zabbix配置操作详解（三）

51niux.com/

柴少鹏的官方网站 技术在分享中进步，水平在学习中升华

2017年07月21日 14:50:25

http://www.51niux.com/?id=147 #这里已经对文件、命令都做了详解，通过命令的操作也简单的了解了下想对于agent端来说的主动监控和被动监控，也简单了解了数据库的表结构。这里我们延续上面，继续记录。

一、zabbix-server对数据的存储

其他的监控系统是将数据存储在rrd数据库里面，不存在数据库越来越大的情况，这跟数据库的环形存储特性有关系。zabbix不管是采用分布式架构还是传统的服务端与客户端直接相连的模式，最终都是将数据存储在mysql里面。

1.1 zabbix对数据存储

数据存储的大小与每秒处理的数据量有关，所以数据存储取决于两个因素：更新的数据量和删除的数据量（Housekeeper）。

zabbix-server将采集到的数据主要存储在History和Trends表中，其表结构中的数据类型如下：

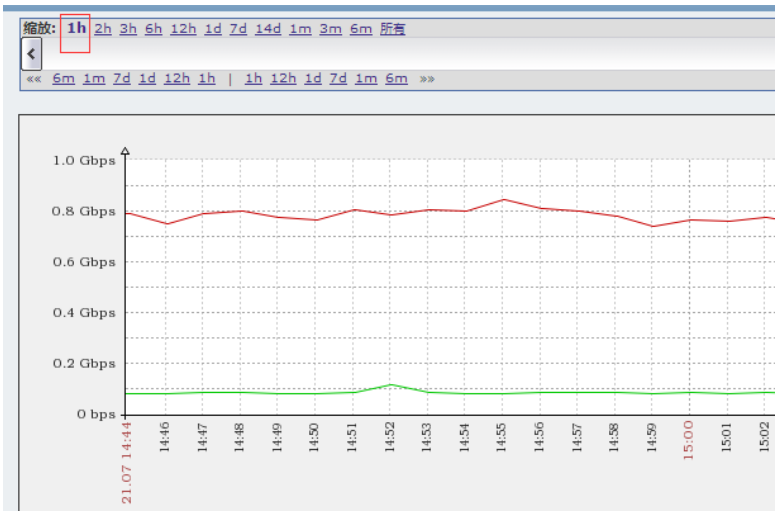
表	数 据 类 型	最大值范围	
history	保存历史数据	数字（浮点数）	double(16,4)~999999999999.9999
history_uint	保存历史数据	数字（非符号）	bigint(20) - 2^{64}+1
history_str	保存短的字符串数据	字符	varchar(255)~255
history_text	保存长的字符串数据	文本	text~65535
history_log	保存日志字符串	日志	text~65535
trends	保存趋势数据	数字（浮点数）	double(16,4)~999999999999.9999
trends_uint	保存趋势数据	数字（非符号）	bigint(20) - 2^{64}+1

另外,acknowledges、alerts、auditlog、events和service_alarms表的数据也较大。

在History表中，主要存储数据到的历史数据，而Trends主要存储经过计算的历史数据（如每小时数据的最小值、最大值和平均值）。其实主要还是history表占的最大，trends表只是相对于其他表大而已。

在Trends表中，主要存储的是趋势数据，Trends 基本上是收集到的按小时合并的数据（仅仅是数值类型）。Zabbix 服务器按小时把所有的值从 history 表中提取出来，并按每小时计算最小值，平均值和最大值。

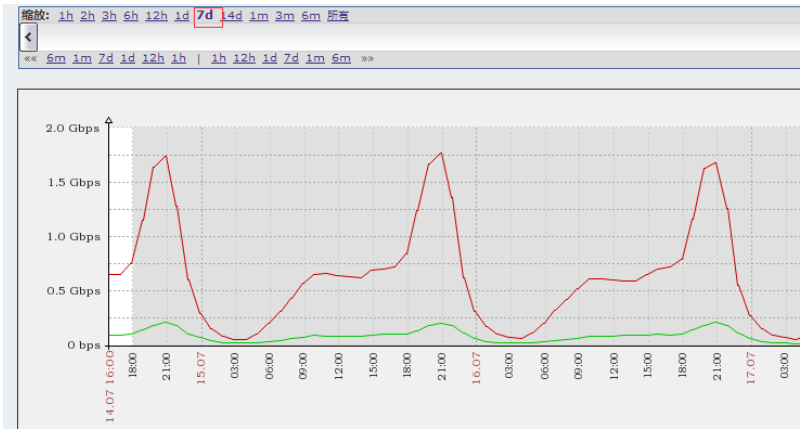
历史图：



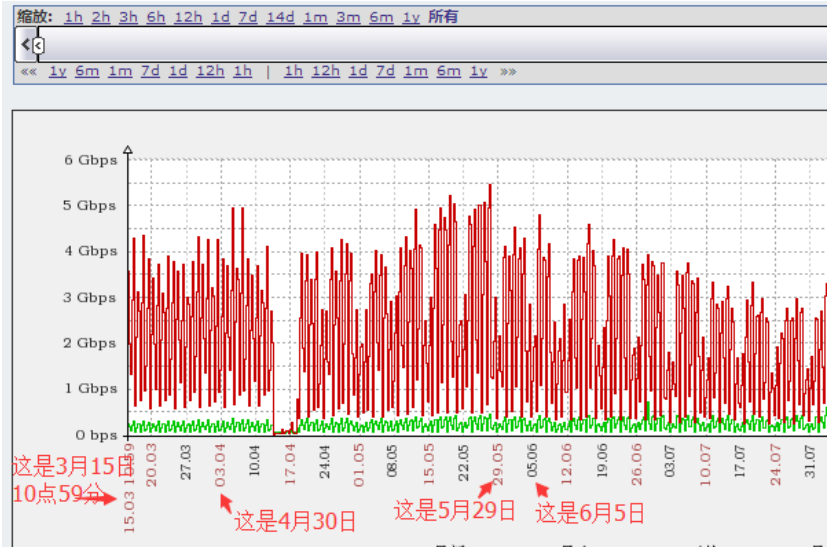
#由上图可以看出，一分钟一个格（我们一般都是60秒采集一次），这种的话就是历史数据，就是数据之间的时间相差不大，数据是以分钟为单位存储的，如果300台主机每个是10个监控项，一分钟就是3000条数据。所以一般历史数据的保存时间一般都不会设置太久，比如7天半个月之类的，虽然默认的是90天的历史保留时间。

#当然有的除外，比如我这里有个zabbix是专门用来存储各机房交换机出口的流量，因为cacti趋势的比较狠，所以我这里的历史保留时间和趋势保留时间都比较久。但是一般的都不应该保留这么久。

