Nagios at SLAC

Summary

https://nagios.slac.stanford.edu/

What is Nagios?

Nagios is an open-source monitoring tool. It is used at SLAC to automatically watch key hosts and services, and to contact appropriate personnel when/if these services go down.

The primary SLAC Nagios instance is run by SCS. The web interface is available at https://nagios.slac.stanford.edu/.

How do I use this service?

To view https://nagios.slac.stanford.edu/ you must authenticate with a SLAC-based Unix or Windows account and password, when prompted by a webauth dialogue.

(Put link for how to request a slac-based (*nagios*) account here.)

Please contact unix-admin@slac.stanford.edu to discuss adding your hosts and services to the central SLAC Nagios service; to adjust existing checks; or to request that you be included in alert notifications for a specific host/service.

Nagios for Users

Checks

Each host/service check is performed by invoking a command on the nagios server. These checks are generally extremely light-weight and specific:

- Is this host pingable?
- Can I reach the web port on this box?
- What is the system load on this host, and is it within a certain range?

Each check is run every 5-10 minutes (configurable), and returns one of three values - 'OK', 'WARNING', or 'CRITICAL' - as well as some explanatory text. The return values are used to determine the current state of the service. Anything other than 'OK' will eventually trigger alerts.

If a problem is found, the Nagios server will immediately schedule another check; if the problem continues, then Nagios will mark the service as 'WARNING' or 'CRITICAL', and send out an appropriate alert to the pre-defined parties. Alerts will continue to be sent periodically (about once an hour) until the service recovers; at that point, a 'RECOVERY' alert is sent.

As of the time of writing, the nagios server runs ~6000 of these checks. This is likely to grow.

Acknowledging Alerts and Scheduling Downtime

By acknowledging an alert for a host or service, you can keep nagios from sending out notifications for the problem (although it will continue to probe the host/service). You don't **have** to do this, but it will keep your email load lighter, as well as tell other interested parties that somebody is looking into problems.

Similarly, there is a method to schedule downtime for hosts or services. If you know that you're about to reboot a system, and don't want to be notified, you can schedule 10 minutes of downtime with nagios; if you know that you're about to tinker with a service, you may want to schedule a few hours of downtime.

Alerts are acknowledged and downtimes are scheduled from the command-line via remctl. remctl is a Kerberos-based client-server protocol that provides authenticated per-user/principle access to specific backend commands; in this case, the back end is a locally-maintained script that allows users to interact with nagios directly.

This generally works as so:

```
# Acknowledge an alert; stop sending emails
remctl nagios02.slac.stanford.edu nagios ack host HOSTNAME COMMENT
remctl nagios02.slac.stanford.edu nagios ack service HOSTNAME SERVICENAME COMMENT

# Pre-emptively mark a host/service as down, don't contact for a while
remctl nagios02.slac.stanford.edu nagios downtime host HOSTNAME HOURS COMMENT
remctl nagios02.slac.stanford.edu nagios downtime service HOSTNAME SERVICENAME HOURS COMMENT

# Tell nagios to run the check for this host/service in MINUTES minutes
remctl nagios02.slac.stanford.edu nagios schedule host HOSTNAME MINUTES COMMENT
remctl nagios02.slac.stanford.edu nagios schedule service HOSTNAME SERVICENAME MINUTES COMMENT

# Help documents and man pages
remctl nagios02.slac.stanford.edu nagios help
remctl nagios02.slac.stanford.edu nagios man
```

This requires that you have the 'remctl' binary installed (generally in /usr/local/bin on our systems, or you can install the 'remctl-client' RPM), and that you have a kerberos ticket.

Access is currently restricted to:

1. SCS Staff
2. Some number of Fermi/SCA staff.

We are investigating opening this up further.