[Install and Configure Nagios 4.0.8 – A Step-by-Step Guide](https://aravindkrishnaswamy.wordpress.com/2014/11/05/install-and-configure-nagios-4-0-8-a-step-by-step-guide/).

So, our task here is to install and Configure Nagios. I choose to host it out of AWS on an AWS Linux instance. I am not going in to the details of how to launch an instance in AWS, once we have a running AWS instance following are the steps to setup Nagios.

The latest stable Nagios build is version 4.0.8 and the latest stable Nagios plugins is version 2.0.3.

**Step 1:** Install the required dependencies

# yum install -y httpd php gcc glibc glibc-common gd gd-devel make net-snmp

This command installs all required dependencies for Nagios to function properly.

**Step 2:** Create Nagios User and Group

# useradd nagios  
# groupadd nagcmd

**Step 3:** Add Nagios and Apache user to the group that we just created

# usermod -G nagcmd nagios  
# usermod -G nagcmd apache

**Step 4:** Download Nagios Core and its corresponding plugins

First create a directory where you want to download Nagios components.

# mkdir /root/downloads  
# cd /root/downloads

Now use wget to download Nagios components to the directory that we just created.

# wget <http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz>  
# wget <http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz>

Once the download completes extract the tar files as follows

# tar -zxvf nagios-4.0.8.tar.gz  
# tar -zxvf nagios-plugins-2.0.3.tar.gz

**Step 5:** Configure Nagios Core

Run the following commands to configure Nagios

# cd nagios-4.0.8  
# ./configure –with-command-group=nagcmd

Once configure is successful, run the following commands to compile and install the binaries

# make all  
# make install  
# make install-init  
# make install-commandmode  
# make install-config

This completes Nagios core installation. Now, open contacts.cfg file to add the email address to which Nagios should alert.

# vi /usr/local/nagios/etc/objects/contacts.cfg

**Step 6:** Install and Configure Nagios Web Interface

In this step with will install Nagios Web Interface, create and assign a password for the user nagiosadmin and start Apache

# make install-webconf  
# htpasswd -s -c /usr/local/nagios/etc/htpasswd.users nagiosadmin  
# service httpd start

**Step 7:** Compile and Install Nagios Plugins

# cd /root/nagios  
# nagios-plugins-2.0.3.tar.gz  
# ./configure –with-nagios-user=nagios –with-nagios-group=nagios  
# make  
# make install

This completes the installation of Nagios and its plugins. Now to verify Nagios installed successfully without errors run the following command

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

This command should return

Total Warnings: 0  
Total Errors:   0

if the installation was successful.

**Step 8:**Add Nagios Services to System Startup

# chkconfig –add nagios  
# chkconfig –level 35 nagios on  
# chkconfig –add httpd  
# chkconfig –level 35 httpd on

Restart Nagios service for the changes to take effect.

# service nagios restart

If all the above steps completed without error you should be able to access the Nagios Web Interface using “[**http://server-IP/nagios**](http://server-ip/nagios)” with user **nagiosadmin**and password we created in **Step 6.**

In this post I am installing Nagios server and NRPE plugin to monitor the whole infrastructure from one system. Here I have created an Amazon Centos Fresh instance Installing the packages required by Nagios.  
  
First of all set the hostname of the server.

hostname nagios

Install the required packages

yum install httpd php gcc gd glibc-common gd-devel

Adding the User Nagios and setting up the password.

useradd -m nagios

passwd nagios

Creating the group nagcmd and adding nagios user in it.

groupadd nagcmd

usermod -a -G nagcmd nagios

usermod -a -G nagcmd httpd

usermod -a -G nagcmd apache

Downloading the source and Install.

cd /usr/src/

wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-3.2.3.tar.gz

wget http://prdownloads.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.11.tar.gz

tar xvf nagios-3.2.3.tar.gz

cd nagios-3.2.3

Run configure script to check the dependencies, passing the name of the group you created earlier like so.

./configure –with-command-group=nagcmd

make all

make install

make install-init

make install-config

make install-commandmode

Change the Contact email in the following file.

vi /usr/local/nagios/etc/objects/contacts.cfg

Install the web content and create a login for the nagiosadmin.

make install-webconf

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

/etc/init.d/httpd restart

cd /usr/local/nagios/

vi /usr/local/nagios/etc/nagios.cfg

/etc/init.d/nagios status

/etc/init.d/nagios start

Install the Plugin.

tar xvf nagios-plugins-1.4.11.tar.gz

cd nagios-plugins-1.4.11

/etc/init.d/nagios stop

./configure –with-nagios-user=nagios –with-nagios-group=nagios

make

make install

chkconfig –add nagios

chkconfig nagios on

Test the Nagios Configuration by the following command.