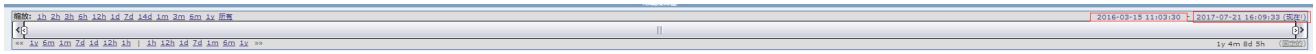
趋势图：



#当前时间减7天，就是7月14日16点，然后看图是3小时一个格，这明显图的颗粒变得很粗，这就是趋势，1小时一条数据，24*7可比1*60*24*7，这个数据量级要少太多了。所以历史数据保存时间短一点，数据库空间增长会少很多。



#再拉大呢，我拉倒所有，这就跨年了，可以看到是7天一格，真的就是看一个大概趋势了。这个日期是从上往下念，我图中也有标准了。那么我这数据起始位置只有月日，没有年啊，可以通过下面的图来查看。

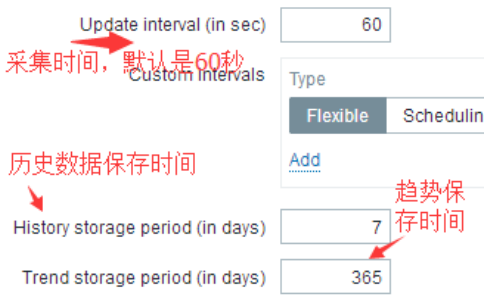


#看标红的地方就能看出下面的趋势图从什么时间开始到什么时间结束，左边那什么1h、2h、12h，7d等等，都是可以点的，就是你想看哪个时间跨度的图。

Housekeeper

上面展示的history和Trends都存储在mysql中，直到 Zabbix 服务器的 Housekeeper 基于 Keep history 和 Keep trends 配置删除它们：

#看上图，就是items里面的设置，首先采集时间是60秒，也就是一分钟去采集一次，也就是说此条item一分钟会往mysql的history表里面插入一条数据。历史数据是保留7天的，然后这个item在history表里面保存的总条数为：1（分钟）*60*24*7=10080行记录，一般来说一条记录需要占用50个字节，也就是说这一个监控项所占用的history表在实际物理空间上面的实际使用量为：504000字节=500KB左右。



#趋势数据设置的是365天，趋势数据是一小时计算一次，一般趋势数据可以存放的时间长一点，比如5年10年的，也是看什么数据有没有必要存那么长时间。它保存的是每小时收集到数据的最小值，最大值和平均值以及每小时收集到值的总数，也就是一小时一条数据。我们先按1年的时间来算，1*24（小时）*365=8760（条数据），如果换算成占用物理空间大小呢：趋势数据一条记录大概占用128字节，就是：8760*128=1095KB约等于1MB。

#而Housekeeper就是根据上图定义的历史数据在mysql表里面的保存时间和趋势数据的保存时间，去通过查询数据的时间戳去删除超过时限的老数据。

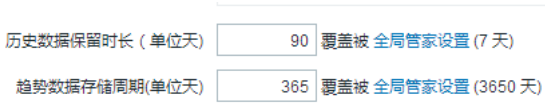
#当然还有个事件（alerts）也会占用点磁盘空间：报警、警告、恢复等等事情，一个事件大概占用130个字节，这个就是按条算了，一般也占用不了太大的空间，一般也不会老报警啥的。

注：

如果是很大型的监控系统，一般都会禁止Housekeeper，因为数据库里面数据太多了，每次去数据库里面搜索并删除消耗和耗时太大，一般都会采用优化数据库表结构、分库等其他的方法来搞。

默认历史保存时间是90天，趋势保存时间是365天，如果你想将历史保存时间变短或者变长的话，不去改全局的话，那么创建的每个items都要去修改，这工作量就大了，如何全局修改呢？

再添加items会看到：



#这是一种方式，不用每次添加Items都手工修改一下，但是更好的方式还是尽量用模板的形式，这样能大大的减少工作量，那么多主机和items肯定不能一个个来加，除非自定义的。

博文来自：www.51niux.com

二、相关配置

2.1 配置流程

zabbix完整的监控配置流程可以简单描述为：

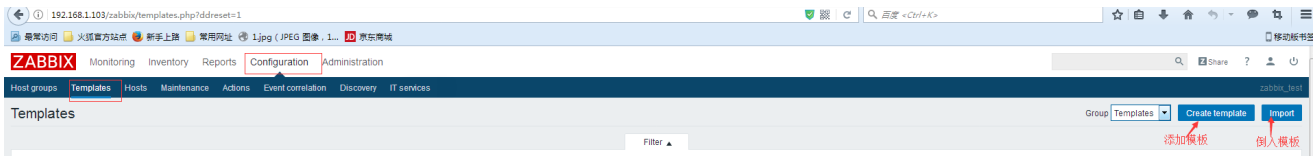
Host groups（主机组）-》Hosts（主机）->Applications（监控项组）->Items（监控项）->Triggers(触发器)->Event（事件）->Actions（处理动作）->User groups（用户组）->Users(用户)->Medias（告警方式）->Audit(日志审计)。

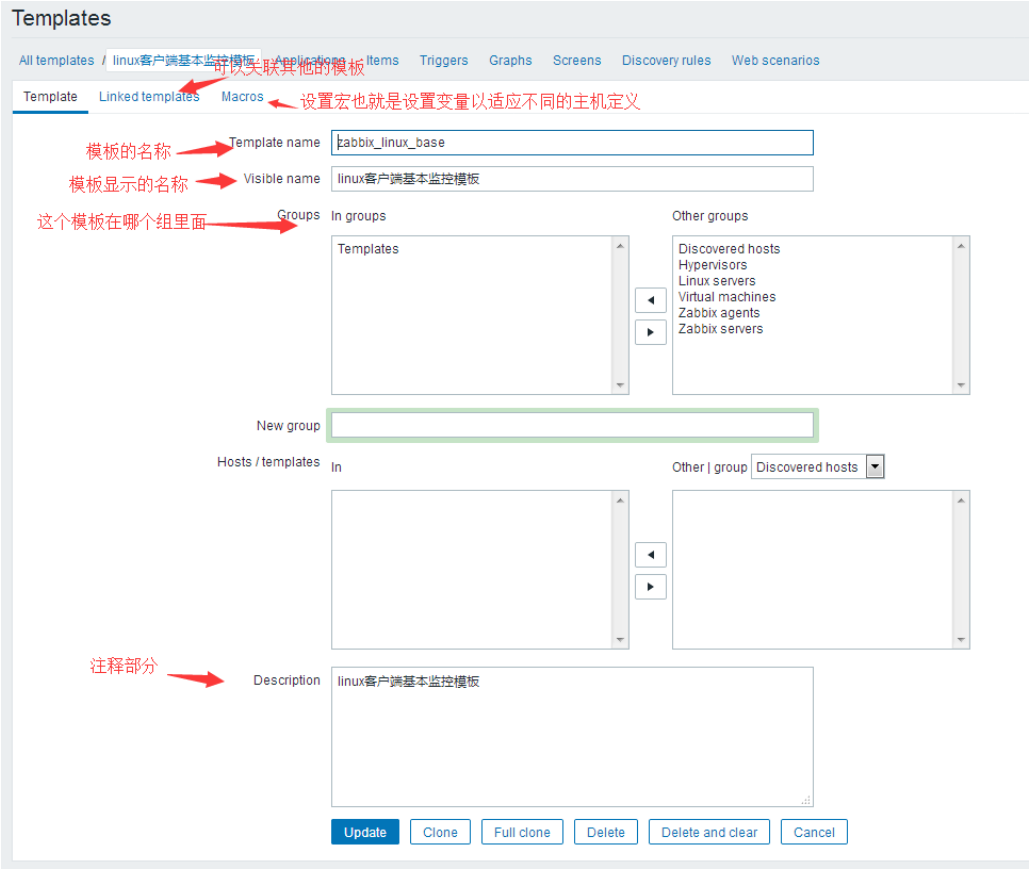
实际使用的时候Items、Trigger、Graph通常采用模板进行监控配置，模板的特点就是可以对相同需求的监控项重复使用，无须对每台主机进行逐个设置。

