ZABBIX 从入门到精通

由浅入深 中文手册

凉白开

2016/2/25

感谢浏览，欢迎拍砖（虽有错别字，但核心内容犹在，谅解谅解!）

目录

前言

感谢

感谢一直陪伴在我身边的老婆：兰若，给予我支持和鼓励，才能坚持把 zabbix 系列教程写到现在，并且完成这份《zabbix 教程从入门到精通》PDF 电子书。

感谢靠谱云（www.kaopuyun.com）免费为本站提供云服务器。

感谢漠北、tonyty163 以及其他网友为本站提供内容

关于运维生存时间

网站名称并没有特别的意思，运维时常接触到 TTL 以及职业本身便是 SA（system administraotr），连起来便得到了 ttlsa.com。域名有了，名称也就这么来了 - 运维生存时间。

运维生存时间历程

日期	历程
2010 年	8 月 4 日：购买域名
2011 年	2 月 13 日：发布本站第一篇文章，团队陆续发布上百篇技术文章
2012 年	本站放空的一年，偶尔更新文章
2013 年	6 月 22 日：重启 TTLSA 团队，用心经营本站。 将网站从国内空间迁移至 linode，并运行了将近一年时间，之后迁移至阿里云 2 个月
2014 年	7 月 7 日：迎来了本站的首次赞助，kaopuyun.com 免费提供国内云服务器。 12 月 14:Alexa 排名首次进入 10 万以内，排名 97901

本站用心更新，用户量也不断创新高，感谢作者们对 TTLSA 的无私奉献.

TTLSA 原创系列教程

当然，zabbix 只是我们系列之中的一份，我们还有更多系列教程，截至 2015 年 1 月 20 日星期二，我们一共有六个原创系列教程，如下：

- 《Zabbix 教程从入门到精通》

  作者：凉白开

  电子书：http://ebook.ttlsa.com/monitor/

  文章列表：http://www.ttlsa.com/zabbix/follow-ttlsa-to-study-zabbix/

- 《Nginx 教程从入门到精通》

作者：漠北、凉白开

电子书：http://ebook.ttlsa.com/nginx/

文章地址：http://www.ttlsa.com/nginx/nginx-tutorial-from-entry-to-the-master-ttlsa/

- 《Mongodb 实战教程》

  作者：漠北

  文章地址：http://www.ttlsa.com/mongodb/mongodb-study/

- 《Mongodb 官方监控：MMS》

  作者：漠北

  文章地址：http://www.ttlsa.com/mms/follow-me-to-use-mongodb-mms-services/

- 《MySQL 管理工具利器：MySQL Utilities》

  作者：漠北

  文章地址：http://www.ttlsa.com/mysql/mysql-manager-tools-mysql-utilities-tutorial/

- 《TTLSA 带你学习 Thinkphp》

  作者：tonyty163

  文章地址：http://www.ttlsa.com/php/follow-ttlsa-to-study-thinkphp/

业务合作

一切运维有关业务可与我们联系洽谈

联系我们

E-mail：support@ttlsa.com

QQ：757444407

QQ 群：6690706、39514058(已满)

捐助

如果你喜欢我们的内容，也想支持我们，那么捐助我们吧！有钱，任性，那就打赏吧

版本记录

内容在不断更新中，请大家保持关注 TTLSA，最新的 PDF 会同步放到官网

版本	更新内容	日期	操作人
V1.0	初始化文档	2015 年 01 月 20 日	凉白开
V2.0	增加自动注册、发现、LLD、监控实例以及其他内容	2015 年 11 月 29 日	凉白开
V3.0	内容添加	2016 年 02 月 25 日	凉白开

第一章：简介

Zabbix 介绍

Alexei Vladishev 创建了 Zabbix 项目，当前处于活跃开发状态，Zabbix SIA 提供支持.

- Zabbix 是一个企业级的、开源的、分布式的监控套件
- Zabbix 可以监控网络和服务的监控状况. Zabbix 利用灵活的告警机制，允许用户对事件发送基于 Email 的告警. 这样可以保证快速的对问题作出相应. Zabbix 可以利用存储数据提供杰出的报告及图形化方式. 这一特性将帮助用户完成容量规划.
- Zabbix 支持 polling 和 trapping 两种方式. 所有的 Zabbix 报告都可以通过配置参数在 WEB 前端进行访问. Web 前端将帮助你在任何区域都能够迅速获得你的网络及服务状况. Zabbix 可以通过尽可能的配置来扮演监控你的 IT 基础框架的角色，而不管你是来自于小型组织还是大规模的公司.
- Zabbix 是零成本的. 因为 Zabbix 编写和发布基于 GPL V2 协议. 意味着源代码是免费发布的.
- Zabbix 公司也提供商业化的技术支持.

Zabbix 特性

在知道 zabbix 是什么之后，我们最关心的是 zabbix 有什么特性，了解特性之后，我们才能决定是否会使用 zabbix，以及 zabbix 是否适合我们.Zabbix 是一个高度集成的网络监控套件，通过一个软件包即可提供如下特性

- 数据收集
  a) 可用性及性能检测
  b) 支持 SNMP(trapping 及 polling)、IPMI、JMX 监控
  c) 自定义检测
  d) 自定义间隔收集收据
  e) server/proxy/agents 吸能

- 灵活的阀值定义
  a) 允许灵活地自定义问题阀值，Zabbix 中称为触发器(trigger)，存储在后端数据库中

- 高级告警配置
  a) 可以自定义告警升级(escalation)、接收者及告警方式
  b) 告警信息可以配置并允许使用宏(macro)变量
  c) 通过远程命令实行自动化动作(action)

- 实时绘图
  a) 通过内置的绘图方法实现监控数据实时绘图

- 扩展的图形化显示
  b) 允许自定义创建多监控项视图
  c) 网络拓扑(network maps)
  d) 自定义的面板(screen)和 slide shows，并允许在 dashboard 页面显示
  e) 报告
  f) 高等级(商业)监控资源

- 历史数据存储

a) 数据存储在数据库中

b) 历史数据可配置

c) 内置数据清理机制

- 配置简单

a) 主机通过添加监控设备方式添加

b) 一次配置，终生监控(译者注：除非调整或删除)

c) 监控设备允许使用模板

- 模板使用

a) 模板中可以添加组监控

b) 模板允许继承

- 网络自动发现

a) 自动发现网络设备

b) agent 自动注册

c) 自动发现文件系统、网卡设备、SNMP OID 等

- 快速的 web 接口

a) web 前端采用 php 编写

b) 访问无障碍

c) 你想怎么做就能做么做

d) 审计日志

- Zabbix API

a) Zabbix API 提供程序级别的访问接口，第三方程序可以很快接入

- 权限系统

b) 安全的权限认证

c) 用户可以限制允许维护的列表

- 全特性、agent 易扩展
  a) 在监控目标上部署
  b) 支持 Linux 及 Windows

- 二进制守护进程
  a) C 开发，高性能，低内存消耗
  b) 易移植

- 具备应对复杂环境情况
  a) 通过 Zabbix proxy 可以非常容易的创建远程监控

zabbix 进程构成

了解完 zabbix 特性之后，本该进入 zabbix 安装教程，但是我觉得在安装之前我们很有必要了解一下 zabbix 进程组成结构，默认情况下 zabbix 包含 5 个程序：**zabbix_agentd、zabbix_get、zabbix_proxy、zabbix_sender、zabbix_server**，另外 **zabbix_java_gateway** 是可选，需要另外安装。下面来分别介绍下他们各自的作用。

- zabbix_agentd

  客户端守护进程，收集客户端数据，例如 cpu 负载、内存、硬盘使用情况等

- zabbix_get

  zabbix 工具，单独使用的命令，通常在 server 或者 proxy 端执行，用户获取被监控端数据，通常用于排错。例如在 server 端获取不到客户端的内存数据，我们可以使用 zabbix_get 获取客户端的内容的方式来做故障排查。

- zabbix_sender

  zabbix 工具，用于发送数据给 server 或者 proxy，通常用于耗时比较长的 check，并且与 trapper 配合使用。生存环境中，个别非常耗时间 check 经常导致 zabbix 超时。于是我们在脚本执行完毕之后，使用 sender 主动提交数据。

- zabbix_server

  zabbix 服务端守护进程。zabbix_agentd、zabbix_get、zabbix_sender、zabbix_proxy、zabbix_java_gateway 的数据最终都是提交到 server

备注：当然不是数据都是主动提交给 zabbix_server,大多数情况下都是 server 主动去取数据。

- zabbix_proxy

  zabbix 代理守护进程。功能类似 server，唯一不同的是它只是一个中转站，它需要把收集到的数据提交/被提交到 server 里。一般跨机房、地区的环境需要用到 proxy。

- zabbix_java_gateway

  zabbix2.0 之后引入的一个功能。顾名思义：Java 网关，类似 agentd，但是只用于 Java 方面。需要特别注意的是，它只能主动去获取数据，而不能被动获取数据。它的数据最终会给到 server 或者 proxy。

第二章：安装

zabbix 软硬件需求

在了解完 zabbix 进程构成之后，我们接着聊 zabbix 的硬件配置、软件需求，或者说我安装 zabbix 需要什么软件，服务器需要什么样的配置，监控 100 台服务器需要怎样的一台服务器，或者我有一台 8 核 16G 的服务器，我能监控多少台服务器？来，带着困惑往下看.

硬件需求

无非就是 cpu、内存、硬盘之类的

1) CPU

由你的 zabbix 数据库使用情况来做决定，如果你监控的项目越多，那你的 cpu 要越好。具体多好，下面有个表格

2) 内存与硬盘

最基本的需求：128MB 内存、256MB 硬盘，当然这样的机器这年头应该找不到了吧，尤其要说明硬盘的问题，你的监控项越多、历史记录保留时间的越久数据库将会越大。

我所知道的 100 来台服务器，做基本的 cpu、内存、硬盘、网卡流量等监控，长年累月下来大概 60GB 左右。

3) 其他硬件

如果你觉得有必要的话，你再准备一个 GSM 短信猫吧，不过很少人用，基本上都使用 email 或者飞信报警.

4) 硬件需求表

Name	Platform	CPU/Memory	Database	Monitored hosts
Small	Ubuntu Linux	PII 350MHz 256MB	SQLite	20
Medium	Ubuntu Linux 64 bit	AMD Athlon 3200+ 2GB	MySQL InnoDB	500
Large	Ubuntu Linux 64 bit	Intel Dual Core 6400 4GB	RAID10 MySQL InnoDB or PostgreSQL	>1000
Very large	RedHat Enterprise	Intel Xeon 2xCPU 8GB	Fast RAID10 MySQL InnoDB or PostgreSQL	>10000

如上，P2 的 CPU、256MB 内存已经可以监控 20 个主机。AMD 3200+/2G 内存可以监控 500 个主机（05 年大学的时候，中低端主流 cpu，这都快 10 年了，尤其可见 zabbix 对服务器的硬件配置要求有多低）,现在的服务器一般都比上面最高配还来得高，所以我武断的认为，大家手头的服务器都有能力监控 1w+以上的服务器，我再武断的认为手头上有 1w+服务器的公司能有多少.

操作系统

支持如下系统，系统之多让人刮目相看，但是 window 只能跑客户端

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris
- Windows: 2000, Server 2003, XP, Vista, Server 2008, 7, 8, Server 2012 (只能跑 Zabbix agent)

软件需求

数据库

- **MySQL**：5.0.3 或者以上，推荐使用 InnoDB 引擎（TTLSA 推荐使用 MySQL，开源、免费资料多）
- **Oracle**：10g 或者以上
- **PostgreSQL**：8.1 或者以上 .
- **SQLite**：3.3.5 或者以上
- **IBM DB2**：9.7 或者以上

WEB 应用

Apache：1.3.12 或者以上

PHP：5.3.0 或者以上，zabbix 早期版本支持 5.2，但是 2.2 版本最低版本是 5.3

PHP 扩展：

名称	版本	必须/可选
gd	-	必须
bcmath	-	必须
ctype	-	必须
libXML	2.6.15 或以上	必须
xmlreader	-	必须

Xmlwriter	-	必须
Session	-	必须
sockets	-	必须
mbstring	-	必须
gettext	-	必须
ibm_db2	-	可选
mysqli	-	推荐
oci8	-	可选
pgsq	-	可选
sqlite3	-	可选

服务器

以下内容都为可选项，如果你需要监控特定项，安装特定支持即可。

OpenIPMI：IPMI 硬件监控

libssh2：版本 1.0 以上，SSH 支持

fping：icmp 监控项

libcurl：监控 web 项.

libiksemel：支持 jabber 报警（国内应该没什么人用）

net-snmp：SNMP 监控支持

JAVA 网关

如果你需要通过 Java 网关来监控你的 Java 进程，那么你需要增加如下支持

包名	地址	兼容性
logback-core-0.9.27.jar	http://logback.qos.ch/	0.9.27、1.0.13、1.1.1
logback-classic-0.9.27.jar	http://logback.qos.ch/	0.9.27、1.0.13、1.1.1.
slf4j-api-1.6.1.jar	http://logback.qos.ch/	1.6.1、1.6.6,、 1.7.6.
android-json-4.3_r3.1.jar	https://android.googlesource.com/platform/libcore/+/master/json	2.3.3_r1.1、4.3_r3.1

时间同步

最重要的一点在最后提，请确保你所有的服务器时间都是正确的，为了确保时间 ok，请在 crontab 里面加上定时时间同步。

```
# crontab -l

00 00    * * *       /usr/sbin/ntpdate -u 195.13.1.153
```

Zabbix 数据库硬盘容量计算

本次案例：100 台服务器，每台服务器有 30 个监控项，每个监控项 60 秒刷新一次，需要多大的硬盘呢？

众所周知，zabbix 基本都是通过 web 配置，这些配置数据也是存放到数据库里的，但是它对硬盘容量的要求基本可以忽略不计，zabbix 对硬盘空间的决定性因素有 4 个，如下：

1. 每秒处理的数据量

这边的每秒只是一个平均值，例如我有 3000 个监控项，都是每 60 秒刷新一次，那么平均每秒有 50（3000/60）个数据要处理。也就是说每秒有 50 条数据要存储到 MySQL（或者其他数据库）

2. 历史记录保存时间

一般情况下，zabbix 监控项值都要存储到数据库中，并且一般保留几周到几个月，当然了，要保存多久，看你具体的配置了。假如一个数据你需要保留 30 天，而且每秒有 50 个值要保存，这三天需要存储 129，600，000（30 天*24 小时*3600 秒）*50 个值。

一条记录需要多少容量：容量由当前使用的数据库引擎和存储的数据类型（浮点型，整形，字符型等等）共同决定的。通常，一条记录需要占用 50 个字节（一个大概值），在这个案例中 129,600,000 个记录大约需要（129600000*50 字节）6.5GB 的硬盘空间

3. 趋势数据保存时间

什么是趋势数据？当你查看一周或者一月的图表，图表上看到的 MAX/MIN/AVG/COUNT 都是取自趋势数据，趋势数据一小时获取一次。通常，一条趋势数据大概占用 128 字节，如果需要保存 5 年趋势数据，3000 个监控项需要 2.4GB（3000 个*24 小时*356 天*128 字节）每年，5 年一共 16.8G。

4. 事件记录保存时间

报警、警告、恢复等等事件，一个事件大概占用 130 个字节，通常情况下不会太多的事件，除非运维做的太糟糕，或者运维要求太严格，把阀值调的很低。假如这个运维今年本命年，既没拜佛有没烧香，更别说给服务器贴灵符，于是这一年每秒钟就有一个事件发生，那么这一年事件记录占用的数据空间为：1 年*365 天*24 小时*3600 秒*130

字节大概为 4.1G 空间。

5. 数据库空间计算公式

zabbix 配置：固定大小，一般<10MB

历史数据：天数*(监控项总数/更新频率)*24 小时*3600 秒*50 字节

趋势数据：天数*（监控项总数/3600）*24 小时*3600 秒*128 字节

事件数据：天数*事件个数（大概值）*24 小时*3600 秒*130 字节

6. 总结

看到这里，大家都心里有数据了，数据库硬盘空间=配置文件大小+历史记录+趋势记录+事件记录。虽然这个硬盘会不停的增长，但是总有一天会停止增长，空间一直保持不变，为什么?看完了这篇还问为什么的话，你从头再看一次。

Zabbix 安装

LNMP 环境配置

NGINX 安装：http://www.ttlsa.com/nginx/nginx-install-on-linux/

PHP 安装+NGINX 配置：http://www.ttlsa.com/nginx/nginx-php-5_5/

MYSQL 安装：http://www.ttlsa.com/mysql/install-mysql5_6/

（安装前一定要看 1.1 PHP 安装参数）

1. PHP 配置参数

zabbix 对 PHP 参数、PHP 模块有特殊要求。群里经常看到群里问，zabbix 装不下去了，缺少 php 扩展，到头来还需要再额外加一个扩展，太浪费时间了。所以，请大家一定要仔细看 php 扩展需求。

1.1 PHP 安装参数

php 具体安装方法参考上面的链接，不过如下模块要特别留意加上

扩展	参数
Bcmath	--enable-bcmath
mbstring	--enable-mbstring
sockets	--enable-sockets
gd	--with-gd
libxml	--with-libxml-dir=/usr/local
xmlwriter	同上
xmlreader	同上
ctype	默认支持

session	默认支持
gettext	默认支持

以下是我 PHP 的配置参数

```
./configure    --prefix=/usr/local/php-5.5.7 \
--with-config-file-path=/usr/local/php-5.5.7/etc --with-bz2 --with-curl \
--enable-ftp --enable-sockets --disable-ipv6 --with-gd \
--with-jpeg-dir=/usr/local --with-png-dir=/usr/local \
--with-freetype-dir=/usr/local --enable-gd-native-ttf \
--with-iconv-dir=/usr/local --enable-mbstring --enable-calendar \
--with-gettext --with-libxml-dir=/usr/local --with-zlib \
--with-pdo-mysql=mysqlnd --with-mysqli=mysqlnd --with-mysql=mysqlnd \
--enable-dom --enable-xml --enable-fpm --with-libdir=lib64 --enable-bcmath
```

1.2 PHP 配置参数

打开 php.ini 配置文件，修改以下配置（zabbix 硬性要求）

```
max_execution_time = 300
memory_limit = 128M
post_max_size = 16M
upload_max_filesize = 2M
max_input_time = 300
date.timezone = PRC
```

2 zabbix 服务端安装

2.1 下载安装 zabbix

所有版本下载地址：http://www.zabbix.com/download.php

```
# cd /usr/local/src
#
"http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/2.2.2/zabbix-2.2.2.tar.gz?r=http%3A%2F%2Fww
# tar -xzvf zabbix-2.2.2.tar.gz
# cd zabbix-2.2.2
# ./configure --prefix=/usr/local/zabbix-2.2.2/ --enable-server \
--enable-agent --with-mysql --with-net-snmp --with-libcurl --with-libxml2
# make
# make install
```

zabbix server 一般充当两个角色：server、angent（需要监控自己），所以上面的配置参数也同时加上了--enable-agent。

备注：请安装好 MySQL，snmp，curl 开发库。

2.2 创建用户

为了安全考虑 zabbix 只使用普通用户运行，假如你当前用户叫 ttlsa，那么你运行他，他便使用 ttlsa 身份运行。但是如果你在 root 环境下运行 zabbix，那么 zabbix 将会主动尝试以 zabbix 身份来运行。但是如果你的系统没有名叫 zabbix 的用户，你需要创建一个用户，如下：

```
# groupadd zabbix
# useradd -g zabbix zabbix
```

2.3 初始化数据库

zabbix server 与 proxy 需要数据库，angent 不需要。尤其要注意的是 proxy 只需要导入一个 sql 文件，而 server 一共要导入 3 个 sql 文件。我当时在搭建 proxy 的时候导入了 3 个 sql，导致出现报错。后来才发现 proxy 只需要导入一个表结构即可。

我假想你安装好了 MySQL，用户名为 root，密码为 ttlsapwd

```
# mysql -uroot -pttlsapwd
mysql> create database zabbix default charset utf8;
mysql> quit;
# mysql -uroot -pttlsapwd zabbix < database/mysql/schema.sql
```

备注：创建数据库请别忘记加 default charset utf8，有可能会导致你出现中文乱码问题，具体问题请查看《zabbix 中文乱码解决方法》

如果你仅仅是初始化 proxy 的数据库，那么够了。如果初始化 server，那么接着导入下面两个 sql

```
# mysql -uroot -pttlsapwd zabbix < database/mysql/images.sql
# mysql -uroot -ppttlsapwd zabbix < database/mysql/data.sql
```

其他数据库（db2\sqlite\oracle）数据库初始化方法参考：https://www.zabbix.com/documentation/2.2/manual/appendix/install/db_scripts

2.4 配置 zabbix

配置 zabbix_server 配置文件,zabbix 源码目录下

```
# mkdir /etc/zabbix
# cp config/zabbix_server.conf /etc/zabbix/
# vim /etc/zabbix/zabbix_server.conf
DBName=zabbix
DBUser=root
DBPassword=ttlsapwd
DBPort=3306
```

2.5 启动 zabbix server

```
# /usr/local/zabbix-2.2.2/sbin/zabbix_server
```

默认端口 10051

3. 客户端安装配置

3.1 下载安装客户端

所有版本下载地址：http://www.zabbix.com/download.php，可以直接下载已经编译好的二进制文件，或者也可以源码安装。

```
# cd /usr/local/src
#
"http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/2.2.2/zabbix-2.2.2.tar.gz?r=http%3A%2F%2Fww
# tar -xzvf zabbix-2.2.2.tar.gz
# cd zabbix-2.2.2
# ./configure --prefix=/usr/local/zabbix-2.2.2/ --enable-agent
# make
# make install
```

3.2 zabbix 客户端配置

配置 zabbix_server 配置文件,zabbix 源码目录下

```
# vim /usr/local/zabbix-2.2.2/etc/zabbix_agentd.conf
Server=127.0.0.1
ServerActive=127.0.0.1
```

```
Hostname=Zabbix server
```

其中 Server 和 ServerActive 都指定 zabbixserver 的 IP 地址，不同的是，前者是被动后者是主动。也就是说 Server 这个配置是用来允许 127.0.0.1 这个 ip 来我这取数据。而 serverActive 的 127.0.0.1 的意思是，客户端主动提交数据给他。明白了吗？为什么要分主动和被动？后续再来讨论这个问题！

其他主机安装客户端记得添加 zabbix 用户。

3.3 zabbix 客户端启动

```
# /usr/local/zabbix-2.2.2/sbin/zabbix_agentd
```

默认端口 10050

4. zabbix 管理网站配置

4.1 拷贝前端文件

```
# mkdir /data/logs/nginx
# mkdir /data/site/monitor.ttlsa.com/zabbix
# cp -rp frontends/php/* /data/site/monitor.ttlsa.com/zabbix
```

4.2 配置虚拟主机

请相应修改你的配置文件路径

```
# vim /usr/local/nginx-1.5.8/conf/vhost/monitor.ttlsa.com.conf
server {
        listen          80;
        server_name monitor.ttlsa.com;
        access_log  /data/logs/nginx/monitor.ttlsa.com.access.log  main;

        index index.html index.php index.html;
```

```
        root /data/site/monitor.ttlsa.com;


        location /
        {
                try_files $uri $uri/ /index.php?$args;
        }


        location ~ ^(.+.php)(.*)$ {
                fastcgi_split_path_info ^(.+.php)(.*)$;
                include fastcgi.conf;
                fastcgi_pass    127.0.0.1:9000;
                fastcgi_index index.php;
                fastcgi_param    PATH_INFO                $fastcgi_path_info;
        }
}
```

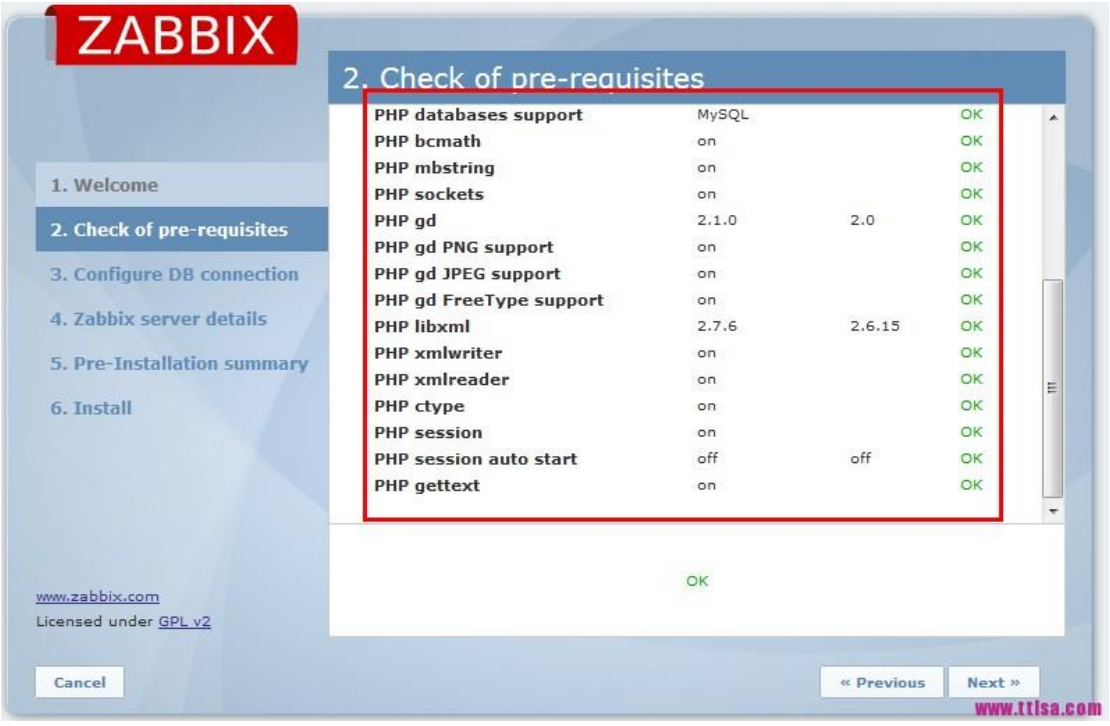
4.3 在线配置 zabbix

浏览器打开 http://monitor.ttlsa.com/zabbix。

如下是 zabbix 2.2 的安装界面，包括欢迎界面一共 6 步.

4.3.1 欢迎界面

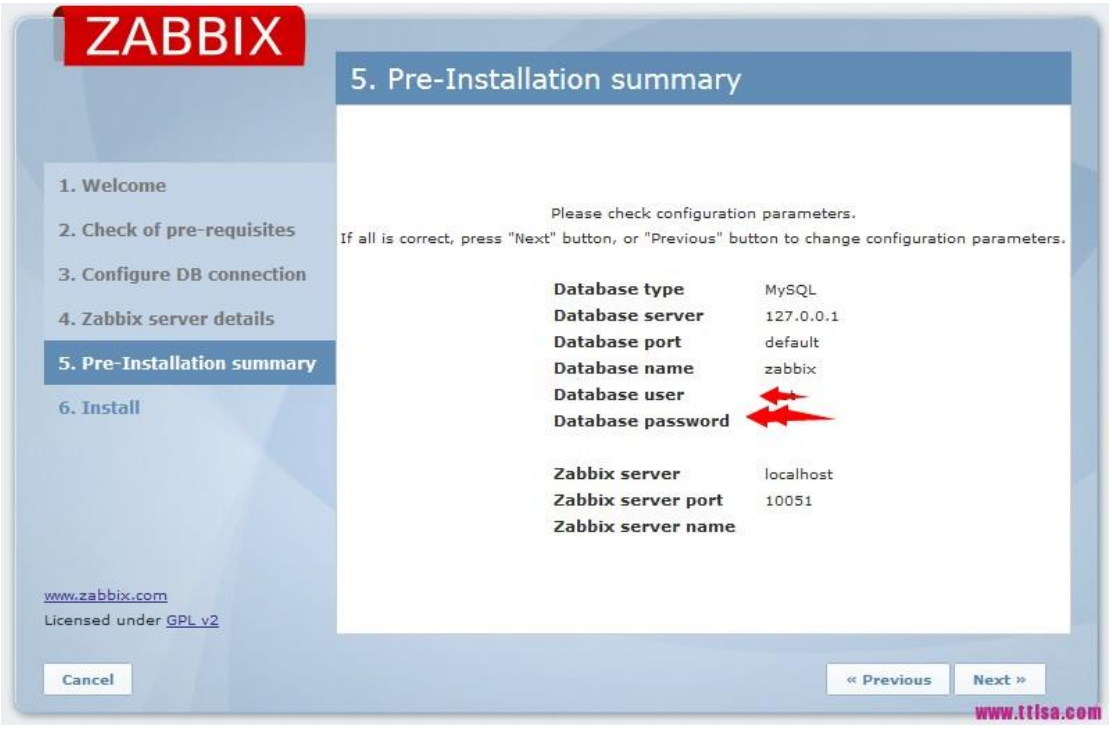
4.3.2 php 需求检查



4.3.3 MySQL 配置

4.3.4 zabbix 服务端详细信息

4.3.5 zabbix 安装前信息列表



4.3.6 安装完成

如果失败了，一般情况是 php 对 zabbix 没有写权限

4.3.7 登陆 zabbix

默认用户名：Admin，密码：zabbix

如下是 zabbix 首页

Zabbix 升级

zabbix 开发者很活跃，版本变更的也比较快，我当前版本是 2.2，但是发现官方 2.4 版本的文档已经悄然上线，不过源码还没放出来。后续也要跟着升级，zabbix 2.0 升 2.2 与 2.2 升 2.4 的方法没一样。如果你从 zabbix 1.6 升级到 1.8 或者 zabbix 1.8 升级到 zbbix 2.0，那么你需要额外执行 sql 文件（path.sql），2.0 之后这些步骤都不需要手动做，如果你的版本很低，那么你需要一步一步升级。本节讲解 2.0 之后的升级方法.

如下为 zabbix 升级方法：

1. 关闭 zabbix server

防止有新的数据提交到数据库中、直接关闭数据库效果也是一样的

2. 备份

2.1 备份数据库

最简单的备份：关闭数据，整个数据库目录 copy 一份。虽说升级一般不会出现什么问题，但是安全起见还是有必要备份一下，就算升级成功，但是不能保证新版本让你喜欢，这个时候回退也有后路。

2.2 备份文件

备份配置文件（通常是/etc/zabbix）、php 网站源码、zabbix 二进制文件（整个程序目录备份就 OK）
不要嫌麻烦，特别是那些运气总是不好的人

3. 安装配置

3.1 安装 Zabbix server

重头安装一次 zabbix，也就是 configure －… make make install.
备注：一般高版本 zabbix server 兼容低版本 zabbix 客户端。如果发现有异常，那么你的 zabbix 客户端也相应升级一下。客户端升级比较简单：更新二进制文件，配置文件对照下是否有修改即可。

3.2 检查配置文件

zabbix_server.conf 配置参数可能会有变化，修改变更后的参数，或者直接修改新的配置文件。

4. 启动 zabbix

启动 zabbix，查看日志（一半在/tmp 目录下），看下 zabbix 的运行是否成功，成功运行之后 zabbix 将会自动更新数据库。启动服务器之前，一定要确保有对 zabbix 数据库有足够的权限（一般情况下，我们都是给所有权限，所以基本不会出现问题）。

5. 部署 zabbix PHP 源码

PHP 源码在 zabbix 源码目录下，不清楚的请参考 zabbix 安装，里面有提到。

6. 总结

总结下 zabbix 的升级方法：备份->重新安装->启动

第三章：快速上手

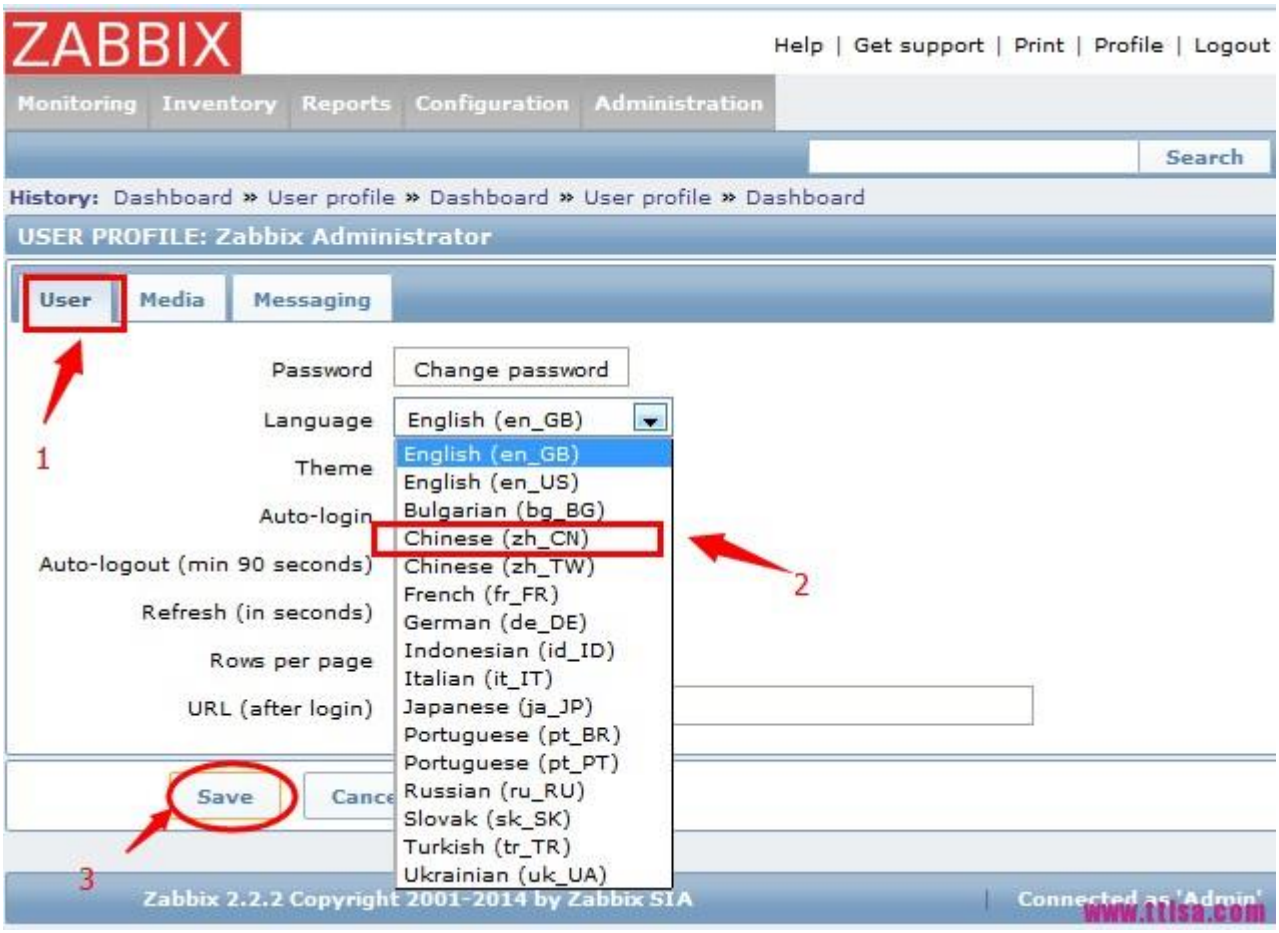
Zabbix 中文语言

zabbix 自带多种语言包，当然也包含中文。登陆到 zabbix web 控制台默认是英文，对有些英文不好或者习惯中文的人来说会有不适应。这边也不是讲 zabbix 汉化，实际上是切换到中文版本。

1. 默认登陆界面（英文版）

2. 点击 Profile（配置）

3. User 标签的 Language 改为 Chinese(zh_CN),点击 save 即可

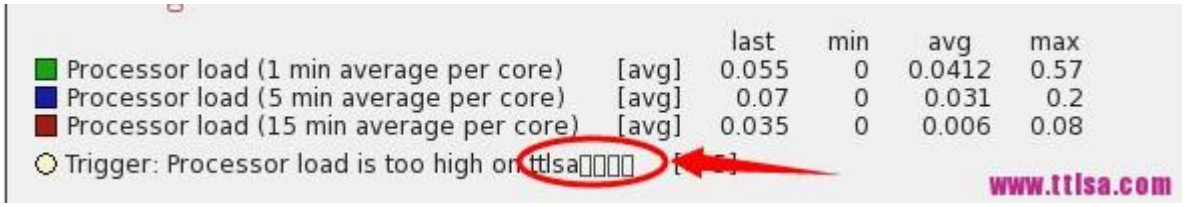

ZABBIX
Help | Get support | Print | Profile | Logout
Monitoring Inventory Reports Configuration Administration
Search
History: Dashboard » User profile » Dashboard » User profile » Dashboard
USER PROFILE: Zabbix Administrator
User Media Messaging
Password Change password
Language English (en_GB)
Theme
Auto-login
Auto-logout (min 90 seconds)
Refresh (in seconds)
Rows per page
URL (after login)
English (en_GB)
English (en_US)
Bulgarian (bg_BG)
Chinese (zh_CN)
Chinese (zh_TW)
French (fr_FR)
German (de_DE)
Indonesian (id_ID)
Italian (it_IT)
Japanese (ja_JP)
Portuguese (pt_BR)
Portuguese (pt_PT)
Russian (ru_RU)
Slovak (sk_SK)
Turkish (tr_TR)
Ukrainian (uk_UA)
1
2
3
Save
Zabbix 2.2.2 Copyright 2001-2014 by Zabbix SIA
Connected as 'Admin'
www.ttlsa.com

4. 汉化后的界面如下


ZABBIX
帮助 | 获取支援 | 打印 | 基本资料 | 登出
监测中 资产记录 报表 组态 管理
总览 | Web | 最新数据 | 触发器 | 事件 | 图形 | 筛选
拓扑图 | 探索 | IT服务
搜寻
历史记录: User profile » Dashboard » User profile » Dashboard » User profile
个人仪表板
www.ttlsa.com

Zabbix 中文乱码

1. 乱码如下图

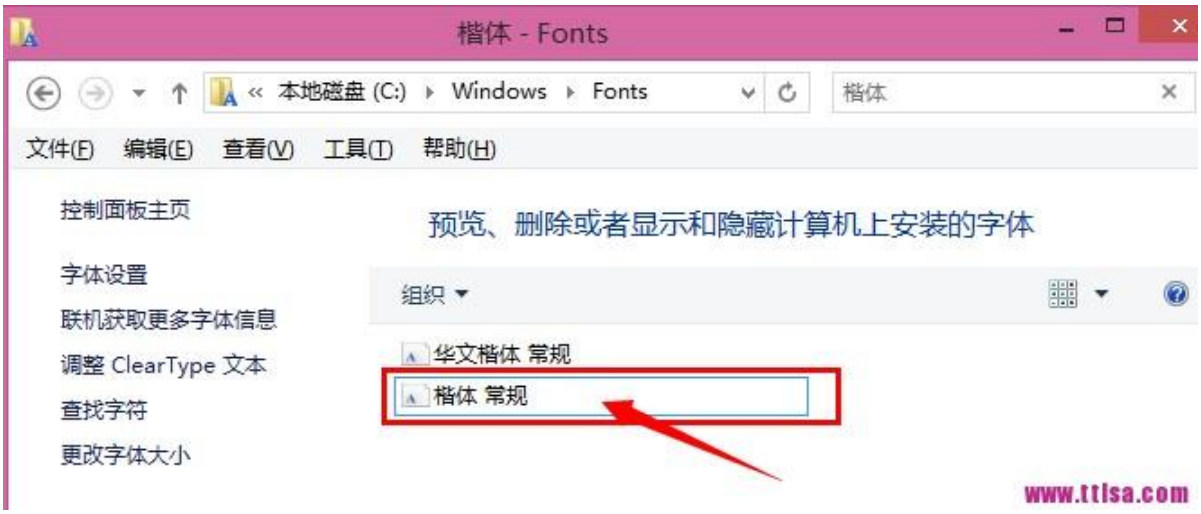


2. 解决 zabbix 乱码方法

2.1 上传文件到 zabbix 中

找到本地 C:\Windows\Fonts\simkai.ttf（楷体）上传到服务器 zabbix 网站目录 fonts 目录下。

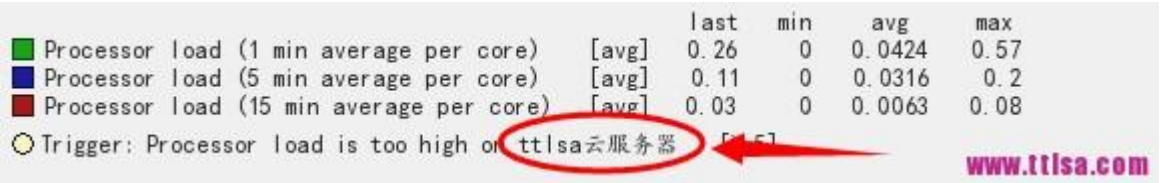
Linux



2.2 修改 zabbix php 配置文件

```
# cd /data/site/www.ttlsa.com/zabbix/ #  我 zabbix 安装目录
# sed -i 's/DejaVuSans/simkai/g' ./include/defines.inc.php
```

3. 查看 zabbix 乱码处理结果



4. 常见问题

依旧乱码：通过以上的操作，大部分同学的乱码问题解决了，但是依旧有一些同学还是乱码？细心的群友提供另外一种情况：初始化数据库的时候未使用 utf8 编码所致.初始化数据库使用命令

```
create database zabbix default charset utf8;
```

或者 my.cnf 增加如下配置

```
default-character-set = utf8
```

zabbix 监控第一台服务器

前面一共花了 9 篇文章介绍 zabbix 的基础内容。今天进入正题，使用 zabbix 监控我们第一台服务器。月初我们 ttlsa.com 刚换到阿里云，zabbix 监控还没部署，今天拿他来当小白鼠。

1. 安装 zabbix 客户端

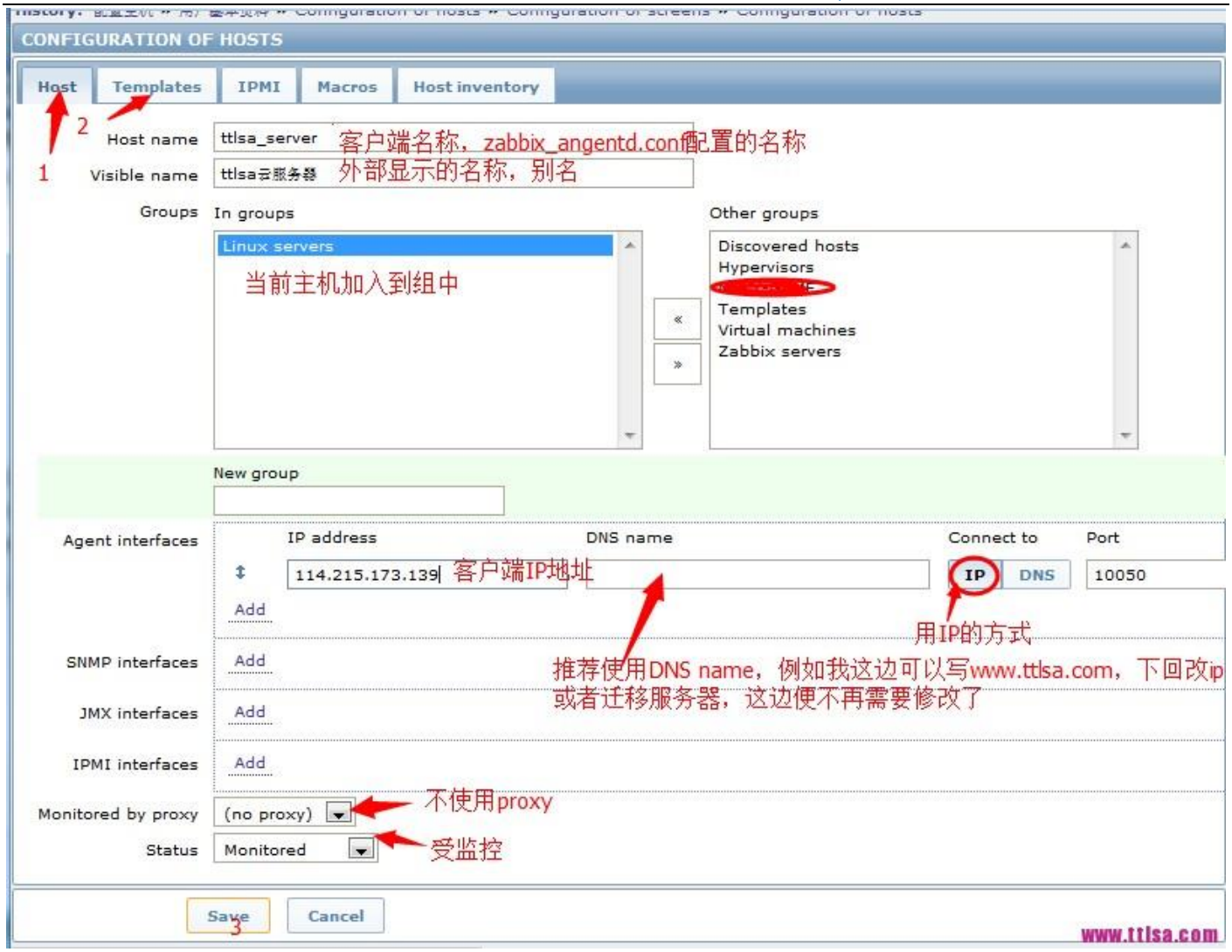
请参考《zabbix 安装》内容中的 zabbix 客户端安装配置。

2. zabbix 监控服务器

创建主机，选择模板以及录入基本信息，过一分钟左右，就可以看到 cpu、内存、硬盘等等使用情况。本节以图文为主。by the way, zabbix 中文翻译很烂，configuration 翻译成组态，screen 翻译成筛选。因此本节以英文图文为主。

2.1 创建主机

configuration（配置）->Hosts（主机）->Create host（创建主机）



2.2 链接监控模板 Template OS Linux

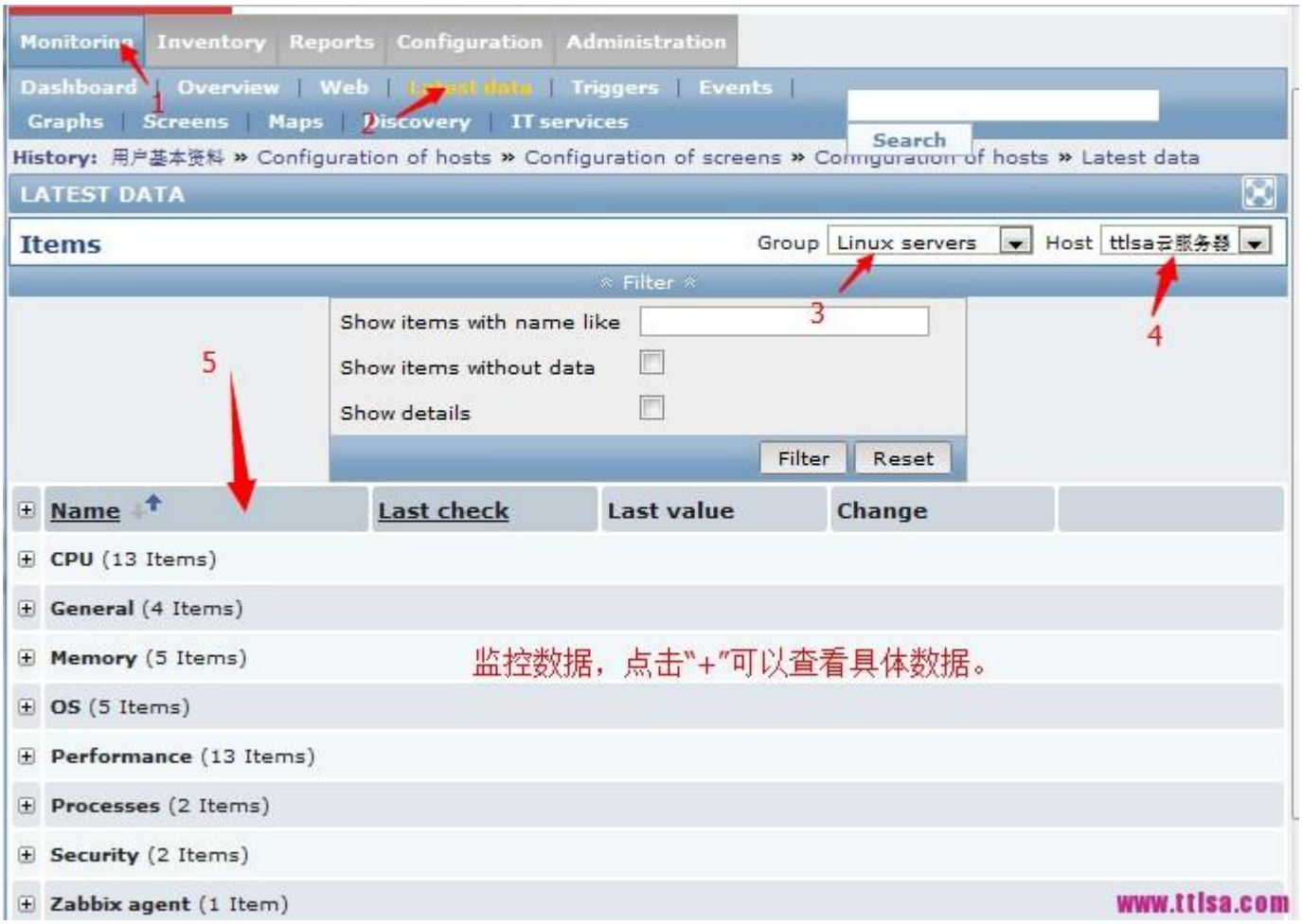
2.3 查看主机列表

绿色的 Z 表示成功的监控了这台客户端，如果是红色 Z 表示失败，此时将鼠标移动到红色 Z 上，会有具体的提示。

3. 查看监控数据

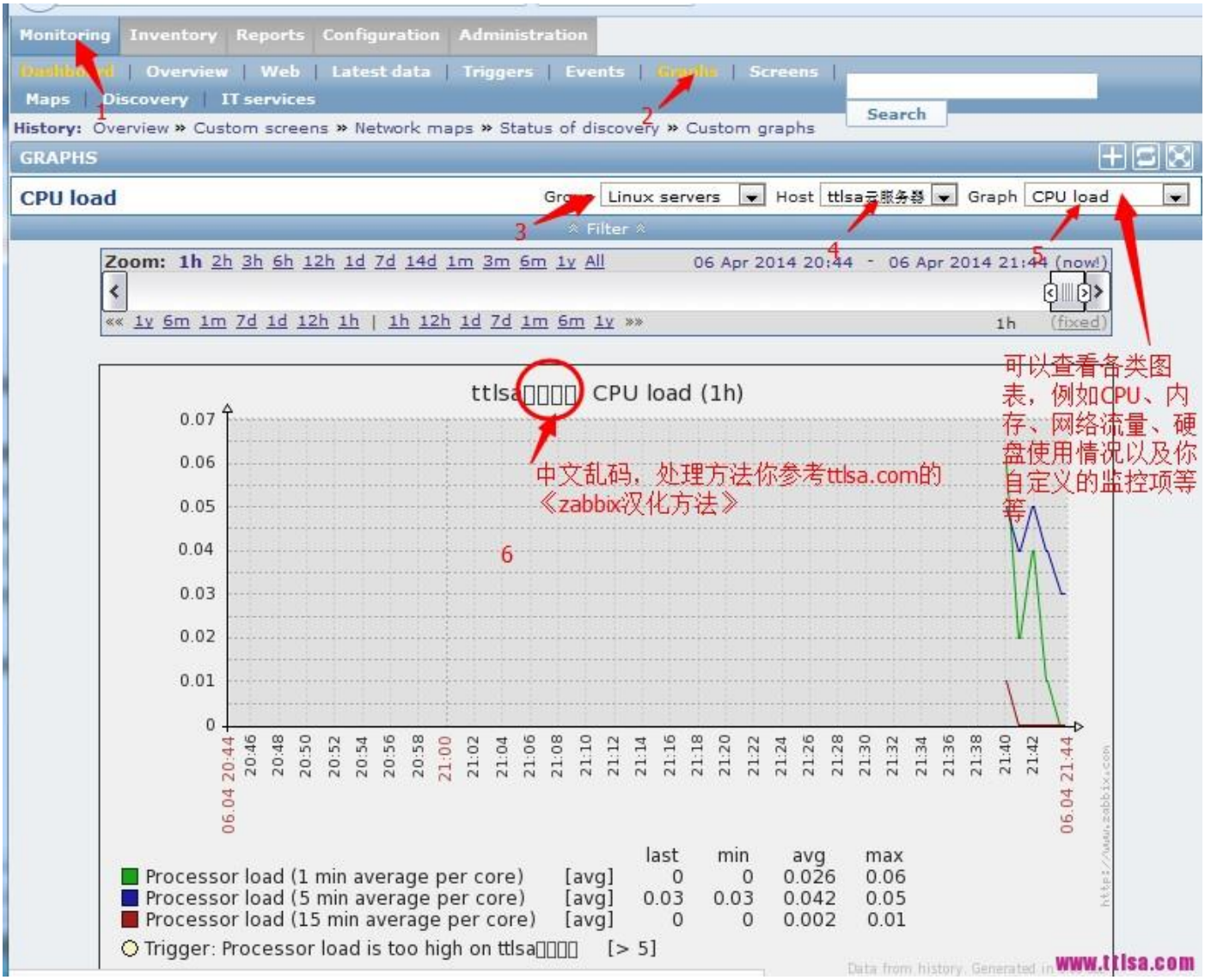
3.1 最新数据

第一台主机添加完成之后，我们便可以查到最新的数据，例如 cpu、内存、硬盘等情况



3.2 图表数据

模板 Template OS Linux 一共包含图表，查看方法如下图



Zabbix 用户管理

这个章节主要介绍 zabbix 用户管理，包括用户增删改查、用户报警媒介管理、用户权限管理。如果你忘记了 zabbix 的管理地址，那么回到《zabbix 安装》一节。

登陆 zabbix

默认账号：Admin，密码：zabbix，这是一个超级管理员。登陆之后在右下角可以看到“connected as Admin“（中文版：连接为 Admin）。

组介绍

我们常用的组有 zabbix administrators，超级管理员组其他没怎么用。至于 Guests 用户，使用 guest 账号，密码为空登陆，只能看到 zabbix 后台，没有具体内容，意义不是很大，但是你可以相应的添加权限，不过这不在本节的范围。

添加用户

有 3 项信息要填写

属性	描述
用户	账号密码、所属组等基本信息
示警媒介	报警相关信息，例如邮箱地址、接受报警时段
许可权	权限，当前用户对哪些主机有权限，我们选择”超级管理员“。

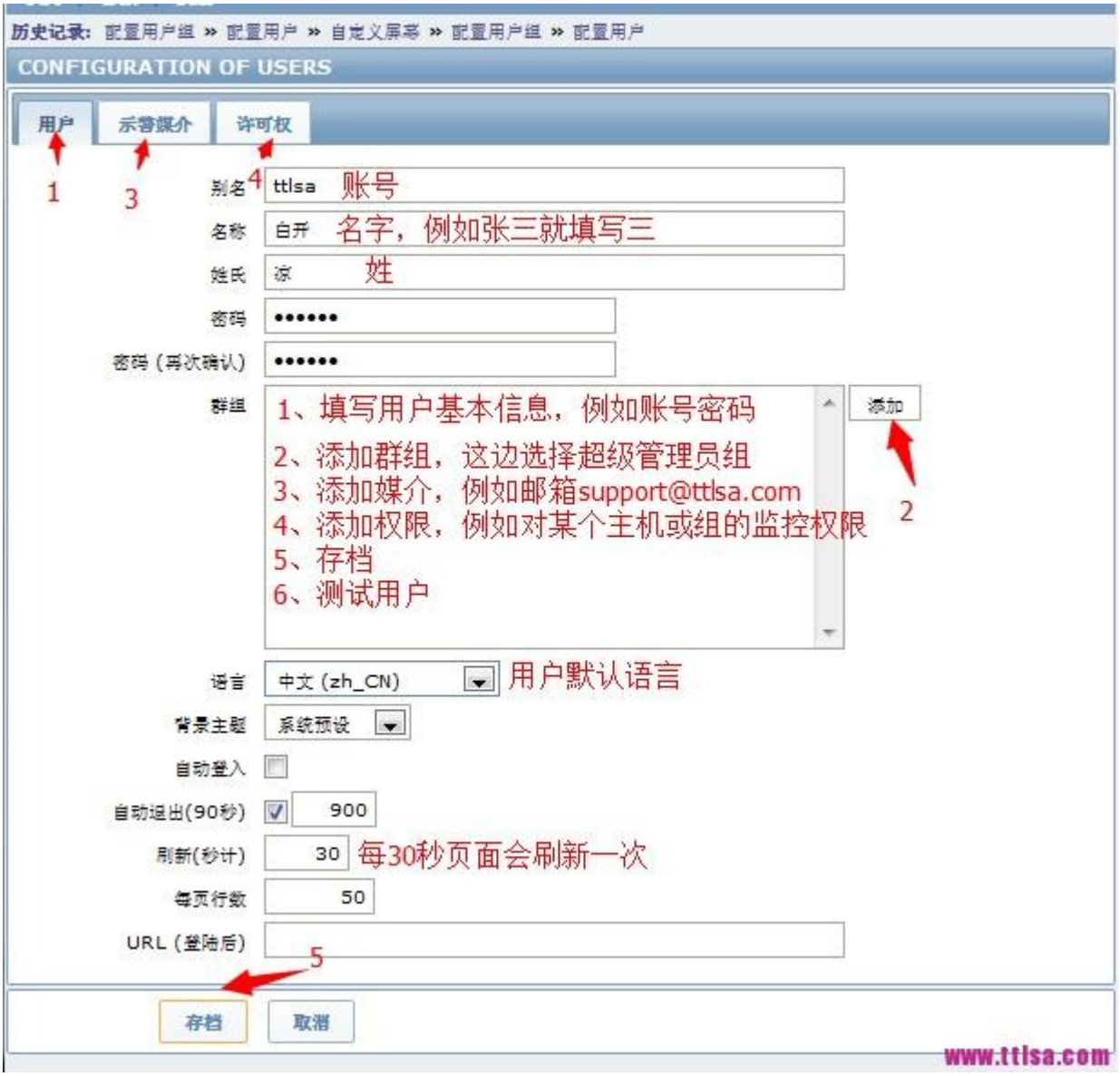
用户信息

1. 创建用户

管理->用户->选择用户->创建用户，如下图


Firefox
配置用户
monitor
ZABBIX
帮助 | 获取支援 | 打印 | 基本资料 | 登出
监测中 资产记录 报表 组态 管理
一般 | 分布管理 | 认证 | 用户 | 示警媒介类型 | 脚本 |
审计 | 队列 | 警报 | 安装
搜寻
历史记录: 自定义图表 » 自定义屏幕 » 配置用户组 » 配置用户 » 配置用户组
CONFIGURATION OF USERS
用户
创建用户
用户
用户群组 所有
显示 1 到 2 共 2 已发现
别名 名称 姓氏 用户类型 群组 是否在线?
Admin Zabbix Administrator 超级管理员 Zabbix administrators 是 (Sun, 06 Apr 2014 11:53:
guest 用户 Guests 不 (Sun, 06 Apr 2014 11:40:
解除封锁所选的
确认 (0)
www.ttlsa.com
1
2
3
4

2、填写用户基本信息


历史记录: 配置用户组 » 配置用户 » 自定义屏幕 » 配置用户组 » 配置用户
CONFIGURATION OF USERS
用户
示警媒介
许可权
1
3
4
别名 ttlsa 账号
名称 白开 名字，例如张三就填写三
姓氏 凉 姓
密码
密码（再次确认）
群组
添加
2
1、填写用户基本信息，例如账号密码
2、添加群组，这边选择超级管理员组
3、添加媒介，例如邮箱support@ttlsa.com
4、添加权限，例如对某个主机或组的监控权限
5、存档
6、测试用户
语言 中文 (zh_CN) 用户默认语言
背景主题 系统预设
自动登入
自动退出(90秒) 900
刷新(秒计) 30 每30秒页面会刷新一次
每页行数 50
URL（登陆后）
5
存档
取消
www.ttlsa.com

3、选择组

媒介信息

1、添加媒介

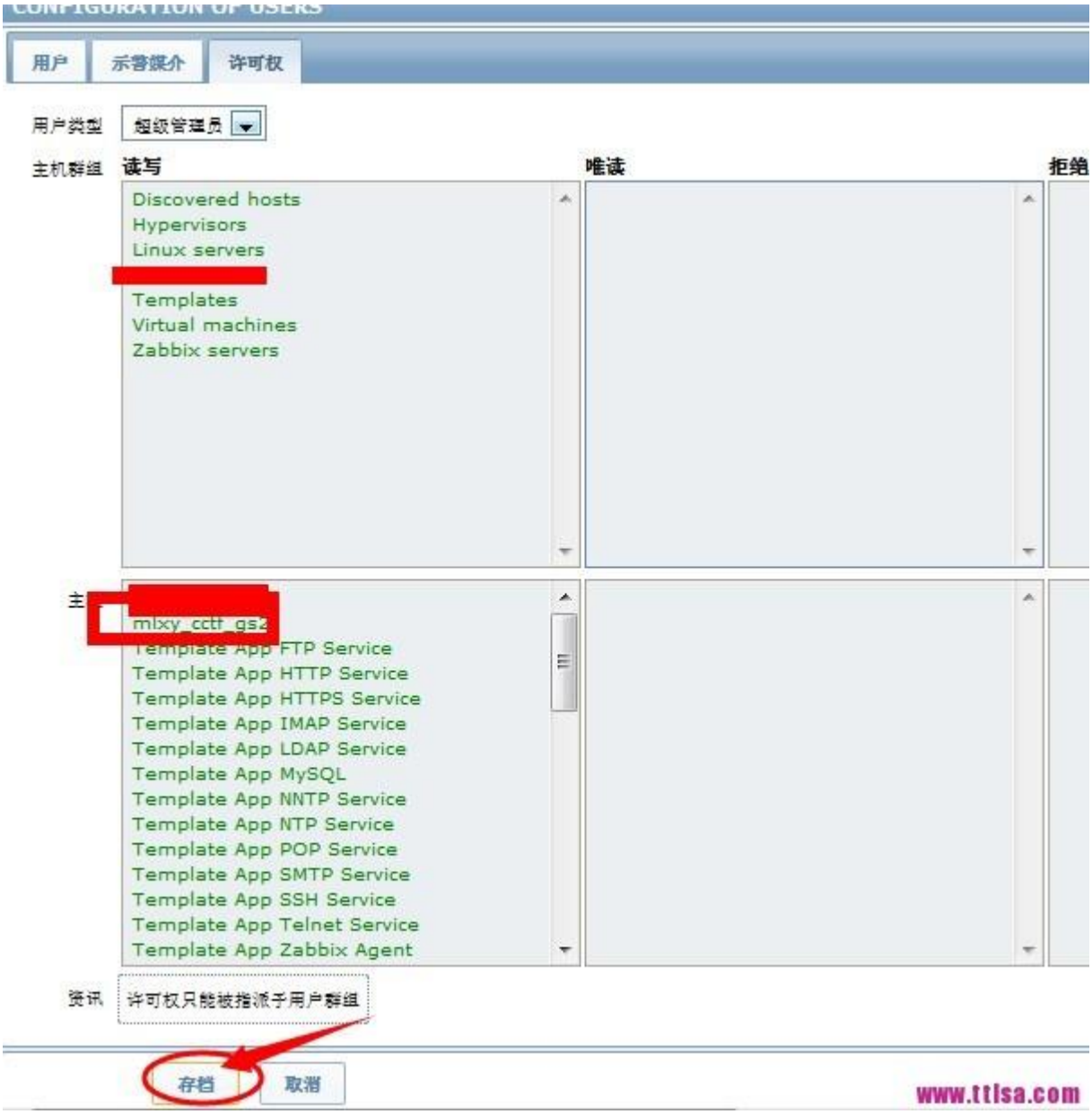
2、录入邮箱信息

如下信息，如果服务器异常、修复异常的时候都会发送邮件到 support@ttlsa.com 中。

权限配置

1、权限信息

因为我们在用户信息以及选择了超级管理员，所以这边默认加入了超级管理员权限，直接存档即可。


用户
示警媒介
许可权
用户类型
超级管理员
主机群组
读写
唯读
拒绝
Discovered hosts
Hypervisors
Linux servers
Templates
Virtual machines
Zabbix servers
Template App FTP Service
Template App HTTP Service
Template App HTTPS Service
Template App IMAP Service
Template App LDAP Service
Template App MySQL
Template App NNTP Service
Template App NTP Service
Template App POP Service
Template App SMTP Service
Template App SSH Service
Template App Telnet Service
Template App Zabbix Agent
资讯
许可权只能被指派予用户群组
存档
取消
www.ttlsa.com

添加完成

zabbix 新用户登陆

用刚才的用户登陆试试，账号 ttlsa，密码：******（我保密了）

登陆成功可以看到 connected as “ttlsa”

第四章：zabbix 配置

配置简介

经过前面十篇文章，我们已经知道如何部署 zabbix 监控，并且使用 zabbix 监控服务器基本的监控项（例如：cpu、内存、硬盘等）。这只是一个入门，zabbix 的功能远不止这些

zabbix 配置内容比较多，我们要分为 9 大块来讲解。分别如下：

- 主机与组

  不用多数，顾名思义，他是添加主机配置与组配置。

- 监控项

  需要监控的项目，例如服务器负载可以使一个监控项。系统自带大部分监控项，一些特定的监控项我们可以自定义，自定义监控项的方法也会在《zabbix 监控项》这节谈到。

- 触发器

  什么情况下出发什么事情，称之为触发器。例如：定义如果系统负载大于 10 那么报警，这个东西可以称之为触发器。

- 事件

  触发器状态变更、Discovery 事件等

- 可视化配置

  图表配置，讲监控的数据绘制成曲线图。或者在一个屏幕中可以看到某台主机所有监控图表。

- 模板配置

  自定义监控模板。例如 Template OS Linux

- 报警配置

  配置报警介质：邮箱、sms 以及什么情况下发送报警通知。

- 宏变量

  用户自定义变量，很有用的一个功能。

- 用户与组管理

  这不是讲过了么？之前简单一笔带过，这次来个详细点的。

zabbix 主机与组配置

什么是主机（Host）？这边有必要介绍一下，主机不单单指类 Linux、window 等服务器，他还包括路由器，交换机等设备。

1. 创建主机方法

1.1 新建主机

configuration（配置）->Hosts(主机)->Create host（创建主机）

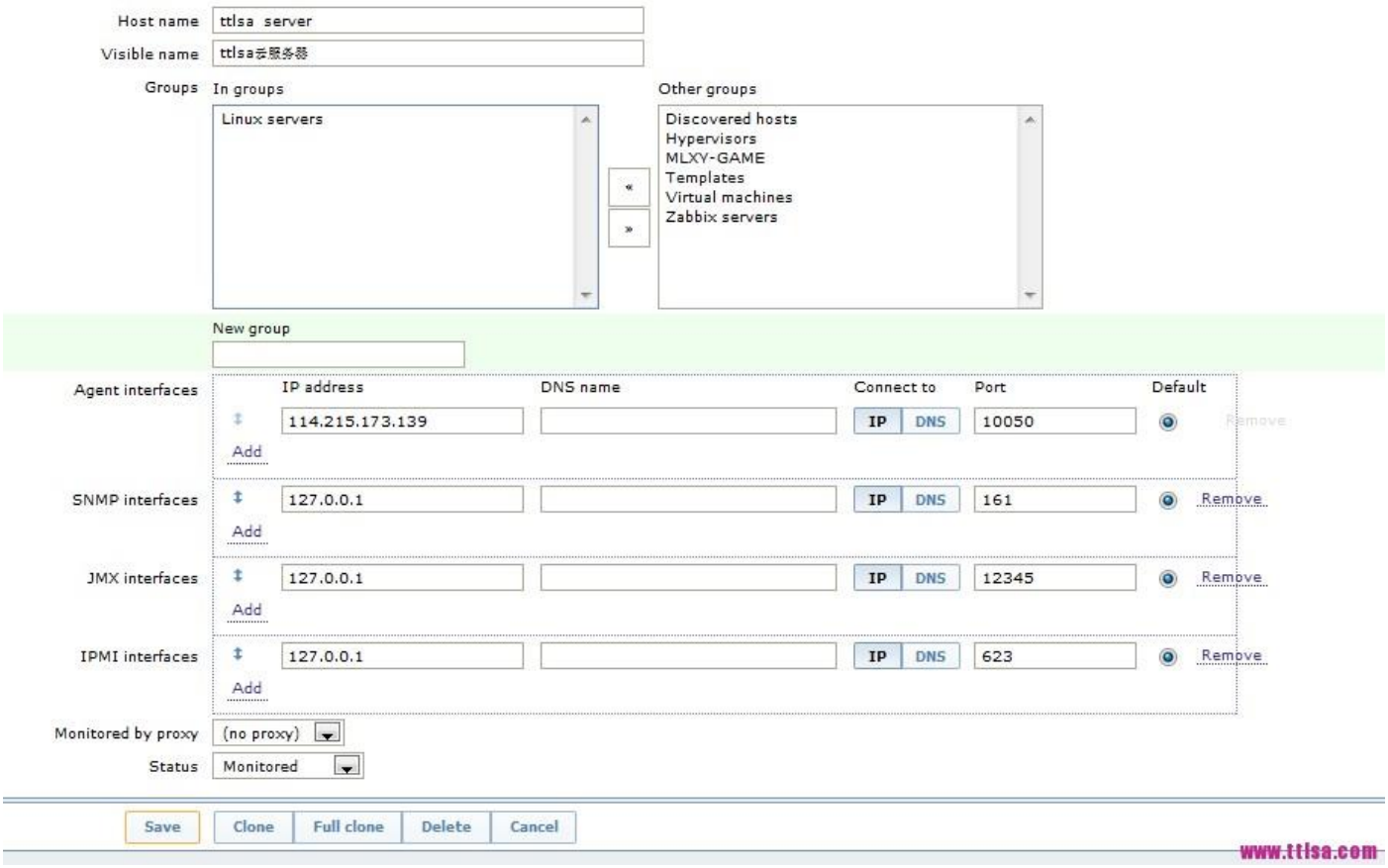
具体的创建方法，请参考《zabbix 监控第一台服务器》

1.2 克隆/完全克隆主机

前面的文章我们有创建名为“ttlsa 云服务器”的主机，configuration（配置）->Hosts(主机)->列表中点击“ttlsa 云服务器”->正下方 save 旁边的 Clone（克隆）或者 Full clone（完全克隆）。然后修改相应资料，最后 save 即可。

2. 主机参数

主机标签相应信息录入，如下图：

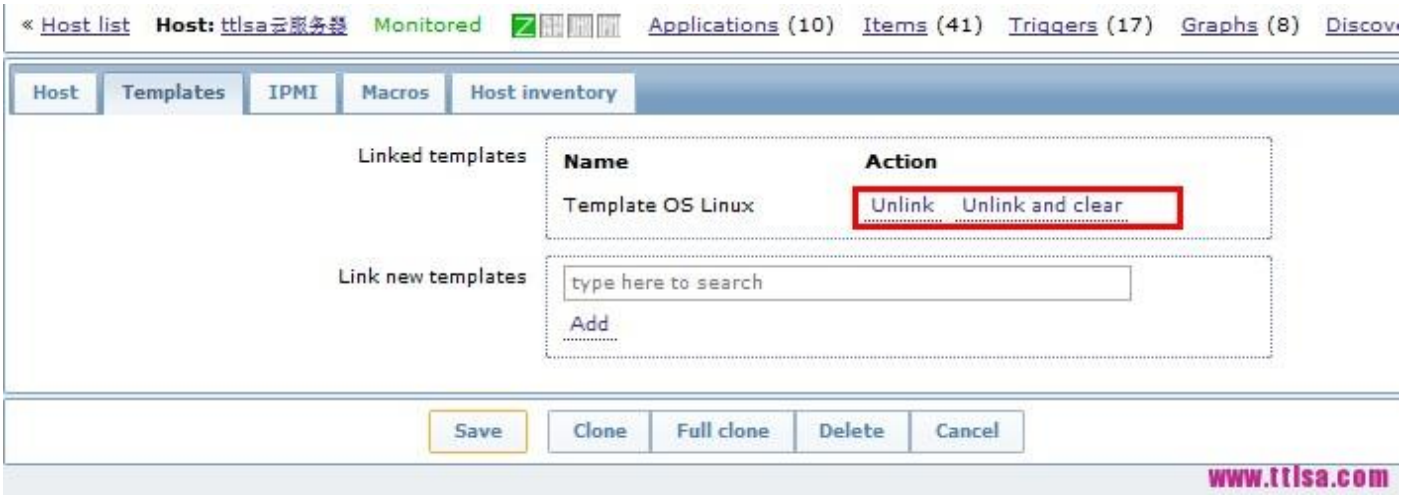


解释如下：

属性	描述
Host name	主机名，只允许数字,空格,句号,下划线,非主流符号它不支持。zabbix 客户端配置文件中的 hostname 名称一定要与这个名字一致。
Visible name	显示名称，可选项。主机名的别名。
Groups	主机要加入的组，一个主机至少要加入一个组
New host group	新主机组，如果已经存在组不是你要的，你可以写上组名，这台主机就加入这个主机组。
Interfaces	主机接口：包含 Agent, SNMP, JMX and IPMI.如果需要增加一个接口，只需要点击“add”即可。键入客户机的 ip 地址即可，推荐使用 ip 地址方式来监控，当然也可以使用域名的方式来监控。zabbix agent 默认端口 10050，snmp 161，jmx 12345，IMPI 623.
Monitored by proxy	是否通过 proxy 监控，默认是 no proxy，由 zabbix server 直接监控。如果选择了”proxy name”（你的代理名称），那么客户机由代理代为收集数据
Status	主机状态，Monitored（被监控）、Not monitored（未被监控）

3. 主机模板

切换到模板选项卡，在文本框里面搜索你要的模板，例如 Linux，会出现 Template OS Linux，Add 即可。如果你想删除模板，选择 unlink 或者 unlink and clear，如下图。



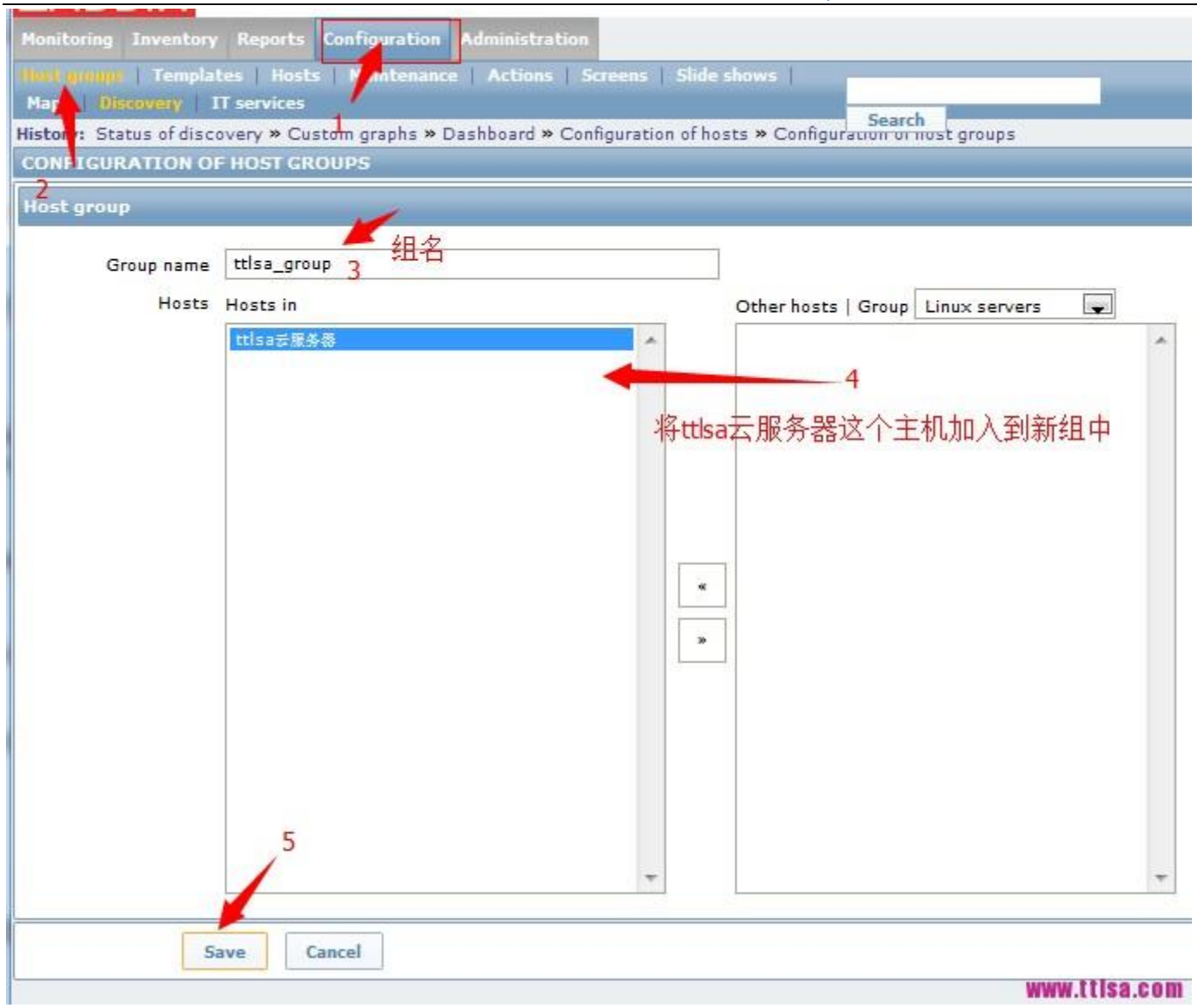
unlink 和 unlink and clear 区别

属性	描述
unlink	取消模板，模板中的 item 依旧保留在 hosts 里
unlink and clear	取消模板，并且删除监控到的数据以及将模板添加到 host 的 item 也删除

4. 主机组

4.1 添加组名

configuration（配置）—>Host Groups（组）—>Create Host Group（创建主机组），录入如下图：



4.2 建组参数

属性	描述
Group name	组名，必须唯一
Hosts	选择主机加入到这个组中，这个组可以为空

zabbix 资产清单 inventory 管理

概述

监控的设备越来越多，有时候搞不清楚哪台服务器是什么配置，大多公司有自己的资产清单，要去专门的系统查询显得多少有点麻烦。为此，zabbix 专门设置了设备资产管理功能。我们创建或者编辑主机的时候，可以看到清单（inventory）功能。里面大致包含 mac 地址、硬件信息等等几十项。

配置资产清单

- 手动模式

当创建或者配置主机的时候，在清单（inventory）选项卡里面，我们选择手动模式，然后输入当前设备的需利好，mac 地址，所在地区，硬件等等信息。

如果相应信息包含带 http 或者 https 的网址，那么在 inventory 选项里面，这个网址会是一个可以点击的超链接。例如，在 os 里面我说如 http://www.ttlsa.com。那么在 inventory 里面，你可以看到 http://www.ttlsa.com 带上了超级链接。

- 自动模式

如上清单选项卡，如果选择了自动模式，部分信息会被自动填充，例如：主机名,系统信息。不过其他的信息还是需要自己输入。这个自动仅仅是把基本的信息给自动获取到，大部分还是要自己手动补充，这项多算个半自动模式。

资产基本信息

点击菜单“资产清单（inventory）”—>主机（选择 ttlsa 云服务器）—>”基本信息（overview）”，可以看到 ttlsa 这台云服务器的基本信息：主机名，监控接口，系统名称，配置打字信息等等。

资产详细信息

点击菜单“资产清单（inventory）”—>主机（选择 ttlsa 云服务器）—>”详细信息（detail）”，可以看到这台服务器更为详尽的信息。

大家可以看到如上信息，除了 Name 和 OS 是自动生成的意外，另外一些是我手动输入的。也可以看到 www.ttlsa.com 是个超级链接

基本信息介绍

参数	描述
Host name	当前主机的名称
Visible name	用来对外显示的名称，例如 ttlsa 云服务器就是 visible name
Host (Agent, SNMP, JMX,	客户端接口地址

IPMI)interfaces	
OS	主机系统
Hardware	主机硬件
Software	主机软件
Latest data	连接到当前最近最新监控数据: Web, Latest data, Triggers, Events, Graphs, Screens.
Configuration	连接到当前主机各种配置: Host, Applications, Items, Triggers, Graphs, Discovery, Web.

