

Nagios 安装手册

○. 废话.....	4
一. 安装 Nagios 及插件	4
1.1. 准备工作.....	4
1.2. 建立相关帐号.....	4
1.3. 编译与安装 Nagios	5
1.4. 配置 WEB 接口	5
1.5. 编译并安装 Nagios 插件.....	5
1.6. 启动 Nagios	6
1.7. 更改 SELinux 设置.....	6
1.7.1. 临时性改动.....	7
1.7.2. 永久性变更.....	7
1.7.3. 只针对 Nagios 变更.....	7
1.8. 登录 WEB 接口	7
1.9. 其他注意事项.....	7
1.10. 模板继承文件（非常重要）	8
二. 定义联系人.....	9
三. 监控 Windows 主机.....	10
3.1. Windows 服务器的准备工作.....	10
3.2. 允许 Nagios 主机监控 Windows.....	11
3.3. Nagios 监控 Windows 主机的进程	12
3.3.1. 监控单台主机多项指标及示例.....	12
3.3.2. 监控多台主机进程示例.....	13
3.3.3. 监控页面显示监控结果.....	15
3.3.4. 使用 escalation 逐级通知联系人.....	16
四. 监控 Linux 主机（NRPE）	16
4.1. nagios 主机的安装	17
4.2. 被监控目标的安装和测试.....	17
4.2.1. 安装.....	17
4.2.2. 测试.....	18
4.3. 被控端命令的定义.....	18
4.4. 将被控目标加入 nagios 监控.....	19
五. 监控网络流量.....	20
5.1. 服务端安装 SNMP.....	20
5.2. check_snmp 监控设备启动时间	20
5.3. check_traffic 安装检测流量插件	20
5.3.1. 安装 check_traffic.....	21
5.3.2. 将 check_traffic 加入到 nagios 中.....	21
六. 告警消息的发布.....	22
6.1 Email 方式发布告警	22
6.1.1. 实现.....	22
6.1.2. 邮件测试.....	23

6.2. 短信猫发布告警	24
6.2.1. 安装 Linux 下的串口工具 minicom.....	24
6.2.2. 用 minicom 测试短信猫.....	24
6.2.3. 安装短信工具 Gnokii.....	26
6.2.4. 将 gnokii 的短信功能绑定到 Nagios	27
6.3. 飞信发布告警.....	29
6.3.1. 注册飞信	29
6.3.2. 下载飞信机器人并安装.....	29
6.3.3. 测试飞信	29
6.3.4. 授权 sms 文件可被 nagios 执行	30
6.3.5. 绑定飞信功能到 Nagios.....	30
七. WEB 授权.....	31
7.1. 定义联系人	31
7.2. 为联系人添加 WEB 登录帐号。	31
7.3. 修改 cgi 为新增帐号授权	31
7.4. 帐户登录.....	32
八. 用 MRTG 查看历史状态	32
8.1. 准备工作已完善的安装方法.....	32
8.2. 准备工作不完善的安装方法.....	32
8.3. 历史数据查看.....	33
九. 用 PNP 绘制历史曲线	34
9.1 准备工作	34
9.2 安装 PNP.....	34
9.2.1. 安装 cgilib	34
9.2.2. 安装 rrdtool.....	34
9.2.3. 安装 pnp.....	35
9.2.4. 让 apache 的首页默认页支持 php 格式.....	36
9.3 将绘图功能加入 nagios	36
9.3.1. 编辑 nagios.cfg.....	36
9.3.2. 修改绘图命令	36
9.3.3. 修改监控目标主机文件.....	37
9.3.4. 自定义图表时间范围.....	38
十. 关于浏览器	39
十一. Nagios 3.1.2 的安装.....	40
十二.Nagios 的分布式部署（NSCA）	40
12.1 安装.....	40
12.2 客户端配置.....	41
12.2.1 复制文件	41
12.2.2 修改配置文件	41
12.2.3 新增命令	41
12.2.4 修改客户端配置.....	42
12.3 服务端的安装.....	43
12.3.1 复制文件	43
12.3.2 修改 nsca.cfg 并测试客户端连通性	43

12.3.3 复制客户端的监控目标配置文件.....	43
12.3.4 将复制过来的客户端配置文件加入 nagios.cfg.....	44
12.3.5 重新启动 NSCA 守护进程和 nagios	44
12.3.6 NSCA 客户端和服务端界面	44
十三. Cacti 的安装	46
13.1 安装 mysql、php、rrdtool、snmp	46
13.2. mysql 的配置.....	46
13.3. 修改 cacti 的 config.php 参数	49
13.4. 登陆 cacti.....	49
1. cacti-plugin-arch 的安装.....	51
2 ndoutils-1.4b 安装.....	52
3 Npc 的安装.....	53

○. 废话

Nagios 是 Linux 下的开源工具，可以监控各种操作系统、网络设备的 CPU、内存、进程、服务、流量等，并且支持声音、mail、短信等实时告警方式。

由于 Linux 发行版和 Nagios 版本众多，不同的版本安装方法可能有所偏差，因此本文中涉及到的大部分软件基本都标识了版本号。在使用不同的操作系统、软件版本时，本文仅供参考。

本文仓促，排版不够规范，部分命令可能会有拼写错误，注意检查。

此外，本文只实现了 nagios 的部分功能，期望值别太高。还有，本文参考大量资料，绝非原创，仅作参考，请随便转摘。

一. 安装 Nagios 及插件

1.1. 准备工作

Linux 操作系统，本文档为 CentOS 5.2，不需 Xwindow。安装过程略。

Develop tools，因为安装时需要用到 glib、gcc 等工具，所以最省事的方法就是安装操作系统时，把 develop tools 一并选上。

httpd-2.12.3，光盘上的 httpd 的 rpm 包，或安装操作系统后下载源码包安装，网上有篇文章说“严重”不推荐使用 Yum 安装 apache，原因不明。

gd-devel-2.0.33-9.4，这个包不装不会影响 nagios 的告警功能，但会导致 Nagios 页面上部分功能的缺失，比如 firefox 插件、mrtg 和 pnp 绘制历史数据等，建议安装时选上。

Nagios -3.0.3

Nagios -plugin-1.4.13

Nagios -2.8.1，只针对被控端为 Linux 的工具。

1.2. 建立相关帐号

创建 nagios 和 apache 的帐号并给定登录口令

```
Useradd nagios
```

```
Useradd apache ;此账户在使用 rpm 安装 httpd 时会自动增加
```

创建一个用户组名为 **nagcmd** 用于从 Web 接口执行外部命令。将 nagios 用户和 apache 用户都加到这个组中。

```
/usr/sbin/groupadd nagcmd
```

```
/usr/sbin/usermod -G nagcmd nagios
```

```
/usr/sbin/usermod -G nagcmd apache
```

1.3. 编译与安装 Nagios

前面说过建议安装操作系统时选上 `gd-devel`，如果没选，在安装 `nagios` 之前可以先安装 `gd-devel`，这个包的依赖比较多，最便捷的方法就是使用 `yum` 安装：

- `yum install gd-devel`
安装 `gd-devel` 后，就可以安装 `nagios` 了。
- `tar zxvf nagios-3.x.tar.gz`
- `cd nagios-3.x`

运行 Nagios 配置脚本并使用先前开设的用户及用户组：

- `./configure --with-command-group=nagcmd`

编译 Nagios 程序包源码

- `make all`

安装二进制运行程序、初始化脚本、配置文件样本并设置运行目录权限

- `make install`
- `make install-init`
- `make install-config`
- `make install-commandmode`

安装完毕后，`/usr/local/nagios` 目录下会出现 5 个子目录：`bin`、`etc`、`sbin`、`share`、`var`。

如果没少了表示安装没成功。

1.4. 配置 WEB 接口

在源码包路径下，安装 Nagios 的 WEB 配置文件到 Apache 的 `conf.d` 目录下。当然，前提是已经安装好 `httpd`。

`make install-webconf`

创建一个 `nagiosadmin` 的用户用于 Nagios 的 WEB 接口登录。记下你所设置的登录口令，用于通过 WEB 登录 `nagios`。

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

上面的命令注意粗体字部分，中间是个点，不是空格。

再一个注意事项是，如果新增用户的话，不需要加参数 `c`。具体帮助可以执行 `htpasswd` 看看。

重启 Apache 服务以使设置生效。

`service httpd restart`

1.5. 编译并安装 Nagios 插件

安装插件的目的是在 `/usr/local/nagios/libexec` 目录下生成监控的命令文件。

如果需要 `mysql`、`snmp` 的支持，在编译之前确认安装了 `mysql-devel` 和 `net-snmp` 相关软件。

展开 Nagios 插件的源程序包

```
# tar xzf nagios-plugins-1.4.13.tar.gz
```

```
# cd nagios-plugins-1.4.13
# ./configure --with-nagios-user=nagios --with-nagios-group=nagios
# make
# make install
```

安装完毕后，nagios 目录下出现 libexec 目录，全部是 nagios 监控时需要调用的命令。

1.6. 启动 Nagios

把 Nagios 加入到服务列表中以使之在系统启动时自动启动

```
chkconfig --add nagios
chkconfig --level 345 nagios on
```

验证 Nagios 的样例配置文件

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

以上命令的路径有点长，简化一下看得更清晰：

```
[root@host ~]# cd /usr/local/nagios/
[root@host nagios]# bin/nagios -v etc/nagios.cfg
```

就是用 nagios 可执行文件去检测 nagios.cfg 文件，如果没有错误，就可以运行 nagios 服务了。

（我自己的做法是把上面的命令作成个批处理放到/usr/bin 目录下，每次不需指定路径就直接执行）

```
#### .....
#### 此处省略若干行检测项
Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
[root@host nagios]#
```

如果有错误，提示会具体到某一文件的某一行，比如 windows.cfg 文件有错。

```
[root@host nagios]# bin/nagios etc/nagios.cfg

Error: Unexpected token or statement in file '/usr/local/nagios/etc/objects/windows.cfg' on line 26.
Bailing out due to one or more errors encountered in the configuration files. Run Nagios from the command
line with the -v option to verify your config before restarting. (PID=2058)
```

此外，nagios 的日志存放在../nagios/var/nagios.log 中，可以看到发邮件、短信以及配置文件检测错误的信息。

如果是早期的 Linux 版本，这个时候就可以用浏览器打开 <http://IP/nagios> 查看 nagios 的页面了，如果是比较新的 Linux，还需继续 selinux 修改。

1.7. 更改 SELinux 设置

默认情况下，Fedora 以后的操作系统（包括 RHEL、CentOS）与 SELinux(安全增强型 Linux)同步发行，并且在安装后默认使用强制模式。这会在你尝试联入 Nagios 的 CGI 时导致一个“内部服务错误”消息。直观显示就是无法通过刚才设置的 WEB 帐号登录。解决方法：

1.7.1. 临时性改动

查看当前模式

`Getenforce`

显示结果如果为: `enforcing`, 表示 `selinux` 已经使用了强制模式, 使用下面的命令改变为允许模式:

`setenforce 0`

再次使用 `getenforce` 查看当前模式

`Permissive`

此处表示正常。可以使用前面设置的帐号登录 `nagios` 页面了!

注意, 重启系统后, `SeLinux` 的模式恢复为强制, 必须再次使用 `setenforce 0`。

1.7.2. 永久性变更

彻底禁用 `selinux` 的强制模式。

Vi `/etc/selinux/config`, 将 `SeLinux` 的值修改为 `Disabled`

<code>SELINUX=disabled</code>

重启系统生效。

1.7.3. 只针对 Nagios 变更

不关闭 `SELinux`, 但永久性变更针对 `Nagios` 的方法是让 `CGI` 模块在 `SELinux` 下指定强制目标模式:

<pre>chcon -R -t httpd_sys_content_t /usr/local/nagios/sbin/ chcon -R -t httpd_sys_content_t /usr/local/nagios/share/</pre>

1.8. 登录 WEB 接口

现在可以从 `WEB` 方式来接入 `Nagios` 的 `WEB` 接口了, 输入 <http://ip/nagios>, 提示下输入用户名(`nagiosadmin`)和口令就登录成功了! 当然, 由于此时没定义主机, 没定义联系人, 这个页面除了监控本机外啥都没有, 基本上就是个摆设。

1.9. 其他注意事项

确保 `iptables` 的规则允许 `httpd` 服务被访问

```
iptables -I INPUT -p tcp -m state --state NEW --dport 80 -j ACCEPT
service iptables save
```

1.10. 模板继承文件（非常重要）

/usr/local/nagios/etc/objects/template.cfg。

这个文件非常重要，可以节省极大的工作量。nagios3.x 比之前的版本重要改进之一就是模板继承功能。模板继承的含义大致解释如下：之前定义的主机、服务的监控细节可以在以后的配置文件中通过模板中定义好的名称来直接调用，而不需要每个服务内容都写成冗长的配置。

这个解释可能有些抽象而晦涩，从 template.cfg 中抽取部分配置举例会更好的说明。

```
define host {
    name                generic-host;  自定义的模板名称，可根据喜好命名。
    check_period         24x7        ; 检测周期，7 天 24 小时
    check_interval       10          ; 服务检测周期，每 10 分钟
    retry_interval       1           ; 当服务出现故障时重新检测的间隔，1 分钟
    max_check_attempts   2           ; 最大检测次数。检测 2 次仍故障才会发布设定的告警手段
    check_command        check-host-alive ; 检测命令，在 commands.cfg 中定义的命令
    notification_period  workhours    ; 告警周期，workhours 对应的时间在其他部分。
    notification_interval 30          ; 当故障持续存在时，告警的间隔周期
    notification_options c,u,w,r      ; 告警类型，即何种状态下才发布告警信息。
                                      ; c=critical,u=unreachable,w=warning,r=recovery
    contact_groups       admins       ; 告警信息接收组，具体的组员信息定义在 contacts.cfg 中
    register              0           ; 0 表示这是个不完整的模板。不完整即定义了绝大部分需
                                      ; 要的信息，但使用时需要补充关键信息
                                      ; 更便于扩展。
}
```

当我们有主机需要被 nagios 监控时，我们只需要这么写：

```
define host {
    use                generic-host;  上文中定义的模板
    host_name          webserver      ; nagios 的 web 页面上看到的主机名称
    alias              Intranet web server ; nagios 页面上看到的对主机的描述
    address            10.80.26.33    ; 监控目标的 IP 地址
    check_interval      5              ; 此处与模板中的变量冲突时，此处配置生效
}
```

如此，webserver 中未标明的参数如监控重试次数、告警周期等统统继承了模板中的配置，以模板 generic-host 中定义为准。

同时我们也看到上面这个 webserver 配置中检测周期：check_interval=5，而上面的模板中也存在一个相同的选项，但值为 10。哪个有效？

当本地变量和模板变量同时存在时，本地变量生效。本例中，webserver 每 5 分钟被检查一次。也就是说，当模板中定义的规则不符合你的要求时，你可以自行更改。

模板可以被反复继承，如 webserver 的配置中如果加入 name web 这样一行，那么 web 这个变量也可以被其他配置再次调用，但不建议继承关系的层次太多，乱啊。

联系人、主机、服务、网络设备在 template.cfg 中都有自己的模板。也可以根据自己的需求新增定义模板。

二. 定义联系人

监控目标之前,最好定义联系人,包括邮件、手机号等信息,便于目标出错时告警信息的发布。否则我们就只能用眼睛盯着 nagios 的 web 页面。

联系人和联系人归属组都可以在一个文件下定义,也可以在多个文件下定义。这里介绍一下在一个文件下对多用户多组的定义。

示例:

