# Zabbix系统监控应用详解

目录

[Zabbix系统监控应用详解 1](#_Toc10687)

[1.1 为何需要监控系统 11](#_Toc31702)

[1.2 监控系统的实现 12](#_Toc4744)

[1.3 监控系统开源软件 12](#_Toc18622)

[2.1 Zabbix简介 15](#_Toc5031)

[2.2 Zabbix概述 15](#_Toc5373)

[2.3 选择Zabbix七大理由 15](#_Toc10941)

[2.3 Zabbix的功能特性 16](#_Toc7270)

[3.1 Zabbix安装与部署 18](#_Toc28192)

[3.1.1 Zabbix-Server服务器端的安装 18](#_Toc1668)

[3.1.2安装Zabbix-Server 18](#_Toc20369)

[3.1.2.1 创建zabbix的数据库 19](#_Toc19692)

[3.1.2.2 导入Zabbix-Server数据库 19](#_Toc24007)

[3.1.2.3 配置zabbix\_server.conf 20](#_Toc6439)

[3.1.2.4 启动Zabbix-Server服务 21](#_Toc17188)

[3.1.2.5 Php.ini配置文件的设置(如果不配置是无法安装) 22](#_Toc32513)

[3.1.2 配置Web界面 22](#_Toc29527)

[3.1.3故障处理 24](#_Toc24129)

[3.1.4 启动zabbix-server端 25](#_Toc18096)

[3.1.5 服务器端修改zabbix\_agentd.conf配置文件 25](#_Toc20236)

[3.2 Zabbix-Agent客户端的安装 26](#_Toc31088)

[3.2.1安装Zabbix-Agent 26](#_Toc22152)

[3.2.2 配置zabbix\_agentd.conf 26](#_Toc30313)

[3.3 Zabbix-Get的使用 27](#_Toc19835)

[3.4 Zabbix相关术语(命令) 28](#_Toc20375)

[4.1自动化监控利器-Zabbix深入配置和使用 28](#_Toc28920)

[4.2 配置流程 28](#_Toc7335)

[4.3 添加主机组 29](#_Toc5463)

[4.4 添加新的主机组 30](#_Toc30335)

[4.5 添加模板 32](#_Toc30221)

[4.4.1.配置模板的步骤为： 32](#_Toc16454)

[4.4.2 创建模板 33](#_Toc12572)

[4.4.3 填入模板名称和所属的组， 34](#_Toc12678)

[4.4.4 模板具有继承的功能（将一个模板在另外一个模板中使用）， 34](#_Toc29261)

[4.4.5 模板中也可以设置宏 35](#_Toc16839)

[4.4.6 查看新建的模板， 35](#_Toc24963)

[4.5 添加主机 35](#_Toc31365)

[4.5.1 在图4-1中单击Create host， 35](#_Toc7887)

[4.5.2 选择模板，如图4-3所示。 37](#_Toc7216)

[4.5.3 搜索模板，如图4-4所示。 37](#_Toc31935)

[4.5.4 搜索出来后，打击Add添加该模板， 39](#_Toc19115)

[4.5.5 添加了两个模板，单击Add保存， 39](#_Toc25871)

[5.1 Graphs配置 39](#_Toc20634)

[5.1.1 在所在的主机模板中，选择Graphs，如图5-1所示。 40](#_Toc18497)

[5.1.2 单击“Create graph”按钮，如图5-2所示。 40](#_Toc13092)

[5.1.3 出现新建图形的界面，输入Name，选择所需要的Items， 41](#_Toc15079)

[5.1.4 单击Graphs，可以看到其中已经添加了一个Graphs， 41](#_Toc20617)

[5.1.5 单击刚才创建的Graphs，选择preview，即可查看图形 41](#_Toc13903)

[5.1.6 如果要查看所有的图形，单击Monitoring→Graphs， 42](#_Toc18607)

[5.1.7 选择对应的Group、Host、Graph， 42](#_Toc24427)

[5.1.8 显示图形如图5-8所示 42](#_Toc9940)

[6.1 Screen配置 43](#_Toc3633)

[6.2 单击创建Screen名称， 44](#_Toc15873)

[6.3 单击Change， 44](#_Toc24777)

[6.4 出现如图6-6所示 44](#_Toc30202)

[6.5 选择对应的Graphs，这里选择选择cpuload图形， 46](#_Toc16696)

[6.6 单击update保存， 46](#_Toc29882)

[6.7 配置好后，通过单击Monitor→Screen，选择相应的Screen组， 46](#_Toc14708)

[7.1 添加Items 47](#_Toc12473)

[7.1.1 Items的含义 47](#_Toc4351)

[7.1.2 自定义Items 48](#_Toc6808)

[7.1.3 找到Template OS Linux,单击Items 49](#_Toc18561)

[7.1.4 单击Create items创建Item， 49](#_Toc16605)

[7.1.5 单击Select按钮，列出Agent所有自带的Key， 49](#_Toc27377)

[7.1.6 这个默认的key方括号中是可选的参数，通过Agent官网文档知道，获取总的内存大小的参数是Tota 49](#_Toc9962)

[7.1.7 如何确认我们的Key添加正确？用Zabbix-get来去值,用zabbix\_get来获取,语句如下 50](#_Toc10577)

[7.1.8 最终添加的结果 50](#_Toc4183)

[7.2 自定义Items key 50](#_Toc7435)

[7.2.1.Key自定义的语法格式 51](#_Toc19528)

[7.2.2 自定义Key中的特殊字符处理 51](#_Toc11888)

[7.2.3 自定义Key的例子 51](#_Toc10382)

[7.2.4.子配置文件的配置 51](#_Toc18403)

[7.2.5 用户自定义参数总结 52](#_Toc12930)

[8.1 宏的配置 52](#_Toc957)

[8.2.1选择Macros下拉菜单， 53](#_Toc23546)

[8.3 模板宏 53](#_Toc12778)

[8.4 主机宏 53](#_Toc3286)

[9.1维护时间 54](#_Toc15307)

[9.2 配置维护时间 54](#_Toc11758)

[9.2 这里的维护时间和Action中的配置紧密相关， 56](#_Toc24558)

[10.1 事件确认 56](#_Toc24678)

[10.2 当事件出现状态为Problem的时候，在菜单栏中单击Monitoring→Events，在其中可以查看事件的详细信息， 56](#_Toc19592)

[10.3 单击图10-2中的Ack列的No，弹出如图10-2所示的界面，输入内容，确认并返回结果。 56](#_Toc27829)

[10.4 看到Ack已经变Yes状态后，说明有用户修改过该状态， 57](#_Toc5859)

[10.5 单击Yes，可以查看具体的文字内容 57](#_Toc3817)

[11.1 数据库导入导出配置 57](#_Toc16156)

[11.2 宏 58](#_Toc12769)

[12.1 Zabbix应用监控系列之TCP状态监控 72](#_Toc10978)

[12.2 配置Agent(标准化目录结构) 72](#_Toc31136)

[12.3 编写Shell脚本 72](#_Toc5908)

[12.4 key的linux\_tcp.conf的子配置文件如下： 73](#_Toc15584)

[12.5 重启zabbix-agent,修改配置文件必须重启 74](#_Toc16242)

[12.6 Server测试Agent是否能获取到值，通过Zabbix\_get(不要直接执行脚本) 74](#_Toc20619)

[12.7 展示所有Key(记得将模板关联主机) 74](#_Toc13504)

[12.8 查看图形(图形是自定义创建) 75](#_Toc23504)

[13.1 Zabbix应用监控系列之Nginx状态监控 75](#_Toc23815)

[13.2 配置所有Agent(标准化目录结构) 75](#_Toc25054)

[13.3 在nginx.conf的Server标签中添加如下内容 76](#_Toc14041)

[13.3.本地访问Nginx Status测试 76](#_Toc22164)

[13.4.编写Nginx的Shell脚本(如果端口不一致,只需要修改脚本端口即可) 77](#_Toc32712)

[13.5.key的nginx\_status.conf的子配置文件如下： 79](#_Toc32175)

[13.6.重启zabbix-agent 80](#_Toc8039)

[13.7.测试一定使用Zabbix\_get来获取值（传入2参数） 80](#_Toc4136)

[14.Zabbix应用监控系列之Redis状态监控 80](#_Toc917)

[14.1.配置所有Agent(标准化目录结构) 80](#_Toc16881)

[14.2.编写Shell脚本 80](#_Toc16354)

[15.1 zabbix自定义发现主机 81](#_Toc21041)

[15.2 Configuration--->Discover--->Create discovery rule 81](#_Toc4425)

[15.3 Configuration--->Action----->Create action 81](#_Toc25253)

[15.4 configuration--->Hosts 82](#_Toc12440)

[16.1.1 Apache安装加载mod\_status模块 82](#_Toc30425)

# 1.1 为何需要监控系统

在一个IT环境中会存在各种各样的设备，例如：硬件设备、软件设备、其系统的构成也是非常复杂的。多种应用构成负载的IT业务系统，保证这些资源的正常运转，是一个公司IT部门的职责。而要让这些应用能够稳定地运行，则需要专业IT人员进行设计、架构、维护和调优。在这个过程中，为了及时掌握基础环境和业务应用系统的可用性，需要获取各个组件的运行状态，如CPU的利用率、系统的复制、服务的运行、端口的连通、带宽流量、网站访问状态码等信息。而这一切都离不开监控系统。

# 1.2 监控系统的实现

一个监控系统的组成大体可以分为两部分：数据采集部分（客户端）和数据存储分析告警展示部分（服务器端），这两部分构成了监控系统的基本模型。

数据采集的工作模式可以分为被动模式（服务端到客户端采集数据）和主动模式（客户端主动上报数据到服务器端）。通常，大多数监控系统应该能同时支持这两种模式。被动模式对服务器的开销较大，适合小规模的监控环境：主动模式对服务器的开销较小，适合大规模的监控环境。

采集数据的协议方式可以分为两种：专用客户端采集和共用协议采集（SNMP、SSH、Telnet等）对于采集到的监控数据，可以将其存储到数据库或者文本或者其他方式，具体采用哪一种，应根据实际需求来决定。

# 1.3 监控系统开源软件

MRTG

MRTG（Multi Route

Trffic Grapher）是一套可用来绘制网络流量图的软件，由瑞士奥尔滕的Tobias Oetiker与Dave Rand所开发，以GPL授权。

MRTG最好的版本是1995年推出的，用perl语言携程，可跨平台使用，数据采集用SNMP协议，MRTG将手机到的数据通过Web页面以GIF或者PNG格式绘制出图像。

Cacti

Cacti（英文含义为仙人掌）是一套基于PHP、MySQL、SNMP和RRDtool开发的网络流量监测图形分析工具，它通过snmpget来获取数据使用RRDtool绘图，但使用者无须了解RRDtool复杂的参数。2提供了非常强大的数据和用户管理功能，可以指定每一个用户能查看树状结构、主机设备以及任何一张图，还可以与LDAP结合进行用户认证，同时也能自定义模板。在历史数据展示监控方面，其功能相当不错。

Cacti通过添加模板，使不同设备的监控添加具有可复用性，并且具备可自定义绘图的功能，具有强大的运算能力（数据的叠加功能）

SmokePing

Smokeping主要用于监视网络性能，包括常规的ping、www服务器性能、DNS查询性能、SSH性能等。底层也是用RRDtool做支持，特点是绘制图非常漂亮，玩过丢包和延迟用颜色和阴影来标示，支持将多张图叠放在一起，其做着还开发了MRTG和RRDtll等工具。

Smokeping的站点为：http://tobi.oetiker.cn/hp

Graphite

Craphite是一个用于采集网站实时信息并进行统计的开源项目。Graphite服务支持平均每分钟4800次更新操作，采用简单文本协议，具有绘图功能，其即插即用的功能可方便地用于任何需要监控的系统上。

和其他监控工具不同额是，Graphite本身并部手机具体的数据，这些数据收集的工作通常由第三方工具或插件完成，可以说，Graphite是一个绘图工具。

Nagios

Nagios是一个企业级的个系统，可监控服务的运行状态和网络信息等，并能监视所指定的本地或远程主机参数以及服务，同时提供异常告警通知功能等。

Nagios可运行在Linux和UNIX平台上。同时提供一个可选的基于刘安琪的Web界面，以方便系统管理人员查看网络状态、各种系统问题、以及日志等

Nagios的功能侧重于监控服务的可用性，能急事根据触发条件告警。

目前Nagios也占领了一定的市场份额，不过Nagios并没有与时俱进，已经不能满足于多变的监控需求，架构的扩展性和使用的便捷性有待增强，其高级功能集成在上夜班Nagios XI中。

Zenoss Core

Zenoss Core（简称Zenoss）是开源企业级IT管理软件，它允许IT管理员依靠歹意的WEB控制台来监控网络架构的状态和健康度。

Zenoss Core的强大功能来自深入的列表与配置管理数据库，用于发现和管理公司IT环境的各类资产（包括服务器、网络和其他结构设备）。Zenoss可以创建关键资产清单和对应的组件级别（接口、服务、进程、已安装的软件等）建立好模型后，Zenoss就可以监控和报告IT架构中各种资源的状态和性能状况了。同时还提供与CMDB关联的事件和错误管理系统，以协助提高各类事件和提醒的管理效率，以此提高IT管理人员的效率。

Ganglia

Grnglia是一个跨平台的、可扩展的、高性能的分布式监控系统，如集群和网格。它基于分层设计，使用广泛的技术，用RRDtool存储数据。具有可视化界面，适合对集群系统的自动化监控。其精心设计的数据结构和算法使得监控端到被监控端的连接开销非常低。目前已经有成千上万的集群正在使用这个监控系统，可以轻松的处理2000个节点的集群环境。

OpenTSDB

开源监控系统OpenTSDB用Hbase存储所有时序（无须采样）的数据，来构建一个分布式、可伸缩的时间序列数据库。它支持秒级数据采集，支持永久存储，可以做容量规划，并很容易地接入到现有的告警系统里。

OpenTSDB可以从大规模的集群（包括集群中的网络设备、操作系统、应用程序）中获取相应的采集指标，并进行存储、索引和服务，从而使这些数据更容易让人理解，如Web化、图形化等。

Zabbix

Zabbix是一个分布式监控系统，支持多种采集方式和采集客户端，有专用的Agent代理，也支持SNMP、IPMI、JMX、Telnet、SSH等多种协议，它将采集到的数据存放到数据库，然后对其进行分析整理，达到条件触发告警。其灵活的扩展性和丰富的功能是其他监控系统所不能比的。相对来说，它的总体功能做的非常优秀。

从以上各种监控系统的对比来看，Zabbix都是具有优势的，其丰富的功能、可扩展的能力、二次开发的能力和简单易用的特点，读者只要稍加学习，即可构建自己的监控系统。

# 2.1 Zabbix简介

随着云计算、虚拟化的大规模应用，以及未来移动互联网、物联网等的兴起，Zabbix的使用将越来越广泛，应用场合也越来越多。目前，不少互联网公司、云计算公司、系统集成软件公司、外包服务公司等，都有对Zabbix进行二次开发和大规模使用。所以，可以断言，Zabbix在未来将会引领监控软件的潮流。

Zabbix适合中小型企业、大中型企业的用户使用。单个Server节点可以支持上万台设备，每秒可以处理1.5万次请求，理论上可以支持5万台设备。

Zabbix自身的定位是中型企业和大型企业，如果在特大型环境中使用，需要解决大并发、大压力的问题，这对使用者提出了更高的要求。

# 2.2 Zabbix概述

Zabbix是一个企业级的高度集成开源监控软件，提供分布式监控解决方案，可以用来监控设备、服务等的可用性和性能，其产品不分企业版和社区版，是一个真正的源代码开放产品，用户可以自由下载并使用该软件。

