马哥教育
IT人的高薪职业学院

企业级监控服务 Zabbix

讲师：杰哥(张士杰)
http://www.magedu.com

一：监控服务介绍：
1.1：逻辑布局：
1.2：整体布局：
1.3：常见的监控方案：
1.3.1：Cacti：
1.3.2：Nagios：
1.3.3：SmokePing：
1.3.4：Open-falcon：
1.3.5：Zabbix：
1.3.6：Prometheus：
1.3.7：商业监控解决方案：
1.4：Zabbix使用场景及系统概述：
1.4.1：zabbix使用场景：
1.4.2：zabbix系统概述：
1.4.2.1：数据采集：
1.4.2.2：数据存储：
1.4.2.3：数据类型：
1.4.2.4：阈值：
1.4.2.5：告警机制：
1.4.3：zabbix 核心任务：
1.4.1：数据采集：
1.4.2：数据存储：
1.4.3：数据展示：
1.4.4：告警通知：
二：Zabbix 规划及部署：
2.1：系统环境：
2.1.1：Centos 7.x：
2.1.2：Ubuntu 18.04：

2.2：apt/yum安装 zabbix：
2.2.1：apt安装zabbix：
2.2.1.1：安装zabbix server：
2.2.1.2：准备数据库：
2.2.1.3：导入数据库：
2.2.1.4：数据库验证数据：
2.2.1.5：编辑zabbix server配置文件：
2.2.1.6：重启zabbix服务并：
2.2.1.7：访问web界面：
2.2.1.8：解决时区问题：
2.2.1.9：数据库配置：
2.2.1.10：zabbix server配置：
2.2.1.11：信息确认：
2.2.1.12：配置完成：
2.2.1.13：登录界面：
2.2.1.14：zabbix登录后的监控图形界面：
2.2.2：yum安装zabbix：
2.3：编译安装zabbix：
2.3.1：下载源码包：
2.3.2：解决依赖关系：
2.3.3：编译安装Zabbix：
2.3.4：准备数据库：
2.3.5：导入数据库：
2.3.6：数据库验证数据：
2.3.7：编辑zabbix server配置文件：
2.3.8：启动zabbix server：
2.3.9：配置web界面：
2.3.9.1：拷贝web界面程序：
2.3.9.2：访问web界面：
2.3.9.3：当前报错界面：
2.3.9.4：解决报错：
2.3.9.5：验证当前界面：
2.3.9.6：数据库配置：
2.3.9.7：zabbix server配置：
2.3.9.8：信息确认：
2.3.9.9：创建配置文件：
2.3.9.10：验证zabbix server配置文件：
2.3.9.11：刷新页面:
2.3.9.12：登录界面
2.3.9.13：zabbix 首页：
2.3.10：启动zabbix agent：
2.3.11：验证zabbix 监控数据：
2.3.12：zabbix server与agent启动文件：
2.3.12.1：zabbix server启动脚本：
2.3.12.2：zabbix agent启动脚本：
2.4：Web界面中文菜单环境：
2.4.1：中文选项无法选择：
2.4.2：ubuntu系统安装中文语言环境：
2.4.3：验证web界面：
2.5：监控项与乱码：
2.5.1：图形乱码：
2.5.2：上传字体文件：
2.5.2.1：在windows拷贝字体：

2.5.2.2：上传字体到zabbix web目录：
2.5.2.3：修改zabbix文件调用新字体：
2.5.2.4：验证字体是否生效：
2.6：zabbix server配置文件详解：
三：Zabbix 监控入门基础：
3.1：监控linux系统：
3.1.1：zabbix agent安装：
3.1.2：zabbix agent配置文件：
3.1.3：重启zabbix agent：
3.1.4：验证zabbix agent：
3.1.5：zabbix web界面添加被监控主机：
3.1.6：关联监控模板：
3.1.6.1：选择模板：
3.1.6.2：确认模板选择：
3.1.6.3：确认添加主机：
3.1.6.4：主机添加完成：
3.1.6.5：验证主机监控数据：
3.2：监控tomcat：
3.2.1：准备JDK环境：
3.2.2：准备tomcat：
3.2.3：验证tomcat web页面：
3.2.4：部署java gateway服务器：
3.2.5：配置zabbix server调用java gateway：
3.2.6：验证Java Pollers：
3.2.7：tomcat开启JMX监控：
3.2.8：通过jconsole验证JMX数据：
3.2.9：非安全连接：
3.2.10：验证连接：
3.2.11：zabbix server添加JMX监控：
3.2.12：zabbix server关联模板：
3.2.13：验证当前JMX状态及数据：
3.2.14：JMX监控生产模板使用：
3.2.14.1：选择并导入模板：
3.2.14.2：关联模板：
3.2.14.3：验证当前模板JMX数据：
3.2.15：Linux测试监控JMX方式：
3.3：zabbix 主动与被动监控模式：
3.3.1：被动模式：
3.3.1.1：被动模式端口状态：
3.3.1.2：被动模式工作流程：
3.3.2：主动模式：
3.3.2.1：主动模式工作流程：
3.3.2.2：修改zabbix agent为主动模式：
3.3.2.3：生成主动模式模板：
3.3.2.4：添加主动模式主机并关联主动模板：
3.3.2.5：验证主动模式主机状态：
3.3.2.6：验证主动模式主机数据：
3.3.2.7：验证主动模式主机端口：
四：zabbix proxy：
4.1：zabbix proxy架构：
4.2：zabbix proxy对比zbbbix server：
4.3：zabbix proxy部署与使用：
4.3.1：zabbix proxy版本选择：

4.3.2：zabbix proxy安装：
4.3.2.1：apt/yum安装zabbix proxy：
4.3.2.2：编译安装zabbix proxy：
4.3.3：配置被动zabbix proxy：
4.3.3.1：zabbix proxy被动配置：
4.3.3.2：重启zabbix proxy服务：
4.3.3.3：zabbix 添加被动代理：
4.3.3.4：配置zabbix agent使用被动代理：
4.3.3.4.1：添加被动模式代理主机：
4.3.3.4.2：关联被动模式模板：
4.3.3.4.3：zabbix server配置：
4.3.3.5：zabbix agent配置文件：
4.3.3.6：zabbix web验证当前主机状态：
4.3.3.7：验证主机监控数据及图形：
4.3.4：配置主动模式zabbix proxy：
4.3.4.1：zabbix proxy主动配置：
4.3.4.2：重启zabbix proxy服务：
4.3.4.3：zabbix web添加主动代理：
4.3.4.4：zabbix agent使用主动代理：
4.3.4.5：zabbix agent配置文件：
4.3.3.6：zabbix web验证当前主机状态：
4.3.3.7：验证主机监控数据及图形：
4.3.3.8：交互过程：
五：zabbix 监控案例实战：
5.1：监控Linux TCP连接状态：
5.1.1：TCP端口的十一种连接状态：
5.1.2：端口转换状态：
5.1.3：TCP三次握手与四次断开：
5.1.3.1：TCP三次握手：
5.1.3.2：TCP四次断开：
5.1.4：监控TCP连接数脚本：
5.1.5：zabbix agent添加自定义监控项：
5.1.6：zabbix server测试监控项数据：
5.1.7：zabbix web导入模板：
5.1.7.1：导入模板：
5.1.7.2：模板导入成功：
5.1.8：将TCP监控模板关联至主机：
5.1.9：验证监控数据：
5.2：监控memcache：
5.2.1：安装memcache服务：
5.2.2：监控脚本：
5.2.3：zabbix agent添加自定义监控项：
5.2.4：zabbix server测试监控项数据：
5.2.5：zabbix web制作模板：
5.2.5.1：创建模板：
5.2.5.2：创建监控项：
5.2.5.3：创建触发器：
5.2.5.4：创建图形：
5.2.6：模板关联主机：
5.2.7：验证监控项数据：
5.3：监控Redis：
5.3.1：安装Redis服务：
5.3.2：监控脚本：

5.3.3：zabbix agent添加自定义监控项：
5.3.4：zabbix server测试监控项数据：
5.3.5：zabbix wen模板制作：
5.3.5.1：创建模板：
5.3.5.2：创建触监控项：
5.3.5.2.1：当前连接数监控项：
5.3.5.2.2：已用内存监控项：
5.3.5.3：创建触发器
5.3.5.3.1：当前连接数触发器：
5.3.5.3.2：已用内存触发器：
5.3.5.4：创建图形：
5.3.5.4.1：Redis当前连接数图形：
5.3.5.4.2：Redis已用内存图形：
5.3.6：模板关联主机：
5.3.7：验证监控项数据：
5.3.7.1：redis 当前连接数图形：
5.3.7.2：redis已用内存连接数：
5.4：监控Nginx：
5.4.1：部署Nginx服务：
5.4.2：监控项脚本：
5.4.3：zabbix agent添加自定义监控项：
5.4.4：zabbix server测试监控项数据：
5.4.5：导入Nginx监控模板：
5.4.6：模板关联主机：
5.4.7：验证监控数据：
5.5：SNMP监控：
5.5.1：SNMP组织机构：
5.5.2：SNMP数据交互：
5.5.3：SNMP组织结构：
5.5.4：SNMP MIB：
5.5.5：基于Centos的SNMP：
5.5.6：SNMP OID：
5.5.7：测试SNMP数据采集：
5.5.8：Centos SNMP：
5.5.9：添加SNMP监控主机：
5.5.9.1：添加SNMP主机：
5.5.9.2：关联SNMP模板：
5.5.10：验证主机当前状态：
5.5.11：验证SNMP监控数据：
5.6：监控MySQL：
5.6.1：实现MySQL主从：
5.6.1.1：MySQL Master：
5.6.1.2：MySQL Slave：
5.6.1.3：MySQL Master授权账户：
5.6.1.4：MySQL slave导入数据：
5.6.2：Procona监控MySQL：
5.6.2.1：MySQL Master安装zabbix-agent：
5.6.2.2：MySQL Master安装Percona：
5.6.2.3：zabbix web导入Percona模板：
5.6.2.4：zabbbix web添加主机：
5.6.2.5：zabbix web对主机关联模板：
5.6.2.6：验证MySQL监控数据：
5.6.3：自定义脚本监控MySQL：

5.6.3.1：MySQL slave安装zabbix agent：
5.6.3.1：脚本内容：
5.6.3.2：自定义监控项配置：
5.6.3.3：自定义模板：
5.6.3.3.1：创建模板：
5.6.3.3.2：添加监控项：
5.6.3.3.3：添加触发器：
5.6.3.3.4：添加图形：
5.6.3.4：模板关联至主机：
5.6.3.4.1：添加主机：
5.6.3.4.2：关联模板：
5.6.3.5：验证监控数据：
5.7：自定义端口和进程监控：
5.8：故障自治愈功能：
5.8.1：zabbix agent需要开启远程命令执行：
5.8.2：zabbix用户授权：
5.8.3：创建动作：
5.8.4：执行远程操作：
5.8.5：验证自治愈功能：
5.8.5.1：验证自治愈：
5.8.5.2：zabbix 自治愈日志：
5.9：grafana图形展示：
5.9.1：安装grafana服务：
5.9.2：grafana安装并启用zabbix插件：
5.9.2.1：安装zabbix插件：
5.9.2.2：启动zabbix插件：
5.9.2.3：添加MySQL数据源：
5.9.2.4：测试MySQL数据源并保存：
5.9.2.5：添加zabbix 数据源：
5.2.9.6：保存并添加数据源：
5.2.5.9.7：添加Dashboard：
5.2.5.9.7.1：单独添加图形：
5.2.5.9.7.2：导入模板：
5.10：自定义基础监控模板：
5.11：结合pyhton脚本监控案例：
5.11.1：kubernetes 集群状态监控脚本：
5.11.2：监控MongodbDB复制集状态：
5.11.3：监控Redis列表长度：
5.11.4：监控ELK集群状态：
5.11.5：监控RabbitMQ集群节点状态：
六：Zabbix 事件通知机制：
6.1：邮件通知：
6.1.1：邮箱开启SMTP：
6.1.2：生成授权码：
6.1.2.1：生成授权码：
6.1.2.2：发送验证码：
6.1.2.3：保存授权码：
6.1.2.4：Zabbix Web创建报警媒介类型：
6.1.2.5：给用户添加报警报警媒介：
6.1.2.6：更新报警媒介：
6.1.2.7：创建动作：
6.1.2.8：配置动作信息：
6.1.2.9：验证动作当前状态：

6.1.2.10：配置故障恢复信息：
6.1.2.11：添加动作：
6.1.2.12：验证动作：
6.1.2.12.1：验证事件状态：
6.1.2.12.2：邮箱验证是否收到邮件：
6.2：短信通知：
6.2.1：添加短信报警媒介类型：
6.2.2：添加联系人报警媒介：
6.2.3：创建短信通知动作：
6.2.4：配置短信发送具体内容：
6.2.5：配置恢复操作：
6.2.6：验证动作状态：
6.2.8：测试短信报警：
6.3：微信通知：
6.3.1：企业微信注册及配置：
6.3.1.1：企业微信账号注册：
6.3.1.2：登录PC版：
6.3.1.3：创建应用：
6.3.1.4：填写应用信息：
6.3.1.5：注册完成：
6.3.1.6：创建微信账号：
6.3.1.7：验证通讯录：
6.3.1.8：查看企业信息：
6.3.1.9：测试发送信息：
6.3.1.10：选择消息接收人：
6.3.1.11：确认消息接收人：
6.3.1.12：开始发送信息：
6.3.1.13：确认发送：
6.3.1.14：手机验证消息：
6.3.2：zabbix server配置：
6.3.2.1：编写python脚本：
6.3.2.2：添加微信报警媒介类型：
6.3.2.3：添加联系人报警媒介：
6.3.2.4：更新报警媒介：
6.3.2.5：添加微信报警动作：
6.3.2.6：配置故障操作：
6.3.2.7：配置恢复操作：
6.3.2.8：验证动作状态：
6.3.2.9：测试微信报警：
七：Zabbix 自动化运维：
7.1：Zabbix Agent批量部署：
7.1.1：源码编译安装：
7.1.2：apt/yum在线安装：
7.2：Zabbix API添加主机
7.2.1：API使用基础：
7.2.2：API批量添加主机：
7.3：Zabbix 动态发现主机

第一部分：Zabbix安装部署、实现分布式监控及网络知识

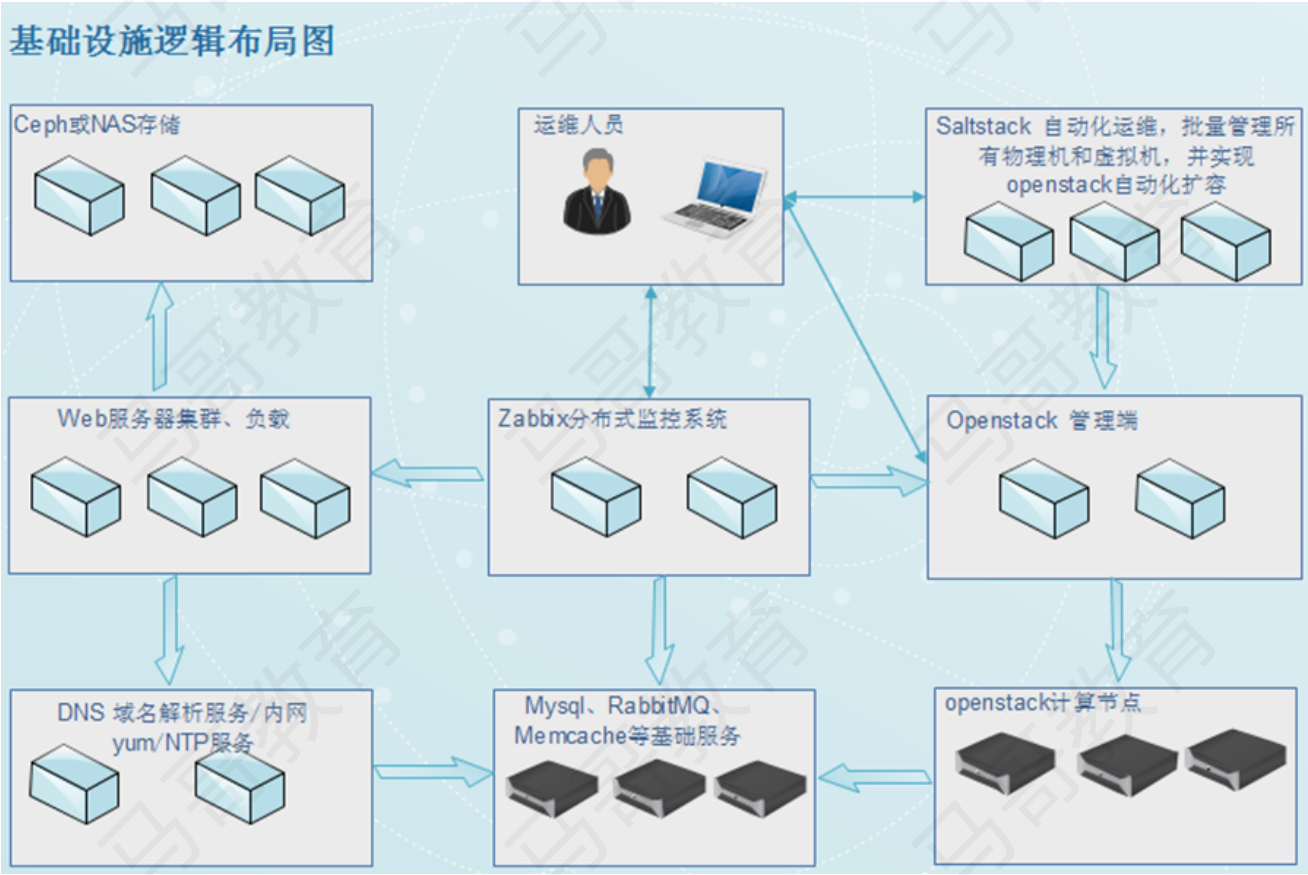
```
Zabbix 监控及业务整体规划图
安装Zabbix Server端和数据库
安装客户端实现被动模式监控
监控tomcat
安装Zabbix proxy端和数据库
配置zabbix 主动模式代理
Zabbix Server通过主动 proxy 监控服务器
```

第二部分：完全自定义模板及监控常用服务

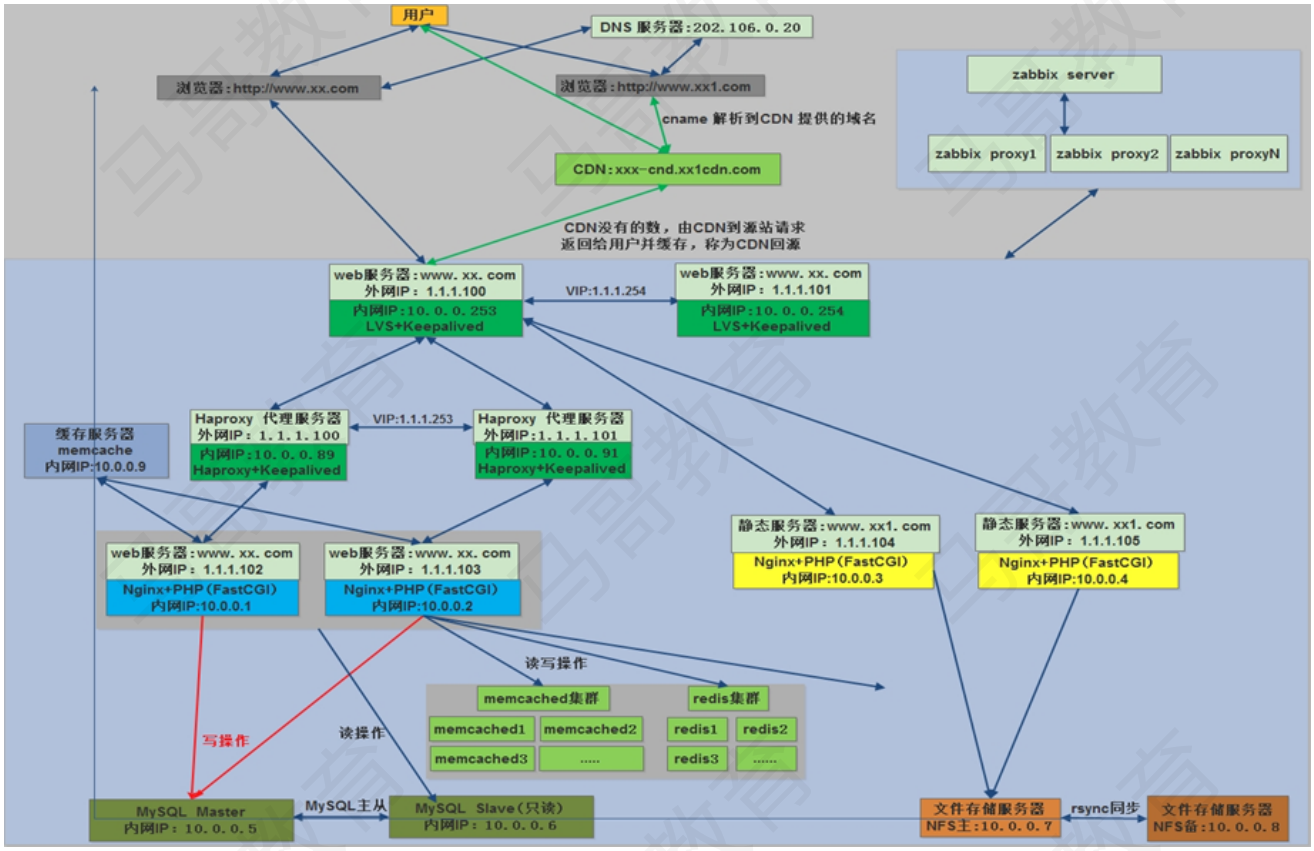
```
监控系统TCP连接数
监控memcache
监控redis
监控mysql
监控nginx
自定义脚本监控端口和进程
自定义系统基础监控模板
编写脚本一键安装zabbix agent
编写shell+python脚本实现短信报警
```

一：监控服务介绍：

1.1：逻辑布局：



1.2：整体布局：



1.3：常见的监控方案：

开源监控软件：cacti、nagios、zabbix、smokeping、open-falcon等

1.3.1：Cacti：

https://www.cacti.net/

https://github.com/Cacti/cacti

> Cacti是基于LAMP平台展现的网络流量监测及分析工具，通过SNMP技术或自定义脚本从目标设备/主机获取监控指标信息；其次进行数据存储，调用模板将数据存到数据库，使用rrdtool存储和更新数据，通过rrdtool绘制结果图形；最后进行数据展现，通过web方式将监控结果呈现出来，常用于在数据中心监控网络设备。

1.3.2：Nagios：

https://www.nagios.org/

> Nagios用来监视系统和网络的开源应用软件，利用其众多的插件实现对本机和远端服务的监控，当被监控对象发生异常时，会及时向管理员告警，提供一批预设好的监控插件，用户可以之间调用，也可以自定义Shell脚本来监控服务，适合各企业的业务监控，可通过web页面显示对象状态、日志、告警信息，分层告警机制及自定义监控相对薄弱。

1.3.3：SmokePing：

https://oss.oetiker.ch/smokeping/

http://blogs.studylinux.net/?p=794

> Smokeping是一款用于网络性能监测的开源监控软件，主要用于对IDC的网络状况，网络质量，稳定性等做检测，通过rrdtool制图方式，图形化地展示网络的时延情况，进而能够清楚的判断出网络的即时通信情况。

1.3.4：Open-falcon：

https://www.open-falcon.org/

https://github.com/XiaoMi/open-falcon

> 小米公司开源出来的监控软件，监控能力和性能较强。

1.3.5：Zabbix：

https://www.zabbix.com/cn/

> 目前使用较多的开源监控软件，可横向扩展、自定义监控项、支持多种监控方式、可监控网络与服务等。

1.3.6：Prometheus：

针对容器环境的开源监控软件

1.3.7：商业监控解决方案：

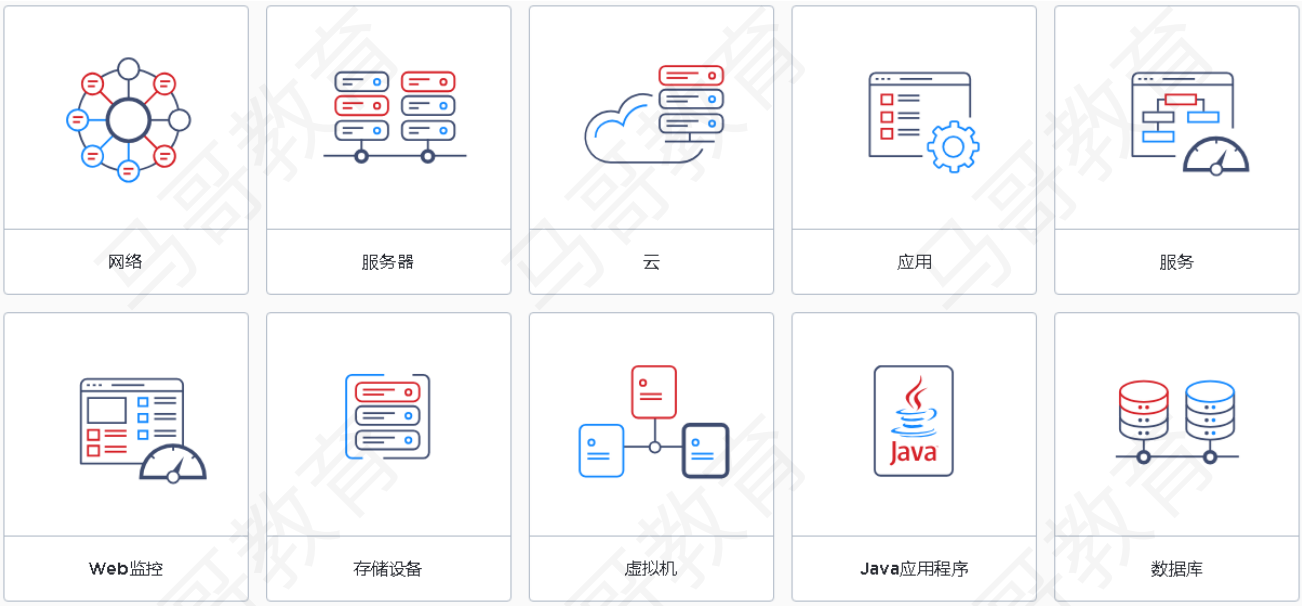
```
监控宝(https://www.jiankongbao.com/)
听云(https://www.tingyun.com/)
```

1.4：Zabbix使用场景及系统概述：

https://www.zabbix.com/cn/features

Zabbix是一个企业级解决方案，支持实时监控数千台服务器，虚拟机和网络设备，采集百万级监控指标，适用于任何IT基础架构、服务、应用程序和资源的解决方案

1.4.1：zabbix使用场景：



1.4.2：zabbix系统概述：

1.4.2.1：数据采集：

周期性时序数据

```
主机/对象：服务器、路由器、交换机、存储、防火墙、IP、PORT、URL、自定义监控对象...
采集目标：监控项，指标数据（metrics data）
```

1.4.2.2：数据存储：

监控数据存储系统

```
SQL：MySQL/MariaDB(Zabbix)
NoSQL：Redis(Open-falcon)
rrd：Round Robin Database(Cacti)
```

1.4.2.3：数据类型：

```
历史数据：每个监控项采集到的每个监控值
趋势数据：趋势表里主要保留某个监控项一个小时内历史数据的最大值、最小值和平均值以及该监控项一个小时内所采集到的数据个数。
```

1.4.2.4：阈值：

可按照预定义的阈值等级实现分层报警

1.4.2.5：告警机制：

email,短信,微信,语音,故障自治愈

1.4.3：zabbix 核心任务：

1.4.1：数据采集：

数据采集方式：zabbix-server，zabbix-proxy，zabbix-agent

```
Agentless：SNMP,Telnet,ssh，IPMI，JMX，
Agent：zabbix agent
```

1.4.2：数据存储：

zabbix database

1.4.3：数据展示：

zabbix web：

```
graph -> screen -> slideshow(将多个screen以幻灯片的方式进行轮流展示)
```

grafana：

```
以zabbix为数据源展示更绚丽的界面
```

1.4.4：告警通知：

```
host (host groups) <- templates #从模板继承告警配置
host -> items -> triggers -> action (条件-conditions，操作-operations) #自定义告警配置
```

二：Zabbix 规划及部署：

部署环境：

服务器系统：ubuntu 18.04.3/centos 7.x

主机类型	IP地址
zabbix server	172.31.0.101
zabbix 主动代理	172.31.0.102
zabbix 被动代理	172.31.0.103
mysql master	172.31.0.104
mysql slave	172.31.0.105
web server1	172.31.0.106
web server2	172.31.0.107

2.1：系统环境：

最小化安装操作系统，然后安装常用依赖包：

2.1.1：Centos 7.x：

```
# yum install  vim iotop bc gcc gcc-c++ glibc glibc-devel pcre pcre-devel openssl
openssl-devel zip unzip zlib-devel   net-tools lrzsz tree ntpdate telnet lsof tcpdump
wget libevent libevent-devel
```

2.1.2：Ubuntu 18.04：

```
# apt  install iproute2  ntpdate  tcpdump telnet traceroute nfs-kernel-server nfs-
common  lrzsz tree  openssl libssl-dev libpcre3 libpcre3-dev zlib1g-dev ntpdate tcpdump
telnet traceroute  gcc openssh-server lrzsz tree  openssl libssl-dev libpcre3 libpcre3-
dev zlib1g-dev ntpdate tcpdump telnet traceroute iotop unzip zip
```

2.2：apt/yum安装 zabbix：

https://www.zabbix.com/documentation/4.0/zh/manual #产品手册

2.2.1：apt安装zabbix：

使用apt在ubuntu 安装zabbix 4.0.x版本

2.2.1.1：安装zabbix server：

https://www.zabbix.com/cn/download?zabbix=4.0&os_distribution=ubuntu&os_version=18.04_bionic&db=mysql&ws=apache

```
# wget https://repo.zabbix.com/zabbix/4.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.0-3+bionic_all.deb
# dpkg -i zabbix-release_4.0-3+bionic_all.deb
# apt update
# apt -y install zabbix-server-mysql zabbix-frontend-php zabbix-agent
```

2.2.1.2：准备数据库：

在mysql maser安装数据库并授权zabbix使用

```
# apt update
# apt install mysql-server mysql-client #或者安装 mariadb-server mariadb-client
# vim  /etc/mysql/mysql.conf.d/mysqld.cnf #修改监听地址
    bind-address            = 0.0.0.0
# systemctl  restart  mysql

# mysql #进入数据库创建账户并授权
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.28-0ubuntu0.18.04.4 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database zabbix_server character set utf8 collate utf8_bin;
Query OK, 1 row affected (0.00 sec)

mysql> grant all privileges on zabbix_server.* to zabbix@"172.31.0.%" identified by
'magedu.zabbix';
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

2.2.1.3：导入数据库：

在zabbix server初始化数据库

```
root@zabbix-server:~# mysql -uzabbix -pmagedu.zabbix -h172.31.0.104 #测试账户权限
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.28-0ubuntu0.18.04.4 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
```

```
+--------------------+
| information_schema |
| zabbix_server      |
+--------------------+
2 rows in set (0.00 sec)
mysql> exit

# zcat /usr/share/doc/zabbix-server-mysql/create.sql.gz  | mysql -uzabbix -
pmagedu.zabbix -h172.31.0.104 zabbix_server #初始化数据库
```

2.2.1.4：数据库验证数据：

```
mysql> use zabbix_server;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------------+
| Tables_in_zabbix_server    |
+----------------------------+
| acknowledges               |
| actions                    |
| alerts                     |
| application_discovery      |
| application_prototype      |
| application_template       |
| applications               |
| auditlog                   |
| auditlog_details           |
| autoreg_host               |
| conditions                 |
| config                     |
```

2.2.1.5：编辑zabbix server配置文件：

```
root@zabbix-server:~# vim /etc/zabbix/zabbix_server.conf
DBHost=172.31.0.104
DBName=zabbix_server
DBUser=zabbix
DBPassword=magedu.zabbix
DBPort=3306
```

2.2.1.6：重启zabbix服务并：

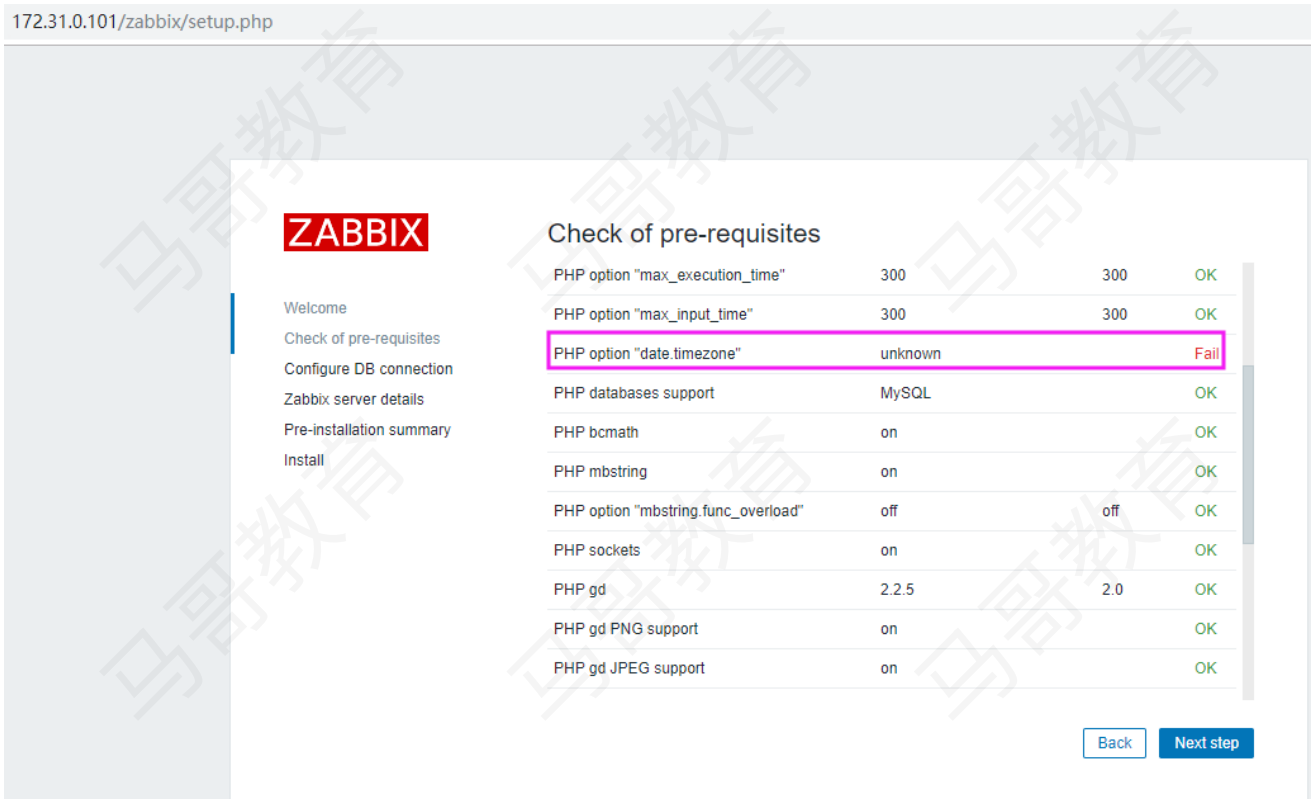
```
# systemctl restart zabbix-server zabbix-agent apache2
```

2.2.1.7：访问web界面：

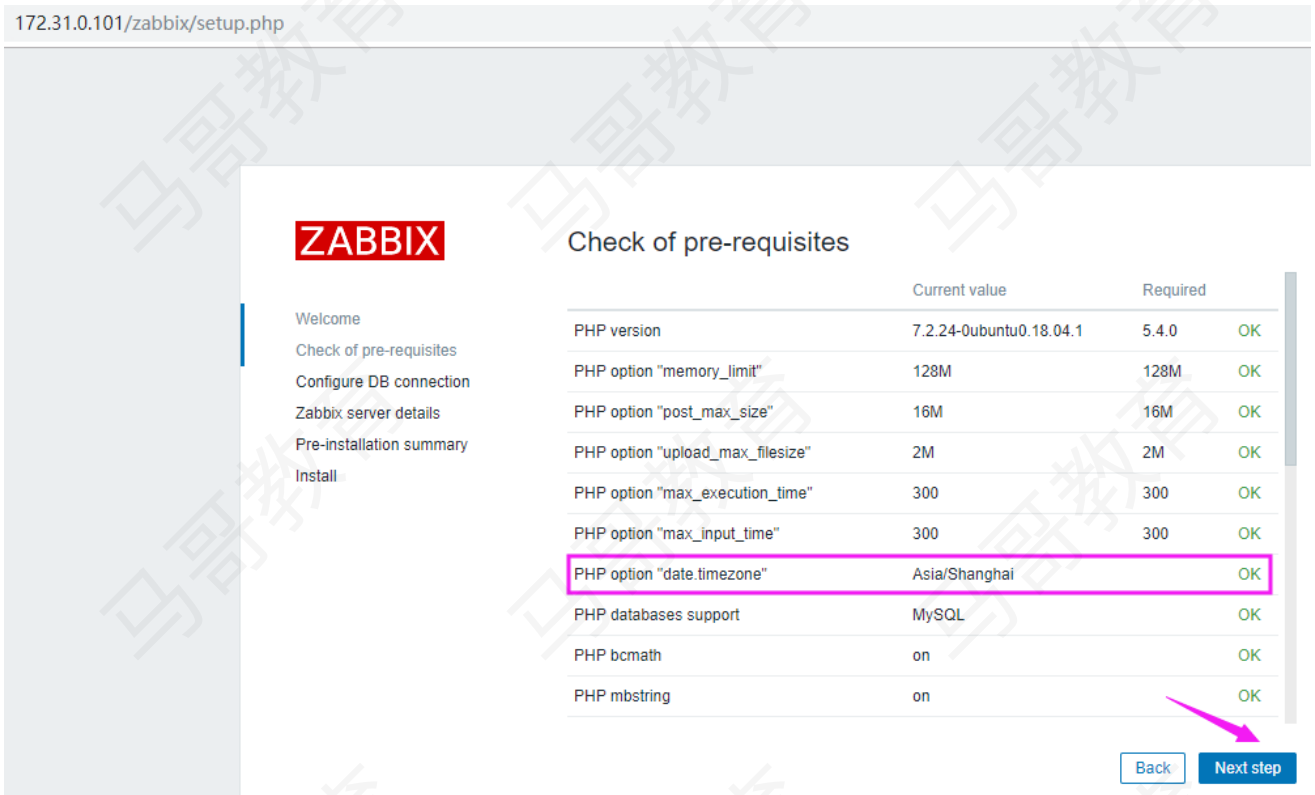
2.2.1.8：解决时区问题：