组名	admins		test_group
归属组员	S	Y	test_user

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

```
##### 定义第一个联系人
define contact{
    contact_name      S ;名字后面不要有空格,
    use                generic-contact ; 这个就是 1.4 节说过的模板继承,具体信息参见模
板内容
    alias              Nagios Admin
    email              testing@netease.com
    pager              133      ;手机号码
}

### 定义第二个联系人
define contact{
    contact_name      Y ;名字后面不要有空格,
    use                generic-contact
    alias              Nagios Admin
    email              yang@sohu.com
    pager              137xxxxxxx
}

##### 定义第三个联系人
define contact{
    contact_name      test_user ;名字后面不要有空格,
    use                generic-contact
    alias              Nagios Admin
    email              test@sina.com
    pager              139
}
```

```
#### 定义联系人组
define contact group {
    contactgroup_name    admins    ;组名
    alias                Nagios Administrators
    members              S,Y      ;多个联系人用逗号分隔
}

define contact group {
    contactgroup_name    test_group
    alias                Test
    members              test_user ;只有一个联系人
}
```

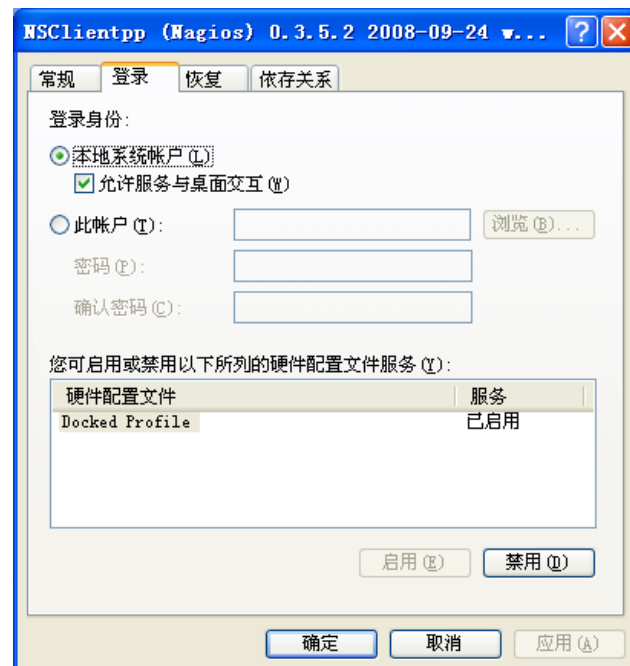
三. 监控 Windows 主机

3.1. Windows 服务器的准备工作

在用 Nagios 监控 Windows 机器的私有服务之前，需要先在机器上安装代理程序。推荐使用 NSClient++外部构件，本文用到的是 NSClient++-Win32-0.3.5.msi。

3.1.1. 安装，用过 windows 的都知道。

3.1.2. 系统中新增服务“NSClientpp (Nagios)”，打开服务管理器并确认 NSClientpp 服务可以在桌面交互。



3.1.3. 编辑 NSC.INI 文件(位于 C:\NSClient++目录)并做如下修改：

- 去掉在 [modules] 段里的列出模块程序的注释，除了 CheckWMI.dll 和 RemoteConfiguration.dll、NSCAAgent.dll;

- 最好是修改一下在[Settings]段里的'password'选项;

Password=abcd1234

这个密码在后面针对目标主机服务、进程监控时会用到。用于 check_command 命令的最后面，后面还会再次描述。

- 去掉在[Settings]段里的'allowed_hosts'选项注释，把 Nagios 服务所在主机的 IP 加到这一行里，例如：

Allowed_hosts=192.168.12.0/24

或是置为空，让全部主机都可以联入，但不建议这样做。

- [NSClient]段里的'port'选项里已经去掉注释并设置成'12489'(默认端口);

[NSClient]

port=12489

3.1.4. 启动 NSClient++服务。如果安装正确，一个新的图标会出现在系统托盘里；有的电脑没出现这个图标，不过问题不大，只要 12489 端口处于侦听状态就行。



```
C:\>netstat -na|find "12489"
```

TCP	0.0.0.0:12489	0.0.0.0:0	LISTENING
-----	---------------	-----------	-----------

3.1.5. 检查防火墙，允许 TCP 12489 端口被访问。

3.1.6. 注意，NSC++的密码是明文传输，所以，当使用互联网作为传输媒介时，必须也加上一定的防火墙策略。

3.2. 允许 Nagios 主机监控 Windows

编辑 Nagios 的主配置文件

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

不监控本机，在下面这行前面增加“#”注释掉。

```
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
```

允许监控 Windows 服务器，确保下面这行配置前无“#”

```
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

保存配置文件并退出。

对于第一台 Windows 机器，可以只是简单地修改里面已经有的主机与服务对象定义而不要新创建一个。

3.3. Nagios 监控 Windows 主机的进程

3.3.1. 监控单台主机多项指标及示例

3.3.1.1 首先了解一下配置文件的内容

在 `./nagios/etc/objects/template.cfg` 中定义了大部分参数的默认值，比如监控周期、最大重试次数等，此为模板。Nagios3 版本以后提供了模板继承的功能，即模板中定义的参数，在以后的配置项中可以不用写，使用 `use` 来调用即可。如果配置文件中没指定具体参数的话，都按照此配置中的默认值处理。

3.1.1.2 添加第一台设备

给 Windows 机器加一个新的主机对象定义以便监控。如果是被监控的第一台 Windows 机器，可以只是修改 `windows.cfg` 文件里的对象定义。

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

修改 `host_name`、`alias` 和 `address` 域以符合那台 Windows 机器。

示例，监控 192.168.1.33 的 CPU、内存、磁盘空间、`cmd`、Firefox 等进程。

注意事项 1: `define host` 中的 `host_name` 并非操作系统中指定的主机名；命名规则统一，否则 web 页面排序就会很乱。

注意事项 2: `define service` 中，进程名全部用大写，不管其在 windows 资源管理器中是如何显示；而目前看来，Windows 自带的系统进程好像无所谓，具体情况还有待测试。

第二章中我们定义了两个组。默认情况下，被控目标的告警信息发布给组 `admins`，这里增加一行配置，将 `cmd.exe` 的告警信息发布给组 `test_group`，即第二章中定义的 `test_user` 这个人。

```
定义主机
define host{
    use                windows-server ;这个 windows-server 就是 1.11 和 3.3.1.1 中提到的模板。

    host_name          Host_33      ;如果是短信告警，建议写短点。
    alias               S's notebook ; A longer name associated with the host
    contact_group       test_group   ;当主机无法 ping 通时，告警信息发给 test_group 组成员。
    address             192.168.1.33 ; IP address of the host
}

##### 监控进程 CMD
define service{
    use                generic-service
    host_name          Host_33
    service_description Cmd
    contact_group       test_group   ;告警信息发给 test_group 组成员。
    check_command       check_nt!PROCSTATE!-d SHOWALL -l CMD.EXE
}

##### 监控进程 firefox ，告警信息发给 admins 组成员
define service{
    use                generic-service
```

host_name	Host_33
service_description	firefox
check_command	check_nt!PROCSTATE!-d SHOWALL -l FIREFOX.EXE
}	

3.1.1.3 其他常用监控 windows 的命令

check_nt!USEDISKSPACE!

目标启动时间	check_nt!UPTIME
CPU 负载, 80% 告警, 90% 紧急	check_nt!CPULOAD!-l 5,80,90
内存, 阈值同上	check_nt!MEMUSE!-w 80 -c 90
磁盘空间, 阈值同上	check_nt!USEDISKSPACE!-l d -w 80 -c 90
检查 Windows 的服务状态	check_nt!SERVICESTATE! -d SHOWALL -l

更多的命令可以参见/nagios/libexec 目录, 每个命令后面加-h 即可查看该命令的帮助。

3.3.2. 监控多台主机进程示例

3.3.2.1 新增自定义配置文件, 文件路径和文件名可自定义, 前提 nagios 必须有权限。

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

以下为分别监控 192.168.1.2 和 192.168.1.3 上的 a.exe、b.exe、c.exe 进程的示例。

##定义服务器 192.168.1.3 的名称	
define host{	
use	windows-server
host_name	<i>Host_3</i>
alias	My Windows Server
address	<i>192.168.1.3</i>
}	
define host{	
use	windows-server
host_name	<i>Host_2</i>
alias	My Windows Server
address	<i>192.168.1.2</i>
}	
### 定义主机归属组, 其中 192.168.1.11 的主机被监控目标另有配置文件, 这里不再列举	
define hostgroup{	
hostgroup_name	aiyo-windows-servers
alias	Windows Servers
members	192.168.1.2,192.168.1.3,192.168.1.11
}	
#### 定义要监控的进程	
define service{	
use	generic-service
host_name	<i>Host_3</i>

```

        service_description      T
        check_command             check_nt!PROCSTATE!-d SHOWALL -l T.EXE
    }

define service{
    use                             generic-service
    host_name                       Host_3
    service_description            C
    check_command                   check_nt!PROCSTATE!-d SHOWALL -l C.EXE
}

define service{
    use                             generic-service
    host_name                       Host_3
    service_description            IVR
    check_command                   check_nt!PROCSTATE!-d SHOWALL -l I.EXE
}

define service{
    use                             generic-service
    host_name                       Host_2
    service_description            T
    check_command                   check_nt!PROCSTATE!-d SHOWALL -l T.EXE
}

```

整理好上面的配置文件后，修改../nagios/etc/nagios.cfg 配置文件，将新增的 winhosts.cfg 添加到其中。

```

# Definitions for monitoring a Windows machine
cfg_file=/usr/local/nagios/etc/objects/windows.cfg
cfg_file=/usr/local/nagios/etc/objects/winhosts.cfg

```

Service nagios restart

注意事项：

- 检测进程的命令也可以将多个进程合并到一个监控目标中，流量更小。

```

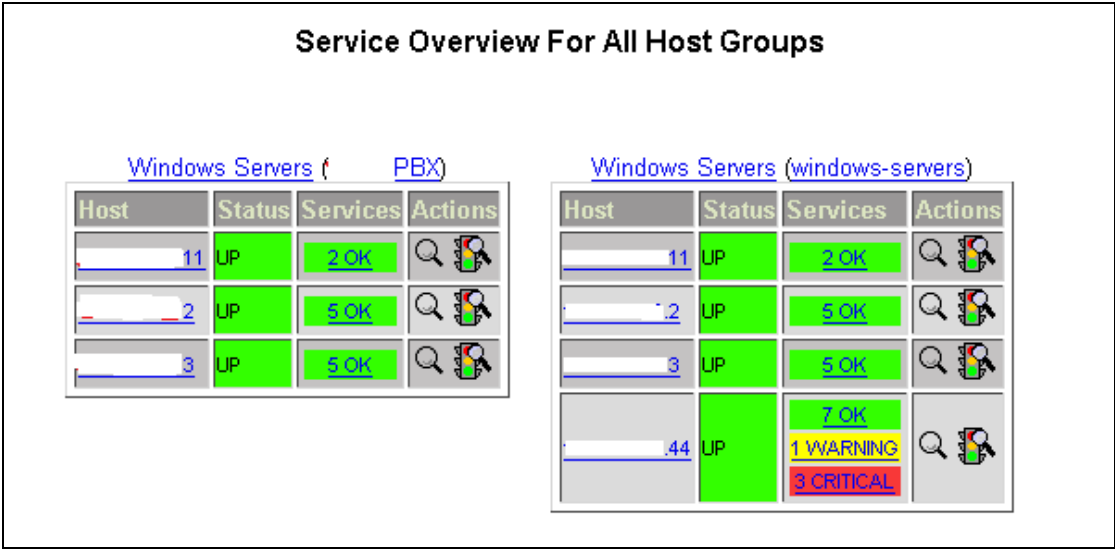
        check_command             check_nt!PROCSTATE!-d SHOWALL -l
        MANAGER.EXE,CTIDRVEXE

```

- 当目标进程文件名不够标准，比如带括号或空格时——cmd(1).exe、ab cd.exe，检查命令的内容按如下格式输入：
“CMD(1).EXE”、“AB\ CD.EXE”，注意“\”后有空格。

3.3.2.2 分组显示的页面如下：

同一服务器可以归纳到多个服务器组下。



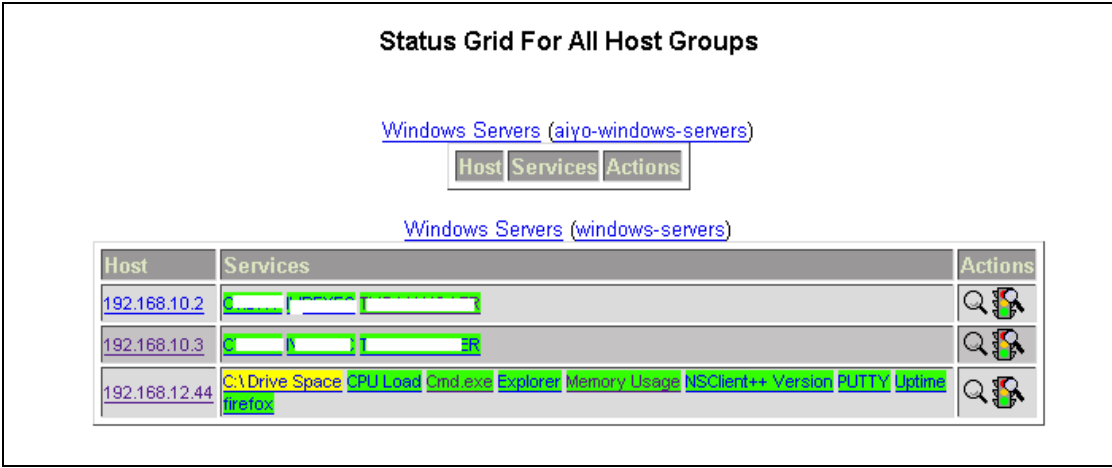
3.3.3. 监控页面显示监控结果

Host	Service	Status	Last Check	Duration	Attempt	Status Information
12	...	OK	12-07-2008 21:31:44	1d 17h 53m 8s	1/3	C...exe: Running
	...	OK	12-07-2008 21:27:41	1d 17h 50m 59s	1/3	i...exe: Running
	...	OK	12-07-2008 21:26:01	0d 0h 49m 14s	1/3	M...exe: Running
3	...	OK	12-07-2008 21:28:10	1d 18h 1m 6s	1/3	C...exe: Running
	...	OK	12-07-2008 21:30:18	1d 17h 59m 17s	1/3	i...exe: Running
	...	OK	12-07-2008 21:32:27	1d 18h 0m 5s	1/3	M...exe: Running
44	C:\Drive Space	WARNING	12-06-2008 03:16:32	1d 20h 3m 28s	3/3	c: - total: 7.81 Gb - used: 6.57 Gb (84%) - free 1.24 Gb (16%)
	CPU Load	OK	12-07-2008 21:29:50	1d 18h 58m 8s	1/3	CPU Load 9% (5 min average)
	Cmd.exe	CRITICAL	12-07-2008 21:32:44	0d 0h 36m 31s	3/3	CMD.EXE: not running
	Explorer	OK	12-07-2008 21:28:53	1d 18h 26m 26s	1/3	Explorer.EXE: Running
	Memory Usage	OK	12-07-2008 21:31:01	1d 18h 21m 13s	1/3	Memory usage: total:2450.39 Mb - used: 639.69 Mb (26%) - free: 181
	NSClient++ Version	OK	12-07-2008 21:31:58	1d 18h 56m 57s	1/3	NSClient++ 0.3.5.2 2008-09-24
	PUTTY	OK	12-07-2008 21:34:07	1d 18h 20m 28s	1/3	putty058.exe: Running
	Uptime	OK	12-07-2008 21:27:27	1d 18h 55m 46s	1/3	System Uptime - 0 day(s) 1 hour(s) 43 minute(s)
	firefox	OK	12-07-2008 21:29:36	1d 18h 9m 11s	1/3	firefox.exe: Running

3.3.3.1 从上图监控页面的 Service detail 上可以看到如下信息：

- Host_3 和 Host_2 的进程都正常。
- Host_44 的 C 盘空间剩余 1.24G，磁盘空间占用率达到了 84%。之前的设定是 80% 告警，90% 紧急，因此显示黄色警告信息。带红叉的喇叭的意思是此服务不发布任何信息。粗体的暗红色的叉表示不再检测，上图中表示 nagios 不检查本机存活状态和 C 盘的空间。
- CPU 负载 9%
- 内存占用率 26%
- 系统运行时间 1 小时 43 分钟。
- 因为没检测到 CMD.EXE 的进程，因此显示红色背景，并标识为 critical（紧急、危急的意思）。

3.3.3.2 也可以以栅格形式概略显示各监控目标的状态



3.3.4. 使用 escalation 逐级通知联系人

在实际应用中，会遇到告警消息连续发布，但因为各种原因导致故障迟迟未能处理，此时需要通知更高级的联系人，此时就用到 escalation 功能。

下面是一个被控目标的配置文件，

