L2 Nagios (W4-6)

OPS635-lab-nagios —nagios-event-handlers-nrpe — assets.nagios.com/eventhandlers.html

nagioscore/4/en/toc.html (DOC) — HowTo Nagios

continuous monitoring network hosts and services

- periodically performs checks - report status/escalate if wrong

Installation and web-console

yum install epel-release / nagios / nagios-plugins / nagios-plugins-* --skip-broken (include httpd) / nrpe

- disable selinux
- enable and start: nagios/httpd/postfix
  - chkconfig httpd on (apache)
  - nagios -v /etc/nagios/nagios.cfg
  - service nagios restart
  - ; — is used for comments

Nagios web-console

▼ firewalld port 80 config

```
firewall-cmd --add-service http/https --zone=public
firewall-cmd --runtime-to-permanent
firewall-cmd --zone=public --add-port=80/tcp --permanent
service firewalld reload
```

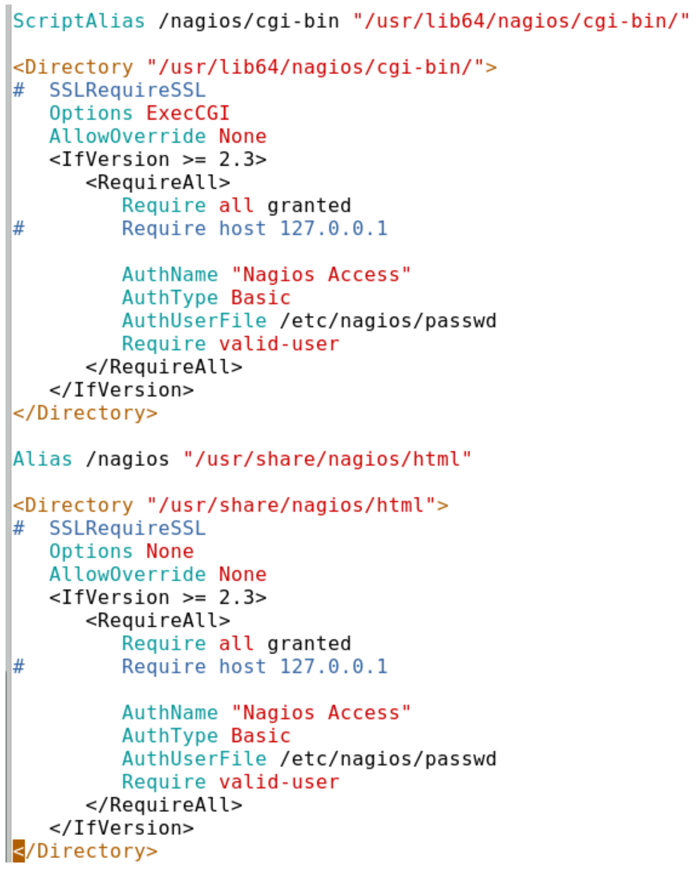
http://<host>/nagios/

- usr/passd: nagiosadmin

`/etc/httpd/conf.d/nagios.conf`
restrict access based on IP (require ip), where to log

- htpasswd /etc/nagios/passwd nagiosadmin
  New password:

▼ # AuthUserFile (/etc/ngaios/passwd), RequireHost (restrict access)



```
ScriptAlias /nagios/cgi-bin "/usr/lib64/nagios/cgi-bin/"

<Directory "/usr/lib64/nagios/cgi-bin/">
#   SSLRequireSSL
   Options ExecCGI
   AllowOverride None
   <IfVersion >= 2.3>
      <RequireAll>
         Require all granted
#        Require host 127.0.0.1

         AuthName "Nagios Access"
         AuthType Basic
         AuthUserFile /etc/nagios/passwd
         Require valid-user
      </RequireAll>
   </IfVersion>
</Directory>

Alias /nagios "/usr/share/nagios/html"

<Directory "/usr/share/nagios/html">
#   SSLRequireSSL
   Options None
   AllowOverride None
   <IfVersion >= 2.3>
      <RequireAll>
         Require all granted
#        Require host 127.0.0.1

         AuthName "Nagios Access"
         AuthType Basic
         AuthUserFile /etc/nagios/passwd
         Require valid-user
      </RequireAll>
   </IfVersion>
</Directory>
```

