目录

1. 实验：集群节点故障诊断软件安装 nagios 安装与配置

1.1. 实验目的

- 安装 nagios 监控管理软件
- 监控 hadoop 平台硬件资源状态 （监测 node1 和 node2，node3 上的资源空间分布情况）
- 监控 Hadoop 日志信息
- 监控 hdfs 的健康状态
- 监控 datanode 个数

1.2. 实验要求

- 熟悉 Hadoop 命令
- 熟悉 Linux 常用命令
- 了解 nagios 功能

1.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1-1 资源环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4
用户名/密码	root/password hadoop/password nagios/password
服务和组件	完成前面章节实验，其他服务及组件根据实验需求安装 nagios-4.0.8

1.4. 实验过程

1.4.1. 实验任务一：Nagios 服务端安装

1.4.1.1. 步骤一：基础套件安装

```
   [root@master ~]# yum install httpd php php-cli gcc glibc glibc-common gd
gd-devel net-snmp
   [root@master ~]# yum -y install openssl-devel
```

1.4.1.2. 步骤二：创建用户和分组

```
[root@master ~]# useradd -m nagios            #新建用户
[root@master ~]# passwd nagios                #修改密码
输入密码 password
[root@master ~]# groupadd nagcmd              #新建分组
[root@master ~]# usermod -a -G nagcmd nagios   #添加用户到分组
```

1.4.1.3. 步骤三：安装 Nagios

```
[root@master ~]#tar xzf /opt/software/nagios-4.0.8.tar.gz  #解压
[root@master ~]#cd nagios-4.0.8
[root@master nagios-4.0.8]# ./configure --with-command-group=nagcmd
[root@master nagios-4.0.8]# make all
[root@master nagios-4.0.8]# make install
[root@master nagios-4.0.8]# make install-init
[root@master nagios-4.0.8]# make install-config
[root@master nagios-4.0.8]# make install-commandmode
[root@master nagios-4.0.8]# make install-webconf    #安装 Web 界面
[root@master             nagios-4.0.8]#             htpasswd             -c
/usr/local/nagios/etc/htpasswd.users nagiosadmin
输入密码 password
#为 web 界面创建登录账号，其中 nagiosadmin 为账号名可更改，同时会提示添加密码。
重启 Apache 服务
[root@master nagios-4.0.8]# cd ~
[root@master ~]# /bin/systemctl restart httpd.service
```

1.4.1.4. 步骤四：安装 plugins

```
[root@master ~]#tar xzf /opt/software/nagios-plugins-2.0.3.tar.gz  #解压
[root@master ~]#cd nagios-plugins-2.0.3
[root@master nagios-plugins-2.0.3]# ./configure --with-nagios-user=nagios
--with-nagios-group=nagios
[root@master nagios-plugins-2.0.3]# make    #编译安装
[root@master nagios-plugins-2.0.3]# make install
[root@master nagios-plugins-2.0.3]# service nagios start
```

默认安装目录：/usr/lib64/nagios/plugins/ ，安装完毕，可以进入 web 界面了

网址: http://master/nagios/ 登陆名 nagiosadmin 密码:password(前面设置的那个密码)

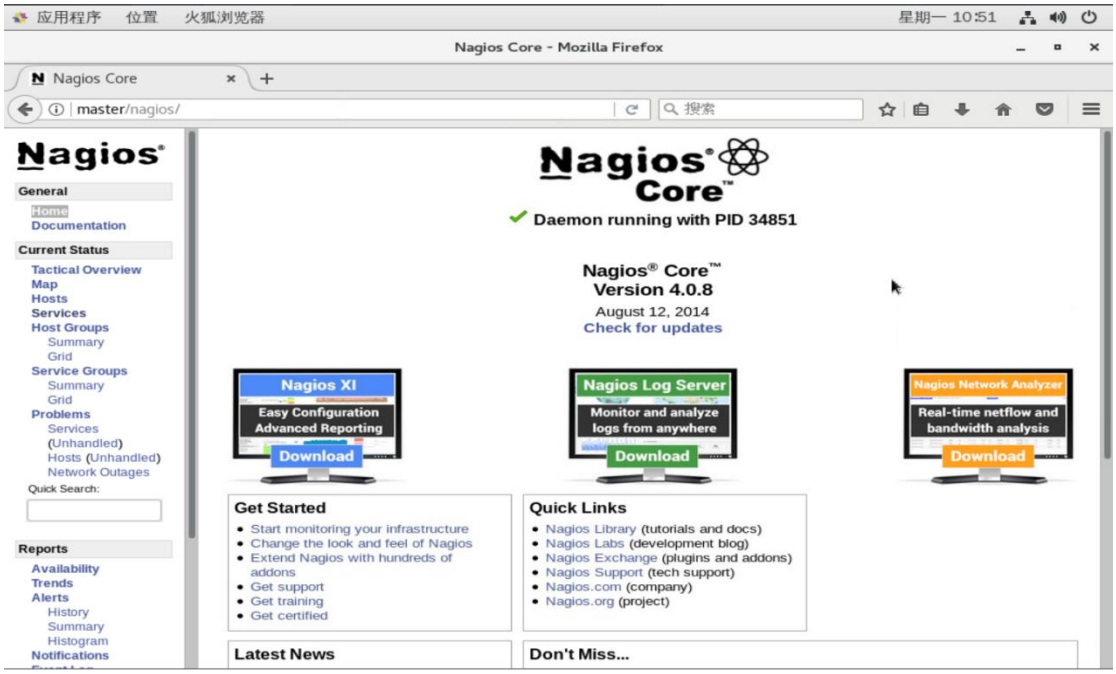


图 1-1 组件首页

若出现 nginx error，则修改

```
[root@master nagios-plugins-2.0.3]#vim /etc/httpd/conf/httpd.conf
```