对于zabbix来说，配置Graph不是必须的，因为没有配置图形，数据获取也不影响，数据获取是Items的功能，我们只需要对比较关心的Items设置Graph进行数据可视化。

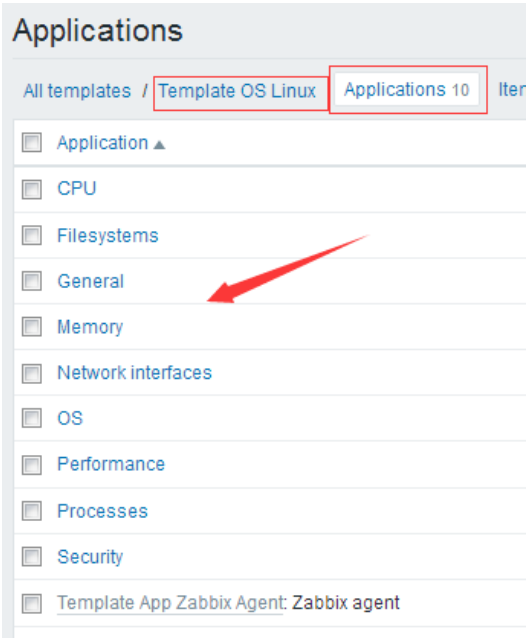
对于zabbix来说，配置Trigger不是必须的，但是对于特别关注的数据，需要对取到的值进行条件判断，这时配置触发器是必须的。

2.2 制作模板

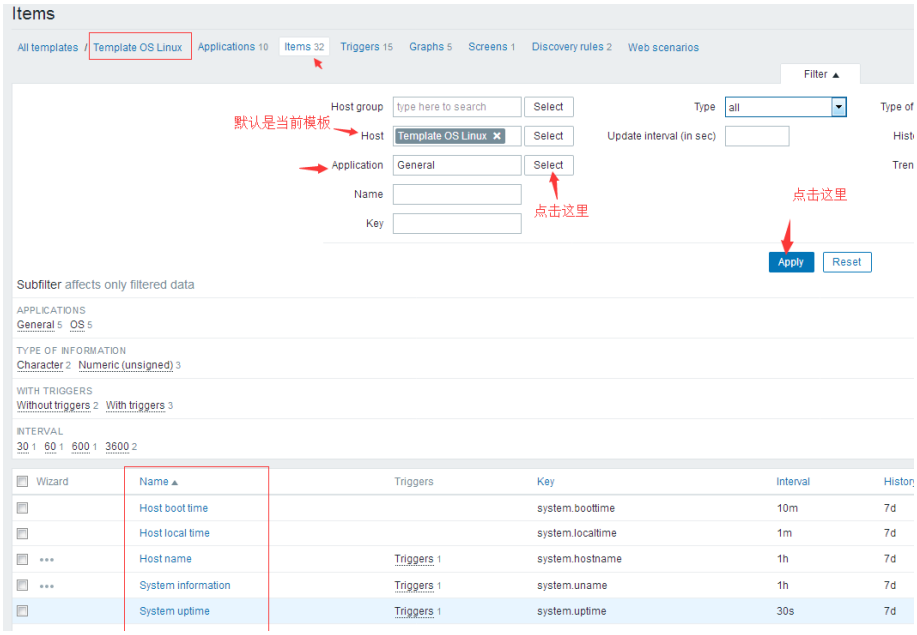




2.3 给模板添加Applications：

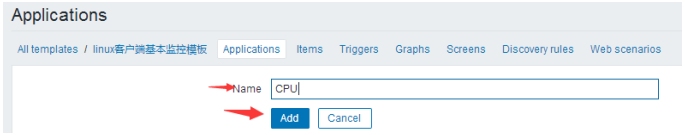


#上图是Template OS Linux的监控项组模板，这就是相当于我们的例子，我们就照着它来创建一些application，当然有些我们可能用不到的就去掉了。

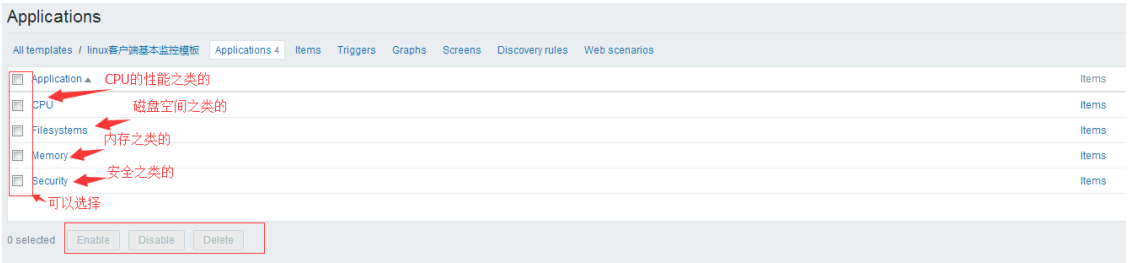


#也可以通过去查看items去看看这个application下面都有哪些items供我们参考

#点击右上角的Create application



下面是搞完之后的applications截图：



2.4 给模板添加items并关联applications：

#这个就是照着Template OS Linux模板里面的items，想加哪个items就加哪个。如下面：

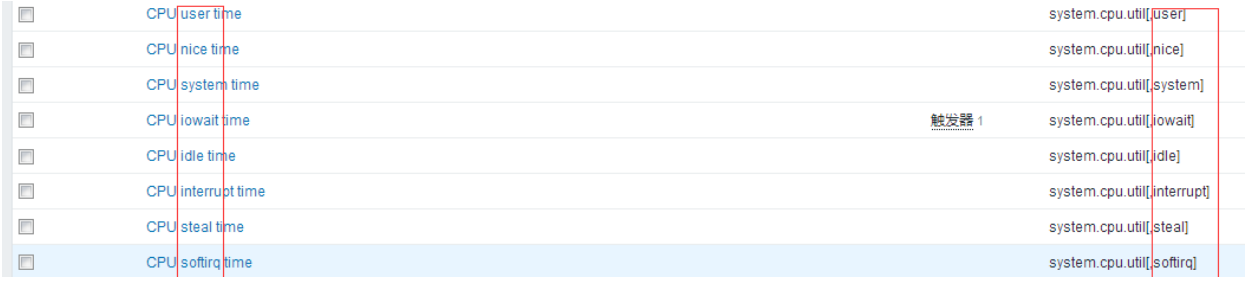
注（Name中的$1..$9）：

名称：CPU $2 time #这个$2代表下面键值中[]中第二个参数，[]里面就是传进来的参数嘛，$1就是以第一个参数，以此类推，所以它的显示因该是CPU system time。

