zabbix文件以及命令详解（二）

51niux.com/

2017年07月19日 15:55:46
zabbix的官网文档：https://www.zabbix.com/documentation/3.2/

一、zabbix配置文件详解

1.1 zabbix_server.conf配置文件详解

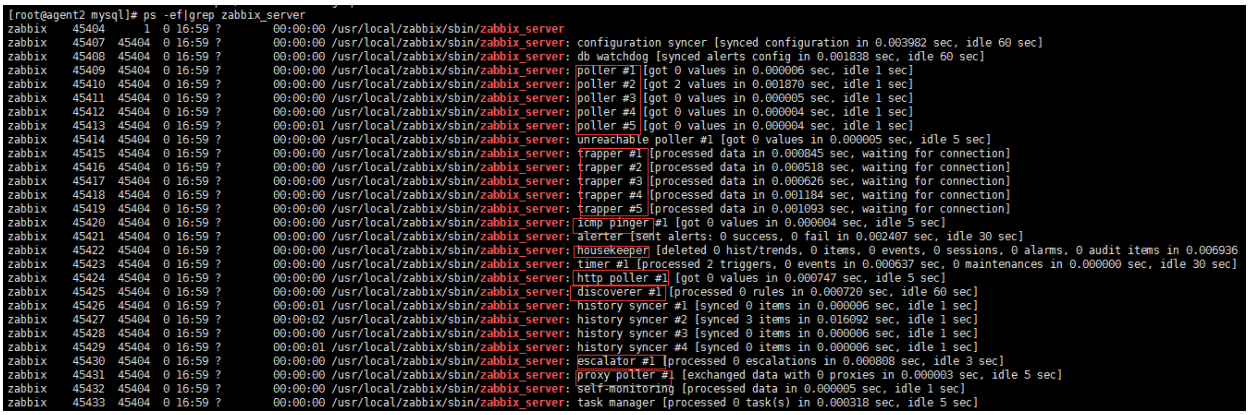
cat /etc/zabbix/zabbix_server.conf

```
############一般参数############
# ListenPort=10051        #trapper的监听端口，端口范围1024-32767，默认是10051
#SourceIP=                #出站连接的源IP地址。默认为空
# LogType=file            #指定日志消息写入的位置：system是写到syslog里面，file是写到本地指定的位置，console是标准输出到屏幕，这里默认是file
LogFile=/var/zabbix/zabbix_server.log   #这里一般要修改到我们指定位置的
# LogFileSize=1           #日志文件的最大大小（MB）就轮询日志。0 - 禁用自动日志旋转。范围是0-1024，默认是1M大小就轮询。
# DebugLevel=3            #日志调试级别，0-有关Zabbix进程启动和停止的基本信息，1-关键信息，2是错误信息，3是warnings级别，4是debug信息，5是扩展调试信息
# PidFile=/tmp/zabbix_server.pid  #pid文件的保存位置，也可以写到指定目录下
# DBHost=localhost        #连接数据库的主机地址，默认是localhost
DBName=zabbix             #连接数据库的库名称
# DBSchema=               #模式名称。用于IBM DB2和PostgreSQL。
DBUser=zabbix             #连接指定数据库的用户名
DBPassword=zabbix         #连接指定数据库用户名的密码
# DBSocket=/tmp/mysql.sock #MySQL套接字的路径。
# DBPort=3306              #数据库的端口号
############高级参数############
#StartPollers=5            #初始化时，启动子进程数量，数量越多，则服务端吞吐能力越强，对系统资源消耗越大。范围是0-1000如非特殊默认即可
# StartIPMIPollers=0       #主要用于IPMI技术用于获取硬件状态场景。若无相关监控项，建议设置为0
# StartPollersUnreachable=1 #默认情况下，ZABBIX会启用指定进程用于探测某些不可达主机的（含IPMI场景）；若使用场景中含有代理端，建议保持默认；若直接agent较多，可视具体情况调整
# StartTrappers=5          #Trappers初始子进程数
# StartPingers=1           #用于设置启用icmp协议PING主机方式启动子进程数量，若单台代理所管理机器超过500台，建议加大此数值
StartDiscoverers=1         #用于设置自动发现主机的子进程数量，若单台代理所管理机器超过500台，可以考虑加大此数值（仅适用于直接AGENT场景）
# StartHTTPPollers=1       #HTTP主动监测的进程数
# StartTimers=1            #计时器的预分支实例数。定时器处理基于时间的触发功能和维护期。 只有第一个定时器进程处理维护期。
# StartEscalators=1        #用于处理动作中的步骤的进程的数量
# JavaGateway=             #Zabbix Java网关的IP地址（或主机名).仅在Java轮询器启动时才需要。
# JavaGatewayPort=10052    #Zabbix Java网关侦听的端口。
# StartJavaPollers=0       #Java轮询器的预分支实例数。
# StartVMwareCollectors=0  #用于设置监控VMWARE Esxi主机实例时使用，若为0则不启用，若要监控ESXI主机，此值最少为1 ；视监控ESXI数量设置对应数值
# VMwareFrequency=60       #Zabbix将连接到VMware服务以获取新数据的频率。默认是60秒
```

```
# VMwarePerfFrequency=60    #Zabbix将连接到VMware服务以获取性能数据的频率。默认是60秒
# VMwareCacheSize=8M        #划出多少共享内存用于存储VMWARE数据,范围是256K-2G
# VMwareTimeout=10          #指定vmware收集器等待VMware服务响应的秒数。
# SNMPTrapperFile=/tmp/zabbix_traps.tmp  #指定SNMP TRAPPER时的临时文件，用于代理端启用SNMP TRAPPER功能时使用，必须与zabbix_trap_receiver.pl或SNMPTT配置文件中的相同。
# StartSNMPTrapper=0        #是否启用 snmptrapper功能 ，默认不启用=0，启用=1（配合参数SNMPTrapperFile使用）
ListenIP=127.0.0.1          #监听地址，留空则会在所有的地址上监听，可以监听多个IP地址，ip之间使用逗号分隔，默认是0.0.0.0