将 Listen 80 改为 888，则网址为：http://master:888/nagios

```
# configuration, error, and log files are kept.
#
# Do not add a slash at the end of the directory path.  If you point
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# Mutex directive, if file-based mutexes are used.  If you wish to share t
he
# same ServerRoot for multiple httpd daemons, you will need to change at
# least PidFile.
#
ServerRoot "/etc/httpd"

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 888
```

图 1-2 配置文件

1.4.2. 实验任务二：Nagios 目录名称及作用

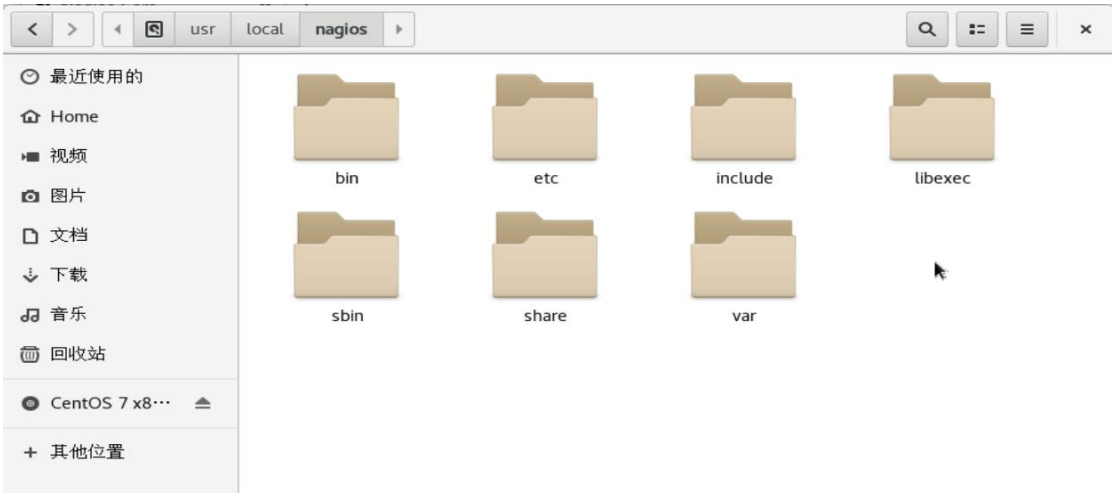


图 1-3 本地文件视图

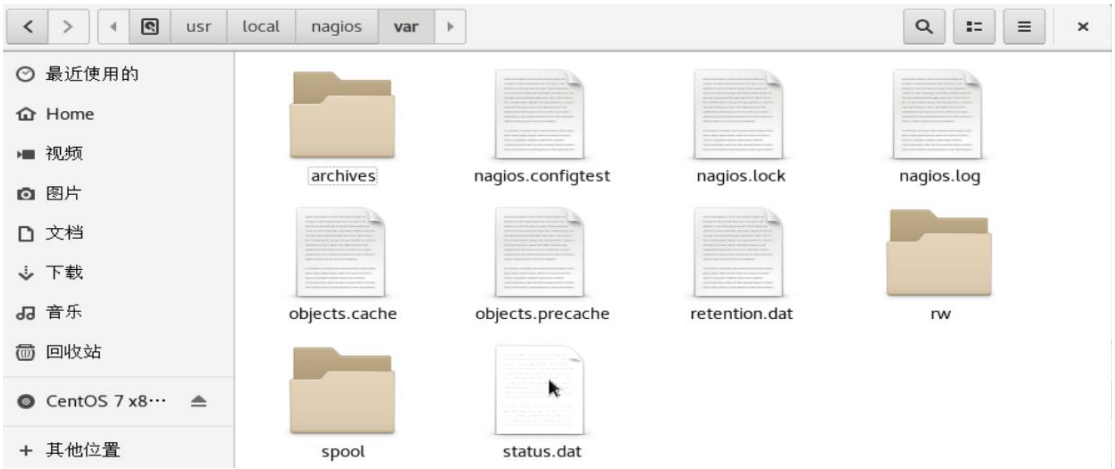


图 1-4 本地文件视图

目录名称	作用
bin	Nagios 可执行程序所在目录
etc	Nagios 配置文件目录
sbin	Nagios cgi 文件所在目录， 也就是执行外部 命令所需要文件所在的目录
share	Nagios 网页存放路径
libexec	Nagios 外部插件存放目录
var	Nagios 日志文件、Lock 等文件所在的目录
var/archives	agios 日志自动归档目录

var/rw	用来存放外部命令文件的目录

1.4.3. 实验任务三：Nagios 配置监控 node1 资源

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：

1.4.3.1. 步骤一：创建 hosts.cfg 文件来定义主机和主机组

```
[root@master ~]# cd /usr/local/nagios/etc/objects
[root@master objects]# vim hosts.cfg
define host{
       use                     linux-server
       host_name               node2
       alias                   Nagios-node2
       address                 192.168.90.146;(slave1)
       }
define host{
       use                     linux-server
       host_name               node3
       alias                   Nagios-node3
       address                192.168.90.20;(slave2)
       }
define hostgroup{
       hostgroup_name          bsmart-servers
       alias                   bsmart servers
       members                 node2,node3
       }
```

1.4.3.2. 步骤二：创建 services.cfg 文件来定义服务

```
[root@master objects]# vim services.cfg
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 node1
       ;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
       check_command check-host-alive
       ;指定检查的命令
       }
```

