QQ 972322817 itnihao@qq.com

ZABBIX 使用手册

QQ 972322817 itnihao@qq.com

文档信息
文档名：zabbix 使用手册
当前版本：v1.4
作者：itnihao
版权：GPL
Mail：admin@itnihao.com
文档修订信息

时间	更新内容
2012-12-07	初创时间 V1.0
2013-01-23	增加自动发现 V1.1
2013-02-08	增加部分文档说明 V1.2
2013-02-11	修正部分错误 V1.2
2013-05-22	增加自定义监控项的例子，邮件报警，触发器等内容 性能优化等内容 V1.3
2013-06-06	增加监控报警分组，一些使用技巧 rpm 打包 v1.4
	后续将继续更新 proxy，优化，api 等内容

本文档涉及内容，zabbix 的安装配置，zabbix 的邮件报警，zabbix 自定义脚本，自动化配置由于 zabbix 功能很丰富，许多功能需要逐步研究整理，后续本文档将继续完善。

1. Zabbix 简介

Zabbix 是一个高度集成的网络监控解决方案，可以提供企业级的开源分布式监控解决方案，由一个国外的团队持续维护更新，软件可以自由下载使用，运作团队靠提供收费的技术支持赢利。

官方网站：http://www.zabbix.com

目前最新版本为 2.0.X,后续版本 2.1，2.2 目前正在开发之中

Zabbix 2.0 官方文档：http://www.zabbix.com/documentation/2.0/manual

Zabbix 通过 C/S 模式采集数据，通过 B/S 模式在 web 端展示和配置。

被监控端：主机通过安装 agent 方式采集数据，网络设备通过 SNMP 方式采集数据

Server 端：通过收集 SNMP 和 agent 发送的数据，写入数据库（MySQL，ORACLE 等），再通过 php+apache 在 web 前端展示。

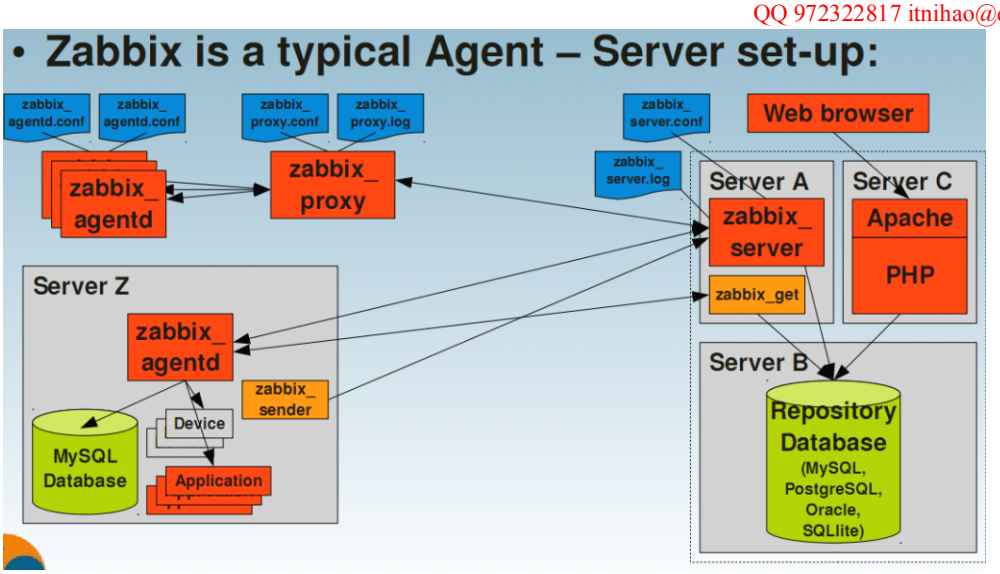
Zabbix 运行条件：

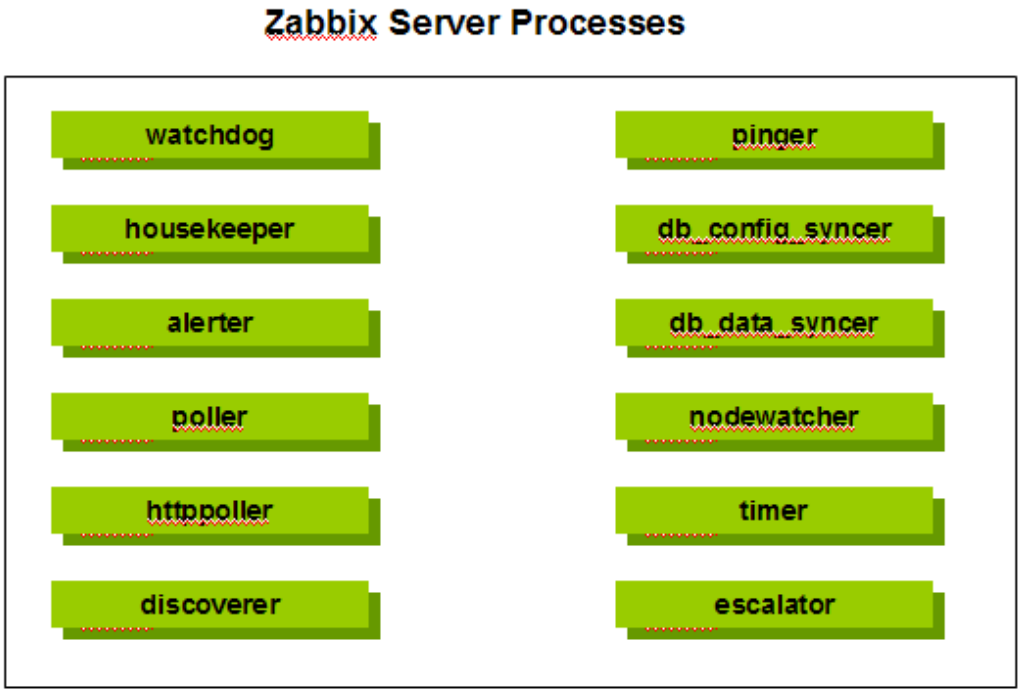
Server：Zabbix Server 需运行在 LAMP（Linux+Apache+Mysql+PHP）环境下（或者 LNMP），对硬件要求低

Agent：目前已有的 agent 基本支持市面常见的 OS，包含 Linux、HPUX、Solaris、Sun、windows

SNMP：支持各类常见的网络设备

监控过程逻辑如图示：





1.1 Zabbix 功能

具备常见的商业监控软件所具备的功能（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）

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器规则来实现）

支持自动发现（low discovery）key 实现动态监控项的批量监控（需写脚本）

支持分布式，能集中展示、管理分布式的监控点

扩展性强，server 提供通用接口（api 功能），可以自己开发完善各类监控（根据相关接口编写程序实现）

编写插件容易，可以自定义监控项，报警级别的设置。

数据收集

可用和性能检测
支持 snmp(包括 trapping and polling)，IPMI，JMX，SSH，TELNET
自定义的检测
自定义收集数据的频率
服务器/代理和客户端模式

灵活的触发器
您可以定义非常灵活的问题阈值，称为触发器，从后端数据库的参考值

高可定制的报警
发送通知，可定制的报警升级，收件人，媒体类型
通知可以使用宏变量有用的变量
自动操作包括远程命令

实时的绘图功能
监控项实时的将数据绘制在图形上面

WEB 监控能力
ZABBIX 可以模拟鼠标点击了一个网站，并检查返回值和响应时间

Api 功能
应用 api 功能，可以方便的和其他系统结合，包括手机客户端的使用。

更多功能请查看
https://www.zabbix.com/documentation/2.0/manual/introduction/features

1.2 优劣势

优点：

开源，无软件成本投入
Server 对设备性能要求低
支持设备多，自带多种监控模板
支持分布式集中管理，有自动发现功能，可以实现自动化监控
开放式接口，扩展性强，插件编写容易
当监控的 item 比较多服务器队列比较大时可以采用被动状态，被监控客户端主动从 server 端去下载需要监控的 item 然后取数据上传到 server 端。这种方式对服务器的负载比较小。
Api 的支持，方便与其他系统结合

缺点：

需在被监控主机上安装 agent，所有数据都存在数据库里，产生的数据据很大,瓶颈主要在数据库。

2. 安装部署

Zabbix Server 可以运行在 CentOS、RedHat Linux、Debain 等 Linux 系统上，这里以 centos6.3_X64 作为部署环境。

预先配置好 yum。

2.1 服务端安装 lamp 环境。

```
yum -y install gcc gcc-c++ autoconf httpd php mysql mysql-server php-mysql
httpd-manual mod_ssl mod_perl mod_auth_mysql php-gd php-xml php-mbstring php-ldap
php-pear php-xmlrpc php-bcmath mysql-connector-odbc mysql-devel libdbi-dbd-mysql
net-snmp-devel curl-devel
```

2.2 服务端配置 lamp 使用环境

修改 php.ini

```
shell#vim /etc/php.ini
```
(注意，这里必须修改，不然后面安装会提示环境不符)

```
date.timezone = Asia/Shanghai
max_execution_time = 300
post_max_size = 32M
max_input_time=300
memory_limit = 128M
mbstring.func_overload = 2
```

如果不想手工修改，可以使用以下 sed 命令操作

```
sed -i "s/;date.timezone =/date.timezone = Asia\/Shanghai/g" /etc/php.ini
sed -i "s#max_execution_time = 30#max_execution_time = 300#g" /etc/php.ini
sed -i "s#post_max_size = 8M#post_max_size = 32M#g" /etc/php.ini
sed -i "s#max_input_time = 60#max_input_time = 300#g" /etc/php.ini
sed -i "s#memory_limit = 128M#memory_limit = 128M#g" /etc/php.ini
sed -i "/;mbstring.func_overload = 0/ambstring.func_overload = 2\n" /etc/php.ini
```

开启 httpd，mysqld 服务，

```
shell#chkconfig mysqld on
sehll#chkconfig httpd on
sehll#service mysqld start
sehll#service httpd start
```

2.3 服务端 server 的安装过程

zabbix 一键安装脚本，可以实现安装过程全自动化，无需人工干预。

下载 zabbix 安装包，这里以 2.0.3 为例（建议大家下载当前最新版本），

http://www.zabbix.com/download.php

Zabbix 2.0 (Stable)

Package	Release	Date	Release Notes	Zabbix Manual	Download
Zabbix Sources Server, Proxy, Agent, GUI	2.0.3	03 October, 2012			Download

http://sourceforge.net/projects/zabbix/files/ZABBIX%20Latest%20Stable/

2.3.1 安装 zabbix 服务端

```
shell#http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/2.0.3/zabbix-2.0.3.tar.gz
增加 zabbix 用户
shell#groupadd zabbix -g 201
shell#useradd -g zabbix -u 201 -m    zabbix
shell#tar xvf zabbix-2.0.3.tar.gz
shell#./configure    --prefix=/usr/local/zabbix    --enable-server    --enable-proxy    --enable-agent
--with-mysql=/usr/bin/mysql_config --with-net-snmp --with-libcurl
shell#make
shell#make install
```

2.3.2 添加 zabbix 到系统服务文件

```
shell#vim    /etc/services
zabbix-agent     10050/tcp                  #Zabbix Agent
zabbix-agent     10050/udp                  #Zabbix Agent
zabbix-trapper   10051/tcp                  #Zabbix Trapper
zabbix-trapper   10051/udp                  #Zabbix Trappe
```

```
[root@kx1d ~]# tail  -n4 /etc/services
zabbix-agent    10050/tcp                   #Zabbix Agent
zabbix-agent    10050/udp                   #Zabbix Agent
zabbix-trapper  10051/tcp                   #Zabbix Trapper
zabbix-trapper  10051/udp                   #Zabbix Trapper
[root@kx1d ~]#
```

2.3.3 导入 zabbix 数据库（此处采用 mysql 数据库）

```
Shell#cd     PATH/zabbix-2.0.3     (确保路径在 zabbix 源码下面)
shell#mysqladmin -uroot    password    'mysql_pass';    （设置 mysql 的 root 密码）
Shell#mysql    -uroot    -p    （登陆数据库）
mysql>create    database    zabbix    character    set    utf8;
mysql>grant    all    privileges    on    zabbix.*    to    zabbix@localhost    identified    by    'zabbix';
mysql>flush    privileges;
#导入 zabbix 数据库
```

```
shell#mysql -uzabbix -pzabbix zabbix < ./database/mysql/schema.sql
shell#mysql -uzabbix -pzabbix zabbix < ./database/mysql/images.sql
shell#mysql -uzabbix -pzabbix zabbix < ./database/mysql/data.sql
```

确保以上过程无误

#创建链接

shell#mkdir /var/log/zabbix

sehll#chown zabbix.zabbix /var/log/zabbix

shell#ln -s /usr/local/zabbix/etc /etc/zabbix

shell#ln -s /usr/local/zabbix/bin/* /usr/bin/

shell#ln -s /usr/local/zabbix/sbin/* /usr/sbin/

2.3.4 拷贝 service 启动脚本

shell#cp misc/init.d/fedora/core/zabbix_* /etc/init.d/

shell#chmod 755 /etc/init.d/zabbix_*

shell#sed -i "s#BASEDIR=/usr/local#BASEDIR=/usr/local/zabbix#g" /etc/init.d/zabbix_server

shell#sed -i "s#BASEDIR=/usr/local#BASEDIR=/usr/local/zabbix#g" /etc/init.d/zabbix_agentd

2.3.5 配置 zabbix_server.conf 服务端文件

DBName=zabbix 数据库名称

DBUser=zabbix 数据库用户

DBPassword=zabbix 数据库密码

```
[root@kx1d zabbix-2.0.3]# cat /etc/zabbix/zabbix_server.conf|grep -v "^#"|grep -v "^$"
LogFile=/var/log/zabbix_server.log
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
[root@kx1d zabbix-2.0.3]#
```

Sed 命令如下

```
shell#sed -i "s/DBUser\=root/DBUser\=zabbix/g" /etc/zabbix/zabbix_server.conf
shell#sed -i "/# DBPassword=/aDBPassword=zabbix\n" /etc/zabbix/zabbix_server.conf
shell#sed-i "s#tmp/zabbix_server.log#var/log/zabbix/zabbix_server.log#g"
/etc/zabbix/zabbix_server.conf
```

2.3.6 配置 zabbix_agentd.conf 文件（监控 server 本身，如对其他服务器进行监控，配置文件相同）

注意：zabbix_agentd.conf 是客户端的配置文件，这里配置的目的是对自身进行监控

修改 4 处

Server=127.0.0.1 此处添加服务端的 ip，如服务器不为本机，则需要填写远端 zabbix_server 的 ip 地址

ServerActive=127.0.0.1 此处修改为服务端的 ip

/tmp/zabbix_agentd.log 修改日志路径

UnsafeUserParameters=0 默认是不启用自定义脚本功能的，要自定义 key，需开启，设置为 1

Include=/etc/zabbix/zabbix_agentd.conf.d/ 自定义的 agentd 配置文件可以写在这个目录下面

```
[root@kx1d zabbix-2.0.3]# cat /etc/zabbix/zabbix_agentd.conf|grep -v "^#"|grep -v "^$"
LogFile=/var/log/zabbix_agentd.log
Server=127.0.0.1,192.168.1.89
ServerActive=192.168.1.89:20051
Hostname=kx1d.localhost
UnsafeUserParameters=1
```

说明，其中的 192.168.1.89 为 zabbix 服务端的 ip（图中的 20051 为 10051）

使用 sed 命令如下

```
shell#sed -i "s/Server\=127.0.0.1/Server\=127.0.0.1,192.168.1.89/g" /etc/zabbix/zabbix_agentd.conf
shell#sed -i "s/ServerActive\=127.0.0.1/ServerActive\=192.168.1.89:10051/g" /etc/zabbix/zabbix_agentd.conf
shell#sed -i "s#tmp/zabbix_agentd.log#var/log/zabbix/zabbix_agentd.log#g" /etc/zabbix/zabbix_agentd.conf
shell#sed -i "#UnsafeUserParameters=0#aUnsafeUserParameters=1\n" /etc/zabbix/zabbix_agentd.conf
```

2.3.7 拷贝网页文件到 apache 目录

```
shell#cp -r ./frontends/php/ /var/www/html/zabbix
shell#chown -R apache.apache /var/www/html/zabbix
```

2.3.8 设置 zabbix 开机启动

```
shell#start zabbix serivce
shell#chkconfig zabbix_server on
shell#chkconfig zabbix_agentd on
shell#service zabbix_server start
shell#service zabbix_agentd start
```

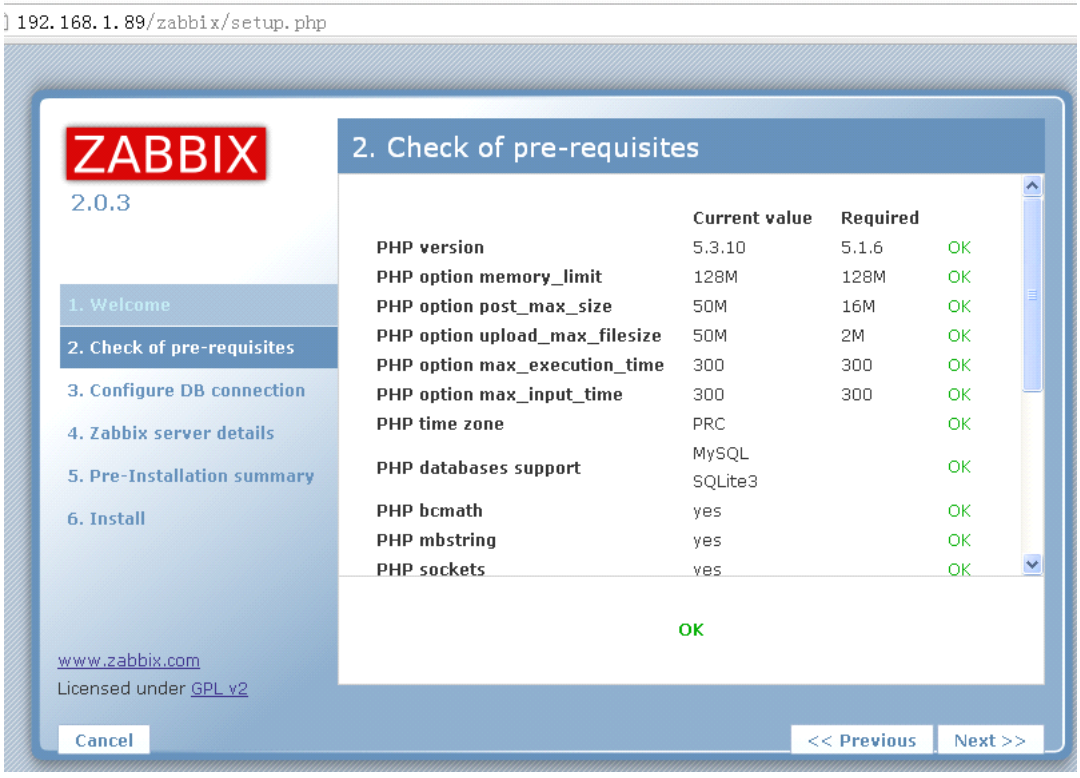
至此，zabbix 的 server 端安装完成。

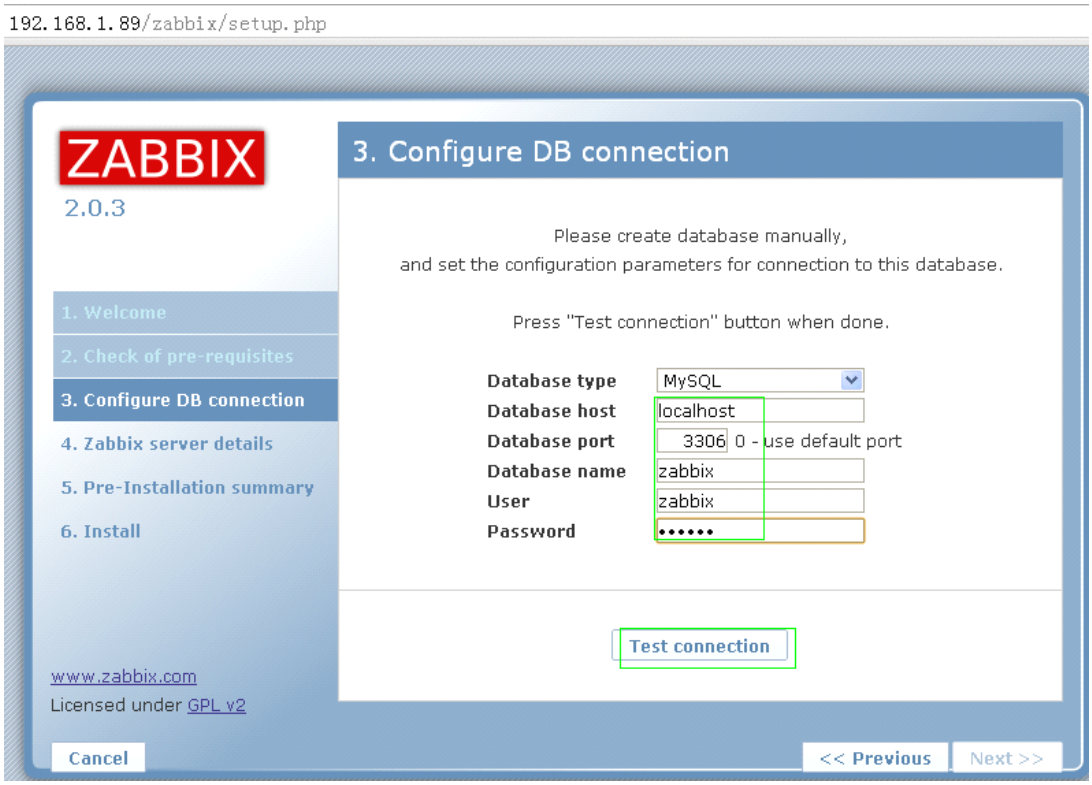
下面开始通过 web 页面进行配置服务端

2.4 通过 web 页面配置 zabbix

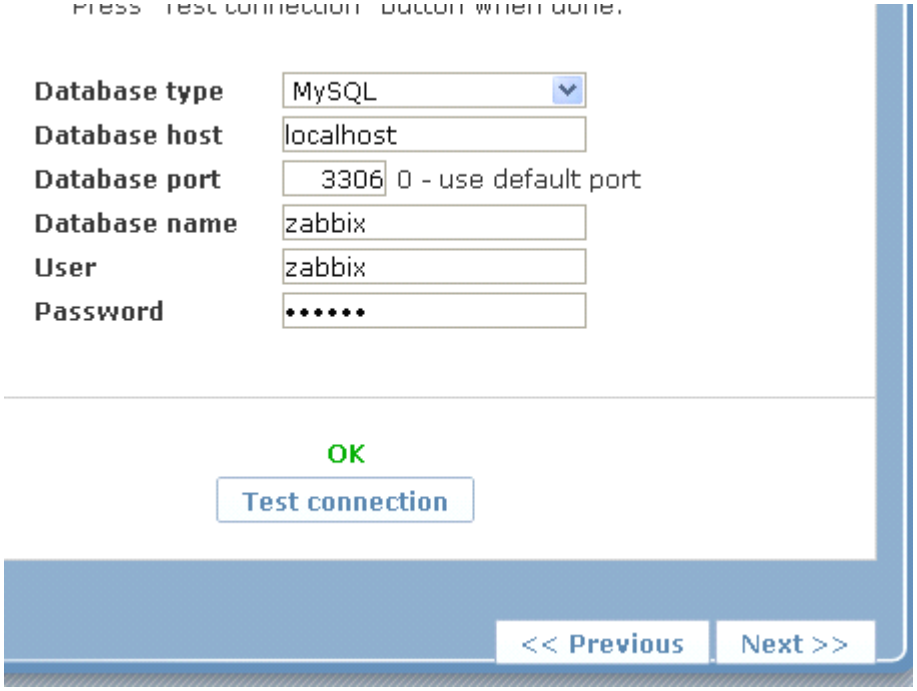
在浏览器输入 http://192.168.1.89/zabbix


192.168.1.89/zabbix/setup.php?ddreset=1&sid=9018900ca179a964
ZABBIX
2.0.3
1. Welcome
2. Check of pre-requisites
3. Configure DB connection
4. Zabbix server details
5. Pre-Installation summary
6. Install
Welcome to Zabbix2.0 setup
www.zabbix.com
Licensed under GPL v2
Cancel
Next >>


192.168.1.89/zabbix/setup.php
ZABBIX
2.0.3
1. Welcome
2. Check of pre-requisites
3. Configure DB connection
4. Zabbix server details
5. Pre-Installation summary
6. Install
2. Check of pre-requisites
Current value
Required
PHP version 5.3.10 5.1.6 OK
PHP option memory_limit 128M 128M OK
PHP option post_max_size 50M 16M OK
PHP option upload_max_filesize 50M 2M OK
PHP option max_execution_time 300 300 OK
PHP option max_input_time 300 300 OK
PHP time zone PRC OK
PHP databases support MySQL SQLite3 OK
PHP bcmath yes OK
PHP mbstring yes OK
PHP sockets yes OK
OK
www.zabbix.com
Licensed under GPL v2
Cancel
<< Previous
Next >>



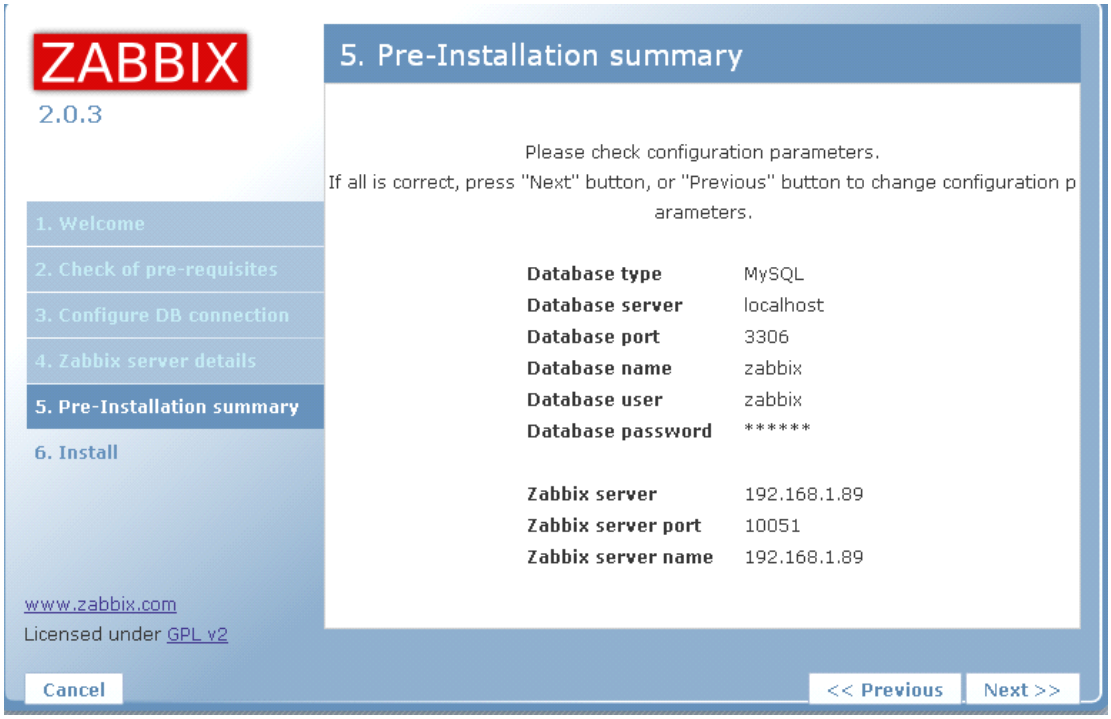
数据库 ip 为 localhost
端口为 3306
数据库名 zabbix
用户为 zabbix
密码为 zabbix
点击 test connection，如果没有问题，则会提示 OK，否则，需要检查数据库授权是否正确



QQ 972322817 itnihao@qq.com

Host 本机的 ip

Name 本机的 ip



如果此处提示文件 zabbix.conf.php 无法创建，则是 apache 目录无法写入，/var/www/html/zabbix 的文件权限不为 apche.apache

输入以下命令解决

```
chown -R apache.apache    /var/www/html/zabbix
```

默认用户名为 admin，密码为 zabbix

2.5 客户端 agentd 的安装过程

2.5.1 linux 服务器客户端的安装

在安装客户端的时候，软件包和服务器是同一个，只是 configure 的配置参数不同而已。

```
shell#groupadd zabbix -g 201
shell#useradd -g zabbix -u 201 -m zabbix
shell#tar xvf zabbix-2.0.3.tar.gz
shell#./configure --prefix=/usr/local/zabbix --enable-agent
shell#make
shell#make install
shell#mkdir /var/log/zabbix
shell#chown zabbix.zabbix /var/log/zabbix
shell#cp    misc/init.d/fedora/core/zabbix_agentd /etc/init.d/
shell#chmod 755 /etc/init.d/zabbix_agentd
shell#sed -i "s#BASEDIR=/usr/local#BASEDIR=/usr/local/zabbix#g" /etc/init.d/zabbix_agentd
shell#ln -s /usr/local/zabbix/etc    /etc/zabbix
shell#ln -s /usr/local/zabbix/bin/* /usr/bin/
shell#ln -s /usr/local/zabbix/sbin/* /usr/sbin/
shell#vim    /etc/services
zabbix-agent       10050/tcp                    #Zabbix Agent
zabbix-agent       10050/udp                    #Zabbix Agent
zabbix-trapper     10051/tcp                    #Zabbix Trapper
zabbix-trapper     10051/udp                    #Zabbix Trappe
shell#sed-                  i                  "s/Server\=127.0.0.1/Server\=127.0.0.1,192.168.1.89/g"
/etc/zabbix/zabbix_agentd.conf
shell#sed              -i              "s/ServerActive\=127.0.0.1/ServerActive\=192.168.1.89:20051/g"
/etc/zabbix/zabbix_agentd.conf
shell#sed              -i              "s#tmp/zabbix_agentd.log#var/log/zabbix/zabbix_agentd.log#g"
/etc/zabbix/zabbix_agentd.conf
shell#sed                  -i                  "#UnsafeUserParameters=0#aUnsafeUserParameters=1\n"
/etc/zabbix/zabbix_agentd.conf
```

```
shell#chkconfig zabbix_agentd on
shell#service zabbix_agentd start
#以下内容为 snmpd 的配置。（agentd 和 snmp 是两种不同的监控方式，如安装了 agentd，此处无需设置 snmp）
shell#yum -y install net-snmp
```

```
shell#mv /etc/snmpd/snmpd.conf /etc/snmpd/snmpd.conf.bak
shell#vim /etc/snmpd/snmpd.conf
com2sec mynetwork 192.168.1.89 public_monitor
com2sec mynetwork 127.0.0.1 public
group MyROGroup v2c mynetwork
access MyROGroup "" any noauth prefix all none none
view all included .1 80
shell#chkconfig snmpd on
shell#service snmpd restar
```

2.5.2 windows 服务器的安装

下载地址 http://www.zabbix.com/downloads/2.0.3/zabbix_agents_2.0.3.win.zip

Windows 下解压客户端包到 c:,下载修改好的 zabbix_agentd.conf 文件也放到 c:,打开 cmd 命令行，执行

```
C:>zabbix_agentd -install
```

安装后会在系统服务里添加一个 zabbix_agentd 服务，会自动开机运行

如果需要将客户端和配置文件放在其他目录，请执行

```
C:>DIR/zabbix_agentd -c DIR/zabbix_agentd.conf -install
```

启动 agentd 服务

```
C:>zabbix_agentd -start
```

或是通过管理->服务找到 zabbix_agentd 来启动

注意：zabbix_agentd.conf 配置文件和 linux 语法意思相同，此处略过。

3. Zabbix 的配置使用

通过本地浏览器访问 http://ServerIP/zabbix 来开始配置和使用 zabbix。
默认的用户名为 admin，密码是 zabbix
使用 zabbix 进行监控之前，要理解 zabbix 监控的流程。
一次完整的监控流程可以简单描述为：
Host Groups（设备组）->Hosts（设备）->Applications（监控项组）->Items（监控项）->Triggers（触发器）->Actions（告警动作）->Medias（告警方式）->User Groups（用户组）->Users（用户）

对于实际使用的时候，一般都是采用模板进行监控配置。使用过 cacti 的都知道，可以先添加主机，然后选择对应模板即可，zabbix 中同样存在此功能。

3.1 Zabbix 支持的监控方式类型

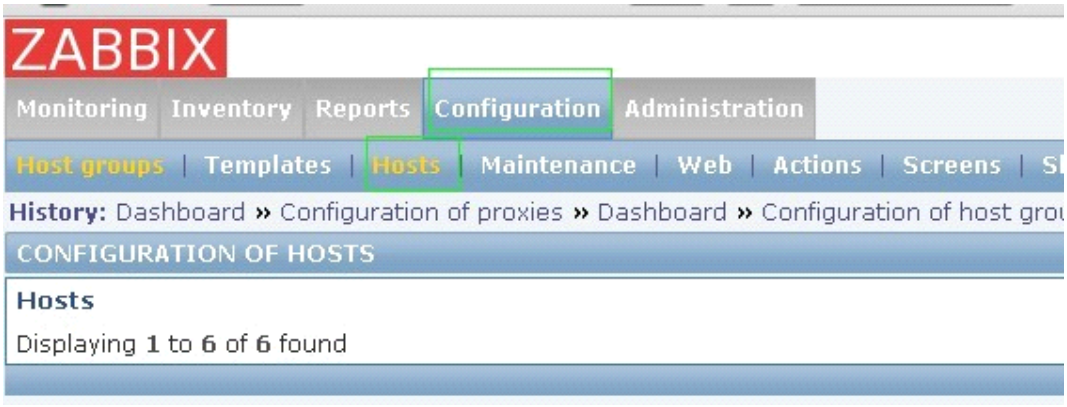
Agentd
Snmp
Jmx

ipmi

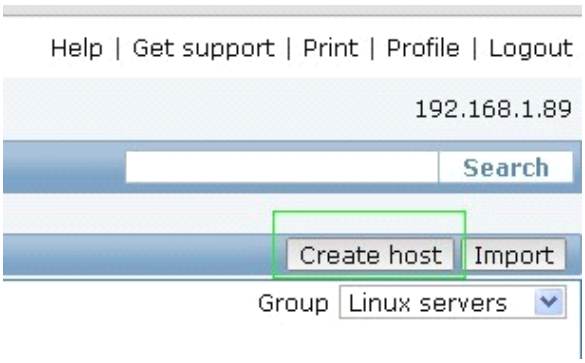
3.2 一个简单的例子--添加 Hosts，并应用模板

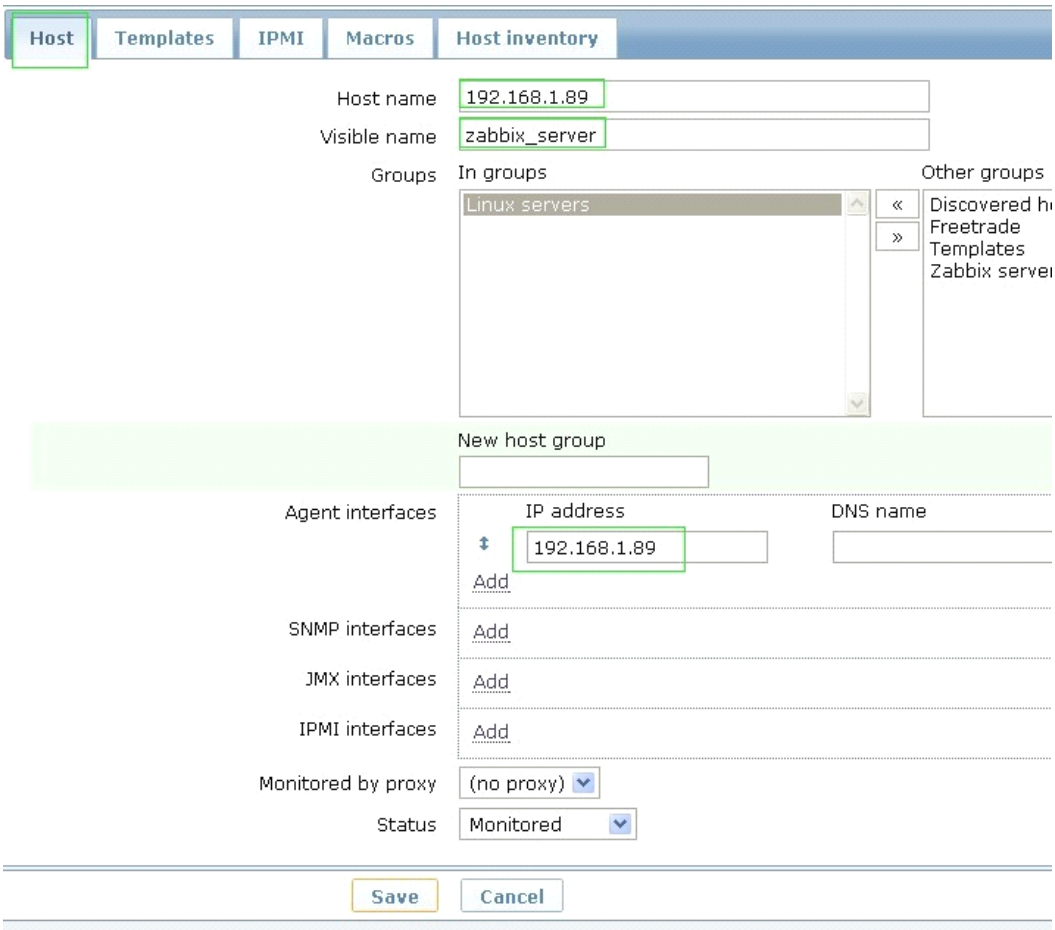
Host 是 Zabbix 监控的基本载体，所有的监控项都是基于 host 的。