```
root@zabbix-server:~# vim /etc/zabbix/apache.conf
php_value date.timezone Asia/Shanghai
root@zabbix-server:~# systemctl restart zabbix-server zabbix-agent apache2
root@zabbix-server:~# systemctl enable  zabbix-server zabbix-agent apache2
```

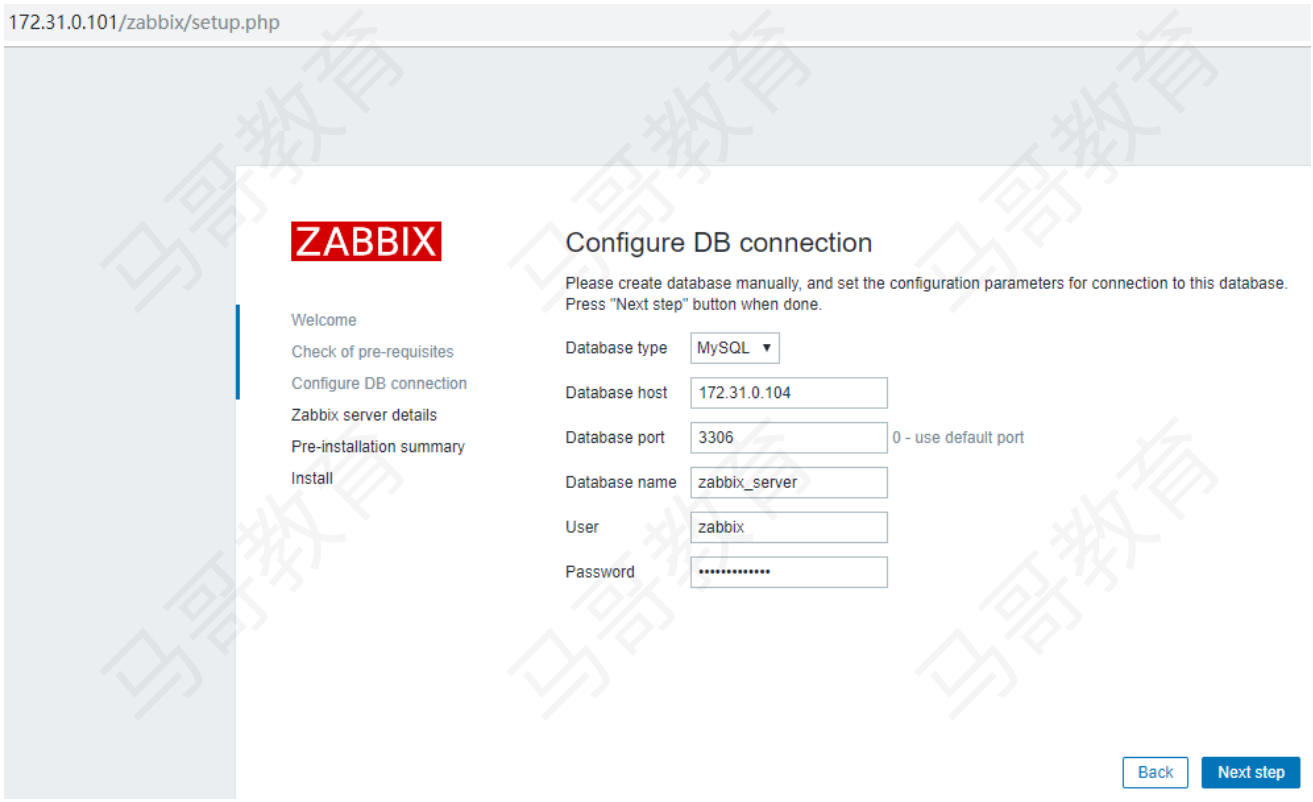
#报错的界面：



#正常的界面：

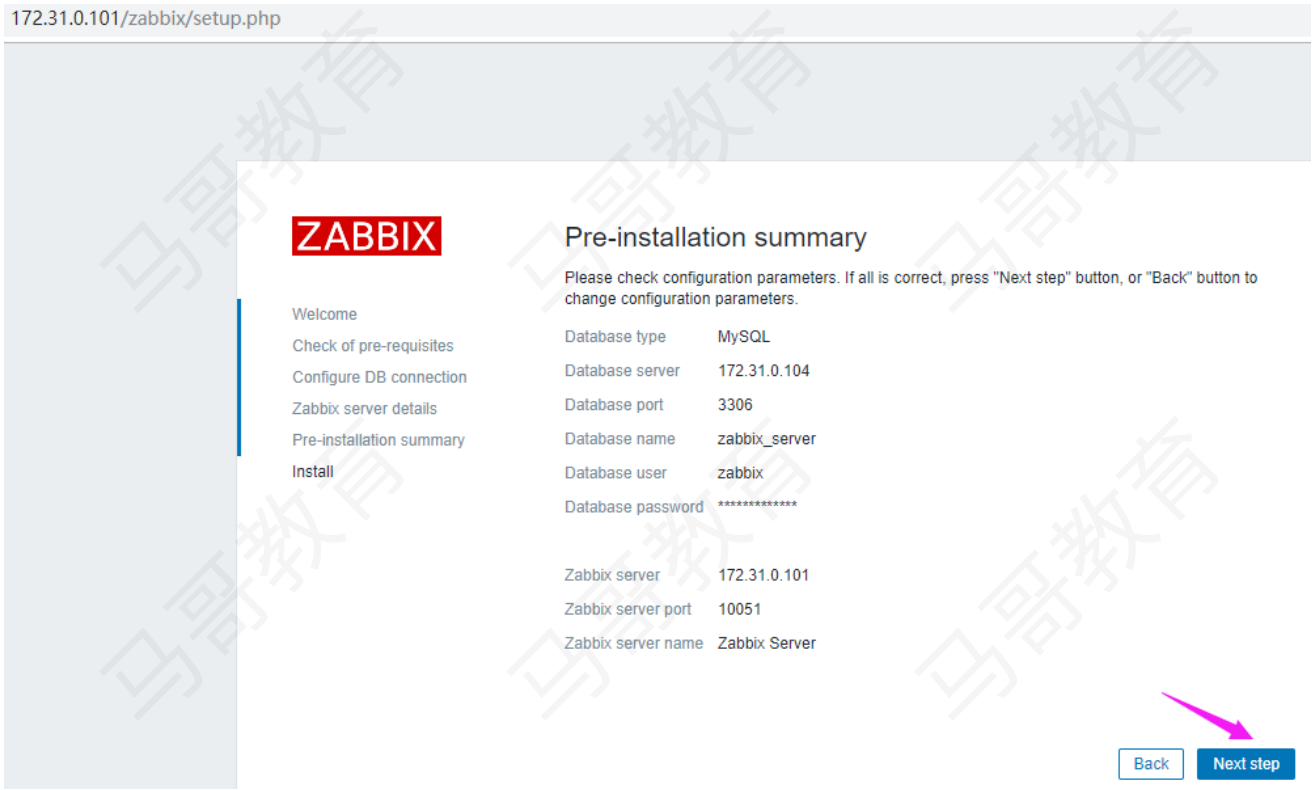


2.2.1.9：数据库配置：

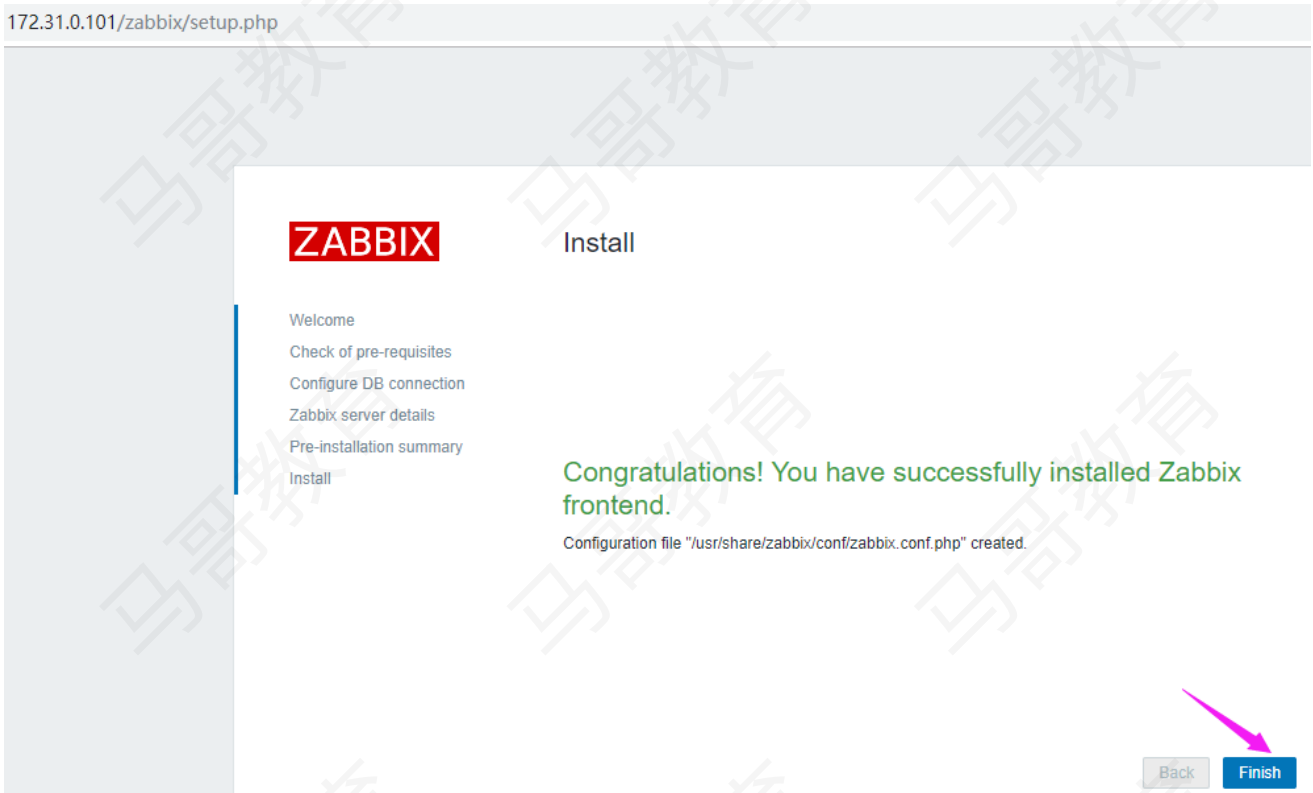


2.2.1.10：zabbix server配置：

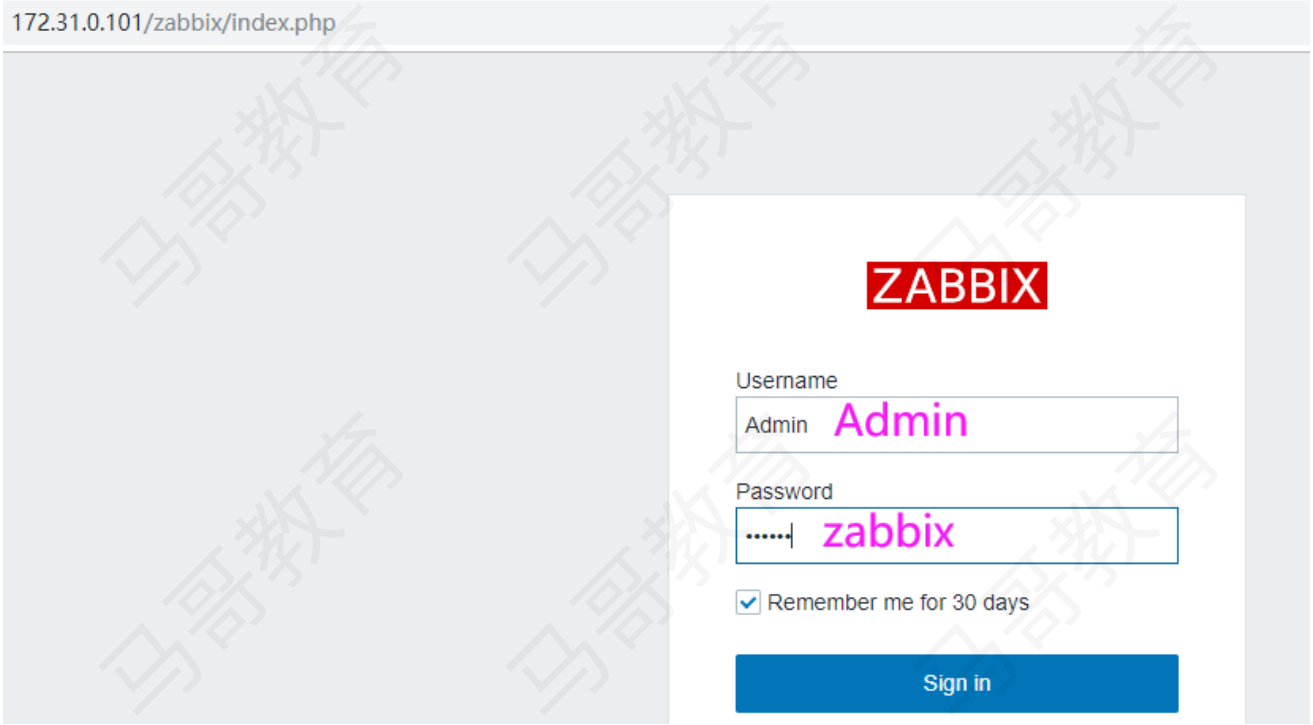
2.2.1.11：信息确认：



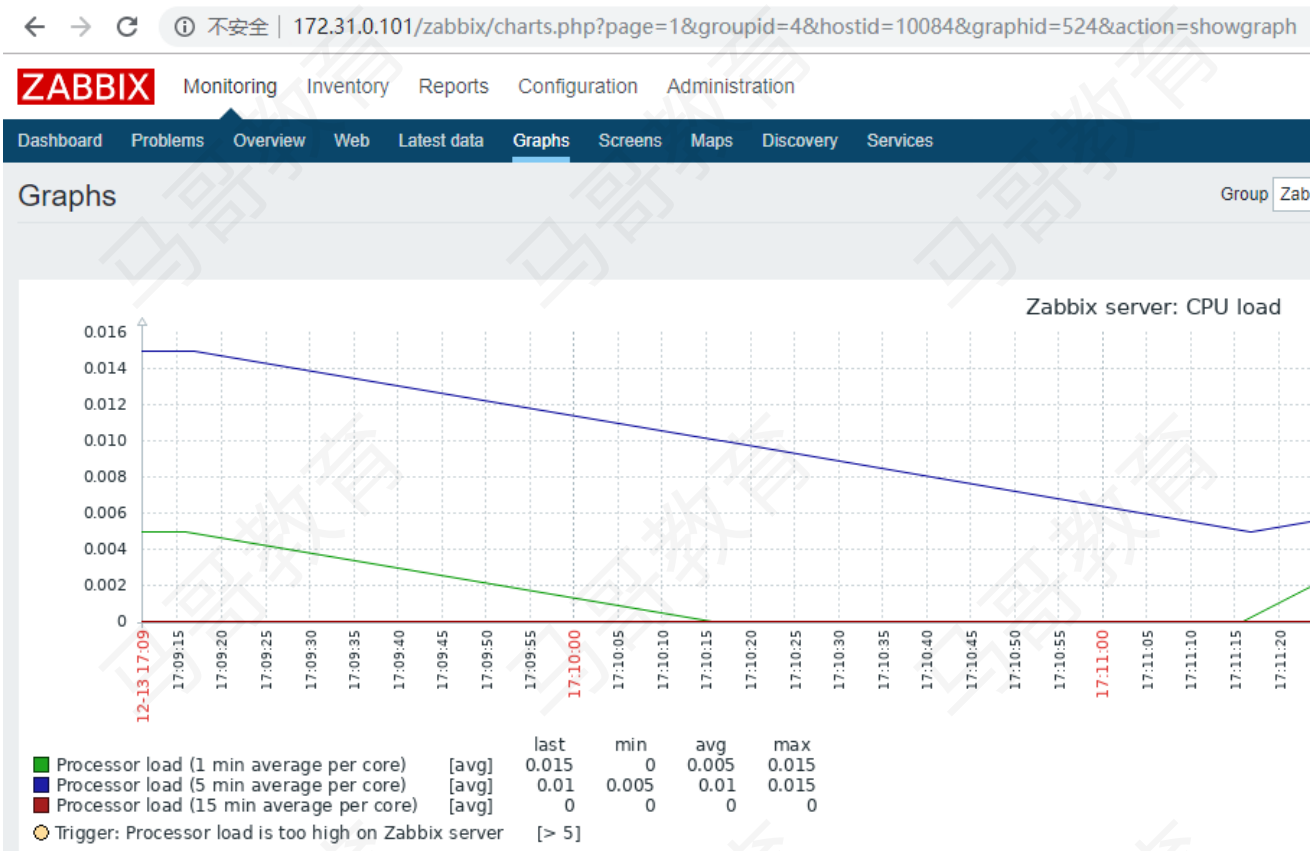
2.2.1.12：配置完成：



2.2.1.13：登录界面：



2.2.1.14：zabbix登录后的监控图形界面：



2.2.2：yum安装zabbix：

略。。。。。。

2.3：编译安装zabbix：

编译安装zabbix server与agent

2.3.1：下载源码包：

https://jaist.dl.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable #下载地址

```
# pwd
/usr/local/src
# groupadd  -g 1001 zabbix #创建zabbix用户和组
# useradd  -u 1001 -g 1001 zabbix
# useradd  -u 1001 -g 1001 zabbix #验证用户id

# wget
https://jaist.dl.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/4.0.15/zabbix-4.0.15.tar.gz
# tar xf zabbix-4.0.15.tar.gz
# cd zabbix-4.0.15/
```

2.3.2：解决依赖关系：

Centos：

```
# yum install gcc  libxml2-devel   net-snmp net-snmp-devel   curl  curl-devel php  php-bcmath  php-mbstring mariadb mariadb-devel  -y
```

Ubuntu：

```
# apt update
# apt-get  install   apache2 apache2-bin apache2-data apache2-utils fontconfig-config fonts-dejavu-core fping libapache2-mod-php   libapache2-mod-php7.2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libfontconfig1 libgd3 libiksemel3   libjbig0 libjpeg-turbo8 libjpeg8 liblua5.2-0 libodbc1 libopenipmi0 libsensors4 libsnmp-base libsnmp30 libsodium23 libssh2-1  libtiff5 libwebp6 libxpm4 php-bcmath php-common php-gd php-ldap php-mbstring php-mysql php-xml php7.2-bcmath php7.2-cli  php7.2-common php7.2-gd php7.2-json php7.2-ldap php7.2-mbstring php7.2-mysql php7.2-opcache php7.2-readline  php7.2-xml snmpd  ssl-cert ttf-dejavu-core     libmysqlclient-dev  libxml2-dev libxml2 snmp  libsnmp-dev   libevent-dev  openjdk-8-jdk curl libcurl4-openssl-dev
```

2.3.3：编译安装Zabbix：

```
# ./configure --prefix=/apps/zabbix_server  --enable-server --enable-agent --with-mysql --with-net-snmp --with-libcurl --with-libxml2  --enable-java
# make install
```

```
root@zabbix-server:/usr/local/src/zabbix-4.0.15# ./configure --prefix=/apps/zabbix_server
t-snmp --with-libcurl --with-libxml2  --enable-java
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
configure: Configuring Zabbix 4.0.15
checking whether make sets $(MAKE)... (cached) yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for cc... cc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether cc accepts -g... yes
checking for cc option to accept ISO C89... none needed
checking whether cc understands -c and -o together... yes
checking for style of include used by make... GNU
```

2.3.4：准备数据库：

```
# apt update
# apt install mysql-server mysql-client
# vim  /etc/mysql/mysql.conf.d/mysqld.cnf
    bind-address            = 0.0.0.0
# systemctl  restart mysql

root@zabbix-mysql-master:~# mysql #进入数据库创建账户并授权
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.28-0ubuntu0.18.04.4 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database zabbix_server character set utf8 collate utf8_bin;
Query OK, 1 row affected (0.00 sec)

mysql> grant all privileges on zabbix_server.* to zabbix@"172.31.0.%" identified by
'magedu.zabbix';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

2.3.5：导入数据库：

```
# apt  install mysql-client #安装mysql客户端命令，用于测试zabbix 数据库账号权限
# mysql -uzabbix  -pmagedu.zabbix  -h172.31.0.104 #测试权限
# pwd
/usr/local/src/zabbix-4.0.15/database/mysql

# cd  database/mysql/
# mysql -uzabbix -pmagedu.zabbix -h172.31.0.104 zabbix_server < schema.sql
# mysql -uzabbix -pmagedu.zabbix -h172.31.0.104 zabbix_server < images.sql
# mysql -uzabbix -pmagedu.zabbix -h172.31.0.104 zabbix_server < data.sql
```

2.3.6：数据库验证数据：

```
mysql> use zabbix_server;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------------+
| Tables_in_zabbix_server    |
+----------------------------+
| acknowledges               |
| actions                    |
| alerts                     |
| application_discovery      |
| application_prototype      |
| application_template       |
| applications               |
| auditlog                   |
| auditlog_details           |
| autoreg_host               |
| conditions                 |
| config                     |
| corr_condition             |
| corr_condition_group       |
```

2.3.7：编辑zabbix server配置文件：

```
# vim /apps/zabbix_server/etc/zabbix_server.conf
# grep "^[a-Z]" /apps/zabbix_server/etc/zabbix_server.conf
LogFile=/tmp/zabbix_server.log
DBHost=172.31.0.104
DBName=zabbix_server
DBUser=zabbix
DBPassword=magedu.zabbix
DBPort=3306
Timeout=4
LogSlowQueries=3000
```

2.3.8：启动zabbix server：

```
# /apps/zabbix_server/sbin/zabbix_server  -c /apps/zabbix_server/etc/zabbix_server.conf

# tail  /tmp/zabbix_server.log
 54340:20191213:182423.876 server #32 started [preprocessing worker #2]
 54341:20191213:182423.876 server #33 started [preprocessing worker #3]
 54339:20191213:182423.976 server #31 started [preprocessing worker #1]
```

2.3.9：配置web界面：

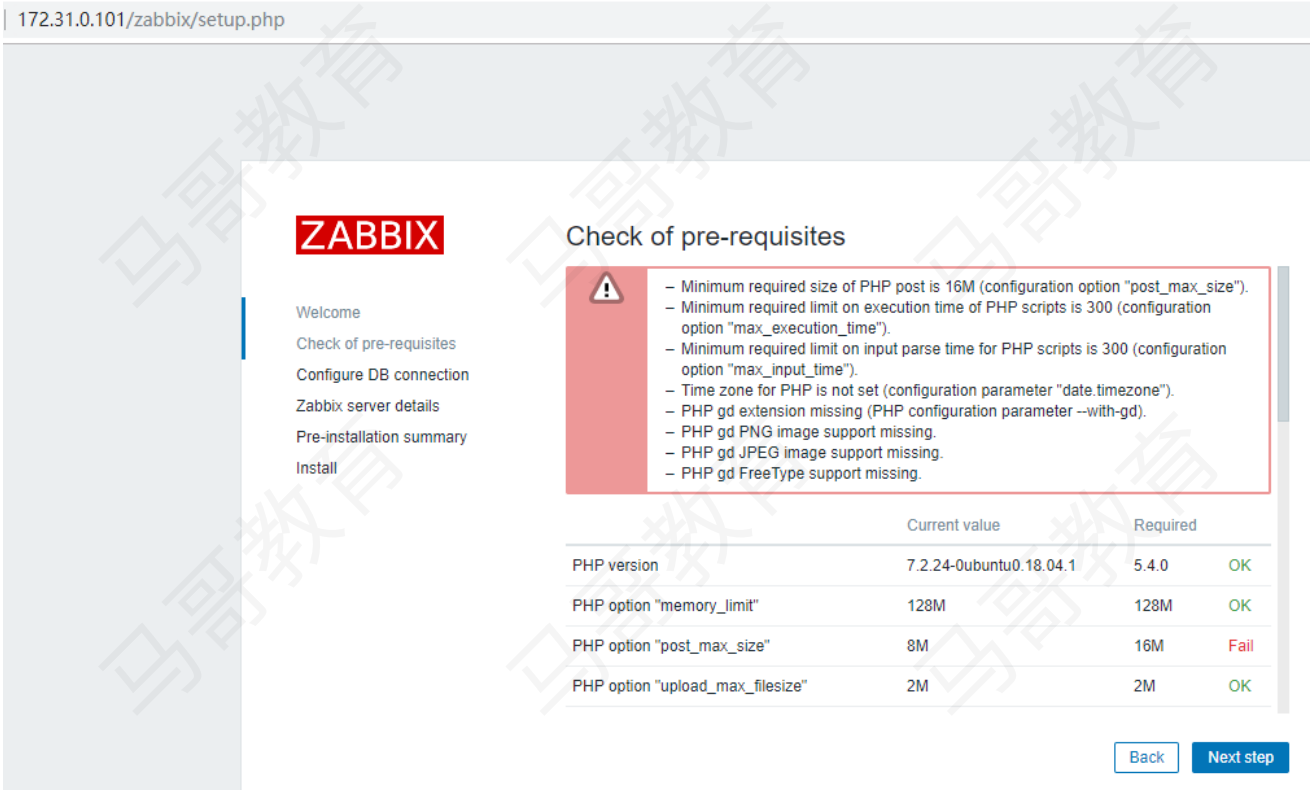
2.3.9.1：拷贝web界面程序：

https://www.zabbix.com/documentation/4.0/zh/manual/installation/install

```
# mkdir  /var/www/html/zabbix
# pwd
/usr/local/src/zabbix-4.0.15
# cd frontends/php/
# cp -a . /var/www/html/zabbix/
```

2.3.9.2：访问web界面：

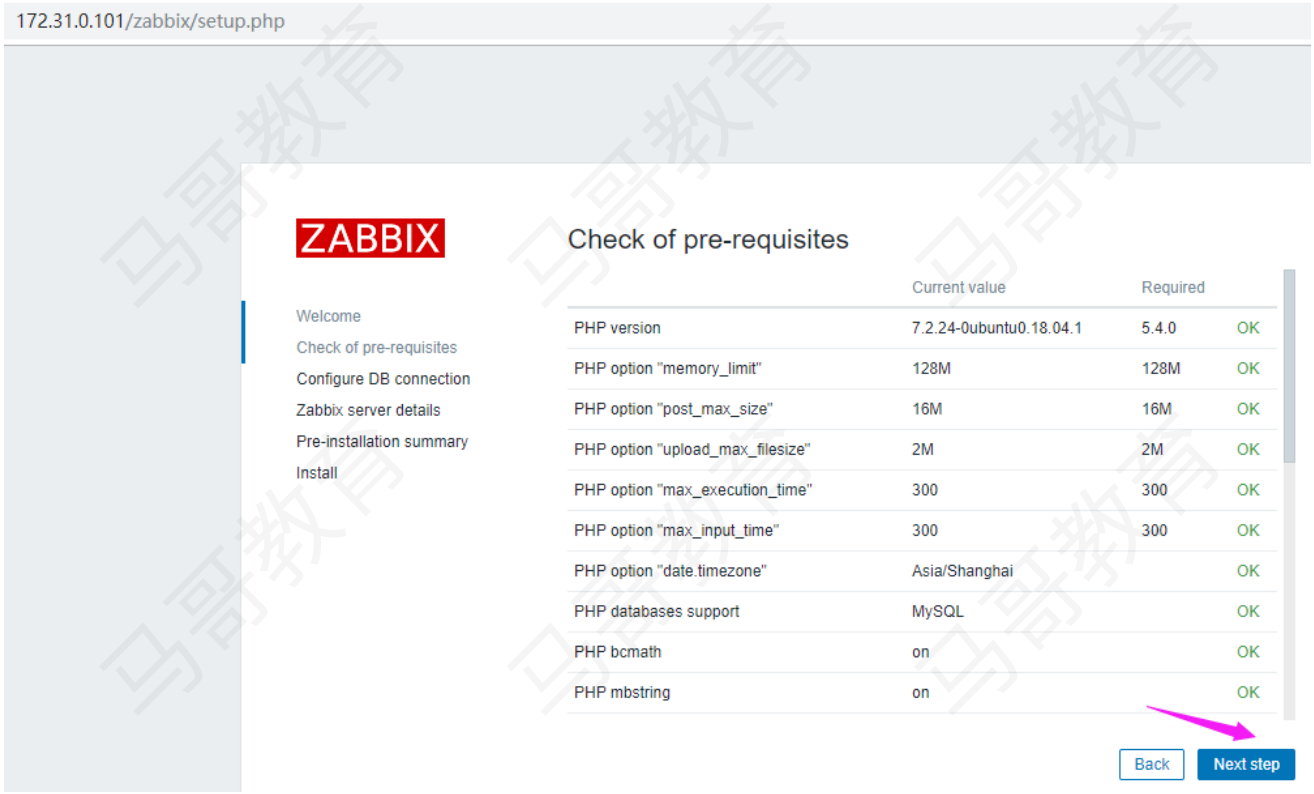
2.3.9.3：当前报错界面：



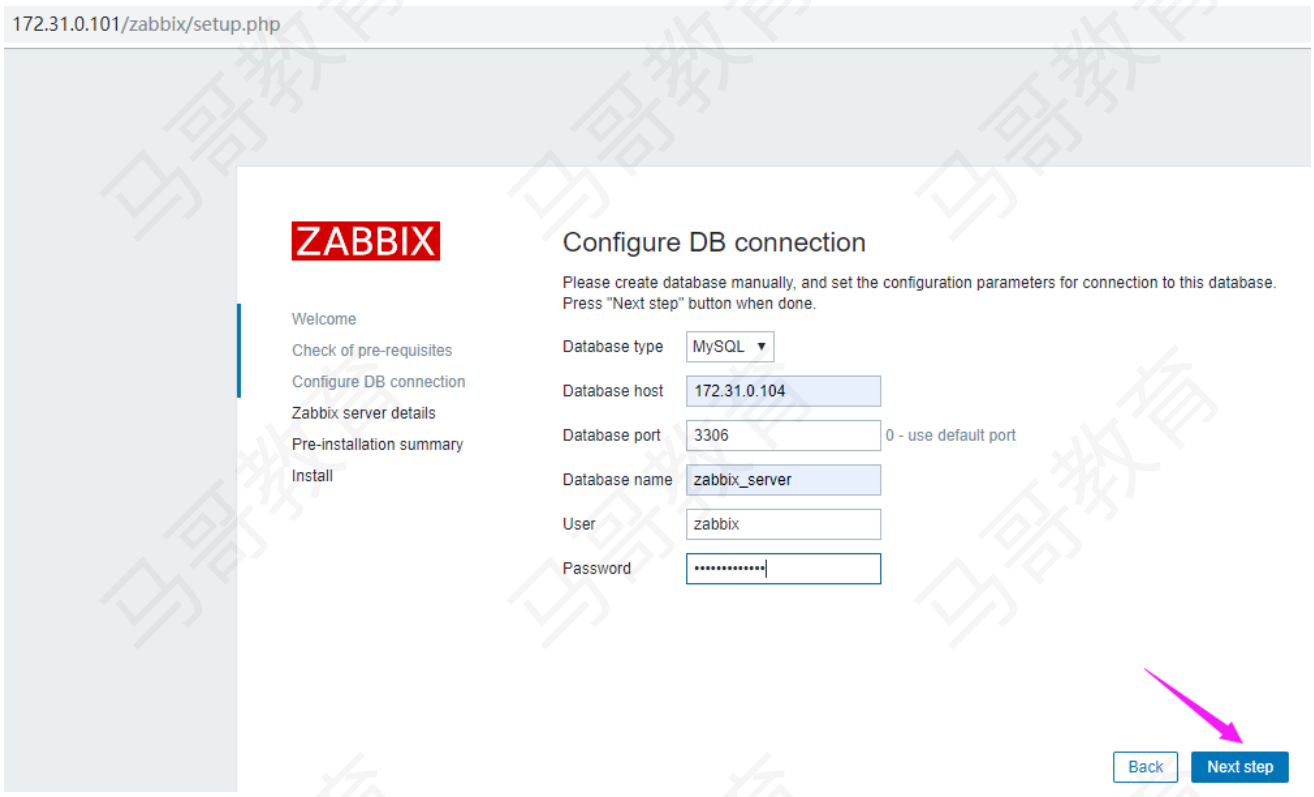
2.3.9.4：解决报错：

```
# apt-get   install php-gettext   php-xml php-net-socket php-gd php-mysql
# systemctl  restart apache2

# vim /etc/php/7.2/apache2/php.ini
# systemctl  restart apache2
```

2.3.9.5：验证当前界面：

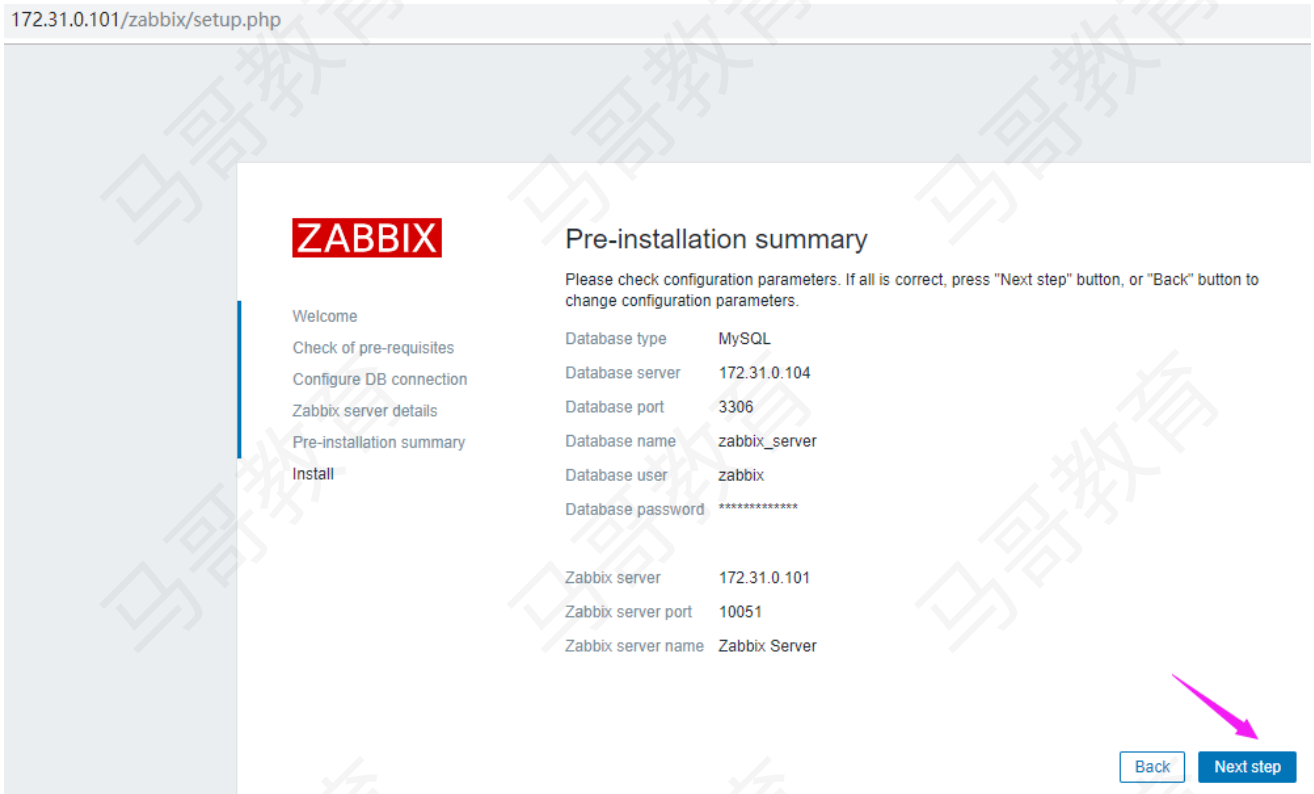


2.3.9.6：数据库配置：



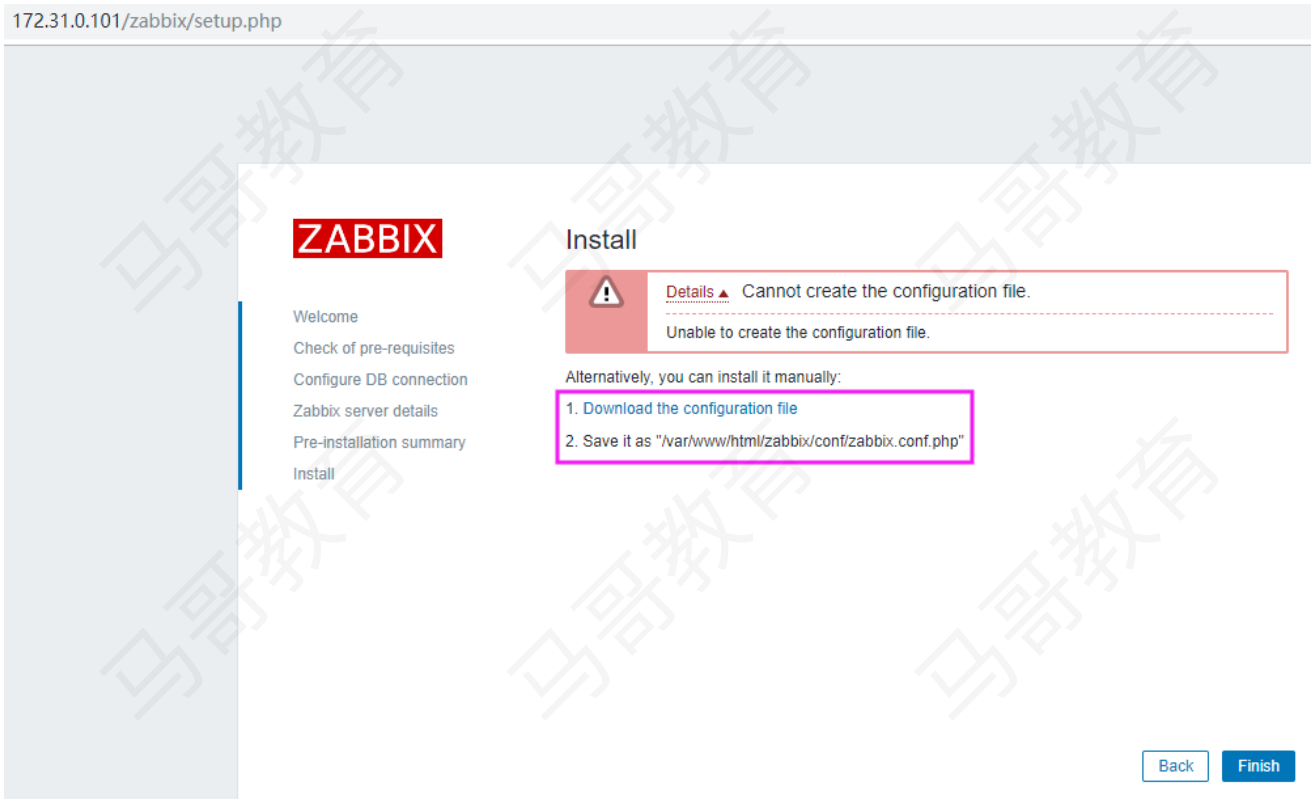
2.3.9.7：zabbix server配置：

2.3.9.8：信息确认：



2.3.9.9：创建配置文件：

需要手动下载配置文件并上传至zabbix server的/var/www/html/zabbix/conf/zabbix.conf.php路径



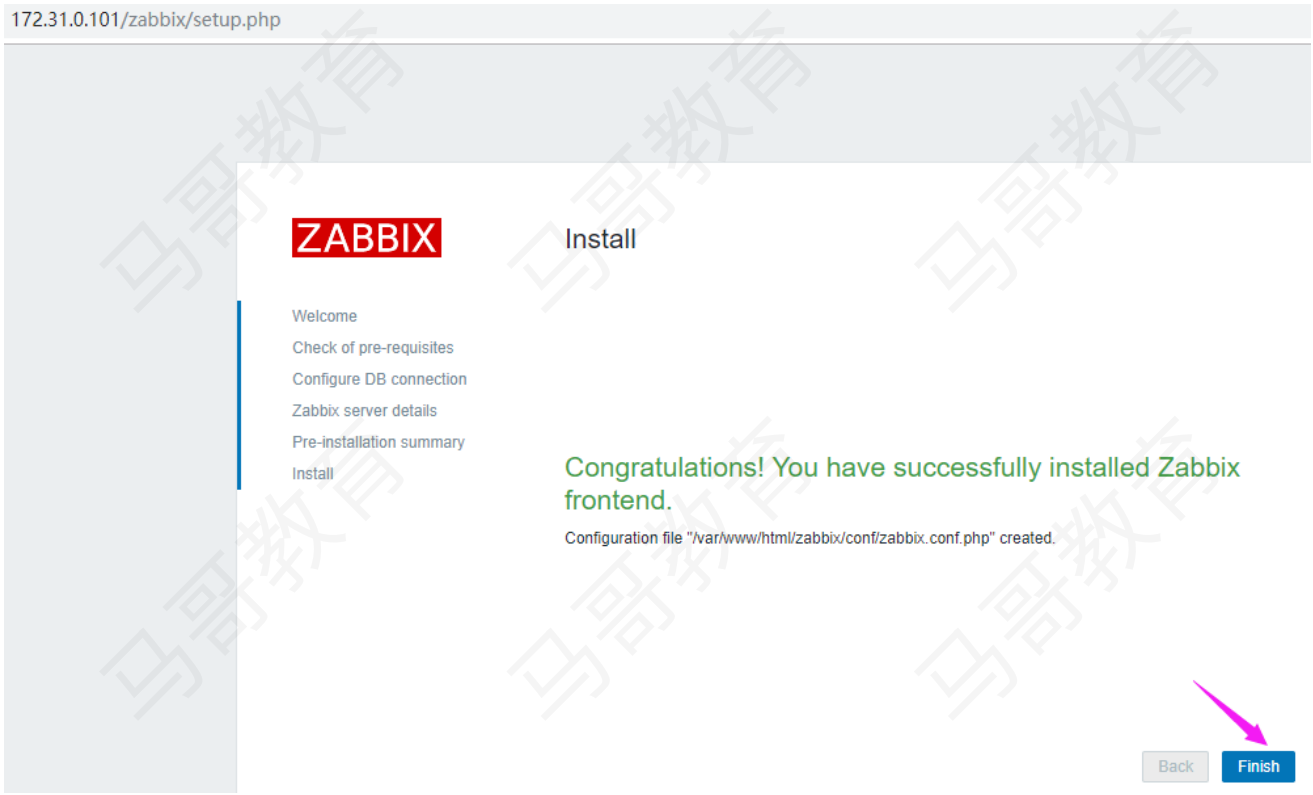
2.3.9.10：验证zabbix server配置文件：

手动上传到服务器指定路径并验证文件内容

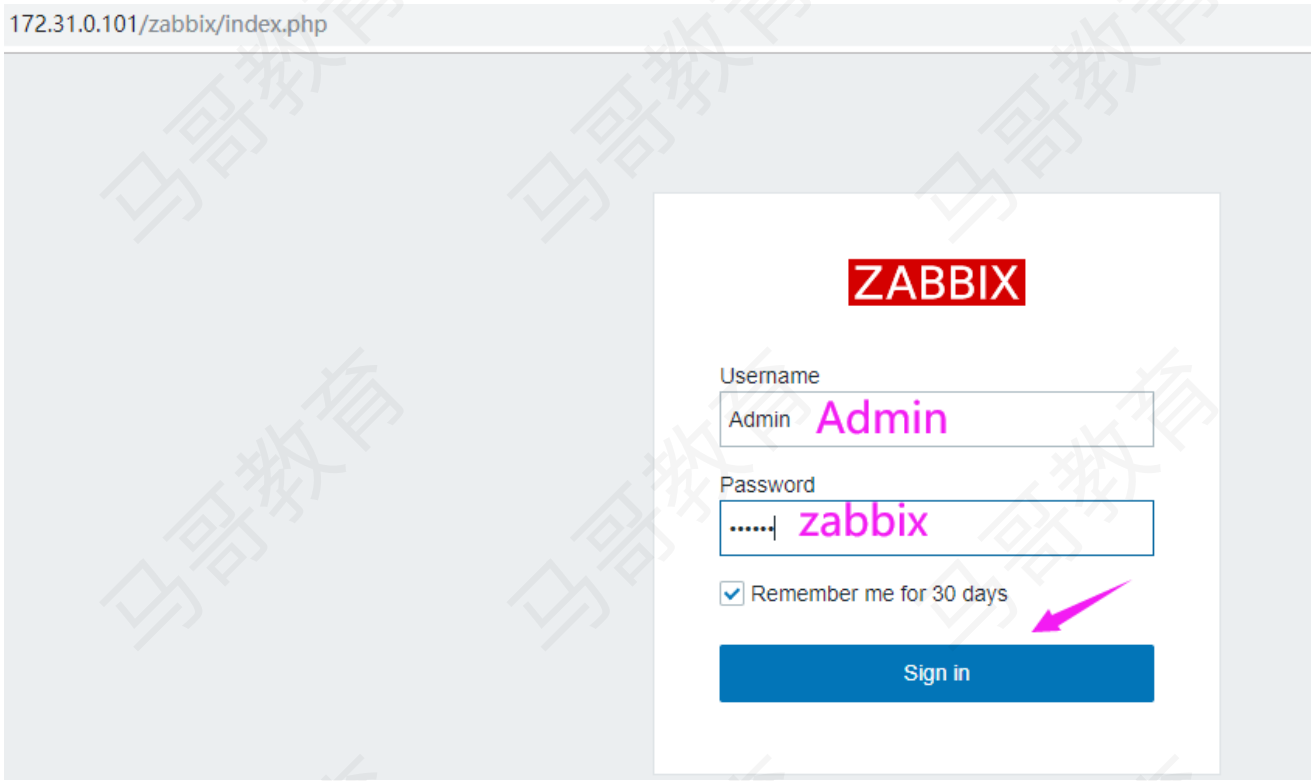
```
# cat /var/www/html/zabbix/conf/zabbix.conf.php
<?php
// Zabbix GUI configuration file.
global $DB;

$DB['TYPE']     = 'MYSQL';
$DB['SERVER']   = '172.31.0.104';
$DB['PORT']     = '3306';
$DB['DATABASE'] = 'zabbix_server';
$DB['USER']     = 'zabbix';
$DB['PASSWORD'] = 'magedu.zabbix';

// Schema name. Used for IBM DB2 and PostgreSQL.
$DB['SCHEMA'] = '';

$ZBX_SERVER      = '172.31.0.101';
$ZBX_SERVER_PORT = '10051';
$ZBX_SERVER_NAME = 'Zabbix Server';

$IMAGE_FORMAT_DEFAULT = IMAGE_FORMAT_PNG;
```

