

Installation and Configuration of Nagios 2.x

These instructions are adapted by Ric Charlton from those found at http://nagios.sourceforge.net/docs/2_0/

The instructions look lengthy, however they are mostly configuration files.

Pre-requisites

- Install a standard base system of Linux – any distro should be OK but these instructions are written using Debian Etch
- Install the Apache web server (v2)
- Install gcc and g++ compilers so that the application can be made
- Download the tarballs for the latest version of Nagios (currently v2.9) and the Nagios Plugins (currently v1.4.8)
- Root access

Installation of Nagios

- Download the latest distribution package to your home directory on the Nagios server
- Unpack the distribution package with the following command:

```
# tar xzvf nagios-version.tar.gz
```

- Create the 'nagios' user – this user will be used to run the Nagios program

```
# adduser nagios
```

- You will be prompted for a password and then some user details... you may leave the user details blank but it's advisable to enter a password ;-)
- Create the installation directory and give ownership to the new nagios user

```
# mkdir /usr/local/nagios
```

```
# chown nagios /usr/local/nagios
```

- Create a group for issuing commands to Nagios from the web interface... you must add your Nagios user and Apache user to this group (in my case this is www-data)

```
# groupadd nagioscmd
```

```
# usermod -G nagioscmd www-data
```

```
# usermod -G nagioscmd nagios
```

- Run the configure script as follows:

```
# ./configure --prefix=/usr/local/nagios --with-cgiurl=/nagios/cgi-bin --with-htmurl=/nagios --with-nagios-user=nagios --with-nagios-group=nagios --with-command-group=nagioscmd
```

- Check that the options are as expected and then run make to compile the application

```
# make all
```

- Run make scripts to install Nagios, the init scripts and configure some permissions

```
# make install
```

```
# make install-init
```

```
# make install-commandmode
```

- Do not run the final script to install the example configs as these are not needed

Installing the Plugins

- Unpack the plugins package

```
# tar xzvf nagios-plugins-version.tar.gz
# ./configure --prefix=/usr/local/nagios --with-cgi-url=/nagios/cgi-bin
# make
# make install
```

Setup Apache Web Server

This section may be different if you are not using a Debian-based distro

- Create a file in /etc/apache2/sites-available called nagios which contains the following

```
ScriptAlias /nagios/cgi-bin /usr/local/nagios/sbin

<Directory "/usr/local/nagios/sbin">
    Options ExecCGI
    AllowOverride None
    Order allow,deny
    Allow from all
    AuthName "Nagios Access"
    AuthType Basic
    AuthUserFile /usr/local/nagios/etc/htpasswd.users
    Require valid-user
</Directory>
```

```
Alias /nagios /usr/local/nagios/share
```

```
<Directory "/usr/local/nagios/share">
    Options None
    AllowOverride None
    Order allow,deny
    Allow from all
    AuthName "Nagios Access"
    AuthType Basic
    AuthUserFile /usr/local/nagios/etc/htpasswd.users
    Require valid-user
</Directory>
```

- Create a link to the above file in the directory /etc/apache2/sites-enabled

```
# cp -s /etc/apache2/sites-available/nagios /etc/apache2/sites-enabled
```

- Restart Apache

```
# /etc/init.d/apache2 restart
```

- Create the directory /usr/local/nagios/etc

```
# mkdir /usr/local/nagios/etc
```

- Create a web-user who can access Nagios

```
# htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

- Enter a password when prompted
- Additional users may be added with the command

```
# htpasswd /usr/local/nagios/etc/htpasswd.users USERNAME
```

- Further details can be found at http://nagios.sourceforge.net/docs/2_0/cgiauth.html

Configuring Nagios

Base Configuration

- You first need to add the important config files to the /usr/local/nagios/etc folder:

nagios.cfg (main configuration file)

```
#####
# NAGIOS.CFG - Main Config File for Nagios
#####
```

```
# 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) ##
# 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 tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below (this helps to tidy the config files):
cfg_dir=/usr/local/nagios/etc/conf.d


# ## OBJECT CACHE FILE ##
# This option determines where object definitions are cached when
# Nagios starts/restarts. The CGIs read object definitions from
# this cache file (rather than looking at the object config files
# directly) in order to prevent inconsistencies that can occur
# when the config files are modified after Nagios starts.
object_cache_file=/usr/local/nagios/var/objects.cache


# ## RESOURCE FILE ##
# This is an optional resource file that contains $USERx$ 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.dat

# ## 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
# Nagios to check for external commands every minute. If you specify a
# number followed by an "s" (i.e. 15s), this will be interpreted to mean
# actual seconds rather than a multiple of the interval_length variable.
# Note: In addition to reading the external command file at regularly
# scheduled intervals, Nagios will also check for external commands after
# event handlers are executed.
# NOTE: Setting this value to -1 causes Nagios to check the external
```

```
# command file as often as possible.
command_check_interval=-1

# ## EXTERNAL COMMAND FILE ##
# This is the file that Nagios checks for external command requests.
# It is also where the command CGI will write commands that are submitted
# by users, so it must be writeable by the user that the web server
# is running as (usually 'nobody'). Permissions should be set at the
# directory level instead of on the file, as the file is deleted every
# time its contents are processed.
command_file=/usr/local/nagios/var/rw/nagios.cmd

# ## COMMENT FILE ##
# This is the file that Nagios will use for storing host and service
# comments.
comment_file=/usr/local/nagios/var/comments.dat

# ## DOWNTIME FILE ##
# This is the file that Nagios will use for storing host and service
# downtime data.
downtime_file=/usr/local/nagios/var/downtime.dat

# ## LOCK FILE ##
# This is the lockfile that Nagios will use to store its PID number
# in when it is running in daemon mode.
lock_file=/usr/local/nagios/var/nagios.lock

# ## TEMP FILE ##
# This is a temporary file that is used as scratch space when Nagios
# updates the status log, cleans the comment file, etc. This file
# is created, used, and deleted throughout the time that Nagios is
# running.
temp_file=/usr/local/nagios/var/nagios.tmp

# ## EVENT BROKER OPTIONS ##
# Controls what (if any) data gets sent to the event broker.
```

```

# Values:  0      = Broker nothing
#          -1     = Broker everything
#          <other> = See documentation
event_broker_options=-1


# ## EVENT BROKER MODULE(S) ##
# This directive is used to specify an event broker module that should
# be loaded by Nagios at startup. Use multiple directives if you want
# to load more than one module. Arguments that should be passed to
# the module at startup are separated from the module path by a space.
#
# Example:
#
#   broker_module=<modulepath> [moduleargs]


# ## LOG ROTATION METHOD ##
# This is the log rotation method that Nagios should use to rotate
# the main log file. Values are as follows..
#   n      = None - don't rotate the log
#   h      = Hourly rotation (top of the hour)
#   d      = Daily rotation (midnight every day)
#   w      = Weekly rotation (midnight on Saturday evening)
#   m      = Monthly rotation (midnight last day of month)
log_rotation_method=d


# ## 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.
use_syslog=1


# ## 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.
log_host_retries=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.
log_event_handlers=1

# ## INITIAL STATES LOGGING OPTION ##
# If you want Nagios to log all initial host and service states to
# 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 state
# statistics reporting, you do not need to enable this option.
log_initial_states=0

# ## EXTERNAL COMMANDS LOGGING OPTION ##
# If you don't want Nagios to log external commands, set this value
# to 0. If external commands should be logged, set this value to 1.
# Note: This option does not include logging of passive service
# checks - see the option below for controlling whether or not
# passive checks are logged.
log_external_commands=1

# ## PASSIVE CHECKS LOGGING OPTION ##
```

```
# If you don't want Nagios to log passive host and service checks, set
# this value to 0.  If passive checks should be logged, set
# this value to 1.
log_passive_checks=1

# ## GLOBAL HOST AND SERVICE EVENT HANDLERS ##
# These options allow you to specify a host and service event handler
# command that is to be run for every host or service state change.
# The global event handler is executed immediately prior to the event
# handler that you have optionally specified in each host or
# service definition. The command argument is the short name of a
# command definition that you define in your host configuration file.
# Read the HTML docs for more information.
#global_host_event_handler=somecommand
#global_service_event_handler=somecommand