```
define service{
    use generic-service
    host_name BJ-2
    service_description TEST1
    contact_group zz ;告警信息发给 zz 组成员。
    check_command check_nt!PROCSTATE!-d SHOWALL -l ASDF.EXE
}
Define serviceescalation{
    Host_name BJ-2
    Service_description TEST1
    First_notification 3 ;自第 3 条消息起，发送给本组员。
    Last_notification 5 ;0 表示无限制通知，5 表示只发送到第 5 条。此后
    的消息仍发送给前一组员。
    Contact_groups S
    Notification_interval 10
    Escalation_options w,c ;只发布告警、紧急的消息，其他消息仍发送给上一
    组员
}
```

四. 监控 Linux 主机（NRPE）

Nagios 监控 Linux 主机需要一个插件——nrpe，本例所用 nrpe2.8.1，这个插件需要分别装在主控端和被控端上。这个插件需要 openssl 的支持，所以还需在 Nagios 主机和被控端安装。

4.1. nagios 主机的安装

- 安装 openssl-devel, 过程略
- 安装 nrpe

```
tar zxvf nrpe-2.8.1.tar.gz
cd nrpe-2.8.1
./configure
make
make all
make install-plugin
```

- 在../nagios/etc/objects/commands.cfg 中增加 nrpe 命令, 定义为 check_nrpe。

```
define command{
    command_name    check_nrpe
    command_line     $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
```

4.2. 被监控目标的安装和测试

4.2.1. 安装

- 安装 openssl-devel, 过程略。
- 创建 nagios 帐户, 过程略。
- 安装 nagios-plugin-1.4.13 插件, 安装过程参照第一章。
- 安装 nrpe, 默认安装到/usr/local/nagios/

```
tar zxvf nrpe-2.8.1.tar.gz
cd nrpe-2.8.1
./configure
make all
make install-plugin
make install-daemon
make install-daemon-config
```

安装完毕后, ../nagios/libexec/目录下出现多个可执行文件, 是检测目标的命令。

- nagios 目录归属到 nagios 帐户下:

```
Chown -R nagios:nagios /usr/local/nagios
```

- 编辑../nagios/etc/nrpe.cfg 文件, 允许 nagios 主机可访问。

Allowed_hosts=192.168.0.1 ;多个 nagios 主机用逗号分隔。

Command 后的[]里的内容是定义在本机上执行的命令, 与主控端定义的 check_nrpe 配合使用。

- 启动 nrpe 守护进程, 默认侦听端口为 TCP5666

```
/usr/local/nagios/bin/nrpe -c /usr/local/nagios/etc/nrpe.cfg -d
```

可以将这个命令放置在/etc/rc.local 中随系统启动。

4.2.2. 测试

- 在被控端服务器上检查安装结果

```
[root@rh90 libexec]# pwd
/usr/local/nagios/libexec           ;当前路径
[root@rh90 libexec]# ./check_swap -w 20 -c 10
SWAP OK - 100% free (382 MB out of 384 MB) |swap=382MB;0;0;384
```

- 在主机端 nagios 服务器上检查被控端安装结果，前提保证 nagios 服务器可不受被控端的防火墙限制。

```
[root@host libexec]# ./check_nrpe -H 192.168.12.226
NRPE v2.8.1
```

- 再检查目标服务器的当前用户数量，只有 1 个用户登录。

```
[root@host libexec]# ./check_nrpe -H 192.168.12.226 -c check_users
USERS OK - 1 users currently logged in |users=1;5;10;0
```

Check_nrpe 的命令使用-h 参数查看帮助的信息。

4.3. 被控端命令的定义

上文检查登录用户数量的命令中，-c 后面的参数必须是在被控端 nrpe.cfg 里定义好的，见下文[]中的内容。并且[]中的内容是自定义的，与 nagios/libexec 目录下的可执行文件无关。

- 比如上文的 check_user 命令在 nrpe.cfg 里的定义是这样的：

```
command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c 10
```

具体含义是当 5 个以上用户登录时告警，10 个用户以上登录时紧急。

- 检查 swap 分区的使用情况，告警状态为剩余的资源比例。

```
command[check_swap]=/usr/local/nagios/libexec/check_swap -w 20 -c 10
```

- 新增检查根分区容量的命令，剩余空间小于 20% 时告警

```
command[check_]=/usr/local/nagios/libexec/check_disk -w 20% -c 10% -p /
```

- 还有一些 Linux 下的进程，尤其是带后缀的长命令需要监控时，首先用 ps 命令查看具体的进程名，然后在 nrpe.cfg 里添加相应的命令名称。比如，要监控一个用 ipython 启用的命令，首先查看包含这个命令的进程：

```
[root@feecenter libexec]# ps ax |grep ipython
2935 ?        S          0:01 xterm -e /usr/local/bin/ipython start_huhu.py
2938 pts/0    Ssl+      163:11 /usr/local/bin/python /usr/local/bin/ipython start_huhu.py
2979 ?        S          0:00 xterm -e /usr/local/bin/ipython start_lulu.py
2982 pts/2    Ssl+      0:05 /usr/local/bin/python /usr/local/bin/ipython start_lulu.py
2987 ?        S          0:00 xterm -e /usr/local/bin/ipython start_rc.py
2990 pts/3    Ssl+      0:35 /usr/local/bin/python /usr/local/bin/ipython start_rc.py
22151 pts/5    S+         0:00 grep ipython
```

如果我们要监控的是第 4 行的进程：

2982 pts/2 Ssl+ 0:05 /usr/local/bin/python /usr/local/bin/ipython start_lulu.py
需要在配置文件中增加：

```
command[check_lulu]=/usr/local/nagios/libexec/check_procs -c 1:1 \
-a '/usr/local/bin/python /usr/local/bin/ipython start_lulu.py'
```

上面的参数中 -c 1:1 的意思是，如果当前系统中没有一条包含如上命令的进程，则状态为 c，即 critical。

注意，修改 nrpe.cfg 后，必须重启 nrpe 才生效。由于 nrpe 没有加载到服务中，所以需要通过杀掉进程再启动的方法来实现 nrpe 的重启，杀掉进程最简单的方法：

```
kill -9 $(ps ax|grep nrpe)
之后再启动 nrpe 守护进程就 OK 了。
```

4.4 将被控目标加入 nagios 监控





- 检查 CPU 负载

```
define service{
    use                generic-service
    host_name           BJ-TEST
    service_description Current Load
    check_command        check_nrpe!check_load
}
```

- 检查根分区容量，注意，此处调用的 6.3 节中最后一例的 check_/命令

```
define service{
    use                generic-service
    host_name           BJ-254
    service_description Root Partition
    check_command        check_nrpe!check_/
}
```

- 部分监控结果

Root Partition		WARNING	12-29-2008 17:33:54	0d 1h 51m 23s	3/3	DISK WARNING - free space: / 1068 MB (18% inode=86%):
SSH		OK	12-29-2008 17:28:27	0d 5h 6m 50s	1/3	SSH OK - OpenSSH_3.5p1 (protocol 1.99)
Swap Usage		OK	12-29-2008 17:33:01	0d 2h 12m 16s	1/3	SWAP OK - 100% free (382 MB out of 384 MB)
Total Processes		OK	12-29-2008 17:35:07	0d 2h 30m 10s	1/3	PROCS OK: 43 processes

Swap 分区的旁边有一个云状图标，这个表示 swap 分区处于抖动状态（flap）。所谓抖动状态即有可能是刚刚从一个不正常的状态返回到当前状态，或频繁出现不正常状态。虽然目前的状态是正常的，但还需要特别关注。

五. 监控网络流量

5.1. 服务端安装 SNMP

不同的设备有不同的安装方法，SNMP 安装过程略。强调两点，一是 SNMP 端口必须只允许可信任的网络连接，二是修改默认的团体字名称 `public` 为其他名称，并设置为只读。

snmp.conf 的内容修改

```
# 修改团体字名称为 Quake3
#com2sec notConfigUser default public
com2sec notConfigUser default Quake3

# 修改 systemview 为 mib2
#access notConfigGroup "" any noauth exact systemview none none
access notConfigGroup "" any noauth exact mib2 none none ;或者 mib2 改成 all
# 取消下行前的“#”
view all included .1 80
```

5.2. check_snmp 监控设备启动时间

如果 `./nagios/libexec` 下没有 `check_snmp` 命令，需要安装光盘上的 `net-snmp-utils` 包，重新安装 nagios 插件。

配置文件不再列举，这里只写命令。

```
define service{
    use generic-service ; Inherit values from a template
    host_name BJ-NGN-ROUTER
    service_description Uptime
    check_command check_snmp!-C public -o sysUpTime.0 ;public 是 snmp 的团体字。下同
}
```

5.3. check_traffic 安装检测流量插件

本文使用 `check_traffic.sh` 来检测网络流量。`check_traffic.sh` 调用了 nagios 插件中的 `check_snmp` 命令。`check_snmp` 命令依赖于 `net-snmp` 安装包中的 `snmpwalk`、`snmpget` 等命令，如果用到检测流量工具，在安装 nagios 插件之前必须确保系统正确安装了如下安装包：

```
net-snmp-5.3.2.2-5.el5.i386.rpm
net-snmp-devel-5.3.2.2-5.el5.i386.rpm
net-snmp-libs-5.3.2.2-5.el5.i386.rpm
net-snmp-perl-5.3.2.2-5.el5.i386.rpm
```

5.3.1. 安装 check_traffic

- 解压
Buzip2 check_traffic.zip
- 放到/usr/local/nagios/libexec/下，并允许所有人可执行，特别是 nagios 帐户
chmod 711 check_traffic.sh
- 该命令需要 nagios 帐户对 check_traffic 里的文件可读可写，将该文件归属 nagios 帐户
chown -R nagios:nagios /var/tmp/check_traffic_123.x.x.x_1.hist_dat

5.3.2. 将 check_traffic 加入到 nagios 中

- 在 command.cfg 里添加命令

```
define command{
    command_name    check_traf
    command_line    $USER1$/check_traffic.sh -V 1 -H $HOSTADDRESS$ -I $ARG1$ -w
$ARG2$ -c $ARG3$ -K -b ;最后两个参数是流量单位，Kbit，可自定义，比如-K -B 为 KByte
}
```

- 获取端口信息，注意 G0/0 的 index 为 1，下一步凭此值监控该端口的流量。

```
[root@host objects]# ../../libexec/check_traffic.sh -H 192.168.0.1 -C public -V 1 -L
List Interface for host 192.168.0.1.
Interface index 1 orresponding to   GigabitEthernet0/0
Interface index 2 orresponding to   GigabitEthernet0/1
Interface index 3 orresponding to   FastEthernet1/0
Interface index 4 orresponding to   FastEthernet1/1
Interface index 5 orresponding to   Null0
Interface index 6 orresponding to   Tunnell
Interface index 7 orresponding to   Loopback0
```

- 监控 G0/0 端口的网络流量

```
define service{
    use                generic-service ; Inherit values from a template
    host_name          Cisco- Router-3825
    service_description Port G0/0 192.168.0.1
    check_command       check_traf!1!700,800!900,1000 -C public
    # 三个参数分别对应 G0/0 端口的 index，流量 700-800Kb 时告警，900-1000Kb 时紧急。
}
```

六. 告警消息的发布

当监控的目标出现问题时，Nagios 可以通过声音、邮件、短信等方式进行通知。

6.1 Email 方式发布告警

6.1.1. 实现

Linux 系统自带 sendmail 服务，实现邮件发布的要素有两条：

6.1.1.1 /usr/local/nagios/object/contact.cfg 文件中的联系人

```
define contact {
    contact_name          sun
    use                   generic-contact
    alias                 Nagios Admin
    service_notification_period 24x7    #服务器监控时间范围，默认
值。
    host_notification_period 24x7      #主机告警时间范围，;默认值。
##### 以下两行配置是主机、服务在什么状态下告警
##### w=warning,u=unknown,r=recover,c=critical
    service_notification_options w,u,c,r ;默认值，无此行也可
    host_notification_options   d,u,r   ;默认值，无此行也可
    email                       testing@netease.com
    pager                       13312345678
##### 多个联系人时，照上述格式在下面增加。
##### 定义联系人组
define contactgroup{
    contactgroup_name      admins
    alias                 Nagios Administrators
    members               S ##### 多个联系人时，以逗号分隔
}
```

6.1.1.2 Sendmail 的设置

在/etc/mail/access 添加必要的信息

```
# by default we allow relaying from localhost...
Connect:localhost.localdomain      RELAY
Connect:localhost                  RELAY
Connect:127.0.0.1                  RELAY
### add by sun
connect:http://mail.cvt.cn         RELAY
connect:nagios@localhost           RELAY
connect:192.168.12                 RELAY
connect:sun@localhost              RELAY
```

connect:testing@netease.com	RELAY
connect:test@sina.com	RELAY
connect:yang@sohu.com	RELAY

在/etc/mail/路径下执行：

```
makemap hash access < access
```

或者

```
makemap -v hash /etc/mail/access.db < /etc/mail/access
```

再执行：

```
chmod 777 /var/spool/mqueue
```

6.1.1.3 commands.cfg

此处的配置文件内容是默认的，不需任何修改，只是为了出现错误时的一个参照。

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

```
# 'notify-host-by-email' command definition
define command{
    command_name    notify-host-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification
Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATES$\nAddress: $HOSTA
DDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIMES$\n" | /bin/mail -s "*** $
NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATES$ ***" $CONTACTEMAILS$
}

# 'notify-service-by-email' command definition
define command{
    command_name    notify-service-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification
Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIASE$\nAddress:
$HOSTADDRESS$\nState: $SERVICESTATES$\n\nDate/Time: $LONGDATETIMES$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$" | /bin/mail -s "*** $NOTIFICATIONTYPE$ Service Alert:
$HOSTALIASE$/$SERVICEDESC$ is $SERVICESTATES$ ***" $CONTACTEMAIL$
}
```

6.1.2. 邮件测试

6.1.2.1 发送队列

/var/spool/mqueue 这个目录存储的是邮件发送队列，如果这个里边满了不但会影响程序的运行，而且 sendmail 容易死掉，所以需按定期检查，必要的时候要删除文件。

6.1.2.2 检查所传送的电子邮件是否送出，或滞留在邮件服务器中

```
/usr/lib/sendmail -bp
```

若屏幕显示以下信息，表示 mail 已送出。

```
[root@host nagios]# /usr/lib/sendmail -bp
/var/spool/mqueue is empty
Total requests: 0
```

```
[root@host nagios]#
```

6.1.2.3 检查邮件发送过程

在 Sendmail 邮件服务器上执行下面的命令.

echo testing | /usr/sbin/sendmail -v testing@netease.com, 如果 sendmail 正常, 邮箱里会收到内容为 testing 的邮件。

6.1.2.4. sendmail 日志

