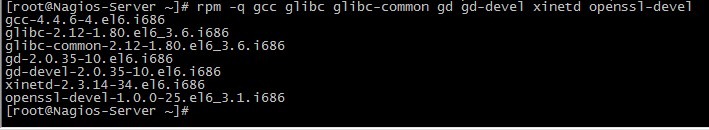
# Nagios服务端安装

# 一、安装

### 1 基础支持套件：

gcc glibc glibc-common gd gd-devel xinetd openssl-devel

# rpm -q gcc glibc glibc-common gd gd-devel xinetd openssl-devel

[](http://images.cnitblog.com/blog/370046/201301/30172023-cbd356b06f1c4fb796b99237b3554b2f.jpg)

如果系统中没有这些套件，使用yum 安装

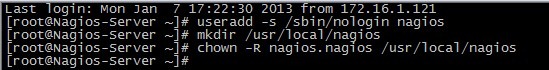
# yum install -y gcc glibc glibc-common gd gd-devel xinetd openssl-devel

### 2 创建nagios用户和用户组

# useradd -s /sbin/nologin nagios

# mkdir /usr/local/nagios

# chown -R nagios.nagios /usr/local/nagios

[](http://images.cnitblog.com/blog/370046/201301/30172026-dd983461e7e24d3fa12c39d213616afc.jpg)

查看nagios 目录的权限

# ll -d /usr/local/nagios/

[7](http://images.cnitblog.com/blog/370046/201301/30172027-2efbcf5f62f34eae857b7c48c0f94d02.jpg)

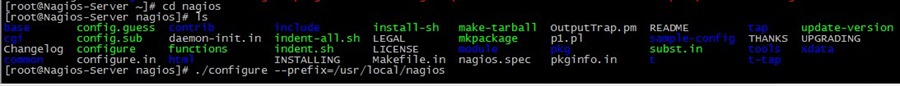
### 3 编译安装Nagios

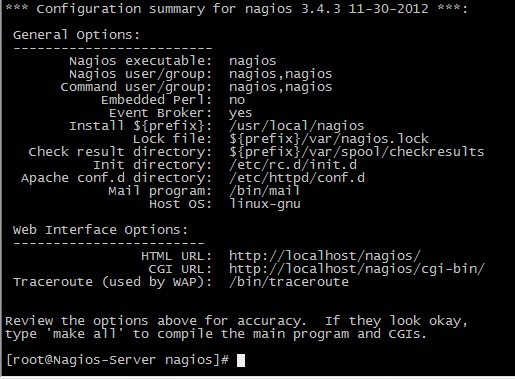
# wget <http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-3.4.3.tar.gz>

# tar zxvf nagios-3.4.3.tar.gz

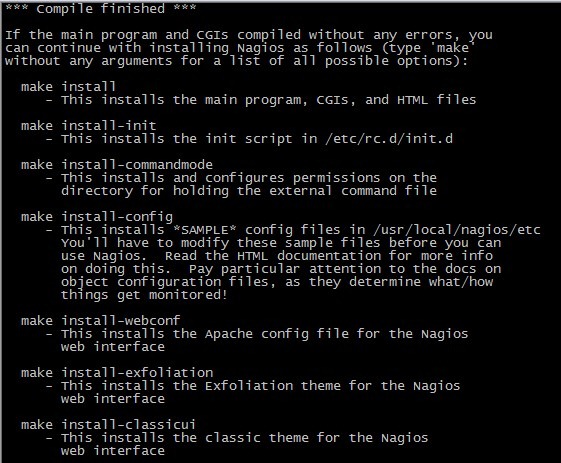
# cd nagios

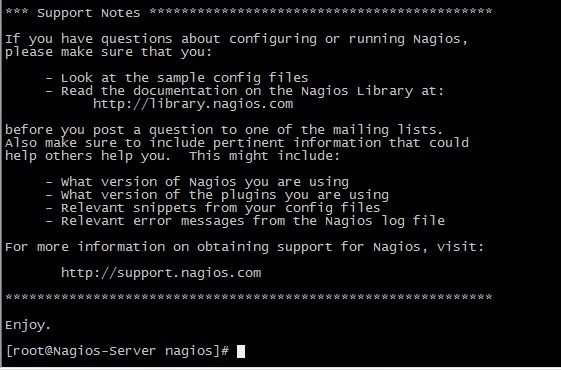
# ./configure --prefix=/usr/local/nagios

[](http://images.cnitblog.com/blog/370046/201301/30172029-51aac459f3a34106af7db6aaaa403d9e.jpg)

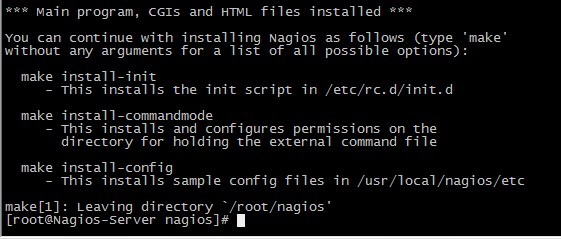
[](http://images.cnitblog.com/blog/370046/201301/30172030-18fdb7961c474b75a2b033e2a4171519.jpg)

# make all

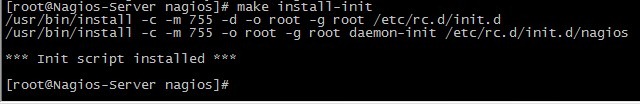
[](http://images.cnitblog.com/blog/370046/201301/30172031-5c6a7463c11d4544bee8f2eed12d7647.jpg)

[](http://images.cnitblog.com/blog/370046/201301/30172034-ebe3fcafffe94564b8161e6b78fe64c8.jpg)

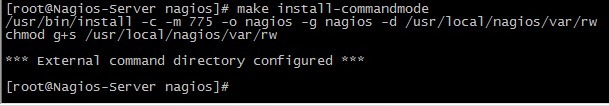
# make install

[](http://images.cnitblog.com/blog/370046/201301/30172035-b01026f0079c477c815351fb70635f40.jpg)

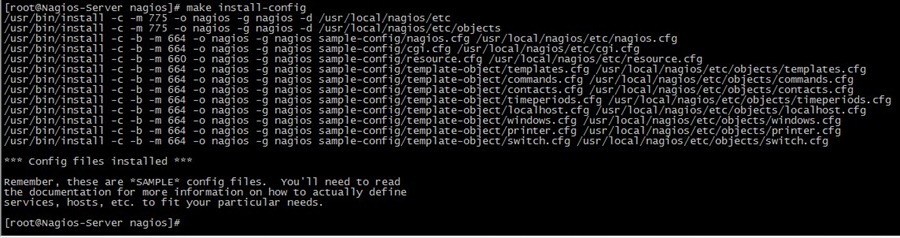
# make install-init

[](http://images.cnitblog.com/blog/370046/201301/30172038-629553af940645d3a1e1b34db39106e7.jpg)

# make install-commandmode

[](http://images.cnitblog.com/blog/370046/201301/30172039-977a570a3633463e905c54ae6b86b8dc.jpg)

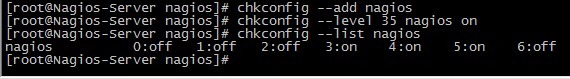
# make install-config

[](http://images.cnitblog.com/blog/370046/201301/30172042-cdce5169280944e3b61f4df6cc33311d.jpg)

# chkconfig --add nagios

# chkconfig --level 35 nagios on

# chkconfig --list nagios

[](http://images.cnitblog.com/blog/370046/201301/30172044-219e4c37980a450c904fb0cb283c73cd.jpg)

### 4 验证程序是否被正确安装

切换目录到安装路径（这里是/usr/local/nagios），看是否存在etc、bin、sbin、share、var 这五个目录，如果存在则可以表明程序被正确的安装到系统了。Nagios 各个目录用途说明如下：

|  |  |
| --- | --- |
| bin | Nagios 可执行程序所在目录 |
| etc | Nagios 配置文件所在目录 |
| sbin | Nagios CGI 文件所在目录，也就是执行外部命令所需文件所在的目录 |
| share | Nagios网页文件所在的目录 |
| libexec | Nagios 外部插件所在目录 |
| var | Nagios 日志文件、lock 等文件所在的目录 |
| var/archives | Nagios 日志自动归档目录 |
| var/rw | 用来存放外部命令文件的目录 |

### 5 安装Nagios 插件

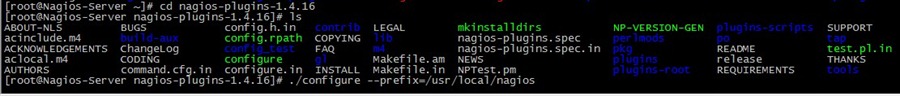
# wget <http://prdownloads.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.16.tar.gz>

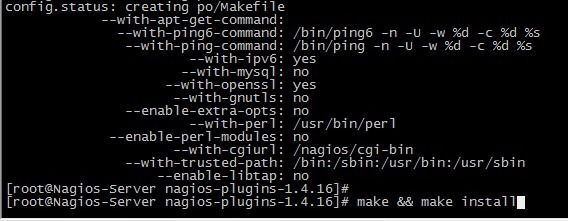
# tar zxvf nagios-plugins-1.4.16.tar.gz

# cd nagios-plugins-1.4.16

# ./configure --prefix=/usr/local/nagios

# make && make install

[](http://images.cnitblog.com/blog/370046/201301/30172045-9f33e94cc4034186b2b81b92370fe259.jpg)

[](http://images.cnitblog.com/blog/370046/201301/30172046-64f935847ba9428d979a2d5c8b0987d2.jpg)

