

Building an enterprise monitoring system using **Linux** and open source tools



Our Objective:

A system that will monitor the state of servers , services and devices in our local or collocated network environment

Our Tools:

Nagios : will monitor hosts services and resources

Syslog: will collect log data from remote hosts

Swatch: will monitor the logs

NAGIOS

- Monitor Hosts , services and resources
 - Notify contacts on any problems
 - And More (which we will not cover)

The screenshot displays the Nagios web interface in a browser window. The browser's address bar shows the URL `https://192.17.9.01/nagios/`. The interface is divided into several sections:

- General:** Includes links for Home and Documentation.
- Monitoring:** Contains links for Tactical Overview, Service Detail, Host Detail, Status Overview, Status Summary, Status List, Status Map, and ID Status Map.
- Reporting:** Includes links for Trends, Availability, Alert Histogram, Alert History, Alert Summary, Notifications, and Event Log.
- Configuration:** A section for system configuration.

The main content area shows the following monitoring data:

- Network Outages:** 0 Outages
- Network Health:** Host Health and Service Health (both shown as red bars).
- Hosts:** 7 Down, 0 Unreachable, 10 Up, 0 Pending. A red box indicates "1 Disabled by 1 user".
- Services:** 21 Critical. A red box indicates "1 Disabled by 1 user".
- Monitoring Problems:** A table with five columns: High Distributions, Notifications, Event Handlers, Active Checks, and Passive Checks. Each column has a status indicator (e.g., "OK" or "Warning") and a brief description of the current state.

Nagios Components

Nagios Daemon :

- schedules & performs hosts and service check
 - notify contacts of events
 - execute event handlers
-

Nagios CGI's :

- displays hosts and services status
 - enable/disable notifications
 - schedule downtime and add comments to hosts
 - annoy everyone around you using alerts
-

Nagios Plugins :

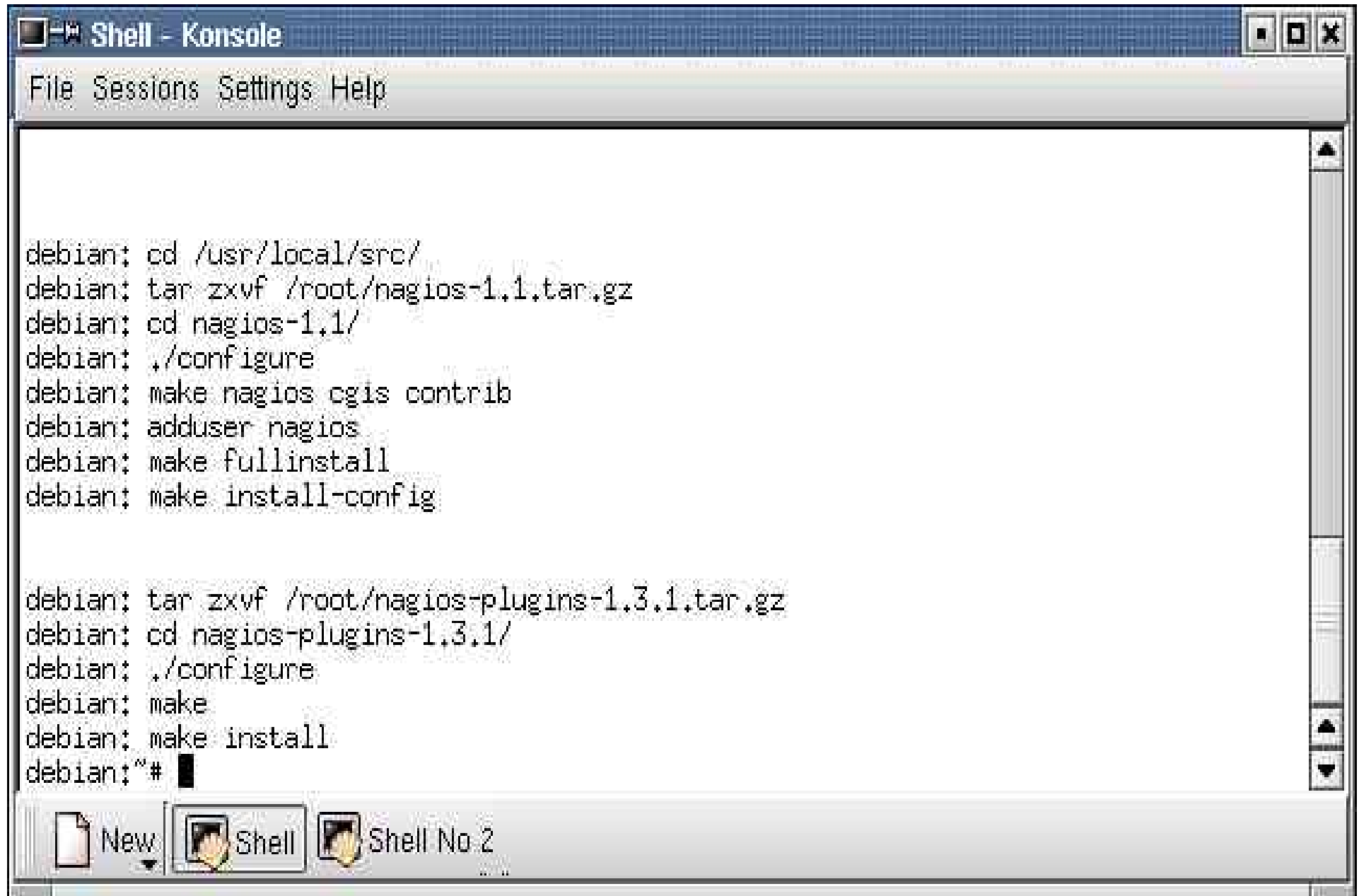
- perform service checks on remote hosts
-