```
[root@localhost log]# pwd
```

```
/var/log
```

vi maillog 就可以看了, 出现任何错误都可以看到。

6.2. 短信猫发布告警

本文中短信下发使用的工具是短信 MODEM, 该设备接在主机的 COM2 口。配置还算简单, 不过首先需要验证短信猫硬件, 其次安装短信工具, 最后整合短信工具到 nagios, 因此会多占一点篇幅。

6.2.1. 安装 Linux 下的串口工具 minicom

Minicom 是与 Windows 超级终端对应的串口工具, CentOS 5.2 光盘自带软件包, 不过需要 lockdev 支持, 同样是光盘自带。

```
rpm -ivh lockdev-1.1.rpm
```

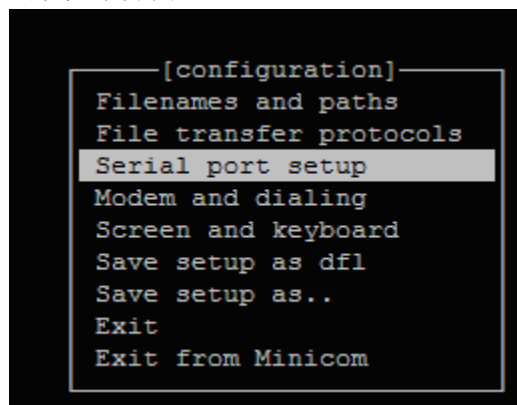
```
rpm -ivh minicom-2.1.rpm
```

6.2.2. 用 minicom 测试短信猫

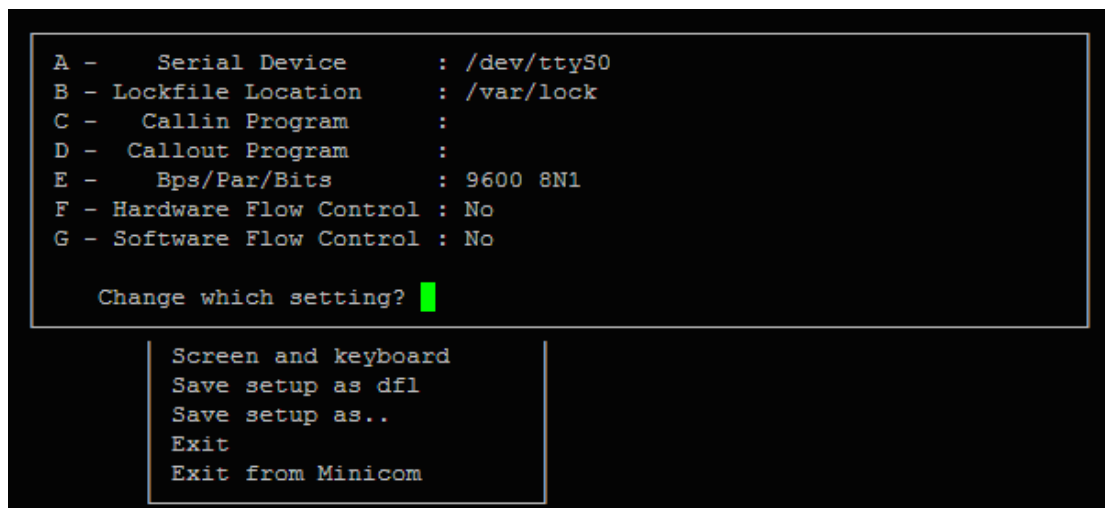
第一步必须要了解短信猫的硬件是否完好, 否则功能再强大的软件也没戏。

运行: *minicom -s*

出现如下界面:



选择: Serial port setup 后出现界面, 输入第一列的字母来修改各项参数:

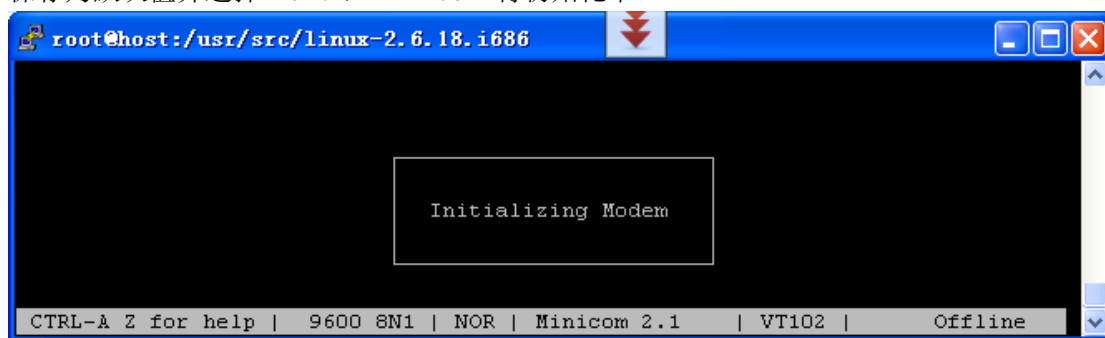


按 A 输入修改设备为/dev/ttyS1（串口 1-4 在 LINUX 下对应的设备名称是/dev/ttyS0-3）

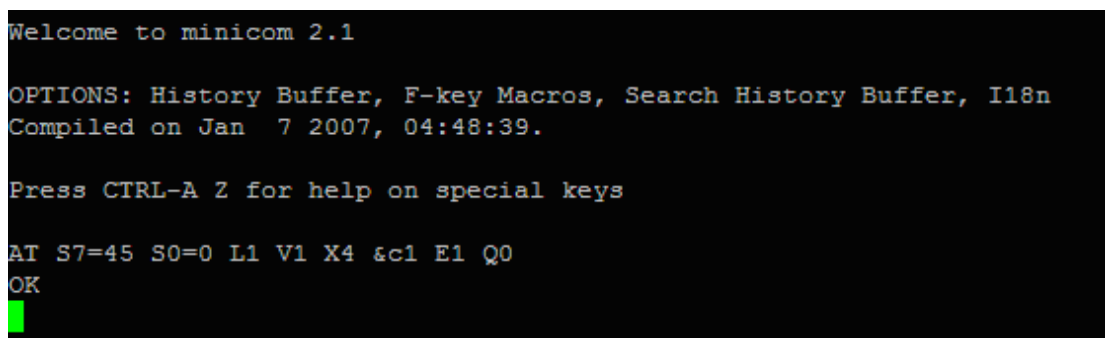
按 E 输入连接串口的速率,短信猫用的是 9600 。

修改完毕后回车确认

保存为默认值并选择“exit”， minicom 将初始化串口



初始化完毕出现下图，表示设备正常：

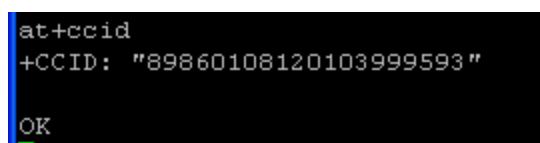


如果没有最后两行，表示配置不成功。

使用用 at 指令测试：

输入 at，弹出 OK 表示短信猫工作正常

at+ccid 可以查看设备号



发短信：

at+cmgs=13912345678 （输入后命令行多出“>”前缀，有的短信猫版本不同，命令需要以分号收尾）

>Test by sun.....

输入完后按 ctrl+z 退出，界面显示如下：

```
Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
OK
at+cmgs= 13912345678
> Test by sunfeng,OS=CentOS 5.2,tools minicom,command is "minicom -s"
>
+CMGS: 8
OK
```

如果手机成功接收到两个“>”之间的内容的短信,说明短信猫已经成功了，如果出现 ERROR 表示有问题。

6.2.3. 安装短信工具 Gnokii

Gnokii 是 Linux 下的短信客户端工具，本文分别介绍 rpm 包安装和源码包安装方式。

6.2.3.1. 用 rpm 包安装

Gnokii 工具的官方网站：<http://www.gnokii.org>，但目前只提供 rpm 安装文件，习惯用源码安装的只能去其他网站寻找。

rpm -ivh gnokii-0.6.28cvs-20081209.i386.rpm

配置文件是/etc/gnokiirc，

修改/etc/gnokiirc 将其中 5 行配置修改

```
[global]
port = /dev/ttyS1
model = AT
##同时需要注释掉 mode = 6510
serial_baudrate = 9600
use_locking = no
#### use_locking
```

运行：

gnokii --identify

注意，有些资料不知是版本还是版面问题，只有一个“-”，正确的格式应该是两个，“--”
输出结果如下：

```
Received message type 06
IMEI      : 012345678901234
Manufacturer : WAVECOM MODEM
Model     : MULTIBAND 900E 1800
Product name : MULTIBAND 900E 1800
Revision  : 532a09gg.2C2 182812
Serial device: closing device
[root@host ~]#
```

gnokii 可以工作了!!!

6.2.3.2 从源码包安装

1) 下载源码，本例中用的是 gnokii-0.6.12。

- `tar xvf gnokii-0.6.12.tar.bz2`
- `cd gnokii-0.6.12`
- `./configure --prefix=/usr/local/gnokii/ --without-x --disable-xdebug` 因为本例无图形界面
- `gmake`
- `gmake install`

2) 将源码包里的: `./gnokii-0.6.12/Docs/sample` 目录下的 `gnokiirc` 文件拷贝到 `/etc` 下，照上面提到的方法修改配置文件。

3) 测试方法同 RPM 包安装。

执行 `gnokii --identify` 的结果稍有区别

```
Received message type 06
IMEI       : 012345678901234
Manufacturer : WAVECOM MODEM
Model      : MULTIBAND 900E 1800
Revision   : 532a09gg.2C2 1828120 040802 18:22
Serial device: closing device
[root@host gnokii]#
```

注意: rpm 包安装后 `gnokii` 的可执行文件在 `/usr/bin` 目录下，而源码安装后的可执行文件默认路径在 `/usr/local/gnokii/` 目录下。

6.2.3.3. 测试 gnokii

用 `gnokii` 向手机发送测试短信。为直观显示，可以打开 `gnokii` 的 `debug` 模式，确认 `/etc/gnokiirc` 配置文件

`[logging]`

`debug = on`

在终端输入:

```
echo -n "test by sun 16:10" | /usr/local/gnokii/bin/gnokii --sendsms 13398765432
```

同样，`sendsms` 前面是两个“-”

如果是 rpm 包安装的，因为可执行文件的路径变更，上面命令的路径也需随之变更。

屏幕将出现一堆的信息,命令执行完毕如果出现:

```
Received message type 21
Message sent okay
Send succeeded!
Serial device: closing device
[root@host gnokii]#
```

号码 13398765432 收到短信——内容为“test by sun 16:10”

6.2.4. 将 gnokii 的短信功能绑定到 Nagios

终于又要到正题了。

6.2.4.1. 修改 nagios 的 `commands.cfg` 配置文件

新增加: `host-by-sms` 和 `service-by-sms` 命令

```
# send sms
```

```

define command{
    command_name host-by-sms
    command_line /bin/echo -n "$NOTIFICATIONTYPE$ $HOSTNAME$/$HOSTADDRESS$ :
host    is    $HOSTSTATE$    $SHORTDATETIME$" | /usr/bin/gnokii    --sendsms
$CONTACTPAGER$ 2>&1 > /dev/null
}
define command{
    command_name service-by-sms
    command_line /bin/echo -n "$NOTIFICATIONTYPE$ $HOSTNAME$/$HOSTADDRESS$ :
Service is $SERVICEDESC$ $SERVICESTATE$ $SHORTDATETIME$" | /usr/bin/gnokii --sendsms
$CONTACTPAGER$ 2>&1 > /dev/null
}

```

强调一下，在 4.2.3.3 小节中，提到了 gnokii 测试短信的命令：echo -n "test by sun 16:10" | /usr/local/gnokii/bin/gnokii --sendsms 13398765432。因此，上面的新增配置文件内容时，需要根据实际情况变更命令路径（斜体字部分）。

6.2.4.2. 修改 Nagios 联系人的配置文件 contacts.cfg

新增加联系人为 sms 的配置，开始修改第二章介绍过的 contacts.cfg 文件，内容增加两行：

```

define contact{
    contact_name                sun                ; Short name of user
    use                          generic-contact    ; Inherit
    default values from generic-contact template (defined above)
    alias                        Nagios Admin      ; Full
    name of user
    service_notification_commands    service-by-sms
    host_notification_commands     host-by-sms
    email                        testing@netease.com
    pager                        13398765432
}

```

6.2.4.3. 应用程序和端口权限

因为 nagios 监控是用 nagios 用户启动，nagios 用户需要对 /dev/ttyS1 有读（4）、写（2）、执行（1）的权限，同时需要可以执行发短信程序的权限，所以如果是因为权限的问题无法发送短信，需要做一下变更。如何判断是不是因为权限问题可以通过 nagios.log 来查看。如果出现发送命令，但又紧跟着出现权限不够的提示，可以检查一下各文件的权限。

默认权限都是 root 的。

```
/dev/ttyS1 root uucp
```

```
/usr/bin/gnokii root root
```

修改权限，将两个文件的权限都归到 nagios 组（root 对任何属主的文件都有最高权限）：

```
chown -R nagios:nagios /usr/bin/gnokii
```

```
chown -R nagios:nagios /dev/ttyS1
```

/dev/ttyS1 每次重启后，权限会变，因此最好将上面这行命令随系统启动。

6.3. 飞信发布告警

由于移动飞信给好友发信息是免费的，所以如果告警的目标手机同样是移动的手机，就可以用飞信方式实现，省点短信费。而且如果目标手机全是移动手机的话，连短信猫都省了。本节使用工具为飞信机器人，有一点需要注意：飞信机器人版本更新很快，每次更新后，旧版本就失效。为避免这种情况的出现，建议每日定时发送测试消息，如果哪天没收到，表示你需要检查网络或者飞信程序了。

6.3.1. 注册飞信

用移动手机注册飞信，并将目标手机设置为好友。过程略。

6.3.2. 下载飞信机器人并安装

- 解压完成后，只有一个可执行文件——`sms`，将其移动到`/usr/bin`目录下，并允许其他用户（主要是 `nagios`）可执行。

```
mv ./sms /usr/bin
chmod 755 /usr/bin/sms
或 chmod +x /usr/bin/sms
```

- 下载 `sms` 依赖的库文件，并解压。解压完毕后将这些文件复制到`/usr/lib`目录下
- ```
tar zxvf lib_lin_32.tar.gz
cp ./lib_lin_32/* /usr/lib
```

### 6.3.3. 测试飞信

帮助文件如下。

```
[root@host nagios]# sms
Usage:
 sms -f mobile -p pwd -t mobile1,... -m message -a message -d
 sms -f mobile -p pwd -t mobile1,... -i file_name[utf8] -a invite_message -d 1
 -f:Fetion mobile account(only supports mobile phone No.)
 -p:Account password
 -t:Destination mobile list
 -m:Message
 -i:File name(only supports utf8)
 -a:Auto send invite using invite_message.
 -d:Debug on.
```

测试：

```
sms -f 139xxxx -p abc123 -t 138 -m "test by sun" -a test -d
成功收到短信表示飞信设置成功！
```

### 6.3.4. 授权 sms 文件可被 nagios 执行

同 gnokii 一样，sms 也需要 nagios 可执行。如果 sms 的属主不是 nagios 帐户，那么需要修改 sms 的权限，保证 nagios 帐户可执行该程序：

```
chmod +x /usr/bin/sms
或 chmod 711 /usr/bin/sms
```

### 6.3.5. 绑定飞信功能到 Nagios

6.3.4.1. 编辑 commands.cfg，加入飞信命令 host-notify-by-fei 和 service-notify-by-fei  
vi ./nagios/etc/objects/commands.cfg

```
define command {
 command_name host-notify-by-fei ;如果有短信猫的话，注意命名要有区别
 command_line /usr/bin/sms -f 13412345678 -p abc123 -t $CONTACTPAGER$ -m
 "Host $HOSTSTATE$ alert for $HOSTNAMES$! on '$LONGDATETIMES$' " $CONTACTPAGER$
}

define command {
 command_name service-notify-by-fei
 command_line /usr/bin/sms -f 13412345678 -p abc123 -t $CONTACTPAGER$ -m
 "$HOSTADDRESS$' $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATES$ on
 '$LONGDATETIMES$' " $CONTACTPAGER$
}
```

6.3.4.2. 编辑联系人，增加飞信的告警方式

两个联系人，一个联系人是 C 网手机，只能用短信猫告警；另一个联系人是移动的手机，可以用飞信。

第一个联系人是前面在 6.2.4.2 小节处介绍过的，在此配置文件的基础上增加一个联系人，使用飞信告警方式，注意两种 command 的区别。

