# zabbix 架构

官网：www.zabbix.com

**Zabbix**是一个基于WEB界面的提供分布式系统监视以及网络监视功能的企业级的开源解决方案。能监视各种网络参数，保证服务器系统的安全运营；并提供灵活的通知机制以让系统管理员快速定位/解决存在的各种问题。

zabbix由2部分构成，zabbix server与可选组件zabbix agent。

常见监控工具zabbix Nagios ganglia prometheus falcon

**Zabbix Server：**可以单独监视远程服务器的服务状态；同时也可以与zabbix agent配合，可以轮询zabbix agent主动接收监视数据（agent方式），同时还可被动接收zabbix agent发送的数据（trapping方式）。zabbix server可以通过SNMP，zabbix agent，ping，端口监视等方法提供对远程服务器/网络状态的监视，数据收集等功能，它可以运行在Linux，Solaris，HP-UX，AIX，Free BSD，Open BSD，OS X等诸多平台上。

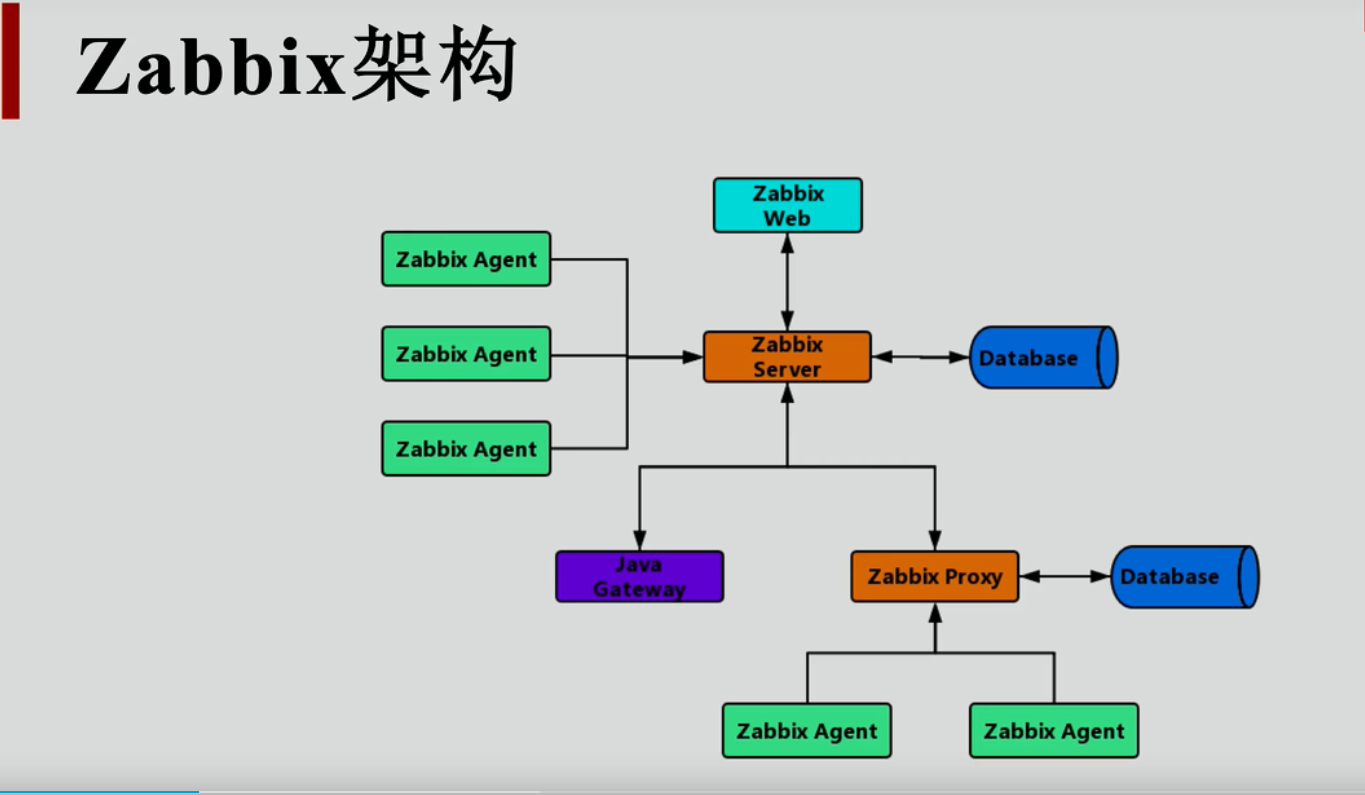
**Zabbix Web：** zabbix前端页面，提供了友好的展示和操作界面，负责数据的展示，监控系统的配置管理，用户权限配置管理等

**Zabbix Databases：** 数据和配置存储数据库，zabbix支持多种数据库，包括mysql,oracle,db2等

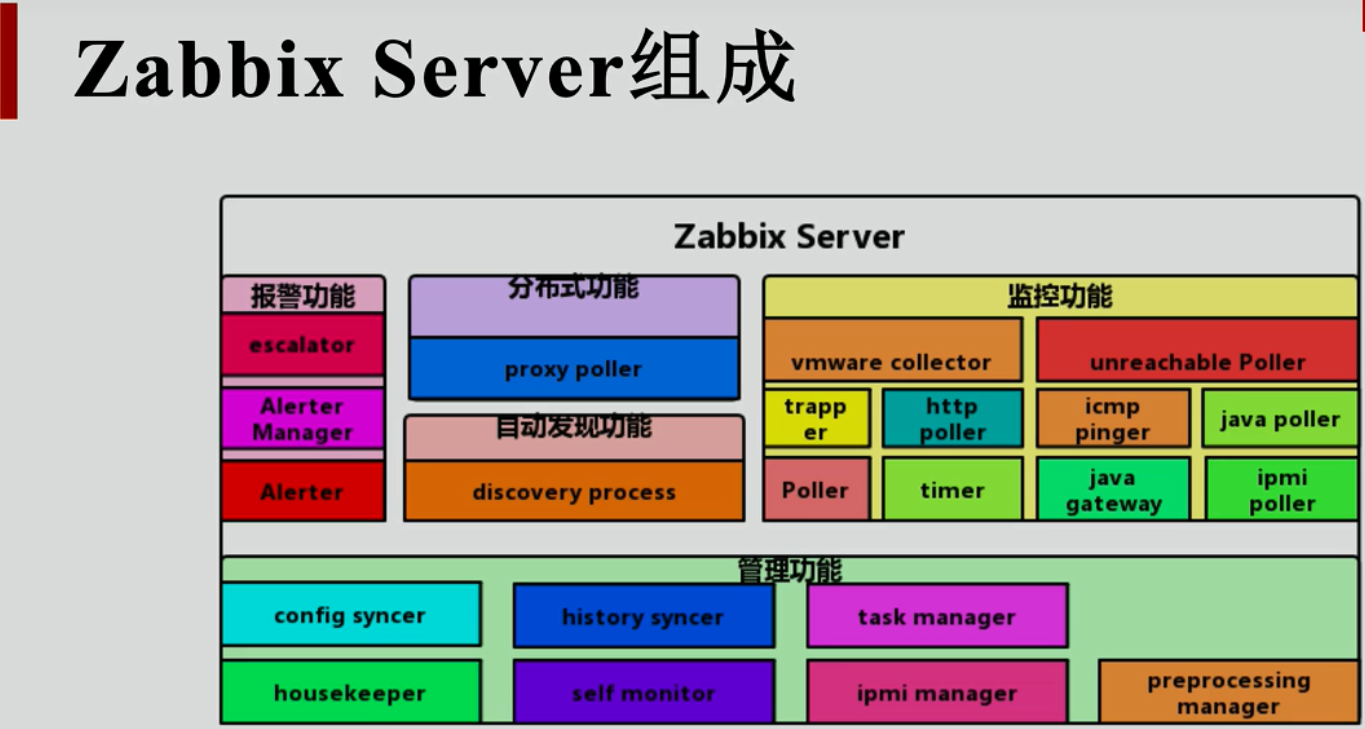
**Java Gateway：** java网关，负责通过JMX监控收集java应用性能数据

**Zabbix Agent：**需要安装在被监视的目标服务器上，它主要完成对硬件信息或与操作系统有关的内存，CPU等信息的收集。zabbix agent可以运行在Linux,Solaris,HP-UX,AIX,Free BSD,Open BSD, OS X, Tru64/OSF1, Windows NT4.0, Windows (2000/2003/XP/Vista)等系统之上。

**Zabbix Proxy：**是一个可以从一个或多个受监控设备收集监控数据，并将信息发送到Zabbix sever的进程，基本上 是代表sever工作的。 所有收集的数据都在本地进行缓存，然后传送到proxy所属的Zabbix sever。设 备在多个地域、机房可通过zabbix proxy来实现分布式。



