Monitoraggio con Nagios

Installazione NRPE Centos/Debian

Installazione panchetti NRP e plugins:

Debian:

`apt-get install nagios-nrpe-server nagios-plugins-basic`

Centos:

`yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm yum install nrpe nagios-plugins-users nagios-plugins-load nagios-plugins-swap nagios-plugins-disk nagios-plugins-procs`

Monitorare Tomcat

Lo script da copiare in /usr/lib/nagios/plugins/

check_tomcat.pl

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

# I'm a Spanish speaker, my English is basic but I try to write everything in
# English.
# Many translations have been made with automatic translators.
# If something is not meant pray excuse me.
# ¡VAMOS A ELLO! (LET'S GO!)

#-------------------------------------------------------------------------------
# Name:        check_tomcat.pl
#
# Author:      Daniel Dueñas
#
# Purpose:
# Plugin para chequeo de servidor tomcat para nagios
# Check tomcat server plugin for nagios
# This plugin uses the tomcat manager webapp, this app usually is located in the
# URL:
# http://tomcat-host-name:xxxx/manager
# which "tomcat-host-name" is the dns name or ip of the tomcat server and "xxxx"
# is the port number of the tomcat service (the tomcat port is 8080 by default)
#
# This plugin can check this items:
# 1- tomcat server status
# 2- tomcat server memory
# 3- tomcat server thread connectors
# 4- application status on tomcat server
#
#
# This plugin conforms to the Nagios Plugin Development Guidelines
# https://nagios-plugins.org/doc/guidelines.html
#
# Created:     18/02/2014
# Copyright:   (c) Daniel Dueñas 2014
# Licence:
#   This program is free software; you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation; either version 2 of the License, or
#   (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
```

```
#   You should have received a copy of the GNU General Public License
#   along with this program; if not, write to the Free Software
#   Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
#-------------------------------------------------------------------------------

import argparse, sys
import urllib2
import socket
import xml.etree.ElementTree as ET
from math import log

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#VARIABLE DEFINITIONS
#-------------------------------------------------------------------------------
version="2.1"   #Plugin version
status = { 'OK' : 0 , 'WARNING' : 1, 'CRITICAL' : 2 , 'UNKNOWN' : 3}
exit_status = 'OK'
output = ""
longoutput = ""
perfdata = ""
mensage = ""
plugin_description ='''Nagios plugin for check an apache tomcat server
'''
mode_help ='''Tomcat monitorizacion mode:
    status: The status of tomcat server
    mem:    Tomcat server used percentage memory status, warning and critical
            values. Requiered in percentage.
    thread: Tomcat connectors Threads used, warning and critical values requiered.
            The parameter connector is optional, if not exists, all connector were shown.
    app:    Application status in tomcat server, the name of the application
            must be defined with the parameter -n or --nameapp.
            This option check the status of java application running on tomcat server
'''
tree_xml=None
tomcat_version=None

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#FUNCTIONS
#-------------------------------------------------------------------------------
#Try if string is a float
def is_float_try(str):
    try:
        float(str)
        return True
    except ValueError:
        return False

#This function define the range of warning and critical values
#Return a range --> (min,max,True/False)
#if True, data must be in a range
#if False, data must be out of a range
def define_range(str):
    if is_float_try(str):
        range = (0,float(str),True)  # x -> in range(0,x)
    elif str.count(":")==1:
        splits = str.split(":")
        if is_float_try(splits[0]) and is_float_try(splits[1]):
            range=(float(splits[0]),float(splits[1]),True) # x:y  -> in range(x,y)
        elif is_float_try(splits[0]) and splits[1]=="":
            range=(float(splits[0]),float("inf"),True)  # x:  -> in range(x,infinite)
        elif splits[0]=="~" and is_float_try(splits[1]):
            range=(float("-inf"),float(splits[1]),True) # ~:x -> in range(-infinite,x)
        if splits[0][0]=="@" and is_float_try(splits[0].replace("@","")) and
```