2.3.9.11：刷新页面:

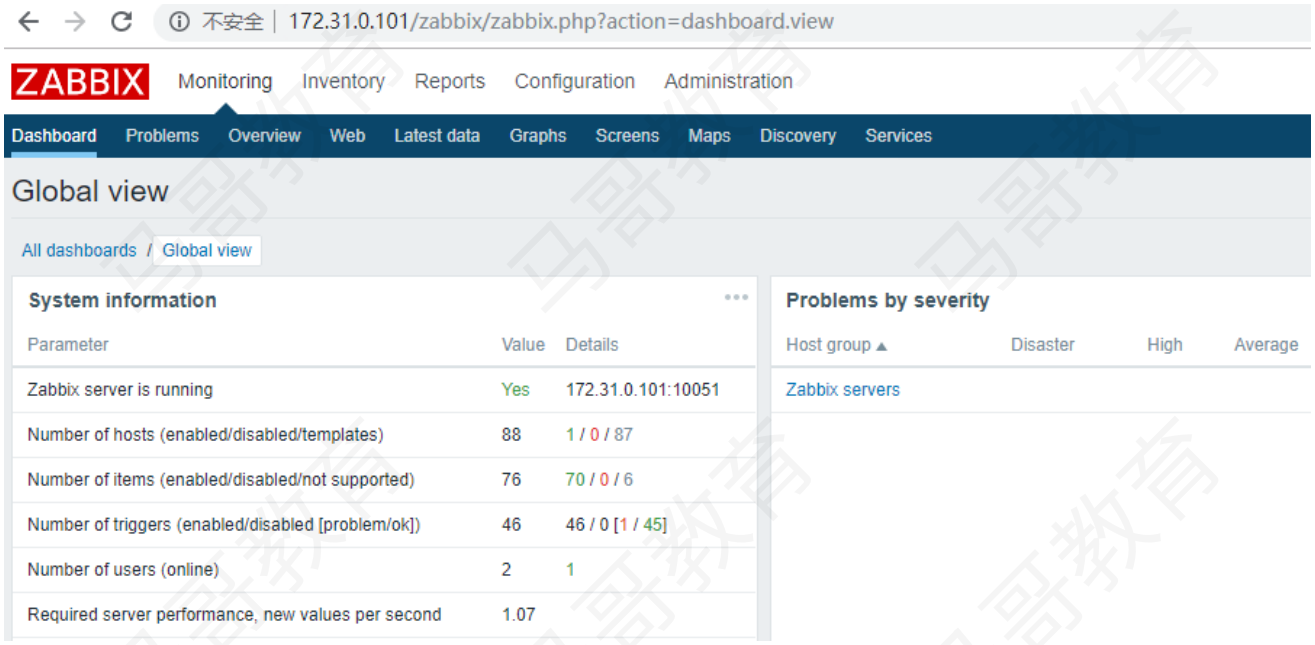
配置文件上传成功后刷新web页面验证是否生效，然后在生效后点finish即可完成zabbix server 的初始化。



2.3.9.12：登录界面



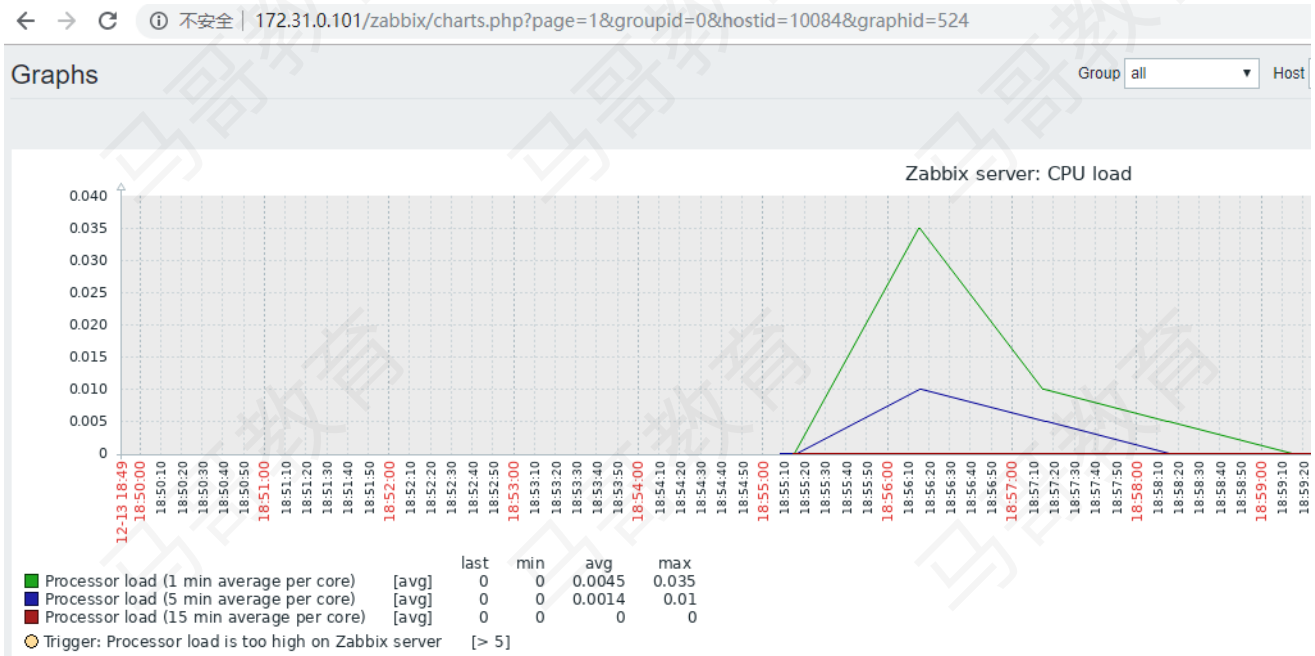
2.3.9.13：zabbix 首页：



2.3.10：启动zabbix agent：

```
# /apps/zabbix_server/sbin/zabbix_agentd
```

2.3.11：验证zabbix 监控数据：



2.3.12：zabbix server与agent启动文件：

通过service启动文件启动zabbix server与zabbix agent。

2.3.12.1：zabbix server启动脚本：

```
#先停止当前zabbix server进程
# pkill zabbix_server
```

```
# cat  /etc/systemd/system/zabbix-server.service
[Unit]
Description=Zabbix Server
After=syslog.target
After=network.target

[Service]
Environment="CONFFILE=/apps/zabbix_server/etc/zabbix_server.conf"
EnvironmentFile=-/etc/default/zabbix-server
Type=forking
Restart=on-failure
PIDFile=/tmp/zabbix_server.pid
KillMode=control-group
ExecStart=/apps/zabbix_server/sbin/zabbix_server -c $CONFFILE
ExecStop=/bin/kill -SIGTERM $MAINPID
RestartSec=10s
TimeoutSec=infinity

[Install]
WantedBy=multi-user.target

# systemctl  restart zabbix-server  && systemctl  enable zabbix-server
```

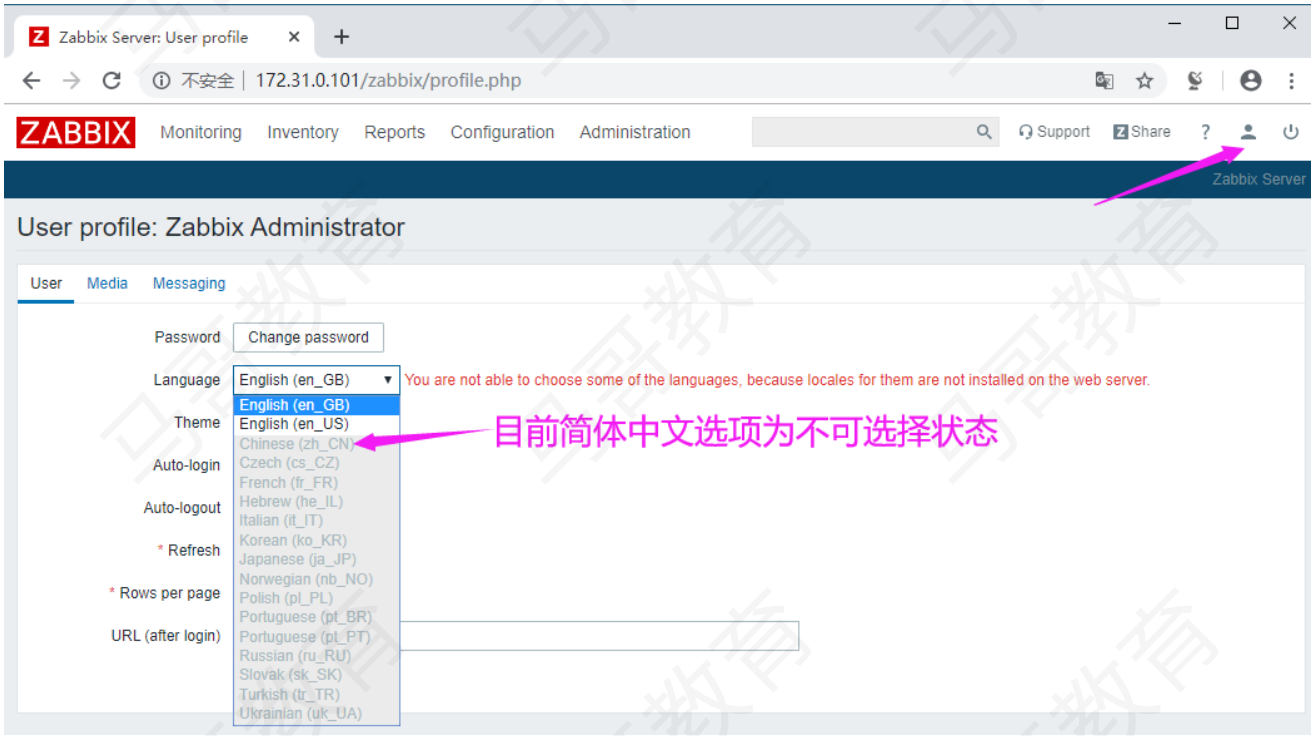
2.3.12.2：zabbix agent启动脚本：

```
#先停止当前zabbix_agent进程
# pkill  zabbix_agentd

# cat /etc/systemd/system/zabbix-agent.service
[Unit]
Description=Zabbix Agent
After=syslog.target
After=network.target

[Service]
Environment="CONFFILE=apps/zabbix_server/etc/zabbix_agentd.conf"
EnvironmentFile=-/etc/default/zabbix-agent
Type=forking
Restart=on-failure
PIDFile=/tmp/zabbix_agentd.pid
KillMode=control-group
ExecStart=/apps/zabbix_server/sbin/zabbix_agentd -c $CONFFILE
ExecStop=/bin/kill -SIGTERM $MAINPID
RestartSec=10s
User=zabbix
Group=zabbix

[Install]
WantedBy=multi-user.target

# systemctl  restart zabbix-agent && systemctl  enable zabbix-agent
```

2.4：Web界面中文菜单环境：

2.4.1：中文选项无法选择：

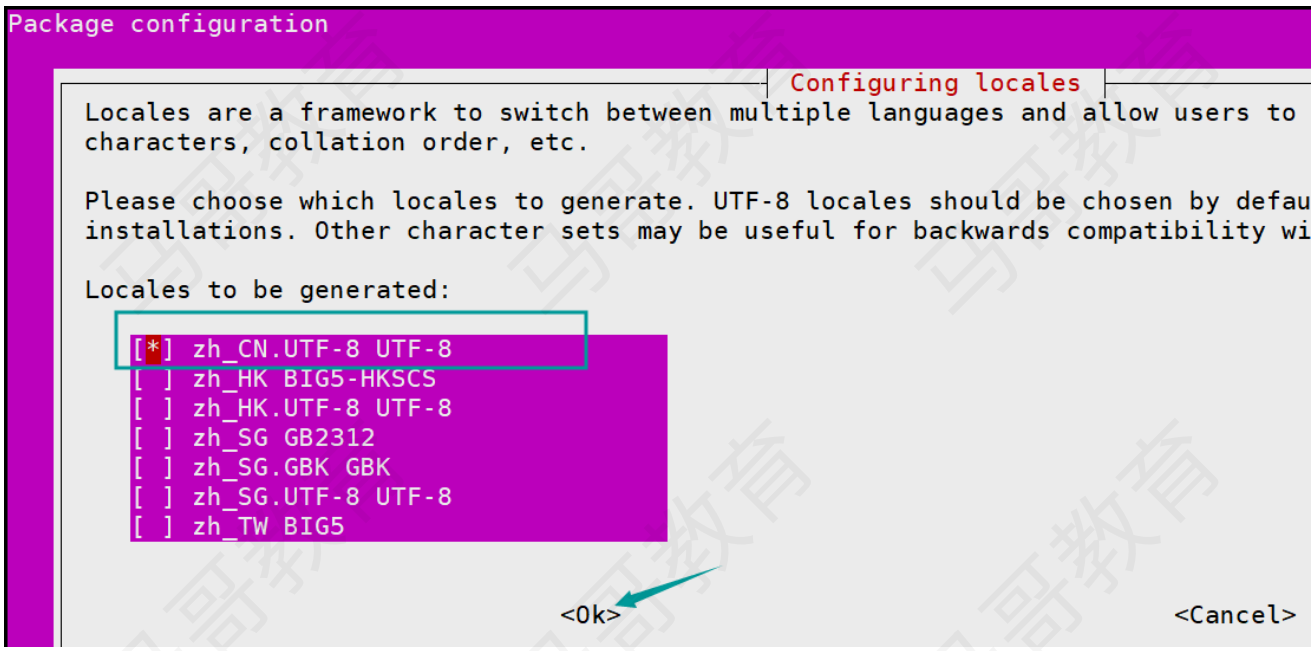
由于ubuntu系统目前未安装中文语言环境所以无法选择中文显示，如下：

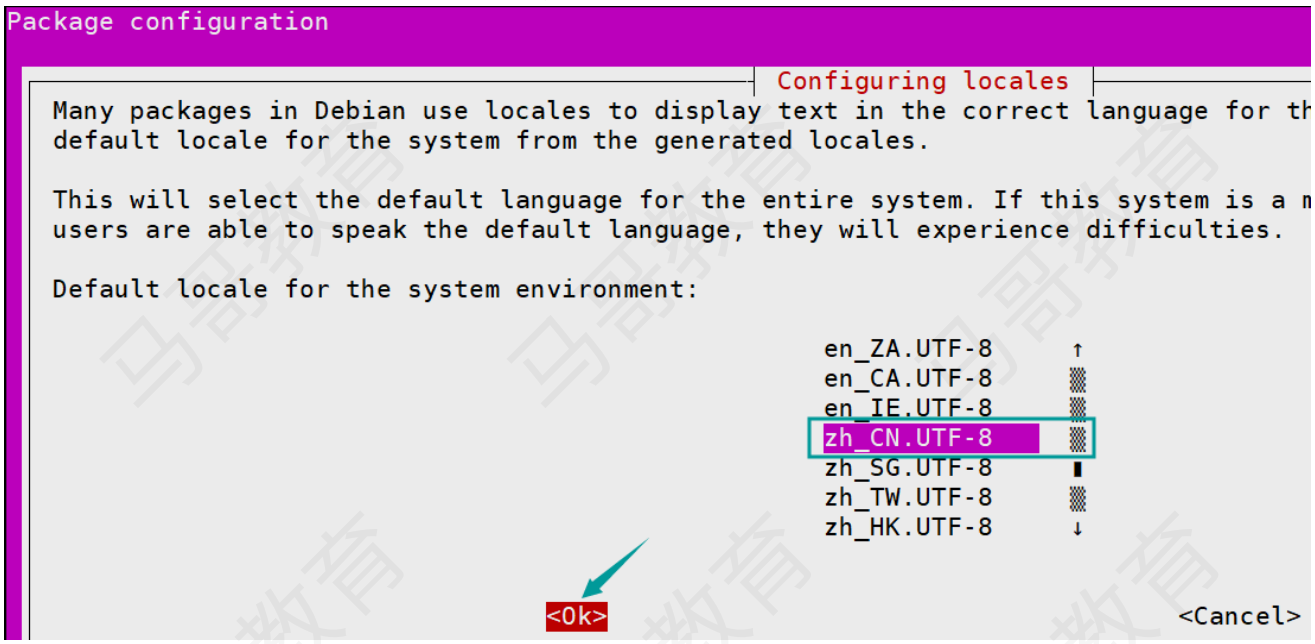


2.4.2：ubuntu系统安装中文语言环境：

安装并设置中文简体语言环境

```
#安装简体中文语言环境
root@zabbix-server:~# sudo apt-get install language-pack-zh*

#增加中文语言环境变量
root@zabbix-server:~# sudo vim /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/
games"
LANG="zh_CN.UTF-8"

#重新设置本地配置
sudo dpkg-reconfigure locales
```


Package configuration
Configuring locales
Locales are a framework to switch between multiple languages and allow users to
characters, collation order, etc.
Please choose which locales to generate. UTF-8 locales should be chosen by defau
installations. Other character sets may be useful for backwards compatibility wi
Locales to be generated:
[*] zh_CN.UTF-8 UTF-8
[ ] zh_HK BIG5-HKSCS
[ ] zh_HK.UTF-8 UTF-8
[ ] zh_SG GB2312
[ ] zh_SG.GBK GBK
[ ] zh_SG.UTF-8 UTF-8
[ ] zh_TW BIG5
<Ok>
<Cancel>

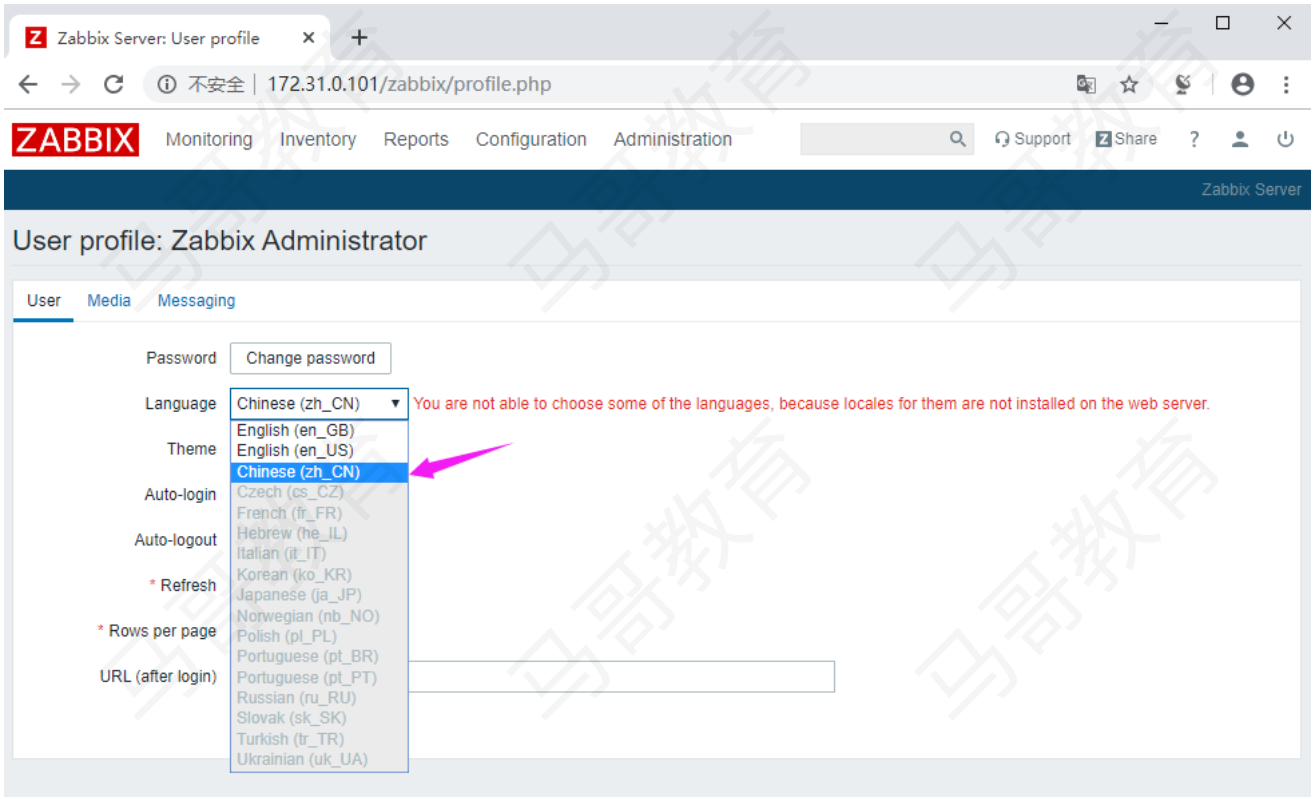

Package configuration
Configuring locales
Many packages in Debian use locales to display text in the correct language for th
default locale for the system from the generated locales.
This will select the default language for the entire system. If this system is a m
users are able to speak the default language, they will experience difficulties.
Default locale for the system environment:
en_ZA.UTF-8
en_CA.UTF-8
en_IE.UTF-8
zh_CN.UTF-8
zh_SG.UTF-8
zh_TW.UTF-8
zh_HK.UTF-8
<Ok>
<Cancel>

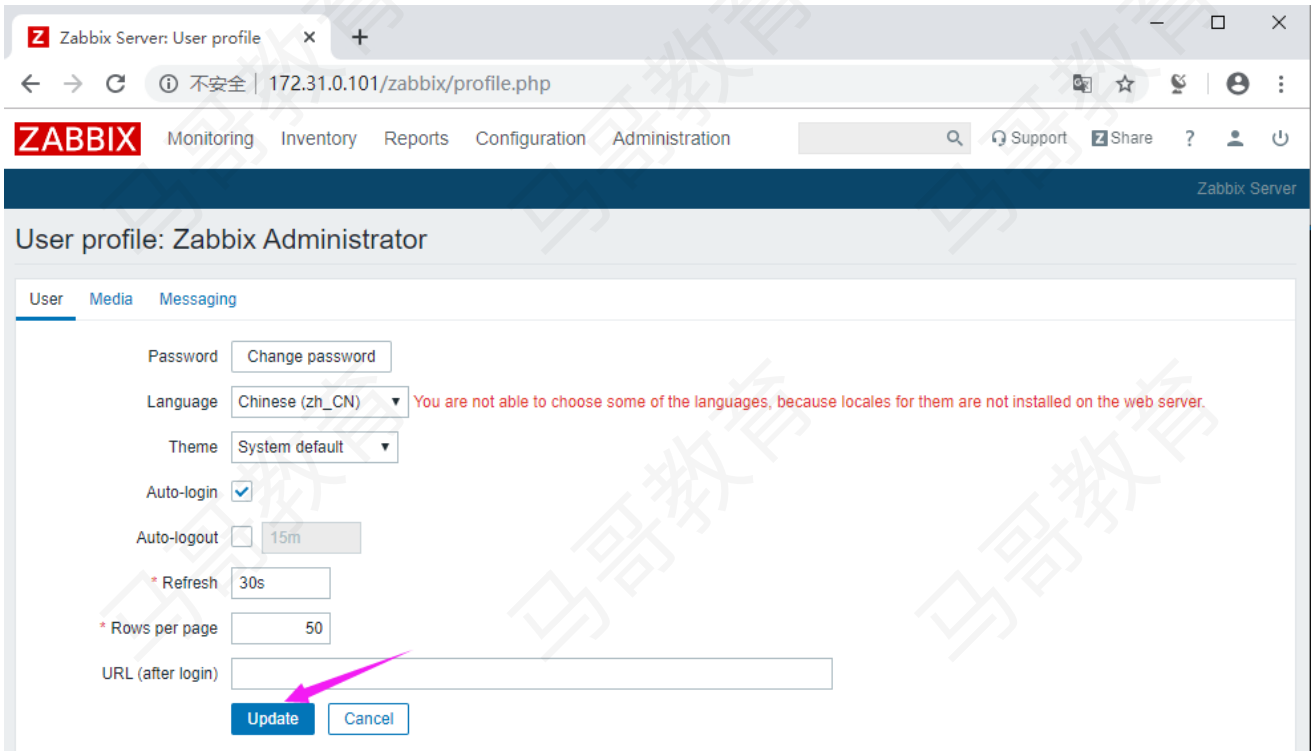
```
Generating locales (this might take a while)...
  en_AG.UTF-8... done
  en_AU.UTF-8... done
  en_BW.UTF-8... done
  en_CA.UTF-8... done
  en_DK.UTF-8... done
  en_GB.UTF-8... done
  en_HK.UTF-8... done
  en_IE.UTF-8... done
  en_IL.UTF-8... done
  en_IN.UTF-8... done
  en_NG.UTF-8... done
  en_NZ.UTF-8... done
  en_PH.UTF-8... done
  en_SG.UTF-8... done
  en_US.UTF-8... done
  en_ZA.UTF-8... done
  en_ZM.UTF-8... done
  en_ZW.UTF-8... done
  zh_CN.UTF-8... done
  zh_HK.UTF-8... done
  zh_SG.UTF-8... done
  zh_TW.UTF-8... done
Generation complete.
root@zabbix-server:~#
```

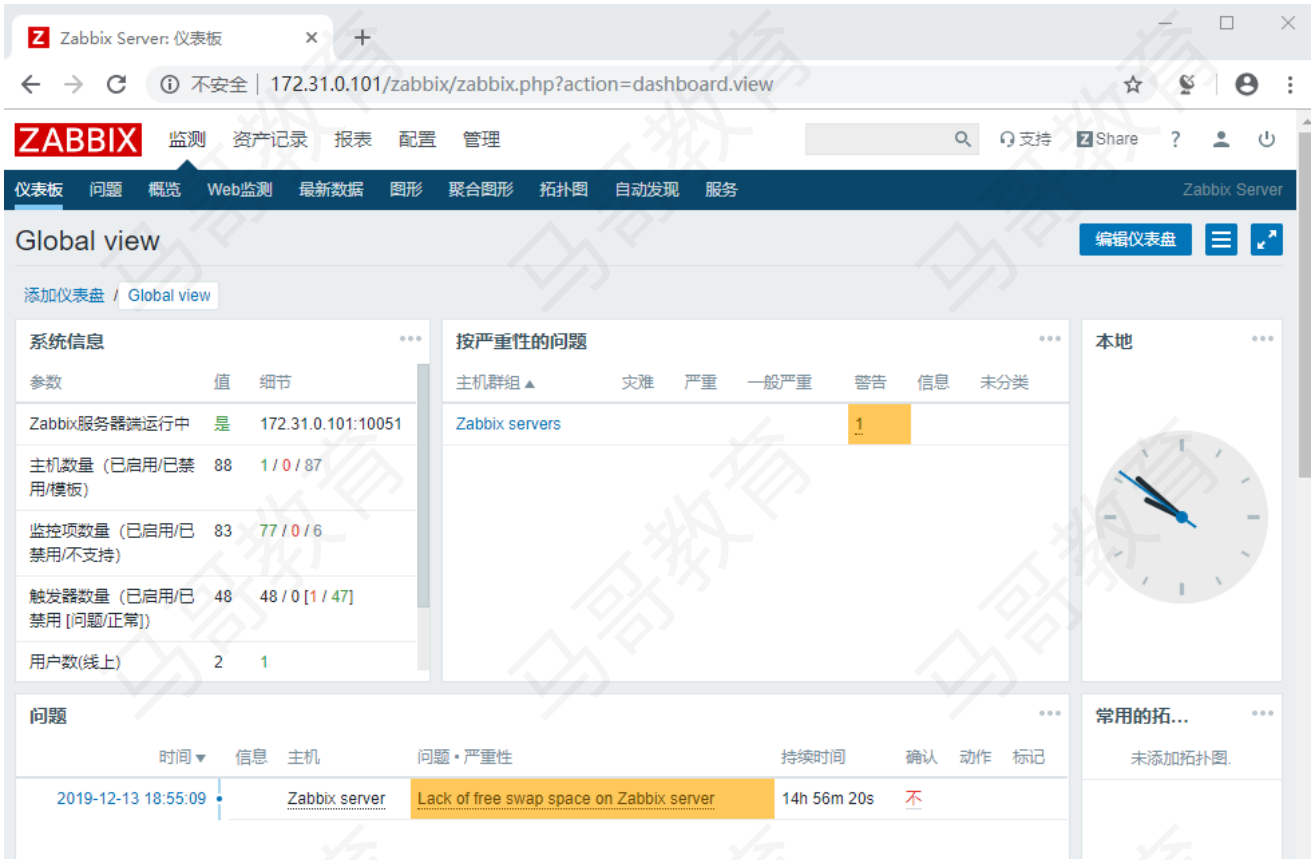
2.4.3：验证web界面：

重启apache并验证wen页面

```
# systemctl  restart apache2
```


Zabbix Server: User profile
不安全 | 172.31.0.101/zabbix/profile.php
ZABBIX
Monitoring
Inventory
Reports
Configuration
Administration
Support
Share
Zabbix Server
User profile: Zabbix Administrator
User
Media
Messaging
Password
Change password
Language
Chinese (zh_CN)
You are not able to choose some of the languages, because locales for them are not installed on the web server.
Theme
Auto-login
Auto-logout
* Refresh
* Rows per page
URL (after login)
English (en_GB)
English (en_US)
Chinese (zh_CN)
Czech (cs_CZ)
French (fr_FR)
Hebrew (he_IL)
Italian (it_IT)
Korean (ko_KR)
Japanese (ja_JP)
Norwegian (nb_NO)
Polish (pl_PL)
Portuguese (pt_BR)
Portuguese (pt_PT)
Russian (ru_RU)
Slovak (sk_SK)
Turkish (tr_TR)
Ukrainian (uk_UA)

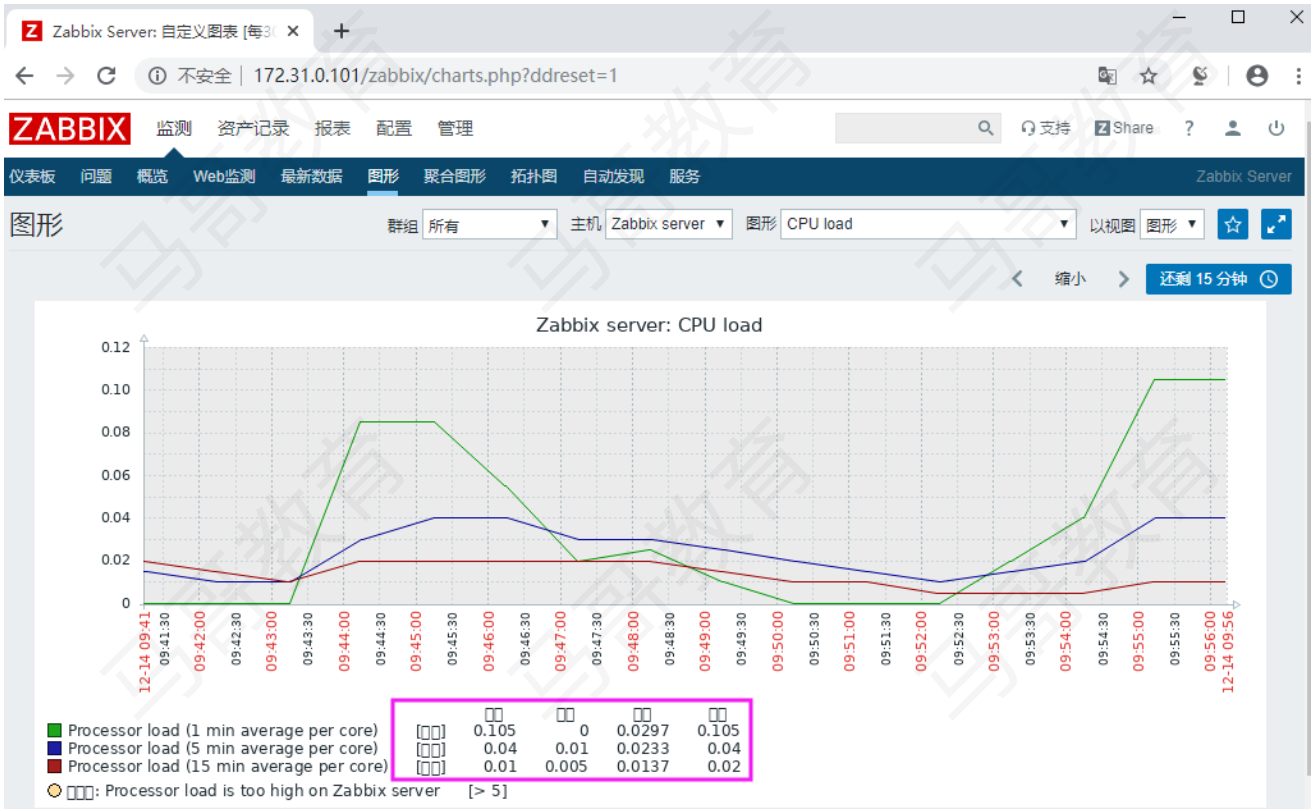

Zabbix Server: User profile
不安全 | 172.31.0.101/zabbix/profile.php
ZABBIX
Monitoring
Inventory
Reports
Configuration
Administration
Support
Share
Zabbix Server
User profile: Zabbix Administrator
User
Media
Messaging
Password
Change password
Language
Chinese (zh_CN)
You are not able to choose some of the languages, because locales for them are not installed on the web server.
Theme
System default
Auto-login
Auto-logout
15m
* Refresh
30s
* Rows per page
50
URL (after login)
Update
Cancel



2.5：监控项与乱码：

2.5.1：图形乱码：

当前系统有些监控项部分显示有乱码，使由于web界面显示为中文但是系统没有相关字体支持，因此需要相关字体的支持才能正常显示，如下：



2.5.2：上传字体文件：

2.5.2.1：在windows拷贝字体：

在Windows上找到控制面板-->字体-->楷体(或者其他个人喜欢的中文字体)，然后将字体拷贝到windows系统其他目录，比如拷贝至windows当前用户的桌面。