# Zabbix Server



监控功能进程：数据通过这些进程从客户端收集上来

poller 进程负责server主动拉去类型的监控数据

trapper进程主要负责Agent主动上报的监控数据

http poller进程主要负责URL监控类型的数据收集

icmp pinger进行负责ping存货监控数据收集

java gateway 进程负责和java gateway通信处理数据

java poll 进程负责拉取JMX类型数据获取

ipmi poller 进程负责IPMI类型数据获取

timer 进程负责处理和时间有关的数据以及告警

vmware collector进行负责收集vmware虚拟化环境监测数据

unreachable poller 负责处理无法到达类的数据监控处理

管理功能进程：

preprocessing manager：会对监控数据进行预处理

history syncer：负责将数据分析并保存至数据库中

housekeeper：负责定期清理历史数据

task manager：

ipmi manager：

self monitor：

config syscer：

报警功能进程：

alerter:告警处理发送

alerter manager：管理alerter进程

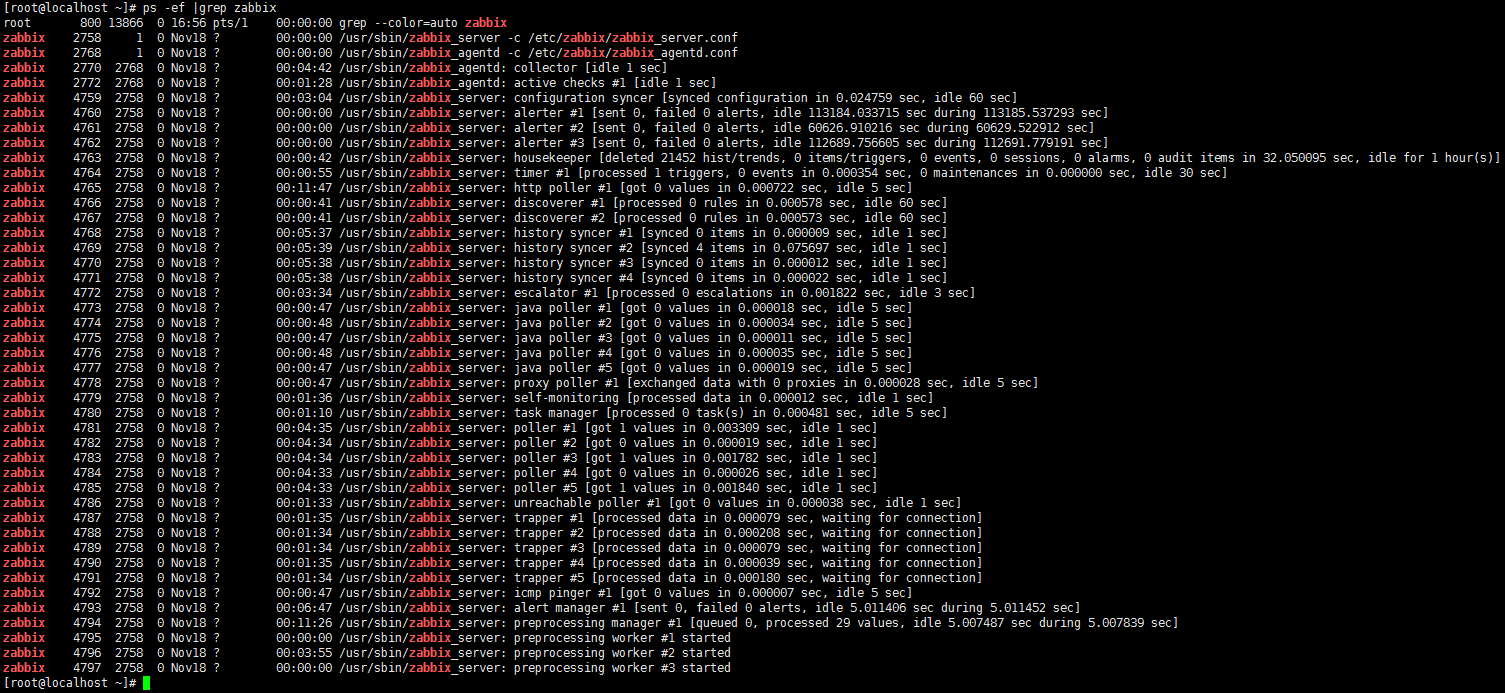
escalator:负责处理告警过程中的各个步骤，比如告警升级之类的

分布式功能：

proxy poller：负责分布式通讯

自动发现功能：

discovery process：负责设备自动发现功能



# zabbix监控方式

zabbix主要Agent，Trapper，SNMP，JMX，IPMI这几种监控方式，

Agent监控方式：zabbix-agent会主动收集本机的监控信息并通过TCP协议与zabbix-server传递信息。Agent监控方式分为主动和被动模式

Trapper监控方式：使用zabbix-sender程序主动向zabbix-server发送数据。key的名称和发送的数据内容都可以灵活定义。发送的信息采用JSON格式，遵循zabbix-sender协议。可以自定义脚本利用zabbix-sender协议来zabbix-server发送信息。

SNMP监控方式： 全称Simple Network Management Protocol，即网络管理协议，包括进程管理和被管理设备两部分。作为一种国际通用的网络管理协议被广泛的应用于各种交换机，路由器等网络设备的管理上，而现在也越来越多被用于对服务器的监控上。

IPMI监控方式：全称Interlligent Platform Management Interface，即智能平台管理接口，原本是Intel架构中企业系统的周边设备所采用的一种工业标准，以后成为业界通用的标准。用户可以利用IPMI监控服务器的物理特性，如温度，电压，电扇工作状态，电源供应以及机箱\*\*\*等指标，也可以管理硬件设备，重启关机、获取硬件日志等等。

JMX监控方式：

zabbix自带的tcp、icmp、ssh、telnet等检测方式，满足不支持以上协议的硬件设备。

zabbix支持自定义监控，通过shell、python、ruby、perl、powershell等可执行脚本收集监控数据

zabbix默认提供URL监控，可监控线上的业务是否正常，异常响应或者没有匹配的关键字及时告警

zabbix 支持8种运算符，29个函数任意组合，完全可满足告警阀值设置需求

zabbix 告警关联，例如：一台服务器上运行多个服务，宕机时，只会发送宕机告警，从而快速定位问题。

zabbix 任意两个组件之间都可以通过TLS加密传输数据，保证异地之间的数据安全

Zabbix主要功能： -应用 - CPU负荷 - 内存使用 - 磁盘使用 - 网络状况 - 端口监视 - 日志监视

Zabbix使用：使用有主机群组、模板、主机、应用集、监控项、数据报表、图形界面，可设置触发条件

zabbix网络自动发现：

网络扫描：通过扫描网段中满足条件的设备，比如安装了agent、snmp，等特定服务，即可设置加入监控，并应相

应监控模板

agent自动注册：安装了agent的设备，并配置为主动上报模式，zabbix服务端根据客户端传过来的信息执行监控

# zabbix术语

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **组件功能** |  | | | |
| frontend | 前端 |  | zabbix api | zabbix接口 |
| zabbix server | zabbix服务端 |  | zabbix agent | zabbix客户端 |
| zabbix proxy | zabbix代理 |  | network discovery | 网络自动发现 |
| Java Gateway | java网关 |  |  |  |
| Zabbix Screen | zabbix屏幕 |  | Dashboard | 仪表板 |
| active agent auto-registration | 主动agent自动注册 |  | low-level discovery | 低级自动发现 |
|  |  |  |  |  |
| **监控收集** |  | | | |
| host | 主机 |  | host group | 主机组 |
| item | 监控项 |  | value preprocessing | 预处理 |
| template | 模板 |  | application | 应用 |
| web scenario | web场景 |  | macros | 宏（变量） |
| **数据展示** |  | | | |
| graph | 图表 |  | screen | 聚合图表 |
| maps | 拓扑图 |  | slide shows | 幻灯片演示 |
| latest data | 最新数据 |  |  |  |
| **告警相关** |  | | | |
| Trigger | 触发器 |  | event | 事件 |
| problem | 异常状态 |  | action | 操作 |
| escalation | 升级 |  | media | 媒介 |
| notification | 通知 |  | remote command | 远程命令 |
| maintenance | 维护模式 |  |  |  |
| **认证和权限** |  | | | |
| user | 用户 |  | user group | 用户组 |
| permission | 权限 |  | user type | 用户类型 |
| **其他** |  | | | |
| active | 主动 |  | passive | 被动 |
| Standar Edition | 标准版 |  | Enterprise Edition | 企业版 |
| prototypes | 原型 |  |  |  |

# zabbix 安装

1、二进制安装 2、源码安装 3、docker安装

## zabbix二进制安装

环境：centos 7.6最小化安装

centos 7 添加阿里云镜像加速常用软件的安装，

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo

yum install epel-release #安装epel源：包含了一些基础镜像中没有的常用软件

安装常用的开发组件

yum groups install "Development Tools"

yum groups info "Development Tools" #查看Development Tools工具包

增加Zabbix镜像源

rpm -ivh https://repo.zabbix.com/zabbix/4.0/rhel/7/x86\_64/zabbix-release-4.0-1.el7.noarch.rpm

vim /etc/selinux/config #关闭SElinux SELINUX=disabled

systemctl stop firewalld.service

systemctl disable firewalld.service

yum install iptables-services

安装Zabbix Server和Frontend

yum install zabbix-server-mysql #yum search zabbix可搜索zabbix相关包名

yum install zabbix-web-mysql

安装MySQL

yum install mariadb-server

systemctl start mariadb.service

systemctl status mariadb.service

mysql\_secure\_installation #初始化

创建数据库

mysql -uroot -p

create database zabbix character set utf8 collate utf8\_bin;

grant all privileges on zabbix.\* to zabbix@localhost identified by 'zabbix@findsec123'; #创建用户设置密码

flush privileges

quit

导入数据结构

zcat /usr/share/doc/zabbix-server-mysql\*/create.sql.gz | mysql -uzabbix -pzabbix@findsec123 zabbix