### 6 安装与配置Apache和Php

Apache 和Php 不是安装nagios 所必须的，但是nagios提供了web监控界面，通过web监控界面可以清晰的看到被监控主机、资源的运行状态，因此，安装一个web服务是很必要的。   
需要注意的是，nagios在nagios3.1.x版本以后，配置web监控界面时需要php的支持。这里我们下载的nagios版本为nagios-3.4.3，因此在编译安装完成apache后，还需要编译php模块，这里选取的php版本为php5.4.10。

a. 安装Apache

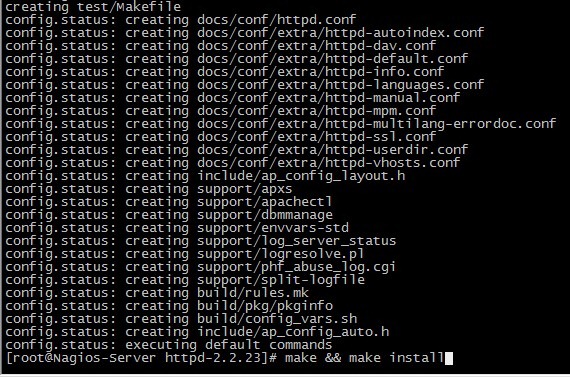
# wget <http://archive.apache.org/dist/httpd/httpd-2.2.23.tar.gz>

# tar zxvf httpd-2.2.23.tar.gz

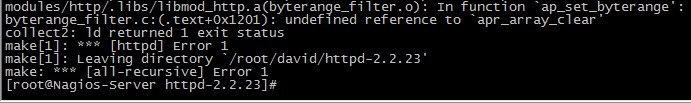
# cd httpd-2.2.23

# ./configure --prefix=/usr/local/apache2

# make && make install

[](http://images.cnitblog.com/blog/370046/201301/30172048-277b391cdc8a4e7da295d6bf9c3dfe6f.jpg)

若出现错误：



则在编译时入加 --with-included-apr 即可解决。

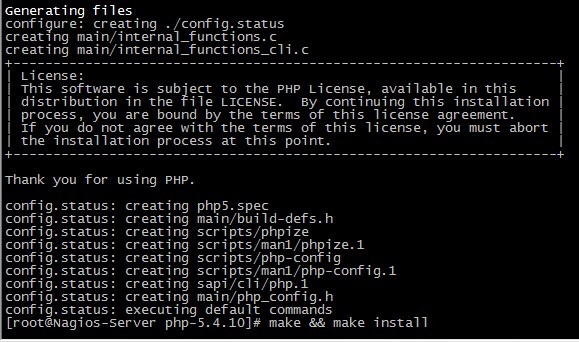
b. 安装Php

# wget <http://cn2.php.net/distributions/php-5.4.10.tar.gz>

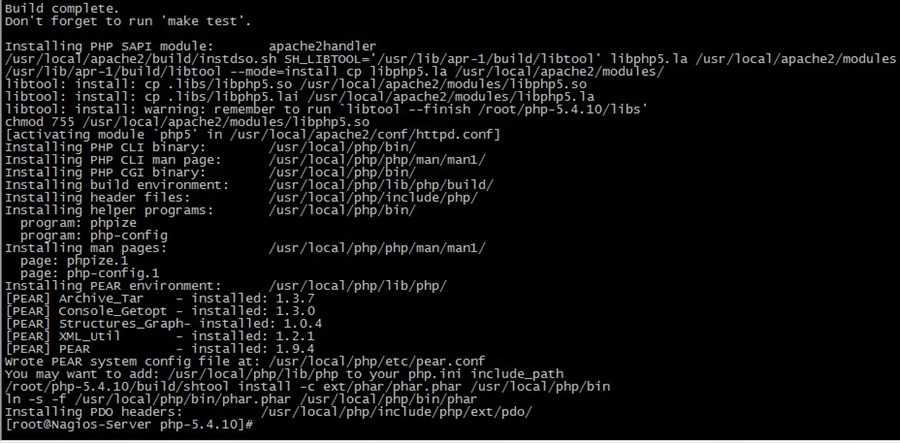
# tar zxvf php-5.4.10.tar.gz

# cd php-5.4.10

# ./configure --prefix=/usr/local/php --with-apxs2=/usr/local/apache2/bin/apxs

[](http://images.cnitblog.com/blog/370046/201301/30172050-6bda75716f614a6ab7f2acf8de422062.jpg)

# make && make install

[](http://images.cnitblog.com/blog/370046/201301/30172054-97af3f5b712041f78d9e44d94ac17bbd.jpg)

c. 配置apache   
找到apache 的配置文件/usr/local/apache2/conf/httpd.conf   
找到：

User daemon

Group daemon

修改为

User nagios

Group nagios

然后找到

<IfModule dir\_module>

　　DirectoryIndex index.html

</IfModule>

修改为

<IfModule dir\_module>

　　DirectoryIndex index.html index.php

</IfModule>

接着增加如下内容：

AddType application/x-httpd-php .php

为了安全起见，一般情况下要让nagios 的web 监控页面必须经过授权才能访问，这需要增加验证配置，即在httpd.conf 文件最后添加如下信息：

[复制代码](javascript:void(0);)

#setting for nagios

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

<Directory "/usr/local/nagios/sbin">

AuthType Basic

Options ExecCGI

AllowOverride None

Order allow,deny

Allow from all

AuthName "Nagios Access"

AuthUserFile /usr/local/nagios/etc/htpasswd //用于此目录访问身份验证的文件

Require valid-user

</Directory>

Alias /nagios "/usr/local/nagios/share"

<Directory "/usr/local/nagios/share">

AuthType Basic

Options None

AllowOverride None

Order allow,deny

Allow from all

AuthName "nagios Access"

AuthUserFile /usr/local/nagios/etc/htpasswd

Require valid-user

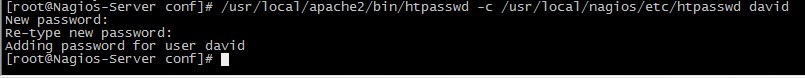
</Directory>

[复制代码](javascript:void(0);)

d. 创建apache目录验证文件

在上面的配置中，指定了目录验证文件htpasswd，下面要创建这个文件：

# /usr/local/apache2/bin/htpasswd -c /usr/local/nagios/etc/htpasswd david

[](http://images.cnitblog.com/blog/370046/201301/30172056-36df7ebedd9d448a8b38a2ccd4633ebe.jpg)   
这样就在/usr/local/nagios/etc 目录下创建了一个htpasswd 验证文件，当通过[http://192.168.1.108/nagios/](http://172.16.1.124/nagios/) 访问时就需要输入用户名和密码了。

e. 查看认证文件的内容

# cat /usr/local/nagios/etc/htpasswd

[26](http://images.cnitblog.com/blog/370046/201301/30172058-a256928a00cf40769bc3a5815401732b.jpg)

f. 启动apache 服务

# /usr/local/apache2/bin/apachectl start

到这里nagios 的安装也就基本完成了，你可以通过web来访问了。



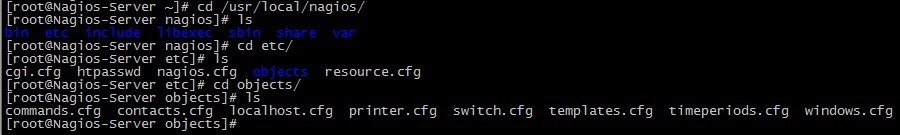


## 二、配置Nagios

Nagios 主要用于监控一台或者多台本地主机及远程的各种信息，包括本机资源及对外的服务等。默认的Nagios 配置没有任何监控内容，仅是一些模板文件。若要让Nagios 提供服务，就必须修改配置文件，增加要监控的主机和服务，下面将详细介绍。

### 1 默认配置文件介绍

Nagios 安装完毕后，默认的配置文件在/usr/local/nagios/etc目录下。

[](http://images.cnitblog.com/blog/370046/201301/30172103-011c4098757a465fa8e4462f89768c84.jpg)

每个文件或目录含义如下表所示：

|  |  |
| --- | --- |
| **文件名或目录名** | **用途** |
| cgi.cfg | 控制CGI访问的配置文件 |
| nagios.cfg | Nagios 主配置文件 |
| resource.cfg | 变量定义文件，又称为资源文件，在些文件中定义变量，以便由其他配置文件引用，如$USER1$ |
| objects | objects 是一个目录，在此目录下有很多配置文件模板，用于定义Nagios 对象 |
| objects/commands.cfg | 命令定义配置文件，其中定义的命令可以被其他配置文件引用 |
| objects/contacts.cfg | 定义联系人和联系人组的配置文件 |
| objects/localhost.cfg | 定义监控本地主机的配置文件 |
| objects/printer.cfg | 定义监控打印机的一个配置文件模板，默认没有启用此文件 |
| objects/switch.cfg | 定义监控路由器的一个配置文件模板，默认没有启用此文件 |
| objects/templates.cfg | 定义主机和服务的一个模板配置文件，可以在其他配置文件中引用 |
| objects/timeperiods.cfg | 定义Nagios 监控时间段的配置文件 |
| objects/windows.cfg | 监控Windows 主机的一个配置文件模板，默认没有启用此文件 |

### 2 配置文件之间的关系

在nagios的配置过程中涉及到的几个定义有：主机、主机组，服务、服务组，联系人、联系人组，监控时间，监控命令等，从这些定义可以看出，nagios各个配置文件之间是互为关联，彼此引用的。

成功配置出一台nagios监控系统，必须要弄清楚每个配置文件之间依赖与被依赖的关系，最重要的有四点：

第一：定义监控哪些主机、主机组、服务和服务组；   
第二：定义这个监控要用什么命令实现；   
第三：定义监控的时间段；   
第四：定义主机或服务出现问题时要通知的联系人和联系人组。

### 3 配置Nagios

 为了能更清楚的说明问题，同时也为了维护方便，建议将nagios各个定义对象创建独立的配置文件：

* 创建hosts.cfg文件来定义主机和主机组
* 创建services.cfg文件来定义服务
* 用默认的contacts.cfg文件来定义联系人和联系人组
* 用默认的commands.cfg文件来定义命令
* 用默认的timeperiods.cfg来定义监控时间段
* 用默认的templates.cfg文件作为资源引用文件

a. templates.cfg文件

nagios主要用于监控主机资源以及服务，在nagios配置中称为对象，为了不必重复定义一些监控对象，Nagios引入了一个模板配置文件，将一些共性的属性定义成模板，以便于多次引用。这就是templates.cfg的作用。