2.5.2.2：上传字体到zabbix web目录：

将windows 字体文件上传至zabbix web目录，具体路径为/ZABBIX/WEB/PATH/**assets/fonts/**，如下：

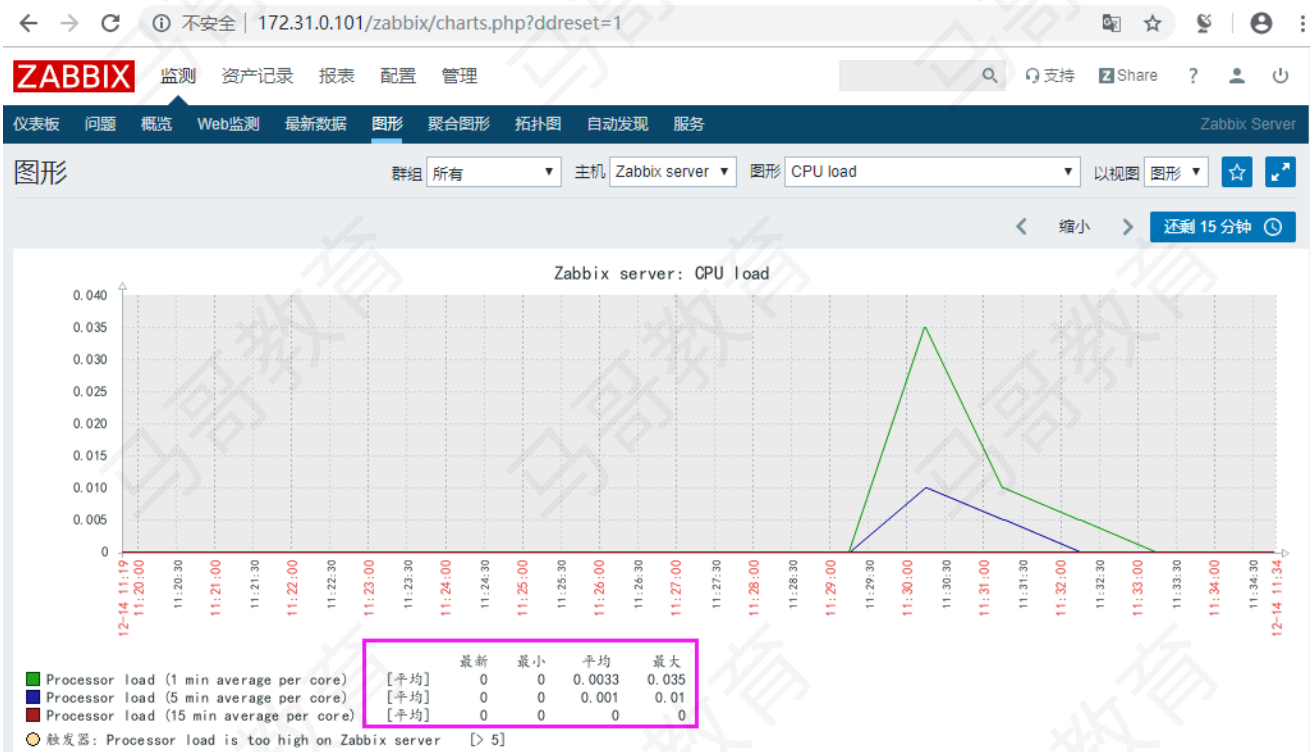
```
# pwd
/var/www/html/zabbix/assets/fonts #上传楷体字体文件到这里

#  chown  zabbix.zabbix ./* #更改权限为zabbix用户和组
# ll
总用量 12260
drwxr-xr-x 2 zabbix zabbix     4096 12月 14 11:28 ./
drwxr-xr-x 5 zabbix zabbix     4096 11月 25 17:06 ../
-rw-r--r-- 1 zabbix zabbix   756072 11月 25 17:05 DejaVuSans.ttf
-rw-r--r-- 1 root   root   11787328 3月   2  2019 simkai.ttf
```

2.5.2.3：修改zabbix文件调用新字体：

```
# pwd
/var/www/html/zabbix
# vim  include/defines.inc.php

#70 define('ZBX_GRAPH_FONT_NAME',          'DejaVuSans'); // font file name
70 define('ZBX_GRAPH_FONT_NAME',          'simkai'); // font file name

#111 define('ZBX_FONT_NAME', 'DejaVuSans');
111 define('ZBX_FONT_NAME', 'simkai');
```

2.5.2.4：验证字体是否生效：

通常不需要重启zabbix及apache，修改后的字体文件即可直接生效。



2.6：zabbix server配置文件详解：

见 **zabbix_server.conf配置文件详解**

三：Zabbix 监控入门基础：

学习通过apt/yum安装zabbix agent、对tomcat进行监控。

3.1：监控linux系统：

在其他linux服务器安装zabbix agent，然后添加到zabbix server以对其进行资源监控。

3.1.1：zabbix agent安装：

```
# wget https://repo.zabbix.com/zabbix/4.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.0-3+bionic_all.deb
# dpkg -i zabbix-release_4.0-3+bionic_all.deb
# apt update
# apt  install  zabbix-agent
```

3.1.2：zabbix agent配置文件：

```
# vim /etc/zabbix/zabbix_agentd.conf
##### Passive checks related #被动检查相关配置
Server=172.31.0.101 #指向当前zabbix server

### Option: ListenPort
ListenPort=10050 #监听端口

### Option: StartAgents
StartAgents=3 #被动状态时默认启动的实例数(进程数)，为0不监听任何端口

### Option: Hostname
Hostname=172.31.0.107 #区分大小写且在zabbix server唯一的值
```

3.1.3：重启zabbix agent：

```
# systemctl  restart zabbix-agent
# systemctl  status  zabbix-agent
● zabbix-agent.service - Zabbix Agent
   Loaded: loaded (/lib/systemd/system/zabbix-agent.service; disabled; vendor preset: enabled)
   Active: active (running) since Sat 2019-12-14 12:06:17 CST; 5s ago
  Process: 2247 ExecStop=/bin/kill -SIGTERM $MAINPID (code=exited, status=0/SUCCESS)
  Process: 2248 ExecStart=/usr/sbin/zabbix_agentd -c $CONFFILE (code=exited, status=0/SUCCESS)
 Main PID: 2250 (zabbix_agentd)
    Tasks: 6 (limit: 2290)
   CGroup: /system.slice/zabbix-agent.service
           ├─2250 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
           ├─2252 /usr/sbin/zabbix_agentd: collector [idle 1 sec]
```

```
            ├─2253 /usr/sbin/zabbix_agentd: listener #1 [waiting for connection]
            ├─2254 /usr/sbin/zabbix_agentd: listener #2 [waiting for connection]
            ├─2255 /usr/sbin/zabbix_agentd: listener #3 [waiting for connection]
            └─2256 /usr/sbin/zabbix_agentd: active checks #1 [idle 1 sec]
```

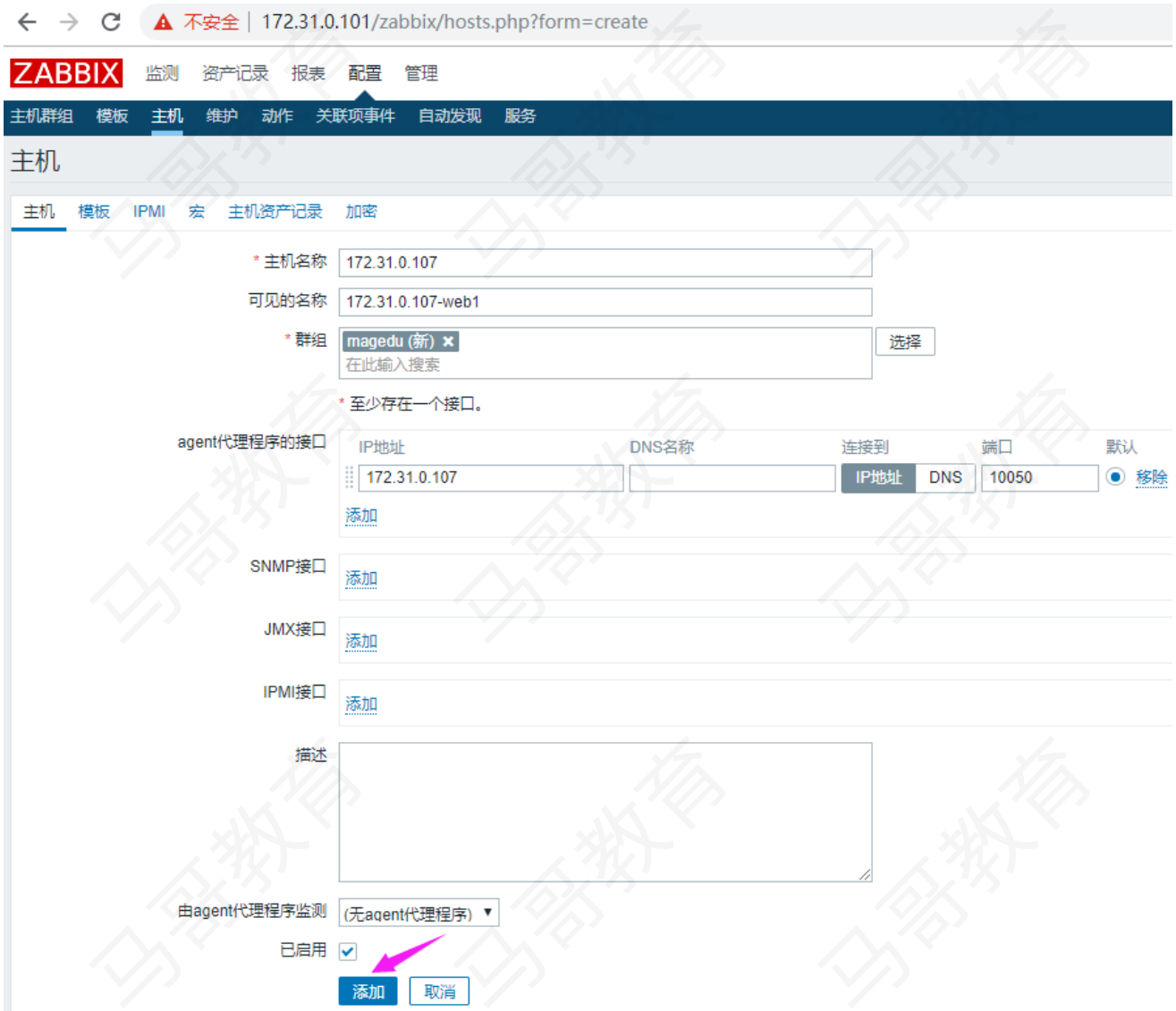
3.1.4：验证zabbix agent：

验证zabbix agent进程是否存在、端口是否监听以及日志是否有异常

```
root@zabbix-node4:~# ps -ef | grep zabbix
zabbix     2250      1  0 12:06 ?        00:00:00 /usr/sbin/zabbix_agentd -c
/etc/zabbix/zabbix_agentd.conf
zabbix     2252   2250  0 12:06 ?        00:00:00 /usr/sbin/zabbix_agentd: collector
[idle 1 sec]
zabbix     2253   2250  0 12:06 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #1
[waiting for connection]
zabbix     2254   2250  0 12:06 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #2
[waiting for connection]
zabbix     2255   2250  0 12:06 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #3
[waiting for connection]
zabbix     2256   2250  0 12:06 ?        00:00:00 /usr/sbin/zabbix_agentd: active
checks #1 [idle 1 sec]
root       2261   1466  0 12:07 pts/0    00:00:00 grep --color=auto zabbix

root@zabbix-node4:~# pstree -p 2250
zabbix_agentd(2250)─┬─zabbix_agentd(2252)
                    ├─zabbix_agentd(2253)
                    ├─zabbix_agentd(2254)
                    ├─zabbix_agentd(2255)
                    └─zabbix_agentd(2256)

root@zabbix-node4:~# ss -tnl | grep 10050
LISTEN   0         128                0.0.0.0:10050            0.0.0.0:*
LISTEN   0         128                   [::]:10050               [::]:*

root@zabbix-node4:~# tail  -n5 /var/log/zabbix/zabbix_agentd.log
  2253:20191214:120617.791 agent #2 started [listener #1]
  2254:20191214:120617.792 agent #3 started [listener #2]
  2256:20191214:120617.794 agent #5 started [active checks #1]
  2255:20191214:120617.794 agent #4 started [listener #3]
```

3.1.5：zabbix web界面添加被监控主机：

在zabbix web管理界面添加上一步安装了zabbix agent的linux主机。



3.1.6：关联监控模板：

在zabbix对主机实现监控，可以通过添加模板或者添加监控项实现对指定的监控目标进行数据采集，通常采用先创建模板，然后再将模板关联至主机的方式，模板关联如下：

3.1.6.1：选择模板：

选择符合当前主机监控项目的模板，如对linux系统监控可以关联zabbix 自带的Template OS Linux模板，如下在搜索框输入linux关键字后会自带匹配符合名称的模板，直接点击即可选择：

3.1.6.2：确认模板选择：

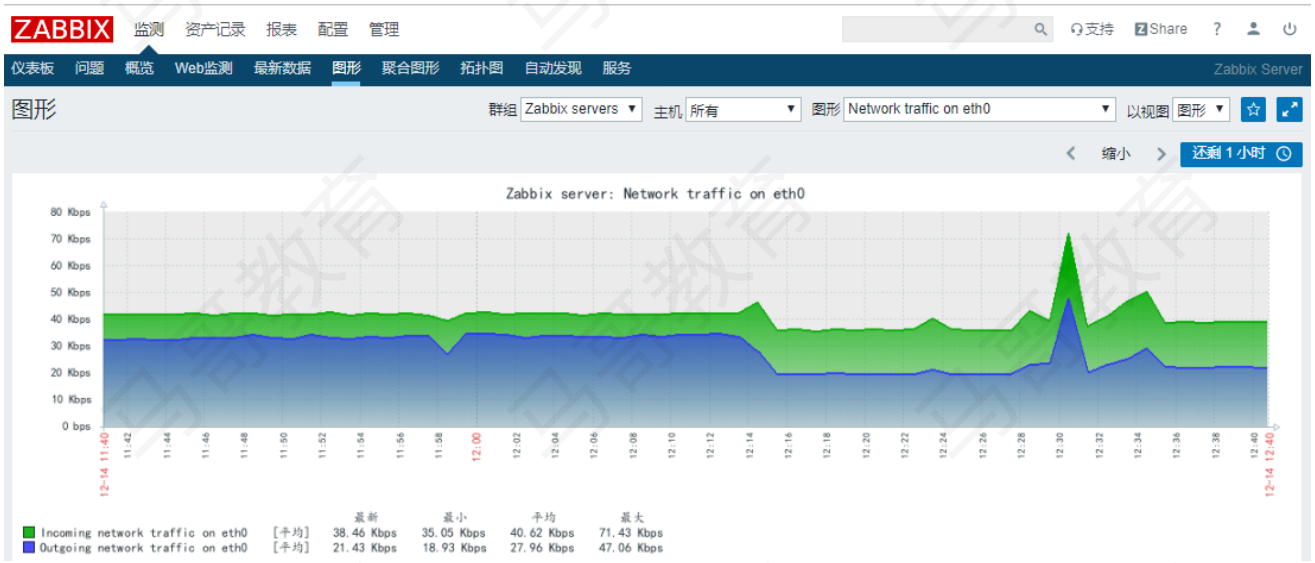
3.1.6.3：确认添加主机：

3.1.6.4：主机添加完成：

需要等待几分钟(具体看模板中监控项的数据采集间隔时间)之后，主机的状态才会变成绿色或者红色，绿色表示zabbix server对zabbix agent有权限进行数据采集并且当前通信正常，红色则表示通信异常，通信异常会有多种原因，具体可以看相关日志或者到时候点击红色方框会有相关报错显示。

3.1.6.5：验证主机监控数据：

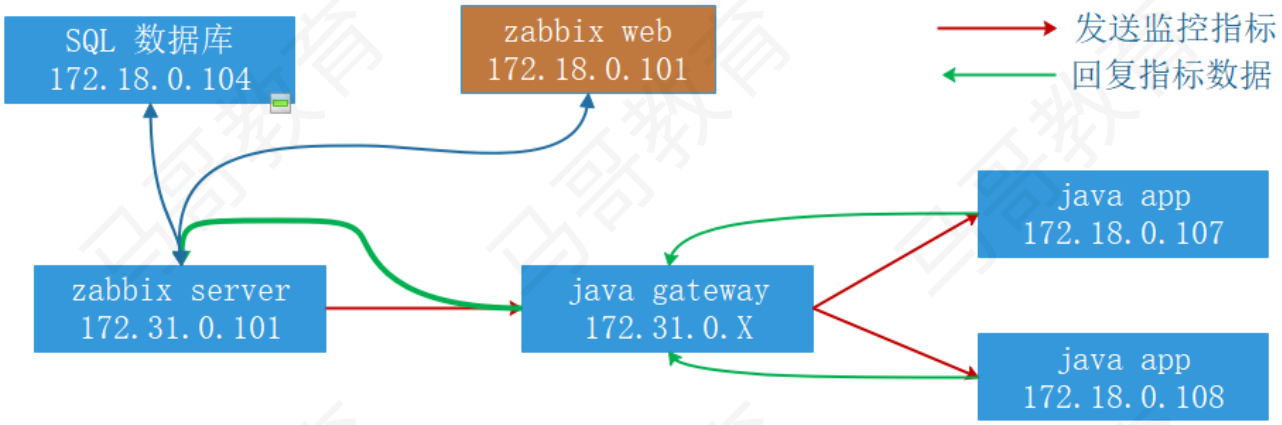
经过一段时间的数据采集后，验证zabbix server是否有刚添加完成主机的监控数据和图形，图形显示如下：



至此，针对linux系统的zabbix agent安装、添加主机和监控就简单完成了。

3.2：监控tomcat：

学习如何通过java gateway实现对tomcat的指标进行数据采集和图形展示，如堆栈内存利用率、当前会话连接数、繁忙线程等。



3.2.1：准备JDK环境：

```
# pwd
/usr/local/src
# tar xf jdk-8u221-linux-x64.tar.gz

# ln -sv /usr/local/src/jdk1.8.0_221 /usr/local/jdk
'/usr/local/jdk' -> '/usr/local/src/jdk1.8.0_221'

# vim /etc/profile
export JAVA_HOME=/usr/local/jdk
export TOMCAT_HOME=/apps/tomcat
export PATH=$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$TOMCAT_HOME/bin:$PATH
export CLASSPATH=.$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib:$JAVA_HOME/lib/tools.jar

# source  /etc/profile
# java -version
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)
```

3.2.2：准备tomcat：

```
#解压tomcat压缩包
# cd /apps/
# tar xf apache-tomcat-8.0.38.tar.gz

#创建测试app
# cd /apps/apache-tomcat-8.0.38/
# mkdir  webapps/magedu
# cat  webapps/magedu/index.html
tomcat web page

#启动tomcat
# /apps/apache-tomcat-8.0.38/bin/catalina.sh  start
```

3.2.3：验证tomcat web页面：

确认tomcat服务运行及访问正常

3.2.4：部署java gateway服务器：

java gateway是一个独立于zabbix server和zabbix agent的组件，也就是java gateway可以是单独的一台服务器，但是也可以和zabbix server或者zabbix agent公用一台服务器，前提是端口不要配置冲突了。

本次使用一台单独的服务器，IP地址为172.31.0.104

```
# wget https://repo.zabbix.com/zabbix/4.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.0-3+bionic_all.deb
# dpkg -i zabbix-release_4.0-3+bionic_all.deb
# apt update
# apt install zabbix-java-gateway

# vim /etc/zabbix/zabbix_java_gateway.conf
# grep "^[a-z]" /etc/zabbix/zabbix_java_gateway.conf
LISTEN_IP="0.0.0.0"
LISTEN_PORT=10052
PID_FILE="/var/run/zabbix/zabbix_java_gateway.pid"
START_POLLERS=50
TIMEOUT=30

# systemctl  restart zabbix-java-gateway
# systemctl  enable  zabbix-java-gateway

# lsof  -i:10052 #验证java gateway端口
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
java    3759 zabbix   11u  IPv6  47414      0t0  TCP *:10052 (LISTEN)
```

或者编译安装java gateway

```
# yum install gcc  libxml2-devel   net-snmp net-snmp-devel   curl  curl-devel php  php-bcmath  php-mbstring  mariadb-devel java-1.8.0-openjdk-devel  -y
# ./configure --prefix=/usr/local/zabbix  --enable-java  --with-net-snmp --with-mysql --with-ssh2
#make install

#/usr/local/zabbix/sbin/zabbix_java/startup.sh  #编译安装的java gateway启动方式
```

3.2.5：配置zabbix server调用java gateway：

```
# vim  /apps/zabbix_server/etc/zabbix_server.conf
JavaGateway=172.31.0.104 #监听地址
JavaGatewayPort=10052 #指定java gateway的服务器监听端口，如果是默认端口可以不写
StartJavaPollers=20 #启动多少个进程去轮训 java gateway
```

3.2.6：验证Java Pollers：

```
# systemctl  restart  zabbix-server
```

```
root@zabbix-server:~# ps -ef | grep java
zabbix     3597   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #1
 sec]
zabbix     3598   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #2
 sec]
zabbix     3599   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #3
 sec]
zabbix     3600   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #4
 sec]
zabbix     3601   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #5
 sec]
zabbix     3602   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #6
 sec]
zabbix     3603   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #7
 sec]
zabbix     3604   3585  0 14:21 ?        00:00:00 /apps/zabbix_server/sbin/zabbix_server: java poller #8
```

3.2.7：tomcat开启JMX监控：

JMX在Java编程语言中定义了应用程序以及网络管理和监控的体系结构、设计模式、应用程序接口以及服务，通常使用JMX来监控系统的运行状态。

https://www.jianshu.com/p/8c5133cab858

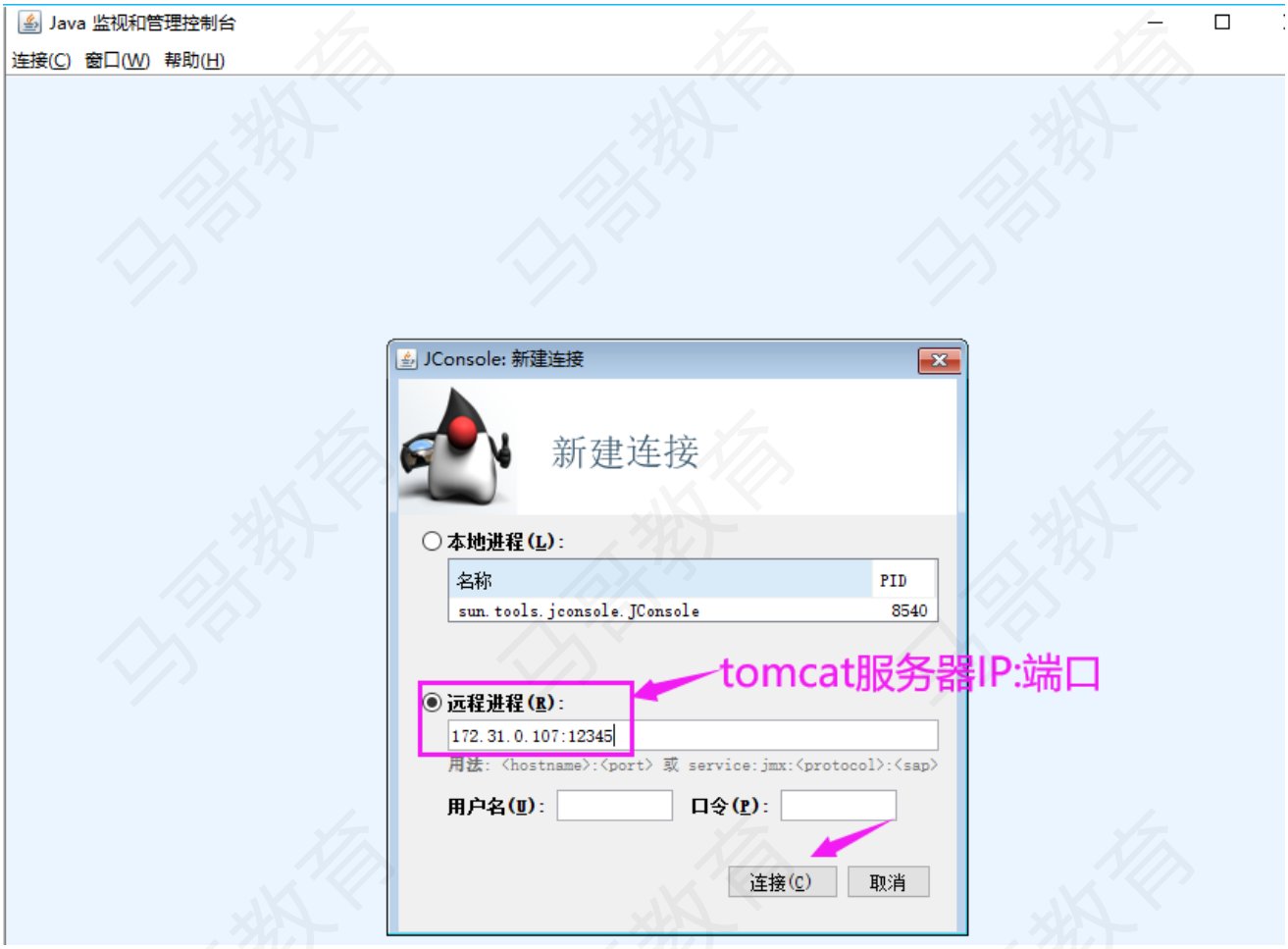
```
# vim /apps/apache-tomcat-8.0.38/bin/catalina.sh
CATALINA_OPTS="$CATALINA_OPTS
-Dcom.sun.management.jmxremote  #启用远程监控JMX
-Dcom.sun.management.jmxremote.port=12345   #默认启动的JMX端口号，要和zabbix添加主机时候的端口一致即可
-Dcom.sun.management.jmxremote.authenticate=false  #不使用用户名密码
-Dcom.sun.management.jmxremote.ssl=false  #不使用ssl认证
-Djava.rmi.server.hostname=x.x.x.x"  #tomcat主机自己的IP地址，不要写zabbix服务器的地址
```

如服务器IP是172.31.0.107，则配置如下：

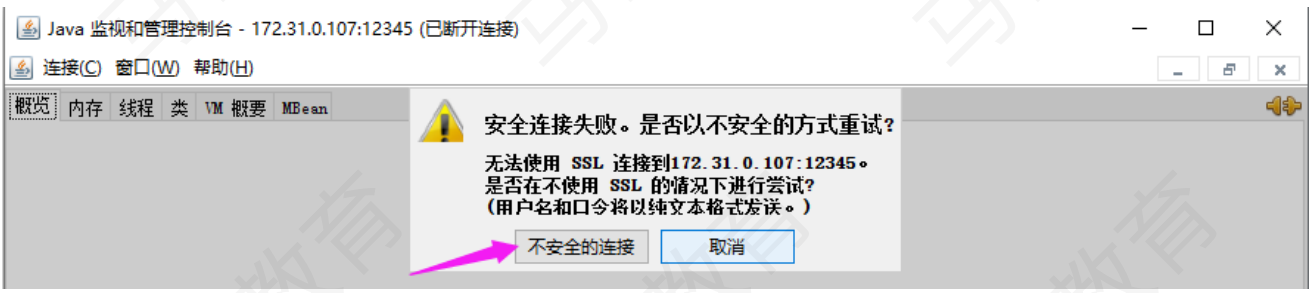
```
# vim /apps/apache-tomcat-8.0.38/bin/catalina.sh
############################
CATALINA_OPTS="$CATALINA_OPTS
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=12345
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=172.31.0.107"

# /apps/tomcat/bin/catalina.sh   stop
# /apps/tomcat/bin/catalina.sh   start
```

3.2.8：通过jconsole验证JMX数据：

在windows安装windows版本的JDK，安装过程略，安装完成之后点击安装目录的jconsole.exe，比如**C:\Program Files\Java\jdk1.8.0_221\bin\jconsole.exe**，如下：

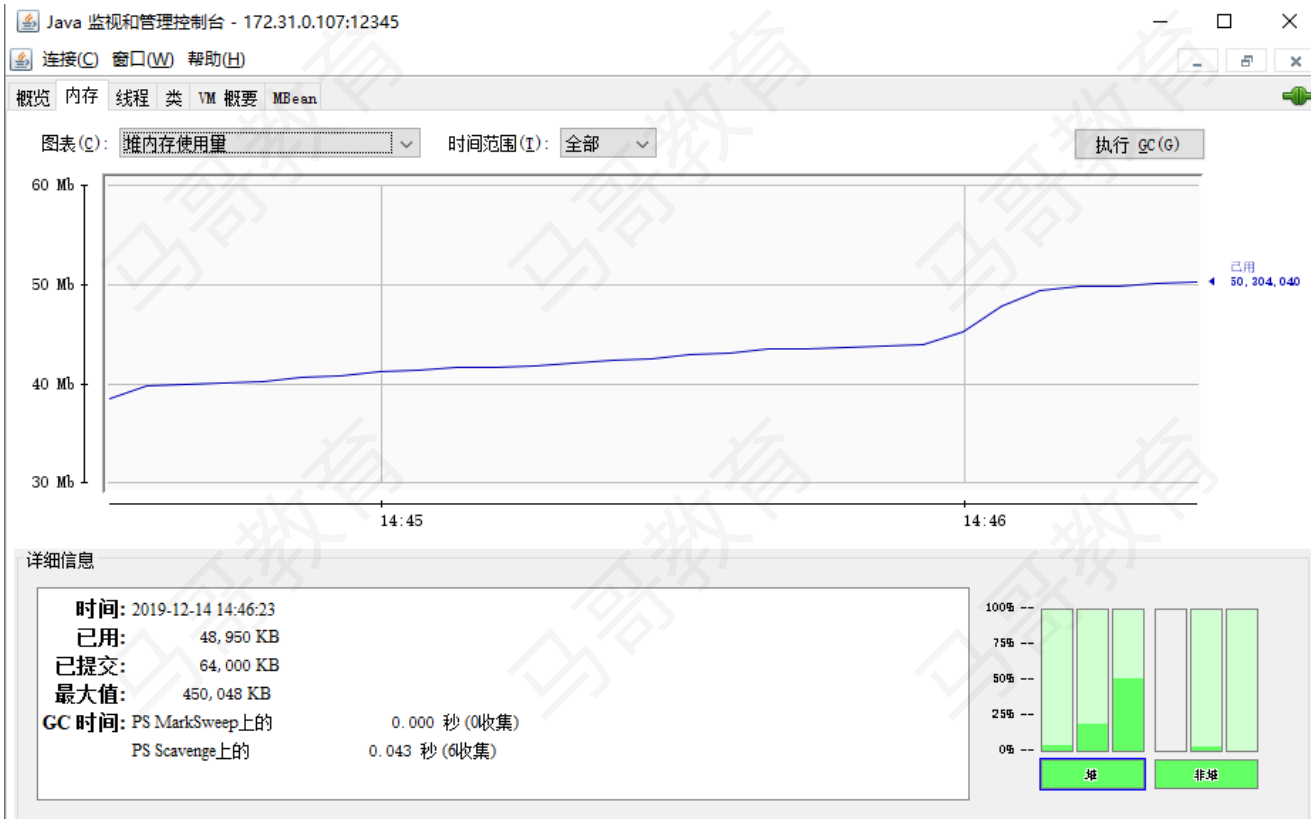


3.2.9：非安全连接：



3.2.10：验证连接：

连接成功后会显示当前JMX数据



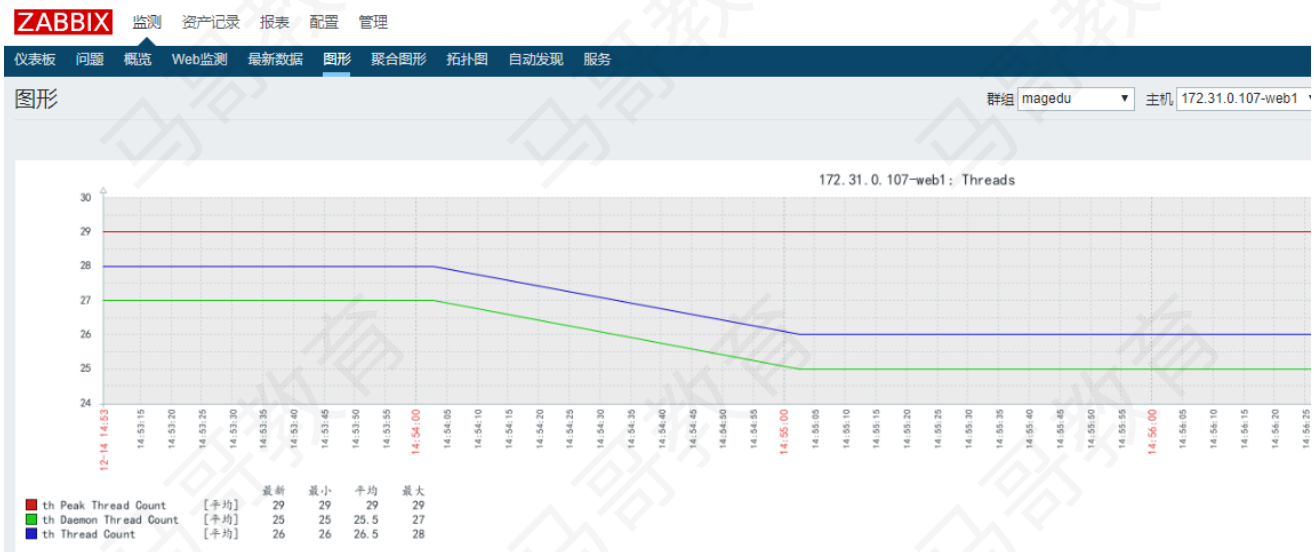
3.2.11：zabbix server添加JMX监控：

3.2.12：zabbix server关联模板：

3.2.13：验证当前JMX状态及数据：

验证JMX状态：

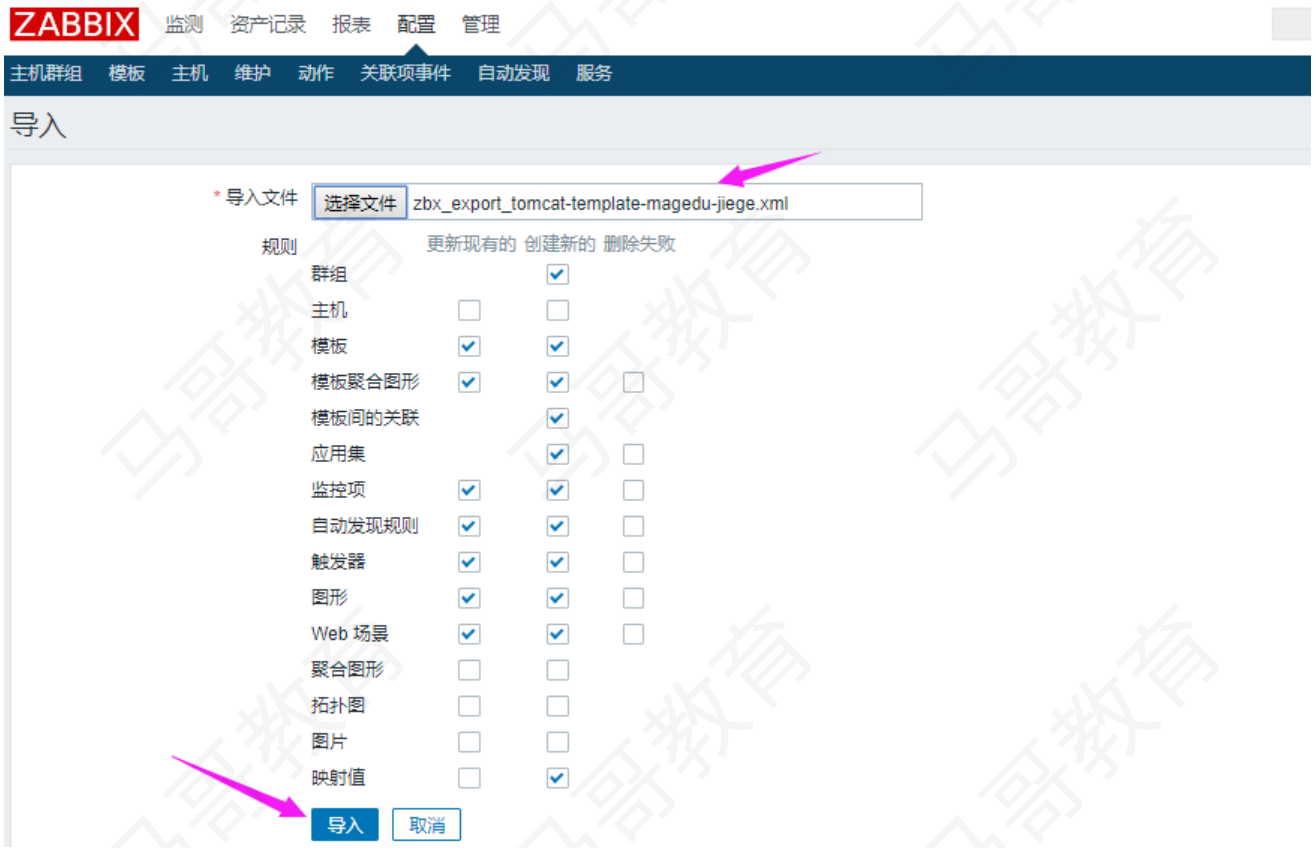
验证JMX数据：



3.2.14：JMX监控生产模板使用：

生产环境的JMX监控模板使用，主要添加了自定义监控项和阈值。

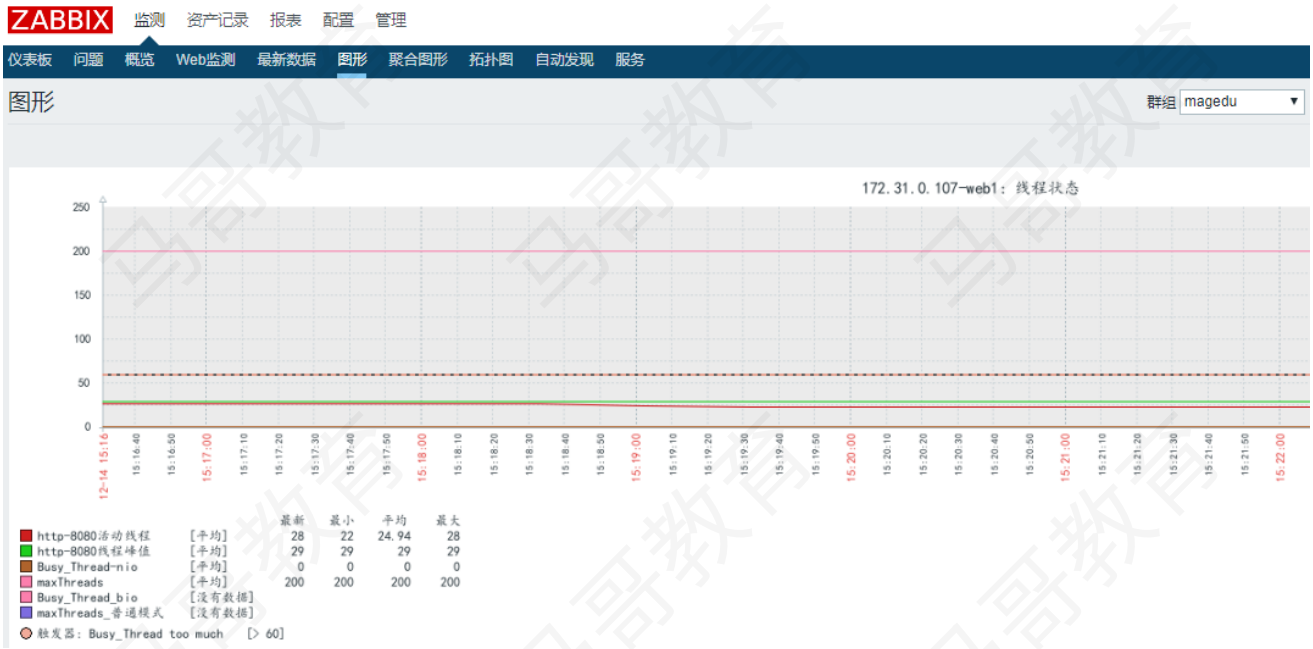
3.2.14.1：选择并导入模板：



3.2.14.2：关联模板：

将上一步骤导入的模板关联至被监控的tomcat服务器，然后取消关联并清理之前的JMX模板，如下：

3.2.14.3：验证当前模板JMX数据：



3.2.15：Linux测试监控JMX方式：

使用java客户端cmdline-jmxclient进行测试

测试能否获取到java 当前线程数和最大线程数：

```
root@zabbix-server:~# java -jar cmdline-jmxclient-0.10.3.jar - 172.31.0.107:12345
'Catalina:name="http-nio-8080",type=ThreadPool' currentThreadCount
12/14/2019 15:22:38 +0800 org.archive.jmx.Client currentThreadCount: 3

# java -jar cmdline-jmxclient-0.10.3.jar - 172.31.0.107:12345  'Catalina:name="http-
nio-8080",type=ThreadPool' maxThreads
12/14/2019 15:23:48 +0800 org.archive.jmx.Client maxThreads: 200
```