▼

`/etc/nagios/cgi.cfg` — use_authentication, path to config. files

`/etc/nagios/nagios.cfg` determines which other files to include

- cfg_file=<absolute path>
    - cfg_file=/etc/nagios/lab.cfg ;newly created cfg file for this lab
    - cfg_dir=/etc/nagios/servers ;for directories
- put everything about one machine in one file, or, put each type of definition in separate files

Interface: Scheduling Queue — which checks are going to be performed next, cancel and/or reschedule

Nagios configuration

Defining the basic object to work with (object-definitions)

hosts

define host{
 use <template name> - default parameters to inherit
 host_name <name> - actual hostname
 address <ip address> - client IP
 }

▼ define host{

use	linux-server
host_name	client
alias	client
address	192.168.1.100
max_check_attempts	5
check_period	24×7
notification_interval	30
notification_period	24×7
contacts	fajton

}

alias – a more human readable name
check_interval – how often to check if the host is up
retry_interval – how often to re-check if the primary check failed
max_check_attempts – how many times the check can fail before reporting it, uses notifications
check_command – the command used to determine host state, by default (check_host_alive) pings the host

notification_options, contact_groups or contacts

▼ More examples

```
# Generic host definitions
define host{
        name                           generic-host    ; Generic 
template name
        notifications_enabled          1               ; Host no
tifications are enabled
        event_handler_enabled          1               ; Host ev
ent handler is enabled
        flap_detection_enabled         1               ; Flap de
tection is enabled
        process_perf_data              1               ; Process
performance data
        retain_status_information       1               ; Retain
status information across program restarts
        retain_nonstatus_information    1               ; Retain
non-status information across program restarts
        register                        0               ; DONT RE
GISTER - NOT A REAL HOST, JUST A TEMPLATE!
        }

# This creates a generic template that any host can use.
# Notifies never, checks 15 times before showing critical on CGI
interface,

define host{
        name                    basic-host
        use                     generic-host
        check_command           check-host-alive
        max_check_attempts      15
        notification_interval   0
        notification_period     none
        notification_options    n
        register                0
        }

# This creates a generic host that your routers can use
# monitors host(s) 24x7, notifies on down and recovery, checks 15
times before going critical,
# notifies the contact_group every 30 minutes

define host{
        name                    your-routers-host
        use                     generic-host
        check_command           check-host-alive
        max_check_attempts      15
        notification_interval   30
        notification_period     24x7
        notification_options    d,r
        register                0
        }
```

```
define host{
        use                         basic-host
        host_name                   mymachine1
        alias                       mymachine1
        address                     192.168.100.101
        contact_groups              einsteins
#       notification_options        d,r  #overrides the basic-host op
tion
        }

define host{
        use                         your-routers-host
        host_name                   router1
        alias                       router1
        address                     192.168.100.100
        contact_groups              einsteins
        }
```

services

define service{
 use <template name> - the template to inherit default settings from
 host_name <name> - the machine to perform the check on
 service_description <NAME> - a brief identifier for the check
 check_command <command name> - the name of the command to perform
 }

