Final Project

APPLYING Nagios
ON THE NETWORK OF THE INDUSTRIAL INFORMATICS DEPARTMENT OF T.E.I. KAVALAS

BY

ESTEBAN MURILLO RANEDO

Kavala, Greece
June 2005
DEDICATION

To my parents, Jose Maria and Mari Ramos, and my sister, Ewa, for being always there for me. They give me the possibility of studying in Alcalá de Henares and the possibility to come to Kavala.

Esteban Murillo Ranedo
ACKNOWLEDGEMENTS

To Prof. Avgoustos Tsinakos for being my thesis supervisor, providing me with great advice and ideas in order to complete successfully my essay.

To Prof. George Kyranastasis for helping me with my exams and supervise my project.

To T.E.I. of Kavala and to my University in Alcala de Henares for giving me the opportunity to do my final project in Kavala.

Thank you to Birgit for showing me the opportunity of the Erasmus program and for being always on my side.

I would like to thank all the Greek people for making me feel like at home. Special thanks to Sotos, Bagelis, Miltos and Mitsos for making me feel as part of their “family” and to Stan for helping me.

Also thank you to all Erasmus people for the great moments that we have spent together. In special to Itziar, Ainara, Jucar, Laura, Cristina and Lucas for making these four months one of the best periods in my life. Thank you to all.

Esteban Murillo Ranado
INDEX

INDEX OF FIGURES  7
INDEX OF TABLES  8

SUMMARY  9

NETWORK MONITORING SOFTWARE  10

WHAT THE PROGRAM MUST DO?  11
WHAT NAGIOS CAN DO  11
ARE MANY PEOPLE USING AND WORKING ON IT?  12
THE NAGIOS’ NUMBERS AND PERCENTAGES  13
THERE IS ENOUGH SUPPORT?  15
NAGIOS’ SUPPORT  15
CONCLUSION  15
OTHER MONITORING UTILITIES  16

NAGIOS  17

FEATURES  17
NAGIOS RECOMMENDATIONS  18
LICENSE  19

PROCEDURE FOR THE INSTALLATION  20

SYSTEM REQUIREMENTS  20
PREREQUISITE BEFORE NAGIOS  21
INSTALLING NAGIOS  22
UNPACKING THE DISTRIBUTION  23
ADDING A USER  23
RUN THE CONFIGURE SCRIPT  24
COMPILE BINARIES  24
INSTALLING THE BINARIES AND HTML FILES  25
INSTALLING AN INIT SCRIPT  25
DIRECTORY STRUCTURE AND FILE LOCATIONS  25
INSTALLING THE PLUGINS  26
NAGIOS PLUGIN REQUIREMENTS  28
SETTING UP THE WEB INTERFACE  28
CONFIGURE SCRIPT ALIAS FOR THE CGIS  28
CONFIGURE ALIAS FOR THE HTML FILES  29
RESTART THE WEB SERVER  30
VERIFY THE CHANGES  30
CONFIGURING WEB AUTHENTICATION  30

Esteban Munilla Raneda
PROCEDURE FOR THE CONFIGURATION

OBJECT CONFIGURATION FILES
CONFIGURING HOSTS
MONITORING SERVICES
CONFIGURING HOSTGROUPS
CONFIGURING CONTACTGROUPS

NETWORK STRUCTURE

CHECKING THE NETWORK

CONFIGURING THE IP’S
  Example

CHECKING THE SERVICES

PING
  PING RESULTS
HTTP
  HTTP RESULTS
FTP
  FTP RESULTS

FUTURE PROPOSALS

APPENDIX A

NAGIOS PLUGIN REQUIREMENTS

APPENDIX B

INSTALLATION OF THE WINDOWS CLIENT SOFTWARE (NSCLIENT)
  OBTAIN THE NSCLIENT PROGRAM
  INSTALLING NSCLIENT
  UNINSTALLATION
  CONFIGURATION

APPENDIX C

INSTALLATION OF THE LINUX CLIENT SOFTWARE (NRPE)
  OBTAIN THE NRPE CLIENT PROGRAM
  INSTALLING NRPE

APPENDIX D

Esteban Murillo Ranedo
## APPENDIX E

**CODE OF THE .CFG FILES CONFIGURED.**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST.cfg</td>
<td>72</td>
</tr>
<tr>
<td>SERVICES.cfg</td>
<td>72</td>
</tr>
<tr>
<td>HOSTGROUPS.cfg</td>
<td>88</td>
</tr>
<tr>
<td>CONTACTGROUPS.cfg</td>
<td>90</td>
</tr>
<tr>
<td>NAGIOS.cfg</td>
<td>91</td>
</tr>
<tr>
<td>TIMEPERIODS.cfg</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>102</td>
</tr>
</tbody>
</table>

**IMAGES REFERENCES**

104

**TABLE REFERENCES**

106

**BIBLIOGRAPHY**

107
INDEX OF FIGURES

Fig.1: Summary of some monitoring programs and service .................... 14
Fig.2: What monitoring software do you use?........................................... 15
Fig.4: Top Organization Types Using Nagios........................................... 16
Fig.5: Network parameters ................................................................... 23
Fig.7: Nagios authentication ................................................................... 35
Fig.8: Network Structure ....................................................................... 46
Fig.9: Nagios system ............................................................................. 47
Fig.10: Nagios script ............................................................................ 48
Fig.11: Nagios welcome page .................................................................. 48
Fig.12: Error: Nagios is not running ..................................................... 49
Fig.13: Route command ....................................................................... 50
Fig.15: Nagios Service Overview ............................................................. 51
Fig.16: Route add command ................................................................... 52
Fig.17: Nagios Service Overview ............................................................. 53
Fig.19: Service Status Details For Host Group ‘UNIX’ .............................. 56
Fig.20: Service Status Details For The Host Group ‘servers’ .................... 58
Fig.21: Service HTTP on host UNIX mailhost (iiwm) ............................... 59
Fig.22: Service HTTP on host Windows Lab Server (router) .................... 60
Fig.23: Service Status Details For The Host Group ‘servers’ .................... 62
INDEX OF TABLES

Fig. 3: Overview Status .............................................................. 16

Fig. 6: Nagios Directory Structure ........................................... 28

Fig 14: Network parameters .................................................... 51

Fig. 18: Services monitored ...................................................... 54
SUMMARY

Nowadays if you want to administer one big network to support its correct operation the best way is to use a system of network monitoring.

Network monitoring is the act of recording all computer and Internet activity that occurs on the PC's in your network.

The project's scope is monitoring the network of the Informatics and Industrial Department of the T.E.I. (Technological Educational Institute) of Kavala, in order to see how are the connexions of the servers among them and with the final hosts, and to check the correctly performance of the different services in each one of the servers. For this finality, I will apply and configure the program of network monitoring Nagios on the testing of three laboratories. The computers of two of them will be running Windows, the third one will be running UNIX, and the services I will check will be PING, HTTP and FTP.
NETWORK MONITORING SOFTWARE

Network monitoring software is designed to help users and administrators monitor local area networks, wide area networks, metropolitan area networks, and all network equipment components. With this software implemented, they will be able to get alerts when network equipment fails, monitor performance of network equipment, track and troubleshoot network related issues, provide reports about the network and equipment, and much more.

A network monitoring software can be used in institutions to monitor the use of their public lab workstations. All workstations and their services can be monitored from a central location to ensure they are working correctly. One user from any location can monitor multiple rooms and networks for convenient real-time monitoring.

Depending on the needs and the network monitoring software, the administrator can hope to gain several benefits from utilizing network monitoring software in his environment. Some of the benefits may include are:

- Network monitoring can help in achieving an SLA (Special Libraries Association) agreement
- Get timely notifications when failures occurs
- Allows you to monitor the health of your network equipment
- Manage your entire network from a central location
- Provides graphical, customized, and real-time reporting of your network
- Reduce necessary down time

There is commercial software out there that seemed to do what the project wanted, but it is often expensive. That open-source is the best option.
There is a large collection of open-source monitoring software. However, unlike some applications where it is know which solution is best - for example, the dominant open-source web server is Apache - in network monitoring there is no clear winner, although Nagios is the most used option. To devise a way to choose, I applied these three criteria:

**What the program must do?**

The program must monitor both UNIX and Windows servers, and monitor those parameters that are of interest: equipment status and service availability and quality.

**What Nagios Can Do**

Nagios monitors a wide variety of system properties, including system-performance metrics such as load average and free disk space; the presence of important services like HTTP and SMTP; and per-host network availability and reachability. It also allows the system administrator to define what constitutes a significant event on each host--for example, how high a load average is "too high"--and what to do when such conditions are detected.

In addition to detecting problems with hosts and their important services, Nagios also allows the system administrator to specify what should be done as a result. A problem can trigger an alert to be sent to a designated recipient via various communication mechanisms (such as email, UNIX message, and pager). It is also possible to define an event handler: a program that is running when a problem is detected. Such programs can attempt to solve the problem encountered, and they can also proactively prevent some serious problems when they get triggered by warning conditions.
In the last figure (fig.1), there is a summary of some open source monitoring programs and services. The only ones that fulfil the necessities of the project are benu, Big Sister, Nagios and Zabbix.

<table>
<thead>
<tr>
<th>The Angel Network Monitor</th>
<th>Access Through</th>
<th>PUBS</th>
<th>CPU</th>
<th>Memory</th>
<th>Ftp:</th>
<th>Processor</th>
<th>SNMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>benu</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Big Sister</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Event Monitor</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ganglia</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nagios</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OpenCST</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>EditW</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SitePro</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SitePro</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zabbix</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Summary of some monitoring programs and services (Juan Antonio, 2004)

In the last figure (fig.1), there is a summary of some open source monitoring programs and services. The only ones that fulfil the necessities of the project are benu, Big Sister, Nagios and Zabbix.

**Are many people using and working on it?**

About other users and developer status are equally important. On open-source sites like SourceForge, someone can easily check a lot of projects that seem to do what he wants, but either aren't used by anyone else or have had no development done on them for years. For software he is going to rely on, it is important to know there is a community that can help. A good way to check is to examine any mailing list that comes with the package. If there is not one, or it shows no postings for a year or so, there may be a reason.

Esteban Murillo Ranedo
The Nagios' numbers and percentages

If someone looks for Nagios in the web, he will realise that there is a big community website at www.nagiosexchange.org, and there are a lot of websites about the program, how to install and configure it, FAQs, forums, documentation, mailing list,... with www.nagios.org like the official web.

In the following figure (fig.2), a poll shows that Nagios is the most used monitoring software.

![Poll showing the most used monitoring software]

**Note:** Nagios was formerly known as Netsaint. Netsaint configuration files are compatible with Nagios, although Nagios has adopted a new, simpler syntax. It is possible also convert Netsaint configurations files with the included convertcfg utility.
In the following figure (fig.3) there is the last information about the number of users, host and services monitored.