通过 Configuration->Hosts->Create Host 来创建监控设备



点击右上角





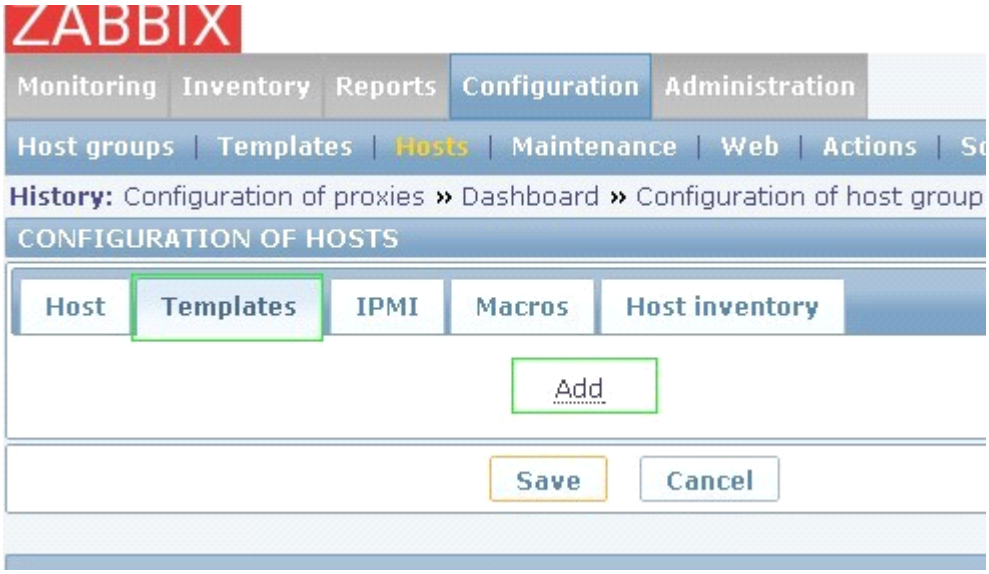
按提示填入 Name、Groups、IP ,其他选项默认即可，Link Templates 处选择一个模板，save 即可成功添加设备。（注：如果 host 上没安装 agent，添加后的状态会是 unmonitor，会采集不到值，Zabbix 自带大量的设备监控模板，我们添加主机时通过 link 到这些模板，就可以快速添加主机的监控项和告警触发条件。一旦采用 Templates 模板后，后面的步骤可以省略）

一类的 hosts 可以归属到同一个 Host Group，便于分类管理同一类设备，在 Configuration->Host Group->Create Host Group 可以添加设备组

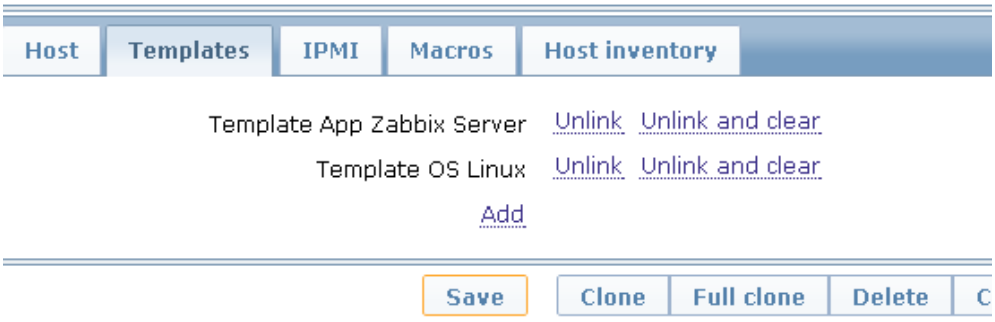
下面对各项参数进行详解

参数	描述
主机名	输入一个不重复的主机名。只允许大小写字母、数字、标点符号和下线 注意：你编辑该名称对应客户端配置文件时，主机名（hostname）这一项必须跟你在此处输入的值是一样的。在主机存活检查时需要这个名字。
访问名	你如果设置该名字，那么它将出现在主机列表,地图等地方。这个属性需要 UTF-8 支持。
群组	选择主机所属的群粗。一个主机必须属于至少一个主机组。
新主机组	一个新的群组将被创建然后自动链接到该主机上。如果空的话，该项将忽略。
接口协议	一个主机支持几种类型的主机接口协议类型:Agent, SNMP, JMX and IPMI 如果想增加一个新的接口协议，点击 Add 然后输入 IP/DNS，连接项，端口等信息
ip 地址	要监控主机的 ip 地址（可选项）
DNS 名称	要监控主机 DNS 能够解析的名称（可选项）。

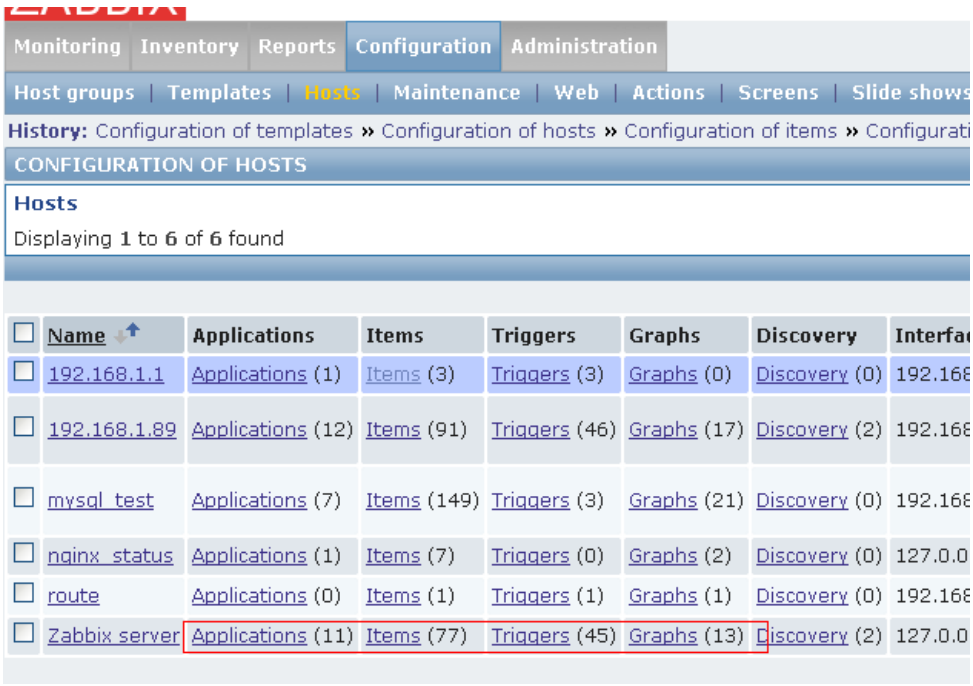
链接	点击各自对应名称的按钮将反馈给 zabbix 服务器用哪个名称（IP 或 DNS）从客户端获得数据。	IP	连接要监控主机的 IP 地址（推荐）
		DNS	链接要监控主机能够正常解析的 DNS 名称
端口	TCP 协议的端口，zabbix 客户端使用的默认值是 10050		
通过代理	主机可以通过 zabbix 服务器或者 zabbix 的一个代理来监控。		
状态	Monitored	主机是活动的，监控就绪	
	Not monitored	主机已停止，因此没被监控	



添加模板



注意，此处的模板选择后，会自动创建监控的对象。



简单的监控添加就完成了，如果需要自定义监控项的监控。则需要继续阅读以下章节。

3.3 如何自定义监控

在自定义监控之前，需要了解几个概念

Items：创建监控项，这里会运用到自定义的 key 值

Triggers：创建触发器，这里是监控项达到报警的阈值

Graphs：添加图形

自定义监控，可以在单台机器上面添加，也可以先定义成模板再把模板应用到主机上面。

首先是创建 key 值，

比如我想监控/etc/passwd 文件的行数，则，首先定义 key，

3.2.1，key 的创建

客户端配置文件如下

#grep -v "#" /etc/zabbix/zabbix_agentd.conf|grep -v "^$"

```
LogFile=/var/log/zabbix/zabbix_agentd.log    #修改日志存储路径，非必要修改
EnableRemoteCommands=0                       #允许远程执行命令，非必要修改
Server=127.0.0.1,192.168.1.254.200           #允许哪些 ip 访问本机的 key
StartAgents=8                                #启动的客户端进程数
ServerActive=192.168.1.254:10051             #被动监控，服务端的 ip
Hostname=zabbixtest.itnihao.cn               #客户端的主机名
Timeout=30                                   #超时时间
Include=/etc/zabbix/zabbix_agentd.conf.d/    #配置文件
UnsafeUserParameters=1                       #用户自定义的 key
```

#cat /etc/zabbix/zabbix_agentd.conf.d/count_line_passwd.conf

```
UserParameter=count.line.passwd,wc -l /etc/passwd|awk '{print $1}'
```

```
[root@nat ~ 18:28 ~]1#cat /etc/zabbix/zabbix_agentd.conf.d/count_line_passwd.conf
UserParameter=count.line.passwd,wc -l /etc/passwd|awk '{print $1}'
```

然后重启客户端

```
#service zabbix_agentd restart
```

Key 值创建后是否成功，可以用命令行来检测

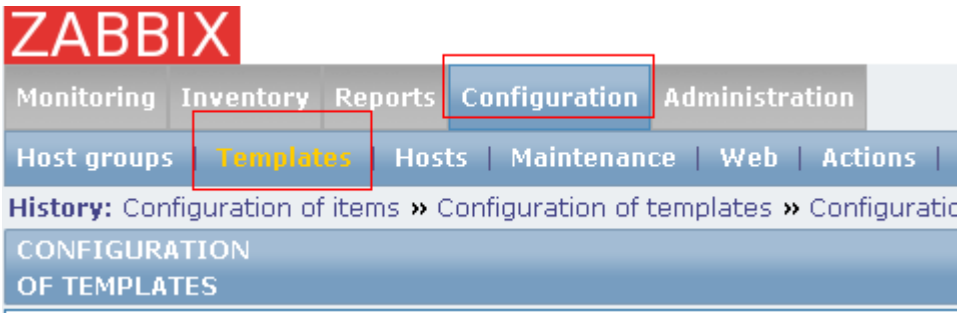
```
#zabbix_get   -s   127.0.0.1   -k   count.line.passwd
```

```
[root@nat ~ 18:28 ~]2#zabbix_get -s 127.0.0.1 -k count.line.passwd
39
```

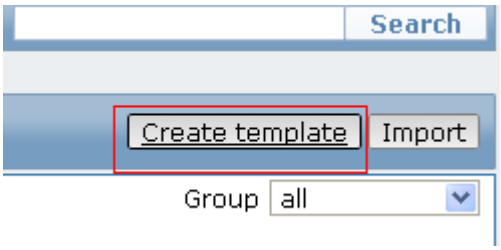
可以看到有返回值，则说明 key 值创建成功。

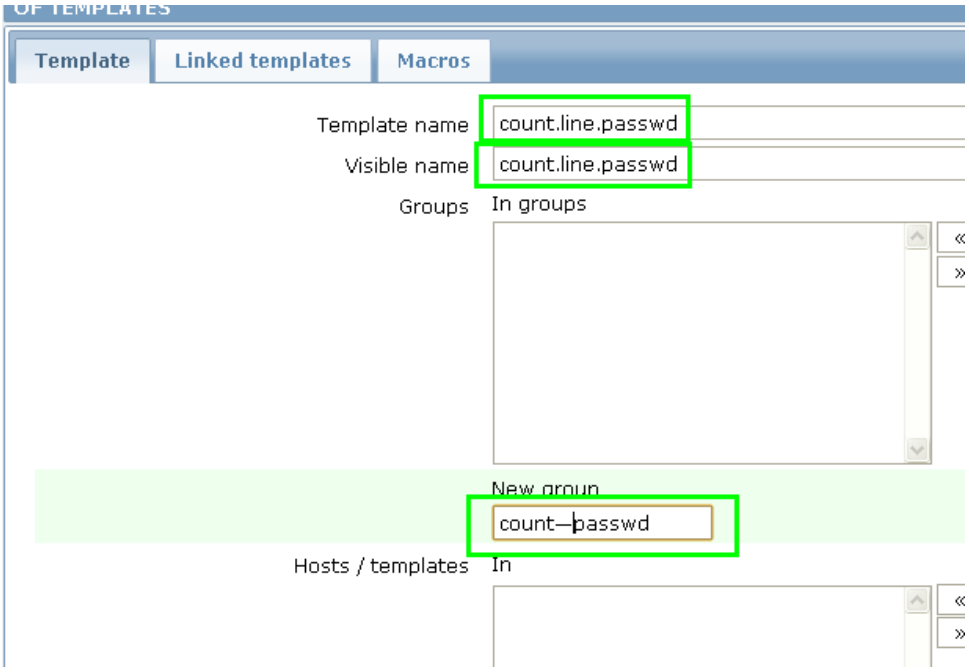
关于更多 zabbix_get 的用法，请查看帮助，此处不过多说明。
Key 值创建的更多内容，参考下一章节。

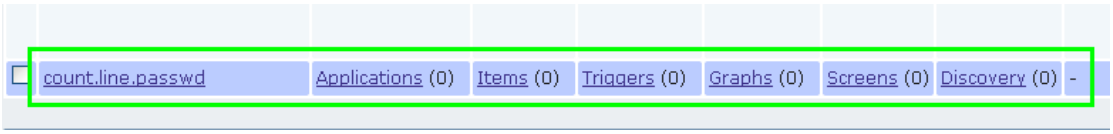
3.2.2，web 页面创建项目

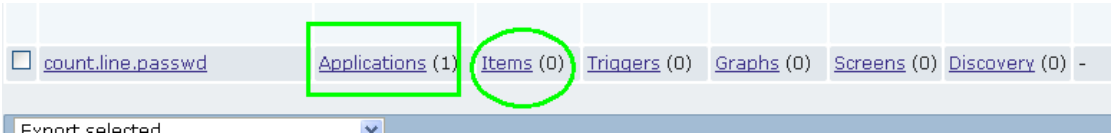


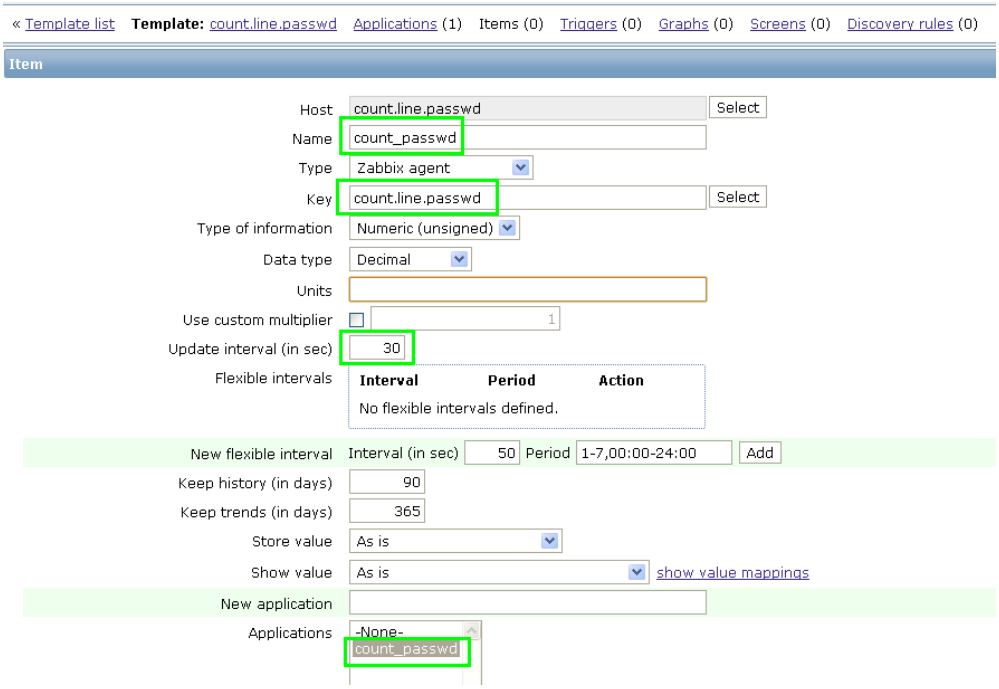
QQ 972322817 itnihao@qq.com







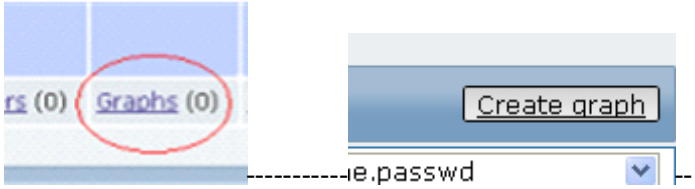


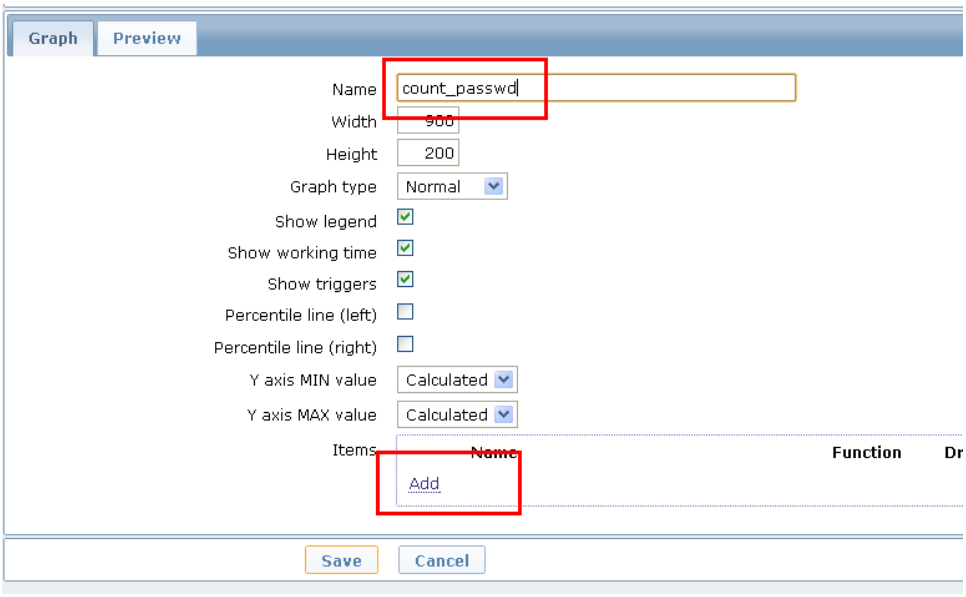


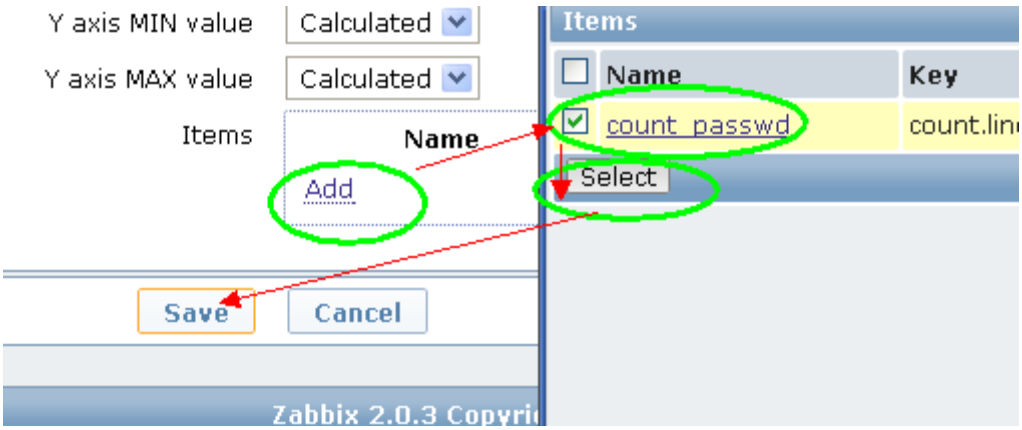
注意，这里的 key 就是前面定义的 key

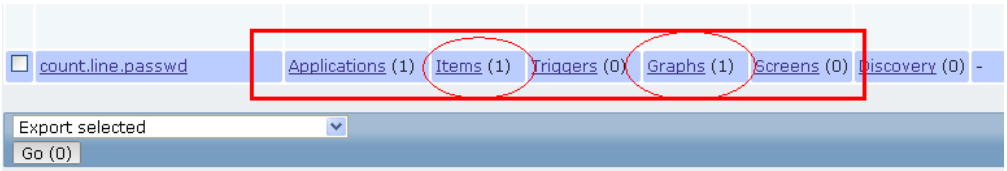
此处略去 triggers 的创建

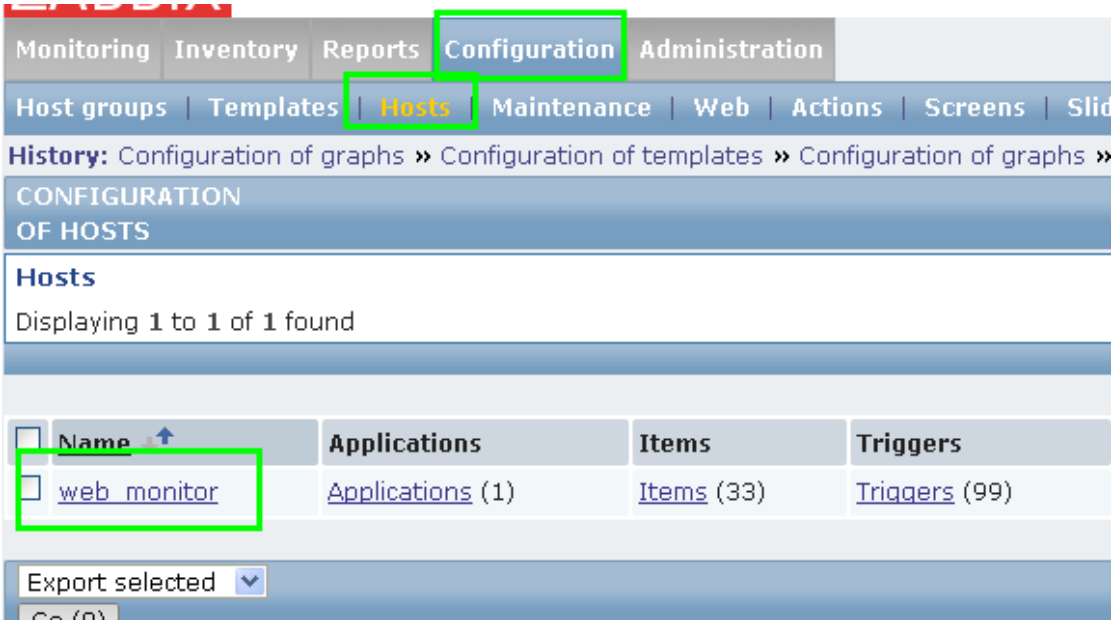
Graphs 的创建

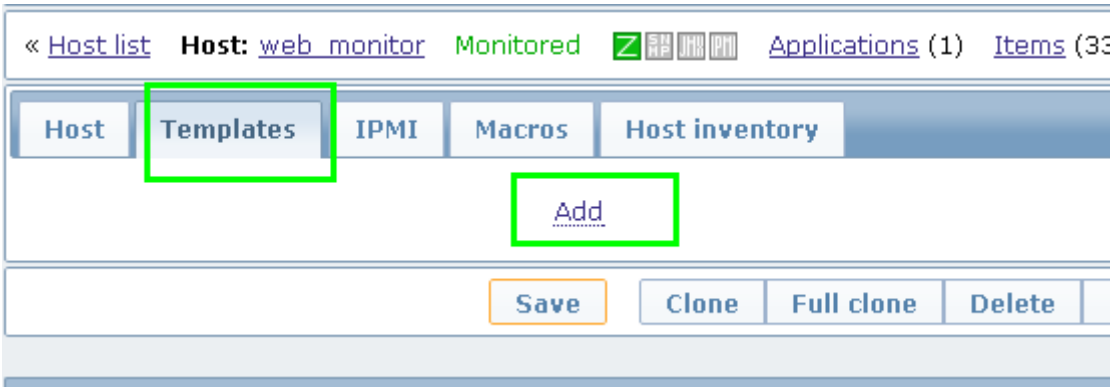


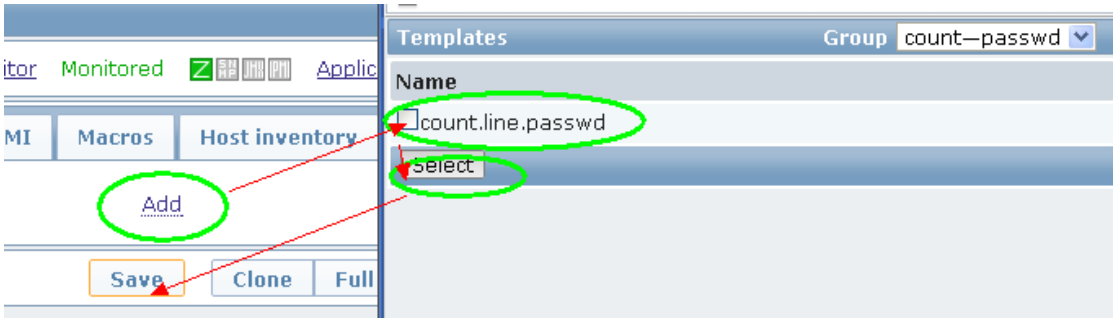

Graph
Preview
Name
count_passwd
Width
900
Height
200
Graph type
Normal
Show legend
Show working time
Show triggers
Percentile line (left)
Percentile line (right)
Y axis MIN value
Calculated
Y axis MAX value
Calculated
Items
Name
Function
Dr
Add
Save
Cancel


Y axis MIN value
Calculated
Y axis MAX value
Calculated
Items
Name
Add
Save
Cancel
Items
Name
Key
count_passwd
count.lin
Select


count.line.passwd
Applications (1)
Items (1)
Triggers (0)
Graphs (1)
Screens (0)
Discovery (0)
Export selected
Go (0)







至此，一个简单的自定义可以就完成了。
如何查看 grahps 呢。
Monitor---grahps--group--host-grahps　选择相应的图形查看即可，此处效果略

3.4 如何自定义 key

虽然上一节已经讲过了 key 如何创建，但这里还是单独列出来详解其他注意事项.
自定义 key 即 User parameters 这个功能，先看一下官方文档
https://www.zabbix.com/documentation/2.0/manual/config/items/userparameters
Key 自定义的语法格式

```
UserParameter=<key>,<command>
```

例子

在 /etc/zabbix/zabbix_agentd.conf 后面添加如下内容

```
UserParameter=get.os.type,head -1  /etc/issue
```

然后重启 zabbix_agentd

```
#service  zabbix_agentd  restart
```

运行测试命令，查看 key

```
#zabbix_get -s 127.0.0.1 -k get.os.type
```

```
[root@nat ~ 13:49 ~]4#zabbix_get -s 127.0.0.1 -k get.os.type
CentOS release 6.3 (Final)
[root@nat ~ 13:49 ~]5#
```

当然，UserParameter 的内容可以单独写一个配置文件，便于维护

需要做以下设置，修改/etc/zabbix/zabbix_agentd.conf

```
Include=/etc/zabbix/zabbix_agentd.conf.d/
```

如果你的 UserParameter 包含\ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @这些字符，则需要开启下面这个参数

修改/etc/zabbix/zabbix_agentd.conf

```
UnsafeUserParameters=1
```

关于 key 的名称定义注意事项

- 所有的数字;
- 所有的小写字母;
- 所有大写字母;
- 下划线;
- 破折号;
- 点.

2. 传递参数

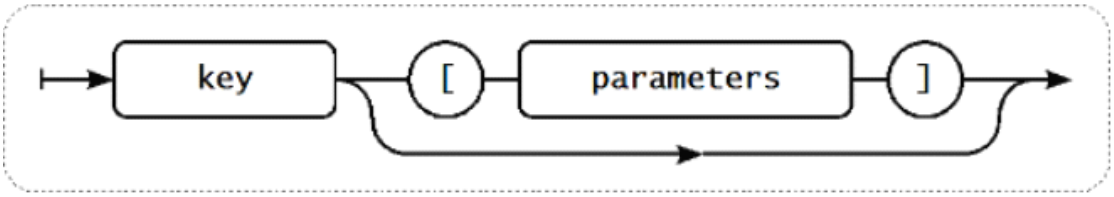
UserParameter=wc[*],grep -c "$2" $1

这里表示把$2,$1 的传递给 key，测试如下

#zabbix_get -s 127.0.0.1 -k wc[/etc/passwd,root]

注意，这里的/etc/passwd 为$2,root 为$1,则 key 最终运行的命令为 grep -c root /etc/passwd

格式如下



如果[]中括号里面有多个参数选项的值，每一个参数用用逗号隔开，如：

```
icmpping[,,200,,500]
```

3.5 添加 Items

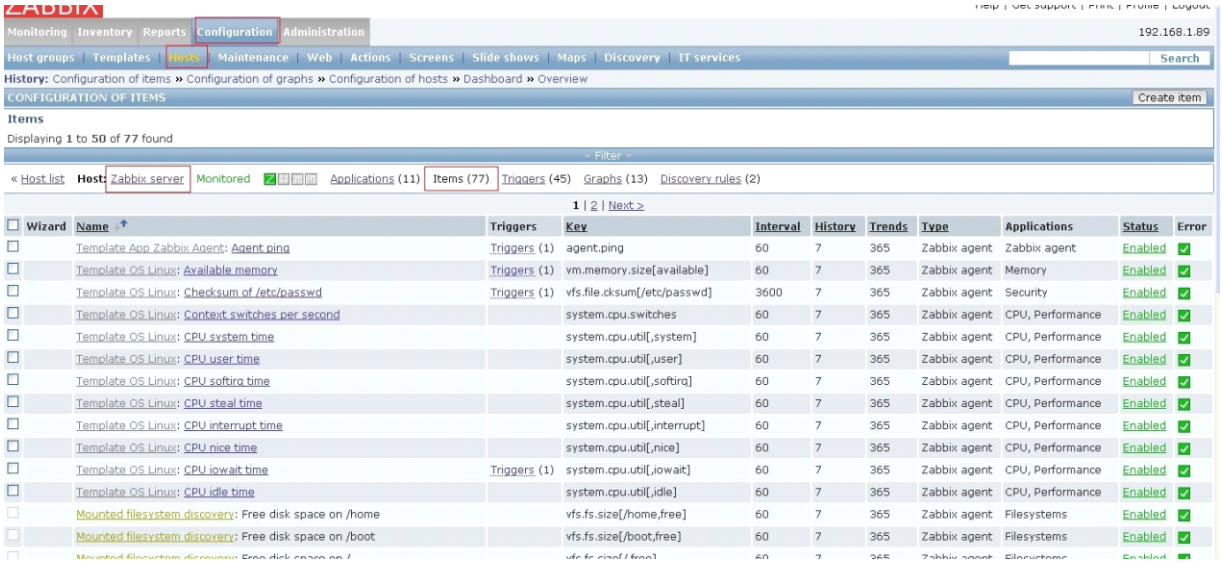
从 2.0 版本起，zabbix 支持 Agent, SNMP, JMX, and IPMI 等 4 种方式的检查

Agentd 支持的 items key 详见

Item 是监控项，是监控的基本元素，每一个监控项对应一个被监控端的采集值。

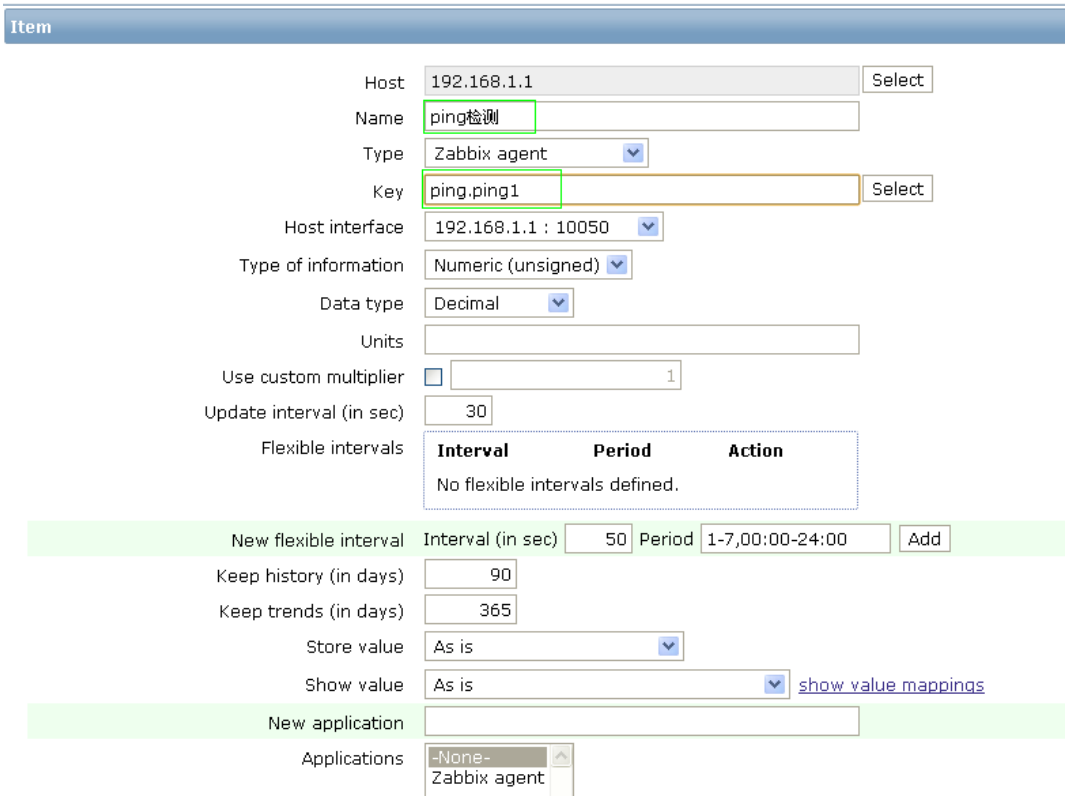
在 Configuration->Hosts 界面，我们能看到每个 host 所包含的 items 总数，点击对应主机的 items 项，可以看到具体的每个 item 信息，这些 items 可以引用自 templates，也可以自己创建。

注意：当我们需要监控的服务器的时候，一般是链接模板，如需自定义 item，则需要在此处添加 Items。



通过点击具体 item 名字可以修改已有监控项的属性，点击 Satus 的链接可以禁用/启用这个监控项。（注：我们可以通过新建一个 template，在 template 中禁用掉所有不需要用到的 items，然后把同一类 hosts link to 这个 template，就不用一台台主机去更改 items）

新增 item 可以通过点击右上角的 create item 来创建



此处的 key 值来自 zabbix_agentd.conf 文件自定义的

```
UserParameter=ping.ping1,/etc/zabbix/scripts/ping ping1
```

如不需要自定义 key，可以点击 select 直接选择

关于如何自定义监控脚本，见后文专题叙述。

关于更多 item，请参考官方文档

http://www.zabbix.com/documentation/2.0/manual/config/items

3.6 添加 Triggers

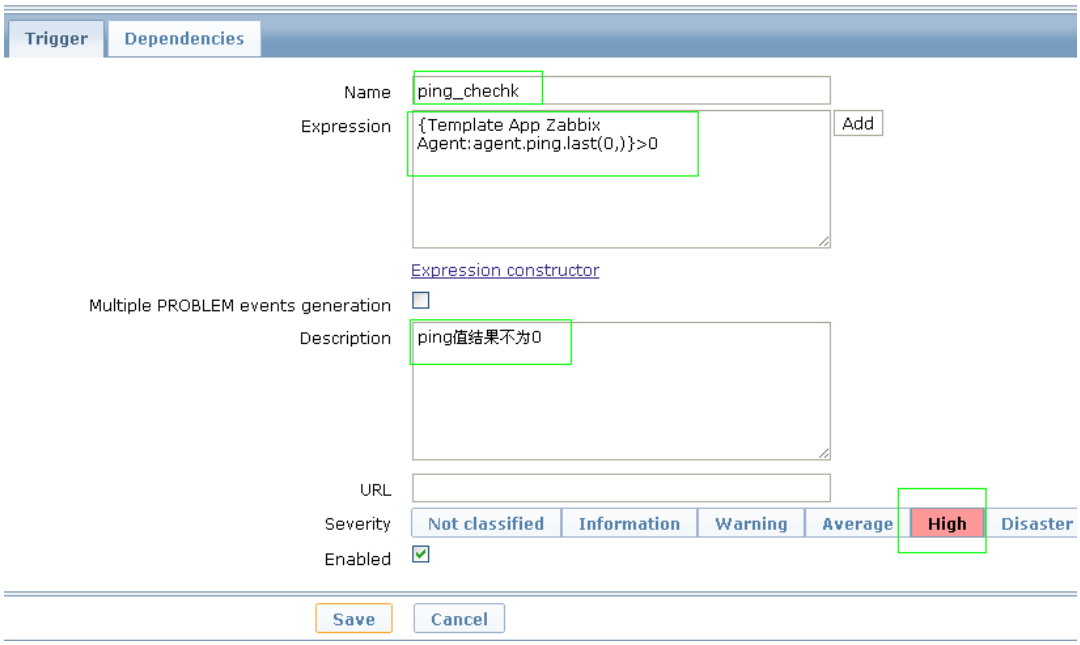
如何配置 Triggers

Trigger 是触发器，当 Items 采集值满足 triggers 的触发条件时，就会产生 actions。

每一个 trigger 必须对应一个 item，但一个 item 可以对应多个 trigger。通过设置多个 trigger，实现触发条件的不同，从而达到不同级别的报警。默认的模板中只有一个 trigger。

同样，通过点击 Configuration->Hosts->Triggers 中某个 trigger 的名字，可以修改 trigger 的属性。（注意：引用自 template 的 trigger 触发值是不能单独修改的，必须在 template 中修改，或是复制一个同样的 trigger 再修改，然后禁用掉之前的）

新增 trigger 可以通过点击右上角的 create trigger 来创建



Expression 中选择对应的 item、触发方式及触发值，Severity 是告警级别，根据 trigger 的严重性来选择。

Zabbix 提供多种 trigger 触发方式供选择，常用的我们可以选择 last value </>/=/#/N(最近一次采集值),或是选择 maximal value for period of time (一段时间内的最大值),等等。可以根据实际需要来设定触发方式。更多的解释请参考：
http://www.zabbix.com/documentation/2.0/manual/config/triggers