```
is_float_try(splits[1]):
            range=(float(splits[0].replace("@","")),float(splits[1]),False) # @x:y -> out of
range(x,y)
    else:
        print "bad range definition in "+str
        sys.exit(1) #Error in range definition
    if range[0]<range[1]:
        return range   # OK
    else:
        print "Second value of range "+str+" is less than first value"
        exit(1)

#Critical and warning function resolve
#This function return exit_status: OK,WARNING,CRITICAL or UNKNOWN string
def define_status(value,warning,critical):
    warning_range = define_range(warning)   #Define warning range
    critical_range = define_range(critical) #Define critical range
    val=float(value)                        #The value
    exit_status="UNKNOWN"                   #status by default

    if args.verbosity:
        print "Value for test: "+str(value)
        print "Warning range (min:%s max:%s
in_range:%s)"%(str(warning_range[0]),str(warning_range[1]),str(warning_range[2]))
        print "Critical range (min:%s max:%s
in_range:%s)"%(str(critical_range[0]),str(critical_range[1]),str(critical_range[2]))
        print ""

    #value into the range range(x:y:True)
    if warning_range[2]==True and critical_range[2]==True:
        if (warning_range[1]>critical_range[1]) or (warning_range[0]<critical_range[0]):
            parser.print_usage()
            parser.exit(3,"ERRROR: critical range (%s) is greater than warning range(%s)\n"
% (critical, warning))
        if value<warning_range[0] or value>warning_range[1]:
            exit_status="WARNING"
            if value<critical_range[0] or value>critical_range[1]:
                exit_status="CRITICAL"
        else:
            exit_status="OK"

    #value out of range range(x:y:False)
    elif warning_range[2]==False and critical_range[2]==False:
        if (warning_range[1]<critical_range[1]) or (warning_range[0]>critical_range[0]):
            parser.print_usage()
            parser.exit(3,"ERRROR: critical range (%s) is greater than warning range(%s)\n"
% (critical, warning))
        if value>warning_range[0] and value<warning_range[1]:
            exit_status="WARNING"
            if value>critical_range[0] and value<critical_range[1]:
                exit_status="CRITICAL"
        else:
            exit_status="OK"
    #warning and critical ranges must be both in or out
    else:
        parser.print_usage()
        parser.exit(status[exit_status],'''
ERROR: Both critical and warning values must be in or out of the ranges:
       warning('''+warning+''') and critical ('''+critical+''')\n''')

    return exit_status

# convert human readable size function
def sizeof_fmt(num):
    # Human friendly size
```

```
    unit_list = zip(['bytes', 'kB', 'MB', 'GB', 'TB', 'PB'], [0, 0, 1, 2, 2, 2])
    if num > 1:
        exponent = min(int(log(num, 1024)), len(unit_list) - 1)
        quotient = float(num) / 1024**exponent
        unit, num_decimals = unit_list[exponent]
        format_string = '{0:.%sf} {1}' % (num_decimals)
        return format_string.format(quotient, unit)
    elif num == 0:
        return '0 bytes'
    elif num == 1:
        return '1 byte'
    elif num < 0:
        return 'negative number'
    else:
        return None

#open and read XML from tomcat manager/status?XML=true
def read_page_status_XML(host,port,url,user,password):

    url_status_xml = url+"/status?XML=true"
    page,error_page = read_page(host,port,url_status_xml,user,password)
    # End of Open manager status
    if(error_page):
        return page,error_page
    else:
    # Read xml
        try:
            root = ET.fromstring(page)
        #If can't parse XML
        except ET.ParseError as e:
            root="ERROR: I Can't understand the XML page. Error: %s" %(e)
            error_page=True
        #show XML tree
        if args.verbosity>1:
            print "XML tree:"
            print ET.dump(root)
            print ""
        return root,error_page