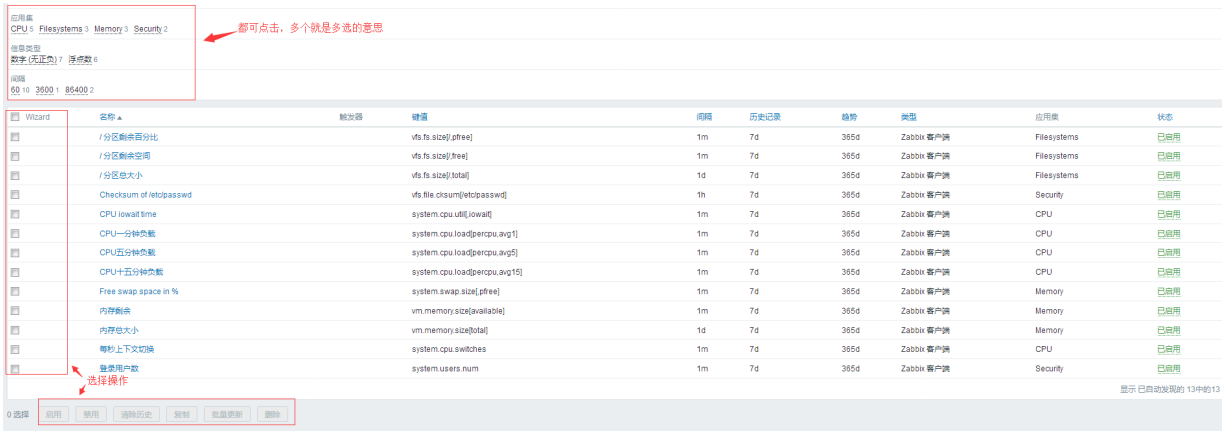
键值：system.cpu.util[,system]

内置了哪些键呢：https://www.zabbix.com/documentation/3.2/manual/config/items/itemtypes/zabbix_agent

下面是主机items显示的效果：



好了下图我们定义了13个items：



#注意关于Units（单位的注意）：

默认情况下，如果原始值超过1000，那么他会先除以1000并且显示出来例如，设置了单位为bps并且收到的值为11102，将会显示为11.1Kbps 如果单位被指定为 B (byte)，Bps (bytes per second) ，那么它会除以1024然后再显示数据。所以大家在监控流量和文件大小的时候不要用错单位，否则会出现数据不一致的情况。
如下为时间单位：
unixtime – 转为 “yyyy.mm.dd hh:mm:ss”. 只能使用正数。
uptime – 转为“hh:mm:ss” 或者“N days, hh:mm:ss”
例如，收到的值为881764秒，他将会显示为“10 days, 04:56:04”
s – 转为“yyy mmm ddd hhh mmm sss ms”;
例如，收到的值为881764(单位秒)，他将会被显示为10d 4h 56m”，只会显示3个单元。有时候只会显示2个单元，例如”1m 5h”（不包含分，秒，毫秒），如果返回的值小于0.001，他只会显示”<1 ms”禁用单位：ms、rpm、RPM、%

博文来自：www.51niux.com

2.5 给模板添加trigger(触发器)：

还是触发器的右上角点击创建触发器：

直接表达式添加触发器：

#{HOST.NAME}是一个内置宏。这里是官网内置宏的列表：https://www.zabbix.com/documentation/3.2/manual/appendix/macros/supported_by_location

#主要是表达式，这里才是核心，如：{zabbix_linux_base:vm.memory.size[available].last(0)}<20M

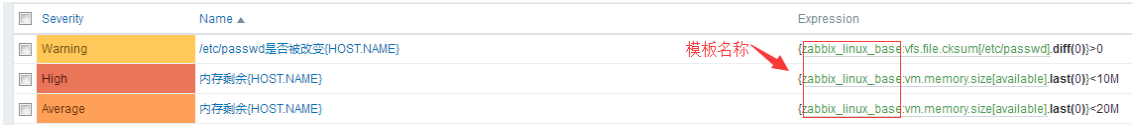
表达式语法格式：

```
{<server>:<key>.<function>(<parameter>)}<operator><constant>
```

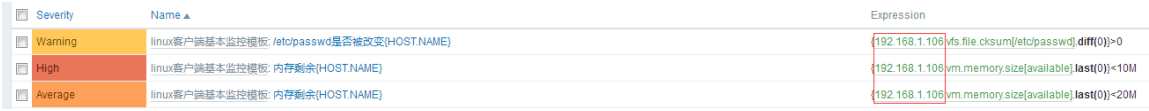
用上面的例子拆分举例就是：

```
<server>     ：模板名称，即zabbix_linux_base，等谁引用这个模板，那这里就是引用模板对应的主机了
<key>        ：即监控模板里的项目items，这里就是内存剩items的键：vm.memory.size[available]
<function>   ：就是我们触发器里面的功能项也就是函数，这里last就是表示last函数，即last()
<parameter>  ：就是上面方法所使用的参数，这里即0，last(0)也就是表示倒数第0个值，也就是最新的值
<operator>   ：可使用的操作符，这里<是小于的意思，操作符有：/ * - + < > #（不等于） = &（逻辑与） |（逻辑或）
<constant>   ： 常量，在这里即是20M
```

模板中trigger效果图：



192.168.1.106主机添加完模板之后trigger效果图：



function都有哪些啊，官网地址：https://www.zabbix.com/documentation/3.2/manual/appendix/triggers/functions