清单变量

我们在发送报警通知可以使用变量{INVENTORY.*}来发送故障服务器的基本信息。或者说这台服务器故障了，我们需要发送邮件给当前服务器的联系人，我们可以发送邮件给{INVENTORY.CONTACT1}或者发短信给{INVENTORY.POC.PRIMARY.PHONE.A1}。{PROFILE.*} 宏变量依旧支持使用，当时现在强烈对剑你使用{INVENTORY.*}来代替

zabbix 监控项 item

介绍

Items 是从主机里面获取的所有数据。通常情况下我叫 itme 为监控项，例如我们 ttlsa 云服务器加入了 zabbix 监控，我需要监控它的 cpu 负载，那么实现这个方法的东西就叫 item。接下来 zabbix 教程中提到的 item 都翻译为监控项。

item 构成

item 由 key+参数组成，item 详细介绍请看下回分析。

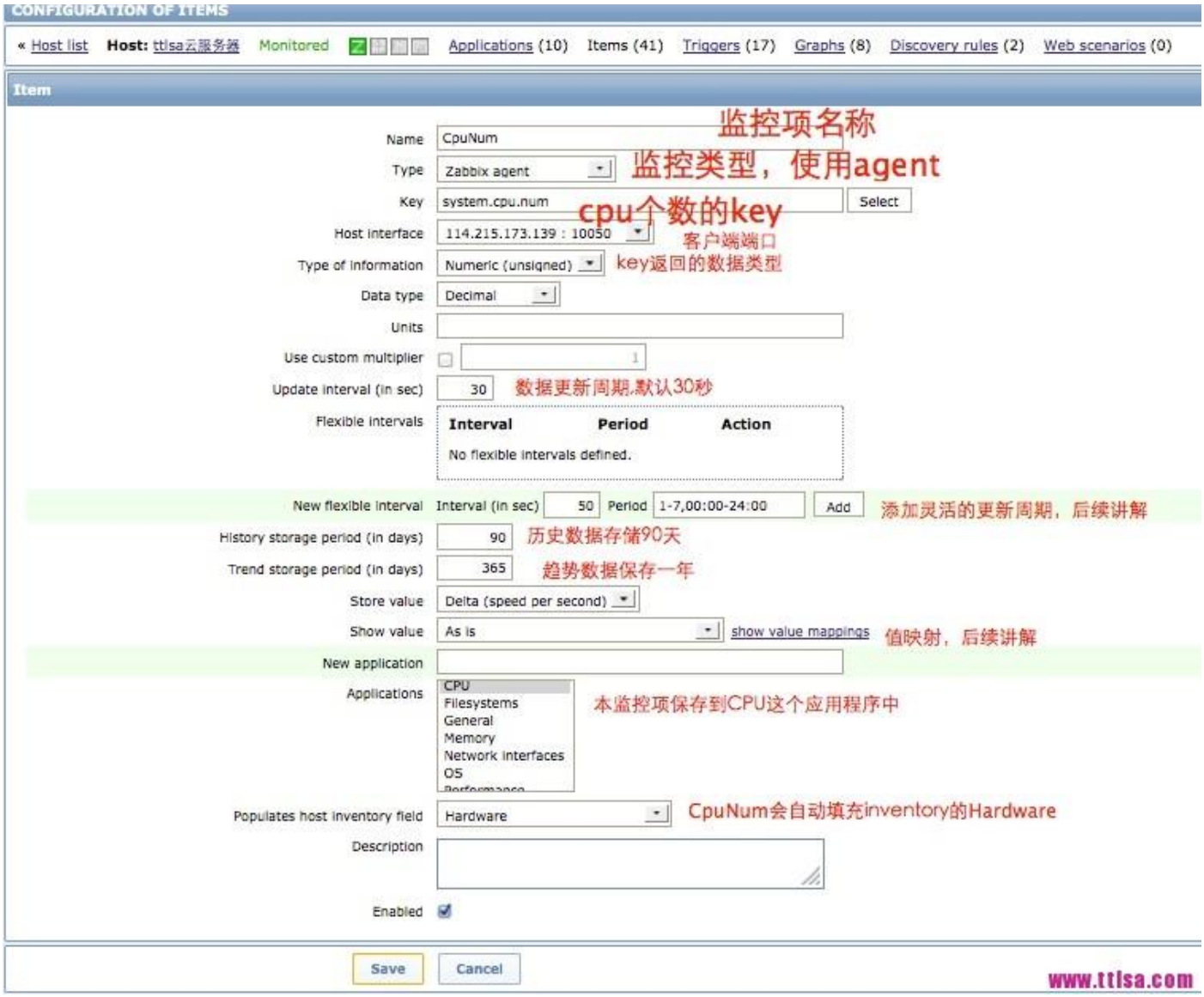
监控项中需要获取 cpu 信息，则需要一个对应的监控 key：system.cpu.load。如果是获取网卡流量，那么获取网卡这个监控项需要 key：net.if.in 或者 net.if.out。

一般情况下 key 要与参数结合起来使用，例如获取 5 分钟的负载情况：system.cpu.load[avg5]，avg5 是对应的参数，如果是 1 分钟则使用 avg1，如果是 15 分钟则使用 avg15(有人会问，如果是 2 分钟是不是 avg2，-！-，没有这种说法)。网卡流量 net.if.in[etho]，使用 etho 作为参数可以获取到 etho 网卡的进入流量，同样看一看换成 eth1，eth2 等等。

zabbix 创建监控项 item

1. 创建监控项

点击配置（configuration）->主机（Hosts）->在你要配置的主机一栏上点击 Items->点击 create item。具体看截图，各个参数我都已经标注清楚了。



Item 属性详解

属性	描述
Host	主机或者模板
Name	监控项 item 名称可以使用如下宏变量： $1, $2…$9，这 9 个参数对应 item key 的参数位置。 例如： Free disk space on $1 如果 item key 为“vfs.fs.size[/,free]”,那么对应的名称会变成”Free disk space on /“， $1 对应了第一个参数”/“,你明白了吗？
Type	item 类型（常见 agent、SNMP、agent（active）等）,请看后续监控项类型相关文章.
Key	监控项 item 的 key.点击 select 可以看到系统很多自带的 key,也可以看到用户自定义的

	key，如何自定义 key，情继续关注 ttlsa 后续的 zabbix 教程.
Host interface	主机接口，例如 agent、SNMP、impi 等
Type of information	获取到得数据类型 **Numeric (unsigned)** - 64bit unsigned integer **Numeric (float)** - floating point **numberCharacter** - 字符串，最长 255 字节 **Log** - 日志文件. key 必须为 log[]. **Text** - 大小无限制的文本
Data type	定义获取到整数数据的数据类型 **Boolean** - 数据为 0 或者 1.’true’表示 1,’false’为 0，不区分大小写。 如下为 True 和 False 的定义： TRUE - true, t, yes, y, on, up, running, enabled, available FALSE - false, f, no, n, off, down, unused, disabled, unavailable 任何非 0 数字都被认为是 TRUE，0 被定义为 FALSE.负数呢? **Octal** - 八进制 **Decimal** - 十进制 **Hexadecimal** - 十六进制 zabbix 将会自动把他们转为数字
Units	默认情况下，如果原始值超过 1000，那么他会先除以 1000 并且显示出来例如，设置了单位为 bps 并且收到的值为 11102，将会显示为 11.1Kbps 如果单位被指定为 B (byte), Bps (bytes per second) ，那么它会除以 1024 然后再显示数据。所以大家在监控流量和文件大小的时候不要用错单位，否则会出现数据不一致的情况。 如下为时间单位: **unixtime** - 转为 “yyyy.mm.dd hh:mm:ss”. 只能使用正数。 **uptime** - 转为“hh:mm:ss” 或者“N days, hh:mm:ss” 例如，收到的值为 881764 秒，他将会显示为“10 days, 04:56:04” **s** - 转为“yyy mmm ddd hhh mmm sss ms”; 例如，收到的值为 881764(单位秒)，他将会被显示为 10d 4h 56m”，只会显示 3 个单元。有时候只会显示 2 个单元，例如”1m 5h”（不包含分，秒，毫秒），如果返回的值小于

	0.001，他只会显示”<1 ms”禁用单位：ms、rpm、RPM、%
Use custom multiplier	如果启用这个选项，所有接收到的整数或者浮点数都会乘以这个文本框里面的值。使用这个选项，zabbix 将会把收到的 KB,MBps 等数据先转为 B,Bps。否则 zabbix 不能正确设置前缀（K,M,G 等等）. zabbix 2.2 开始支持科学计数法，例如：1e+70.
Update interval (in sec)	数据更新时间注意：如果设置为 0，那么这个数据将永久不更新。但是如果在灵活更新间隔（flexible interval）里面设置了一个非 0 间隔，那么以这个为准
Flexible intervals	可以创建例外的更新间隔，例如: Interval:10,Period:1-5,10:00-19:00,表示周一到周五的早上 10 点到晚上 19 点每十秒更新一次数据。其余时间使用默认值。这边最多只能设置 7 个灵活更新间隔.如果设置的多个灵活时间间隔有冲突，那么他会使用最小的时间间隔。 两个注意点：如果时间间隔被设置为 0，那么数据永久不会更新。它不能用在 zabbix 主动方式的 item
Keep history(in days)	历史记录可以在数据库中保存多久，过期的历史数据将会被 Housekeeper 删除. 从 Zabbix2.2 开始，这个值可以被一个全局值覆盖：Administrator->General->Housekeeper->勾选 Keep history（in days），输入你希望历史记录保留的时间。 zabbix 官方推荐大家尽量开启他，尽量使用一个较短的历史记录。如果你想看历史数据的画，你可以将”趋势历史记录 Keep trends”的保留时间设置长一点。
Keep trends(in days)	趋势数据（以小时为单位的 min，max，avg，count 的数据）在数据库中保留时常，过期数据将会被 HouseKeeping 删除。 从 zabbix2.2 开始.这个值可以被一个全局值覆盖（请参考上面的 Keep history） 备注：趋势数据只能存数字类型数据，字符、日志这些都无法存储。
Store value	**As is** – 数据不作处理 **Delta (speed per second)** – 计算值公式为 (value-prev_value)/(time-prev_time) value – 获取到得原始值 value_prev – 上次接收到的值 time – 当前时间 prev_time – 上次接收数据的时间一般用于数据增长的类型，例如：

	网卡流量，每次获取到得都是当前网卡总流量。比如第一次给的值是 0 字节（UNIX 时间为 1），第二获取到得是 3000 字节（UNIX 时间为 31），那么套用公式（3000-0）/（31-30），可以得出数据是 100 字节/秒 备注：如果当前获取到的值比上一个值更小，那么 zabbix 会忽略这个值，等待下一次的值 **Delta (simple change)** – 计算公式为 (value-prev_value),value – 当前值 value_prev – 上次获取到得值
Show value	值映射，需要配置数字映射到字符的映射表。例如： 1=>ttlsa.com 访问正常。如果 key 返回的数据为 1，那么监控页面不会显示 1，而是显示 ttlsa.com 访问正常。key 返回的数据只能为整数，并且不做任何修改保存到数据库中。只有在显示的时候才会根据映射表来展示相应的内容。
Log time format	只可以用在 LOG 类型中，支持占位符: * **y**: 年(0001-9999) * **M**: 月(01-12) * **d**: 日(01-31) * **h**: 小时(00-23) * **m**: 分钟(00-59) * **s**: 秒(00-59)如果时间搓留空不会被解析。 例如： 如下为 zabbix agent 日志” 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).”前面 6 个字符是 PID，后面更上日期，时间和日志内容，日志时间类为“ppppp:yyyyMMdd:hhmmss” 备注：“p” 与 ”:” 为占位符，除了“yMdhms”不能为占位符，其它任意字符都可以作为占位符
New application	创建一个新的应用
Applications	包含多个应用，例如： cpu、disk、network，监控项可以属于多个应用
Populates host inventory field	数据自动填充到 inventory 资产清单的相应属性，前提是你的 inventory 处于自动模式
Description	监控项的描述

Enabled	是否启用这个监控项.

创建 item 快捷方法，编辑一个 item，然后克隆这个 item，修改 name 等等其它数据即可。

不可用的 items

由于各种原因，某些 item 的数据无法获取到，但是 zabbix 依旧会再固定的时间间隔内重新获取数据

zabbix item key 详解

上篇文章详细介绍了 zabbix 创建 item，本节详细介绍 item key 的规范，涉及到 key 的名称如何定义，key 的参数如何定义。看完这篇，以前总看不懂的 key 今天算是明白了。

1. 灵活的参数

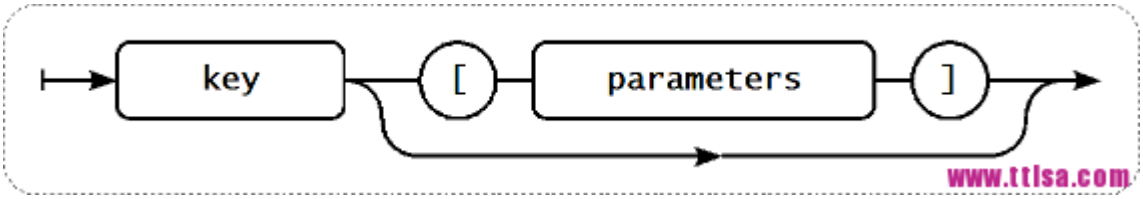
参数位置可用接收任意参数则是灵活的。例如 vfs.fs.size[*]，”*”星号可以使用任意的参数，例如：

vfs.fs.size[/]

vfs.fs.size[/opt]

2. Key 格式

Item key 格式包含 key 名称和他得参数，参数必须符合规范，请看下面的图片。key 的定义要遵循箭头从做到右的规则，如果都符合，那这个 key 合法，否则不合法。大致流程是:首先验证 key 名是否合法，如果存在参数那么验证参数是否合法，如果没有参数那直接跳过。如下图的意思大家现在懂了吗？



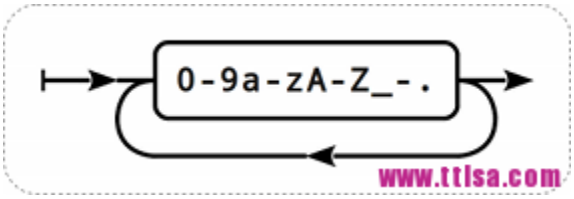
3. Key 名称

key 允许如下字符作为名称：

0-9a-zA-Z_-.

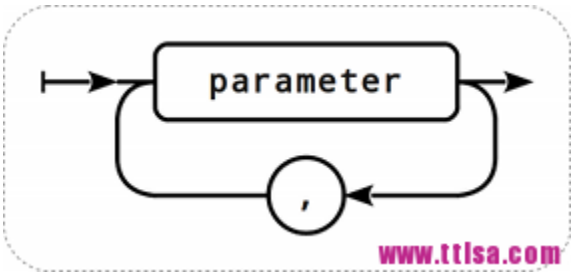
大致意思是说允许字符:所有数字、有大小写字幕、下划线、减号、点.

key 名称从左到右走下规则，只要又一个字符不符合，那么 key 就不合法。

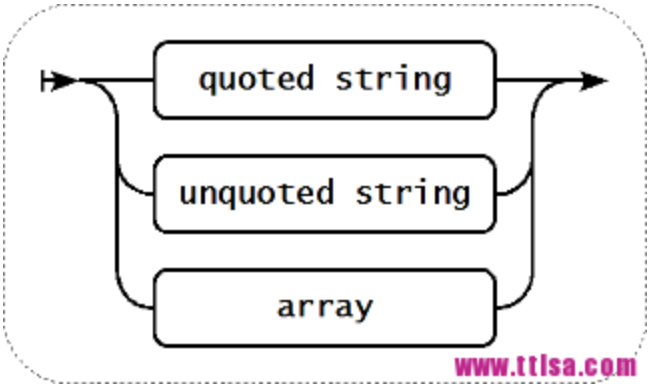


4. Key 参数

item key 可以又多个参数，他们之间用逗号’,’分开。如下图



key 参数可以是带引号的字符串、不带引号的字符串以及数组。如下图。

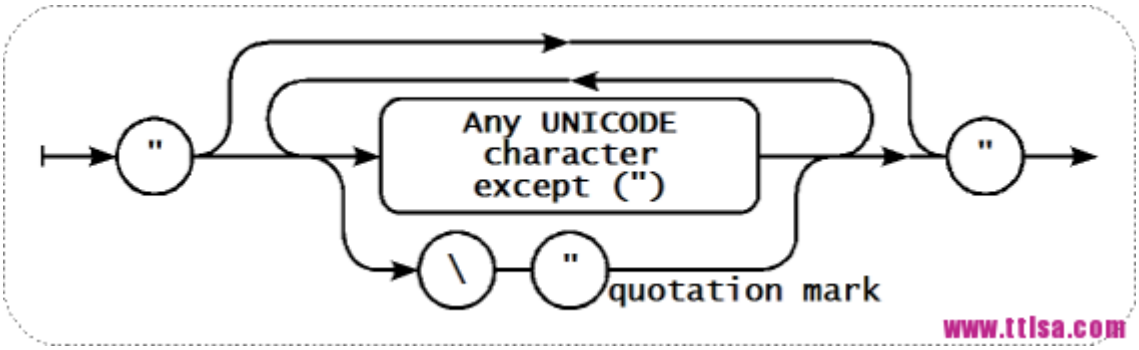


属性	描述
quoted string	带引号字符串
unquoted string	不带引号字符串
array	数组

参数如果为空，那么将会使用他设置的默认值。例如 key icmpping[,,200,,500]，其中 3 个参数都为空，那么每 200ms 会 ping 一次，超时时间为 500ms，其它为空的参数使用默认值。

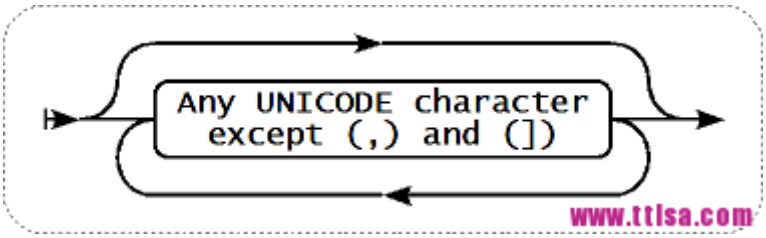
4.1 参数- 带引号

如果 key 参数带引号，那么任何 unicode 数据都合法，如果参数里面带有双引号，那么要使用”\”来转义。具体如下



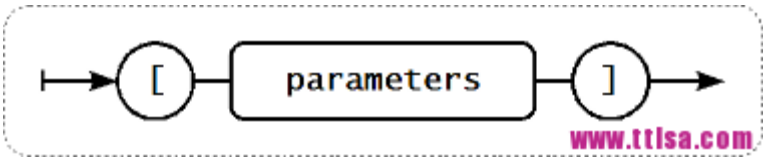
4.2 参数- 不带引号

如果 kye 参数是一个不带引号的字符串，除了逗号和右中括号其他任何 unicode 字符串都合法。具体看如下图



4.3 参数- 数组

如果 key 参数是一个数组，那么数组要多加一对中括号，并且数组里面的参数同样要遵循参数规范，具体如下图

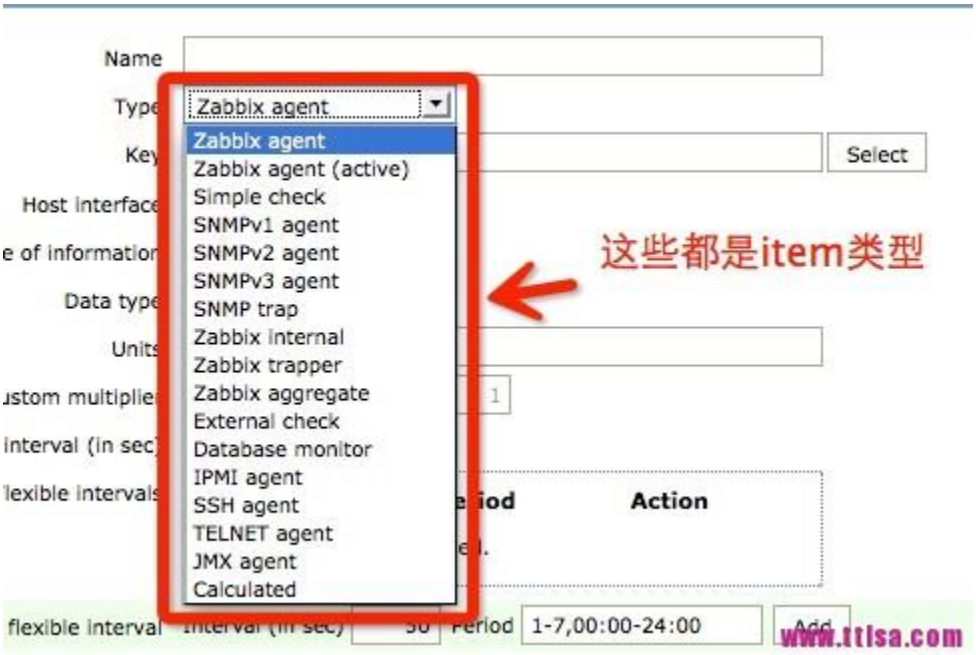


zabbix item types 监控类型

1. 什么是 item types

item types 是由 zabbix 提供的各种类型的检查器（这样翻译很奇怪），大致就是 Zabbix agent, Simple checks, SNMP, Zabbix internal, IPMI, JMX monitoring 等等。

那么在哪里可以看到这些东西呢？在创建或者配置一个监控项的时候。每次创建监控项你都必须选择一个检测类型。看如下图：



2. item types 注意点

从 zabbix 2.0 开始一台主机可以定义多个接口，什么是接口？agent、jmx、impi、snmp 这些都是接口。假如你需要检查他的服务器主板温度等，需要使用 impi； 如果你还需要检测他的 mysql、nginx 之类的，你需要 agent，snmp 等等。

监控什么类型的 item，你需要配置什么类型的接口。如果你配置来多个接口，当需要检查一个 item，zabbix 会依次（Agent→SNMP→JMX→IPMI）调用接口，直到找到合适的接口为止。

有些监控项完全由服务器端来完全，根本不需要 agent，这个大家可以记一下。接下来的大部分时间都要花费在 zabbix

的监控类型。大家要做好心理准备。

zabbix agent 类型所有 key

zabbix 服务器端通过与 zabbix agent 通信来获取客户端服务器的数据，agent 分为两个版本，其中一个是主动一个是被动，在配置主机我们可以看到一个是 agent，另一个是 agent（active）。前者为被动检测，后者为主动检测。那么主动和被动区别在哪里呢？

被动：zabbix server 向 zabbix agent 讨要数据。

主动：zabbix agent 提交数据给 zabbix server。

监控项 keys 列表

以下表格是 zabbix agent 所支持的所有.请大家一一过目，认识他们就行，不需要背下来。

Key			
功能	返回值	参数	描述
agent.hostname			
		-	
agent.ping			
agent.version			
zabbix agent 版本	字符串	-	例如返回： 1.8.2
kernel.maxfiles			
系统支持最大的 open files	整数		
kernel.maxproc			
系统支持最大的进程数量	整数		

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]			
监控日志文件	Log	**file** – 文件详细路径 **regexp** – 正则 **encoding** – 编码 **maxlines** – zabbix agent 向 server 或者 proxy 发送最大的行数。这个参数覆盖配置文件 zabbxi_agentd.conf 中的 'MaxLinesPerSecond' **mode** – 可选值: all (默认), skip (跳过处理老数据).mode 参数从 2.0 版本开始支持 **output** – 可选项，输出格式模板. The \0 escape sequence is replaced with the matched text while an \N(where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups). If <output> is left empty – the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the <output> parameter is ignored. output 是 2.2 中新加入的功能	这个监控项只能设置为 active check（主动模式）. 如果文件不存在或者没有相应的权限，那么监控项状态会转变为 unsupported. 示例: log[/var/log/syslog] log[/var/log/syslog,error] log[/home/zabbix/logs/logfile,,,100]
logrt[file_pattern,<regexp>,<encoding>,<maxlines>,<mode>,<output>]			

监控支持轮转的日志	Log	**file_pattern** – 文件绝对路径 **regexp** – 正则表达式 **encoding** – 编码 **maxlines** – 客户端每秒发送给 server 的最大行数. 这个参数会覆盖客户端配置文件 zabbix_agentd.conf 中 MaxLinesPerSecond 的值 **mode** – 可选值：all（默认), skip (跳过老数据的处理). Mode 参数从 2.0 开始支持 **output** – an optional output formatting template. The \0 escape sequence is replaced with the matched text while an \N(where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups). If <output> is left empty – the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the <output> parameter is ignored. The output parameter	The item must be configured as an active check. Log rotation is based on the last modification time of files. 示例: logrt[“/home/zabbix/logs/^logfile[0-9]{1,3}$”,,,100] – will match a file like “logfile1″ (will not match “.logfile1″) logrt[“ /home/user/logfile_.*_[0-9]{1,3}”,”pattern_to_match”,”UTF-8″,100] – will collect data from files such “logfile_abc_1” or “logfile__001”.

		is supported from version 2.2.	
net.dns[<ip>,zone,<type>,<timeout>,<count>]			
检测 DNS 服务是否开启	0 - DNS 挂了 1 - DNS 运行中	**ip** - DNS 服务器的 ip 地址(留空表示使用本地 DNS, ignored on Windows) **zone** - 需要测试的域名 **type** - 记录类型 (默认为 SOA) **timeout** (ignored on Windows) - 超时时间(默认 1 秒) **count** (ignored on Windows) - 重试次数 (默认值 2)	示例 key: net.dns[8.8.8.8,zabbix.com,MX,2,1] type 可选值: ANY, A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS (除了 windows), HINFO, MINFO, TXT, SRV SRV 记录从 Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows)开始支持 Zabbix 2.0 之前的命名方式: net.tcp.dns（目前依旧支持）
net.dns.record[<ip>,zone,<type>,<timeout>,<count>]			
执行一个 DNS 查询	获取 DNS 查询数据	**ip** - DNS 服务器的 ip 地址(留空表示使用本地 DNS, ignored on Windows) **zone** - 需要测试的域名 **type** - 记录类型 (默认为 SOA) **timeout** (ignored on Windows) - 超时时间(默认 1 秒) **count** (ignored on Windows) - 重试次数 (默认值 2)	示例 key: net.dns.record[8.8.8.8,ttlsa.com,MX,2,1] type 的可选值: ANY, A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS (except for Windows), HINFO, MINFO, TXT, SRV SRV 记录从 Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows)开始支持 Zabbix 2.0 之前的命名方式: net.tcp.dns（目前依旧支持）
net.if.collisions[if]			
Out-of-window collision	Number of collisions	数字	if - 网卡
net.if.discovery			
列出网卡. 通常用于	JSON 对象		Zabbix agent 2.0 开始支持 但是操作系统 FreeBSD, OpenBSD and NetBSD 上的 Zabbix

低级别的discovery			agent 2.2 开始支持
net.if.in[if,<mode>]			
网卡入口流量	整数	**if** - 网卡名称 **mode** - 可用值: **bytes** - 字节数 (默认) **packets** - 包数量 **errors** - 错误数量 **dropped** - 丢包数量	示例 keys: net.if.in[etho,errors] net.if.in[etho] 请使用增量存储方式
net.if.out[if,<mode>]			
网卡出口流量	整数	**if** - 网卡名称 **mode** - 可用值: **bytes** - 字节数 (默认) **packets** - 包数量 **errors** - 错误数量 **dropped** - 丢包数量	范例 keys: net.if.out[etho,errors] net.if.out[etho]请使用增量存储方式
net.if.total[if,<mode>]			
网卡进/出流量的总和	整数	**if** - 网卡名称 **mode** - 可用值: **bytes** - 字节数 (默认) **packets** - 包数量 **errors** - 错误数量 **dropped** - 丢包数量	范例 keys: net.if.total[etho,errors] net.if.total[etho] 请使用增量存储方式
net.tcp.listen[port]			
检测端口是否开启	**0** – 未监听 **1** – 监听中	**port** – TCP 端口	示例: net.tcp.listen[80] linux 下的 zabbix agent 1.8.4 开始支持
net.tcp.port[<ip>,port]			
是否可以连接到指	**0** – 无法连接 **1** – 可以连接	**ip** - IP 地址 (默认是 127.0.0.1)	范例: net.tcp.port[,80]

定的 TCP 端口		**port** – 端口	检测 web 服务器端口是否运行中 老命名方式: check_port[*]
net.tcp.service[service,<ip>,<port>]			
检测服务是否开启，并且端口可用	0 – 服务挂了 1 – 服务运行中	**service** – 如下：ssh, ntp, ldap, smtp, ftp, http, pop, nntp,imap, tcp, https, telnet **ip** – IP 地址 (默认 127.0.0.1) **port** – 端口 (默认情况为标准端口号)	示例 key: net.tcp.service[ftp,,45] – 检测 45 端口上得 FTP 是否运行中 Zabbix 1.8.3 支持的版本请使用 service.ntp 代替 ntp. https 和 telnet 服务从 2.0 和 2.2 开始支持
net.tcp.service.perf[service,<ip>,<port>]			
检测服务器性能	0 – 服务挂了; seconds – 链接到服务器端口消耗的时间	**service** – 如下 :ssh, ntp, ldap, smtp, ftp, http, pop, nntp,imap, tcp, https, telnet **ip** – IP 地址 (默认 127.0.0.1) **port** – 端口 (默认情况为标准端口号)	示例 key: net.tcp.service.perf[ssh] – 检测 SSH 服务器响应速度加密协议检测不被支持 (类似 IMAP 端口 993 或者 POP 端口 995) .但是我们可以使用 net.tcp.service.perf[tcp,<ip>,<port>] 来检测他们.Windows agent 不能检测 LDAP 和 HTTPS. Note that the telnet check looks for a prompt (':' at the end). 老命名方式: check_service_perf[*]注意：that before Zabbix 1.8.3 version service.ntp should be used instead of ntp. Zabbix 2.0 支持 https 和 telnet 服务
net.udp.listen[port]			
检测 UDP 端口是否在监听	**0** – 未监听 **1** – 监听中	**port** – udp 端口	范例: net.udp.listen[68] linux 系统的 Zabbix agent 1.8.4 开始支持
proc.mem[<name>,<user>,<mode>,<cmdline>]			
用户进程消耗的内存	内存使用量 (字节单位).	**name** – 进程名 (默认值“all processes”) **user** – 用户名 (默认值“all	示例 keys: proc.mem[,root] – root 的进程消耗了多少内存

		users”) **mode** – 可选值: avg, max, min, sum (默认) **cmdline** – 命令行过滤(正则表达时)	proc.mem[zabbix_server,zabbix] – zabbix 用户运行的 zabbix_server 使用了多少内存 proc.mem[,oracle,max,oracleZABBIX] – memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line
proc.num[<name>,<user>,<state>,<cmdline>]			
某用户某些状态的进程的数量	进程数量	**name** – 进程名称 (默认“all processes”) **user** – 用户名 (默认 “all users”) **state** – 可用值: all (默认), run,sleep, zomb **cmdline** – 命令行过滤(正则表达时)	示例 keys: proc.num[,mysql] – MySQL 用户运行的进程数量 proc.num[apache2,www-data] – www-data 运行了多少个 apache2 进程 proc.num[,oracle,sleep,oracleZABBIX] – number of processes in sleep state running under oracle having oracleZABBIX in its command line 备注：Windows 系统只支持 name 和 user 两个参数
sensor[device,sensor,<mode>]			
读取硬件传感器		**device** – 设备名称 **sensor** – 传感器名称 **mode** – 可选值: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim). On Linux 2.4, 读取 /proc/sys/dev/sensors.	示例 key: sensor[w83781d-i2c-0-2d,temp1] Prior to Zabbix 1.8.4, the sensor[temp1] format was used. On Linux 2.6+, 读取/sys/class/hwmon. On OpenBSD, 读取 hw.sensors MIB. 示例 keys: sensor[cpuo,tempo] – CPUo 的温度 sensor[cpu[o-2]$,temp,avg] – cpu 平均温度 Zabbix 1.8.4 开始支持 OpenBSD
system.boottime			

系统启动的时间戳	整数		unix 时间戳
system.cpu.intr			
设备中断	整数		
system.cpu.load[<cpu>,<mode>]			
CPU 负载	浮点数	**cpu** - 可用值：all（默认），percpu（所有在线 cpu 的负载） **mode** - 可用值：avg1（1 分钟 默认值），avg5（5 分钟平均），avg15（15 分钟平均值）	范例 key: system.cpu.load[,avg5] 老命令方式：system.cpu.loadX 参数 percpu is Zabbix 2.0.0 开始支持
system.cpu.num[<type>]			
CPU 数量	处理器个数	**type** - 可用值：online（默认值），max	范例: system.cpu.num
system.cpu.switches			
上下文交换	交换次数		老命名方式：system[switches]
system.cpu.util[<cpu>,<type>,<mode>]			
CPU 利用率	百分比	**cpu** - cpu 数量（默认是所有 cpu） **type** - 可用值：idle，nice，user（默认），system（windows 系统默认值），iowait，interrupt，softirq，steal **mode** - 可用值：avg1（一分钟平均，默认值），avg5（5 分钟平均），avg15（15 分钟平均值）	范例 key: system.cpu.util[0,user,avg5] 老命名方式: system.cpu.idleX，system.cpu.niceX，system.cpu.systemX，system.cpu.userX
system.hostname[<type>]			
返回主机名	字符串	**type**（仅用于 windows 系统） - 可用值：netbios（默认）or	例如: on Linux: system.hostname → linux-w7x1

		host	system.hostname → www.zabbix.com on Windows: system.hostname → WIN-SERV2008-I6 system.hostname[host] → Win-Serv2008-I6LonG type 参数从 zabbix 1.8.6 开始支持
system.hw.chassis[<info>]			
返回机架信息	字符串	**info** - full（默认），model, serial, type 或 vendor	例如: system.hw.chassis[full] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop] 需要 root 权限，因为这些信息是从内存中读取的。 Zabbix agent version 2.0 开始支持
system.hw.cpu[<cpu>,<info>]			
返回 CPU 信息	字符/数字	**cpu** - cpu 数量或者 all（默认） **info** - **full**（默认），curfreq, maxfreq, model 或者 vendor	例如: system.hw.cpu[0,vendor] AuthenticAMD 从 /proc/cpuinfo 、/sys/devices/system/cpu/[cpunum]/cpufreq/cpuinfo_max_freq 获取信息. 如果指定了 CPU 数量和 curfreq 或者 maxfreq, 将会返回数值 (Hz). Zabbix agent 2.0 开始支持
system.hw.devices[<type>]			
列出 PCI 或者 USB	文本值	**type** - pci (默认) or usb	范例: system.hw.devices[pci] 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge [..] 返回 lspci 或者 lsusb (不带参数) Zabbix agent 2.0 开始支持
system.hw.macaddr[<interface>,<format>]			
列出 MAC 地址	字符串	**interface** - all（默认）或者正则表达式	范例: system.hw.macaddr[“etho$”,full]

		format – full (默认) 、short	[etho] 00:11:22:33:44:55 列出指定接口 mac 地址 如果 format 指定为 short，MAC 地址相同的将会被忽略掉 Zabbix agent 2.0 开始支持
system.localtime[<type>]			
系统时间	数字或者字符串	**utc** – (默认) (00:00:00 UTC, January 1, 1970) **local** – 本地时间格式 'yyyy-mm-dd,hh:mm:ss.nnn, +hh:mm'	zabbix 2.0 开始支持带参数
system.run[command,<mode>]			
在制定的主机上运行命令	文本	**command** – 命令 **mode** – wait (默认值, 执行超时时间), nowait (不等待) 最大可用返回 512KB 数据，包含空白数据。 命令输出数据必须是文本	例如: system.run[ls -l /] – 列出/的文件和目录. 备注: 启用这个方法, agent 配置文件必须配置 EnableRemoteCommands=1 选项
system.stat[resource,<type>]			
虚拟内存状态	数字	ent - number of processor units this partition is entitled to receive (float) kthr,<type> - information about kernel thread states: r - average number of runnable kernel threads (float) b - average number of kernel threads placed in the Virtual Memory Manager wait queue (float) memory,<type> - information about the usage of virtual and real memory: avm - active virtual pages (integer) fre - size of the free list (integer) page,<type> - information about page faults and paging activity: fi - file page-ins per second (float) fo - file page-outs per second (float) pi - pages paged in from paging space (float)	

		po - pages paged out to paging space (float) fr - pages freed (page replacement) (float) sr - pages scanned by page-replacement algorithm (float) faults,<type> - trap and interrupt rate: in - device interrupts (float) sy - system calls (float) cs - kernel thread context switches (float) cpu,<type> - breakdown of percentage usage of processor time: us - user time (float) sy - system time (float) id - idle time (float) wa - idle time during which the system had outstanding disk/NFS I/O request(s) (float) pc - number of physical processors consumed (float) ec - the percentage of entitled capacity consumed (float) lbusy - indicates the percentage of logical processor(s) utilization that occurred while executing at the user and system level (float) app - indicates the available physical processors in the shared pool (float) disk,<type> - disk statistics: bps - indicates the amount of data transferred (read or written) to the drive in bytes per second (integer) tps - indicates the number of transfers per second that were issued to the physical disk/tape (float) This item is supported starting from version 1.8.1.	
system.sw.arch			
返回软件信息	字符串		范例: system.sw.arch i686 数据来自 uname 方法 Zabbix agent 2.0.开始支持
system.sw.os[<info>]			

返回系统信息	字符串	**info** – full (default), short ,name	范例: system.sw.os[short] Ubuntu 2.6.35-28.50-generic 2.6.35.11 信息来自如下文件 [full] - /proc/version [short] - /proc/version_signature [name] - /etc/issue.net Zabbix agent version 2.0.开始支持
system.sw.packages[<package>,<manager>,<format>]			
已安装软件列表	文本值	**package** - all (默认)或者正则表达式 **manager** - all (默认) or a package manager **format** - full (默认) , short	范例: system.sw.packages[mini,dpkg,short] python-minimal, python2.6-minimal, ubuntu-minimal Lists (alphabetically) installed packages whose names match the given package regexp (“all” lists them all). 包管理: manager (执行命令) dpkg (dpkg -get-selections) pkgtool (ls /var/log/packages) rpm (rpm -qa) pacman (pacman -Q) 如果 format 为 full, packages are grouped by package managers (each manager on a seperate line beginning with it’s name in square brackets). 如果 format 为 short, 包不分组，并且都列在一行上. Zabbix agent 2.0 开始支持
system.swap.in[<device>,<type>]			
交换分区	数字	**device** - 交换分区设备 (默	示例 key:

IN（磁盘交换到内存）		认 all) **type** – 可选值：count (swapins 数量), sectors(sectors swapped in), pages (pages swapped in).	system.swap.in[,pages] 数据采集自：Linux 2.4: /proc/swaps, /proc/partitions, /proc/stat Linux 2.6: /proc/swaps, /proc/diskstats, /proc/vmstat
system.swap.out[<device>,<type>]			
Swap out (f 内存到磁盘)	数字	**device** – swap 设备（默认 all) **type** – possible values: count (number of swapouts), sectors(sectors swapped out), pages (pages swapped out). See supported by platformfor details on defaults.	示例 key: system.swap.out[,pages] 数据采集自: Linux 2.4: /proc/swaps, /proc/partitions, /proc/stat Linux 2.6: /proc/swaps, /proc/diskstats, /proc/vmstat
system.swap.size[<device>,<type>]			
交换分区大小	字节或者百分比	**device** – 交换分区（默认值 all) **type** – possible values: free (free swap space, default), pfree (free swap space, in percent), pused (used swap space, in percent), total (total swap space), used (used swap space)	示例 key: system.swap.size[,pfree] – 空闲 swap 百分比 老命名格式: system.swap.free, system.swap.total
system.uname			
返回主机相信信息	字符串		示例值: FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 Since Zabbix 2.2.0, the value for this item is obtained by using the uname() system call,

			whereas previously it was obtained by invoking “uname -a” on Unix systems. Hence, the value of this item might differ from the output of “uname -a” and does not include additional information that “uname -a”
system.uptime			
系统运行时长(秒)	多少秒		使用 s/uptime 来获取
system.users.num			
登陆用户数量	多少用户		agent 使用 who 命令获取
vfs.dev.read[<device>,<type>,<mode>]			
磁盘读取状态	整数（如果 type 为如下）：sectors,operations,bytes 浮点数（如果 type 为如下）：sps, ops,bps	**device** – 磁盘设备 (默认值“all”1) **type** – 可选值：sectors, operations, bytes, sps, ops, bps(必须指定，不同操作系统下不同). sps, ops, bps stand for: sectors, operations, bytes per second, respectively mode – 可选值：avg1 (一分平均，默认值), avg5(五分内平均), avg15 (15 分内平均值). 备注：只有 type 为 sps, ops, bps 的时候，第三个参数才被支持。 不同操作系统的 TYPE 参数： FreeBSD – bps Linux – sps OpenBSD – operations	示例 key: vfs.dev.read[,operations] 老命名方式: io[*] Usage of the type parameters ops, bps and sps on supported platforms used to be limited to 8 devices (7 individual devices and one “all”). Starting with Zabbix 2.0.1 this limit has been increased to 1024 (1023 individual devices and one for “all”). Zabbix 1.8.6 开始支持 LVM Until Zabbix 1.8.6, only relative device names may be used (for example, sda), since 1.8.6 an optional /dev/ prefix may be used (for example, /dev/sda)

		Solaris - bytes	
vfs.dev.write[<device>,<type>,<mode>]			
磁盘写入状态	整数（如下类型）: sectors,operations,bytes 浮点型（如下类型）: sps, ops,bps	**device** - 磁盘设备 (默认值“all”1) **type** - sectors, operations, bytes, sps, ops, bps (must specify exactly which parameter to use, since defaults are different under various OSes). sps, ops, bps means: sectors, operations, bytes per second respectively **mode** - one of avg1 (default),avg5 (average within 5 minutes), avg15. Note: The third parameter is supported only if the type is in: sps, ops, bps.	Default values of 'type' parameter for different OSes: FreeBSD – bps Linux – sps OpenBSD – operations Solaris – bytes Example: vfs.dev.write[,operations] Old naming: io[*] The type parameters ops, bps and sps on supported platforms used to be limited to 8 devices (7 individual devices and one "all"). Starting with Zabbix 2.0.1 this limit has been increased to 1024 (1023 individual devices and one for "all"). Supports LVM since Zabbix 1.8.6. Until Zabbix 1.8.6, only relative device names may be used (for example, sda), since 1.8.6 optional /dev/ prefix may be used (for example, /dev/sda)
vfs.file.cksum[file]			
计算文件校验	File checksum, calculated by algorithm used by UNIX cksum	**file** - 文件完整路径	例如， 返回值: 1938292000 例如: vfs.file.cksum[/etc/passwd] 老命名规范: cksum
vfs.file.contents[file,<encoding>]			
获取文本内容	Contents of a file or empty string if it is empty or it contains only LF/CR characters	**file** - 文件完整路径	例如: vfs.file.contents[/etc/passwd] 文件不可以超过 64KB. Zabbix agent 2.0 开始支持
vfs.file.exists[file]			

检测文件是否存在	1 - 文件/硬链接/软连接文件存在 0 - 不存在	**file** - 文件完整路径	例如: vfs.file.exists[/tmp/application.pid]
vfs.file.md5sum[file]			
文件 MD5 校验码	文件 MD5 哈希值	**file** - 完整路径	示例返回值: b5052decb577e0fffd622d6ddc017e82 示例: vfs.file.md5sum[/usr/local/etc/zabbix_agentd.conf] 1.8.6 移除了 64MB 文件大小的限制
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]			
文件中搜索字符串	包含字符串的行，或者为空	**file** - 文件完整路径 **regexp** - GNU 正则表达 **encoding** - 编码 **start line** - 从哪一行开始，默认第一行 **end line** - 从哪一行结束，默认最后一行 **output** - an optional output formatting template. The \0 escape sequence is replaced with the matched text while an \N(where N=1 … 9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups). 如果 <output> 为空， 所有包含搜索字符串的行都会被返回 start line 和 end line 从 zabbix	仅返回首先匹配的行 例如: vfs.file.regexp[/etc/passwd,zabbix] vfs.file.regexp[/path/to/some/file, ”([0-9]+)$”,,3,5,\1] vfs.file.regexp[/etc/passwd,^zabbix:.:([0-9]+),,,,\1] - getting the ID of user zabbix

		2.0 开始支持	
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]			
文件中搜索字符串	0 - 为找到 1 - 找到	**file** - 文件完整路径 **regexp** - GNU 正则表达式 **encoding** - 编码 **start line** - 哪行开始，默认第一行 **end line** - 哪行结束，默认最后一行 上面两个参数从 2.0 版本开始支持。	例如: vfs.file.regmatch[/var/log/app.log,error]
vfs.file.size[file]			
文件大小	字节	**file** - 完整路径	zabbix 必须有可读此文件的权限 例如: vfs.file.size[/var/log/syslog]
vfs.fs.discovery			
列出挂载的文件系统	Used for low-level discovery	JSON 对象	从 zabbix agent 2.0 开始支持
vfs.fs.inode[fs,<mode>]			
inodes 数量	数字	**fs** - 文件系统 **mode** - total (默认), free, used, pfree (空闲百分比), pused (使用百分比)	例如: vfs.fs.inode[/,pfree] 老命名规则: vfs.fs.inode.free[*], vfs.fs.inode.pfree[*], vfs.fs.inode.total[*]
vfs.fs.size[fs,<mode>]			
磁盘空间	字节	**fs** - 文件系统 **mode** - total (默认), free, used, pfree (空闲百分比), pused (使用百分比)	返回本地文件系统的使用量. 例如: vfs.fs.size[/tmp,free] 老命名规则: vfs.fs.free[*], vfs.fs.total[*], vfs.fs.used[*], vfs.fs.pfree[*], vfs.fs.pused[*],* 是任意挂载点

vm.memory.size[<mode>]			
内存大小	字节或者百分比	**mode** – total (默认), active, anon, buffers, cached, exec, file, free, inactive, pinned, shared, wired, used, pused, available, available 其中挑一个	老命名规则：vm.memory.buffers, vm.memory.cached, vm.memory.free, vm.memory.shared, vm.memory.total 监控项 vm.memory.size[] 允许三种类型的参数 第一类：包含 total – 总内存 第二类： 系统指定内存类型:active, anon, buffers, cached, exec, file, free, inactive,pinned, shared, wired. 第三类：用户级别，一共使用了多少内存，还有多少内存可用：used, pused, available,pavailable.
web.page.get[host,<path>,<port>]			
获取网页内容	网页源代码	**host** – 主机名/域名 **path** – 文件地址，默认/ **port** – 端口，默认 80	返回空字符串表示失败. 例如: web.page.get[www.ttlsa.com,/,80]
web.page.perf[host,<path>,<port>]			
获取完全加载网页消耗的时长	秒	**host** – 主机名/域名 **path** – html 地址，默认是/ **port** – 端口,默认 80	返回 0 表示失败. 例如： web.page.perf[www.ttlsa.com,/,80]
web.page.regexp[host,<path>,<port>,<regexp>,<length>,<output>]			
在网页中搜索字符串	The matched string, or as specified by the optional<output>parameter. An empty string if no match was found.	**host** – 主机名 **path** – html 文件路径 (默认值 /) **port** – 端口 (默认 80) **regexp** – GNU 正则表达式 **length** – 返回的最大的字符串数量	失败则返回空字符 (不匹配). 示例: web.page.regexp[www.zabbix.com,index.php, 80,OK,2]

		output – 输出格式模板可选项. The \o escape sequence is replaced with the matched text while an \N(where N=1…9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups). If <output> is left empty – the whole line containing the matched text is returned. The output parameter is supported from version 2.2.	
vfs.file.time[file,<mode>]			
文件时间	Unix 时间戳.	**file** – 文件路径 **mode** - one of modify (default, modification time), access - last access time, change - last change time	示例: vfs.file.time[/etc/passwd,modify] The file size limit depends on large file support.

zabbix External checks 外部命令检测

1. 概述

zabbix server 运行脚本或者二进制文件来执行外部检测，外部检测不需要在被监控端运行任何 agentd

item key 语法如下:

参数	定义
script	shell 脚本或者二进制文件名.
parameter(s)	Optional command line parameters.

如果不想传递任何参数，如下：

script[] 或者 script

zabbix server 会再定义好的目录里面执行外部脚本（zabbix server 中的配置 extenalScripts），这个脚本将使用 zabbix 的运行用户身份运行。请注意权限以及只有指定目录中的命令才能够被执行。

zabbix 脚本使用标准方式输出（完整输出但是会删除末尾的空白）,标准错误和退出代码将会被丢弃

备注：请不要过度使用那个外部检测，这会严重降低 zabbix 系统性能

2. 范例

执行带参数脚本 check_oracle.sh “-h <host IP address>”.

```
check_oracle.sh[“-h”,”{HOST.CONN}”]
```

zabbix 将会执行:

```
check_oracle.sh "-h" "192.168.1.4"
```

zabbix Simple checks 基本检测

1. 开始

Simple checks 通常用来检查远程未安装代理或者客户端的服务。使用 simple checks，被监控客户端无需安装 zabbix agent 客户端，zabbix server 直接使用 simple checks 来收据数据，一基本上都是用来检测远程服务器某端口是否在监听。如下为 simple checks:

```
net.tcp.service[ftp,,155]
net.tcp.service[http]
net.tcp.service.perf[http,,8080]
```

使用基本检测去监控 vmware 服务器，需要填写 User 和 Password 字段，其他服务器可以忽略

2. 基本检测

以下列表为 zabbx 支持的基本检测

Key			
描述	返回值	参数	说明
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]			
检测是否支持 icmpping	0 - ICMP ping 失败 1 - ICMP ping 成功	**target** - 主机 IP 或者域名 **packets** - 包数量 **interval** - 连续的数据包之间的时间间隔，以毫秒为单位 **size** - 包大小，以字节为单位 **timeout** - 超时时间,以毫秒为单位	例如: icmpping[,4] - 4 个包只要一个有返回，那么将会返回 1
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]			
返回百分比	丢包率	**target** - 目标 IP 或者域名 **packets** - 包个数 **interval** - 连续包之间的时间间隔,以毫秒为单位	

		size – 包大小，字节为单位 **timeout** – 超市时间，毫秒为单位	
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]			
返回 ICMP 响应时间	秒	**target** – 主机 IP 或者域名 **packets** – 包个数 **interval** – 包得响应时间,毫秒为单位 **size** – 包大小,字节为单位 **timeout** – 超时时间,毫秒为单位 **mode** – min, max, avg (默认值)	如果主机不可用，比如挂掉了，那么会返回 0
net.tcp.service[service,<ip>,<port>]			
检测服务是否运行并且接受 tcp 连接	0 – 服务未开启 1 – 服务运行中	**service** – ssh, ntp, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet 中的一个 **ip** – zabbix 中定义好的 ip **port** – 端口(标准服务端口).	范例: net.tcp.service[ftp,,45] 检测运行在端口号 45 的 FTP 服务是否可用.加密协议 imap993 端口和 pop995 端口目前不支持 zabbix 2.0 开始支持 https 和 telnet
net.tcp.service.perf[service,<ip>,<port>]			
检测服务器性能	0 – 服务停止 sec – 连接到服务器的时间（秒）	**service** – ssh, ntp, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet 之一 **ip** – zabbix 中定义的 IP **port** – 端口（标准端口号）	例如: net.tcp.service.perf[ssh] 可以检测连接到 ssh 所消耗的时间.加密协议服务不支持，例如 IMAP. zabbix 2.0 开始支持 https 和 telnet

超时处理

如果基本检测时间超过了 zabbix 配置文件设置的超市时间,zabbix 将不会做处理.

3. ICMP pings

zabbix 使用外部命令 fling 处理 ICMP ping. fping 不包含在 zabbix 的发行版本中，你需要额外去下载安装,如果你配置了 epel 源，如下命令安装

```
# yum install fping
```

然后修改 zabbix_server.conf,指定 fping 路径，配置参数为 FpingLocation，一般为

```
FpingLocation = /usr/bin/fping
```

如果没有安装 fping 或者 fping 路径指定出错，cmpping，icmppingloss,icmppingsec 都不会处理，执行 fping 命令的 zabbix 用户要设置 setuid，毕竟 fping 命令是需要 root 权限的，如下设置：

```
# chown root:zabbix /usr/sbin/fping
# chmod 4710 /usr/sbin/fping
```

ICMP 检测默认参数:

参数	值	描述	fping flag	Min	Max
packets	3	包数量	-C	1	10000
interval	1000	毫秒，“fping” 默认	-p	20	
size	56 or 68	字节，“fping” 默认; x86 使用 56 字节, x86_64 使用 68 字节	-b	24	65507
timeout	500	毫秒，“fping” 默认	-t	50	

zabbix ODBC 数据库监控

ODBC 是 C 语言开发的、用于访问数据库的中间件接口.zabbix 支持查询任何 ODBC 支持的数据库.zabbix 通过调用 ODBC 来获取数据库的数据以及数据库状态等等信息.

1. 安装 unixODBC

官方主页:http://www.unixodbc.org/download.html.安装方法如下

RedHat/Fedora/Cetnos

```
shell> yum -y install unixODBC unixODBC-devel
```

SUSE zypper

```
# zypper in unixODBC-devel
```

2. 安装 unixODBC 驱动

要监控 mysql 等数据库必须先安装基于 c 开发的 unixODBC 数据库驱动.redhat 直接 yum 安装,suse 使用 zypper 安装.其他系统,源码安装,官方地址：http://www.unixodbc.org/drivers.html.

redhat/centos

```
shell> yum install mysql-connector-odbc
```

SUSE

```
zypper in MyODBC-unixODBC
```

3. 配置 unixODBC

配置 odbcinst.ini 和 odbc.ini 两个配置文件即可,验证配置是否正常,如下命令:

```
# odbcinst -j
unixODBC 2.2.14DRIVERS.....: /etc/odbcinst.iniSYSTEM DATA SOURCES: /etc/odbc.iniFILE DATA SOURCES..:
/etc/ODBCDataSourcesUSER DATA SOURCES..: /root/.odbc.iniSQLULEN Size.....: 8SQLLEN Size.....:
8SQLSETPOSIROW Size.: 8
```

odbcinst.ini 范例:

```
# vi /etc/odbcinst.ini
[mysql]
Description = ODBC for MySQL
Driver      = /usr/lib/libmyodbc5.so
```

参数详解:

属性	描述
mysql	数据库驱动名称.
Description	数据库驱动描述.
Driver	数据库驱动类库具体路径

odbc.ini 范例:

```
[test]
Description = MySQL test database
Driver      = mysql
Server      = 127.0.0.1
User        = root
Password    =
Port        = 3306
```

```
Database        = zabbix
```

参数详解:

属性	描述
test	数据源名称 (DSN).
Description	数据源描述.
Driver	数据库驱动名 – 在 odbcinst.ini 指定
Server	数据库 IP/DNS.
User	数据库用户名.
Password	数据库密码.
Port	数据库端口
Database	数据库名称.

验证 ODBC 是否可用,使用 isql（命令 unixODBC 包提供）命令,如下：

```
# isql test
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

命令解释

参数	描述
help	列出所有表
help tablename	查询表所有数据,类似 select * from tablename
quit	退出

4. 编译支持 ODBC 的 zabbix

增加 ODBC 的支持,需要增加如下参数.起先我没有加这个参数，我需要重新编译一次

```
--with-unixodbc[=ARG]    use odbc driver against unixODBC package
```

5. 配置监控项

配置数据库监控项:

属性	描述
Type	类型,选择监控数据库.
Key	监控项 key db.odbc.select[unique_description,data_source_name]
unique_description	描述，要唯一
data_source_name	odbc.ini 中定义的数据源名称
User name	数据库用户名 (可选,如果 odbc.ini 中已经定义)
Password	数据库密码 (可选,如果 odbc.ini 中已经定义)
SQL query	SQL 语句
Type of information	返回值类型,如果类型选错了,这个监控项会不可用

6. 注意事项

- 查询语句执行时间不能超过配置的超时时间
- 查询只允许返回一个值.
- 如果查询语句返回了多个列,它只读取第一列
- 如果查询语句返回了多行,它读取第一条
- SQL 语句必须是 select 开头,只能是查询语句.
- SQL 语句不能包含换行符

7. 错误消息

从 zabbix 2.08 开始 ODBC 提供如下详细的错误信息:

```
Cannot execute ODBC query:[SQL_ERROR]:[42601][7][ERROR: syntax error at or near ";"; Error while executing the query]|
————————————-  ———  ——  |  ————————————————————————-|
|                |        |    `- Native error code          `- error message.          `- Record separator
|                |        `-SQLState
`- Zabbix message  `- ODBC return code
```

错误消息最长不能超过 128 字节,因此错误消息太长会被截断.

zabbix history trends 历史与趋势数据详解

zabbix 会收集历史数据（所有的数据都成为过去，O(∩_∩)O 哈！），以及还会收集每小时的平均数据作为趋势数据，每小时才收集一次，所以 trends（趋势）暂用的资源很小。

1. 保留历史数据

我们可以通过如下方式来设置保留数据的时长:

- 监控项（item）配置里
- 匹配更新监控项（item）
- 设置 Housekeeper tasks

Housekeeper 会定期删除过期的数据。如果数据不是特别有意义，建议你把保留时间设置短一些。可能你会说，那我想看我以前的数据图怎么办？好说，老数据一般不用精确到分秒，只要小时的平均数据即可，这样的话，趋势数据保留久一点即可。例如保存历史记录 14 天，趋势数据 5 年（5 年，你孩子都从小学一年级到五年级了）

备注：如果 history 设置为 0（设置为 0 的人是不想看历史数据或者硬盘没空间么？），zabbix 只会获取 item 的值，然后用与触发器，然后就没有然后了，不会存到数据库的。

2. 保留趋势数据

你可以通过如下方式设置保留趋势数据的周期

- 监控项配置表单
- 批量更新监控项
- 设置 Housekeeper tasks

一般来说趋势数据都设置的时间都很长，但是时间超过了 Housekeeper 的老数据还是会被它删除。

备注：如果 trends 设置为 0，zabbix server 压根不会计算和存储趋势数据。

3. 注意事项

重启服务器会丢失数据导致这个小时的平均数据不准确。你的数据是什么类型的，那么趋势数据也是什么类型的。整数的数据特别要注意一个问题，如果你一共获取了两个值，其中一个 0，一个 1，那么他们的平均值是 0，而不是

0.5

zabbix 自定义用户 key 与参数 User parameters

为什么要自定义 KEY

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

语法如下:

```
UserParameter=key,command
```

用户自定义参数包含一个 key 和一个命令，key 必须整个系统唯一，配置好之后，重启客户端。然后配置 item,在 key 的位置填上我们自定义的 key 即可。用户自定义参数里指定的脚本由 zabbix agent 来执行，最大可以返回 512KB 的数据.

用户自定义 key 实例

- 简单点的命令示例：

```
UserParameter=ping,echo 1
```

如果调用 ping 这个 key，将会收到返回值 1.

- 更复杂的命令示例:

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

如果返回 1 表示 MySQL 运行中，如果返回 0 表示 MySQL 挂了

- 灵活的自定义 key：

如下为灵活的用户自定义参数

```
UserParameter=key[*],command
```

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

实例

- 示例 1

```
UserParameter=ping[*],echo $1
```

ping[0] – 将一直返回 0

ping[aaa] – 将一直返回 ‘aaa’

- 示例 2

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

如下参数用于监控 MYSQL，并且可以传递用户名和密码。

```
mysql.ping[zabbix,our_password]
```

- 示例 3

统计一个文件中有多少行被匹配?

```
UserParameter=wc[*],grep -c “$2″ $1
```

如下方法将会返回文件中出现指定字符的行数

```
wc[/etc/passwd,root]
wc[/etc/services,zabbix]
```

注意事项

1) 如果需要使用命令行里面出现$2 这种变量，那么你要使用两个$$2，例如 awk ’{ print $$2 }’，之前就遇到过这个问题，不停的测试自己脚本输出正常，但是 zabbix 却拿不到数据，原来是出在这里。为了防止和参数冲突，所以 zabbix 做了这个规定。
2) zabbix 禁止使用一些不安全的参数，如下：

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

3) 从 zabbix 2.0 开始，zabbix 返回文本数据可以是空格。

zabbix 值映射 Value mapping

1. 介绍

zabbix 为了显示更人性化的数据，在使用过程中，我们可以将获取到得数据映射为一个字符串。比如，我们写脚本监控 MySQL 是否在运行中， 一般返回 0 表示数据库挂了，1 表示数据库正常，还有各种各样的监控都是返回 0，1，2，3 这样的数据，监控页上显示的都是数据字，完全不知道是什么意思。 这个时候我们可以使用 zabbix 的值映射，例如这边的 MySQL 架空 0 映射为”离线”，1 映射为“在线”或者用于备份的监控,做如下映射:

- ‘F’ → ‘Full’
- ‘D’ → ‘Differential’
- ‘I’ → ‘Incremental’

在配置 zabbix item 表单中，我们可以看到如下：

点击 show value mappings

CONFIGURATION OF VALUE MAPPING

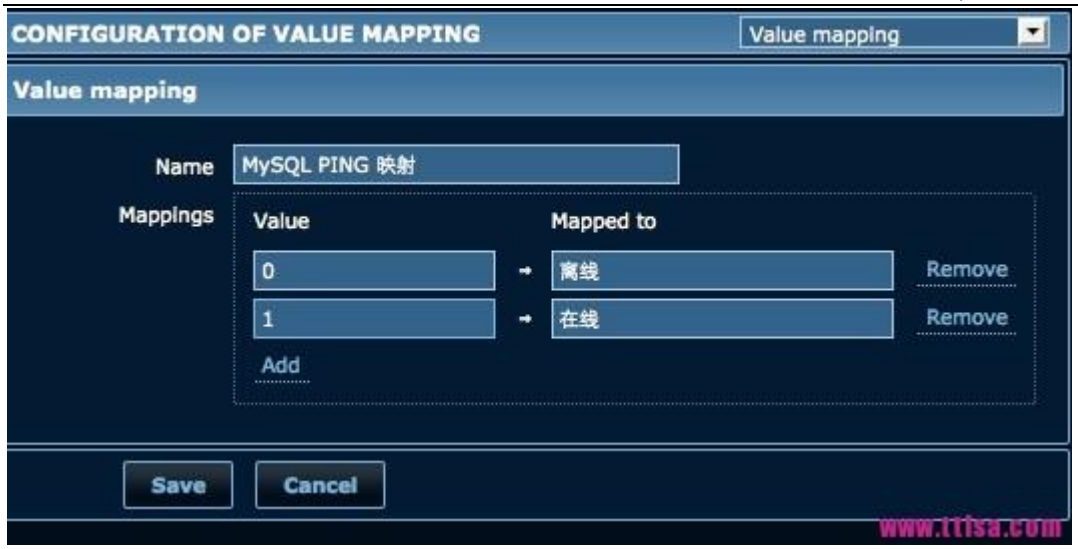
Value mapping

Name	Value map
APC Battery Replacement Status	1 → unknown 2 → notInstalled 3 → ok 4 → failed 5 → highTemperature 6 → replaceImmediately 7 → lowCapacity
APC Battery Status	1 → unknown 2 → batteryNormal 3 → batteryLow
Dell Open Manage System Status	1 → Other 2 → Unknown 3 → OK 4 → NonCritical 5 → Critical 6 → NonRecoverable
Host status	0 → Up 2 → Unreachable
HP Insight System Status	1 → Other 2 → OK 3 → Degraded

选择我们需要的映射即可。在 zabbix 2.2 之前只有数字（无符号）数据 item 才能使用值映射，在之后支持浮点数和字符类型。

如何定义值映射

点击 Administration （管理）→ General（常规），导航栏的右侧，下拉选择“Value mapping”，点击 Create value map (创建值映射)



参数说明:

参数	描述
Name	值映射名称，唯一.
Mapping	映射对.

zabbix Applications 使用介绍

介绍

Applications（我们翻译为应用程序）是 item 的一个组。例如我们要监控 MySQL，我们可以将所有和 MySQL 相关的 item 放到这个应用程序中。例如 MySQL 的 availability of MySQL, disk space, processor load, transactions per second, number of slow queries 都可以放到这个 Applications 里。

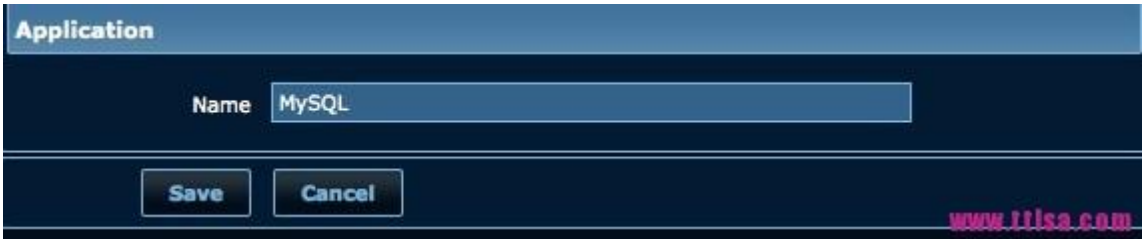
点击 Monitoring → Latest data，可以看到所有数据都是根据 application 来排列的，然后点击 application 前面的加号，可以看到应用程序中相关 items 的数据。一个 item 可以属于多个 application。如果你有这么做，你可以在多个 application 里面看到相关 item 的数据。

配置

先创建应用程序，然后将 item 连接到这个应用程序中即可。

创建应用程序

1) 点击 Configuration（配置） → Hosts（主机）或者 Templates（模板）
2) 点击需要创建应用程序的 Hosts 或者 Templates
3) 点击 Create application（创建应用程序）
4) 键入 application（应用程序）名称



在配置 item 的表单中也可以创建一个应用程序，你可以按住 shif 然后选择多个应用程序，那么在多个应用程序里面都可以见到当前 item 了。

zabbix 触发器 triggers

触发器介绍

触发器(triggers)是什么？触发器使用逻辑表达式来评估通过 item 获取到得数据是处于哪种状态，item 一收回数据，讲解任务交给触发器去评估状态，明白触发器是怎么一回事了把？

在触发器表达式中我们可以定义哪些值范围是合理，哪些是不合理的，如果出现不合理的值，触发器会把状态改为 PROBLEM。接下来就到了报警以及发邮件,这步在讲完触发器之后开始讲。别落下 ttlsa 的 zabbix 教程哦。

触发器状态

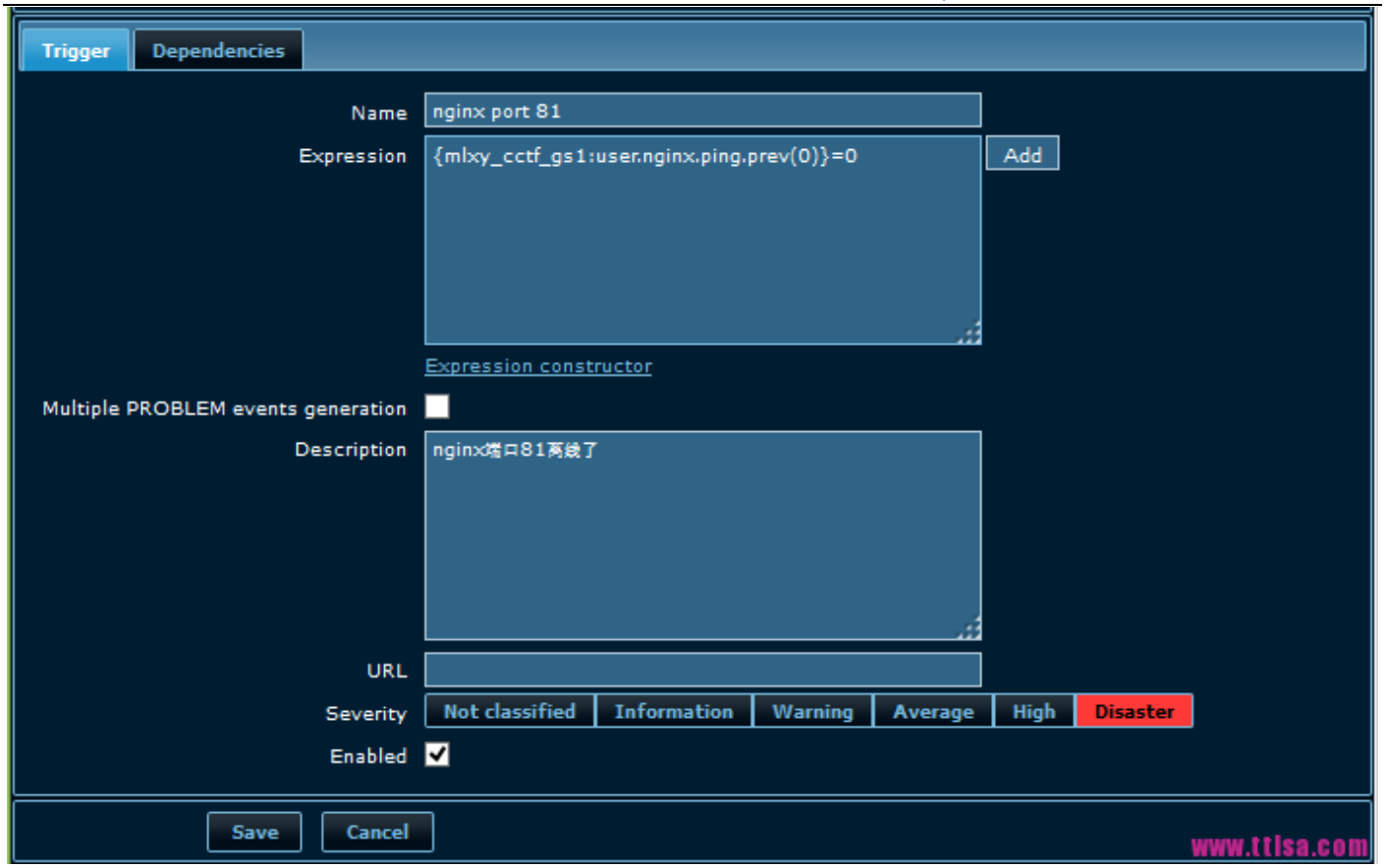
值	描述
OK	触发器的正常状态. 老版本的 zabbix 中叫做 FALSE
PROBLEM	非正常状态,例如数据库挂了,系统负载高了,都会是这个状态. 老版本 zabbix 中叫 TRUE

zabbix server item 每次获取到一个新值都会使用触发器表达式计算它的状态如果使用基于时间的表达式 (例如：nodata(), date(), dayofmonth(), dayofweek(), time(), now()), zabbix timer 每 30 秒会重新计算一次。

创建触发器

创建触发器步骤:

1) 点击 Configuration（配置） → Hosts（主机）
2) 点击 hosts（主机）相关行的 trigger
3) 点击右上角的创建触发器（create trigger），你也可以修改列表中的触发器
4) 在表单中输入相应的信息



参数介绍

参数	描述
Name	触发器名称 名称可以包含宏变量： {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {ITEM.VALUE}, {ITEM.LASTVALUE} and {$MACRO}. $1, $2…$9 可以被用来关联表达式的常量 示例： name：Processor load above $1 on {HOST.NAME}” 表达式：system.cpu.load[percpu,avg1].last(0)}>5 会显示为：Processor load above 5 on ttlsa 云服务器
Expression	计算触发器状态的逻辑表达式，这边设置为上一次值等于 0
Multiple PROBLEM events generation	通过设置该选项，你可以在触发器产生 problem 的时候触发一个事件
Description	触发器的描述,一般 name 写的不清楚，这边可以具体描述这个触发器的作用，例如 nginx 当前离

	线，请处理等等。Zabbix 2.2 版本开始，支持触发器名称。
URL	在 Monitoring → Triggers 中，可以看到 URL 并且可以点击，一般情况下他需要配合触发器 ID 来使用，在 url 中包含触发器 ID（宏变量 {TRIGGER.ID}），这样可以直接点击到具体触发器中。
Severity	设置严重性级别,上图我设置为“灾难”,你可以相应的设置警告、严重等状态的触发器
Enabled	当前触发器是否启用

Trigger severity

severity 通常用来定义当前 item 的一个状态的严重性。我们可以根据不同的严重性来定义不同的事件，例如报警，zabbix 自带如下严重性定义。

严重性	**定义**	**颜色**
Not classified	未知	灰色
Information	一般信息	浅绿
Warning	警告	黄色
Average	一般问题	橙色
High	严重问题	红色
Disaster	灾难,会带来损失的那种	深红

severities 用途

- 可视化显示，不同级别显示不同颜色，例如一般严重性为绿色
- 声音报警,不同的级别不同声音.
- 使用用户自定义媒体报警，例如严重问题发短信，其他问题发送邮件。
- 根据严重性来定义是否报警

可以自定义触发器严重性以及颜色，请参考：customise trigger severity names and colours.

zabbix 自定义触发器严重性

触发器严重性介绍

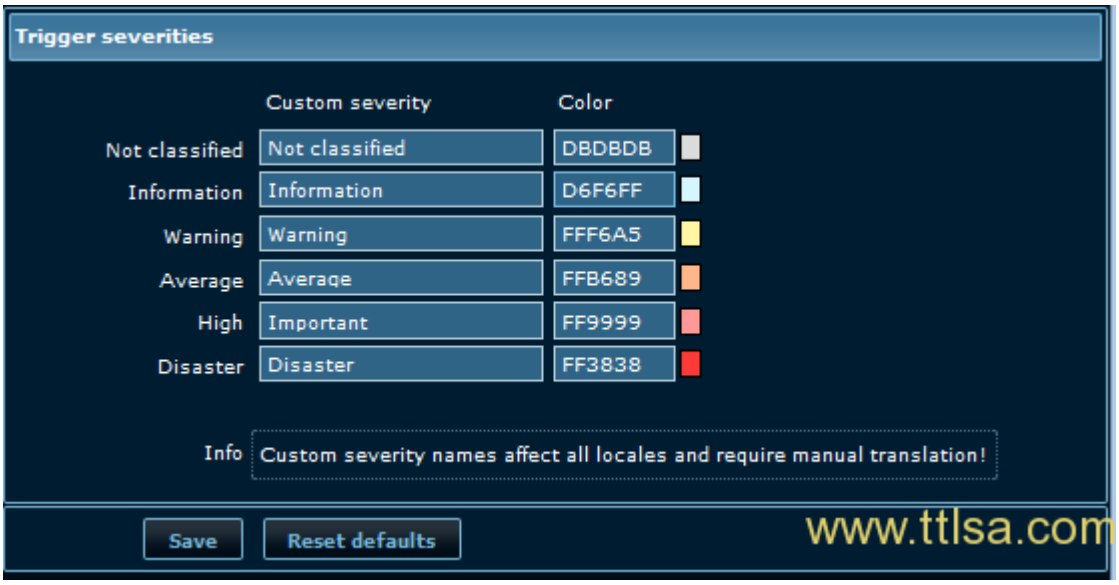
触发器严重性命名以及颜色定义都可以在 zabbix web 后台定义，点击 Administration（管理） → General （常规） → Trigger severities（触发器严重性）。这边定义好的颜色在每个不同主题/风格里面都是一样的。

所有系统默认的触发器名字在各国的语言包中都有翻译，但是你自定义的其他语言包不会给你翻译，因为在语言包里面没有这一个项目。那么怎么保证各国语言包里面都能相应的翻译自定义的严重性呢?

我们知道 zabbix 默认定义了 6 个触发器严重性，分别为：Not classified、Information、Warning、Average、High、Disaster，有些人觉得 High 不好理解或者觉得描述不满意，想改成 Important，请看如下操作：

设置触发器名称

点击 Administration（管理） → General （常规）→ Trigger severities（触发器严重性），将 High 改为 important，当然这里你也可以自定义你的颜色，我们这边就不再赘述了，修改完之后点击保存。



添加内容到 frontend.po

```
# /data/site/monitor.ttlsa.com/locale/en_US/LC_MESSAGES/frontend.po
msgid "Important"
msgstr "very Import"
```

备注：/data/site/monitor.ttlsa.com/是您 zabbix 站点根目录

创建.mo 文件

需要执行 locale 目录下的 make_mo.sh 文件，如果出现./make_mo.sh: line 4: msgfmt: command not found，那么请你先安装 msgfmt

```
# yum install gettext
```

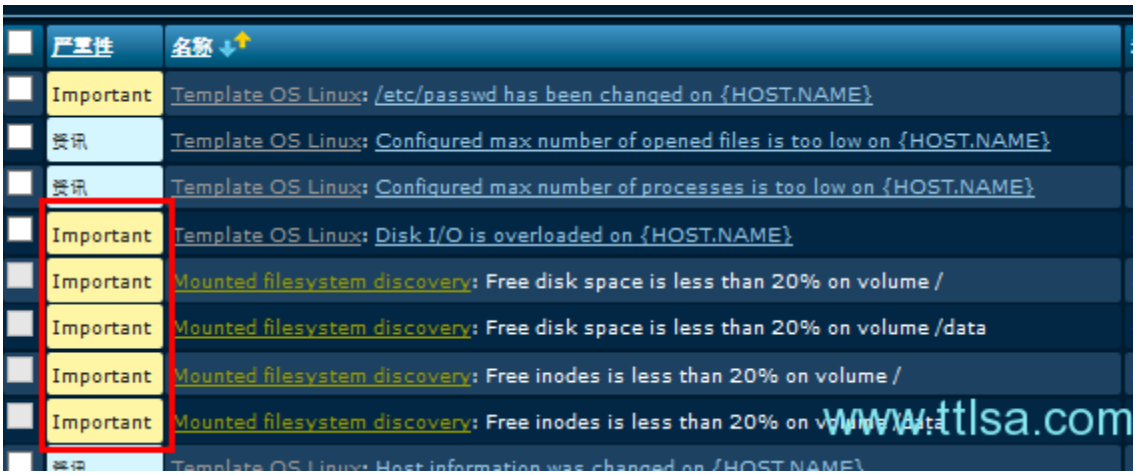
然后执行 make_mo.sh