配置Zabbix Serve：配置数据库连接等相关信息

vim /etc/zabbix/zabbix\_server.conf

DBHost=192.168.10.241

DBName=zabbix

DBUser=zabbix

DBPassword=zabbix@findsec123 #上面创建数据库时创建的数据库用户和密码

启动Zabbix Server

systemctl start zabbix-server.service

systemctl status zabbix-server.service

more /var/log/zabbix/zabbix\_server.log #查看日志

配置Zabbix frontend

vim /etc/php.ini #修改PHP参数

max\_execution\_time = 300 #执行时间，默认30秒，改为300秒

memory\_limit = 128M #内存限制

post\_max\_size = 16M #至少16M

upload\_max\_filesize = 2M #上传文件大小

max\_input\_time = 300 #

max\_input\_vars = 10000 #至少10000

always\_populate\_raw\_post\_data = -1 #修改

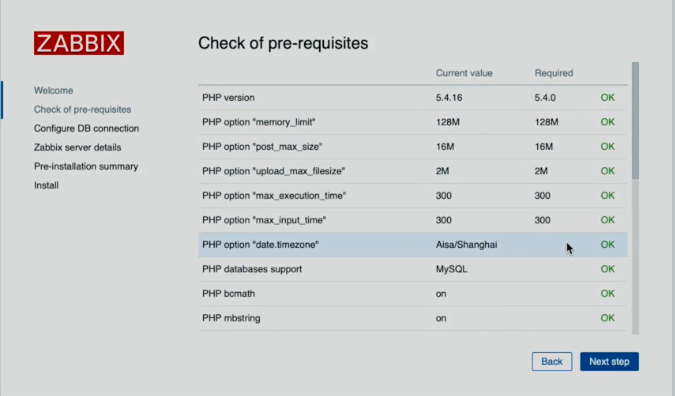
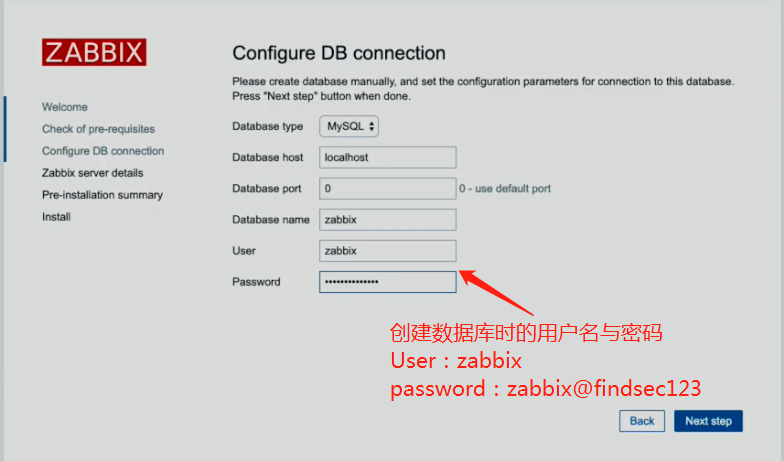
date.timezone = Asia/Shanghai #修改时区

启动httpd

systemctl start httpd.service

systemctl status httpd.service

http://10.211.55.7/zabbix/setup.php



默认登录名：Admin 密码：zabbix

安装Zabbix Agent

yum install zabbix-agent

vim /etc/zabbix/zabbix\_agentd.conf

ServerActive=192.168.10.243 #主动监控以实际案例为例

Hostname=Centos7\_243agent #此主机名将是服务端添加监控的主机名

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

systemctl start zabbix-agent.service

-----------扩展：实际中server1中agent的配置-----------------------------------

vim /etc/zabbix/zabbix\_agentd.conf

ServerActive=192.168.10.243

Hostname=Server1\_agent #此主机名将是服务端添加监控的主机名

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

## zabbix源码安装

环境：centos 7.6最小化安装

下载zabbix server源码包

wget -O zabbix-4.0.4.tar.gz https://sourceforge.net/projects/zabbix/files/ZABBIX%20Latest%20Stable/4.0.4/zabbix-4.0.4.tar.gz/download

安装依赖

yum install wget telnet net-tools python-paramiko gcc gcc-c++ dejavu-sans-fonts python-setuptools python-devel sendmail mailx net-snmp net-snmp-devel net-snmp-utils freetype-devel libpng-devel perl unbound libtasn1-devel p11-kit-devel OpenIPMI unixODBC libevent-devel mysql-devel libxml2-devel libssh2-devel OpenIPMI-devel java-1.8.0-openjdk-devel openldap-devel curl-devel unixODBC-devel

解压并编译安装

tar -zxvf zabbix-4.0.4.tar.gz

cd zabbix-4.0.4/

建立编译安装目录

mkdir -p /data/findsec/zabbix

./config --helpe #可查看安装时编辑参数选择帮助

./configure --prefix=/data/findsec/zabbix --enable-server --enable-agent --enable-java --with-mysql --with-libxml2 --with-unixodbc --with-net-snmp --with-ssh2 --with-openipmi --with-ldap --with-libcurl --with-iconv

make

make install

/data/findsec/zabbix/sbin/zabbix\_server -V #安装完成后验证

---------编译详解：-----------------------------------------------------

--enable-server(启用server) --enable-agent(启用agent) --enable-java(启用java监控) --with-mysql(启用mysql,用mysql作为存储) --with-libxml2() --with-unixodbc(数据库连接方式) --with-net-snmp(网络相关监控) --with-ssh2(执行远程命令用到) --with-openipmi(硬件相关的监控) --with-ldap(ldap认证) --with-libcurl(web监控相关) --with-iconv()

#编译完成后可以看到编译详情，启用了哪些功能选项，如启用分布式需要--enable-proxy

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

数据库安装

yum install mariadb-server

systemctl start mariadb.service

systemctl status mariadb.service

mysql\_secure\_installation #初始化数据库

创建数据库

mysql -uroot -p

create database zabbix character set utf8 collate utf8\_bin;

grant all privileges on zabbix.\* to zabbix@localhost identified by 'zabbix@findsec123';

导入数据结构

mysql -uzabbix -pzabbix@findsec123 zabbix < /root/zabbix-4.0.4/database/mysql/schema.sql

mysql -uzabbix -pzabbix@findsec123 zabbix < /root/zabbix-4.0.4/database/mysql/images.sql

mysql -uzabbix -pzabbix@findsec123 zabbix < /root/zabbix-4.0.4/database/mysql/data.sql

mysql -uroot -p #进入数据库

show databases;

use zabbix;

show tables; #进入数据库查看zabbix 库，导入成功将有144张表

修改zabbix server配置

vim /data/findsec/zabbix/etc/zabbix\_server.conf

DBHost=192.168.10.241

DBName=zabbix

DBUser=zabbix

DBPassword=zabbix@findsec123 #上面创建数据库时创建的数据库用户和密码

增加用户

groupadd --system zabbix #创建一个系统组

useradd --system -g zabbix -d /usr/lib/zabbix -s /sbin/nologin -c "Zabbix Monitoring System" zabbix #-c为简短描述

启动服务

/data/findsec/zabbix/sbin/zabbix\_server -h #可查看用法

/data/findsec/zabbix/sbin/zabbix\_server -c /data/findsec/zabbix/etc/zabbix\_server.conf

tailf /tmp/zabbix\_server.log #查看日志，有警告和错误是因为启用了agent,但是未启动

配置启用server本机的agent

vim /data/findsec/zabbix/etc/zabbix\_agent.conf #配置agent

Server=127.0.0.1 #本机上的server 被动监控

ServerActive=127.0.0.1 #本机上的server 主动监控

Hostname=Zabbix server #日志中能看到,与其一致即可，用来在web端添加主机

/data/findsec/zabbix/sbin/zabbix\_agent -h #可查看用法

/data/findsec/zabbix/sbin/zabbix\_agernt -c /data/findsec/zabbix/etc/zabbix\_agent.conf #启动agent

安装zabbix web 前端

增加Zabbix镜像源

rpm -ivh https://repo.zabbix.com/zabbix/4.0/rhel/7/x86\_64/zabbix-release-4.0-1.el7.noarch.rpm

安装Zabbix Frontend

yum install zabbix-web-mysql

配置Zabbix frontend

vim /etc/php.ini

max\_execution\_time = 300 #执行时间，默认30秒，改为300秒

memory\_limit = 128M #内存限制

post\_max\_size = 16M #至少16M

upload\_max\_filesize = 2M #上传文件大小

max\_input\_time = 300 #

max\_input\_vars = 10000 #至少10000

always\_populate\_raw\_post\_data = -1 #修改

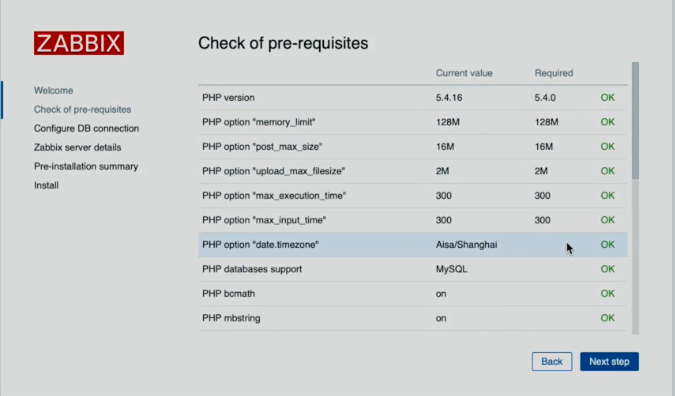
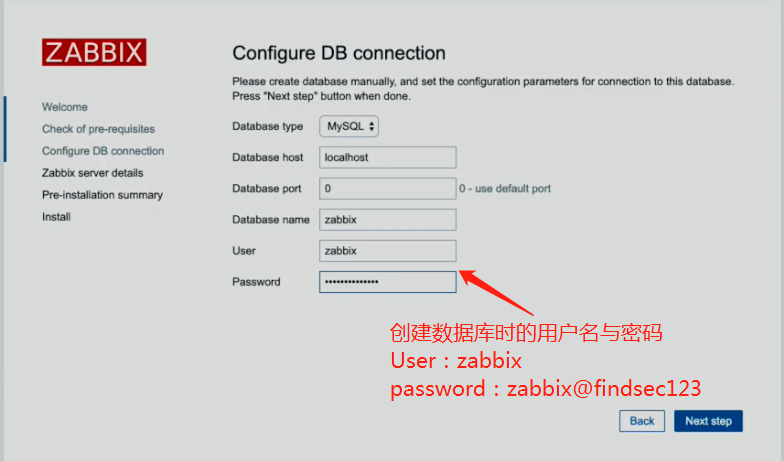
date.timezone = Asia/Shanghai #修改时区

