Nagios、Cacti、Ganglia、Zenoss、Clumon

# Nagios

## 简介

Nagios是一个可运行在Linux/Unix平台之上的开源监视系统，可以用来监视系统运行状态和网络信息。Nagios可以监视所指定的本地或远程主机以及服务，同时提供异常通知功能。

Nagios提供以下几种监控功能：

* 监控网络服务（SMTP、POP3、HTTP、NNTP、Ping等）。
* 监控主机资源（处理器负荷、磁盘利用率等）。
* 简单的插件设计使得用户可以方便地扩展自己服务的检测方法。
* 并行服务检查机制。
* 具备定义网络分层结构的能力，并使用“parent”主机定义来表达网络主机间的关系，这种关系可被用来发现和明晰主机宕机或不可达状态。
* 当服务或主机问题产生与解决时将告警发送给联系人（通过电子邮件、短信、用户定义方式）。
* 具备定义事件处理功能，可以在主机或服务的事件发生时获取更多问题定位。
* 自动的日志回滚。
* 可以支持并实现对主机的冗余监控。
* 可选的Web界面用于查看当前的网络状态、通知和故障历史、日志文件等。

Ganglia没有内置通知系统，而 Nagios 在这方面表现杰出。

Nagios没有适合目标主机的可伸缩内置代理（这是一个争议点），而 Ganglia 在设计之初就考虑到这点。

## 安装Nagios

1. **监控本地机器**

[root@node0 ~]# yum install -y httpd php gcc glibc glibc-common gd gd-devel

[root@node0 ~]# useradd nagios

[root@node0 ~]# passwd nagios

[root@node0 ~]# usermod -a -G nagcmd nagios

[root@node0 ~]# # usermod -a -G nagcmd apache

[root@node0 ~]# # wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.7.tar.gz

[root@node0 ~]# tar zxvf nagios-4.0.7.tar.gz

[root@node0 ~]# cd nagios-4.0.7

//此处修改默认安装目录会出错

[root@node0 nagios] # ./configure --with-command-group=nagcmd

[root@node0 nagios]# make all

[root@node0 nagios]# make install

[root@node0 nagios]# make install-init

[root@node0 nagios]# make install-config

[root@node0 nagios]# make install-commandmode

配置警告接受人email

[root@node0 nagios]# vi /usr/local/nagios/etc/objects/contacts.cfg

define contact{

contact\_name nagiosadmin ; Short name of user

use generic-contact ; Inherit default values from generic-contact template (defined above)

alias Nagios Admin ; Full name of user

email 331274799@qq.com ; <<\*\*\*\*\* CHANGE THIS TO YOUR EMAIL ADDRESS \*\*\*\*\*\*

}

[root@node0 nagios]# make install-webconf

创建登陆网页的用户名密码

[root@node0 nagios]# htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

[root@node0 nagios-plugins-2.0.2]# chkconfig --add httpd

[root@node0 nagios-plugins-2.0.2]# chkconfig httpd on

[root@node0 nagios]# service httpd restart

[root@node0 ~]# # wget http://nagios-plugins.org/download/nagios-plugins-2.0.2.tar.gz

[root@node0 ~]# tar zxvf nagios-plugins-2.0.2.tar.gz

[root@node0 nagios-plugins-2.0.2]# ./configure --with-nagios-user=nagios --with-nagios-group=nagios

[root@node0 nagios-plugins-2.0.2]# make

[root@node0 nagios-plugins-2.0.2]# make install

[root@node0 nagios-plugins-2.0.2]# chkconfig --add nagios

[root@node0 nagios-plugins-2.0.2]# chkconfig nagios on

[root@node0 nagios-plugins-2.0.2]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

[root@node0 nagios-plugins-2.0.2]# service iptables stop

[root@node0 nagios-plugins-2.0.2]# chkconfig iptables off

[root@node0 nagios-plugins-2.0.2]# vi /etc/selinux/config

SELINUX=disabled

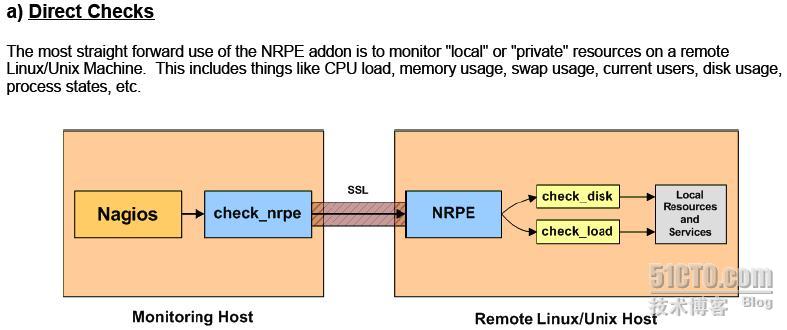
[root@node0 nagios-plugins-2.0.2]# service nagios start

<http://localhost/nagios>

输入nagiosadmin和密码登录

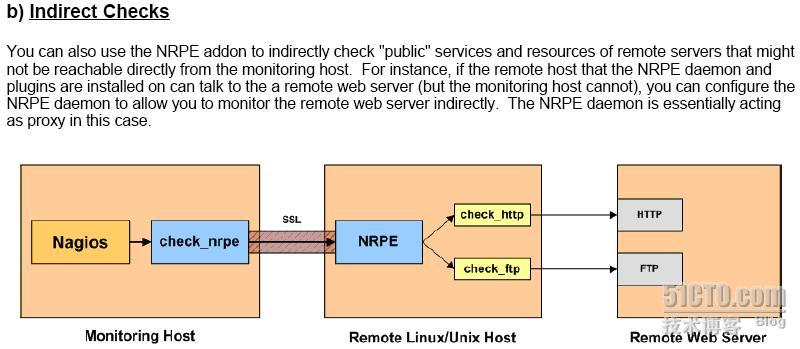
## 安装NRPE

1. **监控远程机器**

****

NRPE运行在远程主机的后台，处理Nagios主机上check\_nrpe发送的请求。当接收到请求时，执行相应命令，再将结果返回给check\_nrpe。

Check\_nrpe运行在nagios主机，向NRPE发送命令请求，等待NRPE执行相应命令并返回结果。Check\_nrpe将远程命令的输出和返回值作为其输出和返回值。