# ## SERVICE INTER-CHECK DELAY METHOD ##
# This is the method that Nagios should use when initially
# "spreading out" service checks when it starts monitoring.  The
# default is to use smart delay calculation, which will try to
# space all service checks out evenly to minimize CPU load.
# Using the dumb setting will cause all checks to be scheduled
# at the same time (with no delay between them)!  This is not a
# good thing for production, but is useful when testing the
# parallelization functionality.
#      n      = None - don't use any delay between checks
#      d      = Use a "dumb" delay of 1 second between checks
#      s      = Use "smart" inter-check delay calculation
#      x.xx   = Use an inter-check delay of x.xx seconds
service_inter_check_delay_method=s

# ## MAXIMUM SERVICE CHECK SPREAD ##
# This variable determines the timeframe (in minutes) from the
# program start time that an initial check of all services should
# be completed.  Default is 30 minutes.
max_service_check_spread=30

# ## SERVICE CHECK INTERLEAVE FACTOR ##
```

```
# This variable determines how service checks are interleaved.
# Interleaving the service checks allows for a more even
# distribution of service checks and reduced load on remote
# hosts. Setting this value to 1 is equivalent to how versions
# of Nagios previous to 0.0.5 did service checks. Set this
# value to s (smart) for automatic calculation of the interleave
# factor unless you have a specific reason to change it.
#      s      = Use "smart" interleave factor calculation
#      x      = Use an interleave factor of x, where x is a
#               number greater than or equal to 1.
service_interleave_factor=s
```

``` # ## HOST INTER-CHECK DELAY METHOD ## ```

```
# This is the method that Nagios should use when initially
# "spreading out" host checks when it starts monitoring. The
# default is to use smart delay calculation, which will try to
# space all host checks out evenly to minimize CPU load.
# Using the dumb setting will cause all checks to be scheduled
# at the same time (with no delay between them)!
#      n      = None - don't use any delay between checks
#      d      = Use a "dumb" delay of 1 second between checks
#      s      = Use "smart" inter-check delay calculation
#      x.xx   = Use an inter-check delay of x.xx seconds
host_inter_check_delay_method=s
```

``` # ## MAXIMUM HOST CHECK SPREAD ## ```

```
# This variable determines the timeframe (in minutes) from the
# program start time that an initial check of all hosts should
# be completed. Default is 30 minutes.
max_host_check_spread=30
```

``` # ## MAXIMUM CONCURRENT SERVICE CHECKS ## ```

```
# 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 process
# the results of services that have been checked.
service_reaper_frequency=10

# ## AUTO-RESCHEDULING OPTION ##
# This option determines whether or not Nagios will attempt to
# automatically reschedule active host and service checks to
# "smooth" them out over time. This can help balance the load on
# the monitoring server.
# WARNING: THIS IS AN EXPERIMENTAL FEATURE - IT CAN DEGRADE
# PERFORMANCE, RATHER THAN INCREASE IT, IF USED IMPROPERLY
auto_reschedule_checks=0

# ## AUTO-RESCHEDULING INTERVAL ##
# This option determines how often (in seconds) Nagios will
# attempt to automatically reschedule checks. This option only
# has an effect if the auto_reschedule_checks option is enabled.
# Default is 30 seconds.
# WARNING: THIS IS AN EXPERIMENTAL FEATURE - IT CAN DEGRADE
# PERFORMANCE, RATHER THAN INCREASE IT, IF USED IMPROPERLY
auto_rescheduling_interval=30

# ## AUTO-RESCHEDULING WINDOW ##
# This option determines the "window" of time (in seconds) that
# Nagios will look at when automatically rescheduling checks.
# Only host and service checks that occur in the next X seconds
# (determined by this variable) will be rescheduled. This option
# only has an effect if the auto_reschedule_checks option is
# enabled. Default is 180 seconds (3 minutes).
# WARNING: THIS IS AN EXPERIMENTAL FEATURE - IT CAN DEGRADE
# PERFORMANCE, RATHER THAN INCREASE IT, IF USED IMPROPERLY
auto_rescheduling_window=180

# ## SLEEP TIME ##
# This is the number of seconds to sleep between checking for system
```

```
# events and service checks that need to be run.
sleep_time=0.25

# ## 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
# oosp 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
oosp_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 on state statistics, etc, but will
# slow Nagios down a bit when it (re)starts. Since its only
# a one-time penalty, I think its well worth the additional
# 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.
# This file is used only if the preserve_state_information
# variable is set to 1.
state_retention_file=/usr/local/nagios/var/retention.dat

# ## RETENTION DATA UPDATE INTERVAL ##
```

```
# This setting determines how often (in minutes) that Nagios
# will automatically save retention data during normal operation.
# If you set this value to 0, Nagios will not save retention
# data at regular interval, but it will still save retention
# data before shutting down or restarting.  If you have disabled
# state retention, this option has no effect.
retention_update_interval=60

# ## USE RETAINED PROGRAM STATE ##
# This setting determines whether or not Nagios will set
# program status variables based on the values saved in the
# retention file.  If you want to use retained program status
# information, set this value to 1.  If not, set this value
# to 0.
use_retained_program_state=1

# ## USE RETAINED SCHEDULING INFO ##
# This setting determines whether or not Nagios will retain
# the scheduling info (next check time) for hosts and services
# based on the values saved in the retention file.  If you
# If you want to use retained scheduling info, set this
# value to 1.  If not, set this value to 0.
use_retained_scheduling_info=0

# ## INTERVAL LENGTH ##
# This is the seconds per unit interval as used in the
# host/contact/service configuration files.  Setting this to 60 means
# that each interval is one minute long (60 seconds).  Other settings
# have not been tested much, so your mileage is likely to vary...
interval_length=60

# ## AGGRESSIVE HOST CHECKING OPTION ##
# If you don't want to turn on aggressive host checking features, set
# this value to 0 (the default).  Otherwise set this value to 1 to
# enable the aggressive check option.  Read the docs for more info
# on what aggressive host check is or check out the source code in
# base/checks.c
use_aggressive_host_checking=0
```

```
# ## 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 SERVICE 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
```

```
# ## HOST CHECK EXECUTION OPTION ##
# This determines whether or not Nagios will actively execute
# host 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 host checks, leave this enabled!
# Values: 1 = enable checks, 0 = disable checks
execute_host_checks=1
```

```
# ## PASSIVE HOST CHECK ACCEPTANCE OPTION ##
# This determines whether or not Nagios will accept passive
# host checks results when it initially (re)starts.
# Values: 1 = accept passive checks, 0 = reject passive checks
accept_passive_host_checks=1
```

```
# ## NOTIFICATIONS OPTION ##
# This determines whether or not Nagios will sent out any host or
# service notifications when it is initially (re)started.
# Values: 1 = enable notifications, 0 = disable notifications
```

```
enable_notifications=1

# ## EVENT HANDLER USE OPTION ##
# This determines whether or not Nagios will run any host or
# service event handlers when it is initially (re)started. Unless
# you're implementing redundant hosts, leave this option enabled.
# Values: 1 = enable event handlers, 0 = disable event handlers
enable_event_handlers=1

# ## PROCESS PERFORMANCE DATA OPTION ##
# This determines whether or not Nagios will process performance
# data returned from service and host checks. If this option is
# enabled, host performance data will be processed using the
# host_perfdata_command (defined below) and service performance
# data will be processed using the service_perfdata_command (also
# defined below). Read the HTML docs for more information on
# performance data.
# Values: 1 = process performance data, 0 = do not process performance data
process_performance_data=1

# ## HOST AND SERVICE PERFORMANCE DATA PROCESSING COMMANDS ##
# These commands are run after every host and service check is
# performed. These commands are executed only if the
# enable_performance_data option (above) is set to 1. The command
# argument is the short name of a command definition that you
# define in your host configuration file. Read the HTML docs for
# more information on performance data.
host_perfdata_command=process-host-perfdata
service_perfdata_command=process-service-perfdata

# ## HOST AND SERVICE PERFORMANCE DATA FILES ##
# These files are used to store host and service performance data.
# Performance data is only written to these files if the
# enable_performance_data option (above) is set to 1.
host_perfdata_file=/tmp/host-perfdata
service_perfdata_file=/tmp/service-perfdata
```

```
# ## HOST AND SERVICE PERFORMANCE DATA FILE TEMPLATES ##
# These options determine what data is written (and how) to the
# performance data files. The templates may contain macros, special
# characters (\t for tab, \r for carriage return, \n for newline)
# and plain text. A newline is automatically added after each write
# to the performance data file. Some examples of what you can do are
# shown below.
host_perfdata_file_template=[HOSTPERFDATA]\t$TIMET$\t$HOSTNAME$\t$HOSTEXECUTIONTIME$\t$HOSTOUTPUT$\t$HOSTPERFDATA$
service_perfdata_file_template=[SERVICEPERFDATA]\t$TIMET$\t$HOSTNAME$\t$SERVICEDESC$\t$SERVICEEXECUTIONTIME$\t$SERVICELATENCY$\t$SERVICEOUTPUT$\t$SERVICEPERFDATA$

# ## HOST AND SERVICE PERFORMANCE DATA FILE MODES ##
# This option determines whether or not the host and service
# performance data files are opened in write ("w") or append ("a")
# mode. Unless you are the files are named pipes, you will probably
# want to use the default mode of append ("a").
host_perfdata_file_mode=a
service_perfdata_file_mode=a

# ## HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING INTERVAL ##
# These options determine how often (in seconds) the host and service
# performance data files are processed using the commands defined
# below. A value of 0 indicates the files should not be periodically
# processed.
host_perfdata_file_processing_interval=120
service_perfdata_file_processing_interval=120

# ## OBSESS OVER SERVICE CHECKS OPTION ##
# This determines whether or not Nagios will obsess over service
# checks and run the oosp_command defined below. Unless you're
# planning on implementing distributed monitoring, do not enable
# this option. Read the HTML docs for more information on
# implementing distributed monitoring.
# Values: 1 = obsess over services, 0 = do not obsess (default)
obsess_over_services=0

# ## OBSESSIVE COMPULSIVE SERVICE PROCESSOR COMMAND ##
```

```
# This is the command that is run for every service check that is
# processed by Nagios.  This command is executed only if the
# obsess_over_service option (above) is set to 1.  The command
# argument is the short name of a command definition that you
# define in your host configuration file. Read the HTML docs for
# more information on implementing distributed monitoring.
#ocsp_command=somecommand

# ## 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 some
# 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_services=0

# ## 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

# ## SERVICE 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.
service_freshness_check_interval=60

# ## HOST FRESHNESS CHECK OPTION ##
# This option determines whether or not Nagios will periodically
# check the "freshness" of host 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_host_freshness=0

# ## HOST FRESHNESS CHECK INTERVAL ##
# This setting determines how often (in seconds) Nagios will
# check the "freshness" of host check results.  If you have
# disabled host freshness checking, this option has no effect.
host_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 are 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)
date_format=euro

# ## P1.PL FILE LOCATION ##
# This value determines where the p1.pl perl script (used by the
# embedded Perl interpreter) is located. If you didn't compile
# Nagios with embedded Perl support, this option has no effect.
pl_file=/usr/local/nagios/bin/p1.pl

# ## ILLEGAL OBJECT NAME CHARACTERS ##
# This option 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 option 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:
#     $HOSTOUTPUT$
#     $HOSTPERFDATA$
#     $HOSTACKAUTHOR$
#     $HOSTACKCOMMENT$
#     $SERVICEOUTPUT$
#     $SERVICEPERFDATA$
#     $SERVICEACKAUTHOR$
#     $SERVICEACKCOMMENT$
illegal_macro_output_chars=~$&|' "<>

## REGULAR EXPRESSION MATCHING ##
# This option controls whether or not regular expression matching
# takes place in the object config files. Regular expression
# matching is used to match host, hostgroup, service, and service
# group names/descriptions in some fields of various object types.
# Values: 1 = enable regexp matching, 0 = disable regexp matching
use_regexp_matching=0

## "TRUE" REGULAR EXPRESSION MATCHING ##
# This option controls whether or not "true" regular expression
# matching takes place in the object config files. This option
# only has an effect if regular expression matching is enabled
# (see above). If this option is DISABLED, regular expression
# matching only occurs if a string contains wildcard characters
# (* and ?). If the option is ENABLED, regexp matching occurs
# all the time (which can be annoying).
# Values: 1 = enable true matching, 0 = disable true matching
use_true_regexp_matching=0

## ADMINISTRATOR EMAIL ADDRESS ##
# The email address of the administrator of *this* machine (the one
# doing the monitoring). Nagios never uses this value itself, but
# you can access this value by using the $ADMINEMAIL$ macro in your
# notification commands.
admin_email=nagios
```

```
# ## ADMINISTRATOR PAGER NUMBER/ADDRESS ##
# The pager number/address for the administrator of *this* machine.
# Nagios never uses this value itself, but you can access this
# value by using the $ADMINPAGER$ macro in your notification
# commands.
admin_pager=pagenagios
```

```
# ## DAEMON CORE DUMP OPTION ##
# This option determines whether or not Nagios is allowed to create
# a core dump when it runs as a daemon. Note that it is generally
# considered bad form to allow this, but it may be useful for
# debugging purposes.
# Values: 1 - Allow core dumps
#          0 - Do not allow core dumps (default)
daemon_dumps_core=0
```

```
# ## EOF (End of file) ##
```