Nagios **Remote** Plugins :

- perform **local** resource checks on remote hosts
-

Installation

- Get source or rpm from www.nagios.org



```
Shell - Konsole
File Sessions Settings Help

debian: cd /usr/local/src/
debian: tar zxvf /root/nagios-1.1.tar.gz
debian: cd nagios-1.1/
debian: ./configure
debian: make nagios cgis contrib
debian: adduser nagios
debian: make fullinstall
debian: make install-config

debian: tar zxvf /root/nagios-plugins-1.3.1.tar.gz
debian: cd nagios-plugins-1.3.1/
debian: ./configure
debian: make
debian: make install
debian: ~#
```

The screenshot shows a terminal window with a menu bar (File, Sessions, Settings, Help) and a taskbar at the bottom with buttons for 'New', 'Shell', and 'Shell No 2'. The terminal output shows the execution of various commands to install Nagios and its plugins on a Debian system.

Nagios Configuration overview:

/usr/local/nagios/

bin/ - contains nagios core program

etc/ - contains config files

sbin/ - contains nagios CGIs

share/ - contains web interface html and online documentation

var/ - contains log files

libexec/ - contains nagios plugins

Config Files

nagios.cfg – main config file ,affects how nagios operates

cgi.cfg – contain directives used by the web interface

resource files – used to store custom configuration information not available to the cgi's (macros, db settings etc)

object files – these contain host,service,contact and time period information

extended information files - are used to define additional information for hosts and service (icons , description etc)

Note: a working apache server is needed for the web interface

Nagios.cfg

File contains A LOT of directives but the sample file is well documented and will work “ out of the box”

A few notes:

- **_file_settings** = these directive are used to point nagios to the correct config file locations
`cfg_file=/usr/local/nagios/etc/hosts.cfg`
 - **check_external_commands=0** – set this to 1 to enable web interface commands
 - **command_file=/usr/local/nagios/var/rw/nagios.cmd** – this files must be writeble by the web server user (apache on redhat , www-data on woody)
 - **interval_length=60** - the seconds per unit interval as used in the host/contact/service configuration files
 - **enable_flap_detection=0** – flap detection will supress notifications
when a host or service changes between states too frequently
-

cgi.cfg

- **url_html_path=/nagios** - name of the apache virtual directory
 - **use_authentication=1** - tells nagios to use apache authentication this will determine who can access the web interface
 - **Access control :**
 - authorized_for_system_information=nagiosadmin,theboss,jdoe
 - authorized_for_configuration_information=nagiosadmin,jdoe
 - authorized_for_system_commands=nagiosadmin**
 - authorized_for_all_services=nagiosadmin,guest
 - authorized_for_all_hosts=nagiosadmin,guest
 - authorized_for_all_service_commands=nagiosadmin
 - authorized_for_all_host_commands=nagiosadmin
-

Apache config

/nagios dir alias :

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

/nagios/cgi-bin script alias:

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

.htaccess:

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

create user using htpasswd -c /usr/local/nagios/etc/htpasswd.users username

Define a Time period

Time periods are used to create different schedules for notifications and host/service checks

Filename: timeperiods.cfg

```
# 'workhours' timeperiod definition  
define timeperiod{  
    timeperiod_name    workhours  
    alias              "Normal" Working Hours  
    monday            10:00-16:00  
    tuesday           10:00-16:00  
    wednesday         10:00-16:00  
    thursday          10:00-16:00  
    friday            09:00-12:00  
}
```

Define a contact and contact group

A contact definition is used to identify someone who should be contacted in the event of a problem on your network

Filename: contacts.cfg

```
define contact{
    contact_name          nagios
    alias                 Nagios Admin
    service_notification_period 24x7
    host_notification_period  24x7
    service_notification_options w,u,c,r
    host_notification_options  d,u,r
    service_notification_commands notify-by-email,notify-by-epager
    host_notification_commands  host-notify-by-email,host-notify-by-epager
    email                 nagios-admin@localhost.localdomain
    pager                 pagenagios-admin@localhost.localdomain
}
```

A contact group definition is used to group one or more contacts together for the purpose of sending out alert/recovery notifications

Filename:contactgroups.cfg

```
define contactgroup{
    contactgroup_name linux-admins
    alias             Linux Administrators
    members          nagios
}
```

Define a host & hostgroup

A host definition is used to define a physical server, workstation, device, etc. that resides on your network.

Filename: hosts.cfg

```
define host{
    use          generic-host
    host_name    linux1
    alias        Linux Server #1
    address      192.169.1.6
    check_command check-host-alive
    max_check_attempts 10
    notification_interval 0 ;
    notification_period 24x7
    notification_options d,u,r ;
}
```

A host group definition is used to group one or more hosts together for the purposes of simplifying notification

Filename: hostgroups.cfg

```
define hostgroup{
    hostgroup_name linux-boxes
    alias          Linux Servers
    contact_groups linux-admins
    members        linux1,linux2
}
```

Define services

A service definition is used to identify a "service" that runs on a host

Filename: services.cfg

```
define service{  
  host_name          linux1  
  service_description PING  
  check_period       24x7  
  max_check_attempts 3  
  normal_check_interval 5  
  retry_check_interval 1  
  contact_groups     linux-admins  
  notification_interval 0  
  notification_period 24x7  
  notification_options w,u,c,r  
  check_command      check_ping!100.0,20%!500.0,60%  
}
```

And another one

Service definition

```
define service{  
    host_name           linux1  
    service_description HTTP  
    check_period       24x7  
    max_check_attempts   3  
    normal_check_interval 2  
    retry_check_interval 1  
    contact_groups      linux-admins  
    notification_interval 0  
    notification_period  24x7  
    notification_options w,u,c,r  
    check_command       check_http!-u index.html  
}
```

Where did those check_commands come from ?