1. **被监控主机**

安装nagios-plugins

[root@node1 ~]# useradd nagios //新增用户

[root@node1 ~]# passwd nagios

[root@node1 ~]# wget http://nagios-plugins.org/download/nagios-plugins-2.0.2.tar.gz

[root@node1 ~]# cd nagios-plugins-2.0.2

[root@node1 nagios-plugins-2.0.2]# ./configure

[root@node1 nagios-plugins-2.0.2]# make

[root@node1 nagios-plugins-2.0.2]# make install

[root@node1 nagios-plugins-2.0.2]# chown -R nagios.nagios /usr/local/nagios/

[root@node1 nagios-plugins-2.0.2]#

安装NRPE

[root@node1 ~]# yum install -y xinetd mod\_ssl openssl openssl-devel

[root@node1 ~]# wget http://sourceforge.net/projects/nagios/files/nrpe-2.x/nrpe-2.15/nrpe-2.15.tar.gz/download

[root@node1 ~]# tar zxvf nrpe-2.15.tar.gz

[root@node1 ~]# cd nrpe-2.15

[root@node1 nrpe-2.15]# ./configure

[root@node1 nrpe-2.15]# make all

[root@node1 nrpe-2.15]# make install-plugin

[root@node1 nrpe-2.15]# make install-daemon

[root@node1 nrpe-2.15]# make install-daemon-config

[root@node1 nrpe-2.15]# make install-xinetd

[root@node1 nrpe-2.15]# vi /etc/xinetd.d/nrpe

only\_from = 127.0.0.1 192.168.0.62在后面增加监控主机的地址，以空格间隔

[root@node1 nrpe-2.15]# vi /etc/services

# Local services

nrpe 5666/tcp # NRPE

//作为xinetd启动

[root@node1 nrpe-2.15]# service xinetd restart

[root@node1 nrpe-2.15]# netstat -utlp | grep nrpe

[root@node1 nrpe-2.15]# netstat -an | grep 5666

//作为stand alone启动

[root@node1 nrpe-2.15]# /usr/local/nagios/bin/nrpe -c /usr/local/nagios/etc/nrpe.cfg -d

[root@node1 nrpe-2.15]# ps aux | grep nrpe

[root@node1 nrpe-2.15]# /usr/local/nagios/libexec/check\_nrpe -H localhost查看本地NRPE版本

check\_nrpe -H 被监控的主机 -c要执行的监控命令

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

定义可被执行的命令

[root@node1 nrpe-2.15]# vi /usr/local/nagios/etc/nrpe.cfg

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

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

1. **监控主机**

安装check\_nrpe插件，只用安装check\_nrpe插件

[root@node0 ~]# yum install openssl-devel

[root@node0 ~]# wget <http://sourceforge.net/projects/nagios/files/nrpe-2.x/nrpe-2.15/nrpe-2.15.tar.gz/download>

[root@node0 ~]# tar zxvf nrpe-2.15.tar.gz

[root@node0 ~]# cd nrpe-2.15

[root@node0 nrpe-2.15]# ./configure

[root@node0 nrpe-2.15]# make all

[root@node0 nrpe-2.15]# make install-plugin

[root@node0 nrpe-2.15]# /usr/local/nagios/libexec/check\_nrpe -H 192.168.0.63 //测试子节点版本

NRPE v2.13

/usr/local/nagios/libexec/check\_nrpe -H node1 -c check\_users

/usr/local/nagios/libexec/check\_nrpe -H node1 -c check\_ping -a node0 500,100% 700,100%

