**Nagios**

**INTRODUCTION:**

Nagios Core is open source software licensed under the GNU GPL V2.

Nagios is mainly for monitors systems, networks and infrastructure. Nagios offers monitoring and alerting services for servers, switches, applications and services. It alerts users when things go wrong and alerts them a second time when the problem has been resolved.

**PURPOSE:**

The main purpose of this Nagios guide is to setup the Nagios Core Server in Ubuntu to monitor various services like MongoDB, RabbitMQ and Redis Server along with Windows Servers services.

**ARCHITECTURE:**

* Simplest setup has central server running Nagios daemon that runs local check scripts which the status of services on that and remote hosts
* A host is a computer running on the network which runs one or more services to be checked
* A service is anything on the host that you want checked. Its state can be one of:

1. OK
2. Warning
3. Critical
4. Unknown

* A check is a script run on the server whose exit status determines the state of the service:

1. 0
2. 1
3. 2
4. -1

**Components:**

* + **nagios** – the main server software and web scripts
  + **nagios-plugins** – the common set of check scripts used to query services
  + **nagios-nrpe** – Nagios Remote Plugin Executor
  + **nagios-nsca** – Nagios Service Check Acceptor

|  |
| --- |
|  |

**Currently it provides:**

* Monitoring of network services (SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH)
* Monitoring of host resources (processor load, disk usage, system logs) on a majority of network operating systems, including Microsoft Windows, using monitoring agents.
* Monitoring of any hardware (like probes for temperature, alarms, etc.) which have the ability to send collected data via a network to specifically written plugins
* Monitoring via remotely run scripts via Nagios Remote Plugin Executor (NRPE)
* Remote monitoring supported through SSH or SSL encrypted tunnels.
* A simple plugin design that allows users to easily develop their own service checks depending on needs, by using their tools of choice (shell scripts, C++, Perl, Ruby, Python, PHP, C#, etc.)
* Available data graphing plugins
* Parallelized service checks
* Flat-text formatted configuration files (integrates with many config editors)
* The ability to define network host using 'parent' hosts, allowing the detection of and distinction between hosts that are down or unreachable
* Contact notifications when service or host problems occur and get resolved (via e-mail, pager, SMS, or any user-defined method through plugin system)
* The ability to define event handlers to be run during service or host events for proactive problem resolution
* Automatic log file rotation
* Support for implementing redundant monitoring hosts and performance data graphing
* Support for database backend (such as NDOUtils)
* A web-interface for viewing current network status, notifications, problem history, and log files, etc.

**NAGIOS AGENTS:**

1. **NRPE**

Nagios Remote Plugin Executor (NRPE) is a Nagios agent that allows remote system monitoring using scripts that are hosted on the remote systems. It allows for monitoring of resources such as disk usage, system load or the number of users currently logged in. Nagios periodically polls the agent on remote system using the **check\_nrpe plugin**.

NRPE allows you to remotely execute Nagios plugins on other Linux/Unix machines. This allows you to monitor remote machine metrics (disk usage, CPU load, etc.). NRPE can also communicate with some of the Windows agent addons, so you can execute scripts and check metrics on remote Windows machines as well.

1. **NRDP**

Nagios Remote Data Processor (NRDP) is a Nagios agent with a flexible data transport mechanism and processor. It is designed with an architecture that allows it to be easily extended and customized. NRDP uses standard ports and protocols (HTTP and XML) and can be implemented as a replacement for Nagios Service Check Acceptor (NSCA).

1. **NSClient++**

This program is mainly used to monitor Windows machines. Being installed on a remote system NSClient++ listens to port TCP 12489. The Nagios plugin that is used to collect information from this add-on is called **check\_nt**. As NRPE, NSClient++ allows to monitor the so-called 'private services' (memory usage, CPU load, disk usage, running processes, etc.) Nagios is a host and service monitor which is designed to inform your network problems.