# HousekeepingFrequency=1  #多少小时清理一次代理端数据库的 history, alert, and alarms，以保持代理端数据库轻便，范围是0-24
# MaxHousekeeperDelete=5000 #每次最多删除历史数据的行数，范围是0-1000000
# SenderFrequency=30        #多少秒后重试发送失败的报警信息，范围是5-3600
# CacheSize=8M              #配置缓存的大小（以字节为单位）。用于存储主机，项目和触发器数据的共享内存大小。范围是128K-8G
# CacheUpdateFrequency=60  #Zabbix更新缓存数据的频率，单位为秒，范围是1-3600
# StartDBSyncers=4          #DB同步进程数量
# HistoryCacheSize=16M      #历史缓存的大小（以字节为单位）。用于存储历史数据的共享内存大小。范围是128K-2G
# HistoryIndexCacheSize=4M #历史索引缓存的大小（以字节为单位）。用于索引历史缓存的共享内存大小。范围是128K-2G
# TrendCacheSize=4M         #用于设置划分多少系统共享内存用于存储计算出来的趋势数据，此参数值从一定程度上可影响数据库读压力，范围是128K-2G
# ValueCacheSize=8M         #历史值缓存的大小，以字节为单位。 用于缓存项历史数据请求的共享内存大小。 设置为0将禁用值缓存。范围是0,128K-64G
Timeout=4                   #指定等待代理，SNMP设备或外部检查的时间（以秒为单位）。
# TrapperTimeout=300        #Trapper处理新数据的最长时间，单位是秒，范围是1-300
# UnreachablePeriod=45      #当主机不可达多少秒后，设置为主机不可用，单位是秒，范围是1-3600
# UnavailableDelay=60       #当主机不可用了，多久检查一次该主机的可用性，单位为秒，范围是1-3600
# UnreachableDelay=15       #当主机不可到达了，多久检查一次该主机的可用性，单位为秒，范围是1-3600
# AlertScriptsPath=${datadir}/zabbix/alertscripts  #监控报警脚本路径,取决于编译时候的datadir参数
# ExternalScripts=${datadir}/zabbix/externalscripts  #自定义脚本存储路径
# FpingLocation=/usr/sbin/fping  #fping的位置 确保fping二进制有root权限和SUID标志设置。
# Fping6Location=/usr/sbin/fping6 #fping6的位置 确保fping6二进制有root权限和SUID标志设置。
# SSHKeyLocation=           #用于SSH检查和操作的公钥和私钥的位置。
LogSlowQueries=3000         #数据库查询在记录之前可能需要多长时间（以毫秒为单位）。仅当DebugLevel设置为3,4或5时才可用。0 - 不记录慢查询。范围是1-3600000
# TmpDir=/tmp               #临时目录
# StartProxyPollers=1       #启用多少子进程与代理端通信，若代理端较多可考虑加大此数值，范围是0-250
# ProxyConfigFrequency=3600 #proxy被动模式下，server多少秒同步配置文件至proxy。该参数仅用于被动模式下的代理。范围是1-3600*24*7
# ProxyDataFrequency=1       #被动模式下，zabbix server间隔多少秒向proxy请求历史数据
# AllowRoot=0              #是否允许以root身份运行服务端，0是不允许如果禁用并且服务器由“root”启动，服务器将尝试切换到用户配置选项指定的用户。，1是允许。
# User=zabbix              #运行使用的用户
# Include=                 #可以将单个文件或所有文件包含在配置文件中的目录中。安装Zabbix将在/usr/local/etc中创建include目录，除非在编译期间进行修改。
# SSLCertLocation=${datadir}/zabbix/ssl/certs   #SSL客户端证书的位置。 此参数仅用于Web监控。
# SSLKeyLocation=${datadir}/zabbix/ssl/keys  #SSL客户端证书的私钥位置。此参数仅用于Web监控。
# SSLCALocation=           #SSL CA钥文件目录
# LoadModulePath=${libdir}/modules    #服务器模块位置的完整路径。 默认值取决于编译选项。
```