在nagios中增加命令

[root@node0 nrpe-2.15]# vi /usr/local/nagios/etc/objects/commands.cfg

define command{

command\_name check\_nrpe

command\_line $USER1$/check\_nrpe -H $HOSTADDRESS$ -c $ARG1$

}

新增一个配置文件

[root@node0 nrpe-2.15]# vi /usr/local/nagios/etc/nagios.cfg

# Definitions for monitoring the remote (Linux) host

cfg\_file=/usr/local/nagios/etc/objects/host.cfg

[root@node0 nrpe-2.15]# vi /usr/local/nagios/etc/objects/host.cfg

define host{

use linux-server

host\_name node1

alias node1

address 192.168.214.145

}

define service{

use generic-service

host\_name node1

service\_description CPU Load

check\_command check\_nrpe!check\_load

}

define service{

use generic-service

host\_name node1

service\_description Current Users

check\_command check\_nrpe!check\_users

}

define service{

use generic-service

host\_name node1

service\_description /dev/hda1 Free Space

check\_command check\_nrpe!check\_hda1

}

define service{

use generic-service

host\_name node1

service\_description Total Processes

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

}

define service{

use generic-service

host\_name node1

service\_description Zombie Processes

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

}

linux-box继承自generic-host模板，其定义在localhost.cfg中。

检查配置文件

[root@node0 nrpe-2.15]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

[root@node0 nrpe-2.15]# service nagios restart

访问nagios网页可看到该远程主机的信息

## 新增远程监控

新增一项远程监控服务步骤：

1. 在远程主机的nrpe.cfg中新增命令定义；
2. 在监控主机的配置文件中新增服务定义。

以check\_swap为例：可执行文件已在libexec中

[root@compute ~]# /usr/local/nagios/libexec/check\_swap -w 20% -c 10%

[root@node1 ~]# vi /usr/local/nagios/etc/nrpe.cfg

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

standalone需要重启NRPE，xinetd不需要

[root@node0 ~]# vi /usr/local/nagios/etc/objects/host.cfg

define service{

use generic-service

host\_name node1

service\_description Swap Usage

check\_command check\_nrpe!check\_swap

}

[root@node0 ~]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

[root@node0 ~]# service nagios restart

## 问题

**问题一：Could not complete SSL handshake**

原因有2个，一个是没有设好访问权限，一个是nrpe无法正常启动

解决方法：

[root@node1 nrpe-2.15]# vi /etc/xinetd.d/nrpe

only\_from = 127.0.0.1 192.168.0.62在后面增加监控主机的地址，以空格间隔

[root@node1 nrpe-2.15]# vi /usr/local/nagios/etc/nrpe.cfg

allowed\_hosts=127.0.0.1,192.168.0.62 以逗号分隔

xinetd方式启动的nrpe会失败，用stand alone方式启动-d，ps aux | grep nrpe查看是否成功

**问题二：Connection refused by host**

解决方法：Nrpe进程没有启动

**问题三：NRPE: Unable to read output**

解决方法：/usr/local/nagios/etc/nrpe.cfg命令配置错误，路径或多余符号

**问题四：check\_nrpe通过-a 调用出错**

/usr/local/nagios/libexec/check\_nrpe -H node1 -c check\_ping -a node0 500,100% 700,100%

./check\_nrpe -H node1 -c check\_by\_ssh -a node0 hostname

解决方法：

[root@node1 nrpe-2.15]# vi /usr/local/nagios/etc/nrpe.cfg

dont\_blame\_nrpe=1 //该处设置为1

[root@node1 nrpe-2.15]# ./configure --enable-command-args //配置时增加该参数

**问题五：CHECK\_NRPE: Received 0 bytes from daemon. Check the remote server logs for error messages.**

登陆到node1，查看/var/log/messages，可知参数出错

原因是安装NRPE时，没有支持参数解析功能，即不能通过-a设置命令参数，且在/usr/local/nagios/etc/nrpe.cfg中没有配置check\_ping

/usr/local/nagios/libexec/check\_nrpe -H node1 -c check\_ping -a node0 500,100% 700,100% 出错

**问题六：在nrpe-2.15中的bug，check\_nrpe –H localhost失败，check\_nrpe -H 127.0.0.1成功**

vi /etc/xinetd.d/nrpe

only\_from       = 127.0.0.1 localhost

**问题七：HTTP WARNING: HTTP/1.1 403 Forbidden**

nagios监控/var/www/html/下面的index.html文件，若没有就会提示错误，touch一个。

文章推荐：

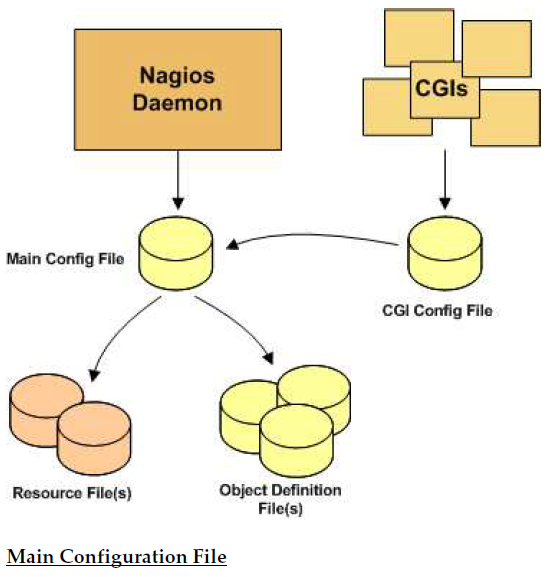
<http://www.ibm.com/developerworks/cn/linux/l-ganglia-nagios-1/index.html>

<http://www.programmer.com.cn/11477/>

<http://yahoon.blog.51cto.com/13184/41893>

<http://sery.blog.51cto.com/10037/20520>

## 用户手册



配置文件

[root@node0 etc]# ls

cgi.cfg htpasswd.users nagios.cfg objects resource.cfg

Main Configuration File：包含控制Nagios Daemon行为的指令集，被Nagios Daemon和CGI调用。

Resource Files：保存用户自定义宏。为了保存敏感信息（如密码），不被CGI调用。通过Main Configuration File中的resource\_file指明一个或多个resource files。

Object Definition Files：定义监视对象和如何监视，包括主机，服务，主机群，联系，联系群，命令等。通过Main Configuration File中的cfg\_file指明一个或多个Object Definition Files。