1. **NCPA**

The Nagios Cross Platform Agent is an open source project maintained by Nagios Enterprises. NCPA installs on Windows, Linux, and Mac OS X. Created as a scale-able API that allows flexibility and simplicity in monitoring hosts. NCPA allows multiple checks such as memory usage, CPU usage, disk usage, processes, services, and network usage. Active checks are queried through the API of the "NCPA Listener" service while passive checks are sent via the "NCPA Passive" service.

1. **Nagios XI**

Nagios XI is an extended interface, config manager, and toolkit using Nagios Core as the back-end, written and maintained by the original author, “Ethan Galstad”, and Nagios Enterprises. It is an enterprise-class application that monitors systems, networks and infrastructure. It offers an extensive user interface, configuration editor, advanced reporting, monitoring wizards, an extensible front-end and back-end, along with many other additions over Nagios Core. CentOS and RHEL are the currently supported operating systems. It combines Nagios Core with other technologies. Its main database and the ndoutils module that is used alongside Nagios Core use MySQL. Prior to XI 5, PostgreSQL was used for one of the three databases it uses, and is no longer used on new installs of Nagios XI. While the front-end of Nagios Core is mainly CGI with some PHP, most of the Nagios XI front-end and back-end are written in PHP including the subsystem, event handlers, and notifications, and Python is used to create capacity planning reports and other reports. **RRDtool and Highcharts** are included to create customizable graphs that can be displayed in dashboards.

**INSTALLATION PROCEDURE:**

**Prerequisites:**

**Step 1: Install Apache:**

The Apache web server is currently the most popular web server in the world, which makes it a great default choice for hosting a website.

We can install Apache easily using Ubuntu's package manager, **apt**. A package manager allows us to install most software pain-free from a repository maintained by Ubuntu

$ sudo apt-get update

$ sudo apt-get install apache2

$ sudo service apache2 start [start/restart/stop/status…etc]

If everything went fine, you can able to browse the Apache Ubuntu default webpage by using the following URL.

http://your\_server\_IP\_address

|  |
| --- |
|  |

**Step 2: Install MySQL:**

Now that we have our web server up and running, it is time to install MySQL. MySQL is a database management system. Basically, it will organize and provide access to databases where our site can store information.

$ sudo apt-get install mysql-server php5-mysql

During the installation, your server will ask you to select and confirm a password for the MySQL "root" user. This is an administrative account in MySQL that has increased privileges. Think of it as being similar to the root account for the server itself (the one you are configuring now is a MySQL-specific account however).

When the installation is complete, we need to run some additional commands to get our MySQL environment set up securely.

First, we need to tell MySQL to create its database directory structure where it will store its information. You can do this by below command.

$ sudo mysql\_install\_db

Afterwards, we want to run a simple security script that will remove some dangerous defaults and lock down access to our database system a little bit. Start the interactive script by running:

$ sudo mysql\_secure\_installation

|  |
| --- |
| Setting the root password ensures that nobody can log into the MySQL  root user without the proper authorization.  You already have a root password set, so you can safely answer 'n'.  Change the root password? [Y/n] n  Remove anonymous users? [Y/n] y  Disallow root login remotely? [Y/n] n  Remove test database and access to it? [Y/n] y  Reload privilege tables now? [Y/n] y  All done! If you've completed all of the above steps, your MySQL installation should now be secure.  Thanks for using MySQL! |

**Step 3: Install PHP**

PHP is the component of our setup that will process code to display dynamic content. It can run scripts, connect to our MySQL databases to get information, and hand the processed content over to our web server to display.

$ sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt

In most cases, we'll want to modify the way that Apache serves files when a directory is requested. Currently, if a user requests a directory from the server, Apache will first look for a file called **index.html**. We want to tell our web server to prefer PHP files, so we'll make Apache look for an **index.php** file first.