#Read a html manager page
def read_page(host,port,url,user,password):
    error=False
    url_tomcat = "http://"+host+":"+port+url
    if args.verbosity:
        print "connection url: %s\n"%(url_tomcat)

    try:
        password_mgr = urllib2.HTTPPasswordMgrWithDefaultRealm()
        password_mgr.add_password(None,url_tomcat,user,password)
        handler = urllib2.HTTPBasicAuthHandler(password_mgr)
        opener=urllib2.build_opener(handler)
        urllib2.install_opener(opener)
        req = urllib2.Request(url_tomcat)
        handle = urllib2.urlopen(req, None)
        # Store all page in a variable
        page = handle.read()
        # End of Open manager status
    except urllib2.HTTPError as e:
        if(e.code==401):
            page="ERROR: Unauthorized, your user not have permissions. %s" %(e)
        elif(e.code==403):
            page="ERROR: Forbidden, yours credentials are not correct. %s" %(e)
        else:
            page="ERROR: The server couldn\'t fulfill the request. %s" %(e)
        error=True
```

```
    except urllib2.URLError as e:
       page = 'ERROR: We failed to reach a server. Reason: %s' %(e.reason)
       error = True
    except socket.timeout as e:
        page = 'ERROR: Timeout error'
        error = True
    except socket.error as e:
       page = "ERROR: Dammit! I can't connect with host "+args.host+":"+args.port
       error = True
    except:
       page = "ERROR: Unexpected error (I'm damned if I know!): %s"%(sys.exc_info()[0])
       error = True

    # Show page if -vvv option
    if args.verbosity>2:
        print "page "+url_tomcat+" content:"
        print page
        print ""
    return page,error

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# ARGPARSE OBJECT DEFINITION
#--------------------------------------------------------------------------------
parser = argparse.ArgumentParser(description=plugin_description,
                                 formatter_class=argparse.RawTextHelpFormatter)
parser.version = parser.prog+" 1.0"
parser.add_argument('-V','--version', action='version',
                    help="Show plugin version",
                    version='%(prog)s '+version)
parser.add_argument('-v', '--verbosity',action="count",
                    help='''increase output verbosity:
                    -v Single line, additional information (eg list processes that fail)\n
                    -vv Multi line, configuration debug output (eg ps command used)
                    -vvv Lots of detail for plugin problem diagnosis
                    ''')

# Connection parameters
conn_parameters = parser.add_argument_group('Connection parameters',
                  'parameters for Tomcat connection')
conn_parameters.add_argument('-H', '--host',
                    help="Name or Ip of tomcat host",
                    required=True)
conn_parameters.add_argument('-p','--port',
                    help="Tomcat port (Example:8080)",
                    required=True)
conn_parameters.add_argument('-u','--user',
                    default = "admin",
                    help="Tomcat user")
conn_parameters.add_argument('-a','--authentication',
                    metavar='PASS',
                    default = "tomcat",
                    help="Tomcat authentication password")
conn_parameters.add_argument('-U','--URL',
                    default = "/manager",
                    help='''Tomcat manager app url "/manager" by default''')
conn_parameters.add_argument('-C','--connector',
                    help='''Connector name, used in thread mode''')
conn_parameters.add_argument('-t', '--timeout',
                    default = "5",
                    help='''Timeout for connection (5 seconds by default)''')
conn_parameters.add_argument('-e','--expire_time',
                    default = "0",
```

```
                    help='''Expire time for sessions created in tomcat manager app
                    value in minutes (0 minutes by default)''')

parameters = parser.add_argument_group('Check parameters',
             'Parameters for tomcat check')
parameters.add_argument('-n','--nameapp',
                    help="Name of the java application you want to check, only for app
mode")
parameters.add_argument('-w','--warning',
                    help="Warning value")
parameters.add_argument('-c','--critical',
                    help="Critical value")
parameters.add_argument('-m','--mode',
                    choices=['status','mem','thread','app'],
                    help=mode_help,
                    required=True)


#Corrects negative numbers in arguments parser
for i, arg in enumerate(sys.argv):
    if arg!="": #prevent an empty character, as ""
        if (arg[0] == '-') and arg[1].isdigit(): sys.argv[i] = ' ' + arg
# arguments parse
args = parser.parse_args()
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



#No arguments
if sys.argv[1:] == []:
    parser.print_usage()
    parser.exit(status['UNKNOWN'],
                "ERROR: No arguments, write '"+parser.prog+" -h' for help\n")

if args.verbosity!=None:
    if args.verbosity>3:
        args.verbosity=3
    print "verbosity level = %i\n"%(args.verbosity)

if args.verbosity:
    print "Arguments: %s" %(str(args))
#plugin logic...