CGI Configuration File：包含控制CGI行为的指令集。

objects定义的对象：

|  |  |  |
| --- | --- | --- |
| 联系人 | contact | 出了问题像谁报告?一般当然是系统管理员了 |
| 监控时间段 | timeperiod | 7X24小时不间断还是周一至周五,或是自定义的其他时间段 |
| 被监控主机 | host | 所需要监控的服务器,当然可以是监控机自己 |
| 监控命令 | command | nagios发出的哪个指令来执行某个监控,这也是自己定义的 |
| 被监控的服务 | service | 例如主机是否存活,80端口是否开,磁盘使用情况或者自定义的服务等 |

define host{

host\_name bogus-router //主机名称

alias Bogus Router #1 //主机别名

address 192.168.1.254 //IP地址

display\_name router //web中显示的名称，默认为host\_name

parents server-backbone //父主机名称，必须已存在

check\_command check-host-alive //当主机启动时检查的命令，必须已在commands.cfg中定义，为空则不检查

check\_interval 5 //检查主机的单位时间的次数，如果修改interval\_length（默认为60），则该值的单位时间为分

retry\_interval 1 //等待重新检查主机的单位时间的次数

max\_check\_attempts 5 //check\_command重试次数

check\_period 24x7 //检查时间段，该值必须在timeperiods.cfg中已定义

process\_perf\_data 0

retain\_nonstatus\_information 0

contact\_groups router-admins

notification\_interval 30

notification\_period 24x7

notification\_options d,u,r

}

# Ganglia

## 简介

Ganglia是UC Berkeley发起的一个开源集群监视项目，设计用于测量数以千计的节点。Ganglia的核心包含gmond、gmetad以及一个Web前端。主要是用来监控系统性能，如：cpu 、mem、硬盘利用率、 I/O负载、网络流量情况等，通过曲线很容易见到每个节点的工作状态，对合理调整、分配系统资源，提高系统整体性能起到重要作用。

## 安装

\* APR (http://apr.apache.org/)

\* libConfuse (http://www.nongnu.org/confuse/)

\* expat (http://expat.sourceforge.net/)

\* pkg-config (http://www.freedesktop.org/wiki/Software/pkg-config)

\* python (http://www.python.org/)

\* PCRE (http://www.pcre.org/)

\* RRDtool (http://oss.oetiker.ch/rrdtool/)

[root@node0 ~]# yum install httpd php

[root@node0 ~]# yum -y install apr-devel apr-util check-devel cairo-devel pango-devel libxml2-devel rpmbuild glib2-devel dbus-devel freetype-devel fontconfig-devel gcc-c++ expat-devel python-devel libXrender-devel

//安装rrdtool

[root@node0 software]# tar zxvf rrdtool-1.4.7.tar.gz

[root@node0 software]# cd rrdtool-1.4.7

[root@node0 rrdtool-1.4.7]# ./configure –prefix=/usr

[root@node0 rrdtool-1.4.7]# make

[root@node0 rrdtool-1.4.7]# make install

//安装libConfuse

[root@node0 software]# tar zxvf confuse-2.7.tar.gz

[root@node0 software]# cd confuse-2.7

[root@node0 confuse-2.7]# ./configure --prefix=/home/dwt/software/confuse

[root@node0 confuse-2.7]# make

[root@node0 confuse-2.7]# make install

//安装ganglia

[root@node0 software]# tar zxvf ganglia-3.4.0.tar.gz

[root@node0 software]# cd ganglia-3.4.0

[root@node0 ganglia-3.4.0]# ./configure --prefix=/home/dwt/software/ganglia --with-gmetad --with-libconfuse=/home/dwt/software/confuse

[root@node0 ganglia-3.4.0]# make

[root@node0 ganglia-3.4.0]# make install

//配置脚本

[root@node0 ganglia-3.4.0]# cp gmetad/gmetad.init / etc/rc.d/init.d/gmetad

[root@node0 ganglia-3.4.0]# cp gmond/gmond.init /etc/rc.d/init.d/gmond

[root@node0 ganglia-3.4.0]# vi /etc/init.d/gmetad

#GMETAD=/usr/sbin/gmetad

GMETAD=/home/dwt/software/ganglia/sbin/gmetad

[root@node0 ganglia-3.4.0]# vi /etc/init.d/gmond

#GMOND=/usr/sbin/gmond

GMOND=/home/dwt/software/ganglia/sbin/gmond

//设置输出目录

[root@node0 ganglia-3.4.0]# mkdir -p /var/lib/ganglia/rrds

[root@node0 ganglia-3.4.0]# chown nobody:nobody /var/lib/ganglia

[root@node0 ganglia-3.4.0]# chmod -R 777 /var/lib/ganglia/dwoo/

//设置配置文件

[root@node0 ganglia-3.4.0]# vi /home/dwt/software/ganglia/etc/gmetad.conf

data\_source "my cluster" node0 node1

[root@node0 ganglia-3.4.0]# gmond –default\_config > /home/dwt/software/ganglia/etc/gmond.conf

globals {

daemonize = yes

setuid = yes

user = nobody

}

cluster {

name = "my cluster"

owner = "nobody"

latlong = "unspecified"

url = "unspecified"

}

//安装web

[root@node0 software]# tar zxvf ganglia-web-3.5.4.tar.gz

[root@node0 software]# cd ganglia-web-3.5.4

[root@node0 ganglia-web-3.5.4]# cp conf\_default.php conf.php

[root@node0 ganglia-web-3.5.4]# make install

[root@node0 ganglia-web-3.5.4]# vi /var/www/html/ganglia/conf.php

$conf['gmetad\_root'] = "/var/lib/ganglia";

$conf['rrds'] = "${conf['gmetad\_root']}/rrds";

$conf['rrdtool'] = "/usr/bin/rrdtool";