启动httpd

systemctl start httpd.service

systemctl status httpd.service

http://10.211.55.7/zabbix/setup.php



默认登录名：Admin 密码：zabbix #正常的话即可看到Zabbix server主机的数据

安装Zabbix Agent

yum install zabbix-agent

vim /etc/zabbix

systemctl start zabbix-agent.service

扩展：

安装Zabbix依赖包

[root@zabbixserver lnmp\_soft]#yum -y install net-snmp-devel curl-devel gcc

[root@zabbixserver lnmp\_soft]# yum -y install libevent-devel-2.0.21-4.el7.x86\_64.rpm

#注意libevent-devel这个软件包在lnmp\_soft目录下有提供

7安装Zabbix

[root@zabbixserver ~]# tar -xf zabbix-3.4.4.tar.gz

[root@zabbixserver ~]# cd zabbix-3.4.4

[root@zabbixserver zabbix-3.4.4]# ./configure --prefix=/usr/local/zabbix --enable-server --enable-proxy \

> --enable-agent --with-mysql=/usr/bin/mysql\_config --with-net-snmp --with-libcurl

....

....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Now run 'make install' \* #提示直接 'make install'

[root@zabbixserver zabbix-3.4.4]# make install

[root@zabbixserver ~]# ls /usr/local/zabbix/ #安装成功后查看安装的目录里文件

bin etc lib sbin share

## zabbix docker安装

yum install docker-latest #安装docker最新版

docker -h

systemctl start docker-latest

systemctl status docker-latest

docker run -d -p 80:80 httpd #启动镜像测试，映射80端口，本地没有将会拉取镜像

docker ps

docker stop 容器ID #停止容器

**docker run命令解析：**

# --name="mysql-server": 为容器指定一个名称”mysql-server”；

# -t: 为容器重新分配一个伪输入终端，通常与 -i 同时使用；

# -d: 后台运行容器，并返回容器ID；

# -p: 指定端口映射，格式为：主机(宿主)端口:容器端口 -P(大写)为随机映射端口)

# -e MYSQL\_DATABASE=”zabbix”: 设置环境变量；

# --link=[]: 添加链接到另一个容器；

启动mysql-server 容器

docker run --name mysql-server -t -e MYSQL\_DATABASE=”zabbix” -e MYSQL\_USER=”zabbix” -e MYSQL\_PASSWORD=”zabbix@findsec123” -e MYSQL\_ROOT\_PASSWORD=”zabbix@findsec123” -d mysql:5.7 --character-set-server=utf8 --collation-server=utf8\_bin

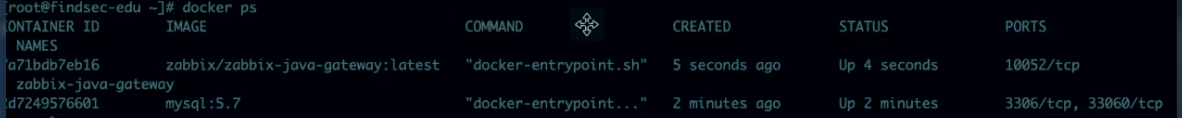
docker ps



启动zabbix-java-gateway 容器

docker run --name zabbix-java-gateway -t -d zabbix/zabbix-java-gateway:latest

docker ps

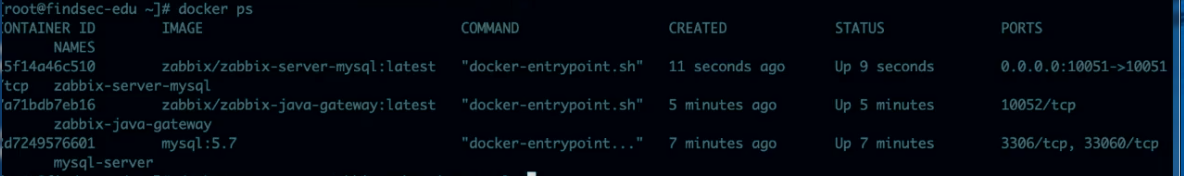


docker logs 7a71bdb7eb16 #查看日志，id为前面启动的java-gateway容器id

启动zabbix-server-mysql 容器

docker run --name zabbix-server-mysql -t -e DB\_SERVER\_HOST=”mysql\_server” -e MYSQL\_DATABASE=”zabbix” -e MYSQL\_USER=”zabbix” -e MYSQL\_PASSWORD=”zabbix@findsec123” -e MYSQL\_ROOT\_PASSWORD=”zabbix@findsec123” -e ZBX\_JAVAGATEWAY=”zabbix-java-gateway” --link mysql-server:mysql --link zabbix-java-gateway:zabbix-java-gateway -p 10051:10051 -d zabbix/zabbix-server-mysql:latest

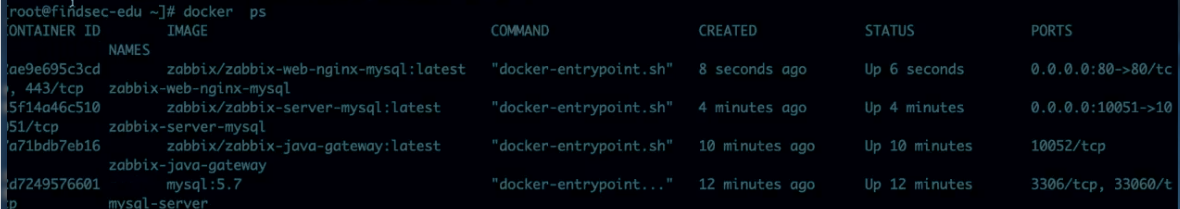
docker ps



启动zabbix-web-nginx-mysql 容器(前端)

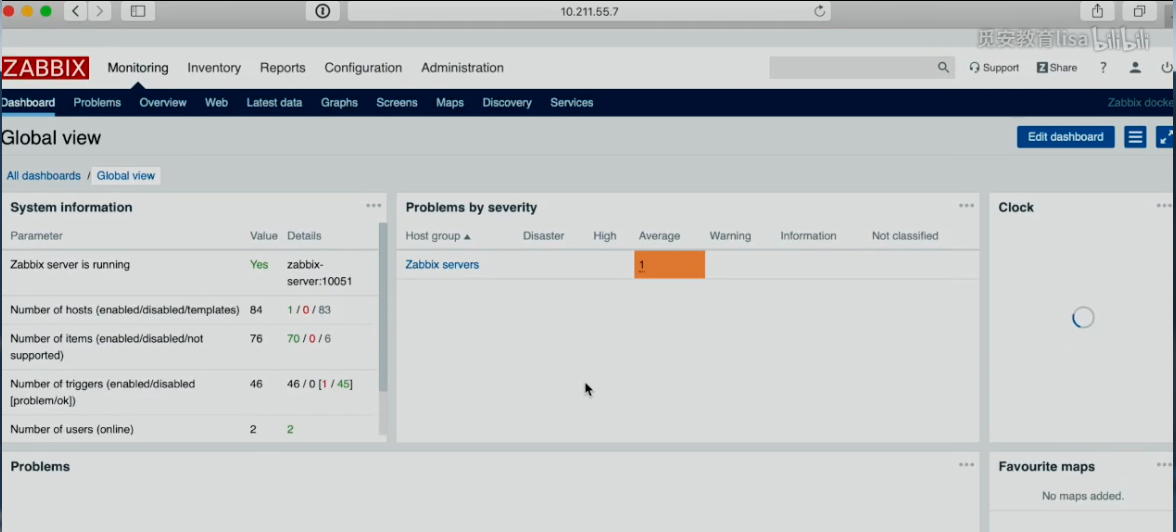
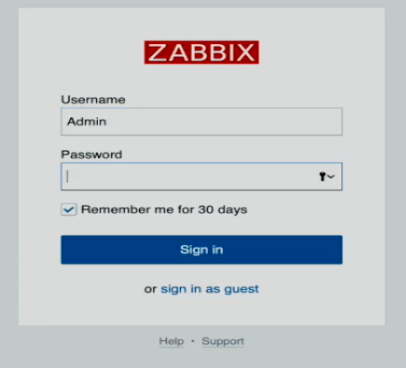
docker run --name zabbix-web-nginx-mysql -t -e DB\_SERVER\_HOST=”mysql-server” -e MYSQL\_DATABASE=”zabbix” -e MYSQL\_USER=”zabbix” -e MYSQL\_PASSWORD=”zabbix@findsec123” -e MYSQL\_ROOT\_PASSWORD=”zabbix@findsec123” --link mysql-server:mysql --link zabbix-server-mysql:zabbix-server -p 80:80 -d zabbix/zabbix-web-nginx-mysql:latest