下面详细介绍下templates.cfg文件中每个参数的含义：

[复制代码](javascript:void(0);)

define contact{

name generic-contact ; 联系人名称

service\_notification\_period 24x7 ; 当服务出现异常时，发送通知的时间段，这个时间段"24x7"在timeperiods.cfg文件中定义

host\_notification\_period 24x7 ; 当主机出现异常时，发送通知的时间段，这个时间段"24x7"在timeperiods.cfg文件中定义

service\_notification\_options w,u,c,r ; 这个定义的是“通知可以被发出的情况”。w即warn，表示警告状态，u即unknown，表示不明状态;  
 ; c即criticle，表示紧急状态，r即recover，表示恢复状态;  
 ; 也就是在服务出现警告状态、未知状态、紧急状态和重新恢复状态时都发送通知给使用者。

host\_notification\_options d,u,r ; 定义主机在什么状态下需要发送通知给使用者，d即down，表示宕机状态;  
 ; u即unreachable，表示不可到达状态，r即recovery，表示重新恢复状态。

service\_notification\_commands notify-service-by-email ; 服务故障时，发送通知的方式，可以是邮件和短信，这里发送的方式是邮件;  
 ; 其中“notify-service-by-email”在commands.cfg文件中定义。

host\_notification\_commands notify-host-by-email ; 主机故障时，发送通知的方式，可以是邮件和短信，这里发送的方式是邮件;  
 ; 其中“notify-host-by-email”在commands.cfg文件中定义。

register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL CONTACT, JUST A TEMPLATE!

}

define host{

name generic-host ; 主机名称，这里的主机名，并不是直接对应到真正机器的主机名;  
 ; 乃是对应到在主机配置文件里所设定的主机名。

notifications\_enabled 1 ; Host notifications are enabled

event\_handler\_enabled 1 ; Host event handler is enabled

flap\_detection\_enabled 1 ; Flap detection is enabled

failure\_prediction\_enabled 1 ; Failure prediction is enabled

process\_perf\_data 1 ; 其值可以为0或1，其作用为是否启用Nagios的数据输出功能;  
 ; 如果将此项赋值为1，那么Nagios就会将收集的数据写入某个文件中，以备提取。

retain\_status\_information 1 ; Retain status information across program restarts

retain\_nonstatus\_information 1 ; Retain non-status information across program restarts

notification\_period 24x7 ; 指定“发送通知”的时间段，也就是可以在什么时候发送通知给使用者。

register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!

}

define host{

name linux-server ; 主机名称

use generic-host ; use表示引用，也就是将主机generic-host的所有属性引用到linux-server中来;  
 ; 在nagios配置中，很多情况下会用到引用。

check\_period 24x7 ; 这里的check\_period告诉nagios检查主机的时间段

check\_interval 5 ; nagios对主机的检查时间间隔，这里是5分钟。

retry\_interval 1 ; 重试检查时间间隔，单位是分钟。

max\_check\_attempts 10 ; nagios对主机的最大检查次数，也就是nagios在检查发现某主机异常时，并不马上判断为异常状况;  
 ; 而是多试几次，因为有可能只是一时网络太拥挤，或是一些其他原因，让主机受到了一点影响;  
 ; 这里的10就是最多试10次的意思。

check\_command check-host-alive ; 指定检查主机状态的命令，其中“check-host-alive”在commands.cfg文件中定义。

notification\_period 24x7 ; 主机故障时，发送通知的时间范围，其中“workhours”在timeperiods.cfg中进行了定义;  
 ; 下面会陆续讲到。

notification\_interval 10 ; 在主机出现异常后，故障一直没有解决，nagios再次对使用者发出通知的时间。单位是分钟;  
 ; 如果你觉得，所有的事件只需要一次通知就够了，可以把这里的选项设为0

notification\_options d,u,r ; 定义主机在什么状态下可以发送通知给使用者，d即down，表示宕机状态;  
 ; u即unreachable，表示不可到达状态;  
 ; r即recovery，表示重新恢复状态。

contact\_groups ts ; 指定联系人组，这个“admins”在contacts.cfg文件中定义。

register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!

}

define host{

name windows-server ; The name of this host template

use generic-host ; Inherit default values from the generic-host template

check\_period 24x7 ; By default, Windows servers are monitored round the clock

check\_interval 5 ; Actively check the server every 5 minutes

retry\_interval 1 ; Schedule host check retries at 1 minute intervals

max\_check\_attempts 10 ; Check each server 10 times (max)

check\_command check-host-alive ; Default command to check if servers are "alive"

notification\_period 24x7 ; Send notification out at any time - day or night

notification\_interval 10 ; Resend notifications every 30 minutes

notification\_options d,r ; Only send notifications for specific host states

contact\_groups ts ; Notifications get sent to the admins by default

hostgroups windows-servers ; Host groups that Windows servers should be a member of

register 0 ; DONT REGISTER THIS - ITS JUST A TEMPLATE

}

define service{

name generic-service ; 定义一个服务名称

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;  
 ; (disabling this can lead to major performance problems)

obsess\_over\_service 1 ; We should obsess over this service (if necessary)

check\_freshness 0 ; Default is to NOT check service 'freshness'

notifications\_enabled 1 ; Service notifications are enabled

event\_handler\_enabled 1 ; Service event handler is enabled

flap\_detection\_enabled 1 ; Flap detection is enabled

failure\_prediction\_enabled 1 ; Failure prediction 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

is\_volatile 0 ; The service is not volatile

check\_period 24x7 ; 这里的check\_period告诉nagios检查服务的时间段。

max\_check\_attempts 3 ; nagios对服务的最大检查次数。

normal\_check\_interval 5 ; 此选项是用来设置服务检查时间间隔，也就是说，nagios这一次检查和下一次检查之间所隔的时间;  
 ; 这里是5分钟。

retry\_check\_interval 2 ; 重试检查时间间隔，单位是分钟。

contact\_groups ts ; 指定联系人组

notification\_options w,u,c,r ; 这个定义的是“通知可以被发出的情况”。w即warn，表示警告状态;  
 ; u即unknown，表示不明状态;  
 ; c即criticle，表示紧急状态，r即recover，表示恢复状态;  
 ; 也就是在服务出现警告状态、未知状态、紧急状态和重新恢复后都发送通知给使用者。

notification\_interval 10 ; Re-notify about service problems every hour

notification\_period 24x7 ; 指定“发送通知”的时间段，也就是可以在什么时候发送通知给使用者。

register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL SERVICE, JUST A TEMPLATE!

}

define service{

name local-service ; The name of this service template

use generic-service ; Inherit default values from the generic-service definition

max\_check\_attempts 4 ; Re-check the service up to 4 times in order to determine its final (hard) state

normal\_check\_interval 5 ; Check the service every 5 minutes under normal conditions

retry\_check\_interval 1 ; Re-check the service every minute until a hard state can be determined

register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL SERVICE, JUST A TEMPLATE!

}

[复制代码](javascript:void(0);)

b. resource.cfg文件

resource.cfg是nagios的变量定义文件，文件内容只有一行：

$USER1$=/usr/local/nagios/libexec

其中，变量$USER1$指定了安装nagios插件的路径，如果把插件安装在了其它路径，只需在这里进行修改即可。需要注意的是，变量必须先定义，然后才能在其它配置文件中进行引用。

c. commands.cfg文件

此文件默认是存在的，无需修改即可使用，当然如果有新的命令需要加入时，在此文件进行添加即可。

[复制代码](javascript:void(0);)

#notify-host-by-email命令的定义   
define command{

command\_name notify-host-by-email #命令名称，即定义了一个主机异常时发送邮件的命令。

command\_line /usr/bin/printf "%b" "\*\*\*\*\* Nagios \*\*\*\*\*\n\nNotification Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /bin/mail -s "\*\* $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ \*\*" $CONTACTEMAIL$ #命令具体的执行方式。

}

#notify-service-by-email命令的定义   
define command{

command\_name notify-service-by-email #命令名称，即定义了一个服务异常时发送邮件的命令

command\_line /usr/bin/printf "%b" "\*\*\*\*\* Nagios \*\*\*\*\*\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional Info:\n\n$SERVICEOUTPUT$\n" | /bin/mail -s "\*\* $NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ \*\*" $CONTACTEMAIL$

}  
#check-host-alive命令的定义

define command{

command\_name check-host-alive #命令名称，用来检测主机状态。

command\_line $USER1$/check\_ping -H $HOSTADDRESS$ -w 3000.0,80% -c 5000.0,100% -p 5   
 # 这里的变量$USER1$在resource.cfg文件中进行定义，即$USER1$=/usr/local/nagios/libexec;  
 # 那么check\_ping的完整路径为/usr/local/nagios/libexec/check\_ping;  
 # “-w 3000.0,80%”中“-w”说明后面的一对值对应的是“WARNING”状态，“80%”是其临界值。  
 # “-c 5000.0,100%”中“-c”说明后面的一对值对应的是“CRITICAL”，“100%”是其临界值。  
 # “-p 1”说明每次探测发送一个包。  
 }

define command{

command\_name check\_local\_disk

command\_line $USER1$/check\_disk -w $ARG1$ -c $ARG2$ -p $ARG3$ #$ARG1$是指在调用这个命令的时候，命令后面的第一个参数。

}

define command{

command\_name check\_local\_load

command\_line $USER1$/check\_load -w $ARG1$ -c $ARG2$

}

define command{

command\_name check\_local\_procs

command\_line $USER1$/check\_procs -w $ARG1$ -c $ARG2$ -s $ARG3$

}

define command{

command\_name check\_local\_users

command\_line $USER1$/check\_users -w $ARG1$ -c $ARG2$

}

define command{

command\_name check\_local\_swap

command\_line $USER1$/check\_swap -w $ARG1$ -c $ARG2$

}

define command{

command\_name check\_ftp

command\_line $USER1$/check\_ftp -H $HOSTADDRESS$ $ARG1$

}

define command{

command\_name check\_http

command\_line $USER1$/check\_http -I $HOSTADDRESS$ $ARG1$

}

define command{

command\_name check\_ssh

command\_line $USER1$/check\_ssh $ARG1$ $HOSTADDRESS$

}

define command{

command\_name check\_ping

command\_line $USER1$/check\_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5