<table>
<thead>
<tr>
<th>Stat</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Registered User Profiles</td>
<td>857</td>
</tr>
<tr>
<td># of Monitoring Hosts</td>
<td>2,157</td>
</tr>
<tr>
<td># of Hosts Monitored</td>
<td>118,832</td>
</tr>
<tr>
<td># of Services Monitored</td>
<td>505,354</td>
</tr>
<tr>
<td>Avg. (Mean) # of Monitoring Hosts Per Profile</td>
<td>2.5</td>
</tr>
<tr>
<td>Avg. (Mean) # of Hosts Monitored Per Profile</td>
<td>138.7</td>
</tr>
<tr>
<td>Avg. (Mean) # of Services Monitored Per Profile</td>
<td>589.7</td>
</tr>
<tr>
<td>Avg (Mean) Service To Host Ratio</td>
<td>4.3 : 1</td>
</tr>
</tbody>
</table>

**Fig. 3**
*Overview Status* (Nagios, 2004)

In the next figure (fig.4), there is a resume with the organizations types that are using the program Nagios.

**Top Organization Types Using Nagios**

Legend:
- 37.1% (320) Corporation
- 15.2% (132) Other
- 14.7% (120) Small Business
- 13.3% (116) Unspecified
- 10.9% (95) Educational Institution
- 5.8% (51) Government
- 2.9% (25) Non-Profit Organization
- 0.1% (1) Military

**Fig. 4**
*(Nagios, 2004)*

Esteban Murillo Ranedo
There is enough support?

When someone has a problem is good to know that he will be able to repair it as soon as possible and with the best guarantee. It is a very important requirement for a program of which he is going to rely on.

Nagios' support

If someone needs assistance getting Nagios up and running, there are a number of options available to him. Support options for Nagios include several self-service resources, as well as professional consulting and support services.

Self-Service Support Options

• Online documentation
• FAQs
• Mailing lists
• Forums

Professional Support Options

If someone needs professional consulting services for Nagios, or if he is looking to purchase a support contract for Nagios, help is available in http://www.nagios.org/support/commercial/

Conclusion

Nagios is a very powerful, very flexible monitoring solution, with many plugins available to do almost anything, and with a seemingly endless number of options for notification and service monitoring. The best part, though, is that the layout and design of Nagios makes it amazingly easy to drop in your own ideas that may be specific to your environment's needs.
Other Monitoring Utilities

There are other network monitoring utilities available besides Nagios. I think Nagios is a good contender, but I am obviously biased. Here there are the names of some of them:

Angel Network Monitor
  Autostatus
  Big Brother
  HiWAYS
  MARS
  Mon
  NocMonitor
  NodeWatch
  Penemo
  PIKT
  RITW
  Scotty
  Spong
  Sysmon

In [www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html](http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html) there is a classification with more network monitoring utilities (public and commercial) and their links.
"How can a system administrator monitor a large number of machines and services to proactively address problems before anyone else suffers from them?

The answer is Nagios."

Network monitoring with Nagios, part one. Bernard Golden, 13 Apr 2005

"nagios-1.0.1 – Extremely powerful network monitoring system."

Free BSD, package information

Nagios® is a host and service monitor designed to inform you of network problems before your clients, end-users or managers do. It has been designed to run under the Linux operating system, but works fine under most *NIX variants as well. The monitoring daemon runs intermittent checks on hosts and services you specify using external "plugins" which return status information to Nagios. When problems are encountered, the daemon can send notifications out to administrative contacts in a variety of different ways (email, instant message, SMS, etc.). Current status information, historical logs, and reports can all be accessed via a web browser.

Features

Nagios has a lot of features, making it a very powerful monitoring tool. Some of the major features are listed below:

- Monitoring of network services (SMTP, POP3, HTTP, NNTP, PING, etc.)
- Monitoring of host resources (processor load, disk and memory usage, running processes, log files, etc.)
- Monitoring of environmental factors such as temperature
- Simple plug-in design that allows users to easily develop their own host and service checks
- Ability to define network host hierarchy, allowing detection of and distinction between hosts that are down and those that are unreachable
- Contact notifications when service or host problems occur and get resolved (via email, pager, or other user-defined method)
- Optional escalation of host and service notifications to different contact groups
- Ability to define event handlers to be run during service or host events for proactive problem resolution
- Support for implementing redundant and distributed monitoring servers
- External command interface that allows on-the-fly modifications to be made to the monitoring and notification behaviour through the use of event handlers, the web interface, and third-party applications
- Retention of host and service status across program restarts
- Scheduled downtime for suppressing host and service notifications during periods of planned outages
- Ability to acknowledge problems via the web interface
- Web interface for viewing current network status, notification and problem history, log file, etc.
- Simple authorization scheme that allows you restrict what users can see and do from the web interface

**Nagios Recommendations**

Like all network management tools, Nagios is fairly complex to set up and requires ongoing tuning to ensure that the level of information provided is
right -- neither too much detail nor too little information. Here are some recommendations about how to get the best use of your Nagios implementation:

- Begin by planning what it is needed to keep track of, prioritized by most important resources first.
- Work incrementally, first getting those most-important resources under management before moving on to less-important resources. For example, in most organizations e-mail is more important than FTP availability, so begin by putting e-mail under Nagios management. Working incrementally can reduce the burden of implementing a network management system.
- Plan on regular reviews of the type and level of information you are getting, especially in the first few months. The purpose of these reviews is to get the system configured properly so that you can then use it on an ongoing (relatively) easy basis.
- Take advantage of the Nagios community. A large number of sample configurations, dashboard extensions, and custom plugins are available, which can make it easier to get Nagios implementation up and running.
- Document configuration. Comment configuration files are clear to know what resources are managing and what plugins are running. Also, write up some external documentation on Nagios implementation so that someone can pick it up later and get a good overview of how management scheme works.

License

Nagios is licensed under the terms of the GNU General Public License Version 2 (www.gnu.org/copyleft/gpl.html#TOC1) as published by the Free Software Foundation. This gives legal permission to copy, distribute and/or modify Nagios under certain conditions.
“Installing and configuring Nagios is rather involved. You can’t just compile the binaries, run the program and sit back. There’s a lot of setup before you can start monitoring anything. Relax, take your time and read all the documentation – you’re going to need it. Okay, let’s get started....”

Ethan Galstad (main developer of the Nagios)
“Nagios version 1.0 Documentation”, Installing Nagios

This section provides only an example configuration. The description that follows has been successfully installed and tested as a viable platform for running Nagios. At the same time, this method of configuration is only one of a wide variety of setups that can be used, and is therefore not exhaustive.

The first step is to be sure the workstation or server that will be hosting Nagios fulfills the necessary requirements.

**System Requirements**

The only requirement of running Nagios is a machine running Linux (or UNIX variant) and a C compiler. I also want to have TCP/IP configured, as most service checks will be performed over the network. In addition, it is recommended at least 256 megabytes of RAM, and a good starting point for the hard drive size is twenty gigabytes.

It is not required to use the CGIs included with Nagios. However, if someone decides to use them, he will need to have the following software installed...
1. A web server (preferably Apache)
2. Thomas Boutell's gd library version 1.6.3 or higher (required by the statusmap and trends CGIs)

The computers of the UNIX laboratory do not fulfill these requirements, at least with respect to minimum RAM memory and hard disk size recommended, so it is necessary to run the program in my own laptop. However, for this finality it is necessary Linux and Apache running in my computer, so it is the next step.

**Prerequisite before Nagios**

Install Linux Mandrake 10.0 in my laptop, and download and install the Apache web server from the [www.apache.org](http://www.apache.org) website. Nagios uses a web interface for the front-end of the program. Additionally, I must be prepared to make modifications to the httpd.conf file to accommodate Nagios. An additional program that can be downloaded is the Comanche user-interface that serves as a front-end for the Apache program. Refer to the documentation that comes with the download for proper installation.

It is necessary to configure the network parameters to connect the laptop to the network of the T.E.I.

![Network parameters](image)

**Fig. 5** Network parameters (Esteban, June 2005)
The last figure (fig.5) is a printscreen of the Manage connections of the computer that is running Nagios.

Now all this prepared to begin with the installation of Nagios.

**Installing Nagios**

As far as the install goes, it really consists of the following: unpacking the distribution, adding a user, building, configuring apache, and getting/installing the plug-ins. The app is still cute without plug-ins, but unfortunately worthless.

The first step is obtaining the Nagios core program from the appropriate website. At this writing, the URL for the website is [http://www.nagios.org](http://www.nagios.org).

```
nagios-1-x-cvs.tar.gz
```

Then obtain the plug-ins that Nagios uses to monitor the network. This is an important point. The core program is strictly the engine of the program, and does not have any monitoring functionality of its own. The plugins are necessary for the network monitoring that will be taking place. I found the plug-in files in the same website:

```
nagios-plugins-1.4.tar.gz
```

The various nodes on the network will need a client program in order for them to be able to communicate with the Nagios application server. There are two such programs: NSClient (Windows) and NRPE (Linux). The homepage of NSClient is [http://nsclient.ready2run.nl](http://nsclient.ready2run.nl), while NRPE can be downloaded from the Nagios homepage (see Appendix B and C for the installation information).
Unpacking the Distribution

To unpack the Nagios distribution, type the following two commands at a shell prompt:

```
unzip nagios-1.0.zip
```

Note: If someone downloaded the ZIP version of the distribution, he must type the following:

```
unzip nagios-1.0.zip
```

After finished executing these commands, find a nagios-1.0 directory that has been created in the current directory. Inside that directory, find all the files that compromise the core Nagios distribution.

Create the installation directory for the program. The root or user account should be used when this and future steps are done to insure there are no permissions issues later during the installation. The syntax to use

```
mkdir /nagios
```

Adding a User

To run Nagios under a normal user account, add a new user (and group) to the system with the following commands (these will vary depending on what OS you are running):
Run the Configure Script

Run the configure script to initialize variables and create a Makefile. This shell script will automatically create the necessary directory structure for the program. The generic version of the script is the following

```
./configure --prefix=prefix --with-cg
with-htmurl=htmurl --with-nagios-user=someuser
nagios-grp=somegroup
```

- Replace `prefix` with the installation directory that was created in the step above (default is `/usr/local/nagios`)
- Replace `cgiurl` with the actual URL that will be used to access the CGIs (default is `/nagios/cgi-bin`). Do not append a slash at the end of the url.
- Replace `htmurl` with the actual url that will be used to access the HTML for the main interface and documentation (default is `/nagios/`)
- Replace `someuser` with the name of a user on the system that will be used for setting permissions on the installed files (default is `nagios`)
- Replace `somegroup` with the name of a group on the system that will be used for setting permissions on the installed files (default is `nagios`)

Compile Binaries

Compile Nagios and the CGIs. These are the executable files needed for Nagios' operation. The command syntax is:
Installing the Binaries and HTML Files

Install the Binaries and HTML Files. After the binaries are compiled (created), the next step is to actually install them in their appropriate directories. This is coupled with doing the same for the html files. The command to accomplish this task is:

```
Installing the Binaries and HTML Files
```

Installing an Init Script

This is the start-up script used by Nagios to automatically start the program upon system boot. The utility program is created in the /etc/rc.d/init.d/nagios directory, and can be modified to reflect the correct paths for the operating system and Nagios installation. The command syntax is:

```
It is necessary to edit the init script to make sense with particular OS and Nagios installation by editing paths, etc.
```

Directory Structure and File Locations

Change to the root of your Nagios installation directory with the following command:

```
Verify the directory structure of Nagios is correct. The location of the program, by default, is /usr/local/nagios. Executing the command ls -l will provide a detailed view of the infrastructure. They should be five different
```

Esteban Murillo Raneda
subdirectories. A brief description of what each directory contains is given in the table below (fig.6).