1.4.3.3. 步骤三：编辑 localhost.cfg 文件

```
 [root@master objects]# vim localhost.cfg
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             node1
          alias                 node1
          address              192.168.90.233;(master)
          }
   define hostgroup{
          hostgroup_name  node1 ; The name of the hostgroup
          alias          node1 ; Long name of the group
          members         node1     ; Comma separated list of hosts that belong
to this group
          }
   define service{
          use                           local-service        ; Name of service
template to use
          host_name                     node1
          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                     node1
          service_description            Root Partition
       check_command          check_local_disk!20%!10%!/
          }



   # Define a service to check the number of currently logged in
   # users on the local machine.  Warning if > 20 users, critical
   # if > 50 users.

   define service{
          use                           local-service        ; Name of service
template to use
          host_name                     node1
          service_description            Current Users
       check_command          check_local_users!20!50
          }
   define service{
          use                           local-service        ; Name of service
template to use
          host_name                     node1
          service_description            Total Processes
       check_command          check_local_procs!250!400!RSZDT
          }



   # Define a service to check the load on the local machine.
```

```
   define service{
          use                             local-service         ; Name of service
template to use
          host_name                       node1
          service_description             Current Load
       check_command          check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
          }



   # Define a service to check the swap usage the local machine.
   # Critical if less than 10% of swap is free, warning if less than 20% is
free

   define service{
          use                             local-service         ; Name of service
template to use
          host_name                       node1
          service_description             Swap Usage
       check_command          check_local_swap!20!10
          }
   define service{
          use                             local-service         ; Name of service
template to use
          host_name                       node1
          service_description             SSH
       check_command          check_ssh
       notifications_enabled       1
          }



   # Define a service to check HTTP on the local machine.
   # Disable notifications for this service by default, as not all users may
have HTTP enabled.

   define service{
          use                             local-service         ; Name of service
template to use
          host_name                       node1
          service_description             HTTP
       check_command          check_http
       notifications_enabled       1
          }
```

1.4.3.4. 步骤四：编辑 nagios.cfg 挂载各个文件文件

```
    [root@master objects]# cd ..
    [root@master etc]# vim nagios.cfg
   添加
   cfg_file=/usr/local/nagios/etc/objects/hosts.cfg
   cfg_file=/usr/local/nagios/etc/objects/services.cfg
```

注：在以后修改配置文件后，可以利用以下命令行检测配置文件是否正确，可以根据错误提示修改

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

修改完配置文件后 web 界面不会立马修改，需要重新加载、启动 nagios 服务。

```
[root@master ~]# service nagios reload
[root@master ~]# service nagios restart
```

登陆。点击 services,视图 1-5 如下(需要等待一段时间)

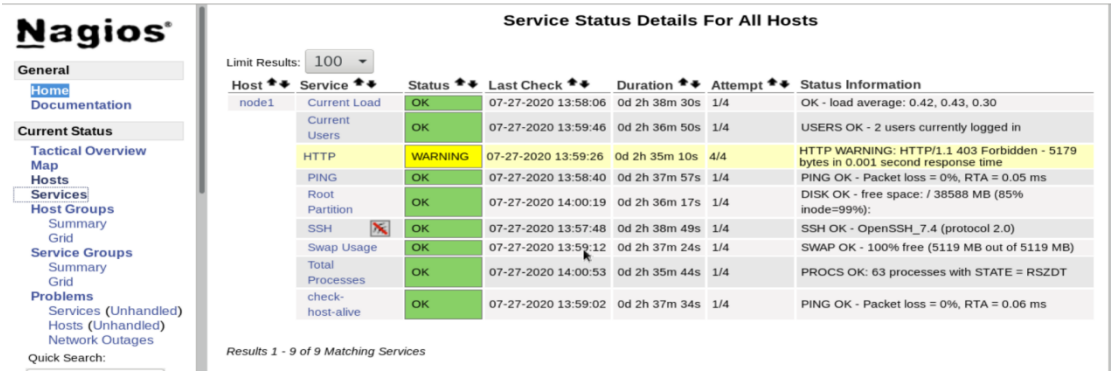


图 1-5 本地资源视图

监控状态分为：

正常 OK、警告 Warning、不知道 Unknown、严重错误 Critical、监控中 Pending

status information 可以看到检测对应信息，各 service 对应作用如下：

名称	作用	插件名
Current Load	CPU 负载	check_load
Current Users	登录系统用户数	check_users
HTTP	网站运行状态	check_http
PING	ping	check_ping
ROOT Partition	根分区	check_disk
SSH	监控 ssh	check_ssh
Swap Usage	交换分区	check_swap
Total Processes	总的进程数量	check_procs