▼ define service{

```
use                    generic-service
host_name              client
service_description     SSH
check_command          check_ssh
notifications_enabled 0
}
```

check_interval – how often to perform this check, check if the service is up
retry_interval – how often to re-check if the initial check failed
max_check_attempts – how many times the check can fail before reporting it, like hosts, uses notifications

notification_options, contact_groups or contacts

▼ Examples, nested templates, last one overrides the parents

```
# Generic service definition template
define service{
        name                            generic-service ; Generic
service name
        active_checks_enabled           1               ; Active
service checks are enabled
        passive_checks_enabled          1               ; Passive
service checks are enabled/accepted
        parallelize_check               1               ; Active
service checks should be parallelized
        obsess_over_service             1               ; We shou
ld 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           1               ; Service
event handler is enabled
        flap_detection_enabled          1               ; Flap de
tection is enabled
        process_perf_data               1               ; Process
performance data
        retain_status_information       1               ; Retain
status information across program restarts
        retain_nonstatus_information    1               ; Retain
non-status information across program restarts
        register                        0               ; DONT RE
GISTER NOT A REAL SERVICE, JUST A TEMPLATE!
        }

# Generic for all services
define service{
        use                             generic-service
        name                            basic-service
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              15
        normal_check_interval           10
        retry_check_interval            2
        notification_interval           0
        notification_period             none
        register                        0
        }

define service{
        use                             basic-service
        name                            ping-service
        notification_options            n
        check_command                   check_ping!1000.0,20%!200
0.0,60%
        register                        0
        }

define service{
        use                             ping-service
        service_description             PING
        contact_groups                  einsteins
        hostgroup_name                  basic-clients,your-router
s
#       host_name                       one_client
        }


# SMTP - ensure SMTP services are available.
define service{
        use                             basic-service
        name                            smtp-service
        service_description             SMTP
        notification_interval           15
        contact_groups                  einsteins
        notification_options            c,r
        notification_period             24x7
        check_command                   check_smtp
        register                        0
        }
```

```
define service{
        use                             smtp-service
        hostgroup_name                  smtp-servers
#       host_name                       one_client  #would have t
o be hosts defined in hosts.cfg
        }
```

templates

same monitoring behaviors, no need to add definitions every time

- can then be overridden in each individual host, service, etc.

add the parameter 'register 0' inside a definition — tells Nagios not to monitor this entity

add the 'use' parameter to identify a template to inherit values from

pre-written templates: `/etc/nagios/objects/templates.cfg`

timeperiods

when checks should be run, or notifications sent (or when not)

- don't run checks at a time when something is not in use or during scheduled down-time

▼ examples: `/etc/nagios/objects/timeperiods.cfg`

```
# '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
define timeperiod{
        timeperiod_name workhours
        alias           "Normal" Working Hours
        monday          08:00-17:00
        tuesday         08:00-17:00
        wednesday       08:00-17:00
        thursday        08:00-17:00
        friday          08:00-17:00
        }

# 'nonworkhours' timeperiod definition
define timeperiod{
        timeperiod_name nonworkhours
        alias           Non-Work Hours
        sunday          00:00-24:00
        monday          00:00-09:00,17:00-24:00
        tuesday         00:00-09:00,17:00-24:00
        wednesday       00:00-09:00,17:00-24:00
        thursday        00:00-09:00,17:00-24:00
        friday          00:00-09:00,17:00-24:00
        saturday        00:00-24:00
        }

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

define timeperiod{
 timeperiod_name <name>
 <period definition> - One or more.
 }

period definition, time range is: day hh:mm-hh:mm — monday 09:00-17:00 #(every monday from 9AM to 5PM)

exclude <name> — inside a timeperiod, specify the name of another timeperiod to exclude (statutory holidays)

Add timeperiods (name) to host or service definitions:

- check_period <name> - when to perform checks

- notification_period <name> - when to send notifications that something is wrong

commands / custom plugins

scripts used by nagios to check the state of host/services

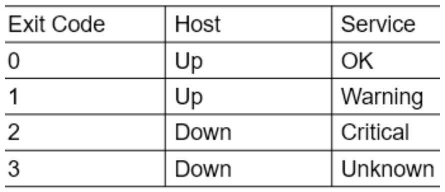
Installed as nagios-plugins-*

/usr/lib64/nagios/plugins

- <full_plugin_path> --help — to get help on a plugin

command_name — check_command in host/service definition

command_line - might include macros (variables), values to indicate what result should be a warning (-w), and what is critical (-c)



Exit Code	Host	Service
0	Up	OK
1	Up	Warning
2	Down	Critical
3	Down	Unknown

define command{
 command_name <name>
 command_line <call an actual executable>
 }

▼ Example