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

/etc/init.d/nagios start

Also check the sendmail/postfix is running and configured on your server so you can get email notification from nagios.

/etc/init.d/sendmail status

netstat -ntulp

/etc/init.d/sendmail start

At this point Nagios is installed and running, you can access the nagios by http://your-ip-address/nagios If you want access add more clients to Nagios so you can monitor the whole infrastructure from one centralized location, we need to setup NRPE. Following are details for that. **NRPE Installation on the client end.**

cd /home/toqeer/

/usr/sbin/useradd nagios

passwd nagios

cd /usr/src/

wget http://prdownloads.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.16.tar.gz

tar xvf nagios-plugins-1.4.16.tar.gz

cd nagios-plugins-1.4.16

./configure

make

You might get SSL error (check\_http.c:312: error: âssl\_versionâ undeclared (first use in this function)

)

in that case run this command on centos //yum install openssl-devel and rerun the following commands

./configure

make

make install

chown nagios:nagios /usr/local/nagios/

chown nagios:nagios /usr/local/nagios/libexec/ -R

yum install xinetd // For Debina based use apt-get install xinetd

cd ..

wget http://prdownloads.sourceforge.net/sourceforge/nagios/nrpe-2.13.tar.gz

tar xvf nrpe-2.13.tar.gz

cd nrpe-2.13

./configure

make all

make install-plugin

make install-daemon

make install-daemon-config

make install-xinetd

vi /etc/xinetd.d/nrpe

Add the IP address of the Nagios server. //only\_from = 127.0.0.1, IP-Address-of Nagios Server

Also add the the following line in /etc/services.

vi /etc/services

nrpe 5666/tcp # NRPE

/etc/init.d/xinetd restart

netstat -ntulp | grep nrpe

netstat -at | grep nrpe

/etc/init.d/iptables status

/usr/local/nagios/libexec/check\_nrpe -H localhost -c check\_users

/usr/local/nagios/libexec/check\_nrpe -H localhost -c check\_load

netstat -at | grep nrpe

/usr/local/nagios/libexec/check\_nrpe -H localhost -c check\_hda1

If you run the above command it ill not be able to find hda1

(DISK CRITICAL - /dev/hda1 is not accessible: No such file or directory)

because there is no hda1 on our system so open the file nrpe.cfg and replace the line with sda1.

vi /usr/local/nagios/etc/nrpe.cfg

Replace

command[check\_hda1]=/usr/local/nagios/libexec/check\_disk -w 20% -c 10% -p /dev/hda1

With

command[check\_sda1]=/usr/local/nagios/libexec/check\_disk -w 20% -c 10% -p /dev/sda1

/usr/local/nagios/libexec/check\_nrpe -H localhost -c check\_sda1

NRPE setup on Nagios Server

wget http://prdownloads.sourceforge.net/sourceforge/nagios/nrpe-2.13.tar.gz

tar xvf nrpe-2.13.tar.gz

cd nrpe-2.13

./configure

yum install openssl-devel

./configure

make all

make install-plugin

/usr/local/nagios/libexec/check\_nrpe -H Hosr-IP-Addr

Open the port 5888 on Amazon AWS console in the concerned security group and then telnet to verify things are working fine.

telnet Host-IP-Addr 5888

/usr/local/nagios/libexec/check\_nrpe -H Hostname

cd /usr/local/nagios/etc/

cd objects/

Now create a new file or open a new file with vi and add the following lines for the initial configuration and then put that file in nagios.cfg.

vi host1.cfg

define host{

use linux-server ; Inherit default values from a template

host\_name Test-App ; The name we're giving to this server

alias Centos 5 ; A longer name for the server

address 192.168.1.33 ; IP address of the server

}

define service{

use generic-service

host\_name Test-App

service\_description CPU Load

check\_command check\_nrpe!check\_load

}

define service{

use generic-service

host\_name Test-App

service\_description Current Users

check\_command check\_nrpe!check\_users

}

define service{

use generic-service

host\_name Test-App

service\_description /dev/sda1 Free Space

check\_command check\_nrpe!check\_sda1

}

define service{

use generic-service

host\_name Test-App

service\_description Total Processes

check\_command check\_nrpe!check\_total\_procs

}

define service{

use generic-service

host\_name Test-App

service\_description Zombie Processes

check\_command check\_nrpe!check\_zombie\_procs

}

vi /usr/local/nagios/etc/nagios.cfg

**cfg\_file=/usr/local/nagios/etc/objects/host1.cfg**

Save the file and reload nagios and verify the configuration with the following command.