3.3：zabbix 主动与被动监控模式：

学习主动模式与被动模式工作原理，主动模式模板制作，然后添加主机并关联主动模式模板，然后验证主动模式监控数据和图形。

3.3.1：被动模式：

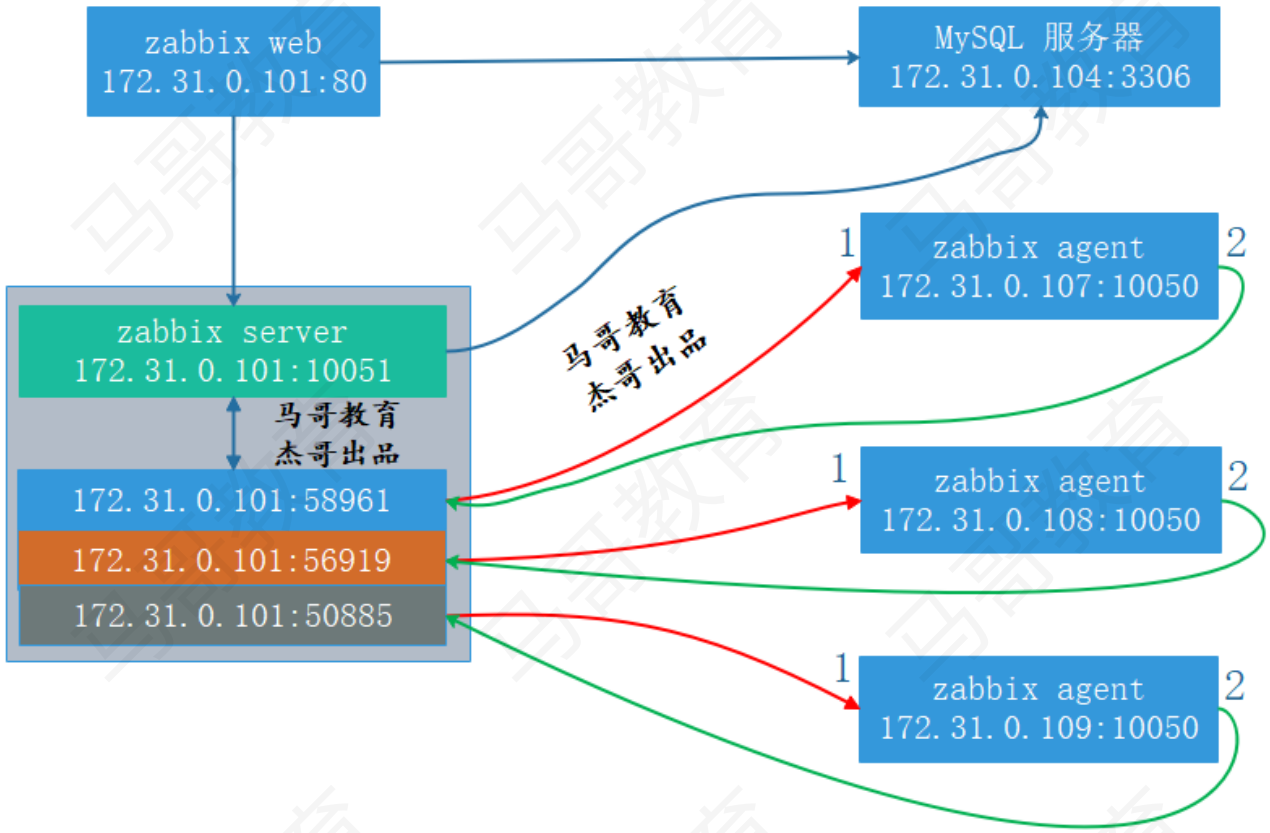
无论是模式还是被动模式，都是站在zabbix agent角度来说的工作模式，比如被动模式，是说zabbix agent被动的接受zabbix server周期性发送过来的数据收集指令，在被动模式之下，zabbix server会根据主机关联的模板中的监控项和数据采集间隔时间，周期性的打开随机端口并向zabbix agent服务器的10050发起tcp连接，然后发送获取监控项数据的指令，即zabbix server发送什么指令那么zabbix agent就收集什么数据，zabbix server什么时候发送zabbix agent就什么时候采集，zabbix server不发送zabbix agent就一直不响应，所以zabbix agent也不用关心其监控项和数据采集周期间隔时间。

被动模式的优点就是配置简单，安装后即可使用，因此也成为zabbix 的默认工作模式，但是被动模式的最大问题就是会加大zabbix server的工作量，在数百甚至数千台服务器的环境下会导致zabbix server需要轮训向每个zabbix agent发送数据采集指令，如果zabbix server负载很高还会导致不能及时获取到最新数据，但由于无需其他复杂配置，被设置为了默认的工作方式。

3.3.1.1：被动模式端口状态：

```
root@zabbix-node4:~# netstat  -tanlp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:57741           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:37425           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:44505           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:10050           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:33411           0.0.0.0:*               LISTEN
tcp        0      0 172.31.0.107:10050      172.31.0.101:51332      TIME_WAIT
tcp        0      0 172.31.0.107:10050      172.31.0.101:51214      TIME_WAIT
tcp        0      0 172.31.0.107:10050      172.31.0.101:51190      TIME_WAIT
tcp        0      0 172.31.0.107:10050      172.31.0.101:51334      TIME_WAIT
tcp        0      0 172.31.0.107:10050      172.31.0.101:51218      TIME_WAIT
tcp        0      0 172.31.0.107:10050      172.31.0.101:51262      TIME_WAIT
tcp        0      0 172.31.0.107:10050      172.31.0.101:51224      TIME_WAIT
tcp        0      0 172.31.0.107:10050      172.31.0.101:51220      TIME_WAIT
```

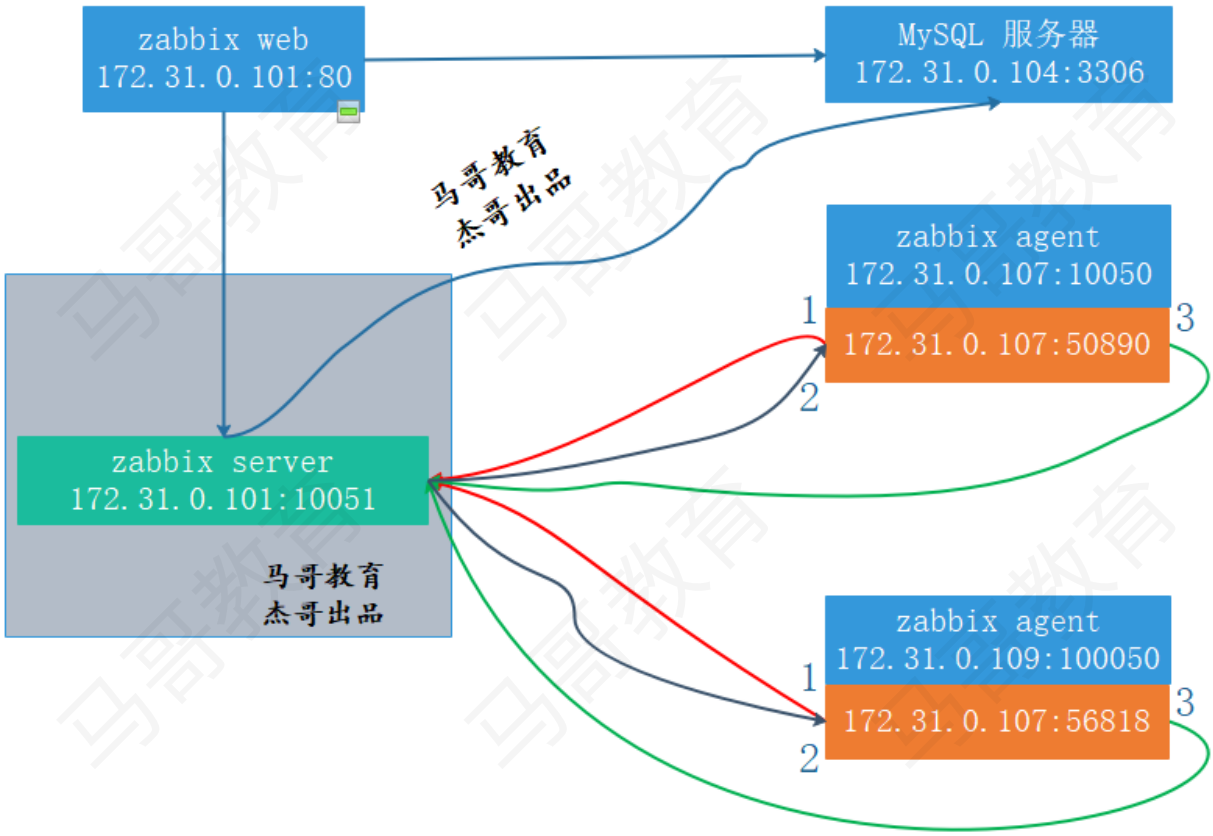
3.3.1.2：被动模式工作流程：



3.3.2：主动模式：

主动模式是由zabbix agent主动向zabbix server的10051端口发起tcp连接请求，因此主动模式下必须在zabbix agent配置文件中指定zabbix server的IP或者主机名(必须可以被解析为IP地址)，在连接到zabbix server之前zabbix agent是不知道自己要采集那些数据以及间隔多久采集一次数据的，然后在连接到zabbix server以后获取到自己的监控项和数据采集间隔周期时间，然后再根据监控项采集数据并返回给zabbix server，在主动模式下不再需要zabbix serve向zabbix agent发起连接请求，因此主动模式在一定程度上可减轻zabbix server打开的本地随机端口和进程数，在一定程度就减轻看zabbix server的压力。

3.3.2.1：主动模式工作流程：



3.3.2.2：修改zabbix agent为主动模式：

```
# grep "^[a-Z]" /etc/zabbix/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
Server=172.31.0.101
ListenPort=10050
StartAgents=3
ServerActive=172.31.0.101 #主动模式的zabbix server地址
Hostname=172.31.0.107
Include=/etc/zabbix/zabbix_agentd.d/*.conf
```

3.3.2.3：生成主动模式模板：

生成主动模式模板

Template OS Linux-active: Available memory	触发器 1	vm.memory.size[available]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: Checksum of /etc/passwd	触发器 1	vfs.file.cksum[/etc/passwd]	1h	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: Context switches per second		system.cpu.switches	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU guest nice time		system.cpu.util[,guest_nice]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU guest time		system.cpu.util[,guest]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU idle time		system.cpu.util[,idle]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU interrupt time		system.cpu.util[,interrupt]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU iowait time	触发器 1	system.cpu.util[,iowait]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU nice time		system.cpu.util[,nice]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU softirq time		system.cpu.util[,softirq]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU steal time		system.cpu.util[,steal]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU system time		system.cpu.util[,system]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: CPU user time		system.cpu.util[,user]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: Free swap space		system.swap.size[,free]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: Free swap space in %	触发器 1	system.swap.size[,pfree]	1m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: Host boot time		system.boottime	10m	1w	365d	Zabbix客户端(主动式)
Template OS Linux-active: Host local time		system.localtime	1m	1w	365d	Zabbix客户端(主动式)

3.3.2.4：添加主动模式主机并关联主动模板：

3.3.2.5：验证主动模式主机状态：

3.3.2.6：验证主动模式主机数据：

3.3.2.7：验证主动模式主机端口：

```
root@zabbix-server:~# netstat  -tanlp | grep 106
tcp        0      0 172.31.0.101:10051      172.31.0.106:56050      TIME_WAIT   -
tcp        0      0 172.31.0.101:10051      172.31.0.106:56052      TIME_WAIT   -
```

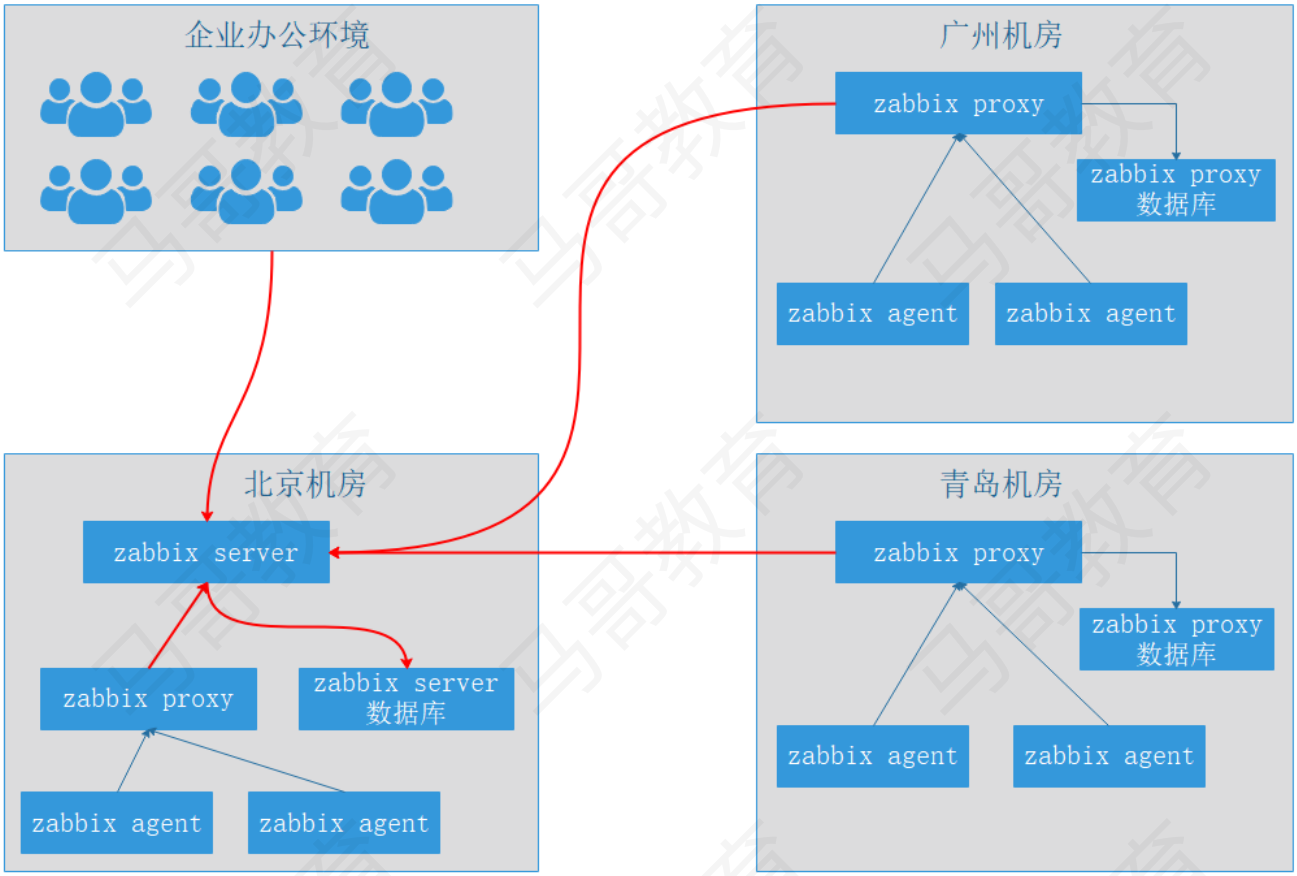
四：zabbix proxy：

https://www.zabbix.com/documentation/4.0/zh/manual/distributed_monitoring

zabbix 作为一个分布式监控系统(分布式监控解决方案)，支持通过代理(proxy)收集zabbix agent的监控数据然后由zabbix proxy再把数据发送给zabbix server，也就是zabbix proxy 可以代替 zabbix server 收集监控数据，然后把数据汇报给 zabbix server，所以zabbix proxy可以在一定程度上分担了zabbix server 的数据收集压力，从而降低了数据的采集时间、也相应的增加了zabbix server的监控能力。

另外zabbix proxy也区分主动模式和被动模式，通信方式与zabbix server主动模式和被动模式一样，区别是zabbix proxy由于没有zabbix agent的配置，所以zabbix proxy在主动模式下要向zabbix server周期性的向zabbix server申请获取zabbix agent的监控项信息，但是zabbix proxy在被动模式下也是等待zabbix server的连接并接受zabbix server发送的监控项指令，然后再有zabbix proxy向zabbix agent发起请求获取数据。

4.1：zabbix proxy架构：



4.2：zabbix proxy对比zbbbix server：

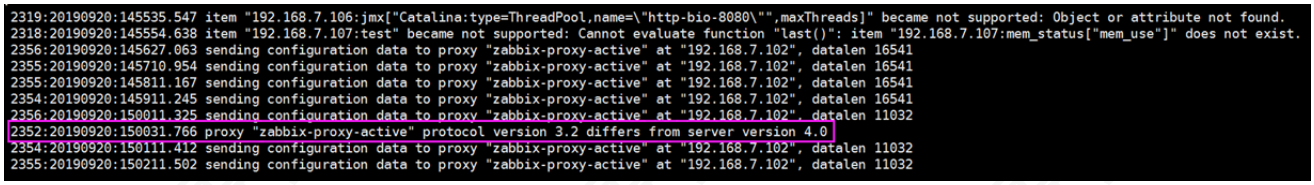
功能	zabbxy proxy	zabbix server
轻量级	是	相对重量级
图形	无	带图形控制界面
可以独立工作	是，可以独立采集数据并存储	是，即数据采集、存储、分析、展示于一体
易维护	是，配置完成后基本无需管理	维护也不难
独立数据库	保留少量最近数据	保留指定时间内的所有数据
报警通知	否，代理服务器不发送邮件通知	支持邮件、短信等告警机制

4.3：zabbix proxy部署与使用：

```
学习zabbix proxy工作原理
安装主动模式与被动模式的zabbix proxy服务器
配置zabbix gent工作模式为主动模式与被动模式
```

4.3.1：zabbix proxy版本选择：

zabbix proxy的大版本必须要和zabbix server版本一致，否则会导致出现zabbix server与zabbix proxy不兼容问题，如下：



4.3.2：zabbix proxy安装：

https://www.zabbix.com/documentation/4.0/zh/manual/installation/install

zabbix 的不同安装方式

```
zabbix server: 172.31.0.101
zabbix proxy 主动模式: 172.31.0.102 #采用apt或者yum安装
zabbix proxy 被动模式: 172.31.01.03 #采用编译安装
```

4.3.2.1：apt/yum安装zabbix proxy：

https://www.zabbix.com/documentation/4.0/zh/manual/installation/install_from_packages/debian_ubuntu

```
# wget https://repo.zabbix.com/zabbix/4.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.0-3+bionic_all.deb
# dpkg -i zabbix-release_4.0-3+bionic_all.deb
# apt update

#安装zabbix proxy
# yum install gcc  libxml2-devel   net-snmp net-snmp-devel   curl  curl-devel php  php-bcmath  php-mbstring mariadb mariadb-devel java-1.8.0-openjdk-devel  -y

# apt install libmysqld-dev  libmysqlclient-dev  libxml2-dev  libxml2 snmp  libsnmp-dev  libevent-dev  curl  libcurl4-openssl-dev

# apt install zabbix-proxy-mysql

#主动模式数据库导入:
# zcat /usr/share/doc/zabbix-proxy-mysql/schema.sql.gz | mysql -uproxy -p123456 -h172.31.0.104 zabbix_proxy_active
```

4.3.2.2：编译安装zabbix proxy：

```
# useradd  zabbix  -s /usr/sbin/nologin

# pwd
/usr/local/src/zabbix-4.0.15
# ./configure --prefix=/apps/zabbix_proxy  --enable-proxy --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-libxml2
# make install

#创建zabbix proxy数据库
# mysql
mysql> create database zabbix_proxy_active  character set utf8 collate utf8_bin;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> grant all privileges on  zabbix_proxy_active.* to proxy@'172.31.0.%' identified
by '123456';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> create database zabbix_proxy_pasive  character set utf8 collate utf8_bin;
Query OK, 1 row affected (0.00 sec)
mysql> grant all privileges on  zabbix_proxy_pasive.* to proxy@'172.31.0.%' identified
by '123456';
Query OK, 0 rows affected, 1 warning (0.00 sec)

#被动模式数据库导入
# pwd
/usr/local/src/zabbix-4.0.15
# mysql -uproxy -p123456 -h172.31.0.104 zabbix_proxy_pasive < database/mysql/schema.sql

#zabbix proxy启动service文件:
# cat /lib/systemd/system/zabbix-proxy.service
[Unit]
Description=Zabbix Proxy
After=syslog.target
After=network.target

[Service]
Environment="CONFFILE=//apps/zabbix_proxy/etc/zabbix_proxy.conf"
EnvironmentFile=-/etc/default/zabbix-proxy
Type=forking
Restart=on-failure
PIDFile=/tmp/zabbix_proxy.pid
KillMode=control-group
ExecStart=/apps/zabbix_proxy/sbin/zabbix_proxy -c $CONFFILE
ExecStop=/bin/kill -SIGTERM $MAINPID
RestartSec=10s
TimeoutSec=infinity

[Install]
WantedBy=multi-user.target
```

4.3.3：配置被动zabbix proxy：

配置并使用被动模式(Passive Proxy Mode)的zabbix proxy收集zabbix agent监控数据。

```
zabbix server：172.31.0.101
被动模式服务器IP：172.31.0.103
web服务器IP：172.31.0.107
```

4.3.3.1：zabbix proxy被动配置：

```
# vim /apps/zabbix_proxy/etc/zabbix_proxy.conf
ProxyMode=1 #0为主动，1为被动
Server=172.31.0.101 #zabbix server服务器的地址或主机名
Hostname=magedu-jiege-proxy-active  #代理服务器名称，需要与zabbix server添加代理时候的proxy
name是一致的!
```

```
ListenPort=10051 #zabbix proxy监听端口
LogFile=/tmp/zabbix_proxy.log
EnableRemoteCommands=1 #允许zabbix server执行远程命令
DBHost=172.31.0.104  #数据库服务器地址
DBName=zabbix_proxy_active #使用的数据库名称
DBUser=proxy #连接数据库的用户名称
DBPassword=123456 #数据库用户密码
DBPort=3306 #数据库端口
ProxyLocalBuffer=720  #已经提交到zabbix server的数据保留时间
ProxyOfflineBuffer=720 #未提交到zabbix server的时间保留时间
HeartbeatFrequency=60  #心跳间隔检测时间，默认60秒，范围0-3600秒，被动模式不使用
ConfigFrequency=5  #间隔多少秒从zabbix server获取监控项信息
DataSenderFrequency=5 #数据发送时间间隔，默认为1秒，范围为1-3600秒，被动模式不使用
StartPollers=20  #启动的数据采集器数量
JavaGateway=172.31.0.104 #java gateway服务器地址,当需要监控java的时候必须配置否则监控不到数据
JavaGatewayPort=10052 #Javagatewa服务端口
StartJavaPollers=20  #启动多少个线程采集数据
CacheSize=2G #保存监控项而占用的最大内存
HistoryCacheSize=2G #保存监控历史数据占用的最大内存
HistoryIndexCacheSize=128M  #历史索引缓存的大小
Timeout=30 #监控项超时时间，单位为秒
LogSlowQueries=3000 #毫秒，多久的数据库查询会被记录到日志
```

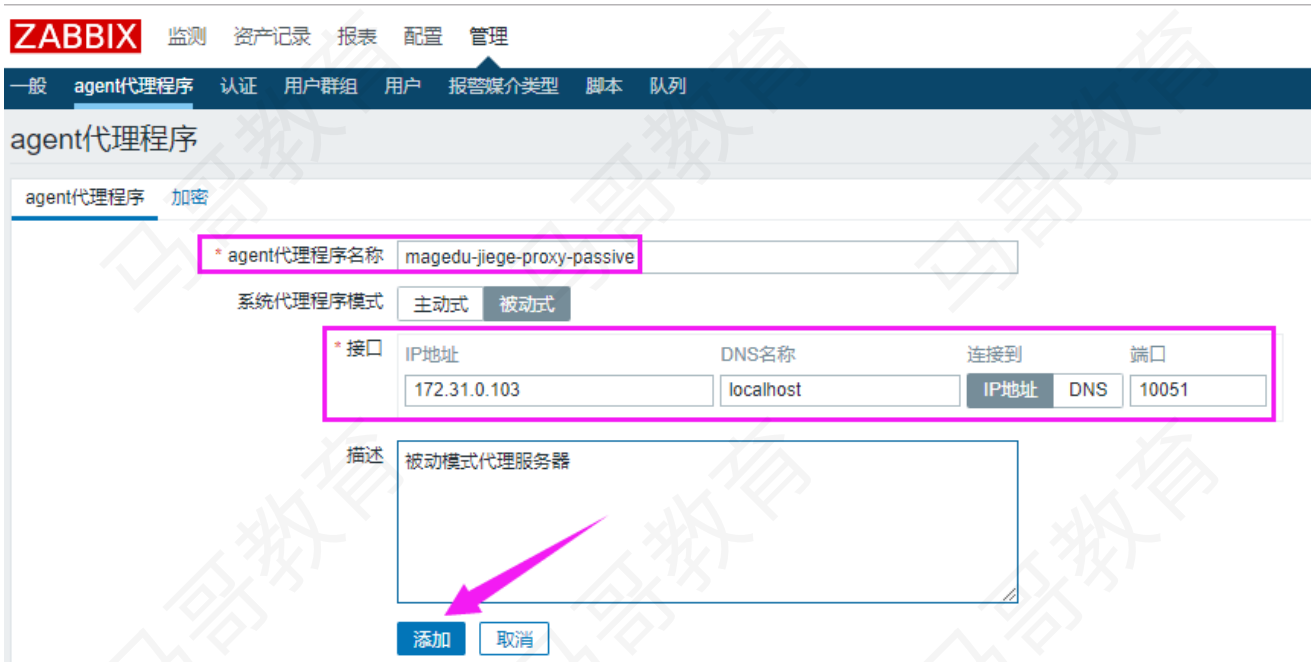
4.3.3.2：重启zabbix proxy服务：

重启zabbix proxy服务并验证日志

```
root@zabbix-proxy-passive:~# systemctl  restart zabbix-proxy
root@zabbix-proxy-passive:~# tail  /tmp/zabbix_proxy.log
  1971:20191216:102255.620 proxy #34 started [poller #5]
  1974:20191216:102255.623 proxy #37 started [trapper #2]
  1972:20191216:102255.624 proxy #35 started [unreachable poller #1]
  1973:20191216:102255.624 proxy #36 started [trapper #1]
  1969:20191216:102255.628 proxy #32 started [poller #3]
  1970:20191216:102255.630 proxy #33 started [poller #4]
  1977:20191216:102255.631 proxy #40 started [trapper #5]
  1978:20191216:102255.632 proxy #41 started [icmp pinger #1]
  1975:20191216:102255.633 proxy #38 started [trapper #3]
  1976:20191216:102255.634 proxy #39 started [trapper #4]
```

4.3.3.3：zabbix 添加被动代理：

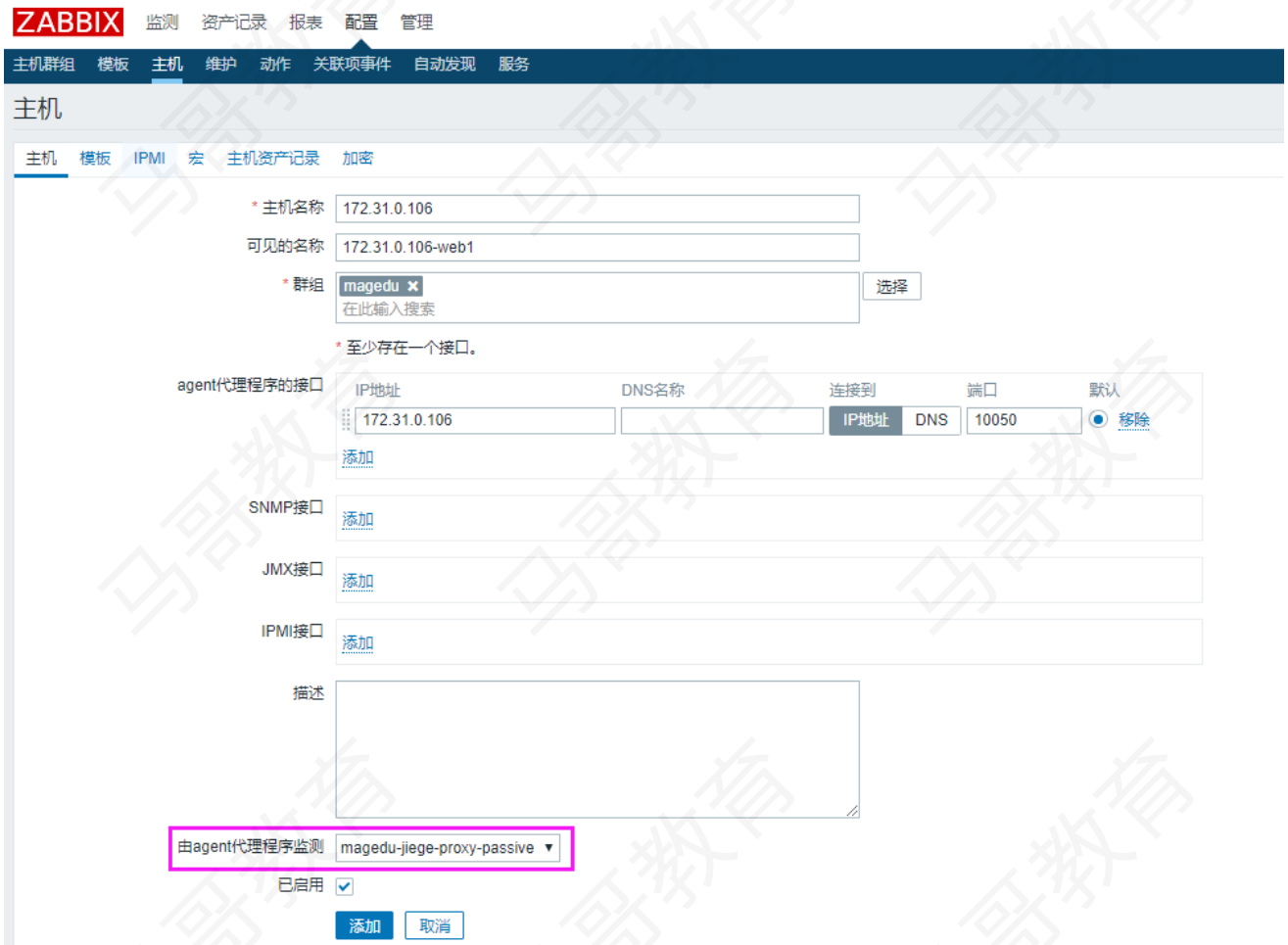
管理-->agent代理程序-->创建代理：



4.3.3.4：配置zabbix agent使用被动代理：

配置-->主机-->创建 主机：

4.3.3.4.1：添加被动模式代理主机：



4.3.3.4.2：关联被动模式模板：

4.3.3.4.3：zabbix server配置：

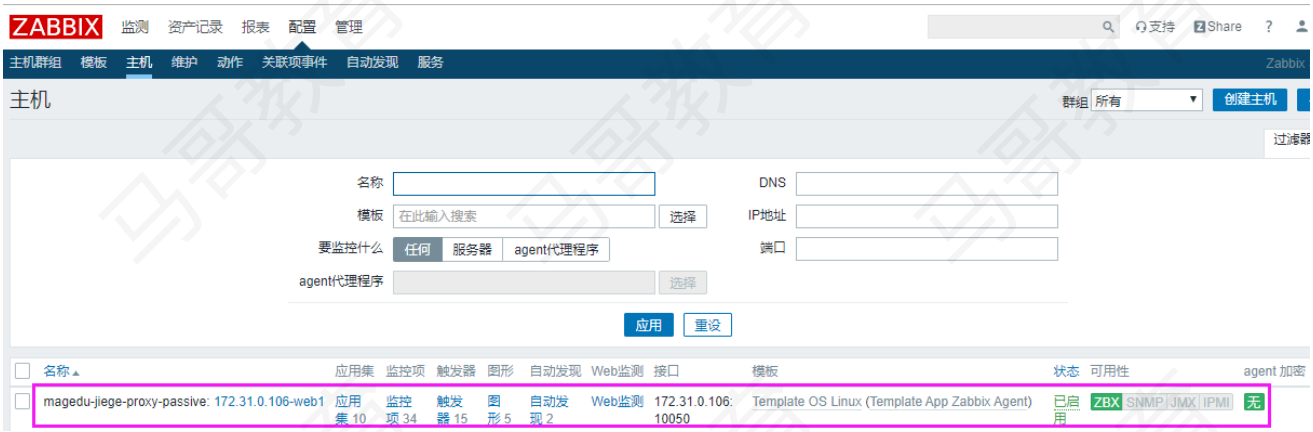
修改zabbix server向zabbix获取监控数据的频率等参数

```
### Option: StartProxyPollers
#       Number of pre-forked instances of pollers for passive proxies.
#
# Mandatory: no
# Range: 0-250
# Default:
StartProxyPollers=20 #启用多少子进程与代理端通信，若代理较多可考虑加大此数值，范围是0-250

### Option: ProxyConfigFrequency
#       How often Zabbix Server sends configuration data to a Zabbix Proxy in seconds.
#       This parameter is used only for proxies in the passive mode.
#
# Mandatory: no
# Range: 1-3600*24*7
# Default:
ProxyConfigFrequency=60 #proxy被动模式下，server多少秒同步配置文件至proxy，该参数仅用于被动模式下的代理，范围是1-3600*24*7

### Option: ProxyDataFrequency
#       How often Zabbix Server requests history data from a Zabbix Proxy in seconds.
#       This parameter is used only for proxies in the passive mode.
#
# Mandatory: no
# Range: 1-3600
# Default:
ProxyDataFrequency=60  #被动模式下，zabbix server间隔多少秒向proxy请求历史数据
```

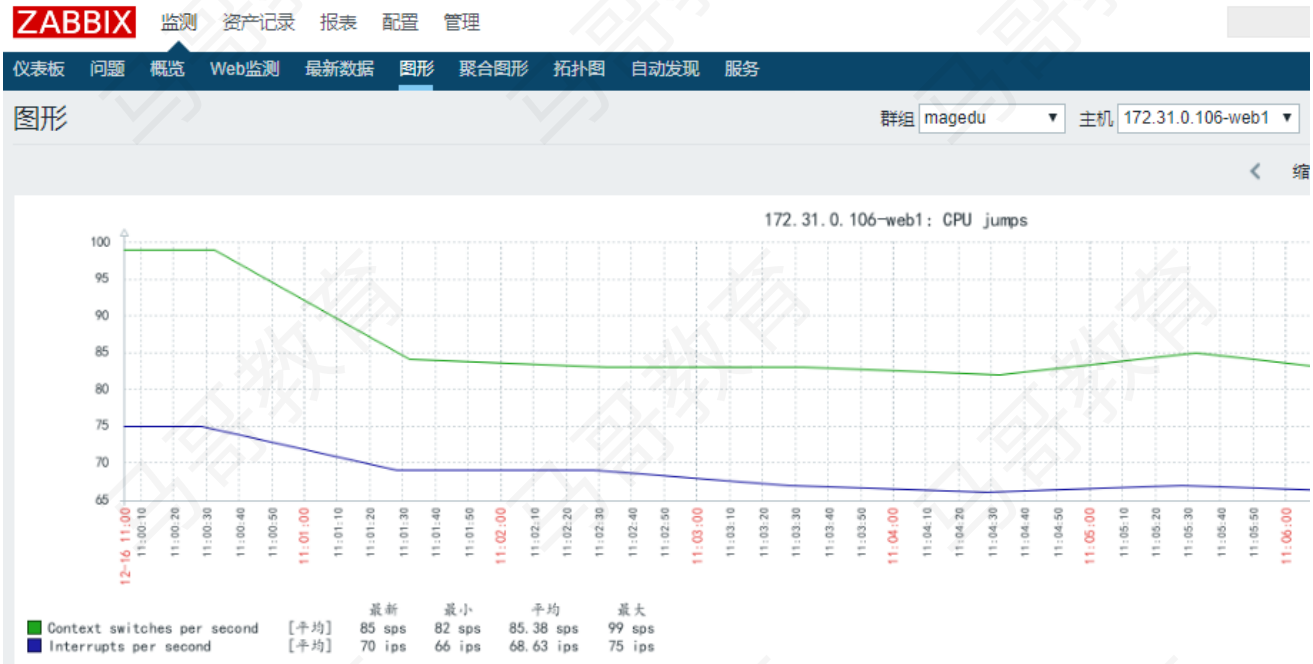
4.3.3.5：zabbix agent配置文件：

```
root@zabbix-node3:~# grep "^[a-Z]" /etc/zabbix/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
Server=172.31.0.101,172.31.0.102
ServerActive=127.0.0.1
Hostname=172.31.0.106
Timeout=30
Include=/etc/zabbix/zabbix_agentd.d/*.conf
```

4.3.3.6：zabbix web验证当前主机状态：



4.3.3.7：验证主机监控数据及图形：



4.3.4：配置主动模式zabbix proxy：

配置并使用主动模式(Active Proxy Mode)的zabbix proxy收集zabbix agent监控数据。

```
zabbix server：172.31.0.101
主动模式服务器IP：172.31.0.102
web服务器IP：172.31.0.107
```

4.3.4.1：zabbix proxy主动配置：

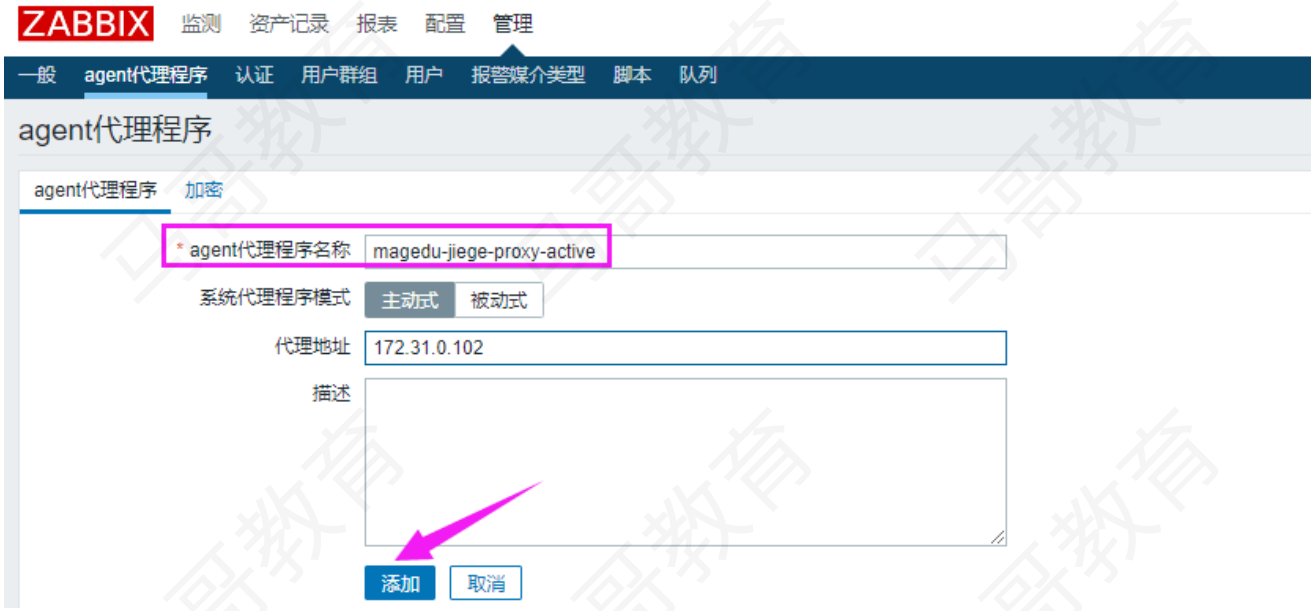
```
# vim /etc/zabbix/zabbix_proxy.conf
ProxyMode=0 #0为主动，1为被动
Server=172.31.0.101 #zabbix server服务器的地址或主机名
Hostname=magedu-jiege-proxy-active  #代理服务器名称，需要与zabbix server添加代理时候的proxy name是一致的!
ListenPort=10051 #zabbix proxy监听端口
LogFile=/tmp/zabbix_proxy.log
EnableRemoteCommands=1 #允许zabbix server执行远程命令
DBHost=172.31.0.104  #数据库服务器地址
DBName=zabbix_proxy_active #使用的数据库名称
DBUser=proxy #连接数据库的用户名称
DBPassword=123456 #数据库用户密码
DBPort=3306 #数据库端口
ProxyLocalBuffer=720  #已经提交到zabbix server的数据保留时间
ProxyOfflineBuffer=720 #未提交到zabbix server的时间保留时间
HeartbeatFrequency=60  #心跳间隔检测时间，默认60秒，范围0-3600秒，被动模式不使用
ConfigFrequency=5  #间隔多少秒从zabbix server获取监控项信息
DataSenderFrequency=5 #数据发送时间间隔，默认为1秒，范围为1-3600秒，被动模式不使用
StartPollers=20  #启动的数据采集器数量
JavaGateway=172.31.0.104 #java gateway服务器地址,当需要监控java的时候必须配置否则监控不到数据
JavaGatewayPort=10052 #Javagatewa服务端口
StartJavaPollers=20  #启动多少个线程采集数据
CacheSize=2G #保存监控项而占用的最大内存
HistoryCacheSize=2G #保存监控历史数据占用的最大内存
HistoryIndexCacheSize=128M  #历史索引缓存的大小
Timeout=30 #监控项超时时间，单位为秒
LogSlowQueries=3000 #毫秒，多久的数据库查询会被记录到日志
```