大概来说就是：

abschange #作用：返回最近获得值与之前获得值的差值，对于字符串0表示相等，1表示不同

```
举例：
change(0)>n:忽略参数一般输入0，表示最近得到的值与上一个值的差值大于n
```

avg #返回一段时间的平均值

```
举例：
avg(5):最后5秒的平均值
avg(#5）：表示最近5次得到值的平均值
avg(3600,86400）：表示一天前的一个小时的平均值
如果仅有一个参数，表示指定时间的平均值，从现在开始算起，如果有第二个参数，表示漂移，从第二个参数前开始算时间，
#n表示最近n次的值
```

band #

change #最后的区别和之前的值

count #返回时间间隔内数值的统计

```
举例：
count(600)最近10分钟得到值的个数
count(600,12)最近10分钟得到值的个数等于12
count(600,12,"gt")最近10分钟得到值的个数大于12
count(#10,12,"gt")最近10个值中，值大于12的个数
count(600,12,"gt",86400)24小时之前的10分钟内值大于12的个数
count(600,6/7,"band")-thenumberofvaluesforlast10minuteshaving'110'(inbinary)inthe3leastsignificantbits.
count(600,,,86400)24小时之前的10分钟数据值的个数
第一个参数：指定时间段
第二个参数：样本数据
第三个参数：操作参数
第四个参数：漂移参数
#支持的操作类型
eq: 相等
ne: 不相等
gt: 大于
ge: 大于等于
lt: 小于
le: 小于等于
like: 内容匹配
```

date #时间，返回当前的时间，格式YYYYMMDD

dayofmonth #返回当前是本月的第几天

dayofweek #返回当前是本周的第几天

delta #返回时间间隔内的最大值与最小值的差值

diff #返回值为1表示最近的值与之前的值不同，0为其他情况

forecast #预测未来的值

fuzzytime #返回值为1表示监控项值的时间戳与ZabbixServer的时间多N秒,0为其他.常使用system.localtime来检查本地时间是否与Zabbixserver时间相同.

iregexp #不区分大小写

last #最近的值，如果为秒，则忽略，#num表示最近第N个值，请注意当前的#num和其他一些函数的#num的意思是不同的

```
last(0)等价于last(#1)last(#3)表示最近第3个值(并不是最近的三个值)
本函数也支持第二个参数time_shift，例如last(0,86400)返回一天前的最近的值
如果在history中同一秒中有多个值存在，Zabbix不保证值的精确顺序
#num从Zabbix1.6.2起开始支持,timeshift从1.8.2其开始支持,可以查询avg()函数获取它的使用方法
```

logeventid #

logseverity #

logsource #

max #返回指定时间间隔的最大值.

min #返回指定时间间隔的最小值.

nodata #检查收到的任何数据。

now #返回距离Epoch(1970年1月1日00:00:00UTC)时间的秒数

percentile #百分比

prev #返回之前的值，类似于last(#2)

regexp #第一个参数为string,第二个参数为秒或#num。检查最近的值是否匹配正则表达式，参数的正则表达式为POSIX扩展样式,第二个参数为秒数或收集值的数目，将会处理多个值.本函数区分大小写。当返回值为1时表示找到，0为其他.

str #第一个参数为string,第二个参数为秒或#num。查找最近值中的字符串。第一个参数指定查找的字符串，大小写敏感。第二个可选的参数指定秒数或收集值的数目，将会处理多个值。当返回值为1时表示找到，0为其他.

strlen #指定最近值的字符串长度(并非字节),参数值类似于last函数.例如strlen(0)等价于strlen(#1),strlen(#3)表示最近的第三个值,strlen(0,86400)表示一天前的最近的值.

sum #返回指定时间间隔中收集到的值的总和.时间间隔作为第一个参数支持秒或收集值的数目(以#开始).从Zabbix1.8.2开始，本函数支持time_shift作为第二个参数。可以查看avg函数获取它的用法

```
sum(600):表示在600秒之内接收到所有值的和
sum(#5):表示最后5个值的和
```

time #返回当前的时间，HHMMSS格式的当前时间。

timeleft #一个项目达到所需的时间间隔,以秒为单位指定的阈值。

表达式语法举例：

官网的一些举例：https://www.zabbix.com/documentation/3.2/manual/config/triggers/expression

示例1：

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5
#www.zabbix.com这个主机的监控项，最新的CPU负载值如果大于5，那么表达式会返回true，这样一来触发器状态就改变为“problem”了。
```

示例2：

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5 or {www.zabbix.com:system.cpu.load[all,avg1].min(10m)}>2
#当前cpu负载大于5或者最近10分内的cpu负载大于2，那么表达式将会返回true.
```

示例3：

```
{www.zabbix.com:vfs.file.cksum[/etc/passwd].diff()}=1
#/etc/passwd最新的checksum与上一次获取到的checksum不同，表达式将会返回true.
```

示例4：

```
{www.zabbix.com:net.if.in[eth0,bytes].min(5m)}>100K
#当前主机网卡eth0最后5分钟内接收到的流量超过100KB那么触发器表达式将会返回true
```

示例5：

```
{smtp1.zabbix.com:net.tcp.service[smtp].last()}=0 and {smtp2.zabbix.com:net.tcp.service[smtp].last()}=0
#当smtp1.zabbix.com和smtp2.zabbix.com两台主机上的SMTP服务器都离线，表达式将会返回true.
```

示例6：

```
{zabbix.zabbix.com:agent.version.str("beta8")}=1
#如果当前zabbix agent版本包含beta8（假设当前版本为1.0beta8），这个表达式会返回true.
```

示例7：

```
{zabbix.zabbix.com:icmpping.count(30m,0)}>5
#最近30分钟zabbix.zabbix.com这个主机超过5次不可到达返回true
```

示例8：

```
{zabbix.zabbix.com:tick.nodata(3m)}=1
#tick为Zabbix trapper类型，首先我们要定义一个类型为Zabbix trapper，key为tick的item。我们使用zabbix_sender定期发送数据给tick，如果在3分钟内还未收到
zabbix_sender发送来的数据，那么表达式返回一个true，与此同时触发器的值变为“PROBLEM”。
```

示例9：

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2 and {zabbix:system.cpu.load[all,avg1].time()}>000000 and {zabbix:system.cpu.load[all,avg1].time()}
<060000
#只有在凌晨0点到6点整，最近5分钟内cpu负载大于2，表达式返回true，触发器的状态变更为“problem”
```

示例10：

```
{MySQL_DB:system.localtime.fuzzytime(10)}=0
#主机MySQL_DB当前服务器时间如果与zabbix server之间的时间相差10秒以上，表达式返回true，触发器状态改变为“problem”
```

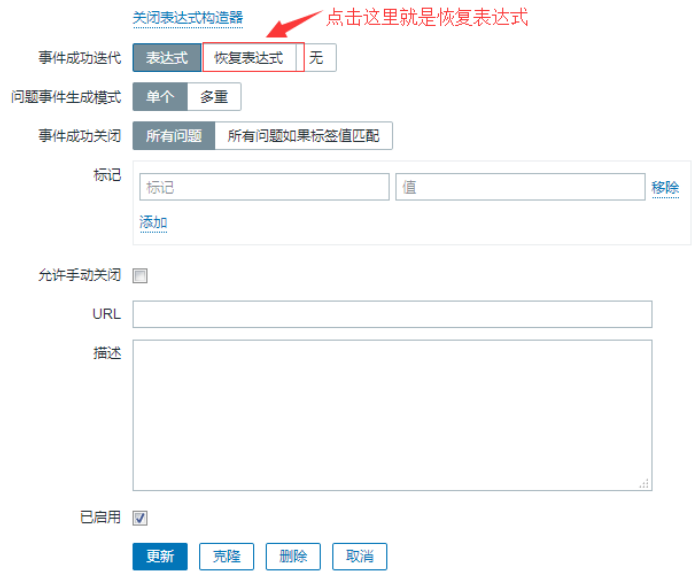
示例11：