cgi.cfg (configuration for the CGIs)

```
#####
# CGI.CFG - CGI Configuration File for Nagios
#####
```

```
# ## MAIN CONFIGURATION FILE ##
# This tells the CGIs where to find your main configuration file.
# The CGIs will read the main and host config files for any other
# data they might need.
```

```
main_config_file=/usr/local/nagios/etc/nagios.cfg
```

```
# ## PHYSICAL HTML PATH ##
# This is the path where the HTML files for Nagios reside. This
# value is used to locate the logo images needed by the statusmap
# and statuswrl CGIs.
physical_html_path=/usr/local/nagios/share
```

```
# ## URL HTML PATH ##
# This is the path portion of the URL that corresponds to the
# physical location of the Nagios HTML files (as defined above).
# This value is used by the CGIs to locate the online documentation
```

```
# and graphics. If you access the Nagios pages with an URL like
# http://www.myhost.com/nagios, this value should be '/nagios'
# (without the quotes).
url_html_path=/nagios

# ## CONTEXT-SENSITIVE HELP ##
# This option determines whether or not a context-sensitive
# help icon will be displayed for most of the CGIs.
# Values: 0 = disables context-sensitive help
#         1 = enables context-sensitive help
show_context_help=0

# ## NAGIOS PROCESS CHECK COMMAND ##
# This is the full path and filename of the program used to check
# the status of the Nagios process. It is used only by the CGIs
# and is completely optional. However, if you don't use it, you'll
# see warning messages in the CGIs about the Nagios process
# not running and you won't be able to execute any commands from
# the web interface. The program should follow the same rules
# as plugins; the return codes are the same as for the plugins,
# it should have timeout protection, it should output something
# to STDOUT, etc.
# Note: The command line for the check_nagios plugin below may
# have to be tweaked a bit, as different versions of the plugin
# use different command line arguments/syntaxes.
nagios_check_command=/usr/local/nagios/libexec/check_nagios
/usr/local/nagios/var/status.dat 5 '/usr/local/nagios/bin/nagios'

# ## AUTHENTICATION USAGE ##
# This option controls whether or not the CGIs will use any
# authentication when displaying host and service information, as
# well as committing commands to Nagios for processing.
#
# Read the HTML documentation to learn how the authorization works!
#
# NOTE: It is a really *bad* idea to disable authorization, unless
# you plan on removing the command CGI (cmd.cgi)! Failure to do
# so will leave you wide open to kiddies messing with Nagios and
# possibly hitting you with a denial of service attack by filling up
```

```
# your drive by continuously writing to your command file!
#
# Setting this value to 0 will cause the CGIs to *not* use
# authentication (bad idea), while any other value will make them
# use the authentication functions (the default).
use_authentication=1

# ## DEFAULT USER ##
# Setting this variable will define a default user name that can
# access pages without authentication. This allows people within a
# secure domain (i.e., behind a firewall) to see the current status
# without authenticating. You may want to use this to avoid basic
# authentication if you are not using a secure server since basic
# authentication transmits passwords in the clear.
#
# Important: Do not define a default username unless you are
# running a secure web server and are sure that everyone who has
# access to the CGIs has been authenticated in some manner! If you
# define this variable, anyone who has not authenticated to the web
# server will inherit all rights you assign to this user!
#default_user_name=guest

# ## SYSTEM/PROCESS INFORMATION ACCESS ##
# This option is a comma-delimited list of all usernames that
# have access to viewing the Nagios process information as
# provided by the Extended Information CGI (extinfo.cgi). By
# default, *no one* has access to this unless you choose to
# not use authorization. You may use an asterisk (*) to
# authorize any user who has authenticated to the web server.
authorized_for_system_information=nagiosadmin

# ## CONFIGURATION INFORMATION ACCESS ##
# This option is a comma-delimited list of all usernames that
# can view ALL configuration information (hosts, commands, etc).
# By default, users can only view configuration information
# for the hosts and services they are contacts for. You may use
# an asterisk (*) to authorize any user who has authenticated
# to the web server.
authorized_for_configuration_information=nagiosadmin
```

```
# ## SYSTEM/PROCESS COMMAND ACCESS ##
```

```
# This option is a comma-delimited list of all usernames that  
# can issue shutdown and restart commands to Nagios via the  
# command CGI (cmd.cgi). Users in this list can also change  
# the program mode to active or standby. By default, *no one*  
# has access to this unless you choose to not use authorization.  
# You may use an asterisk (*) to authorize any user who has  
# authenticated to the web server.
```

```
authorized_for_system_commands=nagiosadmin
```

```
# ## GLOBAL HOST/SERVICE VIEW ACCESS ##
```

```
# These two options are comma-delimited lists of all usernames that  
# can view information for all hosts and services that are being  
# monitored. By default, users can only view information  
# for hosts or services that they are contacts for (unless you  
# you choose to not use authorization). You may use an asterisk (*)  
# to authorize any user who has authenticated to the web server.
```

```
authorized_for_all_services=nagiosadmin,guest
```

```
authorized_for_all_hosts=nagiosadmin,guest
```

```
# ## GLOBAL HOST/SERVICE COMMAND ACCESS ##
```

```
# These two options are comma-delimited lists of all usernames that  
# can issue host or service related commands via the command  
# CGI (cmd.cgi) for all hosts and services that are being monitored.  
# By default, users can only issue commands for hosts or services  
# that they are contacts for (unless you you choose to not use  
# authorization). You may use an asterisk (*) to authorize any  
# user who has authenticated to the web server.
```

```
authorized_for_all_service_commands=nagiosadmin
```

```
authorized_for_all_host_commands=nagiosadmin
```

```
# ## STATUSMAP BACKGROUND IMAGE ##
```

```
# This option allows you to specify an image to be used as a  
# background in the statusmap CGI. It is assumed that the image  
# resides in the HTML images path (i.e. /usr/local/nagios/share/images).  
# This path is automatically determined by appending "/images"  
# to the path specified by the 'physical_html_path' directive.
```

```
# Note: The image file may be in GIF, PNG, JPEG, or GD2 format.
# However, I recommend that you convert your image to GD2 format
# (uncompressed), as this will cause less CPU load when the CGI
# generates the image.
#statusmap_background_image=smbbackground.gd2

# ## DEFAULT STATUSMAP LAYOUT METHOD ##
# This option allows you to specify the default layout method
# the statusmap CGI should use for drawing hosts. If you do
# not use this option, the default is to use user-defined
# coordinates. Valid options are as follows:
#     0 = User-defined coordinates
#     1 = Depth layers
#     2 = Collapsed tree
#     3 = Balanced tree
#     4 = Circular
#     5 = Circular (Marked Up)
default_statusmap_layout=5

# ## DEFAULT STATUSWRL LAYOUT METHOD ##
# This option allows you to specify the default layout method
# the statuswrl (VRML) CGI should use for drawing hosts. If you
# do not use this option, the default is to use user-defined
# coordinates. Valid options are as follows:
#     0 = User-defined coordinates
#     2 = Collapsed tree
#     3 = Balanced tree
#     4 = Circular
default_statuswrl_layout=4

# ## STATUSWRL INCLUDE ##
# This option allows you to include your own objects in the
# generated VRML world. It is assumed that the file
# resides in the HTML path (i.e. /usr/local/nagios/share).
#statuswrl_include=myworld.wrl

# ## PING SYNTAX ##
# This option determines what syntax should be used when
```

```

# attempting to ping a host from the WAP interface (using
# the statuswml CGI.  You must include the full path to
# the ping binary, along with all required options.  The
# $HOSTADDRESS$ macro is substituted with the address of
# the host before the command is executed.
# Please note that the syntax for the ping binary is
# notorious for being different on virtually ever *NIX
# OS and distribution, so you may have to tweak this to
# work on your system.
ping_syntax=/bin/ping -n -U -c 5 $HOSTADDRESS$

# ## REFRESH RATE ##
# This option allows you to specify the refresh rate in seconds
# of various CGIs (status, statusmap, extinfo, and outages).
refresh_rate=90

# ## SOUND OPTIONS ##
# These options allow you to specify an optional audio file
# that should be played in your browser window when there are
# problems on the network.  The audio files are used only in
# the status CGI.  Only the sound for the most critical problem
# will be played.  Order of importance (higher to lower) is as
# follows: unreachable hosts, down hosts, critical services,
# warning services, and unknown services.  If there are no
# visible problems, the sound file optionally specified by
# 'normal_sound' variable will be played.
#
# <varname>=<sound_file>
# Note: All audio files must be placed in the /media subdirectory
# under the HTML path (i.e. /usr/local/nagios/share/media/).
#host_unreachable_sound=hostdown.wav
#host_down_sound=hostdown.wav
#service_critical_sound=critical.wav
#service_warning_sound=warning.wav
#service_unknown_sound=warning.wav
#normal_sound=noproblem.wav

# ## End of File ##