```
# ./make_mo.sh
```

效果

随意选择一个 Host 的触发器列表，看左边信息，请看如下图

在自定义触发器名称之前应该显示 High 的，这边被我们修改成了 Very Import。zabbix 是一个多语言监控系统，如果你想切换到中文环境，那么你需要修改 zh_CN 下的 frontend.po，然后 make_mo.sh 创建 frontend.mo。否则将只会显示 Important，如下：



这边直接显示触发器的 MSGID，因为你没翻译。

结束语

6 个触发器级别，名称随你修改，但是大多数情况下我们都不需要修改，默认的挺好的，不是吗？非要自定义，我不拦你。

zabbix 触发器依赖关系详解

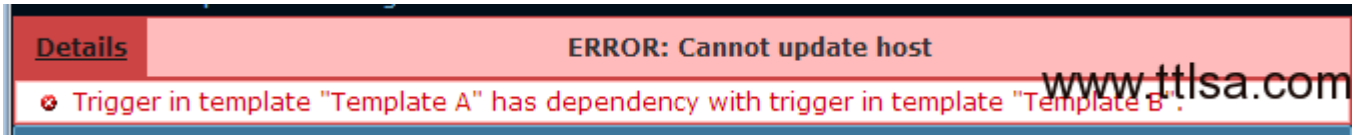
概述

zabbix 触发器可以设置依赖性，例如我配置了两个触发器，一个触发器定义 www.ttlsa.com 这个 HOST 是否在运行中，另一个是 www.ttlsa.com 的网络是否通畅。假如网络出现故障，但是 ttlsa 服务器并未出现故障，我们依旧会收到这两个触发器给到的故障通知。

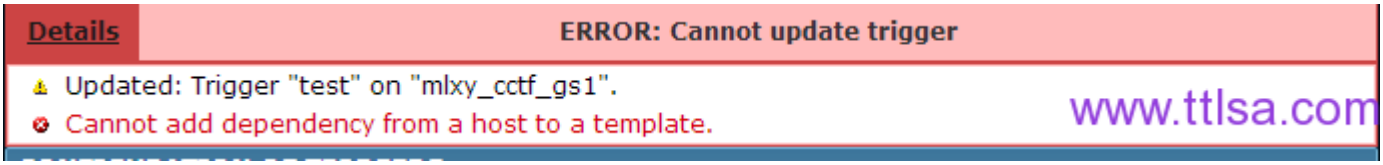
现在的问题很明显，HOST 是正常的，肯定不希望收到他的故障信息，因为它正常工作。所以在配置 HOST 触发器时，我们需要增加依赖关系，依赖网络是否畅通这个触发器。一旦网络出现故障，将不会出发 HOST 故障的通知。单个触发器可以依赖于多个触发器。

触发器依赖要点

- 一台 HOST 的触发器可以依赖其他 HOST 的触发器，但是注意不要有死循环依赖。比如 A 依赖 B，B 依赖 C，C 又依赖 A。一个圆圈，没完没了。如下是 A 依赖 B，B 又依赖 A，根本依赖不了，有如下报错。



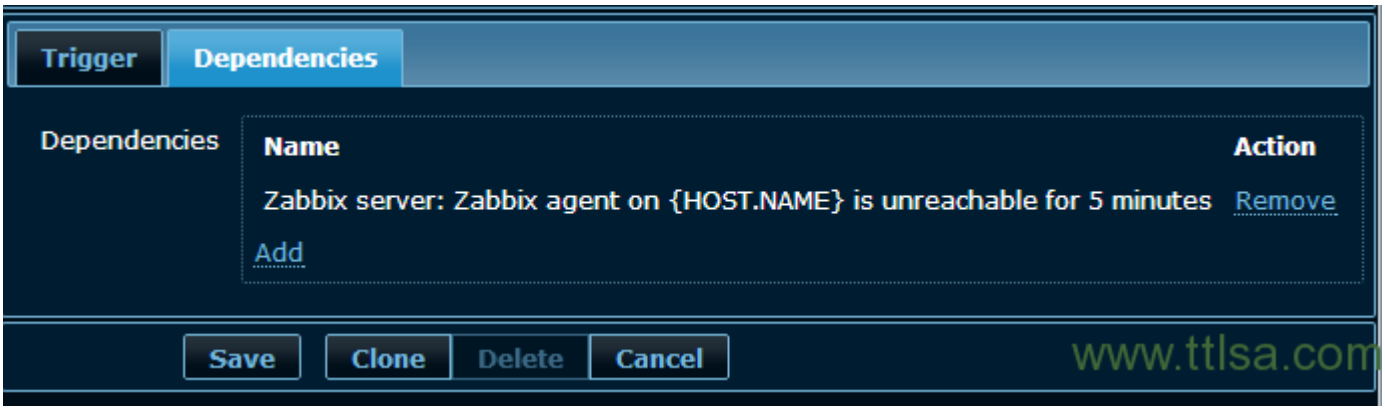
- 一个模板的触发器可以依赖另外一个模板的触发器，例如模板 A 触发器依赖模板 B 触发器。一个 HOST 要链接 A 模板，那么它同时要链接 A 模板（因为模板 A 中的触发器依赖了模板 B 中的触发器），但是 HOST 可以单独链接模板 B（B 是被依赖）。在一个 host 单独链接 template A，结果出现如下错误，所以别忘记了一起把 template B 也链接到 HOST 中。



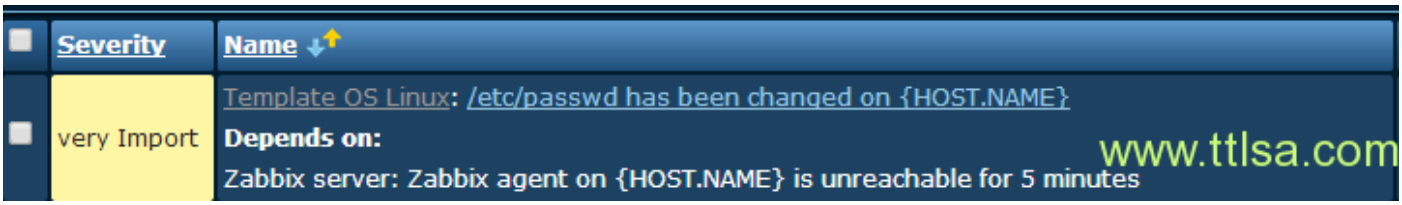
- 模板中的触发器可以依赖 HOST 中的触发器。如果某个 HOST 链接这类模板，那么 HOST 创建的相应的触发器也同样会依赖那个 HOST 的触发器。举个官方的例子，某个模板中的一些触发器依赖了 route/主机的触发器，凡事链接（理解为套用）了这个模板的机器都会依赖这些 router/主机。说了那么多，其实就是继承了。HOST 中的触发器不能依赖模板中的触发器。

配置

编辑触发器，选择选项卡“dependencies”，点击 Add，选择你需要依赖的触发器，如下图：



然后点击保存，可以看到触发器多了一个 depend on



多个依赖实例

借用官方文档的示例，Host 前面有个 Router2，Router2 前面有 Router1，如下：

```
Zabbix - Router1 - Router2 – Host
```

如果 Router1 挂了，很明显 Router2 和 Host 连不上，我们不希望收到关于连不上 Router2 和 HOST 的通知，因此，我们定义了如下依赖关系：

```
'Host is down' trigger depends on 'Router2 is down'

trigger'Router2 is down' trigger depends on 'Router1 is down' trigger
```

在触发器将 Host 的状态改变为’Host is down’之前，它会检查 host 相关的依赖，这时候如果发现它依赖的触发器只要出现一个 problem 状态，那么当前触发器状态不会变化，这样一来 action 不会执行，报警通知 sms/email 自然也不会发送了。

zabbix 会递归执行检测，如果 router1 或者 router2 有一个出现连不上，那么 Host 的触发器不会有任何的改变。

zabbix 单位符号 Unit symbols

概述

在 zabbix 里面，我们不需要使用大数字来，例如我们可以不使用 86400 来表示一天，这个数字又不容易理解也容易出错。用什么办法来解决大数字问题呢？我们可以使用单位来简化，例如简化 zabbix 触发器表达式或者 item key。所以，我们可以使用 1d 来渠道 86400，‘d’就是单位后缀，表示天。

时间单位后缀

- s - 秒(一般来说来说不写 s，就表示 s 了)
- m - 分
- h - 小时
- d - 天
- w - 周

时间单位后缀支持如下使用场景：

- 触发器表达式
- zabbix internal item 参数，如 zabbix[queue,<from>,<to>]
- last parameter of aggregate checks

单位符号

Zabbix server 和 zabbix 后台都支持使用这些单位显示，触发器表达式也可以使用这些单位。

- K – kilo
- M – mega
- G – giga
- T – tera

如果 item 值的单位不是 B、Bps，那么 1000 为基数，1000 将会别显示成 1k，2000 显示为 2k。如果单位是 B 或者 Bps，那么基数就是 1024 了，item 值为 1024 将会被展示成 1k。所以在单位上面，大家一定要切记。例如流量别忘记单位改为 B，访问量/次数不填写单位

除了以上单位，zabbix 还支持如下单位（单位很大，基本很少用到）

- P – peta
- E – exa
- Z – zetta
- Y – yotta

使用例子

在编写触发器表达式中使用这些单位会显得好理解并且更好维护，如下

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
 {host:system.uptime[].last(0)}<86400
 {host:system.cpu.load.avg(600)}<10
```

可以修改为:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
 {host:system.uptime.last(0)}<1d
 {host:system.cpu.load.avg(10m)}<10
```

可以看到 120 秒改成了 2m（分钟），86400 改为 1d,是不是变得好理解了。

zabbix 触发器表达式详解

概述

触发器中的表达式使用很灵活，我们可以创建一个复杂的逻辑测试监控，触发器表达式形式如下:

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

{主机：key.函数(参数)}<表达式>常数，具体的例子，请接着往下走，很简单

Functions 函数

触发器 functions 可以引用检索到的值，当前时间或者其他元素。触发器表达式支持的 function 完整列表请点击官网地址 supported functions

Function 参数

大多数数值 functions 可以使用秒来作为参数。你可以使用前缀“#”来表示它有不同的含义

函数	描述
sum(600)	600 秒内的总和
sum(#5)	最新 5 个值的和

last 函数使用不同的参数将会得到不同的值，#2 表示倒数第二新的数据。例入从老到最新值为 1,2,3,4,5,6,7,8,9,10，last(#2)得到的值为 9，last(#9)得到的值为 2。last 函数必须包含参数。

AVG，count，last，min 和 max 函数还支持额外的参数，以秒为单位的参数 time_shift（时间偏移量）。例如 avg(1h,1d)，那么将会获取到昨天的 1 小时内的平均数据。

备注：触发器表达式需要使用 history 历史数据来计算，如果 history 不可用（time_shift 时间偏移量参数无法使用），因此 history 记录一定要保留长久一点，至少要保留需要用的记录。

触发器表达式可以使用单位符号来替代大数字，例如 5m 替代 300，或者 1d 替代 86400,1k 替代 1024 字节等等。

操作符

优先级	操作	定义
1	/	除
2	*	乘
3	-	减
4	+	加
5	<	小于，用法如下: A<B ⇔ (A<=B-0.000001)
6	>	大于. 用法如下: A>B ⇔ (A>=B+0.000001)
7	#	不等于.用法如下: A#B ⇔ (A<=B-0.000001) | (A>=B+0.000001)
8	=	等于.T 用法如下: A=B ⇔ (A>B-0.000001) & (A<B+0.000001)
9	&	逻辑与
10	|	逻辑或

触发器示例

■ 示例一

触发器名称：Processor load is too high on www.zabbix.com

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

触发器说明：

参数	说明
www.zabbix.com	host 名称
system.cpu.load[all,avg1]	item 值,一分内 cpu 平均负载值
last(0)	最新值
>5	最新值大于 5

如上所示，www.zabbix.com 这个主机的监控项，最新的 CPU 负载值如果大于 5，那么表达式会返回 true，这样一来触发器状态就改变为“problem”了。

- 示例二

触发器名称：www.zabbix.com is overloaded

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

当前 cpu 负载大于 5 或者最近 10 分内的 cpu 负载大于 2，那么表达式将会返回 true.

- 示例三

触发器名称：/etc/passwd has been changed

使用函数 diff():

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

/etc/passwd 最新的 checksum 与上一次获取到的 checksum 不同，表达式将会返回 true. 我们可以使用同样的方法监控系统重要的配置文件,例如/etc/passwd,/etc/inetd.conf 等等。这些 zabbix 一般都会自带，没带的你自己加上吧。

- 示例四

触发器名称：Someone is downloading a large file from the Internet

使用函数 min:

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

当前主机网卡 etho 最后 5 分钟内接收到的流量超过 100KB 那么触发器表达式将会返回 true

- 示例五

触发器名称：Both nodes of clustered SMTP server are down

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

当 smtp1.zabbix.com 和 smtp2.zabbix.com 两台主机上的 SMTP 服务器都离线，表达式将会返回 true.

- 示例六

触发器名称：Zabbix agent needs to be upgraded

使用函数 str():

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

如果当前 zabbix agent 版本包含 beta8（假设当前版本为 1.0beta8），这个表达式会返回 true.

- 示例七

触发器名称：Server is unreachable

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

如上表达式表示最近 30 分钟 zabbix.zabbix.com 这个主机超过 5 次不可到达。

- 示例八

触发器名称：No heartbeats within last 3 minutes

使用函数 nodata():

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

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

- 示例九

触发器名称：CPU activity at night time

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
```

只有在凌晨 0 点到 6 点整，最近 5 分钟内 cpu 负载大于 2，表达式返回 true，触发器的状态变更为“problem”

■ 示例十

触发器名称：Check if client local time is in sync with Zabbix server time

使用函数 fuzzytime():

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

主机 MySQL_DB 当前服务器时间如果与 zabbix server 之间的时间相差 10 秒以上，表达式返回 true，触发器状态改变为“problem”

■ 示例十一

触发器名称：Comparing average load today with average load of the same time yesterday (使用 time_shift 时间偏移量参数).

```
{server:system.cpu.load.avg(1h)}/{server:system.cpu.load.avg(1h,1d)}>2
```

This expression will fire if the average load of the last hour tops the average load of the same hour yesterday more than two times.

Hysteresis（迟滞,滞后）

简单的说触发器状态转变为 problem 需要一个条件，从 problem 转变回来还需要一个条件才行。一般触发器只需要不满足触发器为 problem 条件即可恢复。明白了么？不明白就看例子吧。

有时候触发器需要使用不同的条件来表示不同的状态，举个官网很有趣的例子：机房温度正常稳定为 15-20°，当温度超过 20°，触发器值为 problem，当问题到了 15°与 20°之间，异常会解除。别整这些没用的，我们看实例.

为了达到这个效果，我们需要使用如下触发器表达式:

■ 示例 1

触发器名称：Temperature in server room is too high

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

如上有两个小括号，前面一个表示触发异常的条件，后面一个表达式表示解除异常的条件。

注意：宏变量 {TRIGGER.VALUE}将会返回当前触发器的值

■ 示例 2

触发器名称：Free disk space is too low

Problem: 最近 5 分钟剩余磁盘空间小于 10GB。（异常）

Recovery: 最近 10 分钟磁盘空间大于 40GB。（恢复）

简单说便是一旦剩余空间小于 10G 就触发异常，然后接下来剩余空间必须大于 40G 才能解除这个异常，就算你剩余空间达到了 39G（不在报警条件里）那也是没用的，有意思不！

```
({TRIGGER.VALUE}=0&{server:vfs.fs.size[/,free].max(5m)}<10G) |
 ({TRIGGER.VALUE}=1&{server:vfs.fs.size[/,free].min(10m)}<40G)
```

zabbix 事件通知

概述

我们前面花了大量时间去讲解 item、trigger、event 都是为发送报警做准备的，什么是事件通知呢？简单的说故障发生了，zabbix 会发邮件或者短信给你，告诉你服务器的一些状况。如果没有通知这块内容，你要盯着 zabbix 的事件才知道服务器发生什么状况了。这和保安有什么区别？

通知条件

发送通知，需要有如下步骤

- 定义一个通知介质

这个介质包含 email、shell 脚本、sms 短信网关（用得比较少）。zabbix 将需要发送的内容传递给这些介质，然后这些介质把事件放到对应的终端。例如通知内容发给 email，email 介质将内容发送到报警邮箱中。

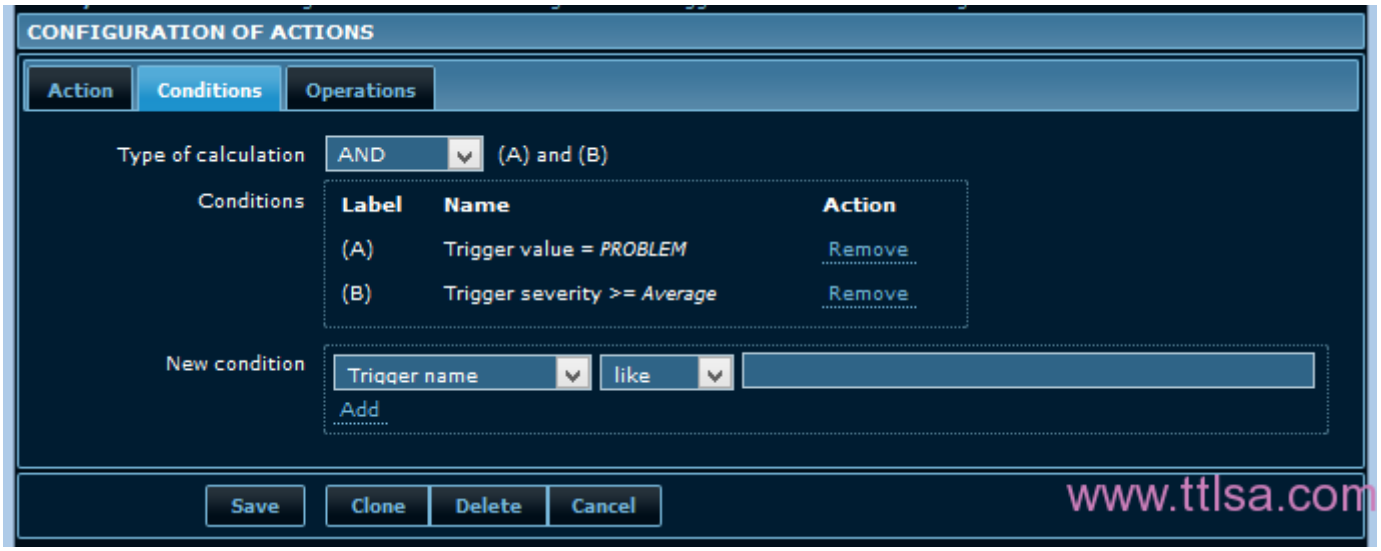
- 配置 action

Action 由 conditions（条件）和 operations（操作）组成。当满足指定的条件，然后执行操作。这就是一个 action。

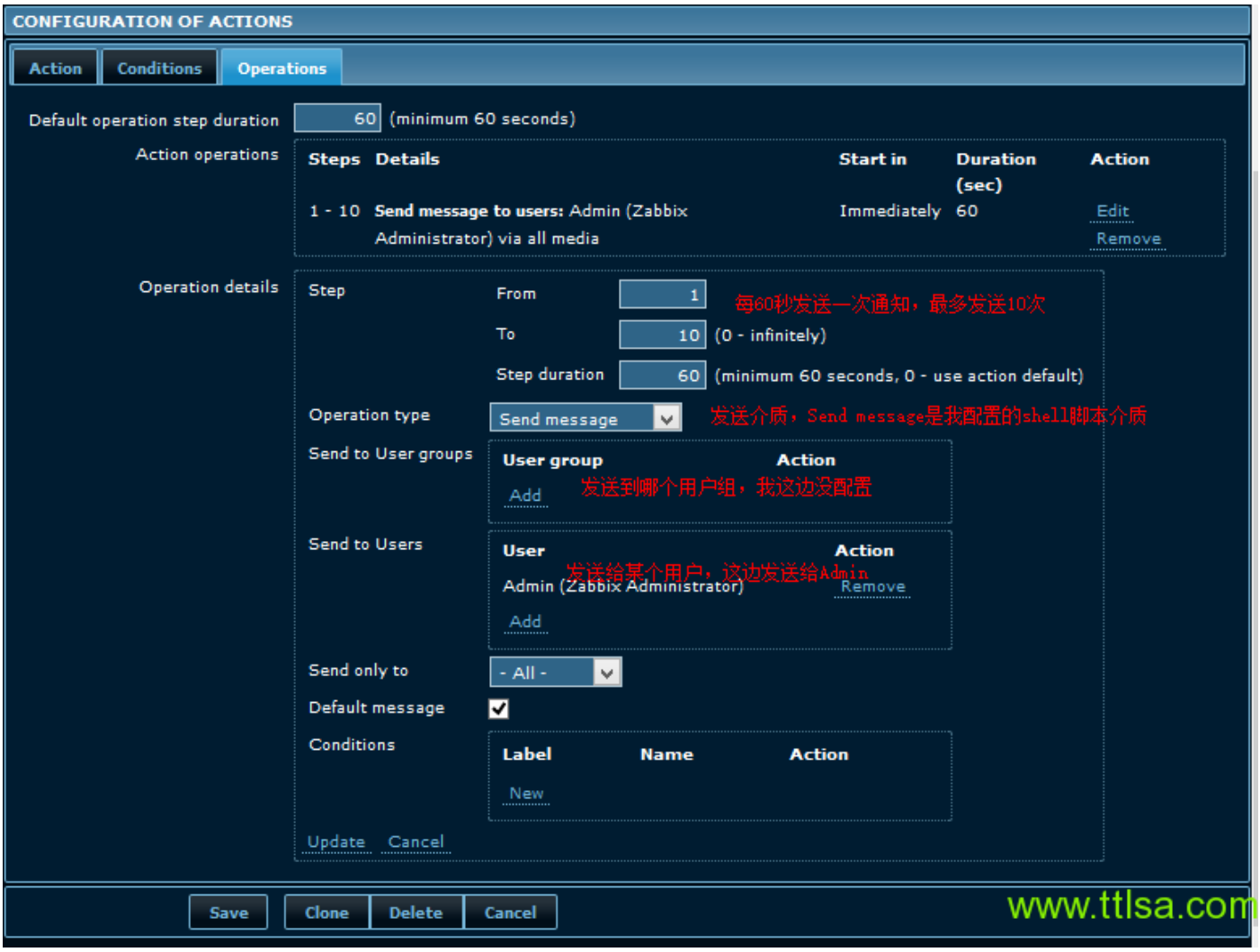
事件通知配置

这么讲大家可能有点不理解，我举个例子吧。

- 条件



- 操作



看图大家就明白了，满足两个条件之后，于是执行后面的操作，操作内容便是通过某个介质发送通知，间隔多久发送一次，发给谁，一共发送多少次。

Zabbix 事件来源

什么是 zabbix 事件

在 trigger 的文章内，我们已经有用到事件，这个事件要讲概念真心不知道怎么说，就拿 trigger 事件来说，如果 trigger 从当前值 ok 转变为 problem，那么我们称之为一个事件，这边便是 trigger 事件。

zabbix 事件有哪些

zabbix 事件一共有三种，分别为：触发器事件、发现事件、内部事件、自动注册事件，接下来我们一一讲解这四种事件类型。

zabbix 内部事件

- 监控项 item 状态从 normal 变为 unsupported，或者从 unsupported 变为 normal
- low-level 发现规则状态从 normal 变为 unsupported，或者从 unsupported 变为 normal
- 触发器状态从 normal 变为 unknown，或者从 unknown 转为 normal

有些人用中文语言，并且英语很过不去的，如上单词解释如下：

属性	描述
normal	正常
unsupported	不支持
unknown	未知

备注:内部事件从 zabbix 2.2 版本才开始引入，主要目的是为了可以通知管理员 zabbix 的某些 item 不可用

zabbix 发现事件

在我们配置完 zabbix 网络发现规则之后，zabbix 会定期按照这个规则去扫描 IP 段，一旦有发现主机和服务，便生成一个事件。事件如下列表：

事件	描述
Service Up	每当 zabbix 检测到 active service

Service Down	每当 zabbix 检测不到 service
Host Up	目标 IP，至少有一个 service 是 UP
Host Down	所有的 services 都没响应.
Service Discovered	service 在维护时间之后恢复或者第一次被发现
Service Lost	service 在 up 之后丢失了
Host Discovered	主机在维护时间之后恢复或者第一次被发现
Host Lost	主机在 up 之后丢失.

关于 zabbix 自动发现，我们后续有专门的文章来讲解。

zabbix 触发器事件

触发器是 zabbix 事件中最重要的事件，同时也是最常用到的事件（例如定义 action 等）。每当触发器状态发生变化，都会生成一个包含详细状态信息的触发器事件：例如事件什么时候发送的以及当前状态是什么。

zabbix 自动注册事件

active agent 主动与 server 通信，zabbix server 使用 agent 通信的 ip 地址与端口来添加主机，并且生成一个自动注册事件。关于自动注册我们后续来讲。

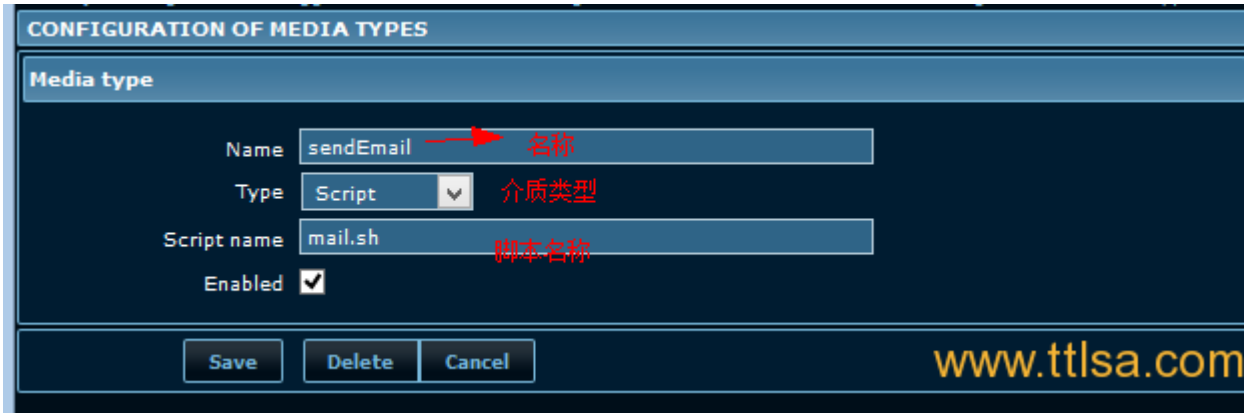
zabbix 脚本报警介质自定义

概述

zabbix 媒介类型包括 mail、sms、自定义脚本，我们用的最多的还是脚本媒介，再次我们就不讲另外两个媒介了。当事件通知到脚本，会传递三个参数它，分别为$1（发送给谁） $2（标题） $3（内容）。例如发送邮件给 support@ttlsa.com，标题为 nginx 离线，内容是 IP 地址：1.1.1.1，nginx 离线，请立即处理。

媒介配置

点击 administrator->media types->create media types



触发器名称

属性	描述
Type	介质类型
script name	脚本名称（需要先定义 AlertScriptsPath,mail.sh 放在这个目录下,写绝路路径没用）
Enabled	状态

配置 AlertScriptsPath

```
# mkdir /usr/local/zabbix-2.2.1/alertscripts
# cat /usr/local/zabbix-2.2.1/etc/zabbix_server.conf| grep -i aler
 AlertScriptsPath=/usr/local/zabbix-2.2.1/alertscripts
```

脚本编写

```
# cd /usr/local/zabbix-2.2.1/alertscripts
 # cat mail.sh
 #!/bin/sh
 to=$1
 subject=$2
 body=$3
/usr/local/bin/sendEmail -f support@ttlsa.com -t "$to" -s smtp.ttlsa.com -u "$subject" -o message-content-type=html -o
message-charset=utf8 -xu monitor@ttlsa.com -xp123456 -m "$body" 2>>/tmp/22.log
```

备注：sendEmail 需要额外安装，请参考《sendEmail 发送邮件》

用户媒介

给用户指定媒介:

*点击 Administration→Users->打开用户属性表单->在 Media tab 点击 Add

参数介绍

属性	描述
Type	选择媒介类型，这边选自定义媒介
Send to	发送到哪，例如 monitor@ttlsa.com,他就是脚本中的$1
When active	报警时间限定，例如 1-5,09:00-18:00，只有工作日的 9 点到 18 点才会通知，实际工作中，我们应该是相反。
Use if severity	严重性类型，只接收指定的类型,例如 info 不想接收，那我不勾选即可。

Status	媒介状态 Enabled – 启用中. Disabled – 已禁用.

zabbix action 报警配置

介绍

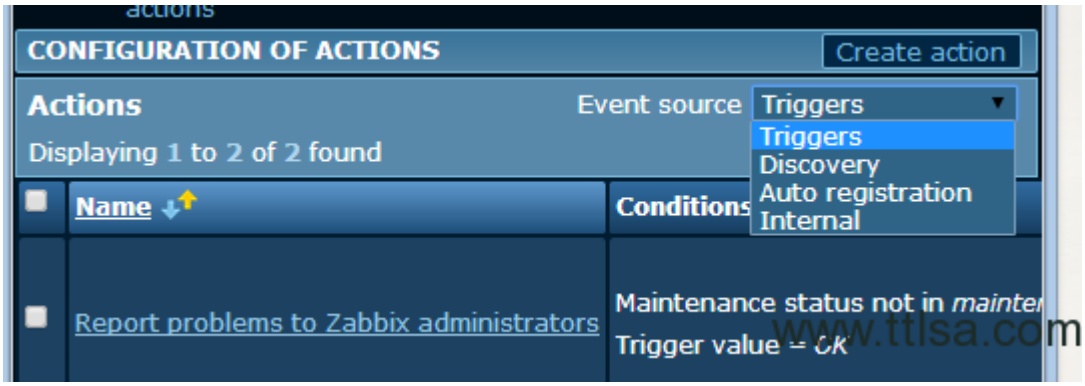
这节内容我们讲解如何配置 zabbix 报警，也就是 zabbix 的 action 配置，action 支持如下事件：

- 触发器事件 - 触发器状态在 OK 和 PROBLEM 之间变化
- 发现事件
- 自动注册时间 - 新的客户端注册进来
- 内部事件 - item 转变为 unsupported 状态，触发器转变为 unknown 状态

配置 action

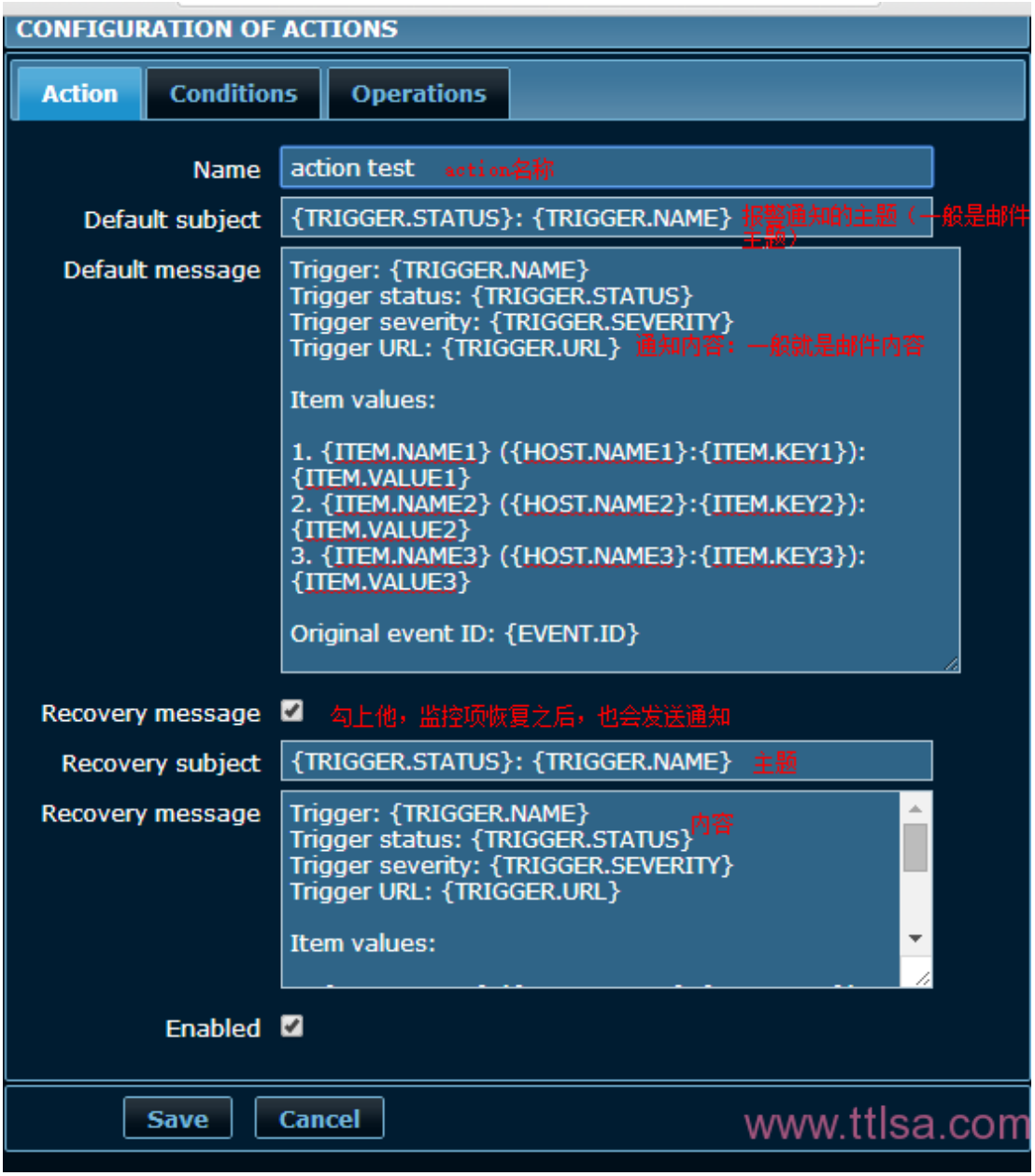
Action 创建

点击 configuration（配置）->Actions（报警）->选择事件来源



Action 配置

如上图，我们可以发现这四种事件来源正好是我们前面提到的，选择好你的事件来源之后，点击 create action



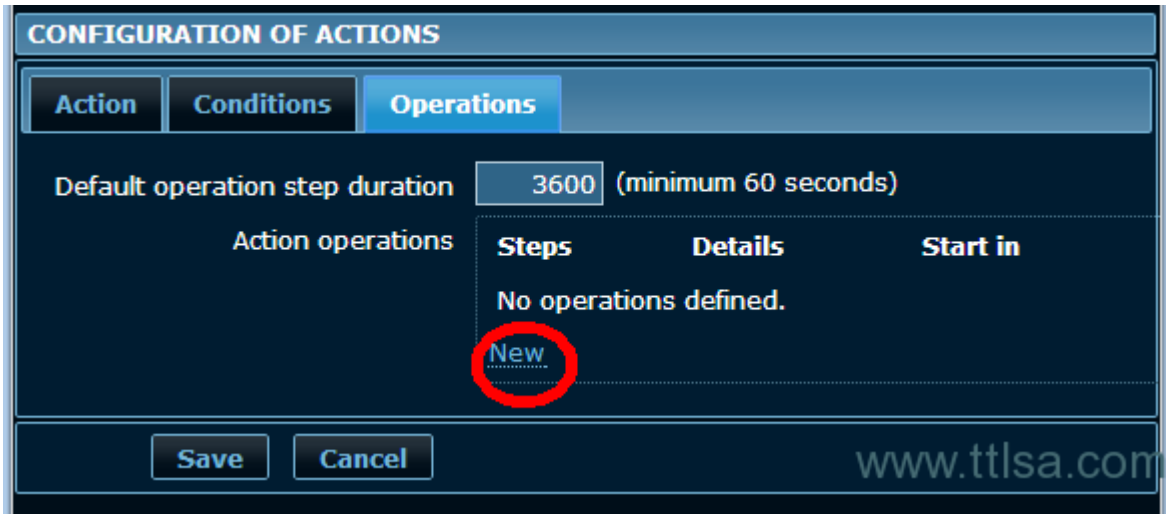
配置完 action 的基本内容之后，接下来配置条件

条件配置

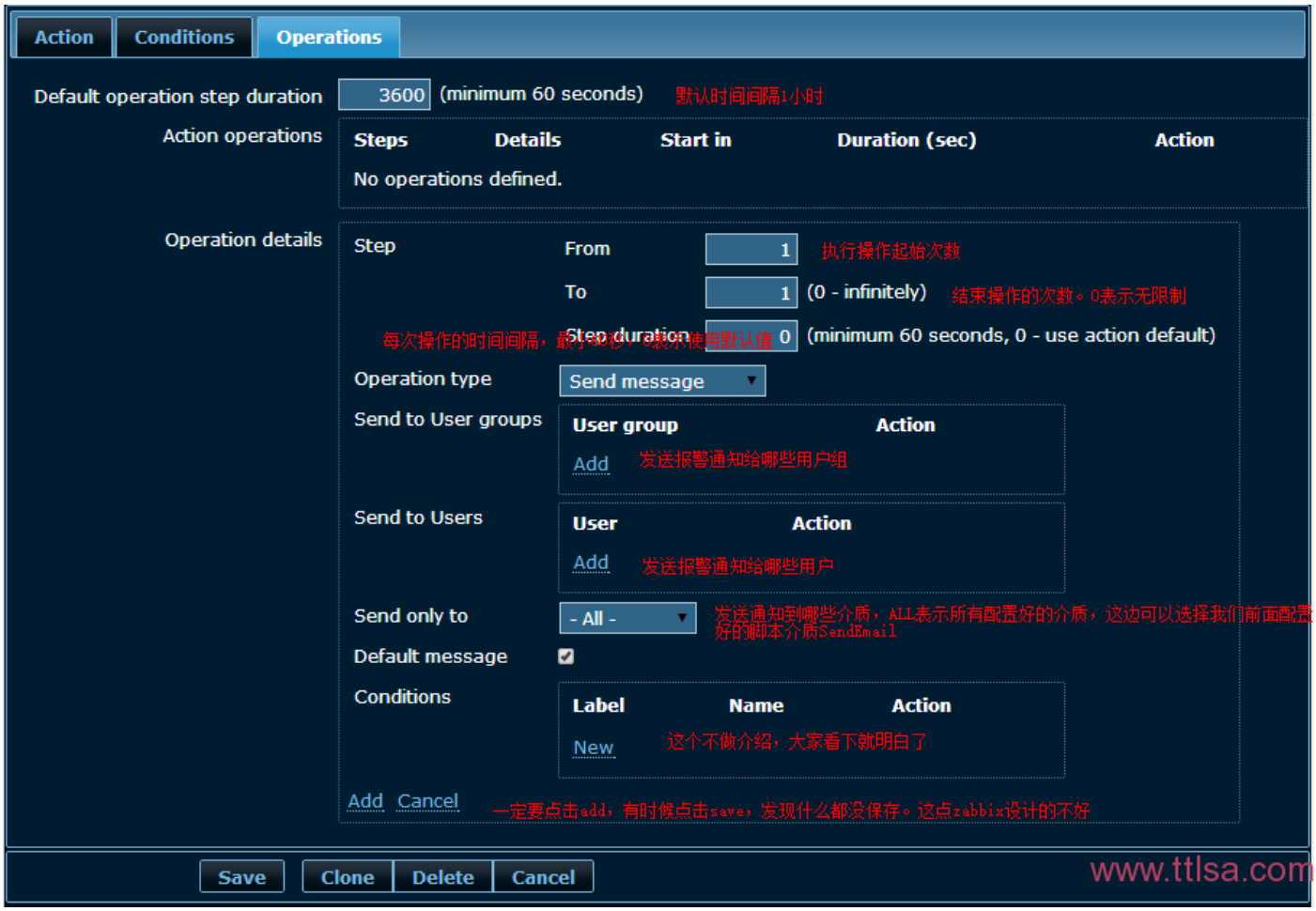
Type Of calculation：各种条件之间的关系，包含 AND、OR 以及 AND/OR，如上图是 AND 关系，同时要满足以上机器不在维护状态以及触发器值为 PROBLEM 才会触发报警的动作。

operaations 配置

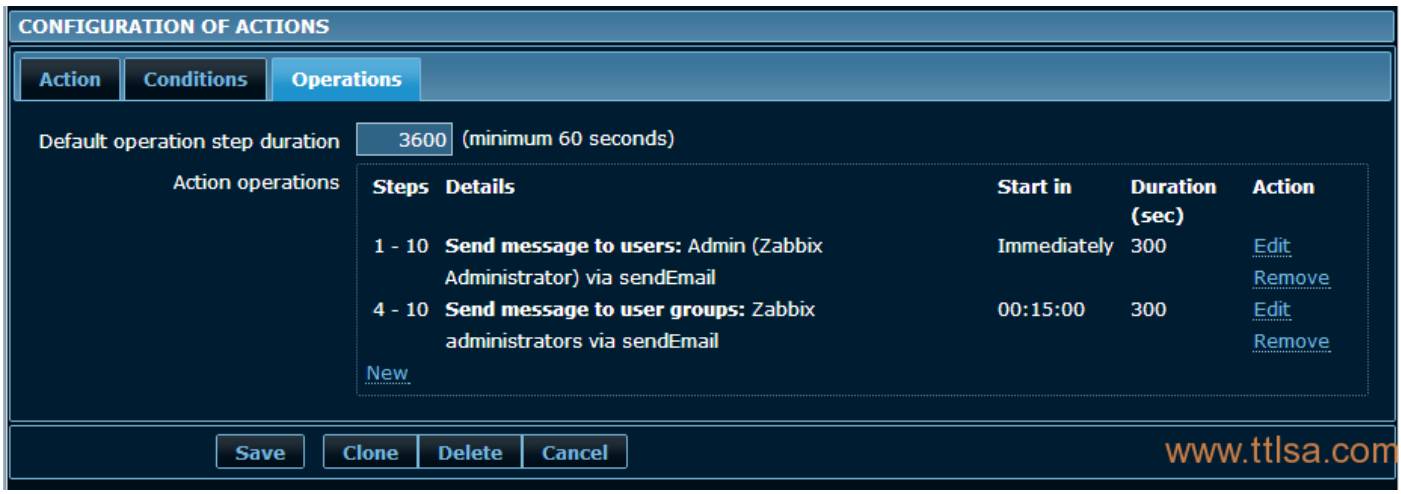
接下来是“操作”标签，如下：



此处没有报警的动作，当你满足了报警条件也没有任何意义，因为你不执行任何报警的操作，那还要 action 做什么，对吧？话说回来，每个 action 都必须配置 operations。



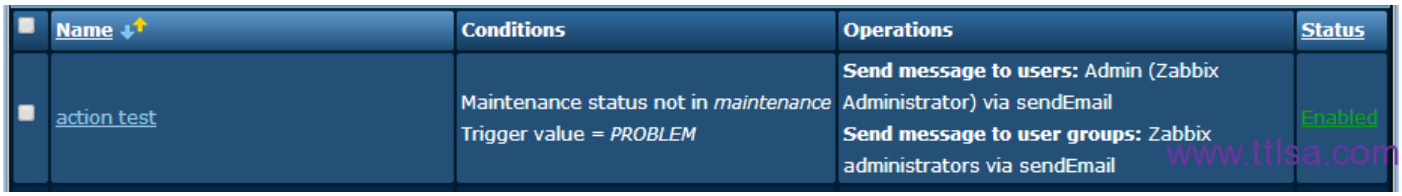
图片上的 step 说的可能不是很明白，表示阶段，1 表示第一次报警，如果 2 表示第二次报警。action operations 可以添加多个，如下：



action 如上图，我们可以看出第 1-10 次报警都会发邮件给 Admin 这个用户，每次邮件间隔为 300 秒，第 4-10 次开始（故障发生 15 分钟后）便会发送邮件给 administrators 这个组。这边可以实现故障开始时发送邮件给值班运维，

多少分钟还没处理好发送邮件给主管或者经理，甚至是老板，呵呵。

最后保存之后可以看到我们配置好的 action 了，只要满足 action 条件便会出发报警操作。



Name	Conditions	Operations	Status
action test	Maintenance status not in *maintenance* Trigger value = *PROBLEM*	**Send message to users:** Admin (Zabbix Administrator) via sendEmail **Send message to user groups:** Zabbix administrators via sendEmail	Enabled

zabbix 用户宏变量详解 macro

zabbix 宏变量让 zabbix 变得更灵活，变量可以定义在主机、模板以及全局，变量名称类似：{$MACRO}，宏变量都是大写的。认识了宏变量，你会感叹 zabbix 越发的强大。

变量可以用于如下地方:

- item 名称
- item key 参数
- 触发器名称和描述
- 触发器表达式
- 其他地方

宏名称

宏变量名称定义只允许后面包含后面的字符: A-Z , 0-9 , _ , .

zabbix 宏变量优先级:

- 主机宏(checked first)
- 主机模板定义的宏,如果有多个模板，那么按照模板越靠前那么宏的优先级越高
- 全局宏(checked last)

话句话说，zabbix 找一个宏的值，首先检查 hosts 有没有，如果没有那么找当前主机的第一个模板，如果还不在找第二个模板，一直到最后一个模板，如果还是没有找到那将会使用全局宏。如果全局宏也不存在，那么宏变量不会被替换掉。

自定义宏

定义全局宏， Administration → General → Macros，例如宏名称${TTLSA_SITE}，值 www.ttlsa.com.

定义主机/模板级宏变量，编辑主机或者模板，找到 Macros 选项卡，定义宏变量

宏变量经常用于替代账号、端口、密码等，例如你的某个监控想有用到账号、密码，可以定义为宏，假如下次账号密码有修改，只需要修改宏即可。而不需要每个监控项都去修改账号密码。

宏变量使用实例

示例一

主机 SSH 服务:

```
net.tcp.service[ssh,{$SSH_PORT}]
```

如果你监控的 ssh 端口可能会有不同，那么你只需要定义 host 级别宏变量 SSH_PORT

示例二

主机 CPU 负载触发器

```
{ca_001:system.cpu.load[,avg1].last(0)}>{$MAX_CPULOAD}
```

例如有三台主机 A\B\C，A 主机定义的 MAX_CPULOAD 为 1，B 定义的 MAX_CPULOAD 为 2，C 定义 MAX_CPULOAD 为 3，这三台 link 同一个模板即可。

示例三

主机 CPU 负载触发器（使用 2 个宏）

```
{ca_001:system.cpu.load[,avg1].min({$CPULOAD_PERIOD})}>{$MAX_CPULOAD}
```

宏可以作为 function 的一个参数。

zabbix 执行远程命令

概述

监控,有的人只把他当做报警使用,出现问题之后打开跑回家打开电脑，巴拉巴拉的处理掉，大多数时候都是一些小问题，为何不让 zabbix 帮你把这些事情处理掉呢？和朋友具体，收到 xx 硬盘空间慢了、xx 服务器高负载等问题，你要回家处理？多扫兴

瞧瞧 zabbix 远程执行命令可以做些什么吧:

- 重启应用（Apache、nginx、MySQL 等等）
- 使用 IPMI 接口重启服务器
- 自动释放磁盘空间（删除老文件，清除/tmp 目录等等）
- CPU 过载时讲一个虚拟机迁移到另外一台物理服务器
- 云环境下，一台服务器 CPU\硬盘\内存\其他硬件资源不足的情况下，可以自动添加过去

创建一个报警，记得使用邮件报警吗？呵呵，实际上，我们把发送邮件的操作改成执行远程命令就行了

备注：zabbix 代理不支持远程命令

远程命令最大长度为 255 字符，同时支持多个远程命令，如需要执行多条命令，只需要另起一行写命令即可，还有，远程命令可以使用宏变量。接下来我将一步一步告诉大家如何设置远程命令

配置

首先我们需要在 zabbix 客户配置文件开启对远程命令的支持，编辑 zabbix_agentd.conf，修改

```
EnableRemoteCommands = 1
```

重启客户端

备注：Aive zabbix 不支持远程命令

然后配置 action，Configuration->Actions，选择 Operation 选项卡，operation type 改成 Remote Command，选择远程命令类似（IPMI, Custom script, SSH, Telnet, Global script），输入远命令

配置 Action

1) 在 Operations 选显卡中选择 Remote command

2) 选择远程命令类型(IPMI, Custom script, SSH, Telnet, Global script)

3) 写上远程命令

示例:

```
sudo /etc/init.d/apache restart
```

上面例子用来在出现状况的情况下重启 Apache，务必全包 zabbix agent 能够执行这个命令.

备注：

1.sudo 不用多说了，zabbix 用户没有运行某些命令的权限,必须加上.

2.远程命令，自然是在远程的主机后台运行。

Conditions 选项卡定义了什么条件下，这个远程命令会被执行，其实这个和前面说的 action 真没什么区别，大家都能看懂。下图的意思是，在非维护时间 Apache 应用出现状况，并且严重性级别为 Disaster。满足条件之后，就开始执行命令了。

访问权限

确保你的 zabbix 用户有执行权限，如果某些命令需要 root 权限，那么请使用 sudo

```
# visudo
```

编辑 sudoer 文件，zabbix 用户便可以执行 Apache restart 命令了

```
# allows 'zabbix' user to run all commands without password.
 zabbix ALL=NOPASSWD: ALL
# allows 'zabbix' user to restart apache without password.
 zabbix ALL=NOPASSWD: /etc/init.d/apache restart
```

备注：在某些情况下，zabbix 需要 sudo 才能执行命令，请先在/etc/sudoer 开启 requiretty.具体的方法，请百度或者 google.

使用多种接口执行远程命令

如果目标系统支持多种接口：zabbix agent、IPMI、远程命令（默认），请看如下一些实例

示例 1

通过 zabbix 检测到的一些问题，然后自动重启 windows

参数	描述
Operation type	Remote command
Type	Custom script
Command	c:\windows\system32\shutdown.exe -r

示例 2

使用 IPMI 重启服务器

参数	描述
Operation type	Remote command
Type	IPMI
Command	reset on

示例三

使用 IPMI 关机

参数	描述
Operation type	Remote command
Type	IPMI
Command	power off

zabbix Trapper 监控项配置

概述

zabbix 获取数据有超时时间，如果一些数据需要执行比较长的时间才能获取的话，那么 zabbix 会出现异常，考虑到这种情况，zabbix 增加了 Trapper 功能，客户端自己提交数据给 zabbix，这个通道便是 trapper.

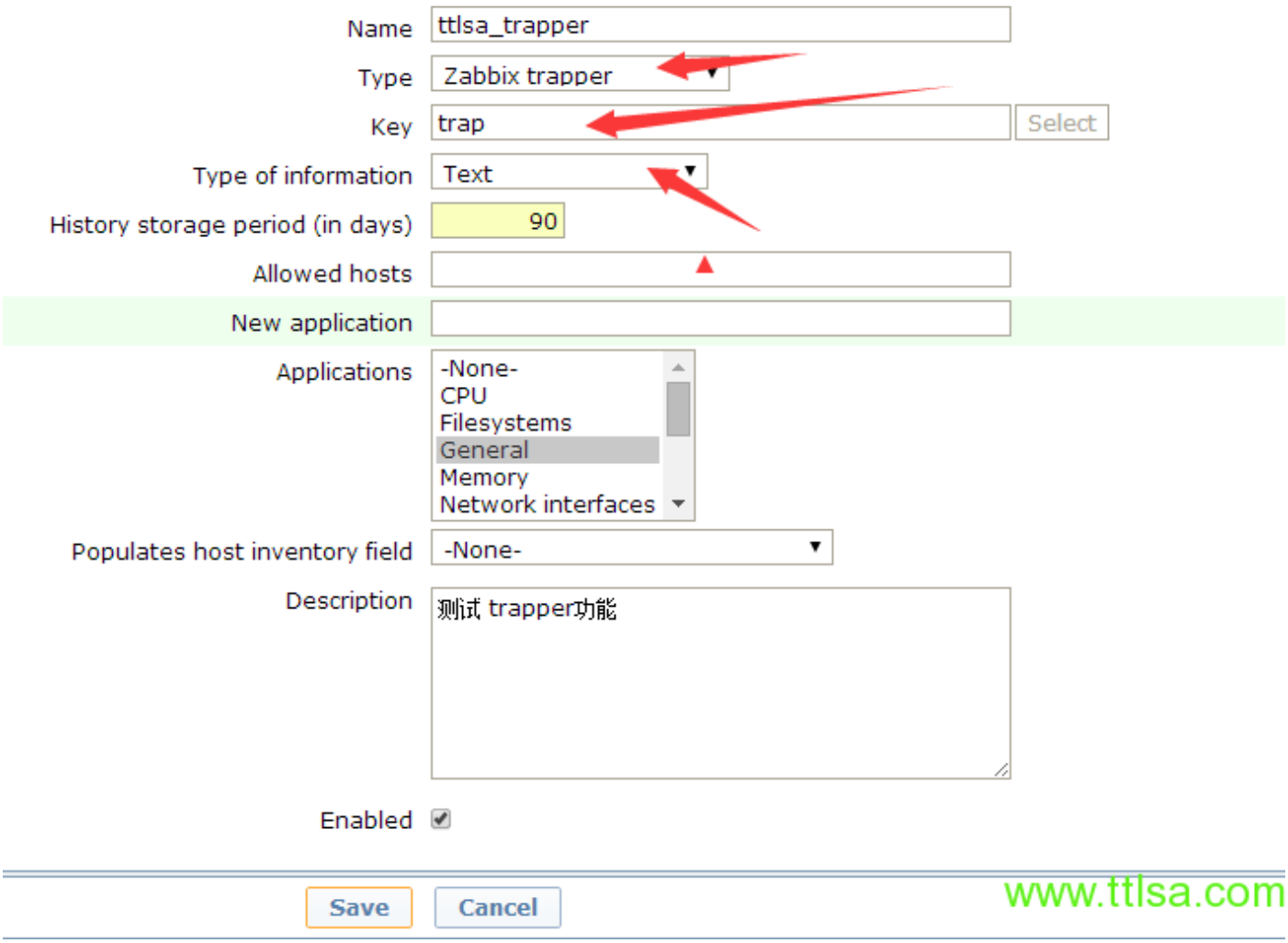
使用 trapper 的步骤如下：

- 在 zabbix 中配置 trapper 监控项
- 传递数据到 zabbix 中

配置

配置监控项

Configuration（配置） → Hosts（主机）－> 选择需要配置的 Host—> 点击右上角的”create item（创建监控项）”－>输入如下参数



参数说明

属性	描述
Type	这边选择 Zabbix trapper.
key	自定义 key 名称，客户端通过 key 来传送数据
Type of information	数据类型，比如数字、文本、浮点等等
Allowed hosts	可选，如果指定了，那么当前监控项只接受指定 IP 地址发送来的数据，多个 IP 使用逗号分隔. 这个参数从 zabbix

传送数据

我们通过一个最简单的例子来讲解，使用 zabbix sender 命令传送“ttlsa.com 你好”给 trapper.

```
# ./bin/zabbix_sender -z 10.9.4.20 -p 10051 -s "test_01" -k trap -o 'ttlsa.com 你好'
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000038"
```

```
sent: 1; skipped: 0; total: 1
```

参数详解:

-z – 指定 zabbix server 的 IP

-p – 指定 zabbix server 的端口，默认为 10051

-s – 指定目标主机，主机名必须是配置中的 hostname 而不是 visible name，切记

-k – 指定 key，我们定义的 trapper 的 key，这边便是我们前面定义的 trap

-o – 指定要传递的数据

备注：如上的 test_01 便是目标主机

展示数据

菜单中 Monitoring → Latest data 可以查看数据

Timestamp	Value
2014.Oct.11 14:06:11	ttlsa.com你好
2014.Oct.11 14:04:47	ttlsa.com你好
2014.Oct.11 14:04:02	ttlsa.com你好

zabbix Aggregate checks 聚合检测

概述

aggregate checks 是一个聚合的检测，例如我想知道某个组的 host 负载平均值，硬盘剩余总量，或者某几台机器的这些数据，简单的说，这个方法就是用来了解一个整体水平，而不需要我们一台台看过去。这个方法的数据全部来之数据库，所以它不需要 agent。文章的最后面我们会有一个简单的图例讲述 aggregate checks.

aggregate item key 语法如下:

```
groupfunc["Host group","Item key",itemfunc,timeperiod]
```

多个组使用逗号分隔.

支持按组的 function

GROUP FUNCTION	描述
grpavg	平均值
grpmax	最大值
grpmin	最小值
grpsum	总和

支持按 item 的 function

ITEM FUNCTION	**描述**
avg	平均值
count	value 个数
last	最新值
max	最大值
min	最小值
sum	总值

参数 timeperiod 为指定的采集时间，可以使用时间单位，例如可以使用 1d 代替 86400（单位默认为秒）,5m 代替 300.

备注：

- 如果第三个参数为 last，那么 timeperiod 参数值将会被 server 忽略掉
- 只有被监控的 HOST 上启用的 item 才会被计入 aggregate check

使用范例

示例 1

组 MySQL Servers 剩余硬盘空间大小

```
grpsum["MySQL Servers","vfs.fs.size[/,total]",last,0]
```

示例 2

组 MySQL Servers 的平均 CPU 负载

```
grpavg["MySQL Servers","system.cpu.load[,avg1]",last,0]
```

示例 3

组 MySQL Servers 5 分钟内的平均查询速度（次/秒）

```
grpavg["MySQL Servers",mysql.qps,avg,5m]
```

示例 4

多个组的 cpu 负载平均值

```
grpavg[["Servers A","Servers B","Servers C"],system.cpu.load,last,0]
```

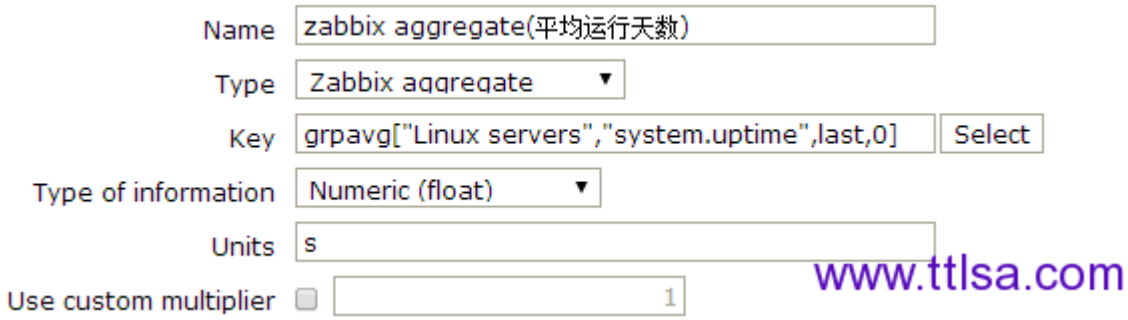
示例（带图）

获取 linux servers 组内所有 HOST 平均运行天数

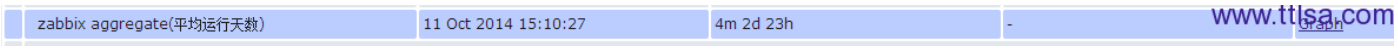
首先在 zabbix server 上配置 item,名字就叫做：zabbix aggregate(平均运行天数)，

key 为：grpavg[“Linux servers”,”system.uptime”,last,0]

具体请看图：



获取到的结果如下：



最后

如果如要对某个监控项有一个整体的了解，zabbix aggregate 是你的不二选择.

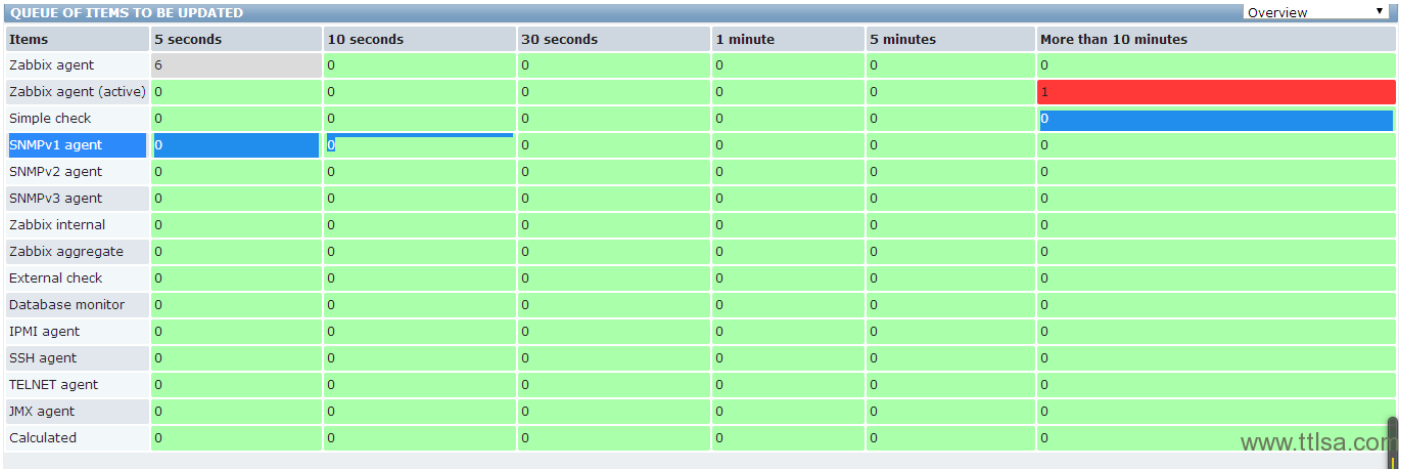
zabbix Queue 队列

概述

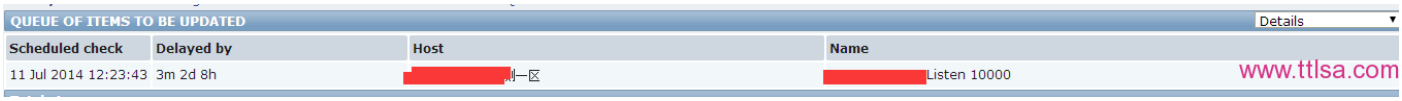
queue（队列）显示监控项等待刷新的时间,可以看到每种 agent 类型刷新时间，通过 queue 可以更好的体现出监控的一个指标.正常情况下，是一片绿色。如果出现过多红色，那么需要留意一下。我们也可以在右上角的下拉条选 detail，可以找出到底是哪个 item 的问题。

读取队列

点击 Administration（管理） → Queue（队列）. 下拉框三个选项，分别为 overview、overview by proxy、detail，如下为 overview



上图大部分为绿色,表示服务器整体 OK,上图可以看到 10 秒和 5 分钟处有两个 item 还未刷新.为了找出是哪个 item，我们可以再下拉框选到 detail。



如上图，可以轻易的找出刷新存在延迟 item 的详细信息（因为延迟很快就恢复了，所以第一张图抓到 7 个延迟，detail 只出现一个）。偶尔出现几个延迟，那是很正常的，一般都会快速恢复的。但是如果比较多的超过 10 分钟的延迟，那么你要好好的留意一下了。有可能出现比较严重的问题。

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	0	1	0	1	0	0
Zabbix agent (active)	0	0	0	0	0	0

远程节点延迟

来自子节点（child node）的信息部都不是最新的。master 节点接受到的数据都存在一定得延迟（通常情况下，多则需要 10 秒）

决定子节点信息延迟因素

- 子节点性能
- 子节点与主节点之间的通行质量
- 子节点与主节点之间的时间差

Queue item

既然 queue 也是一项性能指标，那么我们也有很必要把他加入监控项，zabbix 提供了内建 item zabbix[queue,,] ，from 默认为 6 秒，表示超过多少秒便报警，to 默认为 infinity，也就是无限制.

zabbix 正则表达式应用

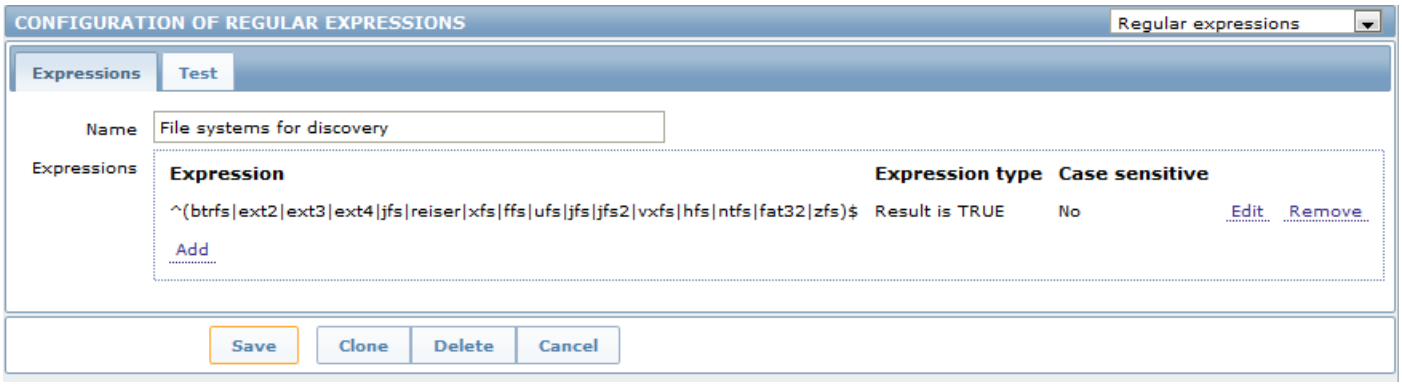
概述

在前面的《zabbix low-level discovery》一文中有 filter 一项，用于从结果中筛选出你想要的结果，比如我们在 filter 中填入^ext|^reiserfs 则表示{#FSTYPE}要符合 ext 与 reiserfs 才能加入到 item 中。这个需求比较简单，假如我们过滤方法更严格怎么办？或者说多个 low-level 的 filter 都有用到同一个正则表达式，并且希望修改一处，所有的 filter 都跟着修改，请使用 zabbix 正则表达式。

配置

点击 Administration >>General>>右侧下拉表选择“ Regular expressions”>>New regular expression.

所有匹配完全匹配 btrfs|ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|ntfs|fat32|zfs 文件类型返回 true，正则表达式为^(btrfs|ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|ntfs|fat32|zfs)$



说明：如果符合如上表达式，那么返回 TRUE，否则返回 FLASE，可以 Add 多个表达式，所有的表达式之间是逻辑与的关系，必须所有的表达式都返回 TRUE，最终的结果才是 TRUE。

正则表达式参数说明

Expression：正则表达式名称

Expression type：正则表达式类型

1. Character string included - 字符串包含，例如表达是处写 abc，你返回的内容是 abcd，那么就算匹配了，将会返回 TRUE

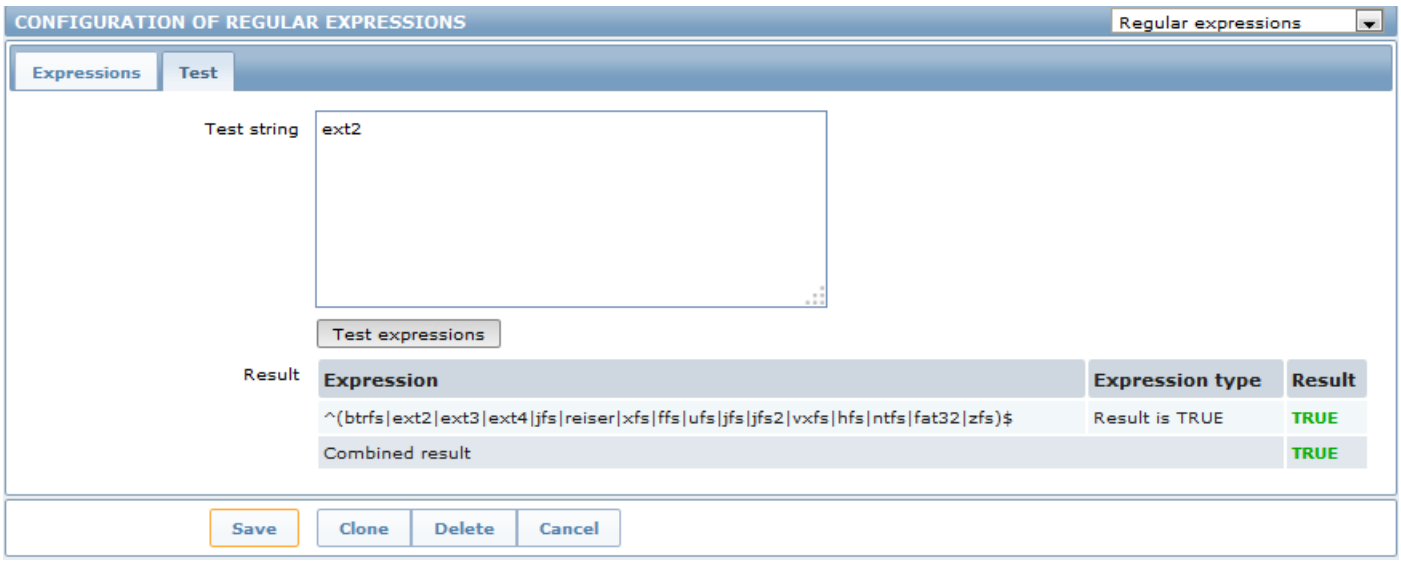
2. Any character string included - 任意字符串包含在内（以逗号','，点号'.'，斜杠'/'分隔），这边使用上有点问题

3. Result is TRUE - 符合正则表达式返回 TRUE、否则返回 FALSE

4. Result is FALSE - 符合正则表达式返回 FALSE、否则返回 TRUE

Case sensitive：是否区分大小写

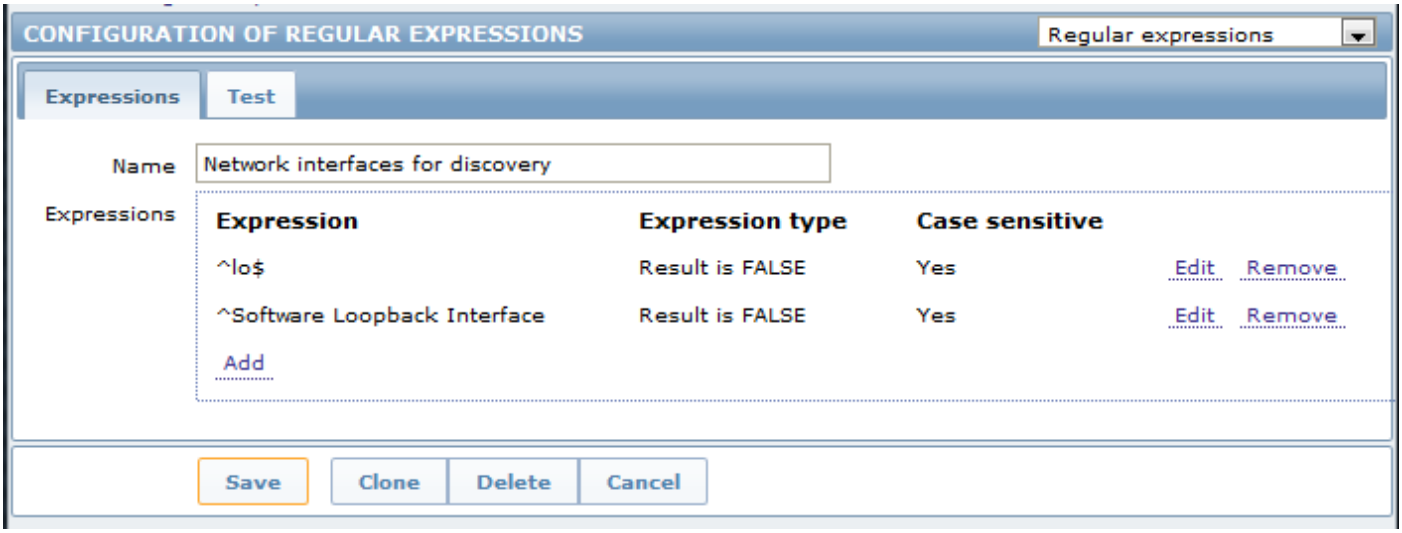
点击 Test 标签测试正则表达式是否符合预期，例如输入 ext2 点击 test，可以发现最后结果是 TRUE，如下图



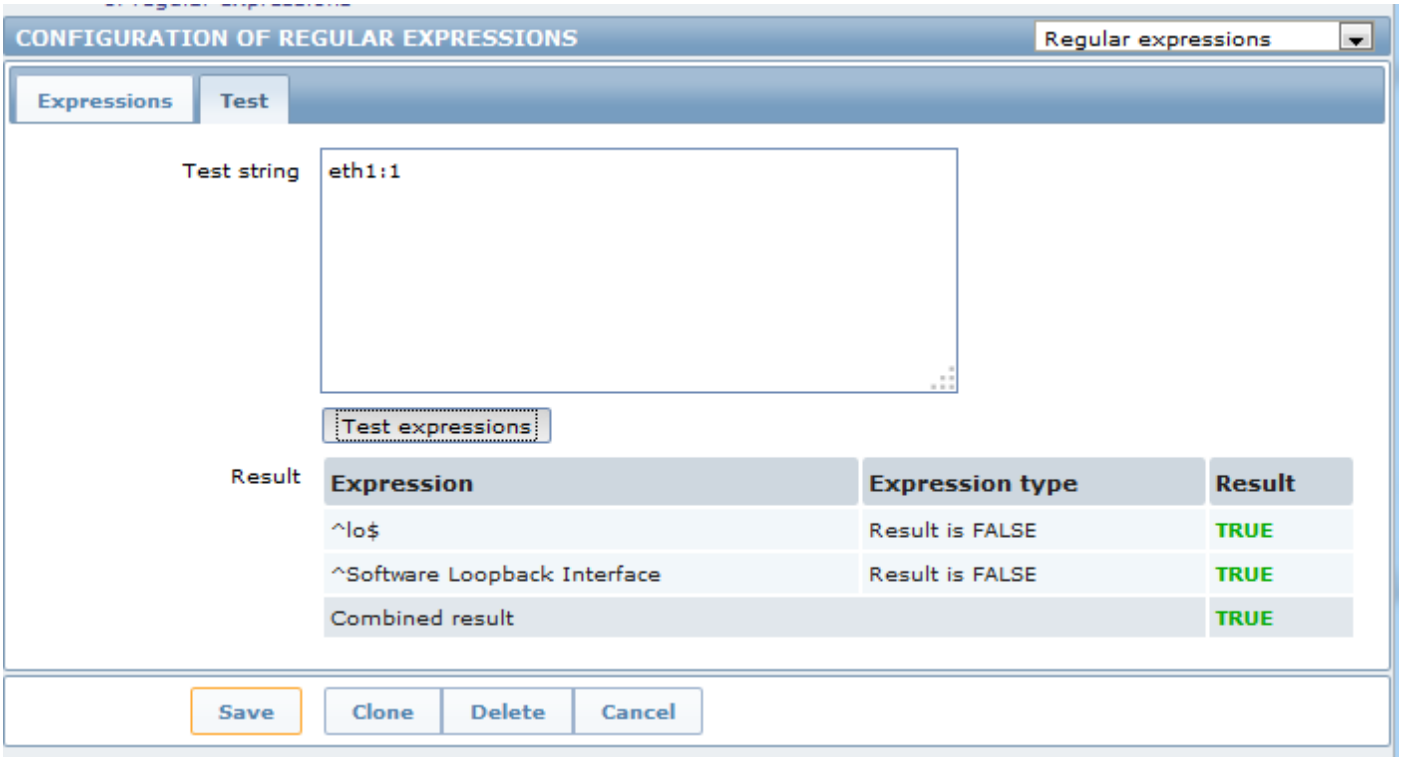
输入不符合预期的值 d:，我们可以发现最后结果是 FLASE，如下图

常用示例

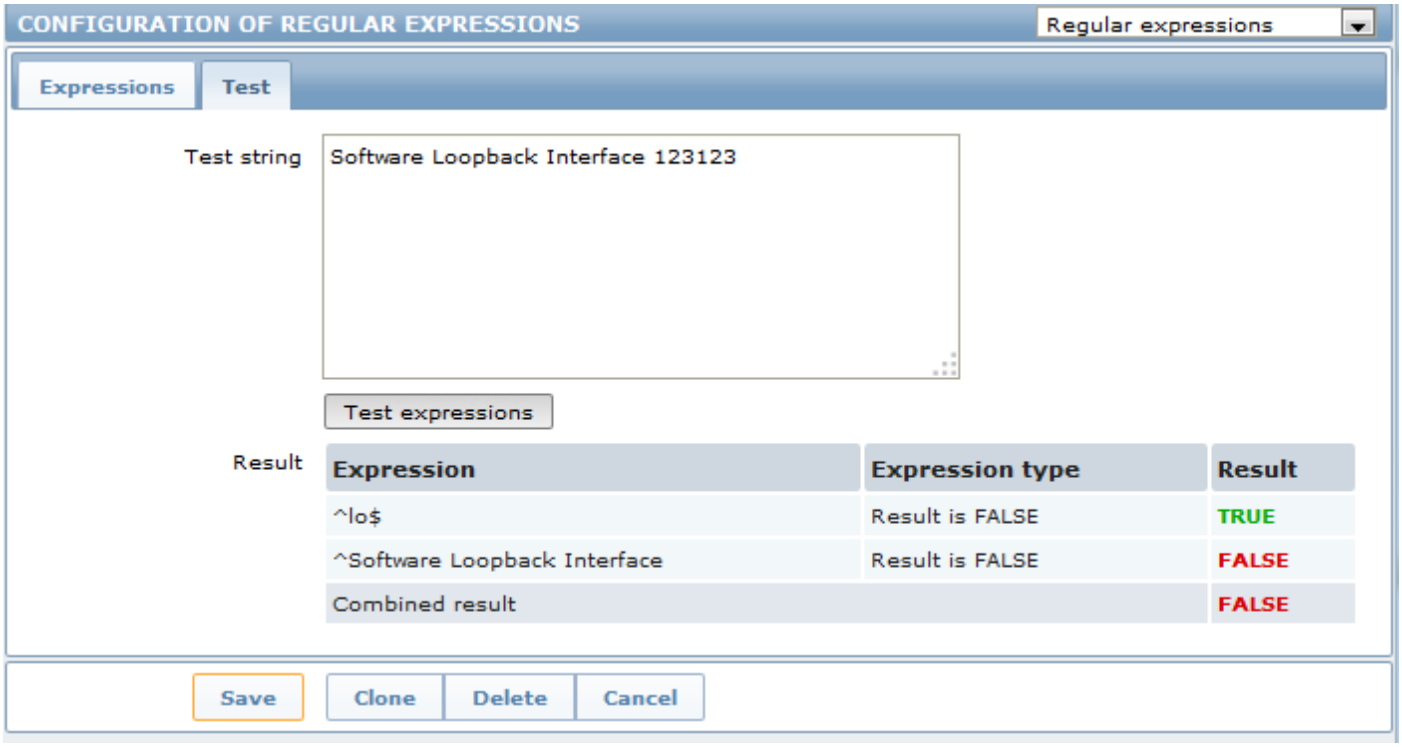
1. 排除 lo 网卡与 Software Loopback interface 开头的内容