Zabbix SIA公司是Zabbix的官方技术团队成立的公司。其运作模式是商业软件的开源—软件的使用免费，服务收费。其为用户提供咨询、技术支持服务、定制开发、解决方案、人员培训等。

# 2.3 选择Zabbix七大理由

对于同类监控产品，有以下理由选择使用Zabbix。

1.Zabbix是一个自由开发源代码的产品，用户可以对源代码进行任意修改和二次开发。Zabbix采用GNU General Public License (GPL) Version2开源协议。

2.安装和配置简单，用户仅仅需要一些简单的学习，即可完成监控的搭建工作。

3.搭建环境简单，基于开源软件构建平台，仅需要Linux、Apache/Nginx、MySQL/PostgreSQL/Oracle、PHP即可，无须专用操作系统支持，也无须专用硬件

4.Zabbix-Agent完全支持Linux、UNIX、Windows、AIX、BSD和Solaris的监控，Server和Agent都采用C语言编码，对系统的资源占用非常小，数据采集的性能和速度非常快。

5.将数据采集持久存储到数据库，便于对监控数据的二次分析。

6.非常丰富的扩展能力，很轻松地自定义监控项和实现数据采集，几乎能监控所有的数据。例如：可以监控网站的访问次数，监控UPS和天气温度等。毫不夸张地说，在Zabbix的世界里，往往有想不到的事情，没有办不到的事情。

7.开源社区的运作模式，有各种论坛、邮件列表、IM及时沟通等。

注意：如果是运维人员，想构建一套自己的监控系统环境，Zabbix将会是最佳的选择。如果你是开发 ，想基于开源软件开发一套属于自己的监控系统，Zabbix也是比较好的选择。

# 2.3 Zabbix的功能特性

Zabbix有常见的商业监控软件所具备的功能，如主机的性能监控、网络设备性能监控、数据库性能监控、FTP等通用协议监控、多种告警方式、详细的报表图表绘制、分布式、可扩展能力、API等。

1)数据收集

可用、性能检测。

支持Agent、SNMP（包括Trapping和Polling）、IPMI、JMX、SSH、Telnet等。

自定义的检测。

自定义收集数据的频率。

服务器端/代理段和客户端模式。

2)灵活的触发器

可以定义非常灵活的 阀值和多种相关联的条件。

3)高度可定制的告警

发送通知，可定制包括告警级别、动作升级、收件人和媒体类型。

通知可以使用全局宏变量和自定义的变量。

自动处理功能包括远程命令的自动调用和执行。

4)实时的绘图功能

监控项将数据实施绘制在图形上。

5)Web监控能力

Zabbix可以模拟浏览器请求一个网站，并检查返回值和响应时间。

6)多种可视化的展示

可以自定义监控的展示图，将多种监控数据集中展示到一张图中。

网络拓扑图。

自定义Screens和Slide shows可以将多种图形集中展示。

报表功能。

资源使用情况的监控展示。

7)历史数据的存储

数据存储在数据库中。

历史数据的存放周期可配置。

定期删除过期的历史数据。

8)配置非常容易

配置比较简单，只需要以下两步即可。

第一步：添加设备。

第二步：应用模板即可完成监控。

9)使用模板

模板可以分组

模板具有可继承性。

10)网络发现

支持自动发现网络设备和服务器（可以通过配置自动发现服务规则实现。）

Agent自动发现

支持自动发现实现动态监控的批量监控（支持自定义）内置的自动发现包括文件系统、网络接口、SNMP OLD，可定制自动发现。

11)快速的访问接口

Web页面基于PHP

远程访问。

日志审计。

12)API功能

应用API功能可以方便地和其他系统结合，包括手机客户端的使用。

13)系统权限

不同的用户展示监控的资源不同。

对用户的身份认证

14)程序特性

用C语言编写，其性能和内存开销非常小。

15)大型环境的支持

利用Zabbix-Proxy方式即可轻松构建远程监控。

# 3.1 Zabbix安装与部署

## 3.1.1 Zabbix-Server服务器端的安装

下面以CentOS6.5\_X64为例介绍如何安装Zabbix-Server服务器端。学习采用RPM包安装方式，该方式较简介。生产建议使用源码安装。

## 3.1.2安装Zabbix-Server

rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/6/x86\_64/zabbix-release-2.4-1.el6.noarch.rpm

wget -O/etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-6.repo #安装epel源

yum -y install zabbix zabbix-server zabbix-web zabbix-server-mysql zabbix-web-mysql zabbix-agent zabbix-get mysql-server mysql

#安装服务端

\cp /usr/share/mysql/my-medium.cnf /etc/my.cnf

sed -i "38a collation-server = utf8\_general\_ci\ninit\_connect = \'SET NAMES utf8\'\ncharacter-set-server = utf8" /etc/my.cnf

#修改字符集为utf-8,否则web页面设置中文会出现乱码

/etc/init.d/mysqld start #启动数据库

echo "/etc/init.d/mysqld start" >>/etc/rc.local

#加入开机启动

## 3.1.2.1 创建zabbix的数据库

mysqladmin -uroot password wpaccp;

mysql -uroot –pwpaccp

mysql> create database zabbix character set utf8;

#创建zabbix命名的数据库为utf-8

mysql> grant all on zabbix.\* to zabbix@localhost identified by 'zabbix';

#给zabbix账户赋予zabbix库的所有权限

mysql> flush privileges; #刷新操作

3.1.2.2 导入Zabbix-Server数据库

注意：如果安装Zabbix-Porxy 之导入schema.sql即可,否则Zabbix-Proxy无法正常工作

cd /usr/share/doc/zabbix-server-mysql-2.4.8/create/

mysql -uzabbix -pzabbix zabbix <schema.sql #导入数据结构

mysql -uzabbix -pzabbix zabbix <images.sql #导入图片

mysql -uzabbix -pzabbix zabbix <data.sql #导入数据

[root@centos65-01 create]# cd /etc/httpd/conf.d/

[root@centos65-01 conf.d]# ll

total 16

-rw-r--r--. 1 root root 674 Mar 22 20:29 php.conf

-rw-r--r--. 1 root root 392 Oct 20 2017 README

-rw-r--r--. 1 root root 299 Sep 19 2017 welcome.conf

-rw-r--r--. 1 root root 1182 May 4 2016 zabbix.conf

[root@centos65-01 conf.d]# grep "date.timezone" zabbix.conf

# php\_value date.timezone Europe/Riga

[root@centos65-01 conf.d]# sed -i 's@# php\_value date.timezone Europe/Riga@ php\_value date.timezone Asia/Shanghai@' zabbix.conf

[root@centos65-01 conf.d]# grep "date.timezone" zabbix.conf

php\_value date.timezone Asia/Shanghai

### 3.1.2.3 配置zabbix\_server.conf

[root@centos65-01 zabbix]# grep 'DBHost' zabbix\_server.conf

### Option: DBHost

# DBHost=localhost

[root@centos65-01 zabbix]# sed -i 's@# DBHost=localhost@DBHost=localhost@' zabbix\_server.conf

[root@centos65-01 zabbix]# grep 'DBHost' zabbix\_server.conf ### Option: DBHost

DBHost=localhost

[root@centos65-01 zabbix]# grep "DBName" zabbix\_server.conf

### Option: DBName

# DBName=

DBName=zabbix

[root@centos65-01 zabbix]# grep "DBPassword" zabbix\_server.conf

# For SQLite3 path to database file must be provided. DBUser and DBPassword are ignored.

### Option: DBPassword

# DBPassword=

sed -i '107aDBPassword=zabbix' /etc/zabbix/zabbix\_server.conf #增加zabbix使用数据库的密码

egrep -v '^#|^$' /etc/zabbix/zabbix\_server.conf #过滤修改后的结果

LogFile=/var/log/zabbix/zabbix\_server.log

LogFileSize=0

PidFile=/var/run/zabbix/zabbix\_server.pid

\*DBHost=localhost #可修改

\*DBName=zabbix #默认

\*DBUser=zabbix #默认

\*DBPassword=zabbix #需修改

DBSocket=/var/lib/mysql/mysql.sock

SNMPTrapperFile=/var/log/snmptt/snmptt.log

AlertScriptsPath=/usr/lib/zabbix/alertscripts

ExternalScripts=/usr/lib/zabbix/externalscripts

以上参数值需要关注\*部分，需要根据实际情况进行调整。默认只修改DBPassword=zabbix即可

。

### 3.1.2.4 启动Zabbix-Server服务

/etc/init.d/zabbix-server start #启动

/etc/init.d/httpd start #启动httpd服务

echo "/etc/init.d/zabbix-server start" >>/etc/rc.local #添加开机启动

echo "/etc/init.d/httpd start" >>/etc/rc.local #添加开机启动

### 3.1.2.5 Php.ini配置文件的设置(如果不配置是无法安装)

1)vim /etc/php.ini,修改如下配置选项

post\_max\_size = 16M

max\_execution\_time = 300

max\_input\_time = 300

date.timezone = Asia/Shanghai

2)在LAMP环境中，也可以按上述方式配置PHP的参数，比修改php.ini更方便。而在Zabbix的官方RPM中，这一步已经配置过了。所以无需修改。

cat /etc/httpd/conf.d/zabbix.conf

<Directory "/usr/share/zabbix">

Options

FollowSymLinks

AllowOverride None

Order allow,deny

Allow from all

php\_value max\_execution\_time 300

php\_value memory\_limit 128M

php\_value post\_max\_size 16M

php\_value upload\_max\_filesize 2M

php\_value max\_input\_time 300

php\_value date.timezone Asia/Shanghai

</Directory>

如果在后面配置Web时提示任何参数不满足安装配置要求(如图3-1所示)

，修改对应的参数后重启Httpd即可。

图3-1.jpg

## 3.1.2 配置Web界面

1)打开浏览器，输入http://IP地址/zabbix，会出现如图3-2所示的界面

图3-2.jpg

2)PHP提示的各项参数都满足后，可以继续往下进行，单机Next按钮。如果提示参数不通过，修改php.ini配置文件，并重启Web(httpd)服务。

3)配置数据库连接的各项参数如图3-3所示。测试结果为OK后，单机Next按钮。

图3-3.jpg

4)Zabbix-Server的链接地址、端口、名称设置（见图3-4）将会显示在Zabbix的前端页面。

注意：如果Zabbix-Server在其他机器中，这里的Host填写Zabbix-Server所在的机器IP (如果是本机,可以直接默认)

图3-4.jpg

5)图3-5中的这些配置信息将会写入到图3-6的配置文件中。

图3-5.jpg

图3-6.jpg

6)如图3-7所示的登录界面，默认账户是Admin，密码是zabbix

图3-7.jpg

7)如图3-8登录成功页面。

图3-8.jpg

如果提示不能正常运行，分别检查zabbix\_server.conf中的各项配置文件、Selinux、Iptables等。配置文件请检查以下文件的正确配置参数。

1./etc/zabbix/zabbix\_server.conf中的参数。

#

DBHost=localhost #数据库的IP(域名)地址

DBName=zabbix #数据库的名称

DBUser=zabbix #数据库的用户

DBPassword=zabbix #数据库的密码

2./etc/zabbix/web/zabbix.conf.php中的配置。

cat /etc/zabbix/web/zabbix.conf.php

<?php

// Zabbix GUI configuration file.

global $DB;

$DB['TYPE'] = 'MYSQL'; #数据库类型

$DB['SERVER'] = 'localhost'; #数据库的IP（域名）地址

$DB['PORT'] = '0'; #数据库的端口

$DB['DATABASE'] = 'zabbix'; #数据库的名称

$DB['USER'] = 'zabbix'; #数据库的用户

$DB['PASSWORD'] = 'zabbix'; #数据库的密码

// Schema name. Used for IBM DB2 and PostgreSQL.

$DB['SCHEMA'] = '';

$ZBX\_SERVER = 'localhost'; #Zabbix-Server的IP（域名）地址

$ZBX\_SERVER\_PORT = '10051'; #Zabbix-Server的端口

$ZBX\_SERVER\_NAME = 'Zabbix-Xuliangwei'; #Zabbix-Server web界面的标识

$IMAGE\_FORMAT\_DEFAULT = IMAGE\_FORMAT\_PNG;

?>

Zabbix是支持多语言的，可以为每个用户配置不同的语言环境。单击菜单栏profile-->User-->Language修改用户的语言，(虽然zabbix支持中文,但并不是很推荐使用。只要是中文界面的翻译不准确。所以后续全都使用英文操作)如图3-9所示。

图3-9.jpg

# 3.1.3故障处理

1.数据库文件找不到的问题。

修改/etc/zabbix/zabbix\_server.conf中的Dbsocket配置。

DBSocket=/tmp/mysql.sock

注意：修改为mysql.sock实际存在的路径后，重启服务。

2.数据库无法连接

解决办法有三种：检查数据库服务是否正常；防火墙是否允许；权限能否访问。

## 3.1.4 启动zabbix-server端

[root@centos65-01 ~]# /etc/init.d/zabbix-server start

Starting Zabbix server: [ OK ]

[root@centos65-01 ~]# netstat -tulnp|grep zabbix

tcp 0 0 0.0.0.0:10051 0.0.0.0:\* LISTEN 2172/zabbix\_server

tcp 0 0 :::10051 :::\* LISTEN 2172/zabbix\_server

## 3.1.5 服务器端修改zabbix\_agentd.conf配置文件

[root@centos65-01 ~]# egrep -v '^#|^$' /etc/zabbix/zabbix\_agentd.conf

PidFile=/var/run/zabbix/zabbix\_agentd.pid

LogFile=/var/log/zabbix/zabbix\_agentd.log

LogFileSize=0

Server=127.0.0.1

ServerActive=127.0.0.1

Hostname=Zabbix server

Include=/etc/zabbix/zabbix\_agentd.d/

#修改该文件85行的配置内容

[root@centos65-01 ~]# sed -i 's#Server=127.0.0.1#Server=192.168.100.100#' /etc/zabbix/zabbix\_agentd.conf

[root@centos65-01 ~]# egrep -v '^#|^$' /etc/zabbix/zabbix\_agentd.conf PidFile=/var/run/zabbix/zabbix\_agentd.pid

LogFile=/var/log/zabbix/zabbix\_agentd.log

LogFileSize=0

Server=192.168.100.100

ServerActive=127.0.0.1

Hostname=Zabbix server

# 3.2 Zabbix-Agent客户端的安装

## 3.2.1安装Zabbix-Agent

前面我们已经知道了Zabbix可以再用Agent/Server的架构，客户端的采集方式分为Agent、SNMP等，这里先学习用Agent方式对服务器进行监控，后续再学习使用其他监控方式（SNMP、JMX、IPMI）这里的客户端依然采用RPM方式安装(安装Zabbix官方的yum源)

|  |
| --- |
| rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/6/x86\_64/zabbix-release-2.4-1.el6.noarch.rpm #使用zabbix官方yum源  yum -y install zabbix zabbix-agent #安装zabbix以及agent |