docker ps



docker logs ID

访问：zabbix-server 主机80端口，



启动容器化zabbix-agent

docker run --name zabbix-agent -e ZBX\_HOSTNAME=”Zabbix server” -e ZBX\_SERVER\_HOST=”zabbix-server-mysql” --link zabbix-server-mysql:zabbix-server -d zabbix/zabbix-agent:latest

docker ps



docker logs ID

# zabbix前端功能

# linux agent安装

环境：centos 7.6最小化安装

centos 7 添加阿里云镜像加速常用软件的安装，

yum install wget

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo

yum install epel-release #安装epel源：包含了一些基础镜像中没有的常用软件

yum install bash-completion bash-completion-extras #安装命令自动补齐

yum insatall vim

yum install tree

增加Zabbix镜像源

rpm -ivh https://repo.zabbix.com/zabbix/4.0/rhel/7/x86\_64/zabbix-release-4.0-1.el7.noarch.rpm

vim /etc/selinux #关闭SELINUX=disabled

getenforce

systemctl stop firewall.service #关闭防火墙

systemctl disable firewall.service #设置开机不自启

yum search zabbix-agent #查询下zabbix包名

yum install zabbix-agent

vim /etc/zabbix/zabbix\_agentd.conf

Server=192.168.10.243 #被动监控，zabbix-server地址

ServerActive=192.168.10.243 #主动监控，zabbix-server地址

Hostname=Zabbix server #用来在web端添加主机

systemctl start zabbix-agent.service

systemctl enable zabbix-agent.service

systemctl status zabbix-agent.service

tailf /var/log/zabbix/zabbix\_agentd.log

配置web ，添加主机，放入主机组，关联对应监控模板，

确认监控是否正常：

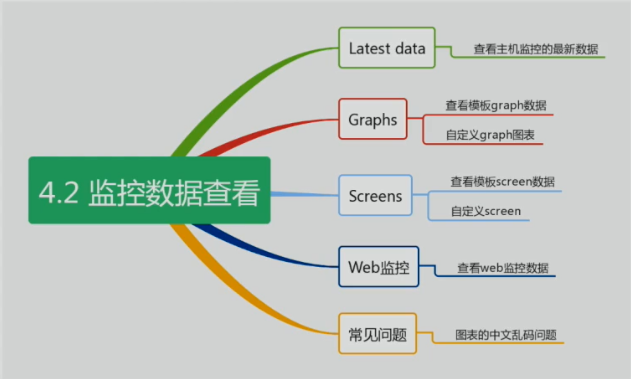
server 主机上telenet agent主机的IP+10050端口

agent主机上tailf /vat/log/zabbix/zabbix\_agentd.log日志

web端查看主机status（状态）是否正常

新建模板：设置模板名，分配所属分组。进入模板，创建application（应用程序），应用程序中创建监控项。

# 监控数据查看



latest data：最新数据 graphs:图表 screens:聚合图表 maps:拓扑图 slide shows：幻灯片演示

graphs:在模板里面创建新的graphs，也可以在主机中创建graphs

screens:在模板里面创建新的screens，也可以在主机中创建screens

web监控：

## zabbix中文字体乱码

windows 字体所在目录：C:\Windows\Fonts

[root@localhost ~]# ls /usr/share/zabbix/fonts/ #将字体文件放入此目录

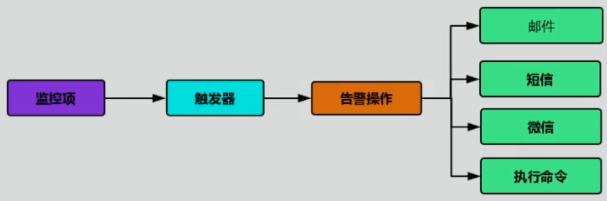
graphfont.ttf SIMHEI.ttf #一定是.ttf文件

[root@localhost ~]# cd /usr/share/zabbix/ #有的版本可能是在zabbxi/assets/fonts

[root@localhost zabbix]# vim include/defines.inc.php #修改zabbix web前端代码配置

54 define('ZBX\_GRAPH\_FONT\_NAME', 'SIMHEI'); // font file name #修改字体文件名，不要后缀

# 监控阀值设置



增加trigger

触发告警信息

trigger未触发

trigger触发频繁

**Zabbix触发器Trigger的作用：**是对收集的Item(监控项)数据做进一步判断，然后产生一个状态值，该状态值只有两个不同的取值”OK”或”Problem”（该状态值可以作为Action的输入，Action根据不同的状态组合条件作出不同的动作如发送报警邮件、重启服务等）。一个Trigger可以对应一个Item，当然了一个Trigger也可能对应了多个Item，配置Trigger的关键就是写好Trigger expression。

1、Trigger基本的表示式是这样的

{ Template OS Linux:vfs.file.cksum[/etc/passwd] . diff( 0 ) }>0

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

各个字段解释如下：

<server> 主机名(或者模板名，最终是显示的主机名，说明一个trigger最终一定是挂在一台Host上面的)

<key> item配置的时候可以选则已存在的item

<function> 函数如last，avg等

<parameter> 函数的参数

<operator> 判断符（简单的运算符）

<constant> 常量

## 函数介绍：

**abschange** 参数：忽略 支持类型：float,int,str,text,log

作用：返回最近获得的值与之前获得值差的绝对值，对于字符串类型：0表示相等，1表示不同

**avg** 参数：秒或#num 支持类型：float,int

作用：返回一段时间的平均值

举例：

avg(5):最后5秒的平均值

avg(#5）：表示最近5次得到值的平均值

avg(3600,86400）：表示一天前的一个小时的平均值

如果仅有一个参数，表示指定时间的平均值，从现在开始算起，如果有第二个参数，表示漂移，从第二个参数前开始算时间，

#n表示最近n次的值

示例：

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.

**max** 参数:秒或#num 支持值类型:float,int

描述:返回指定时间间隔的最大值.时间间隔作为第一个参数可以是秒或收集值的数目(前缀为#).从Zabbix1.8.2开始，函数支持第二个可选参数time\_shift，可以查看avg()函数获取它的使用方法.

示例：

触发器名称：Free disk space is too low Problem: 最近5分钟内剩余磁盘空间小于10GB

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

**min** 参数:秒或#num 支持值类型:float,int

描述:返回指定时间间隔的最小值.时间间隔作为第一个参数可以是秒或收集值的数目(前缀为#).从Zabbix1.8.2开始，函数支持第二个可选参数time\_shift，可以查看avg()函数获取它的使用方法.

示例：

触发器名称：Free disk space is too low Problem:最近10分钟内磁盘空间大于40GB

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

**nodata** 参数:秒 支持值类型:any

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

示例：

触发器名称：No heartbeats within last 3 minutes 使用函数 nodata():

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

**change** 参数：忽略 支持类型：float,int,str,text,log

作用：返回最近获得值与之前获得值的差值，对于字符串0表示相等，1表示不同

举例：

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

**count** 参数：秒或#num 支持类型：float,int,str,text,log

作用：返回指定时间间隔内数值的统计，

举例：

count(600)最近10分钟得到值的个数

count(600,12)最近10分钟得到值的个数等于12

count(600,12,"gt")最近10分钟得到值的个数大于12

count(#10,12,"gt")最近10个值中，值大于12的个数

count(600,12,"gt",86400)24小时之前的10分钟内值大于12的个数

count(600,6/7,"band")-thenumberofvaluesforlast10minuteshaving'110'(inbinary)inthe3leastsignificantbits.

count(600,,,86400)24小时之前的10分钟数据值的个数

第一个参数：指定时间段

第二个参数：样本数据

第三个参数：操作参数

第四个参数：漂移参数

#支持的操作类型

eq: 相等 ne: 不相等 gt: 大于 ge: 大于等于 lt: 小于 le: 小于等于 like: 内容匹配

**date** 参数：忽略 支持类型：any

作用：返回当前的时间，格式YYYYMMDD

**dayofmonth**

返回当前是本月的第几天

**dayofweek**

返回当前是本周的第几天

**delta** 参数：秒或#num 支持类型：float，int

作用：返回时间间隔内的最大值与最小值的差值

**diff** 参数:忽略 支持值类型:float,int,str,text,log

作用:返回值为1表示最近的值与之前的值不同，0为其他情况

示例：

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

**fuzzytime** 参数:秒 支持值类型:float,int

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

示例：

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”

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

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

例子:

last(0)等价于last(#1)last(#3)表示最近\*\*第\*\*3个值(并不是最近的三个值)

本函数也支持第二个参数\*\*time\_shift\*\*，例如

last(0,86400)返回一天前的最近的值

如果在history中同一秒中有多个值存在，Zabbix不保证值的精确顺序

#num从Zabbix1.6.2起开始支持,timeshift从1.8.2其开始支持,可以查询avg()函数获取它的使用方法

logeventid 参数:string 支持值类型:log

描述:检查最近的日志条目的EventID是否匹配正则表达式.参数为正则表达式,POSIX扩展样式.当返回值为0时表示不匹配，1表示匹配。该函数从Zabbix1.8.5起开始支持.