```

resource.cfg (contains variables)

```
#####
```

```

# RESOURCE.CFG - Resource File for Nagios
#
# You can define $USERx$ macros in this file, which can in turn be used
# in command definitions in your host config file(s). $USERx$ macros are
# useful for storing sensitive information such as usernames, passwords,
# etc. They are also handy for specifying the path to plugins and
# event handlers - if you decide to move the plugins or event handlers to
# a different directory in the future, you can just update one or two
# $USERx$ macros, instead of modifying a lot of command definitions.
#
# The CGIs will not attempt to read the contents of resource files, so
# you can set restrictive permissions (600 or 660) on them.
#
# Nagios supports up to 32 $USERx$ macros ($USER1$ through $USER32$)
#
# Resource files may also be used to store configuration directives for
# external data sources like MySQL...
#####

# Sets $USER1$ to be the path to the plugins
$USER1$=/usr/local/nagios/libexec
# Sets $USER2$ to be the path to event handlers
#$USER2$=/usr/local/nagios/libexec/eventhandlers
# Store some usernames and passwords (hidden from the CGIs)
#$USER3$=someuser
#$USER4$=somepassword

# ## DB STATUS DATA ##
# Note: These config directives are only used if you compiled
# in database support for status data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'programstatus', 'hoststatus',
# and 'servicestatus' tables in the database.
#xsddb_host=somehost
#xsddb_port=someport
#xsddb_database=nagios
#xsddb_username=nagios
#xsddb_password=password
#xsddb_optimize_data=1
#xsddb_optimize_interval=3600