Check commands are defined in the checkcommands.cfg file
they make use of nagios plugins and the resources files

'check_ping' command definition

```
define command{  
    command_name check_ping  
    command_line $USER1$/check_ping -H $HOSTADDRESS$  
                                     -w $ARG1$ -c $ARG2$ -p 5  
}
```

'check_http' command definition

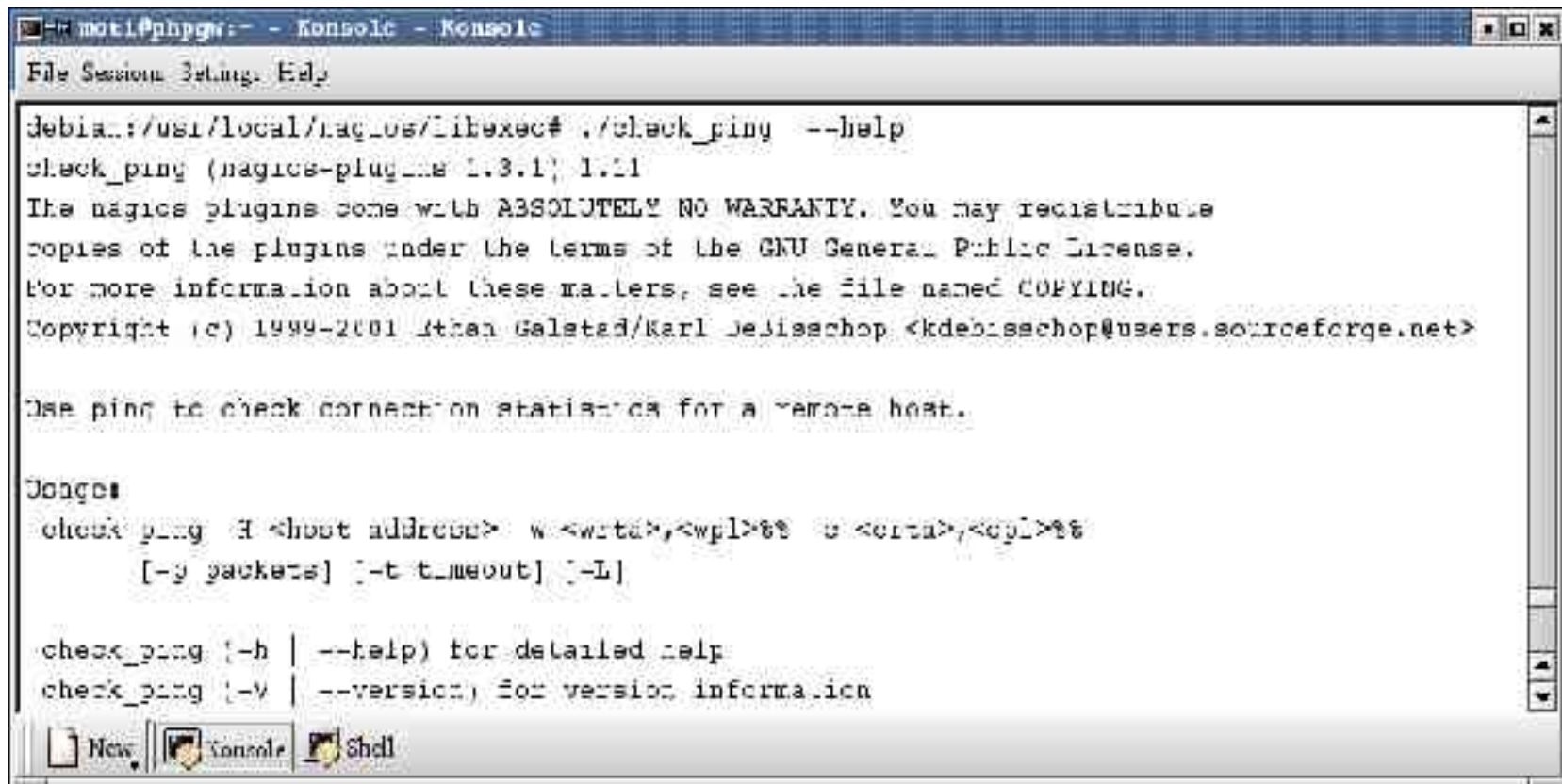
```
define command{  
    command_name check_http  
    command_line $USER1$/check_http -H $HOSTADDRESS$  
}
```

Defining a check command

We need to know two things : **macro names** and **plugin syntax**

Macros are listed in the html documentantation that comes with nagios

Plugin syntax is printed when executing a plugin with the
-- -help switch



```
moti@phpgw:~$ ./check_ping --help
check_ping (nagios-plugin 1.3.1) 1.11
The nagios plugins come with ABSOLUTELY NO WARRANTY. You may redistribute
copies of the plugins under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
Copyright (c) 1999-2001 Ethan Galstad/Karl Deisschop <kdeisschop@users.sourceforge.net>

Use ping to check connection statistics for a remote host.

Usage:
check_ping H <host address> w <write>,<wpl>%% s <size>,<sp>%%
        [-y packets] [-t timeout] [-L]

check_ping [-h | --help) for detailed help
check_ping [-v | --version) for version information
```