$ sudo vim /etc/apache2/mods-enabled/dir.conf

We want to move the PHP index file highlighted above to the first position after the DirectoryIndexspecification, like this:

|  |
| --- |
| <IfModule mod\_dir.c>  DirectoryIndex **index.html** index.cgi index.pl **index.php** index.xhtml index.htm  </IfModule>  <IfModule mod\_dir.c>  DirectoryIndex **index.php** index.html index.cgi index.pl index.xhtml index.htm  </IfModule> |

After this, we need to restart the Apache web server in order for our changes to be recognized. You can do this by typing this:

$ sudo service apache2 restart

**Step 4: Test PHP Processing on your Web Server**

* In order to test that our system is configured properly for PHP, we can create a very basic PHP script.
* We will call this script info.php. In order for Apache to find the file and serve it correctly, it must be saved to a very specific directory, which is called the "web root".
* In Ubuntu 14.04, this directory is located at /var/www/html/. We can create the file at that location by typing:

$ sudo vim /var/www/html/info.php

* This will open a blank file. We want to put the following text, which is valid PHP code, inside the file, save and exit.

**<?php**

**phpinfo();**

**?>**

* Now we can test whether our web server can correctly display content generated by a PHP script. To try this out, we just have to visit this page in our web browser. You'll need your server's public IP address again.

<http://your_server_IP_address/info.php>

|  |
| --- |
|  |

If this was successful, then your PHP is working as expected.

You probably want to remove this file after this test because it could actually give information about your server to unauthorized users. To do this, you can type this:

$ sudo rm /var/www/html/info.php

You can always recreate this page if you need to access the information again later.

**Conclusion:**

Basically, you've installed LAMP stack and it will allow you to install most kinds of websites and web software on your server.

**Step 5: Other Dependencies Installation:**

Make sure you've installed the following packages on your Ubuntu installation before continuing.

* Autoconf
* GCC compiler and development libraries
* GD development libraries
* Unzip

You can use apt-get to install these packages by running the following commands:

$ sudo apt-get install autoconf gcc libc6 build-essential bc gawk dc gettext \

libmcrypt-dev libssl-dev make unzip apache2-utils php5 libgd2-xpm-dev

**Step 6: Create Nagios User Account:**

Create a new nagios user account and give it a password.

$ sudo useradd -m -s /bin/bash nagios

$ sudo passwd nagios

**Step 7: Create Nagios Group:**

You will need to also add a nagios group (if it is not created by default).

$ sudo groupadd nagios

$ sudo usermod -G nagios nagios

Create a new “**nagcmd**” group for allowing external commands to be submitted through the web interface. Add both the **nagios** user and the **apache** user to the group.

$ sudo groupadd nagcmd

$ sudo usermod -a -G nagcmd nagios

$ sudo usermod -a -G nagcmd www-data

**Step 8: Download Nagios and Plugins:**

Create a directory for storing downloads.

$ mkdir downloads

$ cd downloads

Download the source code tarballs of both Nagios Core and the Nagios plugins (visit https://www.nagios.org/download/ for links to the latest versions). These directions were tested with Nagios 4.2.4 and Nagios Plugins 2.1.4.

$ wget <http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.2.4.tar.gz>

$ wget <https://nagios-plugins.org/download/nagios-plugins-2.1.4.tar.gz>

(OR)

$ wget <http://www.nagios-plugins.org/download/nagios-plugins-2.2.0.tar.gz>

**Step 9: Compile and Install Nagios Core:**

Extract the Nagios Core source code tarball.

$ tar xvf nagios-4.2.4.tar.gz

$ ls -lrt

$ cd nagios-4.2.4

Run the Nagios “**configure”** script, passing the name of the group you created earlier, and pointing to the correct Apache configuration directory, like so:

$ ./configure --with-command-group=nagcmd --with-httpd-conf=/etc/apache2/sites-enabled

Compile the Nagios Core source code.

$ make all

Install binaries, init script, sample config files and set permissions on the external command directory.

$ sudo make install

$ sudo make install-init

$ sudo make install-config

$ sudo make install-commandmode

$ sudo update-rc.d nagios defaults

$ sudo /usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-available/nagios.conf

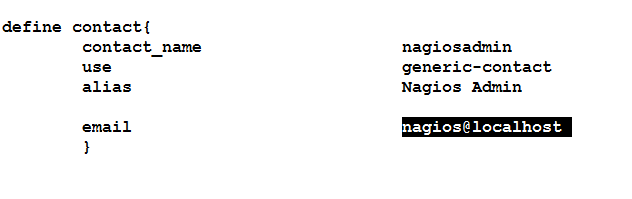
**\*Note:** Don't start Nagios Core yet - there's still more that needs to be done...