其中 HTTP 状态为 WARNING，之所以警告，是因为没有任何索引文件，并且目录列表被禁用

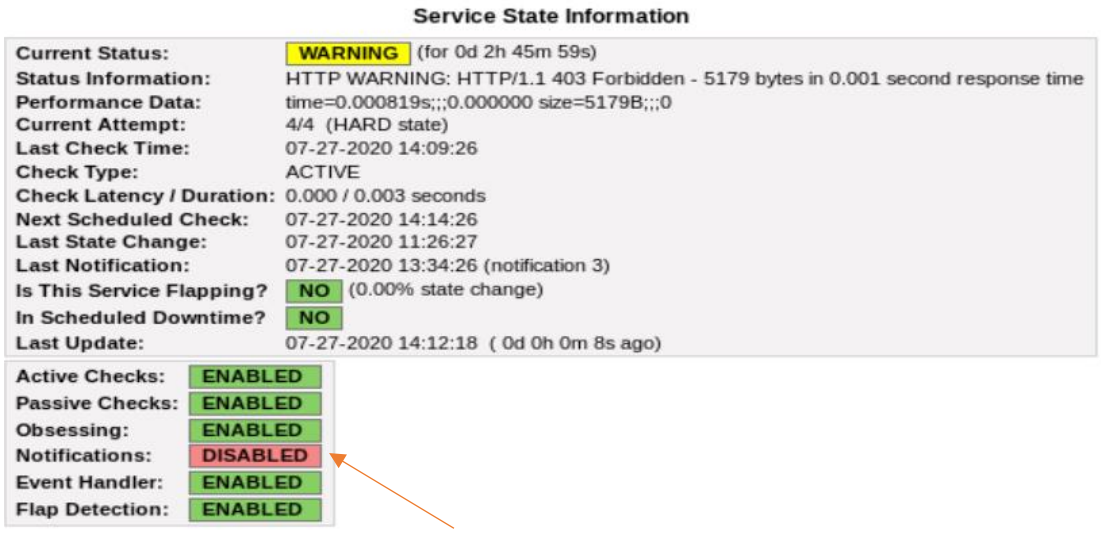


图 1-6 本地资源详情

解决方法

```
[root@master ~]# touch /var/www/html/index.html #创建一个新的 index 文件
[root@master ~]# service nagios reload  #重启服务即可生效
[root@master ~]# service nagios restart
```

图 1-7 本地资源视图

1.4.4. 实验任务四：Nagios 配置监控 node2，node3 资源

1.4.4.1. 步骤一：在被监控端(node2、node3)安装基础套件

```
    [root@slave1 ~]# yum install httpd php php-cli gcc glibc glibc-common gd
gd-devel net-snmp
    [root@slave2 ~]# yum install httpd php php-cli gcc glibc glibc-common gd
gd-devel net-snmp
```

1.4.4.2. 步骤二：在被监控端(node2、node3)安装 plugins

#slave1 和 slave2 操作一致，下面不重复展示

```
[root@slave1 ~]# useradd nagios     #增加用户&设定密码
[root@slave1 ~]# passwd nagios
输入密码 password
[root@master     ~]#     scp     /opt/software/nagios-plugins-2.0.3.tar.gz
root@slave1:/root/
[root@slave1 ~]# tar xzf nagios-plugins-2.0.3.tar.gz
[root@slave1 ~]# cd nagios-plugins-2.0.3
[root@slave1 nagios-plugins-2.0.3]# ./configure --with-nagios-user=nagios
--with-nagios-group=nagios
[root@slave1 nagios-plugins-2.0.3]# make
[root@slave1 nagios-plugins-2.0.3]# make install
#修改目录权限
[root@slave1 nagios-plugins-2.0.3]# chown nagios.nagios /usr/local/nagios
[root@slave1     nagios-plugins-2.0.3]#     chown     -R     nagios.nagios
/usr/local/nagios/libexec
```

1.4.4.3. 步骤三：在被监控端(node2、node3)安装 nrpe

```
[root@slave1 ~]# cd ~
[root@master ~]# scp /opt/software/nrpe-2.15.tar.gz root@slave1:/root/
[root@slave1 ~]# tar -zxf nrpe-2.15.tar.gz
[root@slave1 ~]# cd nrpe-2.15
[root@slave1 nrpe-2.15]# yum -y install openssl-devel
[root@slave1 nrpe-2.15]# ./configure --enable-command-args
[root@slave1 nrpe-2.15]# make all
#接下来安装 NPRE 插件，daemon 和示例配置文件。
 [root@slave1 nrpe-2.15]# make install-plugin   #安装 plugin 这个插件
 [root@slave1 nrpe-2.15]# make install-daemon   #安装 deamon
 [root@slave1 nrpe-2.15]# make install-daemon-config    #安装配置文件
 [root@slave1 nrpe-2.15]# make install-xinetd   #安装 xinted 脚本
[root@slave1 nrpe-2.15]# vim /usr/local/nagios/etc/nrpe.cfg    # 修 改 ##
nagios 服务器主机地址
allowed_hosts=192.168.90.233# (node1 地址)
#可以看到创建了这个文件/etc/xinetd.d/nrpe。
[root@slave1 nrpe-2.15]# vim /etc/xinetd.d/nrpe
#在 only_from 后增加监控主机的 IP 地址
only_from      = 127.0.0.1 192.168.90.233# (node1 地址)
[root@slave1 nrpe-2.15]# vim /etc/services  #编辑/etc/services 文件,增加 NRPE
服务
#最后一行添加
nrpe    5666/tcp        #  nrpe
[root@slave1 nrpe-2.15]# service xinetd restart#重启服务
在启动 xinetd.service 时提示错误信息
         Redirecting to /bin/systemctl restart  xinetd.service
         Failed to issue method call: Unit xinetd.service failed to load:
No such file or directory.
```

1）首先，检查服务器已安装的 tftp-server

```
[root@slave1 ~]# cd ~
[root@slave1 ~]# rpm -qa | grep tftp-server
```

如果存在已安装的 tftp 这里会列出来

2)安装 tftp-server 和 xinetd

使用如下的命令，进行相应服务的安装：

```
[root@slave1 ~]# yum -y install tftp-server
[root@slave1 ~]# yum -y install xinetd
```

3）修改 tftp 配置文件

使用如下命令：

```
[root@slave1 ~]# vim /etc/xinetd.d/tftp  #打开配置文件
    service tftp
    {
        socket_type         = dgram
        protocol        = udp
        wait            = yes
        user            = root
        server        = /usr/sbin/in.tftpd
        server_args         = -s /var/lib/tftpboot
        disable        = no //需要修改的地方，初始时刻为 yes
        per_source         = 11
        cps            = 100 2
        flags            = IPv4
    }
#使用如下命令进行服务的重新启动
[root@slave1 ~]# /bin/systemctl restart xinetd.service
#查看 NRPE 是否已经启动
[root@slave1 ~]# netstat -tnlp
```

```
1013/sshd
tcp6       0      0 :::5666                 :::*                    LISTEN
39120/xinetd
[root@slave1 nrpe-2.15]#
```