```
{server:system.cpu.load.avg(1h)}/{server:system.cpu.load.avg(1h,1d)}>2
#最新一小时的平均负载峰值超过昨天同时段指标两次进行报警
```

Hysteresis（滞后性）：

触发器状态转变为problem需要一个条件，从problem转变回来还需要一个条件才行。一般触发器只需要不满足触发器为problem条件即可恢复。有时候我们需要一个时间间隔一个好的状态和问题,而不是一个简单的阈值。 例如,我们想定义一个触发器,变成了问题当服务器机房温度高于20 c和我们想要呆在这个状态,直到温度低于15度。为了做到这一点,我们首先定义触发器事件的表达式问题。 然后选择“恢复表达”好的事件代和输入一个恢复好事件的表达式。
注意,复苏将计算表达式的值只有当事件是先解决的问题。 复苏是不可能解决问题,如果问题表达条件仍然存在。
示例1：

```
问题表现：
{server:temp.last()}>20
恢复表达式：
{server:temp.last()}<=15
```



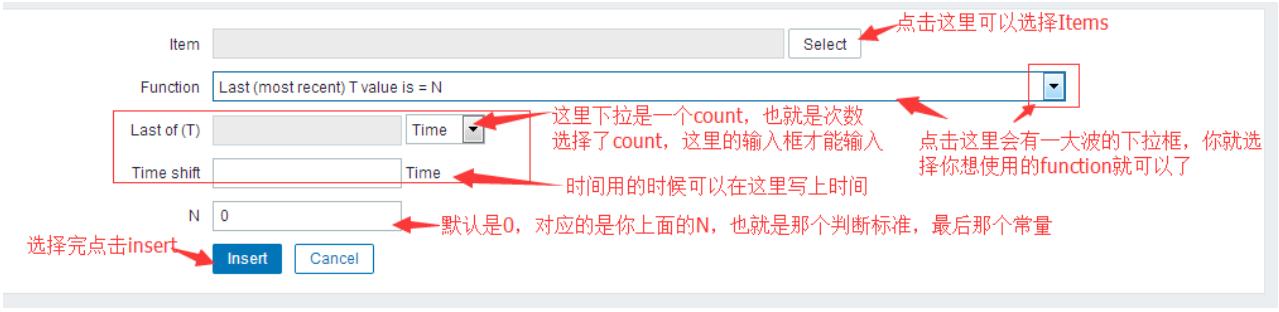
示例2：

问题表达式：
`{server:vfs.fs.size[/,free].max(5m)}<10G  #最近5分钟内剩余磁盘空间小于10GB`
恢复表达式：
`{server:vfs.fs.size[/,free].min(10m)}>40G  #最近5分钟内磁盘空间大于40GB`

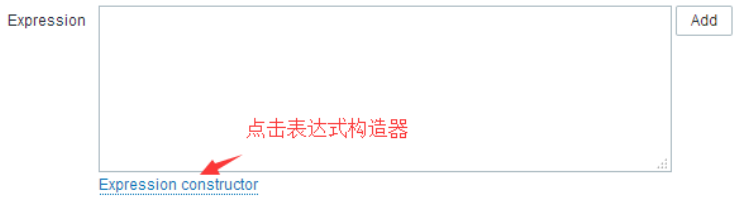
通过表达式构造器添加触发器：

如果说表达式什么的那么多我记不住，我就知道个大概意思就可以了，那就可以通过表达式构造器来添加触发器的表达式。

#通过点击表达式那栏右边的添加按钮或者下方的表达式构造器，都可以进入到表达式添加栏。



`{zabbix_linux_base:vfs.fs.size[/,pfree].last()}<10 or {zabbix_linux_base:vfs.fs.size[/,free].last()}=10  #像这种多条表达式如果根分区磁盘空间百分比低于10%或者根分区磁盘空间小于10B（如果你不加单位就是以你Utils来换算，当然怎么也得写个1G啥的啊），就报警`



#然后上图下面还有个点击这个就是保存设置了。

Close expression constructor

#说到这个Test，当你测试的时候，不能直接输入单位啥的，需要输入整数，比如你内存设置的是1M，那你就得把1M换算成字节的是多少数，然后输入上去测试一下看返回的是True还是False。

改变严重性显示的颜色：

触发器支持的单位：

#触发器支持的单位官网链接：https://www.zabbix.com/documentation/3.2/manual/config/triggers/suffixes

#触发器如果不支持单位的话，那么比如你设置一个磁盘空间的触发器设置，因为它的单位是B，所以你要设置小于1G报警的话，你就要1*1024*1024*1024=1073741824（字节），你的数字就要设置成1073741824

#从上面的例子可以看出，设置成支持的单位就是让程序去算，我们就不用自己算了，因为zabbix采集的时候比如磁盘空间内存剩余空间都是字节的形式，你当然可以设置成字节跟采集上来的字节去比较，如果你设置了单位，它就会将采集上来的数换算成你设置的单位来比较，你再设置的触发器的时候是方便的，不用你自己算了。