4.3.4.2：重启zabbix proxy服务：

```
root@zabbix-proxy-active:~# systemctl  restart zabbix-proxy
root@zabbix-proxy-active:~# systemctl  enable  zabbix-proxy
```

4.3.4.3：zabbix web添加主动代理：

管理-->agent代理程序-->创建代理：



4.3.4.4：zabbix agent使用主动代理：

修改或者新添加一台zabbix agent，并使用zabbix 主动模式代理服务器进行监控：

使用主动模式proxy：

验证当前主机和主动模式proxy 状态：

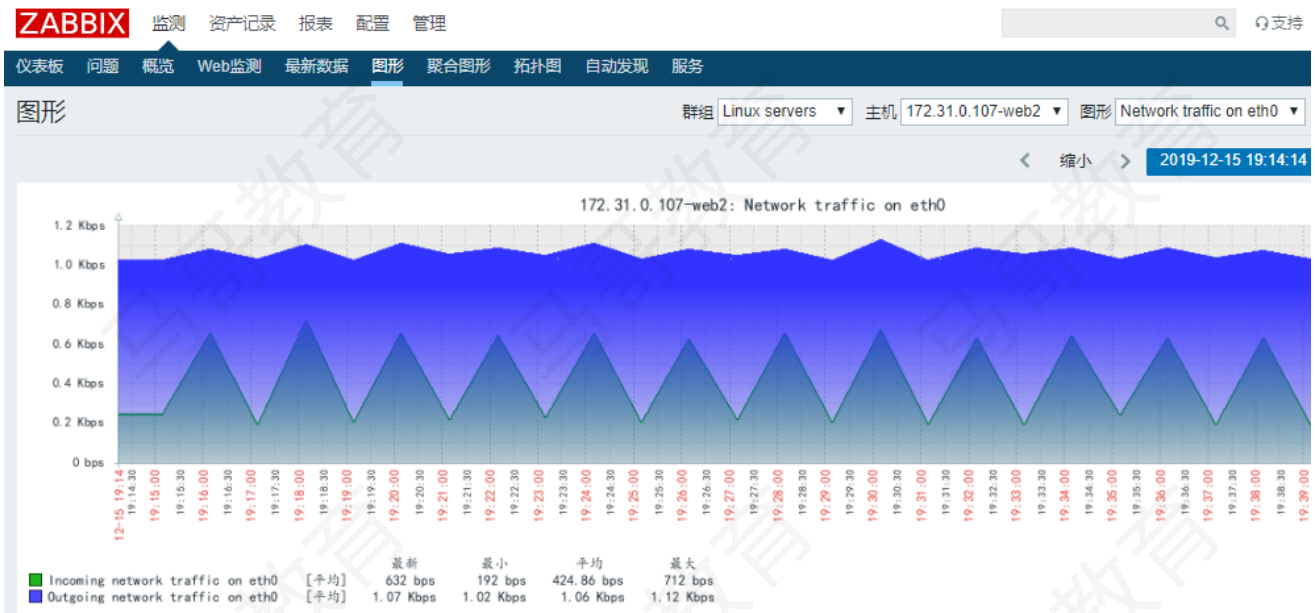
4.3.4.5：zabbix agent配置文件：

需要修改zabbix agent中配置文件ServerActive的值为主动模式zabbix proxy。

```
# grep  "^[a-Z]" /etc/zabbix/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
Server=172.31.0.101,172.31.0.103 #zabbix server与zabbix passive代理的地址
ServerActive=172.31.0.102
Hostname=172.31.0.106
Timeout=30
Include=/etc/zabbix/zabbix_agentd.d/*.conf

# systemctl  restart zabbix-agent
```

4.3.3.6：zabbix web验证当前主机状态：

4.3.3.7：验证主机监控数据及图形：



4.3.3.8：交互过程：

zabbix agent与zabbix proxy与zabbix srver的交互过程

```
# vim /var/log/zabbix/zabbix_proxy.log
zabbix agent向zabbix proxy申请监控项:
  2219:20191215:185229.406 sending [{"request":"active checks","host":"172.31.0.107"}]
  2219:20191215:185229.406 before read
  2219:20191215:185229.416 got [{"response":"failed","info":"host [172.31.0.107] not
found"}]
  2219:20191215:185229.416 In parse_list_of_checks(

#zabbix proxy向zabbix server发起请求获取主机监控项
  4697:20191215:185229.426 In is_ip4() ip:'172.31.0.107'
  4697:20191215:185229.426 In get_hostid_by_host() host:'172.31.0.107' metadata:''
  4697:20191215:185229.426 query [txnlev:0] [select
h.hostid,h.status,h.tls_accept,h.tls_issuer,h.tls_subject,h.tls_psk_identity,a.host_met
adata from hosts h left join autoreg_host a on a.proxy_hostid is null and a.host=h.host
where h.host='172.31.0.107' and h.status in (0,1) and h.flags<>2 and h.proxy_hostid is
null]
  4697:20191215:185229.431 query [txnlev:1] [insert into proxy_autoreg_host
(clock,host,listen_ip,listen_dns,listen_port,host_metadata) values
(1576407149,'172.31.0.107','172.31.0.107','',10050,'')]

#收到zabbix server返回的监控项
  361:20191215:191141.647 Received [{"globalmacro":{"fields":
["globalmacroid","macro","value"],"data":[[2,"{$SNMP_COMMUNITY}","public"]]},"hosts":
{"fields":
["hostid","host","status","available","ipmi_authtype","ipmi_privilege","ipmi_username",
"ipmi_password","ipmi_available","snmp_available","jmx_available","name","tls_connect",
"tls_accept","tls_issuer","tls_subject","tls_psk_identity","tls_psk"],"data":
[[10050,"Template App Zabbix Agent",3,0,-1,2,"","",0,0,0,"Template App Zabbix
Agent",1,1,"","","",""],[10270,"Template OS Linux-
active",3,0,-1,2,"","",0,0,0,"Template OS Linux-active",1,1,"","","",""],[10276,"107-
web2",0,0,-1,2,"","",0,0,0,"172.31.0.107-web2",1,1,"","","",""]]},"interface":
{"fields":
["interfaceid","hostid","main","type","useip","ip","dns","port","bulk"],"data":
[[7,10276,1,1,1,"172.31.0.107","","10050",1]]},"hosts_templates":{"fields":
["hosttemplateid","hostid","templateid"],"data":[[251,10270,10050],
[255,10276,10270]]},"hostmacro":{"fields":
["hostmacroid","hostid","macro","value"],"data":[]}

#收到zabbix agent返回的监控数据
1425:20191215:191315.963 trapper got '{"request":"agent
data","session":"a62af26d487fd475d3f94d36c58880e5","data":
[{"host":"172.31.0.107","key":"proc.num[]","value":"180","id":42,"clock":1576408390,"ns
":885729681},
{"host":"172.31.0.107","key":"system.cpu.intr","value":"164526","id":43,"clock":1576408
390,"ns":886469760},
{"host":"172.31.0.107","key":"system.cpu.load[percpu,avg15]","value":"0.000000","id":44
,"clock":1576408390,"ns":887003663},
{"host":"172.31.0.107","key":"system.cpu.load[percpu,avg1]","value":"0.010000","id":45,
"clock":1576408390,"ns":887473903},
{"host":"172.31.0.107","key":"system.cpu.load[percpu,avg5]","value":"0.005000","id":46,
"clock":1576408390,"ns":887917992},
{"host":"172.31.0.107","key":"system.cpu.switches","value":"218790","id":47,"clock":157
6408390,"ns":889276519},
```

五：zabbix 监控案例实战：

自定义监控项

通过脚本采集监控项数据

zabbix agent获取监控项数据

自定义模板和图形及触发器

验证数据

5.1：监控Linux TCP连接状态：

TCP，全称Transfer Control Protocol，中文名为传输控制协议，它工作在OSI的传输层，提供面向连接的可靠传输服务，TCP的工作主要是建立连接，然后从应用层程序中接收数据并进行传输。TCP采用虚电路连接方式进行工作，在发送数据前它需要在发送方和接收方建立一个连接，数据在发送出去后，发送方会等待接收方给出一个确认性的应答，否则发送方将认为此数据丢失，并重新发送此数据。

在建立连接的时候，所谓的客户端与服务端是相对应的，即要看是谁主动连接的谁，如果A主动连接B那么A就是客户端而B是服务端，如果返过来B主动连接A，那么B就是客户端而A就成了服务端。

5.1.1：TCP端口的十一种连接状态：

CLOSED：端口默认是关闭状态。

LISTEN： 服务器程序开始监听一个端口，就是LISTEN状态。

SYN_RCVD：三次握手的第二次握手后的端口状态，是收到了客户端发送的SYN_SENT数据包之后的状态，这个状态很短暂，正常在服务器上是很少看到的，除非服务器故意不发送最后一次握手数据包，服务器返回给客户端SYN确认之后就会将在自己的端口置为SYN_RCVD。

SYN_SENT：SYN_SENT状态表示客户端已发送SYN=1的请求连接报文，发送之后客户端就会将自己的端口状态置为SYN_SENT。

ESTABLISHED：表示已经连接成功，客户端收到服务器的确认报文会回复服务器，然后就将端口置为ESTABLISHED，服务器第三次收到客户端的Ack确认就会将端口置为ESTABLISHED并开始传输数据。

FIN_WAIT_1：出现在主动关闭方，FIN_WAIT_1状态实际上是当SOCKET在ESTABLISHED状态时，当任意一方想主动关闭连接，向对方发送了FIN=1的断开连接请求报文，此时该SOCKET即 进入到FIN_WAIT_1状态。而当对方回应ACK报文后，则进入到FIN_WAIT_2状态，当然在实际的正常情况下，无论对方何种情况下，都应该马 上回应ACK报文，所以FIN_WAIT_1状态一般是比较难见到的，而FIN_WAIT_2状态还有时常常可以用netstat看到。

FIN_WAIT_2：出现在主动关闭方，当被动方回应FIN_WAIT_1的ACK报文后，则进入到FIN_WAIT_2状态

TIME_WAIT：出现在主动关闭方，表示收到了对方的FIN请求关闭报文，并发送出了ACK报文，就等2MSL后即可回到CLOSED可用状态了。如果FIN_WAIT_1状态下，收到了对方同时带FIN标志和ACK标志的报文时，可以直接进入到TIME_WAIT状态，而无须经过FIN_WAIT_2状态。

CLOSING： 这种状态比较特殊，实际情况中应该是很少见，属于一种比较罕见的例外状态。正常情况下，当你发送FIN报文后，按理来说是应该先收到（或同时收到）对方的 ACK报文，再收到对方的FIN报文。但是CLOSING状态表示你发送FIN报文后，并没有收到对方的ACK报文，反而却也收到了对方的FIN报文。什 么情况下会出现此种情况呢？其实细想一

下，也不难得出结论：那就是如果双方几乎在同时close一个SOCKET的话，那么就出现了双方同时发送FIN报 文的情况，也即会出现CLOSING状态，表示双方都正在关闭SOCKET连接。

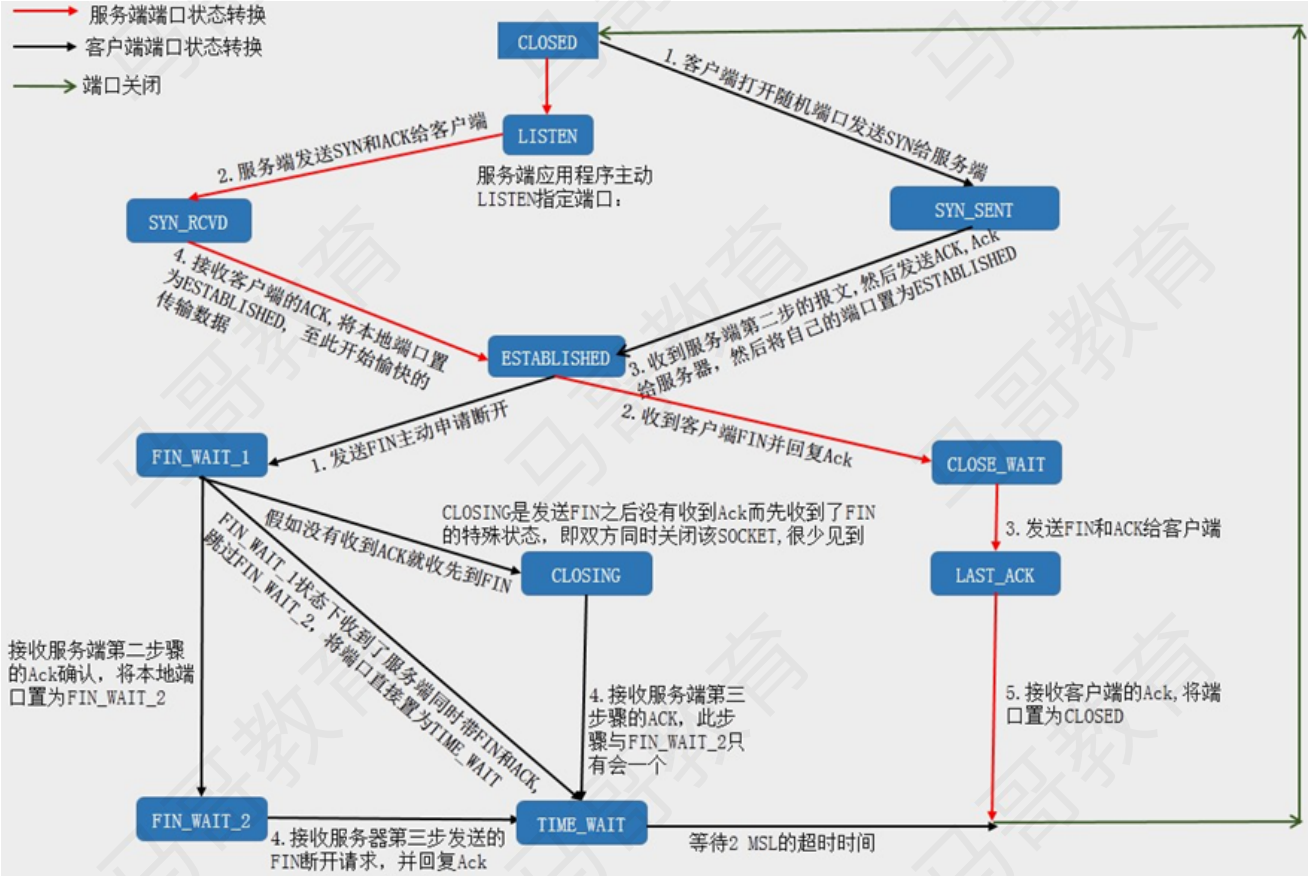
CLOSE_WAIT： 表示在等待关闭端口，这种状态存在于被动关闭的一方。

LAST_ACK： 是被动关闭方在主动关闭一方在发送FIN报文后，最后等待对方的ACK报文，当再次收到ACK报文后，也即可以进入到CLOSED可用状态了。

5.1.2：端口转换状态：

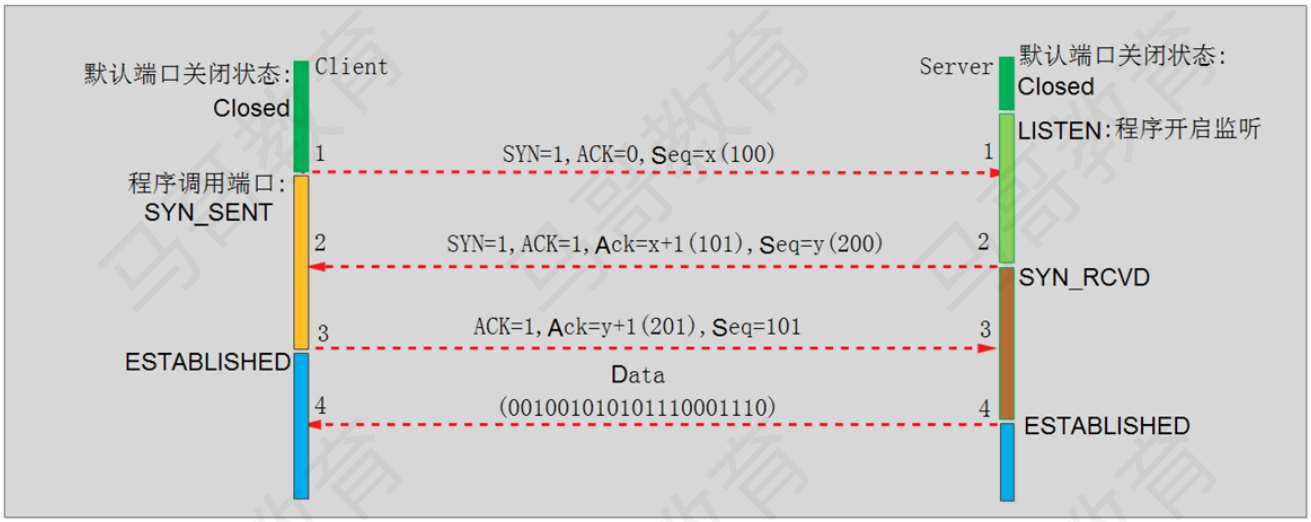
安装服务端与客户端将端口状态区分：

主动端口方：SYN_SENT、FIN_WAIT1、FIN_WAIT2、CLOSING、TIME_WAIT
被动断开方：LISTEN、SYN_RCVD、CLOSE_WAIT、LAST_ACK
都具有的：CLOSED 、ESTABLISHED



5.1.3：TCP三次握手与四次断开：

5.1.3.1：TCP三次握手：



5.1.3.2：TCP四次断开：

5.1.4：监控TCP连接数脚本：

```
# cat /etc/zabbix/zabbix_agentd.d/tcp_conn_plugin.sh
#!/bin/bash
#Author:Zhang ShiJie
tcp_conn_status(){
        TCP_STAT=$1
        ss -ant | awk 'NR>1 {++s[$1]} END {for(k in s) print k,s[k]}' >
/tmp/tcp_conn.txt
        TCP_NUM=$(grep "$TCP_STAT" /tmp/tcp_conn.txt | cut -d ' ' -f2)
        if [ -z $TCP_NUM ];then
                TCP_NUM=0
        fi
        echo $TCP_NUM
}

main(){
```

```
        case $1 in
            tcp_status)
                tcp_conn_status $2;
                ;;
        esac
}

main $1 $2
```

5.1.5：zabbix agent添加自定义监控项：

zabbix agent添加自定义监控项并调用脚本获取到监控项数据

```
# vim /etc/zabbix/zabbix_agentd.conf
...............
297 UserParameter=linux_status[*],/etc/zabbix/zabbix_agentd.d/tcp_conn_plugin.sh "$1"
"$2

# chmod a+x /etc/zabbix/zabbix_agentd.d/tcp_conn_plugin.sh
# systemctl  restart zabbix-agent #重启zabbix agent
```

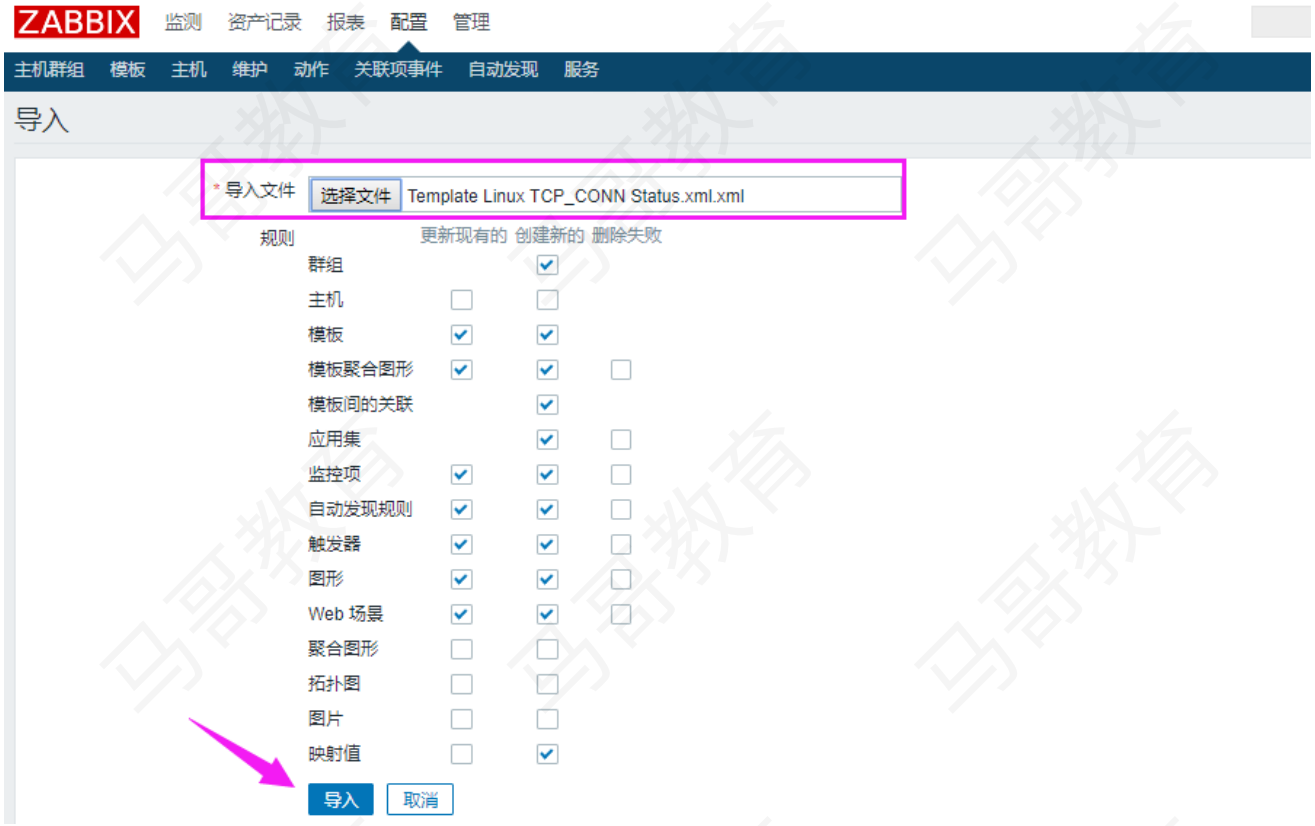
5.1.6：zabbix server测试监控项数据：

```
root@zabbix-server:~# /apps/zabbix_server/bin/zabbix_get  -s 172.31.0.106 -p 10050 -k
"linux_status["tcp_status","LISTEN"]"
19
```

5.1.7：zabbix web导入模板：

配置-模板-导入：

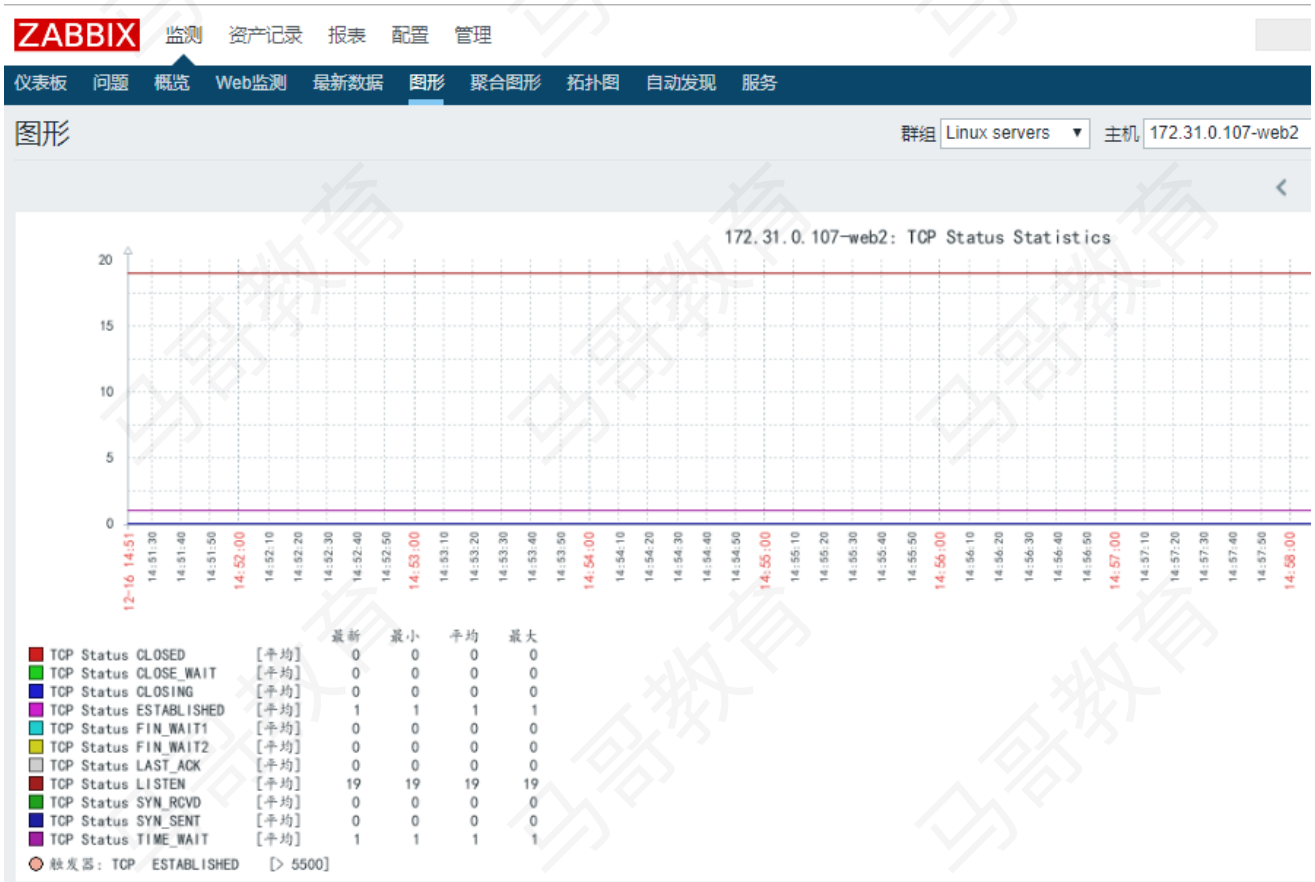
5.1.7.1：导入模板：



5.1.7.2：模板导入成功：

5.1.8：将TCP监控模板关联至主机：

5.1.9：验证监控数据：



5.2：监控memcache：

通过自定义监控模板对memcache进行监控

5.2.1：安装memcache服务：

```
# apt-get install memcached nmap #ubuntu
# yum install memcached nmap #centos
# grep -v "#" /etc/memcached.conf  | grep -v "^$"
```

```
-d
logfile /var/log/memcached.log
-m 512
-p 11211
-u memcache
-l 0.0.0.0
-P /var/run/memcached/memcached.pid


root@zabbix-node4:~# systemctl  restart memcached
root@zabbix-node4:~# systemctl  enable  memcached
```

5.2.2：监控脚本：

传递不同参数传递给脚本，并通过脚本获取监控项

```
# pwd
/etc/zabbix/zabbix_agentd.d


# cat memcache_monitor.sh
#!/bin/bash
#Author:Zhang ShiJie
memcached_status(){
        M_PORT=$1
        M_COMMAND=$2
        echo -e "stats\nquit" | ncat  127.0.0.1 "$M_PORT" | grep "STAT $M_COMMAND" |
awk '{print $3}'   #ubuntu使用ncat, 安装nmap
}
main(){
    case $1 in
        memcached_status)
            memcached_status $2 $3
                ;;
    esac
}

main $1 $2 $3

# chmod  a+x /etc/zabbix/zabbix_agentd.d/memcache_monitor.sh
# bash memcache_monitor.sh memcached_status 11211 curr_connections
1
```

5.2.3：zabbix agent添加自定义监控项：

```
# vim /etc/zabbix/zabbix_agentd.conf
298 UserParameter=memcache_status[*],/etc/zabbix/zabbix_agentd.d/memcache_monitor.sh
"$1" "$2" "$3"

# systemctl  restart zabbix-agent
```

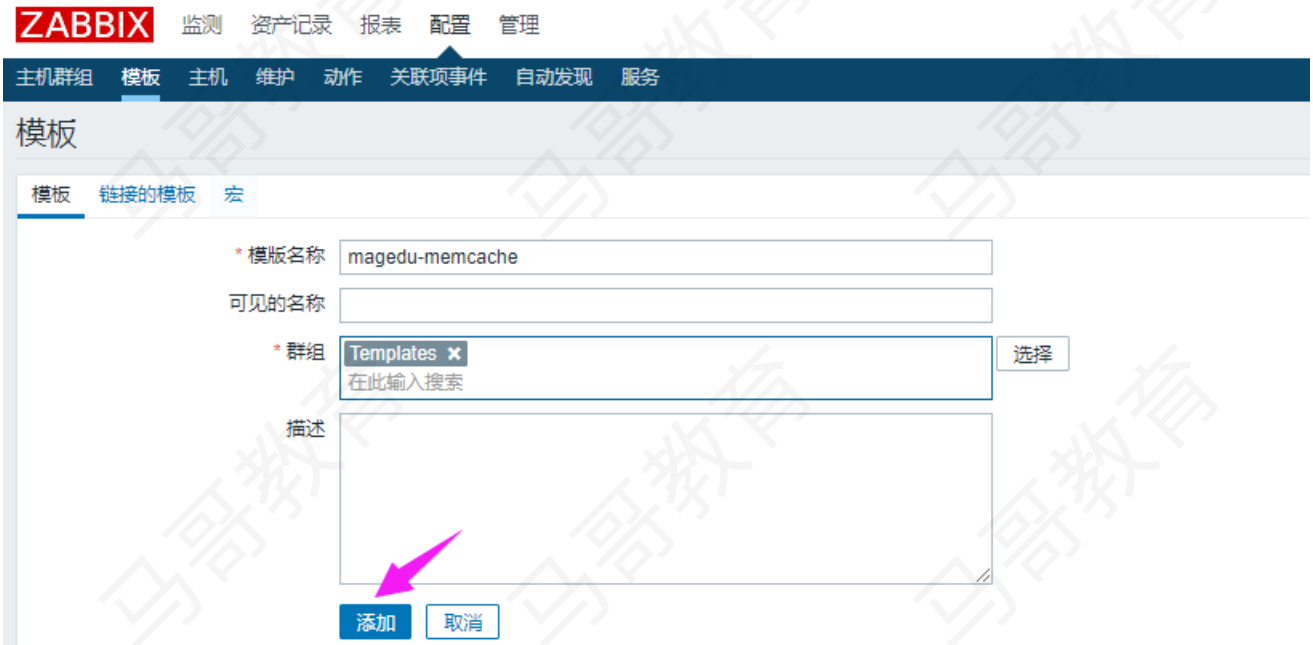
5.2.4：zabbix server测试监控项数据：

```
root@zabbix-server:~# /apps/zabbix_server/bin/zabbix_get  -s 172.31.0.107 -p 10050 -k "memcache_status["memcached_status","11211","curr_connections"]"
1
```

5.2.5：zabbix web制作模板：

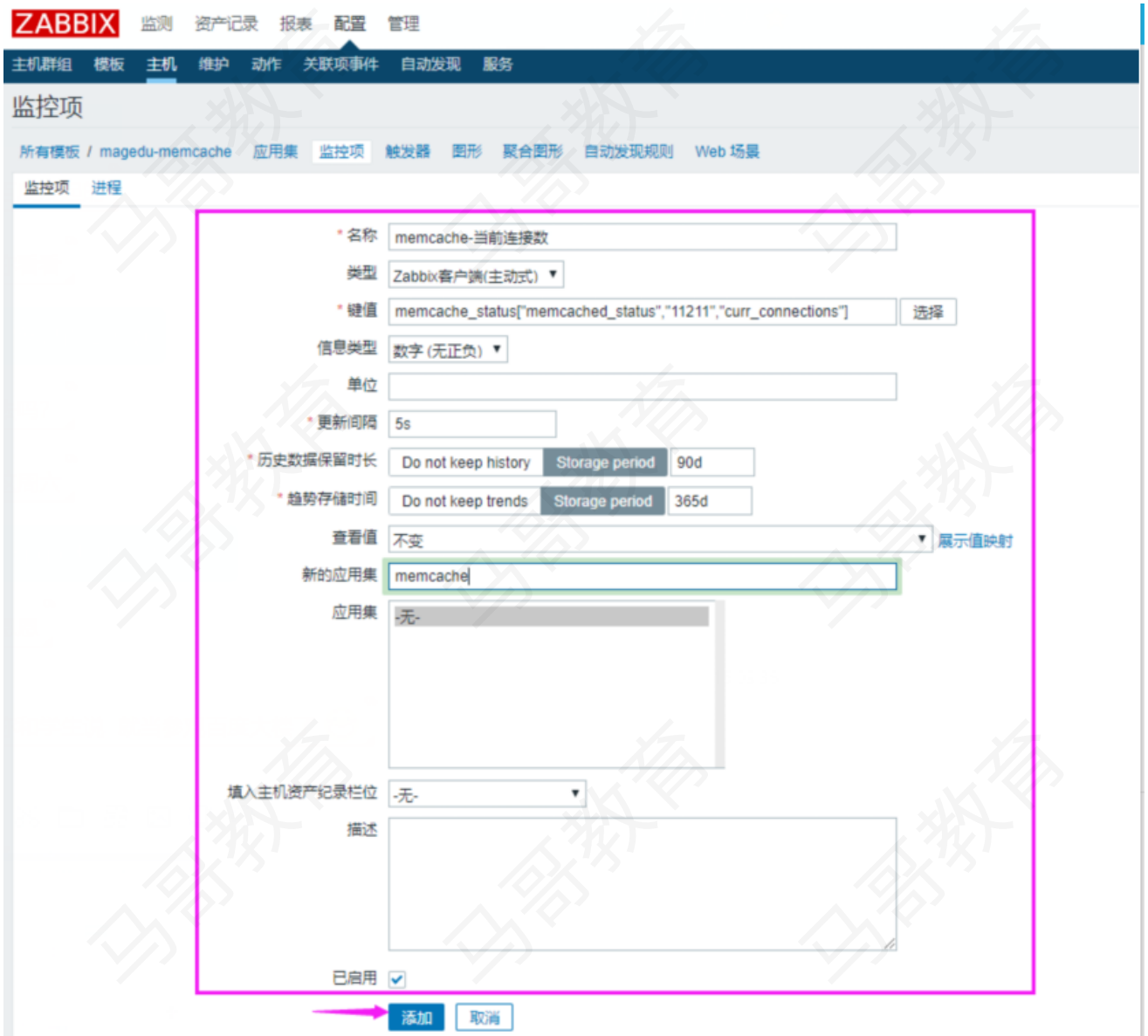
配置-模板-创建模板

5.2.5.1：创建模板：



5.2.5.2：创建监控项：

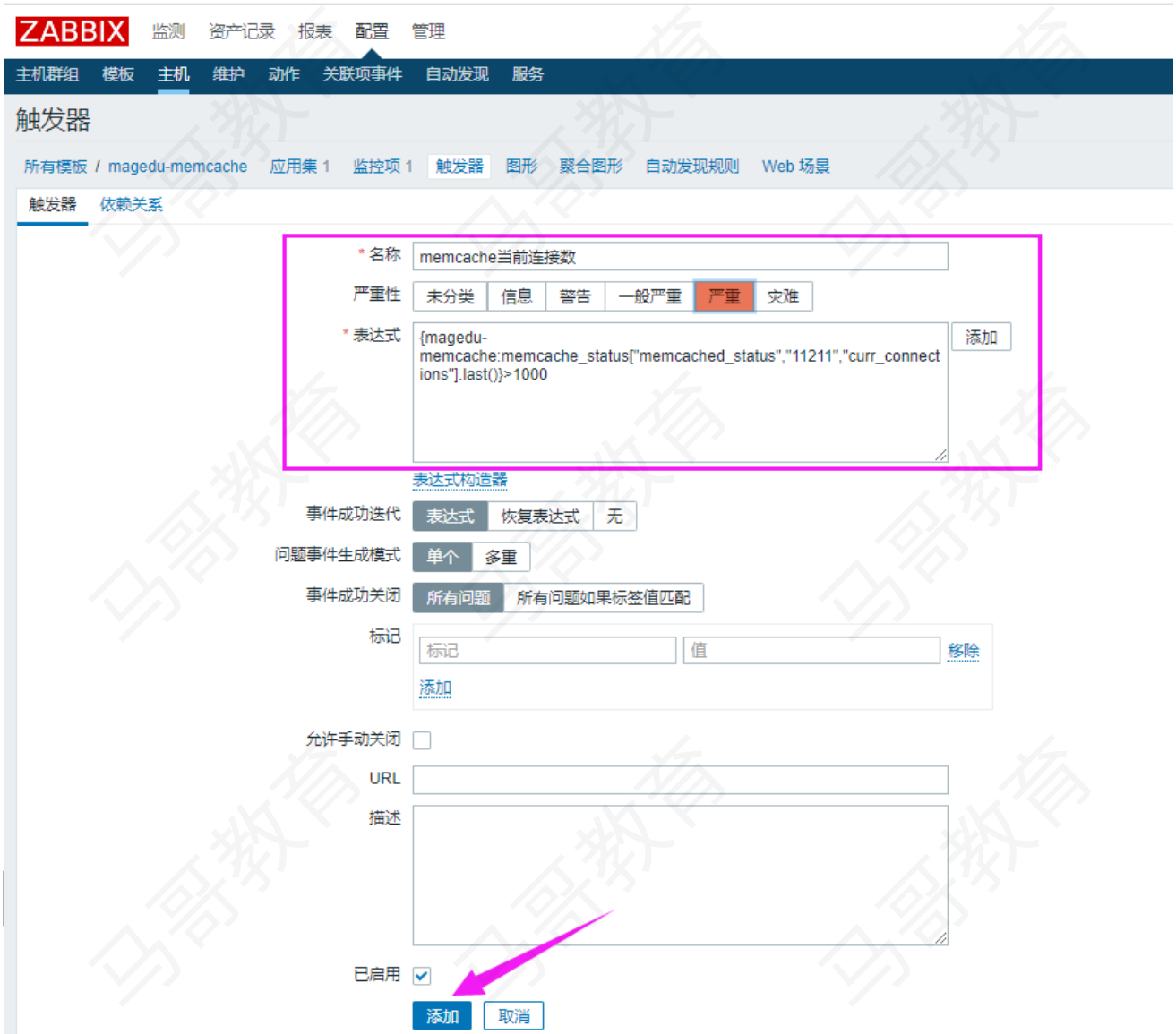
配置-模板-magedu-memcache-监控项-创建监控项：



5.2.5.3：创建触发器：

配置-模板-magedu-memcache-触发器-创建触发器：

点击表达式方框右侧的添加按钮，选择要对那个监控项设置触发器以及触发方式和值的大小，一个模板中可以有多个触发器，一个触发器是根据一个监控项的返回值对比预先设置的阈值，触发器就是监控项返回了不符合预定义的值范围后就进行触发下一步操作的警戒线。



5.2.5.4：创建图形：

配置-模板-magedu-memcache-图形-创建图形：

在图形里面关联监控项，一个图形可以用多个监控项，不同的监控项会自动使用不同的颜色进行区分，也可以手动调整各监控项的颜色和图形类型。

5.2.6：模板关联主机：

配置-主机-选择主机-模板：

5.2.7：验证监控项数据：

5.3：监控Redis：

学习一个模板中创建多个监控项、多个触发器和多个图形

5.3.1：安装Redis服务：

```
# apt install redis -y
# vim /etc/redis/redis.conf
# systemctl  restart redis
# systemctl enable  redis
```