<table>
<thead>
<tr>
<th>SUB-DIRECTORY</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bin/</strong></td>
<td>Nagios core program</td>
</tr>
<tr>
<td><strong>Etc/</strong></td>
<td>Main, resource, object, and CGI configuration files should be put here</td>
</tr>
<tr>
<td><strong>sbin/</strong></td>
<td>CGIs</td>
</tr>
<tr>
<td><strong>share/</strong></td>
<td>HTML files (for web interface and online documentation)</td>
</tr>
<tr>
<td><strong>Var/</strong></td>
<td>Empty directory for the log file</td>
</tr>
</tbody>
</table>

Fig. 6
Nagios Directory Structure (Ethan Galstad)

**Installing the Plugins**

At this point, the Nagios installation is complete. However, it is not very useful at its current state, because it lacks the actual monitoring applications. In order for Nagios to be of any use, it is necessary to download and install some plugins. Plugins are usually installed in the `libexec/` directory of the Nagios installation (i.e. `/usr/local/nagios/libexec`). Plugins are scripts or binaries that perform all the service and host checks that constitute monitoring. It is possible to grab the latest release of the plugins from the Nagios downloads page or directly from the SourceForge project page. Nagios comes with a default set of such plugins, but they have to be downloaded and installed separately.

It is necessary to run the `configure` script that is provided in order to prepare the package for compilation on the system. You will find that the plugins are installed in a fashion similar to the actual Nagios program. Once again, it will be just able to run `configure` if all is OK with the default settings for the username, group, and directory where Nagios is installed.
Download the latest Nagios Plugins package and decompress it:

```
tar xzvf nagios-plugins-1.4.0alpha1.tar.gz
```

```
cd nagios-plugins-1.4.0alpha1/
```

Type the following script on a single line:

```
./configure --with-nagios-group=apache
--prefix=/usr/local/nagios
--with-nagios-user=nagios
--th-nagios-group=nagios
```

It might give notifications about missing programs or Perl modules while configure is running. These are mostly OK, unless the mentioned application was specifically needed to monitor a service.

Once configure is complete, compile all of the plugins.

```
make all
```

If no errors were reported, all is ready to install the plugins.

```
make install
```

The plugins will be installed in the libexec directory of the Nagios base directory (/usr/local/nagios/libexec, in my case).

```
cd /usr/local/nagios/libexec/
```

There are a few rules that all Nagios plugins should implement, making them suitable for use by Nagios. All plugins provide a -help option that displays information about the plugin and how it works. This feature helps a lot when you are trying to monitor a new service using a plugin you have not used before.

For instance, to learn how the check_ssh plugin works, run the following command.
This shows us that the check_ssh plugin accepts one required parameter host, and two optional parameters, timeout and port.

There is nothing especially complicated about the plugins. In fact, it is possible to run the plugins manually to check services on the console.

Nagios Plugin Requirements

Some plugins require have additional programs and/or libraries installed on the system before they can be used. Plugins that are dependent on other programs/libraries that are missing are usually not compiled. Requirements for various plugins are listed in Appendix A.

Setting Up The Web Interface

Apache web server is running in the computer where Nagios is running. If someone is using some other web server, he will have to make changes where appropriate. I am also using the /usr/local/nagios as the installation prefix.

Configure Script Alias for the CGIs

It is necessary to create an alias for the CGIs as well. The default installation expects to find them accessible at http://yourmachine/nagios/cgi-bin, although this can be changed using the --with-cgiurl option in the configure
Anyway, add something as the following to the web server configuration file (i.e. `httpd.conf`) (changing it to match any directory differences on the system)...

```
ScriptAlias /r.·.-:·· ^c/cgi-bin/
  Directory /usr/local/nagios/sbin/
  AllowOverride AuthConfig
  Options ExecCGI
  Order allow,deny
  Allow from all
```

**Important!** The Script-Alias line above must come before the Alias line below. Otherwise, Apache will parse the lines differently.

**Configure Alias for the HTML Files**

In order to make the HTML files accessible via the web server, it is necessary to edit the Apache configuration file.

Add the following to the web server configuration file (i.e. `httpd.conf`) as follows:

```
Alias /nagios/ /usr/local/nagios/share/
  Directory "/usr/local/nagios/share"
  Options None
  AllowOverride AuthConfig
  Order allow,deny
```

This will allow to use an URL like `http://yourmachine/nagios/` to view the HTML web interface and documentation. The alias should be the same value...
that was entered for the \textit{--with-htmurl} argument to the configure script (default is \\
\texttt{/nagios/}).

\textbf{Important!} The \texttt{Alias} directive just added for the HTML files must come after the \texttt{ScriptAlias} directive for the CGIs. If it does not, it will appear a 404 error when attempting to access the CGIs.

\section*{Restart the Web Server}

Once Apache configuration file has been finished editing, it is necessary to restart the web server.

\texttt{.\init.d/httpd restart}

\section*{Verify The Changes}

Do not forget to check and see if the changes made to Apache work. It should be able to point the web browser at \texttt{http://yourmachine/nagios} and get the web interface for Nagios. The CGIs may not display any information, but this will be remedied once configure everything and start Nagios.

\section*{Configuring Web Authentication}

Once the web interface is properly configured, it is necessary to enable web server authentication for accessing the CGIs and configure user authorization information.

Configure Web Authentication by enabling access for specific users and/or groups to the CGI's. The first step is to make sure the \texttt{httpd.conf} file
contains the AuthOverride AuthConfig statement for the Nagios CGI-BIN directory. If it does not, the following entry, shown on the next page, should be added and the web server should be subsequently rebooted.

```
<Directory /usr/local/nagios/sbin>
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
    Options ExecCGI
</Directory>
```

If authentication for access to the HTML pages in Nagios is desired, the following should also be added to the httpd.conf file.

```
<Directory /usr/local/nagios/share>
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

Set up authenticated users for Nagios. Use the htpasswd command that is bundled with Apache. Running the program will create a new file called .htaccess in the next directories:

```
/usr/local/nagios/sbin/.htaccess
/usr/local/nagios/share/.htaccess
```

Each one with the next information:

- AuthName "Nagios access"
- AuthType Basic
- AuthUserFile /usr/local/nagios/etc/htpasswd.users
- Require valid-user
While running the program, a password will be requested to be associated with users who will be authenticating to the web server. The example below shows the syntax of the command when entering the nagios administrator user (lufth_m):

```bash
htpasswd -c /usr/local/nagios/etc/htpasswd.lufth_m
```

The process can be continued to add more users as needed to the list of authenticated users. The syntax used for adding more users is similar to the one shown above except the -c option should not be used. This option actually creates the file, and using it again would overwrite the existing file with another one. Hence, the correct syntax for additional users would be:

```bash
htpasswd /usr/local/nagios/etc/htpasswd.users user
```

Enable authentication and authorization functionality to the CGI files. This is accomplished by editing the cgi.cfg file. The user_authentication variable should be modified to a non-zero value (typically 1) to activate the basic feature.

The correct syntax is the following:

```ini
Lon=l.
authorized_for_system_information=<username>
authorized_for_system_commands=<username>
authorized_for_configuration_information=<username>
authorized_for_all_hosts=<username>
authorized_for_all_host_commands=<username>
authorized_for_all_services=<username>
authorized_for_all_service_commands=<username>
```

Where `username` is the name of the user that was created the last step `lufth_m`.  

Esteban Munkio Ranedo
Each one of these authorization lines allow the user who is logging into the machine to be able to view from the web browser the information that would be displayed on the screen. If the username is not entered, or if the authorization line was left commented out, the screen in question would display a message indicating the permissions were not valid.

Now, when we want to access to the Nagios web interface, we must write the Login and the Password of the Nagios’ allowed user.
PROCEDURE FOR THE CONFIGURATION

First, I will start with a relatively simple configuration. To get anything useful out of Nagios, there are four things, at a minimum, that need to be configured. They are hosts, host groups, contacts, and services. In the directory nagios/etc/sample/ there is sample configuration files to give an idea of how things work.

The logic behind configuring Nagios is not so difficult, but it is needed to spend enough time. There are hosts, on which presumably run services. Hosts providing the same services can be grouped together into host groups for easy summarization in the web front end. Likewise, organization probably has contacts for the different services. If there is more than one contact for a particular service, it is possible to put these contacts together under an alias or contact group. If a machine Nagios monitors goes down or loses a service it has been running, Nagios can be configured to notify the proper contact or group for that host or service.

Object Configuration Files

The bulk of Nagios configuration occurs in the object configuration files. These files define hosts and services to be monitored, how various status conditions should be interpreted, and what actions should be taken when they occur. These files are used to define the following items:

- **HOSTS** Computers and other network devices
- **HOST GROUPS** Named groups of hosts
- **SERVICES**: Important daemons providing specific network services
- **CONTACTS**: User to be contacted in the event of a problem
- **CONTACT GROUPS**: Named groups of contacts
- **Time Periods**: Day and/or time ranges within a week, used to specify when checks are to be performed, notifications are to be sent, and the like
- **Commands**: Commands to be run for all purposes (host/service checking, notifications, event handling, and so on). Nagios provides two files containing many predefined commands: `checkcommands.cfg` and `misccommands.cfg`.
- **Host Dependencies**: Specifications of host reachability dependencies. When an intermediate host is down, checks are skipped for all hosts that are dependent on that one.
- **Service Dependencies**: Specifications of service dependency requirements. When a service host is down, checks are skipped for all other services that are dependent on it.
- **Host Escalations**: Definitions of optional escalation levels for host problems
- **Host Group Escalations**: Definitions of optional escalation levels for host groups
- **Service Escalations**: Definitions of optional escalation levels for failed services

The items in capital letters will need to be defined for virtually every Nagios installation, the ones in small letters are optional. In the sample Nagios configuration provided with the package, each type of object is defined in a separate configuration file (named after the object type, excluding any spaces). However, it is possible to arrange new definitions in any form that makes it easier to understand.

There is a map of the configure files in Appendix D.
Configuring Hosts

The order will be like in the above paragraph in the configuration, so it will be possible to refer back to it if the process gets numb. Let us create two hosts. Here there are two from my test configuration, preceded by the generic host definition template:
This is a small host configuration file. The first entry will save some typing, since "generic-host" is just a template. In the other two entries here, I have put use generic-host that automatically sets generic-host's settings for all of the hosts that use it. Line 1 of the template assigns a name. Line 2 allows to turn notifications on and off, which is great for keeping an inbox from exploding during testing with a large number of hosts. Line 3 enables event handling, which allows to define a set of actions to take when Nagios detects a change in the state of a host or service it has monitoring. Line 4 protects the inbox or pager in the event that a service or host is intermittently (and frequently) changing state due to a network anomaly. Line 5 aggregates the data collected from the various hosts and services to give pretty reports as to the availability of the network environment. Lines 6 and 7 cause Nagios to hold on to "last known values" across restarts of Nagios. Keep in mind that this includes the program's own settings! Read the user guide on ways to get around this. The last line tells Nagios not to look at this entry as a normal host and register it as such. It is just a template!

First, I use the generic-host template for both host definitions, so all things that are true for the template are automatically true for the host definitions that use it. The hostname line is supposed to be the actual hostname...
the machine in question goes by, but the alias line is what the Nagios website
titles will say. The check_command line specifies which built-in Nagios
command to use to determine if the host is even up. It is possible to realize
what the check-host-alive command does by looking in the command.cfg file for
the corresponding entry. The notification settings are such that the machines
are monitored 24x7, every 120 seconds, and a host is considered "dead" if the
check fails 10 times. The notification_options line tells nagios, per host, what
will cause a notification to be sent. In my case, I use "d"own, "u"nreachable, and
"r"ecovered. So, if a machine is down, I get a message when it goes down, and
when it recovers from the down state. Using "unreachable" as an option is a
little obsessive if you are in a network where small occasional glitches can
cause one machine or another to become unreachable temporarily.

**Monitoring Services**

Now, there are two hosts configured. At this point, all Nagios knows how
to do is ping them to see if they are alive, though. Let us set up monitoring for
the individual services on those machines that we care about. The configuration
file format is the same for pretty much everything, so this configuration file
should not be too scary by now:

First, again, there is a template entry where it is possible to set flags that
can then essentially be "included" by the other entries, just as it was added for
the hosts.
With the template defined, it is possible to go about the business of configuring actual services, and reference the template to get those settings on a per-service basis.
Note that for the Ping service, it was used a wildcard ('*') for the host_name. This is because I want to monitor that service on every host configured. Most of the other flags are the same as other configuration files, which is good news. Note the check_command here. This command is defined in command.cfg, which will point off to the actual script used to check the service. This is good for two reasons; first, it allows to tweak the script if needed. Second, this means that it is also possible to drop an own script in place to check whatever wacky service might be running, and define an own command, which can then be applied to whatever hosts.
Configuring Hostgroups

Therefore, what happens if only a subset of the hosts is running HTTP. I do not want to have to list every single host that runs HTTP individually, and I do not want to run HTTP everywhere -- what to do? Configure a hostgroup that contains the hostnames of those hosts in the hosts.cfg file that have some similar characteristic. Here is an example of what a hostgroup looks like:

Now, when the HTTP service is defined, instead of using host_name, just use hostgroup_name instead, and all the right things will happen. This is also nice, because as the service is added to different machines, it is possible just to add their hostnames to the right group, and off you go. One quick note, though: if a host is defined with a non-existent contact_group, it will appear errors from Nagios.

Esteban Murillo Ranedo
Configuring Contactgroups

If a service becomes unavailable, the administrator probably wants someone to know about it PDQ (Pretty Damn Quick). After all, if all Nagios did was make shiny pictures in your browser, nobody would use it! Much like there are hosts and host groups, likewise there are also contacts, and contact groups. You define all of the contacts in the contacts.cfg file, and group those contacts into groups in the contactgroups.cfg file. Here is the contact definition:

Here, I have defined a contact (with an alias for easy viewing in a browser), the periods during which this contact will receive host and service outage notifications, what types of notifications I'll get (only "d"own, "u"nreachable and "r"ecovery messages for hosts, for example), the notification mechanism (email in this case), and finally, an email address. Setting up groups of contacts simply requires that the contacts exist in the contacts.cfg file, and they are put together exactly like hostgroups -- just give them a name, an alias, and a list of members, and you are all set. Here is a quick example:
It is a good idea to create all the groups you are likely to need going forward, even if they only have one member. This way, when an administrator (in this case) will be added, it will be necessary only to add him to the contacts file, and then to the group definition, instead of having to hard-code the contact name everywhere it belong.

I have explained the four things, at a minimum, that Nagios needs to be configured. They are hosts, host groups, contacts, and services. The code of the files I have written to run the program at the T.E.I. network is attached in Appendix E. There someone can easily check the configuration of all the hosts, host groups, contact and services I have included in the tests.
I will taste with Nagios the network of the Industrial Informatics Department of T.E.I. (Technological Educational Institute) Kavalas. In order to make easier the understanding of the next pages, I try to show here a general idea about the structure of this network with the following figure (fig. 8).
CHECKING THE NETWORK

Now, all is ready to run Nagios and see the network monitoring results. The Nagios’ structure is like the following figure (fig.9), where it is possible to see the teamwork between Nagios and data basses, servers, browsers, and with all of the plentiful options of extensions that there are.

To start running Nagios, I have to follow the next steps in order:

1. Apache must be configured and running (on boot).
2. Start running Nagios script:
   ```
   sudo service nagios start
   ```
3. Open the web browser (Konkenor in my case) and go to Nagios address:

   http://my computer ip/Nagios   -- General entrance

4. Write the user and password authorization.

5. Now Nagios is running and it is possible to see the welcome page.

Fig. 10 Nagios script (Esteban, 2005)

---

Esteban Munitlo Ranedo
If I try to go in the Nagios program without start previously the script (fig.10), the browser is going to show the following error message (fig.12):

![Nagios Error Message](image)

### Whoops!

Error: Could not read host and service status information!

The most common cause of this error message (especially for new users), in the fact that Nagios is not actually running. If Nagios is indeed not running, this is a normal error message, it simply indicates that the CGIs could not obtain the current status of hosts and services that are being monitored. If you’ve just installed things, make sure you read the documentation on starting Nagios.

Some other things you should check in order to resolve this error include:

1. Check the Nagios log file for messages relating to startup or status data errors.
2. Always verify configuration options using the --check command-line option before starting or restarting Nagios!
3. Make sure you’ve compiled the main program and the CGIs to use the same status data storage options (i.e. text file or database). If the main program is storing status data in a text file and the CGIs are trying to read status data from a database, you’ll have problems.

Make sure you read the documentation on installing, configuring and running Nagios thoroughly before continuing. If all else fails, try sending a message to one of the mailing lists. More information can be found at [http://www.nagios.org](http://www.nagios.org).

---

**Fig. 12 Error. Nagios is not running (Esteban, June 2005)**

---

**Configuring the ip’s**

I have tried to run the program connecting my laptop to the network with different ip’s:

- **Static ip:** 195.130.92.181
- **Dynamic ip’s:** 10.1.19.114 / 10.1.9.70

---

*Esteban Murillo Ranedo*
If only the parameters of the connection of the computer where Nagios is running are configured, it only will be possible to see the equipments with the seem ip rank.

For example, if I take the static ip 195.130.92.181 (fig.13) and I configure the network 195.130.92.0, I only will see the rest of the computer of the network with static ip's and in the same network: router, server and pej14. However, I will not see power, the iiwb## and the peii## computers, because their network is unreachable for me.

For this reason, it is necessary to configure all the ranks of ip's that I want to check if I want to be able to see all the hosts of the network. To make this, I use the route add command:

route add -net 192.130.92.0 netmask 255.255.255.0 dev eth0
route add -net 192.130.93.0 netmask 255.255.255.0 dev eth0
route add -net 10.1.19.0 netmask 255.255.255.0 dev eth0
route add -net 10.1.9.0 netmask 255.255.255.0 dev eth0

![Fig.13 Route command (Esteban, June 2005)](image-url)
Example

First, I will monitor the network only with the next network parameters (fig. 14) to show the difference between when the needed ranks of ip's are not configured, and when they are:

<table>
<thead>
<tr>
<th>Rank of ip's destinations added</th>
<th>192.130.92.181</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.130.0.92</td>
<td></td>
</tr>
<tr>
<td>10.1.19.0</td>
<td></td>
</tr>
<tr>
<td>127.0.0.0 (local)</td>
<td></td>
</tr>
</tbody>
</table>

Fig 14 Network parameters (Esteban, June 2005)

With this configuration, the Service Overview shows the following information (fig. 15):

Service Overview For All Host Groups

Fig 15 Service Overview (Esteban, June 2005)
Only it is possible to see the status of router and server because they belong to the 195.130.92.0 network, their ip's are 195.130.92.90 and 195.130.92.93, but the rest of the hosts look like down because they belong to not configured networks in my computer (195.130.93.0 and 10.1.9.0). For this reason, it is necessary to add these networks like in the next figure (fig.16), and it will be possible to see the real status of all the hosts.

![Fig 16 Route add command (Esteban, June 2005)](image)

Now the Nagios Service Overview shows the real state of all the host of the network, because it is able to see all of them. Someone can easily check the differences between the fig.15 and the fig.17.

The power server, and the iiwb## and the peii## computers are UP in the fig.17. Nevertheless, they are DOWN in the fig.15, because Nagios could not see them.

---

Esteban Murillo Ranedo
Service Overview For All Host Groups

<table>
<thead>
<tr>
<th>Kyranastasis Computers (computers)</th>
<th>Windows Hosts (hosts)</th>
<th>Servers (servers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Status</td>
<td>Services</td>
</tr>
<tr>
<td>host1</td>
<td>DOWN</td>
<td>CRITICAL</td>
</tr>
<tr>
<td>host2</td>
<td>DOWN</td>
<td>CRITICAL</td>
</tr>
<tr>
<td>host3</td>
<td>UP</td>
<td>OK</td>
</tr>
<tr>
<td>host4</td>
<td>UP</td>
<td>OK</td>
</tr>
<tr>
<td>host5</td>
<td>UP</td>
<td>OK</td>
</tr>
<tr>
<td>host6</td>
<td>UP</td>
<td>OK</td>
</tr>
<tr>
<td>host7</td>
<td>UP</td>
<td>OK</td>
</tr>
<tr>
<td>host8</td>
<td>UP</td>
<td>OK</td>
</tr>
</tbody>
</table>

Fig. 17 Nagios Service Overview (Esteban, June 2005)
CHECKING THE SERVICES

Now it is possible to see all the hosts of the network, and it is able to check which ones are up/down and the correctly operation of the services that are running in each host.

In the next table (fig.18), there is a resume of the services that are checked in each server and host, their name and their alias in the Nagios program:

<table>
<thead>
<tr>
<th>NAME</th>
<th>ALIAS</th>
<th>SERVICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>Kyranastasis Lab Server</td>
<td>Ping, HTTP, FTP</td>
</tr>
<tr>
<td>router</td>
<td>Windows Lab Server</td>
<td>Ping, HTTP, FTP</td>
</tr>
<tr>
<td>server</td>
<td>Unix Lab Server</td>
<td>Ping, HTTP, FTP</td>
</tr>
<tr>
<td>peii##</td>
<td>Kyranastasis Lab Hosts</td>
<td>Ping</td>
</tr>
<tr>
<td>iiwb##</td>
<td>Windows Lab Hosts</td>
<td>Ping</td>
</tr>
<tr>
<td>sun0##</td>
<td>Unix Lab Hosts</td>
<td>Ping</td>
</tr>
<tr>
<td>hp4050</td>
<td>Unix Lab Printer</td>
<td>Ping, Printer Status</td>
</tr>
<tr>
<td>iiwm</td>
<td>Unix Mail host</td>
<td>Ping, HTTP</td>
</tr>
</tbody>
</table>

Fig. 18
Services monitored (Esteban, June 2005)

Esteban Murillo Ranedo
Ping is actually an acronym for the words ‘Packet INternet Groper’. The Ping utility is essentially a system administrator’s tool that is used to see if a computer is operating and also to see if network connections are intact. Ping uses the Internet Control Message Protocol (ICMP) Echo function, which is detailed in RFC 792. A small packet is sent through the network to a particular IP address. This packet contains 64 bytes - 56 data bytes and 8 bytes of protocol reader information. The computer that sent the packet then waits (or ‘listens’) for a return packet. If the connections are good and the target computer is up, a good return packet will be received. PING can also tell the user the number of hops that lie between two computers and the amount of time it takes for a packet to make the complete trip.

I have checked the PING service for the entire host, like is explained in the page 39:

```plaintext
# Service definition
define service {
    use generic-service ; Name of service template

    host_name     
    service_description
    is_volatile
    check_period
    max_check_attempts
    normal_check_interval
    retry_check_interval
    contact_groups
    notification_interval
    notification_period
    notification_options
    check_command
}
```

```
admins
240
24x7
c,r
check_ping!100.0,20%!500.0,60%
```

Esteban Murillo Ranedo
PING results

Here it is possible to see the results of this checking in the Unix Lab. The status information in the following figure (fig.19) shows us some numbers about the service, like packet loss and RTA (Round Trip Average):

- **PING OK – Packet loss = 16%, RTA = 6.29 ms**
- **PING OK – Packet loss = 0%, RTA = 6.51 ms**
- **PING OK – Packet loss = 0%, RTA = 0.81 ms**
- **PING OK – Packet loss = 0%, RTA = 2.49 ms**

![Service Status Details For Host Group 'unix'](Fig. 19 Service Status Details (Esteban, June 2005))

**Esteban Murillo Ranedo**
The program runs correctly, and it detects when some host is Up or Down.

I have checked also the time of update (90 seconds, although it can be changed) shutting down and up one host. The time that the program spend to realize that one host is UP or DOWN is between 2 and 3 updates, so it can be almost five minutes.
Short for HyperText Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

I check the HTTP service in all the servers (power, router and server) and in the mailhost (iiwm) of the Unix Lab. It is possible to see the results in the Service Status Detail For The Host Group 'servers' in the fig 20.

**HTTP results**

The service in the mailhost (iiwm) and in server (Unix Lab Server) is OK, both of them in the Unix Lab. In this case the status information is:

*HTTP OK HTTP/1.1 200 OK – 555 bytes in 0.004 seconds*
The request has succeeded. The information returned with the response is dependent on the method used in the request, for example:

**GET** an entity corresponding to the requested resource is sent in the response;

**HEAD** the entity-header fields corresponding to the requested resource are sent in the response without any message-body;

**POST** an entity describing or containing the result of the action;

**TRACE** an entity containing the request message as received by the end server.

One example of this service is showed in the following image (fig.21):

![Service State Information](image)

<table>
<thead>
<tr>
<th>Current Status:</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Information:</td>
<td>OK - HTTP/1.1 302 Found - 0.004 second response time</td>
</tr>
<tr>
<td>Current Attempt:</td>
<td>1/3</td>
</tr>
<tr>
<td>State Type:</td>
<td>HARD</td>
</tr>
<tr>
<td>Last Check Type:</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>Last Check Time:</td>
<td>06-15-2005 19:07:02</td>
</tr>
<tr>
<td>Status Data Age:</td>
<td>0d 0h 3m 30s</td>
</tr>
<tr>
<td>Next Scheduled Active Check:</td>
<td>06-15-2005 19:12:02</td>
</tr>
<tr>
<td>Latency:</td>
<td>&lt; 1 second</td>
</tr>
<tr>
<td>Check Duration:</td>
<td>&lt; 1 second</td>
</tr>
<tr>
<td>Last State Change:</td>
<td>06-15-2005 17:26:01</td>
</tr>
<tr>
<td>Current State Duration:</td>
<td>0d 1h 44m 31s</td>
</tr>
<tr>
<td>Last Service Notification:</td>
<td>N/A</td>
</tr>
<tr>
<td>Current Notification Number:</td>
<td>0</td>
</tr>
<tr>
<td>Is This Service Flapping?</td>
<td>N/A</td>
</tr>
<tr>
<td>Percent State Change:</td>
<td>N/A</td>
</tr>
<tr>
<td>In Scheduled Downtime?</td>
<td>NO</td>
</tr>
<tr>
<td>Last Update:</td>
<td>06-15-2005 19:10:30</td>
</tr>
</tbody>
</table>

---

(FIG. 21 Service HTTP (Esteban, June 2005))

Esteban Murillo Ranedo
In the case of the rest of the servers (power and router), the service show a message of WARNING (fig.22), with the following status information:

**HTTP WARNING: HTTP/1.1 403 Access Forbidden**

You do not have permission to access to this document on this server. The server understood the request, but is refusing to fulfill it. If the request method was not head and the server wishes to make public why the request has not been fulfilled, it should describe the reason for the refusal in the entity. If the server does not wish to make this information available to the client, the status code 404 (Not Found: The server has not found anything matching the Request-URI) can be used instead.
File Transfer Protocol, or FTP, is a protocol used to upload files from a workstation to a FTP server or download files from a FTP server to a workstation. It is the way that files get transferred from one device to another in order for the files to be available on the Internet. When ftp appears in a URL it means that the user is connecting to a file server and not a Web server and that some form of file transfer is going to take place. Most FTP servers require the user to log-on to the server in order to transfer files.

In contrast, Hyper Text Transfer Protocol, or HTTP, is a protocol used to transfer files from a Web server onto a browser in order to view a Web page that is on the Internet. Unlike FTP, where entire files are transferred from one device to another and copied into memory, HTTP only transfers the contents of a web page into a browser for viewing. FTP is a two-way system as files are transferred back and forth between server and workstation. HTTP is a one-way system as files are transported only from the server onto the workstation's browser. When http appears in a URL, it means that the user is connecting to a Web server and not a file server. The files are transferred but not downloaded, therefore not copied into the memory of the receiving device.

FTP results

I have checked this service in all the servers (power, router and server), but it seem to be down in all of them (fig.23). The Service Status Details For Host Group 'server' show as the following status information for the FTP service:

Connection refused

No data received from host
I think it will be good if there is some FTP service in almost one of the servers to offer the students one way to interchange information, like, for example, this project. I have spent almost of the time to search information about Nagios, about how to install it and configure it. Therefore, at the end I did not have so much time to test it. If there is one FTP server where I can upload this document, the next student that wants to work with this program will find this information there. He will be able to continue from the step that I finish, and he will have more time to add new plugins and monitor more services.

It is also useful to share information about the lessons, exams, timetables, programs... So, I think it will be a very good idea.

Esteban Murillo Ranedo
FUTURE PROPOSALS

This thesis is only an introduction to the large collection of monitoring task that is possible to make with the program Nagios. With the continuous development of the available plugins and commands and the creation of new ones, the possibilities of monitoring of Nagios are almost infinite.

In addition, there are many options in this program that I had no time to config, such as the status maps (2D and 3D), the notifications to the contact groups, optional escalation of host and service notifications to different contact groups, the trends, the availability of external commands to be allowed to change the different parameters from the web browser...

Also I think it is very interesting the possibility of check the status of many internal characteristics of the own hosts, like processor load, disk and memory usage, running processes, log files, free disk space, the temperature, etc. But for this finality it is needed one SNMP+ agent in the final host. Then you only have to define those variable MIBs you want to monitor. If the device to monitoring has not a SNMP agent it is necessary to install what it is know like a "agent of Nagios" in the device (Linux, Windows NT/2000/XP, Solaris, HPUX). This agent is the one in charge to measure the different parameters from the device and communicate them to Nagios when it checks the device.

(1) SNMP: Short for Simple Network Management Protocol, a set of protocols for managing complex networks. The first versions of SNMP were developed in the early 80s. SNMP works by sending messages, called protocol data units (PDUs), to different parts of a network. SNMP-compliant devices, called agents, store data about themselves in Management Information Bases (MIBs) and return this data to the SNMP requesters.

(2) MIB: Short for Management Information Base, a database of objects that can be monitored by a network management system. Both SNMP and RMON use standardized MIB formats that allows any SNMP and RMON tools to monitor any device defined by a MIB.
Other possible propose is to configure Nagios to monitor all the network of the T.E.I. Once the program is forming correctly to monitor the network, all the services that we want and all the options that I said before, add new equipments is as easy as write the cfg. files with the parameters of the new hosts. Nevertheless, I think it is very important to taste the program with a small network before try with a big one, because the configure files will be very big too, and it is easier to resolve possible problems with the configuration of the program if the files are smaller.

Obviously, for apply all these proposals it is necessary to install Nagios in a host that will be continuously connected to the network that it is monitoring, and always running.
APPENDIX A

Nagios Plugin Requirements

Requirements for various plugins are listed below:

- **check_fping:**
  - Requires the fping utility distributed with SATAN. Either download and install SATAN or grab the fping program from
    http://www.stanford.edu/~schemers/docsfpingfping.html

- **check_game:**
  - Requires the qstat utility available from
    http://www.activesw.com/people/steve/qstat.html
    Last tested on qstat 2.3d BETA

- **check_hpjd:**
  - Requires the UCD-SNMP package available from
    http://ucd-snmp.ucdavis.edu
    The snmpget binary is all that is required.

- **check_ldap:**
  - Requires the LDAP libraries available from
    http://www.openldap.org
    Lib: libldap, liblber
- **check_mysql:**

  Requires the MySQL libraries available from
  http://www.mysql.org
  Lib: libmysql, libmysqlclient

- **check_pqsql:**

  Requires the PostgreSQL libraries available from
  http://www.postgresql.org

- **check_radius:**

  Requires the radius client library available from
  http://www.cityline.net/~lf/radius/
  RPM (rpmfind): radiusclient-0.3.1-1, radiusclient-devel-0.3.1-1

- **check_snmp:**

  Requires the UCD-SNMP package available from
  http://ucd-snmp.ucdavis.edu
APPENDIX B

Installation of the Windows Client Software (NSClient)

Obtain the NSClient Program

Download the NSClient program from the appropriate website. At the time of this writing, the URL for the website is http://nsclient.ready2run.nl/

Unpack the distribution file that was downloaded from the internet. Any zip utility will do so long as it successfully expands the executable and library files of the program.

Installing NSClient

Log-on to the Windows machine as Administrator or as a user that has administrator access to the system.

On the Windows machine copy pNSClient.exe in any directory on the machine you want to monitor, i.e., (c:\nsclient).

- Open a command prompt in the installation directory
- Run the following command: >pNSClient /install
- Start the service 'Netsaint NT Agent' in the services applet of the control panel. The installation will create an entry for the service in the registry and create a new key to store parameters. The created key is the following: HKEY_LOCAL_MACHINE\SOFTWARE\NSClient
- Add the following lines to the checkcommands.cfg file:
command[check nt_disk]=$USER1S/check nt -H $HOSTADDRESS$ -p 1248 -v USEDDISKSPACE -i $ARG1$ -w $ARG2$ -c $ARG3$

command[check nt_cpu_load]=$USER1$/check nt -H $HOSTADDRESS$ -p 1248 -v CPULOAD -i $ARG1$

command[check nt_uptime]=$USER1$/check nt -H $HOSTADDRESS$ -p 1248 -v UPTIME

command[check nt_client_version]=$USER1$/check nt -H $HOSTADDRESS$ -p 1248 -v CLIENTVERSION

command[check nt_process]=$USER1$/check nt -H $HOSTADDRESS$ -p 1248 -v PROCSTATE -i $ARG1$

command[check nt_service]=$USER1$/check nt -H $HOSTADDRESS$ -p 1248 -v SERVICESSTATE -i $ARG1$

command[check nt_memuse]=$USER1$/check nt -H $HOSTADDRESS$ -p 1248 -v MEMUSE -i $ARG1$ -w $ARG2$

command[check nt_paging_file]=$USER1$/check nt -H $HOSTADDRESS$ -p 1248 -v COUNTER -i "\Paging File(_Total)\% Usage","Paging File usage is %.2f %" -w $ARG1$ -c $ARG2$

**Uninstallation**

Go to the installation directory and run the following command:
> pNSClient /uninstall. All entries in the registry will be removed as well as the definition of the service.

**Configuration**

There are two parameters you can change: the port (default: 1248) and the password (default: 'None'). These two settings are store in the registry and can only be changed using 'regedit'. Open the following key and change the values if needed: HKEY_LOCAL_MACHINE\SOFTWARE\NSClient\Parms
If you change the password, you will have to use the `-s <password>` with every request you send to NSClient.

Refer to Appendix A for the complete documentation for NSClient. Included is the syntax for the plug-in's. Although NSClient was originally written for Netsaint, it has been successfully proven to work with Nagios.

APPENDIX C

Installation of the Linux Client Software (NRPE)

Obtain the NRPE Client Program

Download the NRPE client program from the appropriate website. At the time of this writing, the URL for the website is http://www.nagios.org.

Installing NRPE

Unpack the distribution file that was downloaded from the Internet. There are two commands:

```
gunzip nrpe-version number.t.
tar -xvf version numb
```

Note: If the zip version of the program was downloaded, the syntax would be the following:
Create the installation directory on the remote host for the program. The root or user account should be used when this and future steps are done to insure there are no permissions issues later during the installation. The syntax to use is:

```bash
mkdir /usr/local/nagios
```

Create the `libexec` directory under the `nagios` directory created in the previous step. The complete path to the directory should read:

```
/usr/local/nagios/libexec
```

The check_nrpe plugin should be placed on the Nagios application server along with the other plugins. The default location is the `/usr/local/nagios/libexec` directory.

The nrpe program and the configuration file (`nrpe.cfg`) should be placed in the directory that was created in the previous steps on the remote host. Additional plugins will need to be installed on the remote host in order for the add-on to function properly. The add-ons can be copied from the application server's `libexec` directory and pasted into the appropriate directory on the remote host.

Configure the nrpe client to run under the control of inetd or xinetd. Refer to Appendix H for detailed information on the proper syntax for this operation.

Configure the proper settings on the Nagios host (application server). Examples for configuring the nrpe daemon are found in the sample `nrpe.cfg` file included in the distribution. The config file resides on the remote host(s) along with the nrpe daemon. The check_nrpe plugin is installed on the Nagios host (application server). In order to use the check_nrpe plugin from within Nagios, several things in the host configuration (`hosts.cfg`) file need to be defined. An example command definition for the check_nrpe plugin would look like this:

---

Esteban Murillo Ranedo
define command{
    command_name check_nrpe
    command_line /usr/local/nagios/libexec/check_nrpe
    $HOSTADDRESS$ -
}

In any service definitions that use the nrpe plugin/daemon to get their results, you would set the service check command portion of the definition to something like this (sample service definition is simplified for this example):

define service{
    host_name someremotehost
    service_description someremoteservice
    check command check_nrpe:yourcommand
    ... etc ...
}

Where "your command" is a name of a command that you define in your nrpe.cfg file on the remote host (see the docs in the sample nrpe.cfg file for more information).

An example of the correct syntax on the application server for running the check_disk plug-in on the remote host would be the following:

define service{
    use generic-service ; name of service template to use

generic-host_name bugzilla.mgh.harvard.edu
    service_description Free Space
    is volatile 0
    check_period 24x
    max_check_attempts 3
    normal_check_interval 5

Esteban Murillo Ranedo
retry_check_interval 1
contact_groups linux-admins
notification_interval 120
notification_period 24x7
notification_options w,u,c,r
check_command check_nrpe!check_disk:

The nrpe.cfg file on the remote machine would have something like or similar to the following to allow the application server to run the above plug-in:

```
command[check_disk?]=/u r/local/nagios/libexec/check_disk 80 95 /dev/hdal
```

It is important to remember that whatever services are to be monitored from the application server, the appropriate plug-ins must be installed on the remote host(s) libexec directory. In addition, the accurate command syntax must be present in the nrpe.cfg file on the remote host as well as the services.cfg file on the application server.
APPENDIX E

Code of the .cfg files configured.

Host.cfg

# Read the documentat: more information on this configuration file. I have provided some, but things may not be so clear without explanation... to read the HTML documentation!

Last Modified: 03-10-2002

### SYNTAX:

Generic host definition template

define host:

- referenced in other host definitions, used for template recursion/resolution
  notifications_enabled 1 ; Host notifications are enabled
event_handler_enabled 1 ; Host event handler is enabled
flop_detection_enabled 1 ; Flap detection is enabled
process_perf_data 1 ; Process performance data
retain_status_information 1 ; Retain status information across program restarts
retain_nonstatus_information 1 ; Retain non-status information across program restarts
register 0 ; DONT REGISTER THIS DEFINITION - NOT A REAL HOST, JUST A TEMPLATE!

# LABORATORIO KYRANASTASIS

# power definition

address 195.130.93.182
check_command check-host-alive
max_check_attempts

Esteban Murillo Ranedo
generic-host

address

check_command
max_check_attempts
notification_interval
notification_period
notification_options

parents
check_command
max_check_attempts
notification_interval
notification_period
notification_options

computer #4
10.1.9.140

Esteban Murillo Ranedo
check_command: check-host_alive
max_check_attempts: 10
notification_interval: 120
notification_period: 24x7
notification_options: d,u,r

#define 'peii06' host definition
define host{
    use: generic-host
    use: N
    host_name: peii06
    alias: computer #6
    address: 10.1.9.144
    parents: power
    check_command: check-host_alive
    max_check_attempts: 10
    notification_interval: 120
    notification_period: 24x7
    notification_options: d,u,r
}

#define 'peii07' host definition
define host{
    use: generic-host
    use: N
    host_name: peii07
    alias: computer #7
    address: 10.1.9.145
    parents: power
    check_command: check-host_alive
    max_check_attempts: 10
    notification_interval: 120
    notification_period: 24x7
    notification_options: d,u,r
}

#define 'peii08' host definition
define host{
    use: generic-host
    use: N
    host_name: peii08
    alias: computer #8
    address: 10.1.9.146
    parents: power
    check_command: check-host_alive
    max_check_attempts: 10
    notification_interval: 120
    notification_period: 24x7
    notification_options: d,u,r
}

#define 'peii09' host definition
define host{
    use: generic-host
    use: N
    host_name: peii09
    alias: computer #9
    address: 10.1.9.147
    parents: power
    check_command: check-host_alive
check_command = check-host-alive
max_check_attempts = 10
notification_interval = 120
notification_period = 24x7
notification_options = d,u,r
}

# 'peii11' host definition
define host{
  use = generic-host ; N
  host_name = peii11
  alias = computer #11
  address = 10.1.9.149
  parents = power
  check_command = check-host-alive
  max_check_attempts = 10
  notification_interval = 120
  notification_period = 24x7
  notification_options = d,u,r
}

# 'peii12' host definition
define host{
  use = generic-host ; N
  host_name = peii12
  alias = computer #12
  address = 10.1.9.152
  parents = power
  check_command = check-host-alive
  max_check_attempts = 10
  notification_interval = 120
  notification_period = 24x7
  notification_options = d,u,r
}

# 'peii13' host definition
define host{
  use = generic-host ; N
  host_name = peii13
  alias = computer #13
  address = 10.1.9.138
  parents = power
  check_command = check-host-alive
  max_check_attempts = 10
  notification_interval = 120
  notification_period = 24x7
  notification_options = d,u,r
}

Esteban Murillo Ranedo
# 'peii6' host definition

---

parents
check_command
max_check_attempts
notification_interval
notification_period
notification_options

---

**Name of host template to use**

10.1.9.134
power
check-host-alive
10
120
24x7
d, w, r

# 'peii7' host definition

---

define host{
    use
generic-host

    ; Name of host template to use
}
Esteban Murillo Ranedo
define host {
  use generic-host ; Name of host template to use
  parents router
  check_command check-host-alive
  max_check_attempts 10
  notification_interval 120
  notification_period 24x7
  notification_options d,u,r
}

host definition

# 'iiwb07' host definition

define host:
    use generic-host

    host_name       iibw07
    alias           host #7
    address         10.1.9.47
    parents         router
    check_command   check-
    max_check_attempts 10
    notification_interval 120
    notification_period 24x7
    notification_options d,u,r

# 'iiwb08' host definition
define host:
    use generic-host

    host_name       iibw08
    alias           host #8
    address         10.1.9.48
    parents         router
    check_command   check-
    max_check_attempts 10
    notification_interval 120
    notification_period 24x7
    notification_options d,u,r

# 'iiwb09' host definition
define host:
    use generic-host

    host_name       iibw09
    alias           host #9
    address         10.1.9.49
    parents         router
    check_command   check-host-alive
    max_check_attempts 10
    notification_interval 120
    notification_period 24x7
define host ipb

parents 10.1.9.53
check_command router
max_check_attempts 10
notification_interval 120
notification_period 24x7
notification_options d,u,r

#define 'ipb' host definition
#define host ipb
# 'iiwbl6' host definition
alias
address
parents
check_command
max_check_attempts
notification_interval
notification_period
notification_options

# 'iiwbl7' host definition
parents
check_command
max_check_attempts
notification_interval
notification_period
notification_options

# 'iiwbl8' host definition
define host:
use
generic-host

host_name
alias
address
<table>
<thead>
<tr>
<th>Interface</th>
<th>IP Address</th>
<th>Netmask</th>
<th>Gateway</th>
<th>MTU</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth0</td>
<td>192.168.1.100</td>
<td>255.255.255.0</td>
<td>192.168.1.1</td>
<td>1500</td>
<td>1000 M</td>
</tr>
<tr>
<td>eth1</td>
<td>192.168.2.100</td>
<td>255.255.255.0</td>
<td>192.168.2.1</td>
<td>1500</td>
<td>1000 M</td>
</tr>
</tbody>
</table>

**Notes:**
- eth0 is the default gateway.
- eth1 is for secondary network connection.
- MTU is set to 1500 for optimal performance.
- Speed is configured for 1000 M for maximum bandwidth.
parents
  check_command  check_host_alive
max_check_attempts
notification_interval
notification_period
notification_options
}

parents
  check_command  check_host_alive
max_check_attempts
notification_interval
notification_period  24x7
notification_options
}

host definition

n004: host definition

use generic-host ; Name of host template to use

host_name: sun005
alias: unix #5
address 195.130.92.98
parents
check_command: check-host-alive
max_check_attempts 10
notification_interval 120
notification_period 24x7
notification_options d,u,r
}

# 'sun006' host definition
define host:
  use generic-host ; Name of host template to use

host_name: sun006
alias: unix #6
address 195.130.92.99
parents
check_command: check-host-alive
max_check_attempts 10
notification_interval 120
notification_period 24x7
notification_options d,u,r
}
alias       unix #7
address     195.130.92.100
parents     server
check_command check-host-alive
max_check_attempts 10
notification_interval 120
notification_period 24x7
notification_options d,s,r
}

# 'sun008' host definition
define host{
  use generic-host ; N

  address 195.130.92.103
  address 195.130.92.104

parents    server
check_command    check-host-alive
max_check_attempts    10
notification_interval    120
notification_period    24x7
notification_options    d,u,r

# 'sun012' host definition
define host{
  use    generic-host    : N
  host_name    sun012
  alias    unix #12
  address    195.130.92.105
  parents    server
  check_command    check-host-alive
  max_check_attempts    10
  notification_interval    120
  notification_period    24x7
  notification_options    d,u,r
}

# 'sun013' host definition
define host{
  use    generic-host    : N
  host_name    sun013
  alias    unix #13
  address    195.130.92.106
  parents    server
  check_command    check-host-alive
  max_check_attempts    10
  notification_interval    120
  notification_period    24x7
  notification_options    d,u,r
}

# 'sun014' host definition
define host{
  use    generic-host    : Name of host template to use
  host_name    sun014
  alias    unix #14
  address    195.130.92.107
  parents    server
  check_command    check-host-alive
  max_check_attempts    10
  notification_interval    120
  notification_period    24x7
  notification_options    d,u,r
}

# 'sun015' host definition
define host{
  use    generic-host    : Name of host template to use
  host_name    sun015
  alias    unix #15
  address    195.130.92.108
  parents    server
  check_command    check-host-alive
  max_check_attempts    10
  notification_interval    120
  notification_period    24x7
  notification_options    d,u,r
}
195.130.92.109
parents
check_command
max_check_attempts
notification_interval
notification_period
notification_options
}

#define 'sun017' host definition
define host{
use

generic-host

#define 'sun019' host definition
define host{
use

generic-host

#define 'sun020' host definition
define host{
use

generic-host

Esteban Murillo Ranedo

86
check_command
max_check_attempts
notification_interval
notification_period
notification_options
}

check_host_alive

generic-host ; Name of host template

define host()
use

generic-host ; Name of host template to use

define host()

define host()

define host()
Services.cfg

# Generic service definition template
define service{
    name generic-service ; The 'name' of this service
    template, referenced in other service definitions
        active_checks_enabled 1 ; Active service checks are enabled
        passive_checks_enabled 1 ; Passive service checks are enabled
    enabled/accepted
        parallelize_check 1 ; Active service checks should be parallelized (disabling this can lead to major performance problems)
        obsess_over_service 1 ; We should obsess over this service (if necessary)
    check freshness
        notifications_enabled 1 ; Service notifications are enabled
        event_handler_enabled 1 ; Service event handler is enabled
        flap_detection_enabled 1 ; Flap detection is enabled
        process_perf_data 1 ; Process performance data
        retain_status_information 1 ; Retain status information across program restarts
        retain_nonstatus_information 1 ; Retain non-status information across program restarts
    register
    REAL SERVICE, JUST A TEMPLATE!
}

# Service definition
define service{
    use generic-service ; Name of service template to use

    host_name *
    service_description PING
    is_volatile 0
    check_period 24x7
    max_check_attempts 3
    normal_check_interval 5
    retry_check_interval 1
    contact_groups admins
    notification_interval 240
    notification_period 24x7
    notification_options c,r

Esteban Murillo Ranedo

88
check_command  check_ping!100.0,20%!500.0,60%

### FTP