LoadModule=　　#
 260/5000在服务器启动时加载模块。 模块用于扩展服务器的功能。 格式：LoadModule = <module.so>模块必须位于LoadModulePath指定的目录中。 允许包含多个LoadModule参数。
TLSCAFile=　　#包含用于对等证书验证的顶级CA证书的文件的完整路径名。
TLSCRLFile=　　#包含撤销证书的文件的完整路径名。
TLSCertFile=　　#包含服务器证书或证书链的文件的完整路径名。
TLSKeyFile=　　#包含服务器私钥的文件的完整路径名。

上面配置了那么多，下面我们通过图来看一下进程吧：



#可以看到启动用户是zabbix，然后左边是一堆的PID号，PID号后面是PPID号也就是父进程号都是统一的45404，然后最后边可以看到不同的进程标识,这些进程是干嘛的，可以跟下面的内容核对：

报警器(alerter)—该类型的进程是用来发送报警通知的；
配置同步器(configuration syncer)—用于将配置文件中的配置信息同步到内存中缓存；
数据发送器(data sender)—服务器代理节点用于发送数据的进程（服务器端没有这类进程）；
数据库看门狗(db watchdog)—该进程用于监视zabbix系统的数据库状态，当数据库状态变为不可用时，发送警告信息（服务器代理端不支持这类型进程）。
自动发现器(discoverer)—用于自动发现设备的进程；
步骤(escalator)—用于处理动作中的步骤的进程；
心跳发送器(heartbeat sender)—服务器代理端用于发送心跳信息（服务器端没有这类型的进程）；
历史数据同步器(history syncer)—用于写历史数据表；
管家(housekeeper)—用于清理过期的历史数据的进程；
HTTP 轮询器(http poller)—用于轮询web类的监控项目；
Ping检查器(icmp pinger)—用于定期的进行ICMP PING检查；
ipmi 轮询器（ipmi poller）—用于定期进行ipmi监控项目的检查；
java 轮询器(java poller)—用于轮询java 监控项目；
分布式节点看守器(node watcher)—用于在不同的分布式节点发送历史数据和配置信息更新的进程；
轮询器(poller)—用于普通的被动监控项目的轮询；
服务器代理轮询(proxy poller)—用于服务器代理的被动轮询；
自我监控(self-monitoring)—用于收集Zabbix系统内部的监控信息；
定时器(timer)—用于处理触发器中也时间相关的函数和维护模式的进程；
陷入器(trapper)—用于处理主动采集、陷入以及分布式节点间或服务器代理的通信；
不可到达轮询器(unreachable poller)—用于轮询不可到达到的设备；
vmware 收集器(vmware collector)—负责从vmware服务进程中收集数据（服务器代理端不支持这种类型的进程）；

下面是一个链接可以看看，

zabbix的内部数据采集：http://blog.chinaunix.net/xmlrpc.php?r=blog/article&id=4210971&uid=9411004

zabbix系统数据采集方法总结 :http://blog.chinaunix.net/xmlrpc.php?r=blog/article&uid=9411004&id=4115731