#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#MODE OPTIONS LOGIC
#--------------------------------------------------------------------------------

#Set timeout global
socket.setdefaulttimeout(float(args.timeout))

#read serviceinfo
url_serverinfo = args.URL+"/serverinfo"
page_serverinfo,error_serverinfo =
read_page(args.host,args.port,url_serverinfo,args.user,args.authentication)
if args.verbosity>2:
    print "serverinfo:"
    print page_serverinfo

# if error, try the manager/text/serverinfo, because in tomcat 7 change the path
# of the manager app commands
if(error_serverinfo):
    url_serverinfo = args.URL+"/text/serverinfo"
    page_serverinfo,error_serverinfo =
```

```
read_page(args.host,args.port,url_serverinfo,args.user,args.authentication)
# Now it is an error yes or yes
if(error_serverinfo):
    output = page_serverinfo
    exit_status='UNKNOWN'
if(error_serverinfo==False):
    # read tomcat version in serverinfo
    serverinfo = page_serverinfo.splitlines()
    if args.verbosity>1:
        print "Server info split: "
        print serverinfo
        print ""
    tomcat_version_string = (serverinfo[1].split(":"))[1]
    tomcat_status_string = serverinfo[0]
    if args.verbosity>2:
        print "tomcat_version_string: "+tomcat_version_string
        print "tomcat_status_string: "+tomcat_status_string
    #tomcat version is read in line 2 of serverinfo
    #example:" Apache Tomcat/7.0.53", choose the "7"
    tomcat_version = (tomcat_version_string.split("/"))[1].split(".")[0]
    if args.verbosity:
        print "tomcat_version: "+tomcat_version
    #If i can't read the tomcat version because it is not a number
    if (tomcat_version.isdigit()==False):
        tomcat_version=0
        if args.verbosity:
            print "WARNING: I can't read the tomcat version"

# status option
#--------------------------------------------------------------------------------
if args.mode == 'status':
    #Default state is CRITICAL
    exit_status='CRITICAL'
    # If serverinfo page is correct
    if (error_serverinfo!=True):
        # check if the first line of serverinfo content "OK"
        if (tomcat_status_string.find("OK")!=-1):
            output = tomcat_version_string+" server is OK"
            exit_status='OK'
        else:
            output="This server is not a tomcat server or "+url_serverinfo+" is not the
manager app server info page"
            exit_status='UNKNOWN'
    # if serverinfo page is not correct try with th status xml page
    else:
        tree_xml,error_status_xml =
read_page_status_XML(args.host,args.port,args.URL,args.user,args.authentication)
        #check if page status xml is OK
        if (error_status_xml!=True):
            if (tree_xml!=None):
                if tree_xml.tag=='status':    #The first tag of xml is "status"
                    output="The Tomcat server is OK, but page serverinfo not work"
                    exit_status='OK'
                else:
                    output="This server is not a tomcat server or not status xml page"
                    exit_status='UNKNOWN'
            else:
                output="I can't read either serviceinfo or serverstatus, this server not
seems a Tomcat Server"
                exit_status='CRITICAL'
        else:
            output=tree_xml
            exit_status='CRITICAL'
```