图 1-8 监听端口

可以看到 5666 端口已经在监听了。

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。

```
[root@slave1 ~]# systemctl stop firewalld.service
[root@slave1 ~]# cd /usr/local/nagios/etc
[root@slave1 etc]# vim nrpe.cfg #修改路径为/dev/sda1
```

```
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/s
da1
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
```

图 1-9 修改配置文件

```
[root@slave1 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 #修改为 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
```

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。

1.4.4.4. 步骤四：在监控端(node1)安装 check_nrpe

```
[root@master ~]# tar -zxf /opt/software/nrpe-2.15.tar.gz
[root@master ~]# cd nrpe-2.15
[root@master nrpe-2.15]# ./configure --enable-command-args
[root@master nrpe-2.15]# make all
[root@master nrpe-2.15]# make install-plugin
[root@master nrpe-2.15]# make install-daemon
[root@master nrpe-2.15]# make install-daemon-config#安装配置文件
[root@master nrpe-2.15]# make install-xinetd    #安装 xinted 脚本
[root@master nrpe-2.15]# vim /usr/local/nagios/etc/nrpe.cfg    # 修 改 ##
nagios 服务器主机地址
allowed_hosts=192.168.90.233 (node1 地址)
#可以看到创建了这个文件/etc/xinetd.d/nrpe。
[root@master nrpe-2.15]# vim /etc/xinetd.d/nrpe
#在 only_from 后增加监控主机的 IP 地址
only_from       = 127.0.0.1 192.168.90.233 (node1 地址)
[root@master nrpe-2.15]# vim /etc/services  #编辑/etc/services 文件,增加 NRPE
服务
#最后一行添加
nrpe     5666/tcp        #  nrpe
```

```
[root@master nrpe-2.15]# service xinetd restart #重启服务
```

在启动 xinetd.service 时提示错误信息

```
        Redirecting to /bin/systemctl restart  xinetd.service
        Failed to issue method call: Unit xinetd.service failed to load:
No such file or directory.
```

1）首先，检查服务器已安装的 tftp-server

```
[root@master ~]# cd ~
[root@master ~]# rpm -qa | grep tftp-server
```

如果存在已安装的 tftp 这里会列出来

2)安装 tftp-server 和 xinetd

使用如下的命令，进行相应服务的安装：

```
[root@master ~]# yum -y install tftp-server
[root@master ~]# yum -y install xinetd
```

3）修改 tftp 配置文件

使用如下命令：

```
[root@master ~]# vim /etc/xinetd.d/tftp  #打开配置文件
    service tftp
    {
        socket_type        = dgram
        protocol        = udp
        wait            = yes
        user            = root
        server        = /usr/sbin/in.tftpd
        server_args        = -s /var/lib/tftpboot
        disable        = no //需要修改的地方,初始时刻为 yes
        per_source        = 11
        cps            = 100 2
        flags            = IPv4
    }
#使用如下命令进行服务的重新启动
[root@master ~]# /bin/systemctl restart xinetd.service
#查看 NRPE 是否已经启动
[root@master ~]# cd ./nrpe-2.15
[root@master nrpe-2.15]# netstat -tnlp
```

```
tcp6       0      0 :::5666                 :::*
  LISTEN      48306/xinetd
[root@master nrpe-2.15]#
```

图 1-10 监听端口

可以看到 5666 端口已经在监听了。关闭本地防火墙

```
[root@master nrpe-2.15]# systemctl stop firewalld.service
```

1.4.4.5. 步骤五：在监控端 node1 监测 node2

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。

```
[root@master    nrpe-2.15]#    /usr/local/nagios/libexec/check_nrpe    -H
```

```
192.168.90.146   #(node2 地址)监测 node3 则修改地址
```

会返回当前 NRPE 的版本

```
[root@master nrpe-2.15]# /usr/local/nagios/libexec/check_nrpe -H 1
92.168.90.119
NRPE v2.15
[root@master nrpe-2.15]#
```

图 1-11 返回版本号

若没返回，检查/usr/local/nagios/etc/nrpe.cfg 文件 allowed-hosts 网址，为 node1 网址

看到已经正确返回了 NRPE 的版本信息，说明一切正常。

1.4.4.6. 步骤六：配置 node1 各个文件

在 commands.cfg 中增加对 check_nrpe 的定义

```
[root@master nrpe-2.15]# vi /usr/local/nagios/etc/objects/commands.cfg
```

在最后面增加如下内容：

```
# '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.
        }
```

-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。定义对 node2 主机的监控(node3 一样的配置，修改 host_name 即可)

下面就可以在 services.cfg 中添加对 Nagios-Linux 主机的监控了。

```
[root@master nrpe-2.15]# vi /usr/local/nagios/etc/objects/services.cfg
define service{
        use                     local-service
        host_name               node2
        service_description     check-host-alive
        check_command           check-host-alive
        }

define service{
        use                     local-service
        host_name               node2
        service_description     Current Load
        check_command           check_nrpe!check_load
        }

define service{
    use                     local-service
        host_name               node2
        service_description     Check Disk sda1
```