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

Also you can verify manually the NRPE and applying the following command and replace the IP-Address with the IP of the NRPE client.

/usr/local/nagios/libexec/check\_nrpe -H IP-Addrees

[Nagios is a amazing tool](http://www.nagios.com/solutions/aws-monitoring) that can help anyone setup a robust and flexible systems monitoring environment. It works great in Amazon Web Services and can add a extra layer of visibility into the application services you are managing there.  Amazon Web Services has a service called [Cloudwatch that does basic systems monitoring](http://aws.amazon.com/cloudwatch/" \t "_blank).  Nagios can be used to augment this service. Amazon CloudWatch Basic Monitoring collects and reports metrics for CPU utilization, data transfer, and disk usage activity from each Amazon EC2 instance at a five-minute frequency. Amazon CloudWatch Detailed Monitoring provides these same metrics at one-minute intervals, and also enables data aggregation by Amazon EC2 AMI ID and instance type. While this information that is collected and aggregated by Cloudwatch is certainly useful to you, you may want to do custom alerts and service checks. That is where Nagios really shines. This post will go over a basic example of setting up Nagios to monitor a application server in a VPC with one subnet.

## What is Nagios?

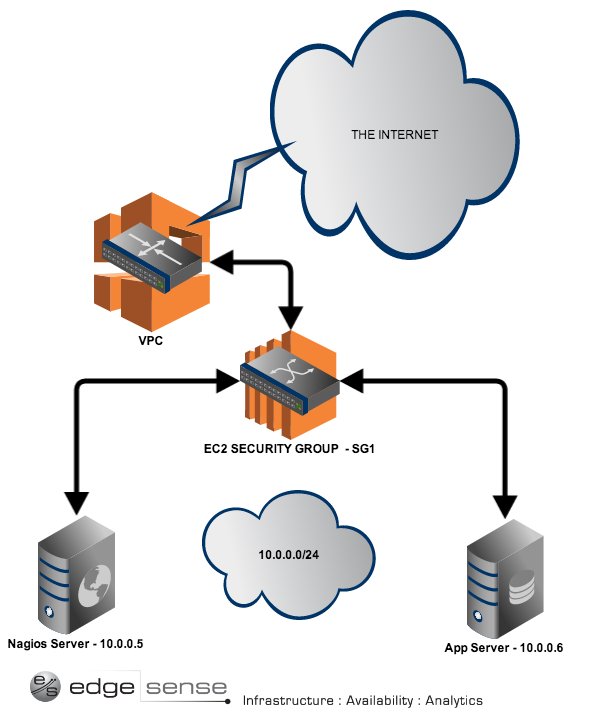
Nagios is a flexible and extensible open source systems monitoring solution.  Nagios is widely considered to be one of the best systems and network monitoring tool available, open source or otherwise. It is scalable and well supported by a dedicated community of developers. There is a paid version currently named [Nagios XI](http://www.nagios.com/products/nagiosxi) and a open source version version named [Nagios Core](http://www.nagios.com/products/nagioscore).  Nagios Core is open source by GPLv2. Want to create a systems monitoring environment that has visibility into the physical, network,  and server infrastructure of your data center? Nagios can do it. Want to deliver alerts when your highly available database server suddenly becomes unavailable? Nagios can do it.

## Why Use Nagios?

You should use Nagios because your systems will fail. When they do fail, you need to have a high level of awareness and visibility to understand exactly why the system failed. Is it the upstream network provider? Did a package update introduce a incompatibility on the application side? Did the network administrator change some firewall rules without running the changes by the application developers? Nagios excels at interoperability and customization. No two applications are architected the same. The variations in underlying computing infrastructure and management structures mean you need to be able to tailor your alert and messaging to very specific failure conditions. You should use Nagios to better manage failure and recover from it gracefully.

## Setting up Nagios on AWS

Nagios Core on AWS is easy to setup. We will walk through a very basic example of using Nagios Core to help us manage a [AWS Virtual Private Cloud (VPC)](http://aws.amazon.com/vpc/) that has two servers. We assume that you have a VPC with two servers running Ubuntu 12.04 LTS. One server will be for Nagios. The other server will be our App Server and we will use Nagios to monitor it. Both of these servers are in security group SG1. Here is a basic diagram:



A example of a VPC with two servers, one subnet, one security group.