触发器的表达式

感谢 http://www.linuxmr.com/2012/zabbix2_0614/161.html 的翻译

概述

在触发器中使用表达式是非常灵活的。你可以用它们复杂的逻辑来测试关于监控统计。
一个简单有用的触发器可能像下面的情况：

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

1、function

触发器函数允许参考收集到的数据，当前时间和其他因素。一个已经完成的支持函数表是可用的。

2、function parameter

大多数数值型函数接受秒数作为一个变量（argument）
你可以使用前缀＃来指定一个变量（argument）有不同的意思：

函数调用（FUNCTION CALL）	意思（MEANING）
sum(600)	600秒内所有值的和

sum(#5)	最近5秒值的和

函数 last 在用 hash 标记前缀时有不同的意义 －－ 它返回给定的第 n 个值，因此给定值 3, 7, 2, 6, 5，last(#2)将返回第二个值 7，last(#5)将返回第五个值 5。
要忽略的函数也必须给它一个参数，例如：last（0）
avg, count, last, minand max 支持在某个时间段之前的，例如 avg(1h,1d)，表示 1 小时之前的 1 天的平均值。
你可以在触发器表达式中使用支持的单位符号，例如'5m'（分钟）代替'300s'（秒）或者'15'（天）代表'86400s'秒。

3、运算符

触发器支持下列运算符（优先级渐降）

优先级（PRIORITY）	运算符（OPERATOR）	定义（DEFINITION）
1	/	整除（division）
2	*	乘（Multiplication）
3	-	减（Arithmetical minus）
4	+	加（Arithmetical plus）
5	<	小于（Less than）运算符可以这样定义: A<B ⇔ (A<=B-0.000001)
6	>	大于（More than）.运算符可以这样定义: A>B ⇔ (A>=B+0.000001)
7	#	不等于（Not equal）.运算符可以这样定义: A#B ⇔ (A<=B-0.000001) | (A>=B+0.000001)
8	=	等于（Is equal). 运算符可以这样定义: A=B ⇔ (A>B-0.000001) & (A<B+0.000001)
9	&	逻辑与（Logical AND）
10	|	逻辑或（Logical OR）

4、触发器例子

例子一
www.zabbix.com 上的处理器负载太高

```
{www.zabbix.com:system.cpu.load[all,avg1].last(0)}>5
```

'{www.zabbix.com:system.cpu.load[all,avg1]'给出了监控参数的名称。它指定服务器是'www.linuxmr.com'，被监控关键字是'system.cpu.load[all,avg1]'，通过使用函数'last()'，我们指定最近的值。最后，'>5'表示来自 www.linuxmr.com 的最后负载测量大于5则触发器进入 PROBLEM 状态。

例子二

www.zabbix.com 过载了

```
{www.zabbix.com:system.cpu.load[all,avg1].last(0)}>5|{www.zabbix.
com:system.cpu.load[all,avg1].min(10m)}>2
```

无论当前处理器负载大于5还是最近10分钟的负载大于2，该表达式的值都是真

例子三

文件/etc/passwd 被更改了。

使用函数 diff：

```
{www.zabbix.com:vfs.file.cksum[/etc/passwd].diff(0)}>0
```

当文件/etc/passwd 之前的 checksum 值于最近的值不同，则该表达式为真

相似的表达式也可以用在监控重要的文件（如/etc/passwd, /etc/inetd.conf, /kernel 等）变更上

例子四

有人从因特网上下载大文件

使用函数 min：

```
{www.linuxmr.com:net.if.in[eth0,bytes].min(5m)}>100K
```

当最近 5 分钟内，eth0 接收的字节数大于 100KB，则该表达式为真。

例子五

两个 SMTP 服务器的集群节点都停止了

注意在一个表达式中使用两个不同的主机

```
{smtp1.linuxmr.com:net.tcp.service[smtp].last(0)}=0&{smtp2.linuxm
r.com:net.tcp.service[smtp].last(0)}=0
```

当 SMTP 服务器 smtp1.linuxmr.com 与 smtp2.linuxmr.com 都停止时，表达式为真

例子六

zabbix 客户端代理需要更新

使用函数 str():

```
{zabbix.linuxmr.com:agent.version.str("beta8")}=1
```

当 zabbix 客户端代理有版本 beta8时该表达式为真。

例子七

服务器不可达

```
{zabbix.linuxmr.com:icmpping.count(30m,0)}>5
```

主机 zabbix.linuxmr.com 在最近 30 分钟内超过 5 次不可达该表达式为真

例子八

最近三分钟内没有回应

使用函数 nodata（）：

```
{zabbix.linuxmr.com:tick.nodata(3m)}=1
```

'tick'必须使用类型'Zabbix trapper'。为了这个触发器工作，tick 必须定义。该主机应该使用 zabbix_sender 定期为该参数发送数据。如果 180 秒都没有收到数据，该触发器的值变为 PROBLEM。

例子九

CPU 在夜间活度

使用函数 time（）

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2&{zabbix:system.cpu.load[all,avg1].time(0)}>000000&{zabbix:system.cpu.load[all,avg1].time(0)}<060000
```

触发器只在晚上(00:00-06:00)为可用。

例子十

检查客户端本地时间是否与 zabbix 服务器时间同步

使用函数 fuzzytime():

```
{MySQL_DB:system.localtime.fuzzytime(10)}=0
```

当 MySQL_DB 的本地时间与 zabbix server 的时间相差超过 10 秒，触发器变为 PROBLEM 状态。

5、滞留状态

有时候触发器必须在不同情况下有不同条件。例如：我们想定义当服务器房间的温度超过 20 摄氏度时触发器变为 PROBLEM 状态，然后触发器一直停留在这个状态除非温度低于 15 摄氏度。

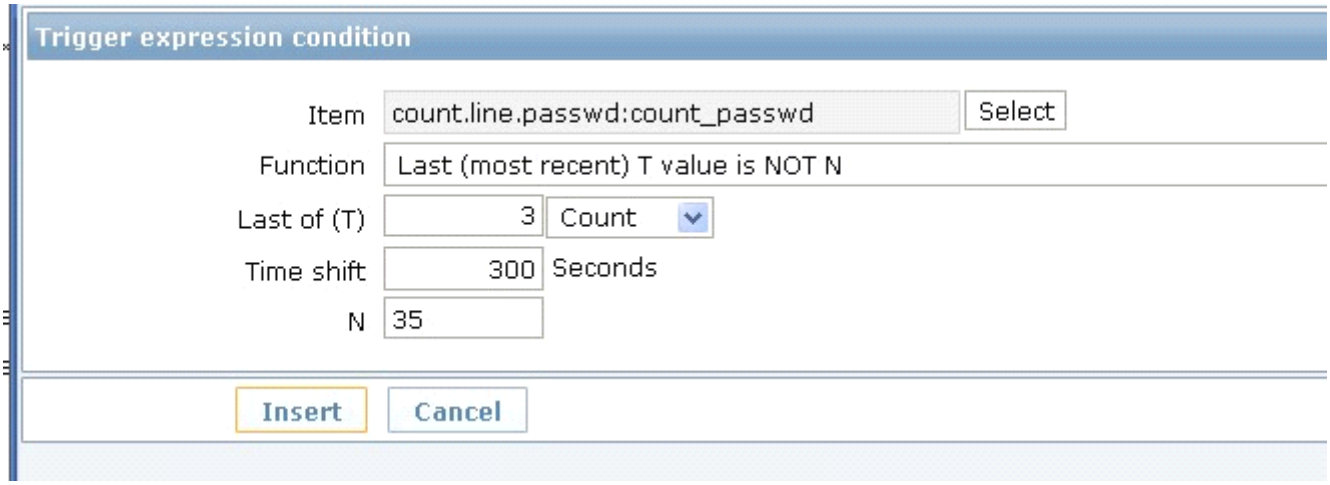
为了实现这种功能，我们定义下面的触发器。

例子一

服务器房间温度过高

```
({TRIGGER.VALUE}=0&{server:temp.last(0)}>20)|
({TRIGGER.VALUE}=1&{server:temp.last(0)}>15)
```

注意使用了一个宏{TRIGGER.VALUE}，这个宏返回当前触发器的值




```
{count.line.passwd:count.line.passwd.last(#3,300)}#35
&{count.line.passwd:count.line.passwd.last(#1,300)}#35
```

表示 300 秒内，第 1 次和第 3 次取的值同时不为 35，则触发规则

更多触发器的内容详细参考 http://pengyao.org/zabbix-triggers-functions.html，整理如下

【翻译】Zabbix 触发器支持的函数说明

2013-05-06 by pengyao

- 原文出处:
- https://www.zabbix.com/documentation/2.0/manual/appendix/triggers/functions
- 译者: pengyao

abschange

- **参数**: 直接忽略后边的参数
- **支持值类型**: float, int, str, text, log
- **描述**: 返回最近获取到的值与之前的值的差值的绝对值. 对于字符串类型，0表示值相等，1表示值不同

avg

- **参数**: 秒或#num
- **支持值类型**: float, int
- **描述**: 返回指定时间间隔的平均值. 时间间隔可以通过第一个参数通过秒数设置或收集的值的数目(需要前边加上#,比如#5表示最近5次的值) 。如果有第二个，则表示时间漂移(time shift),例如像查询一天之前的一小时的平均值，对应的函数是avg(3600,86400)，时间漂移是 Zabbix 1.8.2加入进来的

change

- **参数**: 直接忽略掉后边的参数
- **支持值类型**: float, int, str, text, log
- **描述**: 返回最近获取到的值与之前的值的差值. 对于字符串类型，0表示值相等，1表示值不同

count

- **参数**: 秒或#num
- **支持值类型**: float, int, str, text, log
- **描述**: 返回指定时间间隔内的数值统计。 时间间隔可以通过第一个参数通过秒数设置或收集的值数目（需要值前边加上#）。本函数可以支持第二个参数作为样本(pattern)数据，第三个参数作为操作(operator)参数，第四个参数作为时间漂移(time shift)参数. 对于样本， 整数(iteeger)监控项实用精确匹配，浮点型(float)监控项允许偏差0.0000001

支持的操作(operators)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

对于整数和浮点型监控项目支持 eq(默认), ne, gt, ge, lt, le；对于 string、text、log 监控项支持 like(默认), eq, ne

例子:

count(600): 最近10分钟的值的个数

count(600,12): 最近10分钟，值等于12的个数

count(600,12,"gt"): 最近10分钟，值大于12的个数

count(#10,12,"gt"): 最近的10个值中，值大于12的个数

count(600,12,"gt",86400): 24小时之前的前10分钟数据中，值大于12的个数

count(600,,,86400): 24小时之前的前10分钟数据的值的个数

#num 参数从 Zabbix 1.6.1起开始支持, time shift 参数和字符串操作支持从 Zabbix 1.8.2开始支持

date

- **参数**: 直接忽略掉后边的参数
- **支持值类型**: 所有(any)
- **描述**: 返回当前日期(格式为 YYYYMMDD)，例如20031025

dayofmonth

- **参数**: 直接忽略掉后边的参数
- **支持值类型**: 所有(any)
- **描述**: 返回当前是本月第几天(数值范围:1-31)，该函数从 Zabbix 1.8.5起开始支持

dayofweek

- **参数**: 直接忽略掉后边的参数
- **支持值类型**: 所有(any)
- **描述**: 返回当前是本周的第几天(数值返回:1-7)，星期一是 1，星期天是7

delta

- **参数**: 秒或#num
- **支持值类型**: float, int
- **描述**: 返回指定时间间隔内的最大值与最小值的差值(max()-min())。时间间隔作为第一个参数可以是秒或者收集值的数目．从 Zabbix 1.8.2开始，支持可选的第二个参数 time_shift.

diff

- **参数**: 忽略
- **支持值类型**: float, int, str, text, log
- **描述**: 返回值为1 表示最近的值与之前的值不同，0为其他情况

fuzzytime

- **参数**: 秒
- **支持值类型**: float, int
- **描述**: 返回值为1表示监控项值的时间戳与 Zabbix Server 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 Zabbix server 时间相同.

iregexp

- **参数**: 第一个为字符串，第二个为秒或#num
- **支持值类型**: str, log, text
- **描述**: 与 **regexp** 类似，区别是不区分大小写

last

- **参数**: 秒或#num
- **支持值类型**: float, int, str, text, log
- **描述**: 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

例子:

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 Zabbix 1.6.2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

- **参数**: string
- **支持值类型**: log
- **描述**: 检查最近的日志条目的 Event ID 是否匹配正则表达式. 参数为正则表达式,POSIX 扩展样式. 当返回值为0时表示不匹配，1表示匹配。 该函数从 Zabbix 1.8.5 起开始支持.

logseverity

- **参数**: 忽略
- **支持值类型**: log
- **描述**: 返回最近日志条目的日志等级(log severity). 当返回值为0时表示默认等级，N 为具体对应等级(整数，常用于 Windows event logs). Zabbix 日志等级来源于 Windows event log 的 Information 列.

logsource

- **参数**: string
- **支持值类型**: log
- **描述**: 检查最近的日志条目是否匹配参数的日志来源. 当返回值为0时表示不匹配，1表示匹配。通场用于 Windows event logs 监控. 例如 logsource["VMWare Server"]

max

- **参数**: 秒或#num
- **支持值类型**: float, int
- **描述**: 返回指定时间间隔的最大值. 时间间隔作为第一个参数可以是秒或收集值的数目(前缀为#). 从 Zabbix 1.8.2开始，函数支持第二个可选参数 time_shift，可以查看 avg()函数获取它的使用方法.

min

- **参数**: 秒或#num
- **支持值类型**: float, int

- **描述**: 返回指定时间间隔的最小值. 时间间隔作为第一个参数可以是秒或收集值的数目(前缀为#). 从 Zabbix 1.8.2开始，函数支持第二个可选参数 time_shift，可以查看avg()函数获取它的使用方法.

nodata

- **参数**: 秒
- **支持值类型**: any
- **描述**: 当返回值为1表示指定的间隔(间隔不应小于30秒)没有接收到数据, 0表示其他.

now

- **参数**: 忽略
- **支持值类型**: any
- **描述**: 返回距离 Epoch(1970年1月1日 00:00:00 UTC)时间的秒数

prev

- **参数**: 忽略
- **支持值类型**: float, int, str, text, log
- **描述**:返回之前的值，类似于 last(#2)

regexp

- **参数**: 第一个参数为 string，第二个参数为秒或#num
- **支持值类型**: str, log, text
- **描述**: 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

- **参数**: 第一个参数为 string，第二个参数为秒或#num
- **支持值类型**: str, log, text
- **描述**: 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

- **参数**: 秒或#num
- **支持值类型**: str, log, text
- **描述**: 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. 该函数从 Zabbix 1.8.4起开始支持

sum

- **参数**: 秒或#num
- **支持值类型**: float, int

- **描述**: 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 Zabbix 1.8.2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

time

- **参数**: 忽略
- **支持值类型**: any
- **描述**: 返回当前时间，格式为 HHMMSS，例如123055

触发器的故障等级

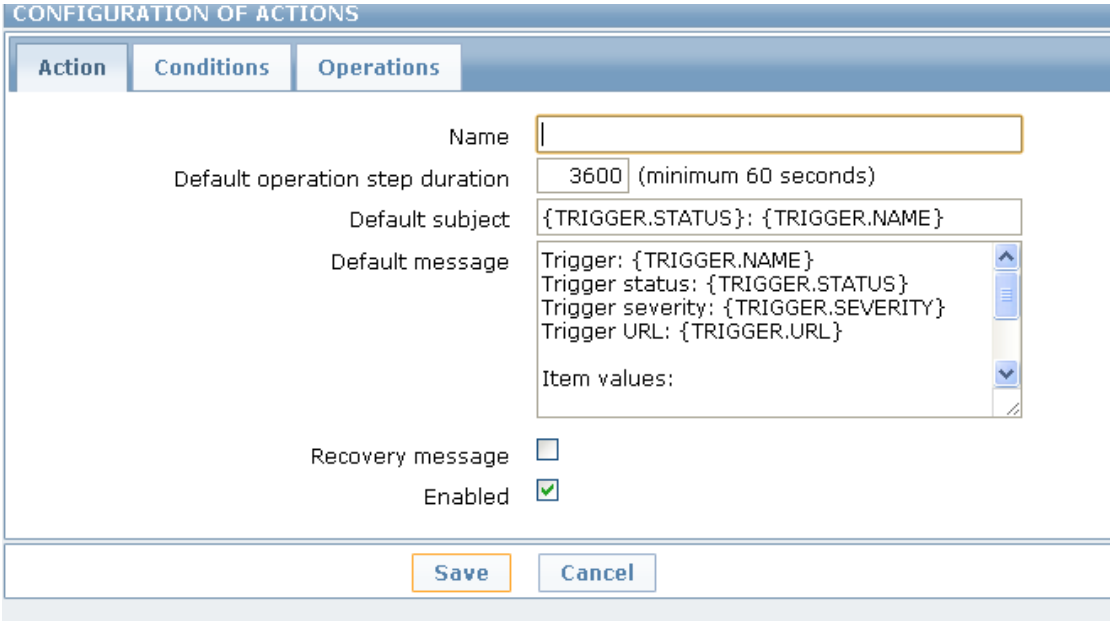
触发器严重性定义了一个触发器的重要程度。zabbbix 支持下列触发器严重性：

严重性（SEVERITY）	定义（DEFINITION）	颜色（COLOUR）
未分类（Not classified）	未知严重性（Unknown severity）	灰色（Grey）
信息（Information）	信息（For information purposes）	亮绿（Light green）
警告（Warning）	因此注意（Be warned）	黄色(Yellow)444444
平均（Average）	平均问题（Average problem）	橘黄（Orange）
高（High）	重要的事情发生（Something important has happened）	红（Red）
灾难（Disaster）	灾难，资产损失等（Disaster. Financial losses, etc）	亮红（Bright red）

3.7 添加 Actions

Action 是告警动作，当触发器条件被满足时，就会执行指定的 action。

通过 Configuration->Actions->Create Action 来创建 Action



Event source:来源

triggers，即所有的 triggers 条件满足时都会执行这个 action

Discovery:自动发现模块

Auto registration：事件产生记录

Escalations：告警是否升级，及升级时间

Subject、Message：告警标题和内容，此处可引用 zabbix 的宏变量；例如{{HOSTNAME}:{TRIGGER.KEY}.last(0)}表示最后一次采集值，更多宏变量参考:

http://www.zabbix.com/documentation/2.0/manual/config/macros/usermacros

Recovery Message：告警恢复信息，不勾选系统会用默认的，勾选后自定义

Conditions：trigger 产生的条件，条件可以多选

Operation：选择 media 及 user

更多资料参考

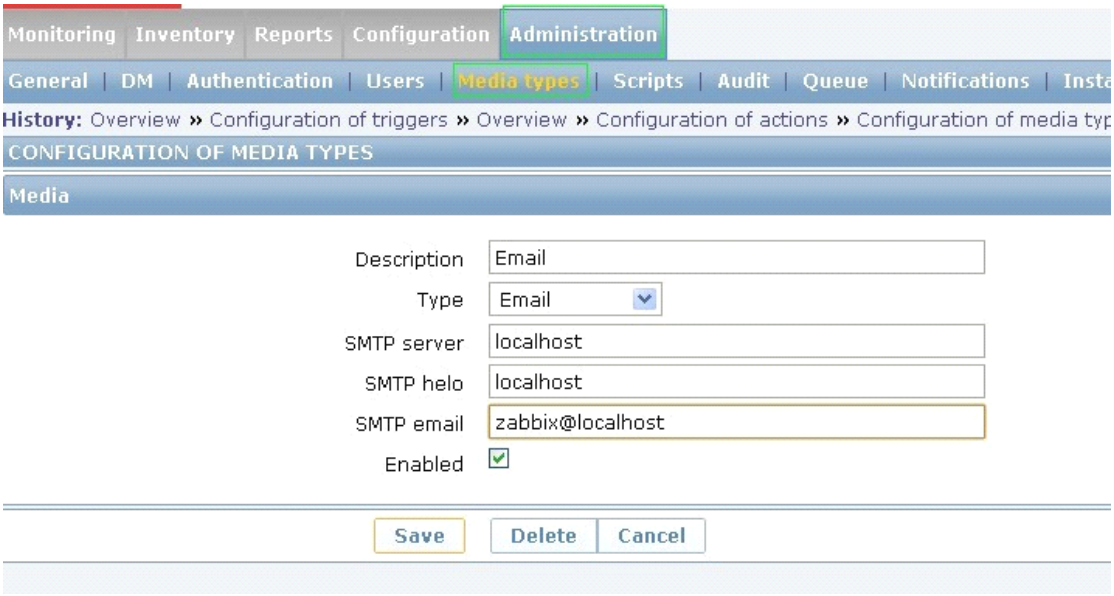
http://www.zabbix.com/documentation/2.0/manual/config/notifications/action

另请参考 3.27 章节

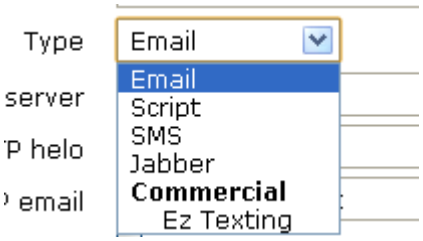
3.8 添加 Medias

Media，即告警方式，Zabbix 可以提供四类 Media：Email/SMS/Jabber/Script，通过 Administrator->Media Type 来修改或新增告警方式

Email 方式最常用的，填入相关的 SMTP 信息，即可通过邮件方式发送告警。



其中 type 有以下几种方式



Email 方式用邮件

Script 方式可以通过自己编写程序或脚本的方式发送告警信息。

SMS 方式要在 server 主机上接入短信 modem。

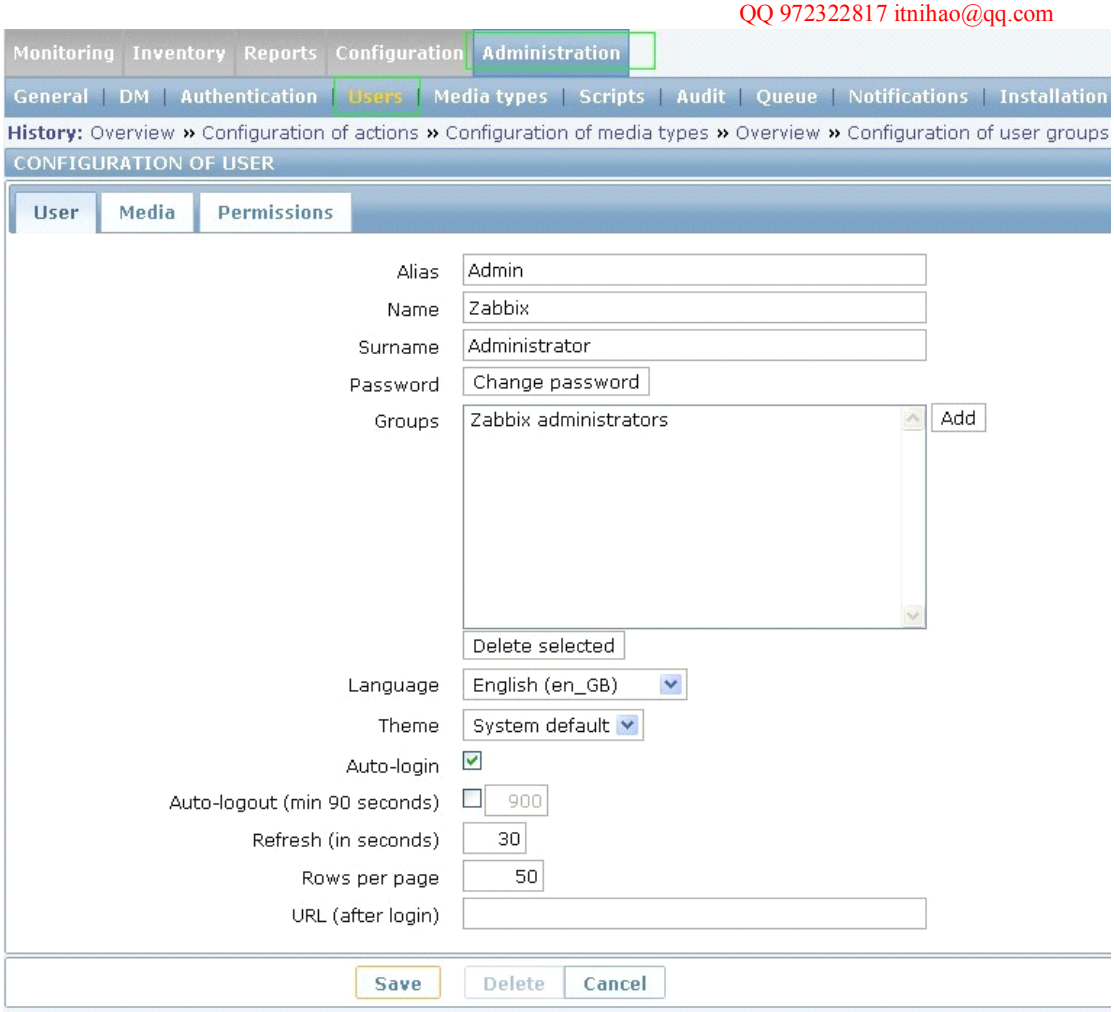
Jabber 方式是一种 linux 下的即时通讯工具,通过 Jabber 发送即时消息。

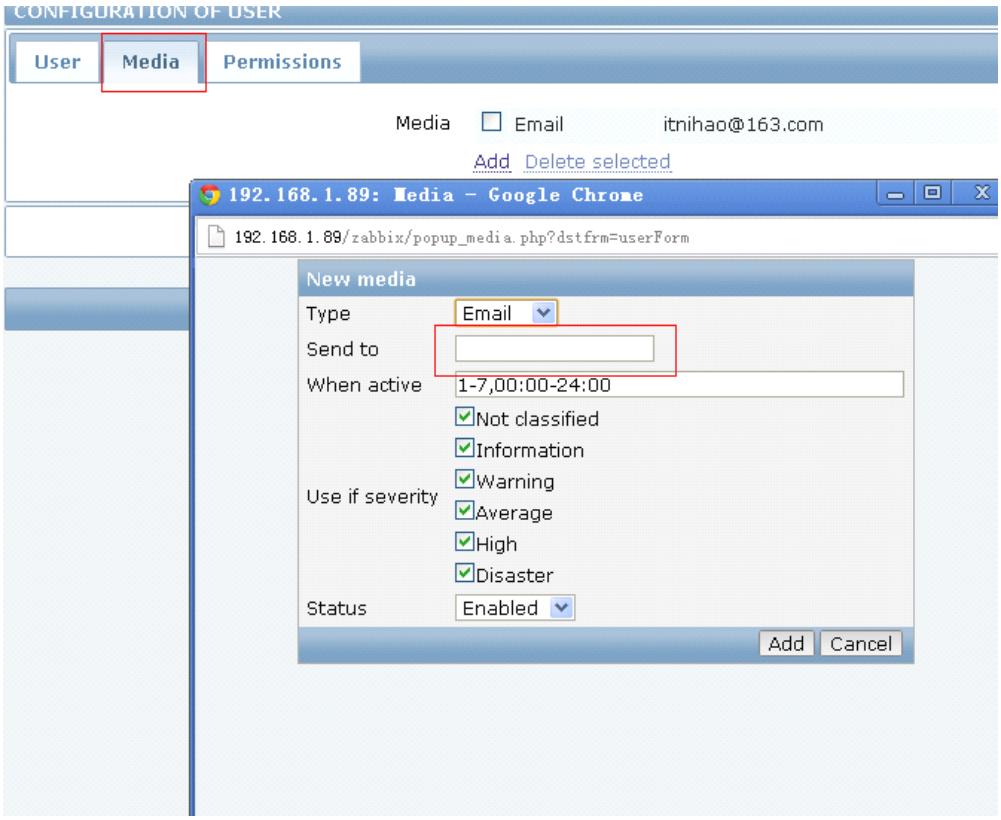
3.9 添加 Users

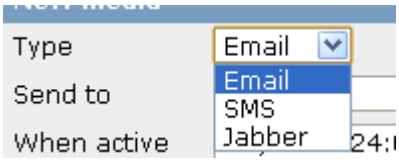
在 Administrator->Users 可以添加用户和用户组

通过 User Group 可以限制用户的权限，zabbix 自带的用户组的权限限制基本能满足我们的要求。

创建用户时可以根据用户的不同作用划分到不同的组，media 中填入告警接受地址及告警接受时间等信息。

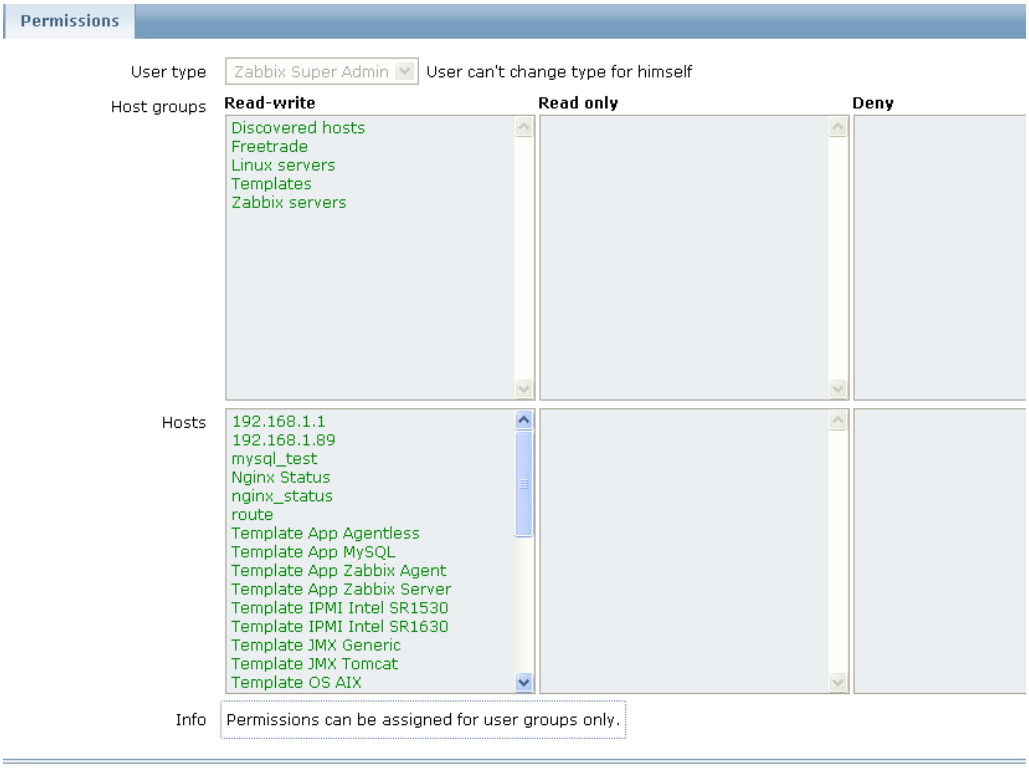

Monitoring
Inventory
Reports
Configuration
Administration
General
DM
Authentication
Users
Media types
Scripts
Audit
Queue
Notifications
Installation
History: Overview » Configuration of actions » Configuration of media types » Overview » Configuration of user groups
CONFIGURATION OF USER
User
Media
Permissions
Alias
Admin
Name
Zabbix
Surname
Administrator
Password
Change password
Groups
Zabbix administrators
Add
Delete selected
Language
English (en_GB)
Theme
System default
Auto-login
Auto-logout (min 90 seconds)
900
Refresh (in seconds)
30
Rows per page
50
URL (after login)
Save
Delete
Cancel


CONFIGURATION OF USER
User
Media
Permissions
Media
Email
itnihao@163.com
Add
Delete selected
192.168.1.89: Media - Google Chrome
192.168.1.89/zabbix/popup_media.php?dstfrm=userForm
New media
Type
Email
Send to
When active
1-7,00:00-24:00
Use if severity
Not classified
Information
Warning
Average
High
Disaster
Status
Enabled
Add
Cancel



Type：报警类型

Send to ：收件人地址，可以增加多个



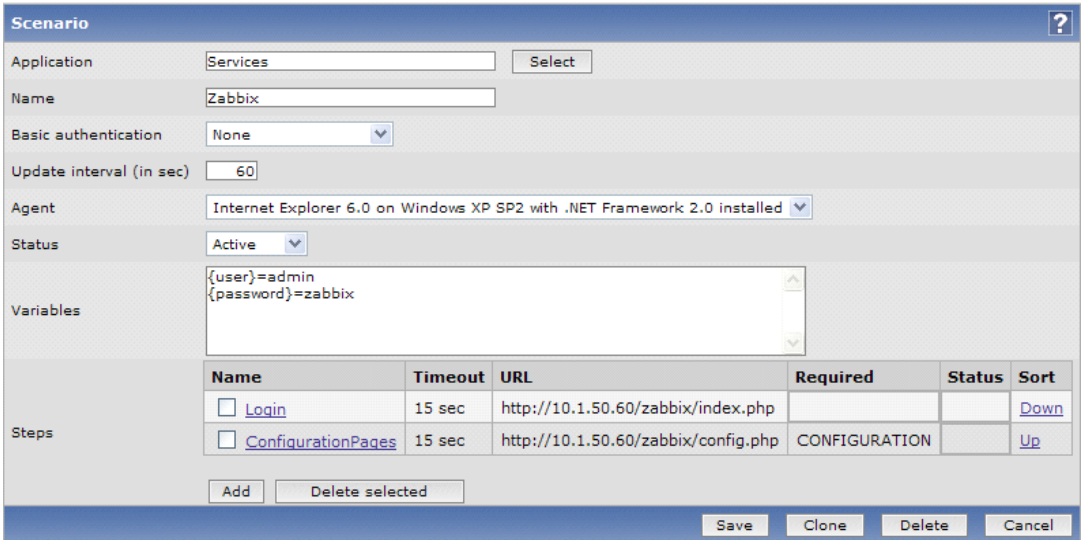
用户权限的控制

3.10 添加 WEB Monitorings

Web Monitoring 是用来监控 web 程序的，可以监控到 web 程序的下载速度、返回码及响应时间，还支持把一组连续的 web 动作作为一个整体来监控。

下面我们以监控登陆 zabbix 的 web 程序为例，来展示如何使用 web monitoring。

Configuration->web->Create Scenario 创建一个 Scenario(注：必须选择 host 后才能创建 scenario，zabbix 的所有 items 都必须创建在 hosts 上)



图表 1

Application：选择这个 scenario 所在的 application 组

Name：scenario 的名字

Basic authentication：鉴权

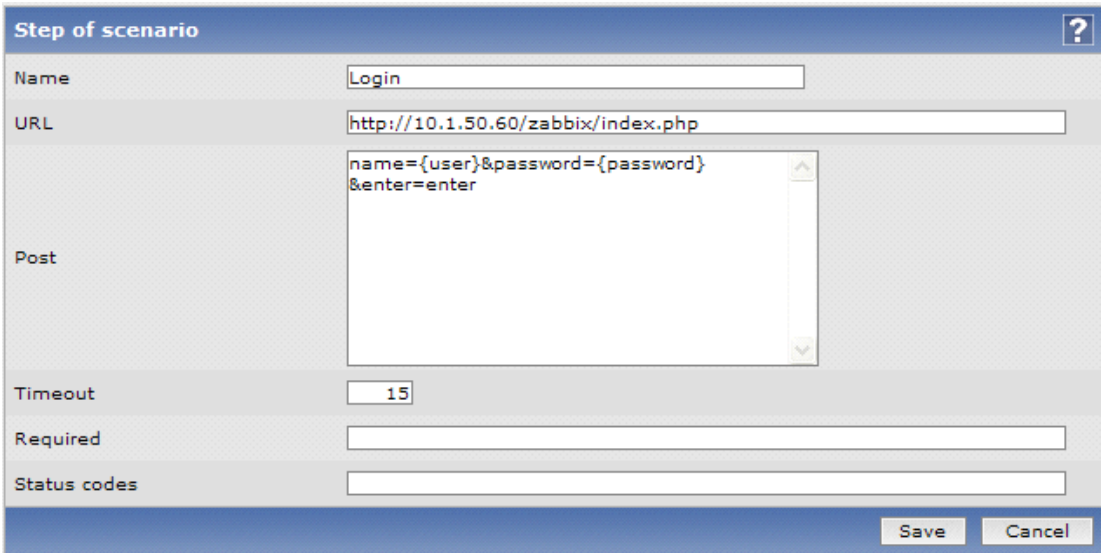
Update interval：监控频率，s 为单位

Agent：选择要使用的浏览器客户端，可能同样的 web 程序对不同的客户端展示的内容会不一样

Status：默认为 active

Variables：变量定义，这里定义的变量可在后续的 steps 中使用，这里我们定义了用户和密码的变量

Steps：web 程序的各个步骤，选择 add 新增一个 Login 的 step，来模拟用户登陆，传递用户和密码给 index.php 页面



图表 2

URL：监控的 web 页面(注:必须是全路径带页面名)

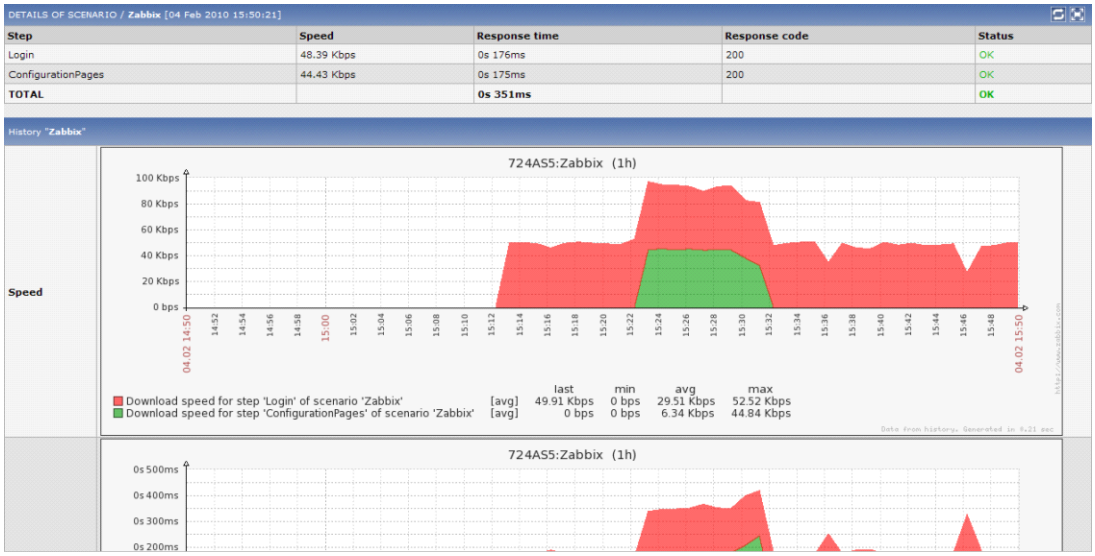
Post:传递给页面的参数，多个参数之间用&连接，此处可引用前面定义的变量

Timeout：超时时间

Required：页面中能匹配到字符,匹配不到即认为错误

Status codes:页面返回码

添加完 step 后，我们在 Monitoring->web 页面即能看到监控的状态和图示



图表 3

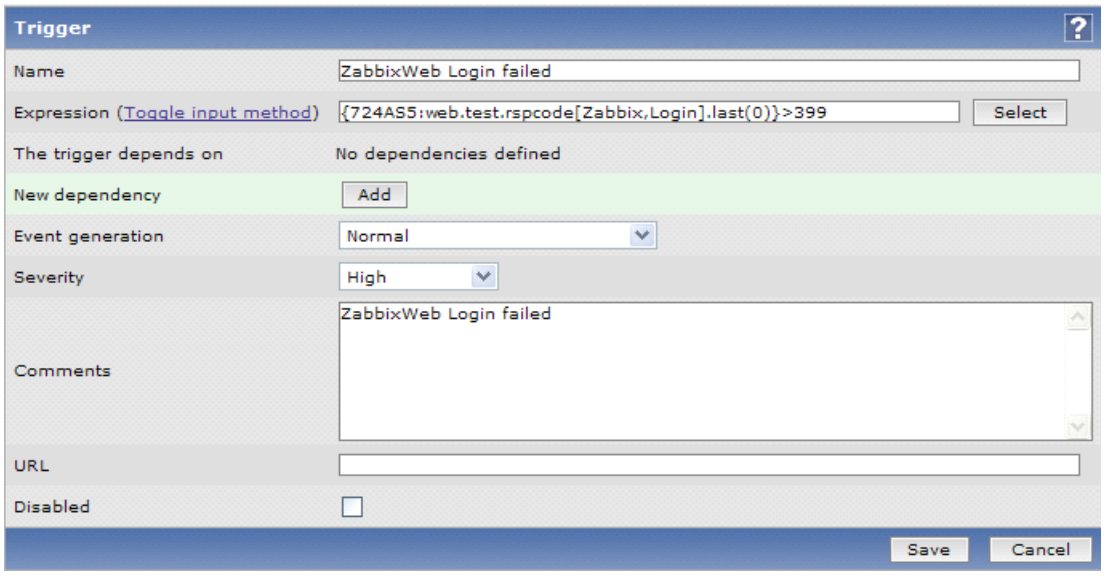
创建完 scenario 后，zabbix server 会自动创建相关的 items，所以我们只需为这些 items 添加 triggers 即可让 web scenario 出错时产生告警

Configuration->hosts->点击 scenario 所在的 host 条目的 trigger，直接 create trigger，在 select items 的时候就可以看到系统自动创建的 items（注：自动创建的 items 在 host 的 items 列表中直接是看不到的，需要在创建 trigger 时选择 items 时才能看到）

图表 4

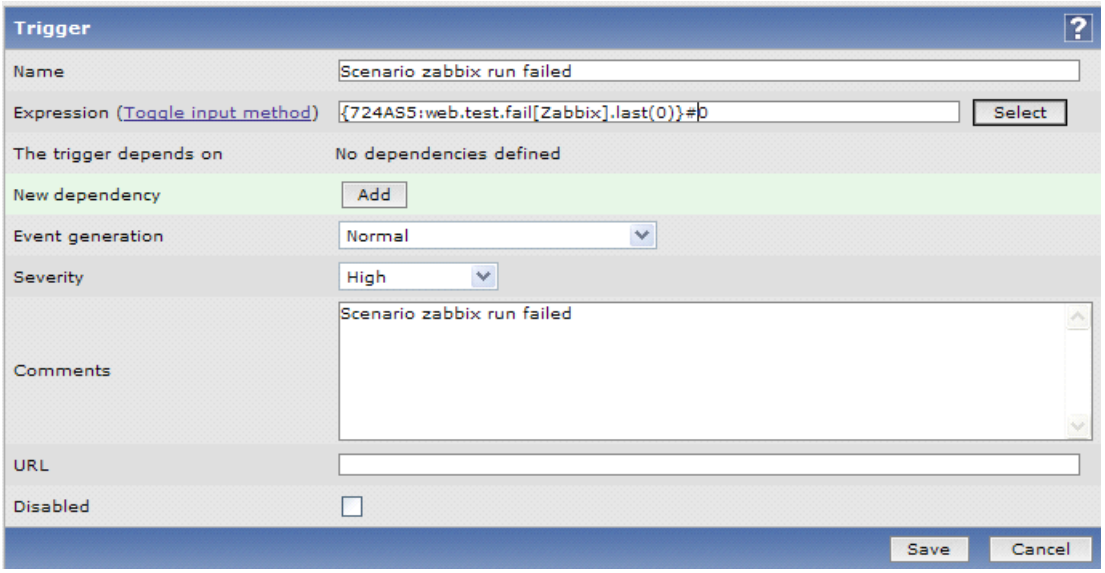
可以在 items 列表中看到，系统为每个 step 创建了 3 个 item，Download Speed/Response Code/Response Time,为整个 scenario 创建了一个 test.fail 的 item，可以分别为其创建 trigger

下例我们创建一个 Login 页面返回码的 trigger，大于等于 400 即为错误



图表 5

再创建一个整个 scenario 所有 step 运行是否成功的 trigger，采集值为 0 表示整个 scenario 的所有 step 都执行成功了，第几步的 step 执行失败就返回数字几，且后续的 step 都不会继续执行下去。



图表 6

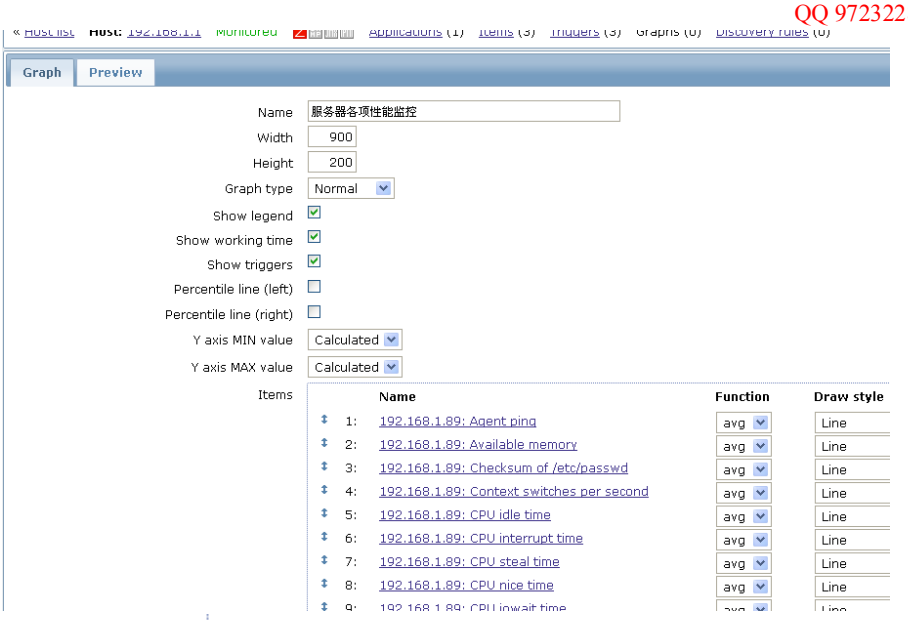
这样，一个完整的 web monitoring 就配置完成了。

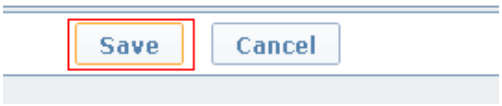
Web monitoring 还有更多强大的功能，未能一一研究了解，有待挖掘

3.11 添加 Graphs

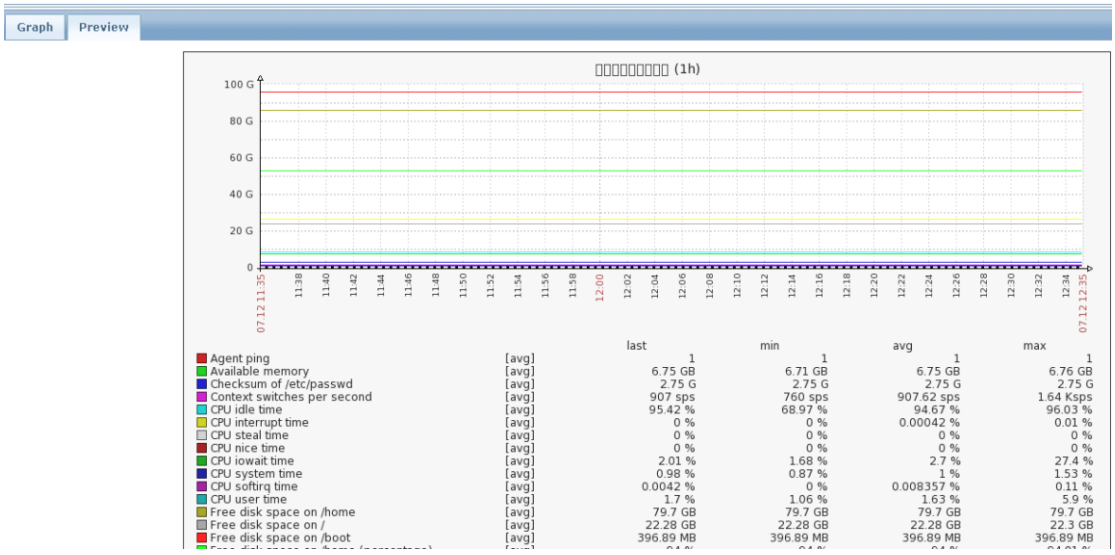
Zabbix 的 Graphs 功能很强大，可以为每一个 item 绘制图表，也可以把多个 items 绘制在一张图表内。

通过 configuration->hosts 选择要绘制图表的 host，点击 graphs，create graphs 即可创建图表。





注意，此处选择的数据如果为多项，则多项数据呈现在一张图表上面



（此处对中文的支持不好，建议在监控过程中全部使用英文字符）

Graph type：图表样式，有线状、柱状、饼状

还可以自定义图表大小，及 Y 轴最大最小值

通过 add items 可以添加在同一个图表中展示的多个 items（注：注意每个 item 的颜色及取值范围，范围相差太大图表会显示不全）

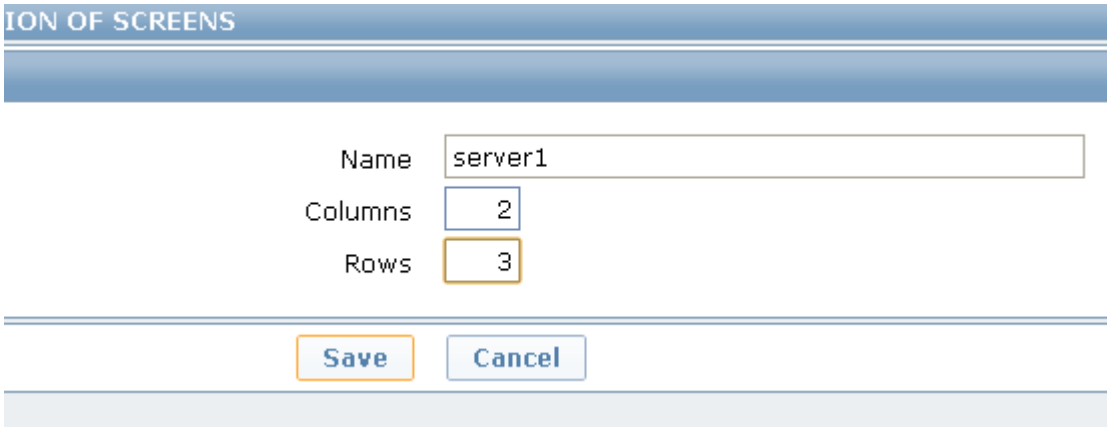
配置好的 graphs 在 monitoring->graphs 中查看

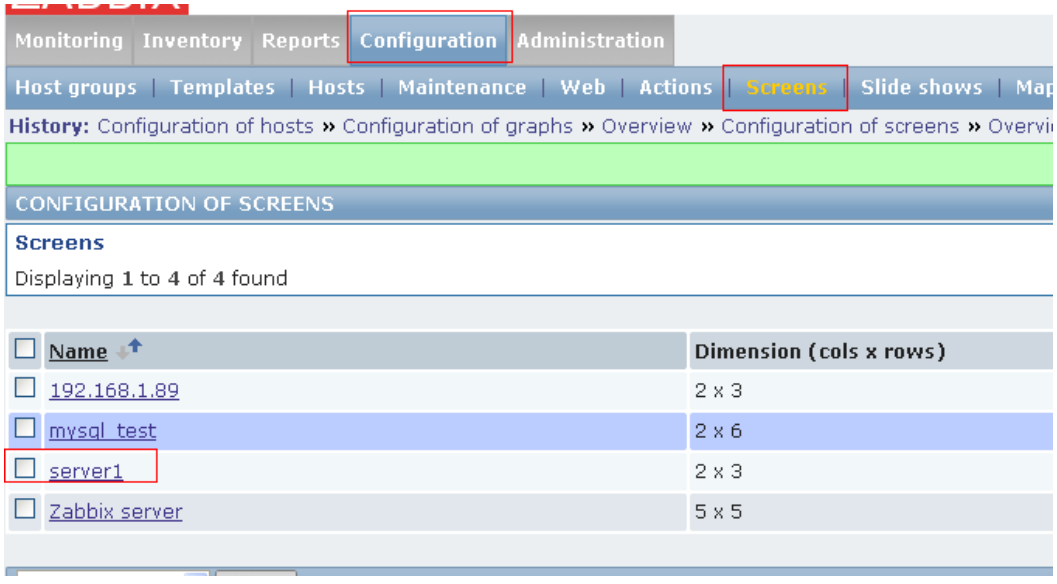
在 monitoring->last data 下能快速查看每个 host 的每个 item 的 graph

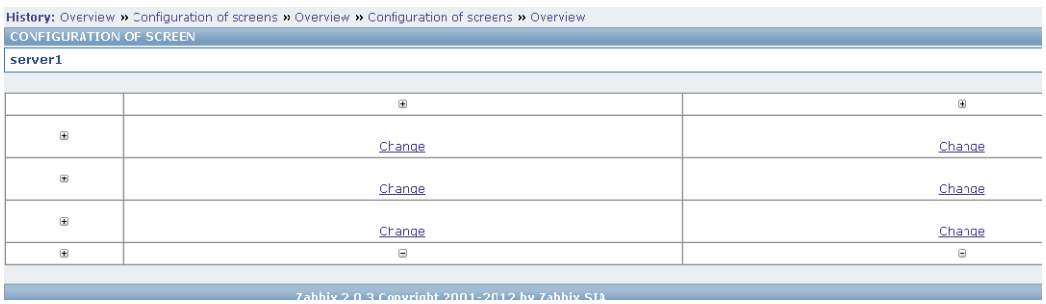
3.12 添加 Screens

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，或是比较多个 hosts 的同一种信息,这些信息可以为 graphs、maps、server infos 等等，几乎涵盖 zabbix 所有的监控信息。

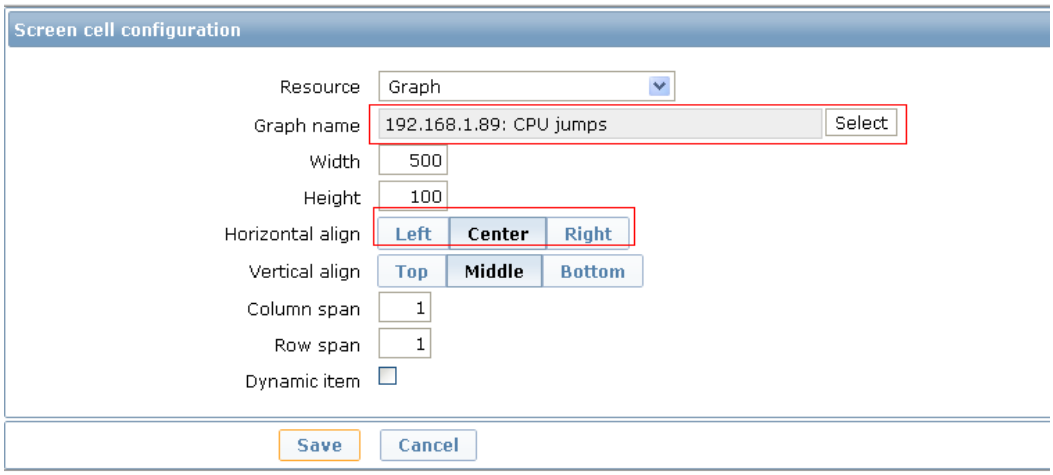
通过 configuration->screen->creat screen 来创建，创建时定义 screen 的行数和列数，点击对应单元格内的 change，添加相应的信息



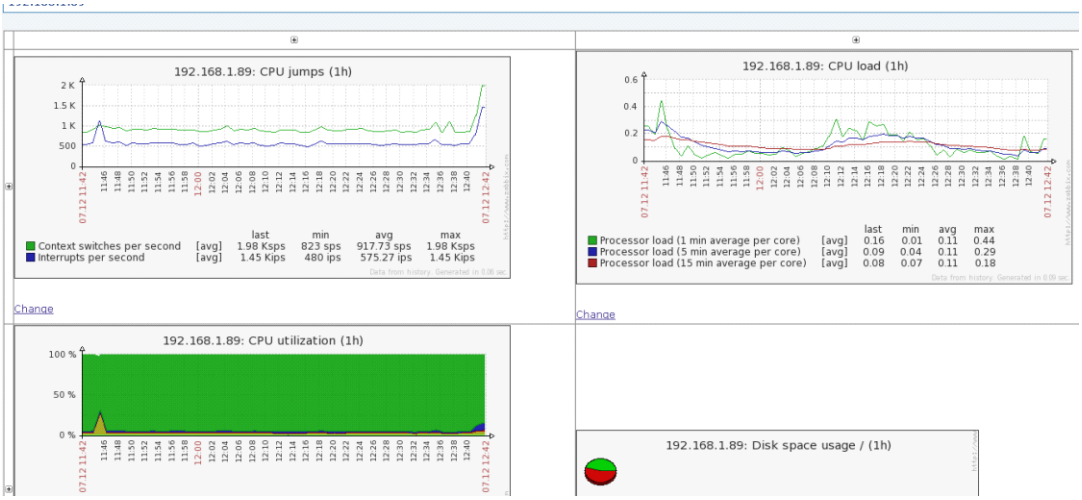




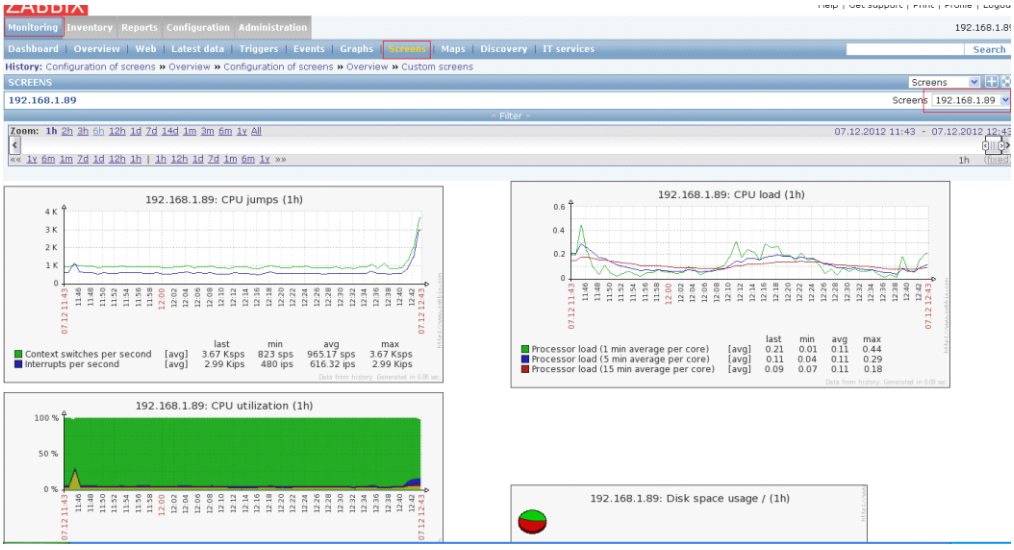
点击 change 添加图像



选择图像类型，对齐方式等



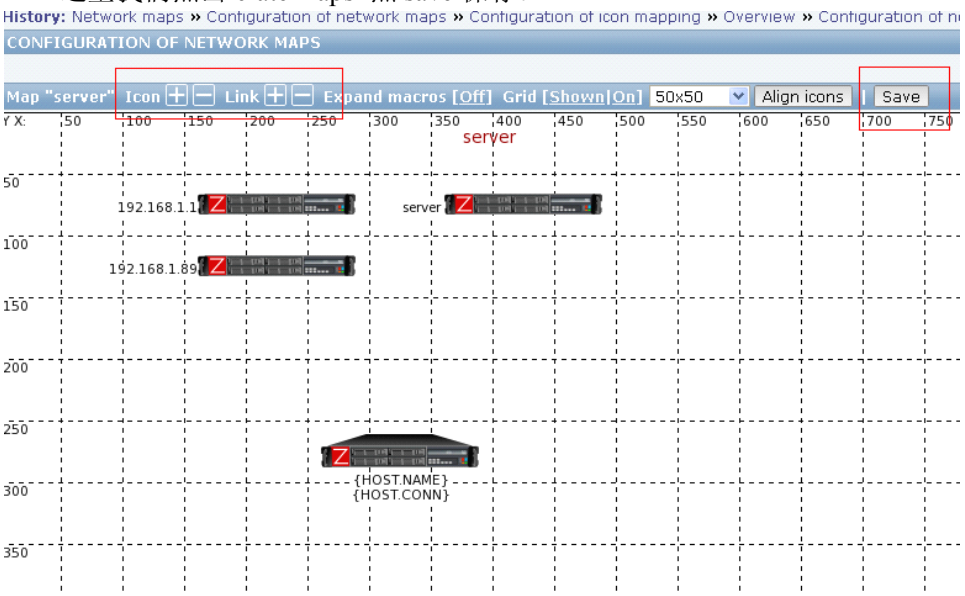
通过 monitoring-screen，可以查看之前配置好的信息

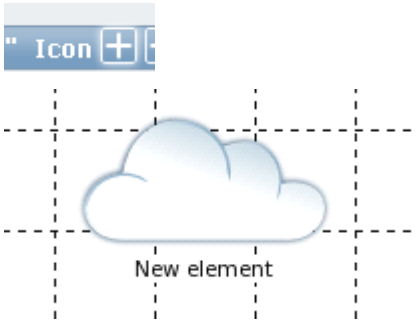


3.13 添加 Maps

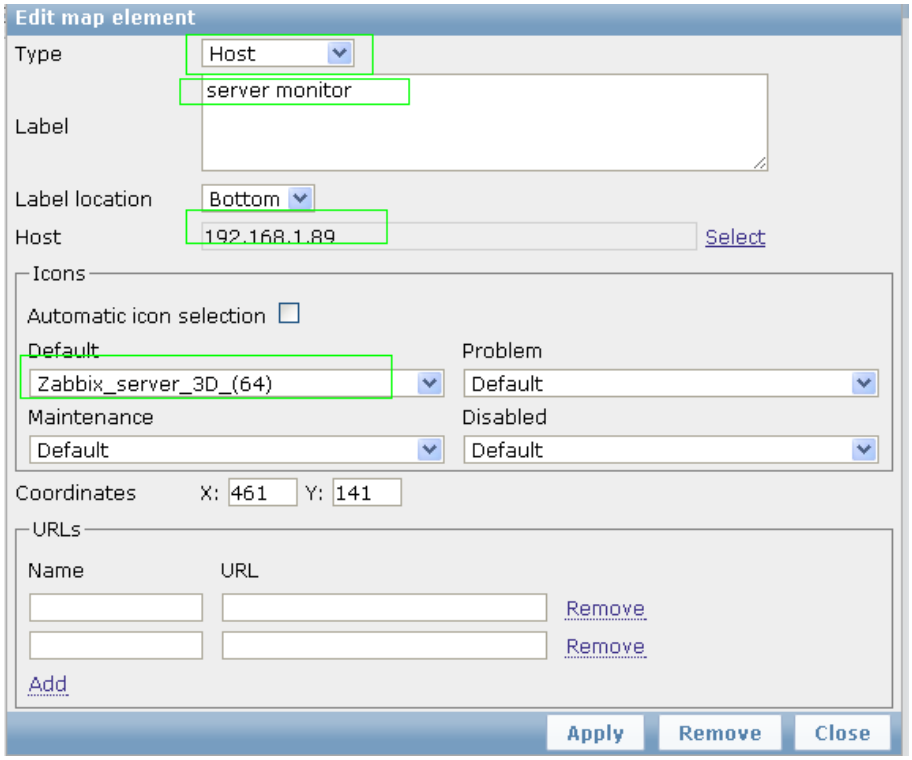
这里可以添加关于主机的拓扑图： configration-maps,在右上角可以 create maps 或者 import map

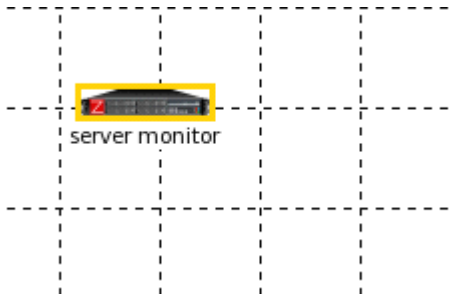
这里我们点击 crate maps 点 save 保存。

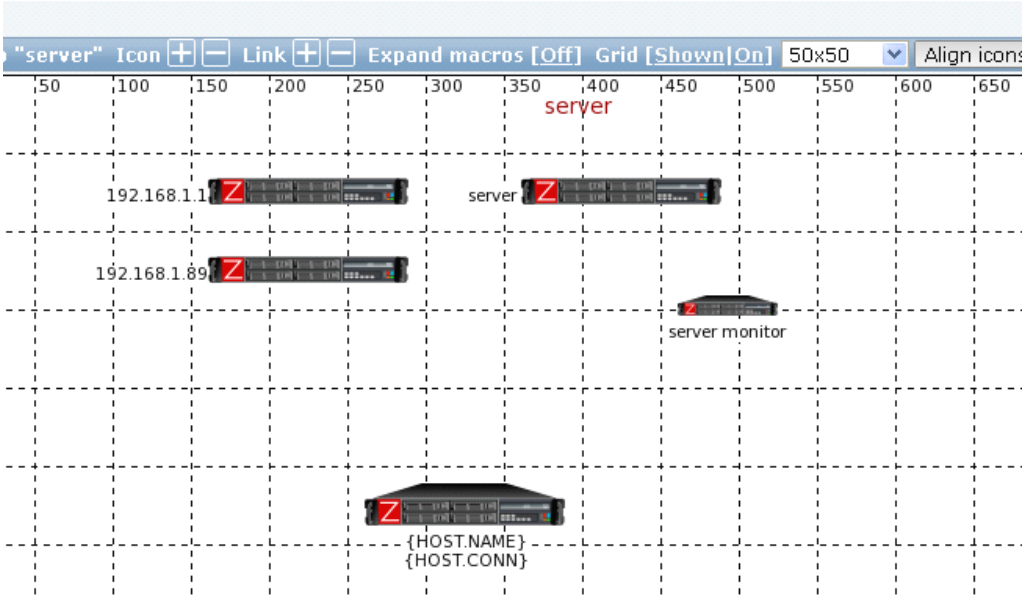




双击







上面加号和减号可以增加主机 link 说明主机的连接情况，可以进行自定义。

3.14 添加 MySQL 监控

Zabbix 自带有 MySQL 的监控模板，可以做一些简单的监控。

1、更改 agentd 配置

#vim /etc/zabbix/zabbix_agentd.conf