//启动

[root@node0 ganglia-3.4.0]# chkconfig --add gmond

[root@node0 ganglia-3.4.0]# chkconfig gmond on

[root@node0 ganglia-3.4.0]# chkconfig --add gmetad

[root@node0 ganglia-3.4.0]# chkconfig gmetad on

[root@node0 ganglia-3.4.0]# service gmetad start

[root@node0 ganglia-3.4.0]# service gmond start

客户端安装

[root@node1 ~]# service gmond start

**问题一：**DWOO compiled templates directory is not writeable. Please adjust $conf['dwoo\_compiled\_dir']

**解决方法：**[root@node0 ganglia-3.4.0]# chmod -R 777 /var/lib/ganglia/dwoo/

**问题二：无法显示图表，都是X**

**解决方法：rrdtool安装的前缀设为/usr，让头文件和库文件都自动放在/usr/include和/usr/lib下面，ganglia安装时能自动找到这些地方**

# JNRPE

项目分为几个模块：

Jnrpe-server是一个独立的应用，调用jnrpe-lib打成的jnrpe-lib-xxx-beta-11.jar包中的JNRPE类，

|  |  |
| --- | --- |
| **Name** | **Description** |
| [jnrpe-server](http://jnrpe.sourceforge.net/jnrpe-server/index.html) | The JNRPE server standalone application |
| [jnrpe-lib](http://jnrpe.sourceforge.net/jnrpe-lib/index.html) | A library that implements the NRPE protocol for JAVA applications |
| [jnrpe-plugins](http://jnrpe.sourceforge.net/jnrpe-plugins/index.html) | The JNRPE bundled plugins |
| [JNRPE Installation](http://jnrpe.sourceforge.net/jnrpe-install/index.html) | The JNRPE Server installation package |
| [jcheck\_nrpe](http://jnrpe.sourceforge.net/jcheck_nrpe/index.html) | jcheck\_nrpe is a java implementation of the check\_nrpe Nagios plugin. It can be easily embedded inside your code to talk to JNRPE or NRPE. Both plain socket and SSL are supported. |

## JNRPE Server

JNRPE最初作为NRPE Nagios Addon的java实现，2.0版本后，分为几个工程，增加更多特性和模块。JNRPE能完全集成到你的应用中，调用check\_nrpe，返回Nagios的检测值。

### 下载运行

下载jnrpe-server-2.0.3-beta-01-bin.zip，jnrpe-server-2.0.3-beta-01-src.zip文件

解压jnrpe-server-2.0.3-beta-01-bin.zip，有以下文件夹：

jnrpe-server-xxx

|-bin

|-etc

|-lib

|-docs

|-logs

|-plugins

jnrpe：unix脚本，jnrpe.bat：windows脚本。Etc目录包含jnrpe配置文件：jnrpe.ini和log4j。lib目录包含所有JNRPE Server运行所需的jar包。Log目录包含JNRPE Server日志。Plugin目录放置希望安装的JNRPE插件。

新建工程jnrpe-server，将jnrpe-server-2.0.3-beta-01-src.zip解压后的it目录拷入src目录

将jnrpe-server-2.0.3-beta-01目录下的etc,lib,docs,logs,plugins目录拷入工程。

刷新工程，将lib下的包都导入工程中，Run Configurations的Arguments页面的Program arguments中增加-c etc/inrpe.ini，打开JNRPEServer.java运行。

**问题1：log4j:WARN No appenders could be found for logger**

方法1：将etc/log4j.properties放到src目录下，log4j默认搜索路径；

方法2：在main函数第一行增加，即可设为其他路径

PropertyConfigurator.configure(System.getProperty("user.dir") + "/etc/log4j.properties");

### 命令参数

JNRPE Server命令参数：

* -c,--conf file : tells to JNRPE where is the JNRPE configuration file. This option is REQUIRED.
* -l,--list : lists all the plugin installed inside the JNRPE server
* -v,--version : shows the server version

举例：

cd $JNRPE\_HOME/bin

./jnrpe -c ../etc/jnrpe.ini

### jnrpe.ini

Jnrpe.ini文件分为两部分：

[server] //Server配置：绑定地址/端口号，客户端ip，插件安装目录。

accept-params : true //true：JNRPE扩展$ARG?$宏，false：忽略-a参数，即传给check\_nrpe的参数

bind-address : 127.0.0.1:5666 //JNRPE监听127.0.0.1 5666端口，可重复这部分使JNRPE监听多个地址和端口，如果希望加密，则在最后添加ssl

plugin-path : /usr/local/jnrpe/plugins //JNRPE在/usr/local/jnrpe/plugins下找安装的插件

allow-address : 127.0.0.1 //JNRPE接收来自127.0.0.1的命令，重复可使JNRPE接收多个地址

[commands] //命令配置格式：

[COMMAND\_NAME] : [PLUGIN\_NAME] [COMMAND\_LINE]

check\_disk\_C : CHECK\_DISK --path C: --warning $ARG1$ --critical $ARG2$

check\_disk\_E : CHECK\_DISK --path E: --warning $ARG1$ --critical $ARG2$

## jnrpe-lib

JNRPE Library是JNRPE的核心，负责接收check\_nrpe查询。

### 下载运行

下载解压jnrpe-lib-1.0.1-beta-11-src.zip，新建工程jnrpe-lib，将src/main/java下的it目录拷贝到工程src目录中，将jnrpe-server lib目录下的commons-cli-1.2.jar，commons-lang-2.6.jar拷贝到工程lib目录中加入工程依赖库。

## jcheck\_nrpe

check\_nrpe的java实现，能够嵌入到JNRPE中，支持SSL。

命令参数：

* -n/--nossl instruct jcheck\_nrpe to not use SSL for communication.
* -u/--unknown instructs jcheck\_nrpe to return an unknown status (instead of critical) on connection timeout errors
* -H/--host tells jcheck\_nrpe where to find the JNRPE/NRPE server
* -p/--port tells jcheck\_nrpe which port JNRPE/NRPE is listening to. Defaults to 5666
* -t/--timeout configure the connection timeout in seconds
* -c/--command configure the name of the command to be invoked
* -a/--args The list of arguments to be passed to the JNRPE/NRPE command. All the parameters must be joined using as separator the exclamation mark ('!')
* -h/--help shows the jcheck\_nrpe command line help