```
# Service definition
define service{
    use template to use
    
    host_name
    service_description
    is_volatile
    check_period
    max_check_attempts
    normal_check_interval
    retry_check_interval
    contact_groups
    notification_interval
    notification_period
    notification_options
    check_command
}
```

### HTTP

```
define service{
    use template to use
    
    host_name
    service_description
    is_volatile
    check_period
    max_check_attempts
    normal_check_interval
    retry_check_interval
    contact_groups
    notification_interval
    notification_period
    notification_options
    check_command
    }
```
use generic-service

host_name
service_description
is_volatile
check_period
max_check_attempts
normal_check_interval
retry_check_interval
contact_groups
notification_interval
notification_period
notification_options
check_command

---

Hostgroups.cfg

---

= documentation for more information c

# Last Modified: 03-07-2002

# UOP DEFINITIONS

# SYNTAX:

# 'Servers' host group definition
define hostgroup{
  hostgroup_name servers
  contact_groups admins
  members power, router, server

# I form a group with all the servers because check the same services in all of them

# 'Windows - server' host group definition
#define hostgroup:
#  hostgroup_name windows-server
#  alias Windows Server
#  contact_groups admins
#  members router

Esteban Munlo Ranoedo
ix - server' host group definition

net hostgroup{
  hostgroup_name    unix-server
  alias             Windows Server
  contact_groups    admins
  members           server
}

'computers' host group definition
fine hostgroup{
  hostgroup_name    computers
  alias             Kyranastasis Computer
  contact_groups    admins
  members           peii01, peii02, peii03, peii04, peii05, peii06, peii07, peii08, peii09, peii10, peii11, peii12, peii13, peii14, peii16, peii17, peii18, peii19, pe-xx, pej14
}

'hosts' host group definition
fine hostgroup{
  hostgroup_name    hosts
  alias             Windows Hosts
  contact_groups    admins
}

ix' host group definition
fine hostgroup{
  hostgroup_name    unix
  alias             UNIX Hosts
  contact_groups    admins

  sun001, sun002, sun003, sun004, sun005, sun006, sun007, sun008, sun009, sun010, sun011, sun012, sun013, sun014, sun015, sun016, sun017, sun018, sun019, sun020, sun021, hp4050, iiwm

Contactgroups.cfg

# the documentation for this configuration file.

# Last Modified: 10-03-2002

Esteban Murillo Ranedo
Nagios.cfg

# Nagios.cfg - Sample Main Config File for Nagios
#
# Read the documentation for more information on this configuration
# file. I have provided some comments here, but things may not be so
# clear without further explanation.
#
# Last Modified: 07-04-2002
#

# LOG FILE
# This is the main log file where service and host events are logged
# for historical purposes. This should be the first option specified
# in the config file!!!
log_file=/usr/local/nagios/var/nagios.log

# OBJECT CONFIGURATION FILE(s)
# This is the configuration file in which you define hosts, host
# groups, contacts, contact groups, services, etc. I guess it would
# be better called an object definition file, but for historical
# reasons it isn't. You can split object definitions into several
# different config files by using multiple cfg_file statements here.
# Nagios will read and process all the config files you define.
# This can be very useful if you want to keep command definitions
# separate from host and contact definitions...

# Plugin commands (service and host check commands)
# Arguments are likely to change between different releases of the
# plugins, so you should use the same config file provided with the
# plugin release rather than the one provided with Nagios.
cfg_file=/usr/local/nagios/etc/checkcommands.cfg

# Misc commands (notification and event handler commands, etc)
cfg_file=/usr/local/nagios/etc/misccommands.cfg

# You can split other types of object definitions across several
# config files if you wish (as done here), or keep them all in a
# single config file.

cfg_file=/usr/local/nagios/etc/contactgroups.cfg

cfg_file=/usr/local/nagios/etc/contacts.cfg

cfg_file=/usr/local/nagios/etc/dependencies.cfg

cfg_file=/usr/local/nagios/etc/escalations.cfg

cfg_file=/usr/local/nagios/etc/hostgroups.cfg

cfg_file=/usr/local/nagios/etc/hosts.cfg

cfg_file=/usr/local/nagios/etc/services.cfg

cfg_file=/usr/local/nagios/etc/timeperiods.cfg

# RESOURCE FILE
# This is an optional resource file that contains $USER$ macro
# definitions. Multiple resource files can be specified by using
# multiple resource_file definitions. The CGIs will not attempt to
# read the contents of resource files, so information that is
# considered to be sensitive (usernames, passwords, etc) can be
# defined as macros in this file and restrictive permissions (600)
# can be placed on this file.

resource_file=/usr/local/nagios/etc/resource.cfg

# STATUS FILE
# This is where the current status of all monitored services and
# hosts is stored. Its contents are read and processed by the CGIs.
# The contents of the status file are deleted every time Nagios
# restarts.

status_file=/usr/local/nagios/var/status.log

# NAGIOS USER
# This determines the effective user that Nagios should run as.
# You can either supply a username or a UID.

nagios_user=nagios

# NAGIOS GROUP
# This determines the effective group that Nagios should run as.
# You can either supply a group name or a GID.

nagios_group=nagios

# EXTERNAL COMMAND OPTION
# This option allows you to specify whether or not Nagios should check
# for external commands (in the command file defined below). By default
# Nagios will *not* check for external commands, just to be on the
# cautious side. If you want to be able to use the CGI command interface
# you will have to enable this. Setting this value to 0 disables command
# checking (the default), other values enable it.

cHECK_EXTERNAL_COMMANDS=1

# EXTERNAL COMMAND CHECK INTERVAL
# This is the interval at which Nagios should check for external commands.
# This value works of the interval_length you specify later. If you leave
# that at its default value of 60 (seconds), a value of 1 here will cause
This is the file that Nagios will write commands that
will execute to check the file as the file is deleted every
night (midnight every day)

This is the lock file that Nagios uses to lock the file, as the file is deleted every night (midnight on Saturday every week)

This is the comment file, etc. This file

The comment file, etc. This file
- Monthly rotation (midnight last day of month)

# LOG ARCHIVE PATH
# This is the directory where archived (rotated) log files should be placed (assuming you've chosen to do log rotation).

log_archive_path=/usr/local/nagios/var/archives

# LOGGING OPTIONS
# If you want messages logged to the syslog facility, as well as the NetAlarm log file set this option to 1. If not, set it to 0.

# NOTIFICATION LOGGING OPTION
# If you don't want notifications to be logged, set this value to 0. If notifications should be logged, set the value to 1.

log_notifications=1

# SERVICE RETRY LOGGING OPTION
# If you don't want service check retries to be logged, set this value to 0. If retries should be logged, set the value to 1.

log_service_retries=1

# HOST RETRY LOGGING OPTION
# If you don't want host check retries to be logged, set this value to 0. If retries should be logged, set the value to 1.

# EVENT HANDLER LOGGING OPTION
# If you don't want host and service event handlers to be logged, set this value to 0. If event handlers should be logged, set the value to 1.

# INITIAL STATES LOGGING OPTION
# If you want Nagios to log all initial host and service states the main log file (the first time the service or host is checked) you can enable this option by setting this value to 1. If you are not using an external application that does long term statistics reporting, you do not need to enable this option. In this case, set the value to 0.

log_initial_states=0

# EXTERNAL COMMANDS LOGGING OPTION

Esteban Munilo Ranedo
I'd use any delay between external commands.

external commands should

external commands=

command definition that

-- nacular=somecommand

-- nacular=somecommand

I would expect the initial

result of a child process to be printed

Continue in the file

It's ok to use delay between

-- > line comment at end of

command

End
# This option allows you to specify the maximum number of
# service checks that can be run in parallel at any given time.
# Specifying a value of 1 for this variable essentially prevents
# any service checks from being parallelized. A value of 0
# will not restrict the number of concurrent checks that are
# being executed.

max_concurrent_checks=0

# SERVICE CHECK REAPER FREQUENCY
# This is the frequency (in seconds!) that Nagios
# will check the results of services that have been checked.

service_reaper_frequency=10

# SLEEP TIME
# This is the number of seconds to sleep between checking for system
# events and service checks that need to be run. I would recommend
# not changing this from its default value of 1 second.

# TIMEOUT VALUES
# These options control how much time Nagios will allow various
# types of commands to execute before killing them off. Options
# are available for controlling maximum time allotted for
# service checks, host checks, event handlers, notifications, the
# ocsp command, and performance data commands. All values are in
# seconds.

service_check_timeout=60
host_check_timeout=30
event_handler_timeout=30
notification_timeout=30
ocsp_timeout=5
perfdata_timeout=5

# RETAIN STATE INFORMATION
# This setting determines whether or not Nagios will save state
# information for services and hosts before it shuts down. Upon
# startup Nagios will reload all saved service and host state
# information before starting to monitor. This is useful for
# maintaining long-term data: state statistics, etc, but will
# also lead to a one-time penalty, loss of additional state,
# and a startup delay.

retain_state_information=1

# STATE RETENTION FILE
# This is the file that Nagios should use to store host and
# service state information before it shuts down. The state
# information in this file is also read immediately prior to
# starting to monitor the network when Nagios is restarted.
# The file should only be used if the preserve_state_information
# option is enabled.

state_retention_file=/usr/local/nagios/var/status.i8192
```
# service check execution option
# This determines whether or not Nagios will actively execute
# service checks when it initially starts. If this option is
# disabled, checks are not actively made, but Nagios can still
# receive and process passive check results that come in. Unless
# you're implementing redundant hosts or have a special need for
# disabling the execution of service checks, leave this enabled!
# Values: 1 = enable checks, 0 = disable checks
execute_service_checks=1