```

```
# ## DB COMMENT DATA ##
# Note: These config directives are only used if you compiled
# in database support for comment data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'hostcomments' and 'servicecomments'
# tables in the database.
#xcddb_host=somehost
#xcddb_port=someport
#xcddb_database=nagios
#xcddb_username=nagios
#xcddb_password=password
#xcddb_optimize_data=1
```

```
# ## DB DOWNTIME DATA ##
# Note: These config directives are only used if you compiled
# in database support for downtime data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'hostdowntime' and 'servicedowntime'
# tables in the database.
#xdddb_host=somehost
#xdddb_port=someport
#xdddb_database=nagios
#xdddb_username=nagios
#xdddb_password=password
#xdddb_optimize_data=1
```

```
# ## DB RETENTION DATA ##
# Note: These config directives are only used if you compiled
# in database support for retention data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'programretention', 'hostretention',
# and 'serviceretention' tables in the database.
#xrddb_host=somehost
#xrddb_port=someport
#xrddb_database=nagios
#xrddb_username=nagios
#xrddb_password=password
#xrddb_optimize_data=1
```

```
# ## End of File ##
```

misccommands.cfg (contains notification and performance tracking bits)

```
#####
```

```
# NOTIFICATION COMMANDS
```

```
#####
```

```
# 'host-notify-by-email' command definition
```

```
define command{
```

```
    command_name        host-notify-by-email
```

```
    command_line        /usr/bin/printf "%b" "***** Nagios
```

```
*****\n\nNotification Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState:
```

```
$HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time:
```

```
$LONGDATETIME$\n" | /usr/bin/mail -s "Host $HOSTSTATE$ alert for $HOSTNAME$!"  
$CONTACTEMAIL$
```

```
    }
```

```
# 'host-notify-by-epager' command definition
```

```
define command{
```

```
    command_name        host-notify-by-epager
```

```
    command_line        /usr/bin/printf "%b" "Host '$HOSTALIAS$' is
```

```
$HOSTSTATE$\nInfo: $HOSTOUTPUT$\nTime: $LONGDATETIME$" | /usr/bin/mail -s
```

```
"$NOTIFICATIONTYPE$ alert - Host $HOSTNAME$ is $HOSTSTATE$" $CONTACTPAGER$
```

```
    }
```

```
# 'notify-by-email' command definition
```

```
define command{
```

```
    command_name        notify-by-email
```

```
    command_line        /usr/bin/printf "%b" "***** Nagios
```

```
*****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost:
```

```
$HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time:
```

```
$LONGDATETIME$\n\nAdditional Info:\n\n$SERVICEOUTPUT$" | /usr/bin/mail -s "**
```

```
$NOTIFICATIONTYPE$ alert - $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **"
```

```
$CONTACTEMAIL$
```

```
    }
```

```
# 'notify-by-epager' command definition
```

```
define command{
```

```
    command_name        notify-by-epager
```

```
    command_line        /usr/bin/printf "%b" "Service: $SERVICEDESC$\nHost:
```

```
$HOSTNAME$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\nInfo:
```

```
$SERVICEOUTPUT$\nDate: $LONGDATETIME$" | /usr/bin/mail -s "$NOTIFICATIONTYPE$:
```

```
$HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$" $CONTACTPAGER$
```

```
    }
```

```
#####
```

```
# PERFORMANCE DATA COMMANDS
```

```
#
```

```

# These are sample performance data commands that can be used to
# send performance data output to two text files (one for hosts, another
# for services).  If you plan on simply writing performance data out to a
# file, consider using the host_perfdata_file and service_perfdata_file
# options in the main config file.
#
#####

# 'process-host-perfdata' command definition
define command{
    command_name        process-host-perfdata
    command_line        /usr/bin/printf "%b"
"$LASTHOSTCHECK$\t$HOSTNAME$\t$HOSTSTATE$\t$HOSTATTEMPT$\t$HOSTSTATETYPE$\t$HOST
EXECUTIONTIME$\t$HOSTOUTPUT$\t$HOSTPERFDATA$\n" >> /usr/local/nagios/var/host-
perfdata.out
    }

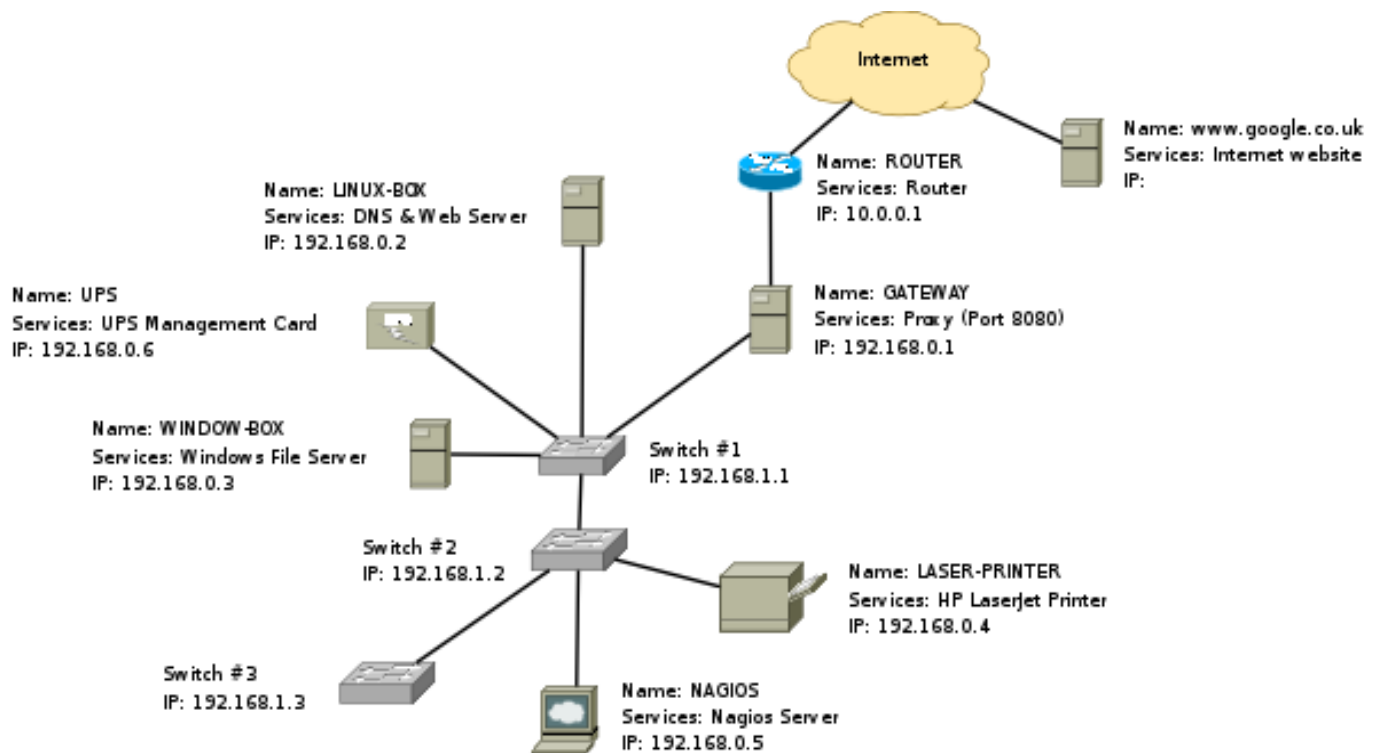
# 'process-service-perfdata' command definition
define command{
    command_name        process-service-perfdata
    command_line        /usr/bin/printf "%b"
"$LASTSERVICECHECK$\t$HOSTNAME$\t$SERVICEDESC$\t$SERVICESTATE$\t$SERVICEATTEMPT$
\t$SERVICESTATETYPE$\t$SERVICEEXECUTIONTIME$\t$SERVICELATENCY$\t$SERVICEOUTPUT$
\t$SERVICEPERFDATA$\n" >> /usr/local/nagios/var/service-perfdata.out
    }

# ## End of File ##

```

Monitoring services

In this document I will be refering to the following network:



- This simplified network will show how to monitor the common services. It is then easy to extend this to any number of hosts.

Keeping the config files neat and tidy

It is possible to have all the config in one file. This will work fine but can be a pain to debug and making changes are not as easy. Instead, we have already specified that the directory `/usr/local/nagios/etc/conf.d` will contain a number of `.cfg` files containing the config data.

In this example, we will split the hosts into separate groups:

- Servers
- Printers
- Switches
- Misc

Each group can then be split into two config files containing info about

- Hosts
- Services

The files for these will be called:

- `hostsServers.cfg`
- `servicesServers.cfg`
- `hostsPrinters.cfg`
- `servicesPrinters.cfg`
- `hostsSwitches.cfg`
- `servicesSwitches.cfg`
- `hostsMisc.cfg`
- `servicesMisc.cfg`

In addition to the eight files above we will also add the following files:

- `contacts.cfg`

- Who to inform
- contactgroups.cfg
 - How the contacts are grouped
- timeperiods.cfg
 - When to notify people
- hostgroups.cfg
 - How the hosts are grouped
- extinfo.cfg
 - Extended information (e.g. which images the CGIs should use for the different hosts)

By using templates we can also keep common syntax outside of the hosts and services files. This makes it quicker to add to the files and makes them considerably shorter!