不符合表达式的 eth1:1 返回 TRUE

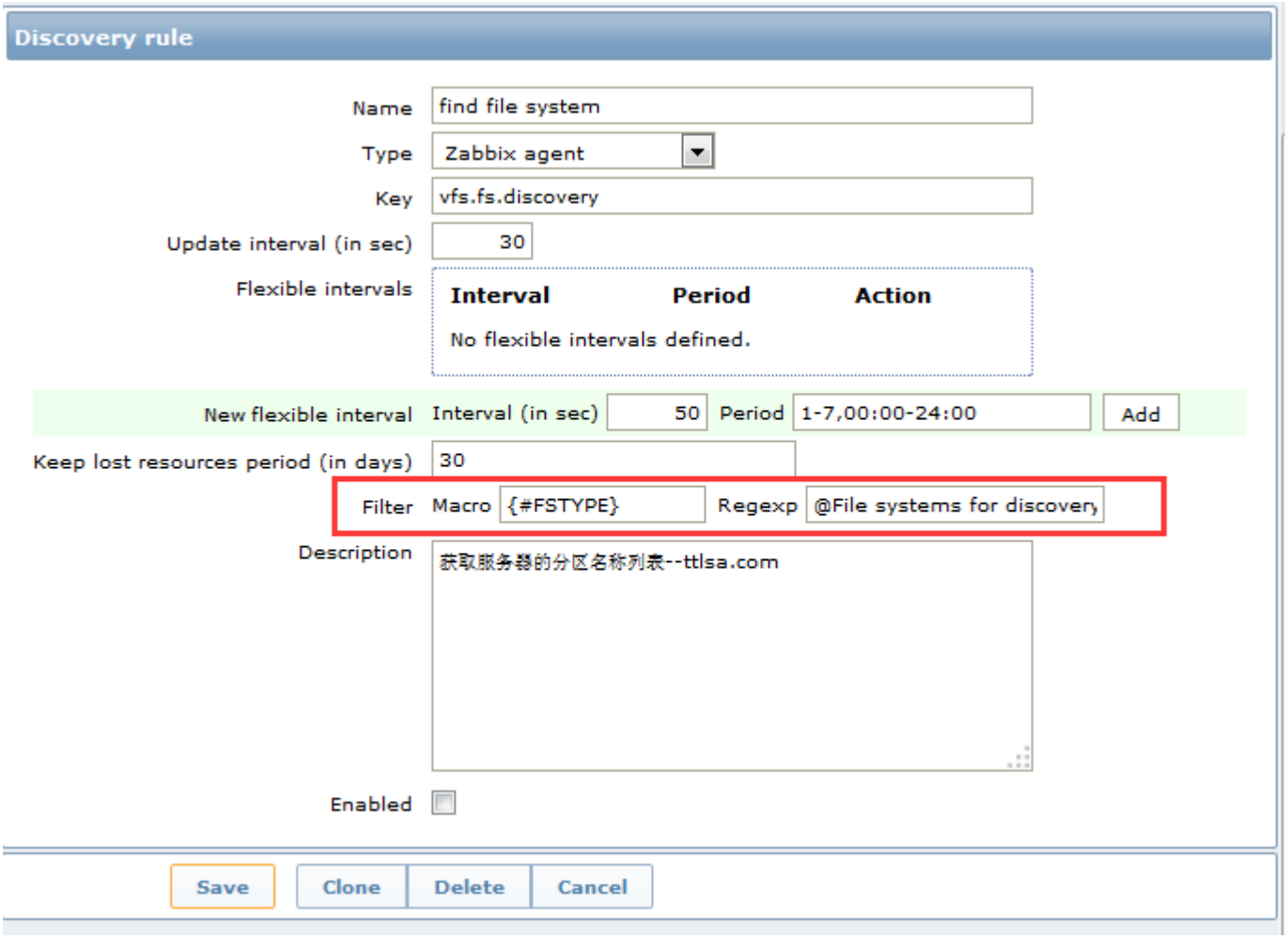


符合表达式的 Software Loopback Interface 123 返回 FLASE



如何引用正则表达式

例如 low-level filter 需要引用表达式，在表达式名称前加@即可，例如@你的正则表达式名称、@File systems for discovery



正则表达式名称规范

名称可以包含逗号与空格，例如@network， ttlsa，个人不推荐你使用逗号，让人觉得很奇怪，并且使用逗号的时候，你需要用双引号将整个名称括起来，例如“@network， ttlsa”。

zabbix 导入/导出配置文件

通过导入/导出 zabbix 配置文件，我们可以将自己写好的模板等配置在网络上分享，我们也可以导入网络上分享的配置文件，配置文件有两种格式，分为为 xml 与 json，通过 zabbix 管理界面可以导出 xml，通过 zabbix api 可以导出 json 与 xml 格式配置。

可导出的项目

- host groups (只能通过 zabbix api 导出)
- templates (包含所有预其直接关联的 items, triggers, graphs, screens, discovery rules 和 link 的 template)
- hosts (包含所有与它直接相关的 items, triggers, graphs, discovery rules 和 link 的 template)
- network maps (包含所有相关图片; map export/import 从 Zabbix 1.8.2 开始被支持)
- images
- screens

导出详细说明

- 导出为单个文件

a) Host and template 从模板中 link 过来的实体 (items, triggers, graphs, discovery rules)不会导出，通过 low-level 创建的实体不会导出。但是导如之后，会重新创建 link 的实体。

b) 与 low-level 实体相关联的实体不会被导出，例如触发器。

c) web items 相关的 Triggers and graphs 不支持导出

- 导入详细说明

a) 导如过程中遇到任何错误，都会停止导入

b) 支持导如 xml 与 json 格式文件，并且文件名后缀必须为.xml 或者.json

XML 文件基本格式

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>2.0</version>
    <date>2015-02-09T05:58:54Z</date>
</zabbix_export>
```

xml 默认头

```
<?xml version="1.0" encoding="UTF-8"?>
```

zabbix xml root 元素

```
<zabbix_export>
```

导出版本

```
<version>2.0</version>
```

导出日期

```
<date>2015-02-09T05:58:54Z</date>
```

导出模板

configuration>>templates>>勾选需要导出的模板>>左下角下拉列表选择"Export selected",点击 Go，保存 xml 即可。

如下是我的一个测试模板内容

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>2.0</version>
    <date>2015-02-09T07:34:25Z</date>
    <groups>
        <group>
            <name>Templates</name>
        </group>
    </groups>
    <templates>
        <template>
            <template>A_Template_For_Discovery</template>
            <name>A_Template_For_Discovery</name>
```

```
        <groups>
            <group>
                <name>Templates</name>
            </group>
        </groups>
        <applications>
            <application>
                <name>Network</name>
            </application>
        </applications>
        <items/>
        <discovery_rules>
            <discovery_rule>
                <name>PING</name>
                <type>0</type>
                <snmp_community/>
                <snmp_oid/>
                <key>net.if.icmpping</key>
                <delay>30</delay>
                <status>0</status>
                <allowed_hosts/>
                <snmpv3_contextname/>
                <snmpv3_securityname/>
                <snmpv3_securitylevel>0</snmpv3_securitylevel>
                <snmpv3_authprotocol>0</snmpv3_authprotocol>
                <snmpv3_authpassphrase/>
                <snmpv3_privprotocol>0</snmpv3_privprotocol>
                <snmpv3_privpassphrase/>
                <delay_flex/>
                <params/>
```

```
<ipmi_sensor/>
<authtype>0</authtype>
<username/>
<password/>
<publickey/>
<privatekey/>
<port/>
<filter>{#LOC}:^cn|^jp</filter>
<lifetime>30</lifetime>
<description/>
<item_prototypes>
    <item_prototype>
        <name>PING IP $1</name>
        <type>3</type>
        <snmp_community/>
        <multiplier>1</multiplier>
        <snmp_oid/>
        <key>icmpping[{#IPADD},4,,,]</key>
        <delay>30</delay>
        <history>90</history>
        <trends>365</trends>
        <status>0</status>
        <value_type>3</value_type>
        <allowed_hosts/>
        <units/>
        <delta>0</delta>
        <snmpv3_contextname/>
        <snmpv3_securityname/>
        <snmpv3_securitylevel>0</snmpv3_securitylevel>
        <snmpv3_authprotocol>0</snmpv3_authprotocol>
```

```
                        <snmpv3_authpassphrase/>
                        <snmpv3_privprotocol>0</snmpv3_privprotocol>
                        <snmpv3_privpassphrase/>
                        <formula>1000</formula>
                        <delay_flex/>
                        <params/>
                        <ipmi_sensor/>
                        <data_type>0</data_type>
                        <authtype>0</authtype>
                        <username/>
                        <password/>
                        <publickey/>
                        <privatekey/>
                        <port/>
                        <description/>
                        <inventory_link>0</inventory_link>
                        <applications>
                            <application>
                                <name>Network</name>
                            </application>
                        </applications>
                        <valuemap/>
                    </item_prototype>
                </item_prototypes>
                <trigger_prototypes/>
                <graph_prototypes/>
                <host_prototypes/>
            </discovery_rule>
        </discovery_rules>
        <macros/>
```

```
            <templates>
                <template>
                    <name>Template OS Linux</name>
                </template>
            </templates>
            <screens/>
        </template>
    </templates>
</zabbix_export>
```

与模板相关的数据都在 xml 里，它 link 的模板"Template OS Linux"并未导出。而是通过如下元素将他关联起来，下回导入还会 link 一次。

```
<templates>
    <template>
        <name>Template OS Linux</name>
    </template>
</templates>
```

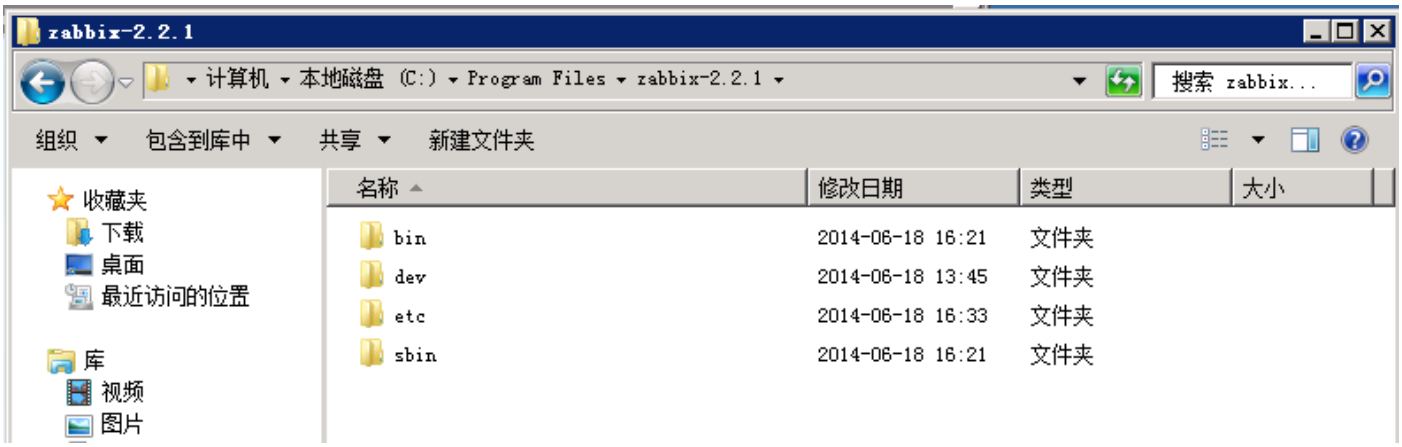
通过此方式，大家可以互相共享配置文件，提高效率。

windows 安装 zabbix 监控

在 windows 下安装 zabbix agent，方法非常简单。首先到 zabbix 官方下载 windows 版本 agent，地址：http://www.zabbix.com/download.php，找到“Zabbix pre-compiled agents”选择相应的版本下载。安装方法很简单，下载-->解压到目录-->修改配置文件-->安装服务-->启动-->web 中添加 HOST 即可。

文件解压到指定目录

如下图即可



bin：zabbix_get.exe、zabbix_sender.exe 连个文件

sbin：zabbix_agentd.exe 客户端主进程

etc：配置文件

dev：不用理会

修改配置文件

修改 etc/zabbix_agentd.conf，只要修改以下几个配置即可（与 linux 配置文件没什么差别），只需要修改以下三个，其他参数都默认。

```
Server=10.0.0.20 # zabbix 服务器 ip 地址
ServerActive=10.0.0.20 # zabbix 服务器 ip 地址
Hostname=win-server-ttlsa # 客户端机器名
```

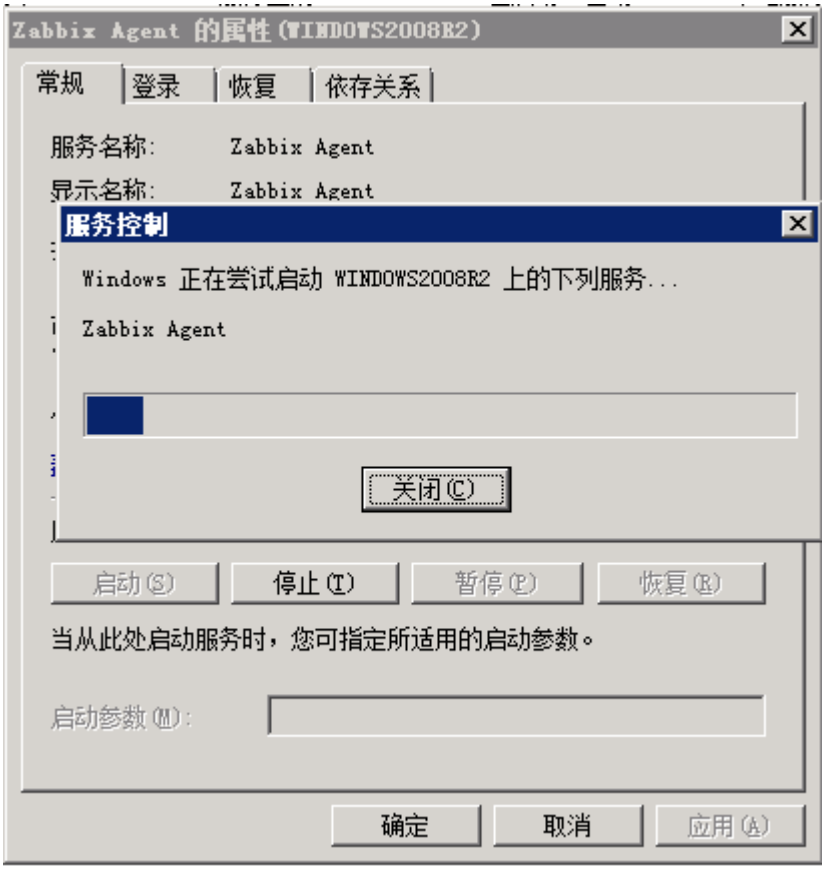
安装服务

```
C:\Program Files\zabbix-2.2.1\sbin>zabbix_agentd.exe -i -c ..\etc\zabbix_agentd.conf
zabbix_agentd.exe [2304]: service [Zabbix Agent] installed successfully
```

```
zabbix_agentd.exe [2304]: event source [Zabbix Agent] installed successfully
```

启动 zabbix agentd

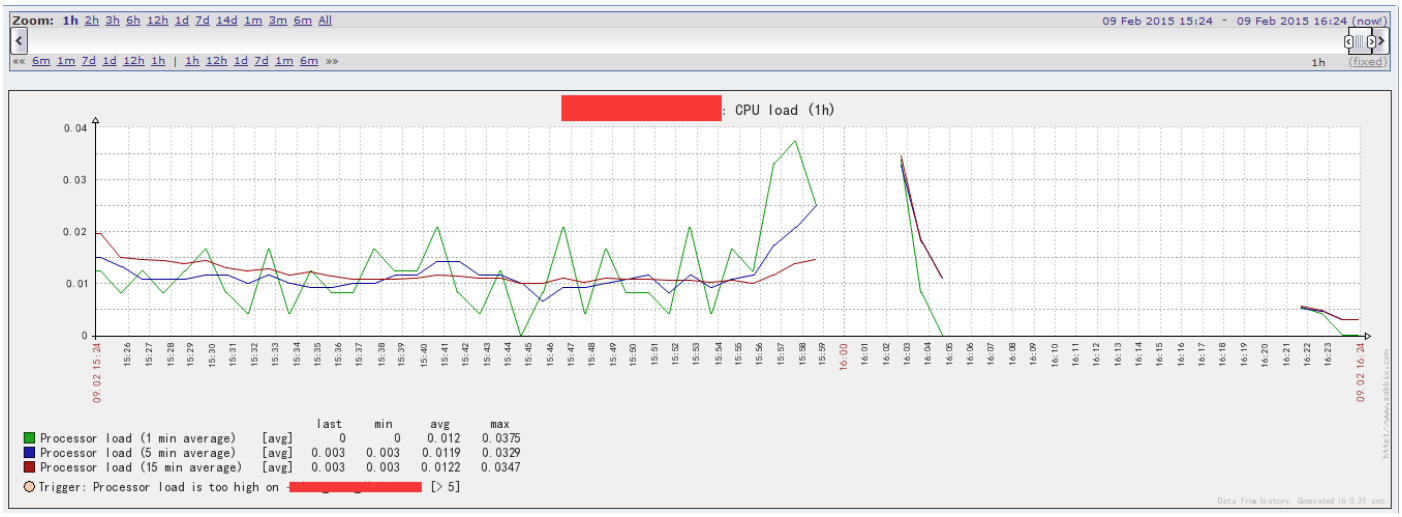
开始->>运行->> services.msc，双击 zabbix agent，点击启动。



添加 Host

这步略过，请参考前面的《zabbix 主机与组配置（12）》

效果图


Zoom: 1h 2h 3h 6h 12h 1d 7d 14d 1m 3m 6m All
09 Feb 2015 15:24 - 09 Feb 2015 16:24 (now!)
«« 6m 1m 7d 1d 12h 1h | 1h 12h 1d 7d 1m 6m »»
1h (fixed)
: CPU load (1h)
0.04
0.03
0.02
0.01
0
09 02 15:24
09 02 16:24
last min avg max
Processor load (1 min average) [avg] 0 0 0.012 0.0375
Processor load (5 min average) [avg] 0.003 0.003 0.0119 0.0329
Processor load (15 min average) [avg] 0.003 0.003 0.0122 0.0347
Trigger: Processor load is too high on [> 5]

zabbix windows 性能计数器使用详解

概述

windows 下的性能计数器让 zabbix 监控更加轻松，直接获取性能计数器的数值即可完成 windows 监控。性能计数器如下：

```
perf_counter["\Processor(0)\Interrupts/sec"]
```

或

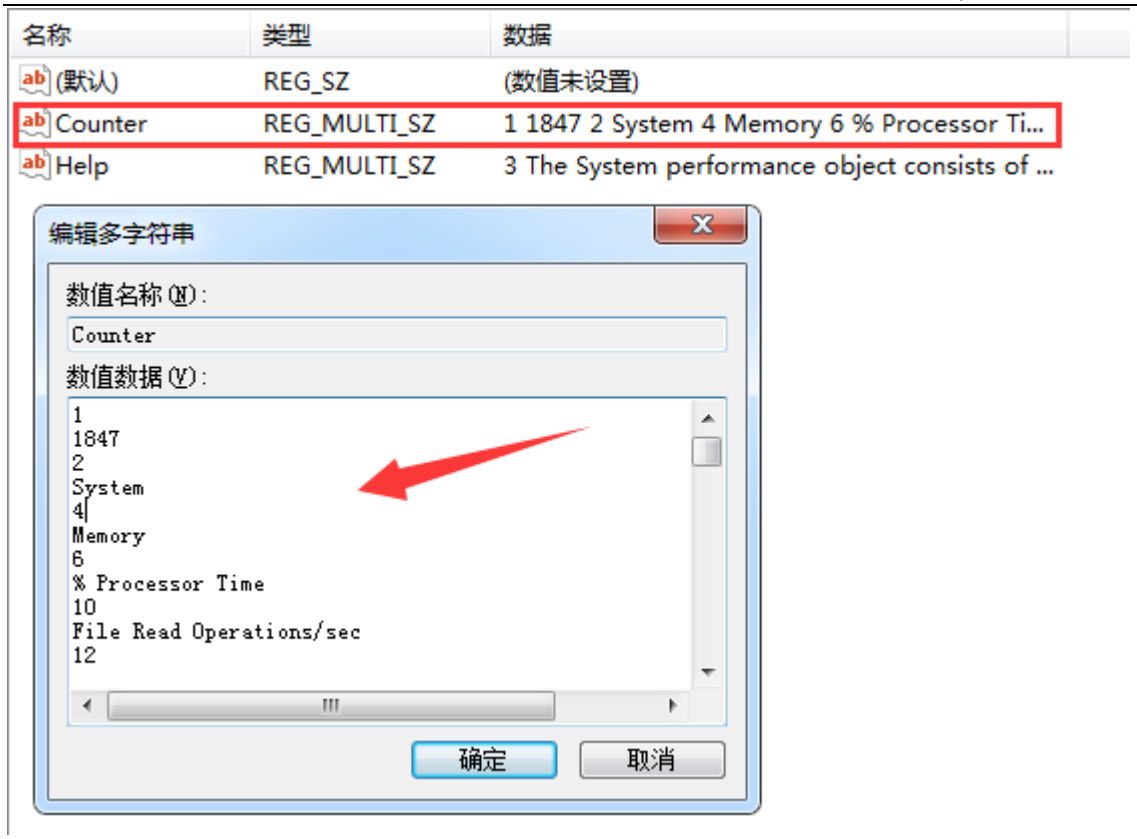
```
perf_counter["\Processor(0)\Interrupts/sec", 10]
```

获取所有性能计数器命令：

```
typeperf -qx
```

数字对应

如上的 perf_counter["\Processor(0)\Interrupts/sec"]，里面的\Processor(0)\Interrupts/sec 很难记忆，而且不同的 windows 系统名称不可能不相同，这可能会导致获取到错误的值。基于此，windows 有相应的数字与名称对应，比如：system 对应 2，Memory 对应 4，有几千个性能计数器名称与数字对。那怎么找到名称对应的数字呢？打开注册表 Regedit，找到 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009，打开 key counter



在 win2008 下，有 5000 多行，大概 2000 多对。贴一部分文字出来吧

```
1
1847
2
System
4
Memory
6
% Processor Time
10
File Read Operations/sec
```

自定义性能计数器 key

编辑 agentd 配置文件，添加 PerfCounter 自定义内容

```
PerfCounter=UserPerfCounter1,"\Memory\Page Reads/sec",30
```

或者

```
PerfCounter=UserPerfCounter2,"\4\24",30
```

UserPerfCounter1 与 UserPerfCounter2 其实是一样的，4 取代了 Memory，24 取代了 Page Reads/sec，虽然说可读性差一点，但是推荐大家使用数值。

zabbix 加载扩展模块 第三方库支持

zabbix 从 2.2 版本开始增加了使用动态库来扩展 zabbix 功能。loadable modules 实际上和我们前面提到的用户自定义 key 是一样的功能，不同的是，他用加载 lib 库的方式，并且 zabbix 不需要 fork 一个新的进程，性能更好。目前类似的功能包含 user parameters 、 external checks 、 system.run[] ，如果这些脚本逻辑过于复杂、耗时太长会出现比较严重的问题。

工作中，我们可以使用 c 开发一些适用于我们自己生产环境的模块。当然你也可以将它分享给出来，而不需要公布你的源代码，如果你对自己写的代码不自信的话。当 agentd、server、proxy 启动的时候同时将模块加载进来，退出的时候也会释放。

zabbix 模块 API

zabbix 代码中有提供 api 所需的头文件.h，目前模块有两类接口需要实现，一类是必须实现的，一类是可选的。

必须实现的接口

两个接口： zbx_module_api_version()、zbx_module_init()

```
int     zbx_module_api_version(void);
```

用于返回 API 版本，必须实现，默认返回常量 ZBX_MODULE_API_VERSION_ONE（数值 1）

```
int     zbx_module_init(void);
```

模块必要的一些初始化，初始化成功返回 ZBX_MODULE_OK，否则返回 ZBX_MODULE_FAIL。

可选接口

可选接口有 zbx_module_item_list()、 zbx_module_item_timeout()、 zbx_module_uninit()

ZBX_METRIC *zbx_module_item_list(void);

返回模块内定义的 item 列表，包含 key，如：agent.ping、agent.version，每个 item 都使用结构体 ZBX_METRIC

```
void	zbx_module_item_timeout(int timeout);
```

超时时间设置，秒为单位

```
int	zbx_module_uninit(void);
```

释放资源，如：文件描述等

定义 item 结构体

```
typedef struct
{
	char		*key;
	unsigned	flags;
	int		(*function)();
	char		*test_param;
}
```

ZBX_METRIC;

key：item key 名称，例如 agent.ping、mysql.version 等

flags：CF_HAVEPARAMS 或者 0

function：将要调用的函数

test_param：参数列表

示例

```
static ZBX_METRIC keys[] =
{
	{ "dummy.random", CF_HAVEPARAMS, zbx_module_dummy_random, "1,1000" },
	{ NULL }
}
```

在定义 function 需要接收两个参数 AGENT_REQUEST 、AGENT_RESULT ， 如下

```
int     zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    ...

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}
```

编译模块

编译准备

zabbix 提供了一份用于测试的模块源码，在 zabbix 源码目录下

```
# cd /usr/local/src/zabbix-2.4.3/src/modules/dummy
# ll
total 32
-rw-r--r-- 1 1001 1001 9024 Dec 16 07:37 dummy.c
-rw-r--r-- 1 1001 1001   73 Dec 16 07:37 Makefile
-rw-r--r-- 1 1001 1001  245 Dec 16 07:37 README
```

请一定记住所有的源代码最好放到 modules 目录下来编译，因为他需要一些接口都在源码中。例如 include/module.h、include/sysinc.h、 include/config.h，前面两个.h 文件解压就存在，而 config.h 需要在源码根目录下执行./configure（不能带参数，否则会报错）。

开始编译

```
# cd  /usr/local/src/zabbix-2.4.3/
# ./configure
.....内容忽略.....
# cd /usr/local/src/zabbix-2.4.3/src/modules/dummy
```

```
# make
gcc -fPIC -shared -o dummy.so dummy.c -I../../../include
# ll
total 32
-rw-r--r-- 1 1001 1001 9024 Dec 16 07:37 dummy.c
-rwxr-xr-x 1 root root 8526 Feb 10 10:48 dummy.so
-rw-r--r-- 1 1001 1001   73 Dec 16 07:37 Makefile
-rw-r--r-- 1 1001 1001  245 Dec 16 07:37 README
```

加载模块

拷贝 so 文件到 zabbix 目录下

```
# cp dummy.so /usr/local/zabbix-2.4.3/lib/
```

修改配置文件

```
LoadModulePath=/usr/local/zabbix-2.4.3/lib/ #  可自定义
LoadModule=dummy.so #  可以加载多个
```

测试模块

重启 zabbix_agentd

```
# killall zabbix_agentd
# /usr/local/zabbix-2.4.3/sbin/zabbix_agentd
```

测试 key

```
# ./zabbix_get -s 127.0.0.1 -k dummy.echo[ttlsa.com]

ttlsa.com

#./zabbix_get -s 127.0.0.1 -k dummy.ping

1

#./zabbix_get -s 127.0.0.1 -k dummy.random[10,100]

73
```

可以看到定义好的三个 key 都成功了。学好 linux c 开发自己的 zabbix 模块吧。

zabbix telnet 监控类型

概述

zabbix 监控的方式很多，例如前面讲到的 agent、snmp 以及后续后续要讲到 ssh 和今天要讲到的 telnet。流程很简单，创建 item-->配置 ip、用户、密码、端口、脚本->zabbix server telnet 目标 ip->执行制定脚本，脚本最后返回数据给 server。

目标：获取 linux 系统 15 分钟负载

telnet key

语法：

```
telnet.run[<unique short description>,<ip>,<port>,<encoding>]
```

<unique short description>：描述

<ip>：服务器 ip

<port>：服务器端口

<encoding>：编码,可为空

telnet 配置

请看《linux 下 telnet 安装与使用》

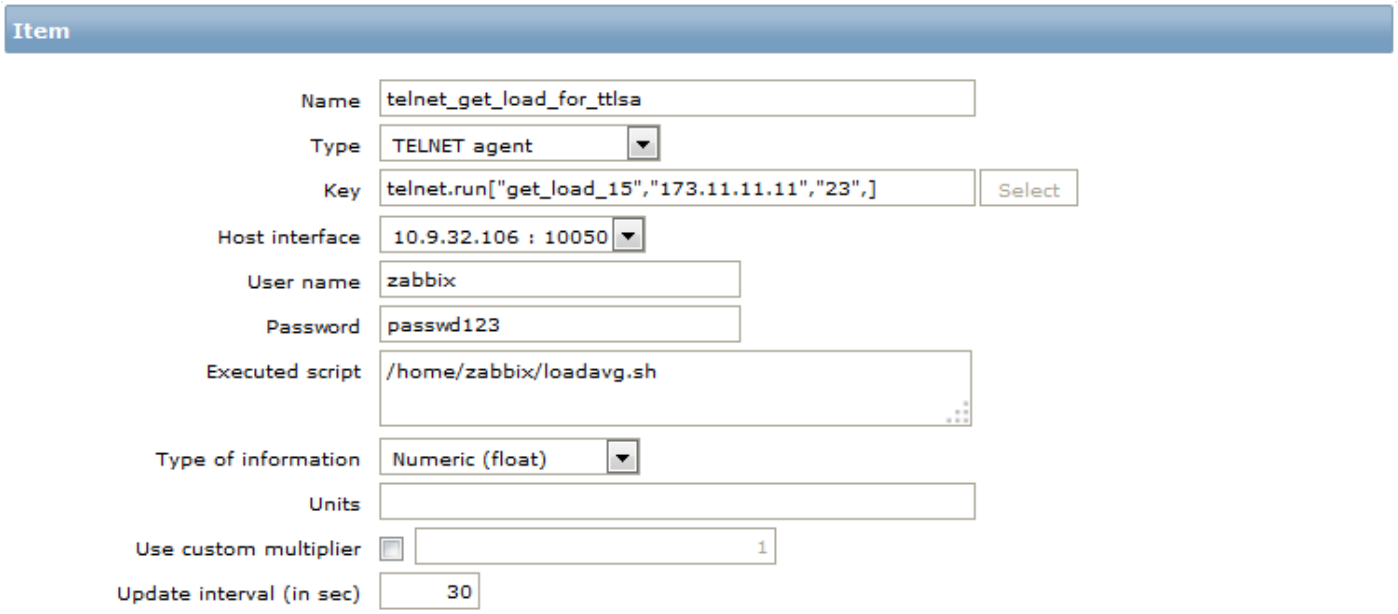
创建脚本

获取系统负载脚本 loadavg.sh

```
# cat /home/zabbix/loadavg.sh
#!/bin/bash
/bin/awk '{print $3}' /proc/loadavg
```

创建 item

configuration>>host>>你的主机>>item>>create item，如下：



属性说明：

user name：telnet 账号

Password：telnet 密码

获取到的结果如下

优缺点都很明显，只需要通过 telnet 就可以监控服务器，但是账号密码是明文配置在 item 中的，而且一旦网络不好，item 状态很容变为 unspport。一直相类似的还有 ssh 监控类型，想了解的同学情关注下一篇文章。再会~

zabbix 用户认证方式 内建、HTTP Basic、LDAP

公司大大小小众多系统，不同系统不同的账号密码，管理上相当复杂，后来慢慢出现了 SSO 等账号统一验证，其他 zabbix 也提供了类似的方法，或许有些公司便可以使用公司提供的账号来登录 zabbix 了。

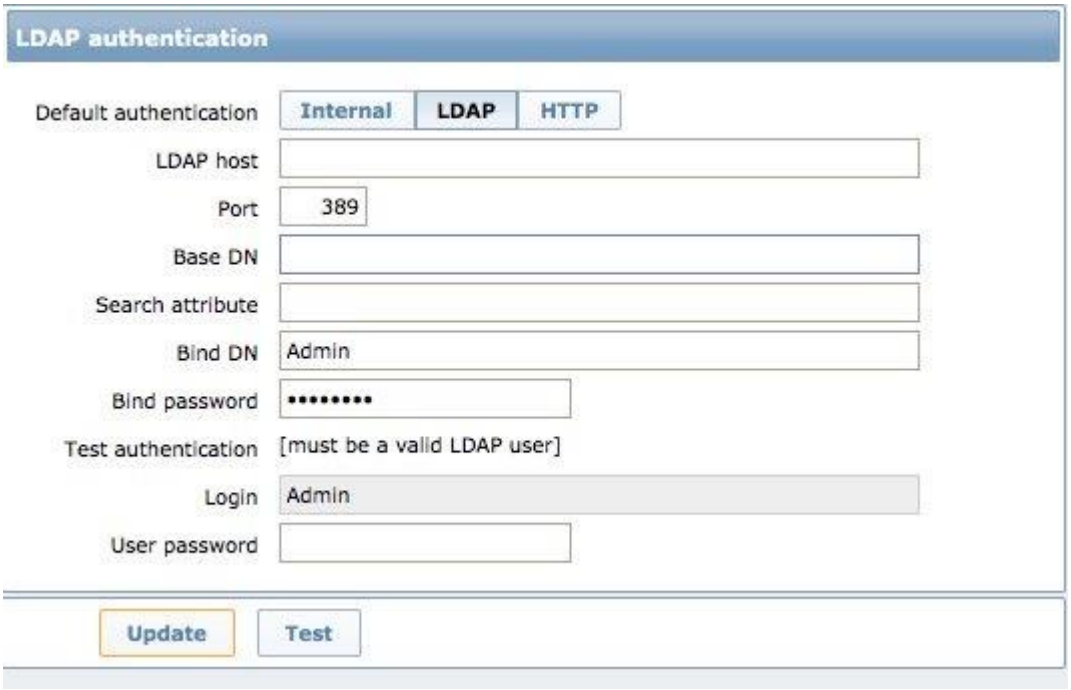
zabbix 提供了三种验证方式：

- 内建验证，也就是 zabbix 自带的账户系统
- HTTP BASIC，Apache 的基本验证，大家懂得
- LDAP 验证

zabbix 验证方式修改

Administration>> Authentication ，如下图：

默认是 Internal，系统内建认证



很多账户系统里面都用 LDAP、或者 window 域，我们后续文章会有实例，请关注我们！

基于 Apache http basic auth，估计用的人会相对较少，但是后续我们会做一个演示。

其实，我更希望他能提供一个最基础的 api 认证接口，我们实现她得接口，这样 zabbix 能更轻松的继承到各种不一样的账户系统中

zabbix 第三方认证之 http(nginx basic auth)

从《zabbix 用户认证方式 内建、HTTP Basic、LDAP（96）》我们了解到 zabbix 有三种认证方式，内建的不用多说，今天我们聊聊 HTTP Basic Auth 认证方式，我们将在实例中使用 nginx 来演示，Apache 也类似。

zabbix 认证配置

Administration>> Authentication，将 http authentication 改为 HTTP，保存即可，如下图：

接下来在 nginx 中创建 Admin 用户，或者创建 zabbix 已经存在的其他用户。

nginx 用户认证配置

nginx 配置如下

```
server{
    server_name   www.ttlsa.com;

     index index.html index.<a href="http://www.ttlsa.com/php/" title="php"target="_blank">php</a>;
     root /data/site/www.ttlsa.com;
          ..... 省略部分配置 .....
     location ~ ^/zabbix(/.*)$
     {
          auth_basic "nginx basic http test for ttlsa.com";
          auth_basic_user_file htpasswd;

          location ~ .php$ {
               fastcgi_pass 127.0.0.1:9000;
```

```
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param SCRIPT_NAME $fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
            include fastcgi_params;
        }
    }
        ..... 省略部分配置 .....
```

创建用户密码

```
# printf "Admin:$(openssl passwd -crypt 123456)\n" >>conf/htpasswd
# cat conf/htpasswd
Admin:xyJkVhXGAZ8tM
```

重启 nginx

```
# /usr/local/nginx-1.5.2/sbin/nginx -s reload
```

更多关于 nginx 认证的内容，请参考前面的文章：nginx 用户认证配置（ Basic HTTP authentication）

zabbix http 认证效果

像往常一样打开 zabbix 管理地址，此时会弹出账号密码框，账号是 Admin，密码 123456。假如哪天你取消了 http

认证，那么 zabbix 会使用 zabbix 系统内的密码。

输入完账号密码之后，直接进入了 zabbix 界面。

ZABBIX 各版本之间的兼容性

zabbix 更新很快，从 2009 年到现在已经更新多个版本，为了使用更多 zabbix 的新特性，随之而来的便是升级版本，zabbix 版本兼容性是必须优先考虑的一点

客户端 AGENT 兼容

zabbix1.x 到 zabbix2.x 的所有 agent 都兼容 zabbix server2.4：如果你升级 zabbix server，客户端是可以不做任何改变，除非你想使用 agent 的一些新特性。

Zabbix 代理（proxy）兼容

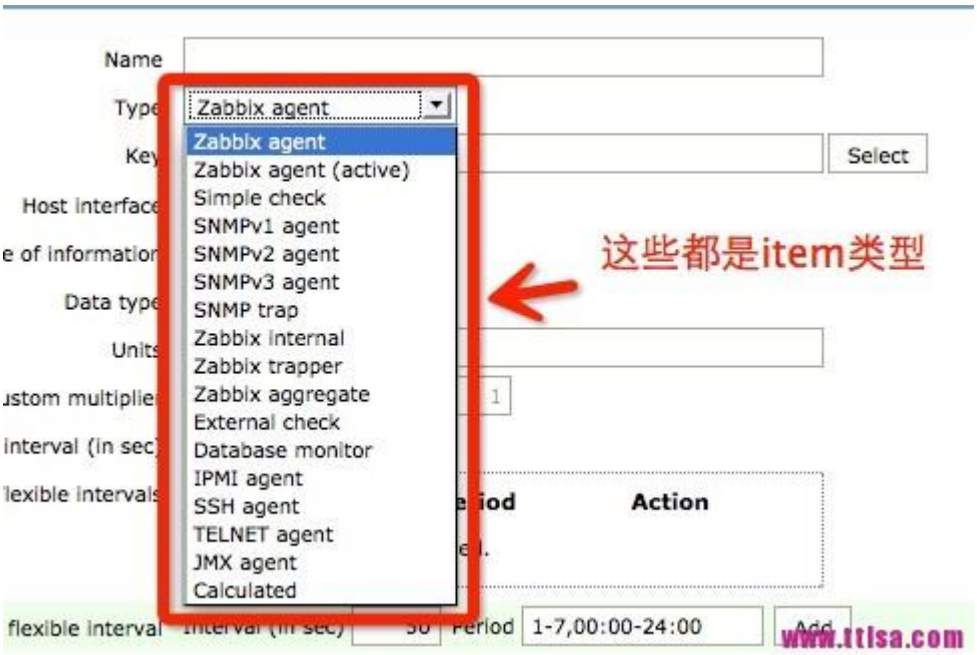
zabbix proxy 很挑剔，2.4 版本的 proxy 必须和 2.4 版本的 server 配套使用。其他 zabbix1.x 到 2.2 的 proxy 都不能与 2.4 的 server 配套。也就是说，如果你升级 zabbix server，那么 zabbix proxy 也要同步升级。

XML 文件兼容

zabbix 1.8、2.0、2.2 导出的 xml 文件都可以导入到 zabbix2.4 中。曾经在 zabbix 2.4 上辛辛苦苦做好了一个模板，结果无法导入到 2.2 版本中，想想就觉得苦。

zabbix 如何选择适合的监控类型

zabbix 提供十几种监控类型，包括：Zabbix agent, Simple checks, SNMP, Zabbix internal, IPMI, JMX monitoring 等等，那我们应该如何选择呢？凉白开在此给大家一一作一个说明



zabbix agent

zabbix 自带的客户端程序（被动模式），zabbix server 主动向它收集监控数据。agent 提供丰富的 key，包括不限于 cpu、内存、网络、磁盘、web 等等。如果你不介意或者系统支持安装此程序，那么他是首选的。需要注意的是，server 检索数据有超时限制，最大超时时间 30 秒，如果检索数据经常超过 30 秒，那么，不建议你使用主动模式的 agent，可以使用如下类型 agent active

zabbix agent（active）

也需要安装 agent（主动模式），和上一个相同。但是数据由 zabbix agent 主动提交至 zabbix server

参考文章：

simple check

基本的检测，可以检测网络、端口、fping 这些，功能很少并且无需安装客户端。

snmp check

snmp v1 check、snmp v2 check、snmp v3 check 的功能都是一样的。推荐如下场景：

客户基于安全考虑，不同意安装 agent

路由器、打印机等等设备无法安装，但是支持 snmp 协议

不喜欢频繁对 agent 升级

zabbix internal

zabbix 系统内部用，比如趋势数据记录数了、历史记录数量等等，日常业务监控用不上他。

zabbix trapper

也需要安装 agent（主动模式），你需要借助 bin/zabbix_sender 将数据提交至 zabbix server。如下情况适合使用：

- 检索数据时间较长
- 同一时间有大量的数据要提交，例如 redis info 信息，里面包含五六十项数据，通过 zabbix_sender 来一次性提交，显然比 agent 来取几十次要方便。

zabbix aggretage

aggregate checks 是一个聚合的检测，例如我想知道某个组的 host 负载平均值，硬盘剩余总量，或者某几台机器的这些数据，简单的说，这个方法就是用来了解一个整体水平，而不需要我们一台台看过去。这个方法的数据全部来之数据库，所以它不需要 agent。

external check

zabbix server 运行脚本或者二进制文件来执行外部检测，外部检测不需要在被监控端运行任何 agentd。

备注：请不要过度使用外部检测，这会严重降低 zabbix 系统性能

database monitor

zabbix 通过调用 ODBC 来获取数据库的数据以及数据库状态等等信息

IPMI agent

用于监控硬件设备，例如 Dell 或者 hp 服务器的主板温度、cpu 电压、盖子是不是被打开等等

SSH agent

zabbix 使用提供的 ssh 信息（服务器用户密码或者证书）登录服务器，执行指定的脚本来检索数据。如下人适合用

不会安装 agent、不想相撞 agent

不担心账号密码/证书放在 zabbix 里

Telnet agent

同上，Windows 不支持 ssh，可以使用 Telnet agent。

jmx agent

通过 jmx 监控 java jvm，比如 tomcat。目前有一个很大的不足，如果 tomcat 多实例，jmx agent 只能监控一个。如果是多实例，建议使用 agent + cmdline-jmxclient-0.10.3.jar

zabbix Calculated

计算类型，在几个 key 值之间做计算，例如 redis 自带的 info 命令可以监控 keyspace_hits 和 keyspace_misses 这两个值，但是 redis 的命中率不能直接获取，可以通过 zabbix 的 calculated 实现监控 redis 的命中率

通过如上的介绍，你知道如何选择了吗？

zabbix LTS 和标准发行版的区别

在 zabbix 官方下载页面，我们可以发现有一个 LTS 版本。如下图：分别为 Zabbix 2.4、Zabbix 2.2 LTS

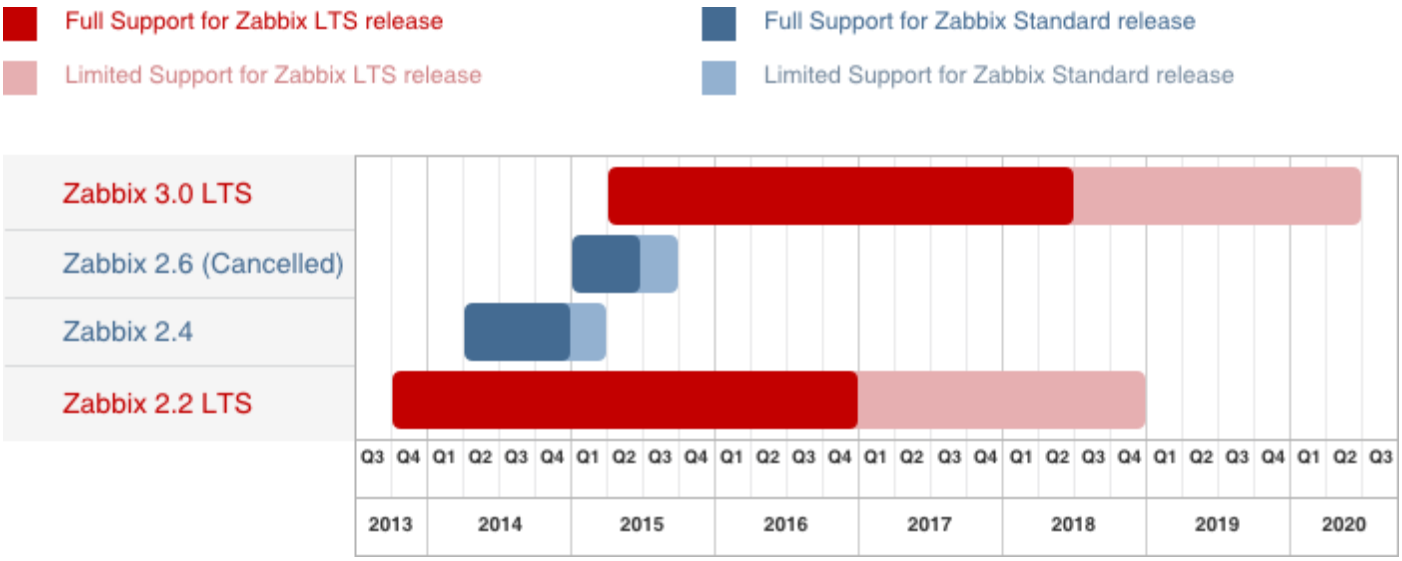
Zabbix Packages

Package	Distribution	Version	Architecture	Download	Documentation
Zabbix 2.4	Red Hat Enterprise Linux CentOS Oracle Linux	7	x86_64	Download	
		6	i386	Download	
			x86_64	Download	
		5	i386	Download	
			x86_64	Download	
	Debian	7 (Wheezy)	i386	Download	
			amd64	Download	
	Ubuntu	14.04 LTS (Trusty)	i386	Download	
			amd64	Download	
Zabbix 2.2 LTS	Red Hat Enterprise Linux CentOS Oracle Linux	7	x86_64	Download	
		6	i386	Download	
			x86_64	Download	
		5	i386	Download	
			x86_64	Download	
	Debian	6 (Squeeze), 7 (Wheezy)	i386	Download	
			amd64	Download	
	Ubuntu	12.04 LTS (Precise) 14.04 LTS (Trusty)	i386	Download	
			amd64	Download	

经常有人问起，他们俩的区别。

LTS 为 Long Term Support 的简写，zabbix 将为客户提供 5 年的支持服务。前三年完全支持与后两年有限制支持。前三年包括一般、关键、安全性问题解决，后两年包括关键、安全性问题解决。超出时间不提供技术支持服务。然而标准版，只提供 6+1 月支持。

上几个图，更好的解释 zabbix LTS 与标准发型版本的区别



目前 3.0 LTS 版本还在开发中，官方暂未给出 zabbix LTS 发行的具体日期。2.4 与 2.6 没有 LTS，所以只提供几个月的支持。即使这样，目前很大一部分同学在使用 zabbix 2.4

Life cycle of Zabbix Standard release

Production (approx. 6 months)	Release	Full Support (6 months)	Limited Support (1 month)

Month 1	Month 2	Month 3	Month 4	Month 5	Month 6	Month 7	Month 8	Month 9	Month 10	Month 11	Month 12	Month 13

Life cycle of Zabbix LTS release

Production (approx. 1,5 years)	Release	Full Support (3 years)	Limited Support (2 years)

Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7

如上可以看出 zabbix LTS 与 zabbix 标准发行版本的生命周期区别，普通版本 6 个月开发发行，支持 6 个月无限制支持，外加一个月有限制支持。然后 LTS 版本 1.5 年开发发行，3 年无限制支持，2 年有限制支持。

Stable release	Release date	End of Full Support	End of Limited Support
Zabbix 2.2 LTS	November, 2013	November, 2016	November, 2018
Zabbix 2.4	September, 2014	May, 2015	June, 2015
Zabbix 3.0 LTS	TBA	May, 2018	May, 2020

Note: Zabbix 3.0 release is still being actively developed, and the planned release date is subject to change without notice.

如上为当前主流 zabbix 版本生命周期表。

总结

简单来说 zabbix LTS 与 zabbix 标准版本区别在开发周期、支持周期，其他都没有什么区别。希望更新新版本的用户肯定不愿意使用 LTS。

zabbix 主动、被动检测的详细过程与区别

zabbix agent 检测分为主动（agent active）和被动（agent）两种形式，主动与被动的说法均是相对于 agent 来讨论的。简单说明一下主动与被动的区别如下：

- 主动：agent 请求 server 获取主动的监控项列表，并主动将监控项内需要检测的数据提交给 server/proxy
- 被动：server 向 agent 请求获取监控项的数据，agent 返回数据。

zabbix agent 通信过程中的协议均基于 json 格式，格式如下：

```
- "ZBXD\x01" (5 bytes)
 - data length (8 bytes). 1 will be formatted as 01/00/00/00/00/00/00/00 (eight bytes in HEX, 64 bit number)
```

备注：为了避免内存耗尽，server 限制每个通信连接最多使用 128MB 内存。

被动检测

server 发起如下请求

```
<item key>\n
```

Aegent 返回如下响应数据

```
<HEADER><DATALEN><DATA>[\o<ERROR>]
```

supported items 通信过程

- Server 打开一个 TCP 连接
- Server 发送请求 agent.ping\n
- Agent 接收到请求并且响应<HEADER><DATALEN>1
- Server 处理接收到的数据 1
- 关闭 TCP 连接
-

not supported items 通信过程

- Server 打开一个 TCP 连接

- Server 发送请求 vfs.fs.size[/nono]\n
- Agent 接收请求并且返回响应数据 <HEADER><DATALEN>ZBX_NOTSUPPORTED\oCannot obtain filesystem information: [2] No such file or directory
- Server 接收并处理数据，将 item 的状态改为“ not supported ”
- 关闭 TCP 连接

主动检测

如前面所说，zabbix 首先向 ServerActive 配置的 IP 请求获取 active items，获取并提交 active tiems 数据值 server 或者 proxy。很多同学会提出疑问：zabbix 多久获取一次 active items？它会根据配置文件中的 RefreshActiveChecks 的频率进行，如果获取失败，那么将会在 60 秒之后重试

获取 ACTIVE ITEMS 列表

Agent 请求

```
<HEADER><DATALEN>{
    "request":"active checks",
    "host":"<hostname>"
}
```

Server 响应列表

```
<HEADER><DATALEN>{
    "response":"success",
    "data":[
        {
            "key":"log[/home/zabbix/logs/zabbix_agentd.log]",
            "delay":30,
            "lastlogsize":0,
            "mtime":0
        },
        {
            "key":"agent.version",
```

```
            "delay":600,
            "lastlogsize":0,
            "mtime":0
        },
        {
            "key":"vfs.fs.size[/nono]",
            "delay":600,
            "lastlogsize":0,
            "mtime":0
        }
    ]
}
```

备注：获取到的 items 列表中的所有 item 属性 key, delay, lastlogsize ,mtime 都必须存在，获取列表的通信过程如下：

- Agent 打开 TCP 连接（主动检测变成 Agent 打开）
- Agent 请求 items 检测列表
- Server 返回 items 列表
- Agent 处理响应
- 关闭 TCP 连接
- Agent 开始收集数据
- 提交 active items 数据
- Agent 发送请求

```
<HEADER><DATALEN>{
    "request":"agent data",
    "data":[
        {
            "host":"<hostname>",
            "key":"agent.version",
```

```
                "value":"2.4.0",
                "clock":1400675595,
                "ns":76808644
            },
            {
                "host":"<hostname>",
                "key":"log[/home/zabbix/logs/zabbix_agentd.log]",
                "lastlogsize":112,
                "value":" 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision 50000).",
                "clock":1400675595,
                "ns":77053975
            },
            {
                "host":"<hostname>",
                "key":"vfs.fs.size[/nono]",
                "state":1,
                "value":"Cannot obtain filesystem information: [2] No such file or directory",
                "clock":1400675595,
                "ns":78154128
            }
        ],
        "clock": 1400675595,
        "ns": 78211329
}
```

Server 响应数据

```
{
 "response":"success",
```

```
"info":"processed: 3; failed: 0; total: 3; seconds spent: 0.003534"
}
```

备注：如果有些数据提交失败，比如 host、item 被删除或者禁用，agent 不会尝试从新提交

主动检测提交数据过程如下：

- Agent 建立 TCP 连接
- Agent 提交 items 列表收集的数据
- Server 处理数据，并返回响应状态
- 关闭 TCP 连接

zabbix 值缓存(value cache)说明

在 zabbix-2.2 版本之前，zabbix 计算 trigger 与 calculated/aggregate 值都是直接通过 sql 语句查询并处理出来的结果，为了提高这块的性能与效率，zabbix 引入了 value cache，相关配置请参考：《zabbix_server.conf 配置文件详解》。

zabbix 获取到的 item 数据，会暂时保存到 cache 中，等达到一定阀值之后，将会把数据存储到数据库中。此时 trigger、aggregate 等等功能直接从 cache 中抓取数据使用，如果 cache 中没有所有的数据，那么再从数据库中读取，无形中降低了不少的数据库的压力。

zabbix 还提供内部 item，可用于监控 zabbix value chache 的使用情况，如下：

```
zabbix[vcache,buffer,]
zabbix[vcache,cache,]
```

了解 value cache 有主语大家配置，它听过的监控 item 一般用的较少。

第五章：zabbix SNMP

zabbix snmp 类型

概述

如果我们需要监控打印机、路由器、UPS 等设备,肯定不能使用 zabbix agentd,因为他们不能安装软件的,还好他们一般都支持 SNMP 协议,这样我可以使用 SNMP 来监控他们.如果你希望使用 SNMP agent 来获取这些设备的信息，那么在安装 zabbix server 的时候你需要增加 snmp 的支持.

备注：SNMP 检查基于 UDP 协议

注意事项

如果监控基于 SNMPv3 协议的设备,确保 msgAuthoritativeEngineID（通常叫做 snmpEngineID 或“Engine ID”）是唯一的. 以前 SNMPv3 协议只支持 MD5 和 DES 加密,从 zabbix 2.2 开始支持 SHA 与 AES 加密协议.

配置 SNMP 监控

使用 SNMP 来监视设备,需要完成以下步骤

1. 创建主机

创建一个使用 SNMP 接口的主机（请参考 zabbix 创建主机）,创建主机过程中你可以选择相应的模板,路由器、交换机这些设备的监控 itme,zabbix 已经默认自带了

2. 找出你想监控的 SNMP 字符串（或者 OID）

如果你想获取 SNMP 字符串,你可以使用命令 snmpwalk 来实现,当然在安装 zabbix 的时候你需要增加 snmp 的支持

```
# snmpwalk -v 2c -c public .
```

2c 表示 SNMP 标准版本,snmp 推出了 v1,v2,v3 版本,你也可以 写成 1,表示使用 1 版本.上面的命令会获取到一个 SNMP 的列表,包含键值，默认情况下我们 snmp 不加密，使用 public 作为共同体即可,这些列表中 有你需要的一些监控数据.但是线上环境我们不可能获取所有的键值,如果需要获取特定的数据，例如像获取交换机第三个端口的入口流量,需要使用 IF-MIB::ifInOctets.3 字符串，如下：

```
IF-MIB::ifInOctets.3 = Counter32: 3409739121
```

然后使用 snmpget 命令来获取 IF-MIB::ifInOctets.3 的 OID。命令如下

```
# snmpget -v 2c -c public -On 10.62.1.22 IF-MIB::ifInOctets.3
```

最后一个数字 3 表示端口 3,如果你想监控端口 23 那就写上 23. 请参考: Dynamic indexes.
会输出如下值：

```
.1.3.6.1.2.1.2.2.1.10.3 = Counter32: 3472126941
```

同理，OID 的最后一个数字 3 表示端口号，要特别注意的是 3COM 的端口号 1 是 101，3 是是 103，但是 cisco 还是不变，1 号还是数字 1.

创建监控项 item

回 到 zabbix 的主机列表中，在你需要配置的主机那行，点击监控项 item，在右上角点击“create item”，和普通的监控项创建方法差不多，仅有几个地方不同，type 选择 SNMP v2 或者 v1，v3，一般选择 v2，snmp community 使用默认的 public，port 不填写默认为 161. SNMP OID 写上 OID，例如”.1.3.6.1.2.1.2.2.1.10.3″。然后其他内容和其他 item type 一样，然后保存即可。最后去看看是否获取到了数据。

简单实例

参数	值
Community	Public（或者你定义的）
OID	1.2.3.45.6.7.8.0 (或者.1.2.3.45.6.7.8.0)

Key	例如 sysName 等，比 OID 好记

OID 可以写数字也可以使用字符串. 然而，一些项目中，字符串 OID 需要转为数字格式，我们可以使用命令 snmpget 来获取，如下命令:

```
# snmpget -On localhost public enterprises.ucdavis.memory.memTotalSwap.0
```

其他

如果想使用 snmp 来监控，从源码编译安装 zabbix，一定要加上编译参数 –with-net-snmp，从 zabbix 2.2.3 开始 server 和 proxy 都支持使用单个请求获取多个值

snmp 安装配置

snmp 在监控这个行当里面有着举足轻重的地位，一直想写 zabbix 使用 snmp 监控，由于最近懒散了一直没写，也有人提到 ttlsa 能否写 snmp 的监控，那就写吧，前面有两篇文章已经做好了铺垫《SNMP OID 列表 监控需要用到的 OID》《zabbix snmp 监控类型》，今天是最后一篇铺垫，然后下面一篇便是 zabbix 使用 snmp 监控的实例，好了，不说没用的，看看 snmp 的安装配置。

yum 安装 snmp

```
# yum list net-snmp* -y
```

snmp 配置

配置 snmpd.conf

```
# vim /etc/snmp/snmpd.conf
view systemview included .1.3.6.1.2.1.25.1.1 //找到这行,增加下面配置
view systemview included .1 // 这个是新增加的
proc mountd // 找到这些配置，把注释去掉
proc ntalkd 4
proc sendmail 10 1
disk / 10000
load 12 14 14
```

启动 snmpd

```
# service snmpd start
```

通过 snmp 获取数据

需要通过 snmp 获取到数据，首先我们需要对应的 OID，请参考《SNMP OID 列表 监控需要用到的 OID》

获取主机名

```
# snmpwalk -c public -v 2c 173.219.255.122 sysName // 使用名称
 SNMPv2-MIB::sysName.0 = STRING: li519-232
```

```
# snmpwalk -c public -v 2c 173.219.255.122 .1.3.6.1.2.1.1.5.0 //  使用 OID
 SNMPv2-MIB::sysName.0 = STRING: li519-232
```

通过如上两种方式均可获取到数据，如上获取到得数据都是 li519-232

获取服务器剩余内存

```
# snmpwalk -c public -v 2c 173.219.255.122 .1.3.6.1.4.1.2021.4.11.0
 UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 560872 kB
# snmpwalk -c public -v 2c 173.219.255.122 memTotalFree
 UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 559176 kB
```

通过两种方式获取到服务器剩余内存，因为服务器资源使用量都实时变动的，所以两次获取的数值不同.

snmp v3 的安全配置 snmp 认证与加密配置

如果你觉得你得服务器信息暴露在外面没关系，或者说服务器安全限制的很严格，不需要对 snmp 做一道验证，那么你可以打住，否则继续往下看。snmp v2 配置请参考上一节《snmp 安装配置 zabbix snmp 监控准备(52)》

增加 snmp v3 用户

```
# service snmpd stop
# net-snmp-config --create-snmpv3-user -ro -a ttlsapwd    -x DES -X ttlsades ttlsa
 adding the following line to /var/lib/net-snmp/snmpd.conf:
 createUser ttlsa MD5 "ttlsapwd" DES ttlsades
 adding the following line to /etc/snmp/snmpd.conf:
 rouser ttlsa
```

参数说明

参数	描述
ttlsa	用户名
ttlsapwd	密码，密码必须大于 8 个字符
DES	加密方式，这边支持 AES、DES 两种
ttlsades	DES 口令，必须大于 8 位

备注：增加用户的时候，snmp 必须关闭，否则有如下报错

```
Apparently at least one snmpd demon is already running.
 You must stop them in order to use this command.
```

启动 snmpd v3

```
# service snmpd start
```

使用 snmp v3 获取信息

snmp v3 安全级别有三种，分别为 noAuthNoPriv（不认证也不加密）、authNoPriv（认证但是不加密）、authPriv（既认证又加密）

服务器剩余内存

noAuthNoPriv 安全级别

```
# snmpwalk    -v 3 -l authPriv    173.219.255.122 .1.3.6.1.4.1.2021.4.11.0
 UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 560872 kB
```

authNoPriv 安全级别

```
# snmpwalk -v 3 -u ttlsa -a MD5 -A ttlsapwd -l authNoPriv freevpn.ttlsa.com sysName
 SNMPv2-MIB::sysName.0 = STRING: li519-232
```

authPriv 安全级别

```
# snmpwalk    -v 3 -u ttlsa -a MD5 -A ttlsapwd -x DES -X ttlsades -l authPriv    173.219.255.122 .1.3.6.1.4.1.2021.4.11.0
 UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 560872 kB
```

zabbix 如何使用 SNMP 获取数据 SNMP 监控实例

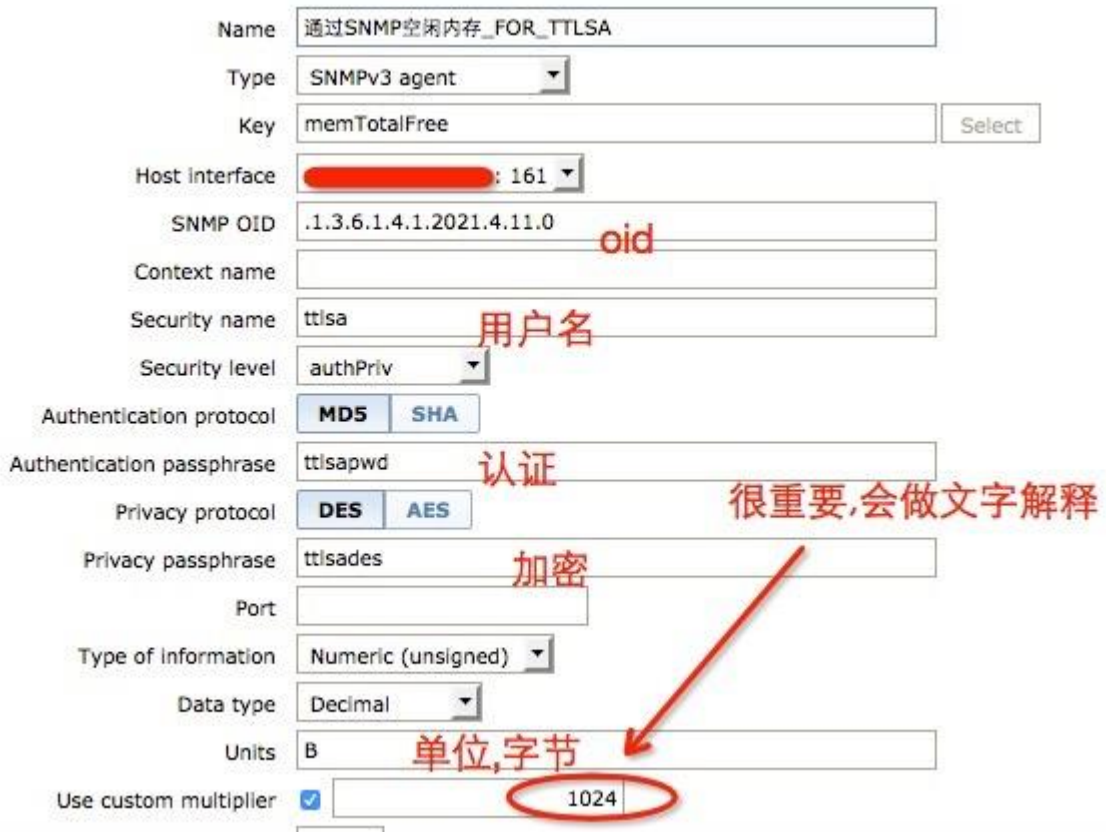
前面几篇文章已经对 zabbix snmp 监控类型以及如何安装配置 snmp 做了几篇讲解，那么接下来我们开始来一个使用 zabbix 监控服务器内存使用情况的实例，大家可以举一反三，可以使用 zabbix+snmp 一一监控 cpu 使用率、硬盘使用率、负载情况等等。

zabbix 增加 snmp 接口

configuration（配置）->Hosts(主题)->您需要配置的主机，找到“SNMP interfaces”，配置类似如下：

创建 SNMP 监控项

configuration（配置）->Hosts(主题)->您需要配置的主机->items，点击 create items,配置如下：



图片里面的账号、口令、oid 我就不多做说明了，特别说一下单位 B 和倍数 1024，更多的单位符号请看文章最后的参考。流量的单位是字节，也就是大 B，那么为什么下面还有一个 1024 呢？因为通过 snmp 获取的数据是 kB，比

如通过 snmp 得到 1024kB，zabbix 以为是 1024，那么数据不准了，所以我们需要额外给它乘以 1024，这样就准确了？不知道能否明白意思？然后到最新数据里面查看 zabbix 是否获取到了 snmp 数据。monitor->last data->找到你得主机以及相应的 item，如下：

Name	Last check	Last value	Change
- other - (2 Items)			
通过SNMP空闲内存_FOR_TTLSA	01 Dec 2014 21:43:17	701.09 MB	+20 KB
通过SNMP获取主机名_FOR_TTLSA	01 Dec 2014 21:43:16	li519-232.members.linode.com	-

zabbix 使用 snmp 说明

大多数设备都支持 snmp，例如路由器、交换机、打印机等等，我想以后的智能家居也会有 snmp 支持，使用 zabbix 监控家里的电视机、冰箱、洗衣机、电饭煲，很有趣。

zabbix snmp 自定义 OID nginx 监控实例

为什么要自定义 OID？

前面的文章我们已经讲过 zabbix 如何使用 snmp 监控服务器，但是他有一个很明显的局限性：只能监控定义好的 OID 项目，假如我们想知道 nginx 进程是否在运行？在没有 zabbix agent 的情况下，我们该怎么做呢？接下来就用这个实例来讲解自定义 OID

确认 SNMP OID 是否存在

首先我们需要找一个 oid 是否被系统暂用，比如.1.3.6.1.4.1.2021.5000

```
# snmpwalk -v 2c -c public www.ttlsa.com .1.3.6.1.4.1.2021.5000
 UCD-SNMP-MIB::ucdavis.5000 = No Such Object available on this agent at this OID
```

如上说明不存在

增加自定 SNMP OID

编写脚本

```
# cat /root/scripts/check_nginx.sh
#!/bin/bash
nginxNum=`/bin/ps aux | /bin/grep nginx | wc -l`
echo $nginxNum
```

修改配置

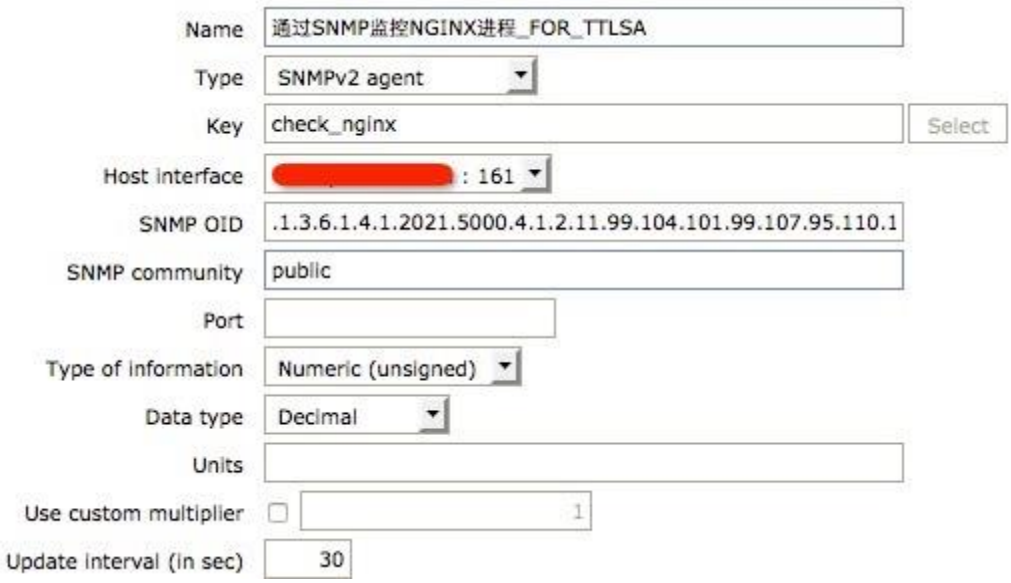
```
# vim /etc/snmp/snmpd.conf
 extend .1.3.6.1.4.1.2021.5000    check_nginx /root/scripts/check_nginx.sh //  增加这一行
```

获取 snmp 信息

以下获取自定义的 oid 的所有数据，第一行便是我们需要获取的数据，那么在 zabbix 中写 oid .1.3.6.1.4.1.2021.5000.4.1.2.11.99.104.101.99.107.95.110.103.105.110.120.1

```
# snmpwalk -v 1 -c public 173.219.255.122 .1.3.6.1.4.1.2021.5000
UCD-SNMP-MIB::ucdavis.5000.1.0 = INTEGER: 1
UCD-SNMP-MIB::ucdavis.5000.2.1.2.11.99.104.101.99.107.95.110.103.105.110.120 = STRING:
"/root/scripts/check_nginx.sh"
UCD-SNMP-MIB::ucdavis.5000.2.1.3.11.99.104.101.99.107.95.110.103.105.110.120 = ""
UCD-SNMP-MIB::ucdavis.5000.2.1.4.11.99.104.101.99.107.95.110.103.105.110.120 = ""
UCD-SNMP-MIB::ucdavis.5000.2.1.5.11.99.104.101.99.107.95.110.103.105.110.120 = INTEGER: 5
UCD-SNMP-MIB::ucdavis.5000.2.1.6.11.99.104.101.99.107.95.110.103.105.110.120 = INTEGER: 1
UCD-SNMP-MIB::ucdavis.5000.2.1.7.11.99.104.101.99.107.95.110.103.105.110.120 = INTEGER: 1
UCD-SNMP-MIB::ucdavis.5000.2.1.20.11.99.104.101.99.107.95.110.103.105.110.120 = INTEGER: 4
UCD-SNMP-MIB::ucdavis.5000.2.1.21.11.99.104.101.99.107.95.110.103.105.110.120 = INTEGER: 1
UCD-SNMP-MIB::ucdavis.5000.3.1.1.11.99.104.101.99.107.95.110.103.105.110.120 = STRING: "6"
UCD-SNMP-MIB::ucdavis.5000.3.1.2.11.99.104.101.99.107.95.110.103.105.110.120 = STRING: "6"
UCD-SNMP-MIB::ucdavis.5000.3.1.3.11.99.104.101.99.107.95.110.103.105.110.120 = INTEGER: 1
UCD-SNMP-MIB::ucdavis.5000.3.1.4.11.99.104.101.99.107.95.110.103.105.110.120 = INTEGER: 0
UCD-SNMP-MIB::ucdavis.5000.4.1.2.11.99.104.101.99.107.95.110.103.105.110.120.1 = STRING: "6"
```

创建 snmp item



获取最新数据

Name	Last check	Last value	Change
TTLSA (27 Items)			
- other - (3 Items)			
通过SNMP监控NGINX进程_FOR_TTLSA	07 Dec 2014 17:58:47	6	-
通过SNMP空闲内存_FOR_TTLSA	07 Dec 2014 17:58:47	671 MB	-1.7 MB
通过SNMP获取主机名_FOR_TTLSA	07 Dec 2014 17:58:45	li519-232.members.linode.com	-

接下来创建触发器以及报警，我就不多说了，大家可以参考《zabbix 触发器》

第六章：zabbix 通知媒介

zabbix 报警媒介介绍

zabbix 触发器到了要发送通知的情况下，需要一个中间介质来接收并传递它的消息给运维们，以往用 nagios，通常用脚本发送邮件或者发送飞信来达到报警。这个脚本实际上就是一个媒介了。

E-mail

使用 sendmail 发送邮件，从这边出去的邮件基本是垃圾邮件，我一直不喜欢用

SMS

需要短信设备，没有，一直都没用过这东西

Jabber

Jabber 有第三方插件，能让 Jabber 用户和 MSN、YahooMessager、ICQ 等 IM 用户相互通讯。因为 Google 遵从 Jabber 协议，并且 Google 已经将 Gtalk 的服务器开放给了其它的 Jabber 服务器。所以 PSI、Giam 等 Jabber 客户端软件支持 GTalk 用户登陆。国内没啥人用

Ez Texting

给用户手机发短信，貌似只支持美国和加拿大

Custom alertscripts

自定义脚本，把信息传递给脚本，我们在脚本里使用 sendEmail（不要和 sendmail 搞混了）、飞信发短信、调用短信接口发送短信等等。

zabbix 报警媒介：email

报警信息将会使用系统自带的 sendmail 发送，配置比较简单

配置媒介 Email

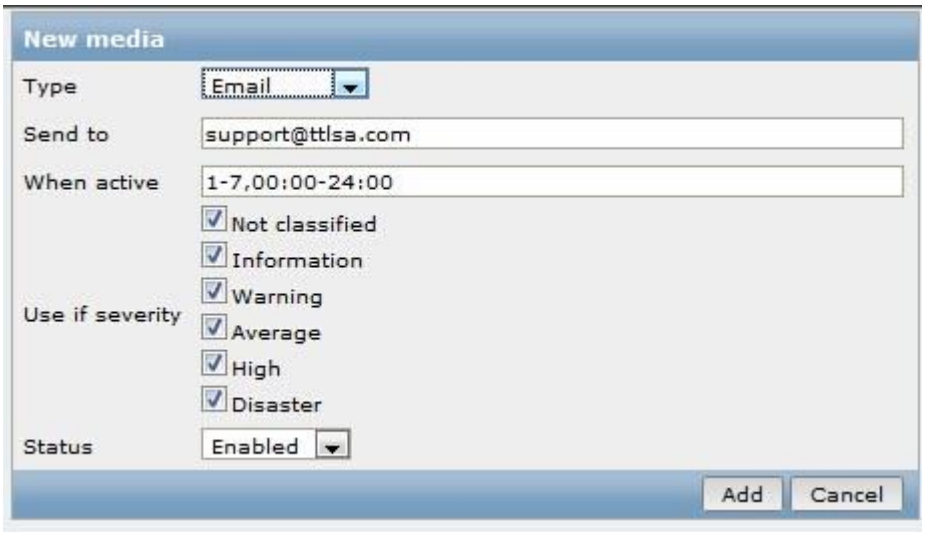
Administration（管理）→Media types->点击 Create media type（创建媒介）

选项	描述
Name	媒介名称，看着起名
Type	选择 Email
SMTP server	SMTP 服务器
SMTP helo	SMTP helo 值，通常情况下是顶级域名
SMTP email	这个邮件地址会显示到收件人的 From 里， **可用邮箱地址** zabbix@company.com (只包含邮箱地址，不需要尖括号括起来) Zabbix HQ <zabbix@company.com> (显示名和邮箱地址，邮箱地址使用尖括号) ∑Ω-monitoring <zabbix@company.com> (显示名称为 UTF8 格式) **不可用的邮箱地址** Zabbix HQ zabbix@company.com (需要尖括号) “Zabbix\@\<H(comment)Q\>” <zabbix@company.com>不支持转义

使用媒介

定义好了媒介之后，我们需要把这媒介指定给用户。

Administration（管理）->Users->打开用户配置->media type 里面添加刚增加的媒介



选项	描述
Type	选择媒介名称，此处选 Email
Send to	发邮件给谁，例如 support@ttlsa.com，也可以使用显示名
When active	发送时间，只有在这个时间段内才会发邮件
Use if severity	发送邮件的触发器级别
Status	当前媒介状态 Enabled – 使用中. Disabled – 禁用中.

zabbix 报警媒介：SMS

服务器安装串口 GSM 短信猫之后，zabbix 可以使用它来发送短信通知给管理员，如下注意事项：

- 串行设备速度要与 GSM 猫相匹配（linux 下默认为/dev/ttyS0），zabbix 无法设置设置串行设备速率
- zabbix 有对串行设备的读写全乡，可以使用 ls -l /dev/ttyS0 查看设备权限
- 请禁用你 GSM 手机卡的 PIN 码

zabbix 测试过的 GSM 猫如下

- Siemens MC35
- Teltonika ModemCOM/G10

配置 SMS

点击 Administration（管理）->Media types->媒介类型选择 SMS，和 email 的配置方法是一样的，直接上参数吧。

选项	描述
Description	媒介名称
Type	类型
GSM modem	SM modem 串行设备，默认为：/dev/ttyS0

SMS 使用

Administration->Users->打开用户配置->media type 里面添加刚增加的媒介

选项	描述
Type	选择媒介名称，此处选 SMS
Send to	发短信给哪个手机号码
When active	发送时间，只有在这个时间段内才会发短信
Use if severity	发送短信的触发器级别
Status	当前媒介状态 Enabled – 使用中. Disabled – 禁用中.

用短信猫发送短信的公司都很有钱，我从来只用邮件~

zabbix 报警媒介：Jabber

Jabber 有第三方插件，能让 Jabber 用户和 MSN、YahooMessager、ICQ 等 IM 用户相互通讯。因为 Google 遵从 Jabber 协议，并且 Google 已经将 Gtalk 的服务器开放给了其它的 Jabber 服务器。所以 PSI、Giam 等 Jabber 客户端软件支持 GTalk 用户登陆。

jabberXMPP（可扩展消息处理现场协议）是基于可扩展标记语言（标准通用标记语言下的一个子集、外语缩写：XML）的协议，它用于即时消息（IM）以及在线现场探测。它在促进服务器之间的准即时操作。这个协议可能最终允许因特网用户向因特网上的其他任何人发送即时消息，即使其操作系统和浏览器不同。XMPP 的技术来自于 Jabber，其实它是 Jabber 的核心协定，所以 XMPP 有时被误称为 Jabber 协议。Jabber 是一个基于 XMPP 协议的 IM 应用，除 Jabber 之外，XMPP 还支持很多应用。

IEEE XMPP 工作组（一个工程师和程序员联盟）正在改编 XMPP 以用作互联网工程任务组（IETF）技术。XMPP 最终有望使用鉴定、访问控制、高级隐私、逐跳加密、端端加密以及与其它协议的相容等应用来支持 IM。

zabbix 报警媒介：Ez Texting

Ez Texting 是 zabbix 的技术合作伙伴，主要提供短信服务，用手机注册账号，便可以使用它来发送短信了，不过他只支持美国和加拿大的手机号码，并且应该是收费的。没有美国/加拿大手机号码的朋友请绕行，先了解的请继续往下看。

配置

点击 Administration（管理）→Media types（媒介类型）->点击创建

参数说明：

选项	描述
Name	媒介名称，看着起名
Type	选择 Ez Texting，如果你没有账号，你可以到 https://app.eztexting.com 注册（没有手机号码绕行）
username	你的 ez 账号
Password	Ez 密码
Message text limit	文本消息限制 USA (160 characters)，美国一条短信支持 160 个字符 Canada (136 characters)，加拿大一条短信支持 136 个字符

使用

定义好了媒介之后，我们需要把这媒介指定给用户。Administration->Users->打开用户配置->media type 里面添加刚

增加的媒介

参数说明

选项	描述
Type	选择媒介名称，此处选 Ez Texting
Send to	发短信给谁，填手机号码
When active	发送时间，只有在这个时间段内才会发短信
Use if severity	发送短信的触发器级别
Status	当前媒介状态 Enabled - 使用中. Disabled - 禁用中.

zabbix 报警媒介： Custom alertscripts

老板抠门不给买 SMS 短信猫，投错胎导致没有美国/加拿大手机号码，根本搞不清楚 jabber 是个什么玩意儿，sendmail 又不靠谱，那都不是事，想要轻轻松松报警，那么用上自定义脚本媒介。zabbix 会将信息传递给脚本，接下来你在脚本里面随意处理，一共会传递三个参数,按顺序接受也就是$1,$2,$3 了，为了方便记忆，一般分别给他们赋值到 To\Subject\body.