### Nagios Terminology

Before we go any further, we need to explain some basic Nagios terminology. Keep these terms fresh in your mind:

* Host:: is a server, workstation, or network device that is being monitored.
* Host Group: a group of similar hosts. You could have a 'DB Servers' group.
* Service: the service being monitored on the host. Such as HTTP, SSH, NFS, etc.
* Service Group: allows you to group multiple services together.
* Contact: person to be notified when an event takes place.

Now that you have this terminology down, we are ready to get hands on.

### 1) Setting up the Nagios Server

Open up port 22 and port 80 by adding a rule to security group SG1 so you can SSH into the Nagios Server and see the Nagios status page.  We will install Nagios3 core and the Nagios NRPE plugin. The [NRPE plugin](http://exchange.nagios.org/directory/Addons/Monitoring-Agents/NRPE--2D-Nagios-Remote-Plugin-Executor/details)  allows us to monitor metrics such as disk usage on remote linux/unix machines. There are other ways to accomplish that, but we will use NRPE for now. Install Nagios3 and NRPE by running the following command on the Nagios Server:

sudo apt-get install nagios3 nagios-nrpe-plugin

You will be asked to enter a password for the nagiosadmin user. This is the login and password you will use to access the Nagios status page at http://(your server IP)/nagios3. The user's credentials are stored in /etc/nagios3/htpasswd.users. To change the nagiosadmin password, or add additional users to the Nagios CGI scripts, use the htpasswd utility that is part of the apache2-utils package.

### 2) Setting up the App Server

On the App Server we just want to install the NRPE plugin so Nagios can monitor metrics such as disk usage and CPU load. Install NRPE, MySql, and Apache on the App Server and create a user named 'nagios' on the MySql server:

sudo apt-get install nagios-nrpe-server  
sudo apt-get install mysql-server  
# need a user that nagios can log in as to verify service check  
mysql -u root -p -e "create user nagios identified by 'apassword';"  
# install apache and follow the instructions  
sudo apt-get install apache2

If you need assistance setting up Apache or MySql, [start here at the Ubuntu server guide.](https://help.ubuntu.com/12.04/serverguide/httpd.html)

### 3) Configuring Nagios

Now comes the fun part, configuring. We are back on the Nagios Server now. Nagios Core is configured by editing text files. Nagios XI has a web GUI instead. Here is a overview of the Nagios application directories that you will find on the Nagios Server:

* /etc/nagios3 : Configuration files for the operation of the nagios daemon
* /etc/nagios-plugins : Configuration files for the service checks

On the App Server there is one directory you need to be aware of:

* /etc/nagios : Contains the Nagios NRPE plugin configuration files.

Nagios is configured to check HTTP, disk space, SSH, current users, processes, and load on the localhost (Nagios Server) by default. We can edit the configuration files to do much more. We will start by configuring Nagios to look after our App Server. On the Nagios Server, copy your localhost\_nagios2.cfg configuration file and rename it 'appserver':

sudo cp /etc/nagios3/conf.d/localhost\_nagios2.cfg \  
/etc/nagios3/conf.d/appserver.cfg

Now let's edit this appserver.cfg file to define the details of the new host we want Nagios to monitor:

define host{  
 use generic-host ; Name of host template to use  
 host\_name appserver  
 alias App Server  
 address 10.0.0.6  
}

Now restart Nagios to enable this new configuration:

sudo /etc/init.d/nagios3 restart

Now we want to tell Nagios that there is a MySql service on the App Server host we want to monitor. Add a service definition for the MySql check by adding the following to /etc/nagios3/conf.d/services\_nagios2.cfg:

# service check for MySql servers. Make sure you replace 'apassword' with the one you set.  
define service {  
 hostgroup\_name mysql-servers  
 service\_description MySQL  
 check\_command check\_mysql\_cmdlinecred!nagios!apassword!$HOSTADDRESS  
 use generic-service  
 notification\_interval 0 ; set > 0 if you want to be renotified  
}

We are almost done setting up the Nagios server. The next step is to define a hostgroup for all servers that are running MySql so Nagios can check them. Edit /etc/nagios3/conf.d/hostgroups\_nagios2.cfg and add the following at the end of the file:

# MySql Servers hostgroup.  
define hostgroup {  
 hostgroup\_name mysql-servers  
 alias MySql Servers  
 members appserver  
 }

Now, let's restart Nagios:

sudo /etc/init.d/nagios3 restart

You should be able to go to http://(You Nagios Server IP)/nagios3 and login.