- hostsTemplates.cfg
- servicesTemplates.cfg

Starting with the simple stuff

- To keep things simple we will assume that there is one user that is logged onto the Nagios box as 'nagios' and that email is delivered locally.
- Create the following files:

contacts.cfg

```
#####
# CONTACT DEFINITIONS
#####
# 'nagios' contact definition
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
    host_notification_commands host-notify-by-email
    email                 nagios@localhost
}

# ## End of File ##
```

contactgroups.cfg

```
#####
# CONTACT GROUP DEFINITIONS
#####
# 'windows-admins' contact group definition
define contactgroup{
```

```

        contactgroup_name windows-admins
        alias                Windows Administrators
        members              nagios
    }

# 'linux-admins' contact group definition
define contactgroup{
    contactgroup_name linux-admins
    alias                Linux Administrators
    members              nagios
}

# 'switch-admins' contact group definition
define contactgroup{
    contactgroup_name switch-admins
    alias                Switch Administrators
    members              nagios
}

# 'printer-admins' contact group definition
define contactgroup{
    contactgroup_name printer-admins
    alias                Printer Administrators
    members              nagios
}

# 'misc-admins' contact group definition
define contactgroup{
    contactgroup_name misc-admins
    alias                Misc Device Administrators
    members              nagios
}

# ## End of file ##

```

- It is not necessary to have multiple contact groups but it makes it easier to assign specific systems to different members of a support team.

timeperiods.cfg

```

#####
# TIMEPERIOD DEFINITIONS
#####
# '24x7' timeperiod definition
define timeperiod{

```

```
timeperiod_name 24x7
alias           24 Hours A Day, 7 Days A Week
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
friday          00:00-24:00
saturday        00:00-24:00
}
```

```
# 'workhours' timeperiod definition (The work hours set are
# the times that most staff are working – if people turn their
# printers off overnight you could end up receiving notifications
# for 30mins before they get in the next day!
```

```
define timeperiod{
    timeperiod_name  workhours
    alias             "Normal" Working Hours
    monday            08:30-16:00
    tuesday           08:30-16:00
    wednesday         08:30-16:00
    thursday          08:30-16:00
    friday            08:30-16:00
}
```

```
# 'nonworkhours' timeperiod definition
```

```
define timeperiod{
    timeperiod_name  nonworkhours
    alias            Non-Work Hours
    sunday           00:00-24:00
    monday           00:00-08:30,16:00-24:00
    tuesday          00:00-08:30,16:00-24:00
    wednesday        00:00-08:30,16:00-24:00
    thursday         00:00-08:30,16:00-24:00
    friday           00:00-08:30,16:00-24:00
    saturday         00:00-24:00
}
```

```
# 'none' timeperiod definition
```

```
define timeperiod{
    timeperiod_name  none
    alias            No Time Is A Good Time
}
```

```
}
```

```
# ## End of File ##
```

Configuring the checks

- Firstly we need to create the template files which will contain the common options of the host and service definitions:

hostsTemplates.cfg

```
##### Templates for Host Definitions #####
define host{
    name                generic-host ; The name of this host
template - referenced in other host definitions, used for template
recursion/resolution
    notifications_enabled 1          ; Host notifications are
enabled
    event_handler_enabled 0          ; Host event handler is
disabled
    flap_detection_enabled 0         ; Flap detection is disabled
    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
    check_command          check-host-alive ; Default host check is
a quick ping
    max_check_attempts     10
    notification_interval   120
    notification_period     24x7
    notification_options    d,u,r
    register                0         ; DONT REGISTER THIS DEFINITION
- ITS NOT A REAL HOST, JUST A TEMPLATE!
}

define host{
    name                windows-server ; Template for Windows Servers
    use                 generic-host    ; Use above template for
defaults
    contact_groups      windows-admins ; Who to notify
    register            0
}

define host{
    name                linux-server    ; Template for Linux Servers
    use                 generic-host
    contact_groups      linux-admins
    register            0
```

```

    }

define host{
    name                switch-template  ; Template for Managed Switches
    use                  generic-host
    contact_groups       switch-admins
    register             0
}

define host{
    name                printer-template  ; Template for Printers
    use                  generic-host
    notification_period  workhours
    contact_groups       printer-admins
    register             0
}

define host{
    name                misc-device      ; Template for Misc Network Devices
    use                  generic-host
    contact_groups       misc-admins
    register             0
}

# ## End of file ##

```

servicesTemplates.cfg

```

##### Service Definition Templates #####
##### Generic service definition template #####
define service{
    name                generic-service
    active_checks_enabled 1      ; Active service checks are enabled
    passive_checks_enabled 0      ; Passive service checks are
enabled/disabled
    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      0      ; Default is to NOT check service
'freshness'
    notifications_enabled 1      ; Service notifications are enabled
    event_handler_enabled 0      ; Service event handler is disabled
    flap_detection_enabled 0      ; Flap detection is disabled

```

```

    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
    check_period           24x7
    max_check_attempts     3
    normal_check_interval  3
    retry_check_interval   1
    notification_interval  120
    notification_period    24x7
    notification_options   w,u,c,r
    register               0
}

```

General Service Definition Templates

```
define service{
```

```

    name                ping-service
    use                 generic-service
    service_description PING
    is_volatile         0
    check_command       check_ping!100.0,20%!500.0,60%
    register            0
}

```

```
define service{
```

```

    name                dns-service
    use                 generic-service
    service_description DNS
    is_volatile         0
    check_command       rmc_check_dns!www.google.co.uk!1!2
    register            0
}

```

```
define service{
```

```

    name                proxy-service
    use                 generic-service
    service_description PROXY
    is_volatile         0
    check_command       check_squid!8080!http://www.google.co.uk
    register            0
}

```

```

define service{
    name                http-service
    use                  generic-service
    service_description  HTTP
    is_volatile          0
    check_command         check_http
    register              0
}

##### Printer Checks #####
define service{
    name                printer-status
    use                  generic-service
    service_description  Printer Status
    is_volatile          0
    check_period          workhours
    max_check_attempts    4
    normal_check_interval 5
    retry_check_interval  1
    contact_groups        printer-admins
    notification_interval 960
    notification_period    workhours
    check_command          check_hpjd
    register                0
}

# ## End of File ##

```

checkcommands.cfg (to be placed in /usr/local/nagios/etc)

```

#####
# checkcommands.cfg - Nagios configuration file for local user changes
#####
##### Adapted DNS server check #####
define command{
    command_name    rmc_check_dns
    command_line     $USER1$/check_dns -s $HOSTADDRESS$ -H $ARG1$ -w $ARG2$
-c $ARG3$
}

##### Check JetDirect Status #####
define command{
    command_name    check_hpjd
    command_line     $USER1$/check_hpjd -H $HOSTADDRESS$

```

```

    }

##### Ping-based checks #####
define command{
    command_name      check-host-alive
    command_line      $USER1$/check_ping -H $HOSTADDRESS$ -w 99,99% -c
100,100% -p 1
    }

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

##### HTTP-based checks #####
define command{
    command_name      check_http
    command_line      $USER1$/check_http -H $HOSTADDRESS$
    }

define command{
    command_name      check_squid
    command_line      $USER1$/check_http -H $HOSTADDRESS$ -p 8080 -u
http://www.google.co.uk
    }

# ## End of file ##

```

hostgroups.cfg

```

##### Hostgroup Definitions #####
## This file groups the hosts to allow easy reference to similar hosts later
# 'windows-servers' host group definition
define hostgroup{
    hostgroup_name    windows-servers
    alias             Windows Servers
    members           WINDOW-BOX
    }

# 'linux-boxes' host group definition
define hostgroup{
    hostgroup_name    linux-boxes
    alias             Linux Servers
    members           LINUX-BOX,GATEWAY,NAGIOS