```
# 'check_smtp' command
definition
define command{
        command_name
check_smtp
        command_line
$USER1$/check_smtp -H
$HOSTADDRESS$
        }
```

Standard Macros in Nagios

$MAXHOSTATTEMPTS$ - nr of failures that can occur before a host check goes into hard fail state

$HOSTADDRESS$ - the IP address of the host being tracked
$HOSTNAME$ - the name from that host definition
$HOSTSTATE$ and $HOSTSTATEID$, indicate the current state of that host, up or down, 0 or 2?
$SERVICESTATE$ and $SERVICESTATEID$ are
the string or numeric states of that service the last time it was checked, any of okay, warn, crit, unknown, or for the ID 0, 1, 2, or 3
$SERVICEATTEMPTS$ - record of how many times this particular service check has failed.
$MAXSERVICEATTEMPTS$ - how many times

this service check can fail before it goes into a hard fail state
$USER1$, which is the first user defined variable, it defaults to the path to the standard nagios plugins directory, /usr/lib64/nagios/plugins
$ARGn$ with a number is how we will pass positional command-
line arguments in a host or service definition. We can call a command,
give it some positional arguments,
and they get passed into that command which then passes the macro on to the actual executable

Passing arguments to a command in a host or service definition

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

- check_ping is the actual command
- 100.0,20% is the first argument ($ARG1$)
- 500.0,60% is the second ($ARG2$)

above arguments can get used in the command_line paremeter in the command definition
command_line   /$USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$

Event handler - special type of plugin, commands that can be run automatically when certain conditions are met

- instruct nagios to try simple fixes before sending a notification
- *event_handler* parameter in a host/service definition
- will get called:
    - when a host or service switches into a soft fail state
    - the first time a host or service goes into a hard fail state
        - ***max_check_attempts*** n, the service will go into hard failed state after n attempts
    - when a host or service goes back into an okay state (recovers from a failed state)
- give admin privileges to nagios account to run commands with sudo
    - getsebool -a | grep nagios
    - `setsebool -P nagios_run_sudo 1`
        - disable selinux /etc/selinux/config, setenforce 0
    - vim /etc/sudoers — %nagios ALL=(ALL) NOPASSWD: ALL

Send notifications

Notifications

After a host/service has remained in a failed state for more than *max_check_attempts* and has moved into a hard state

enable_notifications=1 (/etc/nagios/nagios.cfg)

For hosts and services:

notifications_enabled 1
notifications_period
notifications_options — defined in contact definition

Host:	Service:	Both:
d – down	w – warning	r – recovery - things going back to ok state
u – unreachable	u – unknown	n – none - don't send notifications
s – scheduled downtime starts or stops	c – critical	f – flapping starts or stops

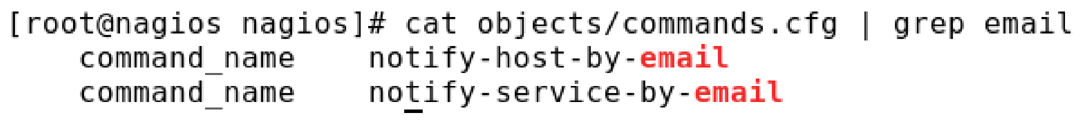
flapping - a host or service is rapidly switching between states, instead of sending d multiple times

Contacts and ContactGroups

determines who to send a notification to (install postfix on nagios vm)

```
define contact{
  contact_name <name used in contact_groups>
  service_notification_period <timeperiod>
  service_notification_options <notification options>
  host_notification_period <timeperiod>
  host_notification_options <notification options>
  email <email address>

  host_notifications_enabled 1
  service_notifications_enabled    1
  host_notification_commands      notify-host-by-email
  service_notification_commands notify-service-by-email
  }
```