### 4) Configuring NRPE to check disk space on the App Server

You would probably want to know if you were running out of disk space on your App Server, so let's configure Nagios to check that using NRPE.  On the Nagios Server, let's tell NRPE to do a disk check on the App Server by adding this to  /etc/nagios3/conf.d/appserver.cfg:

# NRPE disk check.  
define service {  
 use generic-service  
 host\_name appserver  
 service\_description nrpe-disk  
 check\_command check\_nrpe\_1arg!check\_all\_disks!10.0.0.6  
}

Now jump over the to the App Server and let's allow access to the NRPE service to the Nagios Server. Edit /etc/nagios/nrpe.cfg to have the IP of the Nagios Server be a 'allowed host':

allowed\_hosts=10.0.0.5

This will allow the Nagios Server to ask for the metrics on the disk in the App Server. Finally, let's tell the plugin how to run the command by adding the following to the App Server's nrpe.cfg file in the command definition area:

command[check\_all\_disks]=/usr/lib/nagios/plugins/check\_disk -w 20% -c 10% -e

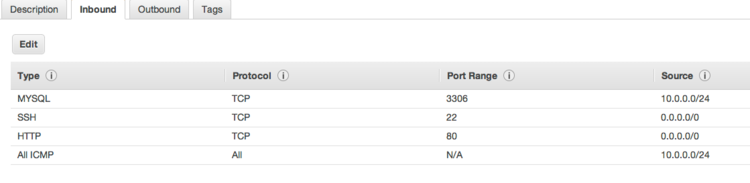
Now since we changed the config files on both the App Server and the Nagios Server we need to restart the Nagios services:

# On the App Server  
sudo /etc/init.d/nagios-nrpe-server restart

# On the Nagios Server  
sudo /etc/init.d/nagios3 restart

### 5) Adjust your VPC Security Group SG1 Rules

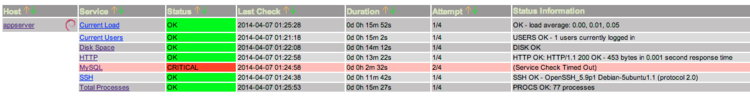
Since we want Nagios to do service checks on the MySql service that is running on the App Server, we need to open up port 3306 on by adding a rule to the SG1 security group. We only need to open up port 3306 for traffic from the subnet of 10.0.0.0/24 at this time. No need to expose this port to the world if we dont have too! We also need to allow Nagios and the App Server to ping on this subnet. Your security group rules might looks a something like this:



Your rules for SG1

### 6) Is MySql up? Ask Nagios

Now that we have Nagios up and port 3306 open on our subnet we should be able to see that the service check on MySql on the App Server went through. Here is what your Nagios 'Services' page should look like right now:



MySql is down and Nagios knows it!

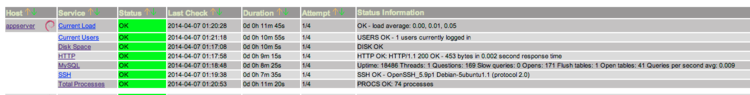
Hmmm...seems Nagios is telling us MySql is down with a critical issue. Thank goodness we found out about it now from that nice services status page on Nagios! Maybe the service is down because we did not change the bind address in the MySql /etc/mysql/my.cnf file on the App Server. Let's change the bind address to be 10.0.0.6 so the App Server can accept network connections to MySql and then Nagios will be able to connect using the 'nagios' user we created. Make the my.cnf file look like this:

bind-address = 10.0.0.6

Now, let's restart MySql on the App Server:

sudo service mysql restart

After a few minutes once the next service check happens, you should see MySql has a OK status:



MySql is OK!

### 7) Next Steps

Now that you have a good grasp of the basics of Nagios and a test environment in AWS you can customize Nagios to suit your needs. Edge Sense would be more then happy to help you get started with Nagios in AWS and help ensure that your services are always available.[Contact us anytime to talk about it](http://www.edgesense.com/contact). Thanks for reading the [libvirtblog](http://www.edgesense.com/libvirtblog). If you are looking to learn more about Nagios on your own, we recommend reading this book on Nagios by David Josephson:

**Introduction**

There are a lot of "addon" software packages that are available for Nagios. Addons can be used to extend Nagios' functionality or integrate Nagios with other applications.

