Setting Up A Nagios Monitoring System

Warren Block, May 2005

What Is Nagios?

NAGIOS (na gee ose) is a system that will monitor the status of other network computers or components. It can watch your network and alert you when things happen. If a computer stops responding or fails somehow, Nagios can send an email, page or let you know in some other way. At the same time, it will create a web page that shows the current status of all the systems being monitored.

Because Nagios has so many capabilities, it can be complex to set up. Here, we'll have it monitor itself and only one other computer. FreeBSD is used for the Nagios system, so we can take advantage of the Ports Collection to make things easier.

Setting Up FreeBSD

Not much of a computer is needed for Nagios. A P166 system could easily handle a hundred monitored machines. 32M of RAM and 4G of disk is adequate, so this can be a good use for old computers. Nagios could also be added to an existing system without adding much load.

A plain-vanilla FreeBSD installation from the FreeBSD 5.3-Release CD is the starting point. Update the ports so all the latest versions are available:

```
# cp /usr/share/examples/cvsup/ports-supfile /root
```
(edit `ports-supfile` to point to the closest cvsup mirror)
```
# pkg_add -r cvsup-without-gui
# rehash
# cvsup /root/ports-supfile
```

Install Nagios

The Ports Collection makes installing Nagios easy:

```
# cd /usr/ports/net-mgmt/nagios
# make install clean
```

To enable Nagios, add this line to `/etc/rc.conf`:

```
nagios_enable="YES"
```

Configure Nagios

Configuration is the biggest part of setting up Nagios. Fortunately, much of it is one-time, and making later additions is easier.

On FreeBSD, the configuration files for added software are stored in the `/usr/local/etc` directory. Because Nagios has many config files, it keeps them all there in a subdirectory called `nagios`.

```
# cd /usr/local/etc/nagios
```

Nagios includes samples of most of the conf files, so they can be copied and modified. First we'll do this with the main config file, `nagios.cfg`:

```
# cp nagios.cfg-sample nagios.cfg
```

There are two changes we need to make in this file. The first is to set `cfg_file` to tell Nagios where our site configuration file is located. This file will contain the list of all the computers and services we want to monitor, and will be called `mysite.cfg`:

```
cfg_file=/usr/local/etc/nagios/mysite.cfg
```

The second change is to comment out all other `cfg_file` lines by prefixing them with a `#` sign. In the minimal install shown here, those other files will not be needed.

To set up the local plugins that will be used on the Nagios machine itself, we also need to copy the resource config file. No internal changes are needed, though:

```
# cp resource.cfg-sample resource.cfg
```

Now we will create that `mysite.cfg` file mentioned earlier. It's based on the minimal sample configuration provided with Nagios:

```
# cp minimal.cfg-sample mysite.cfg
```

The example minimal configuration has Nagios just monitoring itself. This is not greatly useful, so we will add one other computer, or host, to be watched. First, we need another entry in the Hosts section. This is easiest made by copying the existing `localhost` entry and just changing the name and address:

```
define host{
      use                     generic-host
      host_name               paranoid
      alias                   Being Watched
      address                 10.0.0.7
      check_command           check-host-alive
      max_check_attempts      10
      notification_interval   120
      notification_period     24x7
      notification_options    d,r
      contact_groups          admins
}
```

In a similar fashion, we'll add another group:

```
define hostgroup{
     hostgroup_name       watched
     alias                Monitored Machines
     members              paranoid
}
```

If you have multiple hosts being monitored, they would be entered as a comma-separated list on the members line.

Finally, we'll add a service. Each service is one thing that is monitored on an individual host. A check for HTTP would be a service, as would a free disk space check. Our single service will be a simple ping to see if the host is up.

```
define service{
     use                          generic-service
     host_name                    paranoid
     service_description          PING
     is_volatile                  0
     check_period                 24x7
     max_check_attempts           4
     normal_check_interval        5
     retry_check_interval         1
     contact_groups               admins
     notification_interval        960
     notification_period          24x7
     check_command                check_ping!100.0,20%!500.0,60%
}
```

That's it for `mysite.cfg`.

Now we'll set up the last Nagios config file, `cgi.cfg`. As you might expect, it sets up CGI parameters, controlling what can be done by users on the Nagios web page.

```
# cp cgi.cfg-sample cgi.cfg
```

Editing `cgi.cfg`, we'll uncomment the long line that says:

```
nagios_check_command=/usr/local/libexec/nagios/check_nagios \
     /var/spool/nagios/status.log 5 '/usr/local/bin/nagios'
```

We'll also add a line to let our authorized user (guest) see information on all hosts being monitored:

```
authorized_for_all_hosts=guest
```

Finally, we'll have Nagios check the configuration for problems:

```
# nagios -v /usr/local/etc/nagios/nagios.cfg
```

Problems are usually caused by path or punctuation errors. Fix them and try again until Nagios reports that “things look okay.”

Install Apache

Again, an easy FreeBSD port:

```
# cd /usr/ports/www/apache2
# make install clean
# rehash
```

Enable Apache2 in /etc/rc.conf:

```
apache2_enable="YES"
```

Configure Apache

A few settings need to made in the Apache config file, `httpd.conf`. As usual, FreeBSD places these files in /usr/local/etc, and the file we need is in a subdirectory called apache2:

```
# cd /usr/local/etc/apache2
```

We need to tell Apache where the Nagios CGI programs and files live. Adding this section to `httpd.conf` just after the first `ScriptAlias` already in that file will do it:

```
# Nagios
ScriptAlias /nagios/cgi-bin /usr/local/share/nagios/cgi-bin
<Directory "/usr/local/share/nagios/cgi-bin">
    AllowOverride AuthConfig
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>

Alias /nagios /usr/local/share/nagios
<Directory "/usr/local/share/nagios">
    Options None
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
# end of Nagios section
```

Now we will set up the files to get Apache to authenticate our guest user. Create a file called `/usr/local/share/nagios/cgi-bin/.htaccess`with these contents:

```
AuthName "Nagios Access"
AuthType Basic
AuthUserFile /usr/local/etc/nagios/htpasswd.users
require valid-user
```

And finally, create a password for that web user:

```
# htpasswd -c /usr/local/etc/nagios/htpasswd.users guest
```
(enter password for guest user)

Start Nagios And Apache

```
# /usr/local/etc/rc.d/nagios.sh start
# /usr/local/etc/rc.d/apache2.sh start
```

If your Nagios computer has a DNS hostname of “naggy”, you should now be able to see the Nagios screen at

http://naggy/nagios

Click on “Service Detail” on the left, and Apache will ask for a username and password. Use the username **guest** and enter the password you defined above. The status of each host's services should start out as unknown. After a few minutes, they will update as they are tested.

More Information

http://www.freebsd.org

http://www.nagios.org