**Step 10: Customize Configuration:**

Sample configuration files have now been installed in the **/usr/local/nagios/etc** directory. These sample files should work fine for getting started with Nagios Core. You'll need to make just one change before you proceed.

Edit the **/usr/local/nagios/etc/objects/contacts.cfg** config file with your favorite editor and change the email address associated with the **nagiosadmin** contact definition to the address you'd like to use for receiving alerts:

$ sudo vim /usr/local/nagios/etc/objects/contacts.cfg



Replace the **nagios@localhost** with **nagiosadmin** email address (nagiosadmin is the person that who is going to administering the Nagios Server)

Example: vijay.manjeti@xxxx.com

**Step 11: Configure the Web Interface:**

Install the Nagios Core web config file in the Apache conf.d directory.

$ sudo make install-webconf

$ sudo a2enmod rewrite

$ sudo a2enmod cgi

$ sudo service apache2 restart

Create a **nagiosadmin** account for logging into the Nagios Core web interface. Remember the password you assign to this account - you'll need it later to login.

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

Restart Apache to make the new settings take effect.

$ sudo ufw allow apache

$ sudo ufw disable

$ sudo ufw enable

$ sudo ufw reload

**Note**\*: Come out the Nagios source directory by using cd command (ex: cd ..) to proceed further.

**Step 12: Compile and Install the Nagios Plugins**

Extract the Nagios plugins source code tarball:

$ cd ..

$ tar xvf nagios-plugins-2.1.4.tar.gz

$ cd nagios-plugins-2.1.4

Compile and install the plugins.

$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios --with-openssl

$ make

$ sudo make install

**Step 13: Start Nagios Core:**

Verify the sample Nagios Core configuration files.

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

|  |
| --- |
| Running pre-flight check on configuration data...  Checking objects...  Checked 8 services.  Checked 1 hosts.  Checked 1 host groups.  Checked 0 service groups.  Checked 1 contacts.  Checked 1 contact groups.  Checked 24 commands.  Checked 5 time periods.  Checked 0 host escalations.  Checked 0 service escalations.  Checking for circular paths...  Checked 1 hosts  Checked 0 service dependencies  Checked 0 host dependencies  Checked 5 timeperiods  Checking global event handlers...  Checking obsessive compulsive processor commands...  Checking misc settings...  Total Warnings: 0  Total Errors: 0 |

If there are no errors, start Nagios Core.

$ sudo service nagios start

**Step 14: Login to the Web Interface:**

You should now be able to access the Nagios Core web interface at the URL below. You'll be prompted for the username (nagiosadmin) and password you specified earlier. Click on services to get the status of default services.

<http://localhost/nagios/> (or) <http://host-ip-address/nagios>

|  |
| --- |
|  |

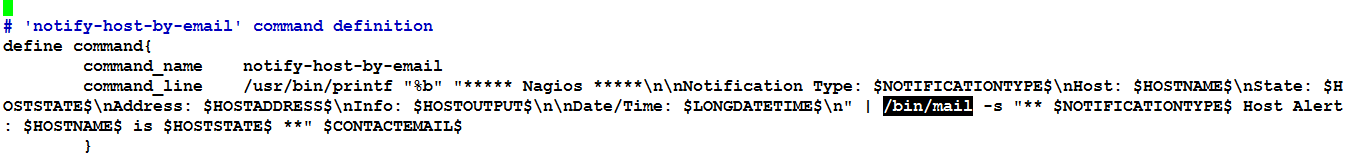
**Other Modifications:**

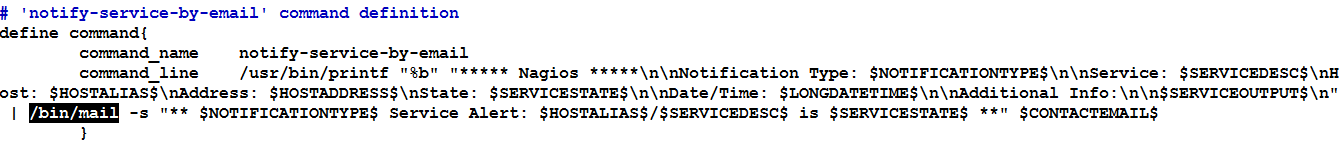
If you want to receive email notifications for Nagios Core alerts, you need to install the mailx (Postfix) package.