## 3.2.2 配置zabbix\_agentd.conf

主动和被动模式

Server：被动模式，允许哪台服务器连接Agent。

ServerActive：主动模式，向哪台服务器传送数据。

修改配置文件的被动模式Ip

|  |
| --- |
| sed -i's#Server=127.0.0.1#Server=10.0.0.3#g' /etc/zabbix/zabbix\_agentd.conf  或者 sed -i 's@\(Server=127.0.0.1\)@\1,192.168.100.196@' /etc/zabbix/zabbix\_agentd.conf #修改被动模式监听地址为zabbix-server的IP地址  egrep -v '^#|^$' /etc/zabbix/zabbix\_agentd.conf #查看修改后信息  PidFile=/var/run/zabbix/zabbix\_agentd.pid  LogFile=/var/log/zabbix/zabbix\_agentd.log  LogFileSize=0  Server=10.0.0.3 #被动模式，zabbix-server的IP（一个Agent是可以同时向多个服务器端发送数据，多个Ip用逗号隔开）  ServerActive=127.0.0.1 #主动模式  Hostname=Zabbix server  Include=/etc/zabbix/zabbix\_agentd.d/  chkconfig zabbix-agent on #加入开机自启动  /etc/init.d/zabbix-agent start #启动agent |

# 3.3 Zabbix-Get的使用

Zabbix-Get是Zabbix中的一个程序，用于Zabbix-Server到Zabbix-Agent的数据获取，通常可以用来检测验证Agent的配置是否正确。用法如下：

|  |
| --- |
| zabbix\_get  usage: zabbix\_get [-hV] -s <host name or IP>  [-p <port>] [-I <IP address>] -k <key>  -s：远程Zabbix-Agent的IP地址或者是主机名。  -p：远程Zabbix-Agent的端口。  -l：本机出去的IP地址，用于一台机器中又多个网卡的情况。  -k：获取远程Zabbix-Agent数据所使用的Key。 |

示例如下：

zabbix\_get -s 10.0.0.4 -k system.uname

#用zabbix\_get命令可以很方便地知道key是否能正常获取到数据，这在测试自定义监控的时候特别有用。

# 3.4 Zabbix相关术语(命令)

zabbix\_server

zabbix\_server是Zabbix服务端的核心程序。

zabbix\_proxy

zabbix\_proxy是Zabbix代理服务的程序，用于分布式监控proxy模式中。

zabbix\_agent

zabbix\_agent是用超级服务(xinetd)的方式来启动的，对应的配置文件为zabbix\_agent.conf

zabbix\_agentd是以独立进程的方式来启动的，对应的配置文件为zabbix\_agentd.conf

zabbix\_agent是Zabbix专用客户端的程序。

zabbix\_java\_gateway

Zabbix的Java采集服务端，用于JMX的监控方式。

zabbix\_sender

Zabbix的Trapping模式，将采集到的数据通过定时任务等主动发送给zabbix\_server。

zabbix\_get

zabbix\_get是一个数据获取测试命令，相当于snmp中的snmpwalk。

# 4.1自动化监控利器-Zabbix深入配置和使用

# 4.2 配置流程

Host groups（主机组）→Hosts（主机）→Applications（监控项组）→Items（监控项）→Triggers（触发器）→Event（事件）→Actions（处理动作）→User groups（用户组）→Users（用户）→Medias（告警方式）→Audit（日志审计）

# 4.3 添加主机组

分组的目的是将同一属性的主机归类，如果你不想将新添加的主机归纳为自定义的分组，可以使用默认的分组。

配置主机组，打开前端Web页面，单击Configuration------>Host groups 如图2-1所示。



#主机组中包含主机，也可以包含模板。

建议：对同一属性的主机或模板，尽量归纳到分组，方便以后管理。分组原则建议如下。

以地理位置的纬度进行划分。

以业务为单位划分组。

以机器用途划分。

以系统版本划分。

以应用程序花分组。

其他方式等。

按组划分的好处是方便分组管理，如图2-2所示，就是以分组划分的实例

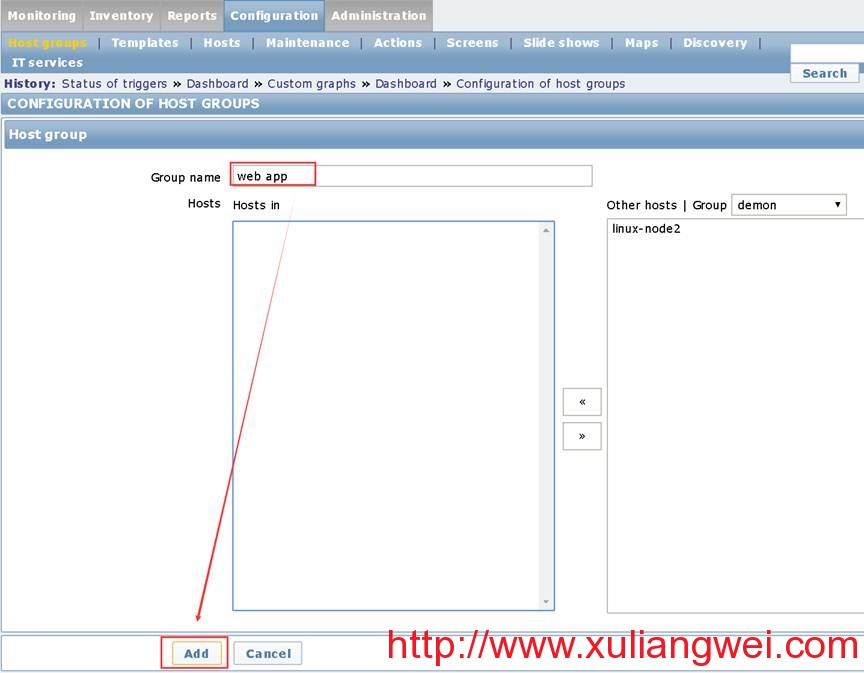


# 4.4 添加新的主机组

1.点击→Host groups上的按钮Create host group



2.单击后会出现一个添加新主机组的界面，如图2-4所示



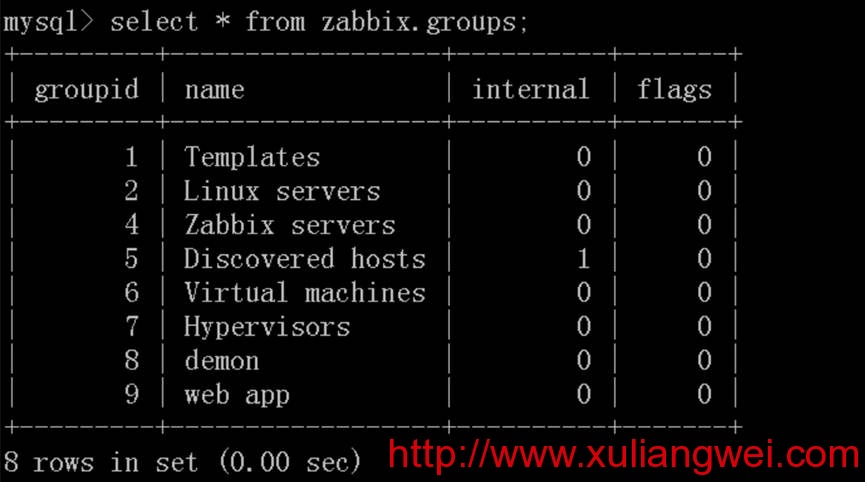
在图2-4中”Group name”是组的名称：”Host Host in”是选择那些设备属于这个新添加的组。

3.单击Save保存，新添加的主机组就可以显示在主机组列表中，如图2-5所示。

图2-5

4.如果想在数据库中查看主机组，运行如下语句即可，如图2-6所示。

mysql> select \* from zabbix.groups; 查询主机组



# 4.5 添加模板

监控项、触发器、图形、Web、Discovery等都是存在于主机之上的，由于多个主机都会用相同的监控配置，因此，可以对这部分同类的数据进行归纳抽象，即将这些数据做成模板。当我们需要对其他监控数据进行配置的时候，只需要对相对应的主句添加对应的模板即可。

## 4.4.1.配置模板的步骤为：

单击→Configuration→Templates---->Create template



## 4.4.2 创建模板

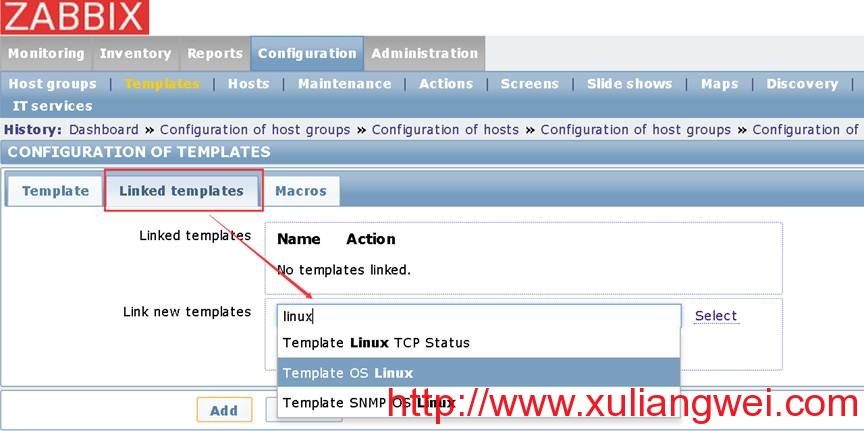
依次单击Templates→Create template,



## 4.4.3 填入模板名称和所属的组，

## 图2-9

## 4.4.4 模板具有继承的功能（将一个模板在另外一个模板中使用），



搜索当前已经存在的模板，然后单击Add添加。

## 4.4.5 模板中也可以设置宏

宏主要是对变量定义，设置宏的作用是方便后面再Items、Trigger中引用，在模板中配置一个宏，在不同的主机对该宏设置不同的变量值，从而达到模板通用的目的。

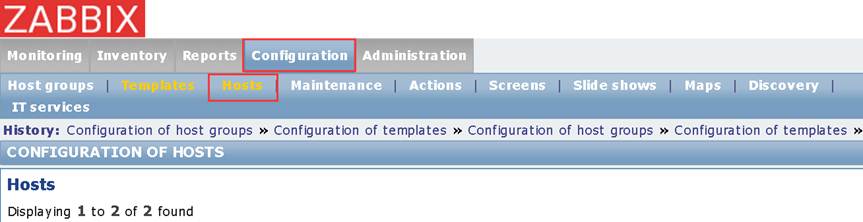
## 4.4.6 查看新建的模板，

图2-11

可以看到，Items中已经有三个Items了，是继承与Template OS Linux模板的.以后会详细讲解,如何在模板中添加其他的Items.

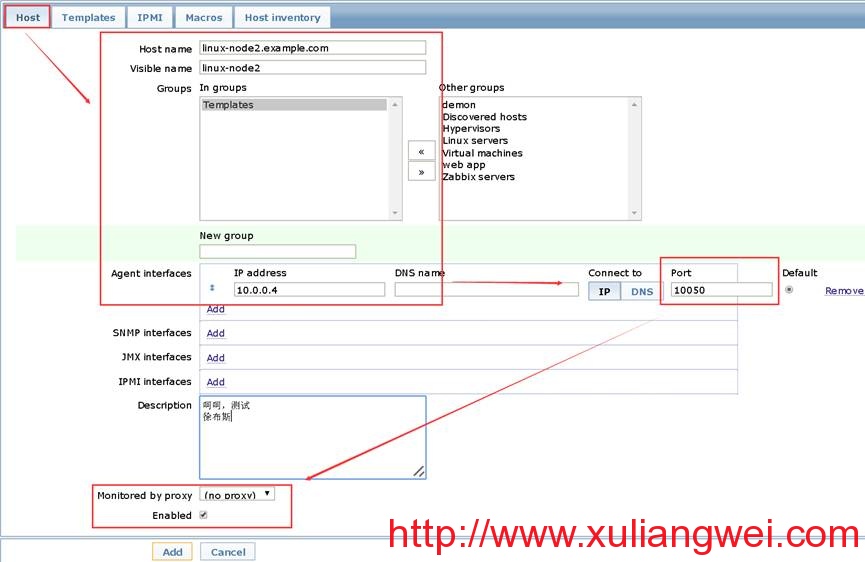
# 4.5 添加主机

添加主机的目的是对具体的设备进行监控，步骤为：依次单击Configuration→Hosts→Create host，如图4-1所示。



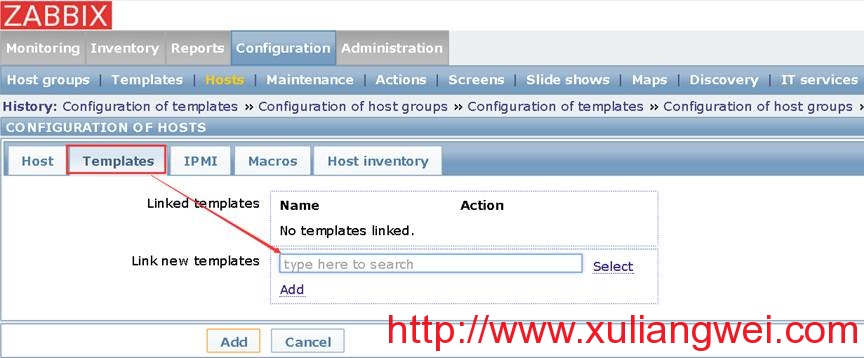
## 4.5.1 在图4-1中单击Create host，

出现如图4-2所示的界面。

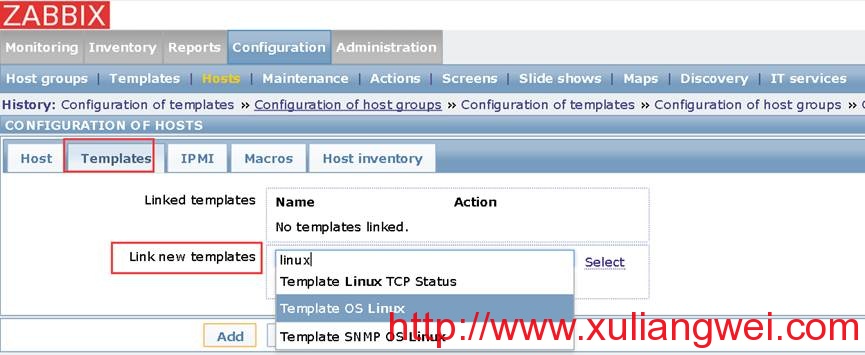


|  |
| --- |
| 参数 描述  主机名(Host name) 输入一个不重复的主机名。编辑该名称对应客户端的配置文件时，主机名这一项必须与此处输入的值相同。  访问名(Visible name) 如果设置该名字，那么它将出现在主机列表、地图等地方。  群组(Group in groups) 选择主机所属的群组。一个主机必须属于至少一个主机组。  新建主机组(New group) 一个新的群组将被 ,然后自动连接到该主机上。如果为空，该项将被忽略。  接口协议(Agent interfaces) 一个主机支持的主机接口协议包括：  Agent、SNMP、JMX、IPMI，如果想增加一个新的接口，单击”Add”按钮，然后输入IP/DNS、连接项、端口等信息。  IP地址(IP address) 要监控主机的IP地址(可选项、默认是IP)  DNS名称(DNS name) 要监控主机DNS能扣解析的名称(可选项)与Agent通信方式(Connect to)  单击对应名称的按钮 IP 连接要监控主机的IP地址(推荐)  DNS 要监控主机能够正常解析的DNS  端口(Port) TCP协议的端口，Zabbix客户端使用的默认值是10050通过代理服务器进行监控  (Monred by proxy) 主机可以通过Zabbix服务器或者  Zabbix的一个代理去监控客户端  状态(enabled) 勾选则开启 |