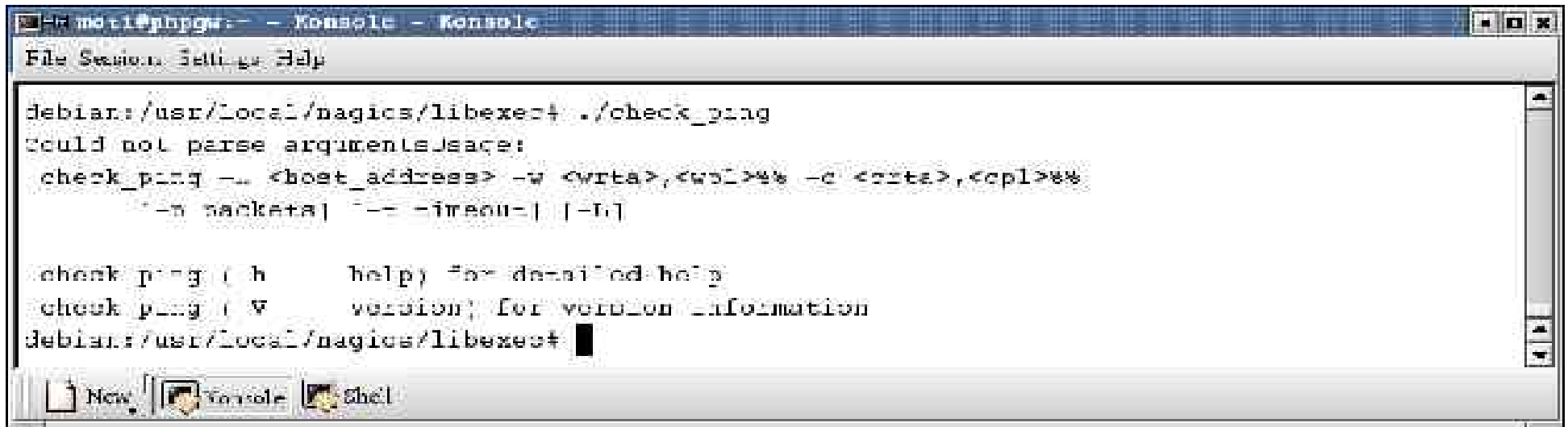
To summarize check commands

host_name **linux1**
check_command **check_ping!100.0,20%!500.0,60%**

+

```
define command{  
    command_name    check_ping  
    command_line    $USER1$/check_ping -H $HOSTADDRESS$  
-w $ARG1$ -c $ARG2$ -p 5  
}
```

+



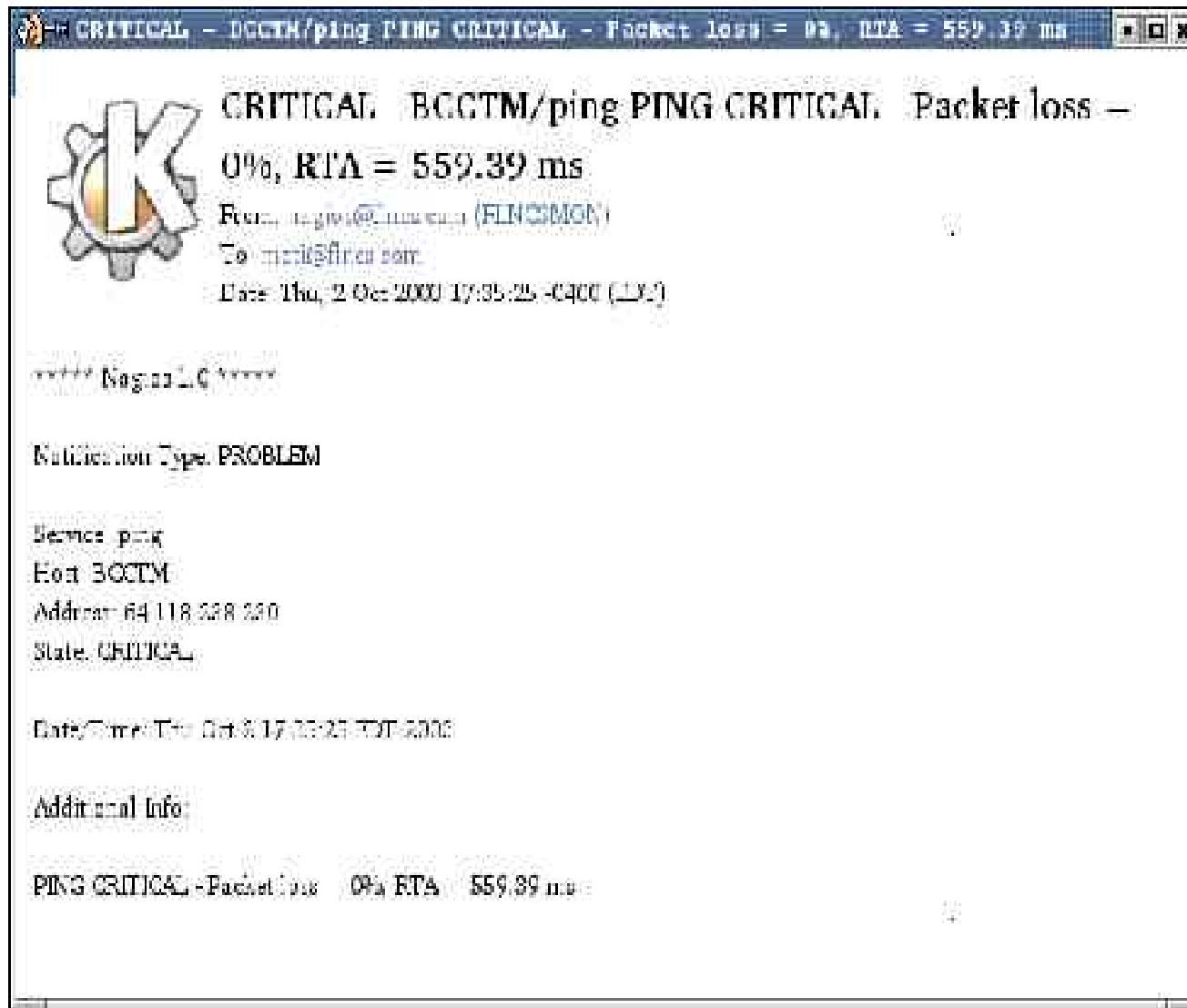
The screenshot shows a terminal window titled "Konsole - Konsole" with the following content:

```
debian:/usr/local/nagios/libexec# ./check_ping  
could not parse arguments usage:  
check_ping -- <host_address> -w <wrta>,<wpl>%% -c <crta>,<cp1>%%  
    [-n packets] [-- timeout] [-h]  
  
check_ping ( h      help) for detailed help  
check_ping ( V      version) for version information  
debian:/usr/local/nagios/libexec#
```