```

```

    }

# 'printers' host group definition
define hostgroup{
    hostgroup_name    printers
    alias             Printers
    members           LASER-PRINTER
}

# 'switches' host group definition
define hostgroup{
    hostgroup_name    switches
    alias             Switches
    members           switch1,switch2,switch3
}

# 'misc-devices' host group definition
define hostgroup{
    hostgroup_name    misc-devices
    alias             Misc Devices
    members           UPS,ROUTER,google
}

# ## End of File ##

```

hostsServers.cfg

```

##### Server Host Definitions #####
# 'WINDOW-BOX' host definition
define host{
    use                windows-server
    host_name          WINDOW-BOX
    alias              WINDOW-BOX (Windows Server)
    address            192.168.0.3
    parents            switch1
}

# 'LINUX-BOX' host definition
define host{
    use                linux-server
    host_name          LINUX-BOX
    alias              LINUX-BOX (Linux Server)
    address            192.168.0.2
    parents            switch1
}

```

```

    }

# 'GATEWAY' host definition
define host{
    use                linux-server
    host_name          GATEWAY
    alias              GATEWAY (linux Server - Proxy)
    address            192.168.0.1
    parents            switch1
}

# 'NAGIOS' host definition
define host{
    use                linux-server
    host_name          NAGIOS
    alias              NAGIOS (Linux Server - Nagios)
    address            192.168.0.5
    parents            switch2
}

# ## End of File ##

```

hostsSwitches.cfg

```

##### Switch Host Definitions #####
# 'switch1' host definition
define host{
    use                switch-template
    host_name          switch1
    alias              Switch #1
    address            192.168.1.1
    parents            switch2
}

# 'switch2' host definition
define host{
    use                switch-template
    host_name          switch2
    alias              Switch #2
    address            192.168.1.2
}

# 'switch3' host definition
define host{

```

```
use                switch-template
host_name          switch3
alias              Switch #3
address            192.168.1.3
parents            switch2
}
```

```
# ## End of File ##
```

hostsPrinters.cfg

```
##### Printers Host Definitions #####
# 'LASER-PRINTER' host definition
define host{
    use                printer-template
    host_name          LASER-PRINTER
    alias              LASER-PRINTER (HP LaserJet)
    address            192.168.0.4
    parents            switch2
}
```

```
# ## End of File ##
```

hostsMisc.cfg

```
##### Misc Host Definitions #####
# 'UPS' host definition
define host{
    use                misc-device
    host_name          UPS
    alias              UPS (APC SNMP Management Card)
    address            192.168.0.6
    parents            switch1
}
```

```
# 'ROUTER' host definition
```

```
define host{
    use                misc-device
    host_name          ROUTER
    alias              ROUTER
    address            10.0.0.1
    parents            GATEWAY
}
```

```
# 'Google UK' host definition
```

```
define host{
```

```

    use                misc-device
    host_name          google
    alias              Google UK
    address             www.google.co.uk ; DNS MUST WORK!!!
    parents             ROUTER
}

```

```
# ## End of File ##
```

- Now that all the hosts are defined, it is necessary to define the services to be checked:

servicesServers.cfg

```

##### Server Service Definitions #####
# Ping all servers
define service{
    hostgroup_name      windows-servers
    use                  ping-service
    contact_groups      windows-admins
}

define service{
    hostgroup_name      linux-boxes
    use                  ping-service
    contact_groups      linux-admins
}

# Check DNS
define service{
    host_name           LINUX-BOX
    use                  dns-service
    contact_groups      linux-admins
}

# Check Web Server
define service{
    host_name           LINUX-BOX
    use                  http-service
    contact_groups      linux-admins
}

# Check Proxy Server
define service{
    host_name           GATEWAY
    use                  proxy-service
}

```

```
        contact_groups      linux-admins
    }
```

```
# ## End of file ##
```

servicesSwitches.cfg

```
##### Switch Service Definitions #####
```

```
# We are not yet monitoring any specific services for now (just ping them)
```

```
# Ping all switches
```

```
define service{
    hostgroup_name      switches
    use                  ping-service
    contact_groups      switch-admins
}
```

```
# ## End of File ##
```

servicesPrinters.cfg

```
##### Printer Service Defintions #####
```

```
# Printers using 'proper' JetDirect cards
```

```
define service{
    hostgroup_name      printers
    use                  printer-status
}
```

```
# ## End of File ##
```

servicesMisc.cfg

```
##### Misc Devices Service Definitions #####
```

```
# Ping devices — all we will do for now ;-)
```

```
define service{
    hostgroup_name      misc-devices
    use                  ping-service
    contact_groups      misc-admins
}
```

```
# ## End of File ##
```

extinfo.cfg (you will need to get the necessary icon packs from nagiosexchange.org)

```
##### Extended Host and Service Information #####
```

```
define hostextinfo{
    hostgroup_name      linux-boxes
    notes                Debian GNU/Linux servers
    icon_image           rack_linux.png
    icon_image_alt       Debian GNU/Linux
    vrmf_image           rack_linux.png
}
```

```
        statusmap_image  rack_linux.png
    }

define hostextinfo{
    hostgroup_name      windows-servers
    notes                Windows servers
    icon_image          rack_windows.png
    icon_image_alt      Windows Server 2003
    vrml_image          rack_windows.png
    statusmap_image     rack_windows.png
}

define hostextinfo{
    hostgroup_name      printers
    notes                Network Printers
    icon_image          hp-printer40.png
    icon_image_alt      Network Printer
    vrml_image          hp-printer40.png
    statusmap_image     hp-printer40.gd2
}

define hostextinfo{
    hostgroup_name      switches
    notes                Switches
    icon_image          switch.png
    icon_image_alt      Switch
    vrml_image          switch.png
    statusmap_image     switch.png
}

define hostextinfo{
    hostgroup_name      misc-devices
    notes                Misc. Devices
    icon_image          black_box.png
    icon_image_alt      Misc. device
    vrml_image          black_box.png
    statusmap_image     black_box.png
}

# ## End of File ##
```

Checking that the config files make sense and starting Nagios

- Before loading the Nagios daemon, a sanity check should be performed on the configuration files:

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

- If there are any error messages, read the output to see what is wrong – it's usually typos ;-)
- If this sanity check works, (re)start the daemon using the command:

```
# /etc/init.d/nagios restart
```

- Now go to <http://servername/nagios> and log in as 'nagiosadmin' to see your handywork

Extending the Configuration

Monitoring Switches

For this section, I will be assuming that you are using managed HP Procurve switches – this is what I have experience of. The checks will be performed using SNMP – Simple Network Management Protocol which is basically a list of information regarding the workings of the switch – and, since a lot of vendors use similar RFCs and reference IDs (oids) they will probably work with kit such as Cisco and 3COM.

I will refer back to the example network above:

- Switch #1 – HP Procurve 2828
- Switch #2 – HP Procurve 4104gl (single PSU)
- Switch #3 – HP Procurve 2650

The reason I have chosen these three switches is because of the slight variations in their hardware (e.g. the 2828 has more memory, the 4104gl has the option of a redundant PSU, etc.). If you have different switches, it is possible to inspect the contents of the oids using snmpwalk and a little trial and error with different threshold values.