**logseverity** 参数:忽略 支持值类型:log

描述:返回最近日志条目的日志等级(logseverity).当返回值为0时表示默认等级，N为具体对应等级(整数，常用于Windowseventlogs).Zabbix日志等级来源于Windowseventlog的Information列.

**logsource** 参数:string 支持值类型:log

描述:检查最近的日志条目是否匹配参数的日志来源.当返回值为0时表示不匹配，1表示匹配。通场用于Windowseventlogs监控.例如logsource["VMWareServer"]

**now** 参数:忽略 支持值类型:any

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

**prev** 参数:忽略 支持值类型:float,int,str,text,log

描述:返回之前的值，类似于last(#2)

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

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

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

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

示例：

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. strlen 参数:秒或#num 支持值类型:str,log,text 描述:指定最近值的字符串长度(并非字节),参数值类似于last函数.例如strlen(0)等价于strlen(#1),strlen(#3)表示最近的第三个值,strlen(0,86400)表示一天前的最近的值.该函数从Zabbix1.8.4起开始支持 sum 参数:秒或#num 支持值类型:float,int 描述:返回指定时间间隔中收集到的值的总和.时间间隔作为第一个参数支持秒或收集值的数目(以#开始).从Zabbix1.8.2开始，本函数支持time\_shift作为第二个参数。可以查看avg函数获取它的用法 time 参数:忽略 支持值类型:any 描述:返回当前时间，格式为HHMMSS，例如123055

**函数案例：**

{test:key.last(0)}<>1 # 最新返回值不等于1就触发

{test:key.last(0)}=1 # 最新返回值等于1就触发

{test:key.avg(3m)}>80 # 连续三分钟平均值在80%就触发

{test:key.max(3m)}>80 # 连续三分钟最大值在80%以上就触发

**内置触发器案例：**

# 如果agent客户端断开连接3分钟就返回1就出发报警

# 注：agent ，需要先建立监控项，可在模板中查看。

触发器：{test:agent.ping.nodata(3m)}=1

# 如果CPU使用率1分钟平均值超80%就报警

# 监控部署：名称：CPU $2 time、键值：system.cpu.util[,idle]、信息类型：浮点数、单位：%

触发器：{test:system.cpu.util[,idle].avg(3m)}<20

# 用户及告警媒介

# 宏（macros）

Zabbix支持许多在多种情况下使用的宏。宏是一个变量，由如下特殊语法标识：MACRO有效地使用宏可以节省时间，并使Zabbix变地更加高效。

例如：在一个的典型用途中，宏可以用于模板中。因此，模板的触发器可能命名为“{HOST.NAME}的mysql{#MYSQLPORT} 端口已宕掉，请注意查看！”。当这个模板应用在主机（飞龙在天roledb-192.168.1.100）时，并且当触发器展示在监控页面上时，触发器的名称将解析为“飞龙在天roledb-192.168.1.100的mysql3306 端口已宕掉，请注意查看！”。

例如：在一个的典型用途中，宏可以用于模板中。因此，模板的触发器可能命名为Processor load is too high on {HOST.NAME}。当这个模板应用于主机（如 Zabbix Server）时，并且当触发器展示在监控页面上时，触发器的名称将解析为Processor load is too high on Zabbix server。

**宏分类：**

系统宏变量{变量}： {macros} 例如：{HOST.NAME}主机名

用户自定义宏变量{$变量}： {$macros} 例如：

低级别自动发现宏变量{#变量}：{#macros} 例如

示例一：主机SSH服务

net.tcp.service[ssh,{$SSH\_PORT}]