博文来自：www.51niux.com

1.2 zabbix_agentd.conf配置文件详解

cat /etc/zabbix/zabbix_agentd.conf #这是zabbix客户端的配置文件，这里好多参数跟server的配置文件里面的意思一致就不过多的解释了

```
############一般参数############
# PidFile=/tmp/zabbix_agentd.pid            #pid文件位置
LogFile=/tmp/zabbix_agentd.log              #log文件的位置，如果不设置则使用syslog也就是写入/var/log/message
# LogFileSize=1                             #日志轮询大小默认是1MB
# DebugLevel=3                              #日志级别
# SourceIP=                                 #出站连接的源IP地址，当系统有多个IP的时候需要制定哪个IP与代理或服务端通信
# EnableRemoteCommands=0                    #是否允许来自Zabbix服务器的远程命令。0为不允许，1为允许。
# LogRemoteCommands=0                       #是否开启日志记录shell命令作为警告，0表示不允许，1表示允许。
#####被动检查相关(被动模式：被动模式下，由代理或服务端主动请求AGENT，去获取所采集到的监控数据)
Server=192.168.1.103                        #在有代理情况下，此IP地址应该填写代理服务器的IP，反之，若无代理服务器，则此IP应设置为服务端，多IP用逗号隔开
# ListenPort=10050                          #agent的监听服务端或者代理的连接端口，范围是1024-32767，默认是10050
# ListenIP=0.0.0.0                          #监听IP
# StartAgents=3                             #在被动模式下，agent启动时启动的子进程数量，范围是0-100，如果设置为0，则禁用被动检查，并且代理将不会在任何TCP端口上侦听。
#####主动检查相关（主动模式：在主动模式下，AGENT端（即采集客户端）将所采集的结果，主动提交给代理服务器或服务器，而此种情况下，代理服务器或服务器将被动接收采集信息）
# ServerActive=                             #主动模式下，代理端口的IP，如果没有指定端口则默认为10051端口，若需要更改端口，则为IP:port的形式
# Hostname=                                 #唯一，区分大小写的主机名。需要活动检查，并且必须与服务器上配置的主机名匹配。 如果未定义，则从HostnameItem获取值。手工自定义一个主机名，可以和系统的主机名一样，也可以不一样
# HostnameItem=system.hostname              #这里的优先级低于上面的Hostname。system.hostname是ZABBIX内置的一个自动获取主机名的方法
# HostMetadata=                             #用于定义当前主机唯一标识符，范围是0-255，仅适用于自动发现情况下，默认不定义。如果未定义，则将从HostMetadataItem获取值。
# HostMetadataItem=                         #定义用于获取主机元数据的项目的可选参数。主机自动注册过程使用主要的元数据。 在自动注册请求期间，如果指定项目返回的值超过255个字符，代理将记录一条警告消息。 仅当未定义HostMetadata时才使用此选项。
# RefreshActiveChecks=120                   #被监控的主机多久（秒）重新请求代理或服务端刷新一监控列表。范围为60-3600秒。ZABBIX运行原理为：, zabbix客户端启动后，在等待RefreshActiveChecks秒后，
#开始从代理或服务端请求并下载监控项信息，保存在本地专门的buffersend中，再过RefreshActiveChecks秒后，重新获取监控项信息。这就是为什么当配置监控项，要过一会才能生效的原因。这个数值，就是等待时间。建议，不要将此数值设置过小，以免加大AGENT端和服务端及数据库的压力，建议为120秒。
# BufferSend=5                              #多少秒后，将BUFFER中的数据提交到代理或服务端。范围（1-3600）此数值的大小决定了采集后，提交数据的及时性，数值越小，则提交得越频繁，
#对服务器压力越大，同时对AGENT端系统资源消耗越大，则表现出来的现象是报警非常及时，建议根据实际情况自行考虑，也可保持默认，若发现ZABBIX消耗资源较多，建议加大此数值。
# BufferSize=100                            #此参数作用设置保存采集数据在内存中的容量大小。若此agent端监控项较多，建议加大此数值。BufferSize与BufferSend之间有联系的。当达到bUFFERSEND或
```

```
Buffersize已满时，都会触发数据提交动作。范围是2-65535
# MaxLinesPerSecond=100                    #代理将每秒发送到Zabbix服务器或代理处理的最大新行数，范围是1-1000
############高级参数############
# Default:                                 #设置项目键的别名。 它可以用来代替较长和更简单的长而复杂的项目密钥。可能存在多个别名参数。 不允许具有相同别名键的多个参数。 不同的别名键可以引用相同的项目键。例如：Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,^zabbix:.:([0-9]+),,,,\1]
# Timeout=3                                #gant采集一个数据的超时时间，但是是秒，范围是1-30
# AllowRoot=0                              #是否允许ROOT帐号运行此客户端。0：不允许，1:允许
# User=zabbix                              #运行agent的用户
# Include=                                 #加载目录路径或扩展配置文件路径
#######用户定义的监控参数#######
# UnsafeUserParameters=0                   #是否启用用户自定义监控脚本，1启用，0不启用。由于ZABBIX实现监控方法的多样性，一般都采用脚本来实现监控数据的采集，所以，建议开启，否则功能将受限。
# UserParameter=                           #用户定义的参数进行监控。 可以有几个用户定义的参数。格式：UserParameter = <key>，<shell命令>请参见“zabbix_agentd”目录中的示例。
#######扩展模块#######
# LoadModulePath=${libdir}/modules         #扩展模块路径
# LoadModule=                              #扩展模块路径
```

1.3 zabbix_proxy.conf配置文件详解

cat /etc/zabbix/zabbix_proxy.conf #这是zabbix代理的配置文件

```
############一般参数############
# ProxyMode=0                 #代理操作模式。 0 - 代理在主动模式，1 - 代理在被动模式
# Server=                     #Zabbix服务器的IP地址（或主机名）。Active Proxy将从服务器获取配置数据。当 Proxy 处于被动模式时，该参数将被忽略。
# ServerPort=10051            #Zabbix Server 监听端口，同上只在 Proxy 为主动模式时生效
Hostname=Zabbix proxy         #手工设置zabbix获取的主机名称
# HostnameItem=system.hostname  #如果上面没设置就按这里通过zabbix的内置函数获取
# ListenPort=10051            #自己本地的监听端口
# SourceIP=                   #多IP下要设置
# LogType=file                #日志的存储类型
LogFile=/tmp/zabbix_proxy.log #日志的存储位置
# LogFileSize=1               #日志轮询大小
# DebugLevel=3                #日志级别
# PidFile=/tmp/zabbix_proxy.pid  #pid位置
# DBHost=localhost            #连接哪个主机库
DBName=zabbix_proxy           #数据库名称
# DBSchema=
DBUser=zabbix                 #数据库用户名
# DBPassword=                 #数据库密码
# DBSocket=/tmp/mysql.sock    #数据库sock位置
# DBPort=3306                 #数据库端口
#########代理特定参数#########
# ProxyLocalBuffer=0          #即使数据已经与服务器同步，代理将在本地保留数据N小时。如果本地数据将被第三方应用程序使用，则可以使用此参数。范围是0-720
# ProxyOfflineBuffer=1        #如果与Zabbix Server无连接，代理将保留数据N小时。 较旧的数据将丢失。范围是1-720
# HeartbeatFrequency=60       #心跳消息的频率（秒）用于监视服务器端的代理服务器的可用性。
 0 - 心跳消息被禁用。 对于被动模式下的代理，该参数将被忽略。范围是0-3600
# ConfigFrequency=3600        #代理在几秒钟内从Zabbix Server检索配置数据的频率。 对于被动模式下的代理，该参数将被忽略。范围是1-3600*24*7
# DataSenderFrequency=1       #代理将每N秒将收集的数据发送到服务器。 对于被动模式下的代理，该参数将被忽略。范围是1-3600
#########高级参数############（这就跟zabbix_agentd.conf一样了，就不重复了）
```