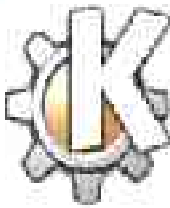
=

Check the host linux1 for ping if response time is over 100.0 miliseconds and packet loss is over 20% issue a warning state If the response time is 500 miliseconds and packet loss is 60% issue a critical state

Critical state warning



CRITICAL - BOCTM/ping PING CRITICAL - Packet loss = 0%, RTA = 559.39 ms

 **CRITICAL** BOCTM/ping PING CRITICAL. Packet loss = 0%, RTA = 559.39 ms
From: nagios@nagios.com (FLINCSMON)
To: mario@flinc.com
Date: Thu, 2 Oct 2008 17:35:25 -0400 (EDT)

***** Nagios *****

Notification Type: PROBLEM


Service: ping
Host: BOCTM
Address: 64.118.228.220
State: CRITICAL

Date/Time: Thu Oct 2 17:35:25 EDT 2008

Additional Info:

PING CRITICAL - Packet loss = 0% RTA = 559.39 ms

Recovery state warning



OK BCCTM/ping PING OK Packet loss —
0%, RTA — 106.19 ms
From: nagios@ti.com (fr.MCMON)
To: mcm@ti.com
Date: Thu, 2 Oct 2008 18:25:29 -0400 (EDT)

Nagios - C Pxxxx

Notification Type: RECOVERY

Screen: ping
Item: BCCTM
Address: 64.1.8.238.225
State: OK

Date/Time: Thu Oct 2 18 25 29 EDT 2008

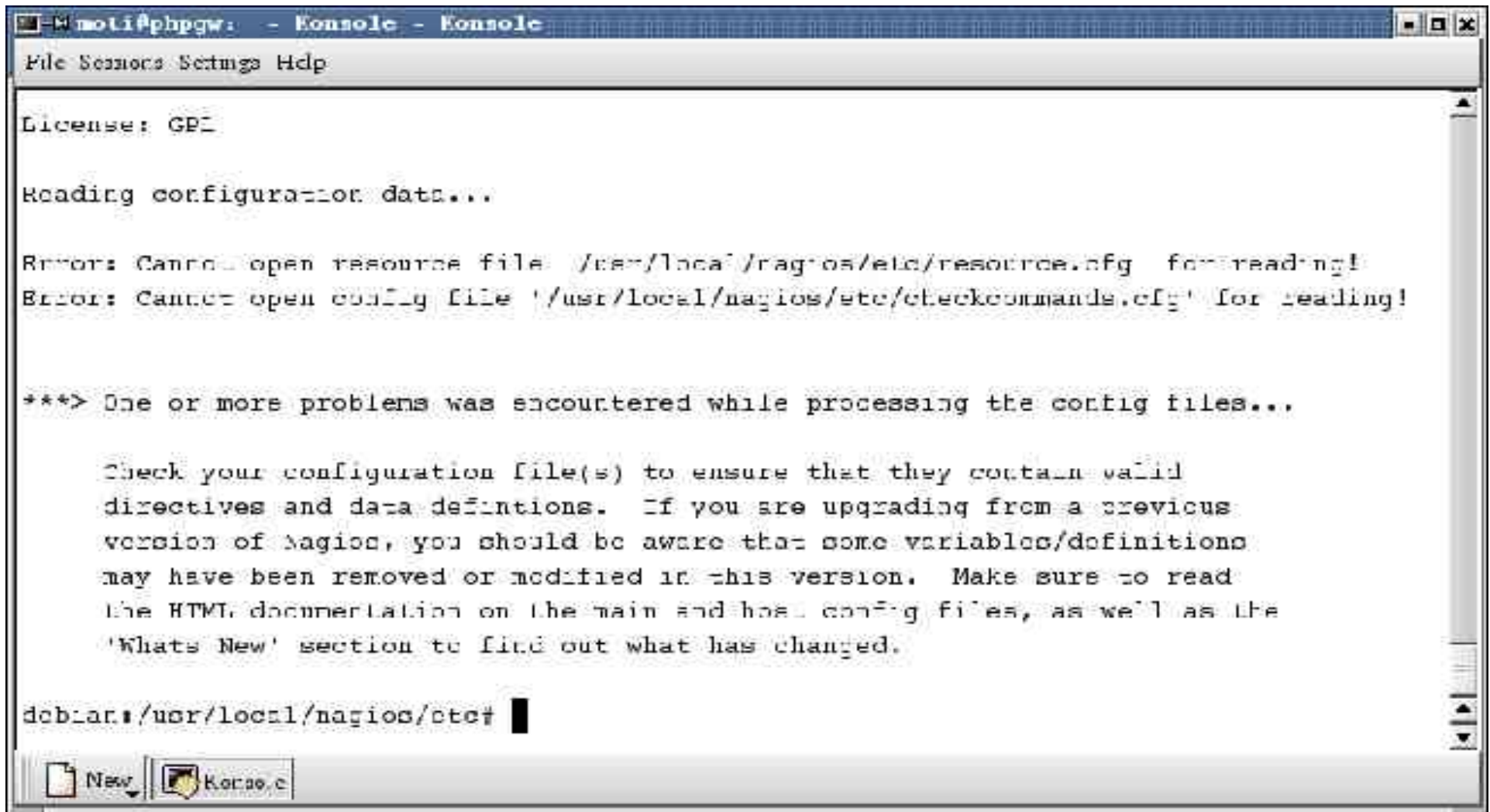
Additional info:

PING OK - Packet loss: 0%, RTA: 106.19 ms

Verify the config

Nagios -v nagios.cfg – will run a check on all services and will print a summary of problems it detected.

Oh No

A screenshot of a terminal window titled 'moti@phpgw: - Konsole - Konsole'. The window shows the output of the command 'nagios -v nagios.cfg'. The output includes the license type 'GPL', a message 'Reading configuration data...', and two error messages: 'Error: Cannot open resource file '/usr/local/nagios/etc/resource.cfg' for reading!' and 'Error: Cannot open config file '/usr/local/nagios/etc/checkcommands.cfg' for reading!'. A summary message states: '***> One or more problems was encountered while processing the config files...'. Below this, a detailed message explains that the user should check their configuration files for valid directives and data definitions, especially if upgrading from a previous version. The terminal prompt is 'debiana:/usr/local/nagios/etc#'. The window has a standard Linux desktop environment with a taskbar at the bottom showing 'New' and 'Konsole' icons.

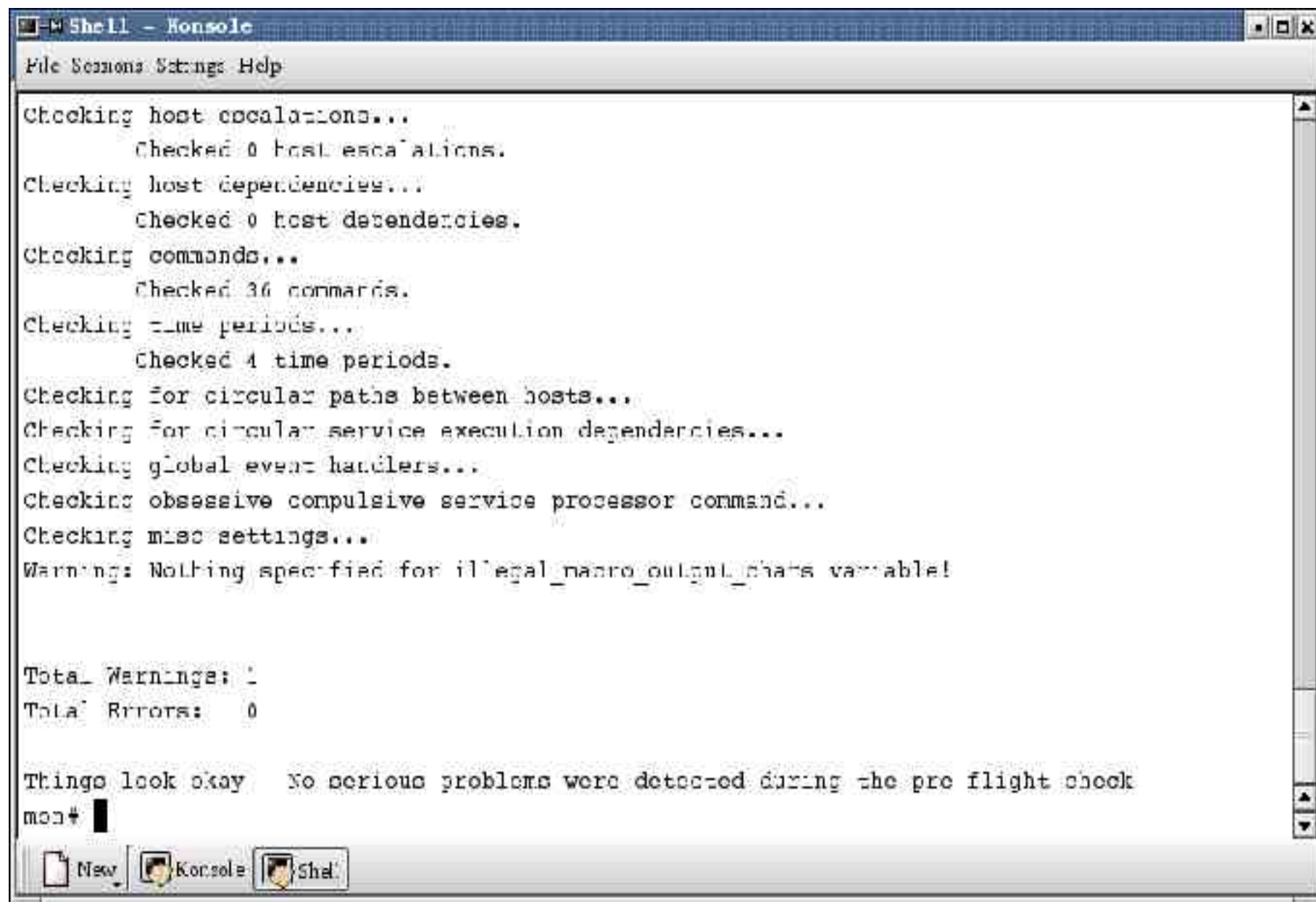
```
moti@phpgw: - Konsole - Konsole
File Sensors Settings Help
License: GPL
Reading configuration data...
Error: Cannot open resource file '/usr/local/nagios/etc/resource.cfg' for reading!
Error: Cannot open config file '/usr/local/nagios/etc/checkcommands.cfg' for reading!

***> One or more problems was encountered while processing the config files...

Check your configuration file(s) to ensure that they contain valid
directives and data definitions.  If you are upgrading from a previous
version of Nagios, you should be aware that some variables/definitions
may have been removed or modified in this version.  Make sure to read
the HTML documentation on the main and host config files, as well as the
'Whats New' section to find out what has changed.

debiana:/usr/local/nagios/etc#
```