}

define command{

command\_name check\_nt

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

}

[复制代码](javascript:void(0);)

d. hosts.cfg文件

此文件默认不存在，需要手动创建，hosts.cfg主要用来指定被监控的主机地址以及相关属性信息，根据实验目标配置如下：

[复制代码](javascript:void(0);)

define host{

use linux-server #引用主机linux-server的属性信息，linux-server主机在templates.cfg文件中进行了定义。

host\_name Nagios-Linux #主机名

alias Nagios-Linux #主机别名

address 192.168.1.111 #被监控的主机地址，这个地址可以是ip，也可以是域名。

}

#定义一个主机组

define hostgroup{

hostgroup\_name bsmart-servers #主机组名称，可以随意指定。

alias bsmart servers #主机组别名

members Nagios-Linux #主机组成员，其中“Nagios-Linux”就是上面定义的主机。   
 }

[复制代码](javascript:void(0);)

**注意：**在/usr/local/nagios/etc/objects 下默认有localhost.cfg 和windows.cfg 这两个配置文件，localhost.cfg 文件是定义监控主机本身的，windows.cfg 文件是定义windows 主机的，其中包括了对host 和相关services 的定义。所以在本次实验中，将直接在localhost.cfg 中定义监控主机（Nagios-Server），在windows.cfg中定义windows 主机（Nagios-Windows）。根据自己的需要修改其中的相关配置，详细如下：

localhost.cfg

[复制代码](javascript:void(0);)

define host{

use linux-server ; Name of host template to use

; This host definition will inherit all variables that are defined

; in (or inherited by) the linux-server host template definition.

host\_name Nagios-Server

alias Nagios-Server

address 127.0.0.1

}

define hostgroup{

hostgroup\_name linux-servers ; The name of the hostgroup

alias Linux Servers ; Long name of the group

members Nagios-Server ; Comma separated list of hosts that belong to this group

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description PING

check\_command check\_ping!100.0,20%!500.0,60%

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description Root Partition

check\_command check\_local\_disk!20%!10%!/

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description Current Users

check\_command check\_local\_users!20!50

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description Total Processes

check\_command check\_local\_procs!250!400!RSZDT

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description Current Load

check\_command check\_local\_load!5.0,4.0,3.0!10.0,6.0,4.0

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description Swap Usage

check\_command check\_local\_swap!20!10

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description SSH

check\_command check\_ssh

notifications\_enabled 0

}

define service{

use local-service ; Name of service template to use

host\_name Nagios-Server

service\_description HTTP

check\_command check\_http

notifications\_enabled 0

}

[复制代码](javascript:void(0);)

windows.cfg

[复制代码](javascript:void(0);)

define host{

use windows-server ; Inherit default values from a template

host\_name Nagios-Windows ; The name we're giving to this host

alias My Windows Server ; A longer name associated with the host

address 192.168.1.113 ; IP address of the host

}

define hostgroup{

hostgroup\_name windows-servers ; The name of the hostgroup

alias Windows Servers ; Long name of the group

}

define service{

use generic-service

host\_name Nagios-Windows

service\_description NSClient++ Version

check\_command check\_nt!CLIENTVERSION

}

define service{

use generic-service

host\_name Nagios-Windows

service\_description Uptime

check\_command check\_nt!UPTIME

}

define service{

use generic-service

host\_name Nagios-Windows

service\_description CPU Load

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

}

define service{

use generic-service

host\_name Nagios-Windows

service\_description Memory Usage

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

}

define service{

use generic-service

host\_name Nagios-Windows

service\_description C:\ Drive Space

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

}

define service{

use generic-service

host\_name Nagios-Windows

service\_description W3SVC

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

}

define service{

use generic-service

host\_name Nagios-Windows

service\_description Explorer

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

}

[复制代码](javascript:void(0);)

e. services.cfg文件

此文件默认也不存在，需要手动创建，services.cfg文件主要用于定义监控的服务和主机资源，例如监控http服务、ftp服务、主机磁盘空间、主机系统负载等等。Nagios-Server 和Nagios-Windows 相关服务已在相应的配置文件中定义，所以这里只需要定义Nagios-Linux 相关服务即可，这里只定义一个检测是否存活的服务来验证配置文件的正确性，其他服务的定义将在后面讲到。

[复制代码](javascript:void(0);)

define service{

use local-service #引用local-service服务的属性值，local-service在templates.cfg文件中进行了定义。

host\_name Nagios-Linux #指定要监控哪个主机上的服务，“Nagios-Server”在hosts.cfg文件中进行了定义。

service\_description check-host-alive #对监控服务内容的描述，以供维护人员参考。

check\_command check-host-alive #指定检查的命令。  
 }

[复制代码](javascript:void(0);)

f. contacts.cfg文件

contacts.cfg是一个定义联系人和联系人组的配置文件，当监控的主机或者服务出现故障，nagios会通过指定的通知方式（邮件或者短信）将信息发给这里指定的联系人或者使用者。

[复制代码](javascript:void(0);)

define contact{

contact\_name David #联系人的名称,这个地方不要有空格

use                             generic-contact #引用generic-contact的属性信息，其中“generic-contact”在templates.cfg文件中进行定义

alias                           Nagios Admin

email david.tang@bsmart.cn

}

define contactgroup{

contactgroup\_name ts #联系人组的名称,同样不能空格

alias Technical Support #联系人组描述

members David #联系人组成员，其中“david”就是上面定义的联系人，如果有多个联系人则以逗号相隔

}

[复制代码](javascript:void(0);)

g. timeperiods.cfg文件

此文件只要用于定义监控的时间段，下面是一个配置好的实例：

[复制代码](javascript:void(0);)

#下面是定义一个名为24x7的时间段，即监控所有时间段

define timeperiod{

timeperiod\_name 24x7 #时间段的名称,这个地方不要有空格

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

}

#下面是定义一个名为workhours的时间段，即工作时间段。

define timeperiod{

timeperiod\_name workhours

alias Normal Work Hours

monday 09:00-17:00

tuesday 09:00-17:00

wednesday 09:00-17:00

thursday 09:00-17:00

friday 09:00-17:00

}

[复制代码](javascript:void(0);)

h. cgi.cfg文件

此文件用来控制相关cgi脚本，如果想在nagios的web监控界面执行cgi脚本，例如重启nagios进程、关闭nagios通知、停止nagios主机检测等，这时就需要配置cgi.cfg文件了。  
由于nagios的web监控界面验证用户为david，所以只需在cgi.cfg文件中添加此用户的执行权限就可以了，需要修改的配置信息如下：

[复制代码](javascript:void(0);)

default\_user\_name=david

authorized\_for\_system\_information=nagiosadmin,david

authorized\_for\_configuration\_information=nagiosadmin,david

authorized\_for\_system\_commands=david

authorized\_for\_all\_services=nagiosadmin,david

authorized\_for\_all\_hosts=nagiosadmin,david

authorized\_for\_all\_service\_commands=nagiosadmin,david

authorized\_for\_all\_host\_commands=nagiosadmin,david

[复制代码](javascript:void(0);)

i. nagios.cfg文件

nagios.cfg默认的路径为/usr/local/nagios/etc/nagios.cfg，是nagios的核心配置文件，所有的对象配置文件都必须在这个文件中进行定义才能发挥其作用，这里只需将对象配置文件在Nagios.cfg文件中进行引用即可。

[复制代码](javascript:void(0);)

log\_file=/usr/local/nagios/var/nagios.log # 定义nagios日志文件的路径

cfg\_file=/usr/local/nagios/etc/objects/commands.cfg # “cfg\_file”变量用来引用对象配置文件，如果有更多的对象配置文件，在这里依次添加即可。  
cfg\_file=/usr/local/nagios/etc/objects/contacts.cfg  
cfg\_file=/usr/local/nagios/etc/objects/hosts.cfg  
cfg\_file=/usr/local/nagios/etc/objects/services.cfg  
cfg\_file=/usr/local/nagios/etc/objects/timeperiods.cfg

cfg\_file=/usr/local/nagios/etc/objects/templates.cfg  
cfg\_file=/usr/local/nagios/etc/objects/localhost.cfg # 本机配置文件  
cfg\_file=/usr/local/nagios/etc/objects/windows.cfg # windows 主机配置文件

object\_cache\_file=/usr/local/nagios/var/objects.cache # 该变量用于指定一个“所有对象配置文件”的副本文件，或者叫对象缓冲文件

precached\_object\_file=/usr/local/nagios/var/objects.precache  
resource\_file=/usr/local/nagios/etc/resource.cfg # 该变量用于指定nagios资源文件的路径，可以在nagios.cfg中定义多个资源文件。

status\_file=/usr/local/nagios/var/status.dat # 该变量用于定义一个状态文件，此文件用于保存nagios的当前状态、注释和宕机信息等。

status\_update\_interval=10 # 该变量用于定义状态文件（即status.dat）的更新时间间隔，单位是秒，最小更新间隔是1秒。

nagios\_user=nagios # 该变量指定了Nagios进程使用哪个用户运行。  
nagios\_group=nagios # 该变量用于指定Nagios使用哪个用户组运行。

check\_external\_commands=1 # 该变量用于设置是否允许nagios在web监控界面运行cgi命令;  
 # 也就是是否允许nagios在web界面下执行重启nagios、停止主机/服务检查等操作;  
 # “1”为运行，“0”为不允许。

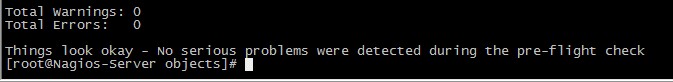
command\_check\_interval=10s # 该变量用于设置nagios对外部命令检测的时间间隔，如果指定了一个数字加一个"s"(如10s);  
 # 那么外部检测命令的间隔是这个数值以秒为单位的时间间隔;  
 # 如果没有用"s"，那么外部检测命令的间隔是以这个数值的“时间单位”的时间间隔。  
interval\_length=60 # 该变量指定了nagios的时间单位，默认值是60秒，也就是1分钟;  
 # 即在nagios配置中所有的时间单位都是分钟。