```
UnsafeUserParameters=1
UserParameter=mysql.ping,mysqladmin  -uroot -pharry ping|grep alive|wc -l
```

修改后重启 host 上的 agentd，使配置文件生效。

```
#service  zabbix_agentd  restart
```

2、添加 items

web 端编辑 mysql 所在的 host，使之 link 到 template_APP_MySQL 模板，然后在 host 的 items 里就能看到刚才定义的这些 MySQL 的监控项了，修改相应的 trigger 值即可。

这个 zabbix 自带的 mysql 监控功能比较弱，只是通过 mysqladmin 工具去查询 mysql 的一些状态而已。

我们可以自己编写或是找一些功能更强的 mysql 监控脚本，加到 zabbix 监控里，后面会讲到如何自己添加监控。

另外更详细的方法可参考 zabbix wiki 上的 mysql 监控方法，这个监控的就非常详细 :http://www.zabbix.com/wiki/howto/monitor/db/mysql/extensive_mysql_monitoring_including

_replication

3.15 添加 SNMP 监控

Zabbix snmp 的监控要在 configration 中 hosts 中添加相关主机的模板和 snmp 版本，一般用 snmp 监控网络设备就可以了.

3.16 添加自定义监控

对于 zabbix 功能上无法实现的监控，我们可以通过自己编写程序或脚本来辅助完成，并将脚本的结果通过 agent 递交给 zabbix server 统一管理，一样可以绘制 graph 报表等。

UserParameters 的定义方法，请参考：
http://www.zabbix.com/documentation/2.0/manual/config/items/userparameters
如何自定义监控

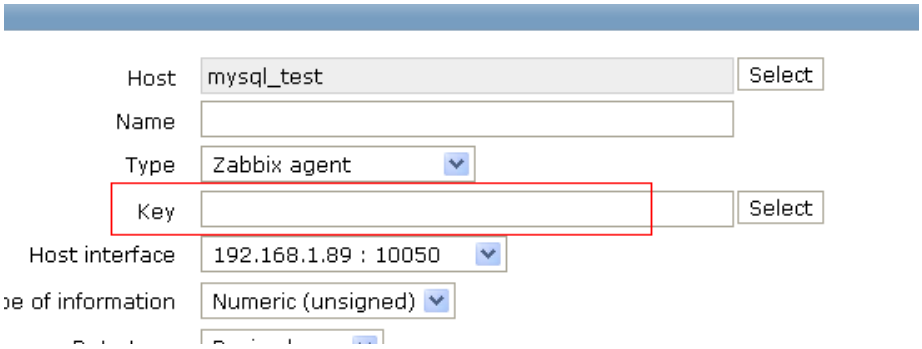
修改/etc/zabbix/zabbix_agentd.conf

UnsafeUserParameters=1 此处默认是 0（不允许自定义脚本）

然后在后面添加 UserParameter=key,command

key,command 为格式

此处 key 是在创建 item 的时候填写的



在修改 zabbix_agentd.conf 后，重启 zabbix_agentd 服务

```
UserParameter=ping.ping1,/etc/zabbix/scripts/ping ping1
```

例如 ping.ping1

用 zabbix_get 命令获取得到的值

```
[root@kx1d zabbix-2.0.3]# zabbix_get -s 192.168.1.89 -p 10050 -k ping.ping1
0
```

```
UserParameter=nginx.accepts,/etc/zabbix/scripts/nginx_status accepts
UserParameter=nginx.handled,/etc/zabbix/scripts/nginx_status handled
UserParameter=nginx.requests,/etc/zabbix/scripts/nginx_status requests
UserParameter=nginx.connections.active,/etc/zabbix/scripts/nginx_status active
UserParameter=nginx.connections.reading,/etc/zabbix/scripts/nginx_status reading
UserParameter=nginx.connections.writing,/etc/zabbix/scripts/nginx_status writing
UserParameter=nginx.connections.waiting,/etc/zabbix/scripts/nginx_status waiting
[root@kx1d zabbix-2.0.3]# zabbix_get -s 192.168.1.89 -p 10050 -k nginx.connections.active
8
```

下面以一个实例来说明此问题

例如要监控 mysql

#vim /etc/zabbix/zabbix_agentd.conf

```
UnsafeUserParameters=1
```

```
UserParameter=mysql.ping_test[*],mysqladmin -u$1 -p$2 ping|grep alive|wc -l
```

#service zabbix_agentd restart

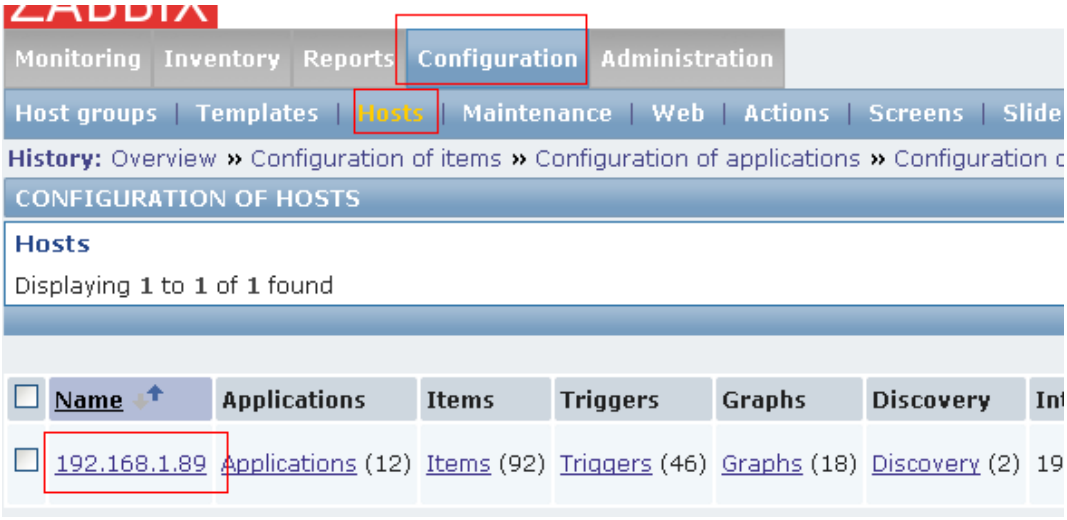
用 zabbix_get 检测设置是否生效