Oh Yeah



```
Shell - Konsole
File Sessions Settings Help

Checking host escalations...
    Checked 0 host escalations.
Checking host dependencies...
    Checked 0 host dependencies.
Checking commands...
    Checked 36 commands.
Checking time periods...
    Checked 4 time periods.
Checking for circular paths between hosts...
Checking for circular service execution dependencies...
Checking global event handlers...
Checking obsessive compulsive service processor command...
Checking misc settings...
Warnings: Nothing specified for illegal_macro_output_chars variable!

Total Warnings: 1
Total Errors: 0

Things look okay. No serious problems were detected during the pre flight check
man#
```

Local Resources

In order to monitor local resources we need to components

- 1) a daemon (*nix) or service (m\$) running on the remote hosts
- 2) a nagios plugin to collect data from the above

nagios web site has a few options for both *nix & m\$ addons , I have had great success with the following two :

Statd – will run on almost any unix (even sco ...) , it is a python script that will enable us to monitor the following resources :
cpu,mem usage,running process,disk space and more

NSClient – is an nt service which will run on NT4,2000 & Xpect Problems ,It will enable us to monitor cpu,mem usage ,
running process ,service states and disk space

Statd install

- Verify you have python installed
- Extract the tar ball
- sbin/nagios-statd – runs on the client , should be started on boot (rc.local)
- sbin/nagios-stat – copy to nagios plugin directory (/usr/local/nagios/libexec)

NSclient install

- Download and unzip the files
- Copy pnsclient.exe to a folder of your choice and run pnsclient / install from a command prompt – this will install “netsaint NT agent “ service
- Copy check_nt to nagios plugin directory (/usr/local/nagios/libexec)

Sample statd service check

first we define the check command

Filename: checkcommands.cfg

```
define command{
    command_name    nagios-stat-disk
    command_line    $USER1$/nagios-stat -d $ARG1$ -w $ARG2$ -c $ARG3$
$ARG4$ $HOSTADDRESS$
}
```

then we define a service check

Filename: services.cfg

```
define service {
    host_name        linux-mail-server
    service_description /var partition
    check_command    nagios-stat!/dev/ida/c0d0p3!80!90!disk
    max_check_attempts 3
    normal_check_interval 5
    retry_check_interval 1
    check_period     24x7
    notification_interval 0
    notification_period 24x7
    notification_options w,c,r
    contact_groups   admin
}
```

Sample nsclient service check

first we define the check command

Filename: checkcommands.cfg

```
define command {  
    command_name    check_nt_disk  
    command_line    $USER1$/check_nt -H $HOSTADDRESS$ -p 1248 -v  
USEDISKSPACE -l $ARG1$ -w $ARG2$ -c $ARG3$  
}
```

then we define a service check

Filename: services.cfg

```
define service{  
    host_name        windows mail server ( no , really )  
    service_description    DISK  
    check_period      24x7  
    max_check_attempts    3  
    normal_check_interval    10  
    retry_check_interval    1  
    contact_groups    admin  
    notification_interval    30  
    notification_period    24x7  
    notification_options    w,c,r  
    check_command      check_nt_disk!C!85%!90%  
}
```

Hey , WAKE UP ! It's almost over

- \$SRC/daemon-init is a start/stop script you can use
- Use the sample config files as reference
- READ the documentation
- Stuck ? Try the FAQ
- Setup done ? Start tweaking ...
- Tweaking done ? Read the documentation again !
 - Dependencies
 - Escalations
 - Templats
 - Icons
 - Wap
 - And more ...
- **Monitor the monitor !**
- Feel free to contact me : moti@flncs.com