## 4.5.2 选择模板，如图4-3所示。



## 4.5.3 搜索模板，如图4-4所示。



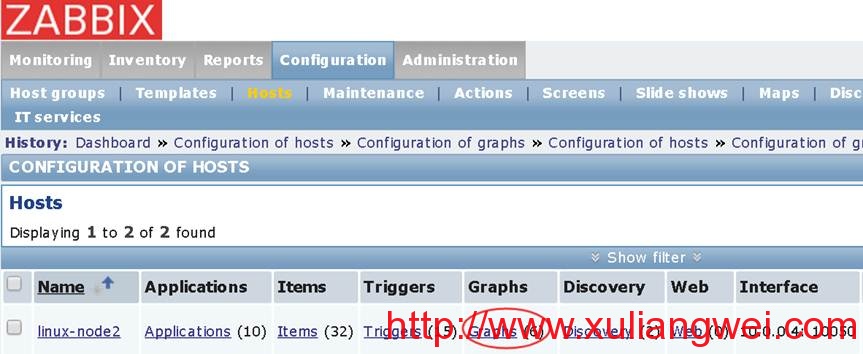
## 4.5.4 搜索出来后，打击Add添加该模板，

## 图2-164.5.5 添加了两个模板，单击Add保存，

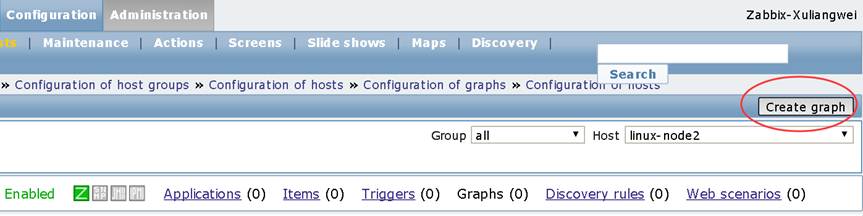
# 图2-175.1 Graphs配置

Graphs是将数据展示为图像，以视觉化形式展示，Graphs的配置存在于主机和模板中

## 5.1.1 在所在的主机模板中，选择Graphs，如图5-1所示。



## 5.1.2 单击“Create graph”按钮，如图5-2所示。



## 5.1.3 出现新建图形的界面，输入Name，选择所需要的Items，

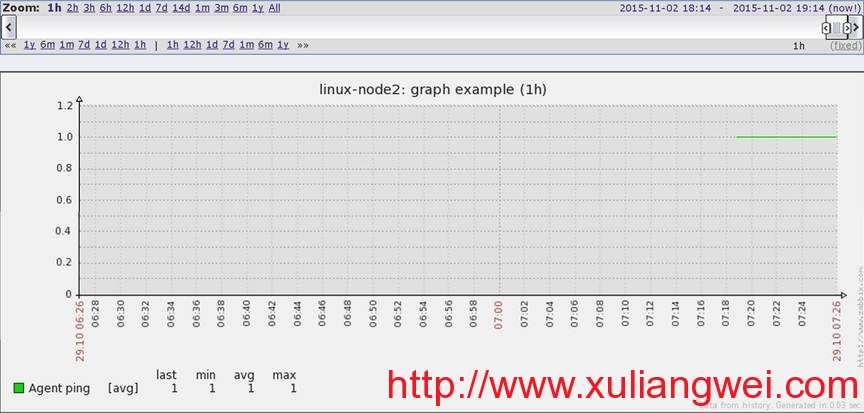
## 图5-35.1.4 单击Graphs，可以看到其中已经添加了一个Graphs，

图5-45.1.5 单击刚才创建的Graphs，选择preview，即可查看图形，

## 图2-185.1.6 如果要查看所有的图形，单击Monitoring→Graphs，

## 图5-55.1.7 选择对应的Group、Host、Graph，

## 图5-65.1.8 显示图形如图5-8所示

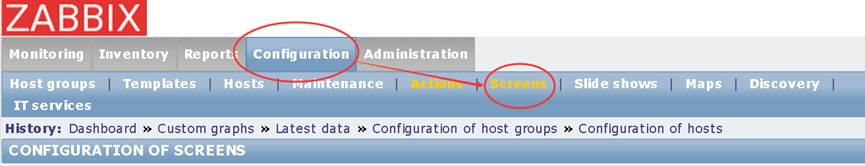


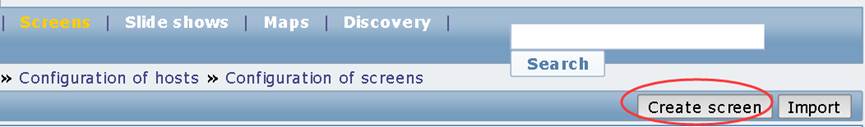
# 6.1 Screen配置

Screen将多种信息放在一起展示，便于集中展示某个Host的多个信息，或是将多个Host的同一种信息放在一起显示，这些信息可以为Graphs、Maps、Serverinfo等，几乎涵盖Zabbix所有的监控信息。

1.通过单击Configuration-→Screens→Create screen来创建，

创建时定义Screen的行数和列数(如图6-1至图6-3所示)单击单元格内的Change，添加相应的元素。



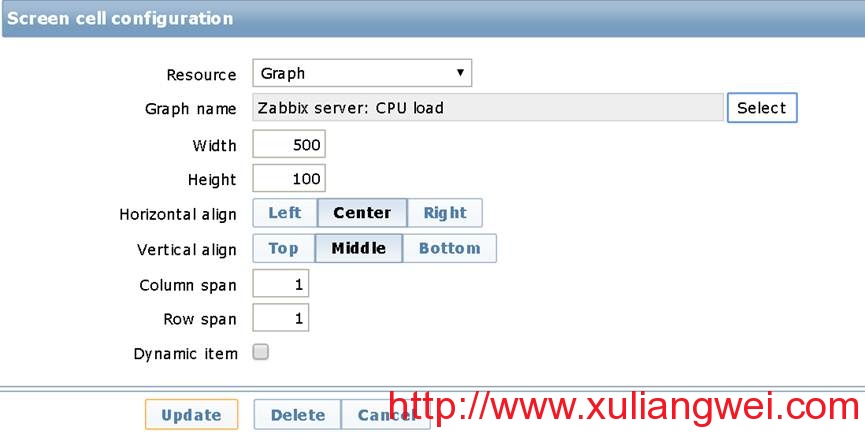




# 6.2 单击创建Screen名称，

# 图6-46.3 单击Change，

## 图6-56.4 出现如图6-6所示



# 6.5 选择对应的Graphs，这里选择选择cpuload图形，

# 图6-76.6 单击update保存，

# 图6-96.7 配置好后，通过单击Monitor→Screen，选择相应的Screen组，

# 图5-97.1 添加Items

## 7.1.1 Items的含义

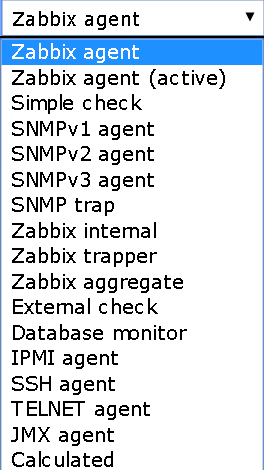
Items就是监控项可以配置获取监控数据的方式、取值的数据类型、获取数值的间隔、历史数据保留时间、趋势数据保存时间、监控Key的分组等。

检测周期的长短直接影响了数据的获取，也影响了ZabbixServer的性能。

监控的方式非常多，Zabbix支持的监控方式如图7-1所示。

监控项存在于Zabbix.items表中。

mysql> select \* from zabbix.items;



## 7.1.2 自定义Items

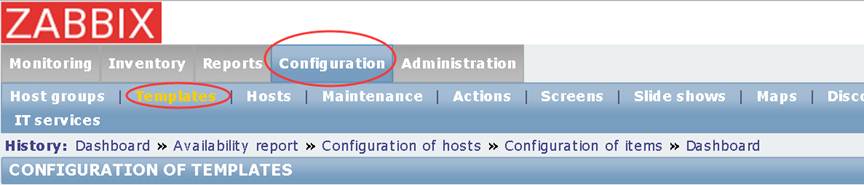
Items可以存在于模板（Template）中，也可以存在于主机（Host）中，模板的作用是可以复用，对需要重复配置的监控项归类。

下面演示如何在Template OS

Linux模板中添加一个Items,Key为vm.memory.size[total]

Template OS

Linux 模板中默认不存在统计总内存的大小的Items，故我们需要自己添加，添加方法：单击Configuration→Templates，



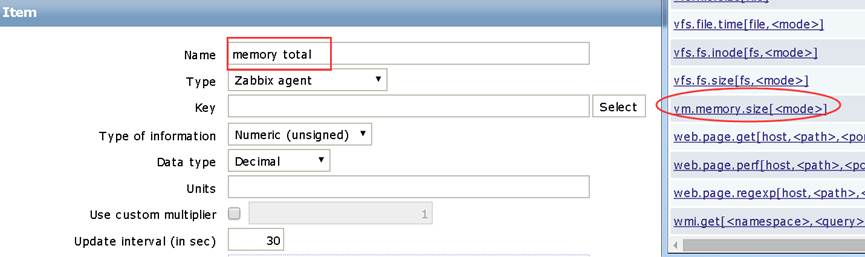
## 7.1.3 找到Template OS Linux,单击Items

## 图7-37.1.4 单击Create items创建Item，

## 图7-47.1.5 单击Select按钮，列出Agent所有自带的Key，

如图7-5所示。

由于vm.memory.size[]这个key是Zabbix自带的，所以会列出来。这里需要注意，如果是自己定义的Key，单击Select送无法看到的。自己定义的key在这里手动添加即可



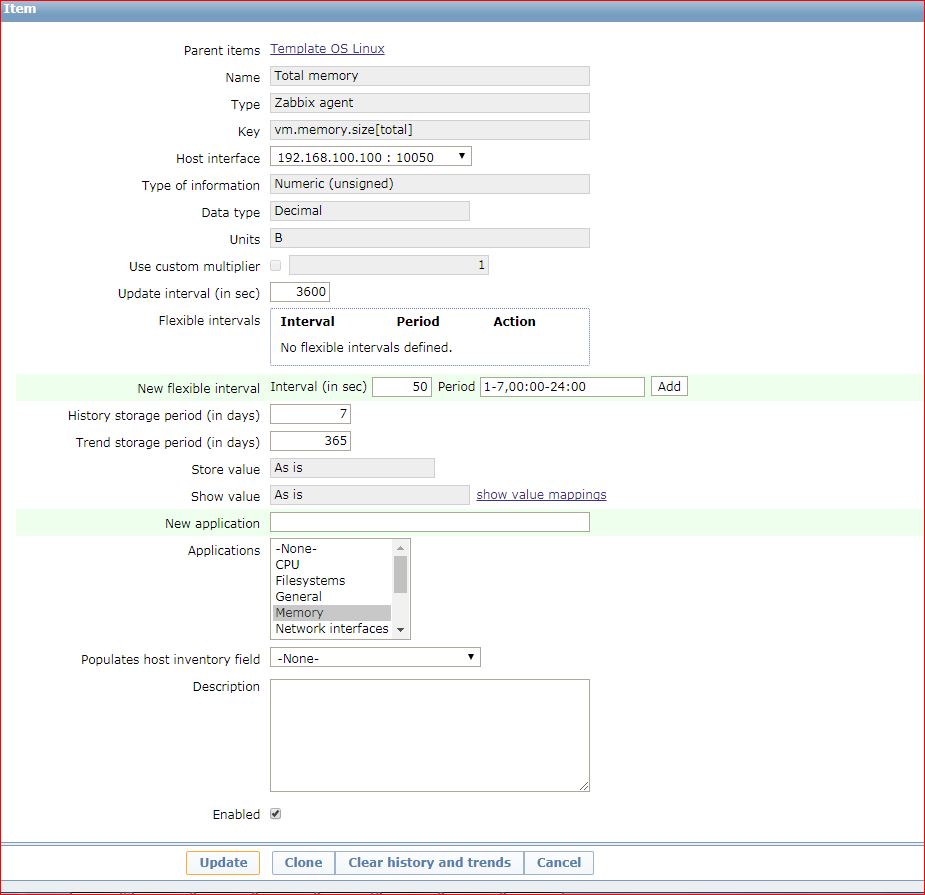
## 7.1.6 这个默认的key方括号中是可选的参数，通过Agent官网文档知道，获取总的内存大小的参数是Total

## 图7-67.1.7 如何确认我们的Key添加正确？用Zabbix-get来去值,用zabbix\_get来获取,语句如下

zabbix\_get -s 10.0.0.4 -k vm.memory.size[total]

确保能获取到的值，就说明Key配置是正确的。提示：zabbix\_get仅能测试获取Agent监控方式的Key值，不能获取其他类型的数据。

# 7.1.8 最终添加的结果



# 7.2 自定义Items key

用户自定义参数：用户自定义参数（UserParameter）仅支持Agent方式，对于其他方式，它是不支持的。

## 7.2.1.Key自定义的语法格式

在/etc/zabbix/zabbix\_agentd.conf中配置参数，写法如下。

UserParameter=key,command

除了上面这种写法，还支持参数传递的写法，具体如下。

UserParameter=key[\*],command $1 $2 $3 ….

## 7.2.2 自定义Key中的特殊字符处理

如果Userparameter包含\’”`\*？[]{}~$!&;()<>|#@这些字符，默认情况下，Zabbix对这些参数是无法正常处理的，需要在Zabbix\_agentd.conf中开启参数UnsafeUserParameters，并将其值设置为1，语句如下。

vim /etc/zabbix/zabbix\_agentd.conf

UnsafeUserParameters=1

## 7.2.3 自定义Key的例子

UserParameter=login-user,uptime|awk -F ' ' '{print $4}'

然后重启zabbix\_agent 服务（注意：修改配置文件必须重启。）

zabbix\_get -s 10.0.0.4 -k login-user #运行测试命令，查看下Key

2

## 7.2.4.子配置文件的配置

为了便于维护和分类管理，UserParameter的内容可以单独写一个配置文件。

vim /etc/zabbix/zabbix\_agentd.conf

Include=/etc/zabbix/zabbix\_agentd.d/

Include是zabbix\_agentd.conf中部分内容的子配置文件，注意：zabbix\_agentd.conf.d/文件下所有的配置文件都会被当做子配置文件，如果有重复，会导致客户端Agent启动失败。