配置 AlertScriptsPath

在 server 的配置文件中配置，这是用来定义脚本目录，这样一来 zabbix 就能找到脚本了

```
# cat /usr/local/zabbix-2.2.1/etc/zabbix_server.conf | grep AlertScriptsPath
### Option: AlertScriptsPath
    AlertScriptsPath=/usr/local/zabbix-2.2.1/alertscripts
# mkdir /usr/local/zabbix-2.2.1/alertscripts
```

创建发邮件脚本

```
# cat /usr/local/zabbix-2.2.1/alertscripts/mail.sh
#!/bin/sh
to=$1
subject=$2
body=$3
/usr/local/bin/sendEmail -f support@ttlsa.com -t "$to" -s smtp.ttlsa.com -u "$subject" -o message-content-type=html -o
message-charset=utf8 -xu support@ttlsa.com -xp 123456 -m "$body" 2>>/tmp/22.log
# chmod a+x /usr/local/zabbix-2.2.1/alertscripts/mail.sh
```

脚本里面使用 sendEmail 发送邮件，sendEmail 的用法请点击《使用 sendEmail 发送邮件》，不一定非要发送邮件，也可以发飞信或者调用短信平台接口

配置自定义脚本媒介

Administration->Media types->创建

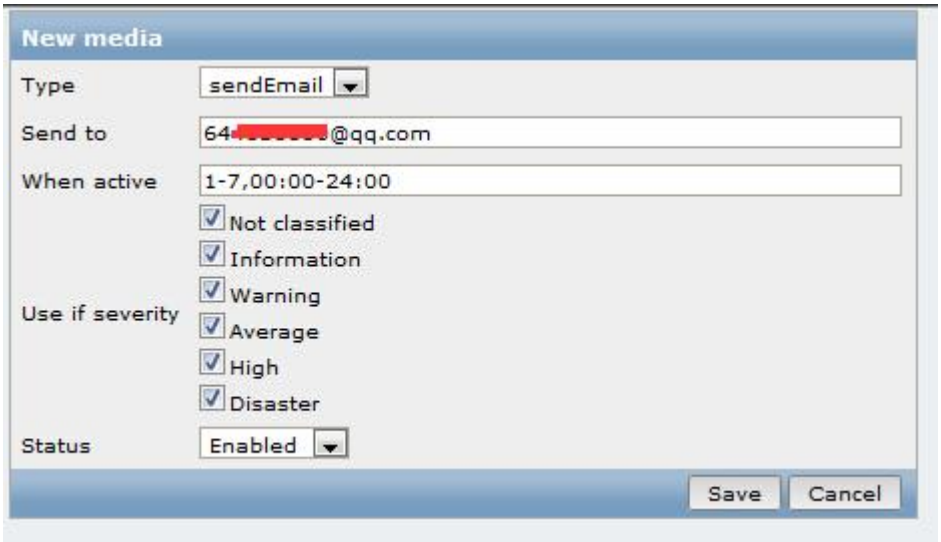
参数说明

选项	描述
Description	媒介名称，看着起名,这边叫 sendEmail
Type	选择 custom scripts
Script name	脚本名称，这边写 mail.sh，只要写名称就行了，不要写绝对路径

使用自定义脚本媒介

定义好了媒介之后，我们需要把这媒介指定给用户。

Administration->Users->打开用户配置->media type 里面添加刚增加的媒介



参数说明

选项	描述

Type	选择媒介名称，此处选 sendEmail
Send to	发邮件给谁，例如 support@ttlsa.com
When active	发送时间，只有在这个时间段内才会发邮件
Use if severity	发送邮件的触发器级别
Status	当前媒介状态 Enabled – 使用中. Disabled – 禁用中.

第七章：zabbix 模板

zabbix 模板介绍

zabbix 模板是做什么的?

我们不提概念，通过一个案例来说明他是干什么的。

王小明是某公司系统管理员，负责 100 台 Linux 服务器，还有几台 windows 服务器。他选择了 zabbix 作为监控服务器基本性能，如 cpu、内存、硬盘、网络这些基本的东西，主管要求他一天内搞定。于是他开始做，第一台服务器添加 cpu、内存、硬盘、网络的 items，然后第二台在一个个添加，添加了两天两夜，他一夜没合眼，突然看到 zabbix 模板的功能，卒了。

这是一个忧伤的故事，没文化害死人是真的。如果他一开始创建一个模板，然后每个服务器套用/链接这个模板，那么只要在创建主机的过程中在 link（套用/链接）这个模板，一个服务器就完成了。也不会发生这么悲伤的事情….

平时工作中，我们需要监控 web、mysql、redis、nginx 这些服务器，众多服务器的业务都是一样的，所以我们只要事先创建好模板，然后所有服务器链接这个模板即可，如果后续有修改、新增功能，只需要修改模板即可。

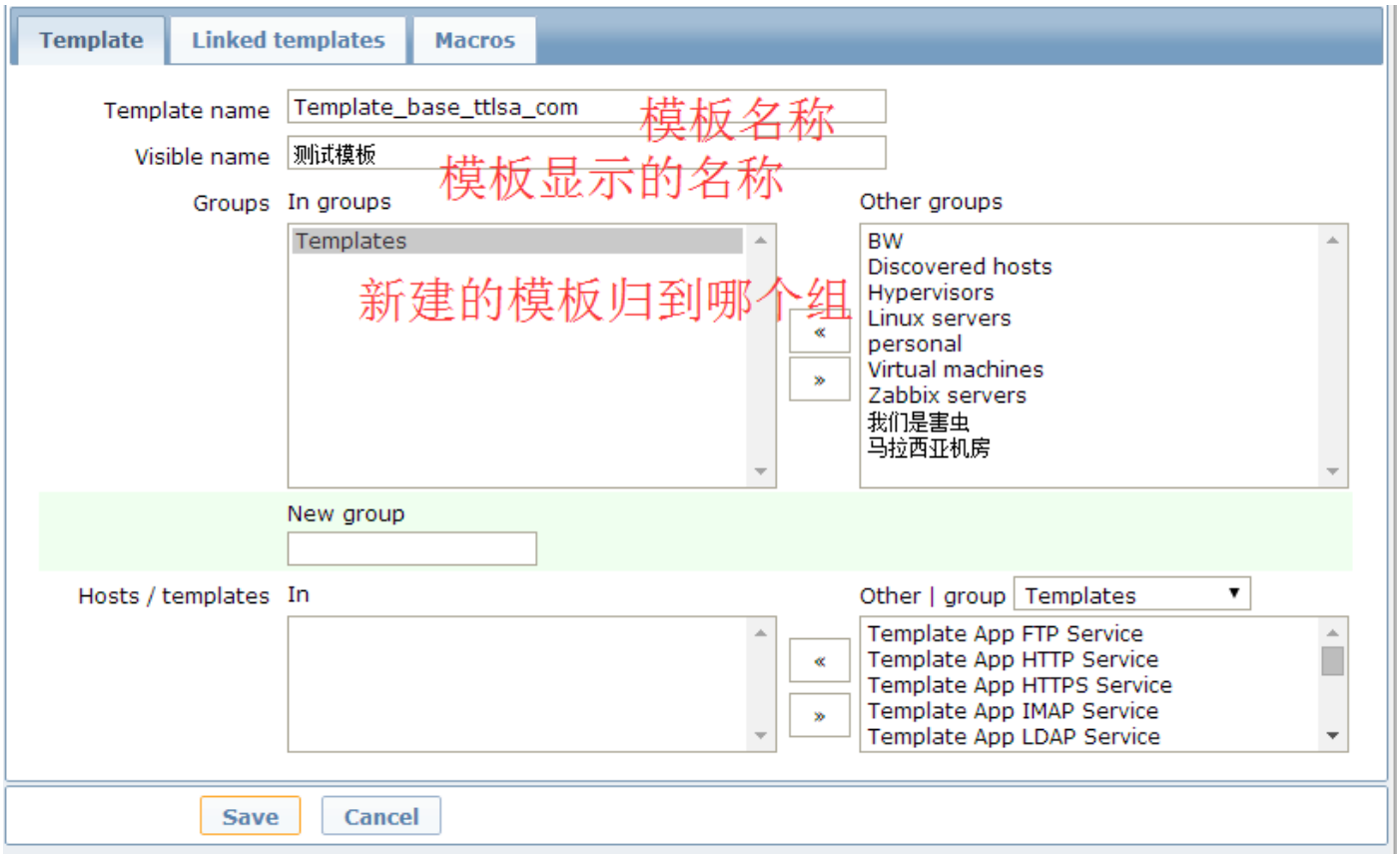
zabbix 模板创建

zabbix 模板中可以包含监控项、触发器、web 监控、图表等等项目，一一创建这些项目之后，在后续的主机只需要套用这个模板，那么主机便可以监控模板里面所配置的监控项目。

创建 zabbix 模板

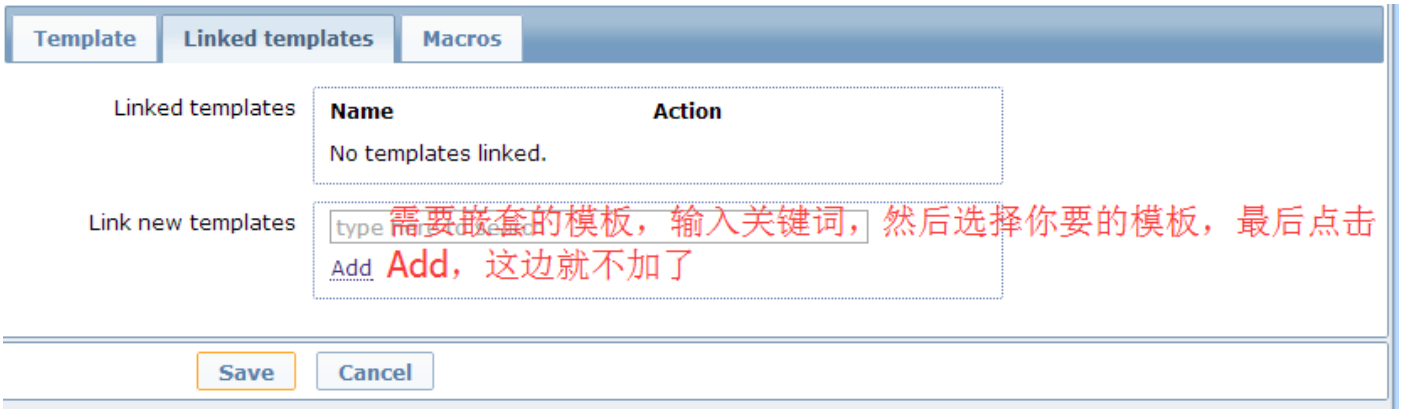
点击 Configuration（配置） —Templates（模板）—create template（创建模板），

template 标签信息如下



参数	描述
Template name	模板名称，在嵌套模板中，都使用 template name
Visible name	显示的名称，template 显示是 visible name，方便识别
Groups Host/template	当前模板归到哪个组
New group	创建一个新组，当前模板便会加入这个组，可以为空
Hosts/Templates	把模板链接到主机

linked template 标签如下

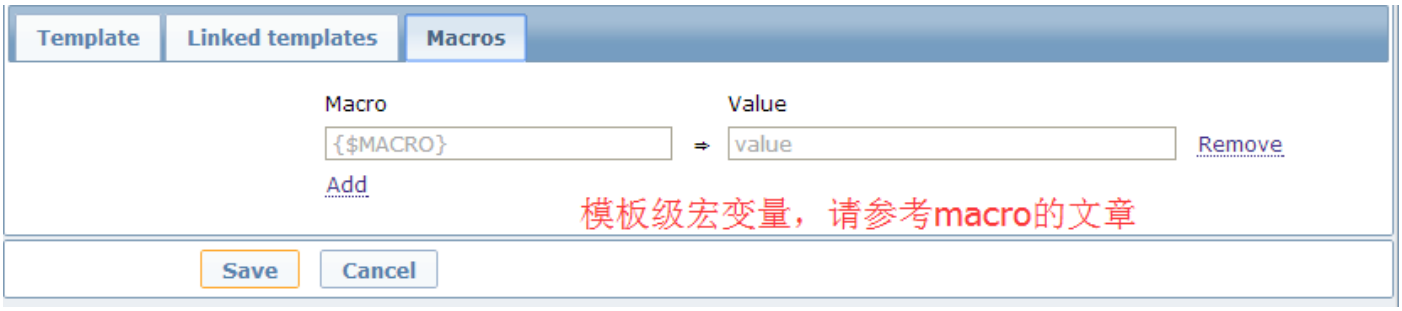


模板嵌套，是一个继承的关系。例如我们定义了一个基础模板，里面 item 有 cpu、内存、硬盘、网卡等等基本信息监控，我们有需要定义个 MySQL 与 WEB 监控模板，那么这两个模板分别嵌套这个基础模板即可，而不需要重复定义监控项。

使用方法：

文本框里输入关键词，例如 Linux 即可搜索到名称含有 linux 的模板，在下拉列表中选择你要的模板，最后点击 Add（千万不要忘记）。

Macro 标签如下



Macro 变量名称，value 为变量值。更多的使用方法请参考《zabbix 自定义用户 Macro》

添加 items, triggers, graphs,low-level discovery rules,web scenarios,screens

一一添加 items, triggers, graphs,low-level discovery rules,web scenarios,screens。与在单台 host 添加 item，trigger，graphs 等等的方法是一样，这边我就不再重复了。添加完毕之后，一个模板也就这么完成了。

编辑 zabbix 模板

点击 Configuration（配置） —Templates（模板）—你需要编辑的模板，当前的底部要比创建模板要多几个按钮，我们分别来讲下这下按钮都是做什么的

按钮名称	描述
save	保存，没什么好说的
Clone	克隆模板，克隆一个与当前模板一模一样的模板，此时你只需要修改下模板名称，以及在其基础上做修改，便能很快的完成一个模板
full clone	完全克隆，比 clone 多一点东西，例如 screen
delete	删除模板，如果主机有嵌套当前模板，那么这些 item 依旧保留在主机上，主机不受影响
delete and clear	删除模板，如果主机有嵌套当前模板，那么这些 item 也被删除掉。
cancel	取消

zabbix 链接及解除模板链接

上一节讲到《zabbix 创建模板》，其中就已经涉及到了链接与解除模板链接（link 与 unlink），这篇文章除了说明怎么链接模板以外，还会特别讲到一些需要特别注意的细节。HOST 链接模板之后，便继承了模板里定义的 item，trigger 等等，使用这个方法，配置 zabbix 监控会减少很多重复的体力劳动，并且更加灵活。

备注：模板只能被链接到 host，不是链接到组里面。

zabbix 主机链接模板

Configuration→Hosts→点击你需要链接模板的主机→切换到 templates（模板）选项，Link new templates 的文本框里面输入你需要 link 的模板名称（关键词就可以了），选择你需要添加的模板，点击 Add，最后 save。最后，当前 host 便获得了模板所有的 item，trigger，web 等等实体。

备注：主机 link 多个模板必须注意，模板们不能含有相同的 item key。trigger 和 graphs 中使用的 items 不能是来自多个模板。

当实体 (items, triggers, graphs 等等)添加之后，内部操作如下：

- host 原有的项目与模板的相同，那么 host 原有的监控项目将会被模板所有的覆盖
- 模板中的所有实体添加到主机中

关于 item 列表

link 模板之后，我们可以发现，item 的名称也有些变化。凡是从模板带来的 item，名称前缀带有灰色的模板名称。没有任何前缀的，那么表示这个 item 是在当前 host 里定义的。

实体唯一性规则

通过前面描述，我们可以了解到，zabbix link 多个模板，这些模板不能有相同的实体。如果模板里的实体与当前 host 实体冲突，那么当前 hosts 的实体将会被覆盖，基于此，我们需要了解实体唯一性的规则由什么决定!

属性	说明

items	item key
trigger	trigger 名称与表达式
自定义图表	图表名称与它的 items
applications	application 的名称

多台主机批量 link 模板

批量 link 主机的方法

Configuration - Templates，点击你需要选择的模板， Other | group 里面选择你的主机，点击«,讲主机们添加到左边 Hosts / templates，最后点击 save 即可，如果想移除主机，只需要点击»。

批量更新 template

Configuration — Hosts — 勾上你需要批量更新的主机，左下角下拉框选择 Mass update，然后点击 Go,切换到 template，选择你需要的模板。最后点击 update 即可。

备注：

zabbix 默认提供了很多模板，有什么不懂的可以参考他，但是不推荐你直接在自带的模板上修改，确实有修改修改，我倾向于让你去克隆一个模板。

编辑 link 实体

在 host 里面，点击 zabbix 实体，大部分文本框都是灰色/不可编辑状态，只有更新间隔等等少量内容可以修改。因为很多 host 使用同一个模板，一旦你修改了一个实体，所有 host 都会跟着变化，所以 zabbix 不允许直接修改 link 过来的实体类。如果你确实需要修改他，那么你只能去修改 zabbix 模板，不过记住，修改之前要谨记，所有 link 当前模板的 host 都会一起变动。

Unlinking 模板

Configuration - Hosts - 切换到 Templates 选项 - 点击 Unlink 或者 Unlink and clear，最后点击 save。

unlink 与 unlink and clear 的区别

- unlink：仅仅移除 template，原先的实体会继续保留在 host 上

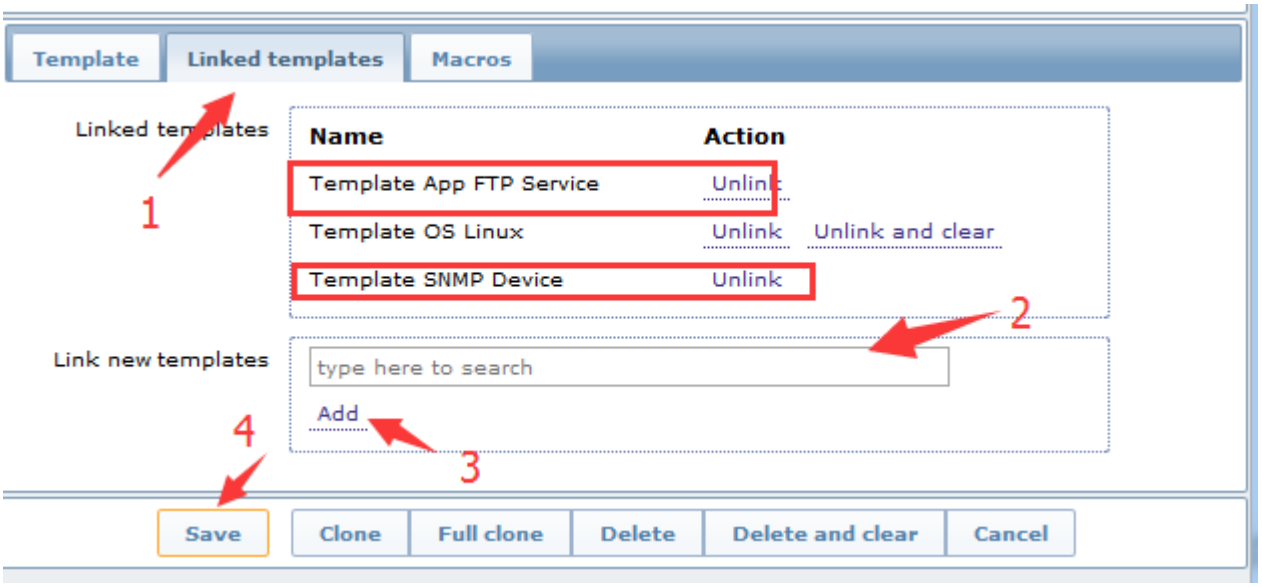
- unlink and clear：移除 template，template 所包含的实体也会一起移除，相对比较彻底。

zabbix 模板嵌套 Templates Nesting

在 zabbix 使用过程中，在某些情况下，一个 host 需要 link 多个模板。这么做显得比较麻烦，很容易忘记到底要 link 哪些模板，我想 link 一个模板就达成这个目标，行不行？然没问题，zabbix 模板内嵌就是这么做的。实际上模板内嵌在《zabbix 创建模板》一文就提到了，简单的说就是：模板 link 多个模板，这便是内嵌。

zabbix 模板内嵌步骤

configuration（配置） - Templates（模板），点击你的目标模板，切换到 linked templates 选项，在文本框里面搜索你需要的模板，然后点击 Add，如下：



说明下两个参数

- Unlink and clear：移除模板，并且移除所有 hosts 上的实例（什么是实体不用我说了吧？前面章节有提过）。比如之前 host 有使用这个模板，那么从这个模板来的实体全部被移除掉。
- Unlink：仅仅是移除模板，实体依旧保留在 host 上。

第八章：zabbix 可视化

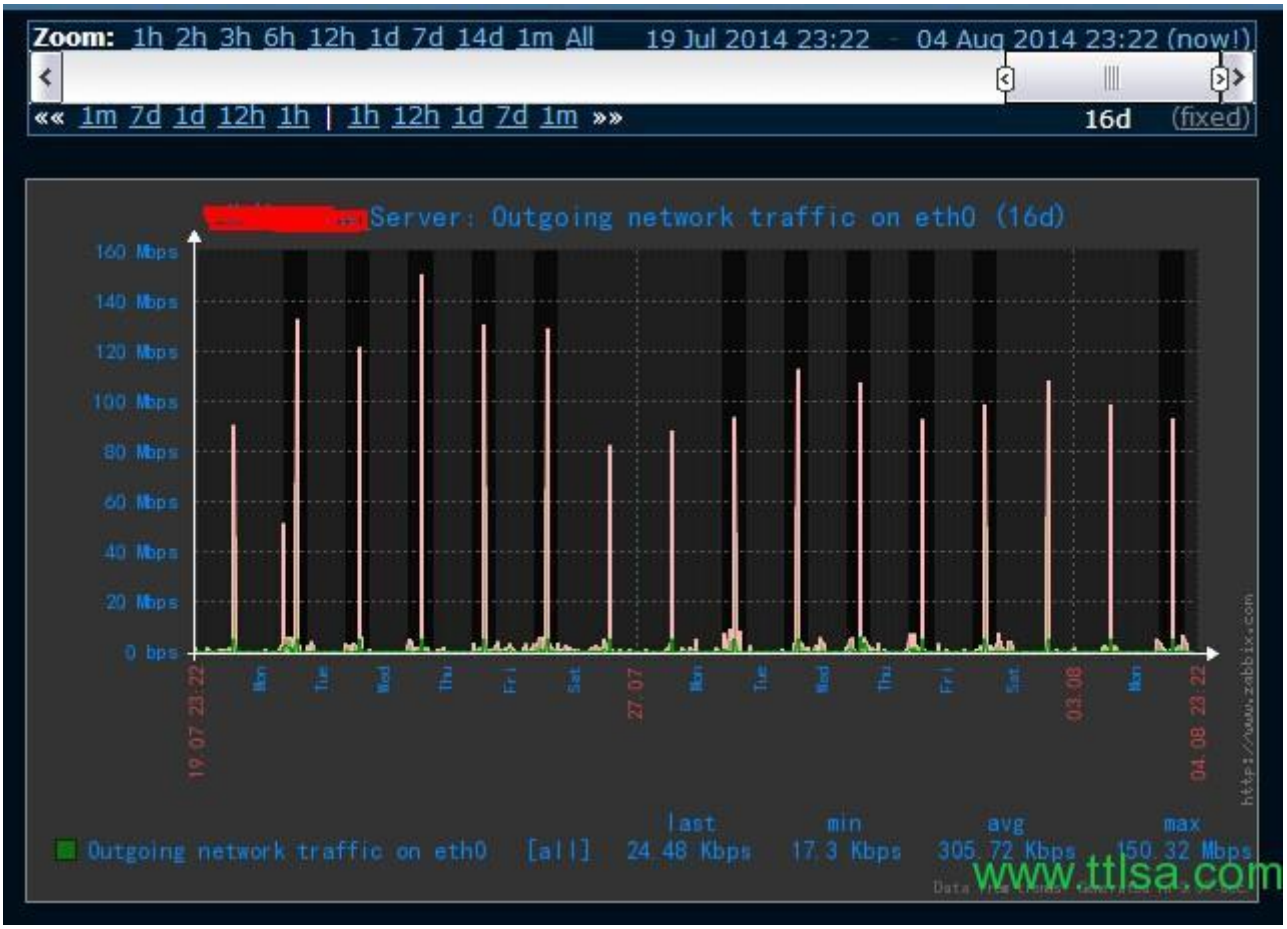
zabbix 图表功能介绍

zabbix 可视化图表绝对是他的一个大特性，我相信很多人也是被它的制图能力吸引到，它可以把任何数值数据转成可读性很强的图表。用过 nagios 的运维们基本都使用 cacti 来绘图，遗憾的是我没用过 cacti，所以我不发表评论。说心里话 cacti 画的图真心比 zabbix 画的漂亮，不过 zabbix 还是用它的简单实用打动了我.

zabbix 简易图表详解

概述

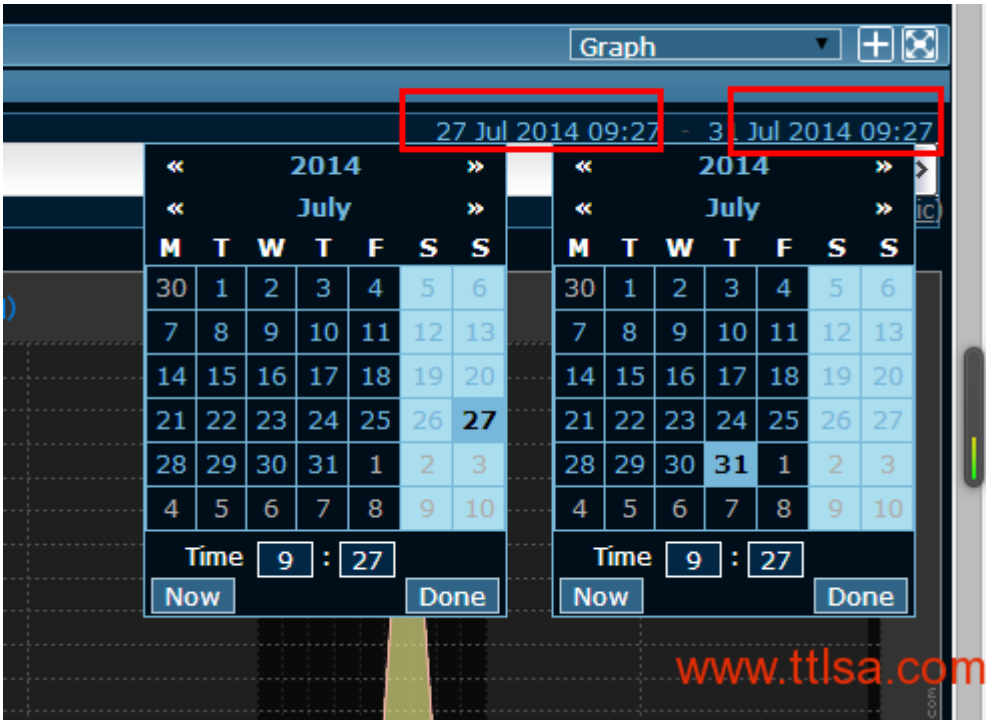
在 zabbix 中，所有数值 item 值都可以绘制成简易的图表。在 Monitoring->Latest data->任意一个数值 item 列上有个 Graph,点击便会出现一个简易图表。如下图：



时间段选择

请看上面的图片，我们可以通过如下方式来查看你需要的数据。

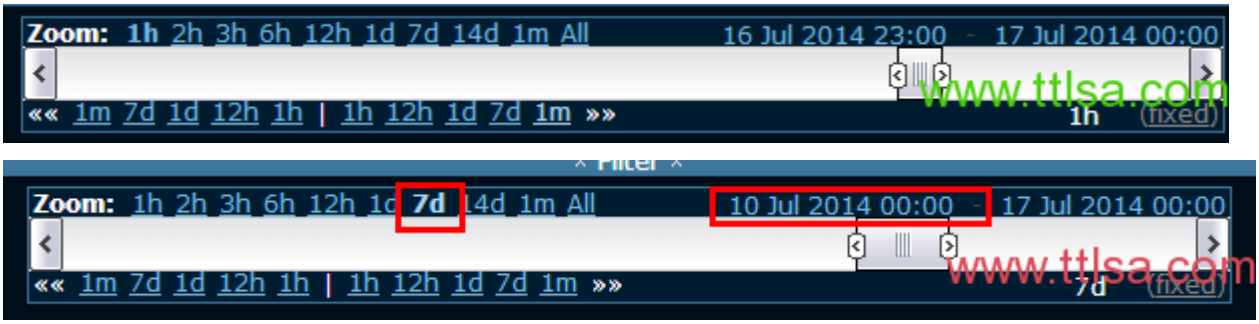
- **通过日历选择时间段**



点击右上角两个日期会弹出两个日历，选择起止两个日期图表便能显示这个时间段的简易图表

■ **点击 1h 1d 等日期前移**

以右上角结束时间为准，如果你点击 1h，那么开始时间会往左边移动，点击 7d，那么会往左边移动 7 天。请看下面两个图片，可以发现点击 7d 之后，开始时间变成了 7 天前。



■ **时间前/后变更**

在滑动条下方可以看到”«« 1m7d1d12h1h | 1h12h1d7d1m »»”，中间有个分割线，表示整个时间段往左/右移动，或者起始时间往左移动/终止时间往右边移动。为什么会有或者的关系呢？是不是有点晕。看看滑动条右下角有个 fixed/dynamic。来讲讲他们的区别吧。

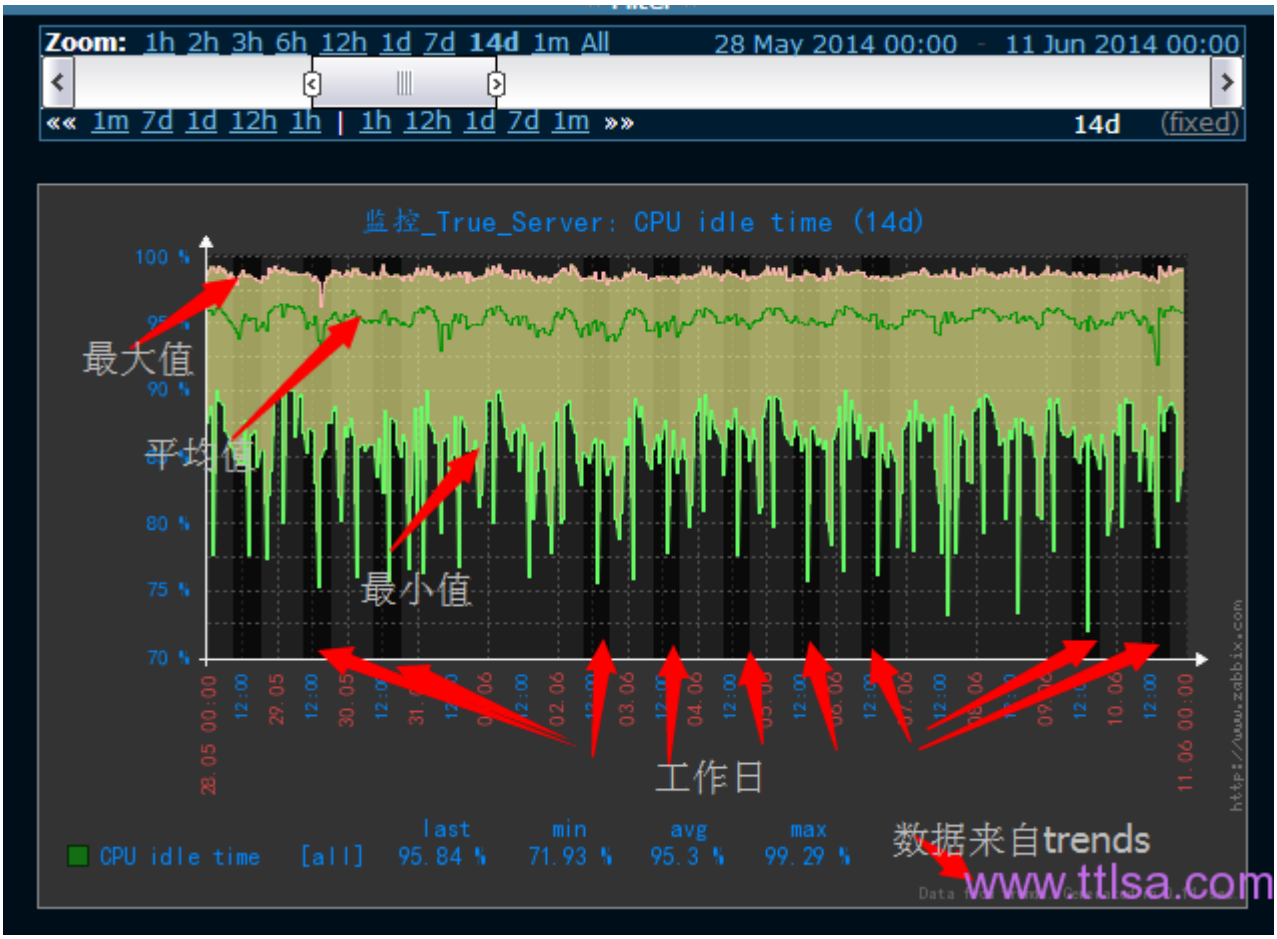
- fixed 翻译过来就是固定的意思，就是说你选择的时间段是固定的，例如你选择的时间段是 1 天，你点击左边的 1d，那么起止时间都会往左边移动，也就是说你怎么移动都只能看 1 天的数据。
- dynamic 翻译过来便是动态的意思，就是说你选择的时间段是动态变化的，例如你选择的时间段是一天，你点击左边的 1d，那么起始时间会往左边移动，终止时间不懂，也就是说你可以看到 2 天的数据。如果你点击右边的 1d，那么终止时间会往右边移动，这样的话，你可以看到 3 天的数据。

■ **4. 通过滑动条**

滑动条，你自己可以左右拖动。也可以点击滑动条左右两侧的'< ‘、’ >’按钮，重点说明一下这两个按钮，如果处在 fixed 下，默认情况下，你点击一次按钮将会往左/右滑动一个小时。如果当前 zoom 你选择的是 2d，那么点击一次将以 2d 为单位左右滑动.如果是 dynamic,不管怎么样，每次都以 1 天为单位.

最新数据 vs 老数据

对于最新的数据，简易图表中只会有一条曲线，每个点坐标点代表一个 item 值，如果数据比较老，来自 trends，那么图表会有 3 条线，分别代表最大值、平均值、最小值，还有图表背景深色部分表示工作日，具体的请看我给的截图



备注：简易图表都会自动显示工作时间的背景，但是在自定义图表（咱们 ttlsa 下一篇的内容）里面这需要配置，图表时间段超过 3 个月，工作时间不会显示，到时候如果不出现，别太诧异。

使用历史/趋势数据生成图表

图表都是基于历史或者趋势数据生成的，在图表的右下角我们可以判断图表是使用什么数据生成的，如果是”data

from history”表示使用历史数据生成。如果是“data from trends”表明图表数据来自趋势数据。关于《zabbix 历史与趋势记录》请参考以前的文章，有做详细的介绍。

关于使用趋势数据:

- 较老的 item 历史数据，例如 item 的历史记录只保留半年，这个时候你查看半年以前的数据，因为历史数据已经被删除了，所以只能使用趋势数据来绘制图表。
- 数据拥挤，如果图表水平像素超过 3600/16，那么不管你的历史记录是否存在，他一定会使用趋势记录，你想想，如果一个 item 每隔一秒去获取数据，你要查看他 10 天的数据，那张图片该多乱，这个时候使用趋势记录来绘制图片的效果实际上是一样的。
- 趋势记录被禁用，如果存在当前时间段 item 的历史数据，那么将会使用历史记录来绘制图表. 这个特性从 Zabbix 2.2.1 开始支持 (以往, 如果禁用了趋势记录，那么只会显示一张空白图表，不管历史记录是否存在.

切换到原始值

右上角的下拉菜单，可以选择 Graph/values/500 latest values，分别可以查看简易图表/值/最新的 500 个值.如果觉得图表不是很直观，可以切换到原始值.

zabbix 自定义图表 Graph

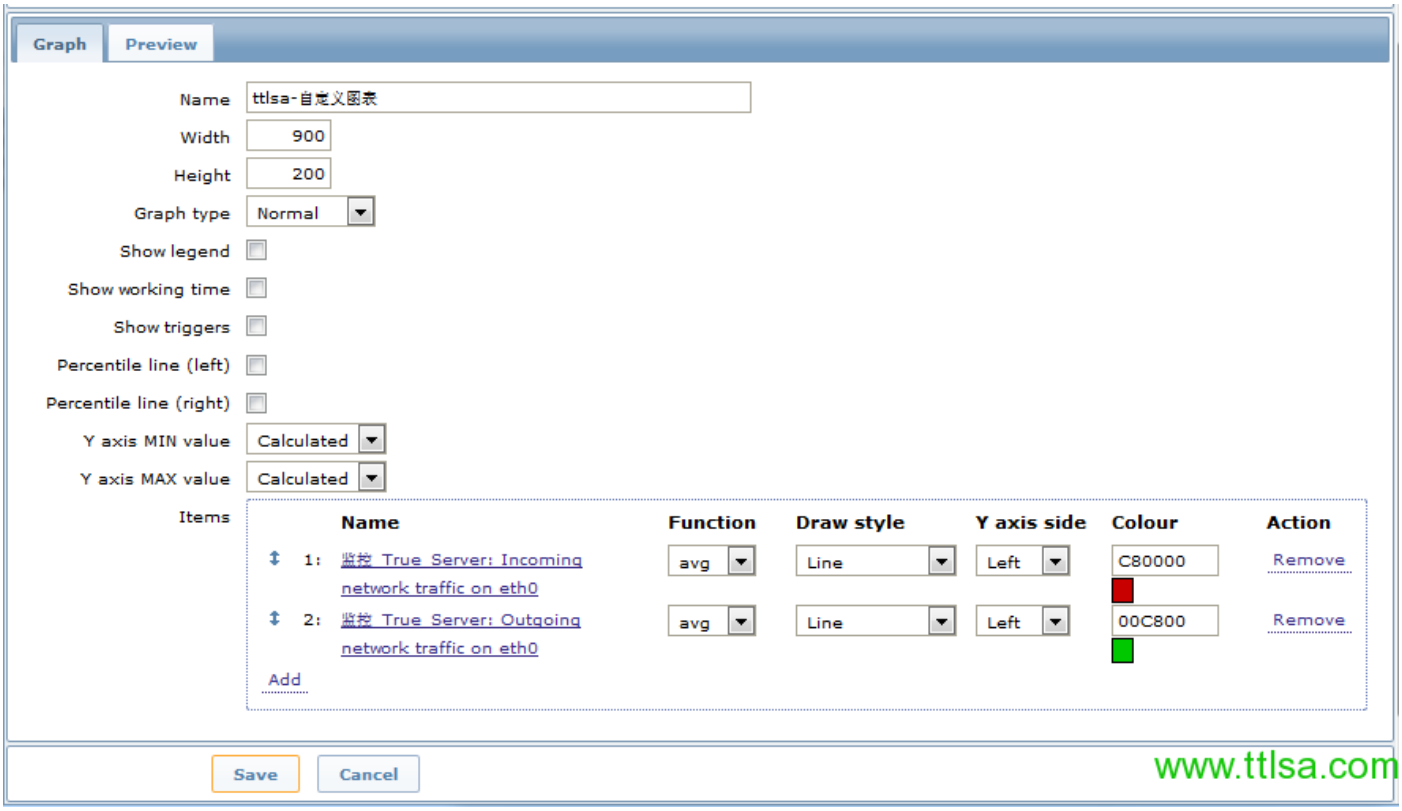
概述

今天我们要讲的是 zabbix 自定义图表功能，这个自定义图表是什么呢？顾名思义,zabbix 提供了一个自定义图表的功能，这不是废话么？呵呵~前面文章 讲到的《zabbix 简易图表》只能显示单个 item 的数据图表。如果我们想显示多个信息到一个图表上，那必须使用 zabbix 自定义图表功能，比如，我 们最常用的网卡流量监控，一张流量图上会包含进/出的流量信息。一个图表的数据可以来源一台主机，也可以来源于多台主机

配置定义图表

创建自定义图表步骤如下：

Configuration→Hosts（或者 templates）,点击 hosts/template 列的 Graphs，点击右上角的 Create graph，出现如下表单。



Graph 属性:

属性	描述

Name	图表名称（唯一）
Width	图表宽度（单位：像素）(仅用于预览和 pie/exploded 图表).
Height	图表高度（单位：像素）
Graph type	图表类型: **Normal** – 常规图表, 值显示为线条 **Stacked** – 叠图, 显示填充区域 **Pie** – 饼图 **Exploded** – “裂开的”饼图，显示部分切出的饼图
Show legend	显示图例,例如 item 名称与最大、平均、最小的数据，一般显示在图表的下方
Show working time	是否显示工作时间，如果选择这个复选框，那么非工作时间背景为灰色。备注：饼图和爆炸式饼图没有这个参数
Show triggers	如果选择现象，那么触发器将会用红线表示. 两种饼图不包含这个功能
Percentile line (left)	左 Y 轴百分数
Percentile line (right)	右 Y 轴百分数
Y axis MIN value	Y 轴最小值： **Calculated** – 自动计算 Y 轴最小值（取 item 最小值） **Fixed** – 固.定 Y 轴最小值. 饼图与裂变式饼图没有这个参数 **Item** – 选中 item 的最新值（例如你选中某网卡，那么它的最小值将来自这个网卡 item 的最新值）
Y axis MAX value	Y 轴最大值： **Calculated** – 自动计算 Y 轴最大值（取 item 最大值） **Fixed** – 固.定 Y 轴最大值. 饼图与裂变式饼图没有这个参数 **Item** – 选中 item 的最新值（例如你选中某网卡，那么它的最大值将来自这个网卡 item 的最新值）
3D view	立体风格图表，仅适用于饼图与爆炸式饼图
Items	监控项，图表的数据来源

配置图表 items

图表的数据来源于 items，点击 add 选择需要显示到图表的 item，可以添加多个。

Item 展示属性:

参数	描述
Sort order (0→100)	绘图顺序，可以上下拖动 items 来改变他们的顺序.这个顺序用来决定图层的顺序。
Name	item 名称
Type	(仅用于两个饼图图表): **Simple** – 按比例显示 **Graph sum** – 充满整个饼图 一 张图表只允许有一个 items 是 Graph sum，否则报错：ERROR: Cannot display more than one item with type “Graph sum”，通常用于影片，硬盘大小 item 使用 Graph sum，剩余空间则使用 simple。这样一个饼图的硬盘使用情况便一目了然。
Function	当一个 item 有多种数值时,选择一种数值用于图表展示 **all** – 所有值 (最小、平均、最大) **min** – 仅最小值 **avg** – 仅平均值 **max** – 进最大值
Draw style	绘制风格(只有常规图表存在该选项): **Line** – 绘制线条 **Filled region** – 绘制填充区域 **Bold line** – 画粗线 **Dot** – 画点 **Dashed line** – 画虚线
Y axis side	Y 轴在左边还是右边
Colour	颜色

图表预览

在创建图表的过程中，我们可以随时预览修改的配置图表，点击标签 preview 即可。备注：如果是 template 预览时没有意义的，毕竟没有任何数据。

关于触发器限制

如果图表的高度小于 120 像素，那么图标上将不会展示触发器相关信息。

zabbix Screens 视图配置

screen 翻译成中文为“屏幕”，在超市、单位等等地方都比较常见到监控视频，视频上有多块小视频，实际上 zabbix screen 和这个功能类似。你可以设置多个 screen，每个 screen 可以显示特定信息，例如某台主机的 cpu、内存、硬盘、网卡流量使用状况，也可以显示 text 文本，甚至能够嵌入其他 screen。不明白？看图

以上图片请点击查看大图，zabbix screen 就是上面那个样子，上面都是小图表，zabbix screen 支持很多格式。接着往下看。

zabbix screen 支持的元素类型

simple graphs

user-defined custom graphs

maps

other screens

plain text information

server information (overview)

hosts information (overview)

trigger information (overview)

host/hostgroup issues (status of triggers)

system status

data overview

clock

history of events

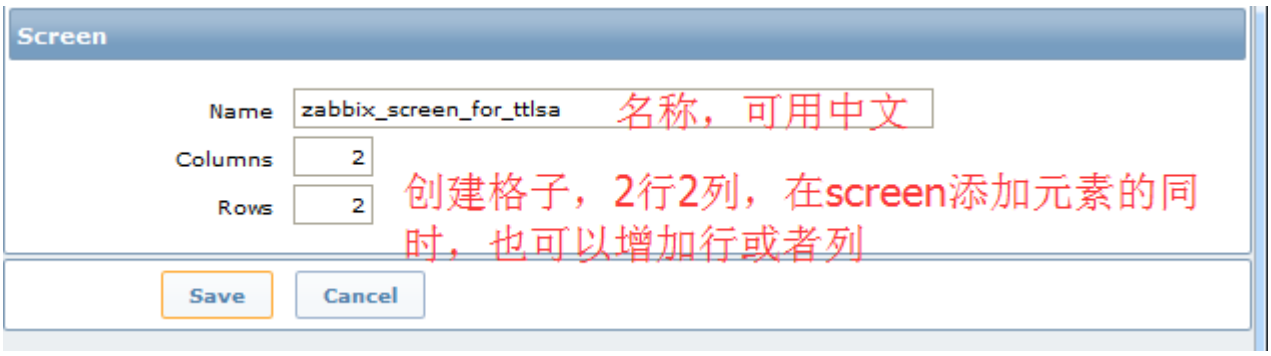
history of actions

URL (data taken from another location)

一共 15 种，最常用的是 simple graphs，想对这些有更大的一个了解，请看后续章节，会有一个专门的演示

创建 screen

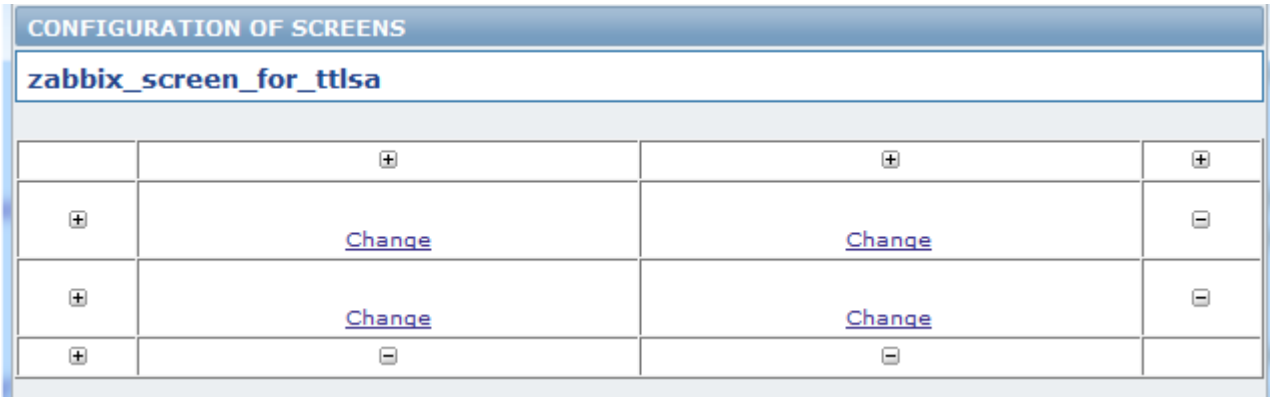
点击 configuration - screen - create screen，输入如下信息：



属性	描述
Name	screen 名称，可以用中文
Columns	列个数
Rows	行个数

screen 元素添加

点击 configuration－screen－“zabbix_screen_for_ttlsa”



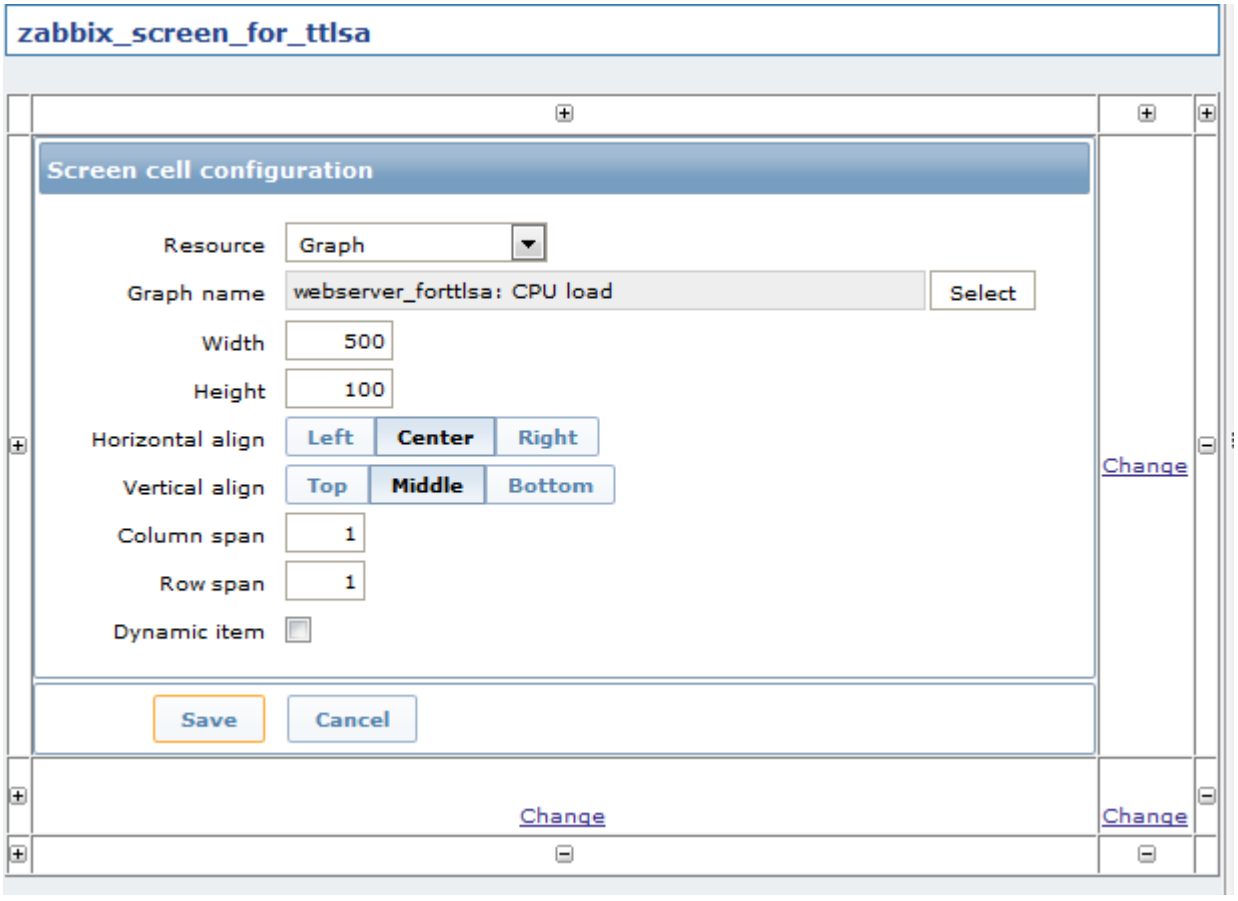
说明：

1. change：可添加元素，例如图表、map、text 等等元素

2. 加号：相应增加行或者列

3.减号：响应删除行或者列

点击 Change，如下图：



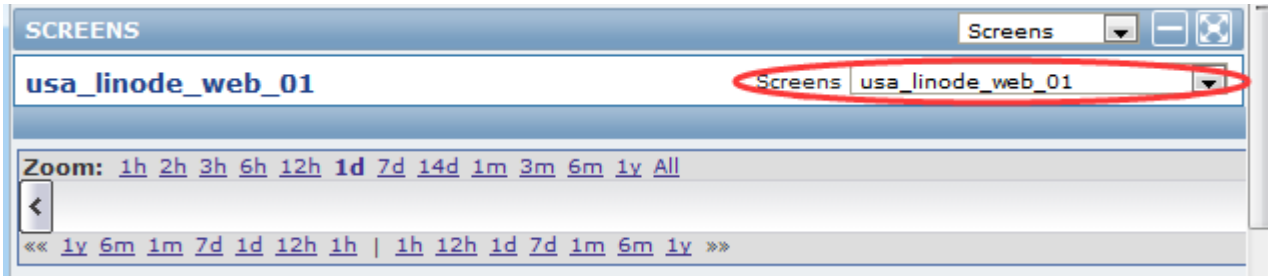
参数说明

参数	说明
Resource	simple graphs user-defined custom graphs maps other screens plain text information server information (overview) hosts information (overview) trigger information (overview) host/hostgroup issues (status of triggers) system status data overview clock history of events history of actions URL (data taken from another location)
Horizontal align	水平对齐方式： Center（居中） Left（左对齐） Right（右对齐）
Vertical align	垂直对齐方式： Middle（中间） Top（顶部） Bottom（底部）
Column span	列跨度，当前元素占用几个列
Row span	行跨度，当前元素占用几个行
Dynamic elements	动态元素

Dynamic elements

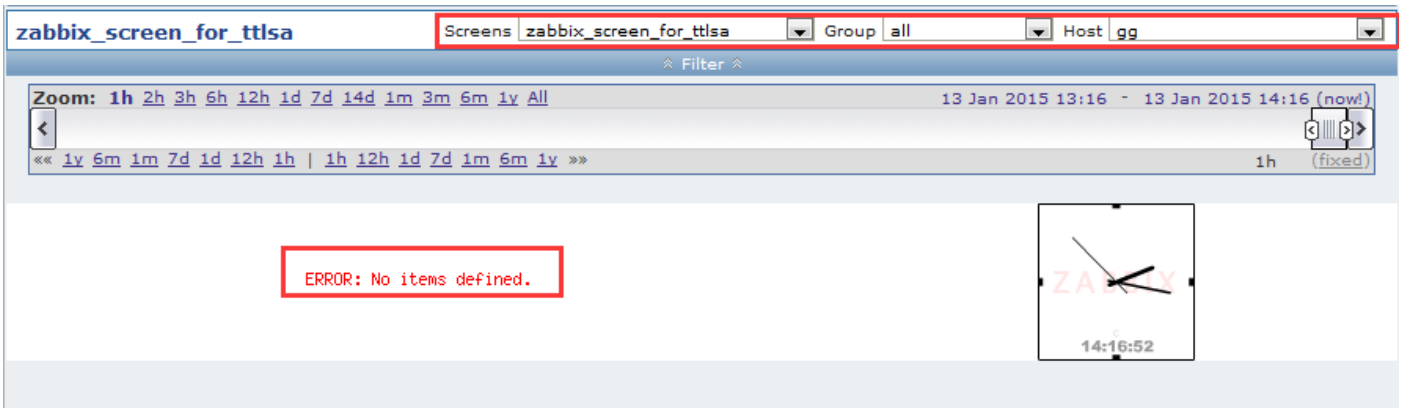
Dynamic elements 动态元素，有点不好理解。如果这个 screen 有一个元素选中了 Dynamic elements，在 monitoring->screen,screen 下拉列表选择这个 screen，会出现 hosts 下拉表，选中一个 hosts，如果当前元素不属于这个主机，那么不会显示，如果属于这个主机便会显示。

普通的 screen



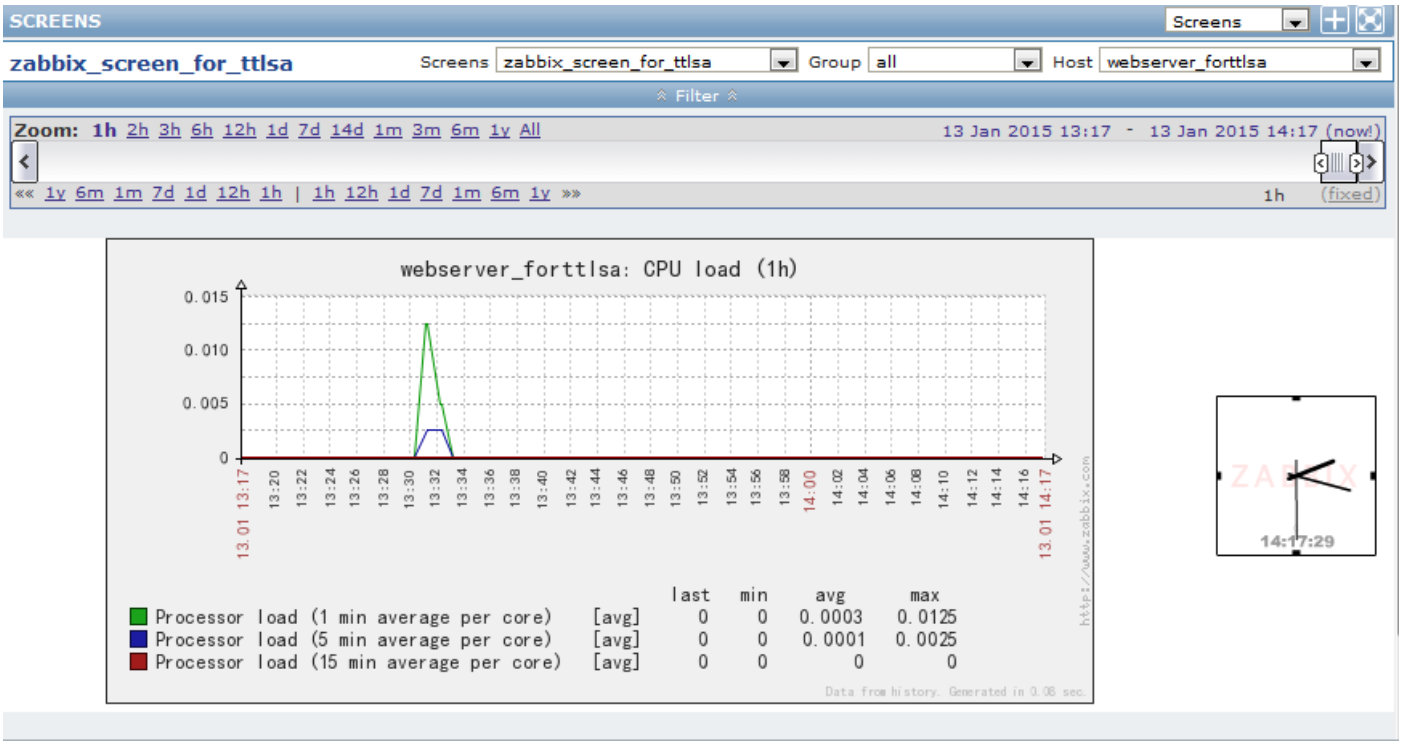
可以看到仅有 screens 一个下拉列表

带动态元素的 screen：



区别如下

- 多了两个下拉列表，可以选择主机
- 有一个时钟，当前时钟不属于任何 hosts，所以不管你选哪一个 host，他都能正常显示
- 当前 Host”gg”左侧出现”ERROR:No items defined”，这里是一个动态元素，因为动态元素不属于 gg，所以显示出错了。我们选择对的 host，情况如下：



备注：

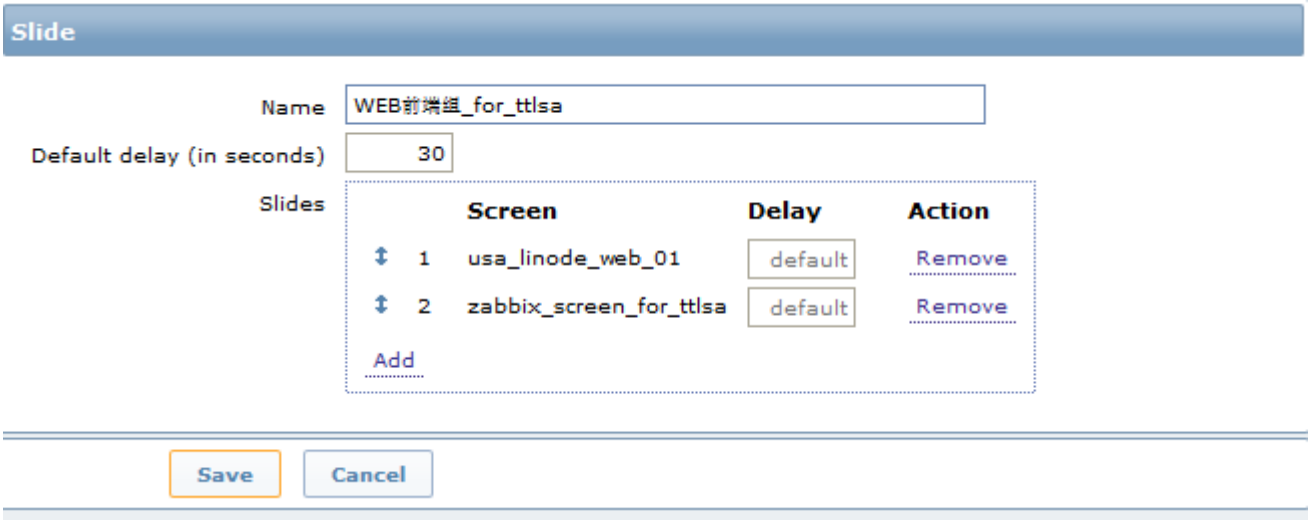
动态元素支持的类型：Graphs (custom graphs)、Simple graphs、Plain text

zabbix Slide shows 幻灯片展示

定义好 screen 之后，我们想了解服务器状况之时，一般会一个个 screen 点过去，zabbix 提供了幻灯片展示方法，可以定义多个 Slide，一个 slide 中可以包含多个 screen。

创建 slide shows

点击 Configuration — Slide shows - Create slide show

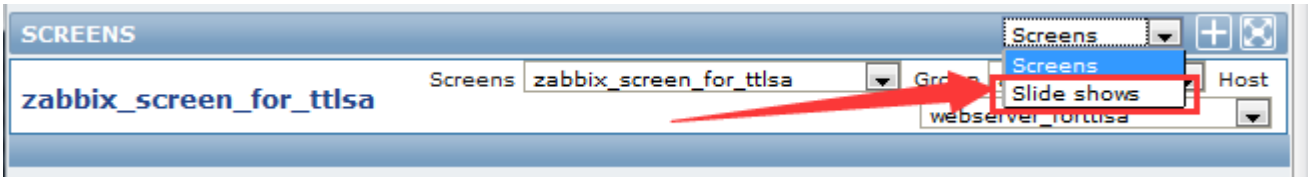


参数说明

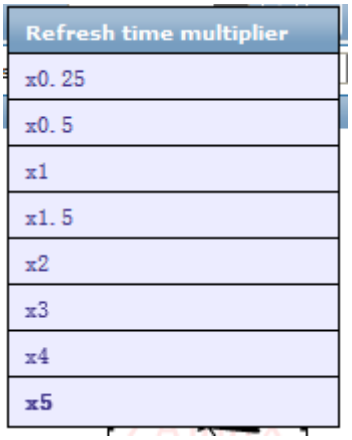
参数	说明
Name	幻灯片名称
Default delay (in seconds)	默认幻灯片切换间隔时间
Slides	轮换显示的 screen 列表
Screen	Screen 名称.
Delay	当前 screen 需要被显示多久，如果设置为 0 或者不填写，将会使用默认值
Remove	移除 screen

slide shows 展示

虽然 slide shows 配置与 screen 是分开的，但是展示却在一个菜单里，点击 monitoring - screen，screens 下拉菜单选择 slide shows



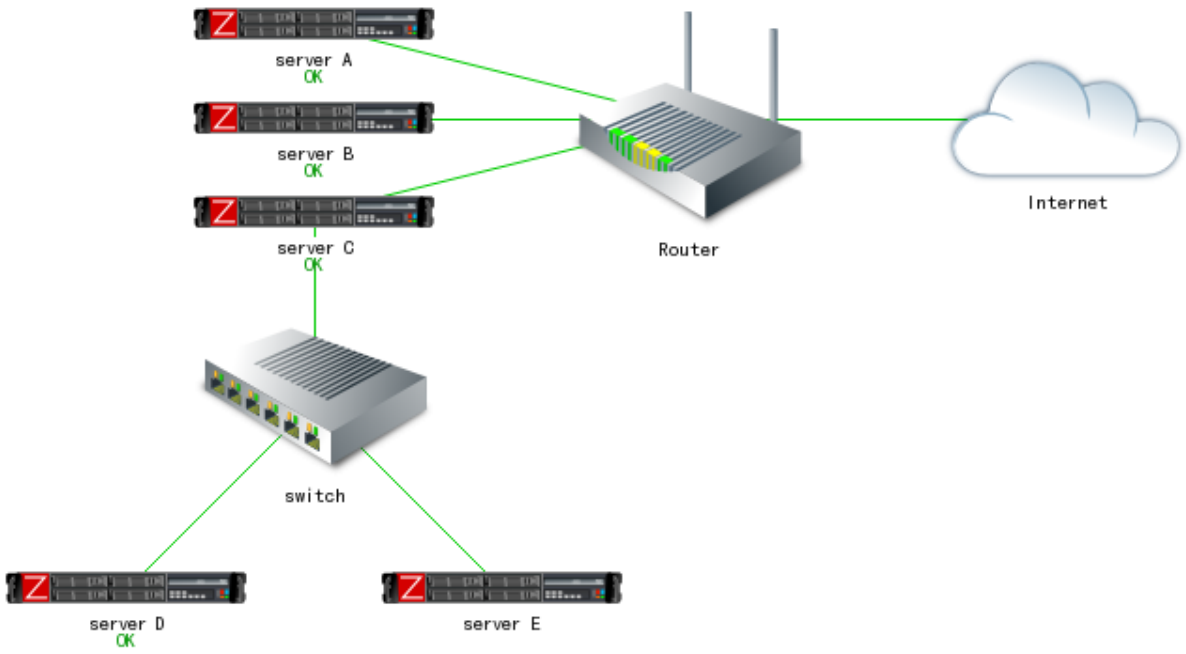
如果你觉得切换速度过慢/过慢，你可以点击加速图标（在加号后面一个，有点像书本），在弹出的菜单选择你调整的速率



zabbix 网络拓扑图配置 network map

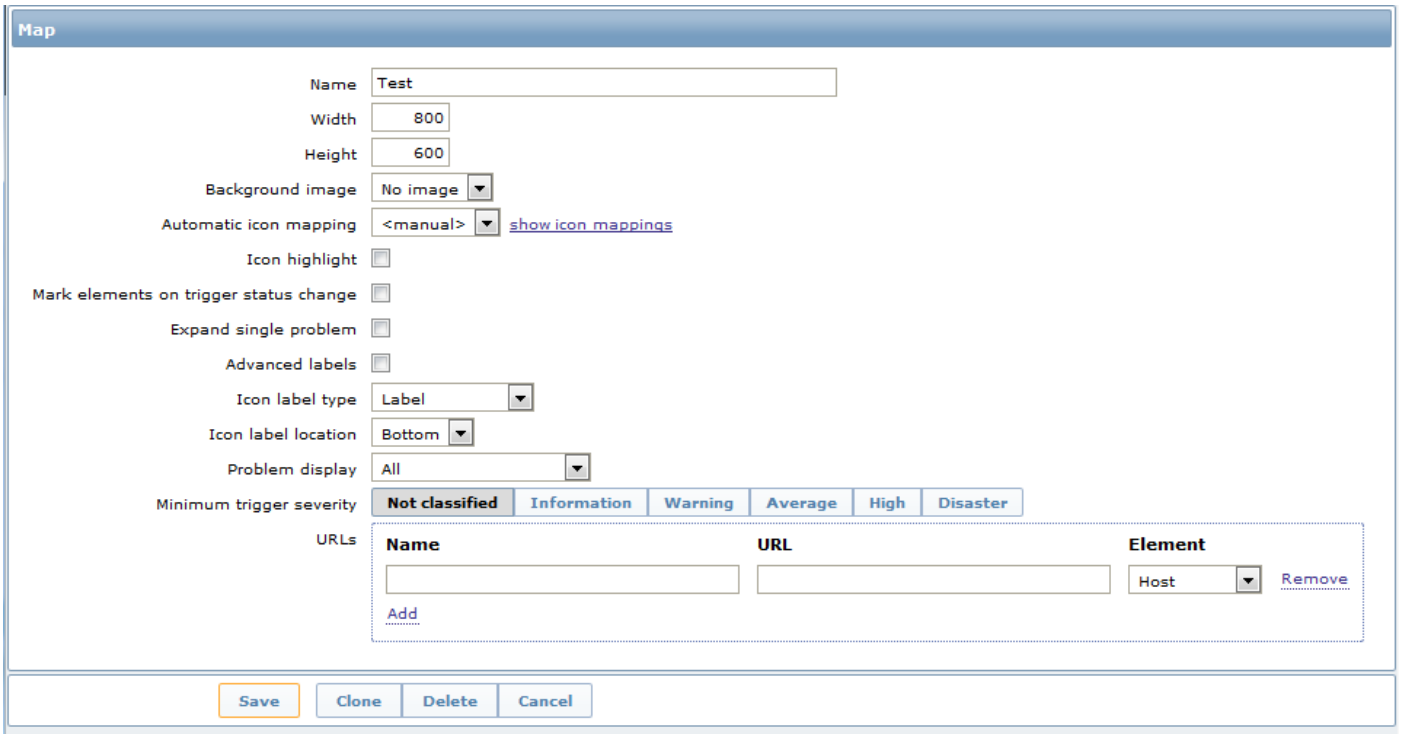
介绍

“zabbix network map”可以简单的理解为动态网络拓扑图，可以针对业务来配置 zabbix map，通过 map 可以了解应用的整体状况：服务器是否异常、网络是否有故障、应用当前什么状态。如果你不需要这些东西，至少你可以通过 network map 绘制一张网络拓扑图。看看效果



创建 network map

点击 Configuration → Maps - > Create map



参数说明

参数	描述
Name	名称，不能重复
Width	宽度，像素为单位
Height	高度，像素为单位
Background image	背景图像: No image – 无背景图像(白色背景) Image – 可以选择图片作为背景，不支持缩放（为啥我的 zabbix 没有这个选项）
Automatic icon mapping	图标映射， Administration → General → Icon mapping.
Icon highlighting	图表突出显示
Mark elements on trigger status change	突出显示触发器状态
Expand single problem	显示故障名称
Advanced labels	为不同类型元素定义不同标签
Icon label type	图标名称: Label – icon 标签名 IP address – IP 地址 Element name – 元素名称(如：主机名)

	Status only – 状态(OK 或者 PROBLEM) Nothing – 不显示
Icon label location	图标名称位置: Bottom – 图标下方 Left – 图标左边 Right – 图标邮编 Top – 图标上方
Problem display	显示故障次数: All – 所有次数 Separated – 分别显示未确认的故障与总故障数 Unacknowledged only – 只显示未确认故障的数量
Minimum trigger severity	低于选择故障严重性级别的故障将不会显示在 map 中。例如，选择了“Warning ”，故障级别为“Information ”和“Not classified”的触发器事件都不会反映到 map 中。Zabbix 2.2 加入此参数.
URLs	monitoring - map - 你的 map - 点击你的元素会出现一个菜单，如果有指定 urls，那么 url 会出现在当前菜单中。你可以点击当前 url 来跳转到具体页面。urls 可以使用 macros: {MAP.ID}, {HOSTGROUP.ID}, {HOST.ID}, {TRIGGER.ID}

添加元素到 map 中

点击上方的图标”+”可以添加元素（host、group、trigger 等），然后左上角会出现一个主机，这时候我们任意拖动它，也可以点击图标”-“来删除它。点击这个元素，在弹出的属性框里面录入它的一些信息，一个元素就添加完成了。

我们可以注意到在上方有“Grid [Shown|On] 20×20”，点击 shown 当前 map 的表格消失（此时文章变成了 hidden），再点击一次，表格又回来了。On 表示当前 map 里面的元素都会按着表格对齐（和 windows 的桌面一样），点击 On 文字变为 Off，表示当前 map 里的元素可以任意拖动摆放。后面的 20×20 是一个下拉列表，表示表格的大小。

看看 map 元素属性

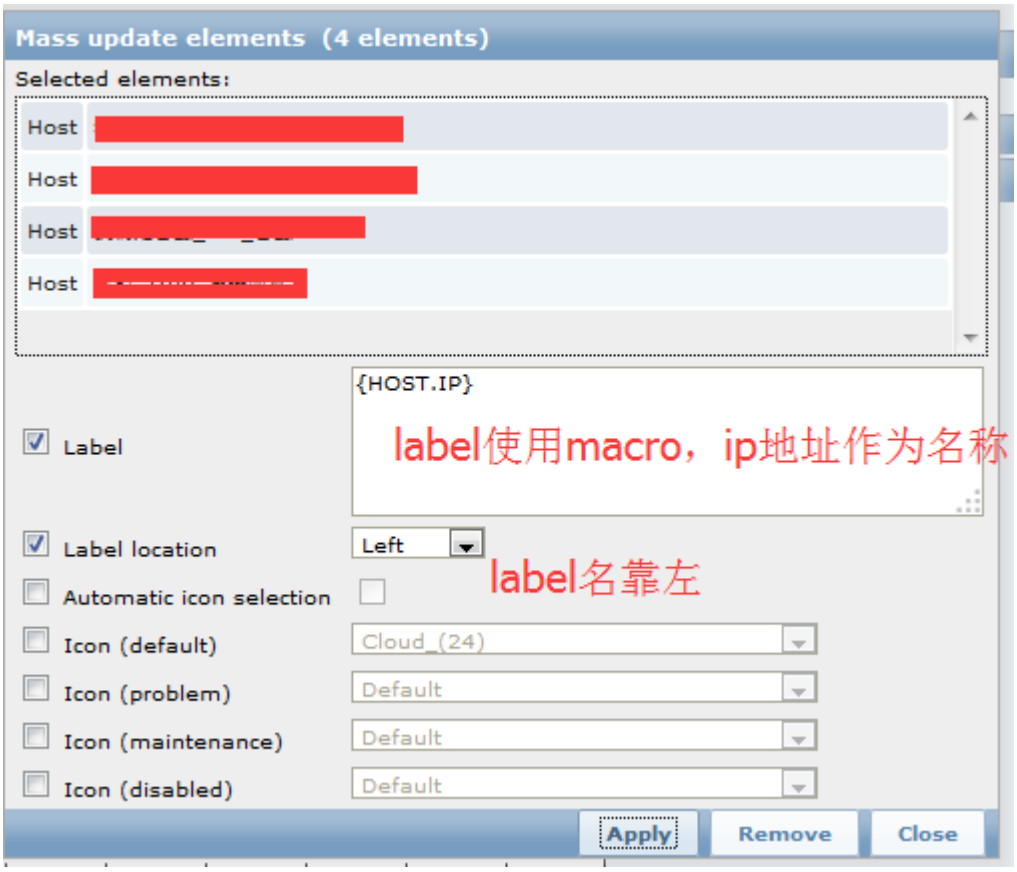
属性	描述

Type	元素类型: Host – 代表主机，他所有的触发器状态都会反映到图标上 Map – map 元素图标，点击之后会链接到相应的 map Trigger – 单个触发器状态 Host group – 代表组机组图标，组内所有主机的触发器状态都会反映到图标上 Image – 图标，不指向任何资源
Label	元素名称，可以使用 macros，支持多行文本
Label location	名称所在位置: Default – 默认位置，一般是 bottom Bottom – 图标底部 Left – 图标左边 Right – 图标右边 Top – 图标上方
Host	如果当前元素类型为 Host，可选择相应的 host，需要搜索
Map	如果当前元素类型为 map，可选择相应的 map
Trigger	如果当前元素类型为 trigger，可选择相应的 trigger
Host group	如果当前元素类型为 Host Group，可选相应的 group，需要搜索
Icon (default)	默认图标
Automatic icon selection	使用 icon mapping 来决定使用哪个图标
Icons	元素在不同状态下不同的图标: default, problem, maintenance, disabled.
Coordinate X	map 元素横坐标
Coordinate Y	map 元素纵坐标
URLs	monitoring - map - 你的 map - 点击你的元素会出现一个菜单，如果有指定 urls，那么 url 会出现在当前菜单中。你可以点击当前 url 来跳转到具体页面。urls 可以使用 macros: {MAP.ID}, {HOSTGROUP.ID}, {HOST.ID}, {TRIGGER.ID}

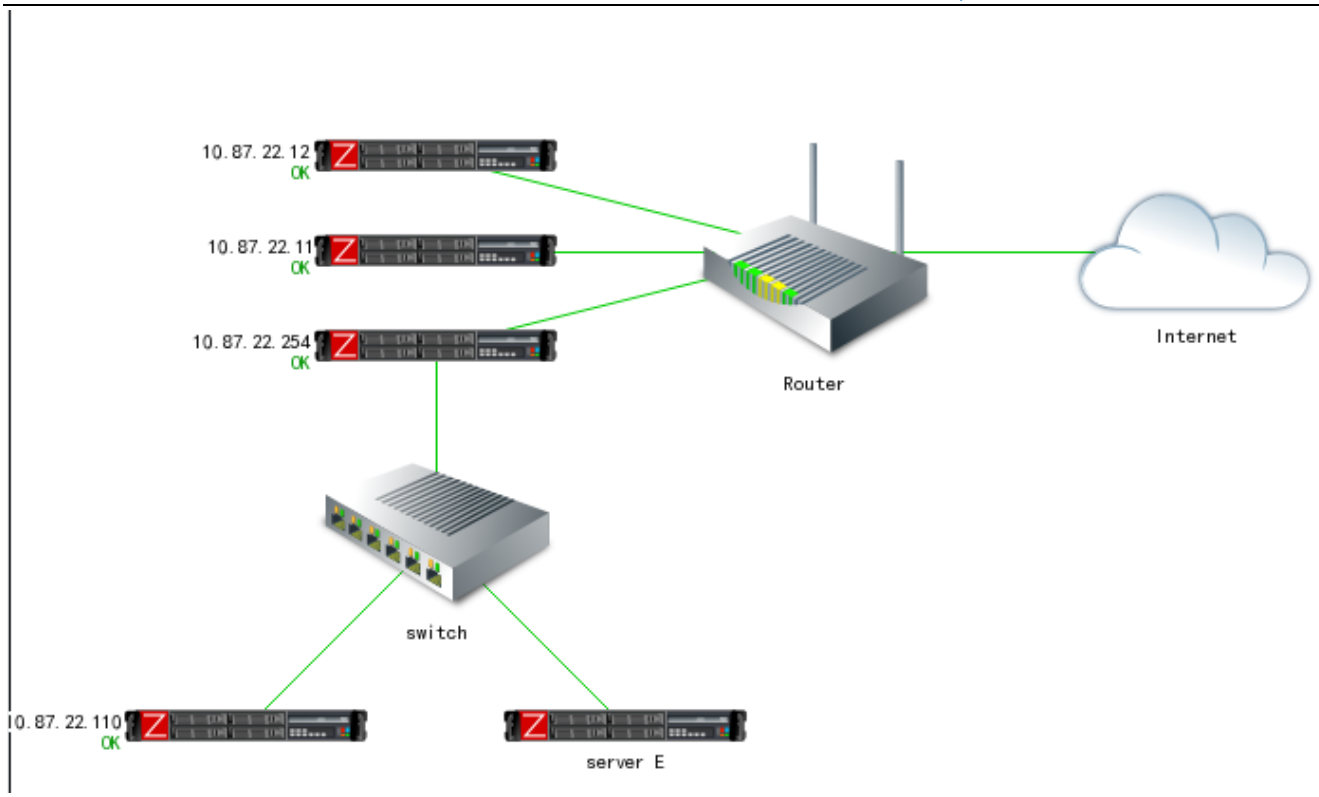
备注：大家一定要记得点击上面的“save”按钮，否则你所有的更改都白费了，zabbix 这点很讨厌，不会自动保存，我已经多次忘记点击 save，然后一切重来。在我们未保存的情况下离开 map 页面，zabbix 提示我们保存，那多好，可惜 zabbix 竟然没有这么做。为什么？