举例：

./jcheck\_nrpe.sh -H 127.0.0.1 -t 10 -c check\_test -a 'HELLO WORLD!critical'

# NDOUtils

## 概述

NDOUtils用于将监控数据导入MySQL数据库中。

NDOMOD：运行时被nagios加载，用于将所有数据导入TCP socket或标准文件或Unix socket。

NDO2DB：从TCP或Unix socket中读入NDOMOD的数据，解析数据并导入MySQL数据库，可同时处理多个客户端连接。

FILE2SOCK：从标准文件中读取数据并导入TCP或Unix Socket中。

LOG2NDO：导入历史日志到NDO2DB中。

## 安装

[root@node0 ~]# yum install -y mysql-server mysql mysql-devel

[root@node0 ~]# service mysqld start

[root@node0 ~]# mysql -u root

mysql> create database nagios;

mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON nagios.\* TO nagios@'localhost' IDENTIFIED BY '123456';

mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON nagios.\* TO nagios@'controller' IDENTIFIED BY '123456';

mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON nagios.\* TO nagios@'%' IDENTIFIED BY '123456';

mysql> flush privileges;

[root@node0 ~]# wget <http://sourceforge.net/projects/nagios/files/ndoutils-2.x/ndoutils-2.0.0/ndoutils-2.0.0.tar.gz/download>

[root@node0 ~]# tar zxvf ndoutils-2.0.0.tar.gz

[root@node0 ~]# cd ndoutils-2.0.0

[root@node0 ndoutils-2.0.0]# ./configure --with-mysql-lib=/usr/lib64/mysql

# ./configure --enable-mysql --disable-pgsql --with-mysql-lib=/usr/lib64/mysql/ --with-mysql-inc=/usr/include/mysql/ --with-ndo2db-user=nagios --with-ndo2db-group=nagcmd

[root@node0 ndoutils-2.0.0]# make

[root@node0 ndoutils-2.0.0]# cd db

[root@node0 ndoutils-2.0.0]# /usr/bin/mysql -u root -p -D nagios -h localhost < ./mysql.sql

[root@node0 ndoutils-2.0.0]# cp src/ndomod-4x.o /usr/local/nagios/bin/ndomod.o

[root@node0 ndoutils-2.0.0]# cp config/ndomod.cfg-sample /usr/local/nagios/etc/ndomod.cfg

[root@node0 ndoutils-2.0.0]# vi /usr/local/nagios/etc/nagios.cfg

在broker\_module下增加如下一行，确保nagios下次启动时能正确加载NDOMOD

broker\_module=/usr/local/nagios/bin/ndomod.o config\_file=/usr/local/nagios/etc/ndomod.cfg

确认让nagios向NDOMOD发送数据

event\_broker\_options=-1

[root@node0 ndoutils-2.0.0]# cp src/ndo2db-4x /usr/local/nagios/bin/ndo2db

[root@node0 ndoutils-2.0.0]# cp config/ndo2db.cfg-sample /usr/local/nagios/etc/ndo2db.cfg

[root@node0 ndoutils-2.0.0]# vi /usr/local/nagios/etc/ndo2db.cfg

db\_user=nagios 修改为MySQL的用户名密码

db\_pass=123456

[root@node0 ndoutils-2.0.0]# /usr/local/nagios/bin/ndo2db -c /usr/local/nagios/etc/ndo2db.cfg

[root@node0 ndoutils-2.0.0]# service nagios restart

[root@node0 ndoutils-2.0.0]# tail -30 /var/log/messages

[root@node0 ndoutils-2.0.0]# tail -20 /usr/local/nagios/var/nagios.log

将NDOMOD设为开机启动

[root@node0 ndoutils-2.0.0]# cp daemon-init /etc/xinetd.d/ndo2db

[root@node0 ndoutils-2.0.0]# chmod 775 /etc/xinetd.d/ndo2db

[root@node0 ndoutils-2.0.0]# chkconfig --add ndo2db

[root@node0 ndoutils-2.0.0]# chkconfig ndo2db on

## 问题

**问题：Support for the specified database server is either not yet supported, or was not found on your system.**

找不到mysql的lib库，可能的原因有：1、没有安装mysql-devel；2、安装的是64位的mysql，其目录在/usr/lib64/mysql，而不是默认的/usr/lib/mysql，需要用--with-mysql-lib指定

**问题：ndoutils启动报:Could not bind socket: Address already in use**

rm -f /usr/local/nagios/var/ndo.sock

**多个实例必需修改成不同的实例名**

在/usr/local/nagios/etc/ndomod.cfg中实例名默认为instance\_name=default，如果有多个实例必需修改成不同的实例名，如：instance\_name=nagios1

**ndomod: Could not open data sink!**

1. 所有的目录权限都设为nagios，chown -R nagios.nagios nagios
2. 检查数据库配置是否正确

ndo2db\_user=nagios

ndo2db\_group=nagcmd (注意组名)

db\_host=localhost

db\_name=nagios

db\_user=nagios

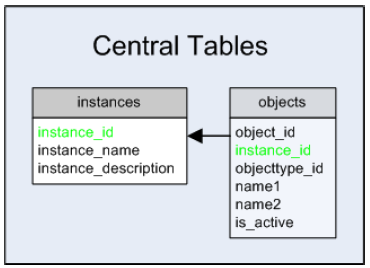
db\_pass=123456

1. 采用配置登陆mysql

mysql -u nagios -p -h localhost -D nagios

## 表结构

### Central Tables



一般根据instance\_id，object\_id来标识唯一实体，如host，timeperiod，contact等。

#### instances

多个Nagios实例，每个实例代表一个Nagios进程，实例通过名称区分，不可重复。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| instance\_id | SMALLINT | 主键 |
| instance\_name | VARCHAR(64) | 实例名称，通过NDOUtils传入 |
| instance\_description | VARCHAR(128) | 描述实例 |