## 7.2.5 用户自定义参数总结

自定义Key的语法。

特殊字符的处理。

子配置文件的注意事项。

注意：

1.用户自定义参数一定要包含一个key和一个命令，也可以是一个脚本

2.key在整个系统是唯一的

3.用户自定义参数返回值，最大返回512兆

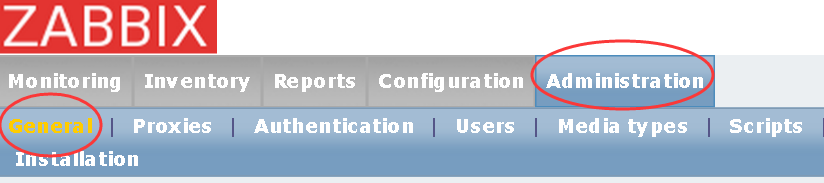
4.自定义完后，重启服务测试Key

# 8.1 宏的配置

宏的作用是便于在模板、Items、Trigger中的引用。宏的名称为{$名称}，宏的字符范围为A~Z、0-9、\_

8.2 作用范围为：模板、主机。配置步骤为：单击Administration→General→Macros,

如图8-1所示。

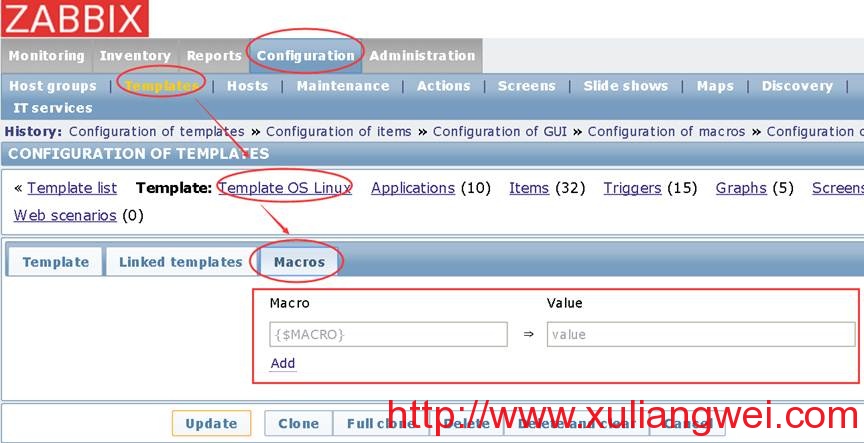


## 8.2.1选择Macros下拉菜单，

# 图8-28.3 模板宏

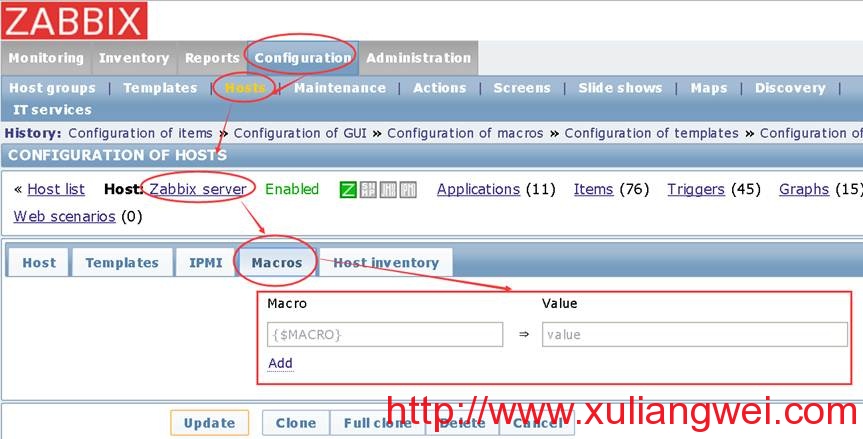
作用范围：当前模板。

配置步骤为：单击Configuration→Templates,再单击模板名称(图中为Templates OS Linux)最后单击Macros，如图8-3所示。



# 8.4 主机宏

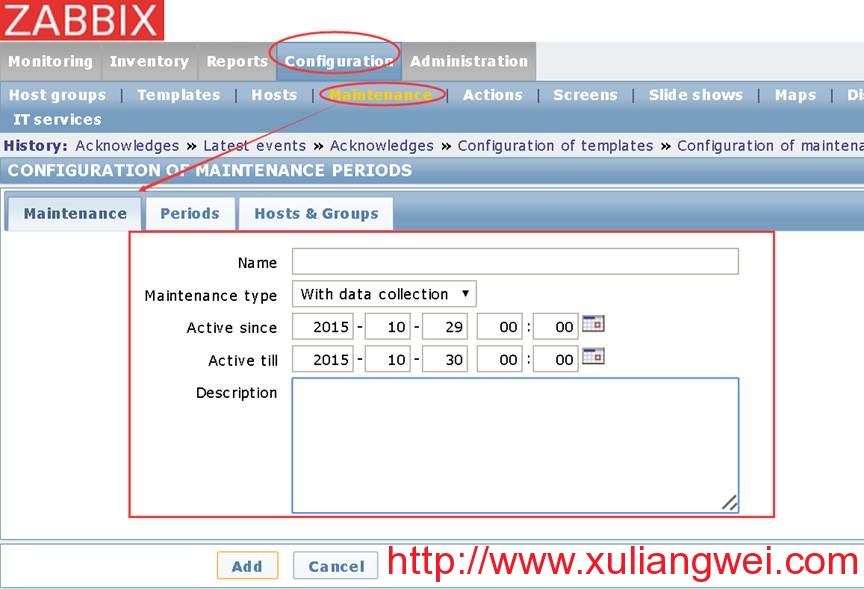
作用范围为：当前主机。配置步骤为：Configuration→Hosts→Zabbix server→Macros，如图8-4所示。



# 9.1维护时间

在某些场合中，我们不需要进行告警，例如：业务的正常维护，所以维护时间就显得特别有用.

# 9.2 配置维护时间



# 9.2 这里的维护时间和Action中的配置紧密相关，

# 图9-110.1 事件确认

# 10.2 当事件出现状态为Problem的时候，在菜单栏中单击Monitoring→Events，在其中可以查看事件的详细信息，

# 图10-110.3 单击图10-2中的Ack列的No，弹出如图10-2所示的界面，输入内容，确认并返回结果。



# 10.4 看到Ack已经变Yes状态后，说明有用户修改过该状态，

图10-210.5 单击Yes，可以查看具体的文字内容，

# 图10-311.1 数据库导入导出配置

Zabbix提供将所有的配置导出为标准的XML格式的文件，同样，也支持导入标准格式的XML配置文件。

导入/导出功能可以把之前做的功能进行模板化，与Cacti模板的原理相同。

单击Configuration→Templates/Host→Export/Import→Export,勾选要导出的模板（设备），选择Export，即可导出XML文件到本地。

# 11.2 宏

Zabbix宏是一种抽象(Abstraction),它根据一系列预定义的规则替换一定得文本模式，而解释器或编译器在遇到宏时会自动进行这一模式替换；

Zabbix基于宏保存预设文本模式，并且在调用时将其替换为其中的文本；

Zabbix有许多内置的宏，如{HOST.NAME}、{HOST.IP}、{TRIGGER.DESCRIPTION}、{TRIGGER.NAME}、

{TRIGGER.EVENTS.ACK}等

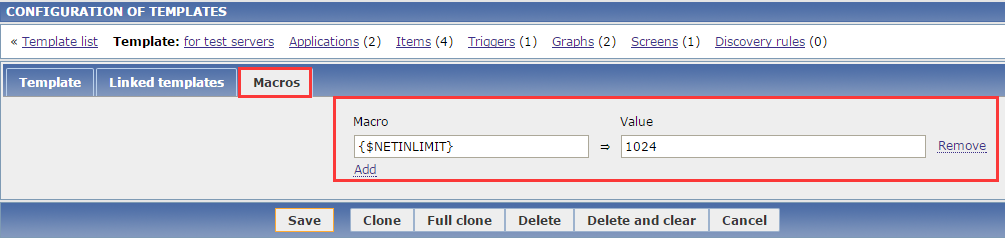
Zabbix为了更强的灵活性，zabbix还支持在全局、模板或主机级别使用用户自定义宏(user macro);

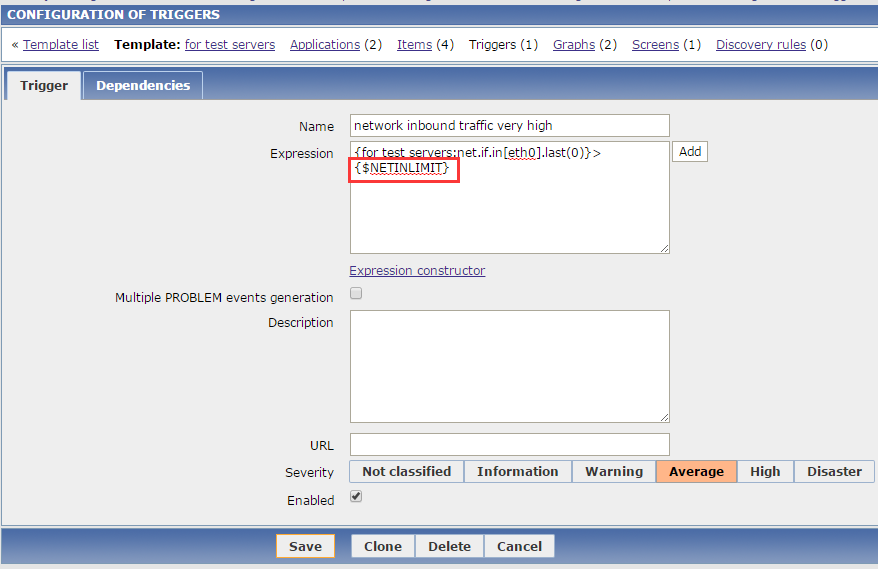
用户自定义宏要使用"{$MACRO}"这种特殊的语法格式；宏可以应用在item keys和descriptions、trigger名称和表达式、主机接口IP/DNS及端口、discovery机制的SNMP协议的相关信息中等；宏名称只能使用大写字母、数字及下划线

Zabbix宏的替换次序:首先是主机级别宏；其次是当前主机上一级模板中(直接链接至主机的模板)的宏，多个一级模板按其ID号排序；再接着是二级模板中的宏；而后依次类推；最后检查的是全局宏；zabbix如果无法查找到某主机定义使用的宏，则不会对其进行替换操作。要使用用户自定义宏，有以下两种途径:

Macros使用示例:

-->在主机级别定义一个名为{$NETINLIMIT}的宏，以定义当前主机所接受的入站流量的合理次数:



 用户自定义监控:

为什么要自定义KEY:

有时候我们想让被监控端执行一个zabbix没有预定义的检测，zabbix的用户自定义参数功能提供了这个方法。我们可以在客户端配置文件zabbix\_angentd.conf里面配置UserParameter.

语法如下:

UserParameter=key,command

用户自定义参数包含一个key和一个命令，key必须整个系统唯一，配置好之后，重启客户端。然后配置item,在key的位置填上我们自定义的key即可。

用户自定义参数里指定的脚本由zabbix agent来执行，最大可以返回512KB的数据.

用户自定义KEY实例:

简单点的命令示例：

UserParameter=ping,echo 1

如果调用ping这个key，将会收到返回值1.

更复杂的命令示例:

UserParameter=mysql.ping,mysqladmin -uroot ping|grep -c alive

如果返回1表示MySQL运行中，如果返回0表示MySQL挂了

灵活的自定义key：

如下为灵活的用户自定义参数:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| UserParameter=key[\*],command   |  |  | | --- | --- | | 参数 | 描述 | | Key唯一. | [\*]表示里面可以传递多个参数  需要执行的脚本，key的[]  里面的参数一一对应$1到$9，一共9个参数。$0表示脚本命令. | | Command | |

注意事项

1、如果需要使用命令行里面出现$2这种变量，那么你要使用两个$$2，例如awk ’{ print $$2 }’，之前就遇到过这个问题，不停的测试自己脚本输出正常，但是zabbix却拿不到数据，原来是出在这里。为了防止和参数冲突，所以zabbix做了这个规定。

2、zabbix禁止使用一些不安全的参数，如下:

\ ' ” ` \* ? [ ] { } ~ $ ! & ; ( ) < > | # @

3、从zabbix 2.0开始，zabbix返回文本数据可以是空格。

示例:获取内存空间大小

(1)配置被监控端agent配置文件如下:

# vim /etc/zabbix/zabbix\_agentd.conf

UserParameter=memory.free,/usr/bin/free |awk '/^Mem:/{print $4}'

重启被监控agent服务

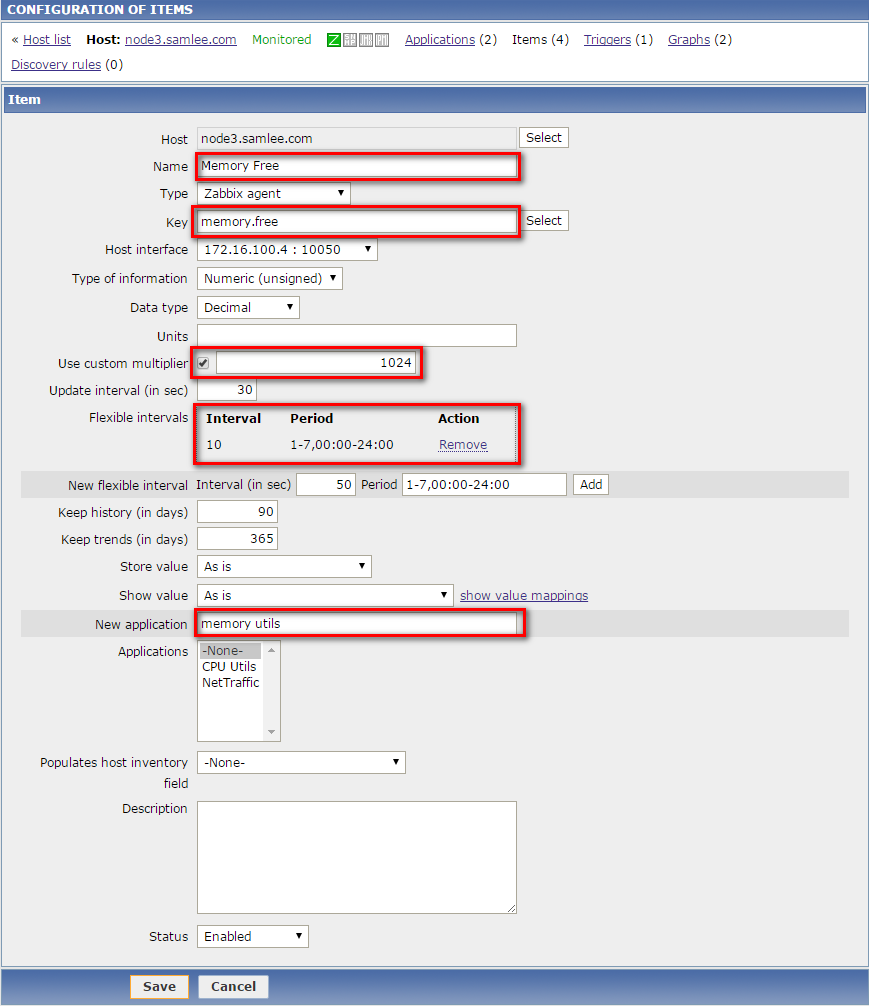
# service zabbix-agent restart

(2)在服务器使用shell方式获取被监控端自定义监控项数据

#zabbix\_get -s 172.16.100.4 -k "memory.free"

297712

(3)添加为被监控对象



示例:自定义key传递多个参数:监控主机内存信息

(1)配置被监控端agent配置文件如下:

# vim /etc/zabbix/zabbix\_agentd.conf

UserParameter=memory.usage[\*],/bin/cat /proc/meminfo | awk '/^$1:/{print $$2}'