批量修改 map 元素

按住 Ctrl 的同时，鼠标选中多个 map 元素，如下图：

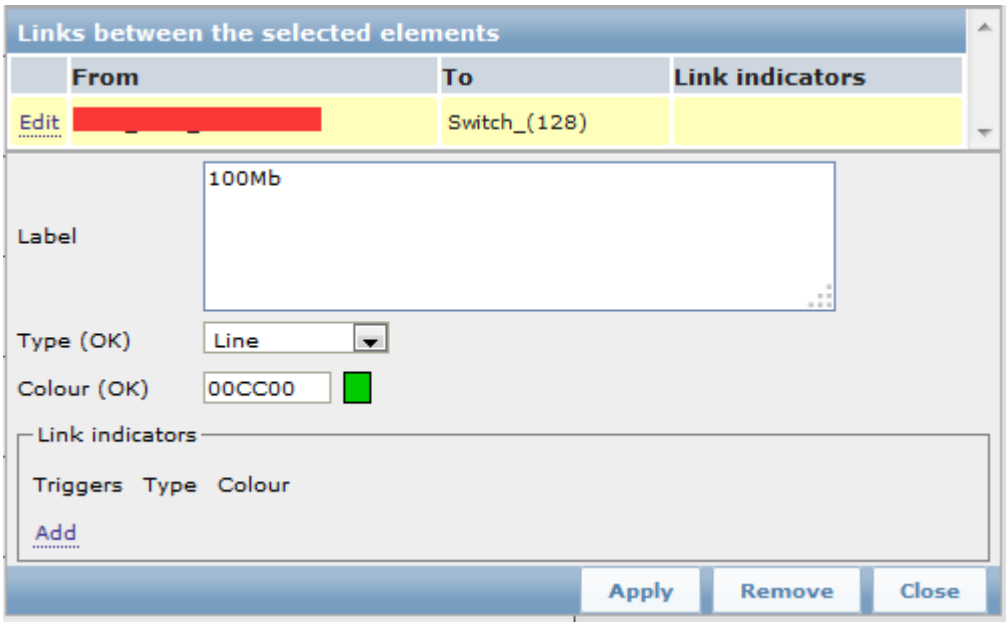


我们批量修改了元素名称，使用 macro{HOST.IP}，并且 label 名称在元素的左边，效果如下



map 元素相连接

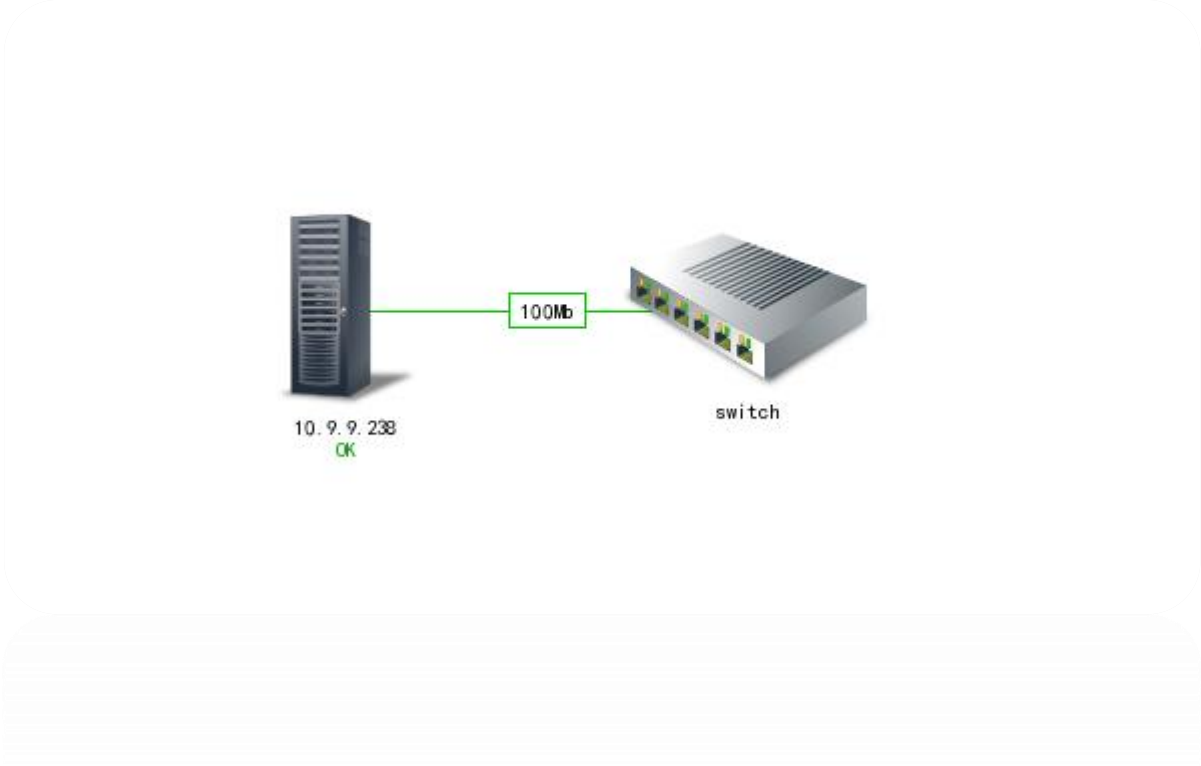
网络拓扑上有了服务器、交换机、路由器，还差一条网线把他们连在一起，并且标明他们之间的链路速度。按住 Ctrl 并且选中两个设备，点击上方 LINK 后边的”+”，在弹出的属性框最后将会增加一条链路属性，点击 edit，输入相关信息，如下：



属性说明

属性	说明
Label	线路名称，可以使用 macro
Connect to	当前元素与哪个元素连接
Type (OK)	连接线风格: Line - 细线 Bold line - 粗线 Dot - 点线 Dashed line - 虚线
Lable	线路名称
Colour (OK)	线条颜色.
Link indicators	链路状态，触发器有故障都会显示到链路上

链路效果如下



zabbix 拓扑图展示链路状况 Link indicators

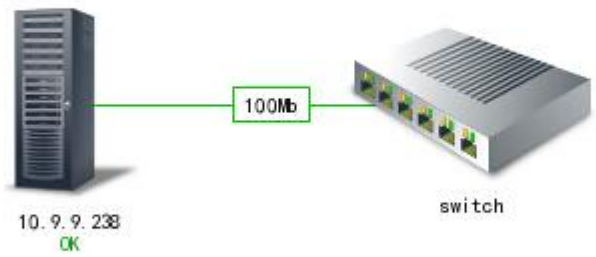
介绍

上一篇文章《zabbix 网络拓扑图配置》我们已经了解了如何配置 zabbix map，也提到了如何连接两个 map 元素，这节我们来讲两个 map 元素之间的链路指示配置。我们需要在链路上配置 trigger，如果 trigger 出现问题，那么线路颜色发生相应的变化，这样很容易判断设备的网络状况。

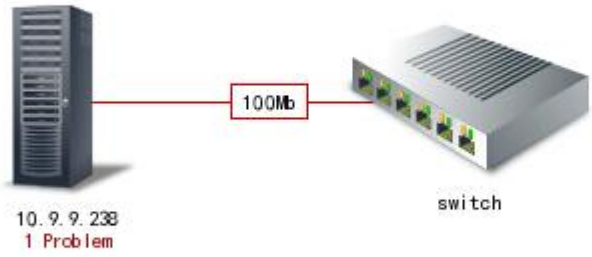
Link indicators 配置

首先需要连接两个设备，在弹出的链路属性上点击 edit（如何连接两个 map 设备，请看上一篇文章），在 trigger 处点击”add”，选择用于判断链路状态的 trigger，例如我选择的“Zabbix agent on webserver_forttlsa is unreachable for 5 minutes”，一旦 zabbix agent 连不上，链路颜色将会变成红色。

链路正常图如下



链路异常如下



IT services

什么是 IT Services

服务器或者某项服务、业务的可用率，不懂技术的上级领导会过问最近服务器可用率如何、所有 api 的状况怎么样？通常一些技术人员会说负载怎么样，哪些 cpu 使用率怎么样，硬盘使用情况，api 的响应速度都保持在多少、响应时间都在多少？还没等说完，领导就打断了。他不关心这些细节，更不懂这些技术。他想要的是一个结果。比如说服务器故障率在 0.001，api 的响应率在 99.99%。这就是 IT Services 的功能。

IT service 结构如下：

```
IT Service
|
|-Workstations
||
||-Workstation1
||
||-Workstation2
|
|-Servers
```

IT Sverices 示例

举个例子，API 的 SLA，各个子 Service 都有他的可用率，然后 XXX 网站 API 可以统计到整个 API 的可用率，当领导过问起来，给他看这个就行了。

```
IT Service
|
|-XXX 网站 API
||
```

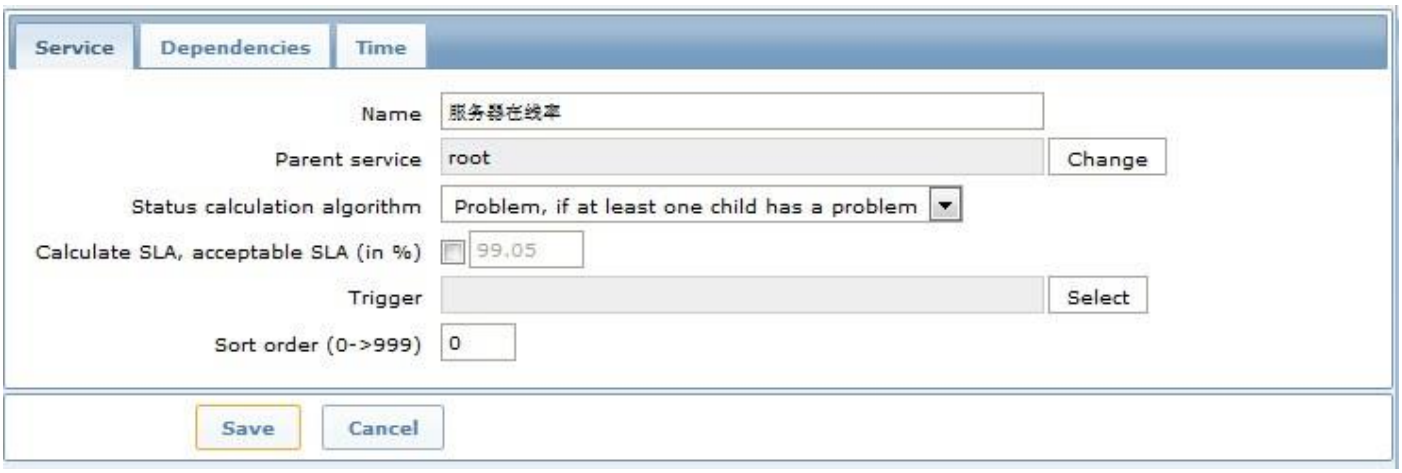
```
||-天气 API
||
||-新闻 API
||
||-用户 API
|
||-....xxxAPI（省略各种 api）
|
|-Servers（其他 services）
```

那这些可用率是怎么计算出来的呢？根据你的触发器，除了未分类和信息这两类，其他严重性级别，例如警告（warnning）等等都会记入故障率

配置 IT Services

configuration->IT Services->单击 root->Add services

创建服务器在线率



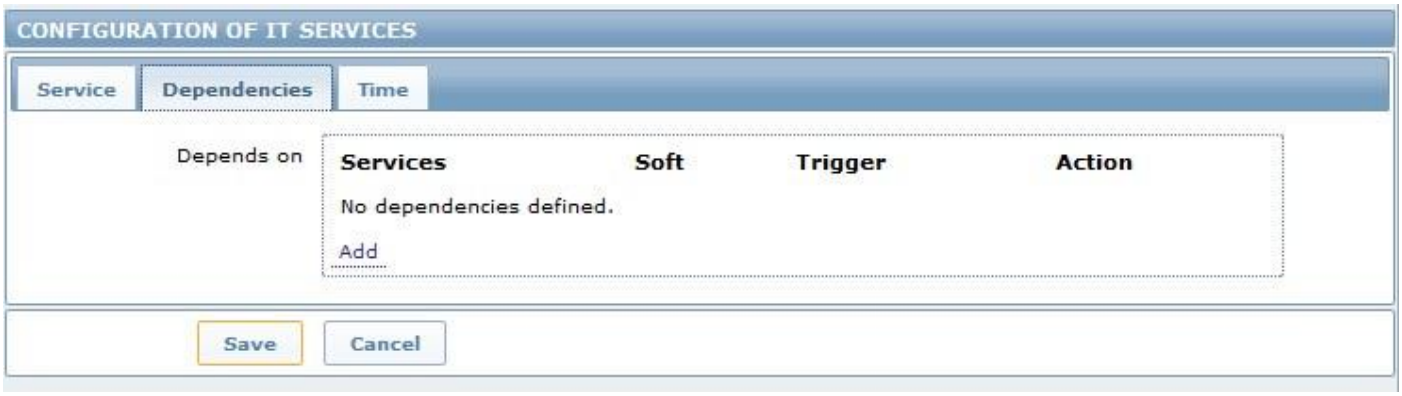
service 说明

属性	说明
name	名称
Parent service	上级节点，这边是 root
Status calculation algorithm	计算付费，共有三个选项: **Do not calculate** - 不加入计算 **Problem, if at least one child has a problem** - 子项至少一个发生故障（一般用这个） **Problem, if all children have problems** - 所有子项都发生故障，才加入计算
Acceptable SLA (in %)	可接受的可用率百分比，如果在大于这个百分比那么现实绿色，如果小于那么就是红色显示
Trigger	触发器，可以选触发器也可以不选，不过大家要记住，可用率计算的就是这些触发器的可用率，如果没有触发器根本无法计算。最上级的可以不选触发器，子项一定记得选择触发器，否则就失去意义了。

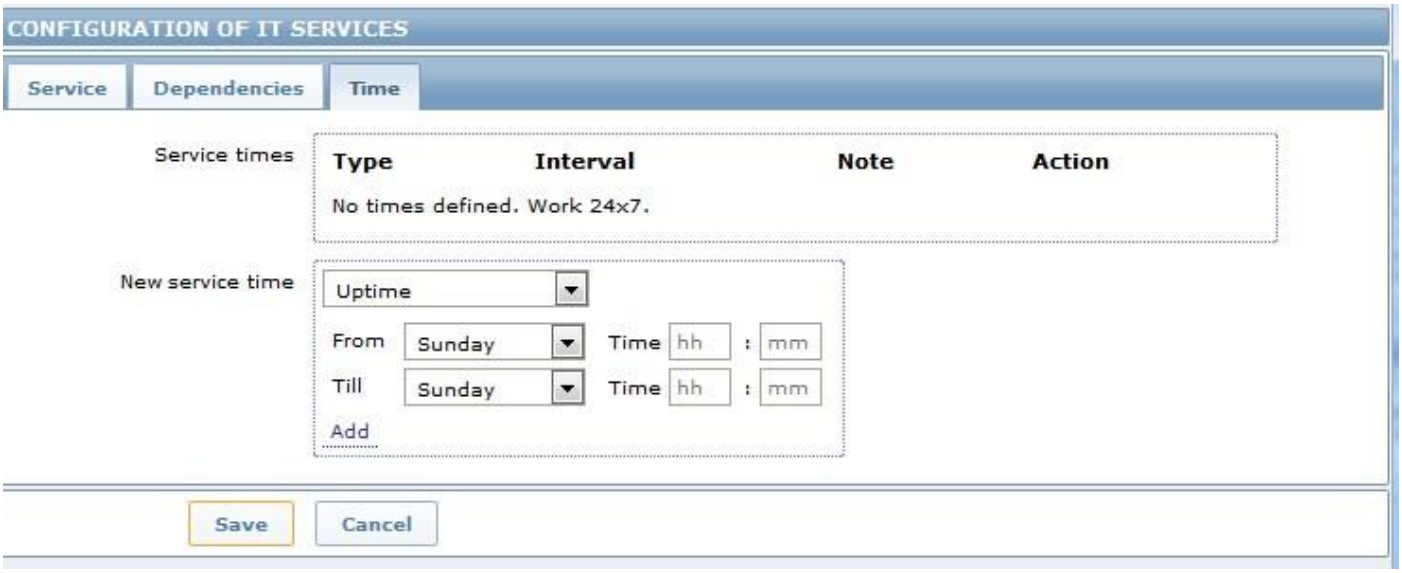
添加子 service

依赖标签

这边我们不增加依赖，在后面我们专门来谈谈这个依赖



Time 这边如果默认，那么就是 24×7



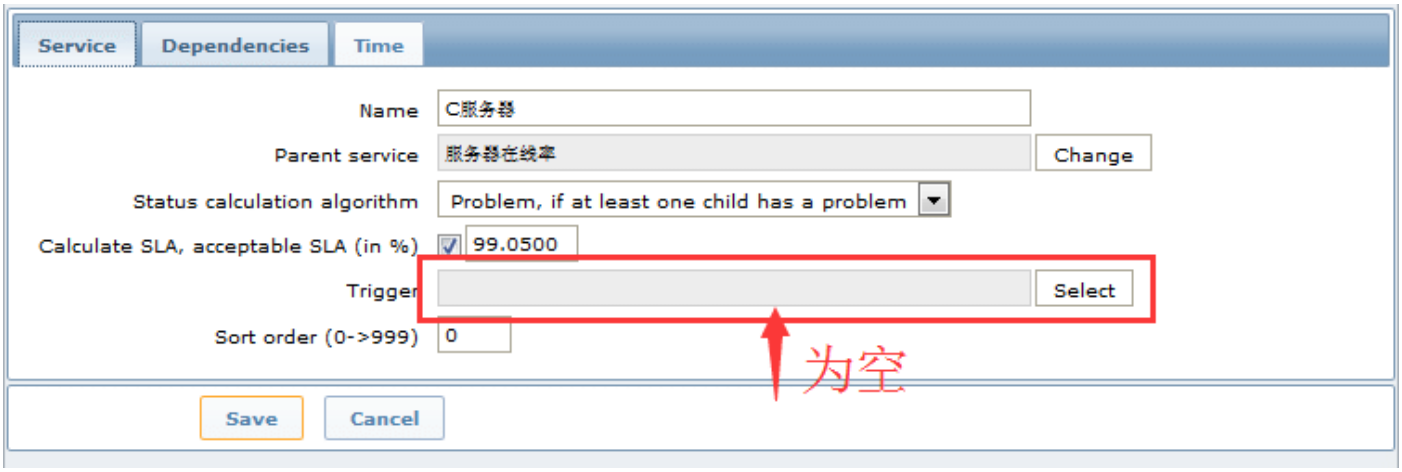
Time 说明

属性	描述
Service times	定义好的工作时间
New service time	一共有三个选项 **Downtime** – 在这个时间段，不计入 SLA **One-time downtime** – 在这个时间段，不计入 SLA，指定一个时间（只有一次） **Uptime** ： 工作时间，在这个时间内出现故障都计入 SLA

看看效果，monitoring – >IT services

IT Services 依赖

分为 hard 和 soft 依赖，例如我们增加一个 C 服务器，他需要依赖其他 IT 树下的 services，首先它不能链接触发器，在依赖那边选择其他树下依赖即可，可以添加多个，软依赖是灰色的标识，硬件依赖则是直接把整个 service 挪过来。如果 C 服务器使用软依赖，那么可以直接删除 C 服务器 Service，如果是硬依赖，需要先移除依赖，才能删除。

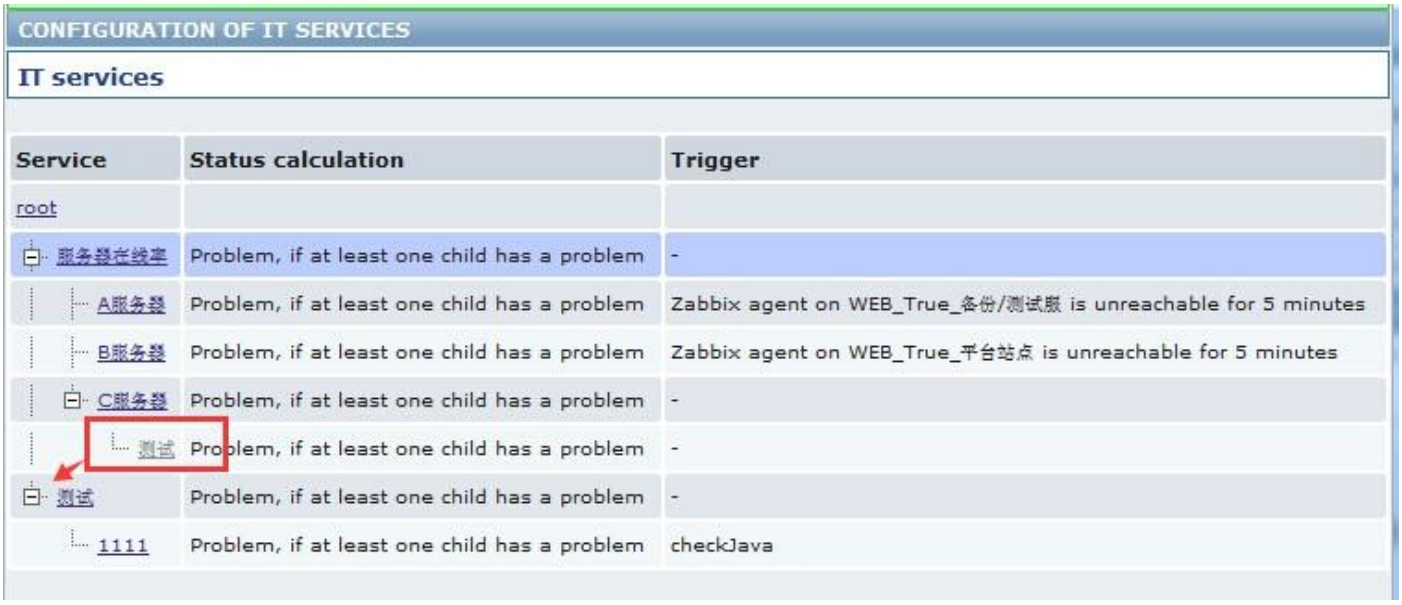


soft 不勾选，表示为硬依赖

如下，原本“测试”和“服务器在线率”在同一个层级，都归属于 root，但是加了硬依赖之后，直接到了 C 服务器只下了

接着来看看软依赖,勾选 soft，就是软依赖了

看下图，和硬依赖很不相同，C 服务器下的测试是灰色的，并且“测试”依旧和“服务器在线率”在同一个层次。



此时你可以直接删除 C 服务器，但是硬依赖的情况下不行哦。zabbix IT SERVICES 就到这里了，可以给领导开个权限，这样他也可以了解到服务器整体状况了。运维们也需要经常看，毕竟这是调整的一个一句。

第九章：WEB 监控

zabbix 监控 web 服务器访问性能

在 host 列可以看到 web(0),在以前的版本这项是独立出来的，这个主要实现 zabbix 对 web 性能的监控，通过它可以了解 web 站点的可用性以及性能。最终将各项指标绘制到图形中，这样我们可以了解到一个站点的下载速度、响应速度等。需要注意的是在安装 zabbix server 需要增加 libcurl 的支持。

我们只需要配置后 web 监控项，那么 zabbix server 会定时按照你的规则去执行性能监控。特性下，如果配置都差不多，大家可以先创建模板，然后套用下模板即可

web 检测数据搜集说明

web 整个检测中会收集如下数据

- 整个 web 监控规则中的页面平均下载速度，秒为单位
- 检测阶段发生的错误次数
- 最后一个错误消息

web 检测的任何一个阶段都会收集如下数据

- 每秒的下载速度
- 响应时间
- 响应代码（http code，如 200、301 等）

zabbix web 监控说明

zabbix 可以检测 http、https 协议，而且 zabbix 也支持重定向，执行过程中的所有 cookies 也会被保留。如果需要的话，zabbix 会检索某个页面是否包含特定的字符，如果有表示成功，没有表示失败，例如检测 zabbix 登陆是否正常，它会检索响应的 html 页面中是否包含 Admin，如果有表示登陆成功。

zabbix web 数据保存

每次执行完之后的数据都会保存到 zabbix 数据中，这些数据可以用户绘制成图表以及用户 zabbix 触发器和发送报警通知

zabbix web 监控项 item 详解

一旦我们创建好 web 监控之后，我们便可以查看 web 站点的性能状况。zabbix 一共给我们提供了 6 个 item key，实际上就三个，分别针对单个阶段和整个阶段，三个 item 分别为 web.test.in、web.test.fail、web.test.error，下面看看它的具体用法。

web 方案监控项

当 web 监控项创建好之后，下面的 key 会被自动添加好

key	描述
web.test.in[Scenario,,bps]	整个阶段中的下载速度，单位字节/秒 类型: Numeric(float)
web.test.fail[Scenario]	整个检测阶段，失败的阶段个数，如果所有的阶段（step）都成功，那么返回 0 类型: Numeric(unsigned)
web.test.error[Scenario]	返回最后一个错误信息（文本）

web 监控项实例

- 创建触发器“Web scenario failed”,表达式如下

```
{host:web.test.fail[Scenario].last(0)}#0
```

- 创建触发器“Web application is slow”,表达式如下

```
{host:web.test.in[Scenario,,bps].last(0)}<10000
```

备注：Scenario 改成你 web 方案的名称即可

web 方案阶段监控项

key	描述

web.test.in[Scenario,Step,bps]	检索指定阶段的下载速度，字节每秒 类型: Numeric(float)
web.test.time[Scenario,Step]	获取指定阶段响应时间，时间计算从开始请求道获取到所有响应信息之后 类型: Numeric(float)
web.test.rspcode[Scenario,Step]	检索指定阶段的 http 响应代码 类型: Numeric(unsigned)

step item 使用实例

创建触发器 “Zabbix GUI login is too slow” trigger，触发器表达式如下

```
{zabbix:web.test.time[ZABBIX GUI,Login].last(0)}>3
```

说明：ZABBIX GUI 是 web 方案的名称，Login 为阶段（step）名称

web 监控项数据保留时间

web 监控历史数据数据保存 30 天，趋势数据保存 90 天，老数据将被清除

zabbix 实战监控 WEB 网站性能

一直在纠结用什么实例来给大家演示呢？想来想去还是官方的好，那我们怎么用 zabbix 监控 web 性能和可用性呢？我们这边分为几个步骤：打开网站、登陆、登陆验证、退出，一共 4 个小 step，看实例。

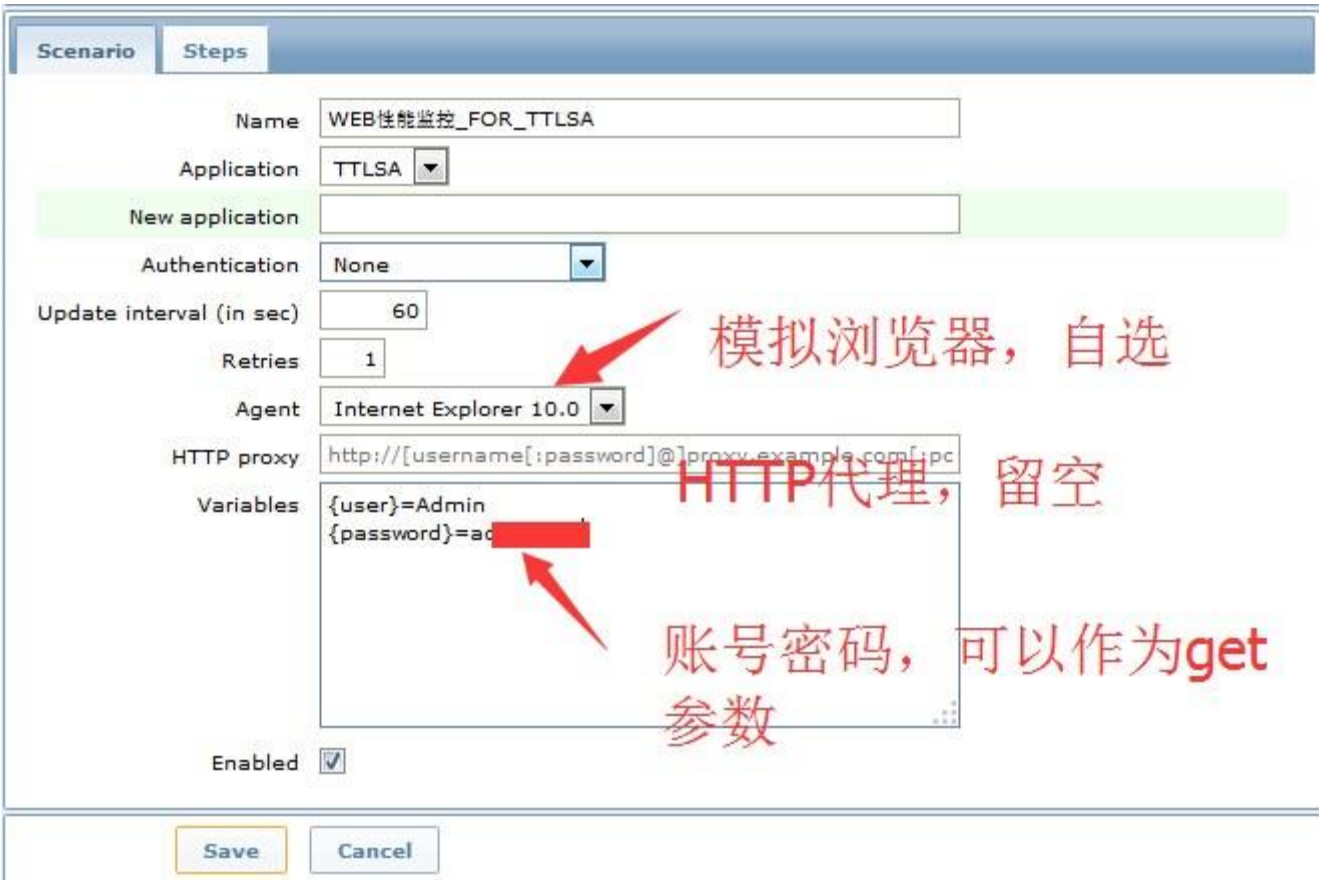
检测流程

1) 打开网站：如果 http code 为 200，并且响应的 html 中包含 Zabbix SIA 表示打开成功（zabbix 页面有这个标示）
2) 登陆后台：post 用户名和密码到 index.php，如果响应 200，那表示 post 成功。并且通过正则表达式从响应的 html 中匹配 sid，这个 sid 也就是一个宏变量，退出可以使用到
3) 验证登陆：打开首页，检索 html 中是否包含 Profile（只有登陆成功，才会有 Profile 出现）
4) 退出账号：传递参数 sid 给 index.php 即可退出，响应 200 即表示退出成功。

我们可以使用上节讲到的 item key 来获取每个 step 的速度以及响应时间或者说最新的一个错误消息，大家自己去研究吧，不难

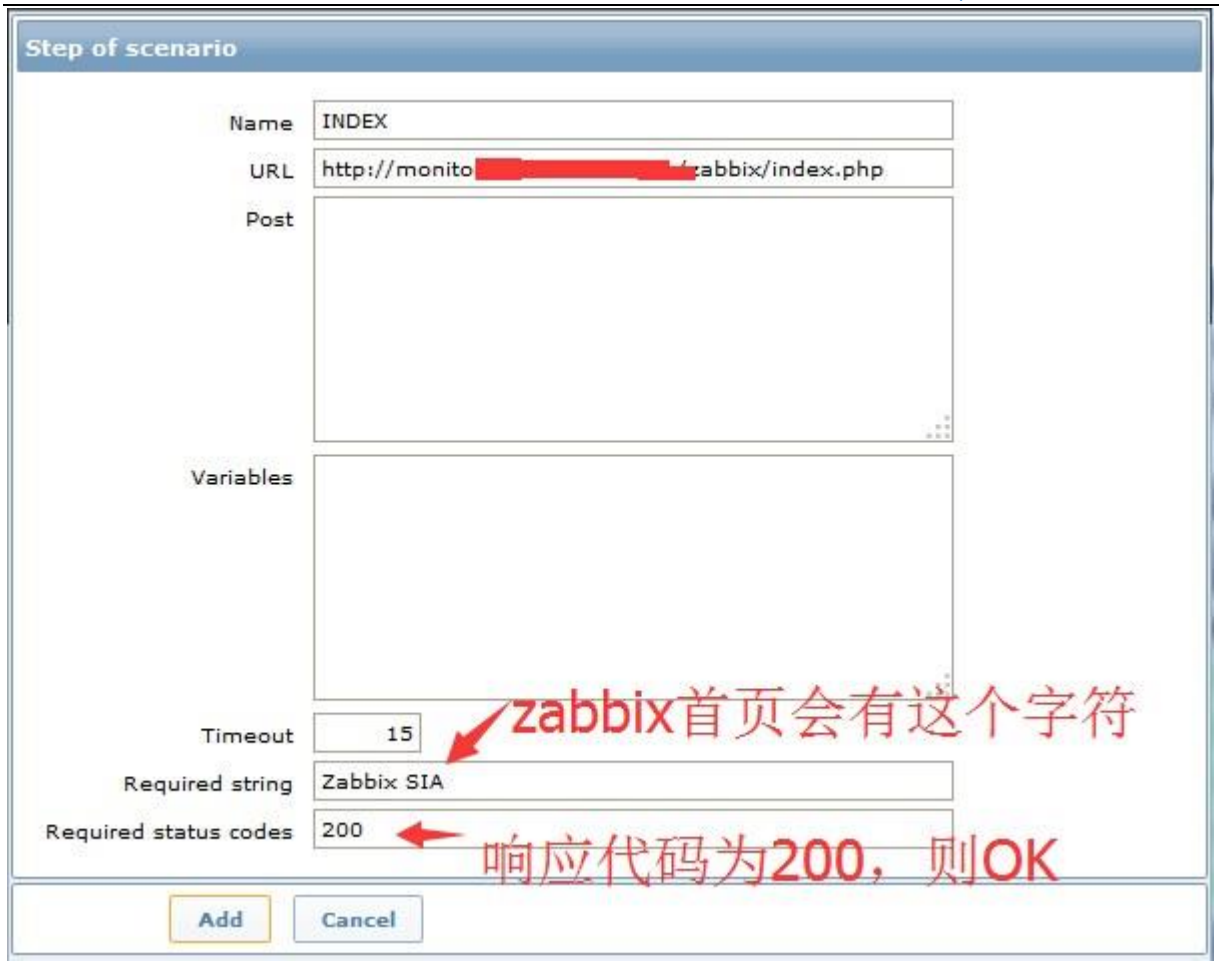
创建 WEB 场景

configuration->Host->你的主机->web->右上角 Create scenario

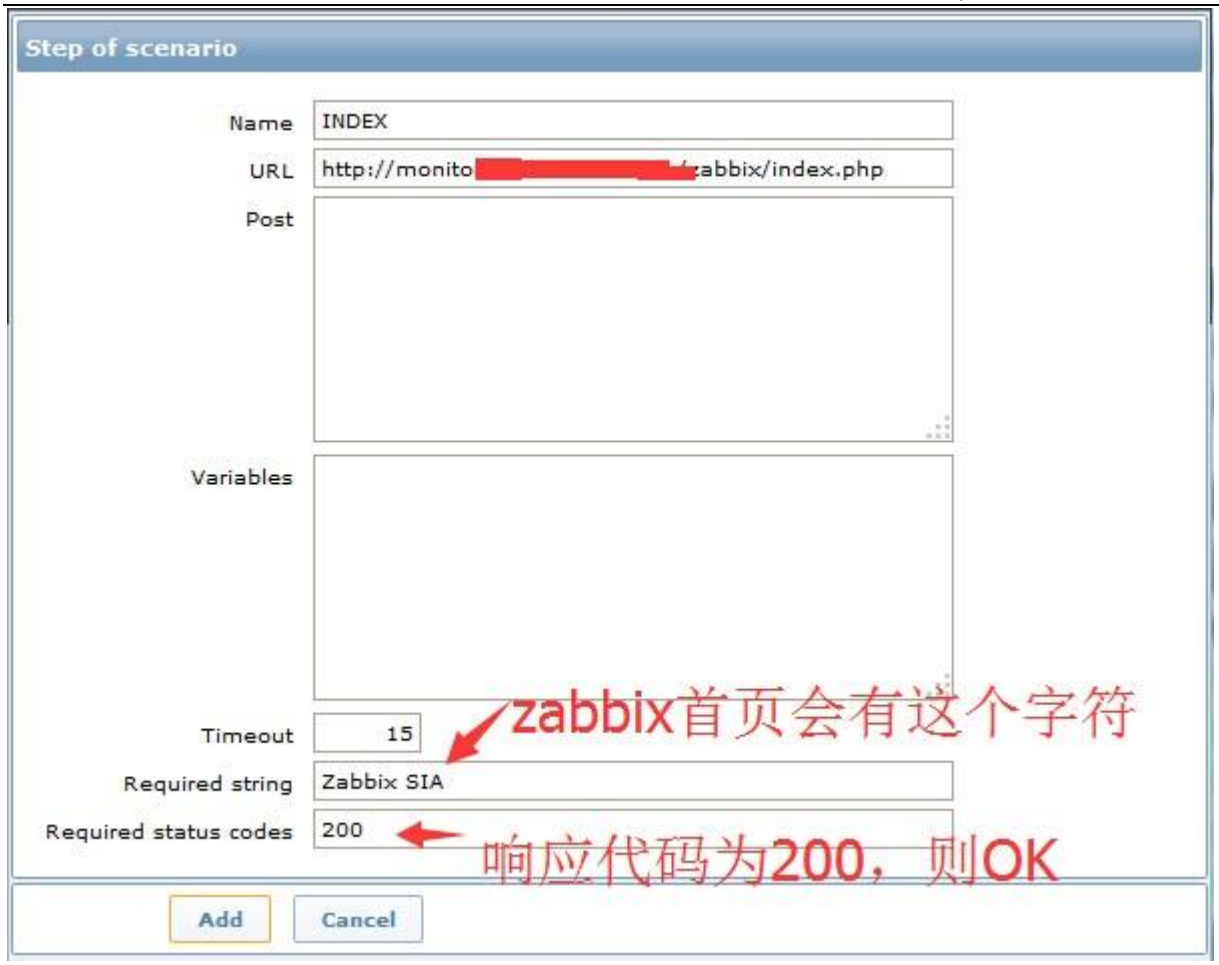


属性表

属性	描述
Name	监控项的名称
Application	放到哪个应用中，《什么是 Application》
Authentication	是否有 http 的基本认证，大部分情况下是 None，难不成用户进来还需要经过一次认证？
Update interval	更新周期，默认 60 秒,多久跑一次
Retries	重试次数
Agent	模拟浏览器
HTTP proxy	代理，如果你的站点有多台服务器，那么请写上你目标服务器 ip 和端口，例如 http://10.9.0.2:80，默认端口可不是 80，别忘记 80 了
Variables	宏变量，后面会用到。想了解请点《zabbix 用户宏 macro》



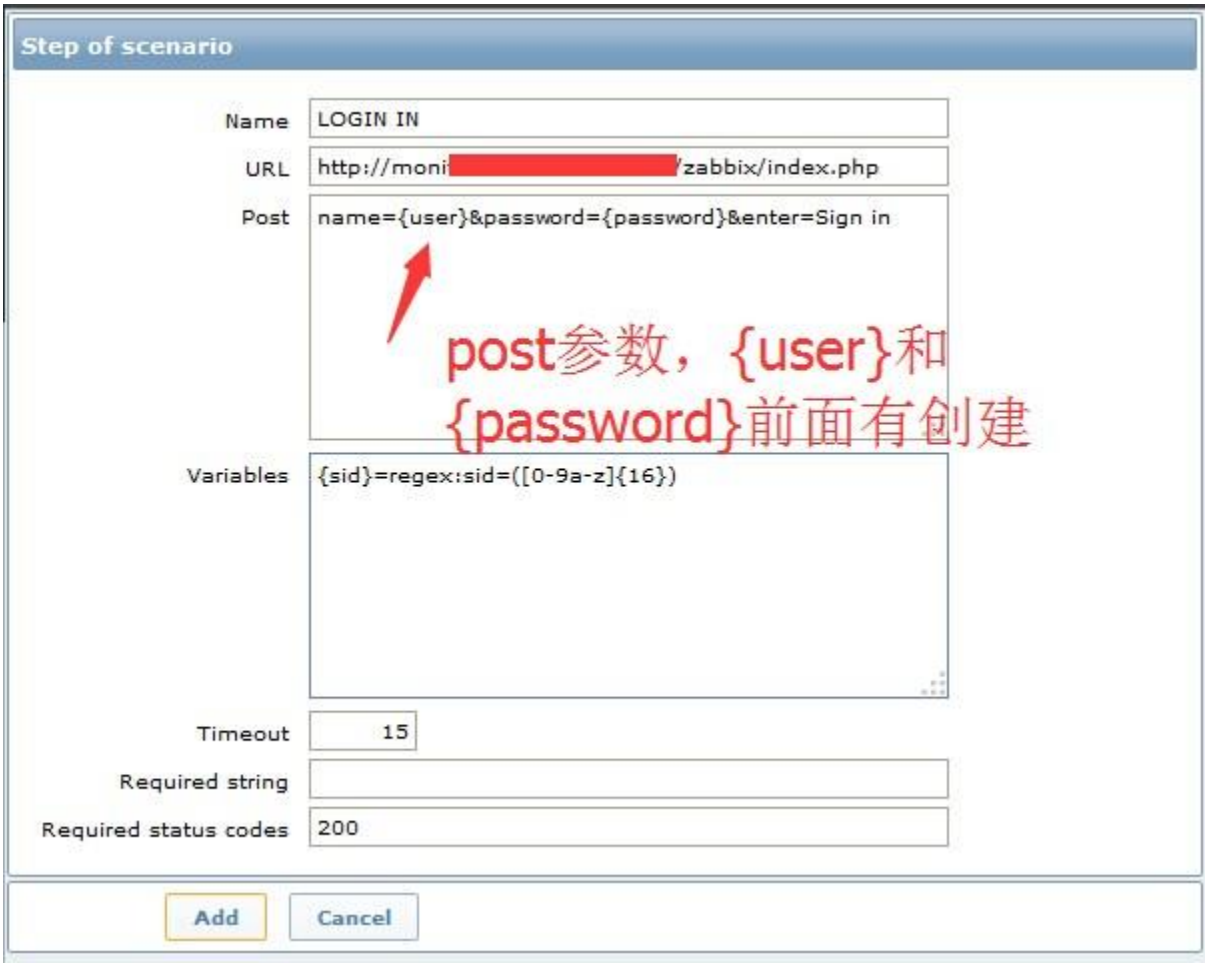
web 监控阶段 1：打开首页



对 step 做一个说明：

属性	说明
name	当前 step 名称，item key 中可以用到
url	需要检测的网址
POST	你需要 post 提交上去的内容，例如 user=123&password=123456,，或者使用宏变量 user={user}&password={password}，如果支持 GET，那么可以直接写到 URL 里面
variables	变量，这边定义宏变量后续的 step 可以使用
Timeout	超时时间，默认 15 秒
Required string	响应的内容中必须包含的字符串，否则失败
Required status codes	响应代码必须包含在里面，多个响应代码用逗号分隔，例如 200,301,302

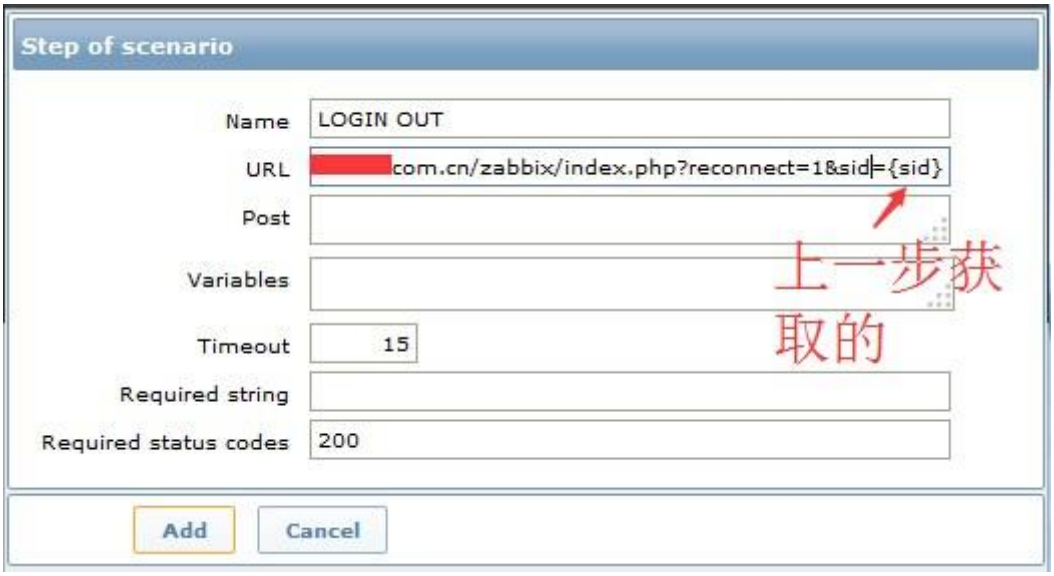
web 监控阶段 2：登陆



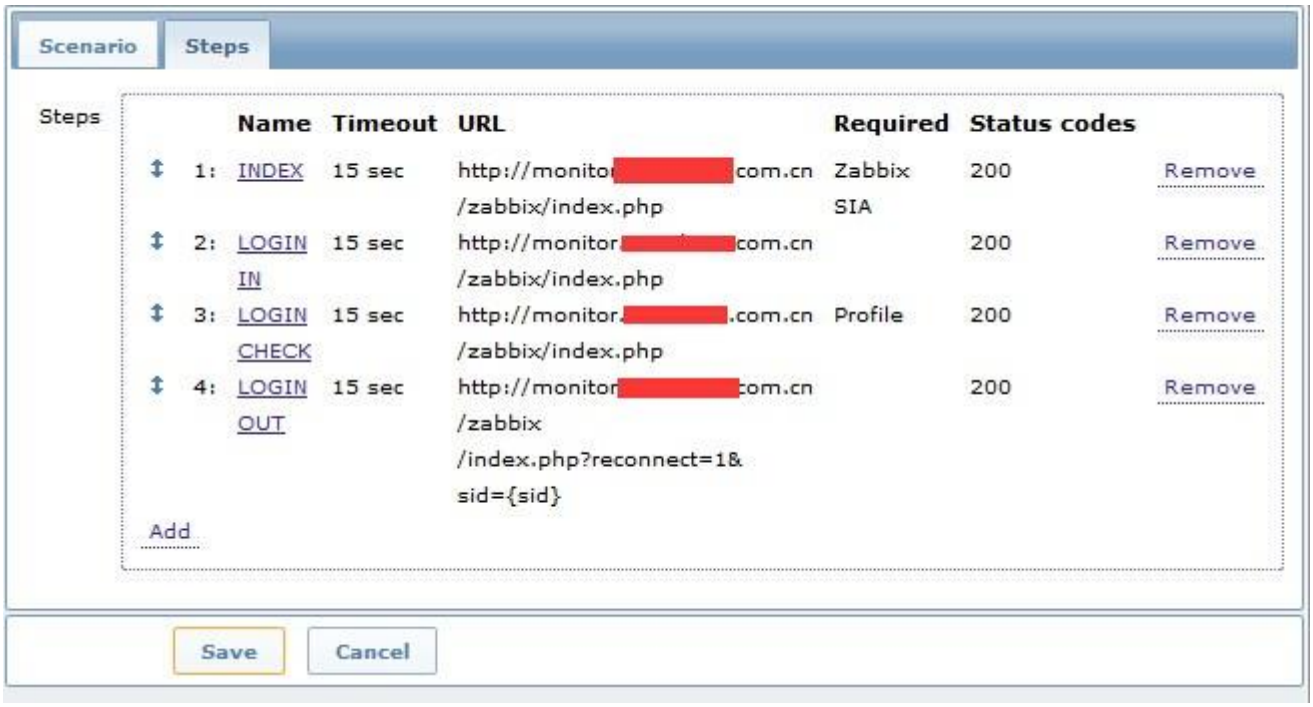
post 账号和密码上去，关于 post 在前面已经提过了。

WEB 监控阶段 3：验证登陆

WEB 监控阶段 4：退出账号



WEB 网站检测配置完成，记得保存



查看结果

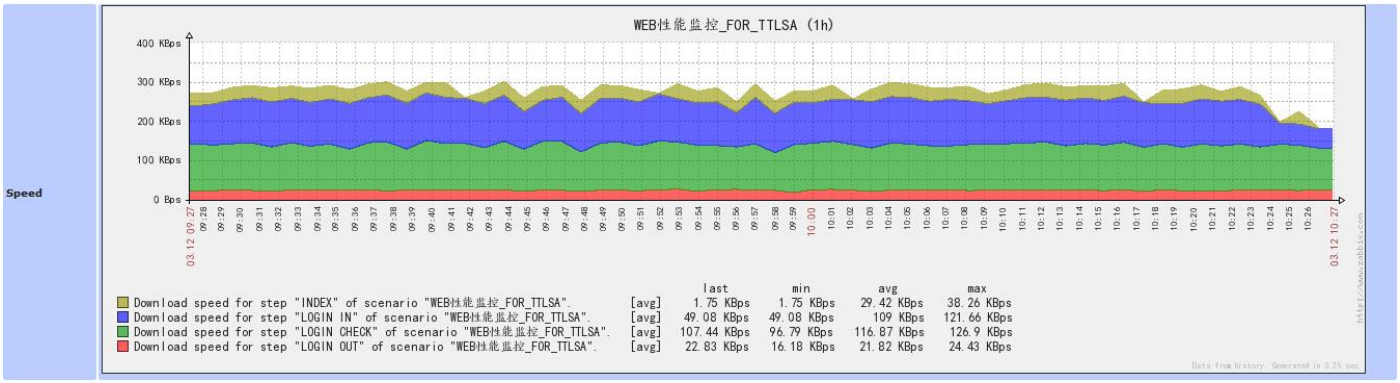
monitorning->web->筛选出你的主机->查看“WEB 性能监控_FOR_TTLSA”，结果如下图，各个阶段的响应时间、速度、返回状态码以及总的响应时间

DETAILS OF SCENARIO WEB性能监控_FOR_TTLSA [02 Dec 2014 18:42:09]

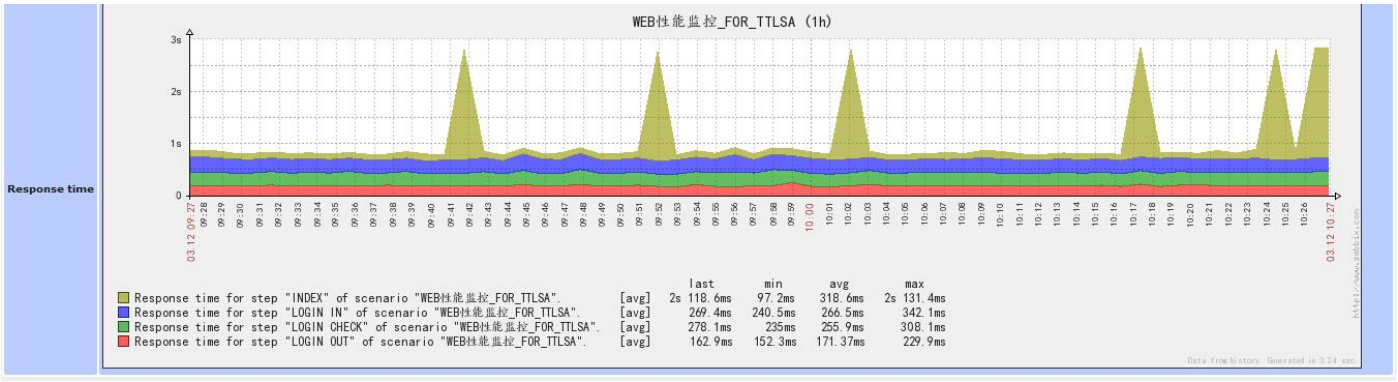
Step	Speed	Response time	Response code	Status
INDEX	38.98 KBps	95.4ms	200	OK
LOGIN IN	57.88 KBps	228.2ms	200	OK
LOGIN CHECK	112.72 KBps	257.9ms	200	OK
LOGIN OUT	20.81 KBps	178.8ms	200	OK
TOTAL		**760.3ms**		**OK**

Filter

下图是下载速度的图表，包含各个阶段



下图是响应时间的图表



以上是没问题的信息，那么出现故障是什么样子呢？我把密码改掉，演示给大家看看下图，在 LOGIN IN 这个 step 就出错了，拿不到 SID

DETAILS OF SCENARIO WEB性能监控_FOR_TTLSA [03 Dec 2014 10:31:41]

Step	Speed	Response time	Response code	Status
INDEX	32.09 KBps	115.9ms	200	OK
LOGIN IN	41.51 KBps	90.1ms	200	Error: error in step variables "{sid}=regex:sid=([0-9a-z]{16})": cannot extract the value of "{sid}" from response
LOGIN CHECK	-	-	-	Unknown
LOGIN OUT	-	-	-	Unknown
TOTAL		**206ms**		**Error: error in step variables "{sid}=regex:sid=([0-9a-z]{16})": cannot extract the value of "{sid}" from response**

那么 Required String 不匹配又是什么样子呢？我们把阶段 3Login CHECK 的 required string 的 Profile 改成 Profile1 试试。看看结果

DETAILS OF SCENARIO WEB性能监控_FOR_TTLSA [03 Dec 2014 10:36:45]

Step	Speed	Response time	Response code	Status
INDEX	31.64 KBps	117.6ms	200	OK
LOGIN IN	53.58 KBps	247.4ms	200	OK
LOGIN CHECK	118.29 KBps	252.9ms	200	Error: required pattern not found
LOGIN OUT	-	-	-	Unknown
TOTAL		**617.9ms**		**Error: required pattern not found**

好了，web 监控的实例就完成了。

zabbix 监控 API

现在各种应用都走 api，例如淘宝，天气预报等手机、pad 客户端都是走 api 的，那么平时也得对这些 api 做监控了。

怎么做呢？zabbix 的 web 监控是不二选择了。今天就以天气预报 api 作为一个例子。

天气预报 API

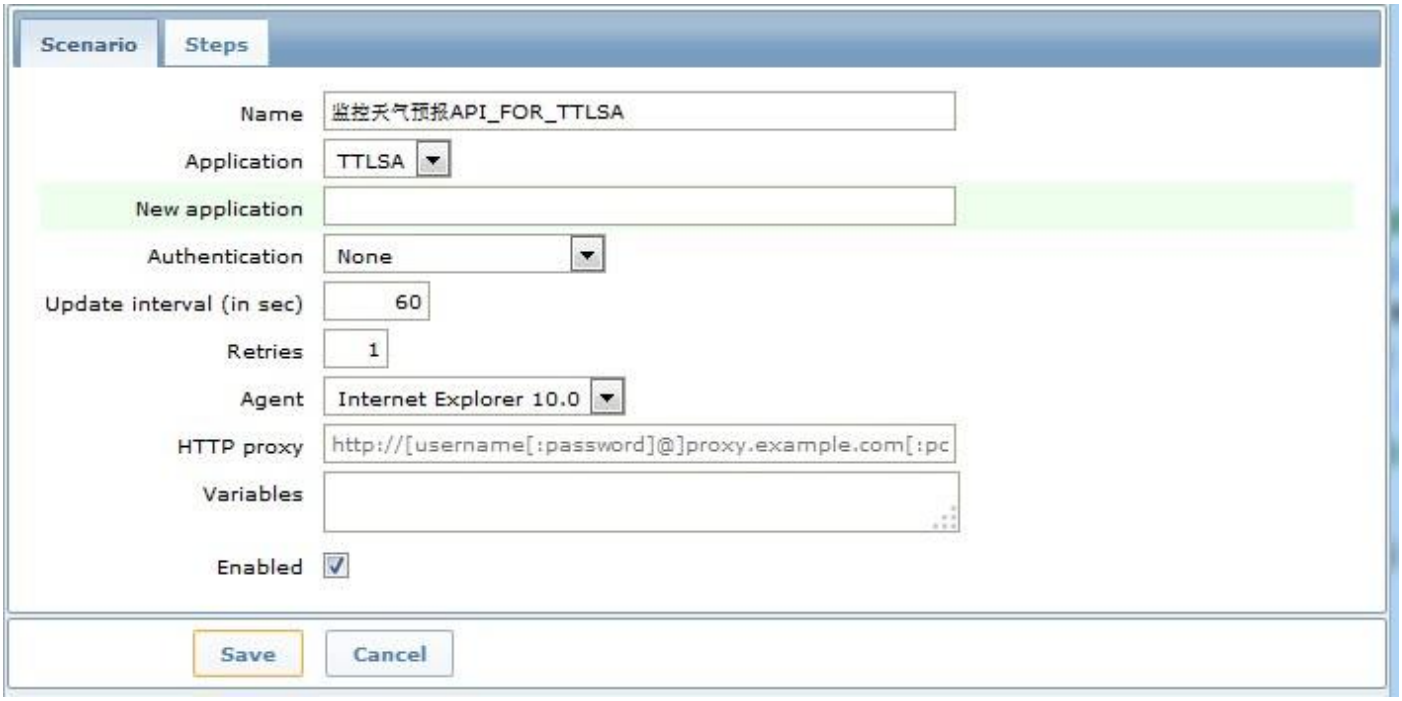
天气预报 api 地址：http://www.weather.com.cn/data/sk/101010100.html

api 正常情况下会返回如下数据：

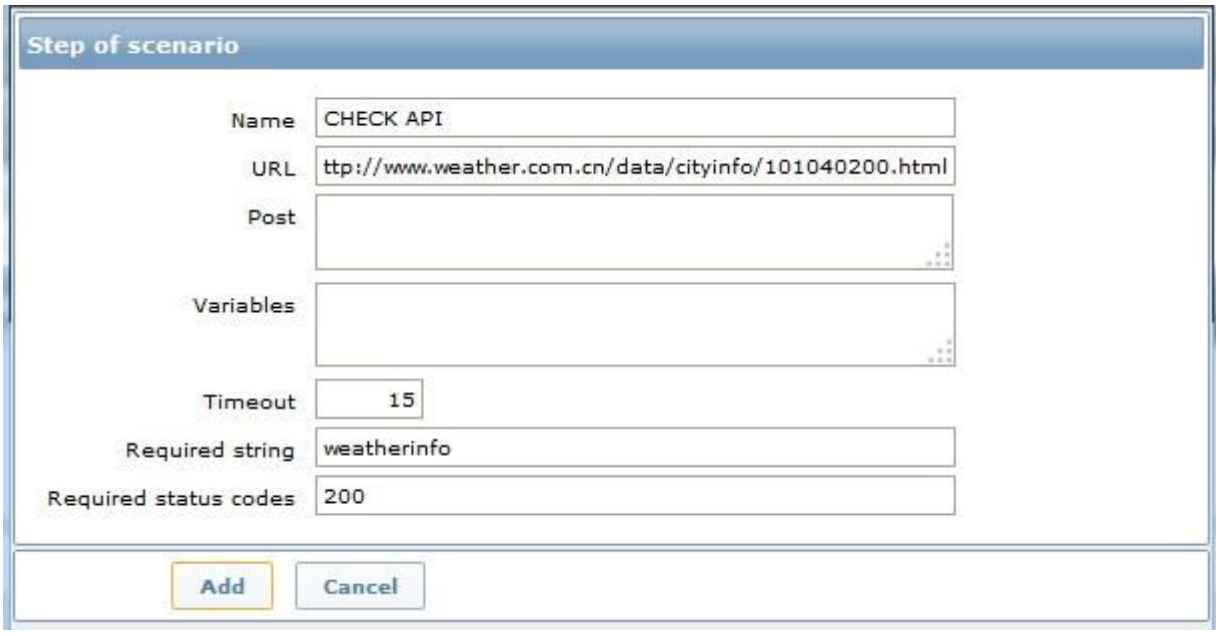
```
{
    "weatherinfo": {
        "city": "北京",
        "cityid": "101010100",
        "temp": "-1",
        "WD": "北风",
        "WS": "3 级",
        "SD": "12%",
        "WSE": "3",
        "time": "11:15",
        "isRadar": "1",
        "Radar": "JC_RADAR_AZ9010_JB",
        "njd": "暂无实况",
        "qy": "1021"
    }
}
```

ZABBIX WEB 场景配置

configuration->host->您的主机->web->点击右上角 create scenario



点击 step，输入如下



查看监控结果

monitoring->web->选择相应的 hosts，点击如下的“监控天气预报 API_FOR_TTLSA”

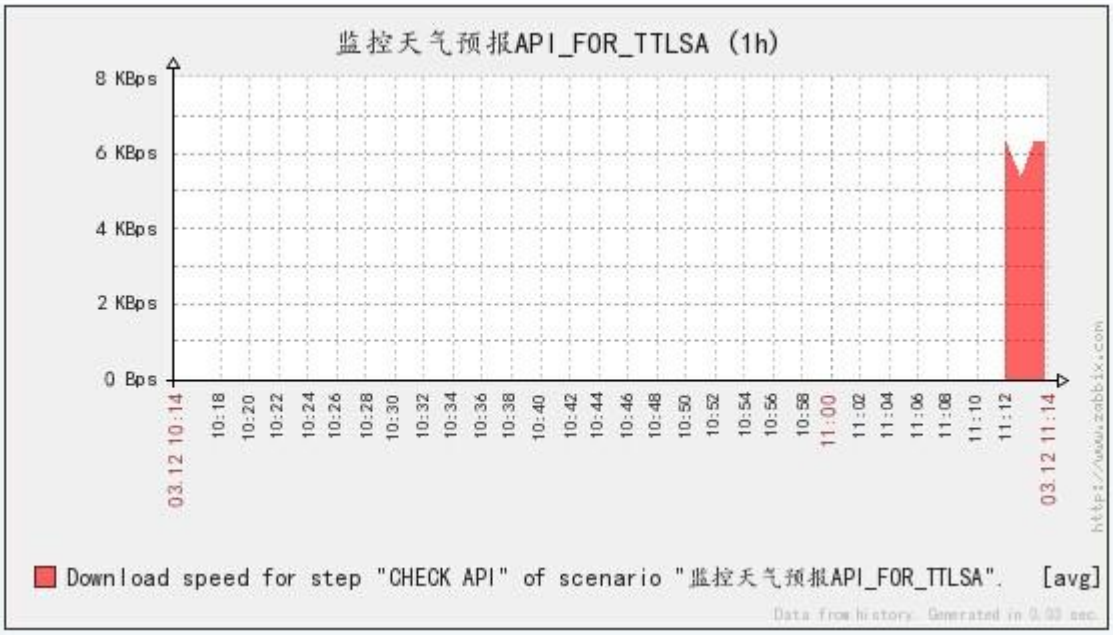
Name	Number of steps	Last check	Status
WEB性能监控 FOR TTLSA	4	03 Dec 2014 11:11:33	Step "LOGIN CHECK" [3 of 4] failed: required patter
检测G TTLSA COM	1	03 Dec 2014 11:11:20	OK
监控天气预报API FOR TTLSA	1	Never	Unknown

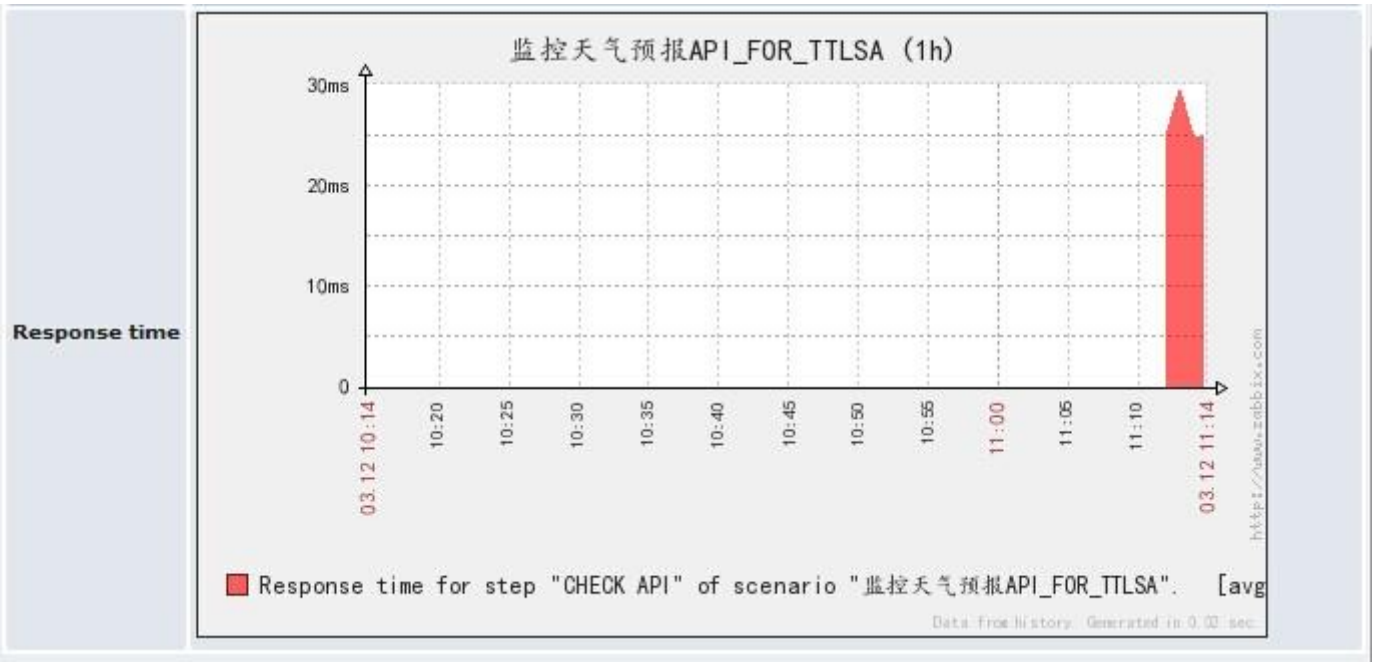
DETAILS OF SCENARIO 监控天气预报API_FOR_TTLSA [03 Dec 2014 11:15:10]

Step	Speed	Response time	Response code	Status
CHECK API	3.6 KBps	42.8ms	200	OK
TOTAL		**42.8ms**		**OK**

Filter

Speed





创建触发器

至于怎么创建触发器，我这边就不多说了，请看关于触发器的文章《zabbix 创建触发器》，当有故障发生，便可以发送故障报警。

zabbix 监控 api 说明

以上只是一个简单的例子，具体应用看大家了，比如说可以监控注册、获取新闻列表、获取评论等等接口是否可以使用，以及这些接口的一些性能。

第十章：维护模式

zabbix Maintenance 维护周期

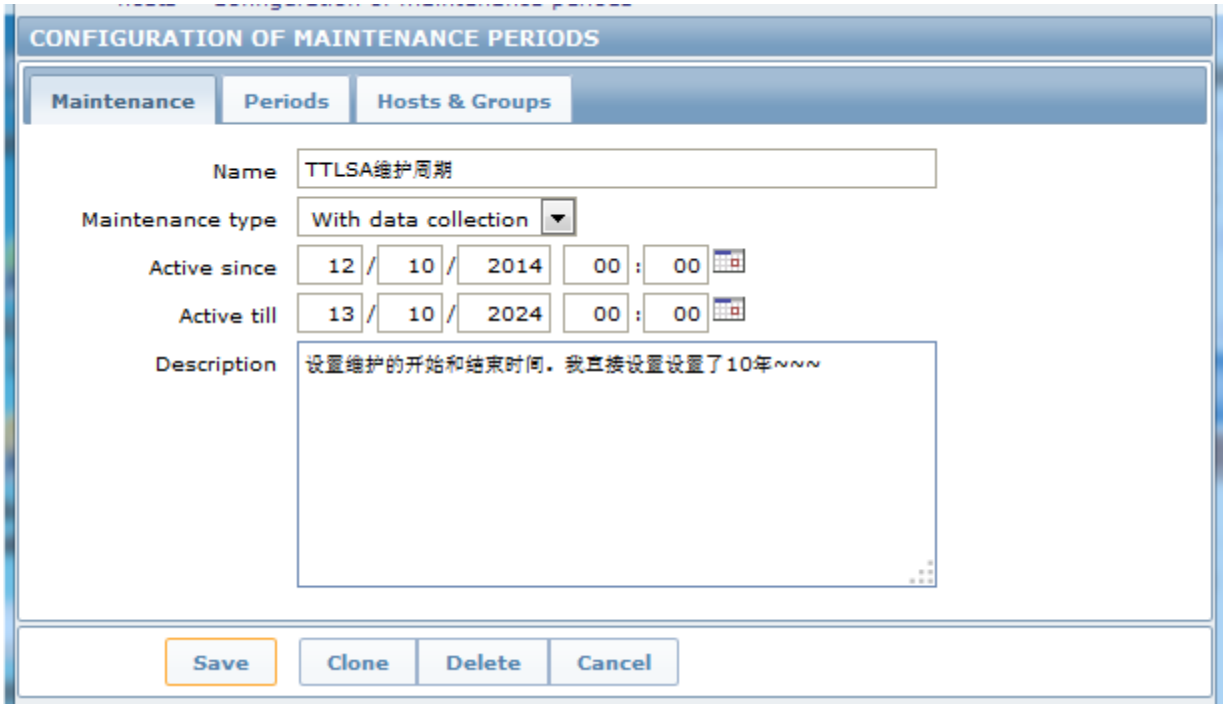
概述

我们可以给 zabbix 某些组或者某些 Hosts 设置维护时间,zabbix 提供两种维护类型：依旧收集数据、暂停收集数据，在服务器维护期间不会生成报警（前提：触发器设置了’Maintenance status = not in “maintenance”’），如果在维护期间出现故障，并且没有解决掉，那么在维护周期结束之后，服务器会生成报警.如果你想在维护期间也能收到报 警，那么触发器不需要设置’Maintenance status = not in “maintenance”’ .

配置

配置维护周期

点击 Configuration（配置） → Maintenance（维护）—>点击 Create maintenance period （创建维护周期），配置如下



参数	描述
Name	维护名称

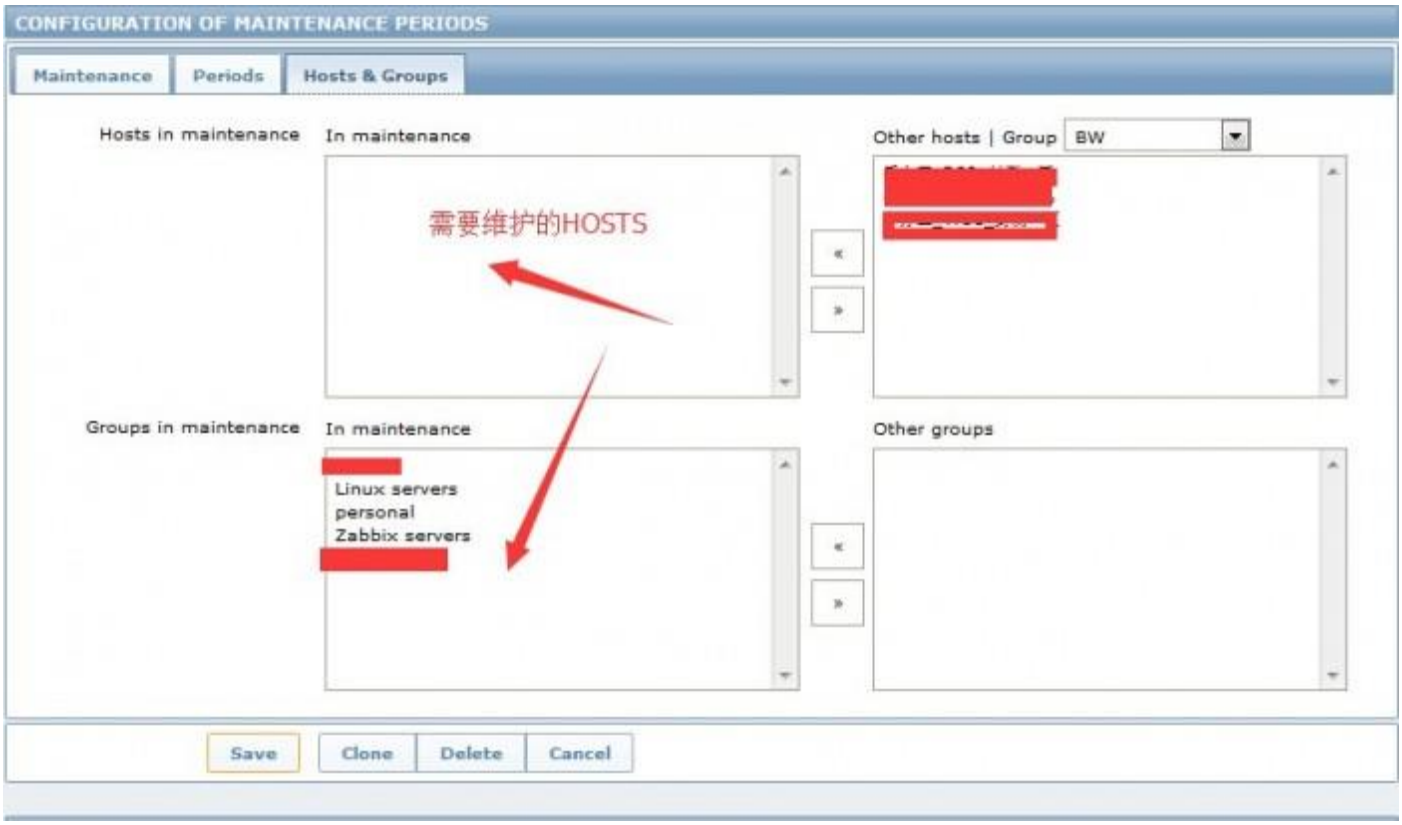
Maintenance type	两种维护类型可选: **With data collection** – 依旧收集数据 **No data collection** – 暂停收集数据
Active since	维护周期开始时间
Active till	维护结束时间
Description	描述

Periods

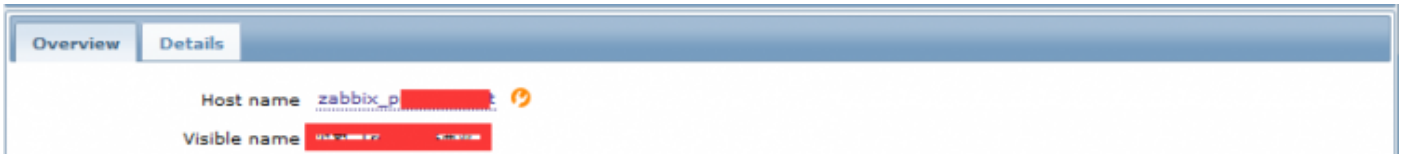
选项卡是维护周期的，可以选择 daily, weekly, monthly or one-time，我这边的例子是每周一凌晨 6 点开始维护，持续 2 个小时，也就是到八点结束.如果你想每天执行，也可以选择 daily 或者在 weekly 里选择周一到周天.具体用法我不在详讲了，大家看了就明白.

Hosts & Groups

选项卡里面，选择需要维护的主机或者组.



维护标识在 inventory－>HOSTS->host inventory 的 overview 里面可以看到维护的标示（扳手），如下图



或者在 HOSTS 列表里面，status 显示 In maintenance.


Status
Availability
In maintenance
In maintenance
In maintenance
In maintenance
In maintenance
In maintenance
In maintenance
In maintenance

第十一章：事件确认

事件确认（Event acknowledgment）

概述

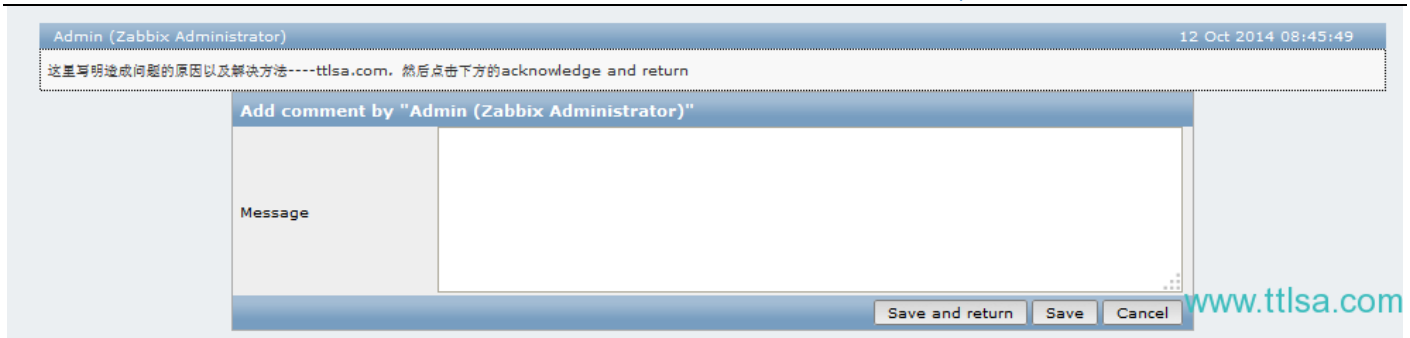
以往服务器出现报警，运维人员处理完事之后，报警自动取消，但是下一次出现同样一个错误，但是换了一个运维人员，他可能需要重新排查问题，直到问题处理完毕。针对这种情况，zabbix 提供了 event acknowledgment（事件确认）功能，一旦处理好某个问题，运维认为可以再里面写上备注，说明造成此问题的原因以及处理方法，下一次运维人员遇到这个报警先看前一次的事件确认。

Acknowledgment 也可以在 action 中使用，一旦运维人员没有及时填写事件确认，可以向他的主管或者经理发送一个通知：xxx 人员没有写事件确认.

事件确认界面

在 zabbix 首页的 last 20 issues，在每条报警列都有 Ack，如果是 NO，说明还没有对事件进行确认，如果是 Yes，表明已经提交了事件描述.