```
[root@kx1d zabbix-2.0.3]# zabbix_get -s 192.168.1.89 -k mysql.ping_test[root,harry]
1
[root@kx1d zabbix-2.0.3]#
```

返回值为 1，说明添加成功

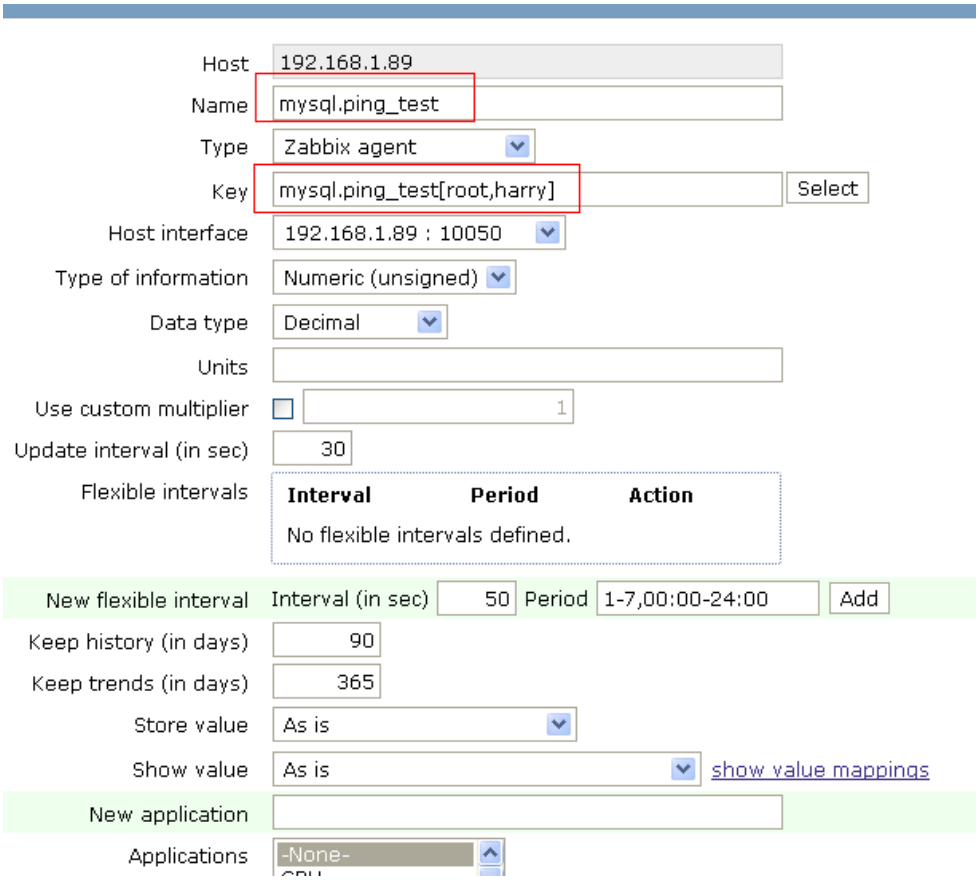
下面添加到主机。

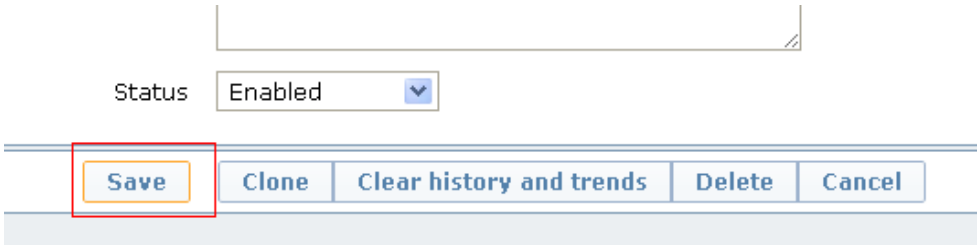
选择一个 host 192.168.1.89



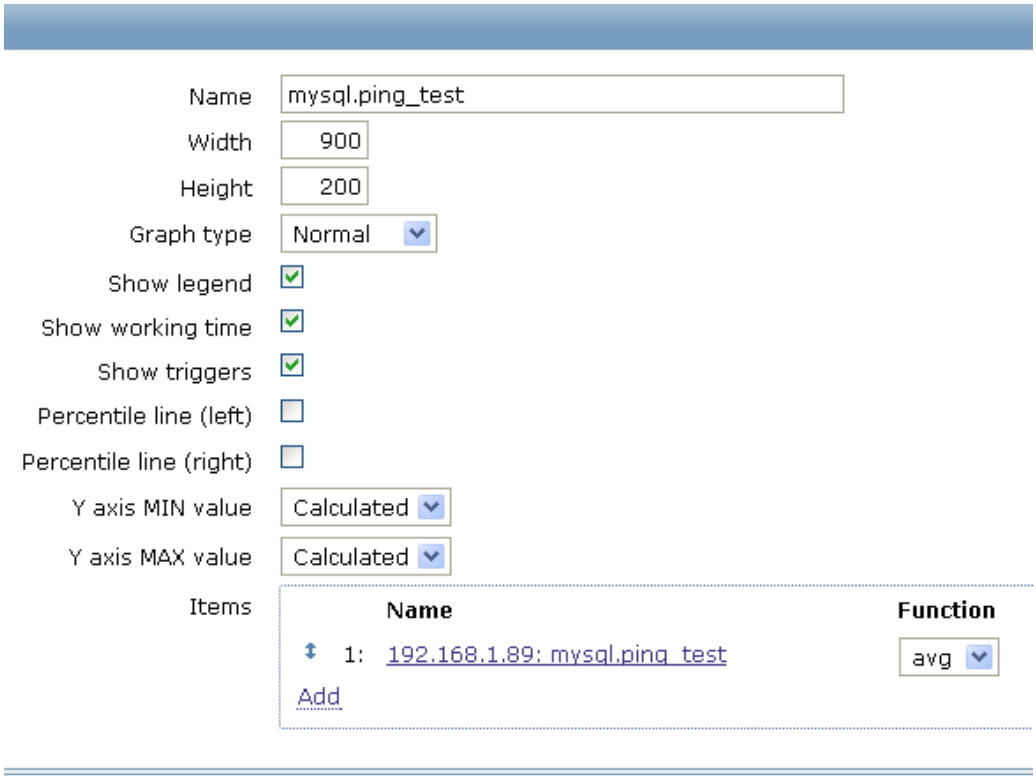
点击 Items-create Itmes

输入 key 值为 mysql.ping_test[root,harry]





点击 graphs---create graph



在 ADD 处选择 Items 为 mysql.ping_test--save,稍等会儿就出图

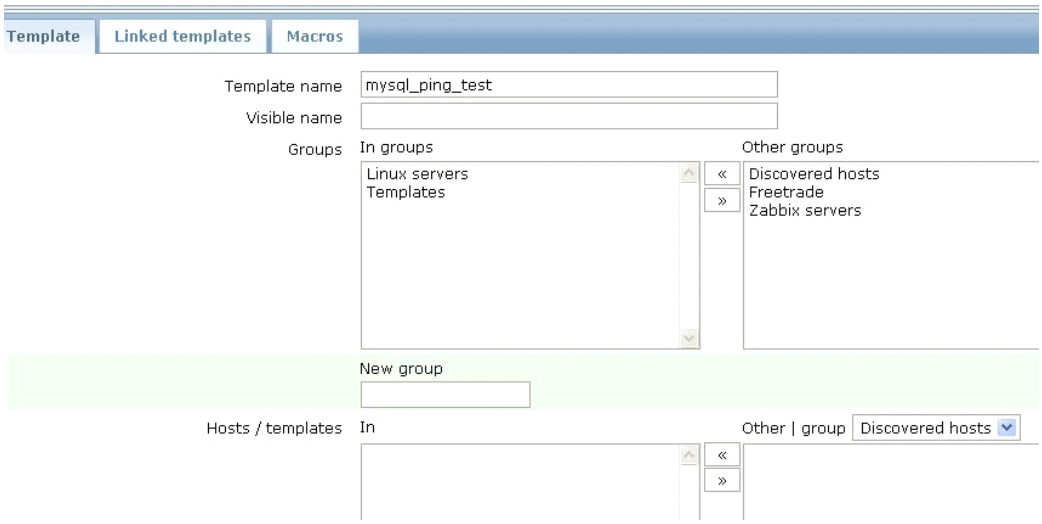
当然，此处为一个简单的实现，如果需要监控复杂的应用，需要写脚本来实现，但方法是类似的。

3.17 添加 Templates

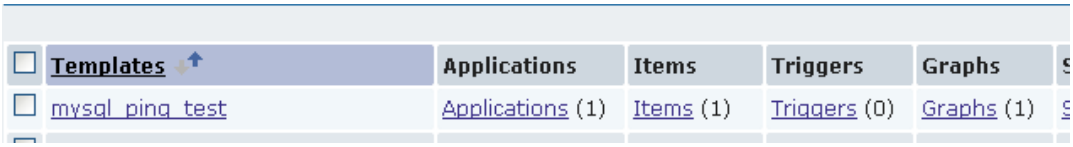
如果有大量的同一类设备，需要监控的信息也大致类似，一个个去修改相关参数比较麻

烦，我们可以通过创建一个 template 来简化操作。

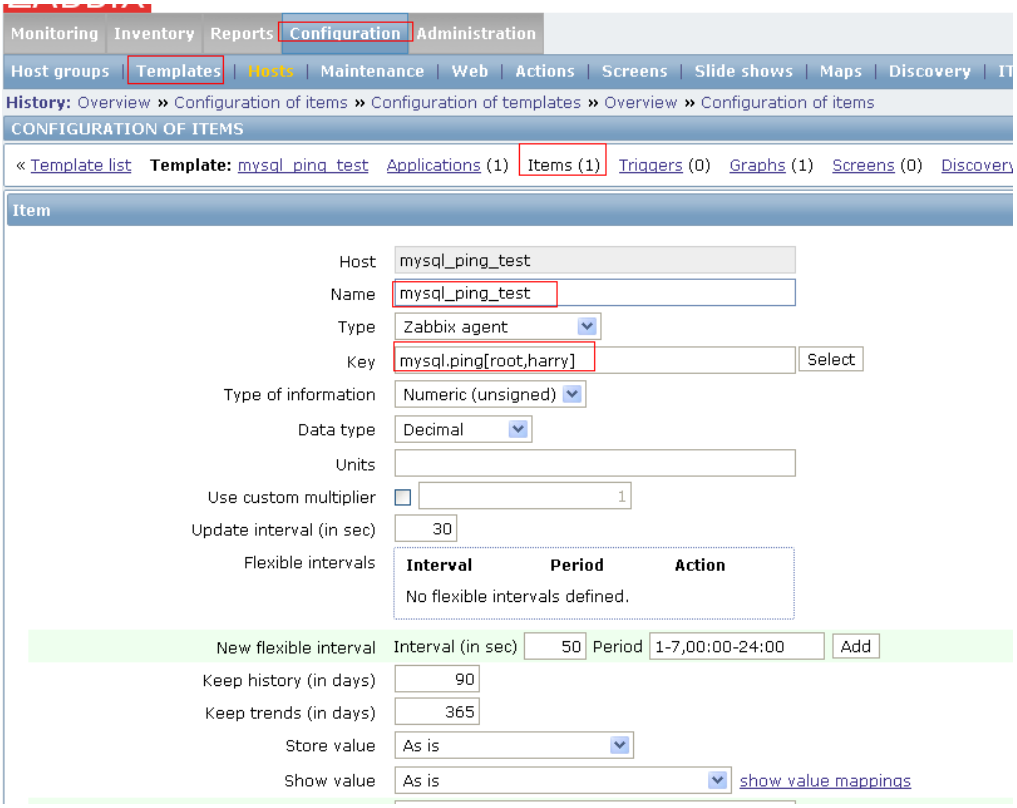
Configuration->Host Groups->Template->Create Template



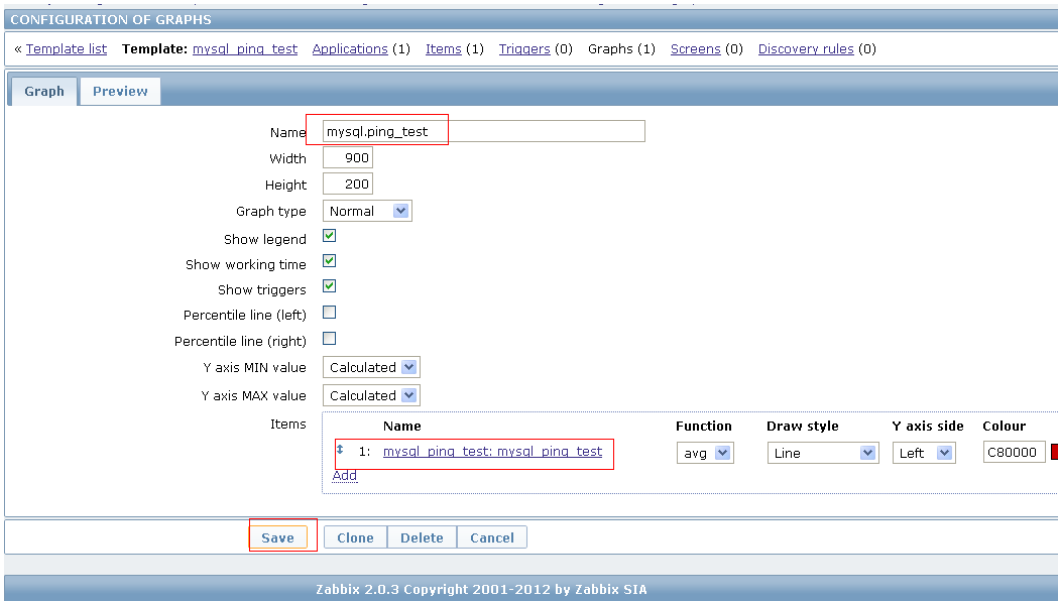
点击 Items---create Items-



cteate Items



graph--create graph--

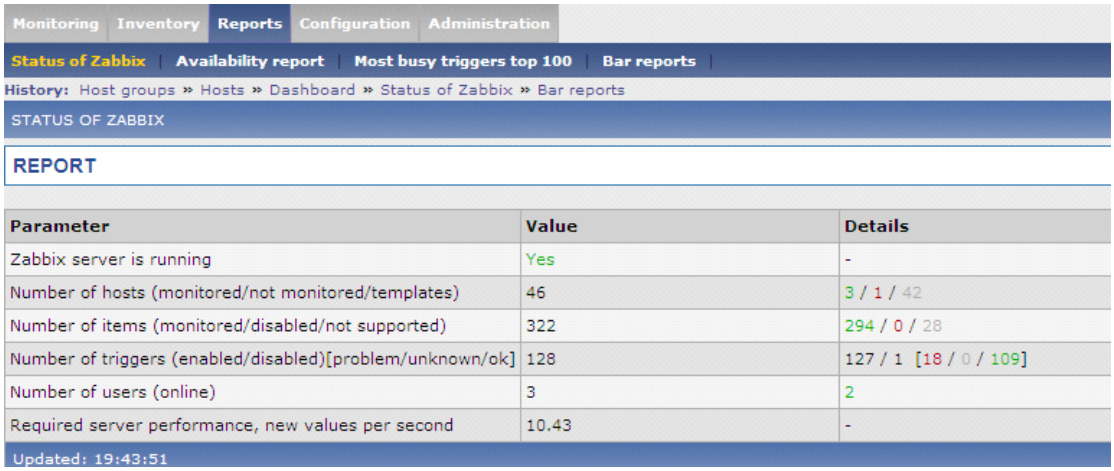


添加到模板，后面就可以引用改模板了。

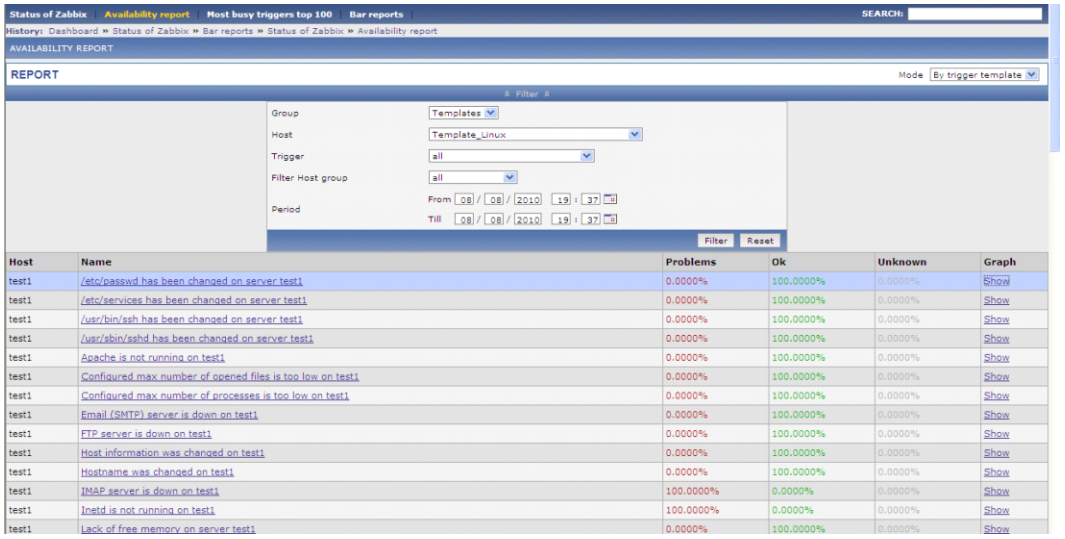
3.18 添加 Reports（定制报表）

在 zabbix 中关于报表的功能有三项：

Status of zabbix:这是关于整个 zabbix 监控系统的



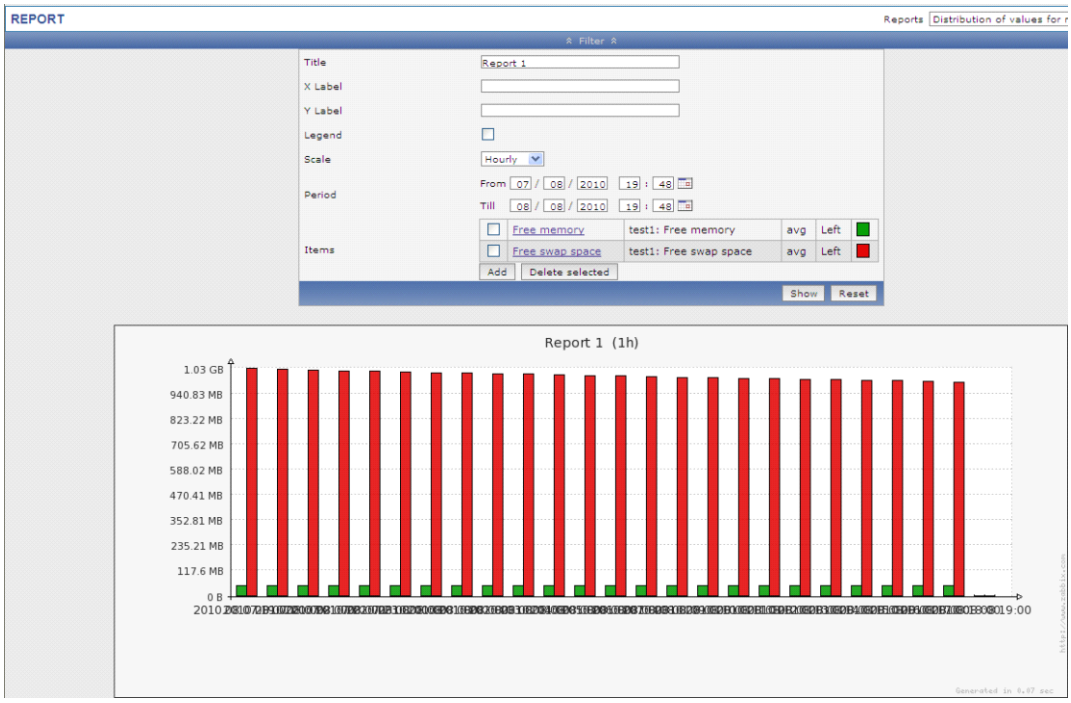
Avaliability report：整个系统可用的系统报表提供过滤功能。



Most busy triggers top 100：提供最常用的 triggers 预览：

Bar report ：可定制报表可以报多个报表整合到一起。

如下图是对 test1 server 的 free memory 和 swap free 每小时报表：



3.19 添加 Macros

Macros 指宏变量，定义的宏变量可以在 trigger、actions 等多种场景中引用。

Macros 分系统自带全局宏的及自定义的宏。

系统自带的全局 macros 列表及解释参考：

http://www.zabbix.com/documentation/2.0/manual/config/macros

引用 macros 的例子可参考上述 action 中添加{{Hostname}:{trigger.key}.last(0)}的例子。

Zabbix 还支持自定义 macros，在添加 host 或是 template 时，我们可以在 macros 项中定义好后续要用到的宏变量，格式为:

{macroname}=macrovalue

自定义的宏变量及系统自带的宏变量都可以在 zabbix 场景中引用，zabbix 在遇到引用的宏变量时，会先查找当前场景中定义的宏，接着查找当前 host 的自定义宏，接着查找 link 的 template 的宏，最后查找 zabbix 系统自带的全局宏。所以在自定义宏时注意宏引用的顺序。

3.20 添加自动发现设备

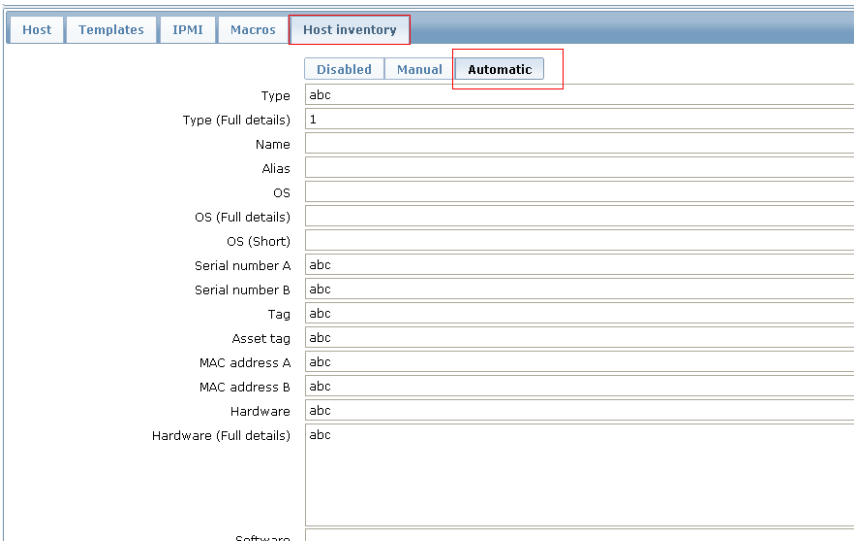
通过 zabbix 的相关设置，zabbix 可以自动添加设备，可以更友好的维护和添加相关设备。

详细参考：
http://www.zabbix.com/documentation/2.0/manual/discovery

3.21 添加 Inventory

Inventory 用来管理设备存档信息的。

在添加 host 时，勾选右侧的 Use profile，我们即可填入该台设备的型号、编码、MAC 地址等详细信息，勾选 Use extended profile 则可以填入更详细的信息。



填写的 inventory 信息在 inventory->hosts 下能看到.可以代替 OCS 一些功能。

3.22 Export/Import XML

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

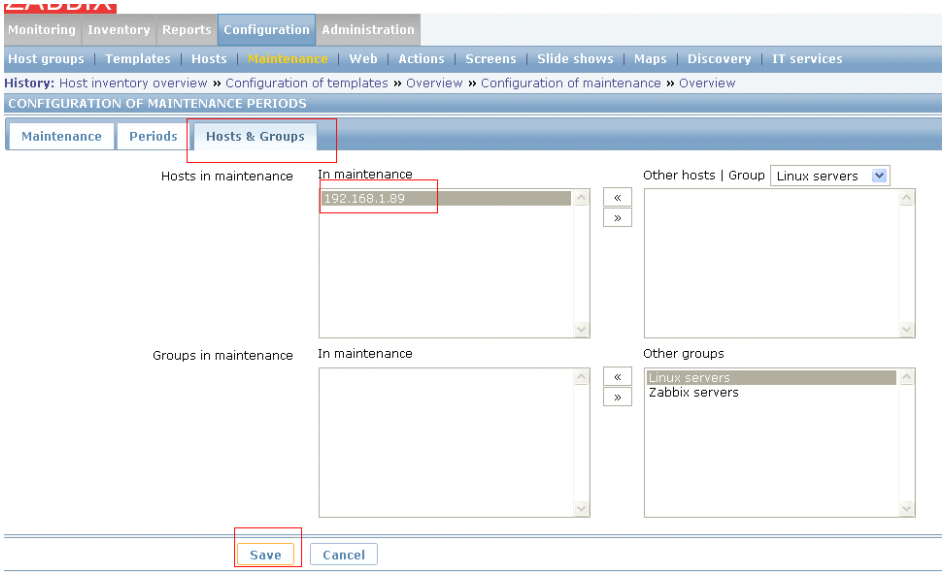
通过 configuration->Templates>Export/Import->Export,勾选要导出的模板，Preview 可以展示要导出的 host 的详细配置，选择 export 即可导出 xml 文件到本地。

3.23 Maintenance（维护时间）

这一点和 Nagios 的 Schedule downtime for this host 差不多，在 Nagios 中可以设置在 downtime 不需要告警，但是 zabbxi 设置的更加详细和可管理。

QQ 972322817 itnihao@qq.com

选择主机



注意，此功能的使用要结合 Action 功能一起使用

3.24 Proxy 的使用

zabbix 中文文档---代理（Proxies）

概述

一个 zabbix 代理（Proxies）可以代表 zabbix 服务器收集性能和可用性数据。这样，代理（Proxies）可以负担采集数据的任务并且减轻 zabbix 服务器负载 同时，使用代理（Proxies）是实施统一和分布式监控的最简单方式，因为所有的客户端和代理（Proxies）向一个 zabbix 服务器报告数据，并且所有数据集中保存在服务器数据库。

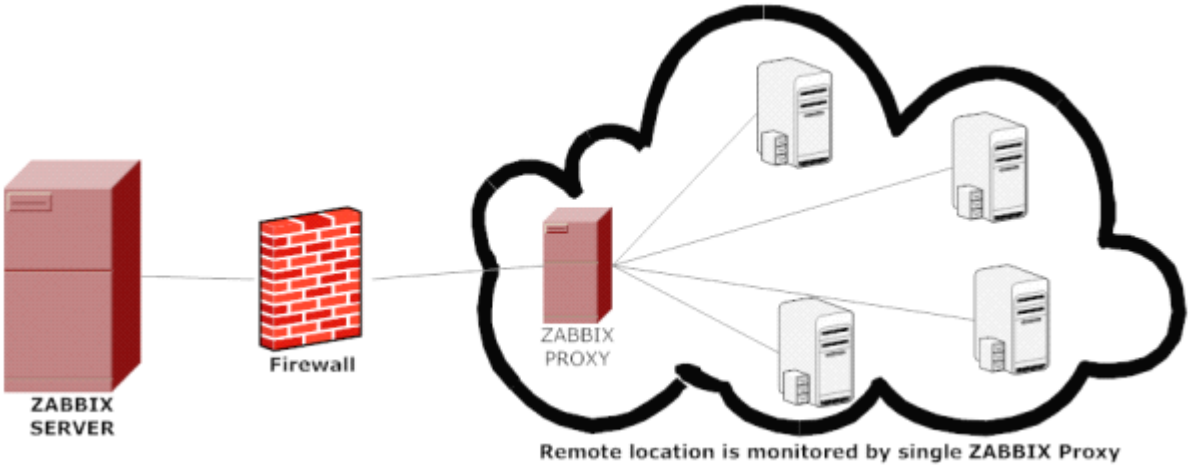
一个 zabbix 代理（Proxies）可以用在以下：

监控远程区域

监控拥有不可靠链接的区域

当监控数以千计的设备时分担 zabbix 服务器的负载

简化分布式监控的维护



代理（Proxies）与服务器之间仅需要一个 TCP 连接。这样将更容易避开防火墙因为你仅需要配置一条防火墙规则

zabbix 代理（Proxies）必须使用一个单独的数据库。代理（Proxies）执行 zabbix 服务器的数据库将打乱配置

所有代理（Proxies）采集到的数据在传送给服务器之前都保存在本地。这样，临时与服务器断开连接也不会导致数据丢失。proxy 配置文件中的参数 ProxyLocalBuffer 和 ProxyOfflineBuffer 控制数据在本地保存多久。

zabbix 代理（Proxies）是一个数据收集器。它不进行触发器计算，处理事件或发送报警信息。要了解代理（Proxies）的全部功能，查看下表

Function	Supported by

	proxy
Items	
Zabbix agent checks	**Yes**
Zabbix agent checks (active)	**Yes** [1]
Simple checks	**Yes**
Trapper items	**Yes**
SNMP checks	**Yes**
SNMP traps	**Yes**
IPMI checks	**Yes**
JMX checks	**Yes**
Log file monitoring	**Yes**
Internal checks	No
SSH checks	**Yes**
Telnet checks	**Yes**
External checks	**Yes**
Built-in web monitoring	**Yes**
Network discovery	**Yes**
Low-level discovery	**Yes**
Calculating triggers	No
Processing events	No
Sending alerts	No
Remote commands	No

为确保客户端代理（agent）实现连接代理服务器（不是 zabbix server）请求主动检测，代理服务器的 ip 地址必须出现在客户代理（Proxies）的参数 ServerActive 的值中

配置

一旦你安装并配置了一个代理服务器，现在该在 zabix 前端配置它了。

添加代理（Adding proxies）

要在 zabbix 前端配置代理，按如下步骤：

点击：高级配置 -> DM

在右上角的下拉列表框中选择 proxies

点击 Create proxy （或已经存在的代理服务器名称）

参数	描述
proxy name	输入代理服务器名称。它必须跟代理服务器配置文件中参数 Hostname 的值一样。
proxy mode	选择代理服务器的模式。 Active - 代理服务器将主动连接 zabbix 服务器并请求配置数据 被动模式 - zabbix 服务器连接代理服务器
hosts	添加被代理服务器监控的主机

主机配置

你可以在主机配置表单中使用 Monitored by proxy 字段指定应该被代理服务器监控的单个主机。

本节参考 http://www.linuxmr.com/2012/zabbix2_0817/265.html

3.25 创建 zabbix 的报警(以 postfix 为例子)

先看一下 zabbix 的运行流程

Host Groups(设备组)->Hosts(设备)->Applications(监控项组)->Items(监控项)->Triggers(触发器)->Actions(告警动作)->Medias(告警方式)->User Groups(用户组)->Users(用户)

在前面的内容中我们了解到了如何添加设备，如何写 items，triggers，但是没有涉及如何发送报警的问题，此处进行讲解。

3.22.1 创建 meida types

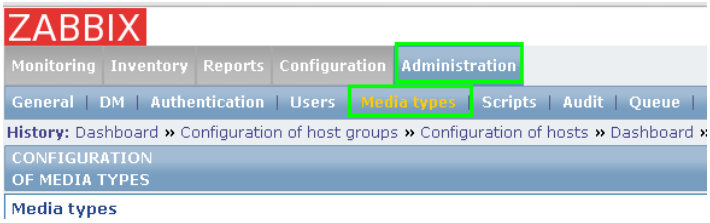
步骤：

登录 zabbix web 页面------Administration--------Media types--------Create Media Type(右上角)

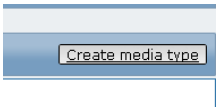
关于 Create Media Type 参数设置的说明

Parameter	Description
Description	Media 类型的名称
Type	选择类型为 Email
SMTP server	设置 SMTP 服务器来处理传出消息
SMTP helo	设置正确的 SMTP helo 值,通常是一个域名.

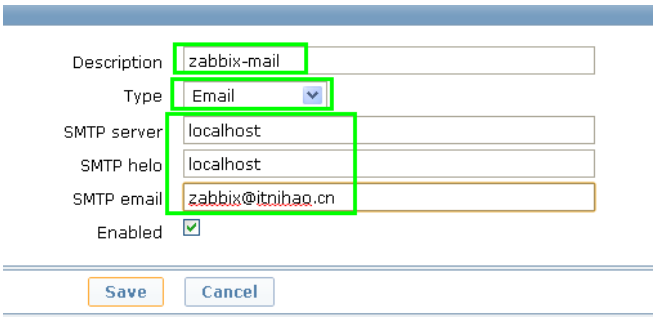
SMTP email	设置发送消息的 email 账号



点击右上角 Create Media Type



输入如下信息(此处采用本机的 postfix(sendmail)发送邮件，postfix 默认安装即可，无需其他设置，为了安全，监听端口应该绑定为127.0.0.1上)，当然，此处是没法设置有密码认证的邮箱，如需要，则可以采用脚本。



点击保存

此处内容的官方文档

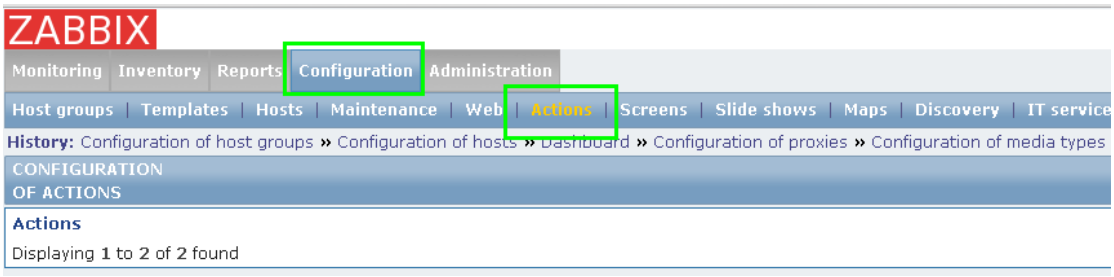
https://www.zabbix.com/documentation/2.0/manual/web_interface/frontend_sections/administration/mediatypes

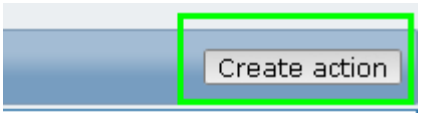
3.22.2 创建 actions

1.登录到 zabbix，进入"Configation" >> "Actions"（“系统配置”>>“操作”），点击右上角"Create Actions"。输入 Name “mysql_baojing” ，其它都默认点击右侧“Action Operations"下的"New"按钮，"Operation Type"选择"Send message"，"Send Message to"选择一个或多个要发送消息的用户组，”Send only to"选择我们之前新增的 mysql_baojing。

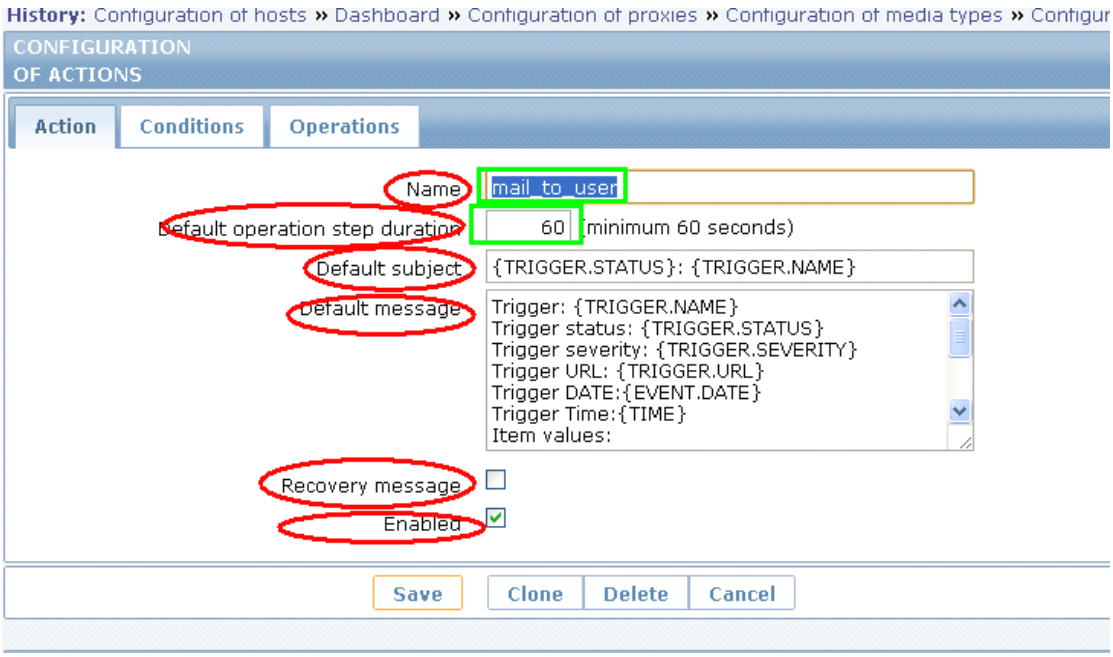
2.点击 save 保存

Zabbix 登陆-----Configation-------Actions-----------Create Actions------





点击 Action



依次填写

Name ：action 的名称

Default operation step duration ：间隔时间

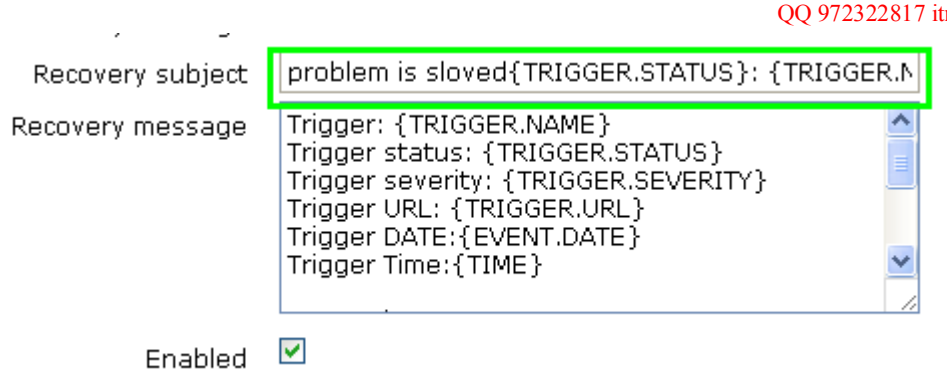
Default subject ：消息发送的默认主题

Default message ：休息发送的内容

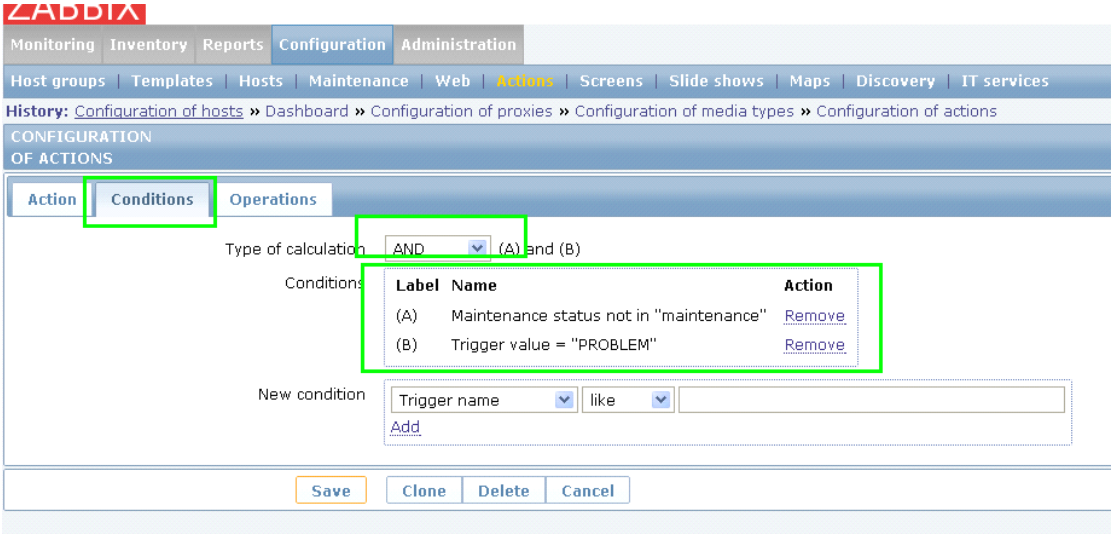
Recovery message ：故障恢复后的消息内容，这里不是必选的，主要用于故障恢复通知用户。如果不开启，则不会发送故障恢复的信息。

Enabled ：是否开启 action

Recovery message 这里勾选，则如下



点击 Conditions



Type of calculation ：逻辑值，and or

Conditions ：条件，此处可以设置多个条件，例子中的条件是不在维护时间，且故障值为 PROBLEM 则满足

New condition ：新的条件，如果想添加自己定义的条件，则点击 add 设置

发送的用户和组

参考 https://www.zabbix.com/documentation/2.0/manual/web_interface/frontend_sections/configuration/actions

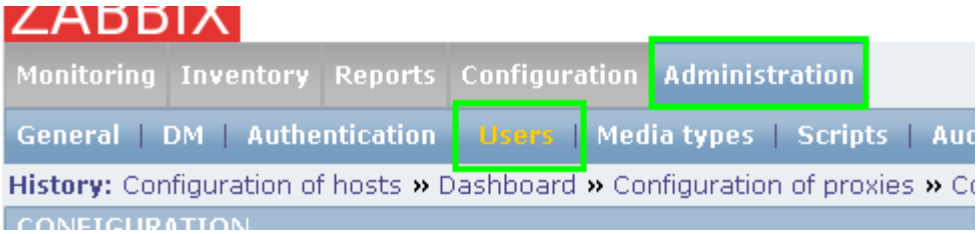
3.22.3 zabbix 用户配置

登录到 zabbix====Adimistration=====Users======Admin 用户。

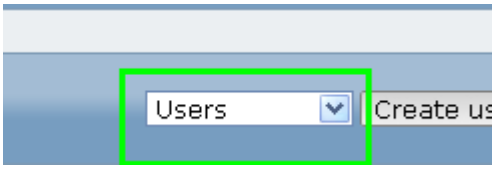
在用户信息修改界面最下方的”Media"处点击"Add"按钮。

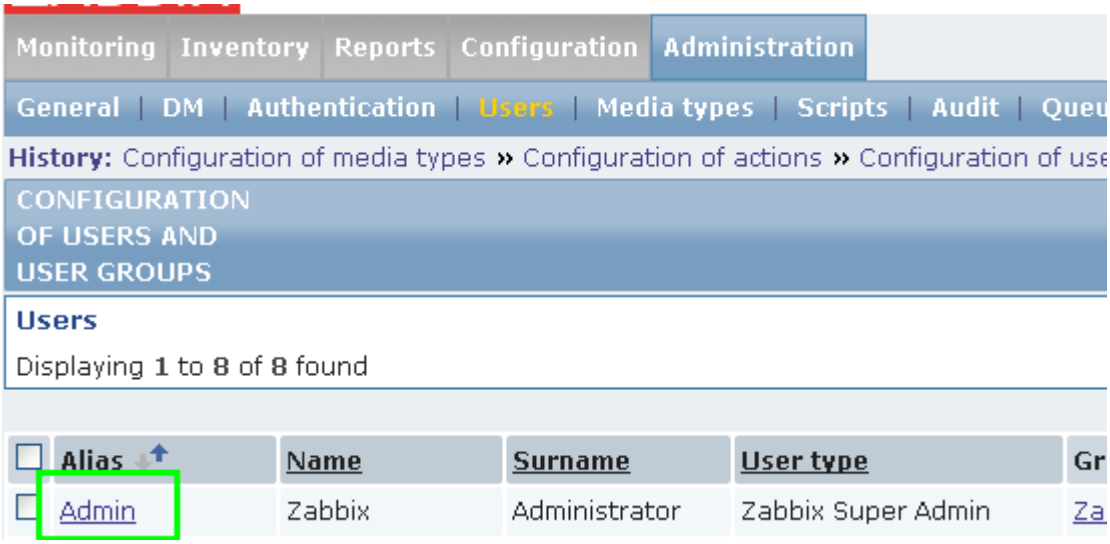
Type 选择"Email"，Send to 填入收件人地址，点击 Add 添加。

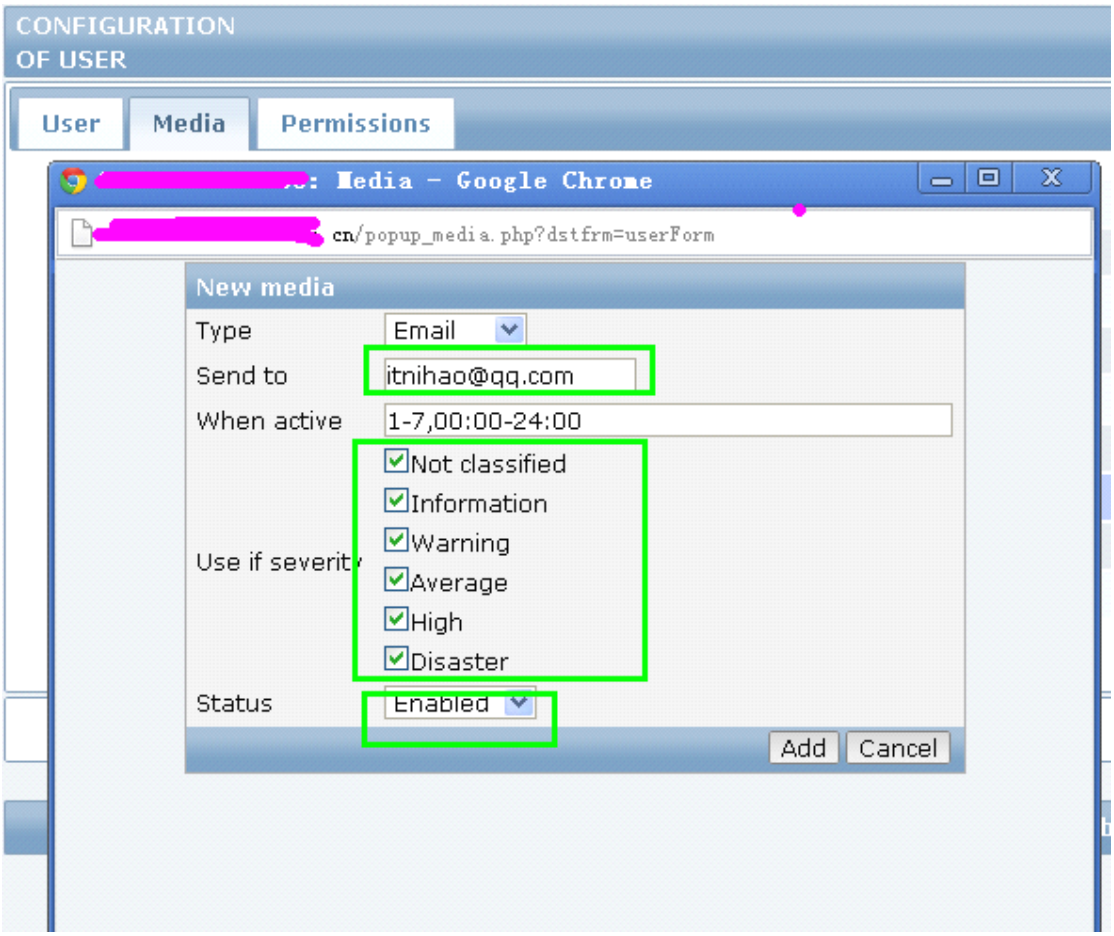
点击"Save"保存配置。



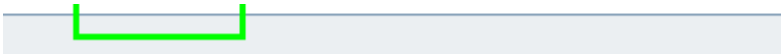
右上角







点击 add 添加保存



除了以上功能，还有用户登陆的默认 url，语言选项配置。

3.26 创建脚本报警（以 mail.py 为例）

此处借用 mail.py 来说明脚本发送邮件的问题，但是不建议用网上公用邮箱--为何？网上公用大部分都有连接数量限制，发送邮件过多，会被服务器拒绝。因此，尽量用自己的邮件服务器发送信息。

这部分内容直接转载 lihuipeng 博客，地址为 http://lihuipeng.blog.51cto.com/3064864/1066915

1、Zabbix 添加处理方法：管理——处理方法——create media type

zabbix 会传给脚本三个参数：接收用户，邮件主题，邮件内容

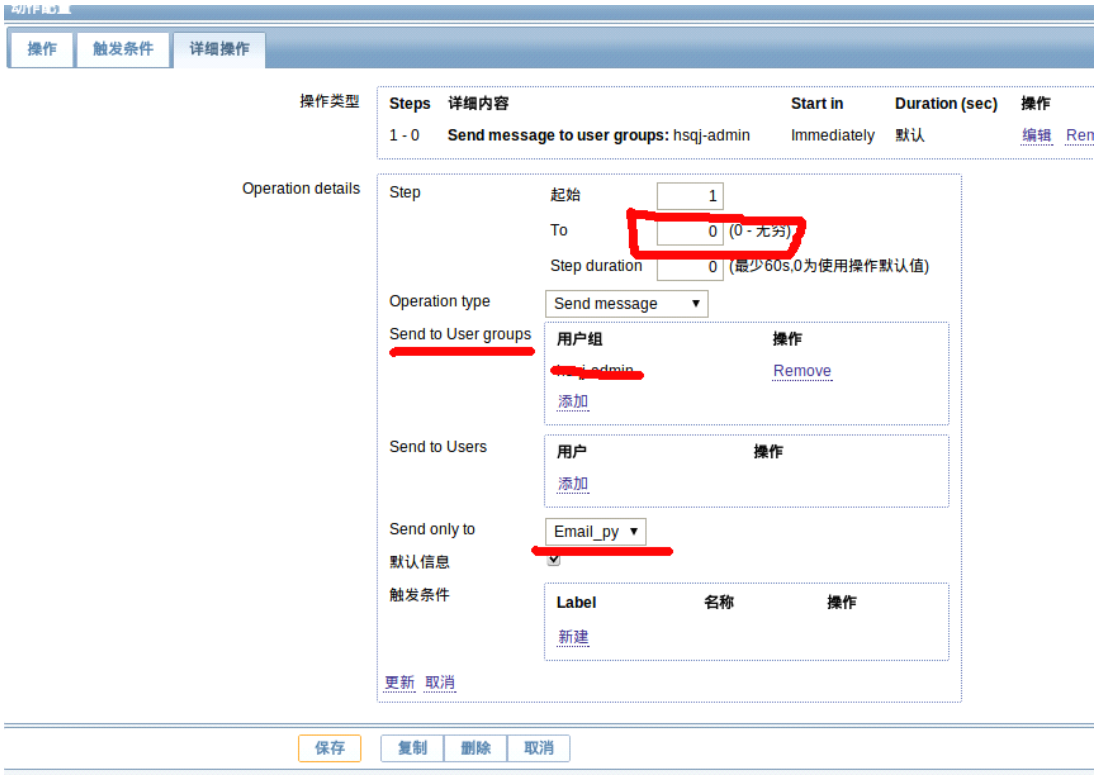
cat /etc/zabbix/alertscripts/zabbix_sendmail.py #注意此文件要具有执行权限