```
联系人 1 #####
define contact{
 contact_name sun ; Short name of user
 use generic-contact ; Inherit
 default values from generic-contact template (defined above)
 alias Nagios Admin ; Full
 name of user
 service_notification_commands service-by-sms ;短信猫发送告警
 host_notification_commands host-by-sms
 email testing@netease.com
 pager 13312345678
}
联系人 2
```

```

define contact{
 contact_name liwei ; Short name of user
 use generic-contact ; Inherit
 default values from generic-contact template (defined above)
 alias Nagios Admin ; Full
 name of user
 service_notification_commands service-notify-by-fei ; 飞信告警
 host_notification_commands host-notify-by-sms
 email liwei@cvtt.cn
 pager 13812345678
}

```

断开短信猫，并断掉一个被监控目标，不出意外的话，在设定的时间范围内会收到短信告警。

## 七. WEB 授权

如果需要为每个收取告警短信的联系人都提供 WEB 接口，让其从 WEB 上查看所负责的设备信息，同时还不能查看到其他设备的信息。如何操作？举例。

### 7.1. 定义联系人

使用 `htpasswd` 来添加新帐户，1.4 节中用过这个命令。需要新增的帐户必须在 `contacts.cfg` 文件中已经定义。

```
vi nagios/etc/object/contacts.cfg
```

```

define contactgroup{
 contactgroup_name shanxitaiyuan
 alias Nagios Administrators
 members testing
}

```

### 7.2. 为联系人添加 WEB 登录帐号。

从上面的配置文件可以看到 `testing` 的帐号归属山西，现为该用户新建一个 WEB 帐号。

```
htpasswd /usr/local/nagios/etc/htpasswd.users testing
```

### 7.3. 修改 cgi 为新增帐号授权

修改 `cgi.cfg` 的 120 行，在行末将新增用户添加进来。多用户以逗号分隔。

```
Vi /usr/local/nagios/etc/cgi.cfg
```

## 7.4. 帐户登录

完成上述工作后，就可以登录了，登录页面只能看到所管理的设备。

| Service Status Details<br>For All Hosts |                                                  |              |                     |                    |               |                                                                                                                                             |
|-----------------------------------------|--------------------------------------------------|--------------|---------------------|--------------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Host<br>↑↓                              | Service<br>↑↓                                    | Status<br>↑↓ | Last Check<br>↑↓    | Duration<br>↑<br>↓ | Attempt<br>↑↓ | Status Information                                                                                                                          |
| <a href="#">SX-<br/>[redacted]</a>      | <a href="#">CPU Load</a>                         | OK           | 12-19-2008 13:09:49 | 0d 0h 7m 57s       | 1/3           | CPU Load 0% (5 min average)                                                                                                                 |
|                                         | <a href="#">Important Process<br/>[redacted]</a> | OK           | 12-19-2008 13:04:21 | 0d 0h 34m 16s      | 1/2           | [redacted].exe: Running -<br>[redacted].exe: Running -<br>[redacted].exe: Running -<br>[redacted].exe: Running -<br>[redacted].exe: Running |
|                                         | <a href="#">NSClient++<br/>Version</a>           | OK           | 12-19-2008 12:38:49 | 0d 0h 38m 57s      | 1/3           | NSClient++ 0.3.5.2 2008-09-24                                                                                                               |
|                                         |                                                  |              |                     |                    |               | System Uptime - 8 day(s) 11                                                                                                                 |

## 八. 用 MRTG 查看历史状态

MRTG 的功能很多，最普遍的用法是检测网络流量，它也可以与 Nagios 很好的结合，以图形方式直观显示监控目标的过去一段时间的状态。在没安装 MRTG 之前，nagios 页面的左侧 REPORT 下的 Trends 项是无法打开的，只能从第二项开始，这样得到的各项数据都是表格形式的，晦涩难懂。Nagios3.0.3 的状态工具默认情况下就可以调用 MRTG 绘制历史数据，基本满足了大部分功能，我们需要做的工作只是把 MRTG 安装完毕。

### 8.1. 准备工作已完善的安装方法

安装 mrtg 前需要注意，/usr/local/nagios/sbin/statusmap.cgi 是否存在，或者看 nagios 左面的 status map 项是否能正常打开。如果 OK，那么 MRTG 的安装会是一件非常容易的工作。

```
rpm -ivh mrtg*.rpm
```

如果有依赖包，还不想自己找的话：

```
yum -y install mrtg
```

当安装完成后，nagios 就可以使用 mrtg 查看主机、进程等目标的历史状态了。

### 8.2. 准备工作不完善的安装方法

如果不能打开 status map 的页面，MRTG 也能装，方法同上，不过没法绘制历史数据，装了白装。解决方法：



8.2.1. 首先/usr/local/nagios/sbin/statusmap.cgi 肯定不存在，导致这个问题的原因就是第一章准备工作的 gd-devel 没装上。确认一下：

```
rpm -qa |grep gd-devel
```

什么都没显示就表示没装上，由于这个包的依赖比较多，建议使用 yum 安装：

```
yum -y install gd-devel
```

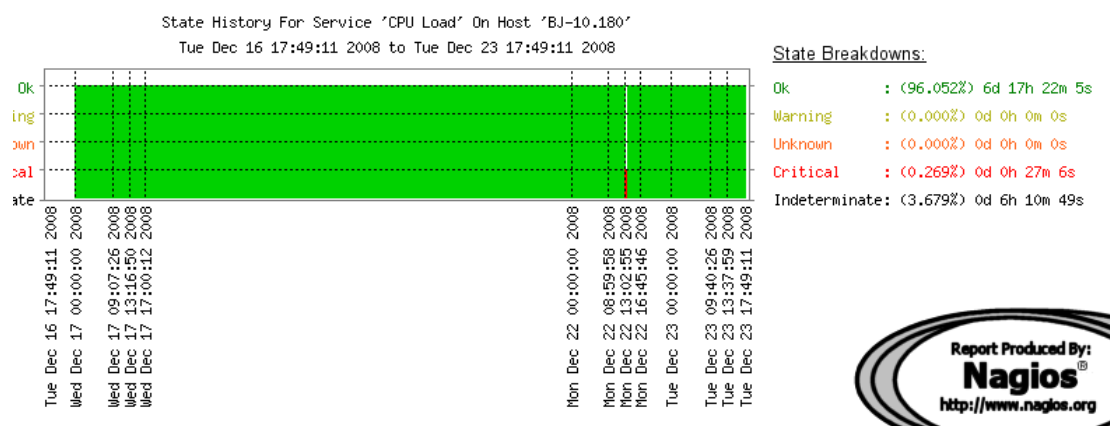
8.2.2. 安装 gd-devel 后，重新编译 nagios-3.0.3，方法同前。

编译完成后，/usr/local/nagios/sbin/statusmap.cgi 就出来了，MRTG 也能用了。

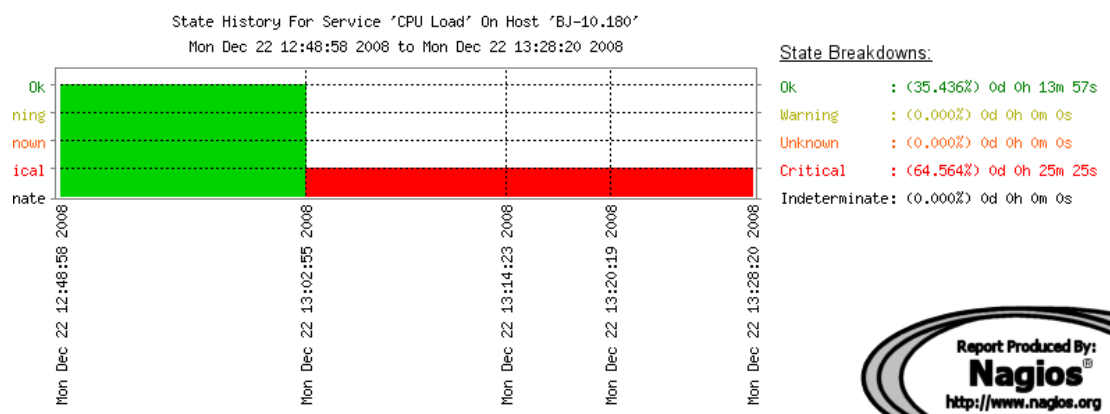
**注意：**重新编译 nagios-3.0.3 前记得把所有的配置文件都备份出来，编译完成后再覆盖回去。如果忘了备份，配置也没丢，只是在原来的文件名后面加上了个“~”，每个文件分别改名，麻烦点而已。

## 8.3. 历史数据查看

用 nagios 查看监控目标的历史记录。点击页面 report 下的第一项 Trends，按提示进行，会显示出指定设备或服务最近一段时间的状态图。下图是某台设备近一周的 CPU 占用情况。其中有一段红色，按前面定义的阈值，有一段时间 CPU 资源被占用超过 90%。本例安装 Linux 的时候没装中文字体，所以不能用中文页面。



点击红色部分可放大，仔细查看 CPU 占用峰值的时间范围



## 九. 用 PNP 绘制历史曲线

Pnp 是一个基于 php 和 perl, 用 rrdtool 将 nagios 采集的数据绘制图表的工具, 所以安装 pnp 之前必须先安装 php、perl 和 rrdtool。

### 9.1 准备工作

光盘中提供如下软件包

- php-common-5.1.6-20.el5.i386.rpm
- php-gd-5.1.6-20.el5.i386.rpm
- php-devel-5.1.6-20.el5.i386.rpm
- php-cli-5.1.6-20.el5.i386.rpm
- php-5.1.6-20.el5.i386.rpm
- php-devel-5.1.6-20.el5.i386.rpm
- perl-URI-1.35-3.noarch.rpm
- perl-String-CRC32-1.4-2.fc6.i386.rpm
- perl-IO-Socket-INET6-2.51-2.fc6.noarch.rpm
- perl-5.8.8-10.el5\_0.2.i386.rpm
- perl-Socket6-0.19-3.fc6.i386.rpm

安装过程略

其他安装包需要从互联网下载:

- rrdtool-1.0.50.tar.gz  
推荐使用 1.0.50, 安装简便, 1.2 以后的版本安装极其不便。而且“据说”1.0.50 的效率要高于 1.2。而且, 1.2 以后的版本需要另找 cgilib 等库文件, 麻烦。
- pnp-0.4.14.tar.gz
- cgilib-0.5.tar.gz (这是 1.2 以后版本需要的)

### 9.2 安装 PNP

#### 9.2.1. 安装 cgilib

```
tar zxvf cgilib-0.5.tar.gz
cd cgilib-0.5
cp libcgi.a /usr/local/lib ##记错咧? 文件咋没咧?
cp cgi.h /usr/include
```

#### 9.2.2. 安装 rrdtool

```
tar zxvf rrdtool-1.0.50.tar.gz
```

```
cd rrdtool
./configure
make
make install
```

### 9.2.3. 安装 pnp

```
tar zxvf pnp-0.4.14.tar.gz
cd pnp-0.4.14
./configure -- with-rrdtool=/usr/local/rrdtool-1.0.50/bin/rrdtool
make all
make install
make install-config
make install-init
```

注意: ./configure 完成后会出现安装结果, 注意下面的红色粗体字, 说 RRDs Perl Modules 在大量安装时可以加速, 暂忽略之, 继续。

\*\*\* Configuration summary for pnp 0.4.14 05-02-2009 \*\*\*

#### General Options:

|                             |                                       |
|-----------------------------|---------------------------------------|
| Nagios user/group:          | nagios nagios                         |
| Install directory:          | /usr/local/nagios                     |
| HTML Dir:                   | /usr/local/nagios/share/pnp           |
| Config Dir:                 | /usr/local/nagios/etc/pnp             |
| Location of rrdtool binary: | /usr/local/rrdtool-1.0.50/bin/rrdtool |

Version 1.0.50

#### **RRDs Perl Modules:**

**\*\*\* NOT FOUND \*\*\***

|                                  |                                       |
|----------------------------------|---------------------------------------|
| RRD Files stored in:             | /usr/local/nagios/share/perfdata      |
| process_perfdata.pl Logfile:     | /usr/local/nagios/var/perfdata.log    |
| Perfdata files (NPCD) stored in: | /usr/local/nagios/var/spool/perfdata/ |

Review the options above for accuracy. If they look okay,  
type 'make all' to compile.

**WARNING:** The RRDs Perl Modules are not found on your System  
Using RRDs will speedup things in larger Installtions.

如果使用加速功能, 从网上找到解决方法 (未测试过), 用下面的命令:

```
ln -sv \
/usr/local/rrdtool/lib/perl/5.8.8/i386-linux-thread-multi/auto/RRDs/RRDs.so \
/usr/lib/perl5/5.8.8/i386-linux-thread-multi/
```

之后继续 make all

## 9.2.4. 让 apache 的首页默认页支持 php 格式

- 编辑 httpd.conf, 在 DirectoryIndex 后面加上 index.php:

`DirectoryIndex index.html index.html.var index.php`

- 重新启动 apache 服务

访问 `http://localhost/nagios/pnp/index.php?host=localhost` 能看到图形

如果 nagios 尚未采集数据, 看到可能是下面的样式。这个是安装 rrdtool-1.2.30 的提示, rrdtool-1.0.50 也是一样。

```

OK Initialising
OK Using /usr/local/nagios/share/perfdata/
OK RRDTool /usr/local/rrdtool-1.2.30/bin/rrdtool found.
OK RRDTool /usr/local/rrdtool-1.2.30/bin/rrdtool is executable
OK PHP Function proc_open is enabled
OK PHP Function fpassthru is enabled
OK PHP Function xml_parser_create is enabled
OK PHP zlib Support found.
OK PHP GD Support found.
OK RRD Base Directory /usr/local/nagios/share/perfdata/ found.
OK Hostname localhost is set.
! Directory /usr/local/nagios/share/perfdata/localhost not found.

```

## 9.3 将绘图功能加入 nagios

### 9.3.1. 编辑 nagios.cfg

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

```
0 改为 1
process_performance_data=1
去掉 “#”

host_perfdata_command=process-host-perfdata
service_perfdata_command=process-service-perfdata
```

### 9.3.2. 修改绘图命令

将原来的 `command_line` 后的内容替换为下面的内容

```

define command{
 command_name process-host-perfdata
 command_line /usr/local/nagios/libexec/process_perfdata.pl
}
define command{
 command_name process-service-perfdata
 command_line /usr/local/nagios/libexec/process_perfdata.pl
}

```

### 9.3.3. 修改监控目标主机文件

在目标主机配置文件中加入如下所示的红色粗体字



















```

define host{
 use linux-server
 host_name localhost
 alias localhost
 address 127.0.0.1
 check_interval 2
 retry_check_interval 1
 max_check_attempts 2
 process_perf_data 1
 action_url /nagios/pnp/index.php?host=$HOSTNAME$
}