5.3.2：监控脚本：

```
# pwd
/etc/zabbix/zabbix_agentd.d

# cat redis_monitor.sh
#!/bin/bash
#Author:Zhang ShiJie
redis_status(){
        R_PORT=$1
        R_COMMAND=$2
        (echo -en "INFO \r\n";sleep 1;) | ncat 127.0.0.1 "$R_PORT" >
/tmp/redis_"$R_PORT".tmp
        REDIS_STAT_VALUE=$(grep ""$R_COMMAND":" /tmp/redis_"$R_PORT".tmp | cut -d ':' -
f2)
        echo $REDIS_STAT_VALUE
}

help(){
        echo "${0} + redis_status + PORT + COMMAND"
}

main(){
    case $1 in
```

```
        redis_status)
            redis_status $2 $3
                ;;
        *)
            help
                ;;
        esac
}
main $1 $2 $

# chmod  a+x redis_monitor.sh
# bash redis_monitor.sh  redis_status 6379 used_memory
841272
```

5.3.3：zabbix agent添加自定义监控项：

```
299 UserParameter=redis_status[*],/etc/zabbix/zabbix_agentd.d/redis_monitor.sh "$1"
"$2" "$3"
# systemctl  restart zabbix-agent
```

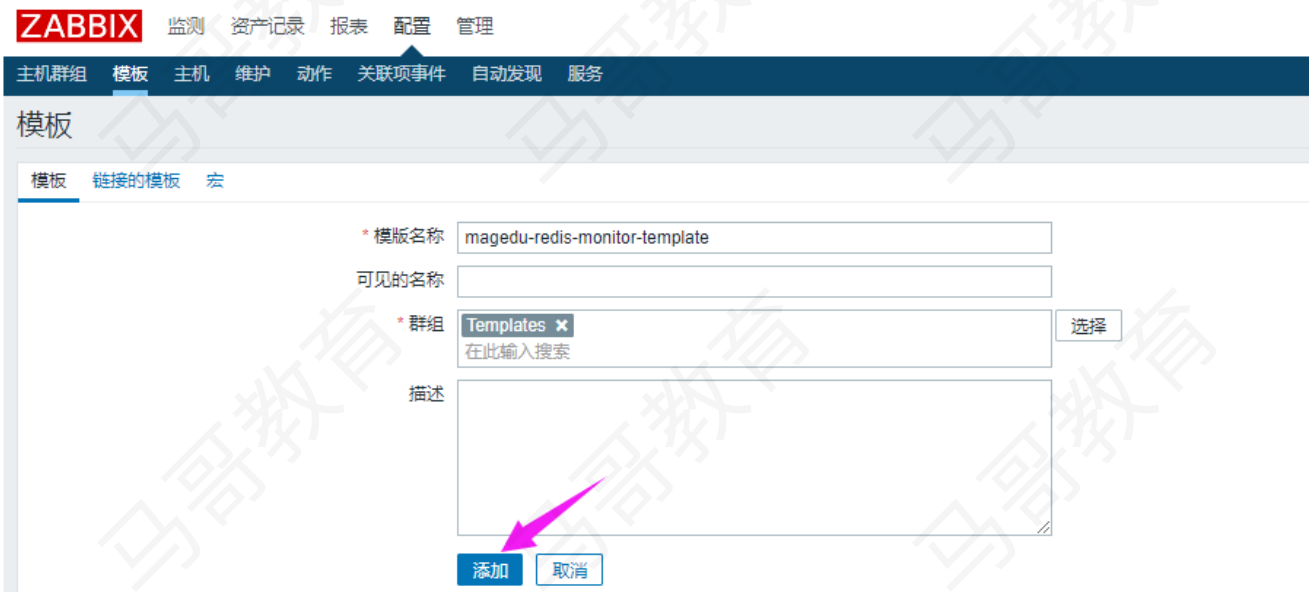
5.3.4：zabbix server测试监控项数据：

```
# /apps/zabbix_server/bin/zabbix_get  -s 172.31.0.107 -p 10050 -k
"redis_status["redis_status","6379","used_memory"]"
841272
root@zabbix-server:~# /apps/zabbix_server/bin/zabbix_get  -s 172.31.0.107 -p 10050 -k
"redis_status["redis_status","6379","connected_clients"]"
1
```

5.3.5：zabbix wen模板制作：

配置-模板-创建模板

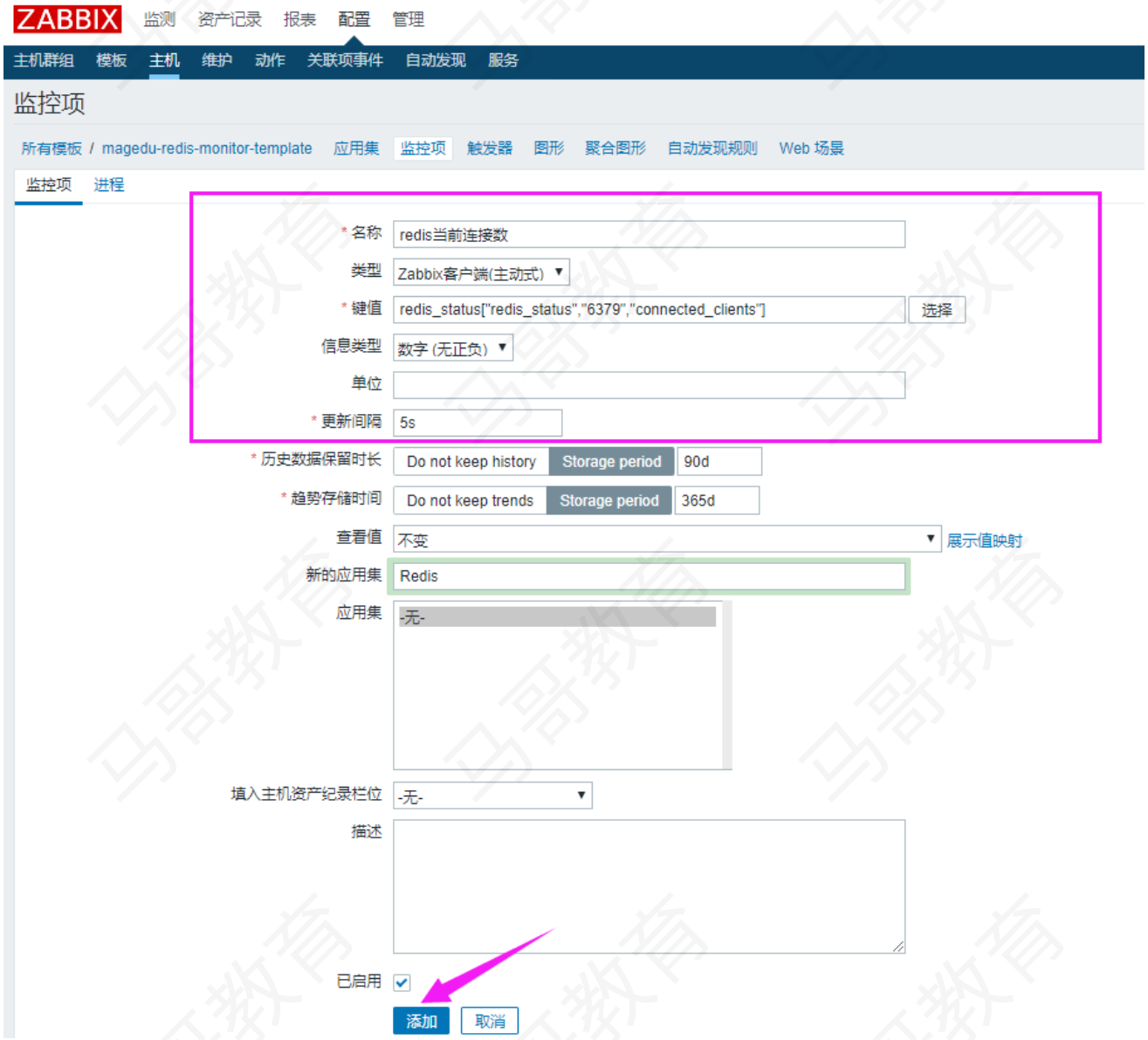
5.3.5.1：创建模板：



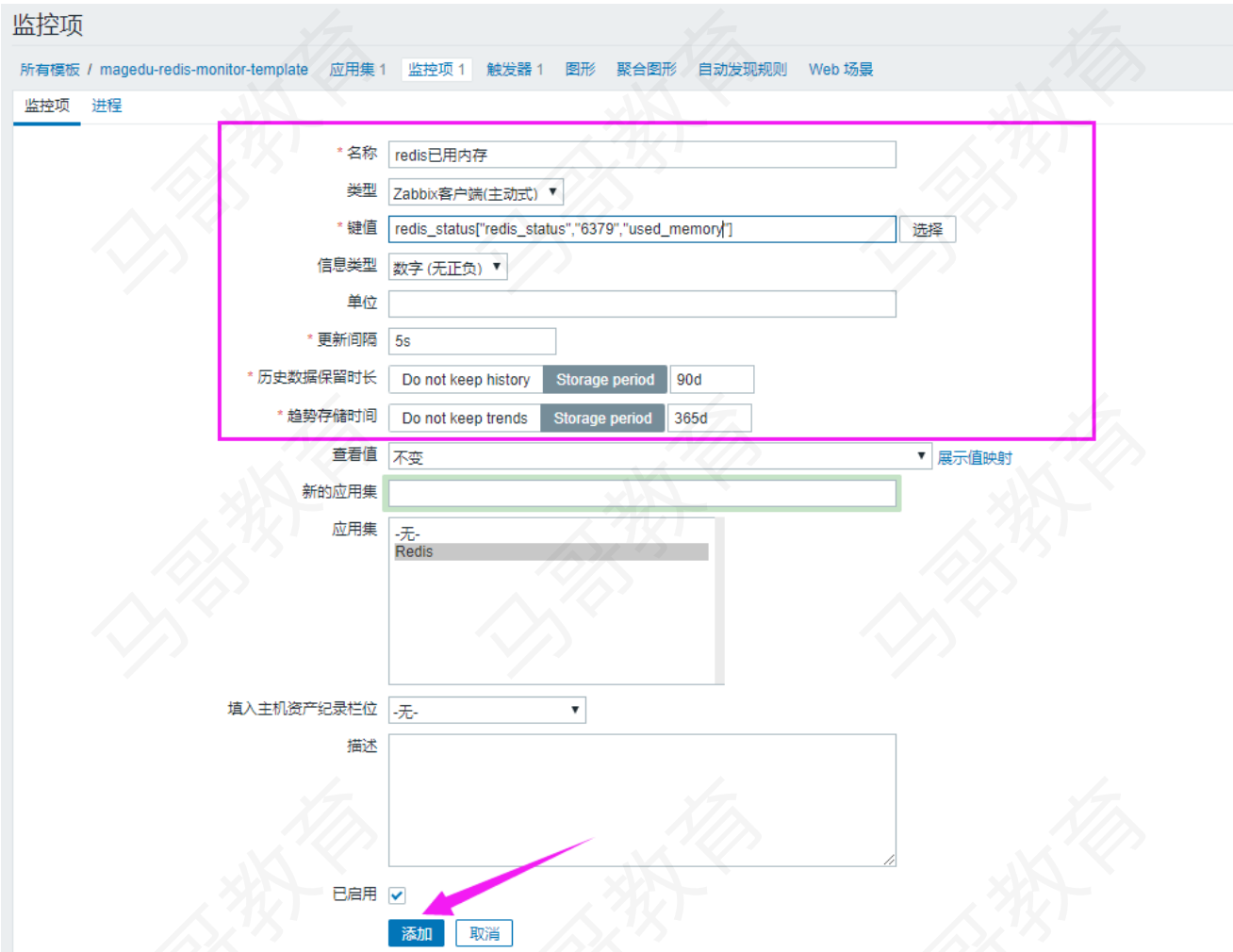
5.3.5.2：创建触监控项：

配置-模板-magedu-redis-monitor-template-监控项-创建监控项：

5.3.5.2.1：当前连接数监控项：



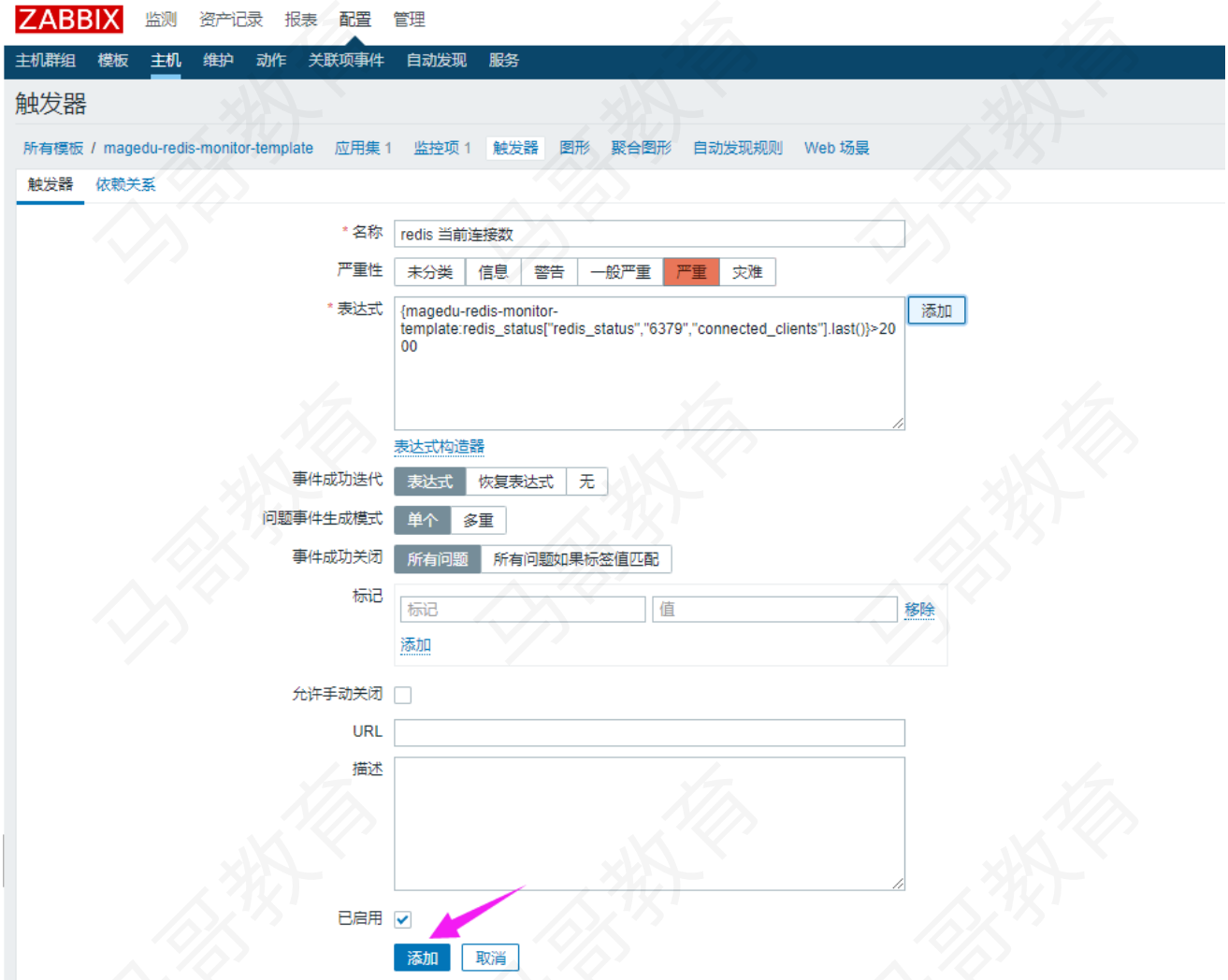
5.3.5.2.2：已用内存监控项：



5.3.5.3：创建触发器

配置-模板-magedu-redis-monitor-template-触发器-创建触发器：

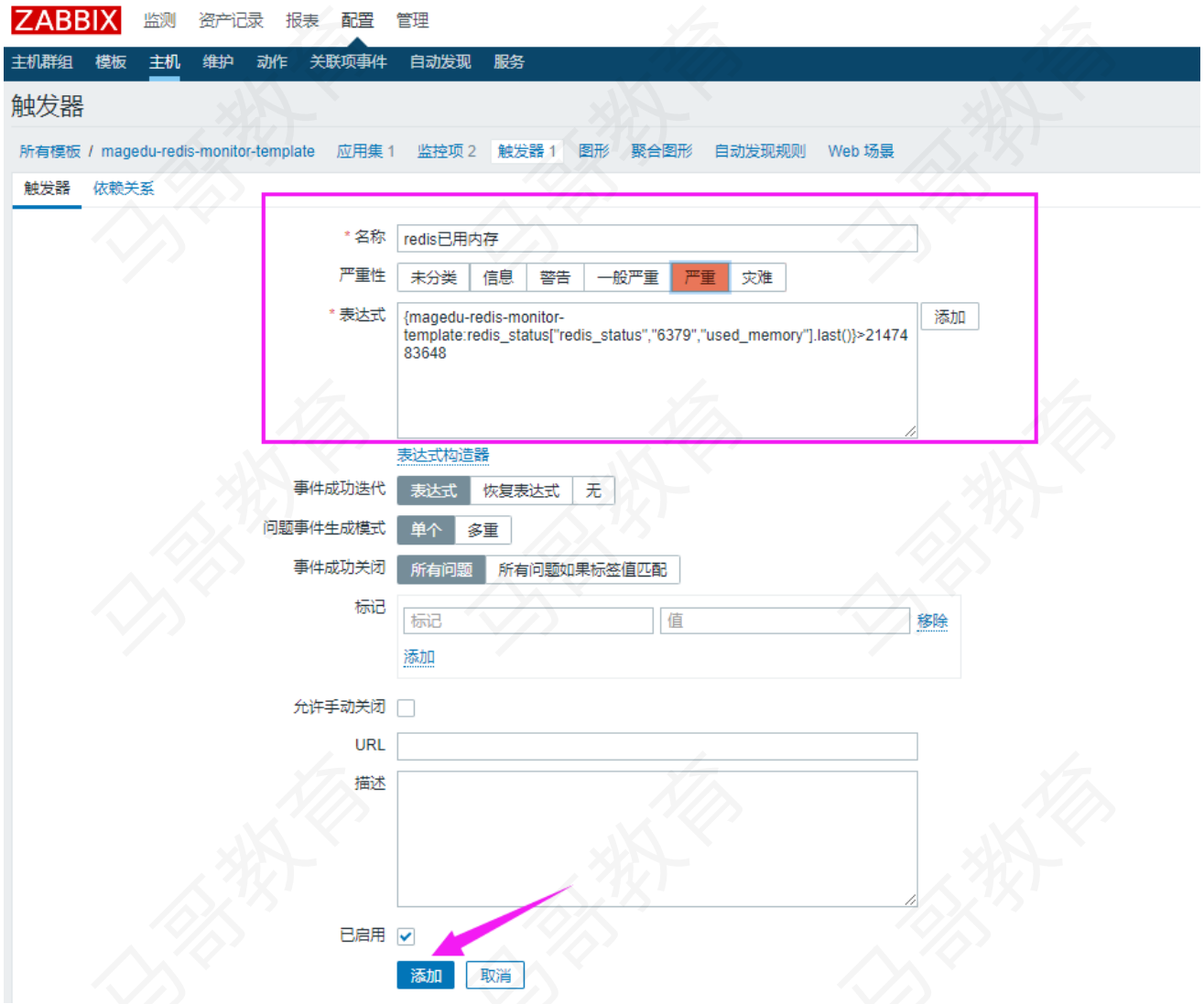
5.3.5.3.1：当前连接数触发器：



5.3.5.3.2：已用内存触发器：

需要将已用内存单位转换，从G转换为字节，例如当内存使用达2G时就进行触发，则换算方式如下：

```
2*1024*1024*1024
```



5.3.5.4：创建图形：

配置-模板-magedu-redis-monitor-template-图形-创建图形：

5.3.5.4.1：Redis当前连接数图形：

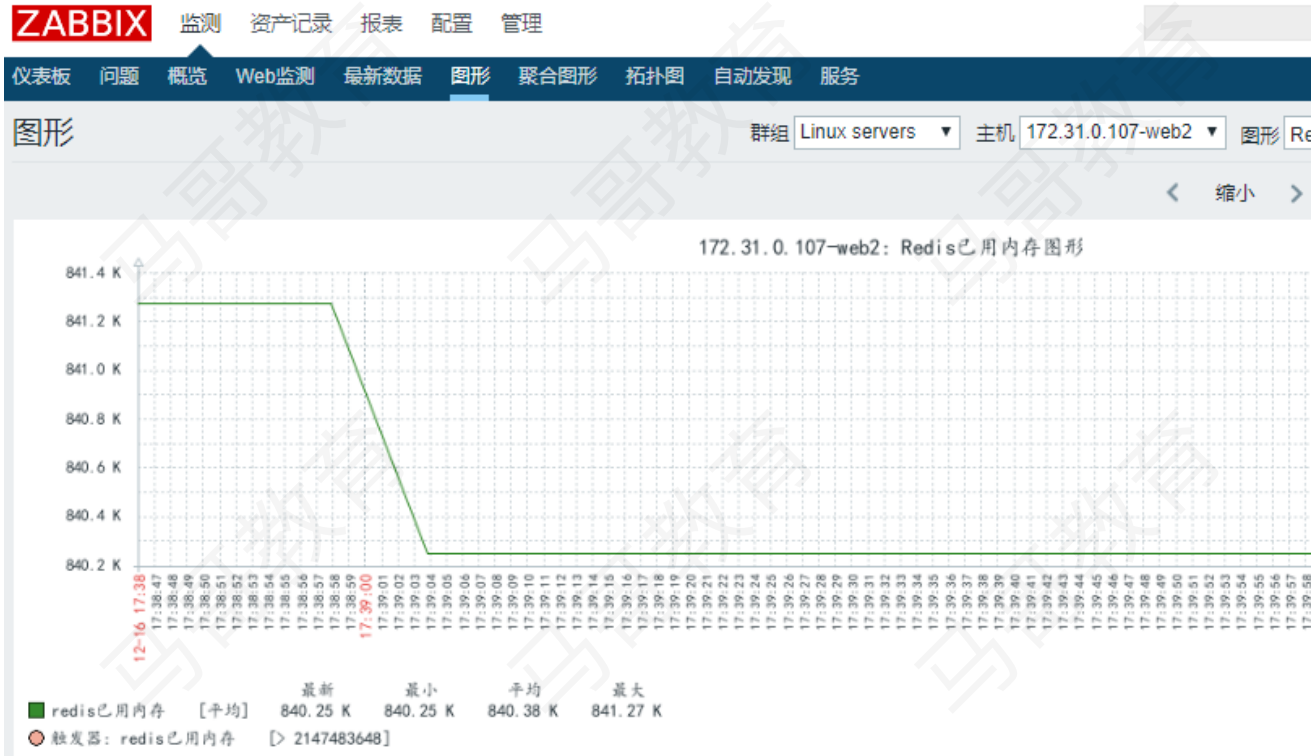
5.3.5.4.2：Redis已用内存图形：

5.3.6：模板关联主机：

5.3.7：验证监控项数据：

5.3.7.1：redis 当前连接数图形：

5.3.7.2：redis已用内存连接数：



5.4：监控Nginx：

```
脚本：nginx_status.sh
模板：nginx_status.yml
```

对nginx的活动连接和当前状态等运行状态进行监控

```
配置示例：
location  /nginx_status {
    stub_status;
    allow 172.31.0.0/16;
    allow 127.0.0.1;
    deny all;
  }

状态页用于输出nginx的基本状态信息：
   输出信息示例：
Active connections: 291
server accepts handled requests
    16630948 16630948 31070465
    上面三个数字分别对应accepts,handled,requests三个值
Reading: 6 Writing: 179 Waiting: 106

Active connections：  当前处于活动状态的客户端连接数，包括连接等待空闲连接数。
accepts：统计总值，Nginx自启动后已经接受的客户端请求的总数。
handled：统计总值，Nginx自启动后已经处理完成的客户端请求的总数，通常等于accepts，除非有因
worker_connections限制等被拒绝的连接。
requests：统计总值，Nginx自启动后客户端发来的总的请求数。
```

```
Reading: 当前状态，正在读取客户端请求报文首部的连接的连接数。
Writing: 当前状态，正在向客户端发送响应报文过程中的连接数。
Waiting: 当前状态，正在等待客户端发出请求的空闲连接数，开启 keep-alive的情况下,这个值等于 active –
(reading+writing),
```

5.4.1：部署Nginx服务：

如果是编译安装需要添加编译参数--with-http_stub_status_module

```
# pwd
/usr/local/src
# wget http://nginx.org/download/nginx-1.16.1.tar.gz
# tar xf  nginx-1.16.1.tar.gz
# ./configure  --prefix=/apps/nginx --with-http_stub_status_module
# make && make install

# vim /apps/nginx/conf/nginx.conf

        location / {
            root   html;
            index  index.html index.htm;
        }

        location  /nginx_status {
          stub_status;
          allow 172.31.0.0/16;
          allow 127.0.0.1;

# /apps/nginx/sbin/nginx  -t
nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /apps/nginx/conf/nginx.conf test is successful
# /apps/nginx/sbin/nginx

# curl  http://127.0.0.1/nginx_status
Active connections: 1
server accepts handled requests
 1 1 1
Reading: 0 Writing: 1 Waiting: 0
```

5.4.2：监控项脚本：

```
# pwd
/etc/zabbix/zabbix_agentd.d

# cat nginx_status.sh
#!/bin/bash
#Date:2016/11/11
#Author: Zhangshijie

nginx_status_fun(){ #函数内容
    NGINX_PORT=$1 #端口，函数的第一个参数是脚本的第二个参数，即脚本的第二个参数是段端口号
    NGINX_COMMAND=$2 #命令，函数的第二个参数是脚本的第三个参数，即脚本的第三个参数是命令
```

```
    nginx_active(){ #获取nginx_active数量，以下相同，这是开启了nginx状态但是只能从本机看到
        /usr/bin/curl "http://127.0.0.1:"$NGINX_PORT"/nginx_status/" 2>/dev/null| grep
'Active' | awk '{print $NF}'
        }
    nginx_reading(){ #获取nginx_reading状态的数量
        /usr/bin/curl "http://127.0.0.1:"$NGINX_PORT"/nginx_status/" 2>/dev/null| grep
'Reading' | awk '{print $2}'
        }
    nginx_writing(){
        /usr/bin/curl "http://127.0.0.1:"$NGINX_PORT"/nginx_status/" 2>/dev/null| grep
'Writing' | awk '{print $4}'
       }
    nginx_waiting(){
        /usr/bin/curl "http://127.0.0.1:"$NGINX_PORT"/nginx_status/" 2>/dev/null| grep
'Waiting' | awk '{print $6}'
       }
    nginx_accepts(){
        /usr/bin/curl "http://127.0.0.1:"$NGINX_PORT"/nginx_status/" 2>/dev/null| awk
NR==3 | awk '{print $1}'
       }
    nginx_handled(){
        /usr/bin/curl "http://127.0.0.1:"$NGINX_PORT"/nginx_status/" 2>/dev/null| awk
NR==3 | awk '{print $2}'
       }
    nginx_requests(){
        /usr/bin/curl "http://127.0.0.1:"$NGINX_PORT"/nginx_status/" 2>/dev/null| awk
NR==3 | awk '{print $3}'
       }
    case $NGINX_COMMAND in
       active)
           nginx_active;
           ;;
       reading)
           nginx_reading;
           ;;
       writing)
           nginx_writing;
           ;;
       waiting)
           nginx_waiting;
           ;;
       accepts)
           nginx_accepts;
           ;;
       handled)
           nginx_handled;
           ;;
       requests)
           nginx_requests;
       esac
}

main(){ #主函数内容
```

```
    case $1 in #分支结构，用于判断用户的输入而进行响应的操作
        nginx_status) #当输入nginx_status就调用nginx_status_fun，并传递第二和第三个参数
            nginx_status_fun $2 $3;
            ;;
        *) #其他的输入打印帮助信息
            echo $"Usage: $0 {nginx_status key}"
    esac #分支结束符
}

main $1 $2 $3

# chmod  a+x nginx_status.sh
# bash nginx_status.sh  nginx_status 80 active
1
```

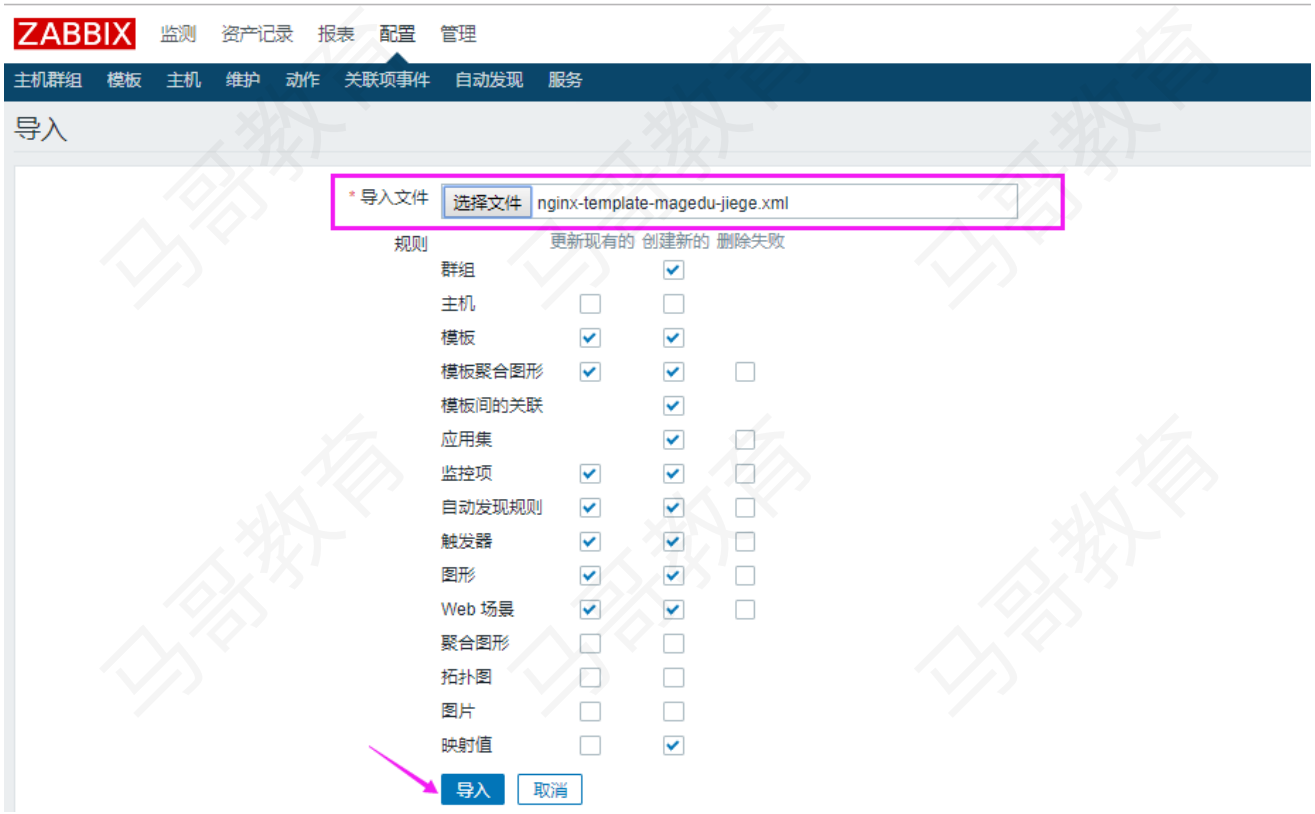
5.4.3：zabbix agent添加自定义监控项：

```
# vim /etc/zabbix/zabbix_agentd.conf
300 UserParameter=nginx_status[*],/etc/zabbix/zabbix_agentd.d/nginx_status.sh "$1" "$2"
"$3"
# systemctl  restart zabbix-agent
```

5.4.4：zabbix server测试监控项数据：

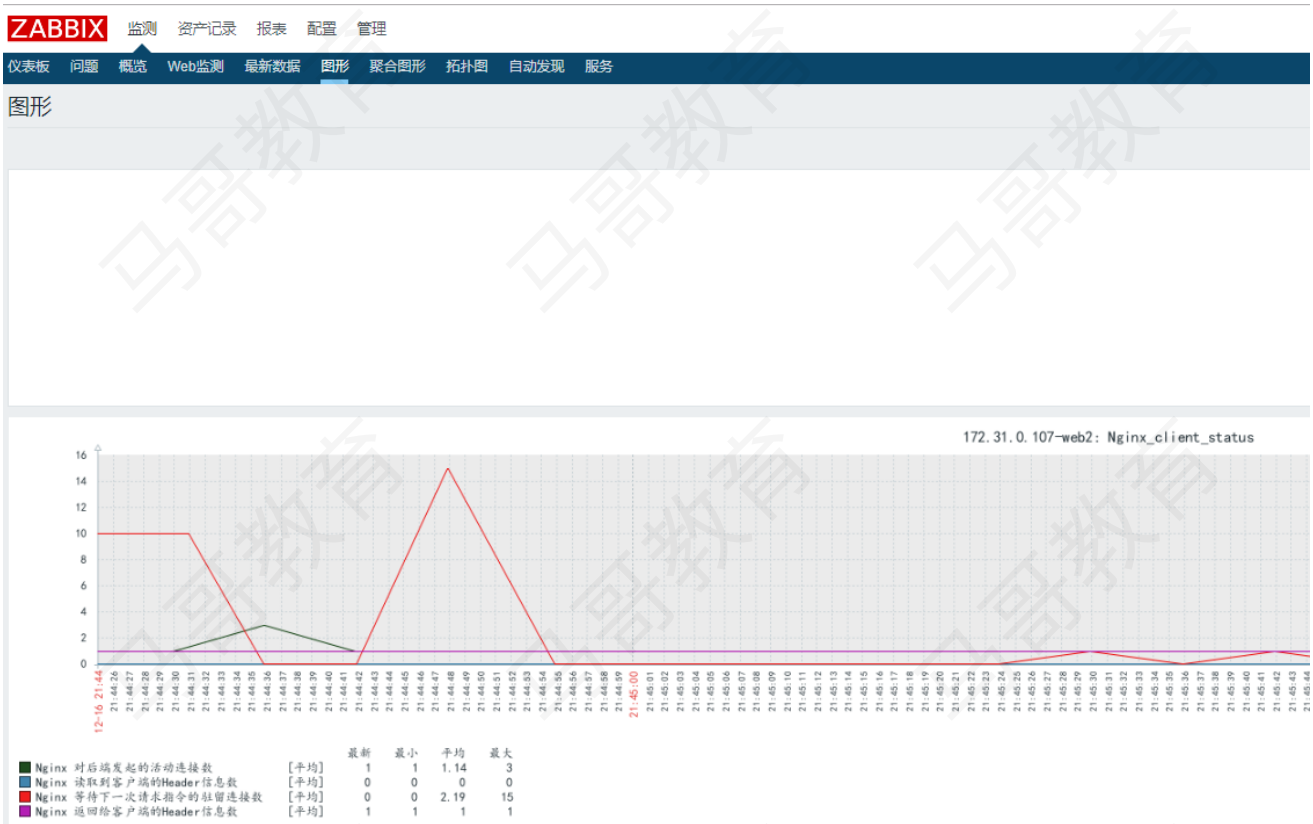
```
# /apps/zabbix_server/bin/zabbix_get  -s 172.31.0.107 -p 10050 -k
"nginx_status["nginx_status",80,"active"]"
1
```

5.4.5：导入Nginx监控模板：



5.4.6：模板关联主机：

5.4.7：验证监控数据：



5.5：SNMP监控：

SNMP是英文"Simple Network Management Protocol"的缩写，中文意思是“简单网络管理协议，SNMP是一种简单网络管理协议，它属于TCP/IP五层协议中的应用层协议，用于网络管理的协议，SNMP主要用于网络设备的理。

SNMP的基本思想：为不同种类的设备、不同厂家生产的设备、不同型号的设备，定义为一个统一的接口和协议，使得管理员可以是使用统一的外观面对这些需要管理的网络设备进行管理。通过网络，管理员可以管理位于不同物理空间的设备，从而大大提高网络管理的效率，简化网络管理员的工作。

SNMP协议是TCP/IP协议簇的一个应用层协议，在1988年被制定，并被Internet体系结构委员会（IAB）采纳作为一个短期的网络管理解决方案，由于SNMP的简单性，在Internet时代得到了蓬勃的发展，1992年发布了SNMPv2版本，以增强SNMPv1的安全性和功能，SNMP的协议版本目前有SNMP v1、SNMP v2c和SNMP v3三种版本，其具体差别如下：

> SNMP v1采用团体名（Community Name）认证，团体名用来定义SNMP NMS和SNMP Agent的关系，如果SNMP报文携带的团体名没有得到设备的认可，该报文将被丢弃，团体名起到了类似于密码的作用，用来限制SNMP NMS对SNMP Agent的访问。
>
> SNMP v2c也采用团体名认证，它在兼容SNMP v1的同时又扩充了SNMP v1的功能，它提供了更多的操作类型（GetBulk和InformRequest）、支持更多的数据类型（Counter64等）、提供了更丰富的错误代码且能够更细致地区分错误。
>
> SNMP v3提供了基于用户的安全模型（USM，User-Based Security Model）的认证机制，用户可以设置认证和加密功能，认证用于验证报文发送方的合法性，避免非法用户的访问，加密则是对NMS和Agent之间的传输报文进行加密，以免被窃听。通过有无认证和有无加密等功能组合，可以为SNMP NMS和SNMP Agent之间的通信提供更高的安全性。

5.5.1：SNMP组织机构：

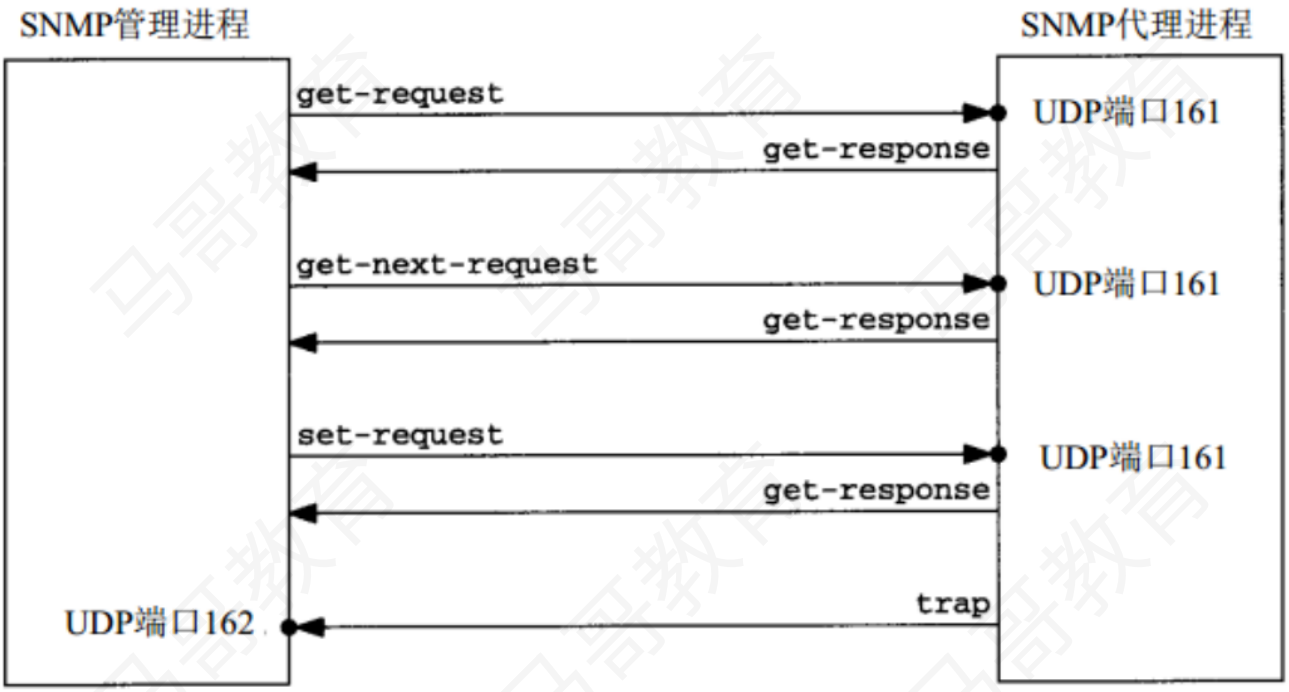
SNMP网络元素分为NMS和Agent两种：

```
NMS（Network Management Station，网络管理站）是运行SNMP客户端管理程序的工作站，能够提供非常友好的人机交互界面，方便网络管理员完成绝大多数的网络管理工作。

Agent是驻留在设备上的一个进程，负责接收、处理来自NMS的请求报文。在一些紧急情况下，如接口状态发生改变等，Agent也会主动通知NMS。

NMS是SNMP网络的管理者，Agent是SNMP网络的被管理者。NMS和Agent之间通过SNMP协议来交互管理信息。
```