#### object

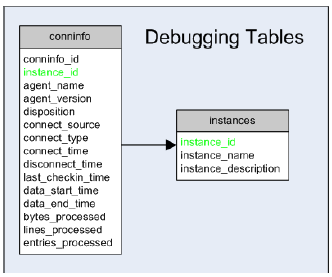
保存所有etc/objects目录下的配置信息，假如删除某个配置，它将不会在数据库中，但想对老数据进行解析时，可以在这找到其含义。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| object\_id | INT | 主键 |
| instance\_id | SMALLINT | instance.instance\_id |
| objecttype\_id | SMALLINT | object类型  1 = Host  2 = Service  3 = Host group  4 = Service group  5 = Host escalation  6 = Service escalation  7 = Host dependency  8 = Service dependency  9 = Timeperiod  10 = Contact  11 = Contact group  12 = Command  13 = Extended host info (deprecated)  14 = Extended service info  (deprecated) |
| name1 | VARCHAR(128) | object的名称 |
| name2 | VARCHAR(128) | object的描述信息 |
| is\_active | SMALLINT | 说明该object目前是否在配置文件中，如果该object已删除，则置为inactive  0=Inactive  1=Active |

### Debugging Tables

#### conninfo

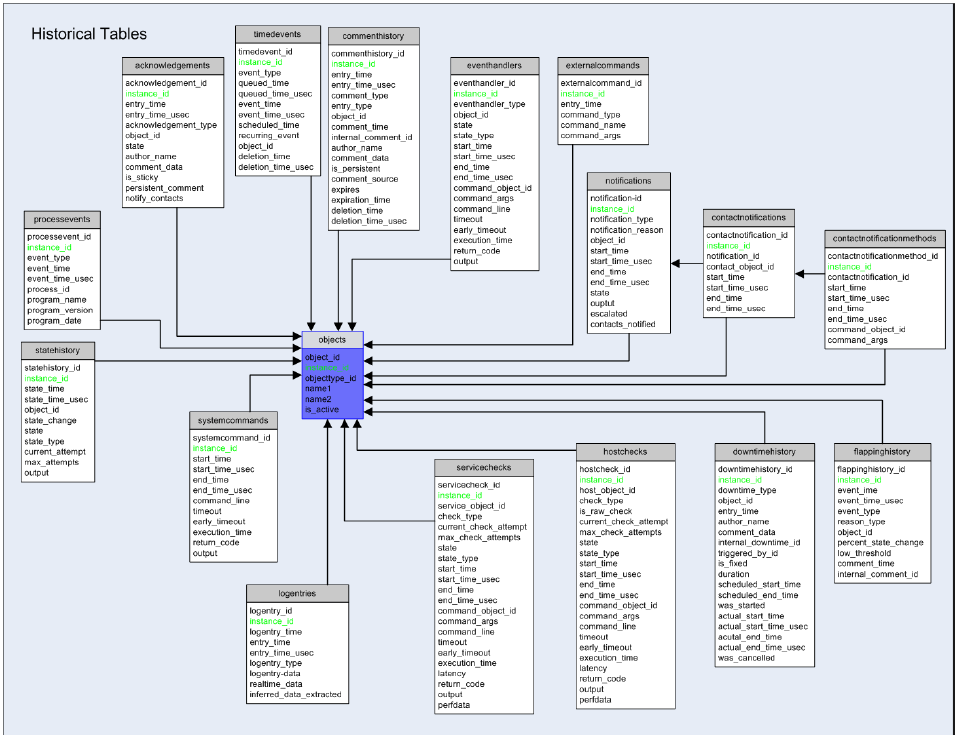
唯一一个用户保存调试信息的表，保存NDO2DB进程，user agents（如Log2DB，NDOMOD）的调试信息。



|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| conninfo\_id | INT | 主键 |
| instance\_id | SMALLINT | instance.instance\_id |
| agent\_name | VARCHAR(32) | 描述向NDO2DB发送数据的user agent，一般为NDOMOD或LOG2NDO |
| agent\_version | VARCHAR(8) | user agent的版本号 |
| disposition | VARCHAR(16) | 描述向NDO2DB发送的数据的配置或类型  REALTIME：直接从Nagios进程发送  ARCHIVED：从文件发送 |
| connect\_source | VARCHAR(16) | user agent连接NDO2DB的方式  TOPSOCKET或UNIXSOCKET |
| connect\_type | VARCHAR(16) | 描述user agent与NDO2DB的连接是新连接还是失败后的重新连接  INITIAL或RECONNECT |
| connect\_time | DATETIME | user agent连接到NDO2DB的初始时间 |
| disconnect\_time | DATETIME | user agent 停止连接到NDO2DB的时间 |
| last\_checkin\_time | DATETIME | user agent在NDO2DB的最后注册时间，表明其还存活 |
| data\_start\_time | DATETIME | user agent向NDO2DB第一次发送数据的时间戳 |
| data\_end\_time | DATETIME | user agent向NDO2DB最后一次发送数据的时间戳 |
| bytes\_processed | INT | user agent向NDO2DB发送数据的大小 |
| lines\_processed | INT | user agent向NDO2DB发送数据的行数 |
| entries\_processed | INT | user agent向NDO2DB发送数据的数据实体个数 |