```
[root@nagios nagios]# cat objects/commands.cfg | grep email
        command_name    notify-host-by-email
        command_name    notify-service-by-email
```

notification_interval — to determine how often notifications should be re-sent

define contactgroup{
 contactgroup_name <name>
 alias <human readable name>
 members <contacts>
 }

▼ Examples

```
#/etc/nagios/objects/contacts.cfg

# service_notification_options are w,u,c,r,f,n
# w=warning u=unknown c=critical r=recovery f=flapping n=none
# host_notification_options d,u,r,f,n
# d=down u=unreachable r=recovery f=flapping n=none

define contact{
        contact_name                    me
        alias                           me
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    c,r
        host_notification_options       d,r
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                           me@myemailaddress.whateve
r
        }

define contact{
        contact_name                    you
        alias                           you
        service_notification_period     workhours
        host_notification_period        workhours
        service_notification_options    c,r
        host_notification_options       d,r
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                           you@youremailaddress.what
ever
        }
```

```
# 'einsteins' contact group definitions
define contactgroup{
        contactgroup_name       einsteins
        alias                   einsteins
        members                 me,you
        }
```

Escalations

define serviceescalation{
  host_name <from host definition>
  service_description <from service definition>
  first_notification <when to start escalating>
  last_notification <when to stop>
  notification_interval <how often to send notifications>
  contacts|contact_groups <who to notify>
  }

define hostescalation{
  host_name <from host definition>
  first_notification
  last_notification
  notification_interval
  contacts|contact_groups
  }

first_notification — when to start sending the escalated notifications. We won't escalate on first notification sent, but maybe we will on the 2nd, or 3rd, or 10th.
last_notification — at what point do we stop sending this escalated notification (0 do not stop)

escalation_period <time period>
escalation_options <notification options> ;exclude flapping state or a warning state

Monitor remote machines with NRPE

NRPE (Nagios remote plugin executor) - execute plugins available on the remote machine

Client configuration

install nrpe and nagios-plugins in monitoring targets

- yum install epel-release/nrpe

- yum install nagios-plugins nagios-plugins-* --skip-broken
- nagios ALL=(ALL) NOPASSWD: /usr/lib64/nagios/plugins
    - or just: usermod -aG wheel nagios/nrpe
- enable/start nrpe, disable selinux

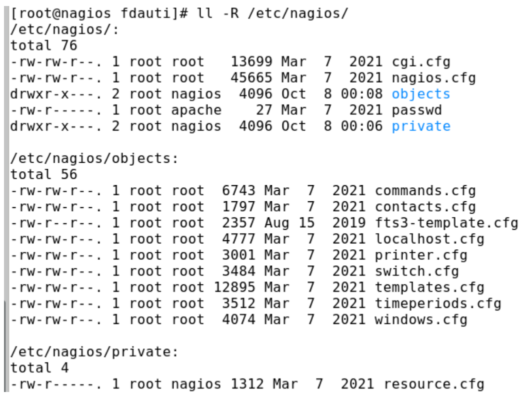
/etc/nagios/nrpe.cfg

- server_port=5666 (>1024, non privileged) the port where nagios server will connect to
    - firewall-cmd --zone=public --add-port=5666/tcp --permanent
- allowed_hosts=127.0.0.1,::1 (list of nagios servers that can execute plugins on this machine)
- command[<command name>]=<plugin path> [<options>]
    - command[check_users]=/usr/lib64/nagios/plugins/check_users $ARG1$
        - needs dont_blame_nrpe = 1

Nagios Server configuration (lab.cfg)

define command{
command_name check_nrpe
command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -c <command>
}

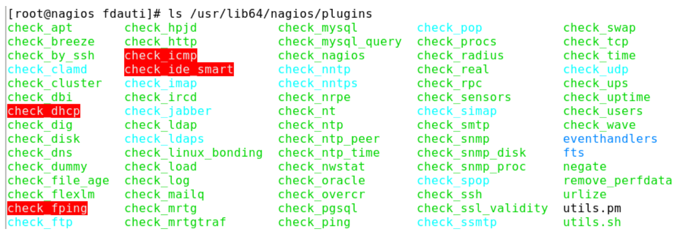
<command> — the command name set in the client's nrpe.cfg file