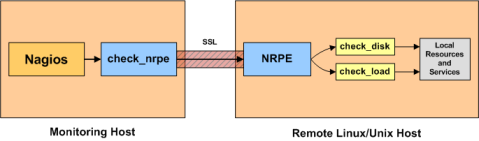
Addons are available for:

* Managing the config files through a web interface
* Monitoring remote hosts (\*NIX, Windows, etc.)
* Submitting passive checks from remote hosts
* Simplifying/extending the notification logic
* ...and much more

You can thousands of addons for Nagios by visiting Nagios Exchange at [https://exchange.nagios.org](https://exchange.nagios.org/).

A few addons that are commonly used with Nagios are described below.

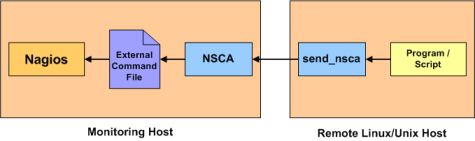
**NRPE**



NRPE is an addon that allows you to execute [plugins](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/plugins.html) on remote Linux/Unix hosts. This is useful if you need to monitor local resources/attributes like disk usage, CPU load, memory usage, etc. on a remote host. Similiar functionality can be accomplished by using the*check\_by\_ssh* plugin, although it can impose a higher CPU load on the monitoring machine - especially if you are monitoring hundreds or thousands of hosts.

The NRPE addon can be found on [Nagios Exchange](https://exchange.nagios.org/directory/Addons/Monitoring-Agents/NRPE--2D-Nagios-Remote-Plugin-Executor/details).

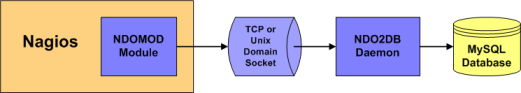
**NSCA**



NSCA is an addon that allows you to send [passive check](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/passivechecks.html) results from remote Linux/Unix hosts to the Nagios daemon running on the monitoring server. This is very useful in[distributed](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/distributed.html) and [redundant/failover](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/redundancy.html) monitoring setups.

The NSCA addon can be found on [Nagios Exchange](https://exchange.nagios.org/directory/Addons/Passive-Checks/NSCA--2D-Nagios-Service-Check-Acceptor/details).

**NDOUtils**



NDOUtils is an addon that allows you to store all status information from Nagios in a MySQL database. Multiple instances of Nagios can all store their information in a central database for centralized reporting. This will likely serve as the basis for a new PHP-based web interface for Nagios in the future.

The NDOUtils addon can be found at [Nagios Exchange](https://exchange.nagios.org/directory/Addons/Database-Backends/NDOUtils/details).

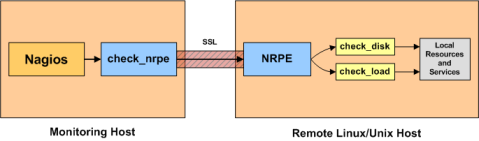
**Nagios Exchange - Hundreds of Other Addons**

[](https://exchange.nagios.org/)

Hundreds of community-developed Nagios addons can be found on the Nagios Exchange website at [exchange.nagios.org](https://exchange.nagios.org/).

# Monitoring Linux/Unix Machines

There are several different ways to monitor attributes or remote Linux/Unix servers. One is by using shared SSH keys and the *check\_by\_ssh* plugin to execute plugins on remote servers. This method will not be covered here, but can result in high load on your monitoring server if you are monitoring hundreds or thousands of services. The overhead of setting up/destroying SSH connections is the cause of this.



Another common method of monitoring remote Linux/Unix hosts is to use the [NRPE addon](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/addons.html#nrpe). NRPE allows you to execute plugins on remote Linux/Unix hosts. This is useful if you need to monitor local resources/attributes like disk usage, CPU load, memory usage, etc. on a remote host.

# Monitoring Windows Machines

**Introduction**

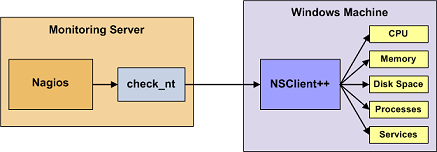
This document describes how you can monitor "private" services and attributes of Windows machines, such as:

* Memory usage
* CPU load
* Disk usage
* Service states
* Running processes
* etc.

Publicly available services that are provided by Windows machines (HTTP, FTP, POP3, etc.) can be monitored easily by following the documentation on [monitoring publicly available services](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/monitoring-publicservices.html).

Note Note: These instructions assume that you've installed Nagios according to the [quickstart guide](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/quickstart.html). The sample configuration entries below reference objects that are defined in the sample config files (*commands.cfg*, *templates.cfg*, etc.) that are installed if you follow the quickstart.