博文来自：www.51niux.com

二、zabbix常用命令

2.1 zabbix_server（zabbix_proxy用法一致）命令

使用语法：

```
zabbix_server [-hV] [-c <file>] [-n <nodeid>] [-R <option>]
```

选项：

```
-c --config config-file   #配置文件的路径
-f --foreground           #在前台运行zabbix_server服务
-R --runtime-control runtime-option   #执行管理功能
   运行时控制选项：
   config_cache_reload   #重新加载配置缓存
   housekeeper_execute   #执行管家
   log_level_increase=target  #增加日志级别，如果未指定目标，则会影响所有进程
   log_level_decrease=target  #降低日志级别，如果未指定目标，则会影响所有进程
```

/usr/local/zabbix/sbin/zabbix_server -c /etc/zabbix/zabbix_server.conf #常用的还是这个

写到启动脚本里面，启动zabbix_server服务

2.2 zabbix_agentd命令

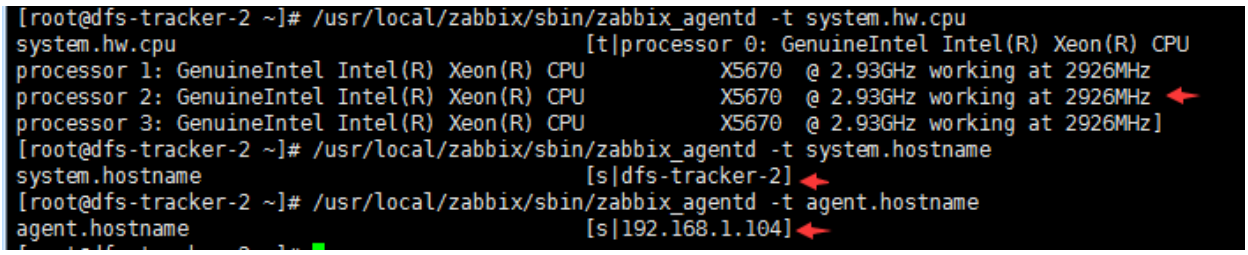
/usr/local/zabbix/sbin/zabbix_agentd -h

```
用法：
    zabbix_agentd [-c config-file]
    zabbix_agentd [-c config-file] -p
    zabbix_agentd [-c config-file] -t item-key
    zabbix_agentd [-c config-file] -R runtime-option
    zabbix_agentd -h
    zabbix_agentd -V
选项（-c,-f,-R同上）：
    -p --print   #打印
    -t --test item-key  #测试指定项目并退出
```

例子如：

/usr/local/zabbix/sbin/zabbix_agentd -p #将客户端自身所有的item-key以及其值都打印出来了，内容太多就不粘贴了

/usr/local/zabbix/sbin/zabbix_agentd -t system.hw.cpu #获取CPU的信息
/usr/local/zabbix/sbin/zabbix_agentd -t system.hostname #获取系统的主机名而非zabbix_agentd配置文件里面定义的那个Hostname
/usr/local/zabbix/sbin/zabbix_agentd -t agent.hostname #这才是获取agentd配置文件里面定义的Hostname



2.3 zabbix_get命令

/usr/local/zabbix/bin/zabbix_get -h #这个主要是zabbix的server端或者是proxy端，来测试是否能从被监控的主机获取数据

```
用法：zabbix_get [-hV] -s <host name or IP> [-p <port>] [-I <IP address>] -k <key>
选项：
    -s --host <host name or IP>     #指定主机的主机名或IP地址
    -p --port <port number>         #指定主机上运行的代理端口号。 默认值为10050
    -I --source-address <IP address> #指定源IP地址
    -k --key <key of metric>          #指定要检索的值的项目的键
```

例子：

/usr/local/zabbix/bin/zabbix_get -s 192.168.1.104 -p 10050 -I 192.168.1.103 -k "system.hostname" #向192.168.1.104发起get请求，-p 是10050端口，-I 来源IP是192.168.1.103，-k 想获取的key

/usr/local/zabbix/bin/zabbix_get -s 192.168.1.105 -k "system.cpu.load[all,avg15]" #向192.168.1.105获取load每15分钟的负载值（这是一个比较简单的写法一般也这么写）