重启被监控agent服务

# service zabbix-agent restart

(2)在服务器使用shell方式获取被监控端自定义监控项数据

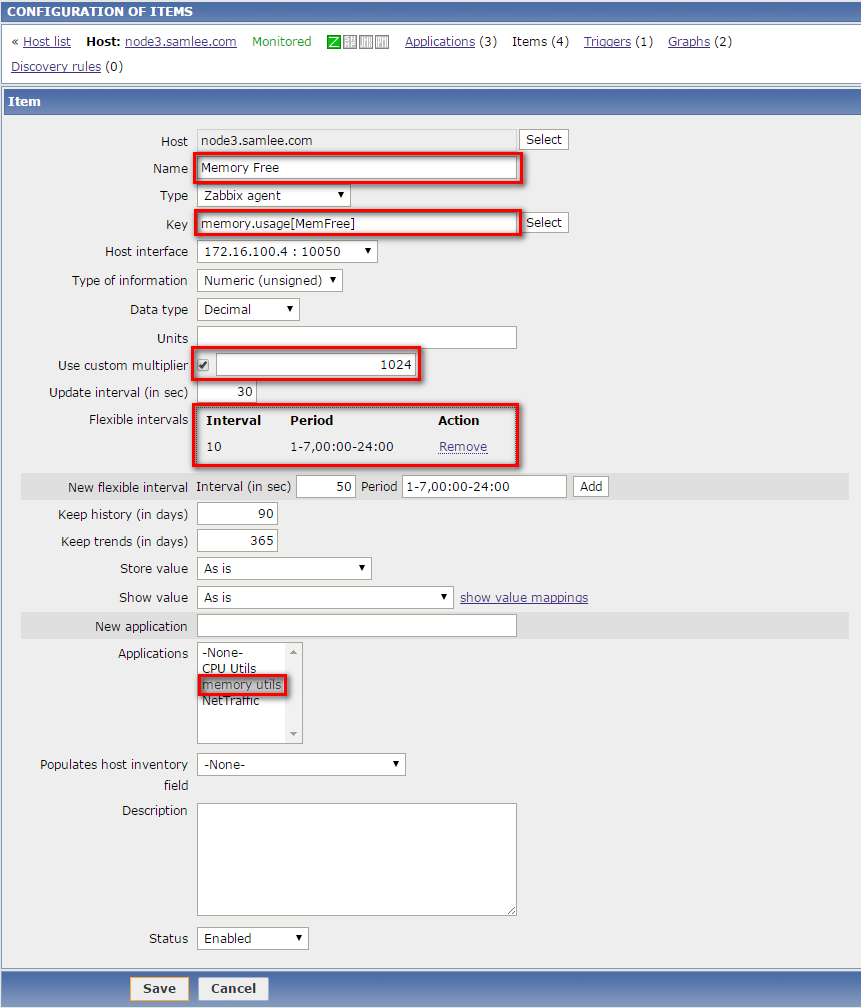
# zabbix\_get -s 172.16.100.4 -k "memory.usage[Buffers]"

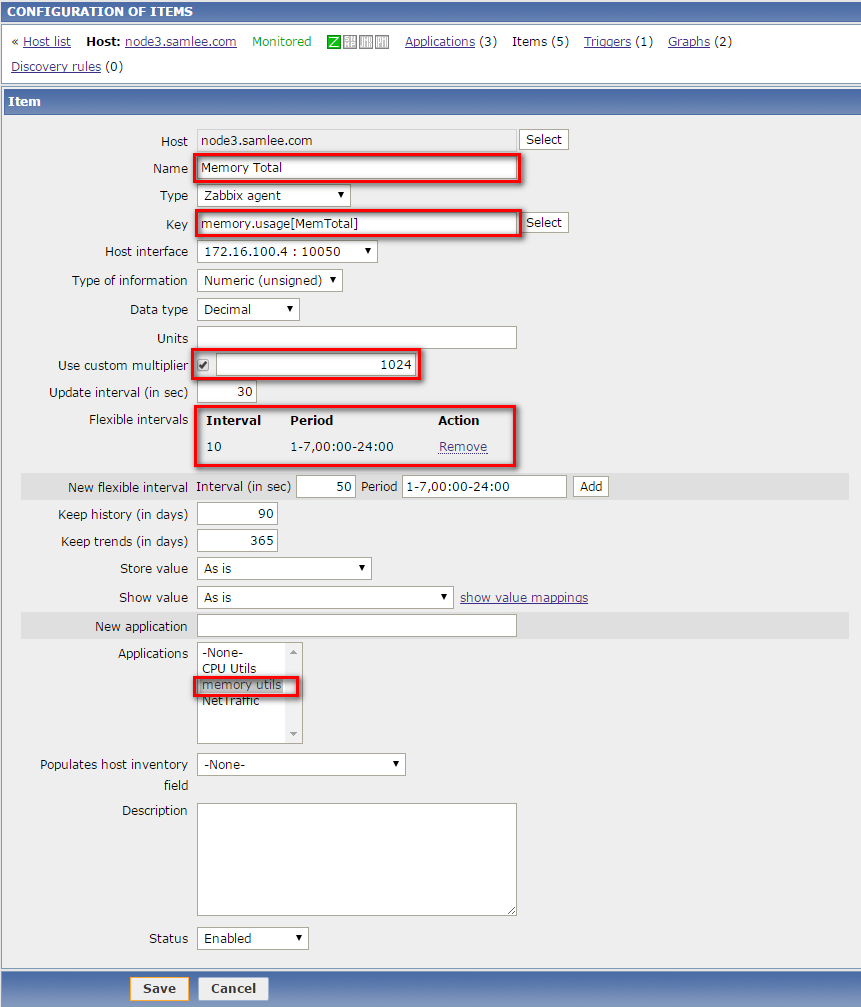
55980

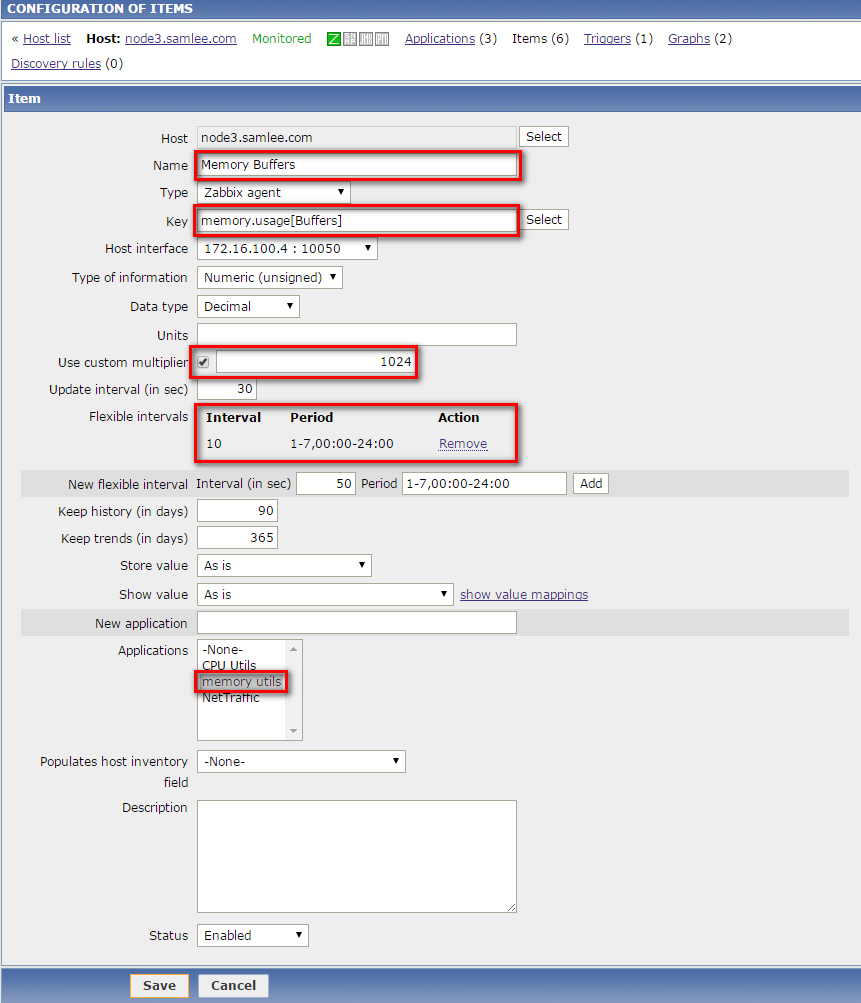
# zabbix\_get -s 172.16.100.4 -k "memory.usage[MemTotal]"

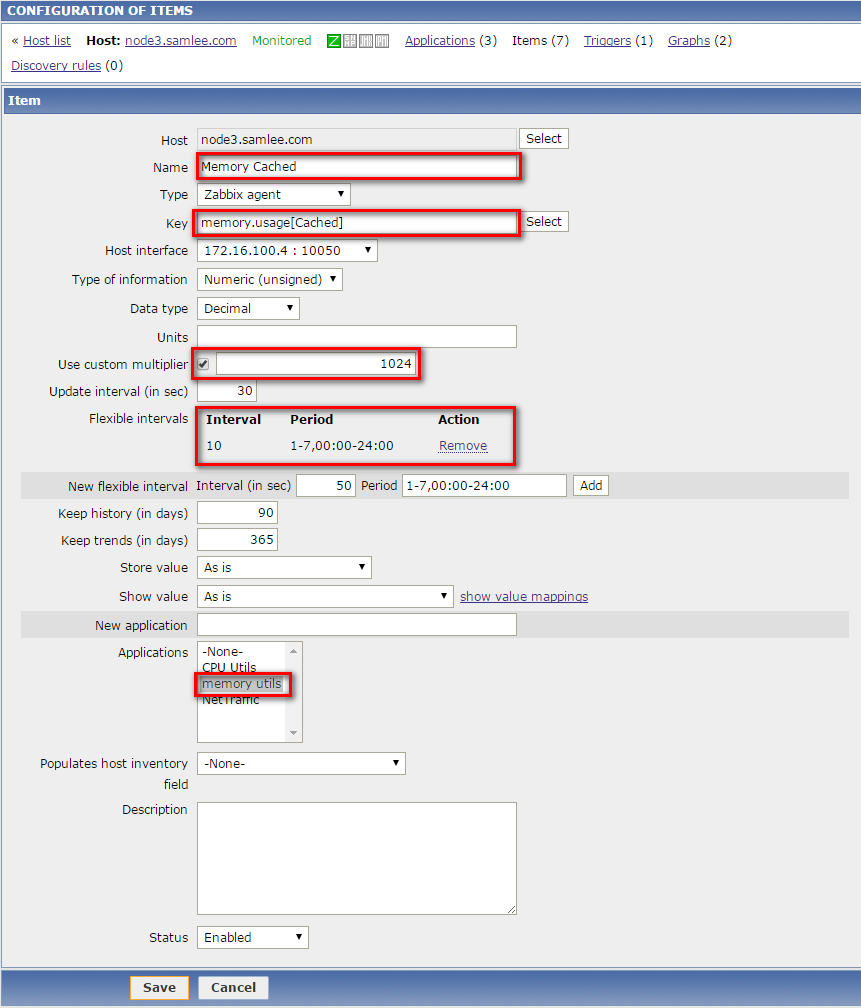
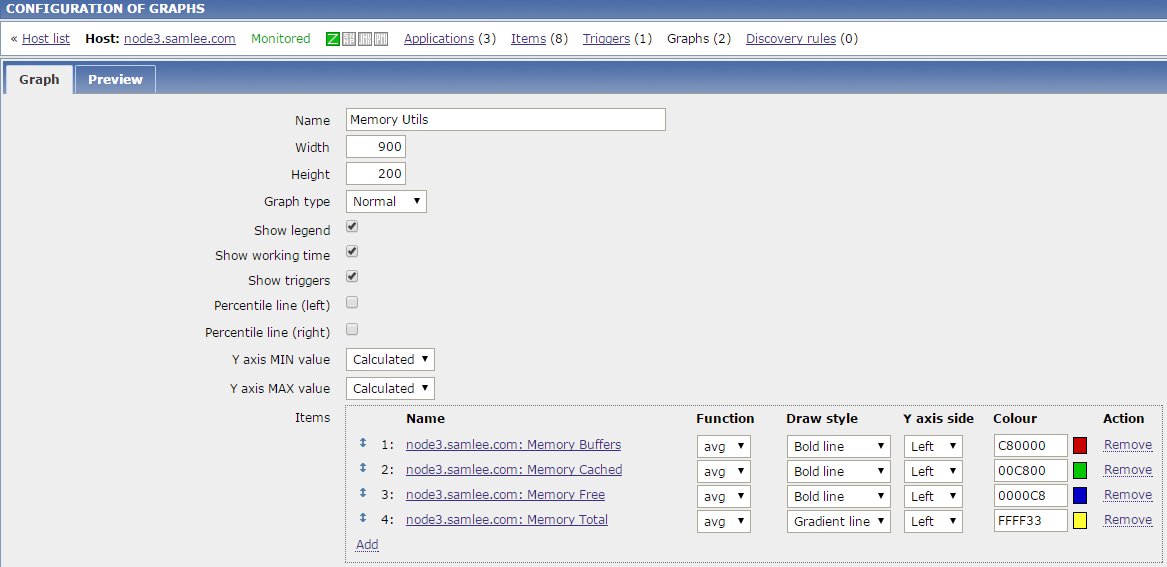
494224

(3)添加为被监控对象







实战示例:使用自定义监控nginx服务

(1)被监控端nginx服务开启status

# vim /etc/nginx/conf.d/default.conf

location /status {

stub\_status on;

access\_log off;

allow 172.16.0.0/16;

deny all;

}

重启nginx服务测试:

# service nginx restart

# curl http://172.16.100.4/status

Active connections: 1

server accepts handled requests

2 2 2

Reading: 0 Writing: 1 Waiting: 0

(2)配置被监控端agent配置文件如下:

# vim /etc/zabbix/zabbix\_agentd.d/nginx.conf

UserParameter=Nginx.active[\*],/usr/bin/curl -s "http://$1:$2/status" | awk '/^Active/ {print $NF}'

UserParameter=Nginx.reading[\*],/usr/bin/curl -s "http://$1:$2/status" | grep 'Reading' | cut -d" " -f2

UserParameter=Nginx.writing[\*],/usr/bin/curl -s "http://$1:$2/status" | grep 'Writing' | cut -d" " -f4

UserParameter=Nginx.waiting[\*],/usr/bin/curl -s "http://$1:$2/status" | grep 'Waiting' | cut -d" " -f6

UserParameter=Nginx.accepted[\*],/usr/bin/curl -s "http://$1:$2/status" | awk '/^[ \t]+[0-9]+[ \t]+[0-9]+[ \t]+[0-9]+/ {print $$1}'

UserParameter=Nginx.handled[\*],/usr/bin/curl -s "http://$1:$2/status" | awk '/^[ \t]+[0-9]+[ \t]+[0-9]+[ \t]+[0-9]+/ {print $$2}'

UserParameter=Nginx.requests[\*],/usr/bin/curl -s "http://$1:$2/status" | awk '/^[ \t]+[0-9]+[ \t]+[0-9]+[ \t]+[0-9]+/ {print $$3}'