### Historical Data Tables

保存Nagios和其监控的hosts/services的历史信息。历史信息并不老，也许就发生在5秒前，所以可利用这些表格报告当前状态。



### Current Status Tables

#### comments

保存当前host和service的评论。历史评论在commenthistory表中。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| comment\_id | INT | 主键 |
| instance\_id | SMALLINT | instance.instance\_id |
| object\_id | INT | objects.object\_id |

#### customvariablestatus

保存当前所有host，service，contact变量的键值对，在Nagios3.x之后才有。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| customvariablestatus\_id | INT | 主键 |
| instance\_id | SMALLINT | instance.instance\_id |
| object\_id | INT | objects.object\_id |

#### hoststatus

描述主机当前状态。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| hoststatus\_id | INT | 主键 |
| instance\_id | SMALLINT | instance.instance\_id |
| host\_object\_id | INT | objects.object\_id |
| timeperiod\_object\_id | INT | objects.object\_id |

#### programstatus

保存Nagios进程的当前状态。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| programstatus\_id | INT | 主键 |
| instance\_id | SMALLINT | instance.instance\_id |

#### runtimevariables

保存Nagios进程的运行时变量。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| runtimevariables\_id | INT | 主键 |
| instance\_id | SMALLINT | instance.instance\_id |

#### scheduleddowntime

保存当前host和service的故障时间。历史故障保存在downtimehistory表中。

# Capistrano

<https://ruby-china.org/topics/18616>

## 安装

ruby version > 1.9

tar zxvf ruby-2.1.5.tar.gz

./configure

make

make install

gem sources --remove https://rubygems.org/

gem sources -a <http://ruby.taobao.org/>

gem sources -l

gem install Capistrano

## 使用

**1、能无密码SSH到服务器；2、能无密码从github下载仓库**

生成密钥

ssh-keygen -t rsa

传到服务器

ssh-copy-id dwt@controller

或者

scp ~/.ssh/id\_rsa.pub [dwt@controller:~/.ssh/authorized\_keys](mailto:dwt@controller:~/.ssh/authrozied_keys)

ssh dwt@controller

chmod 700 .ssh

chmod 600 .ssh/authorized\_keys

exit

尝试登录看是否需要密码

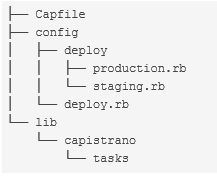
ssh dwt@controller

创建项目

mkdir myproject

cd myproject

执行cap install创建部署目录，deploy.rb为主要配置文件



## Capfile

用于配置插件

# Load DSL and set up stages

require 'capistrano/setup'

# Include default deployment tasks

require 'capistrano/deploy'

# Include tasks from other gems included in your Gemfile

#

# require 'capistrano/rvm'

# require 'capistrano/rbenv'

# require 'capistrano/chruby'

# require 'capistrano/bundler'

# require 'capistrano/rails/assets'

# require 'capistrano/rails/migrations'

# require 'capistrano/passenger'

# Load custom tasks from `lib/capistrano/tasks' if you have any defined

Dir.glob('lib/capistrano/tasks/\*.rake').each { |r| import r }

## deploy.db

# config valid only for current version of Capistrano

lock '3.3.3'

set :application, 'cloudwaf' **//应用名称**

set :scm, :git **//仓库类型，git和subversion**

set :repo\_url, 'https://github.com/dongwenting/cloudwaf.git' **//仓库URL**

set :user, "dwt" **//github用户名**

set :keep\_releases, 5 **//保存发布版本最大值为5**

set :deploy\_to, '/home/dwt/cloudwaf' **//部署路径**

set :log\_level, :debug **//日志等级，调试状态**

set :default\_env, { path: "/opt/ruby/bin:$PATH" } **//默认环境变量**

set :linked\_files, fetch(:linked\_files, []).push('config/database.yml')

set :linked\_dirs, fetch(:linked\_dirs, []).push('bin', 'log', 'tmp/pids', 'tmp/cache', 'tmp/sockets', 'vendor/bundle', 'public/system')

namespace :deploy do

after :restart, :clear\_cache do

on roles(:web), in: :groups, limit: 3, wait: 10 do

# Here we can do anything such as:

# within release\_path do

# execute :rake, 'cache:clear'

# end

end

end

end

## Stage文件

从Role角度配置服务器

role :app, "dwt@controller"

role :web, "dwt@controller"

role :db, "dwt@compute"

从Server角度配置服务器

server 'controller', user: 'dwt', roles: %w{web app}, my\_property: :my\_value

过滤Role，Server的方式：

1. **设置环境变量**

1. **在config/deploy.rb中配置**

1. **在命令行配置**



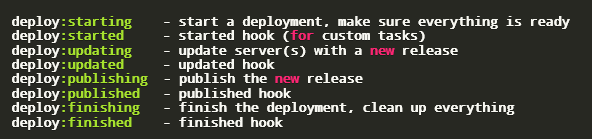
## 部署目录



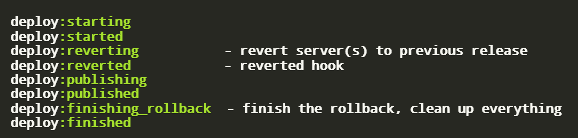
## 命令

cap deploy:check #检查部署目录

部署流程：



回滚流程：



## 问题

**问题1：Stage not set, please call something such as**

v2中可直接执行cap deploy，v3中必须指明stage，例如 cap production deploy

v3根据stage，直接到config/deploy/{stage}.rb文件