如果你监控的ssh端口为8888，那么你只需要定义host级别宏变量SSH\_PORT为指定端口号8888，那么调用 {$SSH\_PORT}就是调用8888

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。

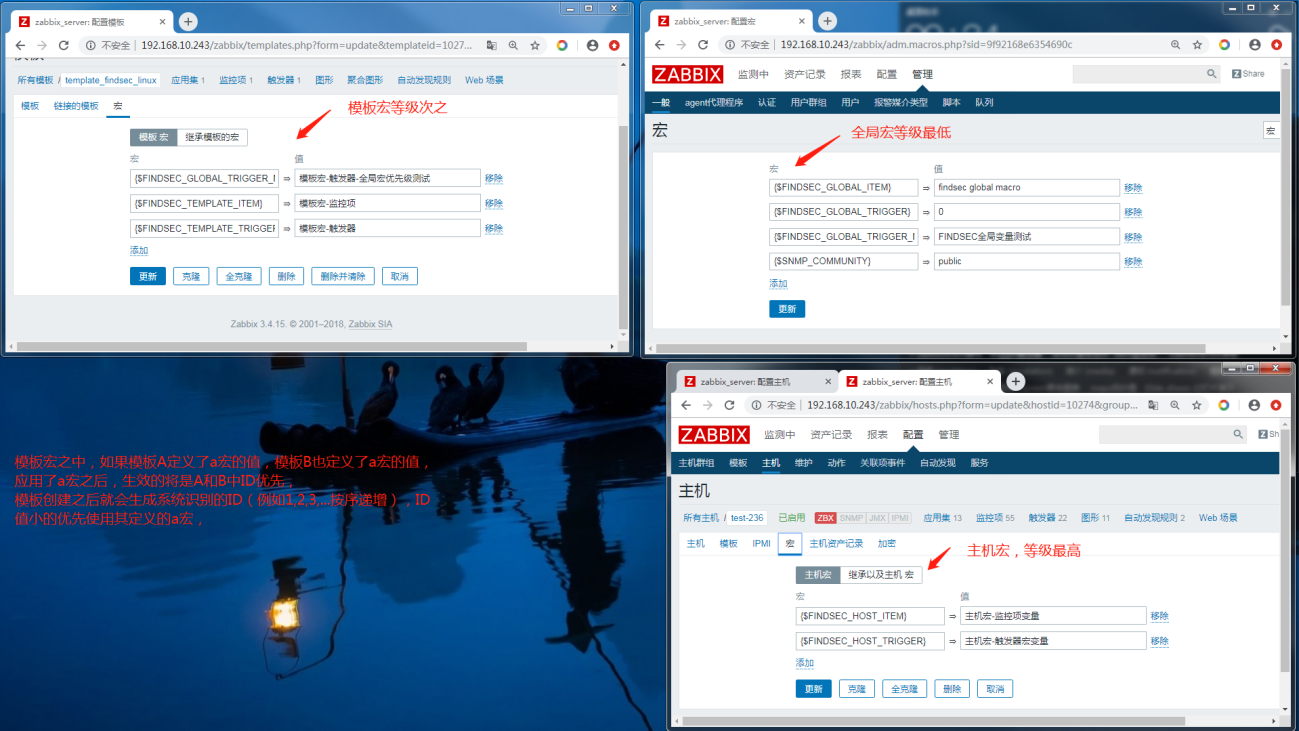
示例三：主机CPU负载触发器（使用2个宏）

{ca\_001:system.cpu.load[,avg1].min({$CPULOAD\_PERIOD})}>{$MAX\_CPULOAD}

宏可以作为function(功能)的一个参数被传入运算。

**宏的优先级：**

主机宏(HOST)--优先级最高。主机模板定义的宏--优先级次之。全局宏(Global)--优先级最低



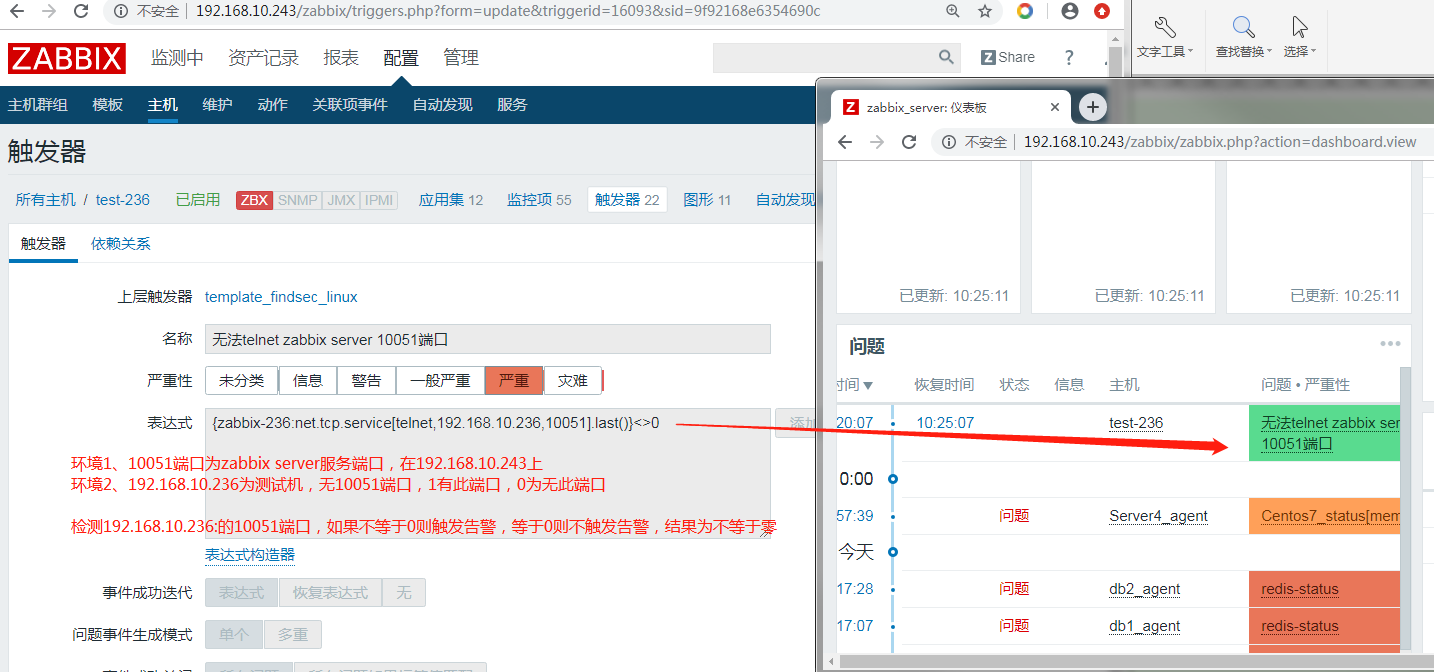
## 系统宏--

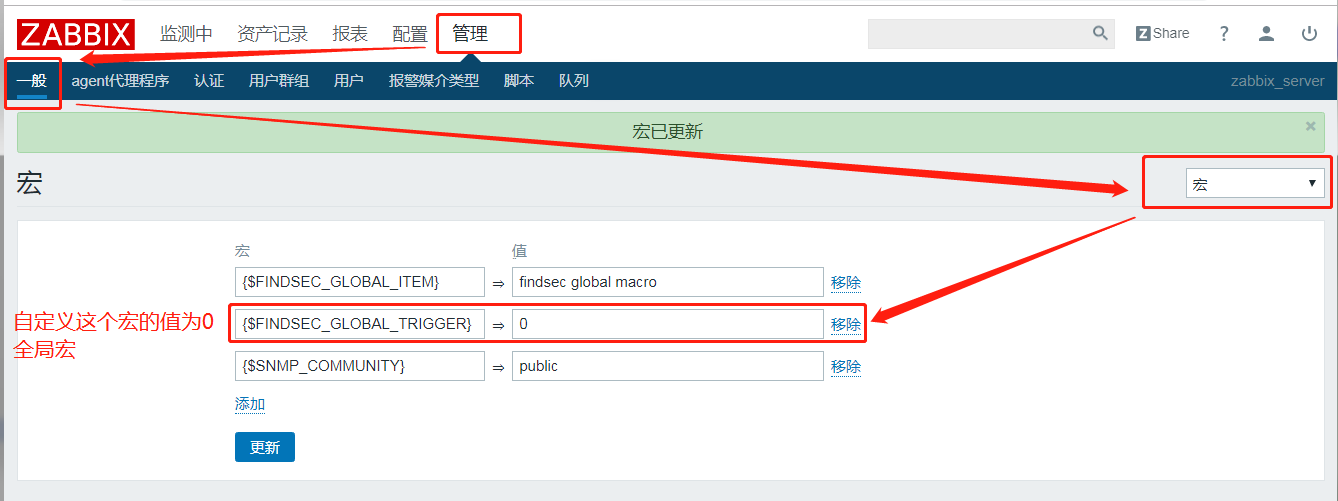
适用范围：每个系统宏适用的场景不同，具体参考官方网站文档<https://www.zabbix.com/manuals>