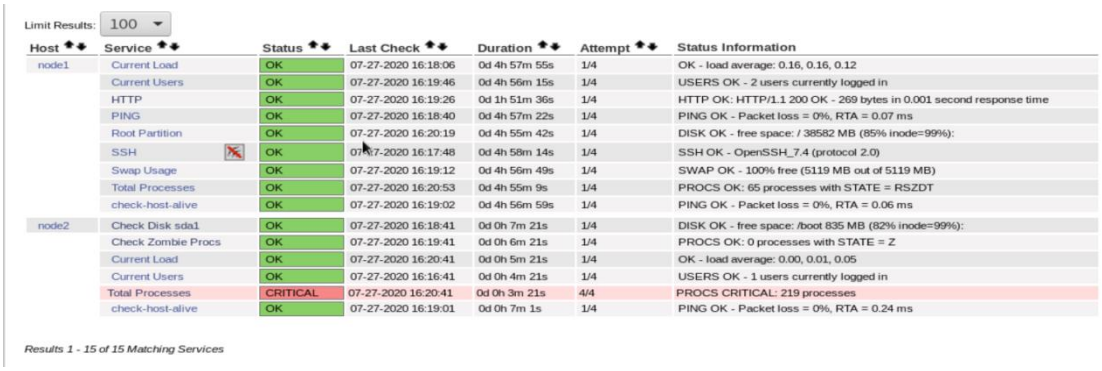
```
        check_command          check_nrpe!check_sda1
        }
define service{
        use                    local-service
        host_name              node2
        service_description     Total Processes
        check_command          check_nrpe!check_total_procs
        }

define service{
        use                    local-service
        host_name              node2
        service_description     Current Users
        check_command          check_nrpe!check_users
        }

define service{
        use                    local-service
        host_name              node2
        service_description     Check Zombie Procs
        check_command          check_nrpe!check_zombie_procs
        }
```

所有的配置文件已经修改好了，现在重启 Nagios。

注：在以后修改配置文件后，可以利用以下命令行检测配置文件是否正确，可以根据错误提示修改

```
[root@master       nrpe-2.15]#       /usr/local/nagios/bin/nagios       -v
/usr/local/nagios/etc/nagios.cfg
[root@master nrpe-2.15]# service nagios reload
[root@master nrpe-2.15]# service nagios restart
```

修改完配置文件后 web 界面不会立马修改，需要重新加载、启动 nagios 服务。打开 web 查看监控状态



Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
node1	Current Load	OK	07-27-2020 16:18:06	0d 4h 57m 55s	1/4	OK - load average: 0.16, 0.16, 0.12
	Current Users	OK	07-27-2020 16:19:46	0d 4h 56m 15s	1/4	USERS OK - 2 users currently logged in
	HTTP	OK	07-27-2020 16:19:26	0d 1h 51m 36s	1/4	HTTP OK: HTTP/1.1 200 OK - 269 bytes in 0.001 second response time
	PING	OK	07-27-2020 16:18:40	0d 4h 57m 22s	1/4	PING OK - Packet loss = 0%, RTA = 0.07 ms
	Root Partition	OK	07-27-2020 16:20:19	0d 4h 55m 42s	1/4	DISK OK - free space: / 38582 MB (85% inode=99%):
	SSH	OK	07-27-2020 16:17:48	0d 4h 58m 14s	1/4	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Swap Usage	OK	07-27-2020 16:19:12	0d 4h 56m 49s	1/4	SWAP OK - 100% free (5119 MB out of 5119 MB)
	Total Processes	OK	07-27-2020 16:20:53	0d 4h 55m 9s	1/4	PROCS OK: 65 processes with STATE = RSZDT
	check-host-alive	OK	07-27-2020 16:19:02	0d 4h 56m 59s	1/4	PING OK - Packet loss = 0%, RTA = 0.06 ms
node2	Check Disk sda1	OK	07-27-2020 16:18:41	0d 0h 7m 21s	1/4	DISK OK - free space: /boot 835 MB (82% inode=99%):
	Check Zombie Procs	OK	07-27-2020 16:19:41	0d 0h 6m 21s	1/4	PROCS OK: 0 processes with STATE = Z
	Current Load	OK	07-27-2020 16:20:41	0d 0h 5m 21s	1/4	OK - load average: 0.00, 0.01, 0.05
	Current Users	OK	07-27-2020 16:16:41	0d 0h 4m 21s	1/4	USERS OK - 1 users currently logged in
	Total Processes	CRITICAL	07-27-2020 16:20:41	0d 0h 3m 21s	4/4	PROCS CRITICAL: 219 processes
	check-host-alive	OK	07-27-2020 16:19:01	0d 0h 7m 1s	1/4	PING OK - Packet loss = 0%, RTA = 0.24 ms

Results 1 - 15 of 15 Matching Services

图 1-12 本地资源视图

可以看到 Total Processes 为 CRITICAL,在 status information 里提示进程为 219 个,为 CRITICAL 是因为我们在 nrpe.cfg 中设置

command[check_total_procs]=/usr/local/nagios/libexec/check_procs -w 150 -c 200

即进程数大于 150 为 WARNING,大于 200 为 CRITICAL,其阈值可根据实际情况设置

1.4.5. 实验任务五：Nagios 配置监控 HDFS 的健康状态

1.4.5.1. 步骤一：新建 check_hdfs.sh 文件并修改权限

```
 [root@master objects]# cd /usr/local/nagios/libexec/
 [root@master libexec]# vim check_hdfs.sh
#!/bin/sh

# set java environment
export JAVA_HOME=/usr/local/src/java
export JRE_HOME=/usr/local/src/java/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

# set hadoop environment
export HADOOP_HOME=/usr/local/src/hadoop
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH

chk_hdfs=`hdfs fsck /user | grep 'filesystem under path'`
case $chk_hdfs in
 *HEALTHY*)
     echo "OK - HDFS is healthy"
     exit 0
 ;;
 *)
     echo "CRITICAL - HDFS is corrupt!"
     exit 2

 ;;
esac

[root@master libexec]# chmod 755 ./check_hdfs.sh
[root@master libexec]# chown nagios:nagios ./check_hdfs.sh
```