2.4 zabbix_sender命令

zabbix获取key值有超时时间，如果自定义的key脚本一般需要执行很长时间，这根本没法去做监控，那怎么办呢？使用zabbix监控类型zabbix trapper，需要配合zabbix_sender给它传递数据。

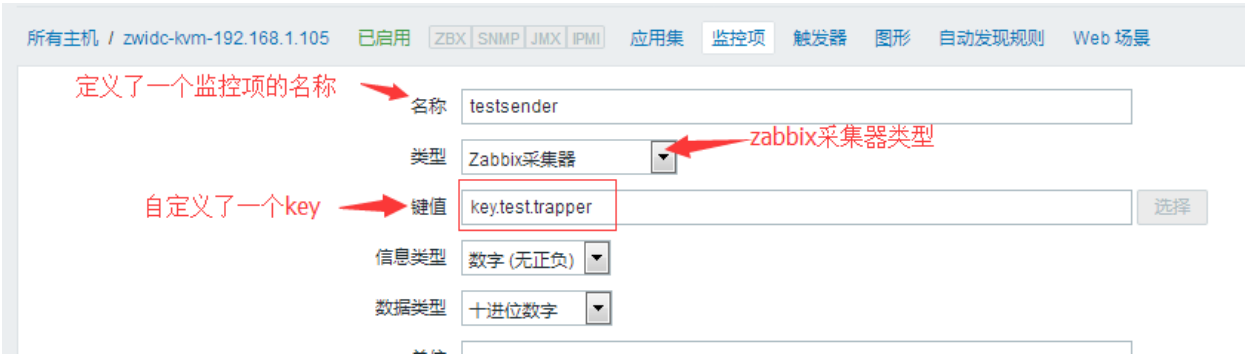
/usr/local/zabbix/bin/zabbix_sender -h

```
用法：usage: zabbix_sender [-Vhv] {[-zpsI] -ko | [-zpI] -T -i <file> -r} [-c <file>]
选项：
-c --config <file>     #配置文件绝对路径
-z --zabbix-server <server>   #zabbix server的IP地址
-p --port <server port>       # zabbix server端口，默认10051
-s --host <hostname>          #主机名，zabbix配置文件里面定义的Hostname
-I --source-address <IP address>  #源IP
-k --key <key>                #监控项的key
-o --value <key value>        #key值
-i --input-file <input file>   #从文件里面读取hostname、key、value 一行为一条数据，使用空格作为分隔符，如果主机名带空格，那么请使用双引号包起来
-T --with-timestamps          #一行一条数据，空格作为分隔符: <hostname> <key> <timestamp> <value>，配合 --input-file option，timestamp为unix时间戳
-r --real-time                #将数据实时提交给服务器
-v --verbose                  #详细模式，-vv 更详细
```

例子：

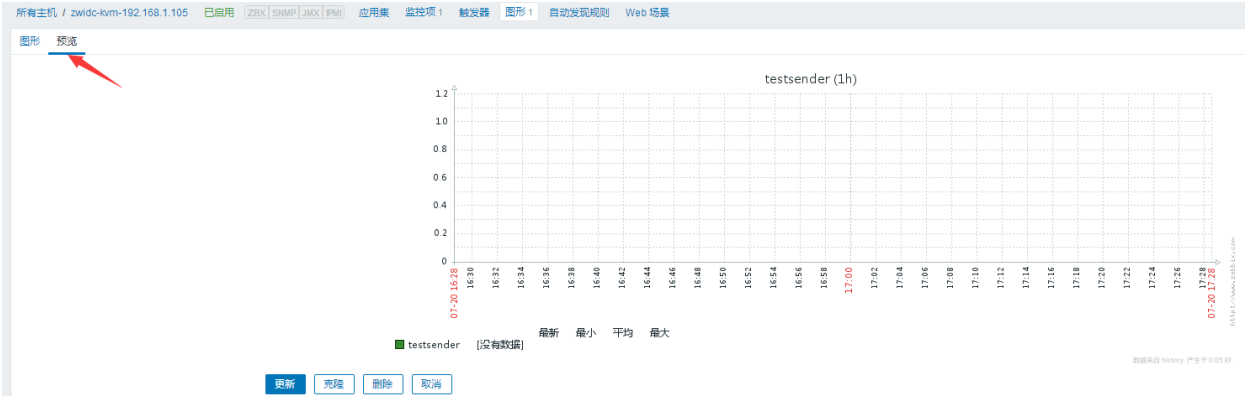
现在我们拿新创建的主机192.168.1.105进行测试

#我们先为192.168.1.105添加一个监控项



#点击监控项右边的图形，我们再为其创建一个图形方便看结果

#这个图形的右边有个预览，可以直接查看效果



#之所以上面没有数据，是因为我们客户端还没有sender数据，下面在客户端发送一下数据

/usr/local/zabbix/bin/zabbix_sender -s zwidc-kvm-192.168.1.105 -z 192.168.1.103 -k "key.test.trapper" -o 1 #参数上面已经解释了注意，-s 是必须要带的，必须要带Hostname

```
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000043"
sent: 1; skipped: 0; total: 1
```