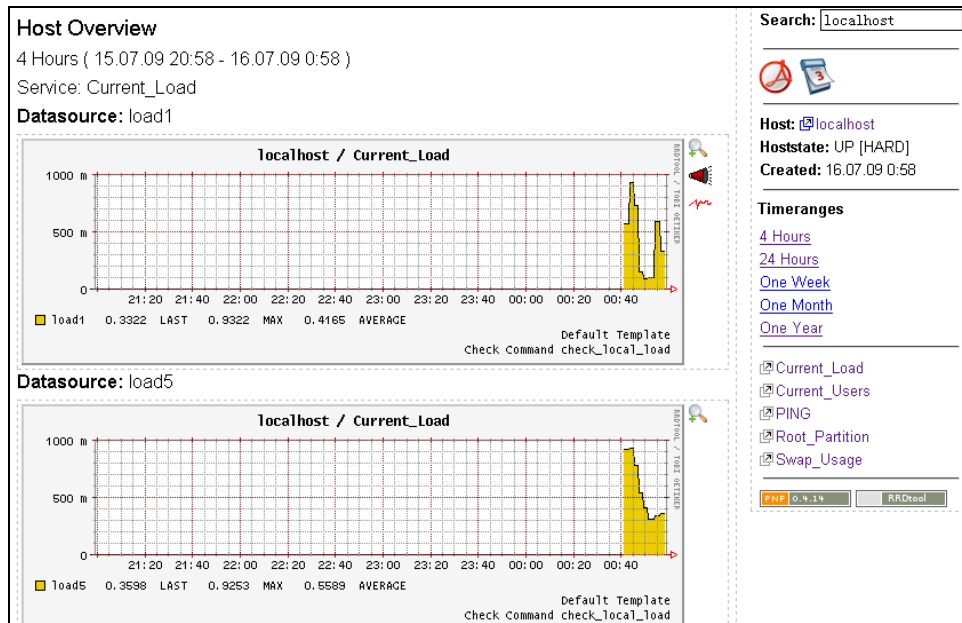
```

如果你要监控的目标主机都需要绘制曲线的话，可以在 1.4 节中介绍的模板中添加上述红色字体，这样就不用每个主机都添加了，又节省好大一笔工作量，赚了。

重启 nagios，打开监控页面后，就能看到监控目标主机旁边有红色的类似小太阳的图标。每个 service 后面都带图标是因为测试时，在每项服务后面都加了上面的那两行命令，但使用中发现没什么用，每次点击单个服务后的太阳图标，照样把其他服务的历史曲线带出来。

| Host ↑↓                   | Service ↑↓                                                                                                                                                                                   | Status ↑↓                                                                                   | Last Check ↑↓       | Duration ↑↓  |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|---------------------|--------------|
| <a href="#">localhost</a> |  <a href="#">Current Load</a>                                                                             |  OK      | 07-16-2009 00:53:51 | 0d 2h 8m 26s |
|                           |  <a href="#">Current Users</a>                                                                            |  OK      | 07-16-2009 00:52:29 | 0d 2h 7m 48s |
|                           |   <a href="#">HTTP</a> |  WARNING | 07-16-2009 00:52:18 | 0d 2h 7m 11s |
|                           |  <a href="#">PING</a>                                                                                     |  OK      | 07-16-2009 00:53:45 | 0d 2h 6m 33s |
|                           |  <a href="#">Root Partition</a>                                                                           |  OK      | 07-16-2009 00:53:45 | 0d 2h 5m 56s |
|                           |   <a href="#">SSH</a>  |  OK      | 07-16-2009 00:52:17 | 0d 2h 5m 18s |
|                           |  <a href="#">Swap Usage</a>                                                                               |  OK      | 07-16-2009 00:52:36 | 0d 2h 4m 41s |
|                           |  <a href="#">Total Processes</a>                                                                          |  OK      | 07-16-2009 00:54:10 | 0d 2h 4m 3s  |

当 nagios 获取到数据以后，点击红色的太阳图标就能看到绘图曲线了。



### 9.3.4. 自定义图表时间范围

上图右侧可以看到绘图的时间范围——Timeranges，最短 4 小时。但有时我们需要查看更短时间范围的曲线时，默认的页面就办不到了，需要手工修改一下配置文件：

/usr/local/nagios/etc/pnp/config.php

关于时间范围定义的默认配置是这样的：

```
$views[0]["title"] = "4 Hours";
$views[0]["start"] = (60*60*4);

$views[1]["title"] = "24 Hours";
$views[1]["start"] = (60*60*24);

.....
```

在配置里增加 30 分钟和 1 小时、2 小时的时间间隔，在上述配置的前面增加下面的内容，注意将 view[] 的数字重新排序。

```
注意需要重新将 views 后面的值重新排序。
$views[0]["title"] = "30 Minutes";
$views[0]["start"] = (60*60*1/2);
$views[1]["title"] = "1 Hour";
$views[1]["start"] = (60*60*1);
$views[2]["title"] = "2 Hours";
$views[2]["start"] = (60*60*2);
```

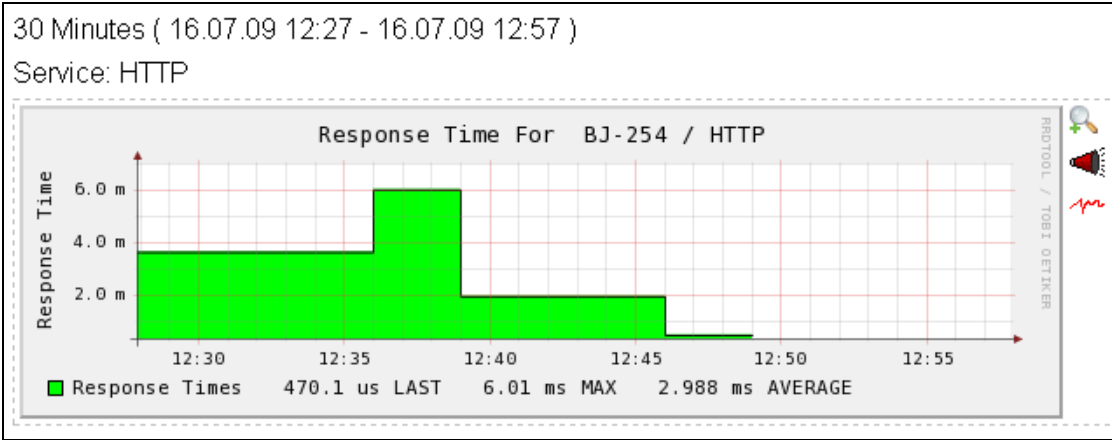
此外，配置文件的第 86 行为默认打开时以哪个时间范围显示图表。

```
$conf['overview-range'] = 1;
```

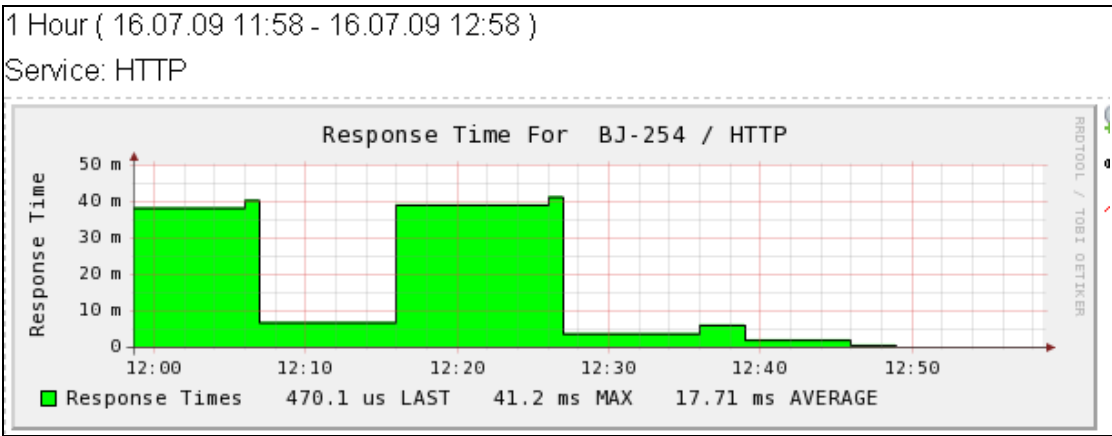
按上面的配置，默认打开时，图表的时间范围是 1 小时内的。可酌情修改。

上述参数修改完毕后重新启动 nagios 就可以了，打开页面在右侧 Timeranges 下可以看到新增加了 3 个时间范围——30 分钟、1 小时、2 小时。依次点击查看。

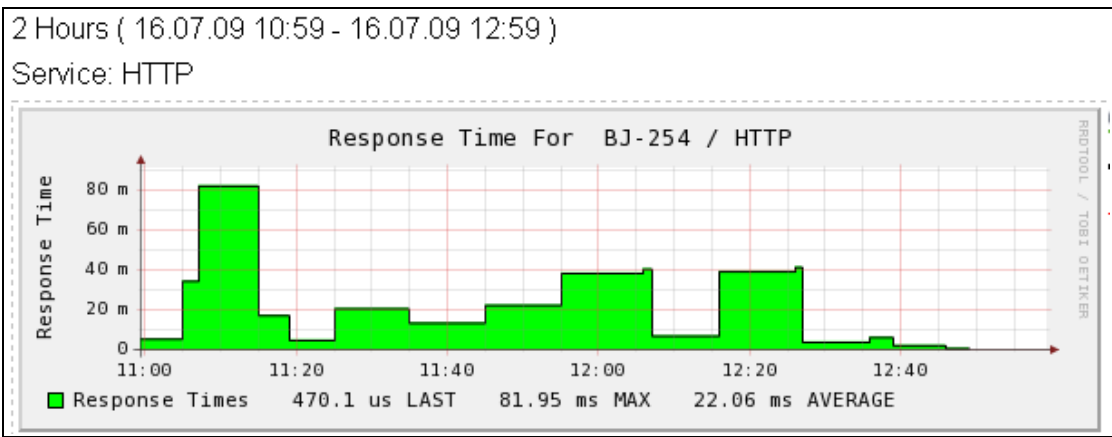
最近半小时的曲线图



最近 1 小时的曲线图



最近两小时的曲线图



## 十. 关于浏览器

近一段时间使用情况看，使用 Firefox 浏览 nagios 页面要比 IE 更好一点。最重要的一点是 Firefox 有个针对 nagios 的插件，只要打开了 Firefox，即使不打开 nagios 页面，也可以从

插件获取到被控目标的告警信息。Firefox 的缺点在于默认的配置占用资源比较高，需要大量的优化配置和插件，很烦。而且，Firefox 的界面风格不符合部分人群的审美观和使用习惯。

# 十一. Nagios 3.1.2 的安装

安装步骤基本相同。由于 nagios3.1.x 的首页不再是 html 文件，而是 php 文件，有几项注意事项：

1. Httpd.conf 里新增 index.php

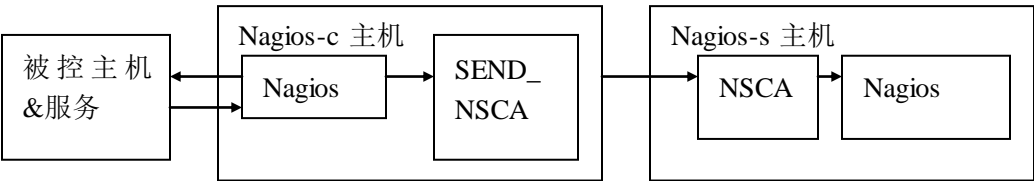
DirectoryIndex index.html index.html.var *index.php*

2. 需要 php 支持。Php 可以在安装 nagios 之前安装，也可以后装，没具体要求。php 的安装包光盘自带，如果你用到了上面的所有功能，那就已经装完了。

3. 如果是升级安装，注意先将原有配置文件备份。如果没有备份，原有的配置文件都会在后面增加一个“~”，需要逐个改名。

# 十二.Nagios 的分布式部署（NSCA）

当 Nagios 想监控数百至数千台主机和数倍于主机数量的服务时，或者监控一组处于防火墙内部的机器时，或者遇到 nagios 主机的性能瓶颈时，或者因网络不可达时而无法实现监控需求，此时就需要用到分布式部署。具体实现方法是保留现有的 Nagios-server，并安装 NSCA。在远端另建一个 Nagios 主机，命名为 Nagios-client，按前面介绍的方法将所要监控的目标纳入 Nagios-client 的监控范围，再安装 NSCA 插件。Nagios-client 通过 NSCA 模块中的 send\_nsc 将获取到的信息发送给 Nagios-server 的 NSCA 中。这样，Nagios-server 不需要耗费本机性能去发起监控请求，也不需要穿越防火墙，在原有监控内容基础上新增了 Nagios-client 的监控内容。



NSCA 是一个软件包同时安装在服务端（Nagios-s）和客户端（Nagios-c 主机）上，配置方法有一定区别，下面分别介绍。

## 12.1 安装

软件包版本为 NSCA-2.7.2，下载地址：

<http://nchc.dl.sourceforge.net/sourceforge/nagios/nsca-2.7.2.tar.gz>

服务端和客户端安装方式一样



```
./configure --with-nsca-user=nagios --with-nsca-grp=nagios
make all
```

## 12.2 客户端(nagios-client 服务器)配置

NSCA 在客户端上所用的命令为 `send_nsca`，配置文件为 `send_nsca.cfg`。安装完毕后，在解压包里解压出来的文件夹中将这两个文件均复制到 `nagios` 安装目录下

### 12.2.1 复制文件

```
cp sample-config/send_nsca.cfg /usr/local/nagios/etc/
cp src/send_nsca /usr/local/nagios/bin/
```

修改这两个文件的属主为 `nagios` 组和用户，命令略。

### 12.2.2 修改配置文件

修改 `send_nsca.cfg` 的内容

```
password=abcd1234 # 此密码必须与服务端的密码一致
ocsp_command=submit_check_result
use_syslog=0
enable_notifications=0
obsess_over_services=1
check_host_freshness=1
check_service_freshness=1
```

### 12.2.3 新增命令

- 在 `/nagios/libexec` 中增加 `submit_check_result` 命令，命令内容：

```
#!/bin/bash
Arguments:
$1 = host_name (Short name of host that the service is
associated with)
$2 = svc_description (Description of the service)
$3 = state_string (A string representing the status of
the given service - "OK", "WARNING", "CRITICAL"
or "UNKNOWN")
$4 = plugin_output (A text string that should be used
as the plugin output for the service checks)
#
Convert the state string to the corresponding return code
```

```

return_code=-1
case "$3" in
OK)
return_code=0
;;
WARNING)
return_code=1
;;
CRITICAL)
return_code=2
;;
UNKNOWN)
return_code=-1
;;
esac

pipe the service check info into the send_nscd program, which
in turn transmits the data to the nscd daemon on the central
monitoring server
/usr/bin/printf "%s\t%s\t%s\t%s\n" "$1" "$2" "$return_code" "$4" |
/usr/local/nagios/bin/send_nscd 192.168.1.100 -c /usr/local/nagios/etc/send_nscd.cfg
注意，上面斜体字的 IP 地址为 Nagios 的 NSCA 服务端所在地址，根据情况自行修改

```

- 修改此文件修改权限为 nagios 用户可执行

```
chown -R nagios:nagios submit_check_result
```

```
chmod 711 submit_check_result
```

- commands.cfg 中增加调用新命令的内容

```

define command{
command_name submit_check_result
command_line /usr/local/nagios/libexec/submit_check_result
$HOSTNAMES$ '$SERVICEDESC$' $SERVICESTATES$ '$SERVICEOUTPUT$'
}

```

## 12.2.4 修改客户端配置

在监控目标的配置文件中主机项和服务项的下面均新增两行配置文件，或者可以修改模板，见 1.4 节。

```

Define hosts{
.....
.....
check_freshness 1
freshness_threshold 20
}

```

客户端的配置到此结束，可以启动或重新启动 Nagios 了

## 12.3 服务端的安装

### 12.3.1 复制文件

服务端的命令和配置文件分别是 `nscd` 和 `nscd.cfg`，复制到 `nagios` 的对应目录下

```
cp sample-config/nscd.cfg /usr/local/nagios/etc/
cp src/nscd /usr/local/nagios/bin/
```

修改这两个文件的属主为 `nagios` 组和用户。

### 12.3.2 修改 `nscd.cfg` 并测试客户端连通性

- Password=abcd1234，与客户端 `send_nscd.cfg` 中的密码相同。
- 启动 NSCA 守护进程  
`/usr/local/nagios/bin/nscd -d -c /usr/local/nagios/etc/nscd.cfg`  
守护进程默认服务端口 TCP 5667。此命令可放在 `rc.local` 下随系统启动。
- 在客户机上如下操作测试连通性  
新建配置文件 `test`，内容随便写。