[复制代码](javascript:void(0);)

### 4 验证Nagios 配置文件的正确性

Nagios 在验证配置文件方面做的非常到位，只需通过一个命令即可完成：

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



Nagios提供的这个验证功能非常有用，在错误信息中通常会打印出错误的配置文件以及文件中的哪一行，这使得nagios的配置变得非常容易，报警信息通常是可以忽略的，因为一般那些只是建议性的。

看到上面这些信息就说明没问题了，然后启动Nagios 服务。

## 三、Nagios的启动与停止

### 1 启动Nagios

a. 通过初始化脚本启动nagios

# /etc/init.d/nagios start  
or  
# service nagios start

b. 手工方式启动nagios

通过nagios命令的“-d”参数来启动nagios守护进程：

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

### 2 重启Nagios

当修改了配置文件让其生效时，需要重启/重载Nagios服务。

a. 通过初始化脚本来重启nagios

# /etc/init.d/nagios reload  
or  
# /etc/init.d/nagios restart  
or  
# service nagios restart

b. 通过web监控页重启nagios

可以通过web监控页的 "Process Info" -> "Restart the Nagios process"来重启nagios



c. 手工方式平滑重启

# kill -HUP <nagios\_pid>

### 3 停止Nagios

a. 通过初始化脚本关闭nagios服务

# /etc/init.d/nagios stop  
or  
# service nagios stop

b. 通过web监控页停止nagios

可以通过web监控页的 "Process Info" -> "Shutdown the Nagios process"来停止nagios



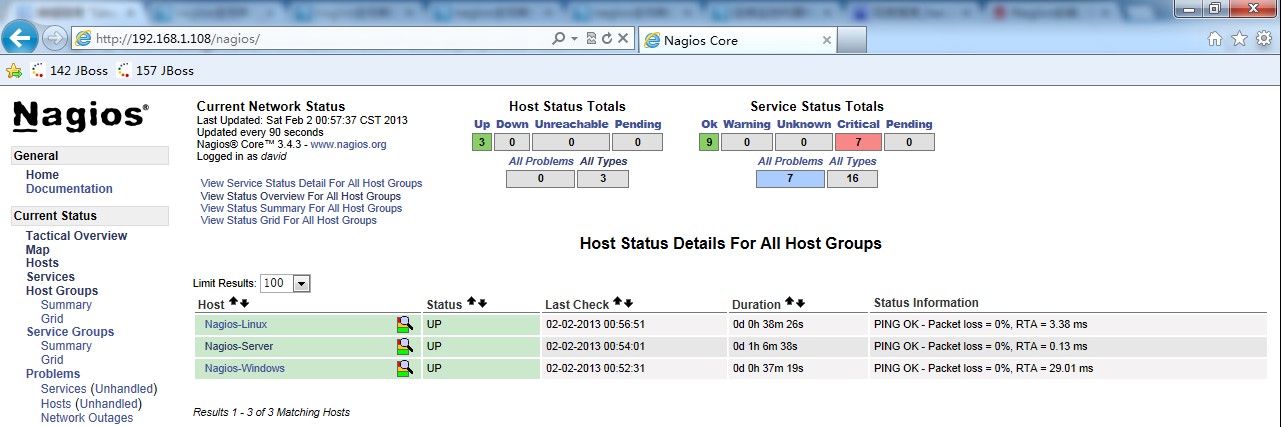
c. 手工方式停止Nagios

# kill <nagios\_pid>

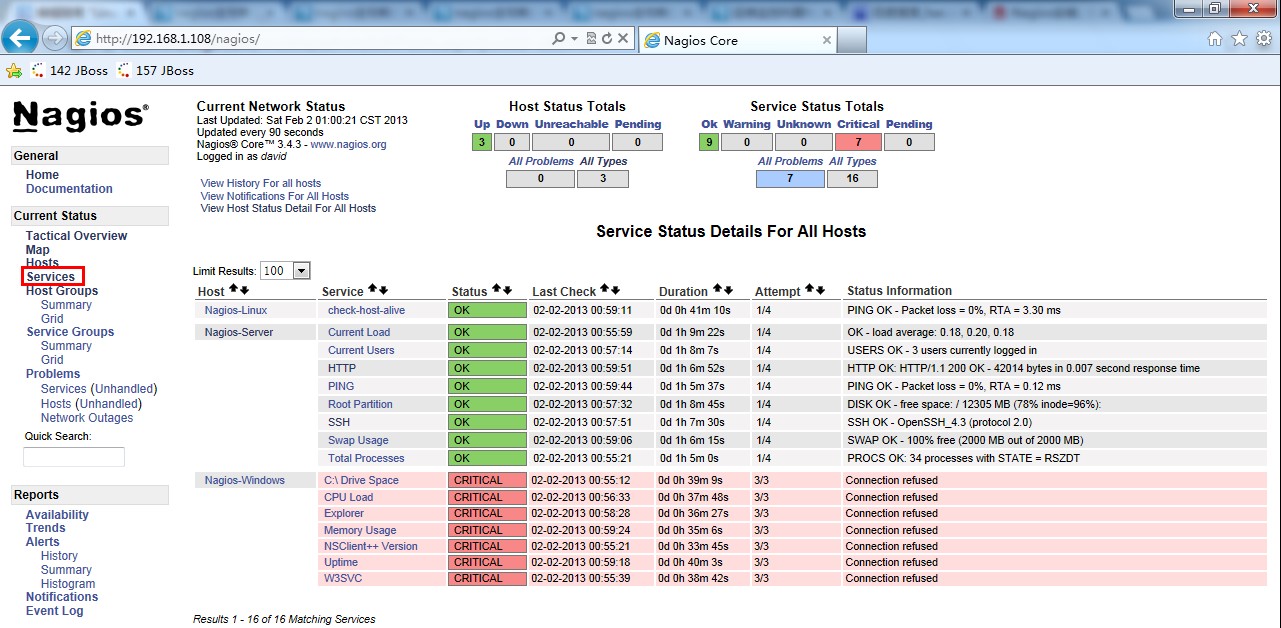
## 四、查看初步配置情况

### 1 启动完成之后，登录Nagios Web监控页[http://192.168.1.108/nagios/](http://172.16.1.124/nagios/) 查看相关信息。

### 2 点击左面的Current Status -> Hosts 可以看到所定义的三台主机已经全部UP了。



### 3 点击Current Status -> Services 查看服务监控情况。

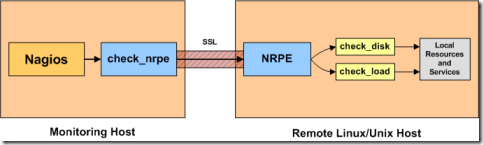


看到Nagios-Linux和Nagios-Server的服务状态已经OK了，但是Nagios-Windows的服务状态为CRITICAL，Status Information 提示Connection refused。因为Nagios-Windows上还未安装插件，内部服务还无法查看，所以出现这种情况。将在下面具体讲解。

## 五、利用NRPE监控远程Linux上的“本地信息”

上面已经对远程Linux 主机是否存活做了监控，而判断远程机器是否存活，我们可以使用ping 工具对其监测。还有一些远程主机服务，例如ftp、ssh、http，都是对外开放的服务，即使不用Nagios，我们也可以试的出来，随便找一台机器看能不能访问这些服务就行了。但是对于像磁盘容量，cpu负载这样的“本地信息”，Nagios只能监测自己所在的主机，而对其他的机器则显得有点无能为力。毕竟没得到被控主机的适当权限是不可能得到这些信息的。为了解决这个问题，nagios有这样一个附加组件--“NRPE”，用它就可以完成对Linux 类型主机"本地信息”的监控。

### 1 NRPE 工作原理

[](http://images.cnitblog.com/blog/370046/201301/30172018-ee257c664e12472590da4a3b503be33e.png)

NRPE 总共由两部分组成：

* check\_nrpe 插件，位于监控主机上
* NRPE daemon，运行在远程的Linux主机上(通常就是被监控机)

按照上图，整个的监控过程如下：

当Nagios 需要监控某个远程Linux 主机的服务或者资源情况时：

1. Nagios 会运行check\_nrpe 这个插件，告诉它要检查什么；
2. check\_nrpe 插件会连接到远程的NRPE daemon，所用的方式是SSL；
3. NRPE daemon 会运行相应的Nagios 插件来执行检查；
4. NRPE daemon 将检查的结果返回给check\_nrpe 插件，插件将其递交给nagios做处理。

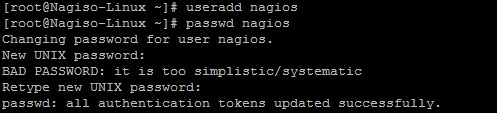
注意：NRPE daemon 需要Nagios 插件安装在远程的Linux主机上，否则，daemon不能做任何的监控。

### 2 在被监控机（Nagios-Linux）上

a. 增加用户&设定密码

# useradd nagios

# passwd nagios



b. 安装Nagios 插件

# tar zxvf nagios-plugins-1.4.16.tar.gz

# cd nagios-plugins-1.4.16

# ./configure --prefix=/usr/local/nagios

# make && make install

这一步完成后会在/usr/local/nagios/下生成三个目录include、libexec和share。

http://images.cnitblog.com/blog/370046/201302/02152814-8ad7b743e94d47e4a273d26f17fc039b.jpg

修改目录权限

# chown nagios.nagios /usr/local/nagios

# chown -R nagios.nagios /usr/local/nagios/libexec

http://images.cnitblog.com/blog/370046/201302/02152955-bba448f8c550421199e95f5abcd2a367.jpg