#从上面的记过看failed:0，说明没有失败上传成功了，这时候你去看zabbix的图形已经出现数据了，这时候为了效果好看一点，我们多上传点数据

cat testsend #我们直接用发送文件内容的形式，下面的文件并非都只能是一种key，还可以多种kye，反正一行就是自己Hostname的名称 key 值

```
"zwidc-kvm-192.168.1.105" key.test.trapper 2
"zwidc-kvm-192.168.1.105" key.test.trapper 3
"zwidc-kvm-192.168.1.105" key.test.trapper 5
"zwidc-kvm-192.168.1.105" key.test.trapper 10
"zwidc-kvm-192.168.1.105" key.test.trapper 11
"zwidc-kvm-192.168.1.105" key.test.trapper 13
"zwidc-kvm-192.168.1.105" key.test.trapper 17
```

/usr/local/zabbix/bin/zabbix_sender -z 192.168.1.103 -i testsend #直接用-i 后面跟文件

```
info from server: "processed: 7; failed: 0; total: 7; seconds spent: 0.000201"
sent: 7; skipped: 0; total: 7
```

#从上面的结果来看没有失败，7条数据都发送给服务端了

#再次查看zabbix_server端的图形，直接看数据，最大是17，最小是1.效果出来了。

博文来自：www.51niux.com

三、zabbix数据库结构

3.1 数据库表总览

MariaDB [zabbix]> show tables;

```
+----------------------------+
| Tables_in_zabbix           |
+----------------------------+
| acknowledges               |
| actions                    |
| alerts                     |
| application_discovery      |
| application_prototype      |
| application_template       |
| applications               |
| auditlog                   |
| auditlog_details           |
| autoreg_host               |
| conditions                 |
| config                     |
| corr_condition             |
| corr_condition_group       |
| corr_condition_tag         |
| corr_condition_tagpair     |
| corr_condition_tagvalue    |
| corr_operation             |
| correlation                |
| dbversion                  |
| dchecks                    |
| dhosts                     |
| drules                     |
| dservices                  |
| escalations                |
| event_recovery             |
| event_tag                  |
| events                     |
| expressions                |
| functions                  |
| globalmacro                |
| globalvars                 |
| graph_discovery            |
```

```
| graph_theme                |
| graphs                     |
| graphs_items               |
| group_discovery            |
| group_prototype            |
| groups                     |
| history                    |
| history_log                |
| history_str                |
| history_text               |
| history_uint               |
| host_discovery             |
| host_inventory             |
| hostmacro                  |
| hosts                      |
| hosts_groups               |
| hosts_templates            |
| housekeeper                |
| httpstep                   |
| httpstepitem               |
| httptest                   |
| httptestitem               |
| icon_map                   |
| icon_mapping               |
| ids                        |
| images                     |
| interface                  |
| interface_discovery        |
| item_application_prototype |
| item_condition             |
| item_discovery             |
| items                      |
| items_applications         |
| maintenances               |
| maintenances_groups        |
| maintenances_hosts         |
| maintenances_windows       |
| mappings                   |
| media                      |
| media_type                 |
| opcommand                  |
| opcommand_grp              |
| opcommand_hst              |
| opconditions               |
| operations                 |
| opgroup                    |
| opinventory                |
| opmessage                  |
| opmessage_grp              |
| opmessage_usr              |
| optemplate                 |
| problem                    |
| problem_tag                |
| profiles                   |
| proxy_autoreg_host         |
| proxy_dhistory             |
| proxy_history              |
```

```
| regexps                   |
| rights                    |
| screen_user               |
| screen_usrgrp             |
| screens                   |
| screens_items             |
| scripts                   |
| service_alarms            |
| services                  |
| services_links            |
| services_times            |
| sessions                  |
| slides                    |
| slideshow_user            |
| slideshow_usrgrp          |
| slideshows                |
| sysmap_element_url        |
| sysmap_url                |
| sysmap_user               |
| sysmap_usrgrp             |
| sysmaps                   |
| sysmaps_elements          |
| sysmaps_link_triggers     |
| sysmaps_links             |
| task                      |
| task_close_problem        |
| timeperiods               |
| trends                    |
| trends_uint               |
| trigger_depends           |
| trigger_discovery         |
| trigger_tag               |
| triggers                  |
| users                     |
| users_groups              |
| usrgrp                    |
| valuemaps                 |
+---------------------------+
127 rows in set (0.01 sec)
```