```
# mem option
#-------------------------------------------------------------------------------
if args.mode == 'mem':
    # read status xml for extract mem data
    tree_xml,error_status_xml =
read_page_status_XML(args.host,args.port,args.URL,args.user,args.authentication)
    if error_status_xml:
        output = tree_xml
        exit_status = 'WARNING'
    else:
        if tree_xml!=None:
            #control warning and critical values
            if (args.warning==None) or (args.critical==None):
                parser.print_usage()
                parser.exit(status['UNKNOWN'],
                            'ERROR: Warning and critical values requiered with mode
"mem"\n')
            memory = tree_xml.find('.//memory')
            free_memory = float(memory.get('free'))
            total_memory = float(memory.get('total'))
            max_memory = float(memory.get('max'))
            available_memory = free_memory + max_memory - total_memory
            used_memory = max_memory - available_memory
            percent_used_memory = float((used_memory * 100)/max_memory)
            if args.verbosity:
                print "mode: mem(memory)"
                if args.verbosity > 1:
                    print "free:%0.1f total:%0.1f max:%0.1f available:%0.1f used:%0.1f
percent_used:%0.2f"%(free_memory,
total_memory,max_memory,available_memory,used_memory,percent_used_memory)
                print "free_memory:%s total memory:%s
max_memory:%s"%(sizeof_fmt(free_memory),sizeof_fmt(total_memory),sizeof_fmt(max_memory))
                print "available_memory = free_memory + max_memory - total_memory -->  %s"
%(sizeof_fmt(available_memory))
                print "used_memory = max_memory - available_memory -->  %s"
%(sizeof_fmt(used_memory))
                print "percent_used_memory = (used_memory * 100)/max_memory  -->
%0.2f%%\n"%(percent_used_memory)

            #Define status whit function
            exit_status=define_status(percent_used_memory,args.warning,args.critical)
            output="Used memory "+sizeof_fmt(used_memory)+" of
"+sizeof_fmt(max_memory)+"(%0.2f%%)" %(percent_used_memory)
            perfdata="'Used_memory'=%0.0f%%;%s;%s"%(percent_used_memory,args.warning,
                                                    args.critical)

# threads option
#-------------------------------------------------------------------------------
if args.mode == 'thread':
    # read status xml for extract mem data
    tree_xml,error_status_xml =
read_page_status_XML(args.host,args.port,args.URL,args.user,args.authentication)
    if error_status_xml:
        output = tree_xml
        exit_status = 'WARNING'
    else:
        if tree_xml!=None:
            #control warning and critical values
            if (args.warning==None) or (args.critical==None):
                parser.print_usage()
                parser.exit(status['UNKNOWN'],
                            'ERROR: Warning and critical values of number of threads open is
requiered with mode "thread"\n')
            if(args.connector==None):
                if (args.verbosity>0): print "Finding all connectors"
```

```
                for connector in tree_xml.findall('./connector'):
                    connector_name = str(connector.get('name'))
                    if (args.verbosity>0): print "Find %s connector"%(connector_name)
                    thread = connector.find('./threadInfo')
                    max_thread = float(thread.get('maxThreads'))
                    busy_thread = float(thread.get('currentThreadsBusy'))
                    iter_status=define_status(busy_thread,args.warning,args.critical)
                    if status[iter_status] > status[exit_status]:
                            exit_status=iter_status
                    output = output + '/connector:%s %0.0f threads busy of %0.0f
'%(connector_name,busy_thread,max_thread)
                    perfdata = perfdata + "'conn %s'=%0.0f;%s;%s;0;%0.0f
"%(connector_name,busy_thread,args.warning,args.critical,max_thread)

            else:
                if (args.verbosity>0): print "Finding %s connector"%(args.connector)
                for connector in tree_xml.findall('./connector'):
                    connector_name = str(connector.get('name'))
                    if (args.connector==connector_name):
                        if (args.verbosity>0):
                            print "Find %s connector"%(connector_name)
                        thread = connector.find('./threadInfo')
                        max_thread = float(thread.get('maxThreads'))
                        busy_thread = float(thread.get('currentThreadsBusy'))
                        exit_status=define_status(busy_thread,args.warning,args.critical)
                        output = output + 'connector:%s %0.0f threads busy of %0.0f
'%(connector_name,busy_thread,max_thread)
                        perfdata = perfdata + "'conn %s'=%0.0f;%s;%s;0;%0.0f
"%(connector_name,busy_thread,args.warning,args.critical,max_thread)