# passive check acceptance option
# This determines whether or not Nagios will accept passive
# service checks results when it initially (re)starts.
# Values: 1 = accept passive checks, 0 = reject passive checks
accept_passive_service_checks=1
```
her or not Nagios will s
; when it is initially (
**ORPHANED SERVICE CHECK OPTION**

This determines whether or not Nagios will periodically check for orphaned services. Since service checks are not rescheduled until the results of their previous execution instance are processed, there exists a possibility that service checks may never get rescheduled. This seems to be a rare problem and should not happen under normal circumstances.

If you have problems with service checks never getting rescheduled, you might want to try enabling this option.

Values: 1 = enable checks, 0 = disable checks

```
check_for_orphaned
```

**SERVICE FRESHNESS CHECK OPTION**

This option determines whether or not Nagios will periodically check the "freshness" of service results. Enabling this option is useful for ensuring passive checks are received in a timely manner.

Values: 1 = enabled freshness checking, 0 = disable freshness checking

```
check_service_freshness=1
```

**FRESHNESS CHECK INTERVAL**

This setting determines how often (in seconds) Nagios will check the "freshness" of service check results. If you have disabled service freshness checking, this option has no effect.

```
freshness_check_interval=60
```

**AGGREGATED STATUS UPDATES**

This option determines whether or not Nagios will aggregate updates of host, service, and program status data. Normally, status data is updated immediately when a change occurs. This can result in high CPU loads if you are monitoring a lot of services. If you want Nagios to only refresh status data every few seconds, disable this option.

Values: 1 = enable aggregate updates, 0 = disable aggregate updates

```
aggregate_status_updates=1
```

**AGGREGATED STATUS UPDATE INTERVAL**

Combined with the aggregate_status_updates option, this option determines the frequency (in seconds!) that Nagios will periodically dump program, host, and service status data. If you are not using aggregated status data updates, this option has no effect.

```
status_update_interval=15
```

**FLAP DETECTION OPTION**

This option determines whether or not Nagios will try and detect hosts and services that are "flapping". Flapping occurs when a host or service changes between states too frequently. When Nagios detects that a...
# Host or service is flapping, it will temporarily suppress notifications for that host/service until it stops flapping. Flap detection is very experimental, so read the HTML documentation before enabling this feature! Values: 1 = enable flap detection 0 = disable flap detection (default)

enable_flap_detection=0

## Flap Detection Thresholds for Hosts and Services
Read the HTML documentation on flap detection for an explanation of what this option does. This option has no effect if flap detection is disabled.

low_service_flap_threshold=5.0
high_service_flap_threshold=20.0
low_host_flap_threshold=5.0
high_host_flap_threshold=20.0

## Date Format Option
This option determines how short dates displayed. Valid options include:

- us (MM-DD-YYYY HH:MM:SS)
- euro (DD-MM-YYYY HH:MM:SS)
- iso8601 (YYYY-MM-DD HH:MM:SS)
- strict-iso8601 (YYYY-MM-DDTHH:MM:SS)

## Illegal Object Name Characters
This options allows you to specify illegal characters that cannot be used in host names, service descriptions, or names of other object types.

illegal_object_name_chars="\$%^&*()_=-";

## Illegal Macro Output Characters
This options allows you to specify illegal characters that are stripped from macros before being used in notifications, event handlers, etc. This DOES NOT affect macros used in service or host check commands.

The following macros are stripped of the characters you specify:

$OUTPUT$, $PERFDATA$

illegal_macro_output_chars="-!$%*()_=-";

## Administrator Email Address
The email address of the administrator doing the monitoring. Nagios can access this value by using the $ADMINEMAIL$ macro in notification commands.

admin_email=nagios

Esteban Murillo Ranedo
# Nagios never uses this value itself, but value by using the $ADMINPAGER$ macro in commands.

admin_pager=pagenagios

# EOF (End of file)

---

**Timeperiods.cfg**

---

```conf
# Read the documentatic configuration file.

# Last Modified: 03-07-2002

# SYNTAX:

'24x7' timeperiod definition

```time
    define timeperiod!
    timeperiod_name 24x7
    24 Hours A Day,
    sunday 00:00-24:00
    monday 00:00-24:00
    tuesday 00:00-24:00
    wednesday 00:00-24:00
    thursday 00:00-24:00
    #

'workhours' timeperiod definition

```time
    define timeperiod!
    timeperiod_name workhours
    "Normal" Working Hours
    monday 09:00-17:00
    tuesday 09:00-17:00
    wednesday 09:00-17:00
    thursday 09:00-17:00
    #

'nonworkhours' timeperiod definition

```time
    define timeperiod!
    timeperiod_name nonworkhours
    alias Non-Work Hours
    sunday 00:00-24:00
    monady 00:00-05:00,17:00-24
```

---

Esteban Murillo Ranedo
tuesday: 00:00-17:00, 17:00-24:00
wednesday: 00:00-09:00, 17:00-24:00
thursday: 00:00-09:00, 17:00-24:00
friday: 00:00-09:00, 17:00-24:00
saturday: 00:00-24:00

' timeperiod definition
define timeperiod:
  timeperiod name none
  alias No Time Is A Good Time
 IMAGES REFERENCES

Fig.1 Summary of some monitoring programs and services (Juan Antonio Valino Garcia, 2004)
"Arquitectura Base para un Sistema de Monitorizacion de Servidores dentro del Marco del
Open Source (Version 0.2)"
Cuadro 5.22: Resumen del tipo de monitorizacion y los parametros monitorizables
Downloaded on June 2005.

Fig.2 What monitoring software do you use? (Poll opened January 24th, 2005)
A total of 907 votes have been cast in this poll so far.
URL: http://dev.mysql.com/tech-resources/quickpolls/monitoring-software.html
Downloaded on June 2005

Fig.4: Top Organization Types Using Nagios (Nagios, Monday December 6th, 2004)
URL: http://www.nagios.org/userprofiles/quickstats.php
Downloaded on June 2005.

Fig.5: Network parameters. (Esteban Murillo, June 2005)
Linux Mandrake 10.1 Manage connections print screen

Fig.7: Nagios authentication (Esteban Murillo, June 2005)
Linux Mandrake 10.0 printscreen

Fig.8: Network Structure. (Esteban Murillo, June 2005)
Network of the Industrial Informatics Department of T.E.I Kavala.

Fig.9: Nagios system
Evaluacion de Nagios para Linux, URL: http://manoloddvirtualave.net
Downloaded on June 2005

Fig.10 Nagios script (Esteban Murillo, June 2005)
Linux Mandrake 10.0 console printscreen

Fig.11 Nagios welcome page (Esteban Murillo, June 14th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen

Esteban Murillo Ranedo
Fig. 12: Error: Nagios is not running (Esteban Murillo, June 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Fig. 13: Route command (Esteban Murillo, June 2005)
Linux Mandrake 10.0 console printscreen.

Fig. 15: Nagios Service Overview (Esteban Murillo, June 16th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Fig. 16: Route add command (Esteban, June 2005)
Linux Mandrake 10.0 console printscreen.

Fig. 17: Nagios Service Overview (Esteban Murillo, June 16th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Fig. 19: Service Status Details For Host Group 'unix' (Esteban Murillo, June 15th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Fig. 20: Service Status Details For The Host Group 'servers' (Esteban Murillo, June 15th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Fig. 21: Service HTTP on host unix mailhost (liwm), (Esteban Murillo, June 15th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Fig. 22: Service HTTP on host Windows Lab Server (router), (Esteban Murillo, June 15th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Fig. 23: Service Status Details For The Host Group 'servers' (Esteban Murillo, June 16th, 2005)
Linux Mandrake 10.0 web browser (Konkeror) printscreen.

Esteban Murillo Ranado
TABLE REFERENCES

Fig.3  Overview Status (Nagios, Monday December 6th, 2004)
URL: http://www.nagios.org/userprofiles/quickstats.php
Downloaded on June 2005.

Fig. 6: Nagios Directory Structure. Ethan Galstad (main developer of the Nagios)
"Nagios version 1.0 Documentation"

Fig 14: Network parameters (Esteban Murillo, June 2005)

Fig. 18: Services monitored (Esteban Murillo, June 2005)
BIBLIOGRAPHY

About Linux:

http://linuxmx.coolfreepage.com/

http://es.wikipedia.org/wiki/Portada

http://usuarios.lycos.es/visualdiego/Programas.php

http://perfparse.sourceforge.net/

About Nagios:

www.nagios.org

http://nagios.uclv.edu.cu/ (Nagios running in the network of a University)

http://wwwunixmexico.org/modules.php?name=Topics

www.nagiosexchange.org

About Apache:

www.apache.org