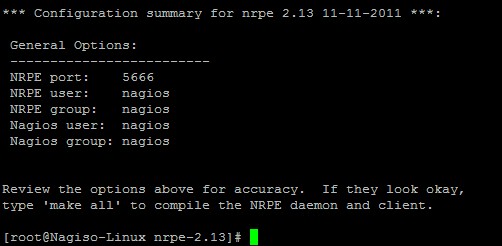
c. 安装NRPE

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

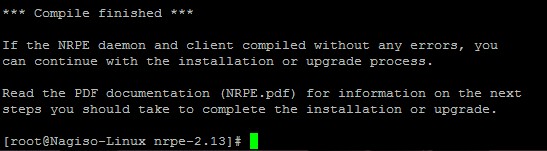
# tar zxvf nrpe-2.13.tar.gz

# cd nrpe-2.13

# ./configure



# make all



接下来安装NPRE插件，daemon和示例配置文件。

c.1 安装check\_nrpe 这个插件

# make install-plugin

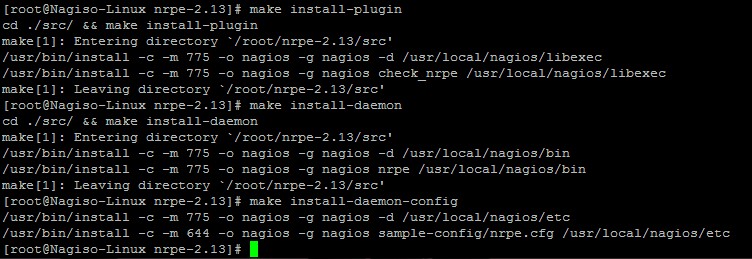
监控机需要安装check\_nrpe 这个插件，被监控机并不需要，我们在这里安装它只是为了测试目的。

c.2 安装deamon

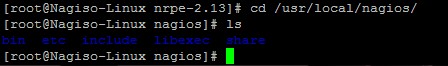
# make install-daemon

c.3 安装配置文件

# make install-daemon-config



现在再查看nagios 目录就会发现有5个目录了



按照安装文档的说明，是将NRPE deamon作为xinetd下的一个服务运行的。在这样的情况下xinetd就必须要先安装好，不过一般系统已经默认安装了。

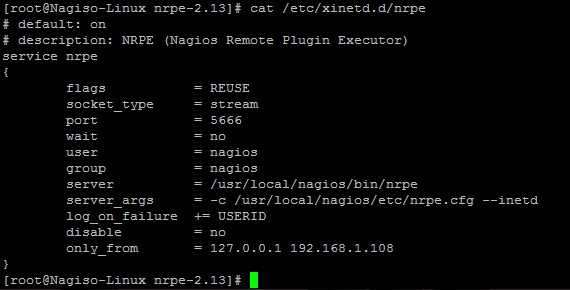
d. 安装xinted 脚本

# make install-xinetd

http://images.cnitblog.com/blog/370046/201302/02154307-bee4a49f8d87420a87899cd9c840c6d9.jpg

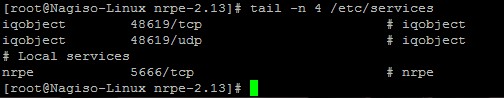
可以看到创建了这个文件/etc/xinetd.d/nrpe。

编辑这个脚本：



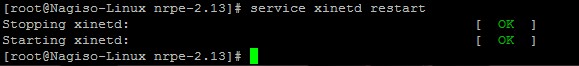
在only\_from 后增加监控主机的IP地址。

编辑/etc/services 文件，增加NRPE服务

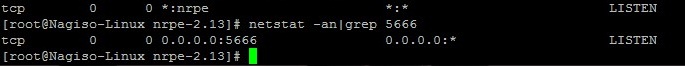


重启xinted 服务

# service xinetd restart



查看NRPE 是否已经启动



可以看到5666端口已经在监听了。

e. 测试NRPE是否则正常工作

使用上面在被监控机上安装的check\_nrpe 这个插件测试NRPE 是否工作正常。

# /usr/local/nagios/libexec/check\_nrpe -H localhost

会返回当前NRPE的版本

http://images.cnitblog.com/blog/370046/201302/02160042-6ae7c0101c264ff8b8076192ace36a23.jpg

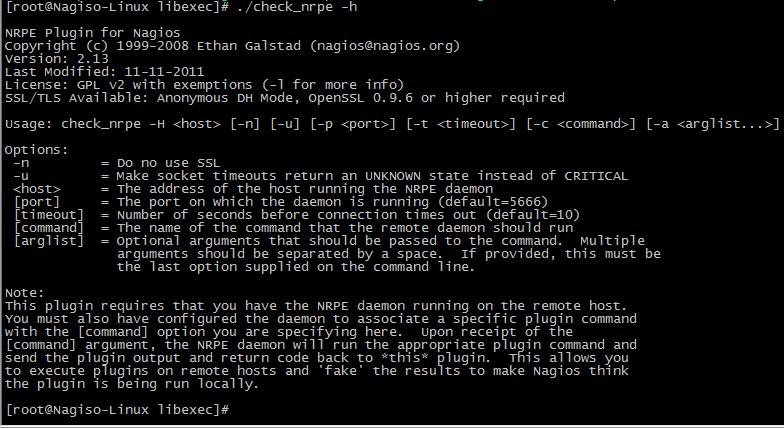
也就是在本地用check\_nrpe连接nrpe daemon是正常的。

注：为了后面工作的顺利进行，注意本地防火墙要打开5666能让外部的监控机访问。

f. check\_nrpe 命令用法

查看check\_nrpe 命令用法

# /usr/local/nagios/libexec/check\_nrpe –h



可以看到用法是：

check\_nrpe –H 被监控的主机 -c 要执行的监控命令

注意：-c 后面接的监控命令必须是nrpe.cfg 文件中定义的。也就是NRPE daemon只运行nrpe.cfg中所定义的命令。

g. 查看NRPE的监控命令

# cd /usr/local/nagios/etc

# cat nrpe.cfg |grep -v "^#"|grep -v "^$"

[复制代码](javascript:void(0);)

[root@Nagiso-Linux etc]# cat nrpe.cfg |grep -v "^#"|grep -v "^$"

log\_facility=daemon

pid\_file=/var/run/nrpe.pid

server\_port=5666

nrpe\_user=nagios

nrpe\_group=nagios

allowed\_hosts=127.0.0.1

dont\_blame\_nrpe=0

debug=0

command\_timeout=60

connection\_timeout=300

command[check\_users]=/usr/local/nagios/libexec/check\_users -w 5 -c 10

command[check\_load]=/usr/local/nagios/libexec/check\_load -w 15,10,5 -c 30,25,20

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

command[check\_zombie\_procs]=/usr/local/nagios/libexec/check\_procs -w 5 -c 10 -s Z

command[check\_total\_procs]=/usr/local/nagios/libexec/check\_procs -w 150 -c 200

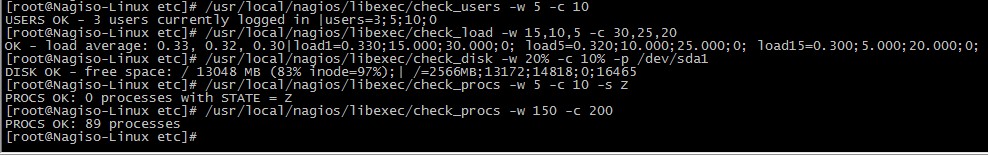
[root@Nagiso-Linux etc]#

[复制代码](javascript:void(0);)

红色部分是命令名，也就是check\_nrpe 的-c 参数可以接的内容，等号 “=” 后面是实际执行的插件程序（这与commands.cfg 中定义命令的形式十分相似，只不过是写在了一行）。也就是说check\_users 就是等号后面/usr/local/nagios/libexec/check\_users -w 5 -c 10 的简称。

我们可以很容易知道上面这5行定义的命令分别是检测登陆用户数，cpu负载，sda1的容量，僵尸进程，总进程数。各条命令具体的含义见插件用法（执行“插件程序名 –h”）。

由于-c 后面只能接nrpe.cfg 中定义的命令，也就是说现在我们只能用上面定义的这五条命令。我们可以在本机实验一下。



### 3 在监控主机（Nagios-Server）上

之前已经将Nagios运行起来了，现在要做的事情是：

* 安装check\_nrpe 插件；
* 在commands.cfg 中创建check\_nrpe 的命令定义，因为只有在commands.cfg 中定义过的命令才能在services.cfg 中使用；
* 创建对被监控主机的监控项目；

### 3.1 安装check\_nrpe 插件

# tar zxvf nrpe-2.13.tar.gz

# cd nrpe-2.13

# ./configure

# make all

# make install-plugin

只运行这一步就行了，因为只需要check\_nrpe插件。

在Nagios-Linux 上我们已经装好了nrpe，现在我们测试一下监控机使用check\_nrpe 与被监控机运行的nrpe daemon之间的通信。

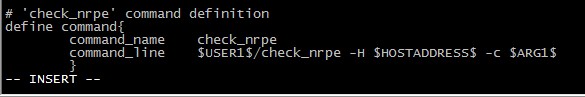
http://images.cnitblog.com/blog/370046/201302/02165718-6aa53e921dc2432a9738a5638aaa17e3.jpg

看到已经正确返回了NRPE的版本信息，说明一切正常。

### 3.2 在commands.cfg中增加对check\_nrpe的定义

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

在最后面增加如下内容：



意义如下：

[复制代码](javascript:void(0);)

# 'check\_nrpe' command definition

define command{

command\_name check\_nrpe # 定义命令名称为check\_nrpe,在services.cfg中要使用这个名称.

command\_line $USER1$/check\_nrpe -H $HOSTADDRESS$ -c $ARG1$ #这是定义实际运行的插件程序.  
 # 这个命令行的书写要完全按照check\_nrpe这个命令的用法,不知道用法的就用check\_nrpe –h查看.