重启被监控agent服务

# service zabbix-agent restart

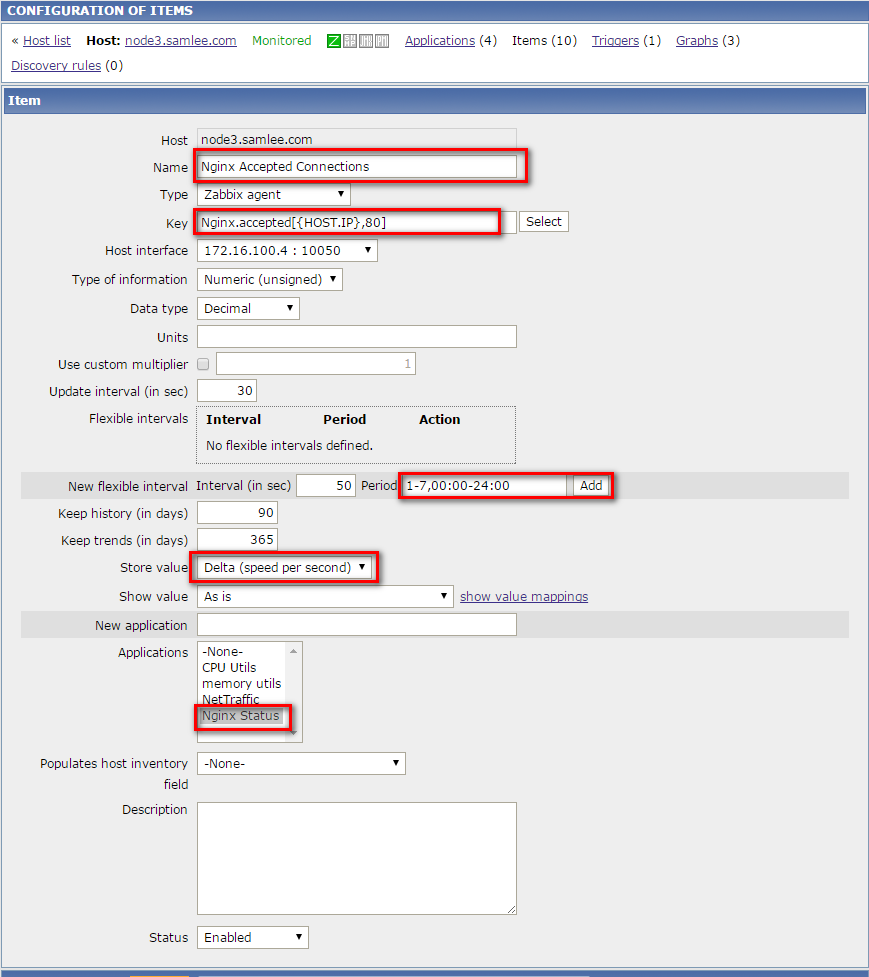
(3)在服务器使用shell方式获取被监控端自定义监控项数据

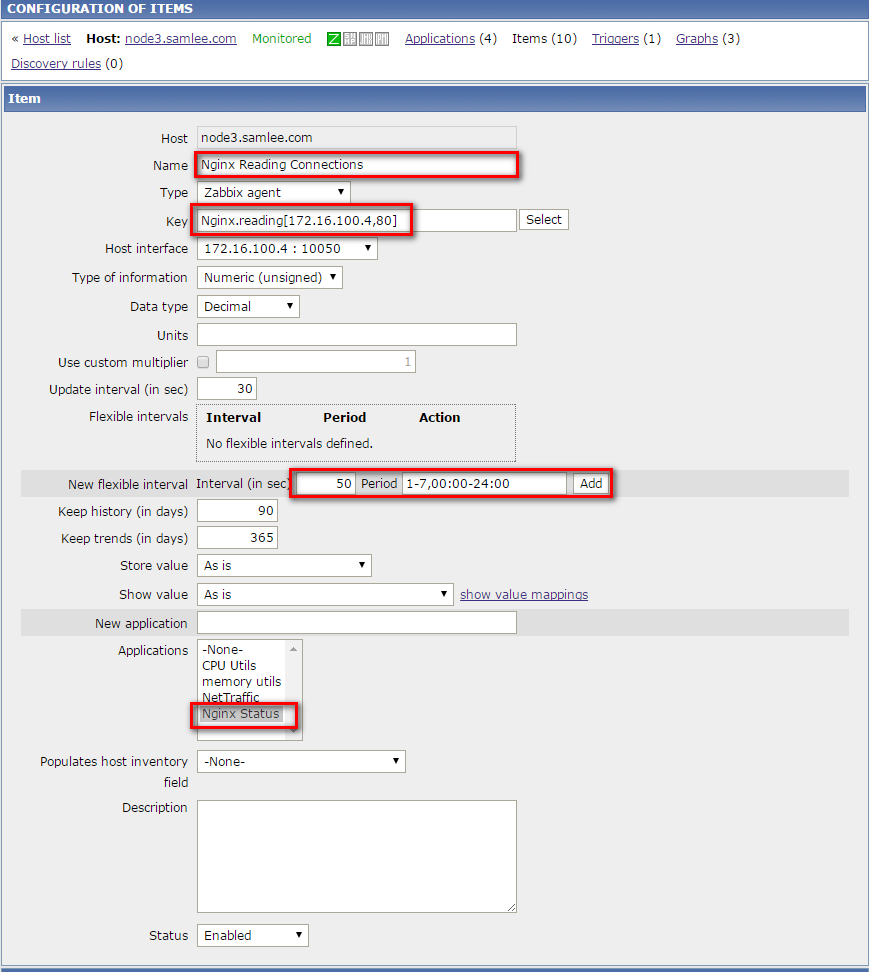
# zabbix\_get -s 172.16.100.4 -k "Nginx.accepted[172.16.100.4,80]"

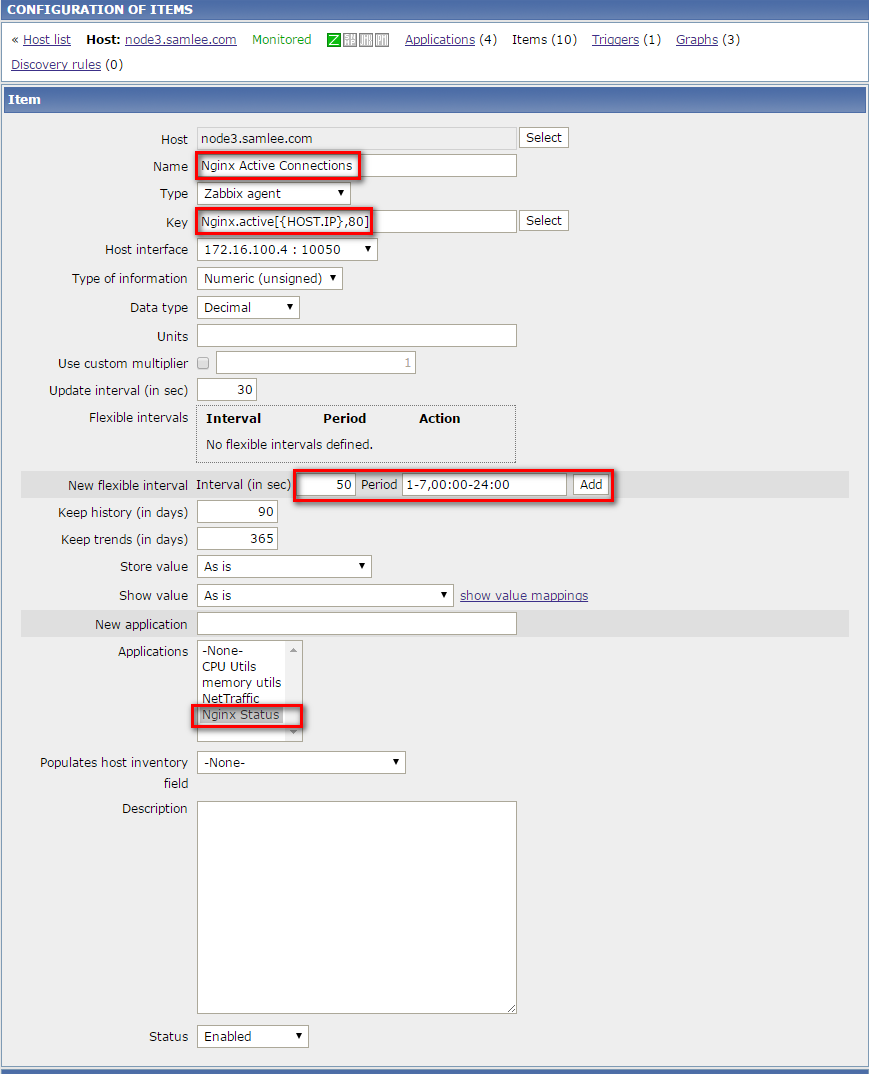
# ab -n 1000 -c 100

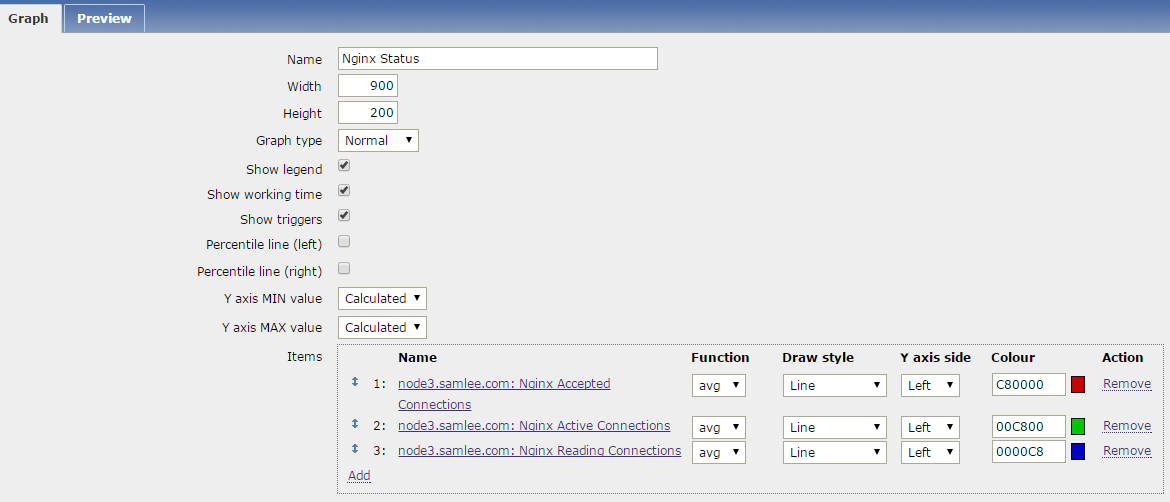
# zabbix\_get -s 172.16.100.4 -k "Nginx.accepted[172.16.100.4,80]"

(4)添加为被监控对象









# 12.1 Zabbix应用监控系列之TCP状态监控

TCP监控

Tcp的连接状态对于我们web服务器来说是至关重要的，尤其是并发量ESTAB；或者是syn\_recv值，假如这个值比较大的话我们可以认为是不是受到了攻击，或是是time\_wait值比较高的话，我们要考虑看我们内核是否需要调优，太高的time\_wait值的话会占用太多端口，要是端口少的话后果不堪设想：所以今天我们来学学如何使用Zabbix监控tcp状态

# 12.2 配置Agent(标准化目录结构)

#vim /etc/zabbix/zabbix\_agentd.conf #编辑配置文件引用key

Include=/etc/zabbix/zabbix\_agentd.d/\*.conf

#mkdir /etc/zabbix/scripts #存放Shell脚本

# 12.3 编写Shell脚本

# cd /etc/zabbix/scripts

# vim tcp\_status.sh

#!/bin/bash

############################################################

# $Name: tcp\_status.sh

# $Version: v1.0

# $Function: TCP Status

# $Author: xuliangwei

# $organization: www.xuliangwei.com

# $Create Date: 2016-06-23

# $Description: Monitor TCP Service Status

############################################################

[ $# -ne 1 ] && echo "Usage:CLOSE-WAIT|CLOSED|CLOSING|ESTAB|FIN-WAIT-1|FIN-WAIT-2|LAST-ACK|LISTEN|SYN-RECV SYN-SENT|TIME-WAIT" && exit 1

tcp\_status\_fun(){

TCP\_STAT=$1

ss -ant | awk 'NR>1 {++s[$1]} END {for(k in s) print k,s[k]}' > /tmp/ss.txt

TCP\_STAT\_VALUE=$(grep "$TCP\_STAT" /tmp/ss.txt | cut -d ' ' -f2)

if [ -z "$TCP\_STAT\_VALUE" ];then

TCP\_STAT\_VALUE=0

fi

echo $TCP\_STAT\_VALUE

}

tcp\_status\_fun $1;

chmod +x tcp\_status.sh #添加执行权限

# 12.4 key的linux\_tcp.conf的子配置文件如下：

# cat /etc/zabbix/zabbix\_agentd.d/tcp.conf

UserParameter=tcp\_status[\*],/bin/bash /etc/zabbix/scripts/tcp\_status.sh "$1"

或者不用写脚本，直接在vi /etc/zabbix/zabbix\_agentd.d/tcp.conf 进入tcp.conf文件

中添加如下内容

UserParameter=netstat.status[\*],/usr/sbin/ss -ant | awk 'NR>1 {++s[$$1]} END {for(k in s) print k,s[k]}'| awk '/$1/{print $$2}'

然后在zabbix服务器上配置item key

# 12.5 重启zabbix-agent,修改配置文件必须重启

#systemctl restart zabbix-agent

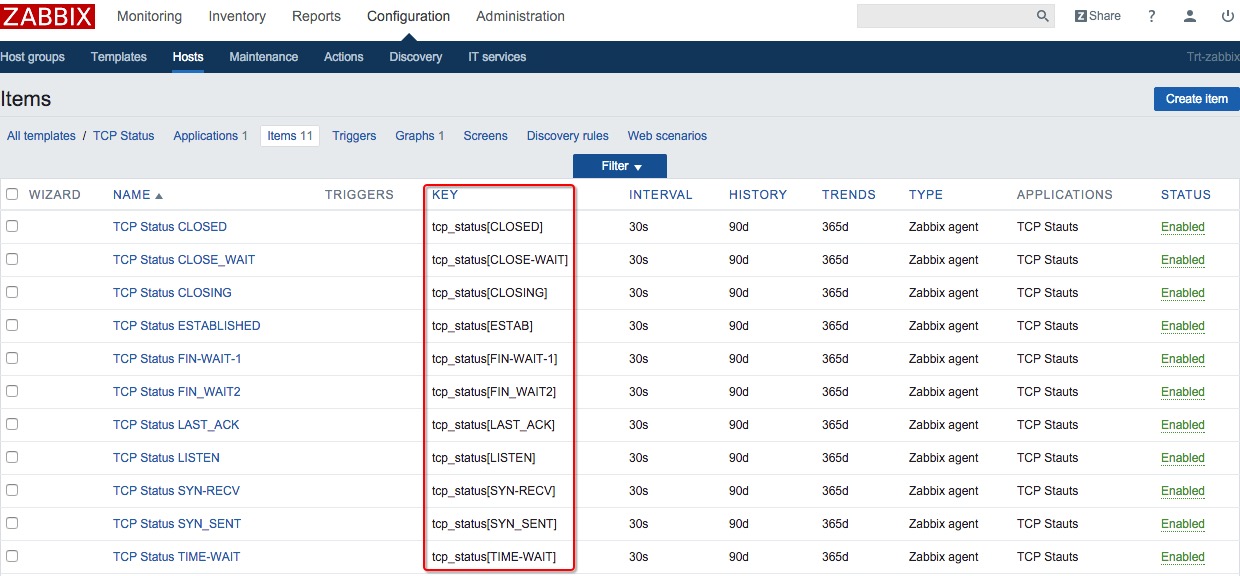
# 12.6 Server测试Agent是否能获取到值，通过Zabbix\_get(不要直接执行脚本)

#zabbix\_get -s 192.168.90.11 -k tcp\_status[ESTAB]

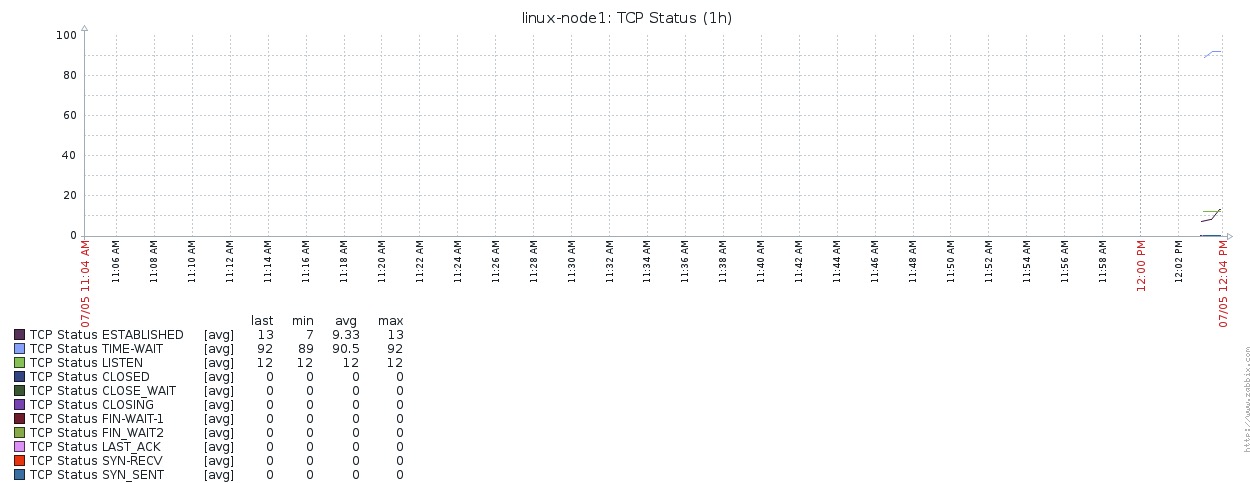
8

# 12.7 展示所有Key(记得将模板关联主机)

(这部分xml可加群索取:471443208)