```
[root@CLIENT ~]# cd /usr/local/nagios/
[root@CLIENT nagios]# bin/send_nscd 192.168.164.141 -c etc/send_nscd.cfg </root/test
0 data packet(s) sent to host successfully.
```

### 12.3.3 复制客户端的监控目标配置文件

用 NSCA 时，客户端下的被控目标配置文件，需要在服务端上建立副本，增加部分内容并重新添加一次，`nagios` 版本已经升级到 3.2.0，仍然没有一个便捷的方式，很头疼但很无奈。

在服务端新建个目录，将客户端监控目标的配置文件复制过来，都放到这个目录下。新建目录的原因是方便管理和避免混淆或文件重名被替换。

修改复制过来的各个监控目标配置文件，这项工作比较讨厌。在原有主机项内容下新增 3 行，注意不要因为这个需求去修改模板中的默认项，新增一个服务项比较好。

```
Define hosts{

 check_freshness 0 # 此处是原有配置，客户端的值为 1
 passive_checks_enabled 1
 active_checks_enabled 0
 flap_detection_enabled 0
}
```

在 Service 项下新增与 Host 相同内容之后再增加一行，同样不要去修改模板。

```
Define hosts{

 is_volatile 0
}
```

### 12.3.4 将复制过来的客户端配置文件加入 nagios.cfg

采用前面讲述的方式将客户端拷贝来的文件路径和文件名写入服务端的 nagios.cfg，由服务端的 Nagios 调用。

### 12.3.5 重新启动 NSCA 守护进程和 nagios






```
/usr/local/nagios/bin/nsca -d -c /usr/local/nagios/etc/nsca.cfg
默认端口为 TCP 5667
service nagios restart
```

### 12.3.6 NSCA 客户端和服务端界面

● 客户端

| Host      | Service         | Status | Last Check          | Duration        |
|-----------|-----------------|--------|---------------------|-----------------|
| localhost | Current Load    | OK     | 05-12-2010 16:16:01 | 215d 3h 16m 37s |
|           | Current Users   | OK     | 05-12-2010 16:16:01 | 215d 3h 4m 18s  |
|           | HTTP            | OK     | 05-12-2010 16:16:01 | 0d 0h 35m 44s   |
|           | PING            | OK     | 05-12-2010 16:16:01 | 215d 3h 3m 3s   |
|           | Root Partition  | OK     | 05-12-2010 16:16:01 | 215d 3h 2m 26s  |
|           | SSH             | OK     | 05-12-2010 16:16:01 | 215d 3h 1m 48s  |
|           | Swap Usage      | OK     | 05-12-2010 16:16:01 | 215d 3h 1m 11s  |
|           | Total Processes | OK     | 05-12-2010 16:16:01 | 215d 3h 0m 33s  |
| winserver | Share Port      | OK     | 05-12-2010 16:16:01 | 0d 3h 22m 10s   |

● 服务端

| Host ↑↓                     | Service ↑↓                      | Status ↑↓                                                                            | Last Check ↑↓       | Duration ↑↓   |
|-----------------------------|---------------------------------|--------------------------------------------------------------------------------------|---------------------|---------------|
| <a href="#">Local_Host</a>  | <a href="#">Current Load</a>    | OK                                                                                   | 05-12-2010 16:17:13 | 0d 3h 7m 3s   |
|                             | <a href="#">Current Users</a>   | OK                                                                                   | 05-12-2010 16:19:06 | 0d 3h 6m 25s  |
|                             | <a href="#">HTTP</a>            | OK                                                                                   | 05-12-2010 16:15:58 | 0d 3h 5m 48s  |
|                             | <a href="#">PING</a>            | OK                                                                                   | 05-12-2010 16:17:51 | 0d 3h 5m 10s  |
|                             | <a href="#">Root Partition</a>  | OK                                                                                   | 05-12-2010 16:14:43 | 0d 3h 4m 33s  |
|                             | <a href="#">SSH</a>             | OK                                                                                   | 05-12-2010 16:16:36 | 0d 3h 3m 55s  |
|                             | <a href="#">Swap Usage</a>      | OK                                                                                   | 05-12-2010 16:18:28 | 0d 3h 3m 18s  |
|                             | <a href="#">Total Processes</a> | OK                                                                                   | 05-12-2010 16:15:21 | 0d 3h 2m 40s  |
| <a href="#">localhost</a> ✖ | <a href="#">Current Load</a>    |  OK | 05-12-2010 16:19:21 | 0d 3h 4m 44s  |
|                             | <a href="#">Current Users</a>   |  OK | 05-12-2010 16:19:21 | 0d 3h 5m 39s  |
|                             | <a href="#">HTTP</a>            |  OK | 05-12-2010 16:19:21 | 0d 3h 1m 50s  |
|                             | <a href="#">PING</a>            |  OK | 05-12-2010 16:19:21 | 0d 3h 5m 32s  |
|                             | <a href="#">Root Partition</a>  |  OK | 05-12-2010 16:19:21 | 0d 3h 5m 35s  |
|                             | <a href="#">SSH</a>             |  OK | 05-12-2010 16:19:21 | 0d 3h 5m 34s  |
|                             | <a href="#">Swap Usage</a>      |  OK | 05-12-2010 16:19:21 | 0d 3h 5m 37s  |
|                             | <a href="#">Total Processes</a> |  OK | 05-12-2010 16:19:21 | 0d 3h 5m 36s  |
| <a href="#">winserver</a> ✖ | <a href="#">Share Port</a>      |  OK | 05-12-2010 16:19:21 | 0d 2h 51m 28s |

两个界面上显示 NSCA 主机运行时间不一致，原因不明，待查。

官方文档上还有一个命令没用上，不知所云。

```
define command{
command_name check_dummy
command_line $USER1$/check_dummy $ARG1$
}
```

## 十三. Cacti 的安装

### 13.1 安装 mysql、php、rrdtool、snmp

#### 13.1.1. 安装 net-snmp

```
rpm -ivh\
net-snmp-utils-5.3.2.2-5.el5.i386.rpm\
net-snmp-perl-5.3.2.2-5.el5.i386.rpm\
net-snmp-devel-5.3.2.2-5.el5.i386.rpm\
beecrypt-devel-4.1.2-10.1.1.i386.rpm\
elfutils-devel-static-0.137-3.el5.i386.rpm\
elfutils-devel-0.137-3.el5.i386.rpm
```

#### 13.1.2. 安装 php

由于 CentOS 5.2/3 的 yum 源提供的 php 目前版本为 5.1.6 不能够支持 json, 因此需要一个 php 的扩展源

在/etc/yum.repos.d 生成 utt erramblings.repo, 内容如下

```
[utt erramblings]
name=Jason's Utter Ramblings Repo
baseurl=http://www.jasonlitka.com/media/EL$releasever/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://www.jasonlitka.com/media/RPM-GPG-KEY-jlitka
```

```
yum -y install php php-mysql php-gd php-pdo php-json php-snmp
```

#### 13.1.3. 安装 rrdtool-1.0.50

```
tar zxvf rrdtool-1.0.50.tar.gz
./configure --prefix=/usr/local/rrdtool/ && make && make install
```

#### 13.1.4. 解压 cacti-0.8.7e 并将解压目录复制或移动到网页默认目录下

```
tar zxvf cacti-0.8.7e
mv cacti-0.8.7e /var/www/html/cacti
```

#### 13.1.5. 安装 mysql

```
mysql-connector-odbc
mysqlclient1
mysql
mysql-server
```

如果需要把 cacti 和 nagios 整合到一起, 还需要 mysql-devel. 最省事的方法就是 mysql\*全装。

### 13.2. mysql 的配置

#### 13.2.1. 创建 cacti 数据库并用 cacti.sql 生成表

```
#mysql
>create database cacti;
>use cacti;
>source /var/www/html/cacti/cacti.sql;
或者
#mysql -u cactiuser -p cacti < /var/www/html/cacti/cacti.sql
```

```
+-----+
| Tables_in_cacti |
+-----+
| cdef |
| cdef_items |
| colors |
| data_input |
| data_input_data |
| data_input_fields |
| data_local |
| data_template |
| data_template_data |
| data_template_data_rra |
| data_template_rrd |
| graph_local |
| graph_template_input |
| graph_template_input_defs |
| graph_templates |
| graph_templates_gprint |
| graph_templates_graph |
| graph_templates_item |
| graph_tree |
| graph_tree_items |
| host |
| host_graph |
| host_snmp_cache |
| host_snmp_query |
| host_template |
| host_template_graph |
| host_template_snmp_query |
| poller |
| poller_command |
| poller_item |
| poller_output |
| poller_reindex |
| poller_time |
| rra |
```

|                         |  |
|-------------------------|--|
| rra_cf                  |  |
| settings                |  |
| settings_graphs         |  |
| settings_tree           |  |
| snmp_query              |  |
| snmp_query_graph        |  |
| snmp_query_graph_rrd    |  |
| snmp_query_graph_rrd_sv |  |
| snmp_query_graph_sv     |  |
| user_auth               |  |
| user_auth_perms         |  |
| user_auth_realm         |  |
| user_log                |  |
| version                 |  |
| +-----+                 |  |

### 13.2.2. 创建 mysql 的帐户 cactiuser，密码 cacti

```
#mysql
>grant all on cacti.* to cactiuser@localhost identified by "cacti";
> flush privileges;
```

有时 flush 会出现错误: **ERROR 1146 (42S02): Table 'mysql.servers' doesn't exist**  
解决方法

```
mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> CREATE TABLE `servers` (
 `Server_name` char(64) NOT NULL,
 `Host` char(64) NOT NULL,
 `Db` char(64) NOT NULL,
 `Username` char(64) NOT NULL,
 `Password` char(64) NOT NULL,
 `Port` int(4) DEFAULT NULL,
 `Socket` char(64) DEFAULT NULL,
 `Wrapper` char(64) NOT NULL,
 `Owner` char(64) NOT NULL,
 PRIMARY KEY (`Server_name`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='MySQL Foreign Servers
table';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```



### 13.3. 修改 cacti 的 config.php 参数

```
vi ../cacti/include/config.php
$database_password = "cacti";
```

新建系统用户 cacti，并将/cacti/rra 属主改为 cacti

```
root#useradd cacti
root#chown -R cacti:cacti /cacti/rra
root# su - cacti
cacti$ php /var/www/html/cacti/poller.php >/dev/null 2>&1
```

定制计划任务，每 5 分钟执行一次

```
root#vi /etc/crontab
*/5 * * * * cacti php /var/www/html/cacti/poller.php > /dev/null 2>&1
```

### 13.4. 登陆 cacti

至此，cacti 安装完毕，可在浏览器上访问 [http://\\$ip/cacti](http://$ip/cacti)

默认密码 admin/admin，第一次登录时 cacti 的会要求修改密码

同时 cacti 会有一个页面要求输入 rrdtool、php、snmp 等应用程序的绝对路径

**Cacti Installation Guide**

Make sure all of these values are correct before continuing.

**[NOT FOUND] RRDTOOL Binary Path:** The path to the rrdtool binary.  
/usr/local/rrdtool-1.0.50/bin  
**[ERROR: FILE NOT FOUND]**

**[FOUND] PHP Binary Path:** The path to your PHP binary file (may require a php recompile to get this file).  
/usr/bin/php  
**[OK: FILE FOUND]**

**[NOT FOUND] snmpwalk Binary Path:** The path to your snmpwalk binary.  
/usr/bin/snmpwalk  
**[ERROR: FILE NOT FOUND]**

**[NOT FOUND] snmpget Binary Path:** The path to your snmpget binary.  
/usr/bin/snmpget  
**[ERROR: FILE NOT FOUND]**

**[NOT FOUND] snmpbulkwalk Binary Path:** The path to your snmpbulkwalk binary.  
/usr/bin/snmpbulkwalk  
**[ERROR: FILE NOT FOUND]**

**[NOT FOUND] snmpgetnext Binary Path:** The path to your snmpgetnext binary.  
/usr/bin/snmpgetnext  
**[ERROR: FILE NOT FOUND]**

**[FOUND] Cacti Log File Path:** The path to your Cacti log file.  
/var/www/html/cacti/log/cacti.log  
**[OK: FILE FOUND]**

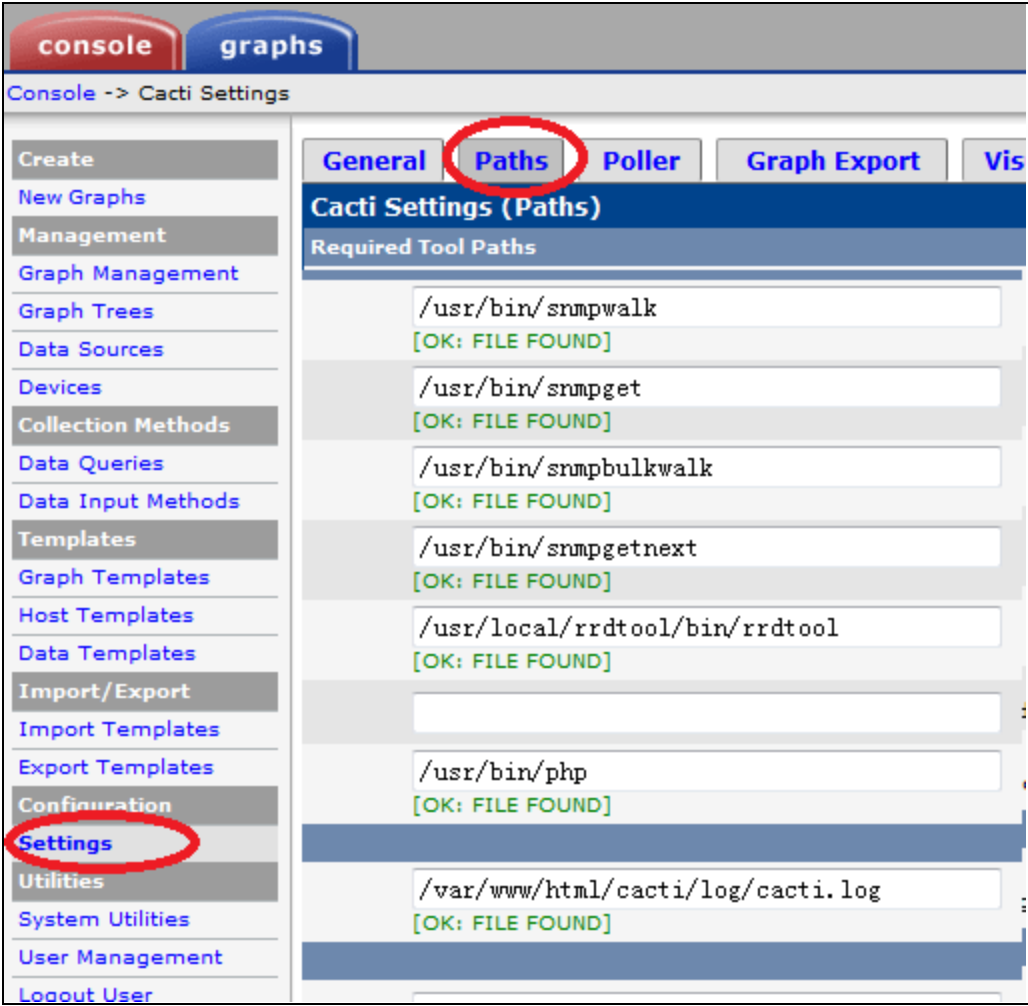
**SNMP Utility Version:** The type of SNMP you have installed. Required if you are using SNMP v2c or don't have embedded SNMP support in PHP.  
NET-SNMP 5.x

**NOTE:** Once you click "Finish", all of your settings will be saved and your database will be upgraded if this is an upgrade. You can change any of the settings on this screen at a later time by going to "Cacti Settings" from within Cacti.