}

[复制代码](javascript:void(0);)

-c 后面带的$ARG1$ 参数是传给nrpe daemon 执行的检测命令，之前说过了它必须是nrpe.cfg 中所定义的那5条命令中的其中一条。在services.cfg 中使用check\_nrpe 的时候要用 “!” 带上这个参数。

### 3.3 定义对Nagios-Linux 主机的监控

下面就可以在services.cfg 中定义对Nagios-Linux 主机的监控了。

[复制代码](javascript:void(0);)

define service{

use local-service

host\_name Nagios-Linux

service\_description Current Load

check\_command check\_nrpe!check\_load

}

define service{

use local-service

host\_name Nagios-Linux

service\_description Check Disk sda1

check\_command check\_nrpe!check\_sda1

}

define service{

use local-service

host\_name Nagios-Linux

service\_description Total Processes

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

}

define service{

use local-service

host\_name Nagios-Linux

service\_description Current Users

check\_command check\_nrpe!check\_users

}

define service{

use local-service

host\_name Nagios-Linux

service\_description Check Zombie Procs

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

}

[复制代码](javascript:void(0);)

还有一个任务是要监控Nagios-Linux 的swap 使用情况。但是在nrpe.cfg 中默认没有定义这个监控功能的命令。怎么办？手动在nrpe.cfg 中添加，也就是自定义NRPE命令。

现在我们要监控swap 分区，如果空闲空间小于20%则为警告状态 -> warning；如果小于10%则为严重状态 -> critical。我们可以查得需要使用check\_swap插件，完整的命令行应该是下面这样。

# /usr/local/nagios/libexec/check\_swap -w 20% -c 10%  
  
在被监控机（Nagios-Linux）上增加check\_swap 命令的定义

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

增加下面这一行

command[check\_swap]=/usr/local/nagios/libexec/check\_swap -w 20% -c 10%

我们知道check\_swap 现在就可以作为check\_nrpe 的-c 的参数使用了

修改了配置文件，当然要重启。

如果你是以独立的daemon运行的nrpe，那么需要手动重启；如果你是在xinetd 下面运行的，则不需要。

由于本实验中nrpe 是xinetd 下运行的，所以不需要重启服务。  
  
在监控机（Nagios-Server）上增加这个check\_swap 监控项目

[复制代码](javascript:void(0);)

define service{

use local-service

host\_name Nagios-Linux

service\_description Check Swap

check\_command check\_nrpe!check\_swap

}

[复制代码](javascript:void(0);)

同理，Nagios-Linux 上我还开启了http 服务，需要监控一下，按照上面的做法，在被监控机（Nagios-Linux）上增加check\_http 命令的定义

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

增加下面这一行

command[check\_http]=/usr/local/nagios/libexec/check\_http -I 127.0.0.1

在监控机（Nagios-Server）上增加check\_http 监控项目

[复制代码](javascript:void(0);)

define service{

use local-service

host\_name Nagios-Linux

service\_description HTTP

check\_command check\_nrpe!check\_http

}

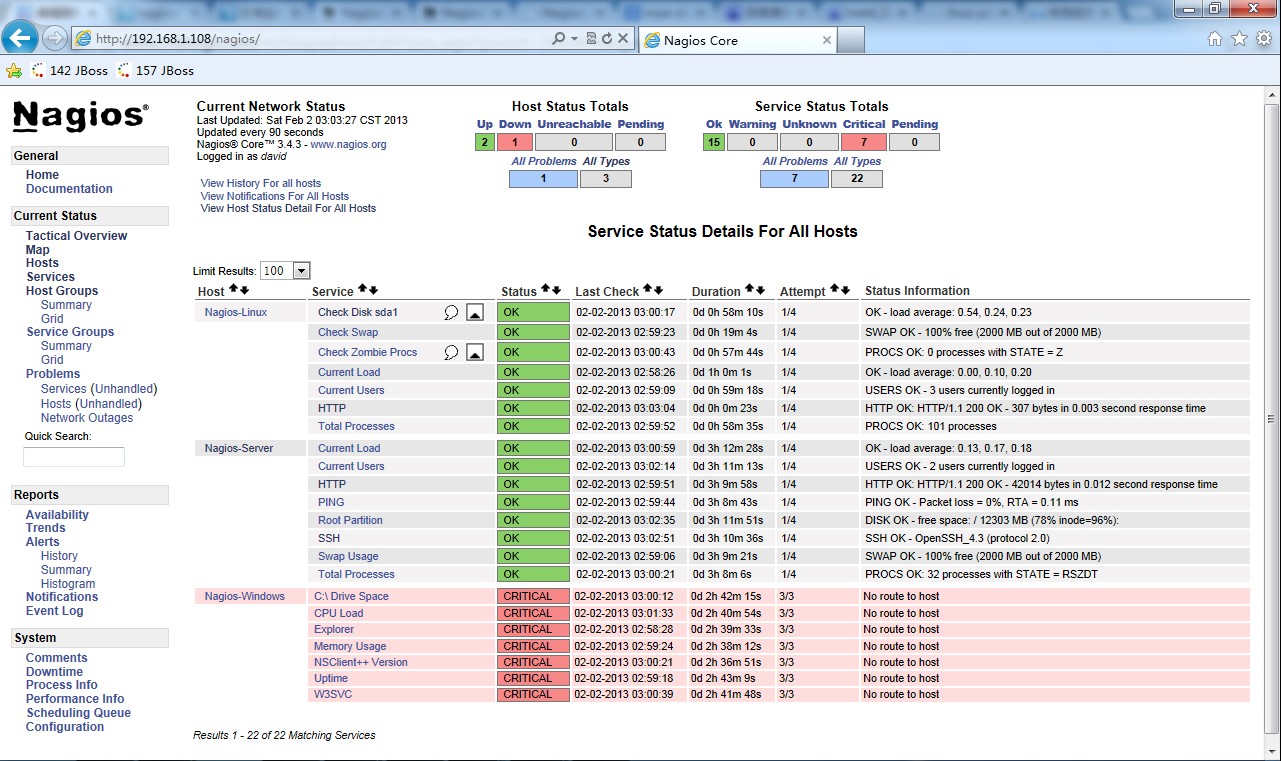
[复制代码](javascript:void(0);)

所有的配置文件已经修改好了，现在重启Nagios。

# service nagios restart

### 3.4 查看配置情况

登录Nagios Web监控页[http://192.168.1.108/nagios/](http://172.16.1.124/nagios/) 查看相关信息。

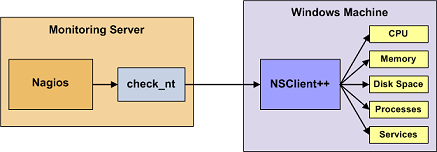


可以看到，对于Nagios-Server 和Nagios-Linux 上的相关服务的监控已经成功了，还有Nagios-Windows 上的服务还没有定义，下面讲到。

## 六、利用NSClient++监控远程Windows上的“本地信息”

在Nagios的libexec下有check\_nt这个插件，它就是用来检查windows机器的服务的。其功能类似于check\_nrpe。不过还需要搭配另外一个软件NSClient++，它则类似于NRPE。

NSClient++的原理如下图



可以看到NSClient与nrpe最大的区别就是：

* 被监控机上安装有nrpe，并且还有插件，最终的监控是由这些插件来进行的。当监控主机将监控请求发给nrpe后，nrpe调用插件来完成监控。
* NSClient++则不同，被监控机上只安装NSClient，没有任何的插件。当监控主机将监控请求发给NSClient++后，NSClient直接完成监控，所有的监控是由NSClient完成的。

这也说明了NSClient++的一个很大的问题：不灵活、没有可扩展性。它只能完成自己本身包含的监控操作，不能由一些插件来扩展。好在NSClient++已经做的不错了，基本上可以完全满足我们的监控需求。

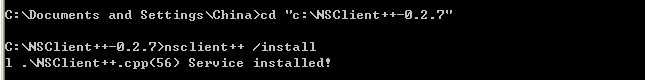
### 1 安装NSClient++

从http://www.nsclient.org/nscp/downloads 下载NSClient++-0.2.7.zip

解压到C盘根目录。

打开cmd 切换到c:\NSClient++-0.2.7

执行nsclient++ /install 进行安装



执行nsclient++ SysTray （注意大小写），这一步是安装系统托盘，时间稍微有点长。

http://images.cnitblog.com/blog/370046/201302/02222739-b0076ad25fbe40d99ff7094f43418657.jpg

在运行里面输入services.msc 打开“服务”



看到下图就说明NSClient服务已经安装上了

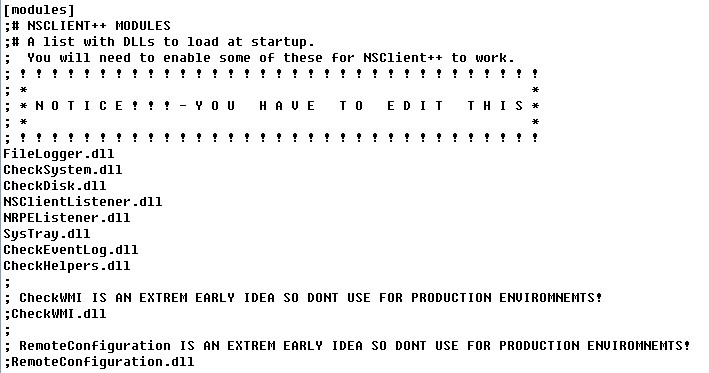
http://images.cnitblog.com/blog/370046/201302/02222842-718e9a7851a348809b4d00244e369544.jpg

双击打开，点“登录”标签，在“允许服务与桌面交互”前打勾。



编辑c:\NSClient++-0.2.7下的NSC.ini文件。

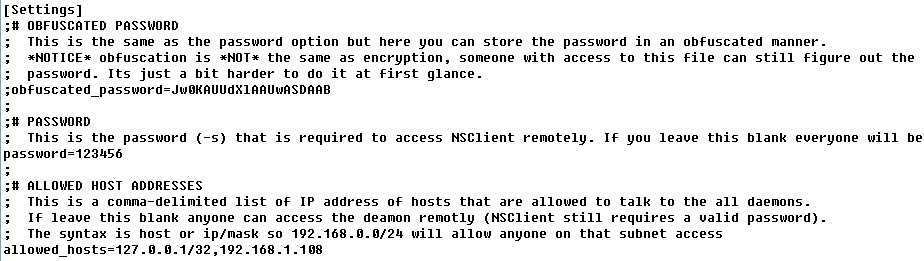
将 [modules]部分的所有模块前面的注释都去掉，除了CheckWMI.dll 和 RemoteConfiguration.dll 这两个。