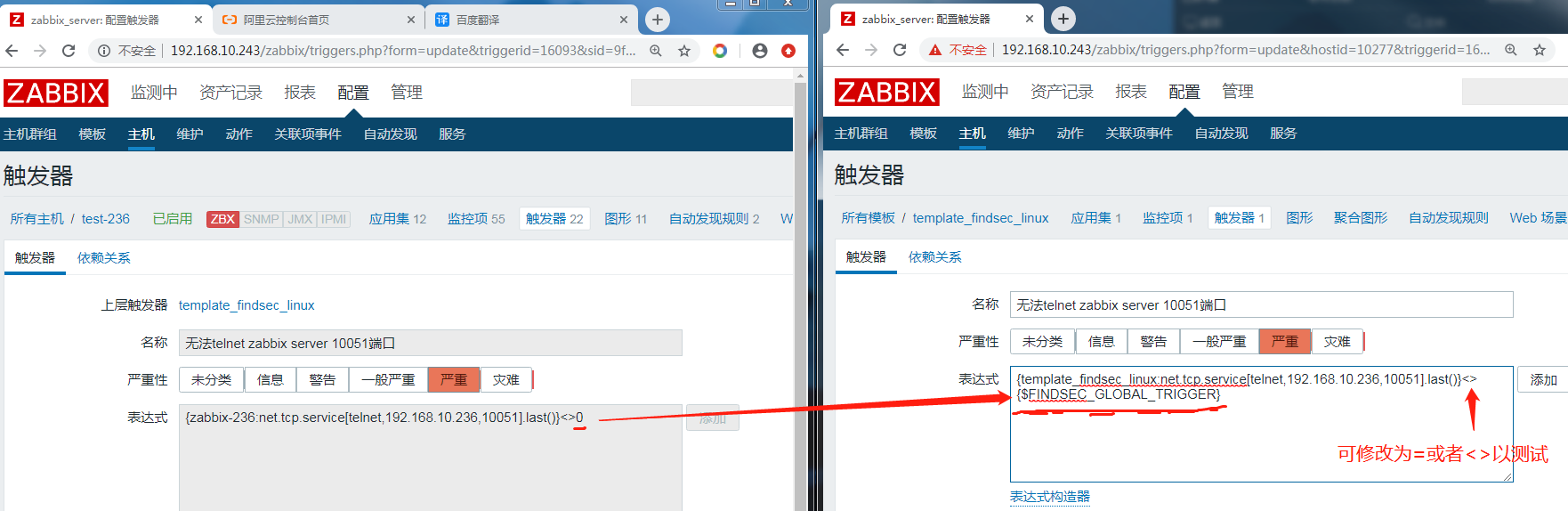
## 用户自定义宏--$

**适用范围：监控项、触发器、自动发现、低级别自动发现、web场景监控、全局宏**

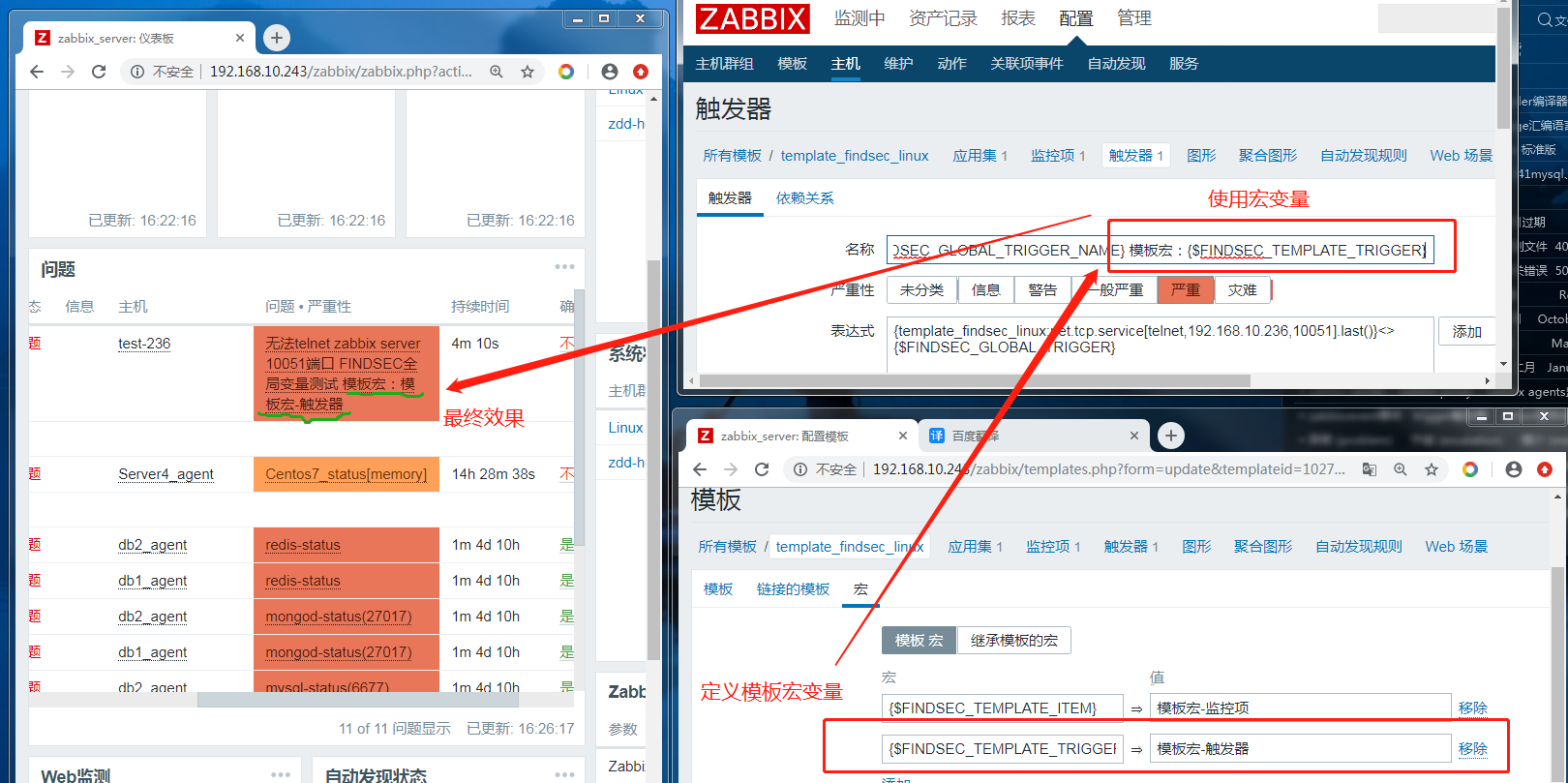
**全局宏变量定义与使用**



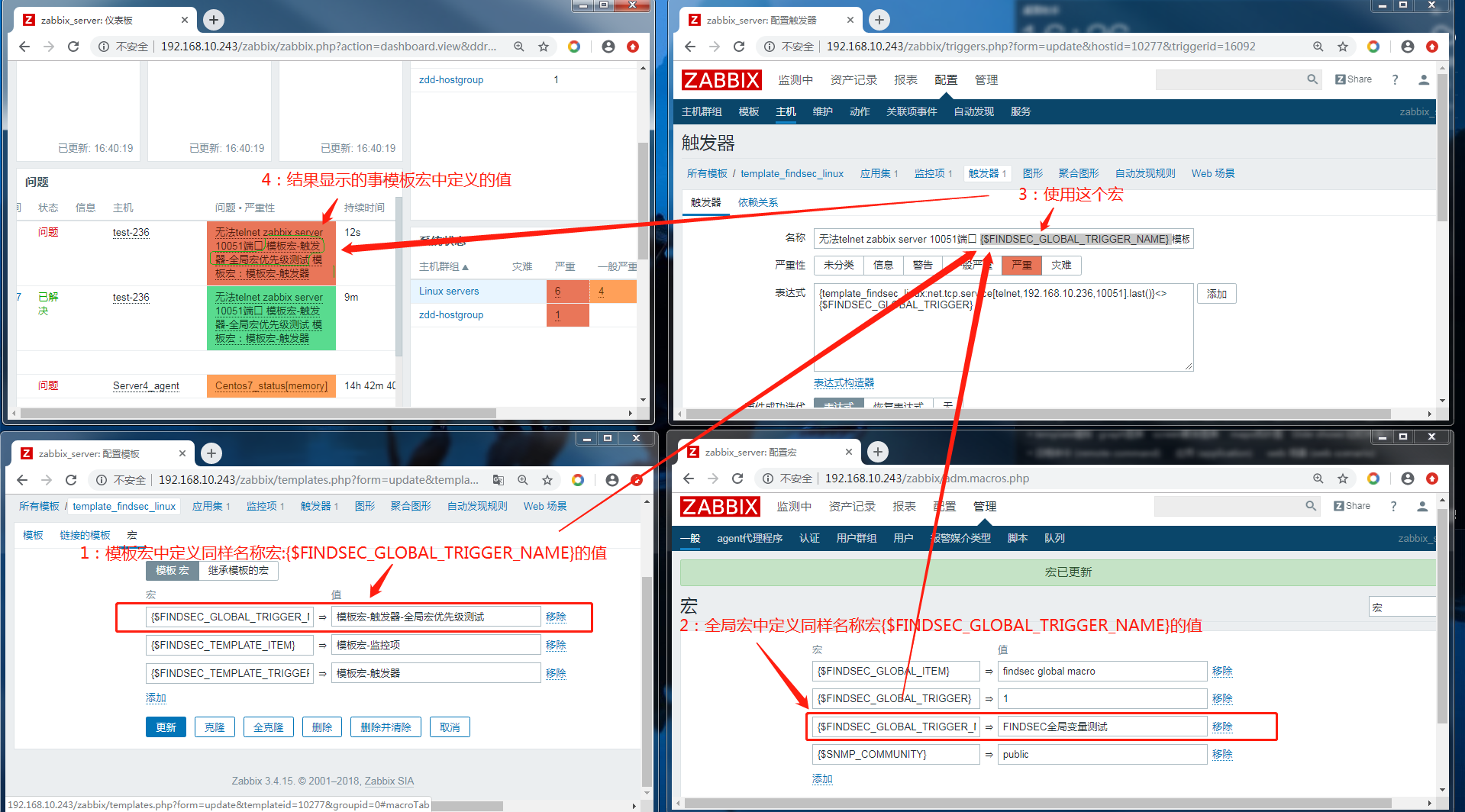
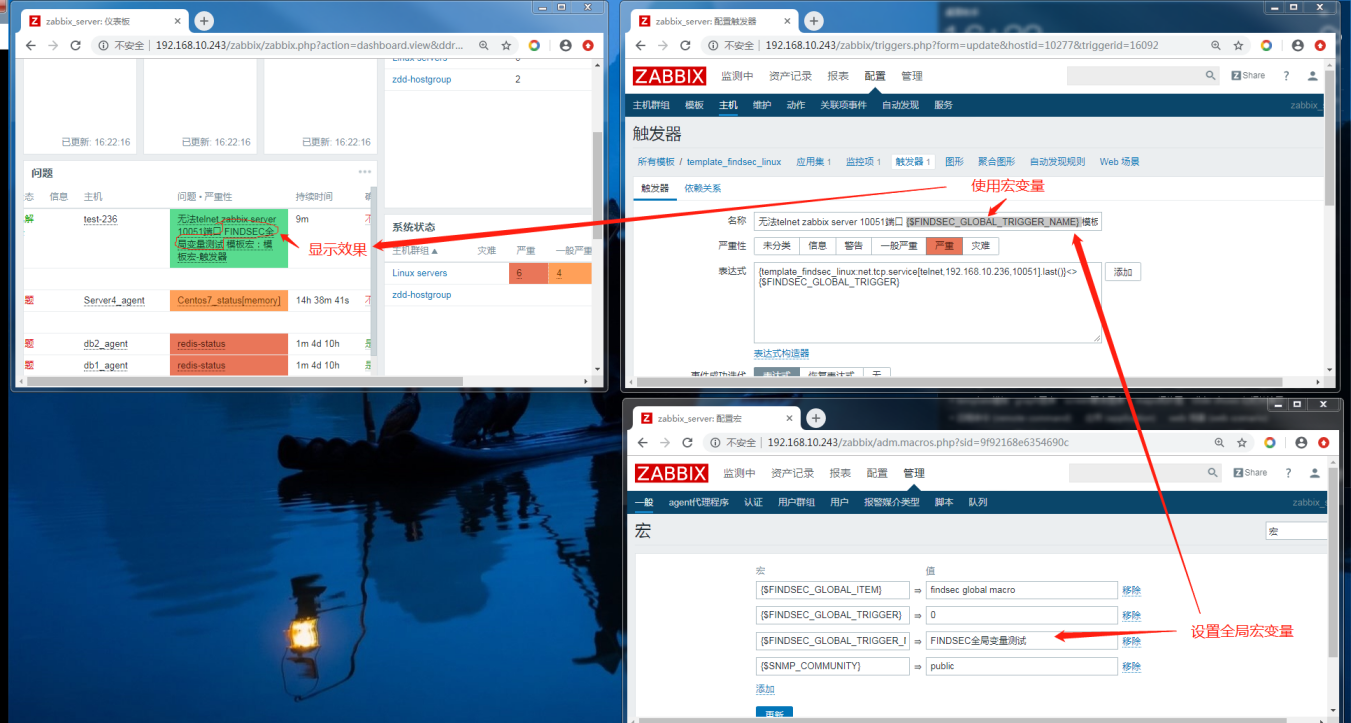




**模板中定义宏**



**优先级：模板宏比全局宏的等级高**



## 低级别自动发现宏（LLD）--#

应用场景：监控项原型、触发器原型、图形原型、主机原型。

## 宏变量函数：

**用来处理宏变量的值，目的是将满足条件的值自定义输出。**

regsub(<pattern>,<output>)区分大小写

iregsub(<pattern>,<output>)不区分大小写 #英语翻译：pattern图案 output输出

#pattern - 匹配的正则表达式

#output - 输出的选项。 \1 - \9 占位符支持被正则表达式匹配的组

**宏函数的语法为：**

{<macro>.<func>(<params>)} #英语翻译：func功能 params参数

其中：

<macro> - 这个参数为要定义的宏 （例如 {ITEM.VALUE}）；

<func> - 要应用的函数；

<params> - 以逗号分隔的函数参数列表。如果他们以 (空格), " 或者包含 ), ,这些符号开始，则参数必须要引用。

例如：{{ITEM.VALUE}.regsub(pattern, output)} #{{ITEM.VALUE}为<macro> regsub(pattern, output)为<func>

例如：{{HOST.NAME}.regsub(pattern, output)}

示例：关于宏函数可用于自定义宏值的方法，在下面的示例中说明，其中包含的 “log line” 作为接收值：

| **序号** | **参数（params）** | **宏函数** | **输出** |
| --- | --- | --- | --- |
| 1 | 123Log line | {{ITEM.VALUE}.regsub(^[0-9]+, Problem)} | Problem |
| 2 | 123 Log line | {{ITEM.VALUE}.regsub("^([0-9]+)", "Problem")} | Problem |
| 3 | 123 Log line | {{ITEM.VALUE}.regsub("^([0-9]+)", Problem ID: \1)} | Problem ID: 123 |
| 4 | Log line | {{ITEM.VALUE}.regsub(".\*", "Problem ID: \1")} | Problem ID: |
| 5 | MySQL crashed errno 123 | {{ITEM.VALUE}.regsub("^([A-Z]+).\*([0-9]+)", " Problem ID: \1\_\2 ")} | Problem ID: MySQL\_123? |
| 6 | 123 Log line | {{ITEM.VALUE}.regsub("([1-9]+", "Problem ID: \1")} | \*UNKNOWN\* (invalid regular expression) |