这里写明造成问题的原因以及解决方法，然后点击下方的 acknowledge and return,如果下次还出现类似故障，运维人员可以看到如下内容


Admin (Zabbix Administrator)
12 Oct 2014 08:45:49
这里写明造成问题的原因以及解决方法----ttlsa.com，然后点击下方的acknowledge and return
Add comment by "Admin (Zabbix Administrator)"
Message
Save and return
Save
Cancel
www.ttlsa.com

第十二章：网络发现

zabbix 网络发现介绍 Discovery

网络发现简介

网络发现有什么用？网络发现怎么配置？我们带着这两个问题开始我们的网络发现之旅。比如小明有 100 台服务器，不想一台台主机去添加，能不能让 zabbix 自动添加主机呢，当然可以，网络发现便是这个功能，当然前提条件是所有服务器都已经安装了 agent 或者 snmp（其实也可以不用，鉴于我们大部分功能都用 agent，所以请安装上 agent），server 扫描配置好的 ip 段，自动添加 host，自动给 host link 模板，自动加到主机组里等等。网络发现功能让我们能更快速的部署 zabbix、简化 zabbix 管理、并且在经常变动的环境里面也不需要花太多的精力，毕竟网络发现也能随时变化。虽然网络发现能干很多事情，但是它无法发现网络拓扑。

zabbix 网络发现基于如下信息

- ip 范围
- 可用的外部服务（FTP, SSH, WEB, POP3, IMAP, TCP, etc）
- 来自 zabbix agent 的信息
- 来自 snmp agent 的信息

网络发现由两个阶段组成：discovery 和 actions

Discovery 发现

zabbix 定期扫描网络发现规则中的 ip 范围，每个规则中都定义了一组需要检测的服务，在这些 ip 范围内一一扫描

网络发现模块每次检测到 service 和 host（ip）都会生成一个 discovery 事件,如下是事件

时间	条件
Service Up	zabbix 检测到可用的 service
Service Down	zabbix 无法检测到 service
Host Up	某个 ip 上至少有一个 service 是 up 状态
Host Down	所有 service 都无响应
Service Discovered	一个 service 首次被发现或者在维护后从新归队

Service Lost	service 在 up 之后又丢失了
Host Discovered	一个 host 首次被发现或者在维护后从新归队
Host Lost	一个 host 在 up 之后又丢失了

Actions 动作

zabbix 所有 action 都是基于发现事件，例如：

- 发送通知
- 添加/移除主机
- 启用/禁用主机
- 添加主机到组
- 从组中移除主机
- 主机 link 模板/unlink 模板
- 执行远程脚本命令

创建主机

discovery 发现主机事件产生之后，接下来需要执行 discovery action，在 action 中选择添加主机操作、并且将主机加入某个组以及 link 某个模板等等。更具体操作请关注下一篇文章。那么主机名怎么定义呢？首先监控端（server/proxy）通过 ip 泛解析主机名（如果失败了，不会重新尝试），如果解析成功了，那么 zabbix 将会使用这个主机名，否则直接使用 ip 地址。如果主机名相同怎么办？比如都叫 ttlsa-server，那么第一台主机名会定义为 ttlsa-server，第二台为 ttlsa-server_2，第三台为 ttlsa-server_3，以此类推。action 配置里的条件包含设备类型、IP、状态、uptime/downtime 等等。

添加主机接口

主机接口规则如下：

- 服务检测 – 例如，成功检测到一个 SNMP 服务，那么创建 snmp 接口
- 如果主机同时 UIzabbix agent 和 snmp 请求作出响应，那么会同时创建这两种接口
- 如果使用 agent 或者 snmp 作为唯一性指标，先通过哪个接口发现主机，那么哪个接口就作为默认接口，其他

的作为附加接口。

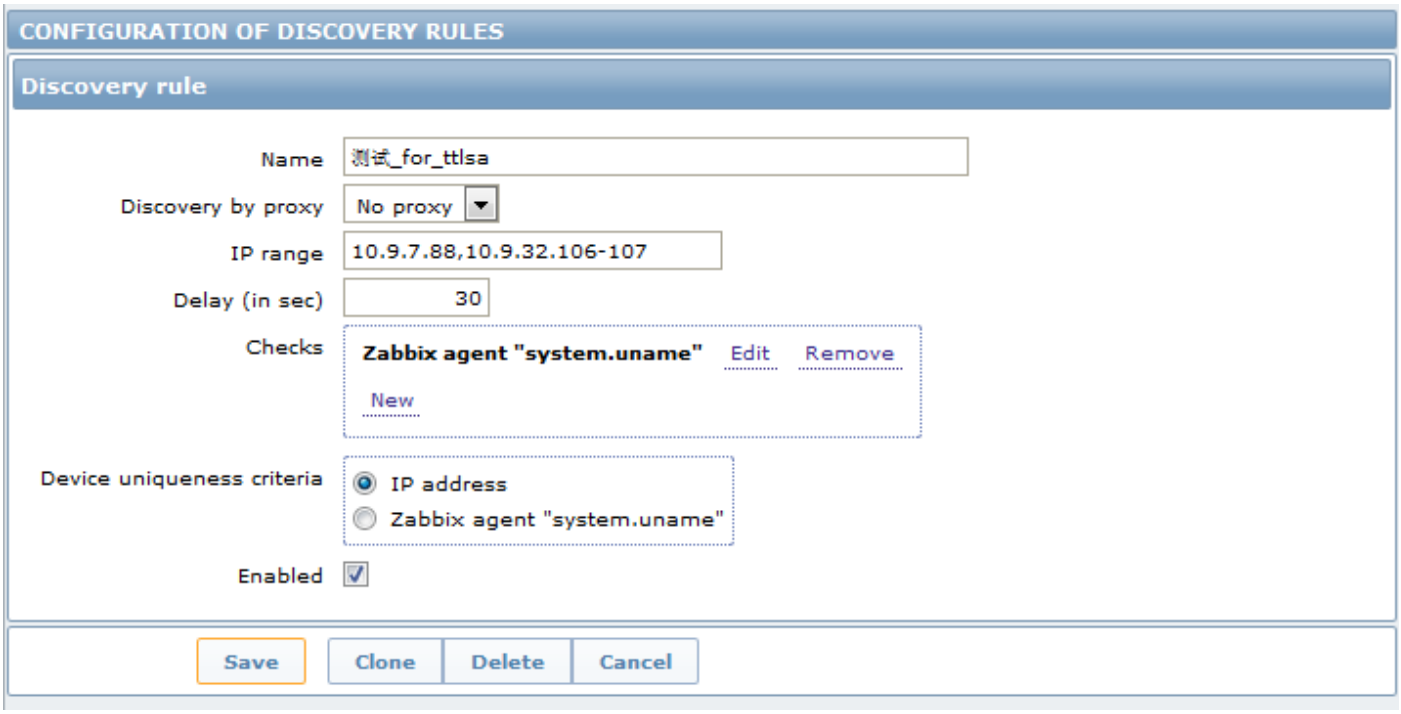
- 如果一开始只响应 zabbix agent 的检测，那么他只会创建 agent 接口。如果后面响应了 snmp 检测，那么他又会增加 snmp 接口
- 如果有三台独立的主机 A\B\C，一开始使用 IP 地址来作为唯一标识。可以看到 discovery 有三条记录。此时我们修改发现规则，让他们有相同的唯一标识。例如自定义一个 keysystem.dis，这个 key 统一输出值”ttlsa”，这样 A 的接口成了默认的，B 和 C 都变成了附加到 A 主机上。我们可以发现一个很明显的变化。在 discovery 接口中依旧有 3 条记录，但是”discovered device”这列显示的 A 主机的接口，”monitored host“这列显示的都是 A 主机的名称，”Uptime/downtime“这列只有 A 主机有值，B 和 C 都为空。从这里我们能看到唯一标识是多重要，如果唯一标识不是唯一，那么有的主机会被认为是同一台。

zabbix 网络发现规则配置实战/详解

在了解了《网络发现规则》是什么之后，我们开始配置。首先，我们需要定义发现规则，用于扫描。步骤如下

第一步

Configuration >>Discovery>>Create rule，编辑网络发现规则



如上配置，zabbix 每 30 秒会扫描 10.9.7.88 与 10.9.32.106-107。会使用 key：agent.uanme 来判断客户端是否存在，并且以 IP 地址作为唯一性的标识。

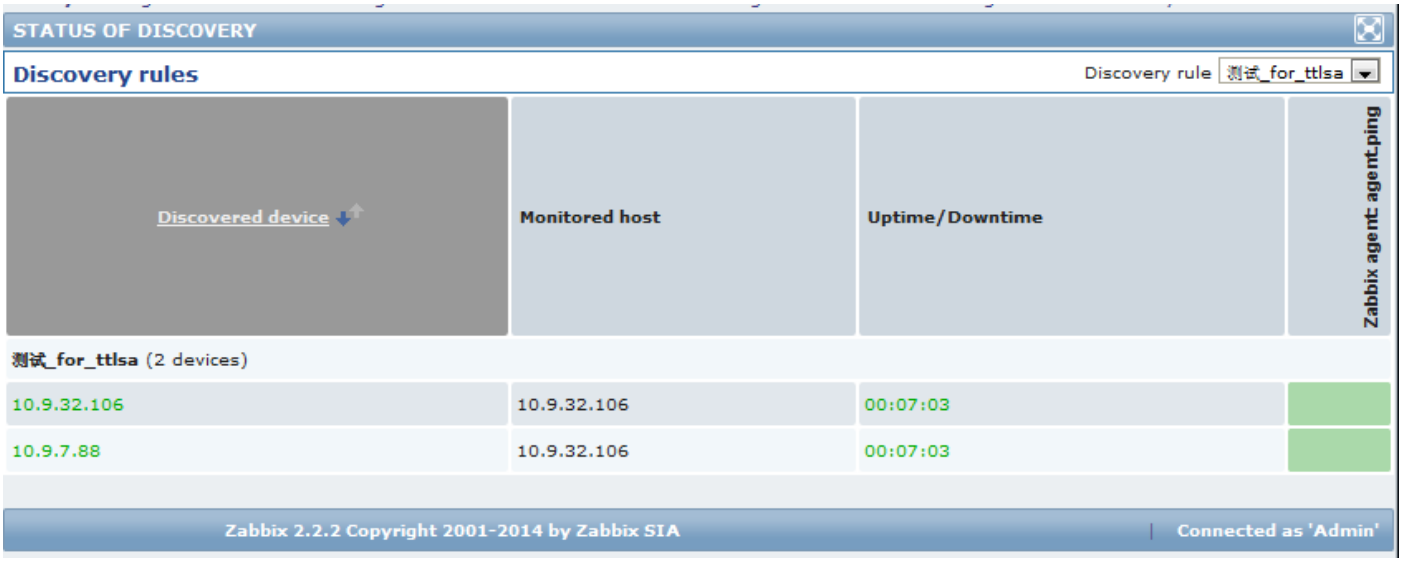
规则属性

属性	描述
Name	规则名称，唯一
Discovery by proxy	谁执行当前发现规则: no proxy - zabbix server <proxy name> - 指定的 proxy
IP range	发现规则中的 ip 范围，格式如下 单 IP: 192.168.1.33 一个 IP 段: 192.168.1.1-255 一个子网: 192.168.4.0/24 支持如下子网掩码: IPV4:/16 - /30

	IPV6:/112 - /128 IP 列表: 192.168.1.1-255,192.168.2.1-100,192.168.2.200,192.168.4.0/24 备注：1. IP 列表中的 IP 不能重复 2. 不同的发现规则里面不要包含相同的 IP，否则可能会出现意想不到的问题
Delay (in sec)	规则执行完毕之后，要多久才执行下一次。
Checks	支持的 checks: SSH, LDAP, SMTP, FTP, HTTP, HTTPS, POP, NNTP, IMAP, TCP, Telnet, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping. Port 属性如下： 单个端口: 22 端口段: 22-45 端口列表: 22-45,55,60-70
Device uniqueness criteria	设备唯一标识: IP address - 使用 IP 地址作为设备唯一性标识 Type of discovery check - 使用 SNMP 或者 Zabbix agent 的 check 作为唯一标识
Enabled	是否启用当前规则

第二步

Monitorning>>Discovery，可以看到已经发现了两台主机，ip 地址作为他们的唯一标识。确保这个标识的唯一性，否则 zabbix 会认为他们是一台主机。



第三步

目前仅仅是可以找到主机，并未自动添加到 Host 中，接下来完成几个步骤：

1. 加入到 Linux Servers 组

2.Linux link linux 模板、windows link windows 模板

3. 主机在线时长 10 分钟的主机添加到 HOST 中

4. 离线 1 天以上的主机从 Host 中移除

创建 Action

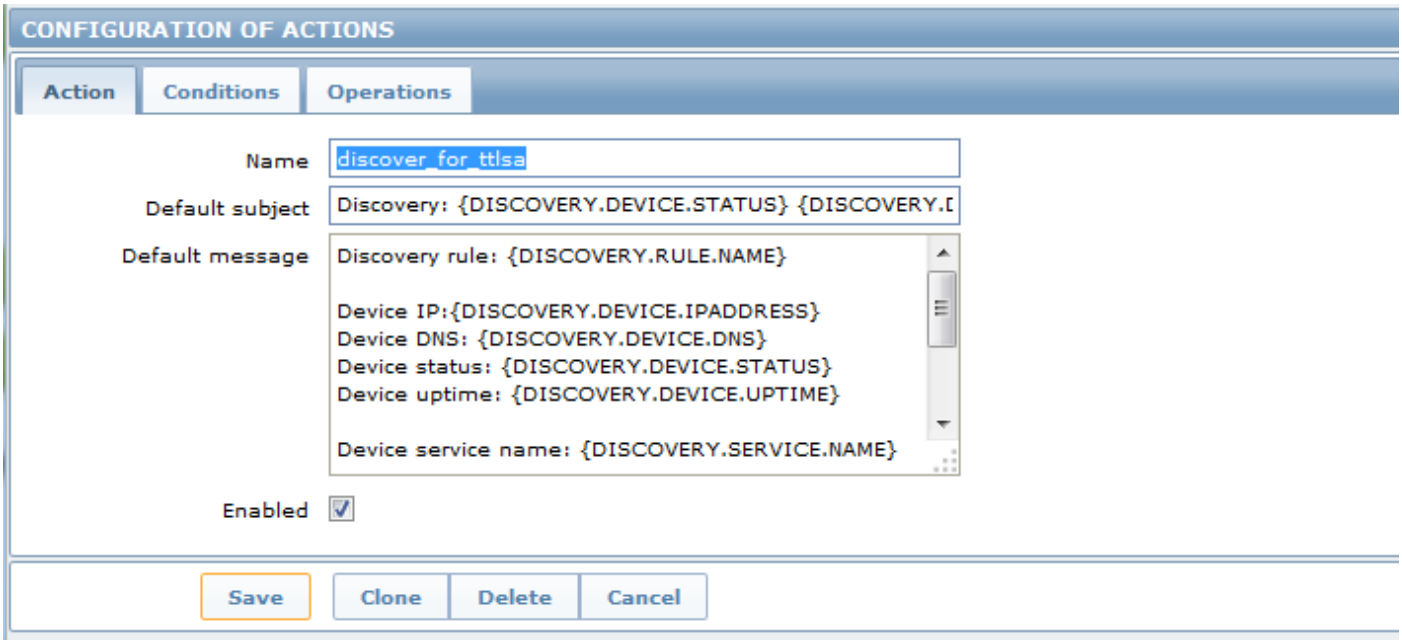
我们需要创建两个 Action，一份正对 windows，一份针对 Linux。我们下面演示一下 Linux 服务器

Action 添加主机

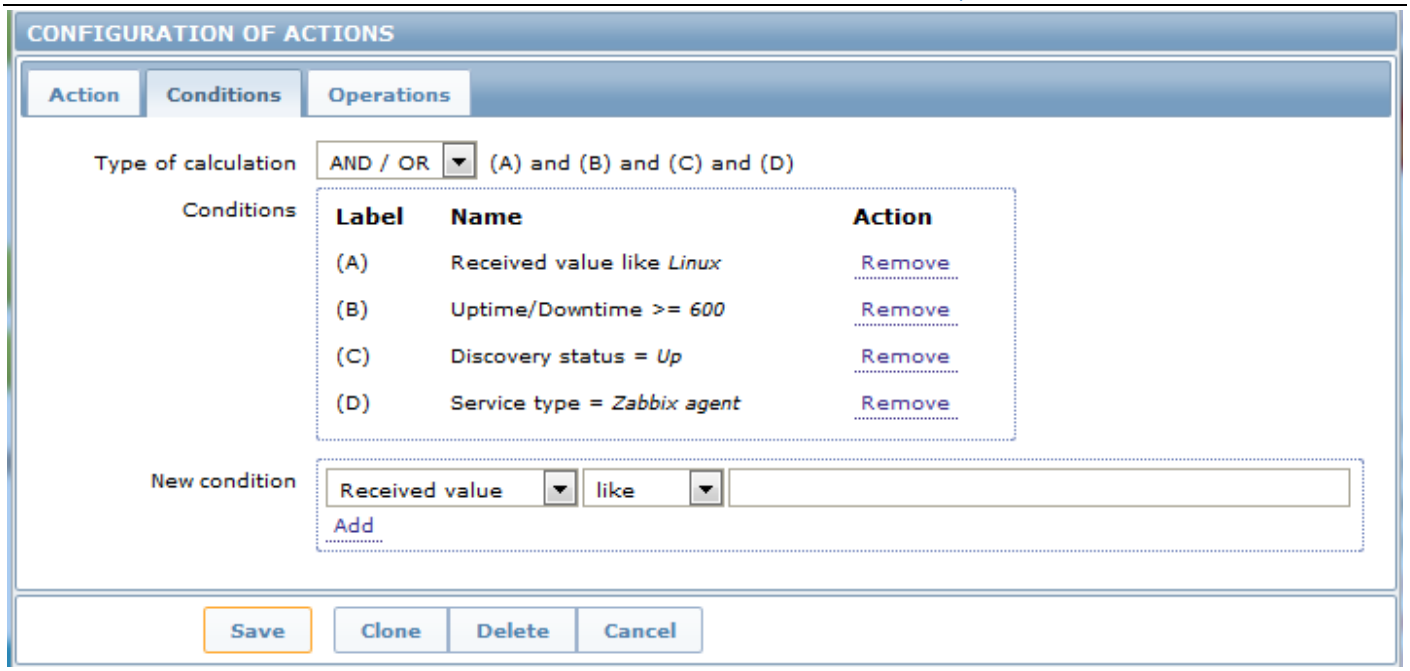
configuration>>action>>Event source（选 discover）>>create action

首先，配置名称，以及定义消息内容，这些使用默认的即可

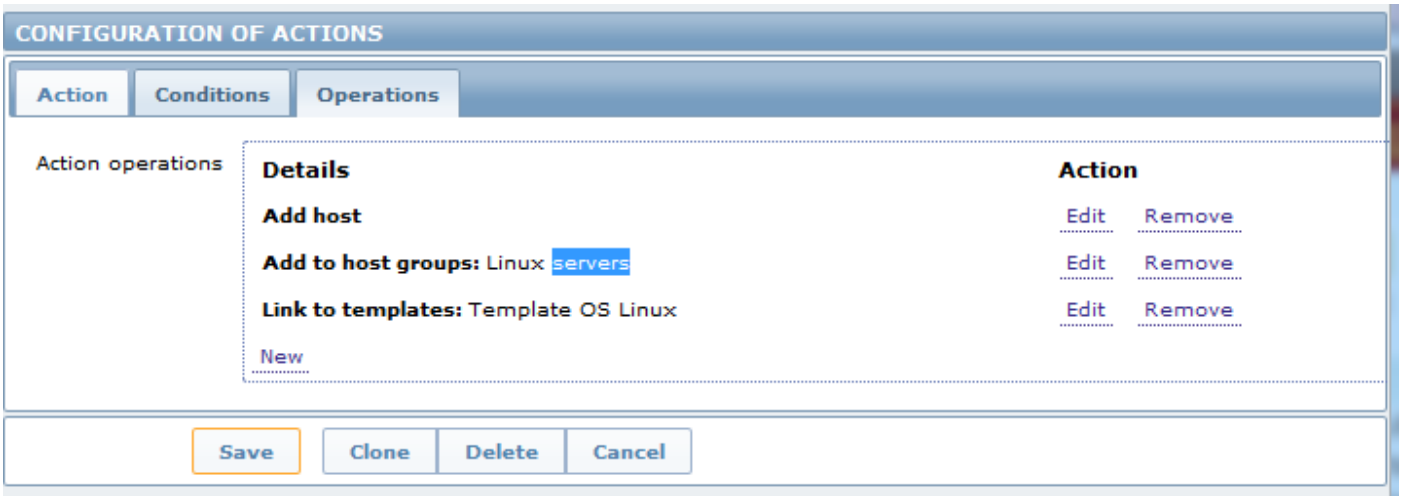
Action



条件配置



操作



回到 HOST 中，我们可以发现已经把主机加到列表里了，并且也 Linux 了模板以及加到了相应的组里

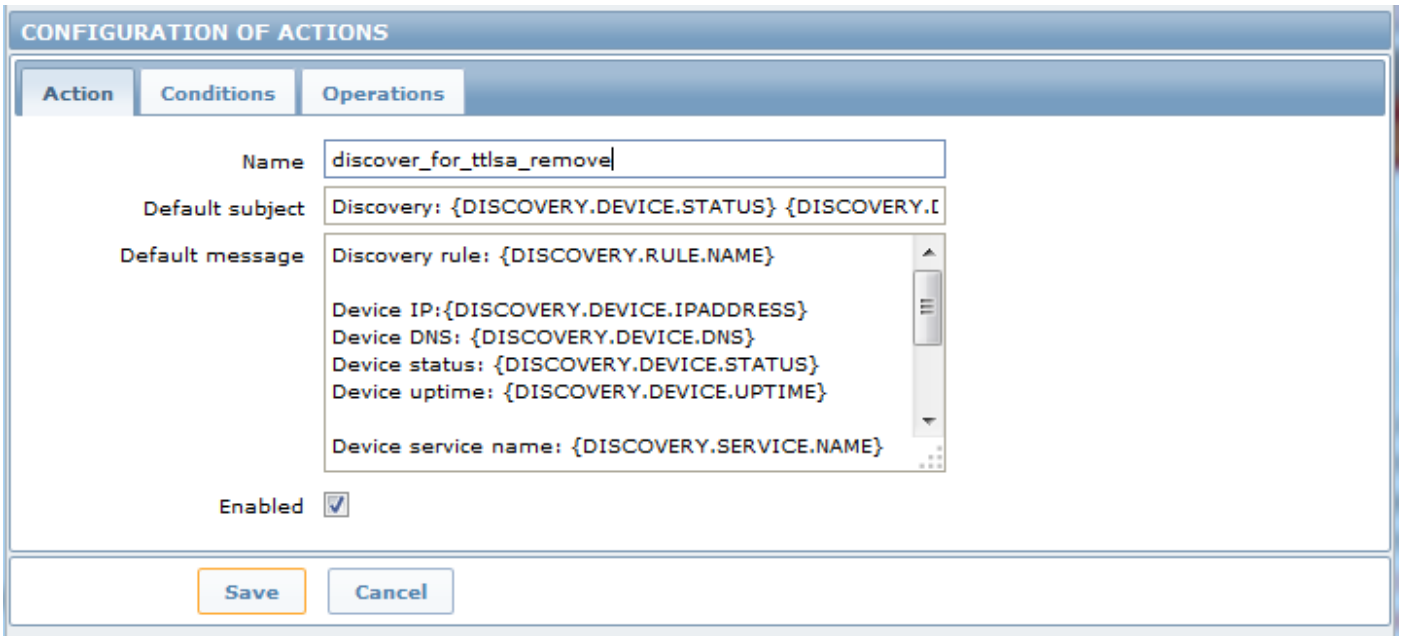
移除主机

configuration>>action>>Event source（选 discover）>>create action

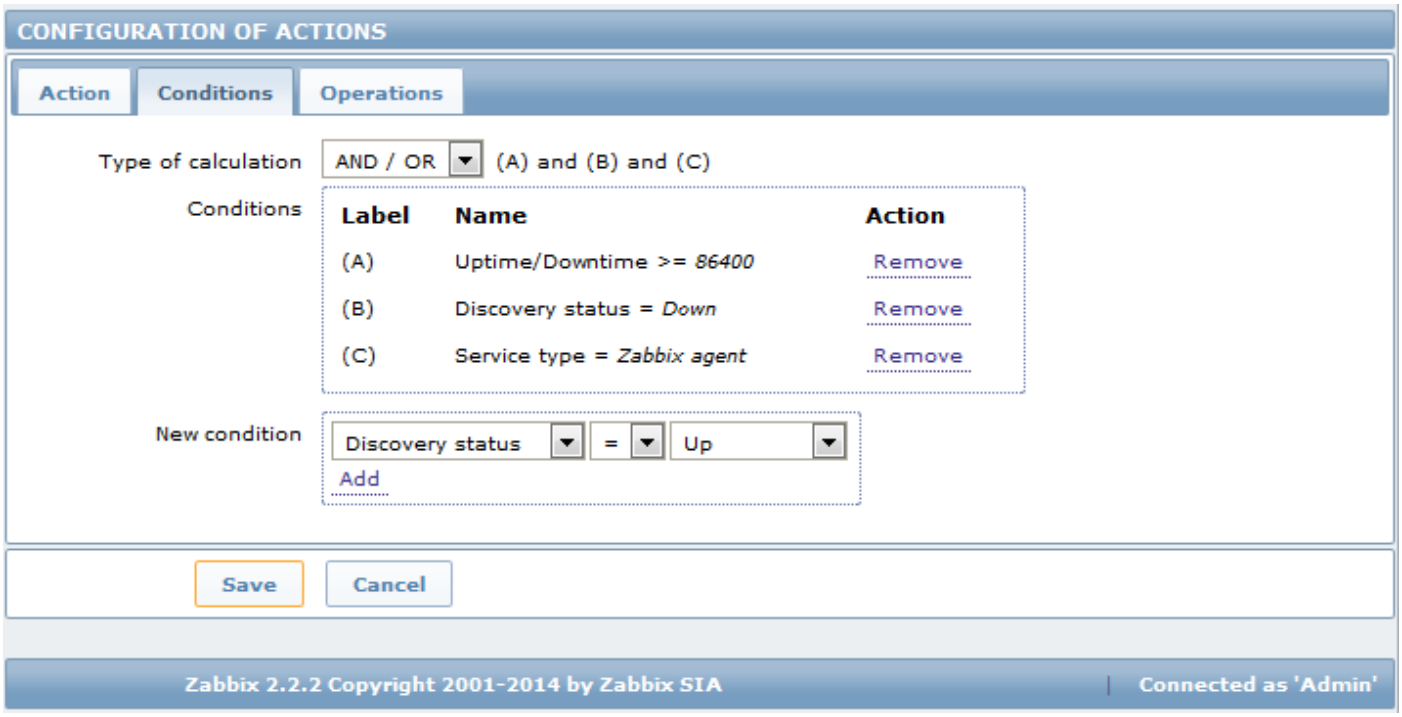
首先，配置名称，以及定义消息内容，这些使用默认的即可

移除主机

Action


CONFIGURATION OF ACTIONS
Action
Conditions
Operations
Name
discover_for_ttlsa_remove
Default subject
Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.[
Default message
Discovery rule: {DISCOVERY.RULE.NAME}
Device IP:{DISCOVERY.DEVICE.IPADDRESS}
Device DNS: {DISCOVERY.DEVICE.DNS}
Device status: {DISCOVERY.DEVICE.STATUS}
Device uptime: {DISCOVERY.DEVICE.UPTIME}
Device service name: {DISCOVERY.SERVICE.NAME}
Enabled
Save
Cancel

条件


CONFIGURATION OF ACTIONS
Action
Conditions
Operations
Type of calculation
AND / OR
(A) and (B) and (C)
Conditions
Label
Name
Action
(A)
Uptime/Downtime >= 86400
Remove
(B)
Discovery status = Down
Remove
(C)
Service type = Zabbix agent
Remove
New condition
Discovery status
=
Up
Add
Save
Cancel
Zabbix 2.2.2 Copyright 2001-2014 by Zabbix SIA
Connected as 'Admin'

动作

移除主机我就不演示了。

通过使用 discovery，zabbix 能够自动完成添加到 host 等等一系列动作，这一切都是基于这个规则来实现的。那么如果离开这个规则，我能完成这一系列动作吗？答案是肯定的

zabbix 客户端自动注册

1. 概述

上一篇内容《zabbix 自动发现配置》，大概内容是 zabbix server 去扫描一个网段，把在线的主机添加到 Host 列表中。我们本篇内容与上篇相反，这次是 Active agent 主动联系 zabbix server，最后由 zabbix server 将这些 agent 加到 host 里。对于需要部署特别多服务器的人来说，这功能相当给力。所有服务器批量装好 zabbix agent，server 配置好 trigger，所有的服务器都配置好了，非常快速。

2. 配置

2.1 配置文件修改

指定 server ip

```
# cat /usr/local/zabbix-2.2.2/etc/zabbix_agentd.conf | grep -E ^ServerActive
 ServerActive=66.175.222.232
```

修改 Hostname

```
# cat /usr/local/zabbix-2.2.1/etc/zabbix_agentd.conf | grep -E ^Hostname
Hostname=auto-reg-for-ttlsa-01
```

关于主机名：如果 zabbix_agentd.conf 配置有定义 Hostname，那么 zabbix 会使用这个 Hostname 命名，否则 agent 的主机名（hostname 得来的）

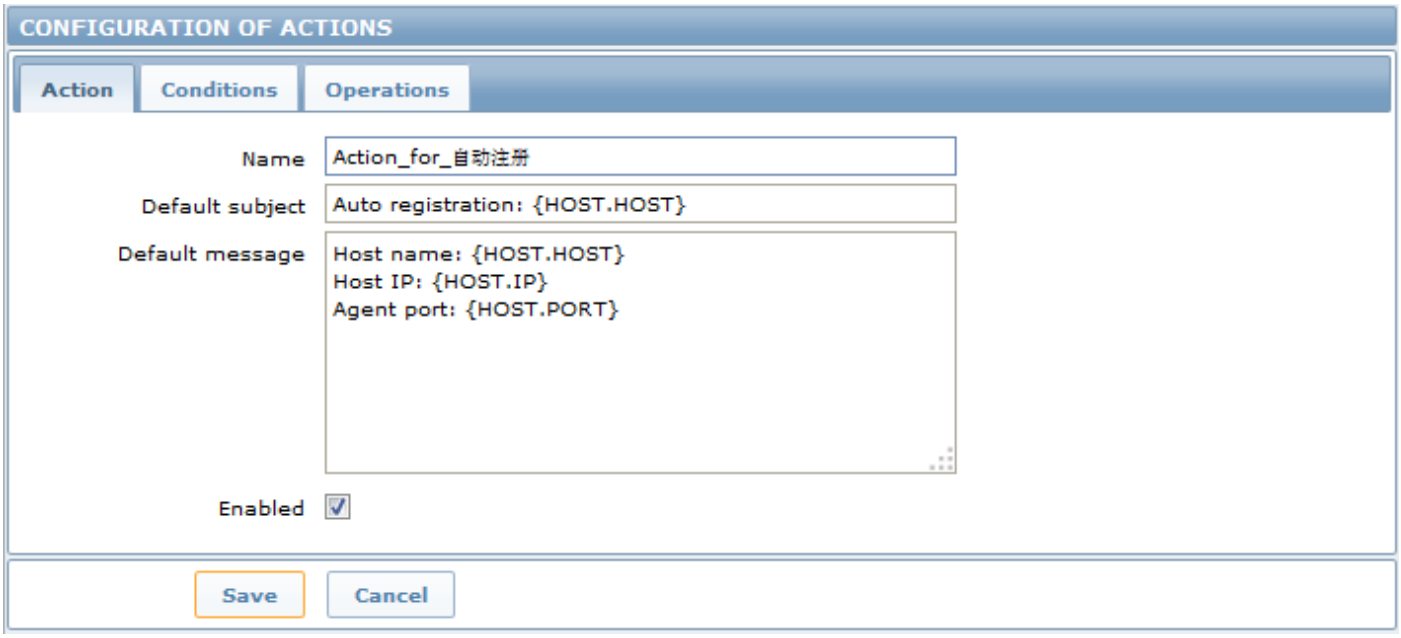
修改 metadataitem

```
cat /usr/local/zabbix-2.2.1/etc/zabbix_agentd.conf | grep HostMetadataItem=
 HostMetadataItem=system.uname
```

2.2 配置 action

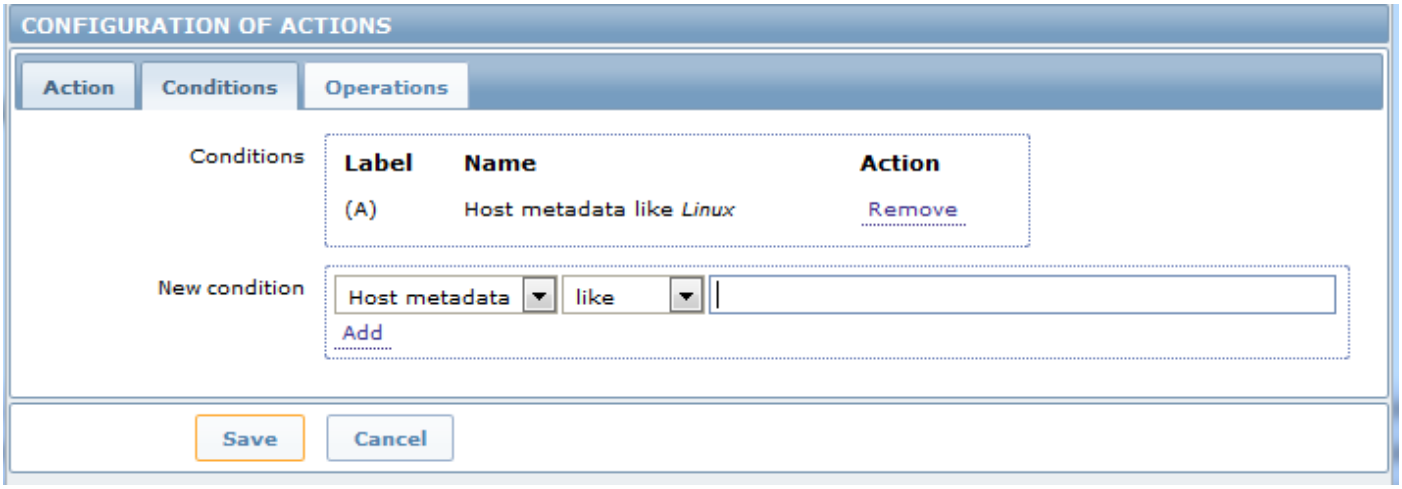
步骤：configuration>>action>>Event source（选择 Auto registration）>>Create Action，我们按如下步骤来定义个 action

2.2.1 action 选项卡



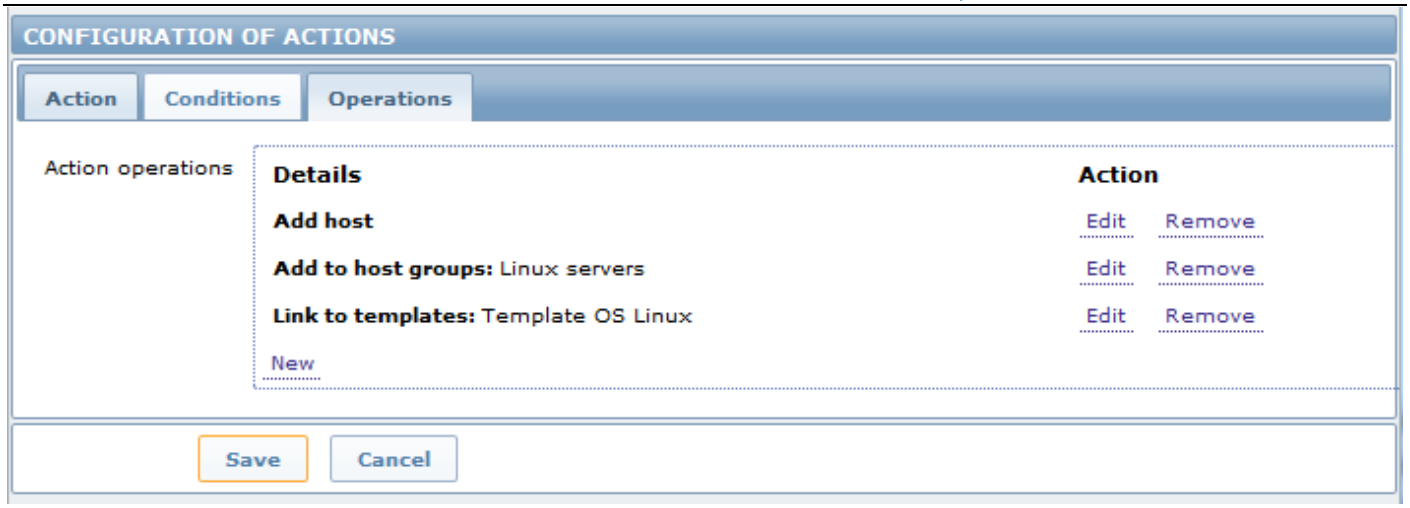
定义 Action 名称，以及发送消息的主题和内容，使用默认的就行了

2.2.2 Conditions 选项卡



Host metadata 包含 Linux 字符的主机将会触发 2.2.3 的操作，什么是 metadata，文章的下半段会专门讲解。

2.2.3 operations 选项卡



满足条件的 active host 发起请求，server 会完成三个动作：

把 agent 加到 host 列表

把 agent 加入 linux servers 组

agent 主机 link 模板 Template OS linux

3. 查看结果

查看/tmp/zabbix_server.log 我们能看到如下内容：

```
16585:20150203:161110.910 enabling Zabbix agent checks on host "auto-reg-for-ttlsa-01": host became available
```

看到如上内容，表明 host 增加成功，此时此刻的 host 列表如下：

4. HostMetadataItem 与 HostMetadata

作用：用于标示主机，通过该符号能够把主机区别开来。比如我们经常用它来区分 linux 与 windows 系统，这样才能分别给他们设置组与 template 等等

4.1 HostMetadataItem 用法

```
HostMetadataItem=system.uname
```

它的值来之 key

4.2 HostMetadata 用法

```
HostMetadata: Linux heheheheheheheheheh xxxxx
```

他的值是直接定义的

通过使用各式各样的 metadata 我们可以用于区分各个主机，来达到我们各种需求。

zabbix 低级别发现

概述

本篇文章是 zabbix 发现的最后一篇，回顾一下前面几篇文章

《zabbix 发现介绍》整个功能的介绍

《zabbix 发现配置》server 通过配置好的规则，自动添加 host、group、template

《zabbix Active agent 自动注册》与 discovery 相反，功能基本相同，active 联系 server，server 自动添加 host、group、template

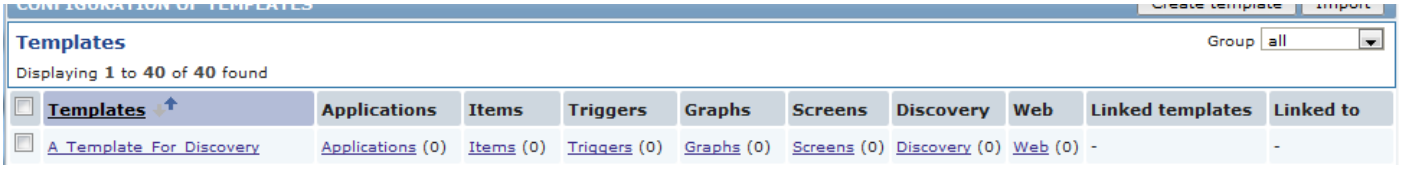
以上目的都是发现 host、添加 host，本文的 low-level discovery 更底层点，用于发现 item、trigger、graph 等等。我们最常用如：filesystem（如/、/home、/proc、C:、D:等），network（etho，eth1 等）

2. Discovery 之文件系统

众多服务器，难免系统以及分区会有所不同。一般存在 linux 和 windows 两种系统，linux 下分区有/、/data、/proc 等等，windows 有 C:D:E:等，A 服务器有/data 分区，B 服务器可能有/site 分区。他有什么分区，我便监控什么分区，这就是 low-level discovery 的功能。

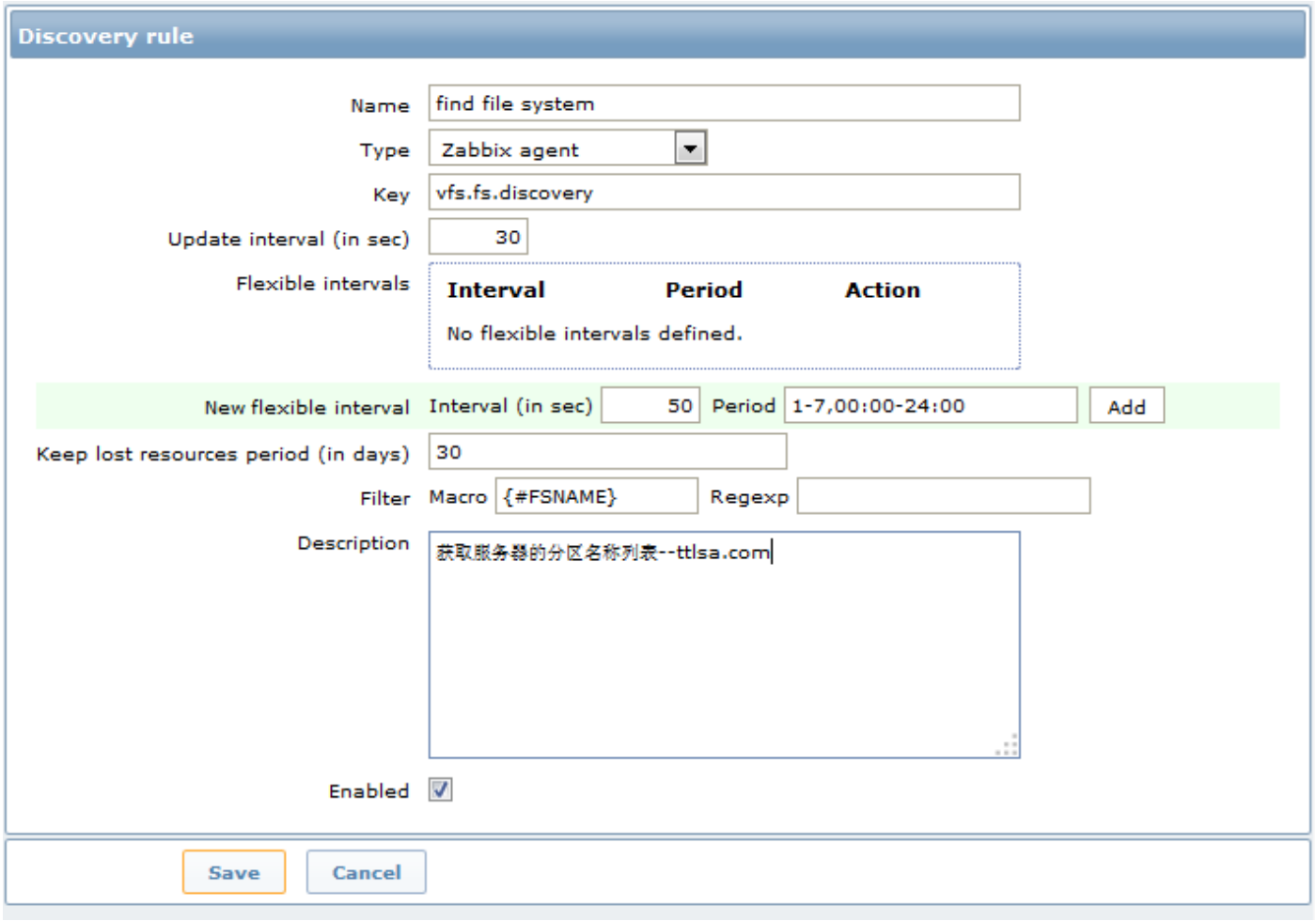
2.1 创建模板

创建模板 A_Template_For_Discovery,.....过程省略....



2.2 配置 discovery 规则

configuration>>templates>>找到模板“A_Template_For_Discovery”>>Discovery(0)>>Create discovery rule



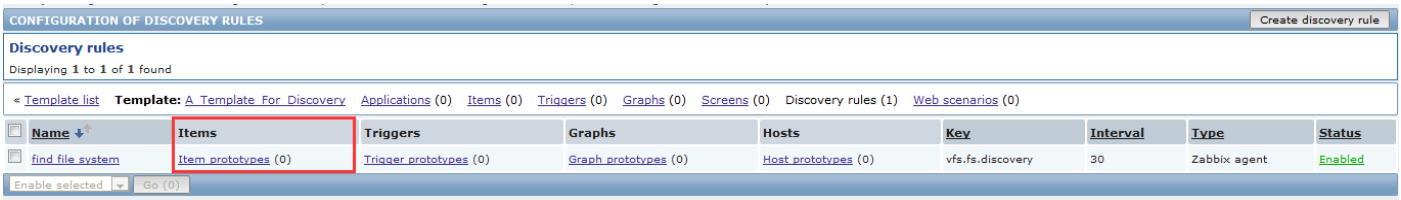
属性说明：

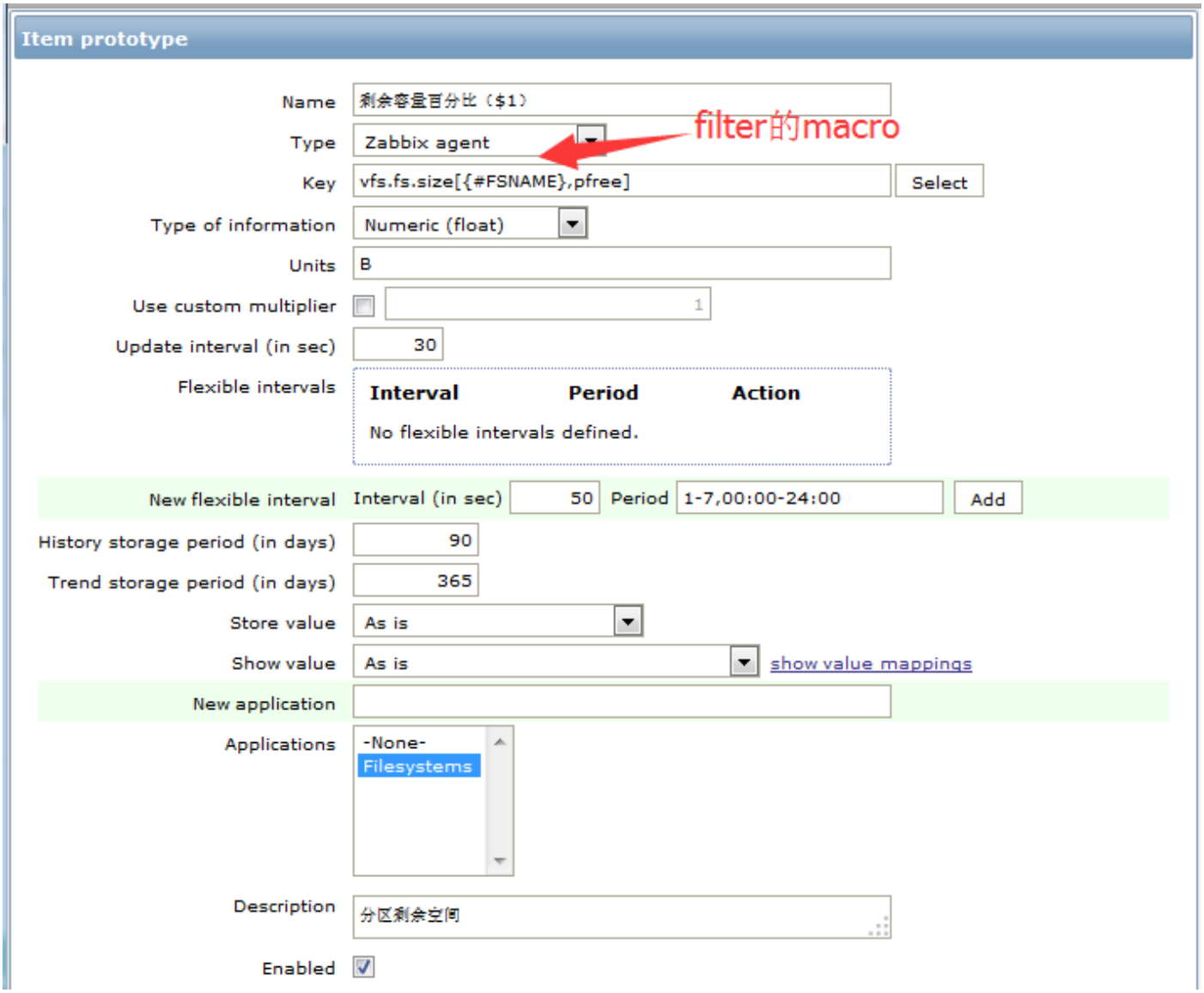
Keep lost resources period（in days）：数据保留天数，默认 30 天

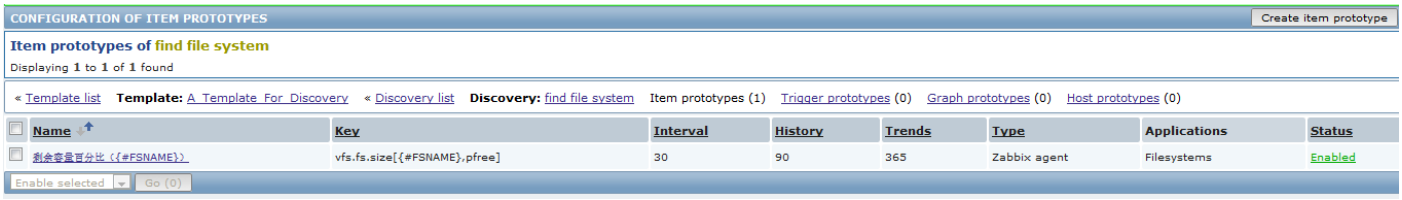
Fileter：Macro 为{#FSNAME}，key “vfs.fs.discovery”返回 json 数据列表，里面内容为{#FSNAME}作为 key，/、/data、C:等等作为 value。regext 可以使用表达式，例如"^/data|/C:"，如果想通过{#FSTYPE}来过滤，那么 Macro 写{#FSTYPE}，regexp 写^(ext2|ext3|swap)$,或者引入 zabbix 中定义好的的正则表达式，@表达式名称。

2.3 创建 Item prototypes

其实就是一个创建一个 item，configuration>>templates>>找到模板“A_Template_For_Discovery”>>Discovery(1)>>find file system>>Item prototypes (0)>>create Item prototypes



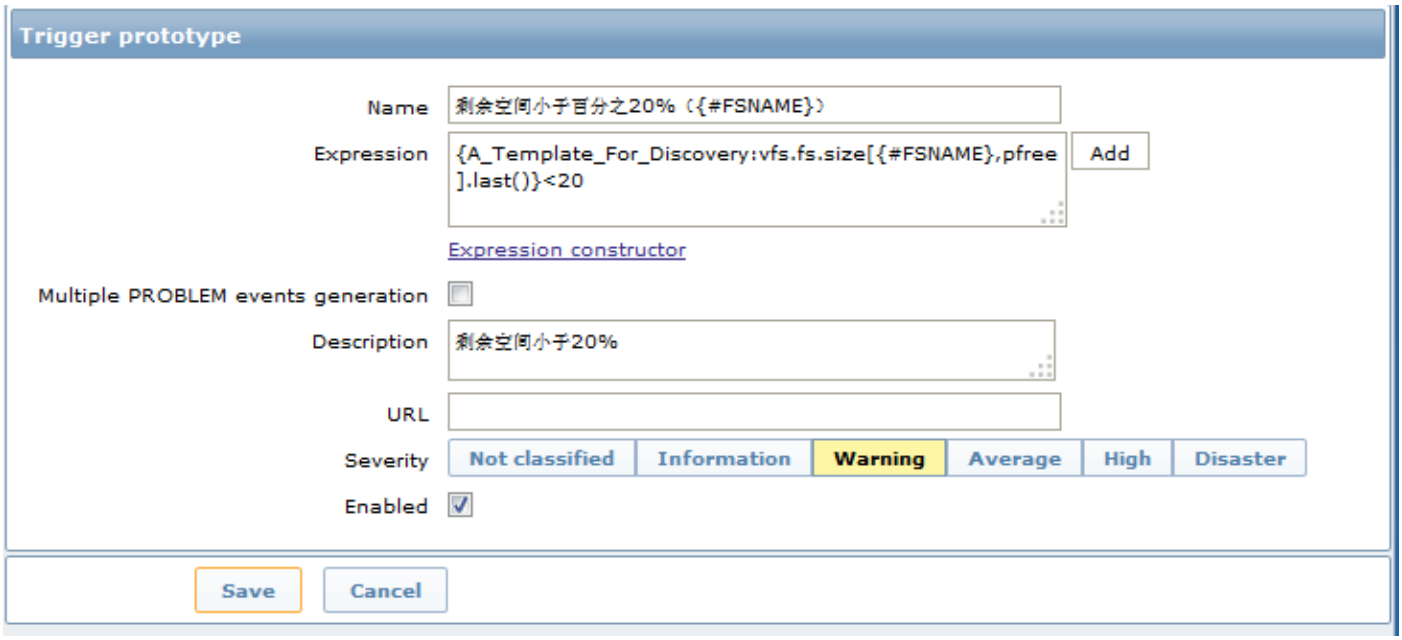




2.4 创建 Trigger

当剩余量小于 20%触发 warnning

configuration>>templates>> 找到模板“A_Template_For_Discovery”>>Discovery(1)>>find file system>>Trigger prototypes (0)>>Create trigger prototypes

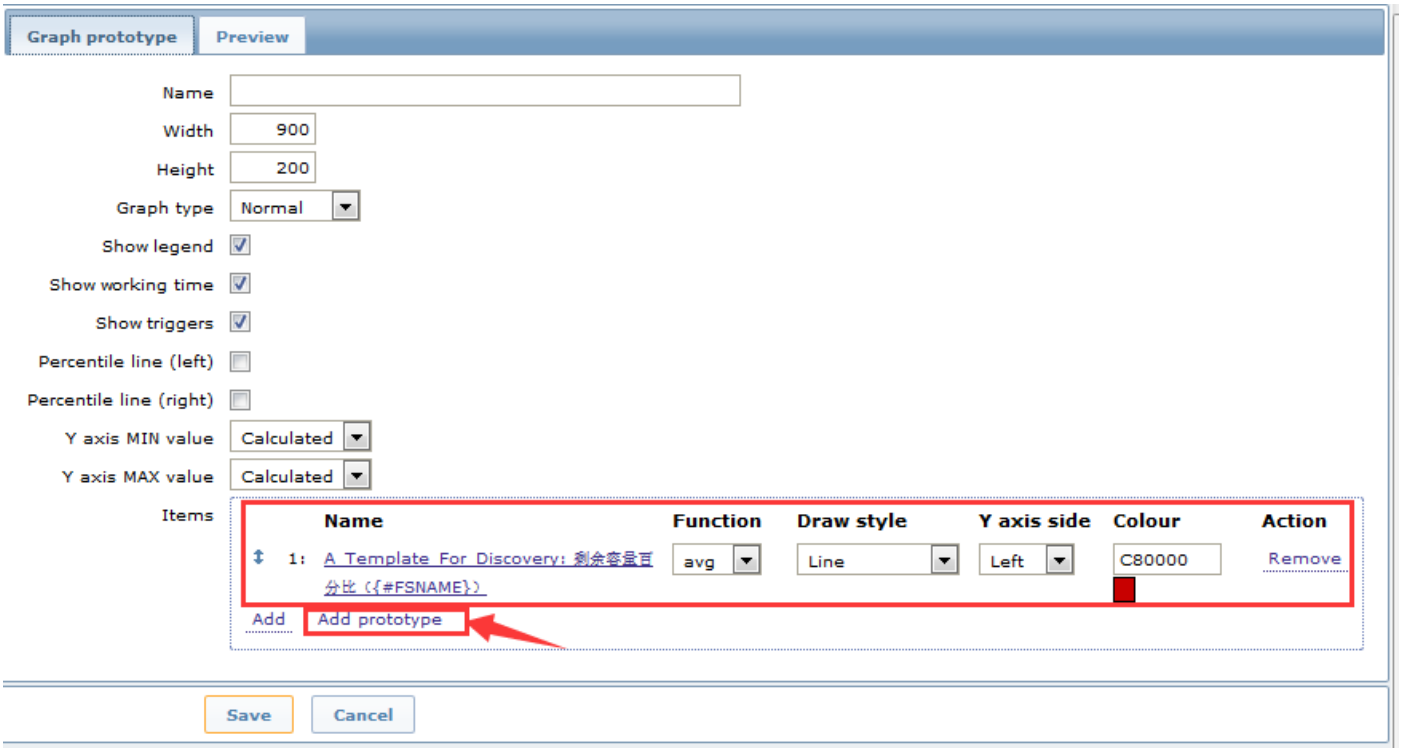


与普通的 trigger 区别在{#FSNAME}

2.4 创建 graph

绘制简单图表

configuration>>templates>> 找到模板“A_Template_For_Discovery”>>Discovery(1)>>find file system>>Graph prototypes (0)>>Create Graph prototypes



3. 自定义 LLD 规则

系统已经内建了文件系统的{#FSNAME}，网络的{#IFNAME}，因为业务的特殊性，我们如何定义我们自己需要的名称呢？

- 编写脚本,脚本输出 json 数据，包含 key 和 value
- 脚本加入 zabbix_agentd.conf UserParameter
- 重启 zabbix_agentd
- 使用定义好的名称配置 low-level discovery

3.1 脚本范例

```
#!/usr/bin/perl

$first = 1;
print "{\n";
print "\t\"data\":[\n\n";

for (`cat /proc/mounts`)
{
        ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;
        $fsname =~ s!/!\\/!g;

        print "\t,\n" if not $first;
        $first = 0;

        print "\t{\n";
        print "\t\t\"{#FSNAME}\":\"$fsname\",\n";
        print "\t\t\"{#FSTYPE}\":\"$fstype\"\n";
        print "\t}\n";
}
```

```
print "\n\t]\n";
print "}\n";
```

3.2 结果范例

执行后得到如下数据，是 json 格式

```
{
        "data":[

        {
                "{#FSNAME}":"\/",
                "{#FSTYPE}":"rootfs"
        }
        ,
        {
                "{#FSNAME}":"\/proc",
                "{#FSTYPE}":"proc"
        }
        ,
        {
                "{#FSNAME}":"\/sys",
                "{#FSTYPE}":"sysfs"
        }
        ,
        {
                "{#FSNAME}":"\/dev",
                "{#FSTYPE}":"devtmpfs"
        }
        ,
        {
```

```
                "{#FSNAME}":"\/dev\/pts",
                "{#FSTYPE}":"devpts"
        }
        ,
        {
                "{#FSNAME}":"\/dev\/shm",
                "{#FSTYPE}":"tmpfs"
        }
        ,
        {
                "{#FSNAME}":"\/",
                "{#FSTYPE}":"ext4"
        }
        ,
        {
                "{#FSNAME}":"\/proc\/bus\/usb",
                "{#FSTYPE}":"usbfs"
        }
        ,
        {
                "{#FSNAME}":"\/proc\/xen",
                "{#FSTYPE}":"xenfs"
        }
        ,
        {
                "{#FSNAME}":"\/proc\/sys\/fs\/binfmt_misc",
                "{#FSTYPE}":"binfmt_misc"
        }
        ]
}
```

第十三章：API

Zabbix API 二次开发

有了 zabbix API 我们可以做很多，自己开发 web 界面、开发手机端 zabbix、获取 zabbix 指定数据、创建 zabbix 监控项等等。

zabbix API 开发库

zabbix API 请求和响应都是 json，并且还提供了各种语法的 lib 库，http://zabbix.org/wiki/Docs/api/libraries，包含 php、c#、Python、Perl、go 等等语言，简单看了下 phpzabbixapi，使用非常方便。

请求 zabbix API

post json 数据到 api 接口地址，例如你得 zabbix 地址是 http://company.com/zabbix，那么你得接口地址是：http://company.com/zabbix/api_jsonrpc.php，必须包含 content-type 头，值为 application/json-rpc, application/json or application/jsonrequest 之一。

```
POST http://company.com/zabbix/api_jsonrpc.php HTTP/1.1
Content-Type: application/json-rpc
{"jsonrpc":"2.0","method":"apiinfo.version","id":1,"auth":null,"params":{}}
```

zabbix API 登陆

获取 auth token（登陆）

在操作 zabbix 之前，我们必须先登陆 zabbix，得到 token，以后的操作带着 token 即可，要不然肯定没权限。

请求的 json 如下：

```
{
 "jsonrpc": "2.0",
 "method": "user.login",
 "params": {
 "user": "Admin",
```

```
"password": "zabbix"
},
"id": 1,
"auth": null
}
```

属性说明

属性	描述
jsonrps	JSON-RPC 版本，基本上用 2.0 就行了;
method	调用的 API 方法，方法列表请上官网;
params	需要传递的参数，这边是 user 和 password;
id	请求标志;
auth	用户 token，这边使用 null，因为还没通过验证

验证成功，会返回如下 json 数据

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

result 便是我们要德 token 数据，id 对应请求的 id。

zabbix api 检索主机

通过验证之后，我们带着 token 使用 host.get 获取主机列表，请求的 json 如下：

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
```

```
                "host"
            ],
            "selectInterfaces": [
                "interfaceid",
                "ip"
            ]
        },
        "id": 2,
        "auth": "0424bd59b807674191e7d77572075f33"
}
```

获取到如下数据

```
{
        "jsonrpc": "2.0",
        "result": [
            {
                "hostid": "10084",
                "host": "Zabbix server",
                "interfaces": [
                    {
                        "interfaceid": "1",
                        "ip": "127.0.0.1"
                    }
                ]
            }
        ],
        "id": 2
}
```

请使用你的程序处理一下即可。

zabbix API 就是这么简单，请求、响应然后处理，更多 API 方法请直接上官方文档，里面有几百个方法等着你。如

果你使用 zabbix 二次开发，千万不要直接操作 zabbix 数据，太…..，为何不使用 zabbix API。

第十四章：zabbix 命令

Zabbix 命令：zabbix_server

介绍

zabbix 可以没有 zabbix_agentd，也可以没用 snmp、也可以没有 proxy，但是 zabbix_server 那是绝对不能少，它是 zabbix 最核心的东西。获取数据、配置主机、发送邮件等等众多事情都是由 zabbix_server 完成，zabbix_server 绝对是日理万机废寝忘食的好员工。

兼容系统

虽然和 zabbix_agentd 相比，zabbix_server 支持的系统少一点，但是它还是支持着众多操作系统。最遗憾的是它不支持 windows 系统。windows sa 们可以哭一会儿。支持的系统如下：

- Linux
- Solaris
- AIX
- HP-UX
- Mac OS X
- FreeBSD
- OpenBSD
- NetBSD
- SCO Open Server
- Tru64/OSF1

命令详解

使用语法

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

命令选项

-c --config <file> 配置文件路径

-n --new-nodeid <nodeid> 将数据转为 nodeid，如果当前服务器想作为一个 nodeid 需要使用名，切记，操作不可逆

-R --runtime-control <option> 执行管理功能，如：config_cache_reload

Runtime 控制项:

config_cache_reload reload 配置缓存

zabbix_server 命令实例

启动 zabbix server

```
/usr/local/zabbix-2.2.1/sbin/zabbix_server -c  /usr/local/zabbix-2.2.1/etc/zabbix_server.conf
```

转为 node 模式（只做转换，不会启动 server）

```
/usr/local/zabbix-2.2.1/sbin/zabbix_server -c  /usr/local/zabbix-2.2.1/etc/zabbix_server.conf -n 12
```

reload 配置缓存

```
/usr/local/zabbix-2.2.1/sbin/zabbix_server -c  /usr/local/zabbix-2.2.1/etc/zabbix_server.conf -R config_cache_reload
```

Zabbix 命令：zabbix_get

zabbix_get 作用

总有人在群里提问，为什么 zabbix 获取不到数据，为什么 zabbix 提示 Not Support，怎么办？别老问，用 zabbix_get 试着获取数据即可。在 zabbix server 上执行 zabbix_get 命令来试着获取 item 值

zabbix_get 命令详解

命令在 zabbix 安装目录 bin 下，如果是 window 命令自然是 zabbix_get.exe，使用方法都是一样的。端口和源（绑定）IP 是可选的，其他参数不能漏掉

```
# /usr/local/zabbix-2.2.1/bin/zabbix_get -h
  Zabbix get v2.2.1 (revision 40808) (09 December 2013)
usage: zabbix_get [-hV] -s <host name or IP> [-p <port>] [-I <IP address>] -k <key>
Options:
  -s --host <host name or IP>                Specify host name or IP address of a host
  -p --port <port number>                    Specify port number of agent running on the host. Default is 10050
  -I --source-address <IP address>           Specify source IP address
-k --key <key of metric>                     Specify key of item to retrieve value for
-h --help                                    Give this help
  -V --version                               Display version number
Example: zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]"
```

参数说明：

-s – host： 指定客户端主机名或者 IP

-p – port：客户端端口，默认 10050

-I – source-address：指定源 IP，写上 zabbix server 的 ip 地址即可，一般留空，服务器如果有多 ip 的时候，你指定一个。

-k – key：你想获取的 key

至于使用长参数还是短的，自己选，我经常使用-s 而不是-host，太长了。来个例子咯

zabbix_get 获取数据

获取负载

```
./zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg15]"

0.270000
```

获取主机名

```
# ./zabbix_get -s 127.0.0.1 -p 10050 -I 127.0.0.1 -k "system.hostname"

 10-9-4-20
```

Zabbix 命令：zabbix_agentd

zabbix_agentd 用途

zabbix_agentd 非常非常重要，它是每个学习 zabbix 必须熟悉也必须接触到得东西，把 zabbix_agentd 安装到被监控服务器上，zabbix_server 便可以和 zabbix_agentd 通信来获取数据。下面来详细聊聊 zabbix_agentd。

兼容系统

作为一个跨平台的监控系统，zabbix_agentd 可以被安装到各式各样的系统中，如下：

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris: 9, 10, 11
- Windows: 2000, Server 2003, XP, Vista, Server 2008, 7

命令详解（Linux）

linux 和 windows 上略有不同，不过基本上差不多

使用方法

```
usage: zabbix_agentd [-Vhp] [-c <config-file>] [-t <item key>]
```

参数

```
-c --config <config-file>   配置文件绝对路径
-p --print                  打印出所有的 item，然后退出
-t --test <item key>        测试指定 item key，然后退出
-h --help                   帮助
-V --version                显示版本号
```

命令详解（Windows）

基本上使用方法和 linux 一样，不过有些特殊，在 window 里，zabbix_agentd 多了服务这块。

```
-i --install              安装为服务
-d --uninstall            移除服务
-s --start                启动服务 zabbix_agentd
-x --stop                 关闭 zabbix_agentd
-m --multiple-agents      多个服务，服务名包含 agentd 的主机名（不常用）
```

实例测试

列出 items

```
# ./zabbix_agentd -p | grep uptime
system.uptime                                 [u|15923945]
```

测试 key

```
# ./zabbix_agentd -t system.uptime -c /usr/local/zabbix-2.2.1/etc/zabbix_agentd.conf
system.uptime                                 [u|15923992]
```

启动 zabbix_agentd

```
# ./zabbix_agentd    -c /usr/local/zabbix-2.2.1/etc/zabbix_agentd.conf
```

Zabbix 命令：zabbix_sender

zabbix_sender 用途

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

执行超长时间脚本，如：脚本去几十台服务器拉去数据，每个日志都上 G，然后日志整合在一起，统计出返回值。这种脚本比如超时，所以我们必须改成让客户端提交数据的方式。

命令详解

语法

```
usage: zabbix_sender [-Vhv] {[-zpsI] -ko | [-zpI] -T -i <file> -r} [-c <file>]
```

使用参数

```
  -c --config <file>                    配置文件绝对路径
  -z --zabbix-server <server>           zabbix server 的 IP 地址
  -p --port <server port>               zabbix server 端口.默认 10051
  -s --host <hostname>                  主机名，zabbix 里面配置的主机名（不是服务器的 hostname），不能使用 ip 地址
  -I --source-address <IP address>      源 IP
  -k --key <key>                        监控项的 key
  -o --value <key value>                key 值
  -i --input-file <input file>          从文件里面读取 hostname、key、value 一行为一条数据，使用空格作为分隔符，如果主机名带空格，那么请使用双引号包起来
  -T --with-timestamps                  一行一条数据，空格作为分隔符: <hostname> <key> <timestamp> <value>，配合 --input-file option，timestamp 为 unix 时间戳
  -r --real-time                        将数据实时提交给服务器
```

-v --verbose	详细模式, -vv 更详细

使用实例

```
# ./zabbix_sender -s 127.0.0.1 -z 127.0.0.1 -k "ttlsa.trapper" -o 1 -r
info from server: "processed: 0; failed: 1; total: 1; seconds spent: 0.000024"
sent: 1; skipped: 0; total: 1
```

ttlsa.trapper：是我们定义好的 key

-o 1：1 是 key 值

failed：错误数，说了-s 不能用 ip 地址，那么我们改成配置文件中得主机名吧

```
# ./zabbix_sender -s "Zabbix server" -z 127.0.0.1 -k "ttlsa.trapper" -o 1 -r
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000035"
sent: 1; skipped: 0; total: 1
```

zabbix_sender 批量传递 key 值

```
#cat f.txt
"Zabbix server" ttlsa.trapper 10
"Zabbix server" ttlsa.trapper 20
"Zabbix server" ttlsa.trapper 30
"Zabbix server" ttlsa.trapper 40
"Zabbix server" ttlsa.trapper 1
# ./zabbix_sender -z 127.0.0.1 -i f.txt
info from server: "processed: 5; failed: 0; total: 5; seconds spent: 0.000085"
sent: 5; skipped: 0; total: 5
```

每行对应一个 key 值，一般是不同的主机名、不同的 key、不同的 key 值。这边方便测试，所以都用了同一个 key

Zabbix 命令：zabbix_proxy

zabbix_proxy 介绍请看《zabbix proxy 分布式配置》 ，zabbix_proxy 收集被监控端的数据，先缓存到本地然后把数据传给 zabbix server，因为 zabbix_proxy 基本上是在收集数据，而不像 server 一样要处理数据，所以 zabbix_proxy 对硬件的要求相对不高，今天主要是来了解 zabbix_proxy 命令的，往下看吧。

兼容系统

和 zabbix server 基本一样，server 支持什么，proxy 就支持什么。请参考《zabbix_server 命令详解》

命令详解

使用语法

```
zabbix_proxy [-hV] [-c <file>] [-R <option>]
```

使用参数

```
  -c --config <file>                 配置文件路径
  -R --runtime-control <option>      执行管理功能

Runtime control options:
  config_cache_reload                Reload 配置缓存
```

启动 proxy

```
# /usr/local/zabbix-2.2.1/sbin/zabbix_proxy -c /usr/local/zabbix-2.2.1/etc/zabbix_proxy.conf
```

Rreload proxy 配置缓存

```
# /usr/local/zabbix-2.2.1/sbin/zabbix_proxy -c /usr/local/zabbix-2.2.1/etc/zabbix_proxy.conf    -R config_cache_reload
```

第十五章：分布式监控

zabbix 分布式监控 proxy vs nodes

概述

zabbix 为 IT 基础设施提供有效和可用的分布式监控,zabbix 提供了两种解决方案：proxy 和 nodes.proxy 代替 zabbix server 在本地检索数据，然后提交给 zabbix server. Nodes 则就是一个完整的 zabbix Server.

Proxy vs. node

服务器一多以及服务器分布在各个不同地区，便需要考虑使用分布式监控，那么我们到底选择 proxy 还是 nodes 呢,请看如下的对照表，看完之后，我想你能选到一个你满意的方式.

功能	Proxy	Node	描述
轻量级	Yes	No	安装完毕即可,Proxy 必须更轻量级
GUI/图形界面	No	Yes	proxy 的配置都在 servers 上，而 node 是一个完整的 server
独立工作	Yes	Yes	
易于维护	Yes	No	
自动生成数据库	Yes	No	
本地管理	No	Yes	
Ready for embedded hardware	Yes	No	
One way TCP connections	Yes	Yes	
集中配置	Yes	No	proxy 配置全部集中在 server 上，node 自己维护自己的配置
通知	No	Yes	

备注：只有 SQLite 才支持自动创建数据库，其他数据都需要手动创建.

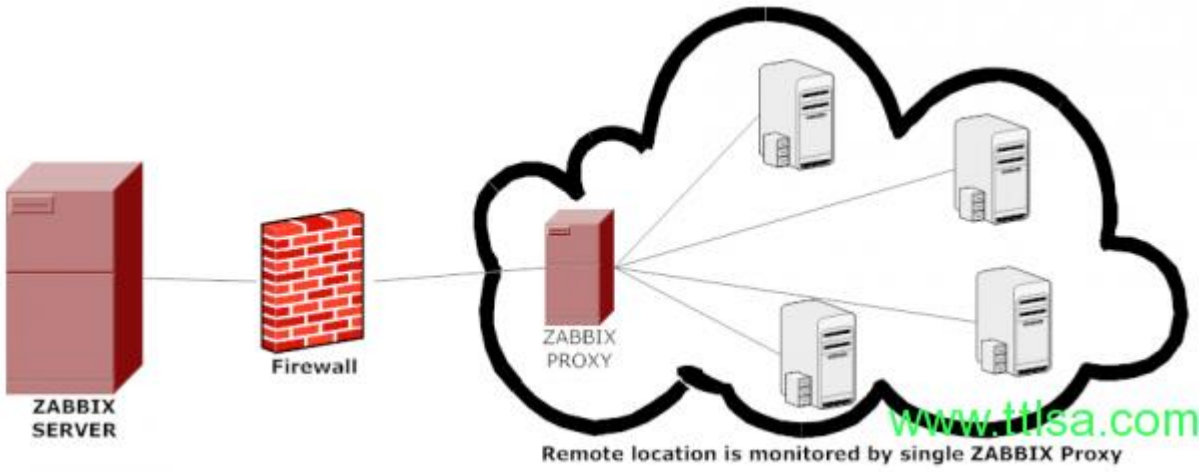
zabbix proxy 分布式监控配置

概述

zabbix proxy 可以代替 zabbix server 检索客户端的数据，然后把数据汇报给 zabbix server，并且在一定程度上分担了 zabbix server 的压力.zabbix proxy 可以非常简便的实现了集中式、分布式监控.

zabbix proxy 使用场景:

- 监控远程区域设备
- 监控本地网络不稳定区域
- 当 zabbix 监控上千设备时，使用它来减轻 server 的压力
- 简化 zabbix 的维护



- zabbix proxy 仅仅需要一条 tcp 连接到 zabbix server,所以防火墙上仅仅需要加上一条规则即可.zabbix proxy 数据库必须和 server 分开，否则数据会被破坏，毕竟这两个数据库的表大部分都相同。总之记住，数据库分开即可。
- proxy 收集到数据之后，首先将数据缓存在本地，然后在一定得时间之后传递给 zabbix server.这个时间由 proxy 配置文件中参数 ProxyLocalBuffer and ProxyOfflineBuffer 决定.
- zabbix proxy 是一个数据收集器，它不计算触发器、不处理事件、不发送报警，如下是 proxy 的功能.

Items	proxy support(yes/no)
Zabbix agent checks	Yes
Zabbix agent checks (active)	Yes
Simple checks	Yes

Trapper items	Yes
SNMP checks	Yes
SNMP traps	Yes
IPMI checks	Yes
JMX checks	Yes
Log file monitoring	Yes
Internal checks	Yes
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

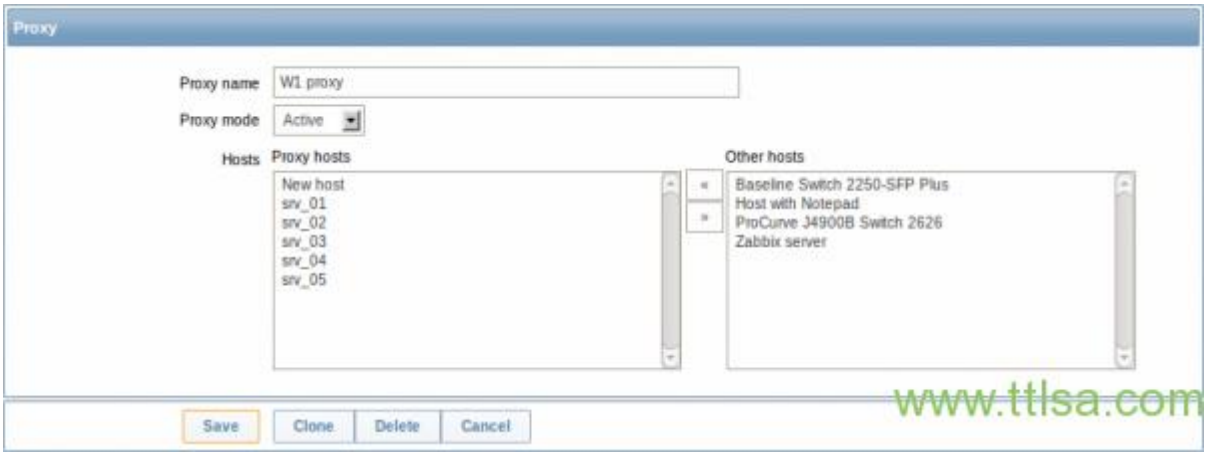
备注：使用 agent active 模式，一定要记住在 agent 的配置文件参数 ServerActive 加上 proxy 的 IP 地址.切记

配置

如果你安装好 proxy（安装方法我们后续讲）之后，我们便可以在 zabbix 管理站点上配置 proxy 了.

添加 proxy

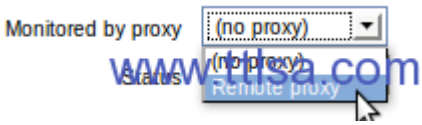
ministration（管理） → DM（分布式监控） - >Create proxy（创建代理）



参数	描述
Proxy name	proxy 名称，必须和 proxy 配置文件中的 hostname 一致
Proxy mode	选择 proxy 模式
Active	proxy 主动连接到 zabbix server 并且请求配置文件数据
Passive	Zabbix server 连接到 proxy
Hosts	哪些主机需要被 proxy 监控

Host 配置

配置主机 HOST 的时候，如果需要被 proxy 代理，那么都选择对应的 proxy 名称



第十六章：性能优化

zabbix 性能优化中的几个中肯建议

随着 zabbix 的广泛应用，少数人的 zabbix 服务器在性能上出现瓶颈，或者在未来会出现性能方面的瓶颈，接下来讨论几个有效并且简单的优化方案。

- 服务器硬件

  想通过几个简单的配置让服务器提高成倍的性能，想法很好，但是基本不太现实。简单的说，你需要搭配更好的 CPU、更大的内存，更快的硬盘：条件允许的花，可以考虑购买 SSD，它比更大的 cpu 和更大的内存带来的效果更好，或者考虑使用 SAS 15K 硬盘，组 raid 等等，总之一句话，配置优化不动的情况，增加硬件投入，别绞尽脑汁搜索：zabbix 如何优化之类的文章，你在浪费时间。

- 操作系统

  使用最新的操作系统，优化、定制化操作系统内核。应该会有些作用，但是肯定不大。

- 数据库优化

  DBsock 优化

  如果 MySQL 和 zabbix server 在同一台服务器上，socket 连接要比 tcp 连接要更快。

- 数据库分离

  将数据库服务器独立，数据库和 zabbix 资源互相独立，例如：可以购买一台 RDS

- 数据库引擎

  使用 MySQL5.6 或者更高版本，自从 MySQL 被 Oracle 收购了，它的性能确实有不少的提升。请一定选择 innodb，别选择 myisam，因为 zabbix 在 innodb 的性能比在 myisam 快 1.5 倍，而且 myisam 不安全，zabbix 监控数据量很大，一旦表坏了，那就是一个悲剧。

  mysql 分区，history 等等表数据量较大，可以试着分区替身性能。

- 其他优化

  a) 减少 history 保存时间

  b) 减少 item 获取间隔时间

  c) 减少不必要的监控项

在条件不允许或者以上方法都无效的情况下，请一定考虑以上建议。在监控环境中，以上三点是大家都在犯的错误，大部分 item 是不需要保存太长的数据，有些监控项根本无意义，有些监控项的间隔时间太短。一直以来我都在犯

这个错，但是因为 zabbix 性能一直不错，暂时不纠正，数据多点总比少点好，是不是~

第十七章 Zabbix 实战

Zabbix API 开发实战：创建维护模式

前言

前段时间，运维生存时间群里有一位兄弟咨询 API 操作，由于时间忙，给了一份写好的 zabbix 维护脚本给他参考，脚本里面包含用户登录或者 token、获取主机信息、创建维护模式。有 python 脚本功力的同学直接看脚本。

脚本功能

zabbix_maintenance.py 执行本脚本，zabbix 将此服务器置为维护状态，周期为 10 分钟。本脚本只需要修改 zabbix 用户 ID、用户名、密码、网址即可

备注：zabbix 中的 hostname 必须与当前主机名一致

API 脚本