**Finish**

如果这些路径不正确或文件缺失，在登陆后可以在侧边栏 configure 的子项 setting-→path 重

新填写



如果主机配置时出现 snmp error 时，修改 snmp.conf 中 62 行的参数，将 systemview 修改为 all。

|        |                   |     |        |       |            |      |      |
|--------|-------------------|-----|--------|-------|------------|------|------|
| access | notConfigGroup "" | any | noauth | exact | <u>all</u> | none | none |
|--------|-------------------|-----|--------|-------|------------|------|------|

注意 snmp 端口的安全

## 第二部分 cacti 整合到 nagios 中

### 1. cacti-plugin-arch 的安装

解压 cacti-plugin 的压缩文件到 cacti-plugin 目录下，，然后将其中的.diff 文件移动或复制到 cacti 的目录，

```
#unzip cacti-plugin-0.8.7e-PA-v2.5.zip -d cacti-plugin
#mv cacti-plugin/cacti-plugin* /var/www/html/cacti/
#cd var/www/html/cacti
#patch -p1 -N < cacti-plugin-0.8.7e-PA-v2.5.diff
输出结果
```

```
patching file auth_changepassword.php
patching file auth_login.php
patching file data_sources.php
patching file graph_image.php
patching file graph.php
patching file graphs_new.php
patching file graphs.php
patching file host.php
patching file include/auth.php
patching file include/bottom_footer.php
patching file include/global_arrays.php
patching file include/global_constants.php
patching file include/global_form.php
patching file include/global.php
patching file include/global_settings.php
patching file include/plugins.php
patching file include/top_graph_header.php
patching file include/top_header.php
patching file index.php
patching file lib/api_device.php
patching file lib/auth.php
patching file lib/functions.php
patching file lib/html_form.php
patching file lib/html.php
patching file lib/plugins.php
patching file lib/poller.php
patching file lib/rrd.php
patching file lib/variables.php
patching file plugins/index.php
```

```
patching file plugins.php
patching file poller.php
patching file user_admin.php
patching file utilities.php
```

执行 cacti-plugin 目录中 pa.sql 的数据库脚本，在 cacti 数据库中创建 4 个表，plugin\_config、plugin\_db\_changes、plugin\_hooks、plugin\_realms

```
#mysql -u$cactiuser -p$cactipasswd $cactidb <pa.sql
$cactiuser=cacti 数据库的用户名，与前面的 u 之间没有空格
$cactipass=cacti 用户的密码，同样无空格
$cactidb=cacti 用到的默认数据库
```

修改 cacti/include/global.php 文件

```
$config['url_path'] = '/cacti/';
```

这个 url 需要修改，按照自己的配置改吧。

## 2 ndoutils-1.4b 安装

这个东西是将 Nagios 的配置及监控信息存储到数据库里，NPC 通过调用 ndo 所存储的数据来展现 Nagios 的信息。

获取：

[http://sourceforge.net/project/showfiles.php?group\\_id=26589&package\\_id=173832&release\\_id=550909](http://sourceforge.net/project/showfiles.php?group_id=26589&package_id=173832&release_id=550909)

这个需要用到 mysql 的 mysql-lib 及 mysql-inc，因此需要安装 mysql-devel (yum install mysql-devel)

解压后，进入目录执行

```
./configure --disable-pgsql --with-ndo2db-user=nagios --with-ndo2db-group=nagios
```

执行 make 来进行编译

编译完成后（不需要 make install）

复制 ndoutils-1.4b7/src 目录下的可执行文件到 nagios/bin 下，复制配置文件到 nagios/etc 下

```
#cp src/ndomod-3x.o nod2db-3x log2ndo file2sock /usr/local/nagios/bin
#cp config/ndo2db.cfg /usr/local/nagios/etc
#cp config/ndomod.cfg etc/ /usr/local/nagios/etc
```

配置 ndo2db.cfg

```
| socket_name=/var/nagios/ndo.sock
| db_name=cacti
| db_prefix=np_
| db_user=cacti
```

```
| db_pass=xxxxx
| debug_level=1
| debug_file=/usr/local/nagios/var/nagios/ndo2db.debug
```

以上为需要根据自己的实际情况配置的信息，其他配置可以根据自己需求来进行配置。

配置 ndomod.cfg，以下指定的两个默认路径是源码安装 nagios 时的路径

```
output=/usr/local/nagios/var/nagios/ndo.sock
buffer_file=/usr/local/nagios/var/nagios/ndomod.tmp
```

配置 nagios.cfg 文件

```
check_external_commands=1
command_check_interval = -1
event_broker_options = -1
broker_module = /usr/bin/ndomod-3x.o config_file=/usr/local/nagios/etc/ndomod.cfg
process_performance_data=1
process_performance
```

以上为需要更改的 nagios.cfg 的内容。

### 3 Npc 的安装

解压到

```
tar zxvf npc-2.0.4.tar.gz
```

解压，然后将整个文件夹 cp 至 cacti/plugins/下

```
cp -a npc /var/www/html/cacti/plugins/
```

修改 cacti/include/global.php

添加：

```
$plugins[] = 'npc';
```

让 cacti 知道有这个插件

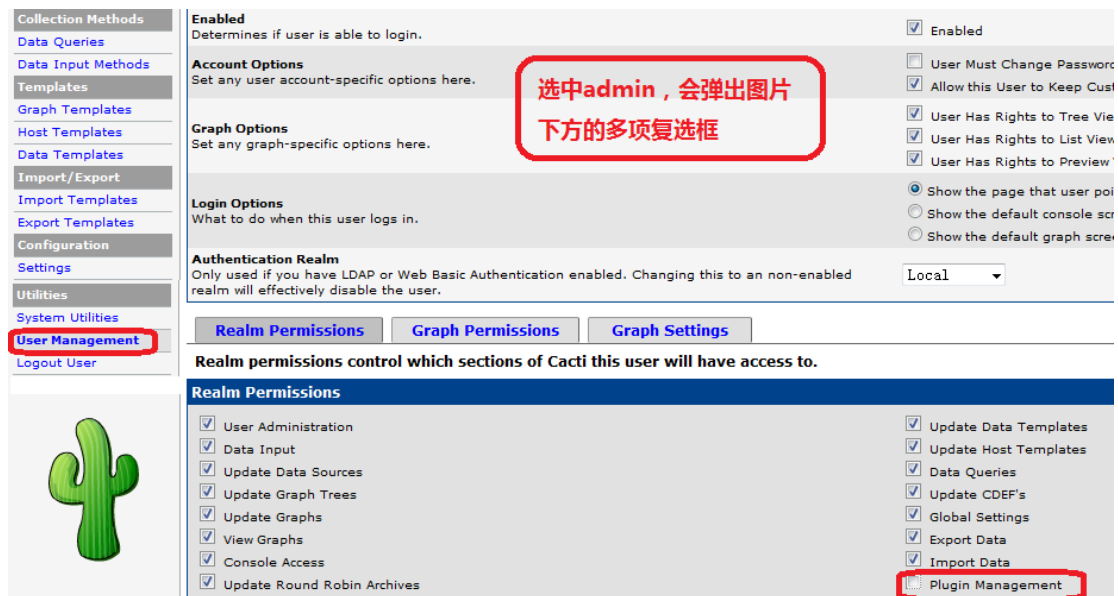
进入 <http://yourserver/cacti/>

点击用户管理

选中 admin

在页面下方会显示出多项复选框

选择插件管理

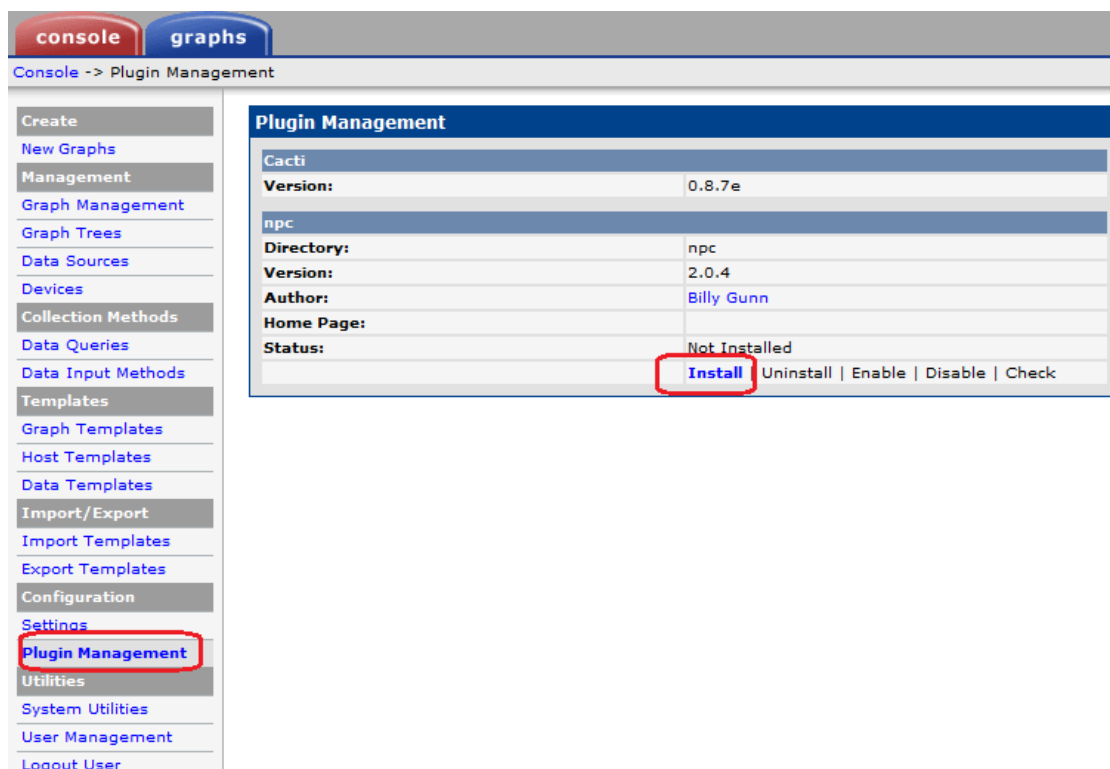


插件管理勾选后，网页侧边栏的 configuration——settings 下方会新增一项插件管理——

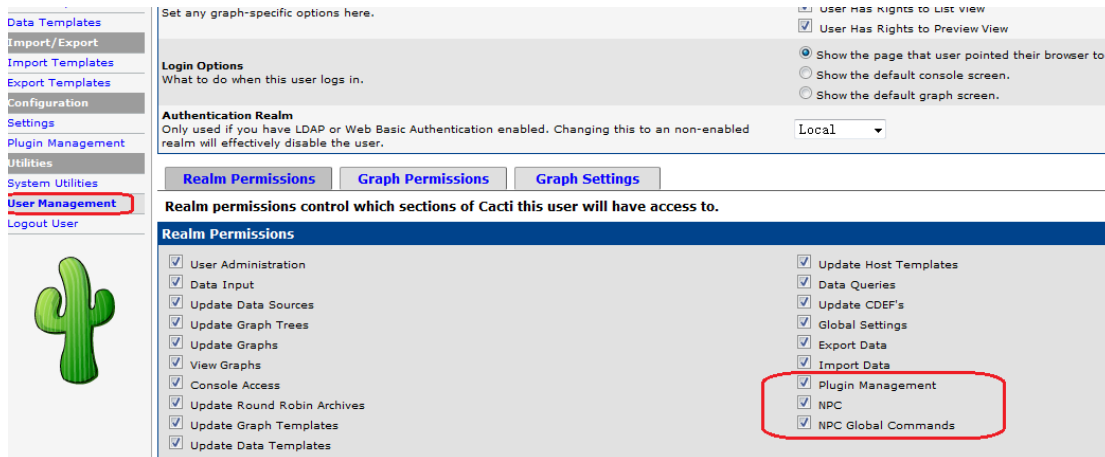
Plugin Managerment

点击插件管理后，右侧出现 npc 安装页面，点击 Install，待页面刷新完毕后点击 Enable。

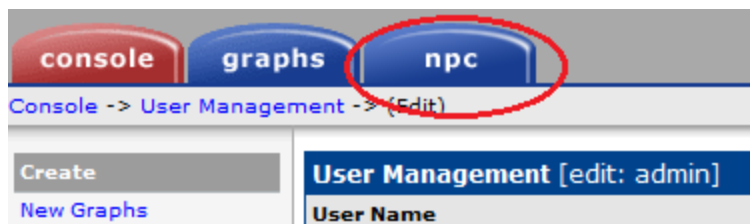
Cacti 数据库中会新增数十个 npc 前缀的表



再次点击 User Managerment，在右侧中间选中 admin，弹出界面的 Realm Permissions 右下方出现 npc 等选项，确认已经勾选。



Cacti 页面上方原有的 console 和 graphs 两个选项的右侧出现 npc 选项



接着在左侧栏目中选择 settings, 點選 npc 的标签

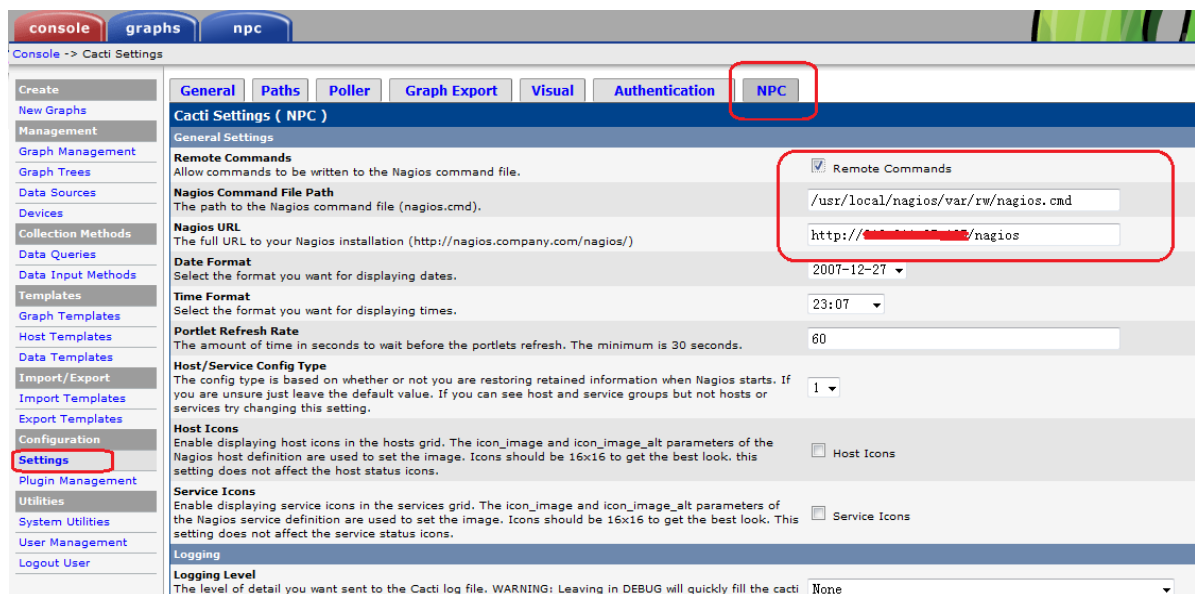
勾选 Remote Commands

Nagios Command File Path: `/var/nagios/rw/nagios.cmd`

<这个文件启动 nagios 后会产生，根据实际的位置写>

Nagios URL: <http://yourserver/nagios/>

保存就可以了。



至此安装就完成了

启动 mysql httpd ndo nagios

```
service mysqld start
service httpd start
/usr/bin/ndo2db-3x -c /etc/nagios/ndo2db.cfg
service nagios start
```