```
#!/usr/bin/python
#coding:utf-8

import smtplib
from email.mime.text import MIMEText
import sys

mail_host = 'smtp.163.com'
mail_user = 'monitor_itnihao'
mail_pass = 'my_password'
mail_postfix = '163.com'

def send_mail(to_list,subject,content):
    me = mail_user+"<"+mail_user+"@"+mail_postfix+">"
    msg = MIMEText(content)
    msg['Subject'] = subject
    msg['From'] = me
    msg['to'] = to_list

    try:
        s = smtplib.SMTP()
        s.connect(mail_host)
        s.login(mail_user,mail_pass)
        s.sendmail(me,to_list,msg.as_string())
        s.close()
        return True
    except Exception,e:
        print str(e)
        return False

if __name__ == "__main__":
    send_mail(sys.argv[1], sys.argv[2], sys.argv[3])
```

2、添加触发设置：系统配置——操作——create action


动作配置
操作
触发条件
详细操作
名称
Default operation step duration
60
(最少60秒)
默认主题
{TRIGGER.STATUS}: {TRIGGER.NAME}
默认信息
Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}
Item values:
Recovery message
Recovery subject
{TRIGGER.STATUS}: {TRIGGER.NAME}
Recovery message
Trigger: {TRIGGER.NAME}
Trigger status: {TRIGGER.STATUS}
Trigger severity: {TRIGGER.SEVERITY}
Trigger URL: {TRIGGER.URL}
Item values:
活跃
保存
复制
删除
取消


动作配置
操作
触发条件
详细操作
计算方式
且
(A) 与 (B) 与 (C)
触发条件
Label
名称
操作
(A)
维护状态 不在 "维护"
Remove
(B)
Trigger value = "PROBLEM"
Remove
(C)
主机组 =
Remove
新的触发条件
Trigger name
相似
添加
保存
复制
删除
取消



3、zabbix_server 添加脚本配置：

mkdir -p /etc/zabbix/alertscripts/

把脚本上传到该目录

修改 zabbix_server.conf 配置：

AlertScriptsPath=/etc/zabbix/alertscripts/

然后重启服务

#注意：以上内容部分做过修改，路径等的修改，请阅读源博客
http://lihuipeng.blog.51cto.com/3064864/1066915

3.27 如何有效的设置监控报警

对于监控的报警信息，处理的好，将会提高我们的故障响应速度，处理的不好，会影响我们的工作情绪，适得其反。试想，当一天收到 1000 封报警信息，是否还会去逐一查看监控报警信息？是否还能分辨是否重大故障，还是一般故障？

对于误报，漏报，会让人对信息的警觉性放松，时间久了，还会导致对接收监控信息有反感。所以，对于监控报警信息的发送，是一件特别慎重的事情。总结一下，对于监控报警信息，我们有以下的需求：

1. 基于业务类型，将报警信息发送给相应的业务用户，例如 IDC 人员，web 运维，cdn 运维，不同的人管理不同的机器，因此需要把故障发送给相关用户处理

2. 基于故障级别，对一个故障，将不同的故障级别发送给不同用户，例如 5 分钟内的故障发送给运维一线人员，10 分钟发送给运维部门主管，30 分钟发送给运维部门经理。重特大故障发送市场部门相关领导。

3. 基于时间发送，比如业务维护期，报警无需发送。

4. 故障的相关依赖关系，当 A 服务发生故障时，发送一般报警，当 A，B 服务故障时候，发送业务故障报警。

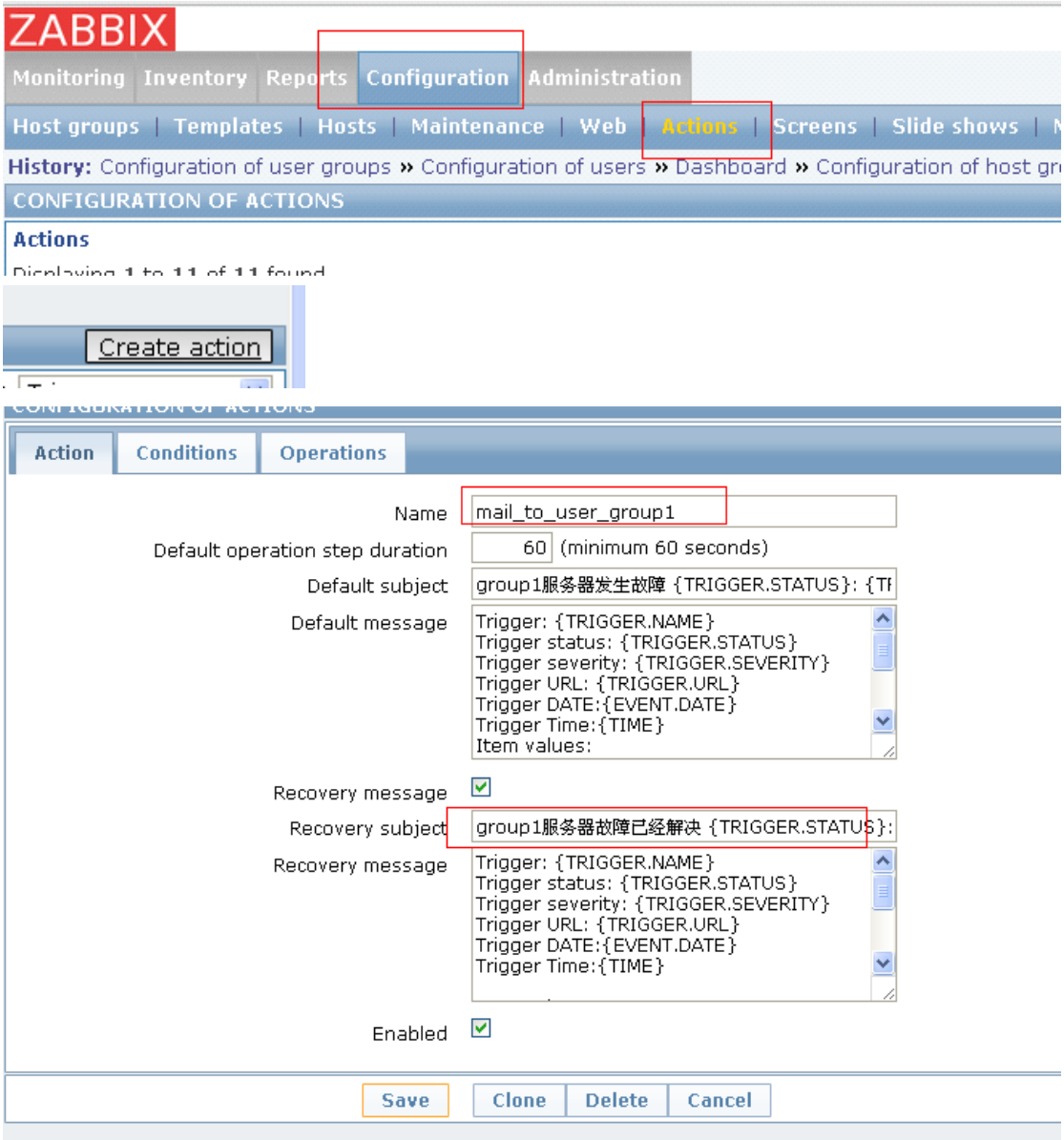
5.对出现故障的服务尝试用相关脚本进进行操作处理，例如重启等。

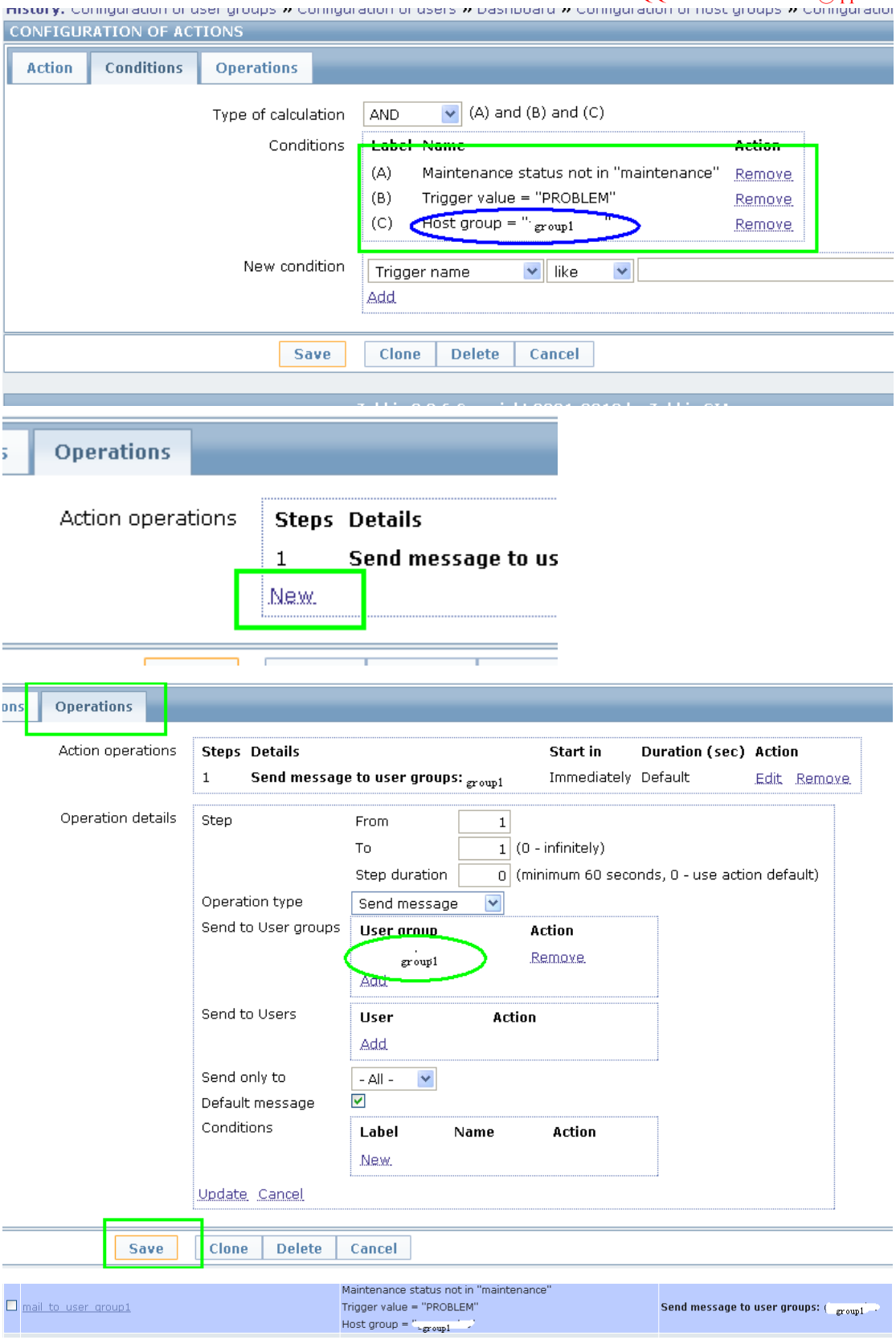
3.27.1 基于业务类型

将不同的用户，分配到不同的用户组

Action 里面定义不同主机组发送信息给对应的用户组

Action 里面可以为不同的服务器组创建不同的 action，因此就可以基于对业务类型的机器分组进行报警信息发送

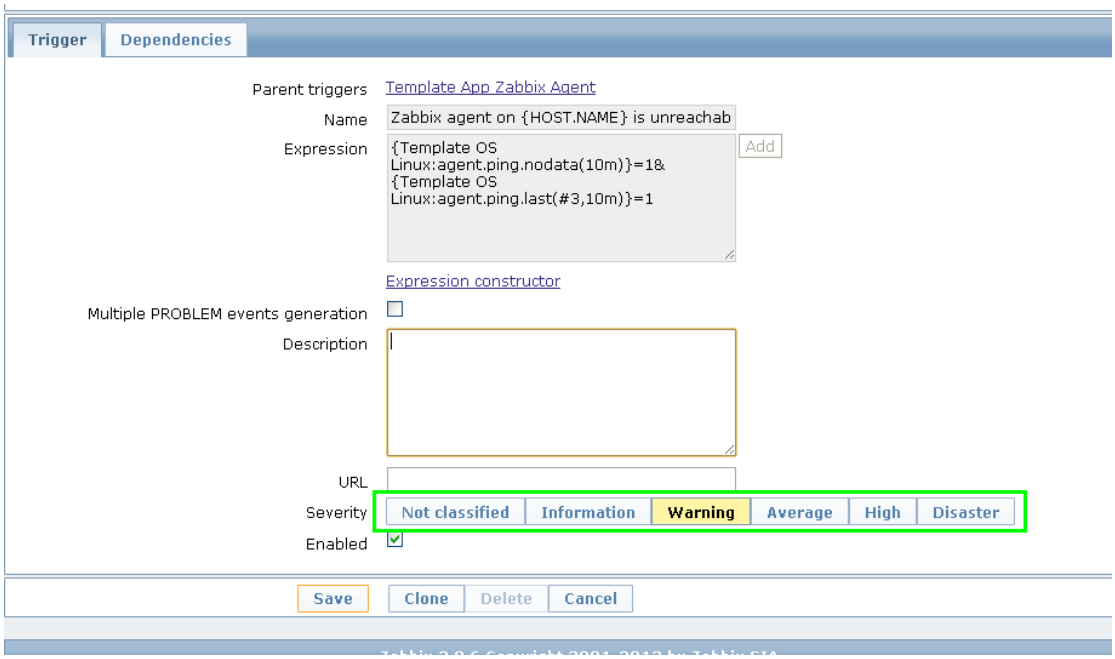




当然，用户组里面的用户需要在用户管理里面定义，此处略(用户管理内容个人觉得比较简单，就不单独写了)

3.27.2 基于故障级别

在定义触发器的时候，有故障级别的选择



可以对不同故障分级别定义

在对用户添加报警媒体的时候，可以定义不同故障等级发送

例如触发器如下

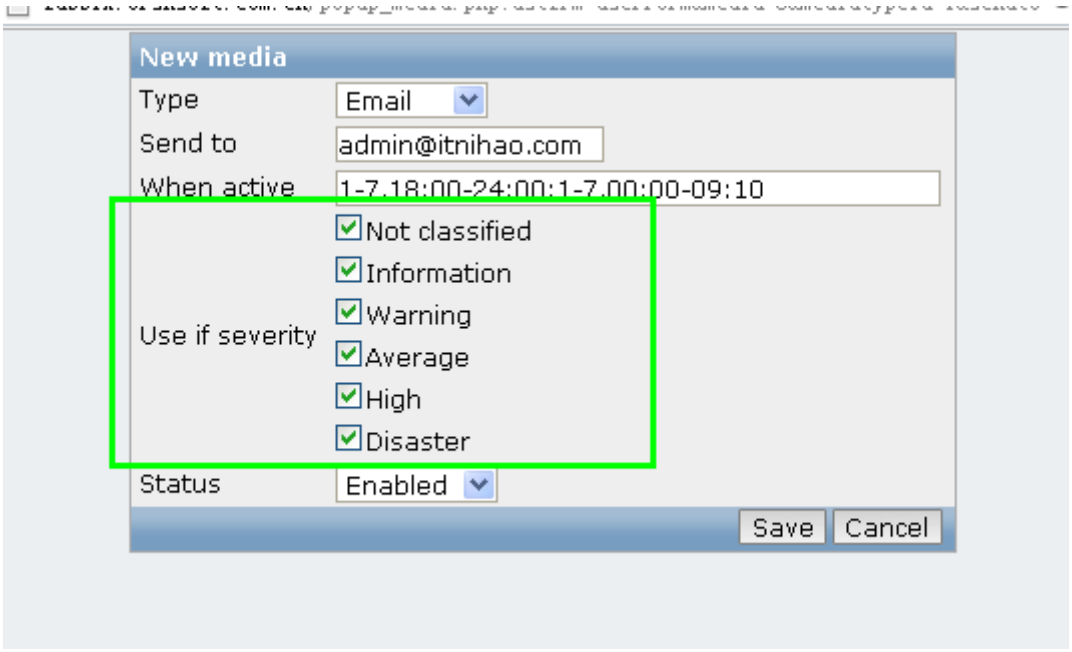
{Template OS Linux:agent.ping.nodata(5m)}=1&{Template OS Linux:agent.ping.last(#3,5m)}=1

{Template OS Linux:agent.ping.nodata(10m)}=1&{Template OS Linux:agent.ping.last(#3,10m)}=1

{Template OS Linux:agent.ping.nodata(15m)}=1&{Template OS Linux:agent.ping.last(#3,15m)}=1

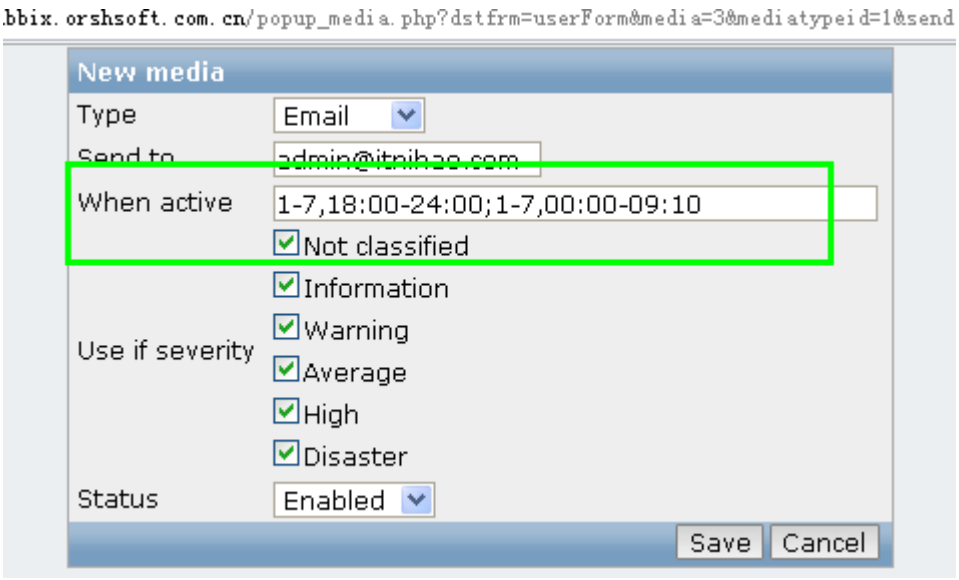
{Template OS Linux:agent.ping.nodata(30m)}=1&{Template OS Linux:agent.ping.last(#3,30m)}=1

然后将不同触发器标记为不同故障级别



基于以上 2 个功能，完全可以做到将不同的故障级别发送给不同的人

3.27.3 基于时间发送



时间可以为 1-7,00：00-24:00 各个时间段之间用分号隔开，可以创建任意的时间段，实际情况为根据需要而定

3.27.4 故障依赖关系

{Template OS Linux:agent.ping.nodata(5m)}=1&{Template OS Linux:agent.ping.last(#3,5m)}=1
例如以上需要 2 个值同时成立才会触发

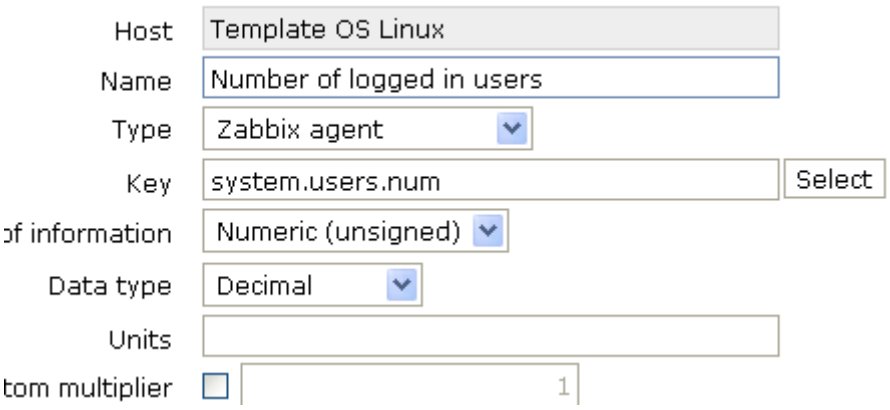
3.27.5 故障处理自动远程命令

除了发送消息，还可以执行远程命令，具体方法参考官方文档

3.28 一些使用的技巧

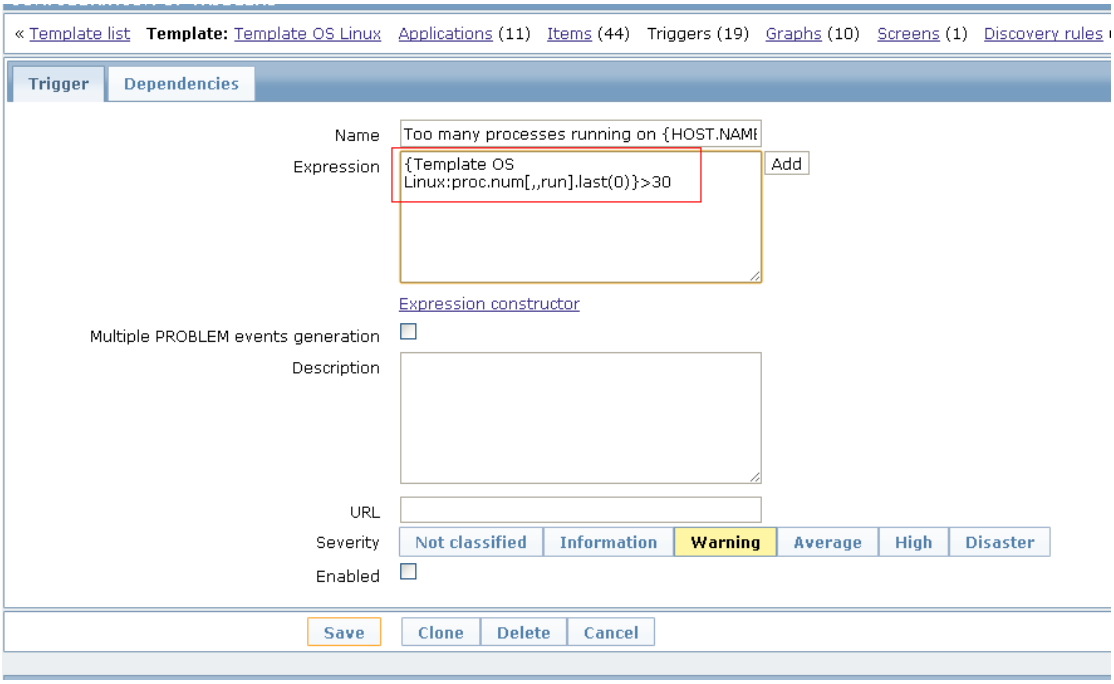
3.28.1 监控项的使用技巧

Agend 已经带大量监控项，某些监控项并没有添加到 Graphs 里面，需要手动添加即可
例如 system.users.num，需要添加图形(如何添加 Graphs 参考前面章节)



3.28.2 触发器的使用技巧

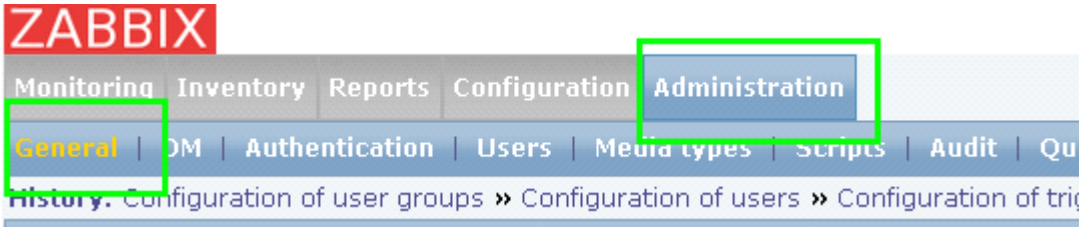
默认的一些触发器由于触发值设置不合理，需要修改后才能适合自己的生产环境



例如此处{Template OS Linux:proc.num[,,run].last(0)}>30 大于 30 即报警，由于服务器应用的不同，需要将此值进行合理设置，否则会产生大量误报。

3.28.3 定义全局变量的使用技巧

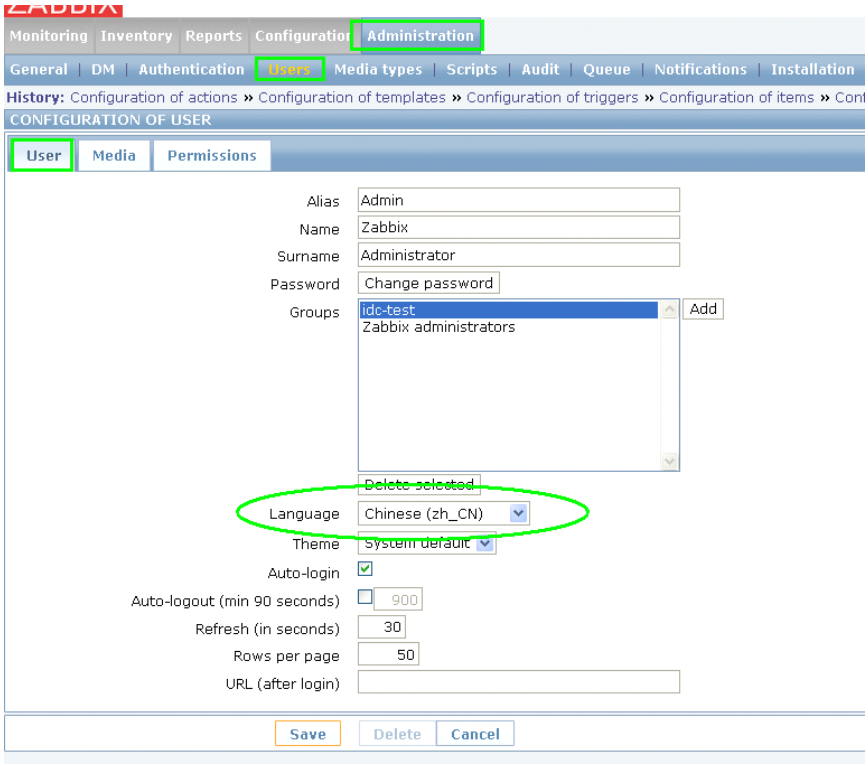
3.28.4.1Snmp 团组名的设置
当使用 snmp 作为客户端监控的时候，需要定义团组名



添加单台主机的时候，也可以定义对单台主机有效的变量

3.28.4 中文语言显示以及中文字体乱码解决方法

如何显示中文语言



如何解决将语言改为中文后乱码的问题

方法 1

将

(1) export ZABBIX_WEB=/var/www/html/zabbix/

(2)wget https://www.zabbix.org/pootle/export/Zabbix-2.0/zh_CN/LC_MESSAGES/frontend.po -O ${ZABBIX_WEB}/locale/zh_CN/LC_MESSAGES/frontend.po

(3)cd ${ZABBIX_WEB}/locale/ && ./make_mo.sh .

(4)刷新图形，看看是否已经恢复正常

方法 2.

将 windows 下的中文字体上传到/var/www/html/zabbix/fonts 下面，替换原文件(与原文件同名)

4. Zabbix 的高级使用-之自动化功能

使用 zabbix 的目的是让监控实现自动化，那么究竟如何才能达到自动化功能呢？首先，zabbix 提供了主机自动发现功能，当客户端安装好之后，zabbix 有自动发现添加主机的功能，其次，zabbix 提供了对多变的监控项目自动发现监控，例如本身有 2 个网卡，新增加 2 个网卡，新增的 2 个网卡会自动监控。基于 zabbix 的这 2 个功能，我们可以做到对服务的自动化监控，从此告别手动添加监控项的痛苦，相信用 cacti 和 nagios 的都有添加监控到手抽筋的经历吧。

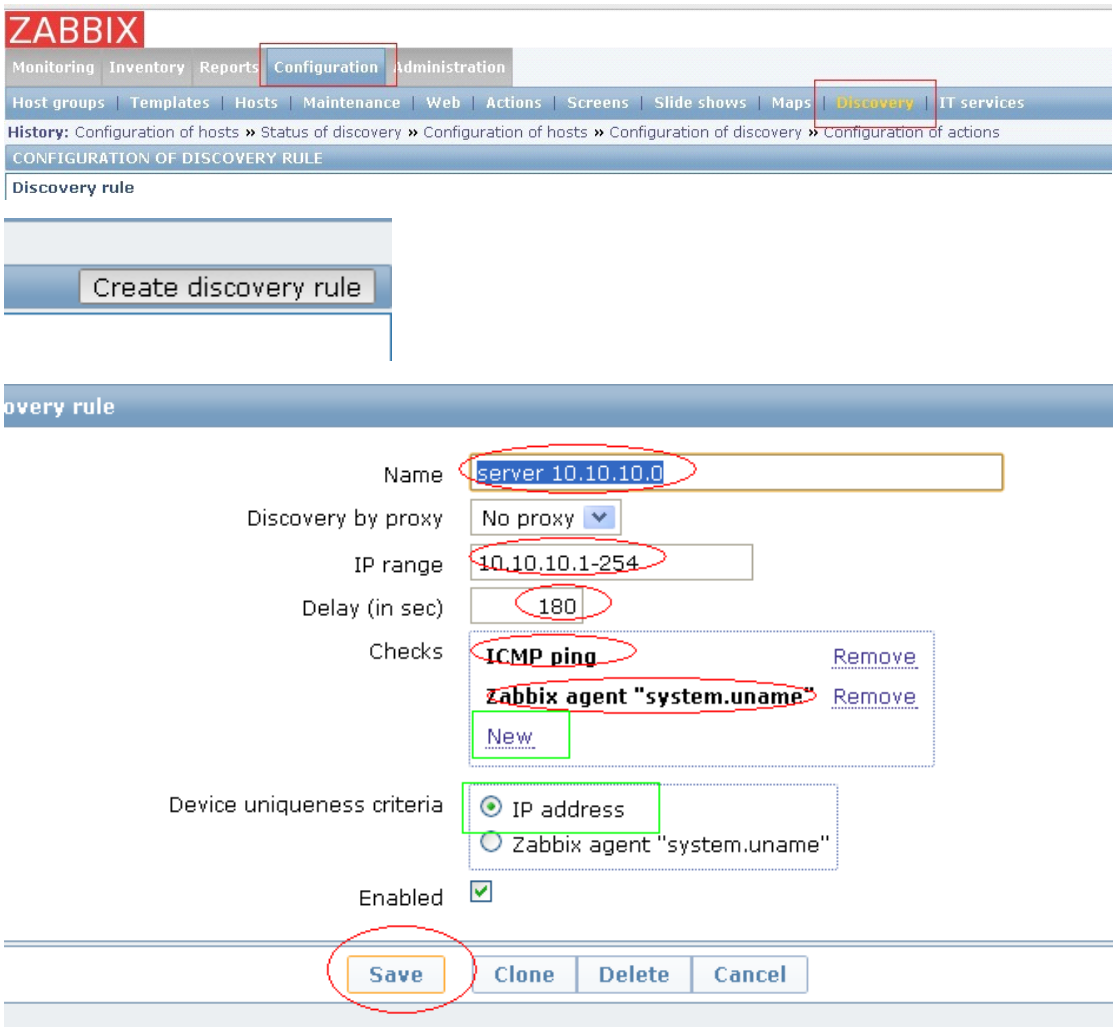
4.1 自动发现添加主机

功能介绍：

官方文档已经给出了很详细的步骤，在此演示一下过程

https://www.zabbix.com/documentation/2.0/manual/discovery/network_discovery/rule

4.1.1 创建自动发现规则



Names：名称，可以写自己能代表服务功能的名称，便于识别

Discovery by proxy：是否通过代理

IP range：ip 地址的范围，可以写一段地址，也可以写多段地址

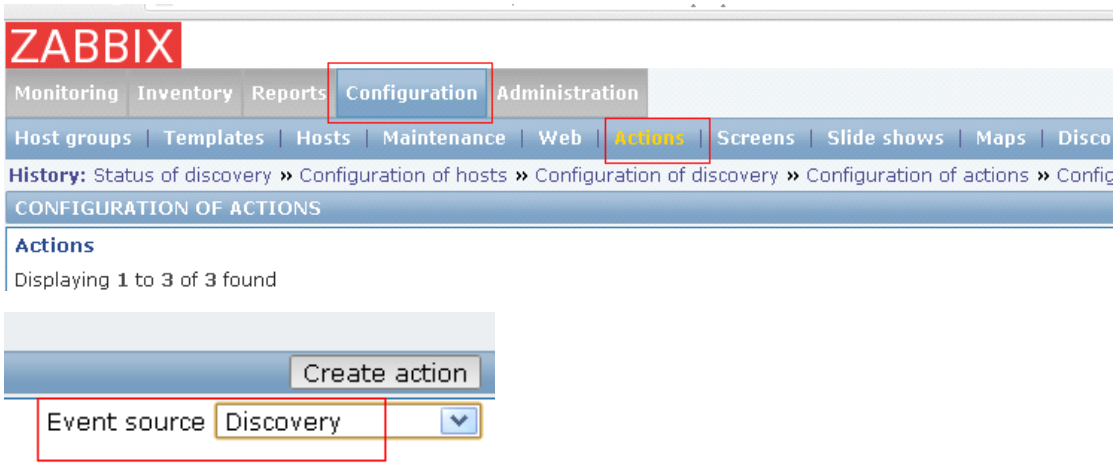
Delay：检测时间周期，注意这个值默认是 3600，即一个小时才能发现服务

Checks:检测命令，这里选择 ICMP，zabbix agentd 来检测

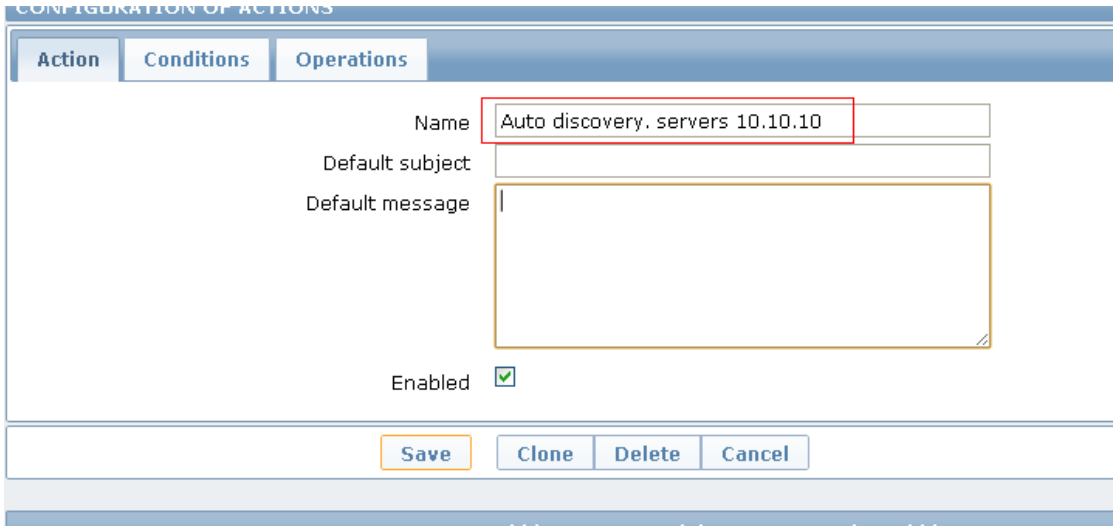
Device uniqueness criteria：设备唯一的名称，此处采用 ip

Enabled：发现功能是否激活

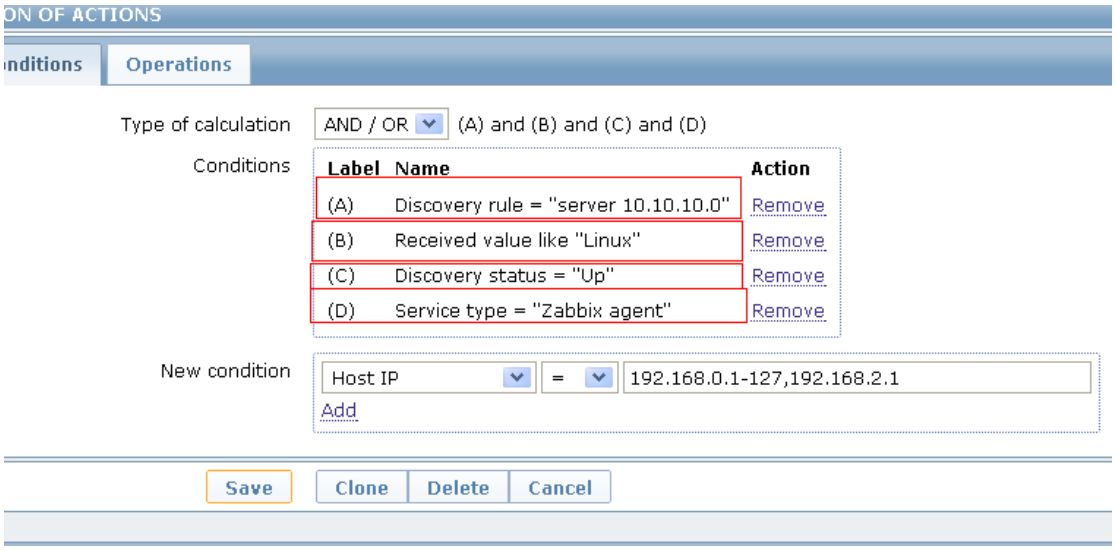
4.1.2 创建自动添加到相应模板规则



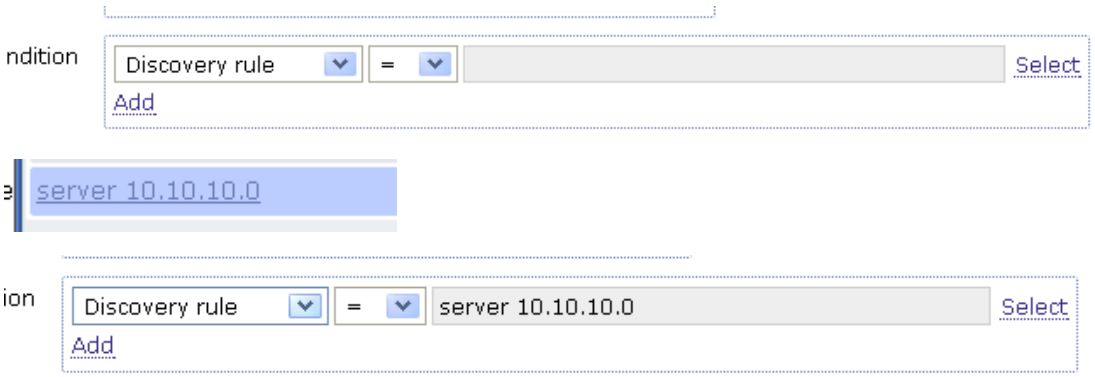
选择 Discovery,



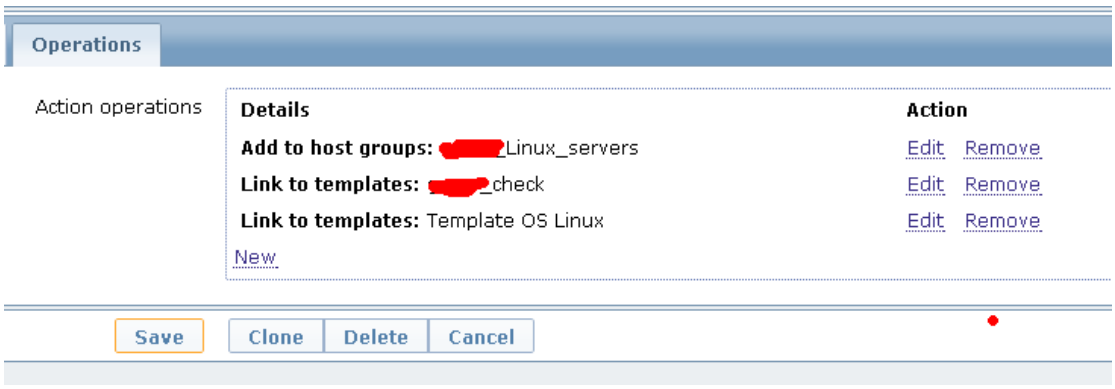
Name 填写业务相关的 Action 名称

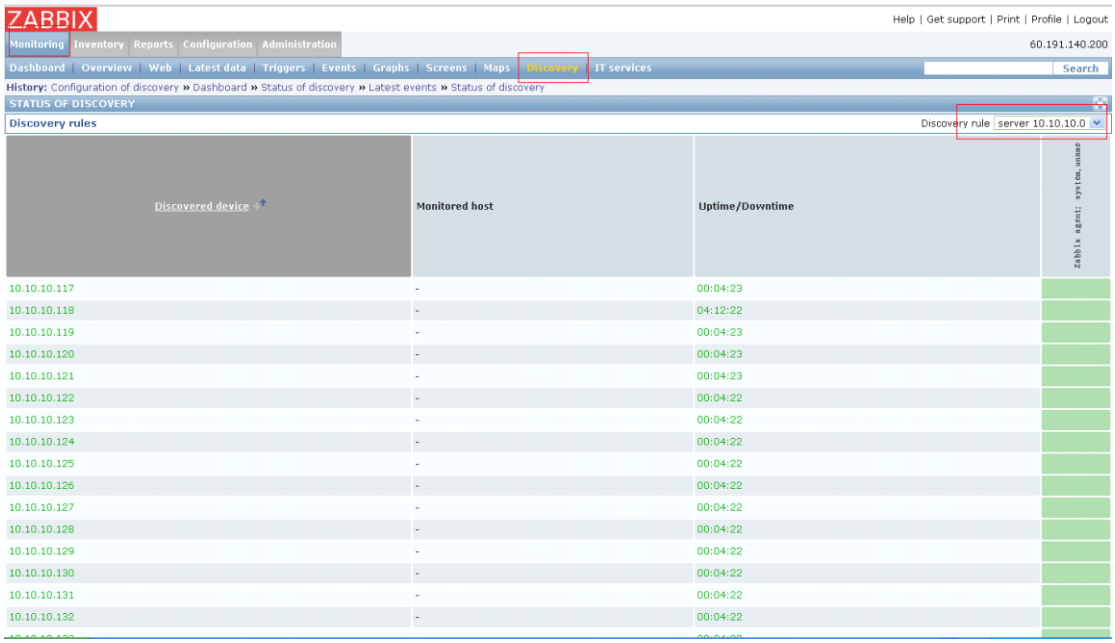


注意：Discovery rule 是添加前面定义的 Discovery rule 项目



此处是添加从自动发现规则里面发现的主机，包含这些规则，会进行下一步的操作





这里是对上面规则的进一步添加到应用，包含上面规则的，将会自动添加到分组，并用指定的模板。此处可以实现的功能，当我们发现主机包含某些规则后，可以用相关的分组，相关份额模板添加监控项。

需要注意的是，选择相关选项后，需点击 add 后，然后才可以保存。

4.2 通过 low-level discovery 发现实现动态监控

自动化运维之监控篇---利用 zabbix 自动发现功能实现批量 web url 监控

需求：

现在有大量 url 需要监控，形式如 http://itnihao.blog.51cto.com ，要求 url 状态不为 200 即报警。

需求详细分析：

大量的 url，且 url 经常变化，现在监控用的是 zabbix，如果手动添加模板，会造成大量重复工作，造成人力财力的浪费，造成休息时间的浪费，得不尝失，如果利用脚本+mail，无法图形呈现

解决方案：

zabbix 有 discovery 功能，利用此功能，即可轻松解决此问题

存在的文件如下