```
[root@nagios fdauti]# ll -R /etc/nagios/
/etc/nagios/:
total 76
-rw-rw-r--. 1 root root   13699 Mar  7  2021 cgi.cfg
-rw-rw-r--. 1 root root   45665 Mar  7  2021 nagios.cfg
drwxr-x---. 2 root nagios  4096 Oct  8 00:08 objects
-rw-r-----. 1 root apache    27 Mar  7  2021 passwd
drwxr-x---. 2 root nagios  4096 Oct  8 00:06 private

/etc/nagios/objects:
total 56
-rw-rw-r--. 1 root root  6743 Mar  7  2021 commands.cfg
-rw-rw-r--. 1 root root  1797 Mar  7  2021 contacts.cfg
-rw-r--r--. 1 root root  2357 Aug 15  2019 fts3-template.cfg
-rw-rw-r--. 1 root root  4777 Mar  7  2021 localhost.cfg
-rw-rw-r--. 1 root root  3001 Mar  7  2021 printer.cfg
-rw-rw-r--. 1 root root  3484 Mar  7  2021 switch.cfg
-rw-rw-r--. 1 root root 12895 Mar  7  2021 templates.cfg
-rw-rw-r--. 1 root root  3512 Mar  7  2021 timeperiods.cfg
-rw-rw-r--. 1 root root  4074 Mar  7  2021 windows.cfg

/etc/nagios/private:
total 4
-rw-r-----. 1 root nagios 1312 Mar  7  2021 resource.cfg
```



```
[root@nagios fdauti]# ls /usr/lib64/nagios/plugins
check_apt        check_hpjd           check_mysql        check_pop            check_swap
check_breeze     check_http           check_mysql_query  check_procs          check_tcp
check_by_ssh     check_icmp           check_nagios       check_radius         check_time
check_clamd      check_ide_smart      check_nntp         check_real           check_udp
check_cluster    check_imap           check_nntps        check_rpc            check_ups
check_dbi        check_ircd           check_nrpe         check_sensors        check_uptime
check_dhcp       check_jabber         check_nt           check_simap          check_users
check_dig        check_ldap           check_ntp          check_smtp           check_wave
check_disk       check_ldaps          check_ntp_peer     check_snmp           eventhandlers
check_dns        check_linux_bonding  check_ntp_time     check_snmp_disk      fts
check_dummy      check_load           check_nwstat       check_snmp_proc      negate
check_file_age   check_log            check_oracle       check_spop           remove_perfdata
check_flexlm     check_mailq          check_overcr       check_ssh            urlize
check_fping      check_mrtg           check_pgsql        check_ssl_validity   utils.pm
check_ftp        check_mrtgtraf       check_ping         check_ssmtp          utils.sh
```

Lab 2 Commands

touch /etc/nagios/lab.cfg

vim /usr/lib64/nagios/plugins/check_sshd

- chmod +x /usr/lib64/nagios/plugins/check_sshd

vim /usr/lib64/nagios/plugins/restart_sshd

- chmod +x /usr/lib64/nagios/plugins/restart_sshd

lab.cfg, from nagios server

nrpe.cfg from nagiosnrpe,

check_sshd plugin,

event handler (restart_sshd)

A2

Create Nagios VM

Use: NRPE, notifications, escalations, time periods, event handlers

- (Nagios remote plugin executor) - execute plugins available on the remote machine
- service groups, and host groups (check_ping)

Nagios (i.e. the nagios.cfg, all other cfg files it refers to)

- a2.cfg and nagios.cfg on server

plugins examples

- check_dns
- restart_dns

nrpe.cfg included on the other machines,

Changes made to service configuration

- my-zone.txt , rev-zone.txt

Firewall information

- firewall-cmd --list-all --zone=public