```
时间可以使用：s（秒）、m（分钟）、h（小时）、d（天）、w(周)
内存大小可以使用：K（千字节）、M（兆字节）、G（千兆字节）、T（太字节）
单位符号可以使用：K,M,G,T
当B，Bps中的项目值显示在前端时，应用基准2（1K = 1024）。 否则使用10的基数（1K = 1000）。
另外还支持：P（peta）、E（exa）、Z（zetta）、Y（yotta）
```

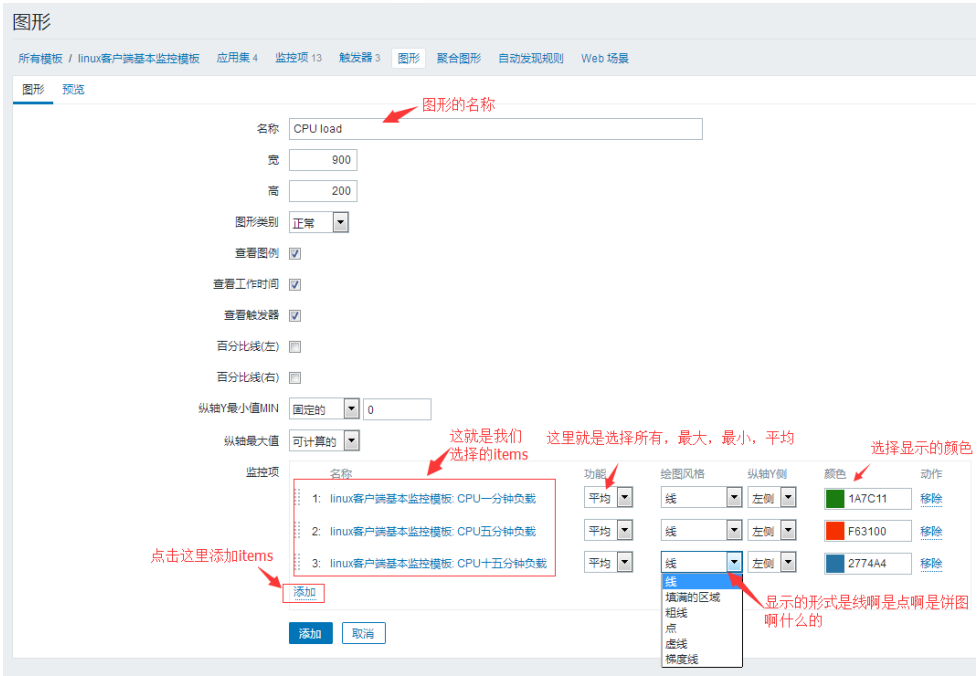
博文来自：www.51niux.com

2.6 给模板添加graphs（图形）配置

大部分数据是不需要出图了，一般CPU和网络还有程序的进程,cache的占用等一些需要图形来直观的查看和分析。

官网链接：https://www.zabbix.com/documentation/3.2/manual/config/visualisation/graphs/custom

这里我们以cpu的负载来举例：



#可以点击图形右边的预览，看看有没有问题没，没问题就点击图中最下面的添加就可以了。

图形的属性：

Name（名称）：Graph唯一的名字
Width(宽度)：图像在屏幕中的宽度（仅对饼行图/分解图有效）
Height（高度）：图像在屏幕中的高度
Graph type(图形类别)：Normal：常规图标，值用线条表示。Stacked:叠图就像cacti的流量图。Pie：饼形图。Exploded：分解饼形图。
Show legend（查看图例）：勾选会显示图标中的说明
Show working time（查看工作时间）：如果被选中，非工作时间用灰色背景显示，不能用于饼形图或分解图中
Show triggers(查看触发器):如果被选中，触发达到阀值会用红色的线条显示，不能用饼形图或是分解图表示，注意，只有部分触发器才支持在此处显示，如min、max函数可支持在图像中显示触发器的值。
Percentile line(left)(百分比线（左）)：左边的Y轴用来显示百分比，例如，给定95%，线条就会在95%的数值处，仅对常规图表适用。
Percentile line(right)（百分比线（右））：右边的Y轴用来显示百分比，例如，给定95%，线条就会在95%的数值处，仅对常规图表使用
Y axis MIN value（纵轴Y最小值MIN）：Y轴表示的最小值，Calculated-Y（可计算的）轴表示自动计算值的最小值，Fixed-Y(固定的)轴表示修正最小值不能用于饼形图或是分解饼形图，Item（监控项）表示选择的items最后一次获取的数值将作为最小值
Y axis MAX value（纵轴最大值）：Y轴表示的最大值

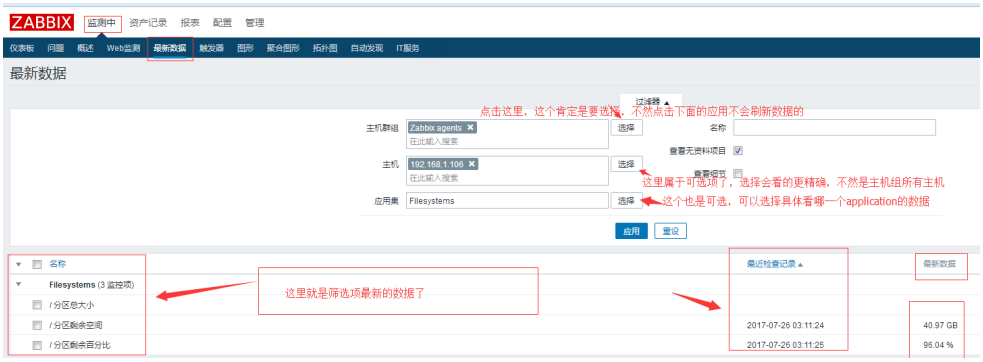
监控项里面的的参数：

Sort order(0->100)：画图时应该从0的顺序开始，可以被用来画线条或区域在另一个之后（或之前），可以在开始的线条箭头处拖放项目，以设定分类顺序或绝对将哪一个项目放在另一个项目的前面
Name(名称)：Item的名称显示的数据
Type：类型（仅对饼形图或是分解图饼形图使用），Simple:单一（简单），Graph sum:图表总数。这里上面的图因为选的不是饼形图，所以这里不显示这两个选项。
Function（功能）：当一个Item存在不止一个值时，决定显示哪一个数据，all：全部（最小值、平均值和最大值），min：仅最小值，avg：仅平均值，max：仅最大值
Draw style（绘图风格）：仅对常规图表使用，对叠图填充区域使用。Line线条：画线条，Filled region（填满的区域），Bold line(粗线)，Dot（圆点），Dashed Line（画虚线），Gradient line（梯度线）
Y axis side（纵轴Y轴）：Y轴的一侧是从左边开始还是右边开始，视图习惯一般从左边开始
Colour（颜色）：在十六进制中引用RGB（红、绿、蓝）法。

2.7 添加主机关联模板

主机的添加上面已经介绍过了，下面主机添加完后关联下模板