```
/etc/zabbix/zabbix_agentd.conf
/etc/zabbix/scripts/web_site_code_status
/etc/zabbix/scripts/WEB.txt
/etc/zabbix/zabbix_agentd.conf.d/web_site_discovery.conf
```

4.2.1 zabbix 客户端配置

```
1.  #########cat /etc/zabbix/zabbix_agentd.conf|grep -v "^#"|grep -v "^$"##########
```

##此处省略 N 多信息，

Include=/etc/zabbix/zabbix_agentd.conf.d/ #配置文件路径

UnsafeUserParameters=1 #自定义 key 里面可以包括特殊字符

4.2.2 自动发现脚本编写

Low-level discovery 的脚本是一个 json 格式，鉴于大多数童鞋习惯使用 shell，故此处采用 shell 来书写，如用 perl，python，则代码会更简洁

```
#######################cat /etc/zabbix/scripts/web_site_code_status#############
#!/bin/bash

# function:monitor tcp connect status from zabbix
# License: GPL
# mail:itnihao@qq.com
# version:1.0 date:2012-12-09
source /etc/bashrc >/dev/null 2>&1
source /etc/profile  >/dev/null 2>&1

#/usr/bin/curl -o /dev/null -s -w %{http_code} http://$1/
WEB_SITE_discovery () {
```

```
WEB_SITE=($(cat  WEB.txt|grep -v "^#"))
        printf '{\n'
        printf '\t"data":[\n'
for((i=0;i<${#WEB_SITE[@]};++i))
{
        num=$(echo $((${#WEB_SITE[@]}-1)))
        if [ "$i" != ${num} ];
                then
        printf "\t\t{ \n"
        printf "\t\t\t\"{#SITENAME}\":\"${WEB_SITE[$i]}\"},\n"
                else
                        printf  "\t\t{ \n"
                        printf  "\t\t\t\"{#SITENAME}\":\"${WEB_SITE[$num]}\"}]}
\n"
        fi
}
}
web_site_code () {
/usr/bin/curl -o /dev/null -s -w %{http_code} http://$1
}
case "$1" in
web_site_discovery)
WEB_SITE_discovery
;;
web_site_code)
web_site_code $2

;;
*)
echo "Usage:$0 {web_site_discovery|web_site_code [URL]}"
;;
esac
```

输出格式如下

```
{
        "data":[
                {
                        "{#SITENAME}":"www.qq.com"},
                {
                        "{#SITENAME}":"www.baidu.com"},
                {
                        "{#SITENAME}":"www.sina.com.cn"}]}
```

如果此处采用 python 来书写，则代码如下

```
#!/usr/bin/env python
# coding=utf8
```

```
# Last modified: 2013-04-12 14:47
# Author: itnihao
# Mail: itnihao@qq.com

import os
import json

r=open('WEB.txt','r').read().split()
devices = []

for devpath in  r:
        device = os.path.basename(devpath)
        devices += [{'{#SITENAME}':device}]

print json.dumps({'data':devices},sort_keys=True,indent=7,separators=(',',':'))
```

输出格式如下

```
{
        "data":[
                {
                        "{#SITENAME}":"www.qq.com"
                },
                {
                        "{#SITENAME}":"www.baidu.com"
                },
                {
                        "{#SITENAME}":"www.sina.com.cn"
                }
        ]
}
```

4.2.3 自定义 key 配置文件

#########cat /etc/zabbix/zabbix_agentd.conf.d/web_site_discovery.conf ###########

UserParameter=web.site.discovery,/etc/zabbix/scripts/web_site_code_status web_site_discovery

UserParameter=web.site.code[*],/etc/zabbix/scripts/web_site_code_status web_site_code $1

域名如下

######### cat /etc/zabbix/scripts/WEB.txt###########

www.qq.com

www.baidu.com

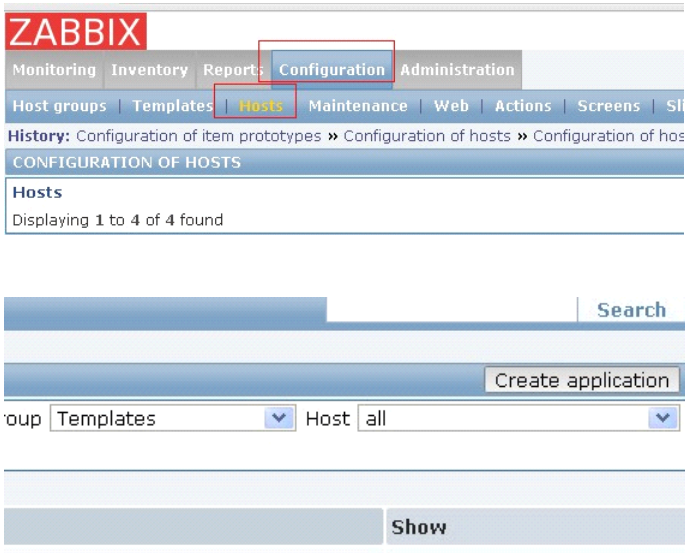
www.sina.com.cn

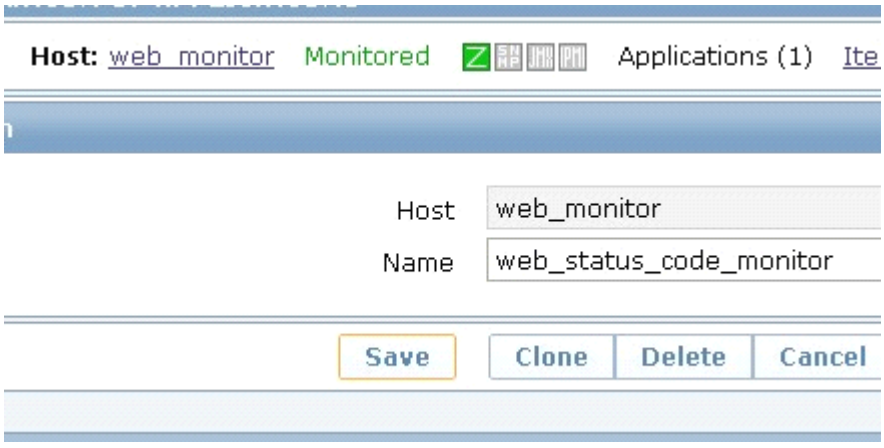
测试：

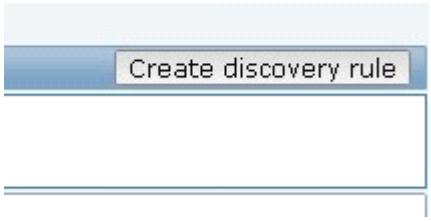
zabbix_get -s 127.0.0.1 -k web.site.discovery

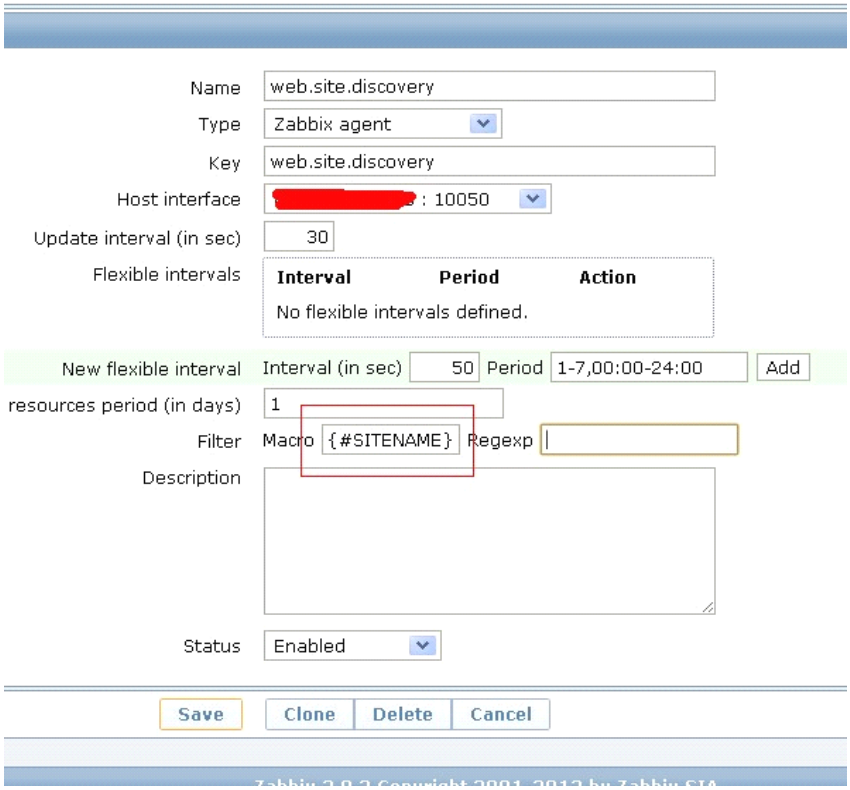
zabbix_get -s 127.0.0.1 -k web.site.code[www.qq.com]
此时返回状态为 200
至此，脚本，客户端配置文件 OK

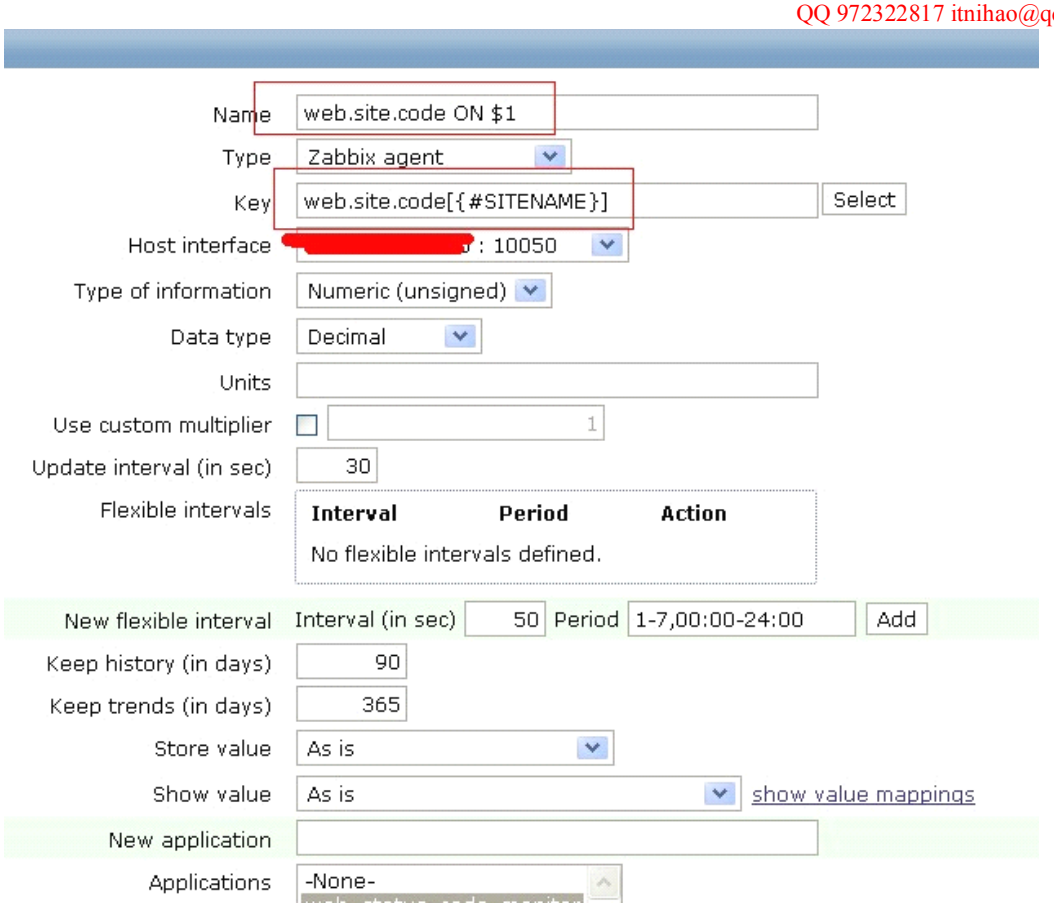
4.2.4 web 页面添加 low-level discovery



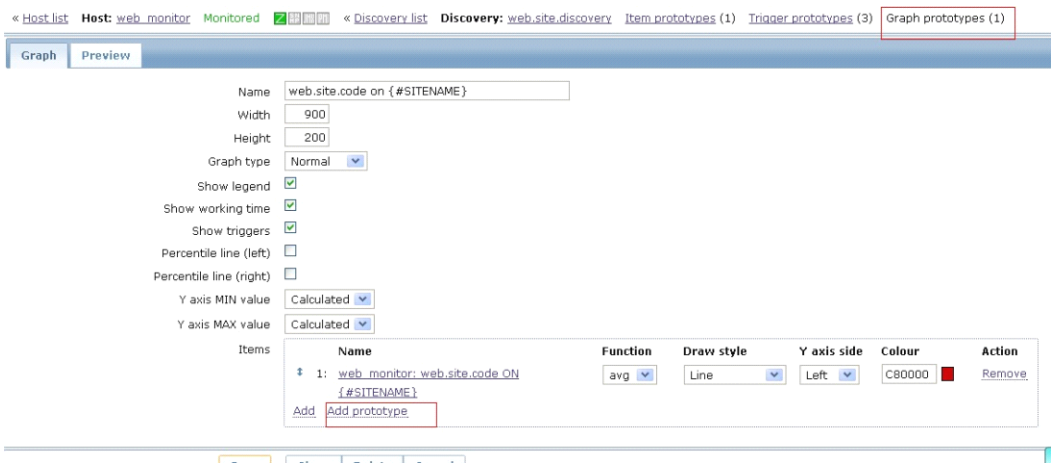


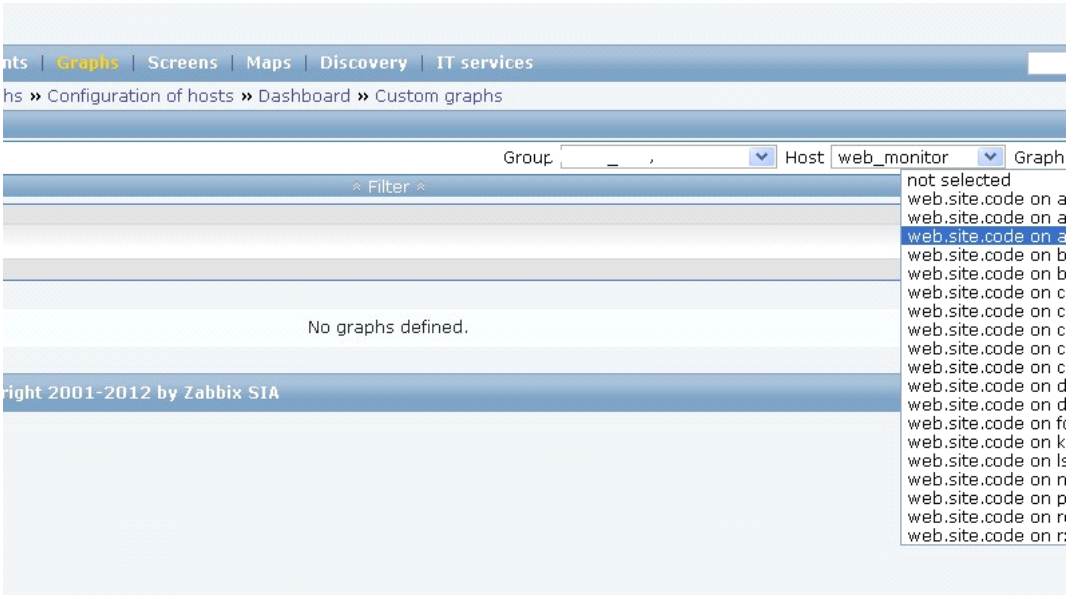

Create discovery rule


Name
web.site.discovery
Type
Zabbix agent
Key
web.site.discovery
Host interface
: 10050
Update interval (in sec)
30
Flexible intervals
Interval
Period
Action
No flexible intervals defined.
New flexible interval
Interval (in sec)
50
Period
1-7,00:00-24:00
Add
resources period (in days)
1
Filter
Macro
{#SITENAME}
Regexp
Description
Status
Enabled
Save
Clone
Delete
Cancel



报警级别的设置：3 个报警级别





利用以上方法，哪怕是 1000 个 url 监控，也是分分钟的事情
总结：工欲善其事，必先利其器
参考文档：https://www.zabbix.com/documentation/2.0/manual/discovery/low_level_discovery
此部分请详细查看官方文档的讲解，此处只是将官方文档实例化。

5. zabbix 性能优化

5.1 Zabbix 优化指标

5.2 zabbix_server.conf 优化

5.3 zabbix_agend.conf 优化

5.4 Zabbix 数据库优化

5.5 zabix 集群扩展的使用

本部分内容转载
http://blog.sina.com.cn/s/blog_704836f40101g0fb.html
本文是我综合了参考资料中的一大堆东西捏在一起的，单纯是翻译整理了一下，另外配了一些图，其他基本没有什么原创内容。

1.如何度量 Zabbix 性能

通过 Zabbix 的 NVPS(每秒处理数值数)来衡量其性能。
在 Zabbix 的 dashboard 上有一个错略的估值。

Status of Zabbix		
Parameter	**Value**	**Details**
Zabbix server is running	Yes	192.168.2.
Number of hosts (monitored/not monitored/templates)	215	182 / 1 / 3
Number of items (monitored/disabled/not supported)	55983	52854 / 25
Number of triggers (enabled/disabled)[problem/unknown/ok]	6443	6443 / 0 [
Number of users (online)	4	1
Required server performance, new values per second	640.08	-

在4核 CPU，6GB 内存，RAID10(带有写入缓存)这样的配置条件下,Zabbix 可以处理每分钟 1M 个数值,大约每秒15000个。

2.性能低下的可见症兆

1. zabbix 队列中有太多被延迟的 item： Administration -> Queue
2. zabbix 绘图中经常性出现断档，一些 item 没有数据
3. 带有 nodata()函数的触发器出现 false
4. 前端页面无响应

3.哪些因素造成 Zabbix 性能低下

因素	慢	快
数据库大小	巨大	适应内存大小
触发器表达式的复杂程度	Min(),max(),avg()	Last(),nodata()
数据收集方法	轮讯(SNMP,无代理,Passive 代理)	Trapping(active 代理)
数据类型	文本，字符串	数值
前端用户数量	多	少

主机数量也是影响性能的主要因素

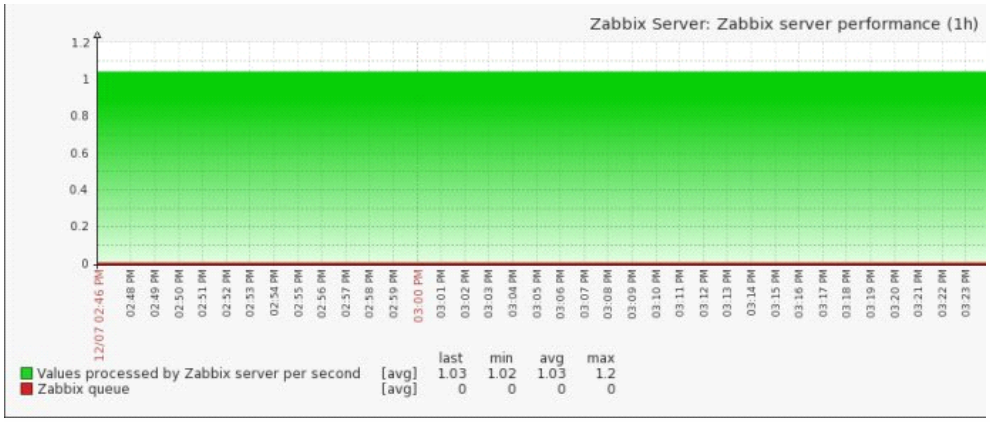
	主机数量	性能(NVPS)
每个主机60个item 每分钟更新一次	10	10
	100	100
	1000	1000
每个主机600个item 每分钟更新一次	10	100
	100	1000
	1000	10000

4.了解 Zabbix 工作状态

获得 zabbix 内部状态

zabbix[wcache,values,all]

zabbix[queue,1m]　　----延迟超过1分钟的 item



获得 zabbix 内部组件工作状态(该组件处于 BUSY 状态的时间百分比)

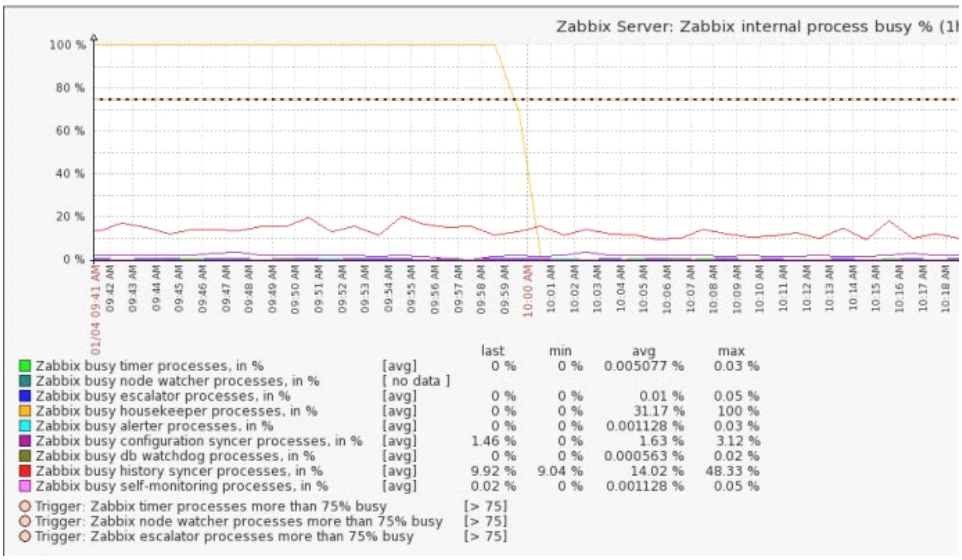
zabbix[process,type,mode,state]

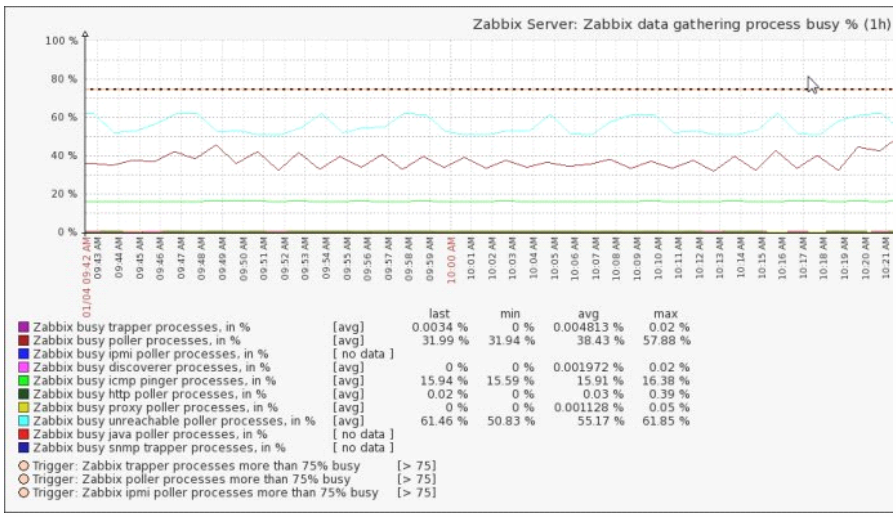
其中可用的参数为:

type: trapper,discoverer,escalator,alerter,etc

mode: avg,count,min,max

state: busy,idel





5.Zabbix 调优大的原则性建议

5 确保 zabbix 内部组件性能处于被监控状态(调优的基础！)
6 使用硬件性能足够好的服务器
7 不同角色分开，使用各自独立的服务器
8 使用分布式部署
9 调整 MySQL 性能
10 调整 Zabbix 自身配置

6.Zabbix 数据库调优

a.使用专用数据服务器，配置应该较高

给一个参考配置，可以处理 NVPS 为3000

Dell PowerEdge R610

CPU: Intel Xeon L5520 2.27GHz (16 cores)

Memory: 24GB RAM

Disks: 6x SAS 10k 配置 RAID10

b.每个 table 一个文件,修改 my.cnf

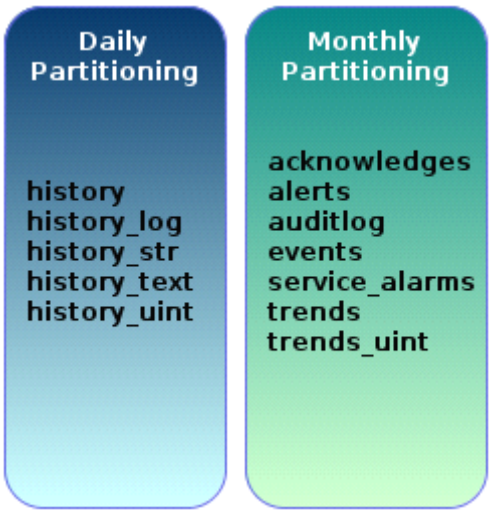
innodb_file_per_table=1

c.使用 percona 替代 MySQL

d.使用分区表，关闭 Houerkeeper

关闭 Houserkeeper,zabbix_server.conf

DisableHousekeeper=1



step 1.准备相关表