1.4.5.2. 步骤二：编辑 services.cfg 文件

```
 [root@master libexec]# cd /usr/local/nagios/etc/objects
 [root@master objects]# vim services.cfg
#最后一行添加
define service{
       use                     generic-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               node1                    ;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_hdfs_health       ;对监控服务内容的描述，>
以供维护人员参考。
    contact_groups      admins
        check_command           check_nrpe_hdfs!check_hdfs      ;指定检查的命
令。
        }
```

1.4.5.3. 步骤三：编辑 commands.cfg 文件

```
    [root@master objects]# vim commands.cfg
    #最后一行添加
    define command{
        command_name    check_nrpe_hdfs
        command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$ -t 30
        }
```

1.4.5.4. 步骤四：编辑 nrpe.cfg 文件

```
    [root@master objects]# cd ..
    [root@master etc]# vim nrpe.cfg
    #添加
    command[check_hdfs]=/usr/local/nagios/libexec/check_hdfs.sh
```

所有的配置文件已经修改好了，现在重启 Nagios。

注：在以后修改配置文件后，可以利用以下命令行检测配置文件是否正确，可以根据错误提示修改

```
    [root@master        ~]#         /usr/local/nagios/bin/nagios        -v
/usr/local/nagios/etc/nagios.cfg
    [root@master ~]# service nagios reload
    [root@master ~]# service nagios restart
```

修改完配置文件后 web 界面不会立马修改，需要重新加载、启动 nagios 服务。打开 web 查看监控状态

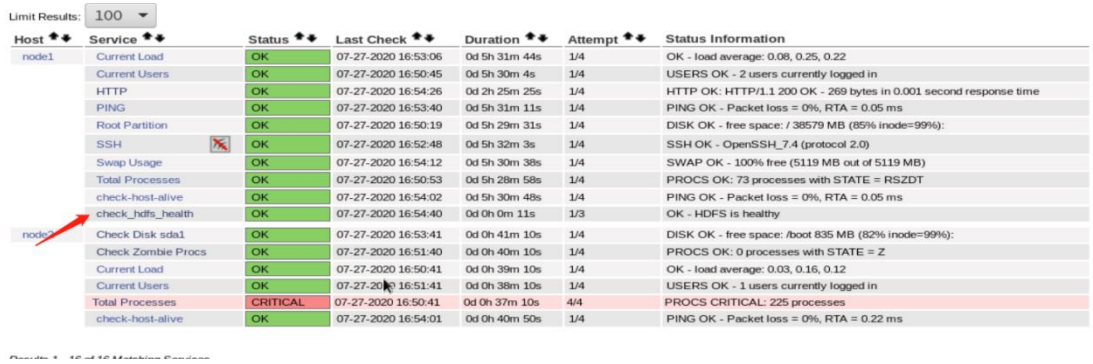


图 1-13 本地资源视图

可以看到 hdfs 状态正常

1.4.5.5. 步骤五：验证 hdfs 的健康状态

```
    [root@master etc]# cd ../libexec
```

```
[root@master libexec]# ./check_nrpe -H 192.168.90.233 -c check_hdfs
OK - HDFS is healthy
```

1.4.6. 实验任务六：Nagios 配置监控 datanode 的个数

1.4.6.1. 步骤一：新建 check_datanodes.sh 文件并修改权限

```
 [root@master libexec]# vim check_datanodes.sh
#!/bin/sh

# set java environment
export JAVA_HOME=/usr/local/src/java
export JRE_HOME=/usr/local/src/java/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

# set hadoop environment
export HADOOP_HOME=/usr/local/src/hadoop
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH

# set hive environment
export HIVE_HOME=/usr/local/src/hive
export PATH=$PATH:$HIVE_HOME/bin
#匹配 2 为正常,匹配非 2 数字为警告,否则报错
chk_hdfs=`su -s /bin/bash - hadoop -c 'hdfs dfsadmin -report' | grep 'Live
datanodes'`

case $chk_hdfs in

 *2*)

    echo $chk_hdfs

    exit 0

;;
*\d[^2]*)
    echo "warning:"$chk_hdfs
    exit 1
;;
*)

    echo "CRITICAL - Live datanodes is non-existent!"

    exit 2

;;

esac

[root@master libexec]# chmod 755 ./check_datanodes.sh
[root@master libexec]# chown nagios:nagios ./check_datanodes.sh
#linux 普通用户无密码切换
[root@master libexec]# vim /etc/pam.d/su
```

```
     1 #%PAM-1.0
     2 auth          sufficient     pam_rootok.so
     3 # Uncomment the following line to implicitly trust users in the "wheel"
group.
     4 #auth         sufficient     pam_wheel.so trust use_uid
     5 # Uncomment the following line to require a user to be in the "wheel"
group.
     6 #auth         required       pam_wheel.so use_uid
     7 auth          include        system-auth
     8 account       sufficient     pam_succeed_if.so uid = 0 use_uid quiet
     9 account       include        system-auth
    10 password      include        system-auth
    11 session       include        system-auth
    12 session       optional       pam_xauth.so
```