3.2 基础表结构简介

actions表：

actions表记录了当触发器触发时，需要采用的动作。可以通过desc actions;来自行查看表结构

alerts表：

alerts 表保存了历史的报警事件。

config表：

config表保存了全局的参数

functions表：

function 表是非常重要的一个表了，记录了trigger中使用的表达式，例如max、last、nodata等函数。

graphs表：

graphs 表包含了用户定义的图表信息。

graphs_items表：

graphs_items 保存了属于某个图表的所有的监控项信息。

groups表：

groups 保存了组名和组的ID 。

history 、history_str、history_log 、history_uint_sync等：

这部分表都差不多，唯一不同的是保存的数据类型。存储着不同类型item的历史数据，最终1小时或者1天等短时间的绘图数据都从其中获取。

trends、trends_uint表：

保留历史数据用的，不过是趋势数据。储存着不同类型item的历史趋势数据，每隔一小时从历史数据中统计一次，并计算统计区间的平均值、最值。长时间区间的绘图数据的数据源。

hosts表：

hosts 非常重要，保存了每个agent、proxy等的IP 、hostid、状态、IPMI等信息， 几乎是记录了一台设备的所有的信息。其他的表一般都时关联hostid的。

hosts_groups表：

hosts_groups 保存了host（主机）与host groups（主机组）的关联关系。

items表：

items 表保存了采集项的信息。字段说明，itemid是每个绘图项目唯一标识，hostid每个主机的标识，name每个item的名字，delay数据采集间隔，history历史数据保存时间，status标识item的状态(0表示正常显示的item)，units保存item的单位。

media表：

media 保存了某个用户的media配置项，即对应的告警方式。

media_type表：

media_type 表与media 表不同的是media_type 记录了某个告警方式对应的执行脚本，注意路径只是相对路径。media 与media_type 通过mediatypeid 键关联。

profiles表：

profiles 表保存了用户的一些配置项。

rights表：

rights 表保存了用户组的权限信息，zabbix的权限一直也是我理不太清的地方， 其实这个表里面有详细的记录。

screeens表：

screeens 表保存了用户定义的图片。

sessions表：

保存了每个用户的sessions,在登陆、注销的时候均会操作该张表的。

triggers表：

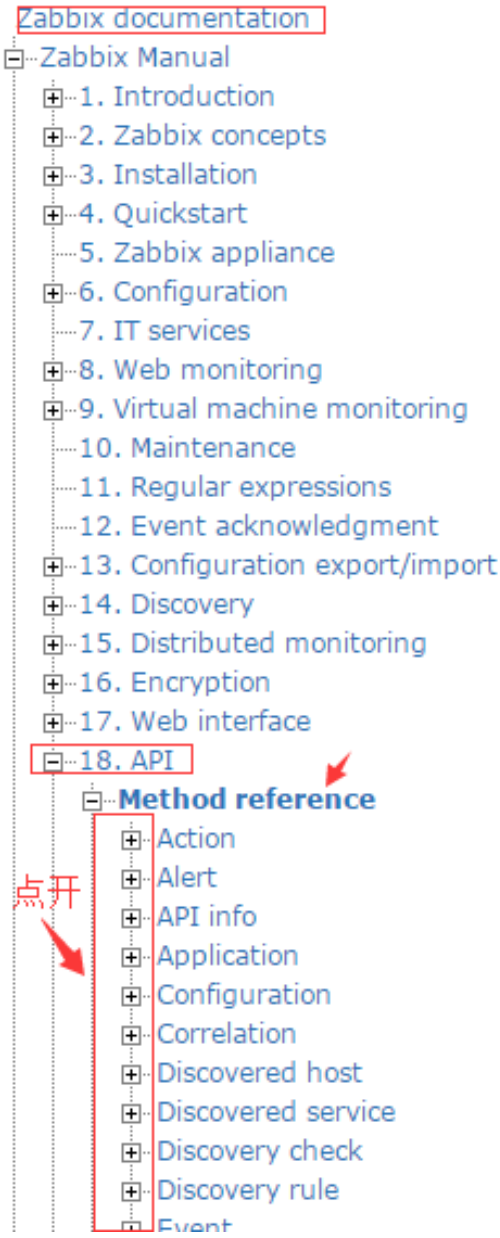
保存了trigger的所有信息。

trigger_depends表：

trigger_depends 保存了trigger的依赖关系。

再详细的内容可以参考：https://www.zabbix.com/documentation/3.2/manual/api/reference

#看到最后的小加号，可以点开仔细的查看里面的内容，里面有很多数据库的字段所对应的意思。



« 上一篇 下一篇 »

发表评论取消回复

一	二	三	四	五	六	日
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

« 2018年6月 »

搜索

网站分类

- 命令使用技巧
- linux性能检测工具
- linux文本编辑工具
- linux网络检测工具
- 环境部署
- 网络文件共享
- 系统无人值守安装
- Web环境搭建

- 邮件服务器
- 隧道与代理
- yum源服务
- DNS服务器
- 负载均衡
- 安全与优化
- 防火墙
- JVM
- 解决小问题
- 自动化与集群运维管理
- ldap
- puppet系列
- SaltStack系列
- ansible系列
- 运维工具使用
- git使用
- 监控系统
- cacti
- nagios
- ganglia
- zabbix
- svn使用
- 虚拟化
- KVM
- OpenStack
- Docker
- 日志收集
- rsyslog日志收集
- ELK
- 数据库与缓存服务
- Redis
- MongoDB
- 分布式文件系统
- 大数据
- Jenkins

友情链接

王玉鹏的官方网站

站点信息

- 文章总数:195
- 页面总数:2
- 分类总数:42
- 标签总数:0
- 评论总数:91
- 浏览总数:236645

控制面板

您好,欢迎到访网站!

[管理登陆] [查看权限]