# app option
#-------------------------------------------------------------------------------
if args.mode == 'app':
    #Watch if nameapp is defined
    if args.nameapp==None:
        parser.print_usage()
        parser.exit(status['UNKNOWN'],
                            'ERROR: nameapp value requiered with mode "app"\n')
    #If serverinfo is not read
    if error_serverinfo:
        output="I can't read the serverinfo page. "+page_serverinfo
        exit_status='UNKNOWN'
    #If serverinfo is read
    else:
        #for versions upper Tomcat 6
        if (int(tomcat_version) > 6):
            url_list = args.URL+"/text/list"
        else:
            url_list = args.URL+"/list"

        #read application list page
        page_list,error_list =
read_page(args.host,args.port,url_list,args.user,args.authentication)
        #If list page is not read
        if error_list:
            output="I can't read the list page of tomcat manager "+page_list
            exit_status='UNKNOWN'
        #If list page is read
        else:
            #Divide page_list in lines, each line is an application in the tomcat server
            applist = page_list.splitlines()
            if args.verbosity>1:
                print "page_list split:"
                print applist
                print ""
```

```
            #Applications in page list are like "/the_name_of_the_app"
            matchapp = "/"+args.nameapp
            match = False     #Flag
            for application in applist:
                application = application.split(":")
                if matchapp == application[0]:
                    match=True
                    break
            #If match app
            if match:
                if application[1]=="running":
                    output = args.nameapp+" is running"
                    exit_status="OK"
                elif application[1]=="stopped":
                    output = args.nameapp+" is stopped"
                    exit_status="WARNING"
                else:
                    output = "I can't understand the state "+application[1]
                    exit_status="WARNING"
                #perfdata is the number of sessions for the application
                perfdata="'sessions'="+application[2]
            #If not match app
            else:
                output = "I can't find the "+matchapp+" application in the tomcat server"
                exit_status="CRITICAL"

#---------------------------------------------------------------------------
#---------------------------------------------------------------------------
#Expire sessions

#If serverinfo is read
if not error_serverinfo:
    if (int(tomcat_version) > 6):
        url_expire = args.URL+"/text/expire?path="+args.URL+"&idle="+args.expire_time
    else:
        url_expire = args.URL+"/expire?path="+args.URL+"&idle="+args.expire_time
    #for versions upper Tomcat 6
    #Expire sessions of manager app
    page_expire,error_expire =
read_page(args.host,args.port,url_expire,args.user,args.authentication)
if (args.verbosity>0):
    if not error_expire:
        print("Expire sessions of "+args.URL)
    if (args.verbosity>1):
        print("Expire information:")
        print(page_expire)



#---------------------------------------------------------------------------
#---------------------------------------------------------------------------
#Outputs

if output=='':
    output = "ERROR: no output"
    exit_status ='UNKNOWN'
message = exit_status + " " + output
if perfdata!="":
    message = message + '|' + perfdata
if longoutput!="":
    message = message + longoutput
print message
sys.exit(status[exit_status])
```

Una volta creato lo script nella cartella indicata assegnare permessi di escuzione.

```
chmod +x /usr/lib/nagios/plugins/check_tomcap.py
```

Editare il file

```
/etc/nagios/nrpe.cfg
```

aggiungiere le seguenti righe:

```
command[check_tomcat_status]=/usr/lib/nagios/plugins/check_tomcat.py -H localhost -p 8080 -u admin
-a 3dinformatica -m status
command[check_tomcat_mem]=/usr/lib/nagios/plugins/check_tomcat.py -H localhost -p 8080 -u admin -a
3dinformatica -m mem -w 80 -c 90
command[check_tomcat_thread]=/usr/lib/nagios/plugins/check_tomcat.py -H localhost -p 8080 -u admin
-a 3dinformatica -m thread -w 150 -c 190
command[check_tomcat_app]=/usr/lib/nagios/plugins/check_tomcat.py -H localhost -p 8080 -u admin -a
3dinformatica -m app $ARG1$
command[check_tomcat_app_jenkins]=/usr/lib/nagios/plugins/check_tomcat.py -H localhost -p 8080 -u
admin -a 3dinformatica -m app -n jenkins
```

NB: check_tomcat_app_jenkins verifica che l'applicazione jenkins sia operativa. Questa riga è personalizzabile e replicabile per tutte le app deployate in Tomcat es: "... -n Docway4"

template

template_monitoraggio_tomcat.cfg

```
define host{
    use         linux-server
        host_name       tomcat-test
        alias           tomcat-test
#   hostgroups      MSA-internal
        address         10.17.61.34
    parents        vmtest
}

define service{
        use                     generic-service
        host_name               tomcat-test
        service_description     PING
        check_command           check_ping!100.0,20%!500.0,60%
}

define service {
        service_description             Tomcat Status
        host_name                       tomcat-test
        check_command                   check_nrpe_1arg!check_tomcat_status
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}

define service {
        service_description             Tomcat Memory
        host_name                       tomcat-test
        check_command                   check_nrpe_1arg!check_tomcat_mem
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}

define service {
        service_description             Jenkins
        host_name                       tomcat-test
        check_command                   check_nrpe_1arg!check_tomcat_app_jenkins
```

```
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}

define service {
        service_description             Tomcat Thread
        host_name                       tomcat-test
        check_command                   check_nrpe_1arg!check_tomcat_thread
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}

define service {
        host_name                       tomcat-test
        service_description             Tomcat Port
        check_command                   check_http!-p 8080
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}
```

Mappa

Icone

Nagios core richiede 4 formati dello stesso file (gd2, png, jpg, gif). Usare icone 40×40 con sfondo trasparente. Ci sono diverse utility ed è stato realizzato anche uno script che dato come argomento un gif realizza la conversione negli altri formati.

nagiosicon.sh

```
#!/bin/bash
# Create Nagios Icon from GIF
# Autor : Marvin Pascale
# Make a 40x40 px GIF image without backgroud
# Usage: nagiosicon.sh image.gif
#install: libgd-tools pnmtopng
#Enjoy nerd

if [ $# -eq 0 ]
  then
    echo ""
    echo "Error:"
    echo "No arguments supplied"
    echo "Usage : ./nagiosicon.sh image.gif"
    echo ""
fi

path="/usr/bin"
for arg
do
  if [ -f "$arg" ]; then
    echo converting $arg
    arg="$(echo $arg | sed 's/\.gif$//')"
    echo "Creating folder" $arg
    mkdir $arg
    cp $arg.gif $arg/
    $path/giftopnm $arg.gif > $arg.pnm
    $path/pnmtopng -transparent rgb:ff/ff/ff $arg.pnm > $arg/$arg.png
    $path/pnmtojpeg -quality=100 -optimize -smooth=0 $arg.pnm > $arg/$arg.jpg
    $path/pngtogd2 $arg/$arg.png $arg/$arg.gd2 0 1
  fi
done

rm -f *.pnm
```

Collegamenti gerarchici

Per visualizzare correttamente le gerarchie nella mappa di Nagios bisogna correttamente utilizzare la specifica “parents” in fase di dichiarazione dell'host. In questa maniera si indica al core di Nagios a chi è fisicamente o virtualmente collegato l'host in quesitone es:

```
define host{
  use             linux-server ; Template host, esistente generic-host
  host_name       tomcat-test  ; il nome che si assegna all'host, servirà per assegnare servizi
  alias           tomcat-test  ; il nome che sarà visualizzato come nome esteso
  address         10.17.61.34  ; Idirizzo ip o hostname
  parents         vmtest       ; Collegamento fisico o virtuale. Anche più valori separati da una
virgola
}
```