```
ALTER TABLE `acknowledges` DROP PRIMARY KEY, ADD KEY `acknowledgedid`
(`acknowledgeid`);
ALTER TABLE `alerts` DROP PRIMARY KEY, ADD KEY `alertid` (`alertid`);
ALTER TABLE `auditlog` DROP PRIMARY KEY, ADD KEY `auditid` (`auditid`);
ALTER TABLE `events` DROP PRIMARY KEY, ADD KEY `eventid` (`eventid`);
ALTER TABLE `service_alarms` DROP PRIMARY KEY, ADD KEY `servicealarmid`
(`servicealarmid`);
ALTER TABLE `history_log` DROP PRIMARY KEY, ADD PRIMARY KEY (`itemid`,`id`,`clock`);
ALTER TABLE `history_log` DROP KEY `history_log_2`;
ALTER TABLE `history_text` DROP PRIMARY KEY, ADD PRIMARY KEY
(`itemid`,`id`,`clock`);
ALTER TABLE `history_text` DROP KEY `history_text_2`;
```

step2.设置每月的分区

以下步骤请在第一步的所有表中重复，下例是为 events 表创建2011-5到2011-12之间的月度分区。

```
ALTER TABLE `events` PARTITION BY RANGE( clock ) (
PARTITION p201105 VALUES LESS THAN (UNIX_TIMESTAMP("2011-06-01 00:00:00")),
PARTITION p201106 VALUES LESS THAN (UNIX_TIMESTAMP("2011-07-01 00:00:00")),
PARTITION p201107 VALUES LESS THAN (UNIX_TIMESTAMP("2011-08-01 00:00:00")),
PARTITION p201108 VALUES LESS THAN (UNIX_TIMESTAMP("2011-09-01 00:00:00")),
PARTITION p201109 VALUES LESS THAN (UNIX_TIMESTAMP("2011-10-01 00:00:00")),
PARTITION p201110 VALUES LESS THAN (UNIX_TIMESTAMP("2011-11-01 00:00:00")),
PARTITION p201111 VALUES LESS THAN (UNIX_TIMESTAMP("2011-12-01 00:00:00")),
PARTITION p201112 VALUES LESS THAN (UNIX_TIMESTAMP("2012-01-01 00:00:00"))
);
```

step3.设置每日的分区

以下步骤请在第一步的所有表中重复，下例是为 history_uint 表创建5.15到5.22之间的每日分

区。

```
ALTER TABLE `history_uint` PARTITION BY RANGE( clock ) (
PARTITION p20110515 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-16 00:00:00")),
PARTITION p20110516 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-17 00:00:00")),
PARTITION p20110517 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-18 00:00:00")),
PARTITION p20110518 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-19 00:00:00")),
PARTITION p20110519 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-20 00:00:00")),
PARTITION p20110520 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-21 00:00:00")),
PARTITION p20110521 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-22 00:00:00")),
PARTITION p20110522 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-23 00:00:00"))
);
```

手动维护分区：
增加新分区

```
ALTER TABLE `history_uint` ADD PARTITION (
PARTITION p20110523 VALUES LESS THAN (UNIX_TIMESTAMP("2011-05-24 00:00:00"))
);
```

删除分区(使用 Housekeepeing)

```
ALTER TABLE `history_uint` DROP PARTITION p20110515;
```

step4.自动每日分区
确认已经在 step3的时候为 history 表正确创建了分区。
以下脚本自动 drop 和创建每日分区,默认只保留最近3天,如果你需要更多天的，请修改 @mindays 这个变量。

不要忘记将这条命令加入到你的 cron 中！

```
mysql -B -h localhost -u zabbix -pPASSWORD zabbix -e "CALL
create_zabbix_partitions();"
```

自动创建分区的脚本:
https://github.com/xsbr/zabbixzone/blob/master/zabbix-mysql-autopartitioning.sql

```
DELIMITER //
DROP PROCEDURE IF EXISTS `zabbix`.`create_zabbix_partitions` //
CREATE PROCEDURE `zabbix`.`create_zabbix_partitions` ()
BEGIN
CALL zabbix.create_next_partitions("zabbix","history");
CALL zabbix.create_next_partitions("zabbix","history_log");
CALL zabbix.create_next_partitions("zabbix","history_str");
CALL zabbix.create_next_partitions("zabbix","history_text");
CALL zabbix.create_next_partitions("zabbix","history_uint");
```

```
CALL zabbix.drop_old_partitions("zabbix","history");
CALL zabbix.drop_old_partitions("zabbix","history_log");
CALL zabbix.drop_old_partitions("zabbix","history_str");
CALL zabbix.drop_old_partitions("zabbix","history_text");
CALL zabbix.drop_old_partitions("zabbix","history_uint");
END //
DROP PROCEDURE IF EXISTS `zabbix`.`create_next_partitions` //
CREATE PROCEDURE `zabbix`.`create_next_partitions` (SCHEMANAME varchar(64),
TABLENAME varchar(64))
BEGIN
DECLARE NEXTCLOCK timestamp;
DECLARE PARTITIONNAME varchar(16);
DECLARE CLOCK int;
SET @totaldays = 7;
SET @i = 1;
createloop: LOOP
SET NEXTCLOCK = DATE_ADD(NOW(),INTERVAL @i DAY);
SET PARTITIONNAME = DATE_FORMAT( NEXTCLOCK, 'p%Y%m%d' );
SET CLOCK = UNIX_TIMESTAMP(DATE_FORMAT(DATE_ADD( NEXTCLOCK ,INTERVAL 1
DAY),'%Y-%m-%d 00:00:00'));
CALL zabbix.create_partition( SCHEMANAME, TABLENAME, PARTITIONNAME, CLOCK );
SET @i=@i+1;
IF @i > @totaldays THEN
LEAVE createloop;
END IF;
END LOOP;
END //
DROP PROCEDURE IF EXISTS `zabbix`.`drop_old_partitions` //
CREATE PROCEDURE `zabbix`.`drop_old_partitions` (SCHEMANAME varchar(64),
TABLENAME varchar(64))
BEGIN
DECLARE OLDCLOCK timestamp;
DECLARE PARTITIONNAME varchar(16);
DECLARE CLOCK int;
SET @mindays = 3;
SET @maxdays = @mindays+4;
SET @i = @maxdays;
droploop: LOOP
SET OLDCLOCK = DATE_SUB(NOW(),INTERVAL @i DAY);
SET PARTITIONNAME = DATE_FORMAT( OLDCLOCK, 'p%Y%m%d' );
CALL zabbix.drop_partition( SCHEMANAME, TABLENAME, PARTITIONNAME );
SET @i=@i-1;
IF @i <= @mindays THEN
LEAVE droploop;
```

```
END IF;
END LOOP;
END //
DROP PROCEDURE IF EXISTS `zabbix`.`create_partition` //
CREATE PROCEDURE `zabbix`.`create_partition` (SCHEMANAME varchar(64), TABLENAME
varchar(64), PARTITIONNAME varchar(64), CLOCK int)
BEGIN
DECLARE RETROWS int;
SELECT COUNT(1) INTO RETROWS
FROM `information_schema`.`partitions`
WHERE `table_schema` = SCHEMANAME AND `table_name` = TABLENAME AND
`partition_name` = PARTITIONNAME;

IF RETROWS = 0 THEN
SELECT CONCAT( "create_partition(", SCHEMANAME, ",", TABLENAME, ",", PARTITIONNAME,
",", CLOCK, ")" ) AS msg;
SET @sql = CONCAT( 'ALTER TABLE `', SCHEMANAME, '`.`', TABLENAME, '`',
' ADD PARTITION (PARTITION ', PARTITIONNAME, ' VALUES LESS THAN (', CLOCK, '));' );
PREPARE STMT FROM @sql;
EXECUTE STMT;
DEALLOCATE PREPARE STMT;
END IF;
END //
DROP PROCEDURE IF EXISTS `zabbix`.`drop_partition` //
CREATE PROCEDURE `zabbix`.`drop_partition` (SCHEMANAME varchar(64), TABLENAME
varchar(64), PARTITIONNAME varchar(64))
BEGIN
DECLARE RETROWS int;
SELECT COUNT(1) INTO RETROWS
FROM `information_schema`.`partitions`
WHERE `table_schema` = SCHEMANAME AND `table_name` = TABLENAME AND
`partition_name` = PARTITIONNAME;

IF RETROWS = 1 THEN
SELECT CONCAT( "drop_partition(", SCHEMANAME, ",", TABLENAME, ",", PARTITIONNAME,
")" ) AS msg;
SET @sql = CONCAT( 'ALTER TABLE `', SCHEMANAME, '`.`', TABLENAME, '`',
' DROP PARTITION ', PARTITIONNAME, ';' );
PREPARE STMT FROM @sql;
EXECUTE STMT;
DEALLOCATE PREPARE STMT;
END IF;
END //
DELIMITER ;
```

e.使用 tmpfs 存储临时文件

mkdir /tmp/mysqltmp
修改/etc/fstab:
tmfs /tmp/mysqltmp tmpfs
rw,uid=mysql,gid=mysql,size=1G,nr_inodes=10k,mode=0700 0 0
修改 my.cnf
tmpdir=/tmp/mysqltmp

f.设置正确的 buffer pool

设置 Innodb 可用多少内存，建议设置成物理内存的70%~80%
修改 my.cnf
innodb_buffer_pool_size=14G

设置 innodb 使用 O_DIRECT，这样 buffer_pool 中的数据就不会与系统缓存中的重复。
innodb_flush_method=O_DIRECT

以下给一个示例 my.cnf，物理内存大小为24G:

```
[mysqld]
# paths
datadir                         = /var/lib/mysql/data
tmpdir                          = /tmp/mysqltmp

# network
connect_timeout                 = 60
wait_timeout                    = 28800
max_connections                 = 2048
max_allowed_packet              = 64M
max_connect_errors              = 1000

# limits
tmp_table_size                  = 512M
max_heap_table_size             = 256M
table_cache                     = 512

# logs
log_error                       = /var/log/mysql/mysql-error.log
slow_query_log_file             = /var/log/mysql/mysql-slow.log
slow_query_log                  = 1
long_query_time                 = 20

# innodb
innodb_data_home_dir            = /var/lib/mysql/data
innodb_data_file_path           = ibdata1:128M;ibdata2:128M:autoextend:max:4096M
innodb_file_per_table           = 1
innodb_status_file              = 1
innodb_additional_mem_pool_size = 128M
innodb_buffer_pool_size         = 14G
innodb_flush_method             = O_DIRECT
innodb_io_capacity              = 2000
innodb_flush_log_at_trx_commit  = 2
innodb_support_xa               = 0
innodb_log_file_size            = 512M
innodb_log_buffer_size          = 128M

# experimental
innodb_stats_update_need_lock   = 0

# other stuff
event_scheduler                 = 1
query_cache_type                = 0
```

g.设置合适的 log 大小

zabbix 数据库属于写入较多的数据库,因此设置大一点可以避免 MySQL 持续将 log 文件 flush 到表中。

不过有一个副作用,就是启动和关闭数据库会变慢一点。

修改 my.cnf

innodb_log_file_size=64M

h.打开慢查询日志

修改 my.cnf

log_slow_queries=/var/log/mysql.slow.log

i.设置 thread_cache_size

这个值似乎会影响 show global status 输出中 Threads_created per Connection 的 hit rate
当设置成4的时候,有3228483 Connections 和5840 Threads_created,hit rate 达到了 99.2%
Threads_created 这个数值应该越小越好。

j.其他 MySQL 文档建议的参数调整

query_cache_limit=1M
query_cache_size=128M
tmp_table_size=256M
max_heap_table_size=256M
table_cache=256
max_connections = 300
innodb_flush_log_at_trx_commit=2
join_buffer_size=256k
read_buffer_size=256k
read_rnd_buffer_size=256k

7.调整 zabbix 工作进程数量,zabbix_server.conf

StartPollers=90
StartPingers=10
StartPollersUnreacheable=80
StartIPMIPollers=10
StartTrappers=20
StartDBSyncers=8
LogSlowQueries=1000

参考文档:
http://www.slideshare.net/xsbr/alexei-vladishev-zabbixperformancetuning
http://zabbixzone.com/zabbix/mysql-performance-tips-for-zabbix/
http://zabbixzone.com/zabbix/partitioning-tables/
http://linux-knowledgebase.com/en/Tip_of_the_day/March/Performance_Tuning_for_Zabbix
http://sysadminnotebook.blogspot.jp/2011/08/performance-tuning-mysql-for-zabbix.html

6. 批量更新参考文档

easy Update on Custom Scripts
http://zabbixzone.com/zabbix/easy-update-on-custom-scripts/

Every time that you need to add or change an UserParameter on zabbix_agentd.conf you need to restart the agent. It's easy if you have less than 10 servers, but could be a trouble if you have more than 30 servers.

Automation Tools like CFEngine and Puppet do this work well done, but unfortunately they aren't used in many companies.

But it's possible using a **Dynamic UserParameter**:

1) add two lines on **zabbix_agentd.conf** and **restart the agent**:

```
UserParameter=custom.getvalue[*],/etc/zabbix/zabbix_agentd/custom_getvalue $1 $2 $3 $4 $5
UserParameter=custom.updatescript,/etc/zabbix/zabbix_agentd/custom_updatescript
```

2) create the script file **/etc/zabbix/zabbix_agentd/custom_getvalue**

```
#!/bin/bash
ACTION=$1
PARAM1=$2
PARAM2=$3
PARAM3=$4
PARAM4=$5
case ${ACTION} in
# key: custom.getvalue[samplescript1,arg]
samplescript1)
  /bin/echo This is a test - ${PARAM1}
  ;;
# key: custom.getvalue[samplescript2,arg1,arg2]
samplescript2)
  /bin/echo This is another test - ${PARAM1} ${PARAM2}
  ;;
*)
  /bin/echo ZBX_NOTSUPPORTED
  ;;
esac
```

Don't forget to set execute permission:

```
1  chmod 0755 /etc/zabbix/zabbix_agentd/custom_getvalue
```

Now when you need create a new script, you must update this script and restart is unnecessary.

Remote Updates

Finally, let's create a script to update **custom_getvalue** script remotely. Before

make sure to publish your**custom_getvalue** script on a WebServer.
Create the file **/etc/zabbix/zabbix_agentd/custom_updatescript**:

```
1 #!/bin/bash
2 /usr/bin/wget "http://yourdomain.com/zabbix/custom_getvalue" -O
  /etc/zabbix/zabbix_agentd/custom_getvalue -o /dev/null
```

Don't forget to set execute permission:

```
1 chmod 0755 /etc/zabbix/zabbix_agentd/custom_updatescript
```

Remote updates can be done from **Zabbix Server/Proxy** using **zabbix_get** utility:

```
1 zabbix_get -s hostserver.yourdomain.com -k custom.updatescript
```

It's a contribution from **laneovcc**:
in my way i config zabbix-agent.conf to include a UserParameter.conf then use system.run[wget http://server/UserParameter.conf -O /path/to/UserParameter.conf] to update the UserParameter.conf and system.run[services zabbix-agentd restart] to restart the agent

7. 将 zabbix 打包成 rpm 包

Rpmbuild 的 spec 文件的写作

官方虽然提供了 rpm，但相关依赖太麻烦，所以改造了一下，仅供参考

```
#
%define zabbix_group zabbix
%define zabbix_user zabbix

Name:                zabbix
Version:             2.0.6
Release:             2%{?dist}.zbx
Summary:             zabbix monitor
Vendor:              itnihao@qq.com

Group:               System Environment/Daemons
License:             GPL
URL:                 http://www.zabbix.com
Source0:
http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/2.0.6/zabbix-2.0.6.
tar.gz
Source1:             zabbix_custom.tar.gz
Source2:             zabbix-web.conf
#BuildRoot:          %{_tmppath}/%{name}-%{version}
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-root
```

```
BuildRequires:    gcc
BuildRequires:    make

Requires(pre):gcc
Requires(post):chkconfig
Provides:Monitor

%description
  Zabbix is the ultimate open source availability and performance monitoring solution. Zabbix offers advanced monitoring, alerting, and visualization features today which are missing in other monitoring systems, even some of the best commercial ones

%package server
Summary:server version of zabbix
Group: System Environment/Daemons
Requires(post):         /sbin/chkconfig
Requires(preun):        /sbin/chkconfig
Requires(preun):        /sbin/service
Requires(postun):       /sbin/service

%description server
Zabbix server common files

%package agentd
Summary:                  Zabbix Agent
Group:                    Applications/Internet
Requires(pre):            shadow-utils
Requires(post):           /sbin/chkconfig
Requires(preun):          /sbin/chkconfig
Requires(preun):          /sbin/service
Requires(postun):         /sbin/service

%description agentd
The Zabbix client agent, to be installed on monitored systems.


%package proxy
Summary:                  Zabbix Proxy
```

```
Group:                  Applications/Internet
Requires(pre):          shadow-utils
Requires(post):         /sbin/chkconfig
Requires(preun):        /sbin/chkconfig
Requires(preun):        /sbin/service
Requires(postun):       /sbin/service
Requires:               fping

%description proxy
The Zabbix proxy



%package web
Summary:                Zabbix Web
Group:                  Applications/Internet
BuildArch:              noarch
Requires(pre):          shadow-utils
Requires(post):         /sbin/chkconfig
Requires(preun):        /sbin/chkconfig
Requires(preun):        /sbin/service
Requires(postun):       /sbin/service
Requires:               dejavu-sans-fonts

%description web
The Zabbix web



%prep
%setup -q

%build
common_flags="
        --enable-dependency-tracking
        --enable-proxy
        --enable-agent
        --enable-ipv6
        --with-net-snmp
        --with-libcurl
```

```
        --disable-java
        --sysconfdir=%{_sysconfdir}/zabbix
        --datadir=%{_sharedstatedir}
        "
    %configure $common_flags --enable-server --with-mysql --with-cc-opt="%{optflags} $(pcre-config --cflags)"
    make %{?_smp_mflags}


    #./configure --prefix=/usr/local --sysconfdir=/etc/zabbix --libdir=/usr/lib --mandir=/usr/share/man --enable-proxy --enable-agent --with-sqlite3
    #make %{?_smp_mflags}


    %install
    %{__rm} -rf $RPM_BUILD_ROOT
    [ "%{buildroot}" != "/" ] && %{__rm} -rf %{buildroot}
    #%{__make} DESTDIR=%{buildroot} install

    %{__install} -d %{buildroot}%{_sbindir}
    %{__install} -d %{buildroot}%{_sysconfdir}/rc.d/init.d
    %{__install} -d %{buildroot}%{_datadir}/%{name}
    %{__install} -d %{buildroot}%{_sysconfdir}/%{name}/scripts
    %{__install} -d %{buildroot}%{_sysconfdir}/%{name}/zabbix_agentd.conf.d
    %{__install} -d %{buildroot}%{_mandir}/man1/
    %{__install} -d %{buildroot}%{_mandir}/man8/
    %{__install} -d %{buildroot}%{_localstatedir}/log/%{name}
    %{__install} -d %{buildroot}%{_localstatedir}/run/%{name}
    %{__install} -d %{buildroot}%{_sysconfdir}/%{name}/externalscripts
    %{__install} -d %{buildroot}%{_sysconfdir}/%{name}/alertscripts
    %{__install} -d %{buildroot}%{_datadir}/%{name}

    %{__make} DESTDIR=$RPM_BUILD_ROOT install


    #%{__mkdir} -p $RPM_BUILD_ROOT%{_initrddir}
    %{__install} -m 755 misc/init.d/fedora/core/zabbix_agentd $RPM_BUILD_ROOT%{_initrddir}/zabbix_agentd
```

```
    %{__install} -m 755 misc/init.d/fedora/core/zabbix_server
$RPM_BUILD_ROOT%{_initrddir}/zabbix_server
    %{__install} -m 755 misc/init.d/fedora/core/zabbix_server
$RPM_BUILD_ROOT%{_initrddir}/zabbix_proxy
    %{__mv}  frontends/php $RPM_BUILD_ROOT/%{_datadir}/zabbix
    %{__sed}  -i  "s@BINARY_NAME=zabbix_server@BINARY_NAME=zabbix_proxy@g"
$RPM_BUILD_ROOT%{_initrddir}/zabbix_proxy
    %{__sed} -i "s@BASEDIR=/usr/local@BASEDIR=/usr@g"
$RPM_BUILD_ROOT%{_initrddir}/zabbix_server
    %{__sed} -i "s@PIDFILE=/tmp@PIDFILE=/var/run/zabbix@g"
$RPM_BUILD_ROOT%{_initrddir}/zabbix_server
    %{__sed} -i "s@BASEDIR=/usr/local@BASEDIR=/usr@g"
$RPM_BUILD_ROOT%{_initrddir}/zabbix_agentd
    %{__sed} -i "s@PIDFILE=/tmp@PIDFILE=/var/run/zabbix@g"
$RPM_BUILD_ROOT%{_initrddir}/zabbix_agentd
    %{__sed} -i "s@BASEDIR=/usr/local@BASEDIR=/usr@g"
$RPM_BUILD_ROOT%{_initrddir}/zabbix_proxy
    %{__sed} -i "s@PIDFILE=/tmp@PIDFILE=/var/run/zabbix@g"
$RPM_BUILD_ROOT%{_initrddir}/zabbix_proxy


    install -m 0755 -p src/zabbix_server/zabbix_server $RPM_BUILD_ROOT%{_sbindir}/
    install -m 0755 -p src/zabbix_proxy/zabbix_proxy $RPM_BUILD_ROOT%{_sbindir}/
    install -m 0755 -p src/zabbix_get/zabbix_get $RPM_BUILD_ROOT%{_sbindir}/
    install -m 0755 -p src/zabbix_sender/zabbix_sender $RPM_BUILD_ROOT%{_sbindir}/
    install -m 0755 -p src/zabbix_agent/zabbix_agent $RPM_BUILD_ROOT%{_sbindir}/
    install -m 0755 -p src/zabbix_agent/zabbix_agentd $RPM_BUILD_ROOT%{_sbindir}/
    install -m 0644 -p conf/zabbix_server.conf
$RPM_BUILD_ROOT%{_sysconfdir}/%{name}/
    install -m 0644 -p conf/zabbix_agent.conf $RPM_BUILD_ROOT%{_sysconfdir}/%{name}/
    install -m 0644 -p conf/zabbix_agentd.conf
$RPM_BUILD_ROOT%{_sysconfdir}/%{name}/
    install -m 0644 -p conf/zabbix_proxy.conf $RPM_BUILD_ROOT%{_sysconfdir}/%{name}/
    install -m 0644 -p man/zabbix_agentd.man
$RPM_BUILD_ROOT%{_mandir}/man8/zabbix_agentd.8
    install -m 0644 -p man/zabbix_server.man
$RPM_BUILD_ROOT%{_mandir}//man8/zabbix_server.8
    install -m 0644 -p man/zabbix_proxy.man
```

```
$RPM_BUILD_ROOT%{_mandir}/man8/zabbix_proxy.8
    install -m 0644 -p man/zabbix_get.man $RPM_BUILD_ROOT%{_mandir}/man1/zabbix_get.1
    install -m 0644 -p man/zabbix_sender.man $RPM_BUILD_ROOT%{_mandir}/man1/zabbix_sender.1

    %{__tar} xf %{SOURCE1} -C $RPM_BUILD_ROOT%{_sysconfdir}/%{name}
    install -m 0644 -p %{SOURCE2}  $RPM_BUILD_ROOT/%{_datadir}/%{name}
    install -d $RPM_BUILD_ROOT/%{_datadir}/%{name}/database/mysql/
    install -m 0644 -p database/mysql/* $RPM_BUILD_ROOT/%{_datadir}/%{name}/database/mysql/
    sed -i \
        -e 's|# PidFile=.*|PidFile=%{_localstatedir}/run/%{name}/zabbix_agentd.pid|g' \
        -e 's|^LogFile=.*|LogFile=%{_localstatedir}/log/%{name}/zabbix_agentd.log|g' \
        -e '/# UnsafeUserParameters=0/aUnsafeUserParameters=1\n' \
        -e  '/#  Include.*zabbix_agentd.conf.d\/aInclude=\/etc\/zabbix\/zabbix_agentd.conf.d\/\n' \
        -e '/StartAgents=3/aStartAgents=5\n' \
        -e 's|# LogFileSize=.*|LogFileSize=0|g' \
        -e 's|Server=127.0.0.1$|Server=127.0.0.1,10.10.10.1,60.191.140.200|g' \
        -e 's|ServerActive=127.0.0.1$|ServerActive=127.0.0.1:10051,10.10.10.1:10051,60.191.140.200:10051|g' \
        -e 's|# EnableRemoteCommands=0|EnableRemoteCommands=1|g' \
        -e 's|# LogRemoteCommands=0|LogRemoteCommands=1|g' \
        -e 's|LogFileSize=0|LogFileSize=10|g' \
        -e 's|/usr/local|/usr|g' \
         $RPM_BUILD_ROOT%{_sysconfdir}/%{name}/zabbix_agentd.conf

    sed -i \
        -e 's|/usr/local|/usr|g' \
        -e '/# UnsafeUserParameters=0/aUnsafeUserParameters=1\n' \
        -e 's@# Include=/usr/etc/zabbix_agentd.conf.d@Include=/etc/zabbix/zabbix_agentd.conf.d@g' \
         $RPM_BUILD_ROOT%{_sysconfdir}/%{name}/zabbix_agent.conf

    sed -i \
        -e 's|# PidFile=.*|PidFile=%{_localstatedir}/run/%{name}/zabbix_server.pid|g' \
```

```
        -e 's|^LogFile=.*|LogFile=%{_localstatedir}/log/%{name}/zabbix_server.log|g' \
        -e 's|# LogFileSize=.*|LogFileSize=0|g' \
        -e 's|^DBUser=root|DBUser=zabbix|g' \
        -e '/# DBPassword=/aDBPassword=zabbix\n' \
        -e 's|# DBSocket=/tmp/mysql.sock|DBSocket=%{_sharedstatedir}/mysql/mysql.sock|g' \
        -e 's|#
ExternalScripts=\${datadir}/zabbix/externalscripts|ExternalScripts=%{_sysconfdir}/%{name}/ext
ernalscripts|' \
        -e 's|/usr/local|/usr|g' \
         $RPM_BUILD_ROOT%{_sysconfdir}/%{name}/zabbix_server.conf

    sed -i \
        -e 's|# PidFile=.*|PidFile=%{_localstatedir}/run/%{name}/zabbix_proxy.pid|g' \
        -e 's|^LogFile=.*|LogFile=%{_localstatedir}/log/%{name}/zabbix_proxy.log|g' \
        -e 's|# LogFileSize=.*|LogFileSize=0|g' \
        -e 's|^DBUser=root|DBUser=zabbix|g' \
        -e '/# DBPassword=/aDBPassword=zabbix\n' \
        -e 's|# DBSocket=/tmp/mysql.sock|DBSocket=%{_sharedstatedir}/mysql/mysql.sock|g' \
        -e 's|#
ExternalScripts=\${datadir}/zabbix/externalscripts|ExternalScripts=%{_sysconfdir}/%{name}/ext
ernalscripts|' \
        -e 's|/usr/local|/usr|g' \
         $RPM_BUILD_ROOT%{_sysconfdir}/%{name}/zabbix_proxy.conf

    %clean
    %{__rm} -rf $RPM_BUILD_ROOT


    %files server
    %defattr(-,root,root,-)
    %doc
    %attr(0755,zabbix,zabbix) %dir %{_localstatedir}/log/%{name}
    %attr(0775,root,zabbix)    %dir %{_localstatedir}/run/%{name}
    %config(noreplace)   %{_sysconfdir}/%{name}/zabbix_server.conf
    %config(noreplace)   %{_sysconfdir}/%{name}/scripts
    %{_sbindir}/zabbix_sender
    %{_sbindir}/zabbix_server
    %{_sbindir}/zabbix_get
```

```
%{_initrddir}/zabbix_server

%config(noreplace) %{_sysconfdir}/%{name}/externalscripts
%config(noreplace) %{_sysconfdir}/%{name}/alertscripts

%{_mandir}/man8/zabbix_server.8*
%{_mandir}/man1/zabbix_get.1*
%{_mandir}/man1/zabbix_sender.1*

%files agentd
%defattr(-,root,root,-)
%doc
%attr(0755,zabbix,zabbix) %dir %{_localstatedir}/log/%{name}
%attr(0775,root,zabbix) %dir %{_localstatedir}/run/%{name}
%attr(0775,root,zabbix) %dir %{_sysconfdir}/%{name}/zabbix_agentd.conf.d
%config(noreplace)    %{_sysconfdir}/%{name}/zabbix_agent.conf
%config(noreplace)    %{_sysconfdir}/%{name}/zabbix_agentd.conf
%config(noreplace)    %{_sysconfdir}/%{name}/scripts
%{_sbindir}/zabbix_sender
%{_sbindir}/zabbix_agent
%{_sbindir}/zabbix_agentd
%{_sbindir}/zabbix_get
%attr(0755,root,zabbix) %{_sysconfdir}/%{name}/scripts/*
%attr(0755,root,zabbix) %{_sysconfdir}/%{name}/zabbix_agentd.conf.d/*

%{_initrddir}/zabbix_agentd
#%{_localstatedir}/run/%{name}
#%{_localstatedir}/log/%{name}
#%config(noreplace) %{_sysconfdir}/%{name}/zabbix_agentd.conf.d

%{_mandir}/man8/zabbix_agentd.8*
%{_mandir}/man1/zabbix_get.1*
%{_mandir}/man1/zabbix_sender.1*



%files proxy
%defattr(-,root,root,-)
```

```
%doc
%attr(0755,zabbix,zabbix) %dir %{_localstatedir}/log/%{name}
%attr(0775,root,zabbix)    %dir %{_localstatedir}/run/%{name}
%config(noreplace)   %{_sysconfdir}/%{name}/zabbix_proxy.conf
%config(noreplace)   %{_sysconfdir}/%{name}/scripts
%{_sbindir}/zabbix_proxy
%{_initrddir}/zabbix_proxy

%{_mandir}/man8/zabbix_proxy.8*
#%{_localstatedir}/run/%{name}
#%{_localstatedir}/log/%{name}
%config(noreplace) %{_sysconfdir}/%{name}/externalscripts
%config(noreplace) %{_sysconfdir}/%{name}/alertscripts

%files web
%defattr(-,root,root,-)
%config(noreplace) %{_datadir}/%{name}/*

%post server
if [ $1 -eq 1 ]; then
/sbin/chkconfig zabbix_server on
fi

%post agentd
if [ $1 -eq 1 ]; then
sed -i "s@Hostname=Zabbix server@Hostname=$HOSTNAME@g" /etc/zabbix/zabbix_agentd.conf
getent group zabbix >/dev/null || groupadd -r zabbix
getent passwd zabbix >/dev/null || useradd -r -g zabbix -d %{_sharedstatedir}/zabbix -s /sbin/nologin -c "zabbix user" zabbix
/sbin/chkconfig zabbix_agentd on
/sbin/service zabbix_agentd start
chown root:zabbix /bin/netstat
chmod 4755 /bin/netstat
fi

%post proxy
if [ $1 -eq 1 ]; then
```

```
/sbin/chkconfig zabbix_proxy on
fi

%post web
mv %{_datadir}/%{name}/php/* %{_datadir}/%{name}
rm -rf %{_datadir}/%{name}/php

[ -d "/etc/http/conf.d" ] && cp %{_datadir}/%{name}/conf/zabbix-web.conf /etc/http/conf.d &&chown -R apache.apache %{_datadir}/%{name} && cat <<EOF
-----------------------------------------------------
you installed Apache Server,the zabbix-web.conf configuration file in /etc/http/conf.d
-----------------------------------------------------
EOF
[ -d "/etc/http/conf.d" ] || cat <<EOF
-------------------------------------------------------------------------------------
you should configure Web Server,the web file in %{_datadir}/%{name}
-------------------------------------------------------------------------------------
EOF
#then
#mv %{_datadir}/%{name} /var/www/html/
 cat <<EOF
-------------------------------------------------------------------------------------
Author:itnihao Mail:itnihao@qq.com Blog: http://itnihao.blog.51cto.com
you can configure web server on directory /usr/share/zabbix
if your web server is Apache,you can use the /etc/httpd/conf.d/zabbix-web.conf file to
start your web server,and others you must configure your web server
to running the server and web, you will install packages;
yum install httpd php mysql mysql-server php-mysql httpd-manual mod_ssl mod_perl mod_auth_mysql php-gd php-xml php-mbstring php-ldap php-pear php-xmlrpc php-bcmath mysql-connector-odbc mysql-devel libdbi-dbd-mysql net-snmp-devel curl-devel
#setting /etc/php.ini for zabbix
sed -i "s/;date.timezone =/date.timezone = Asia\/Shanghai/g" /etc/php.ini
sed -i "s#max_execution_time = 30#max_execution_time = 300#g" /etc/php.ini
sed -i "s#post_max_size = 8M#post_max_size = 32M#g" /etc/php.ini
sed -i "s#max_input_time = 60#max_input_time = 300#g" /etc/php.ini
```

```
sed -i "s#memory_limit = 128M#memory_limit = 128M#g" /etc/php.ini
sed -i "/;mbstring.func_overload = 0/ambstring.func_overload = 2\n" /etc/php.ini
#config apache
sed -i "s/DirectoryIndex index.html index.html.var/DirectoryIndex index.php index.html index.html.var/g" /etc/httpd/conf/httpd.conf
sed -i "s/ServerTokens OS/ServerTokens Prod/g" /etc/httpd/conf/httpd.conf
#create mysql database to zabbix
service mysqld start
chkconfig mysqld on
mysqladmin -u root password 'mysqlpass'
mysql -uroot -pmysqlpass -e "create database zabbix character set utf8"
mysql -uroot -pmysqlpass -e "grant all privileges on zabbix.* to zabbix@localhost identified by 'zabbix'"
mysql -uroot -pmysqlpass -e "flush privileges"

#source zabbix database
cd /usr/share/zabbix
mysql -uzabbix -pzabbix zabbix < ./database/mysql/schema.sql
mysql -uzabbix -pzabbix zabbix < ./database/mysql/images.sql
mysql -uzabbix -pzabbix zabbix < ./database/mysql/data.sql
-----------------------------------------------------------------------------------------
EOF
#fi


%pre server
#add zabbix to services
grep zabbix /etc/services
[ "$?" != 0 ] && cat >> /etc/services <<EOF
zabbix-agent      10050/tcp                    #Zabbix Agent
zabbix-agent      10050/udp                    #Zabbix Agent
zabbix-trapper    10051/tcp                    #Zabbix Trapper
zabbix-trapper    10051/udp                    #Zabbix Trapper
EOF
# Add the "zabbix" user
getent group zabbix >/dev/null || groupadd -r zabbix
getent passwd zabbix >/dev/null || useradd -r -g zabbix -d %{_sharedstatedir}/zabbix -s
```

```
/sbin/nologin    -c "zabbix user" zabbix


    %pre agentd
    #add zabbix to services
    grep zabbix /etc/services
    [ "$?" != 0 ] && cat >> /etc/services <<EOF
    zabbix-agent        10050/tcp                    #Zabbix Agent
    zabbix-agent        10050/udp                    #Zabbix Agent
    zabbix-trapper    10051/tcp                    #Zabbix Trapper
    zabbix-trapper    10051/udp                    #Zabbix Trapper
    EOF
    # Add the "zabbix" user
    getent group zabbix >/dev/null || groupadd -r    zabbix
    getent passwd zabbix >/dev/null || useradd -r -g zabbix -d %{_sharedstatedir}/zabbix -s
/sbin/nologin    -c "zabbix user" zabbix


    %preun server
    if [ "$1" = 0 ]
    then
        /sbin/service zabbix_server stop >/dev/null 2>&1
        /sbin/chkconfig --del zabbix_server
    fi

    %preun proxy
    if [ "$1" = 0 ]
    then
        /sbin/service zabbix_proxy stop >/dev/null 2>&1
        /sbin/chkconfig --del zabbix_proxy
    fi

    %preun agentd
    if [ "$1" = 0 ]
    then
        /sbin/service zabbix_agentd stop >/dev/null 2>&1
        /sbin/chkconfig --del zabbix_agentd
    fi
```

```
%changelog
* Mon Feb 18 2013    changed file for agentd<itnihao@qq.com>
- 2.0.5
* Fri Jan 25 2013    First version is build ok <itnihao@qq.com>
- 2.0.4
```