```
#!/usr/bin/env python
# -*-coding:utf-8-*-
import urllib
import urllib2
import json
import sys
import platform
import time



def auth(uid, username, password, api_url):
    """
    zabbix 认证
    :param uid:
```

```
    :param username:
    :param password:
    :return:
    """
    dict_data = {}
    dict_data['method'] = 'user.login'    #  方法
    dict_data['id'] = uid    #  用户 id
    dict_data['jsonrpc'] = "2.0"    # api 版本
    dict_data['params'] = {"user": username, "password": password}    #  用户账号密码
    jdata = json.dumps(dict_data)    #  格式化 json 数据
    content = post_data(jdata, api_url)    # post json 到接口
    return content    #  返回信息



def post_data(jdata, url):
    """
    POST 方法
    :param jdata:
    :param url:
    :return:
    """
    req = urllib2.Request(url, jdata, {'Content-Type': 'application/json'})
    response = urllib2.urlopen(req)
    # content = response.read()
    content = json.load(response)
    return content



def create_maintenance(name, hostid, active_since, active_till, period, auth_code, api_url):
    """
```

```
    create maintenance
    :return:
    """
    dict_data = {}
    dict_data['method'] = 'maintenance.create'    #  方法
    dict_data['id'] = uid    #  用户 id
    dict_data['jsonrpc'] = "2.0"    # api 版本
    dict_data['auth'] = auth_code    # api 版本
    dict_data['description'] = "UPDATE" + hostid    # api 版本
    # host
    hostids = [hostid]
    # timeperiods
    timeperiods = [{"timeperiod_type": 0, "start_time": 64800, "period": period}]
    dict_data['params'] = {"name": name, "active_since": active_since, "timeperiods": timeperiods,
                                        "active_till": active_till, "hostids": hostids}    #  用户账号密码
    jdata = json.dumps(dict_data)    #  格式化 json 数据
    content = post_data(jdata, api_url)    # post json 到接口
    return content    #  返回信息


def get_hostid(hostname, auth_code, uid, api_url):
    """
    use hostname get hostid
    :param hostname:
    :param auth:
    :param uid:
    :return:
    """
    dict_data = {}
    dict_data['method'] = 'host.getobjects'    #  方法
```

```
    dict_data['id'] = uid    #  用户 id
    dict_data['jsonrpc'] = "2.0"    # api 版本
    dict_data['params'] = {"name": hostname}    #  主机名
    dict_data['auth'] = auth_code    # auth 串
    jdata = json.dumps(dict_data)    #  格式化 json 数据
    content = post_data(jdata, api_url)    # post json 到接口
    return content    #  返回信息



def logout(uid, auth_code, api_url):
    """
    退出
    :param uid:
    :param auth_code:
    :return:
    """
    dict_data = {}
    dict_data['method'] = 'user.logout'    #  方法
    dict_data['id'] = uid    #  用户 id
    dict_data['jsonrpc'] = "2.0"    # api 版本
    dict_data['params'] = []
    dict_data['auth'] = auth_code    # auth 串
    jdata = json.dumps(dict_data)    #  格式化 json 数据
    content = post_data(jdata, api_url)    # post json 到接口
    return content    #  返回信息



if __name__ == '__main__':
    # user info
    uid = 5 #  用户 ID
```

```
    username = 'zabbix 用户名'
    password = 'zabbix 密码'
    api_url = "http://zabbix 网址/zabbix/api_jsonrpc.php"
    res = auth(5, username, password, api_url)  # 认证
    if res['result']:
        auth_code = res['result']  # 认证串
        hostname = platform.node()  # 主机名
        res = get_hostid(hostname, auth_code, uid, api_url)
        if res['result']:
            period = 600  # 维护时长
            active_since = int(time.time())  # 开始时间
            active_till = int(time.time()) + period  # 结束时间
            hostid = res['result'][0]['hostid']  # 主机
            res = create_maintenance('AutoMaintenance_' + hostname + '_' + str(active_since), hostid, active_since,
active_till, period,
                                     auth_code, api_url)  # 创建维护
            logout(uid, auth_code, api_url)  # 退出登录
            print res
    else:
        pass
```

备注：以上脚本适用于 zabbix2.4 及以下版本

如有相关问题，大家有如下问题可以咨询凉白开或者 SA 们：加入 QQ 群（看网站底部）、关注微信公众号 ttlsacom 提问、发送邮件至 suppor@ttlsa.com、在本文里留言！

Zabbix 监控 nginx 性能

需要使用 zabbix 监控 nginx，首先 nginx 需要配置 ngx_status，如果开启请看凉白开之前的文章《启用 nginx status 状态详解》

nginx status 信息

```
# curl http://127.0.0.1/ngx_status
Active connections: 11921
server accepts handled requests
 11989 11989 11991
Reading: 0 Writing: 7 Waiting: 42
```

以上为 nginx 性能计数，我们除了监控以上数据，还需要监控 nginx 进程状态，并且配置触发器！

zabbix 客户端配置

- 编写客户端脚本 ngx_status.sh

```
#!/bin/bash
# DateTime: 2015-10-25
# AUTHOR：凉白开
# WEBSITE: http://www.ttlsa.com
# Description：zabbix 监控 nginx 性能以及进程状态
# Note：此脚本需要配置在被监控端，否则 ping 检测将会得到不符合预期的结果
#  文章地址：http://www.ttlsa.com/zabbix/zabbix-monitor-nginx-performance/

HOST="127.0.0.1"
PORT="80"

#  检测 nginx 进程是否存在
```

```
function ping {
        /sbin/pidof nginx | wc -l
}
# 检测 nginx 性能
function active {
        /usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Active' | awk '{print $NF}'
}
function reading {
        /usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Reading' | awk '{print $2}'
}
function writing {
        /usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Writing' | awk '{print $4}'
}
function waiting {
        /usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Waiting' | awk '{print $6}'
}
function accepts {
        /usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| awk NR==3 | awk '{print $1}'
}
function handled {
        /usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| awk NR==3 | awk '{print $2}'
}
function requests {
        /usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| awk NR==3 | awk '{print $3}'
}
# 执行 function
$1
```

- zabbix 客户端文件配置

将自定义的 UserParameter 加入配置文件，然后重启 agentd，如下：

```
#cat /usr/local/zabbix-3.0.0/etc/zabbix_agentd.conf | grep nginx
UserParameter=nginx.status[*],/usr/local/zabbix-3.0.0/scripts/ngx-status.sh $1
# killall zabbix_agentd
# /usr/local/zabbix-3.0.0/sbin/zabbix_agentd
```

- zabbix_get 获取数据

此步骤可以跳过，但是最好是测试一下，因为通过此命令我们可以检测配置是否正确

```
# /usr/local/zabbix-3.0.0/bin/zabbix_get -s 10.10.1.121 -k 'nginx.status[accepts]'
9570756
# /usr/local/zabbix-3.0.0/bin/zabbix_get -s 10.10.1.121 -k 'nginx.status[ping]'
1
```

在 zabbix server 服务器上执行如上命令，10.10.1.121 为 agentd 机器

zabbix web 端配置

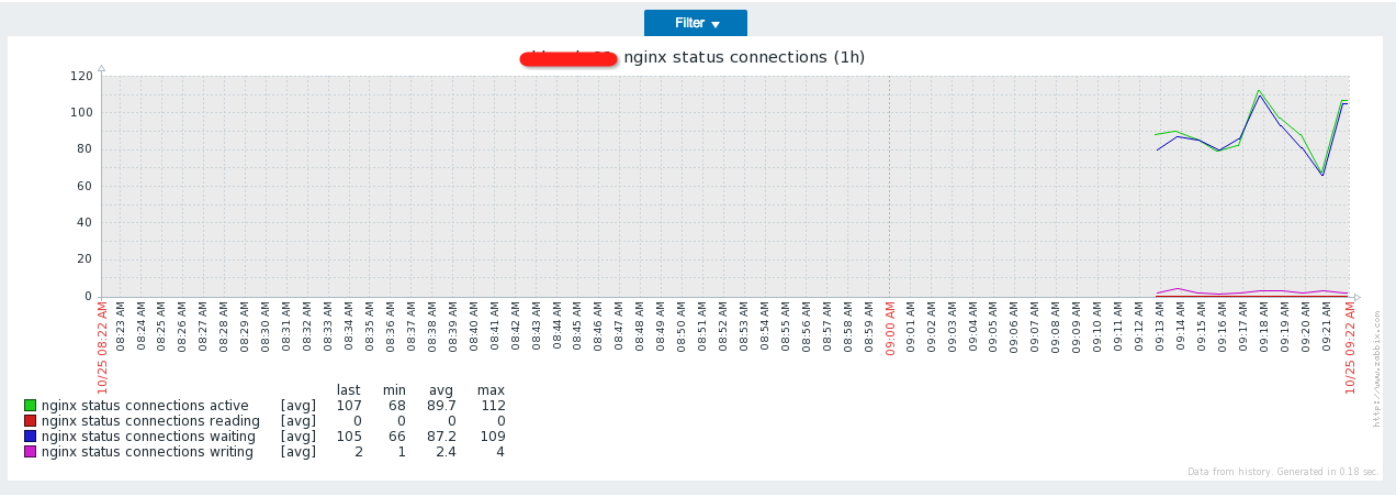
- 导入 Template App NGINX 模板

模板我已经写好了，将会放到附件中。具体导入方式请看前面的文章《zabbix 链接及解除模板链接（76）》

- Link NGINX 模板

到了最后一个阶段，登陆 zabbix 管理端，link 模板到 nginx 服务器：configuration->hosts->点击 nginx 所在服务器->点击 template->Link new templates 输入"Template App NGINX"->Add->最后点击 update。

监控效果



模板附件下载：zabbix_monitor_nginx_template_ttlsa_com

模板请到 ttlsa 官方网站下载

zabbix 监控 php-fpm 性能状态

不多说，首先你需要开启 php-fpm 的状态页，请参考凉白开前面写的文章《启用 php-fpm 状态详解》,然后更我一步一步来完成 zabbix 对 php-fpm 的监控。

zabbix 客户端配置

- 增加自定义 key

```
# cat zabbix_agentd.conf | grep 'php-fpm'
UserParameter=php-fpm.status[*],/usr/bin/curl -s "http://127.0.0.1/status?xml" | grep "<$1>" | awk -F'>|<' '{ print $$3}'
```

在此没有额外使用脚本，比网络上的其他脚本要精练不少！

- 重启 zabbix 客户端

```
# killall zabbix_agentd
# /usr/local/zabbix-3.0.0/sbin/zabbix_agentd
```

依据自己的情况来重启你的 zabbix 客户端

zabbix 管理后台配置

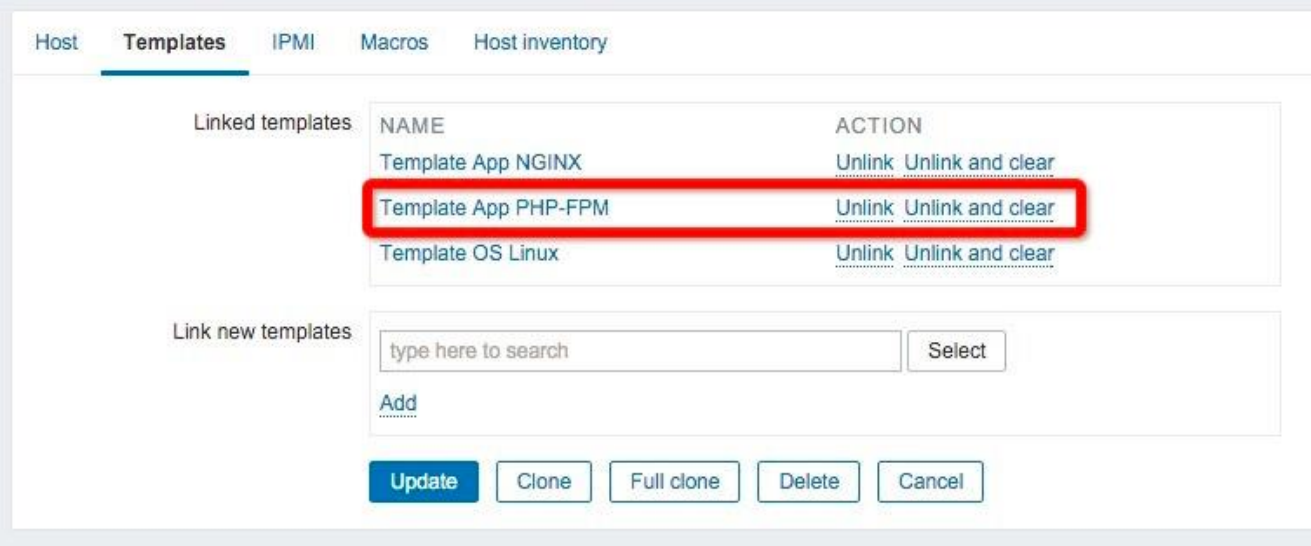
- 导入 php-fpm 模板

附件中会给出模板（备注：更多关于 zabbix 模板内容，请看前面关于模板的章节，我不在多说）

进入后台->configuration->templates->import（右侧）->选择在 ttlsa 下载的模板->最后点击 import。至此 php-fpm 模板已经导入到 zabbix 中。

- Link/关联模板

接下来需要把 php-fpm 模板 link 到你的主机上，进入后台->configuration->点击你的主机->templates->输入 php-fpm 模板名称，点击 add，最后点击 update 即可



还有另外一种更快的方法，点击 php-fpm 模板，然后将主机添加进去即可，都是一些简单的操作，不在截图演示！

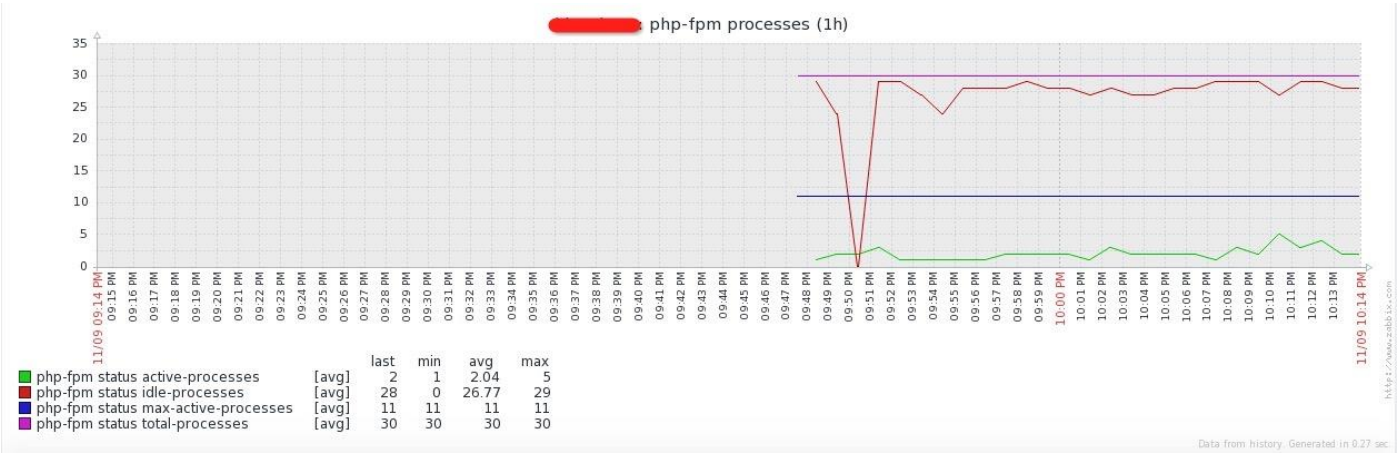
效果展示

- latest 数据

☐	php-fpm is running	11/09/2015 10:09:39 PM	31		Graph
☐	php-fpm status accepted-conn	11/09/2015 10:10:28 PM	271	+184	Graph
☐	php-fpm status active-processes	11/09/2015 10:10:29 PM	5	+3	Graph
☐	php-fpm status idle-processes	11/09/2015 10:10:30 PM	27	-2	Graph
☐	php-fpm status listen-queue	11/09/2015 10:10:32 PM	0		Graph
☐	php-fpm status listen-queue-len	11/09/2015 10:10:31 PM	0		Graph
☐	php-fpm status max-active-processes	11/09/2015 10:10:33 PM	11		Graph
☐	php-fpm status max-children-reached	11/09/2015 10:10:34 PM	0		Graph
☐	php-fpm status max-listen-queue	11/09/2015 10:10:35 PM	1		Graph
☐	php-fpm status slow-requests	11/09/2015 10:10:36 PM	2	-1	Graph
☐	php-fpm status start-since	11/09/2015 10:10:37 PM	01:16:19	+00:01:00	Graph
☐	php-fpm status total-processes	11/09/2015 10:10:38 PM	30		Graph

- 慢日志数量

- php-fm processes



php-fpm 进程退出会触发告警，我不多做演示，有相关问题在此留言。继续关注 zabbix，继续关注 ttlsa。最近比较忙，都没来得及给投稿的小伙伴发红包，实在是抱歉了！

zabbix php-fpm 文件下载

zabbix 监控 php-fpm 模板-zabbix 3.x

zabbix 监控 php-fpm 模板-zabbix 2.x

以上模板，请到官方网站下载

zabbix 监控 Tomcat/JVM 实例性能

最近陆续写了几个 zabbix 监控实例，也提供了监控模板。今天群里有兄弟问什么时候出一个 zabbix 通过 JMX 监控 tomcat 的文章。鉴于 tomcat 用户群体大而且之前也有很多兄弟有过相似需求，今天在此分享 zabbix 监控 tomcat 一文。

安装 Tomcat

既然你需要监控 tomcat，基本说明你已经安装好 tomcat。如有正好你不会安装 tomcat，那么请参考 TTLSA 之前的文章《Tomcat7 安装》

配置 Tomcat JMX

- 配置 jmx

编辑 catalina.sh，加入如下配置

```
# vim /usr/local/tomcat-7.0.65/bin/catalina.sh
CATALINA_OPTS="-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.port=12345"
```

启动 Tomcat

```
# /usr/local/tomcat-7.0.65/bin/startup.sh
```

zabbix server 配置

首先我们需要了解一下他们的对应关系，zabbix_server 开启 java poller, zabbx_java 开启 JavaGateway，端口为 10052，Tomcat JMX 开启 12345 提供性能数据。

数据获取：java poller<-->JavaGateway:10052<-->Tomcat:12345.

Java 支持

编译安装 zabbix server 需要加上--enable-java 以支持 jmx 监控，如果之前的 zabbix server 没加，那么请重新编译安装，分享下我的安装参数：

```
# ./configure --prefix=/usr/local/zabbix-3.0.0/ --enable-server --enable-agent --with-mysql --with-net-snmp --with-libcurl
--with-libxml2 --enable-java
```

启动 zabbix_java

```
# /usr/local/zabbix-3.0.0/sbin/zabbix_java/startup.sh
```

zabbix_server.conf 配置

默认未启用 JavaPollers，需要修改如下配置

```
# cat /usr/local/zabbix-3.0.0/etc/zabbix_server.conf | grep Java | grep =
JavaGateway=127.0.0.1
JavaGatewayPort=10052
StartJavaPollers=5
```

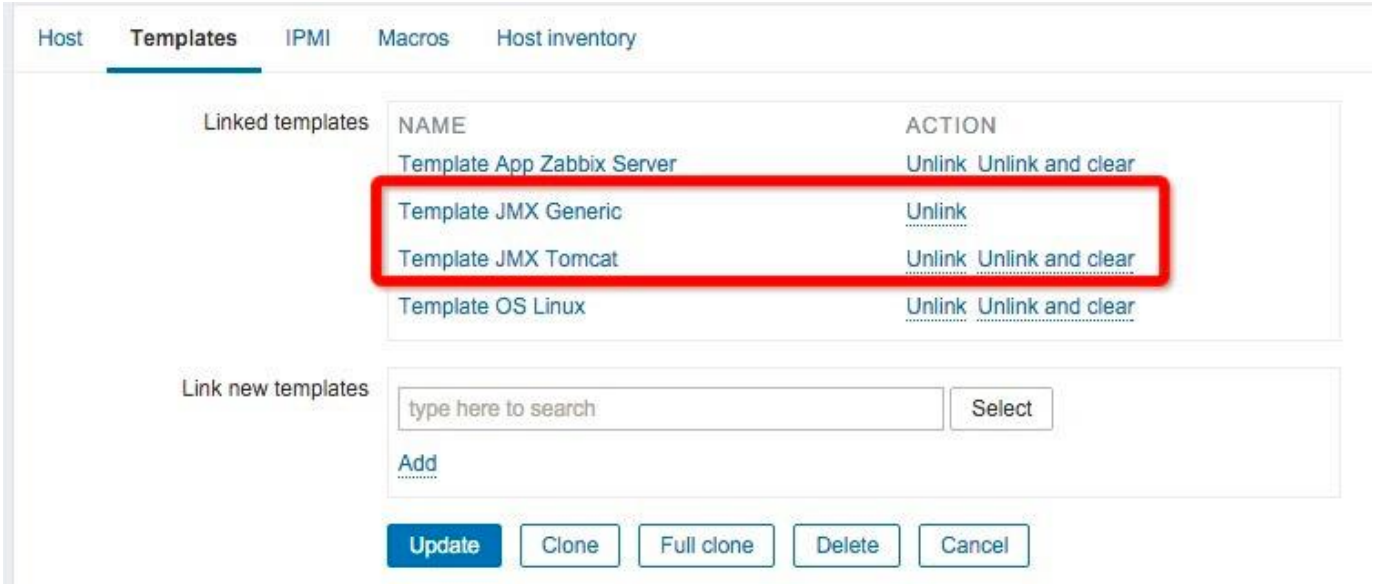
Zabbix 图形界面配置

主机增加 JMX 接口

进入后台，configuration->hosts->选择你的主机->jmx interface 点击 add，输入对应的 tomcat ip 地址和 jmx 端口，如下图：

Link TOMCAT 模板

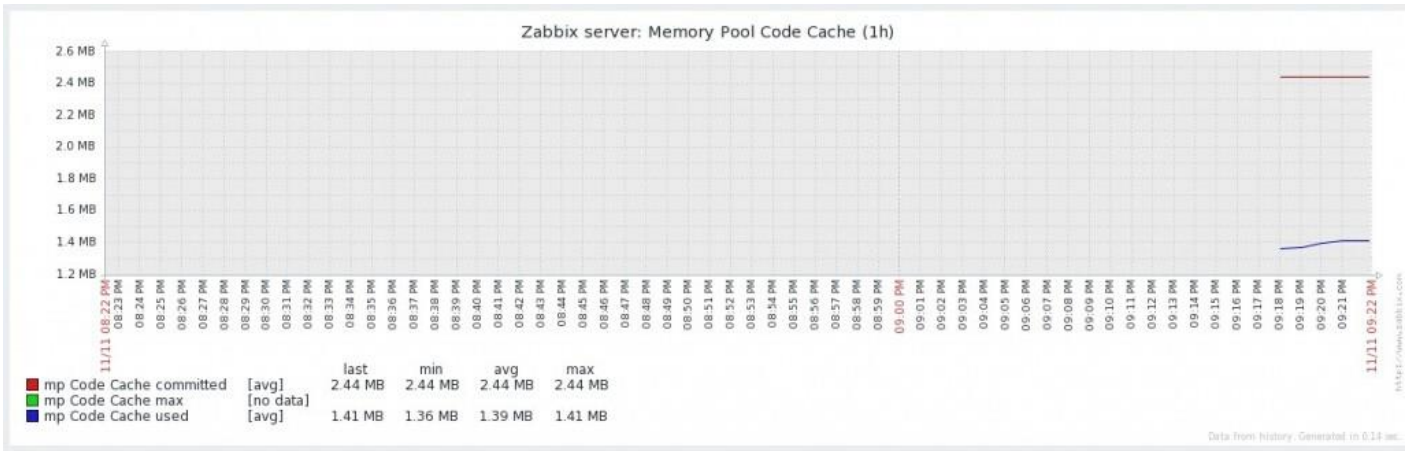
切换到 templates 选项卡，选择 zabbix 自带的 tomcat/JMX 模板，如下



最后 Update 即可。接下去等数据。

效果展示

自带的 jmx 两个模板监控项目非常多，自己去摸索。



zabbix 监控日志文件 MySQL 日志为例

一般情况下，日志最先反映出应用当前的问题，在海量日志里面找到我们异常记录，然后记录下来，并且根据情况报警，大家可以监控系统日志、nginx、Apache、业务日志。这边我拿常见的 MySQL 日志做监控，大家看演示。

监控日志 key

首先要了解 key，

```
log[ file   ,<regexp>,<encoding>,<maxlines>,<mode>,<output>]
```

file：文件名，写绝对路径

regexp：要匹配内容的正则表达式，或者直接写你要检索的内容也可以，例如我想检索带 ERROR 关键词的记录

encoding：编码相关，留空即可

maxlines：一次性最多提交多少行，这个参数覆盖配置文件 zabbxi_agentd.conf 中的’MaxLinesPerSecond’，我们也可以留空

mode：默认是 all，也可以是 skip，skip 会跳过老数据

output：输出给 zabbix server 的数据。可以是\1、\2 一直\9，\1 表示第一个正则表达式匹配出得内容，\2 表示第二个正则表达式匹配错的内容。

备注：我极力推荐大家使用第二个参数，看到网上一些 zabbix 监控日志的教程，几乎只有第一个参数，这样将会导致日志文件里的内容统统丢给 zabbix_server 记录，我想，这一定不是大家想看到的。

日志文件权限配置

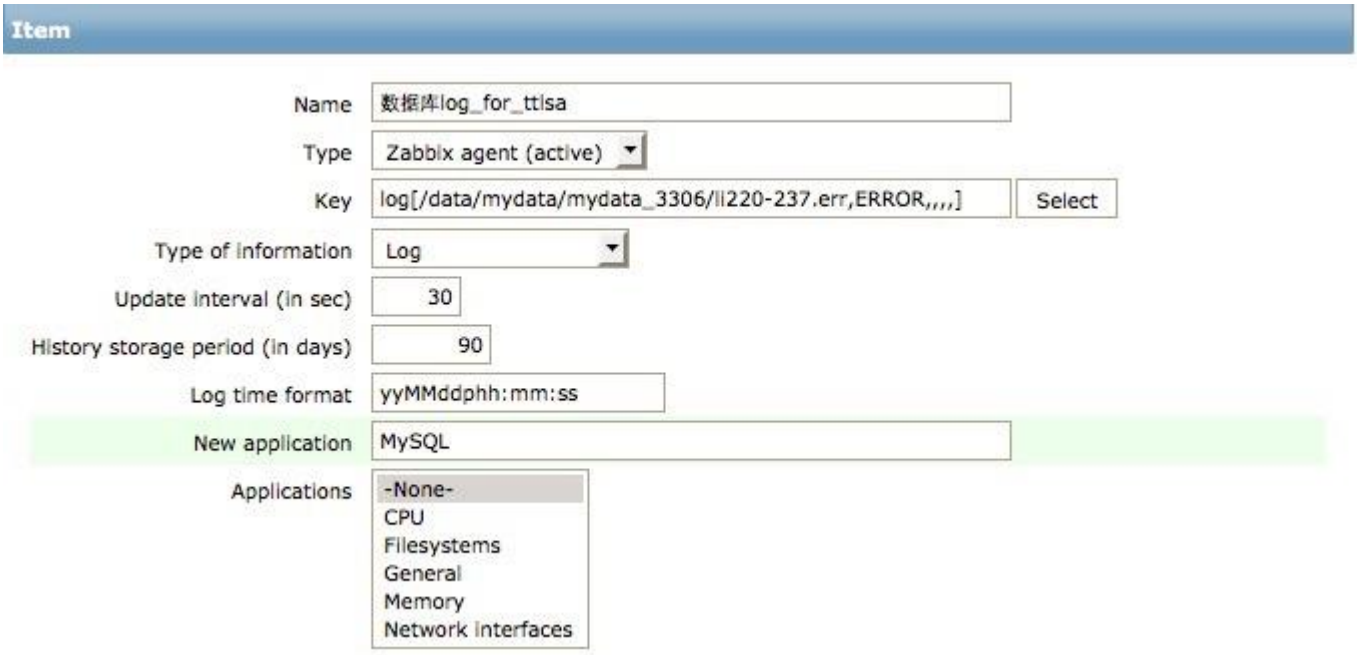
给日志文件加上读取权限，为了演示方便，我直接给 777

```
# chmod 777 /data/mydata/mydata_3306/li220-237.err
如果权限给的不到位，zabbix agent 日志有类似如下报错：
4780:20150311:135742.401 cannot open '/data/mydata/mydata_3306/li220-237.err': [13] Permission denied
 4780:20150311:135742.401 active check "log[/data/mydata/mydata_3306/li220-237.err,ERROR,,,,]" is not supported
```

```
4779:20150311:135742.402 cannot open '/data/mydata/mydata_3306/li220-237.err': [13] Permission denied
4779:20150311:135742.402 active check "log[/data/mydata/mydata_3306/li220-237.err,ERROR,,,,]" is not supported
```

zabbix 配置

Host>>目标主机>>item>>create item，如下：



说明：

- type 必须选择 zabbix agent（active），因为数据是 zabbix 被监控的主动提交给 server
- key：log[/data/mydata/mydata_3306/li220-237.err,ERROR,,,,]，我不多说了，细心的人会说，还有一个叫 logrt 得 key，有什么区别，等会儿讲.
- log time format：yyMMddphh:mm:ss，对应日志的行头 150311 11:47:09，y 表示年、M 表示月、d 表示日、p 和: 一个占位符，h 表示小时，m 表示分钟，s 表示秒。

zabbix 监控 MySQL 日志查看

切换到最新日志里面，找到相应数据，如下是我的监控截图

Timestamp	Local time	Value
2015-03-11 22:02:13	-	150122 17:02:49 [ERROR] /usr/libexec/mysqld: Incorrect key file for table './zhengdazhi_com/wp_o
2015-03-11 22:02:13	-	150122 17:02:49 [ERROR] /usr/libexec/mysqld: Incorrect key file for table './zhengdazhi_com/wp_o
2015-03-11 22:02:13	-	150122 17:02:49 [ERROR] /usr/libexec/mysqld: Incorrect key file for table './zhengdazhi_com/wp_o
2015-03-11 22:02:13	-	150122 17:02:49 [ERROR] /usr/libexec/mysqld: Incorrect key file for table './zhengdazhi_com/wp_o
2015-03-11 22:02:13	-	150122 17:02:49 [ERROR] /usr/libexec/mysqld: Incorrect key file for table './zhengdazhi_com/wp_o
2015-03-11 22:02:13	-	150122 17:02:49 [ERROR] /usr/libexec/mysqld: Incorrect key file for table './zhengdazhi_com/wp_o

接下来便是触发器，大家可以根据自己的情况来创建触发器，例如日志中包含某个字符串等等，如上图，我们可以触发执行 mysql 表修复。

logrt 介绍

key：

```
log[ file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]

logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>]
```

如果仔细看可以发现，第一个参数不一样，logrt 的第一个参数可以使用正则表达式。针对日志回滚用得，例如我们每天都切割 nginx 日志，日志名位 www.ttlsa.com_2017-01-01.log、www.ttlsa.com_2017-01-02.log 等等，使用 log 肯定不合适，如果文件名使用正则，那么新增的日志文件会立即加入监控。

备注：不管新日志、老日志，只要他们有变更，zabbix 都会监控。

zabbix 监控惠普打印机

假设公司有多个楼层或者分布在不同楼，打印机自然分布很广泛，打印机缺少油墨或者卡纸了，都需要员工找 IT 部门。我们使用 zabbix 对打印机进行监控，一旦缺少油墨，zabbix 发出报警，it 人员能够及时更换，让打印机一直处在不间断的工作状态。如果卡纸也能第一时间赶赴现场，迅速解决问题。

我们今天监控的主要项目是油墨，卡纸这块请根据对应的 snmp 来做

开启打印机 SNMP

登陆打印机 web 地址：http://192.168.1.20/（我当前的），网络>>SNMP>>勾选"启用 SNMP v1/v2 只读访问(将 'public' 用于获取社区名称)"

油墨剩余量 OID

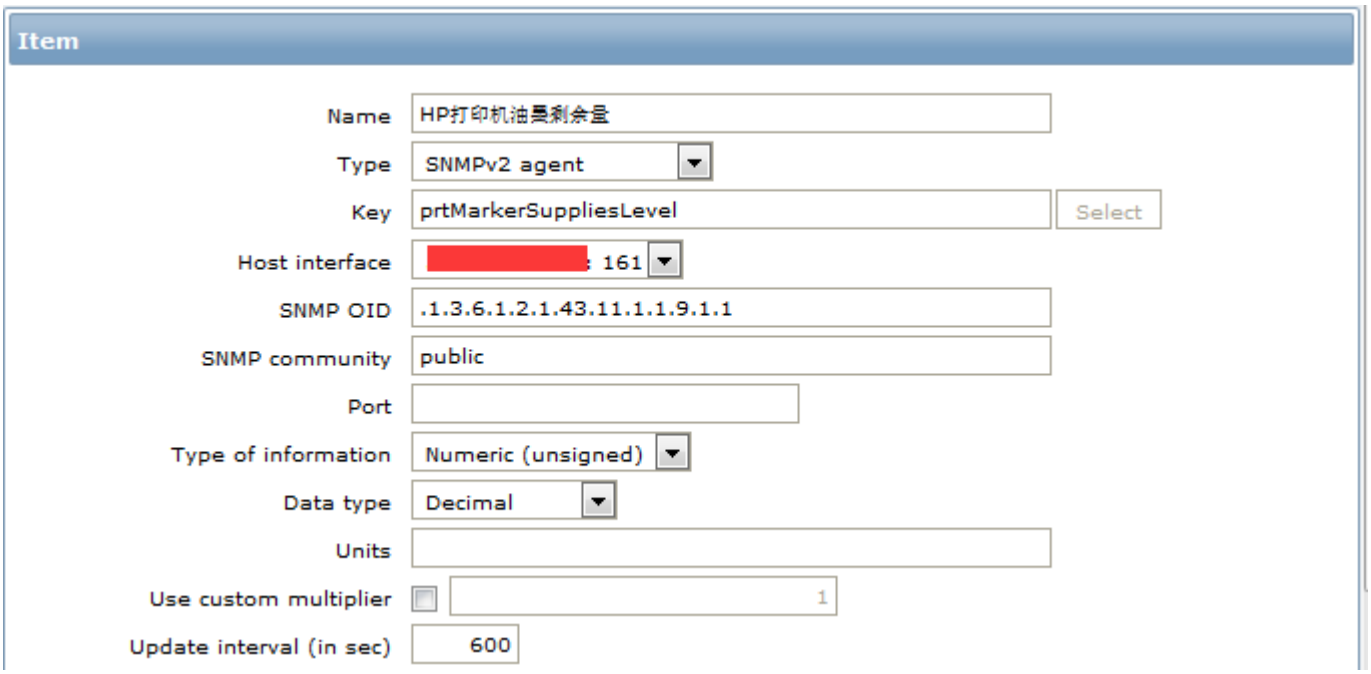
```
# snmpwalk -v 1 -c public 192.168.1.20    .1.3.6.1.2.1.43.11.1.1.9.1.1

SNMPv2-SMI::mib-2.43.11.1.1.9.1.1 = INTEGER: 30
```

可以看出，我们当前油墨剩余量是 30%，与 web 管理后台的剩余量一致

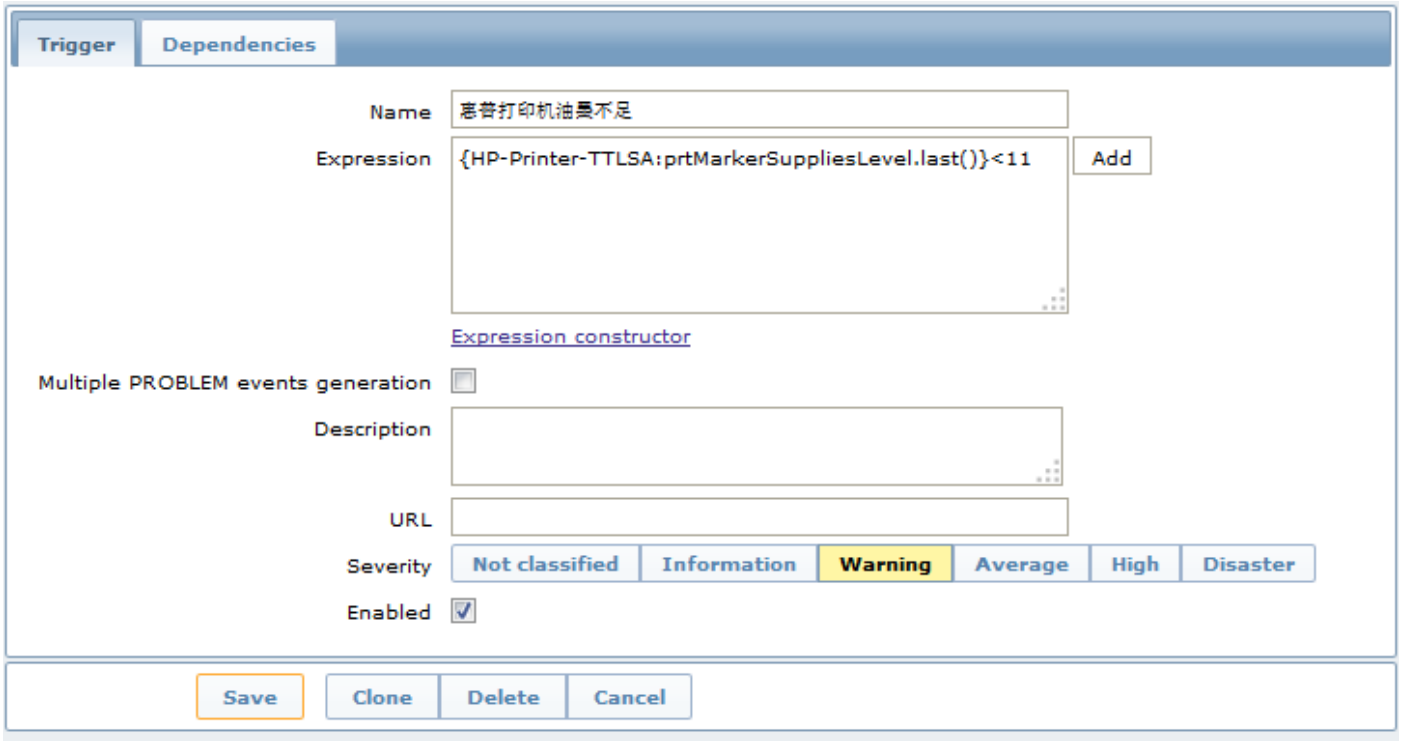
创建主机

configuration>>HOST>>create host，type 选择 SNMPv2 agent，key 其实意义不大，OID：.1.3.6.1.2.1.43.11.1.1.9.1.1，更新时间大家自己发挥，其他都用默认，想了解更多关于 zabbix 使用 snmp 监控，请回头看 ttlsa 相关文章。

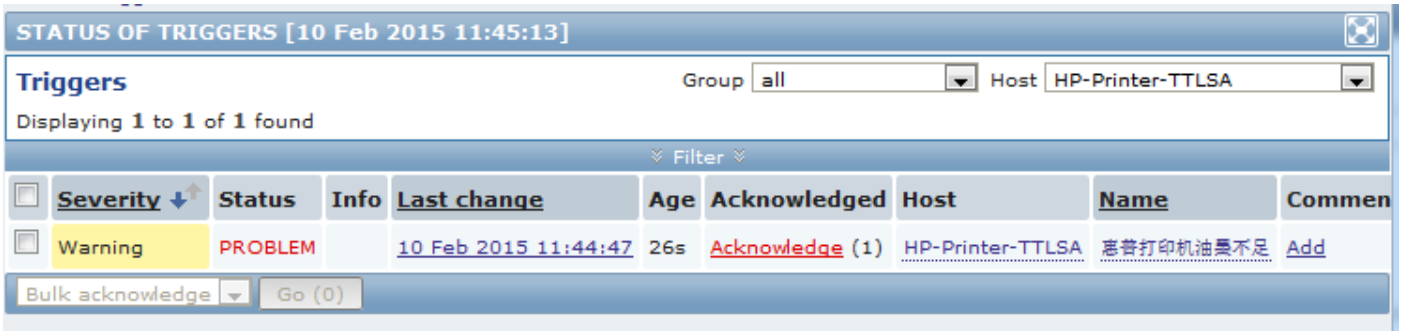


创建触发器

当油墨小于 10%，trigger 触发 warnning。出现 warnning 之后，接下来的便是邮件报警了。



当油墨不足时，trigger 报警如下



打印机 OID：http://www.oidview.com/mibs/o/Printer-MIB.html

zabbix 如何监控多个 JMX/Redis 等实例

本文主要主要是谈思路，不谈具体的配置过程。推荐有 zabbix 基础的人看，特别是有 zabbix 自动发现、lld（low-level-discovery）经验的同学。

zabbix 自动发现

如果没有了解过 zabbix 自动发现，请看以下内容

- 《zabbix 发现介绍》整个功能的介绍
- 《zabbix 发现配置》server 通过配置好的规则，自动添加 host、group、template
- 《zabbix Active agent 自动注册》与 discovery 相反，功能基本相同，active 联系 server，server 自动添加 host、group、template
- zabbix low-level discover zabbix 批量部署必备（85）

特别是最后一篇 LLD

为什么不能监控多个实例

生产环境上一台服务器安装多个 redis、mongodb、mysql、tomcat 等等实例。同一个 zabbix 主机上，不能同时存在一个相同的 key。如果监控 jxm，即使添加两个 jmx 接口也不行！

如何监控多个实例？

答案：把 key 改成不相同。例如：监控 redis key 的数量，一般情况下，key 如下：

```
redis[keys]
```

添加第一个 item，很快就报错了，多实例可以改为如下:

```
redis[keys_10001]
redis[keys_10002]
```

keys 加上端口号，即可解决 key 重复的问题，至于怎么取数据，我不多说了。mongodb、jmx 等等都配合使用。

配合自动发现

我总是不厌其烦的推荐 SA 们一定要看 zabbix 自动发现，这是 zabbix 精髓所在。生产中，一台服务器上可能会存在多个监控实例，比如：A 服务器 2 个，B 服务器 4 个，C 服务器 1 个。单单靠套模板来完成监控，做法那太糟糕了。解决 LLD，所有相关监控实例童童自动被加入监控项。

zabbix 监控 mysql 性能

今天来看看 zabbix 如何监控 mysql 性能，这边使用 mysql 自带的模板，可以监控如下内容：OPS（增删改查）、mysql 请求流量带宽，mysql 响应流量带宽，最后会附上相应的监控图！

编写 check_mysql.sh 脚本

用于获取 mysql 性能指标数据，你需要修改相应的数据库信息

```
# vim /usr/local/zabbix-2.4.4/scripts/chk_mysql.sh
```

脚本如下：

```
#!/bin/bash
# ----------------------------------------------------------------------
# FileName:     check_mysql.sh
# Revision:     1.0
# Date:         2015/06/09
# Author:       DengYun
# Email:        dengyun@ttlsa.com
# Website:      www.ttlsa.com
# Description:
# Notes:        ~
# ----------------------------------------------------------------------
# Copyright:    2015 (c) DengYun
# License:      GPL

#  用户名
MYSQL_USER='zabbix'

#  密码
```

```
MYSQL_PWD='123456'

# 主机地址/IP
MYSQL_HOST='127.0.0.1'

# 端口
MYSQL_PORT='3306'

# 数据连接
MYSQL_CONN="/usr/bin/mysqladmin -u${MYSQL_USER} -p${MYSQL_PWD} -h${MYSQL_HOST} -P${MYSQL_PORT}"

# 参数是否正确
if [ $# -ne "1" ];then
    echo "arg error!"
fi

# 获取数据
case $1 in
    Uptime)
        result=`${MYSQL_CONN} status|cut -f2 -d":"|cut -f1 -d"T"`
        echo $result
        ;;
    Com_update)
        result=`${MYSQL_CONN} extended-status |grep -w "Com_update"|cut -d"|" -f3`
        echo $result
        ;;
    Slow_queries)
        result=`${MYSQL_CONN} status |cut -f5 -d":"|cut -f1 -d"O"`
        echo $result
        ;;
```

```
Com_select)
        result=`${MYSQL_CONN} extended-status |grep -w "Com_select"|cut -d"|" -f3`
        echo $result
                ;;
Com_rollback)
        result=`${MYSQL_CONN} extended-status |grep -w "Com_rollback"|cut -d"|" -f3`
                echo $result
                ;;
Questions)
        result=`${MYSQL_CONN} status|cut -f4 -d":"|cut -f1 -d"S"`
                echo $result
                ;;
Com_insert)
        result=`${MYSQL_CONN} extended-status |grep -w "Com_insert"|cut -d"|" -f3`
                echo $result
                ;;
Com_delete)
        result=`${MYSQL_CONN} extended-status |grep -w "Com_delete"|cut -d"|" -f3`
                echo $result
                ;;
Com_commit)
        result=`${MYSQL_CONN} extended-status |grep -w "Com_commit"|cut -d"|" -f3`
                echo $result
                ;;
Bytes_sent)
        result=`${MYSQL_CONN} extended-status |grep -w "Bytes_sent" |cut -d"|" -f3`
                echo $result
                ;;
Bytes_received)
        result=`${MYSQL_CONN} extended-status |grep -w "Bytes_received" |cut -d"|" -f3`
```

```
                echo $result
                ;;
        Com_begin)
                result=`${MYSQL_CONN} extended-status |grep -w "Com_begin"|cut -d"|" -f3`
                echo $result
                ;;

        *)
        echo
"Usage:$0(Uptime|Com_update|Slow_queries|Com_select|Com_rollback|Questions|Com_insert|Com_delete|Com_com
mit|Bytes_sent|Bytes_received|Com_begin)"
        ;;
Esac
```

修改 zabbix_agentd.conf

增加自定义 key，在最后一行增加如下：

```
# 获取 mysql 版本
UserParameter=mysql.version,mysql -V
# 获取 mysql 性能指标,这个是上面定义好的脚本
UserParameter=mysql.status[*],/usr/local/zabbix-2.4.4/scripts/chk_mysql.sh $1
# 获取 mysql 运行状态
UserParameter=mysql.ping,mysqladmin -uzabbix -p123456 -P3306 -h127.0.0.1    ping | grep -c alive
```

备注：请注意修改你的数据库信息，以及 zabbix 路径信息

重启 zabbix

```
# killall zabbix-agentd
# /usr/local/zabbix-2.4.4/bin/zabbix_agentd
```

或者

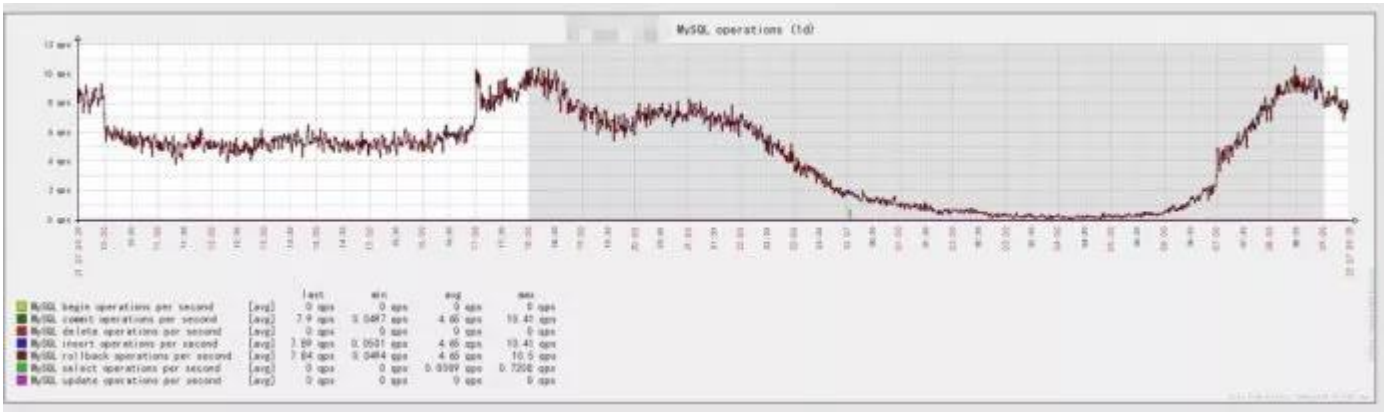
```
# service zabbix_agentd restart
```

Link MySQL 模板

模板是 zabbix 系统提供的，进入 zabbix web 后台，configuration-->hosts-->点击你的主机 name-->选择 template 选项卡，选择模板“Template App MySQL”，最后点击 update 即可

数据查看

如果配置没有异常，那么可以在 graph 中查看到 2 张监控图，分别为请求流量带宽、响应流量带宽、ops，点击 monitoring-->graphs-->选择你的主机，分别选择 Graph“MySQL bandwidth”、“MySQL operations”，监控图分别如下（图片可以点击放大查看）：



常见错误解决思路

如果发现监控没有数据，请排查如下问题

1. zabbix 客户端是否重启
2. 脚本是否有执行权限
3. 数据库是否有权限
4. 环境变量是否有问题
5. 请看 zabbix item 列，鼠标移至红色叉上，有错误提示。

zabbix 监控磁盘 IO low-level-discory 方式

Linux io 监控的方式很多，这次使用 zabbix 对 Linux 磁盘 IO 做一个监控。需要下载三个文件，文章后面我会提供一个下载地址给大家，跟着凉白开的步骤来~

名称	修改日期	类型	大小
discover_disk.pl	2015-7-31 13:40	PL 文件	3 KB
readme.txt	2015-7-31 13:45	文本文档	1 KB
zbx_export_templates.xml	2015-7-31 13:43	XML 文档	40 KB
zbx_parse_iostat_values.sh	2015-7-31 13:40	Shell Script	2 KB

文件介绍

- discover_disk.pl

找出当前系统的分区，例如 sda、sdb、xvda 等等

- zbx_export_templates.xml

写好的模板，导进去即可

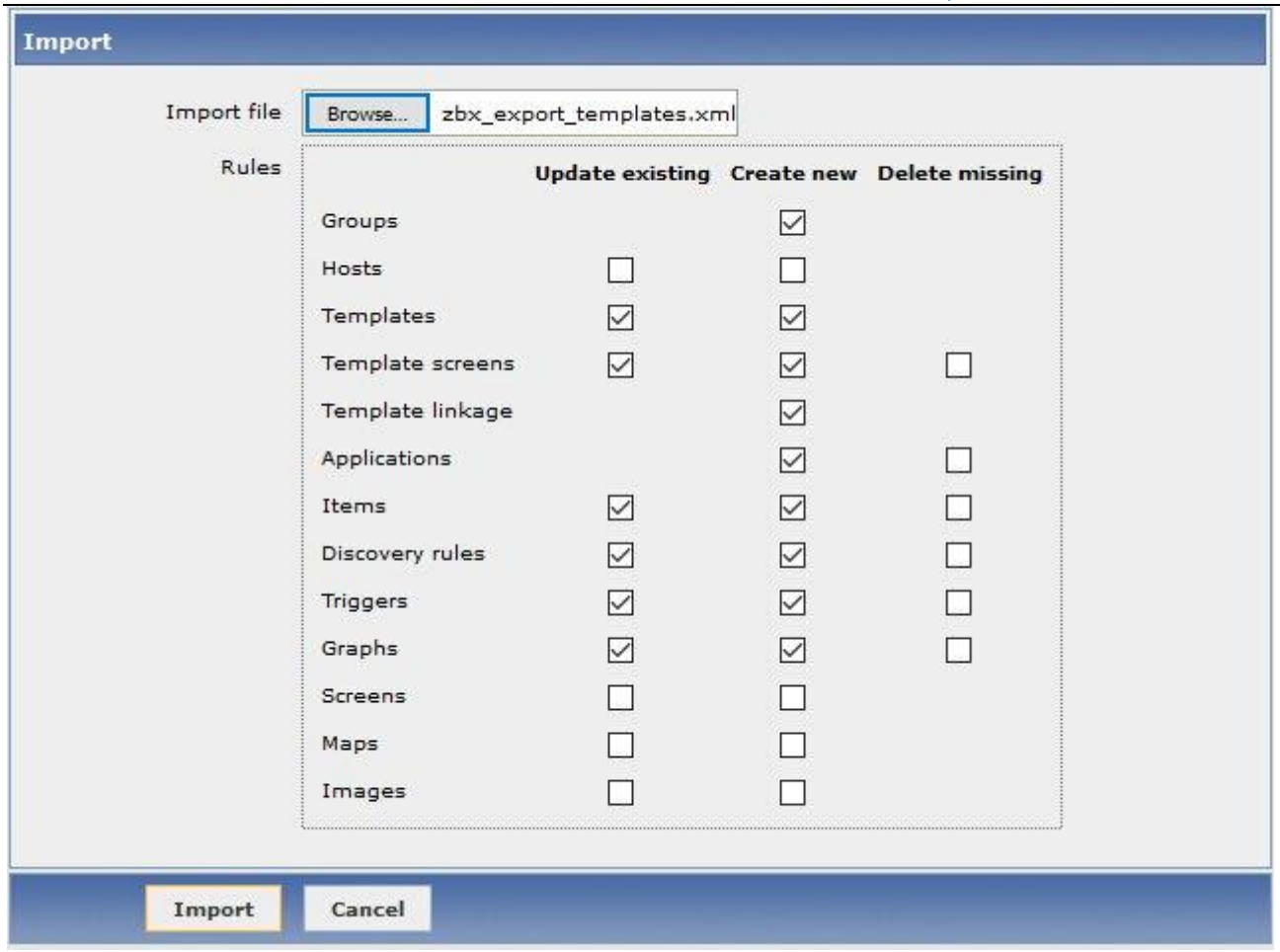
- zbx_parse_iostat_values.sh

定义 key 需要的文件，用于获取 io 监控值

备注：discover_disk.pl、zbx_parse_iostat_values.sh 这两个文件我 copy 到了/usr/local/zabbix-2.4.4/scripts/目录下（这是我新创建用于存放脚本的目录，大家可自行定义）

导入模板

configuration->templates->import，选择你需要导入的模板文件：zbx_export_templates.xml，如下图：



Link 模板

在 host 中 link 模板，configuration->Hosts->点击你的主机->templates->添加磁盘 IO 监控模板，如下：

配置 LLD

① 正则配置

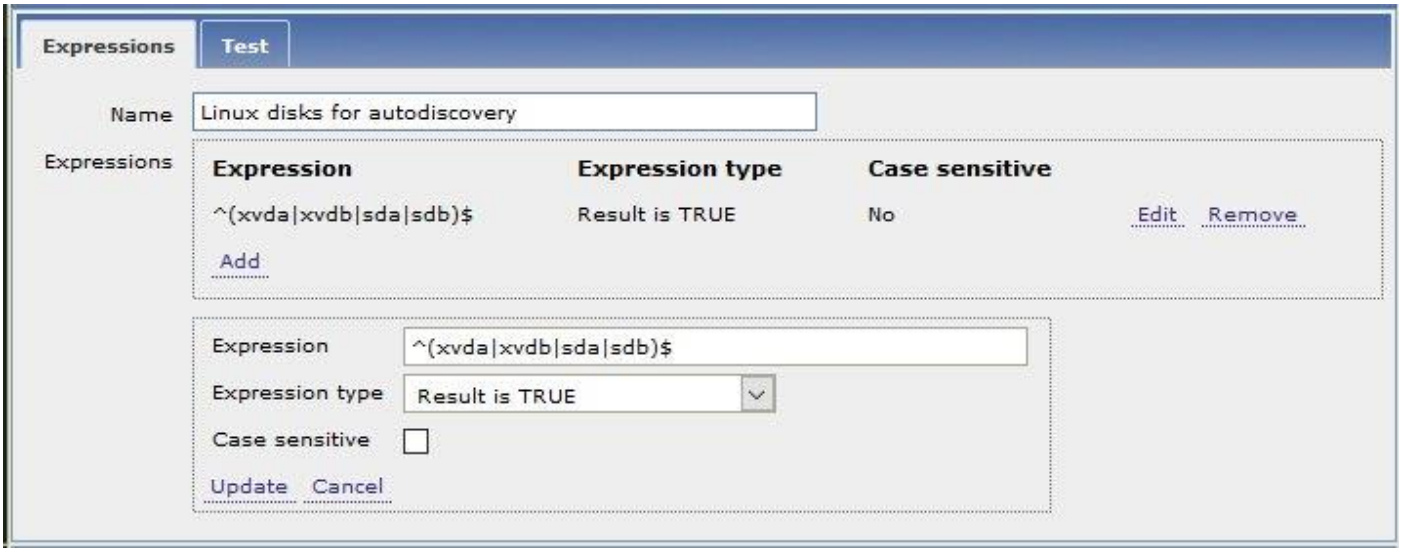
因为扫描出的分区比较多，我们监控特定的分区 IO，需要使用 zabbix 正则表达式过滤，Administration->General->右侧下来条选择“Regular expressions”->New regular expression（创建正则）

expression：^(xvda|xvdb|sda|sdb)$

Expression type：return is True

Case sensitive：留空

最后保存即可，如果你想对正则表达式有更进一步了解，请看凉白开以前写的《zabbix 正则表达式（86）》



② low-level-discovery 配置

模板中已经配置，无需配置！

zabbix agent 配置

配置 zabbix_agentd.conf，在最后加上如下配置：

```
# diskio discovery
UserParameter=discovery.disks.iostats,/usr/local/zabbix-2.4.4/scripts/discover_disk.pl
UserParameter=custom.vfs.dev.iostats.rrqm[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "rrqm/s"
UserParameter=custom.vfs.dev.iostats.wrqm[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "wrqm/s"
UserParameter=custom.vfs.dev.iostats.rps[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "r/s"
UserParameter=custom.vfs.dev.iostats.wps[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "w/s"
UserParameter=custom.vfs.dev.iostats.rsec[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "rsec/s"
UserParameter=custom.vfs.dev.iostats.wsec[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "wsec/s"
UserParameter=custom.vfs.dev.iostats.avgrq[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "avgrq-sz"
```

```
UserParameter=custom.vfs.dev.iostats.avgqu[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "avgqu-sz"
UserParameter=custom.vfs.dev.iostats.await[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "await"
UserParameter=custom.vfs.dev.iostats.svctm[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "svctm"
UserParameter=custom.vfs.dev.iostats.util[*],/usr/local/zabbix-2.4.4/scripts/zbx_parse_iostat_values.sh $1 "%util"
```

重启 agent

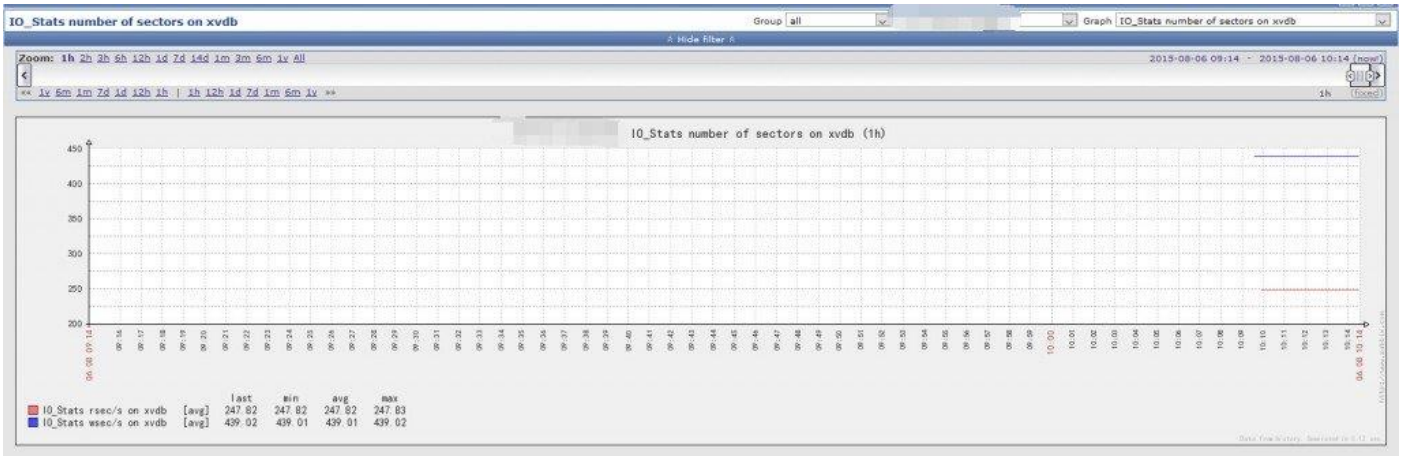
```
killall zabbix_agentd
/usr/local/zabbix-2.4.4/sbin/zabbix_agentd
```

或者

```
service zabbix_agentd restart
```

数据查看

请等待几分钟，查看 latest data，或者查看对应主机的 item，看看是否出现了 io stats 的监控项或者监控数据。如果想查看图形数据，可到 graph 中看。如下：



zabbix io 监控文件

下载地址：zabbix 磁盘监控

请到官方下载 www.ttlsa.com

zabbix 2.4 升级至 zabbix 2.5/3.0 过程

八月十九日 zabbix2.5 发布，可以发现 zabbix2.5 对应的文档却是 zabbix3.0，估计 zabbix2.5 只是一个过渡版本。此次更新较大，整个 web 界面风格发生变化，好在 web 结构变化较少（所以，大家依旧可以学习运维生存时间 ttlsa 系列文章），如果当前版本为 zabbix 2.x，那么可以直接升级至 zabbix 2.5，如果版本比 zabbix 2.x 还要低，那么请先升级到 zabbix 2.x 然后升级到 zabbix 2.5

以下为 zabbix 升级方法：

1. 关闭 zabbix server

防止有新的数据提交到数据库中、直接关闭数据库效果也是一样的

2. 备份

2.1 备份数据库

最简单的备份：关闭数据，整个数据库目录 copy 一份。虽说升级一般不会出现什么问题，但是安全起见还是有必要备份一下，就算升级成功，但是不能保证新版本让你喜欢，这个时候回退也有后路。

2.2 备份文件

备份配置文件（通常是/etc/zabbix）、php 网站源码、zabbix 二进制文件（整个程序目录备份就 OK）

备注：不要懒，一定记得备份，如果出现任何意外，你还有退路

3. 安装配置 zabbix

3.1 安装 Zabbix server

重新安装一次 zabbix，也就是 configure --... make make install，参数与老的 zabbix 一致

3.2 重新配置文件

zabbix_server.conf 配置参数可能会有变化，修改变更后的参数，或者直接修改新的配置文件。

备注：一般高版本 zabbix server 兼容低版本 zabbix 客户端。如果发现有异常，那么你的 zabbix 客户端也相应升级一下。客户端升级比较简单：更新二进制文件，配置文件对照下是否有修改即可。

4. 启动 zabbix

启动 zabbix，查看日志（一般在/tmp 目录下），看下 zabbix 的运行是否成功，成功运行之后 zabbix 将会自动更新数据库表结果。

备注：启动服务器之前，一定要确保有对 zabbix 数据库有足够的权限（一般情况下，我们都是给所有权限，所以基本不会出现问题）。

5. 部署 zabbix PHP 源码

PHP 源码在 zabbix 源码目录下，不清楚的请参考 zabbix 安装，里面有提到。

备注：zabbix 2.5 之后对 php 有严格要求，php 版本必须大于 php5.4

6. 其他问题

6.1 中文语言

《zabbix 汉化方法》

6.2 中文语言不存在

zabbix 开启中文语言 zabbix 没中文语言选项（50）

6.3 中文乱码

zabbix 中文乱码解决方法（13）

7. 总结

总结下 zabbix 的升级方法：备份->重新安装->启动。

第十八章 Zabbix 常见问题解决/FAQ

zabbix Less than 25% free in the configuration cache 解决

在 zabbix server 默认配置下，出现告警：Less than 25% free in the configuration cache，字面意思是：可用的配置缓存少于 25%。报错如下图：

Last 20 issues

Host	Issue	Last change	Age	Info	Ack	Actions
Zabbix server	Less than 25% free in the configuration cache	2015-08-14 16:01:53	2d 17h 22m		No	40

增加 zabbix 配置缓存

修改 zabbix_server.conf 配置文件

找到

```
# CacheSize=8M
```

修改为

```
CacheSize=16M
```

将缓存从 8M 提升到 16M，如果不足你可以调到最高 8G，当然了，能用 8G 那是几乎不可能的。

重启 zabbix server

```
# killall zabbix_server
# /usr/local/zabbix-2.4.4/sbin/zabbix_server
```

或者

```
# service zabbix_server restart
```

等待几分钟，即可恢复~

login as guest zabbix 无法进入登陆界面

很早之前有一位群友告知 zabbix 登陆不了，一直有如下提示：Access denied.Your are logged in as guest. You have no permissions to access this page.If you think this message is wrong, please consult your administrators about getting the necessary permissions.个案没多想，后来又有一位同学有这个问题，决定找出问题，很高兴群友信得过把机器开给我排查，最后找到问题，并解决问题了。

logged in as guest 现象

正常登陆 zabbix 应该是下图：

但是，却出现下图：

解决方法

如果你是 Apache 用户，应该不存在此问题，如果是 nginx 有可能因为配置不当而出现，我们需要修改 php.ini

```
# vim /usr/local/php-5.5.7/etc/php.ini
```

```
cgi.fix_pathinfo=0
```

改为

```
cgi.fix_pathinfo=1
```

问题根源

在 zabbix php 定位问题发现，zabbix 是依据 php 文件名来设置权限，例如 index.php、dashboard.php 这些，如果 cgi.fix_pathinfo 设置为 0，那么 php 无法获取到 url 请求的文件名称，例如访问 index.php，获取不到 index.php 也定位到权限，于是出现这个错误。

Received empty response from Zabbix Agent 问题解决

刚接触 zabbix 新手少部分会出现如下错误：

```
Received empty response from Zabbix Agent at [192.168.1.2]. Assuming that agent dropped connection because of access permission
```

大概意思是说没有权限访问 agent 端口 10050，解决方法如下：

```
# cat zabbix_agentd.conf| grep Server=
Server=192.168.1.2    # zabbix server ip 地址
```

如果你的 server 有多个 IP 地址，使用逗号分隔多个 IP 地址。

zabbix key 总是 not supported 的解决方法

zabbix 定义好 key 之后，总是会出现 Not supported，看到这个问题，大家不用着急，问题其实很容易解决，首先鼠标点击当前 key 的大红叉上，会显示出报错内容。常见的有：

- zabbix_server 取不到值，或者取到空值，在 server 上使用命令 zabbix_get 获取当前 key
- 取到的值和 key 的类型不一样，例如我定义的是 float，但是取到的是字符串，那肯定不会。
- 脚本执行超市，默认情况下 zabbix3 秒就超时，所以要确认下脚本到底要执行多久

这些都是一些常见的问题，但是有一个很奇怪的问题

zabbix_get 能获取到值，但是 item 依旧为 Not Supported。如果你的值类型设置没错的话，那有如下解决方法：

- 等 10 分钟，zabbix 会去重新 check 一次当前 item 的 Supported 状态。
- 删掉当前 item，重新创建
- 修改 zabbix 重新 check 的时间，例如改成 10 分钟，点击 administration--->General--->右侧下拉条选择 "other"--->Refresh unsupported items (in sec)改为 60（单位为秒）---->update。如下图：

问题得到解决！

附录：配置文件详解

zabbix_server.conf 配置文件详解

在 TTLSA 学习 zabbix 的同学们，来看看 zabbix server 配置文件参数详细讲解吧。有助于你更了解 zabbix。直接往下看。

- AlertScriptsPath

默认值：/usr/local/share/zabbix/alertscripts

说明：告警脚本目录

- AllowRoot

默认值：0 说明：是否允许使用 root 启动，0:不允许，1:允许，默认情况下她会使用 zabbix 用户来启动 zabbix 进程，不推荐使用 root

- CacheSize

取值范围： 128K-8G

默认值：8M

说明：配置缓存，用于存储 host，item，trigger 数据，2.2.3 版本之前最大支持 2G，目前最大支持 8G，一般用不了多少的。

- CacheUpdateFrequency

取值范围：1-3600

默认值：60

说明：多少秒更新一次配置缓存

- DBHost

默认值：localhost

说明：数据库主机地址

- DBName

默认值：无

必填：是

- DBPassword：

默认值：孔

说明：数据库密码

- DBPort

取值范围：1024-65535

默认值:3306

说明：SQLite 作为 DB，这个选项请忽略，如果使用 socket 链接，也请忽略。

- DBSchema

说明：Schema 名称. 用于 IBM DB2 、 PostgreSQL.

- DBSocket

默认值：/tmp/mysql.sock

说明：mysql sock 文件路径

- DebugLevel

取值范围：0-5

默认值：3

说明: 指定 debug 级别

0 - 基本信息

1 - critical 信息

2 - error 信息

3 - warnings 信息

4 - 调试日志，日志内容很多，慎重使用

5 - 用于调试 web 和 vmware 监控

- ExternalScripts

默认值： /usr/local/share/zabbix/externalscripts

说明： 外部脚本目录

- Fping6Location

默认值：/usr/sbin/fping6

说明：fping6 路径，不懂 fping 的人可以百度一下，如果 zabbix 非 root 启动，请给 fping6 SUID

- FpingLocation

默认值：/usr/sbin/fping

说明:和上面的一样

- HistoryCacheSize

取值范围：128K-2G

默认值：8M

说明：

历史记录缓存大小，用于存储历史记录

- HistoryTextCacheSize

取值范围：128K-2G

默认值：16M

说明：文本类型历史记录的缓存大小，存储 character, text 、log 历史记录.

- HousekeepingFrequency

取值范围：0-24

默认值：1

说明：housekeep 执行频率，默认每小时回去删除一些过期数据。如果 server 重启，那么 30 分钟之后才执行一次，接下来，每隔一小时在执行一次。

- Include

说明：include 配置文件，可以使用正则表达式，例如：/usr/local/zabbix-2.4.4/conf/ttlsa.com/*.conf

- JavaGateway

说明：Zabbix Java gateway 的主机名，需要启动 Java pollers

- JavaGatewayPort

取值范围：1024-32767

默认值：10052

Zabbix Java gateway 监听端口

- ListenIP

默认值：0.0.0.0

说明：监听地址，留空则会在所有的地址上监听，可以监听多个 IP 地址，ip 之间使用逗号分隔，例如：127.0.0.1,10.10.0.2

- ListenPort

取值范围：1024-32767

默认值：10051

说明：监听端口

- LoadModule

说明：加载模块，格式: LoadModule=，文件必须在指定的 LoadModulePath 目录下，如果需要加载多个模块，那么写多个即可。

- LoadModulePath

模块目录，参考上面

- LogFile

日志文件，例如：/data/logs/zabbix/zabbix-server.log

- LogFileSize

取值范围：0-1024

默认值：1

0 表示禁用日志自动 rotation，如果日志达到了限制，并且 rotation 失败，老日志文件将会被清空掉，重新生成一个

新日志。

- LogSlowQueries

取值范围：0-3600000

默认值：0

多慢的数据库查询将会被记录，单位：毫秒，0 表示不记录慢查询。只有在 DebugLevel=3 时，这个配置才有效。

- MaxHousekeeperDelete

取值范围： 0-1000000

默认值：5000

housekeeping 一次删除的数据不能大于 MaxHousekeeperDelete

- PidFile

默认值：/tmp/zabbix_server.pid

PID 文件

- ProxyConfigFrequency

取值范围：1-604800

默认值：3600

proxy 被动模式下，server 多少秒同步配置文件至 proxy。

- ProxyDataFrequency

取值范围：1-3600

默认值:1

被动模式下，zabbix server 间隔多少秒向 proxy 请求历史数据

- SenderFrequency

取值范围：5-3600

默认值：30

间隔多少秒，再尝试发送为发送的报警

- SNMPTrapperFile

默认值：/tmp/zabbix_traps.tmp

SNMP trap 发送到 server 的数据临时存放文件。

- SourceIP

出口 IP 地址

- SSHKeyLocation

SSH 公钥私钥路径

- SSLCertLocation

SSL 证书目录，用于 web 监控

- SSLKeyLocation

SSL 认证私钥路径、用于 web 监控

- SSLCALocation

SSL 认证,CA 路径，如果为空，将会使用系统默认的 CA

- StartDBSyncers

取值范围：1-100

默认值：4

预先 foke DB Syncers 的数量，1.8.5 以前最大值为 64

- StartDiscoverers

取值范围：0-250

默认值：1

pre-forked discoverers 的数量，1.8.5 版本以前最大可为 255

- StartHTTPPollers

取值范围：0-1000

默认值：1

pre-forked HTTP pollers 的数量，1.8.5 以前最大 255

- StartIPMIPollers

取值范围：0-1000

默认值：0

pre-forked IPMI pollers 的数量，1.8.5 之前，最大为 255

- Timeout

取值范围：1-30

默认值：3

agent，snmp，external check 的超时时间，单位为秒

- TmpDir

默认值：/tmp

- TrapperTimeout

取值范围：1-300

默认值：300

处理 trapper 数据的超时时间

- TrendCacheSize

取值范围：128K-2G

默认值：4M

历史数据缓存大小

- UnavailableDelay

取值范围：1-3600

默认值：60

间隔多少秒再次检测主机是否可用

- UnreachableDelay

取值范围：1-3600

默认值：15

间隔多少秒再次检测主机是否可达。

- UnreachablePeriod

取值范围：1-3600

默认值：45

检测到主机不可用，多久将它置为不可达

- User

默认值：zabbix

启动 zabbix server 的用户，在配置禁止 root 启动，并且当前 shell 用户是 root 得情况下有效。如果当前用户是 ttlsa，那么 zabbix server 的运行用户是 ttlsa

- ValueCacheSize

取值范围：0,128K-64G

默认值：8M

0 表示禁用，history value 缓存大小，当缓存超标了，将会每隔 5 分钟往 server 日志里面记录。养成看日志的好习惯。

zabbix_agentd.conf 配置文件详解

配置文件：zabbix_agentd.conf，不多说，直接看~

- Alias

key 的别名，例如 Alias=ttlsa.userid:vfs.file.regexp[/etc/passwd,^ttlsa:.:([0-9]+),,,,\1]， 或者 ttlsa 的用户 ID。你可以使用 key：vfs.file.regexp[/etc/passwd,^ttlsa:.: ([0-9]+),,,,\1]，也可以使用 ttlsa.userid。

备注: 别名不能重复，但是可以有多个 alias 对应同一个 key。

- AllowRoot

默认值：0

是否允许使用 root 身份运行 zabbix，如果值为 0，并且是在 root 环境下，zabbix 会尝试使用 zabbix 用户运行，如果不存在会告知 zabbix 用户不存在。

0 - 不允许

1 - 允许

- BufferSend

取值范围：1-3600

默认值：5

数据存储在 buffer 中最长多少秒

- BufferSize

取值范围：2-65535

默认值：100

buffer 最大值，如果 buffer 满了，zabbix 将会将检索到的数据发送给 zabbix server 或者 proxy

- DebugLevel

取值范围：0-5

默认值：3

指定日志级别

0 - basic information about starting and stopping of Zabbix processes

1 - critical 级别

2 - error 级别

3 - warnings 级别

4 - debug 级别

5 - extended debugging (与级别 4 一样. 只能使用 runtime control 来设置.)

- EnableRemoteCommands

默认值：0

是否运行 zabbix server 在此服务器上执行远程命令

0 - 禁止

1 - 允许

- HostMetadata

取值范围：0-255 字符

仅用于主机自动注册功能，如果当前值为定义，那么它的值默认为 HostMetadataItem 的值。这个选项在 2.2.0 之后加入，并且确保支付不能超过限制，以及字符串必须是 UTF8，否则服务器无法启动

zabbix 自动注册请参考：zabbix 客户端自动注册（84）

- HostMetadataItem

功能同上，如果 HostMetadata 值未设置，这个配置才有效。支持使用 UserParameters、alias、system.run[]

- Hostname

默认值：HostnameItem 配置的值

主机名，必须唯一，区分大小写。Hostname 必须和 zabbix web 上配置的一直，否则 zabbix 主动监控无法正常工作。为什么呢？因为 agent 拿着这个主机名去问 server，我有配置主动监控项 吗？server 拿着这个主机名去配置里面查询，然后返回信息。

支持字符：数字字母、'.'、' '、 '_'、 '-'，不超过 64 个字符

- HostnameItem

默认值:system.hostname

设置主机名，只有当 HostMetadata 没设置，她才生效。不支持 UserParameters 、aliases，支持 system.run[]

- Include

包含自配置文件，不同的配置写到不同的文件中，然后 include，配置文件会显得规范。例如：/absolute/path/to/config/files/*.conf. Zabbix 2.4.0 开始支持正则表达式。

- ListenIP

默认值：0.0.0.0

监听 IP 地址，默认为所有接口，多个 ip 之间使用逗号分隔

- ListenPort

取值范围：1024-32767

默认值 10050

监听端口

- LoadModule

加载模块文件，可以写多个

格式: LoadModule=

必须配置 LoadModulePath，指定模块目录

zabbix 模块请参考：zabbix 加载扩展模块 第三方库支持（92）

- LoadModulePath

模块路径，绝对路径，如上

- LogFile

日志文件路径

如果未配置，日志会记录到 syslog 中

- LogFileSize

取值范围:0-1024

默认值：1

日志文件大小，单位为 MB。

0 - 关闭自动轮滚.

备注：如果日志文件到达了最大值并且文件轮滚失败，那么老日志文件会被清空掉。

- LogRemoteCommands

默认值：0

记录原型执行的 shell 命令日志，级别为 warrning

0 - disabled

1 - enabled

- MaxLinesPerSecond

取值范围：1-1000

默认值：100

处理监控类型为 log 何 eventlog 日志时，agent 每秒最大发送的行数。默认为 100 行

zabbix 日志监控请参考：zabbix 监控日志文件 MySQL 日志为例（95）

- PidFile

默认值:/tmp/zabbix_agentd.pid

PID 文件名

- RefreshActiveChecks

取值范围：60-3600

默认值：120

多久时间（秒）刷新一次主动监控配置信息，如果刷新失败，那么 60 秒之后会重试一次

- Server

zabbix server 的 ip 地址，多个 ip 使用逗号分隔

- ServerActive

zabbix 主动监控 server 的 ip 地址，使用逗号分隔多 IP，如果注释这个选项，那么当前服务器的主动监控就被禁用了

- SourceIP

zabbix 对外连接的出口 IP 地址

- StartAgents

取值范围：0-100

默认值：3

zabbix 启动之后开启被动监控的进程数量，如果设置为 0，那么 zabbix 被动监控被禁用，并且不会监听相应端口，也就是说 10050 端口不会开启。

- Timeout

默认值：1-30

默认值：3

超时时间

- UnsafeUserParameters

取值范围：0,1

默认值： 0

允许所有字符的参数传递给用户定义的参数。

- User

默认值：zabbix

运行 zabbix 程序的用户，如果 AllowRoot 被禁用，才有效果

- UserParameter

用户自定义 key，格式: UserParameter=,

例如：serParameter=system.test,who|wc -l

更多请看：zabbix 自定义用户 key 与参数 User parameters（24）

后续添加，请关注 TTLSA 官网

http://www.ttlsa.com/