# 12.8 查看图形(图形是自定义创建)



Zabbix应用监控系列之TCP状态监控

# 13.1 Zabbix应用监控系列之Nginx状态监控

# 13.2 配置所有Agent(标准化目录结构)

[root@linux-node1 ~]# vim /etc/zabbix/zabbix\_agentd.conf #编辑配置文件引用key

Include=/etc/zabbix/zabbix\_agentd.d/\*.conf

[root@linux-node1 ~]# mkdir /etc/zabbix/scripts #存放Shell脚本

# 13.3 在nginx.conf的Server标签中添加如下内容

(如果nginx通过saltstack等配置管理工具进行统一管理,则需要在模板中统一加入这段配置)

location /nginx\_status {

stub\_status on;

access\_log off;

allow 127.0.0.1;

deny all;

}

# 13.3.本地访问Nginx Status测试

[root@linux-node1 ~]# curl http://127.0.0.1/nginx\_status

Active connections: 1

server accepts handled requests

1 1 1

Reading: 0 Writing: 1 Waiting: 0

|  |
| --- |
| Nginx状态解释：  Active connections Nginx正处理的活动链接数1个  server Nginx启动到现在共处理了1个连接。  accepts Nginx启动到现在共成功创建1次握手。  handled requests Nginx总共处理了1次请求。  Reading Nginx读取到客户端的 Header 信息数。  Writing Nginx返回给客户端的 Header 信息数。  Waiting Nginx已经处理完正在等候下一次请求指令的驻留链接，开启。Keep-alive的情况下，这个值等于active-（reading + writing）。  请求丢失数=(握手数-连接数)可以看出,本次状态显示没有丢失请求。 |

# 13.4.编写Nginx的Shell脚本(如果端口不一致,只需要修改脚本端口即可)

|  |
| --- |
| [root@linux-node1 ~]# cd /etc/zabbix/scripts  [root@linux-node1 scripts]# vim nginx\_status.sh  #!/bin/bash  ############################################################  # $Name: nginx\_status.sh  # $Version: v1.0  # $Function: Nginx Status  # $Author: xuliangwei  # $organization: www.xuliangwei.com,www,bjstack.com  # $Create Date: 2016-06-23  # $Description: Monitor Nginx Service Status  ############################################################  NGINX\_PORT=$1 #根据具体情况,通过web传入端口参数即可  NGINX\_COMMAND=$2  nginx\_active(){  /usr/bin/curl -s "http://127.0.0.1:"$NGINX\_PORT"/nginx\_status/" |awk '/Active/ {print $NF}'  }  nginx\_reading(){  /usr/bin/curl -s "http://127.0.0.1:"$NGINX\_PORT"/nginx\_status/" |awk '/Reading/ {print $2}'  }  nginx\_writing(){  /usr/bin/curl -s "http://127.0.0.1:"$NGINX\_PORT"/nginx\_status/" |awk '/Writing/ {print $4}'  }  nginx\_waiting(){  /usr/bin/curl -s "http://127.0.0.1:"$NGINX\_PORT"/nginx\_status/" |awk '/Waiting/ {print $6}'  }  nginx\_accepts(){  /usr/bin/curl -s "http://127.0.0.1:"$NGINX\_PORT"/nginx\_status/" |awk 'NR==3 {print $1}'  }  nginx\_handled(){  /usr/bin/curl -s "http://127.0.0.1:"$NGINX\_PORT"/nginx\_status/" |awk 'NR==3 {print $2}'  }  nginx\_requests(){  /usr/bin/curl -s "http://127.0.0.1:"$NGINX\_PORT"/nginx\_status/" |awk 'NR==3 {print $3}'  }  case $NGINX\_COMMAND in  active)  nginx\_active;  ;;  reading)  nginx\_reading;  ;;  writing)  nginx\_writing;  ;;  waiting)  nginx\_waiting;  ;;  accepts)  nginx\_accepts;  ;;  handled)  nginx\_handled;  ;;  requests)  nginx\_requests;  ;;  \*)  echo $"USAGE:$0 {active|reading|writing|waiting|accepts|handled|requests}"  esac |

给脚本添加执行权限

[root@linux-node1 scripts]# chmod +x nginx\_status.sh

# 13.5.key的nginx\_status.conf的子配置文件如下：

[root@linux-node1 ~]# cat /etc/zabbix/zabbix\_agentd.d/nginx\_status.conf

UserParameter=nginx\_status,/bin/bash /etc/zabbix/zabbix\_agentd.d/scripts/nginx/nginx\_status.sh "$1" "$2"

# 13.6.重启zabbix-agent

[root@linux-node1 ~]# systemctl restart zabbix-agent

# 13.7.测试一定使用Zabbix\_get来获取值（传入2参数）

[root@linux-node1 ~]# zabbix\_get -s 192.168.90.11 -k nginx\_status[80,writing]

# 14.Zabbix应用监控系列之Redis状态监控

Redis使用自带的INFO命令，进行状态监控。以一种易于解释且易于阅读的格式，返回关于 Redis 服务器的各种信息和统计数值。

# 14.1.配置所有Agent(标准化目录结构)

vim /etc/zabbix/zabbix\_agentd.conf #编辑配置文件引用key

Include=/etc/zabbix/zabbix\_agentd.d/\*.conf

mkdir /etc/zabbix/scripts #存放Shell脚本

# 14.2.编写Shell脚本

脚本端口、连接redis服务地址根据具体情况进行修改

AUTH认证没有开启，将PASSWD修改为空即可。

脚本端口、连接redis服务地址根据具体情况进行修改

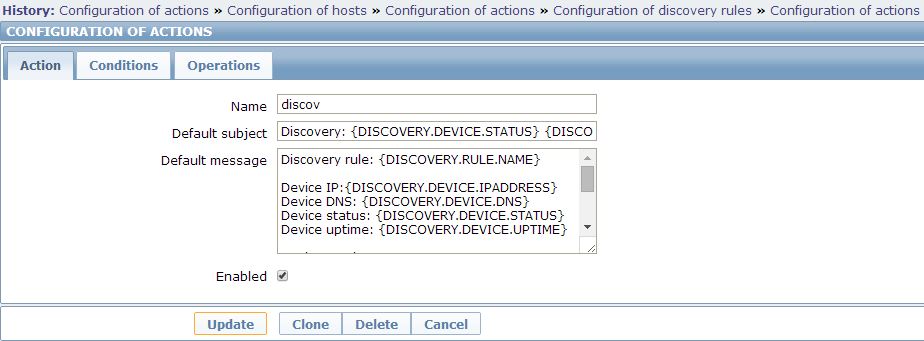
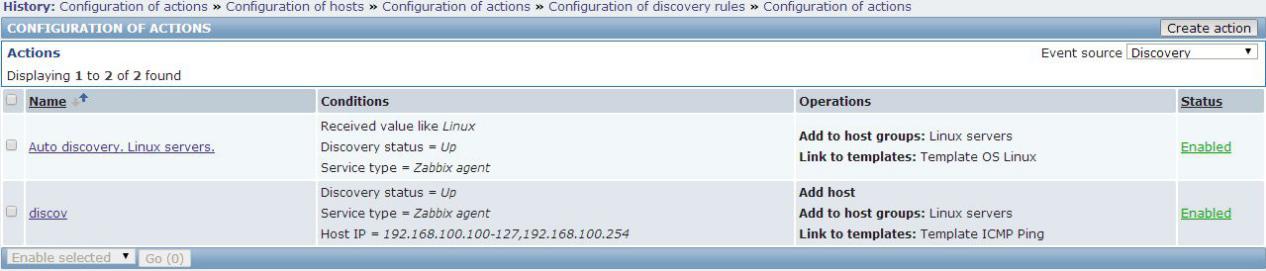
AUTH认证没有开启，将PASSWD修改为空即可。

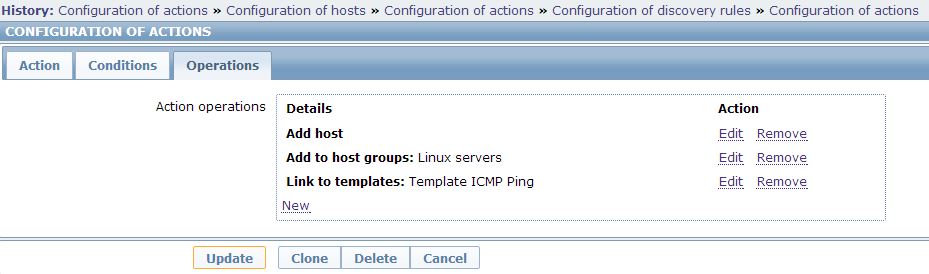
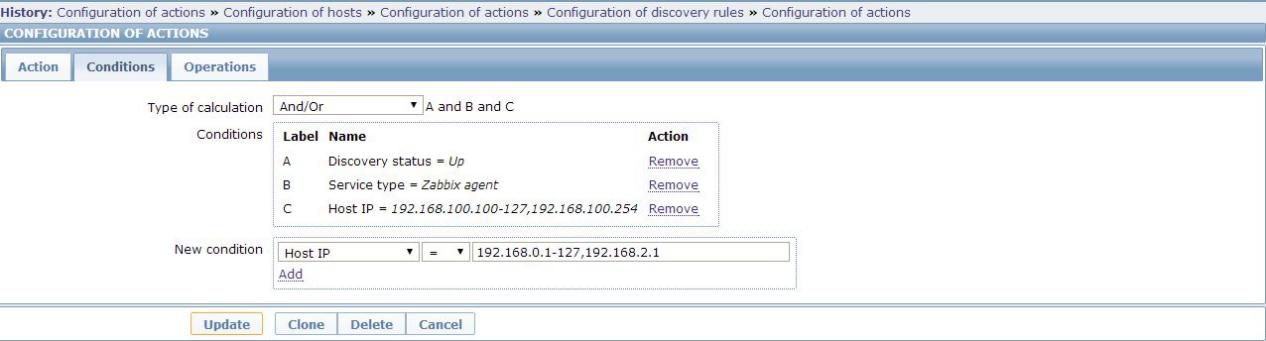
# 15.1 zabbix自定义发现主机

# 15.2 Configuration--->Discover--->Create discovery rule

# 图15-115.3 Configuration--->Action----->Create action

如图15-2所示





# 15.4 configuration--->Hosts



16.1 zabbix应用监控系列之Apache状态监控

## 16.1.1 Apache安装加载mod\_status模块

cd /home/wpaccp/tools/httpd-2.2.27/modules/generators/

/application/apache/bin/apxs -i -a -c mod\_status.c

/application/apache/bin/apachectl stop

/application/apache/bin/apachectl start

vim /application/apache/conf/httpd.conf

修改如下选项为

将#Include conf/extra/httpd-info.conf修改为Include conf/extra/httpd-info.conf

添加ExtendedStatus On参数选项

vim /application/apache/conf/extra/httpd-info.conf

修改如下选项为

<Location /server-status>

SetHandler server-status

Order deny,allow

#Deny from all

#Allow from .example.com

Allow from all

</Location>

/application/apache/bin/apachectl restart

在客户端浏览器上输入:http://192.168.100.192/server-status

-------------------------------------------------------------------------------

UserParameter=nginx.log[\*],/usr/bin/tac /www/log/host.access.log|awk -F "[ ':']" '$$12 ~/$1/{count[$$12]++}END{for(i in count) print i,count[i]}'

awk '/MemTotal/{total=$2}/MemFree/{free=$2}END{print (total-free)}' /proc/meminfo

awk '/MemTotal/{total=$2}/MemFree/{free=$2}END{print (total-free)/total\*100"%"}' /proc/meminfo

awk '/MemTotal/{total=$2}/MemFree/{free=$2}END{print (free/total)\*100"%"}' /proc/meminfo

zabbix\_get -s 127.0.0.1 -k "meminfo\_status[usepre]"

zabbix\_get -s 127.0.0.1 -k "meminfo\_status[freepre]"

free -m | awk '/Mem/{total=$2}/cache:/{use=$3}END{print(use/total)\*100}'

free -m | awk '/Mem/{total=$2}/cache:/{free=$4}END{print(free/total)\*100}'

curl -s "192.168.100.192:80/server-status" | awk -F "[ <>]+" '/Total Traffic:/{print $9}'

curl -s "192.168.100.192:80/server-status" | awk -F "[ <>]+" '/Total accesses/{print $5}'

curl -s "192.168.100.192:80/server-status" | awk -F "[ <>]+" '/idle workers/{print $3}

curl -s "192.168.100.192:80/server-status" | awk -F "[ <>]+" '/idle workers/{print $8}

curl -s "192.168.100.192:80/server-status" | awk -F "[ <>]+" '/idle workers/{idle=$3;request=$8}END{print idle+request}'

-------------------------------------------------------------------------------------

curl -s "$1:$2/server-status" | awk -F "[ <>]+" '/$3:/{print $9}'

curl -s "$1:$2/server-status" | awk -F "[ <>]+" '/$3/{print $5}'

curl -s "$1:$2/server-status" | awk -F "[ <>]+" '/$3/{print $$3}

curl -s "$1:$2/server-status" | awk -F "[ <>]+" '/$3/{print $8}

curl -s "$1:$2/server-status" | awk -F "[ <>]+" '/$3/{idle=$$3;request=$8}END{print idle+request}'

端口监控流量命令：

ifconfig eth0|awk -F "[ :]+" '/RX packets:/{print $4}'

ifconfig eth0|awk -F "[ :]+" '/TX packets:/{print $4}'

ifconfig eth0|awk -F "[ :]+" '/RX bytes:/{print $4}'

ifconfig eth0|awk -F "[ :]+" '/TX bytes:/{print $9}'

/sbin/ifconfig eth0|awk -F "[ :]+" '/RX packets:/{print $4}'

/sbin/ifconfig eth0|awk -F "[ :]+" '/TX packets:/{print $4}'

/sbin/ifconfig eth0|awk -F "[ :]+" '/RX bytes:/{print $4}'

/sbin/ifconfig eth0|awk -F "[ :]+" '/TX bytes:/{print $9}'

monitor\_traffic\_eth0.sh

#分组显示当前的主机的连接数

UserParameter=eth0.traffic[\*],/usr/sbin/ss -ant | awk 'NR>1 {++s[$$1]} END {for(k in s) print k,s[k]}'| awk '/$1/{print $$2}'

常见故障

Received empty response from Zabbix Agent at [192.168.100.100]. Assuming that agent dropped connection because of access permission

大概意思是说没有权限访问agent端口10050，解决方法如下：

[root@centos65-01 ~]# sed -i 's#Server=127.0.0.1#Server=192.168.100.100#' /etc/zabbix/zabbix\_agentd.conf