**Overview**



Monitoring private services or attributes of a Windows machine requires that you install an agent on it. This agent acts as a proxy between the Nagios plugin that does the monitoring and the actual service or attribute of the Windows machine. Without installing an agent on the Windows box, Nagios would be unable to monitor private services or attributes of the Windows box.

For this example, we will be installing the [NSClient++](http://sourceforge.net/projects/nscplus) addon on the Windows machine and using the*check\_nt* plugin to communicate with the NSClient++ addon. The *check\_nt* plugin should already be installed on the Nagios server if you followed the quickstart guide.

Other Windows agents (like [NC\_Net](http://sourceforge.net/projects/nc-net)) could be used instead of NSClient++ if you wish - provided you change command and service definitions, etc. a bit. For the sake of simplicity I will only cover using the NSClient++ addon in these instructions.

**Steps**

There are several steps you'll need to follow in order to monitor a new Windows machine. They are:

1. Perform first-time prerequisites
2. Install a monitoring agent on the Windows machine
3. Create new host and service definitions for monitoring the Windows machine
4. Restart the Nagios daemon

**What's Already Done For You**

To make your life a bit easier, a few configuration tasks have already been done for you:

* A *check\_nt* command definition has been added to the *commands.cfg* file. This allows you to use the *check\_nt* plugin to monitor Window services.
* A Windows server host template (called *windows-server*) has already been created in the *templates.cfg* file. This allows you to add new Windows host definitions in a simple manner.

The above-mentioned config files can be found in the */usr/local/nagios/etc/objects/* directory. You can modify the definitions in these and other definitions to suit your needs better if you'd like. However, I'd recommend waiting until you're more familiar with configuring Nagios before doing so. For the time being, just follow the directions outlined below and you'll be monitoring your Windows boxes in no time.

**Prerequisites**

The first time you configure Nagios to monitor a Windows machine, you'll need to do a bit of extra work. Remember, you only need to do this for the \*first\* Windows machine you monitor.

Edit the main Nagios config file.

vi /usr/local/nagios/etc/nagios.cfg

Remove the leading pound (#) sign from the following line in the main configuration file:

#cfg\_file=/usr/local/nagios/etc/objects/windows.cfg

Save the file and exit.

What did you just do? You told Nagios to look to the */usr/local/nagios/etc/objects/windows.cfg* to find additional object definitions. That's where you'll be adding Windows host and service definitions. That configuration file already contains some sample host, hostgroup, and service definitions. For the \*first\* Windows machine you monitor, you can simply modify the sample host and service definitions in that file, rather than creating new ones.

**Installing the Windows Agent**

Before you can begin monitoring private services and attributes of Windows machines, you'll need to install an agent on those machines. I recommend using the NSClient++ addon, which can be found at <http://sourceforge.net/projects/nscplus>. These instructions will take you through a basic installation of the NSClient++ addon, as well as the configuration of Nagios for monitoring the Windows machine.

1. Download the latest stable version of the NSClient++ addon from <http://sourceforge.net/projects/nscplus>

2. Unzip the NSClient++ files into a new C:\NSClient++ directory

3. Open a command prompt and change to the C:\NSClient++ directory

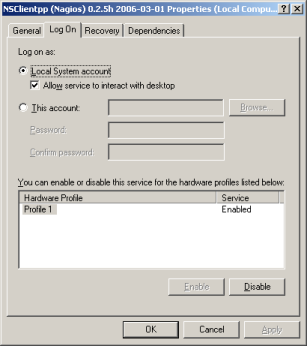
4. Register the NSClient++ system service with the following command:

nsclient++ /install

5. Install the NSClient++ systray with the following command ('SysTray' is case-sensitive):

nsclient++ SysTray

6. Open the services manager and make sure the NSClientpp service is allowed to interact with the desktop (see the 'Log On' tab of the services manager). If it isn't already allowed to interact with the desktop, check the box to allow it to.



7. Edit the NSC.INI file (located in the C:\NSClient++ directory) and make the following changes:

* Uncomment all the modules listed in the [modules] section, except for CheckWMI.dll and RemoteConfiguration.dll
* Optionally require a password for clients by changing the 'password' option in the [Settings] section.
* Uncomment the 'allowed\_hosts' option in the [Settings] section. Add the IP address of the Nagios server to this line, or leave it blank to allow all hosts to connect.
* Make sure the 'port' option in the [NSClient] section is uncommented and set to '12489' (the default port).