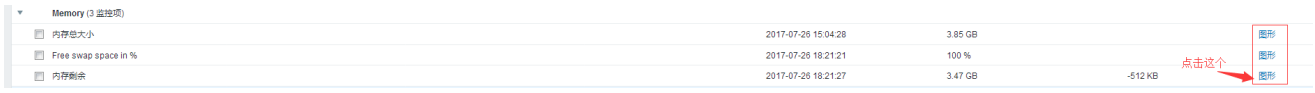
模板也关联了，zabbix_agentd也显示启动连接上了，我们看看是否能采集到新数据了吧：

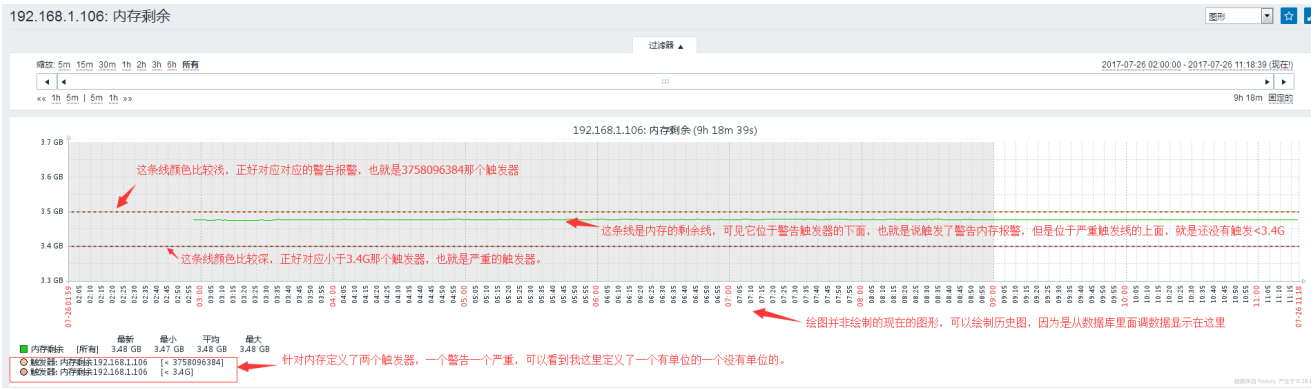


#上面关于items添加的时候utils那里已经说过了，设置的单位很重要，我们磁盘大小那里单位设置的是B，所以zabbix会自动的将收到的值除以1024换算下去，从Byte换算成了GB.内存那里也是一样的。

#去查看唯一的图形，load图形也出来了。就不截图了。

#这个图最后边还有个图形，这里没有截图，就算我们没有事先定义图形，在这里点击图形，也可以查看这个主机这个数据的绘图,如查看一个内存选项的图形：





2.8 模板关联

你说我不想自创模板太麻烦了，图省事那就关联模板白。

#你关联了模板，监控项 触发器什么的不是可以启用和禁用吗，那你就把不想用的禁用掉吧，但是你禁用的是你所关联模板里面的选项，所以如果不想监控那么多项还是自创模板吧灵活一点。

#注意：上图中有个取消链接和取消链接并清理，这个要注意一下，取消链接的话模板会去掉但是应用级什么都会留下，而要什么都不留下的话就要选择取消链接并清理。

2.9 宏（Macros）定义

宏介绍：

宏是一种抽象（Abstraction），它根据一系列预定义的规则替换一定的文本模式，而解释器或编译器在遇到宏时会自动进行这一模式替换。（其实就是变量）类似地，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宏的命名规范：大写字母、0-9、下划线，只能大写字母开头。zabbix有很多内置宏前面已经提到了，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。

宏的优先级：

zabbix有全局宏，模板宏，主机宏。
优先级别：
主机宏(checked first)
主机模板定义的宏,如果有多个模板，那么按照模板（ID）越靠前那么宏的优先级越高
全局宏(checked last)

定义全局宏：

管理（Administration）=》一般(General)=》宏（Macrps）（右上角下拉框）

定义模板宏：

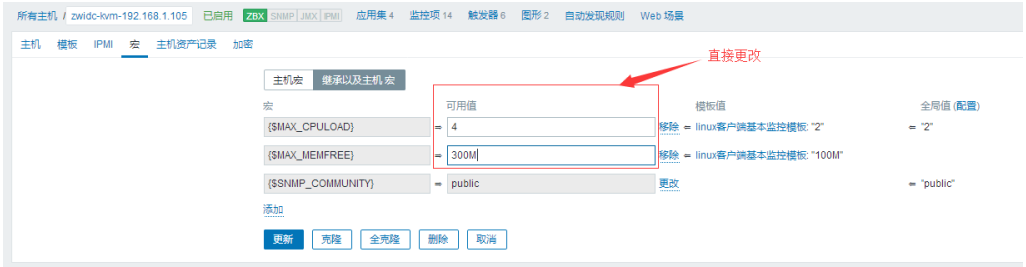
#上面两个全局宏并非单纯的引用，我箭头标注了可以更改，你点击更改，然后这个宏就非全局宏了，就变成模板自己的宏了，你把这个宏删除了，就又变回全局宏了，这就是优先级的问题了。

不管是全局宏还是模板宏，最后还是要被其他的地方引用才有它的价值，我来举例，比如我大部分的虚拟机都是2核CPU4G内存，那么他们的cpu负载值一般设置成2啊，然后内存一般剩余多少MB就该触发器了，触发器一般定义在模板里面，那么这时候有个4核8G的虚拟机，那么他们的cpu负载就非2就应该是4了，那么你主机如果是自定义的触发器当然你可以改这个值了，但是一般触发器都是从模板继承的，你不能改，只能在模板的触发器里面改，但是模板里面的触发器把触发值改成了4，哪些2核CPU的就又不适用了。宏的价值就体现出来了。触发器来引用这个宏，如果个别主机有特殊情况，自己设置个宏就OK了。下面让我们来看例子：

#从上面两张数据库中可以看到触发器引用的宏已经生效了。

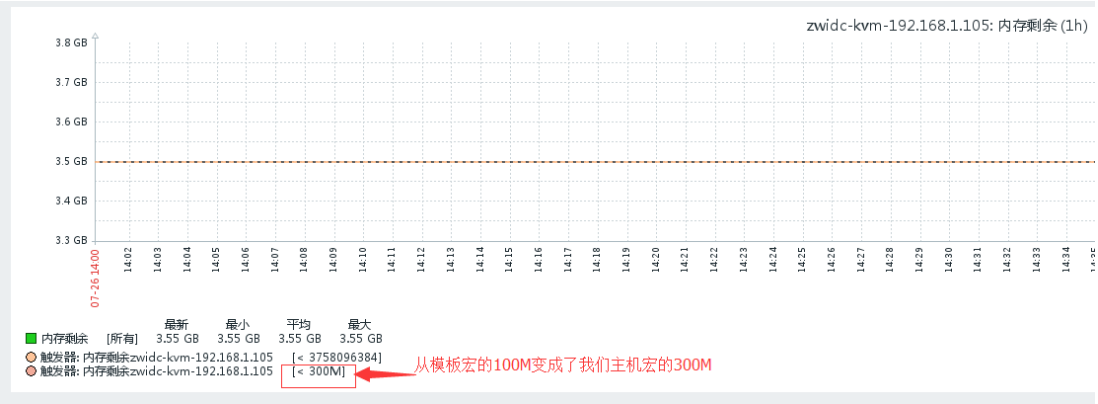
主机宏：

比如现在我有一台新机器了是4C8G的，我也继承了我现在定义的linux客户端基本监控模板，那么一开始我的触发器里面的CPU负载肯定是2内存负载肯定是100M，这个可以自行查看一下。那么我要更改了。



#直接在继承宏那里将这两个值改变一下。

#然后就变成我们自己主机的宏了，当然主机宏的添加就照着上图这个来就行了。



« 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

- 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]