosts cannot be assigned to a specific proxy. The advantage of this is that losing a single proxy doesn't eliminate the ability to monitor a given set of servers. The disadvantage is that you can't set up a particular proxy behind a specific firewall to monitor a subset of nodes. In Zabbix, a single specific proxy is assigned to a specific host. This allows a Zabbix proxy to monitor from a different subnet, for example, but creates an additional point of failure.

Reporting Capabilities

Zabbix has extensive reporting and graphing capabilities as part of the standard package. Unlike Nagios, it also has performance trend graphing built in.

Zabbix does not, however, have time-stamped comments for hosts. Therefore, it is not possible to

retain work histories and logs within the system.

Application Interfaces

Zabbix's primary interface is via a web-based gui. It has limited support for command line or scripted control and changes. Some configuration objects can be modified via xml imports as well, but there are limits. We have not found any way to perform most administrative functions via a command line, which limits scripting greatly.

Zabbix has a passive check capability, as does Nagios, which allows check results to be sent to the server via the `zabbix_sender` utility.

Analysis

Each system has advantages over the other, including features the other lacks.

Here are the key points we've identified:

- Zabbix has a much more attractive and mature user interface. Navigating it can be quite confusing, but the entire system can (must) be configured from the web browser.
- Zabbix has better reporting built in, particularly regarding performance data and trending. Nagios can produce many of the same types of graphs, but only via third-party add-ons and external tools.
- Zabbix has log monitoring capabilities. Nagios does not ship with log monitoring, though it is likely that in the 1500+ third-party plugins someone has implemented this. Alerts cannot be triggered from log events, however, which limits their usefulness. We did not identify this as a critical requirement so we did not pursue it, but we mention it because the Zabbix documentation lists it as a major feature.
- Zabbix, by default, monitors a much larger number of metrics from each server, but that can have an effect on the monitoring server's performance and disk requirements. The CMS group actually trimmed down many of these metrics because of the overhead involved in collecting them and storing them, particularly because the Zabbix MySQL database was growing much faster than they could accommodate. We see some advantages to having all these performance metrics available, but in our testing we also found it necessary to eliminate many of them to limit the resources required to run Zabbix.
- Zabbix is harder to configure, particularly if adding new, custom checks. It takes more steps to build a check in Zabbix and the process must be done through the GUI.
- Zabbix is much more cumbersome to script than Nagios, and interactions with the system must be done through the web-based GUI. Nagios' named pipe interface and simple config files are much easier to script.
- Zabbix currently does not support scheduled maintenance at the server level. The entire Zabbix server can be set offline, or alerts can be disabled by hand during maintenance, but scheduled maintenance cannot be set for specific time periods on specific hosts or services. In Nagios, scheduled maintenance can be assigned to host groups (i.e., clusters), individual hosts, or specific services either through the GUI, or via the named pipe interface. Zabbix has promised some form of scheduled maintenance in their next release, but we don't know what capabilities the feature will provide.
- Zabbix does not allow work logs or comments to be stored with a username and time stamp. It has a single free-form comment field, but does not log individual comments which we like to use as a work log for server issues. Nagios assigns a username and time stamp to each comment, so a better history is available for each host. Having this capability may allow us to entirely eliminate "Faultlog", an internally developed application that does basic monitoring of server availability and work histories.

- Mass changes in Zabbix are very slow and usually require clicking through the interface numerous times. They have added some new capabilities for making mass changes to some attributes, but the interface is clumsy and confusing. Nagios, because of its simple text file configurations, allows for mass changes with simple shell scripts (via perl, sed, etc.).
- We have more internal experience with Nagios. Staff who have used both Nagios and Zabbix believe Nagios is more stable than Zabbix. Zabbix, in our testing, has been unstable and requires restarting occasionally.
- Nagios has a very large user community, with over 1500 plugins (checks) for many common and not-so-common devices and services. Zabbix has a growing community as well, though not as large as Nagios, but far fewer third-party plugins and utilities.

Conclusion

Either of these systems would provide many of the monitoring features we need. There are many additional features of each system that we may want to use at some point, but none of these is critical to our decision. The two systems are fundamentally different in that Zabbix is much more of a monolithic system, with more structure and less flexibility, but a very nice interface with additional performance graphing and reporting. Nagios is much more modular and flexible at the command line level, and has a much simpler method of creating new checks and hosts via simple shell scripts and text-based config files.

With that in mind, the difference lies in how we want to use the system. Our emphasis is on maintainability and scriptability, and in this regard we believe Nagios has an advantage. Further, both because we have experience with Nagios and because of the simpler nature of the Nagios configuration itself, we anticipate that implementing this system will require less effort than Zabbix. Our expectation is that we can implement basic monitoring of new servers directly into our deployment/provisioning procedure with little or no additional overhead for administrators. Additional custom checks for key servers should be easier to implement in Nagios as well. There is a danger that Nagios' method of keeping thousands of servers and tens of thousands of service checks in flat config files may become large and unwieldy, but we believe that by wrapping the creation of new hosts in simple scripts and driving this from our deployment process, we can mitigate that risk.