8. Start the NSClient++ service with the following command:

nsclient++ /start

9. If installed properly, a new icon should appear in your system tray. It will be a yellow circle with a black 'M' inside.

10. Success! The Windows server can now be added to the Nagios monitoring configuration...

**Configuring Nagios**

Now it's time to define some [object definitions](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/objectdefinitions.html) in your Nagios configuration files in order to monitor the new Windows machine.

Open the *windows.cfg* file for editing.

vi /usr/local/nagios/etc/objects/windows.cfg

Add a new [host](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/objectdefinitions.html#host) definition for the Windows machine that you're going to monitor. If this is the \*first\* Windows machine you're monitoring, you can simply modify the sample host definition in *windows.cfg*. Change the *host\_name*, *alias*, and *address* fields to appropriate values for the Windows box.

define host{

use windows-server ; Inherit default values from a Windows server template (make sure you keep this line!)

host\_name winserver

alias My Windows Server

address 192.168.1.2

}

Good. Now you can add some service definitions (to the same configuration file) in order to tell Nagios to monitor different aspects of the Windows machine. If this is the \*first\* Windows machine you're monitoring, you can simply modify the sample service definitions in *windows.cfg*.

Note Note: Replace "*winserver*" in the example definitions below with the name you specified in the *host\_name* directive of the host definition you just added.

Add the following service definition to monitor the version of the NSClient++ addon that is running on the Windows server. This is useful when it comes time to upgrade your Windows servers to a newer version of the addon, as you'll be able to tell which Windows machines still need to be upgraded to the latest version of NSClient++.

define service{

use generic-service

host\_name winserver

service\_description NSClient++ Version

check\_command check\_nt!CLIENTVERSION

}

Add the following service definition to monitor the uptime of the Windows server.

define service{

use generic-service

host\_name winserver

service\_description Uptime

check\_command check\_nt!UPTIME

}

Add the following service definition to monitor the CPU utilization on the Windows server and generate a CRITICAL alert if the 5-minute CPU load is 90% or more or a WARNING alert if the 5-minute load is 80% or greater.

define service{

use generic-service

host\_name winserver

service\_description CPU Load

check\_command check\_nt!CPULOAD!-l 5,80,90

}

Add the following service definition to monitor memory usage on the Windows server and generate a CRITICAL alert if memory usage is 90% or more or a WARNING alert if memory usage is 80% or greater.

define service{

use generic-service

host\_name winserver

service\_description Memory Usage

check\_command check\_nt!MEMUSE!-w 80 -c 90

}

Add the following service definition to monitor usage of the C:\ drive on the Windows server and generate a CRITICAL alert if disk usage is 90% or more or a WARNING alert if disk usage is 80% or greater.

define service{

use generic-service

host\_name winserver

service\_description C:\ Drive Space

check\_command check\_nt!USEDDISKSPACE!-l c -w 80 -c 90

}

Add the following service definition to monitor the W3SVC service state on the Windows machine and generate a CRITICAL alert if the service is stopped.

define service{

use generic-service

host\_name winserver

service\_description W3SVC

check\_command check\_nt!SERVICESTATE!-d SHOWALL -l W3SVC

}

Add the following service definition to monitor the Explorer.exe process on the Windows machine and generate a CRITICAL alert if the process is not running.

define service{

use generic-service

host\_name winserver

service\_description Explorer

check\_command check\_nt!PROCSTATE!-d SHOWALL -l Explorer.exe

}

That's it for now. You've added some basic services that should be monitored on the Windows box. Save the configuration file.

**Password Protection**

If you specified a password in the NSClient++ configuration file on the Windows machine, you'll need to modify the *check\_nt* command definition to include the password. Open the*commands.cfg* file for editing.

vi /usr/local/nagios/etc/objects/commands.cfg

Change the definition of the *check\_nt* command to include the "-s <PASSWORD>" argument (where PASSWORD is the password you specified on the Windows machine) like this:

define command{

command\_name check\_nt

command\_line $USER1$/check\_nt -H $HOSTADDRESS$ -p 12489 -s PASSWORD -v $ARG1$ $ARG2$

}

Save the file.

**Restarting Nagios**

You're done with modifying the Nagios configuration, so you'll need to [verify your configuration files](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/verifyconfig.html) and [restart Nagios](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/startstop.html).

If the verification process produces any errors messages, fix your configuration file before continuing. Make sure that you don't (re)start Nagios until the verification process completes without any errors!