Syslog Server

Any host running syslogd can be a syslog server

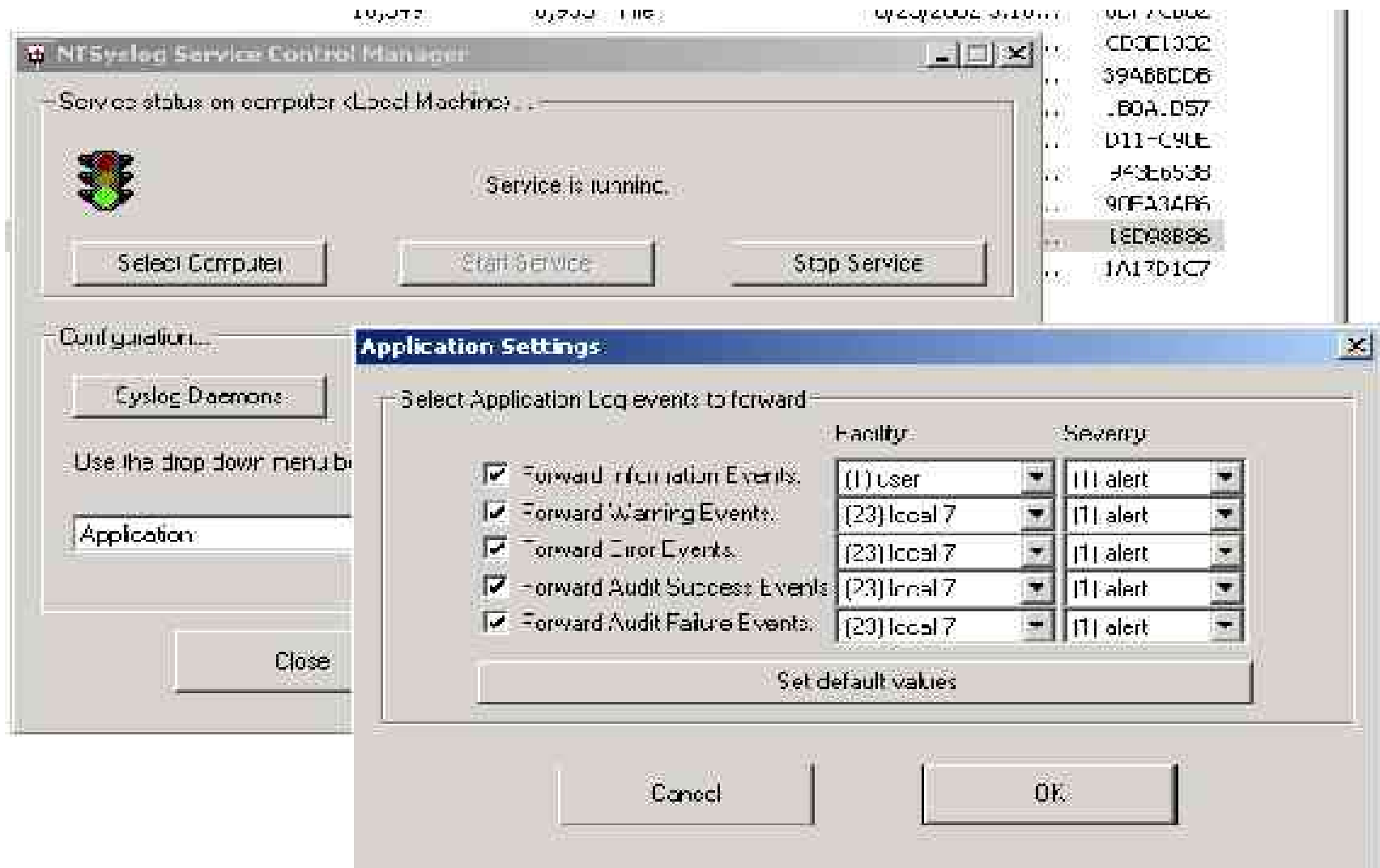
- Linux – syslogd -r (usually)

Syslog Client - Unix

- edit syslogd.conf
 - Mail.* @hostname – direct all mail messages to hostname
 - Kern.* @hostname – direct all kernel messages to hostname
 - *.* @hostname – will direct all traffic to host name
 - Restart syslogd

Syslog Client - microsoft

- M\$ clients – use ntsyslog (ntsyslog.sourceforge.net)
 - Unzip files to %windir%
 - Run ntsyslog –install
 - Use gui (ntsyslogctrl.exe) to configure



Swatch – the simple logfile watcher

- Setup is similar to a cpan module (you can use apt or rpm)
 - Unzip
 - perl Makefile.pl
 - Make
 - Make install
 - Make realclean

Swatch – syntax and options

Watchfor/Ignore /pattern/
action
options

Pattern: any pattern perl will accept

action: echo,bell – swatch must be running in foreground

exec,system,mail,pipe,write – can run in daemon mode

option: throttle,when

Swatch examples

```
watchfor  /\'su root\' failed/  
  echo bold  
  mail =me\@dom.com,subject=Failed root password  
  throttle 30:00
```

```
watchfor  /Sense Key: Media Error/  
  exec echo $0 | mail -s\"syslog alert: disk errors\"admin@email  
  throttle 30:00
```

```
watchfor  /\'Unsuccessful NTLM logon\' failed/  
  echo bold  
  mail =me\@dom.com,subject=Failed nt login  
  throttle 30:00
```

```
watchfor  /\proftpd.*- no such user 'anonymous'/  
  mail =me\@dom.com,subject=ftp pub scan
```

Wait a sec , that was too easy !

That's why it's called simple logfile watcher

- syslog-ng
- mysql
- stunnel

Final Words

I'd rather have a bottle in front of me than a frontal lobotomy

Thank you