$ sudo apt-get install heirloom-mailx bsd-mailx mailutils  **(optional)**

$ sudo vim /usr/local/nagios/etc/objects/commands.cfg

You'll have to edit the Nagios Core email notification commands found in **/usr/local/nagios/etc/objects/commands.cfg** and change any **'/bin/mail'** references to '**/usr/bin/mail'**. Once you do that you'll need to restart Nagios Core to make the configuration changes live.





Replace the text **/bin/mail** to **/usr/bin/mail**

$ sudo service nagios restart

**NAGIOS AGENTS INSTALLATION:**

**Install NRPE:**

Find the source code for the latest stable release of NRPE at the [NRPE downloads page](http://sourceforge.net/projects/nagios/files/nrpe-2.x/). Download the latest version to your Nagios server.

Navigate to your home directory and then go to your downloads directory.

$ cd

$ cd downloads

$ wget <http://downloads.sourceforge.net/project/nagios/nrpe-2.x/nrpe-2.15/nrpe-2.15.tar.gz>

Extract the NRPE archive with this command:

$ tar xvf nrpe-2.15.tar.gz

Then change to the extracted directory:

$ cd nrpe-2.15

Configure NRPE with these commands:

64-bit:

$ ./configure --enable-command-args --with-nagios-user=nagios --with-nagios-group=nagios --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib/x86\_64-linux-gnu

32-bit:

$ ./configure --enable-command-args --with-nagios-user=nagios --with-nagios-group=nagios --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib/i386-linux-gnu

Now build and install NRPE and its **xinetd** startup script with these commands:

$ make all

$ sudo make install

$ sudo apt-get install xinetd

$ sudo make install-xinetd

$ sudo make install-daemon-config

**Configure check\_nrpe Command**

Let's add a new command to our Nagios configuration:

$ sudo vim /usr/local/nagios/etc/objects/commands.cfg

Add the following to the end of the file:

|  |
| --- |
| define command{  command\_name check\_nrpe  command\_line $USER1$/check\_nrpe -H $HOSTADDRESS$ -c $ARG1$  } |

Save and exit. This allows you to use the check\_nrpe command in your Nagios service definitions.

**xinetd conf**

Open the xinetd startup script in an editor:

Modify the **only\_from** line by adding the private IP address of your Nagios server to the end (substitute in the actual IP address of your server):

**Example:** only\_from = 127.0.0.1 192.168.153.135

Save and exit. Only the Nagios server will be allowed to communicate with NRPE.

Restart the xinetd service to start NRPE:

$ sudo service xinetd restart

**Nagios Configuration Management:**

**Nagios Directory Structure:**

|  |
| --- |
|  |

**bin:** contains nagios, nrpe (if installed) and nagiostats scripts to control the nagios server.

**etc:** containsnagios configuration files

* **cgi.cfg –** various cgi script paths
* **htpasswd.users –** Stores nagios user information like username and password.
* **nagios.cfg –** Main configuration file of the nagios server to handle the various configurations files (ex: hosts, commands, and services …..etc)
* **nrpe.cfg -**  If installed (This file holds the nrpe command definitions)
* **resources.cfg –** To configure event handlers and input user resources to set default plugin paths.
* **objects –** Which is created by default, and contains several sample configuration files like localhost.cfg, commands.cfg,…… etc

**Note:** If you want create/monitor other services you can create here, and make sure to include those in nagios.cfg to point the config file or directories which holds configuration.

**include:** By default, it is a empty directory, you can place some third-party libraries.

**libexec:** Default nagios plugins will be installed here, and it can be referred by a command using $USER$, and you can place any other plugins to monitor other services.