Warning! More configuration file changes ahead:

- When adding the lines of config below DO NOT include the ellipses (...) - these simply indicate that there will be config lines below!
- First we need to add some custom commands to the end of checkcommands.cfg file:

```
...
##### Checks for HP Procurve Switches
define command{
    command_name    rmc_check_hpmemoryfree
    command_line    $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.5.1.1.2.1.1.1.6.1 -t 5 -w $ARG2$ -c $ARG3$ -u bytes -l
free
    }
define command{
    command_name    rmc_check_hp_cpu
    command_line    $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.5.1.9.6.1.0 -t 5 -w $ARG2$ -c $ARG3$ -u % -l "5min cpu"
    }
define command{
    command_name    rmc_check_hpfan
    command_line    $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.1.2.6.1.4.1 -w $ARG2$ -c $ARG3$ -l 'Fan status'
    }
define command{
```

```

        command_name    rmc_check_hppower
        command_line     $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.1.2.6.1.4.2 -w $ARG2$ -c $ARG3$ -l 'Power Supply status'
    }
define command{
        command_name    rmc_check_hptemp
        command_line     $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.1.2.6.1.4.4 -w $ARG2$ -c $ARG3$ -l 'Temprature status'
    }
# For some reason the default slot for the 410x power supply is slot 2 :?
define command{
        command_name    rmc_check_hppower_4100
        command_line     $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o
.1.3.6.1.4.1.11.2.14.11.1.2.6.1.4.3 -w $ARG2$ -c $ARG3$ -l 'Power Supply status'
    }

```

- We also need to make some changes at the end of the servicesTemplate.cfg file (bviously replace 'MyReadCommunity' with your SNMP Read community string):

```

...
##### Switch Service Definition Templates #####
define service{
        name                        switch-memory2800-service
        use                        generic-service
        service_description        MEMORY
        is_volatile                0
        contact_groups             switch-admins
        check_command
rmc_check_hpmemoryfree!MyReadCommunity!19000000:10000000!10000000:0
        register                  0
    }

define service{
        name                        switch-memory-service
        use                        generic-service
        service_description        MEMORY
        is_volatile                0
        contact_groups             switch-admins
        check_command
rmc_check_hpmemoryfree!MyReadCommunity!2000:19000000!1000:19000000
        register                  0
    }

define service{
        name                        switch-CPU-service
        use                        generic-service

```

```

    service_description      CPU
    is_volatile              0
    contact_groups           switch-admins
    check_command
rmc_check_hp_cpu!MyReadCommunity$!95:90!100:95
    register                 0
}

define service{
    name                    switch-PSU-service
    use                    generic-service
    service_description     PSU
    is_volatile             0
    contact_groups          switch-admins
    check_command           rmc_check_hppower!MyReadCommunity!4!3:5
    register                0
}

define service{
    name                    switch-PSU4100-service
    use                    generic-service
    service_description     PSU
    is_volatile             0
    contact_groups          switch-admins
    check_command           rmc_check_hppower_4100!MyReadCommunity!4!3:5
    register                0
}

define service{
    name                    switch-temp-service
    use                    generic-service
    service_description     TEMP
    is_volatile             0
    contact_groups          switch-admins
    check_command           rmc_check_hptemp!MyReadCommunity!4!3:5
    register                0
}

define service{
    name                    switch-fan-service
    use                    generic-service
    service_description     FAN

```

```

is_volatile          0
contact_groups       switch-admins
check_command        rmc_check_hpfan!MyReadCommunity!4!3:5
register              0
}

```

- Now we need to assign the services to the switches in servicesSwitches.cfg

```

...
# Check memory
define service{
    host_name          switch1
    use                 switch-memory2800-service
}

define service{
    host_name          switch2,switch3
    use                 switch-memory-service
}

# Check fan
define service{
    hostgroup_name      switches
    use                 switch-fan-service
}

# Check CPU
define service{
    hostgroup_name      switches
    use                 switch-CPU-service
}

# Check PSU
define service{
    host_name          switch1,switch3
    use                 switch-PSU-service
}

define service{
    host_name          switch2
    use                 switch-PSU4100-service
}

# Check temperature

```

```
# 4100-series switches do not appear to do temperature check
define service{
    host_name          switch1,switch3
    use                 switch-temp-service
}
```

A more advanced UPS check

Thanks to a friendly programmer on Nagios Exchange, I found an UPS test that will query the SNMP details from the management card and tell me if the temperature gets silly or the batteries die. You will need to download this from www.nagiosexchange.org :-)

- Edit the checkcommands.cfg file:

```
...
##### Check APC UPS Status #####
define command{
    command_name      rmc_check_snmp_apcups
    command_line      usr/lib/nagios/plugins/check_snmp_apcups -H
$HOSTADDRESS$ -C $ARG1$
}
```

- Edit the servicesTemplate.cfg file:

```
...
# UPS Check
define service{
    name                UPS-check-service
    use                 generic-service
    service_description UPS
    is_volatile         0
    contact_groups      misc-admins
    check_command        rmc_check_snmp_apcups!MyReadCommunity
    register            0
}
```

- Edit the servicesMisc.cfg

```
...
# Check UPS Status
define service{
    host_name          UPS
    use                 UPS-check-service
}
```

What about my Windows Servers?

We all know how clever network devices and Linux boxes are when it comes to giving out information on status but Windows tends to be lacking in this area. To get around this there is a plugin system, called NagiosPluginsNT, written in C# that fires useful information at the Nagios box using a system called NRPE. Both of these can be downloaded from Nagios Exchange. The instructions below are in two parts:

1. On the Nagios server

- Install the libssl-dev package (this is required to compile the plugins)

```
# apt-get install libssl-dev
```

- Download the check_nrpe plugin to your Nagios server from the Nagios download page and save in your home folder
- Unpack the installation package

```
# tar xzvf nrpe-version.tar.gz
```

- Change to the newly created directory and run the configure script

```
# cd nrpe-version
```

```
# ./configure
```

- Check that there are no errors and run the make command

```
# make all
```

- Copy the freshly made plugin to the Nagios plugin directory

```
# cp ./src/check_nrpe /usr/local/nagios/libexec
```

- Define the check in the file /usr/local/nagios/checkcommands.cfg by adding the following at the end:

```
...
# NRPE check
define command{
    command_name      check_nrpe
    command_line      $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
```

- Edit /usr/local/nagios/etc/conf.d/servicesTemplates.cfg to define the NRPE service checks

```
...
##### NRPE Service Checks #####
define service{
    name                nrpe-memory-service
    use                 generic-service
    service_description Memory Usage
    is_volatile         0
    contact_groups      windows-admins
    check_command        check_nrpe!check_mem
    register            0
}

# CPU Utilisation Check
define service{
    name                nrpe-cpu-service
    use                 generic-service
    service_description CPU Utilisation
    is_volatile         0
    contact_groups      windows-admins
    check_command        check_nrpe!check_cpu
```

```

        register          0
    }

# Free Disk Space (C:) Check
define service{
    name                nrpe-diskC-service
    use                  generic-service
    service_description  Disk Usage C:
    is_volatile          0
    contact_groups       windows-admins
    check_command         check_nrpe!check_disk_c
    register             0
}

# Free Disk Space (D:) Check
define service{
    name                nrpe-diskD-service
    use                  generic-service
    service_description  Disk Usage D:
    is_volatile          0
    contact_groups       windows-admins
    check_command         check_nrpe!check_disk_d
    register             0
}

```

- Edit /usr/local/nagios/etc/conf.d/servicesServers.cfg to assign the service checks

```

...
# NRPE Checks
define service{
    host_name    WINDOW-BOX
    use          nrpe-memory-service
}

define service{
    host_name    WINDOW-BOX
    use          nrpe-cpu-service
}

define service{
    host_name    WINDOW-BOX
    use          nrpe-diskC-service
}

```

```
define service{
    host_name    WINDOW-BOX
    use          nrpe-diskD-service
}
```

2. On the Windows server

- Ensure that .NET version 2 is installed on the server
- Download the NRPE_NT daemon from <http://www.miwidv.com/nrpent/>
- Extract the bin folder to c:\nrpe_nt
- Run the command to install NRPE_NT as a service

```
# c:\nrpe_nt\nrpe_nt -i
```

- Download the latest version of the plugins from <http://nagiospluginsnt.getproactivenow.com/download/releases/> - make sure you get the 'bin' version rather than the 'src' version
- Extract the plugin package to c:\nrpe_nt\plugins
- Edit c:\nrpe_nt\nrpe.cfg so that it includes the checks that you wish to perform

```
...
# Check disk space of C: and D: - warning at 90% full and critical at 95% full
command[check_disk_c]=C:\NRPE_NT\Plugins\diskspace_nrpe_nt.exe C: 90 95
command[check_disk_d]=C:\NRPE_NT\Plugins\diskspace_nrpe_nt.exe D: 90 95
# Check CPU utilisation - warning at 70% and critical at 85%
command[check_cpu]=C:\NRPE_NT\NagiosPluginsNT\check_cpu.exe -U % -w 70 -c 85
# Check memory usage - warning at 70% and critical at 85%
command[check_mem]=C:\NRPE_NT\NagiosPluginsNT\check_mem.exe -U % -w 70 -c 85
```

- Restart the nrpe_nt service (Nagios Remote Plugin Executor for NT/W2K in Services management console)