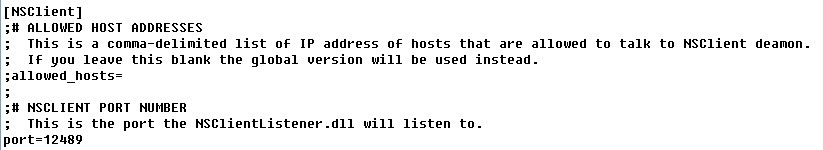
在[Settings]部分设置'password'选项来设置密码，作用是在nagios连接过来时要求提供密码。这一步是可选的，我这里设置为'123456'。

将[Settings]部分'allowed\_hosts'选项的注释去掉，并且加上运行nagios的监控主机的IP。各IP之间以逗号相隔。这个地方是支持子网的，如果写成192.168.1.0/24则表示该子网内的所有机器都可以访问。如果这个地方是空白则表示所有的主机都可以连接上来。

注意是[Settings]部分的，因为[NSClient]部分也有这个选项。



必须保证[NSClient]的'port'选项并没有被注释，并且它的值是'12489'，这是NSClient的默认监听端口。



在cmd 中执行nsclient++ /start启动服务，注意所在目录是c:\NSClient++-0.2.7

http://images.cnitblog.com/blog/370046/201302/02223713-a55666da26bb47338db68d06d04db0ff.jpg

这时在桌面右下角的系统托盘处会出现一个黄色的M字样的图标

http://images.cnitblog.com/blog/370046/201302/02223741-eea5809e0c0a4a75920308e0530a679e.jpg

查看服务

http://images.cnitblog.com/blog/370046/201302/02223800-aa5a5cf0f907417e804e1bf42e3ad95c.jpg

已经正常启动了。

注意服务默认设的是“自动”，也就是说是开机自动启动的。

在cmd 里面执行netstat –an 可以看到已经开始监听tcp的12489端口了。

http://images.cnitblog.com/blog/370046/201302/02223852-f70c360e03204553a28086040839174d.jpg

这样外部就可以访问了吗？

错！

防火墙也要打开tcp的12489端口，否则nagios 检查此服务的时候会报错。

这样被监控机的配置就搞定了，它就等待nagios 发出某个监控请求，然后它执行请求将监控的结果发回到nagios监控主机上。

之前已经在监控主机（Nagios-Server）上对Windows 主机的监控做了配置，但是commands.cfg 中默认没有设置密码项，所以要修改一下，增加"-s 123456"，如下：

# 'check\_nt' command definition

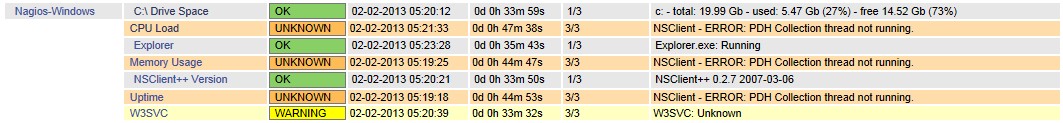
define command{

command\_name check\_nt

command\_line $USER1$/check\_nt -H $HOSTADDRESS$ -p 12489 -s 123456 -v $ARG1$ $ARG2$

}

现在打开Nagios Web监控页便可查看到相关信息了。



可以看到有错误：NSClient - ERROR: PDH Collection thread not running.

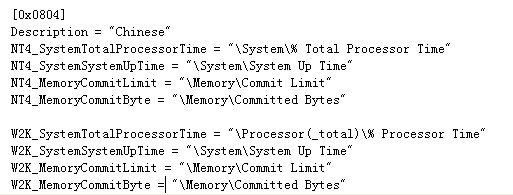
Google 一下，是由于操作系统语言的问题，好像NSClient 默认支持的语言并不多，具体可以百度一下。

查看NSClient的日志C:\NSClient++-0.2.7\nsclient.log，信息如下：

2013-02-02 22:05:30: error:.\PDHCollector.cpp:98: You need to manually configure performance counters!

需要手动配置performance counters。

打开C:\NSClient++-0.2.7\counters.defs文件，复制文件里面"English US"那部分内容，粘贴到counters.defs 文件的最后，修改Description = "Chinese"。



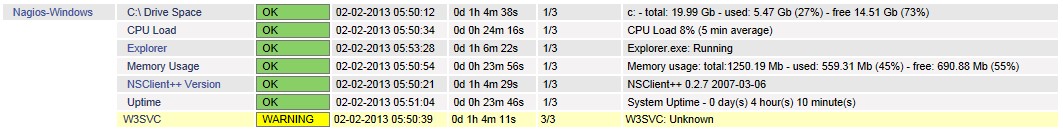
修改完之后，在mmc中重启NSClient 服务。

然后查看日志，内容如下：

http://images.cnitblog.com/blog/370046/201302/03004023-10d2dacf5c6a4d57bbb1c723972a67bd.jpg

在正常执行了。

打开Nagios Web监控页查看。

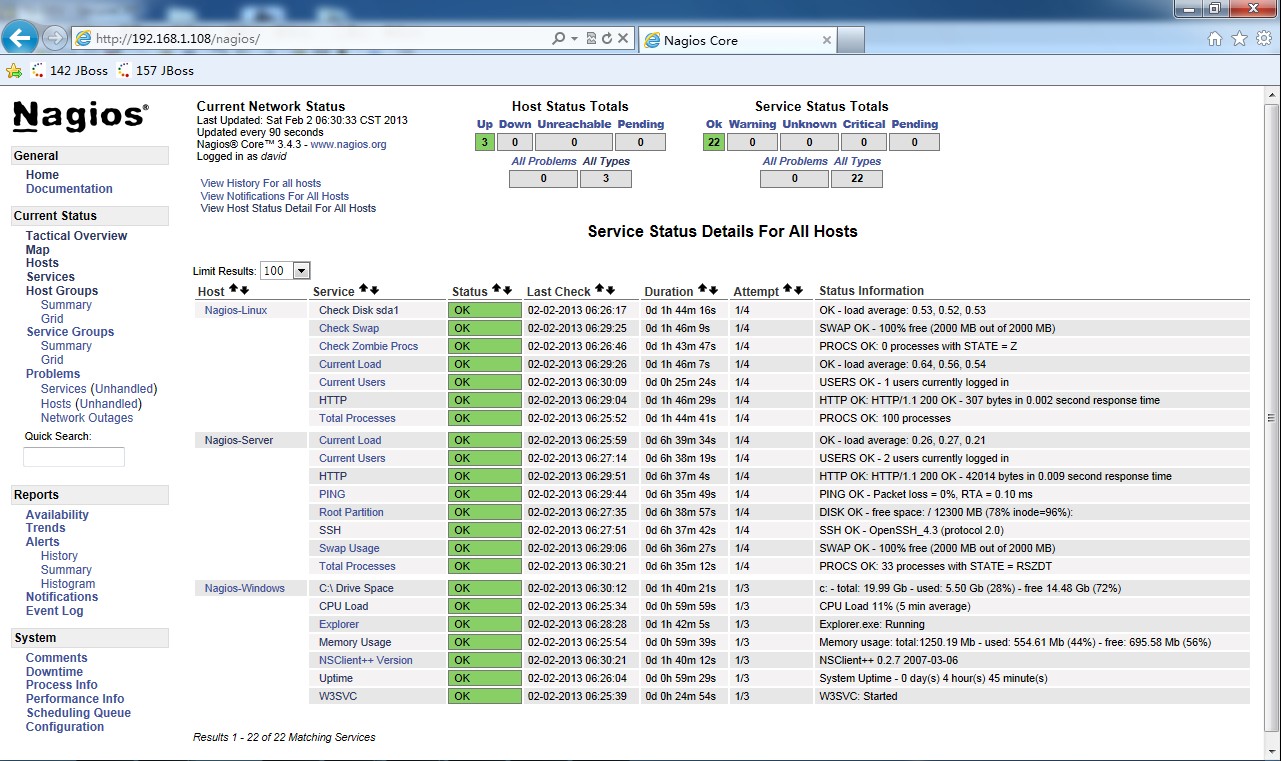


执行成功，但是W3SVC服务为Unknown 状态。查资料，需要开启Windows 的IIS服务。

打开“控制面板”进行安装。



安装完毕后，再到Nagios Web监控页查看，全部监控正常。



## 七、Nagios邮件报警的配置

### 1 安装sendmail 组件

首先要确保sendmail 相关组件的完整安装，我们可以使用如下的命令来完成sendmail 的安装：

# yum install -y sendmail\*

yum install mailutils

然后重新启动sendmail服务：

# service sendmail restart

然后发送测试邮件，验证sendmail的可用性：

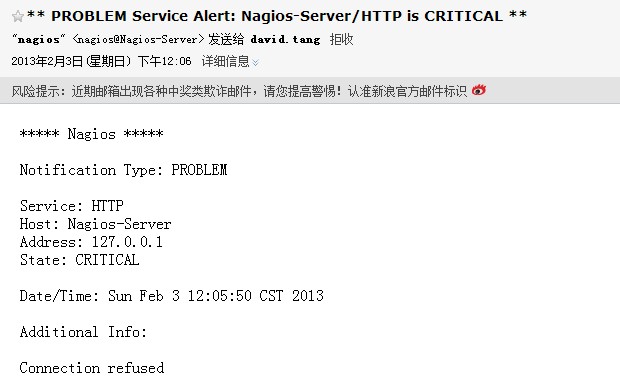
# echo "Hello World" | mail david.tang@bsmart.cn

### 2 邮件报警的配置

在上面我们已经简单配置过了/usr/local/nagios/etc/objects/contacts.cfg 文件，Nagios 会将报警邮件发送到配置文件里的E-mail 地址。

### 3 Nagios 通知

PROBLEM



RECOVERY



Linux下Nagios安装配置完毕。