将第4行的#号去掉

将登陆用户加入 wheel 组,命令如下

```
[root@master libexec]# usermod -G wheel nagios
```

1.4.6.2. 步骤二：编辑 services.cfg 文件

```
[root@master libexec]# cd /usr/local/nagios/etc/objects
[root@master objects]# vim services.cfg
#最后一行添加
define service{
        use                     generic-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                node1              ;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_live_datanodes        ;对监控服务内容的描
述，以供维护人员参考。
        contact_groups          admins
        check_command           check_nrpe_datanodes!check_datanodes       ;
指定检查的命>令。
        }
```

1.4.6.3. 步骤三：编辑 commands.cfg 文件

```
[root@master objects]# vim commands.cfg
#最后一行添加
define command{
        command_name    check_nrpe_datanodes
        command_line      $USER1$/check_nrpe  -H  $HOSTADDRESS$  -u  -t  30  -c
$ARG1$
        }
```

1.4.6.4. 步骤四：编辑 nrpe.cfg 文件

```
[root@master objects]# cd ..
[root@master etc]# vim nrpe.cfg
#添加
command[check_datanodes]=/usr/local/nagios/libexec/check_datanodes.sh
```

所有的配置文件已经修改好了，现在重启 Nagios。

注：在以后修改配置文件后，可以利用以下命令行检测配置文件是否正确，可以根据

错误提示修改

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

修改完配置文件后 web 界面不会立马修改，需要重新加载、启动 nagios 服务。打开 web 查看监控状态

```
[root@master etc]# service nagios reload
[root@master etc]# service nagios restart
```

1.4.6.5. 步骤五：验证 datanode 存活个数

```
[root@master etc]# cd ../libexec
[root@master libexec]# ./check_nrpe -H 192.168.90.233 -c check_datanodes
Live datanodes (3):
```

验证完成可在 web 上查看状态

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
node1	Current Load	OK	07-27-2020 17:08:06	0d 5h 46m 44s	1/4	OK - load average: 0.28, 0.26, 0.22
	Current Users	OK	07-27-2020 17:05:45	0d 5h 45m 4s	1/4	USERS OK - 2 users currently logged in
	HTTP	OK	07-27-2020 17:09:26	0d 2h 40m 25s	1/4	HTTP OK: HTTP/1.1 200 OK - 269 bytes in 0.000 second response time
	PING	OK	07-27-2020 17:08:40	0d 5h 46m 11s	1/4	PING OK - Packet loss = 0%, RTA = 0.05 ms
	Root Partition	OK	07-27-2020 17:05:19	0d 5h 44m 31s	1/4	DISK OK - free space: / 38579 MB (85% inode=99%):
	SSH	OK	07-27-2020 17:07:48	0d 5h 47m 3s	1/4	SSH OK - OpenSSH_7.4 (protocol 2.0)
	Swap Usage	OK	07-27-2020 17:09:12	0d 5h 45m 38s	1/4	SWAP OK - 100% free (5119 MB out of 5119 MB)
	Total Processes	OK	07-27-2020 17:05:53	0d 5h 43m 58s	1/4	PROCS OK: 73 processes with STATE = RSZDT
	check-host-alive	OK	07-27-2020 17:09:02	0d 5h 45m 48s	1/4	PING OK - Packet loss = 0%, RTA = 0.05 ms
	check_hdfs_health	OK	07-27-2020 17:04:40	0d 0h 15m 11s	1/3	OK - HDFS is healthy
	check_live_datanodes	WARNING	07-27-2020 17:08:46	0d 0h 1m 5s	1/3	warning:Live datanodes (3):
node2	Check Disk sda1	OK	07-27-2020 17:08:41	0d 0h 56m 10s	1/4	DISK OK - free space: /boot 835 MB (82% inode=99%):
	Check Zombie Procs	OK	07-27-2020 17:06:40	0d 0h 55m 10s	1/4	PROCS OK: 0 processes with STATE = Z
	Current Load	OK	07-27-2020 17:05:41	0d 0h 54m 10s	1/4	OK - load average: 0.00, 0.02, 0.06
	Current Users	OK	07-27-2020 17:06:41	0d 0h 53m 10s	1/4	USERS OK - 1 users currently logged in
	Total Processes	CRITICAL	07-27-2020 17:05:41	0d 0h 52m 10s	4/4	PROCS CRITICAL: 224 processes
	check-host-alive	OK	07-27-2020 17:09:01	0d 0h 55m 50s	1/4	PING OK - Packet loss = 0%, RTA = 0.22 ms

图 1-14 本地资源视图

HA 分布式有 3 个 datanode 节点,而在 check_datanodes.sh 中设置匹配 2 个节点为 OK,非 2 个为 WARNING，匹配不到为 CRITICAL. 具体匹配数可以根据实际修改脚本.

扩展:可以根据实际情况自定义监控服务,官网 https://www.nagios.org/ 查询更多实用监控,例如监控 app 应用,windows 系统,实现 Nagios 报警功能等

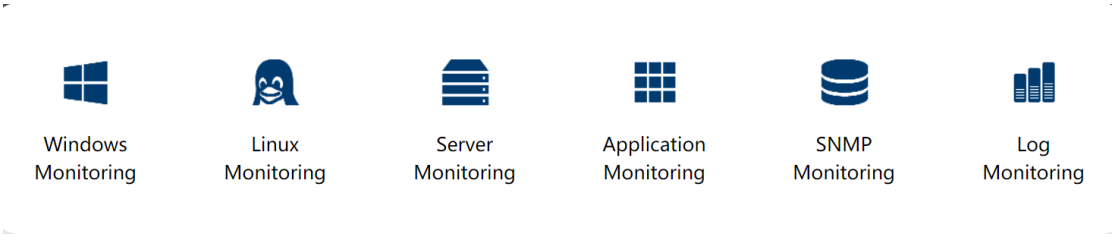


图 1-15 nagios 主要功能图