5.5.2：SNMP数据交互：

SNMP管理进程与代理进程之前为了交互信息，定义了5种报文：

```
get-request操作：从代理进程处提取一个或多个参数值。
get-response操作：返回的一个或多个参数值。这个操作是由代理进程发出的。
trap操作：代理进程主动发出的报文，通知管理进程有某些事情发生。
get-next-request操作：从代理进程处提取一个或多个参数的下一个参数值。
set-request操作：设置代理进程的一个或多个参数值。
```



5.5.3：SNMP组织结构：

一套完整的SNMP系统主要包括以下几个方面：

SNMP报文协议。

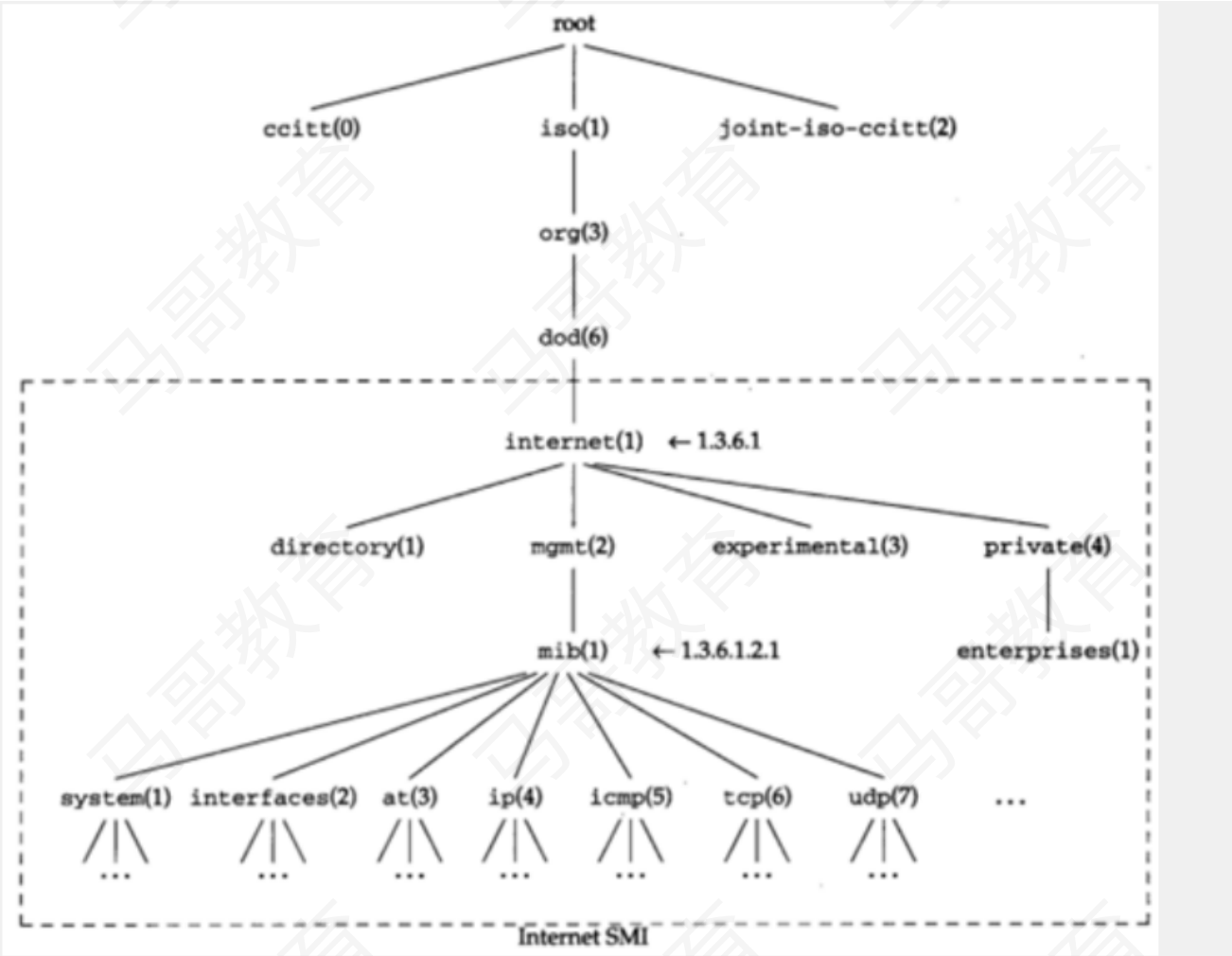
管理信息结构（SMI， Structure ofManagement Information），一套公用的结构和表示符号。

管理信息库（MIB，Management Information Base），管理信息库包含所有代理进程的所有可被查询和修改的参数。

OID（Object Identifiers），一个OID是一个唯一的键值对，用于标识具体某一个设备的某个具体信息(对象标识)，如端口信息、设备名称等。

5.5.4：SNMP MIB：

所谓(MIB)管理信息库，就是所有代理进程包含的、并且能够被管理进程进行查询和设置的信息的集合。MIB是基于对象标识树的，对象标识是一个整数序列，中间以"."分割，这些整数构成一个树型结构，类似于 DNS或Unix的文件系统，MIB被划分为若干个组，如system、 interfaces、 a t（地址转换）和ip组等。iso . org . dod . internet .private . enterprises（1.3 .6 .1.4.1）这个标识，是给厂家自定义而预留的，比如华为的为1.3.6.1.4.1.2011，华三的为1.3.6.1.4.1.25506。



5.5.5：基于Centos的SNMP：

Centos 7.x服务器安装配置SNMP：

```
[root@s6 ~]# yum install -y net-snmp / # apt-get install  snmpd
[root@s6 ~]# vim /etc/snmp/snmpd.conf
#       sec.name  source          community
com2sec notConfigUser  default       123456  #第一步：设置团体认密码，默认为public

group   notConfigGroup v1           notConfigUser
group   notConfigGroup v2c           notConfigUser #第二步：将团体名称notConfigUser 关联至组notConfigGroup

view    systemview    included   .1.3.6.1.2.1.1
view    systemview    included   .1.3.6.1.2.1.25.1.1 #创建一个view，并对其授权可访问的OID范围
view    systemview    included   .1.  #自定义授权，否则zabbix server无法获取数据

access  notConfigGroup “”      any       noauth    exact  systemview none none #将组notConfigGroup关联至systemview  从而完成对组的授权

[root@s6 ~]# systemctl  restart snmpd
```

5.5.6：SNMP OID：

如何测试OID：

snmpwalk是SNMP的一个工具，它使用SNMP的GET请求查询指定OID（SNMP协议中的对象标识）入口的所有OID树信息，并显示给用户。通过snmpwalk也可以查看支持SNMP协议（可网管）的设备的一些其他信息，比如cisco交换机或路由器IP地址、内存使用率等，也可用来协助开发SNMP功能。

要使用snmpwalk需要先按照net-snmp软件包中。

```
[root@s1 ~]# yum -y install net-snmp-utils
[root@s1 ~]# snmpwalk  -h
USAGE: snmpwalk [OPTIONS] AGENT [OID]
-h：显示帮助。
-v：指定snmp的版本，1或者2c或者3。
-c：指定连接设备SNMP密码。
-V：显示当前snmpwalk命令行版本。
-r：指定重试次数，默认为0次。
-t：指定每次请求的等待超时时间，单为秒，默认为3秒。
-l：指定安全级别：noAuthNoPriv|authNoPriv|authPriv。
-a：验证协议：MD5|SHA。只有-l指定为authNoPriv或authPriv时才需要。
-A：验证字符串。只有-l指定为authNoPriv或authPriv时才需要。
-x：加密协议：DES。只有-l指定为authPriv时才需要。
-X：加密字符串。只有-l指定为authPriv时才需要。
```

5.5.7：测试SNMP数据采集：

测试能否通过SNMOP采集数据：

```
[root@s1 ~]# snmpwalk  -v 2c  -c 123456  172.18.200.106  .1.3.6.1.4.1.2021.10.1.3.1
UCD-SNMP-MIB::laLoad.1 = STRING: 0.00
[root@s1 ~]# snmpwalk  -v 2c  -c 123456  172.18.200.106  .1.3.6.1.4.1.2021.4.3.0
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 1048572 kB
```

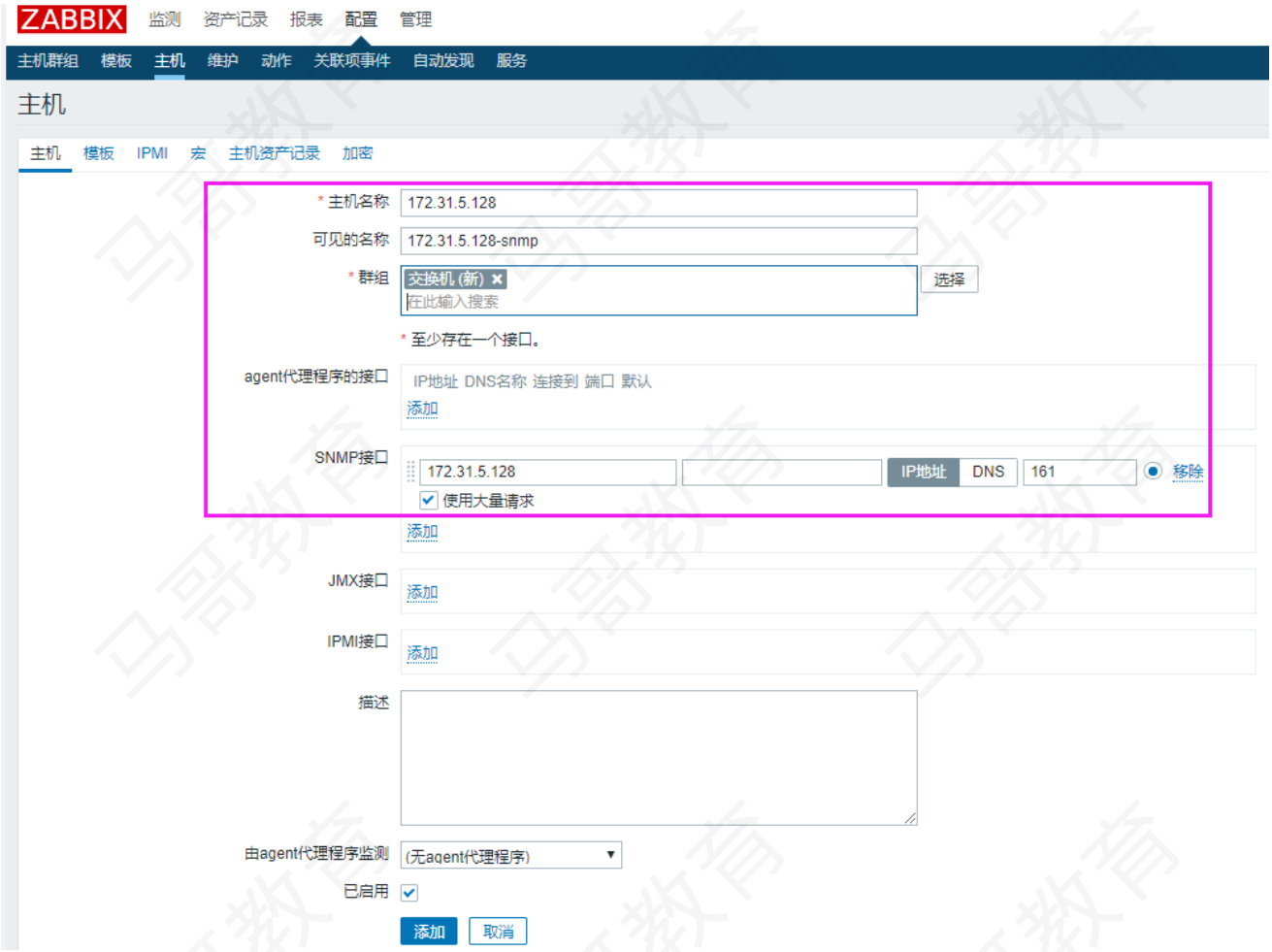
5.5.8：Centos SNMP：

```
CPU 负载:
1 minute Load: .1.3.6.1.4.1.2021.10.1.3.1
5 minute Load: .1.3.6.1.4.1.2021.10.1.3.2
15 minute Load: .1.3.6.1.4.1.2021.10.1.3.3


内存使用:
Total Swap Size: .1.3.6.1.4.1.2021.4.3.0
Available Swap Space: .1.3.6.1.4.1.2021.4.4.0
Total RAM in machine: .1.3.6.1.4.1.2021.4.5.0
Total RAM used: .1.3.6.1.4.1.2021.4.6.0
Total RAM Free: .1.3.6.1.4.1.2021.4.11.0
Total RAM Shared: .1.3.6.1.4.1.2021.4.13.0
Total RAM Buffered: .1.3.6.1.4.1.2021.4.14.0
Total Cached Memory: .1.3.6.1.4.1.2021.4.15.0

硬盘使用:
Path where the disk is mounted: .1.3.6.1.4.1.2021.9.1.2.1
Path of the device for the partition: .1.3.6.1.4.1.2021.9.1.3.1
Total size of the disk/partion (kBytes): .1.3.6.1.4.1.2021.9.1.6.1
Available space on the disk: .1.3.6.1.4.1.2021.9.1.7.1
Used space on the disk: .1.3.6.1.4.1.2021.9.1.8.1
Percentage of space used on disk: .1.3.6.1.4.1.2021.9.1.9.1
Percentage of inodes used on disk: .1.3.6.1.4.1.2021.9.1.10.1

系统信息:
sysDescr 1.3.6.1.2.1.1.1
sysObjectID 1.3.6.1.2.1.1.2
sysUpTime 1.3.6.1.2.1.1.3
sysContact 1.3.6.1.2.1.1.4
sysName 1.3.6.1.2.1.1.5

CPU 信息:
percentage of user CPU time: .1.3.6.1.4.1.2021.11.9.0
raw user cpu time: .1.3.6.1.4.1.2021.11.50.0
percentages of system CPU time: .1.3.6.1.4.1.2021.11.10.0
raw system cpu time: .1.3.6.1.4.1.2021.11.52.0
percentages of idle CPU time: .1.3.6.1.4.1.2021.11.11.0
raw idle cpu time: .1.3.6.1.4.1.2021.11.53.0
raw nice cpu time: .1.3.6.1.4.1.2021.11.51.0
```

5.5.9：添加SNMP监控主机：

使用zabbix 自带的SNMP

5.5.9.1：添加SNMP主机：



5.5.9.2：关联SNMP模板：

5.5.10：验证主机当前状态：

5.5.11：验证SNMP监控数据：

web监控

幻灯片

聚合图形

邮件报警

5.6：监控MySQL：

监控MySQL连接数、主从同步、同步延迟等。

5.6.1：实现MySQL主从：

部署mysql 主从同步，需要配置不同的server id并开启binlog

5.6.1.1：MySQL Master：

```
MySQL 5.7.x:
# vim /etc/mysql/mysql.conf.d/mysqld.cnf
bind-address            = 0.0.0.0
server-id = 10
log-bin = /var/lib/mysql/master-log

MySQL 5.6.x:
# cat  /etc/my.cnf
[mysqld]
socket=/var/lib/mysql/mysql.sock
user=mysql
symbolic-links=0
datadir=/data/mysql
innodb_file_per_table=1
server-id=10
log-bin=/data/mysql/master-log

[client]
port=3306
socket=/var/lib/mysql/mysql.sock

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/tmp/mysql.sock
```

5.6.1.2：MySQL Slave：

```
MySQL 5.7.x:
# vim /etc/mysql/mysql.conf.d/mysqld.cn
bind-address            = 0.0.0.0
server-id = 105
relay-log = /var/lib/mysql/relay-log


MySQL 5.6.x:
# cat /etc/my.cnf
[mysqld]
socket=/var/lib/mysql/mysql.sock
user=mysql
symbolic-links=0
datadir=/data/mysql
innodb_file_per_table=1
relay-log = /data/mysql
server-id=20
```

```
[client]
port=3306
socket=/var/lib/mysql/mysql.sock

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/tmp/mysql.sock
```

5.6.1.3：MySQL Master授权账户：

在mysql master服务器授权账户并导出数据，然后scp到mysql backup服务器。

```
MySQL Master授权账户：
root@zabbix-mysql-master:~# mysql
mysql> GRANT REPLICATION SLAVE,REPLICATION CLIENT ON *.* TO 'magedu'@'172.31.0.%'
IDENTIFIED BY '123456';
Query OK, 0 rows affected, 1 warning (0.00 sec

root@zabbix-mysql-master:~# mysqldump --all-databases  --single_transaction --flush-
logs --master-data=2 --lock-tables > /opt/backup.sql
# scp /opt/backup.sql  172.31.0.105:/opt/
```

5.6.1.4：MySQL slave导入数据：

在MySQL Slave服务器导入数据开始同步数据，Position位置在sql文件：

```
root@zabbix-mysql-slave:~# mysql < /opt/backup.sql
root@zabbix-mysql-slave:~# mysql
mysql> CHANGE MASTER TO
 MASTER_HOST='172.31.0.104',MASTER_USER='magedu',MASTER_PASSWORD='123456',MASTER_LOG_FI
LE='master-log.000003',MASTER_LOG_POS=154;
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> start slave;
Query OK, 0 rows affected (0.00 sec)

mysql> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.31.0.104
                  Master_User: magedu
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-log.000003
          Read_Master_Log_Pos: 980279
               Relay_Log_File: relay-log.000002
                Relay_Log_Pos: 980446
        Relay_Master_Log_File: master-log.000003
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

5.6.2：Procona监控MySQL：

官方文档及下载地址：

```
https://www.percona.com/doc/percona-monitoring-plugins/LATEST/zabbix/index.html  #插件地址
https://www.percona.com/downloads/  #安装包下载地址
https://www.percona.com/doc/percona-monitoring-plugins/LATEST/zabbix/index.html#installation-instructions  #安装教程
```

5.6.2.1：MySQL Master安装zabbix-agent：

```
root@zabbix-mysql-master:~# apt install zabbix-agent
root@zabbix-mysql-master:~# grep "^[a-Z]" /etc/zabbix/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
Server=172.31.0.101
StartAgents=5
ServerActive=127.0.0.1
Hostname=172.31.0.104
Include=/etc/zabbix/zabbix_agentd.d/*.conf

root@zabbix-mysql-master:~# systemctl  restart zabbix-agent
root@zabbix-mysql-master:~# systemctl  enable  zabbix-agent
```

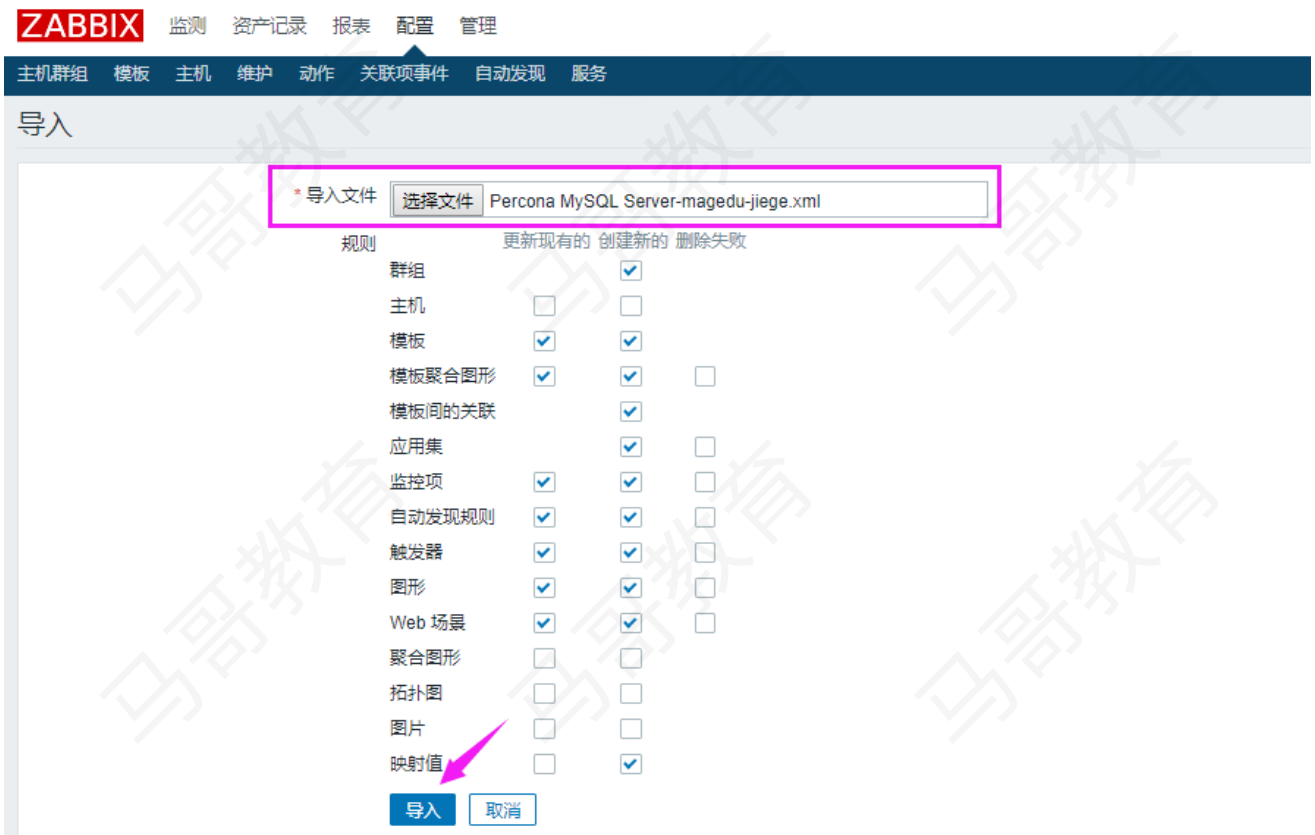
5.6.2.2：MySQL Master安装Percona：

修改zabbix agent启动用户为root，包括zabbix agent配置文件和服务启动文件。

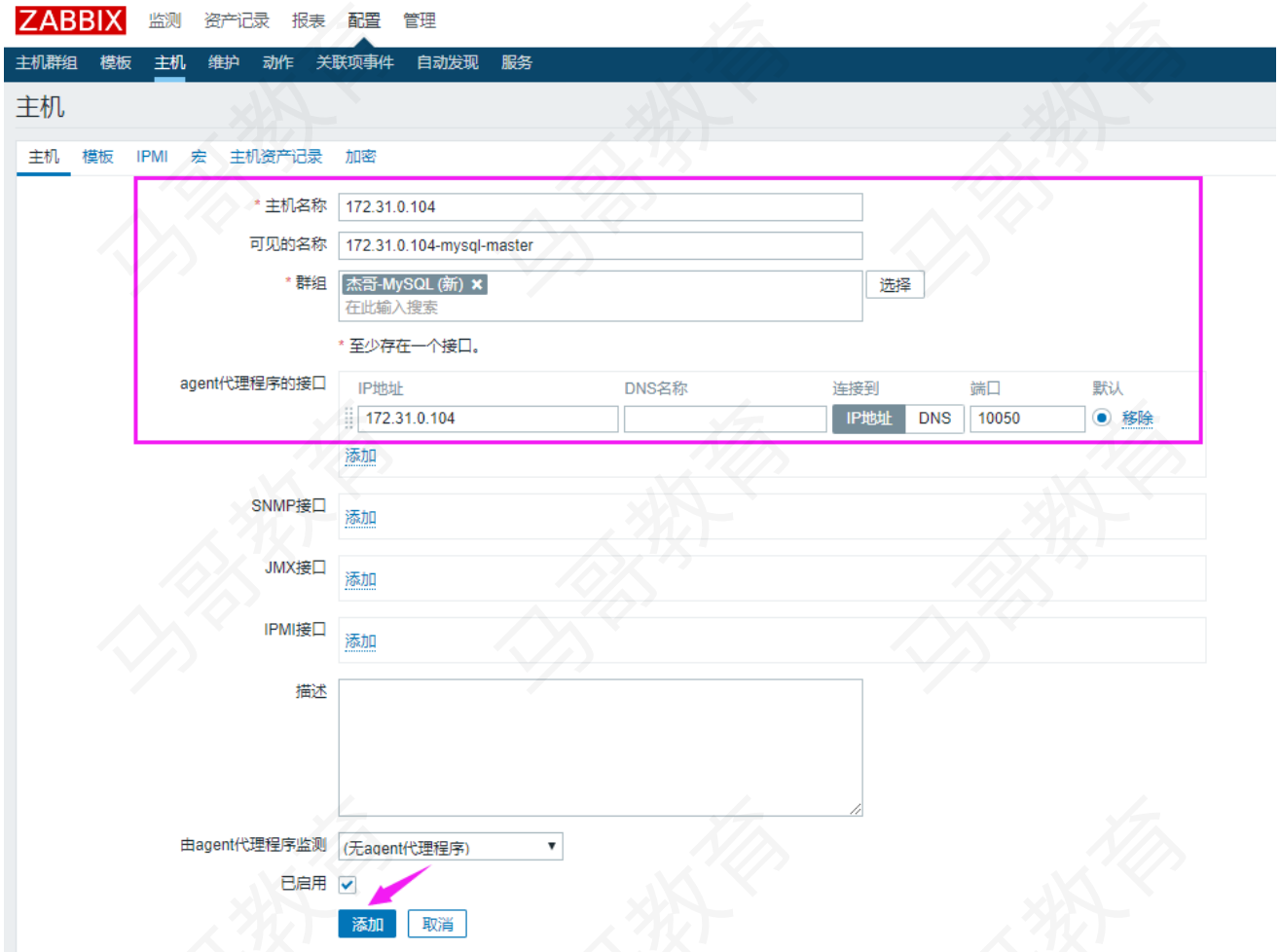
```
安装Percona软件包：
# dpkg -i percona-zabbix-templates_1.1.8-1.artful_all.deb
# cp /var/lib/zabbix/percona/templates/userparameter_percona_mysql.conf /etc/zabbix/zabbix_agentd.d/
# systemctl  restart zabbix-agent

安装php环境：  目前Percona与ubuntu 自带的php 7.2不兼容，需要安装php 5.6版本
# add-apt-repository ppa:ondrej/php
# apt-get -y update
# apt install -y php5.6 php5.6-mysql

创建mysql认证文：
# cat  /var/lib/zabbix/percona/scripts/ss_get_mysql_stats.php.cnf 件
<?php
$mysql_user = 'root';
$mysql_pass = '';

测试脚本能否获取数据：
# /var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh gg
22
```

```
root@zabbix-mysql-master:~# ps -ef | grep zabbix_agent
root      47442      1  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
root      47443  47442  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd: collector [idle 1 sec]
root      47444  47442  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #1 [waiting for connection]
root      47445  47442  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #2 [waiting for connection]
root      47446  47442  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #3 [waiting for connection]
root      47447  47442  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #4 [waiting for connection]
root      47448  47442  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd: listener #5 [waiting for connection]
root      47449  47442  0 11:07 ?        00:00:00 /usr/sbin/zabbix_agentd: active checks #1 [idle 1 sec]
root      47564  16066  0 11:10 pts/0    00:00:00 grep --color=auto zabbix_agent
root@zabbix-mysql-master:~#
```

5.6.2.3：zabbix web导入Percona模板：



5.6.2.4：zabbbix web添加主机：

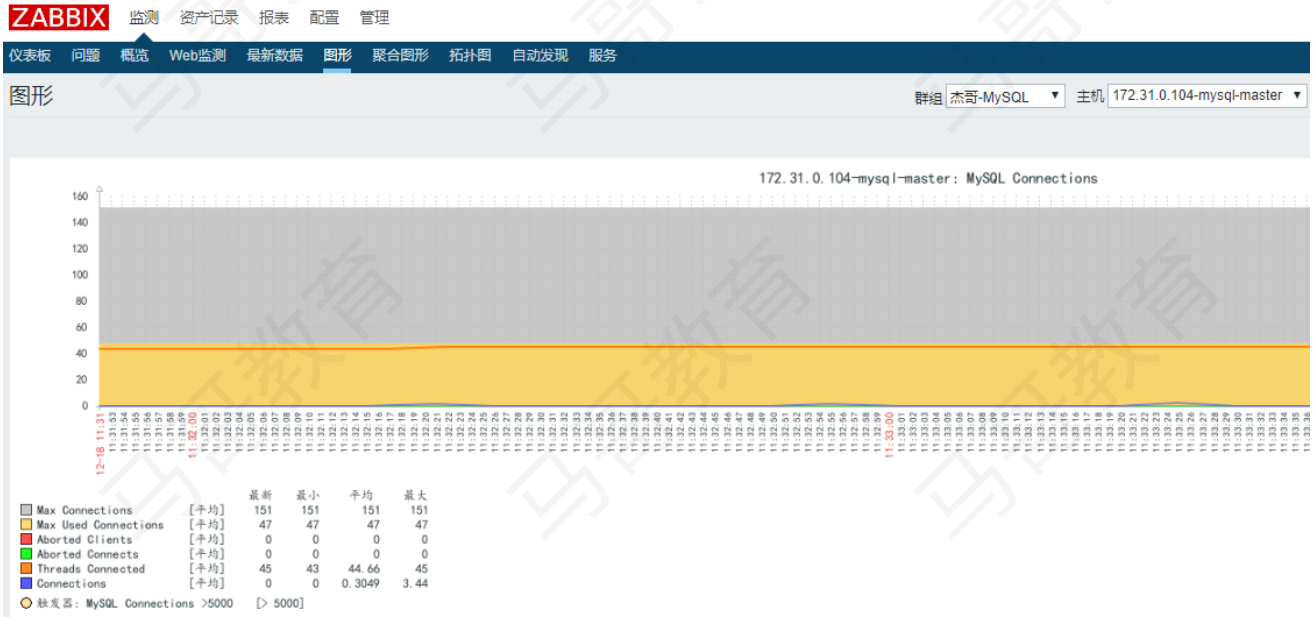


5.6.2.5：zabbix web对主机关联模板：

5.6.2.6：验证MySQL监控数据：

Percona模板中的监控项默认是五分钟收集一次监控项数据，会结合脚本检查agent上报错数据的文件的时间戳是否超过五分钟，脚本位置在/var/lib/zabbix/percona/scripts/get_mysql_stats_wrapper.sh。

```
root@zabbix-server:~# /apps/zabbix_server/bin/zabbix_get  -s 172.31.0.104 -p 10050 -k
"MySQL.Key-read-requests"
22
```



5.6.3：自定义脚本监控MySQL：

编写脚本监控脚本MySQL主从同步及延迟

5.6.3.1：MySQL slave安装zabbix agent：

```
# apt install zabbix-agent
```

5.6.3.1：脚本内容：

mysql_monitor.sh

```
# cat mysql_monitor.sh
#!/bin/bash
#Date:2016/11/11
#Author: Zhangshijie

Seconds_Behind_Master(){
    NUM=`mysql -uroot -e "show slave status\G;"  | grep "Seconds_Behind_Master:" | awk
-F: '{print $2}'`
    echo $NUM
}

master_slave_check(){
NUM1=`mysql -uroot  -e "show slave status\G;"  | grep "Slave_IO_Running" | awk -F:
'{print $2}' | sed 's/^[ \t]*//g'`
#echo $NUM1
NUM2=`mysql -uroot  -e "show slave status\G;"  | grep "Slave_SQL_Running:" | awk -F:
'{print $2}' | sed 's/^[ \t]*//g'`
```

```
#echo $NUM2
if test $NUM1 == "Yes" &&  test $NUM2 == "Yes";then
    echo 50
else
    echo 100
fi
}

main(){
    case $1 in
        Seconds_Behind_Master)
           Seconds_Behind_Master;
           ;;
    master_slave_check)
       master_slave_check
       ;;
    esac
}
main $1

# chmod  a+x mysql_monitor.sh
# bash mysql_monitor.sh  master_slave_check
50
```

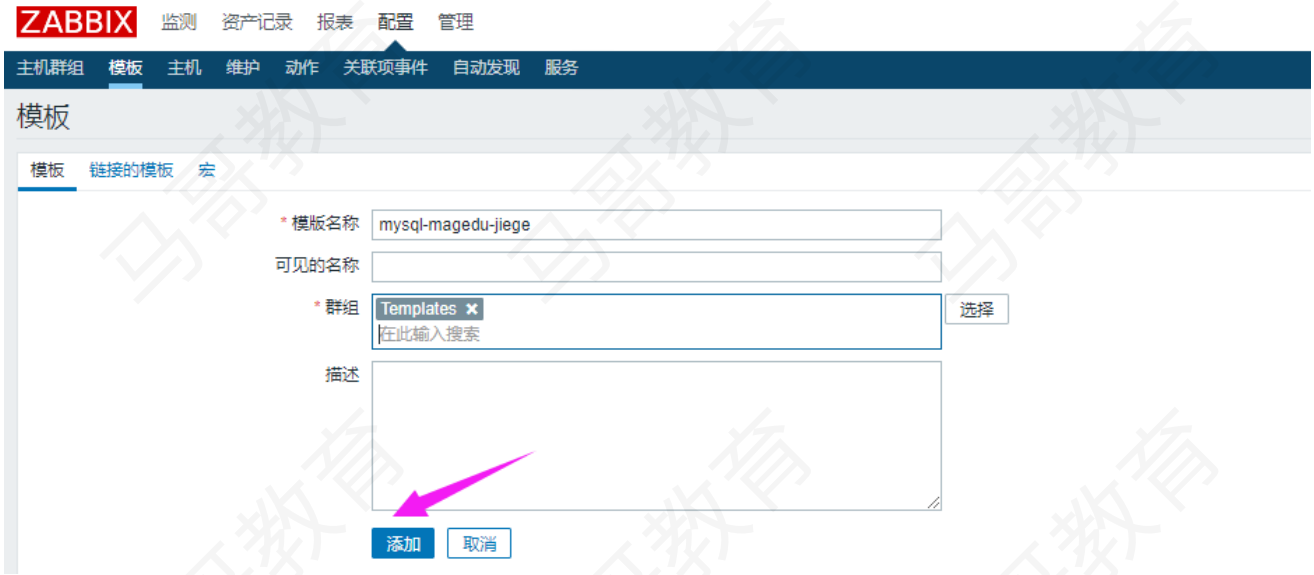
5.6.3.2：自定义监控项配置：

```
# pwd
/etc/zabbix/zabbix_agentd.conf.d
# cat all.conf
UserParameter=mysql_monitor[*],/etc/zabbix/zabbix_agentd.conf.d/mysql_monitor.sh "$1"

# systemctl  restart zabbix-agent

root@zabbix-server:~# /apps/zabbix_server/bin/zabbix_get  -s 172.31.0.105 -p 10050 -k
"mysql_monitor[master_slave_check]" #zabbix server测试获取数据
50
```

5.6.3.3：自定义模板：

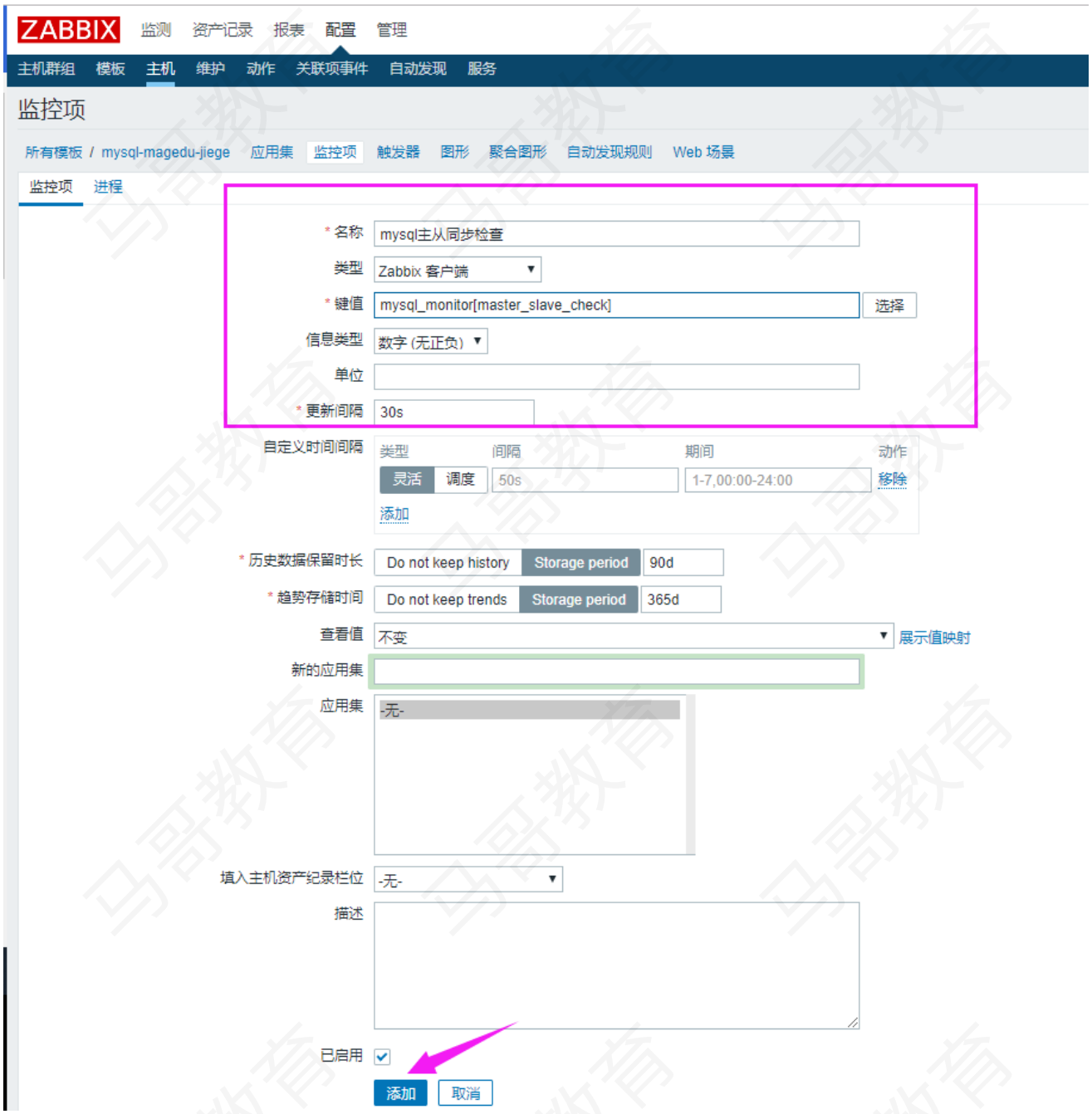
创建模板和MySQL主从同步的监控项、触发器和图形添加过程，其他监控项省略。

5.6.3.3.1：创建模板：



5.6.3.3.2：添加监控项：

添加监控项


ZABBIX
监测
资产记录
报表
配置
管理
主机群组
模板
主机
维护
动作
关联项事件
自动发现
服务
监控项
所有模板 / mysql-magedu-jiege
应用集
监控项
触发器
图形
聚合图形
自动发现规则
Web 场景
监控项
进程
* 名称
mysql主从同步检查
类型
Zabbix 客户端
* 键值
mysql_monitor[master_slave_check]
选择
信息类型
数字 (无正负)
单位
* 更新间隔
30s
自定义时间间隔
类型
间隔
期间
动作
灵活
调度
50s
1-7,00:00-24:00
移除
添加
* 历史数据保留时长
Do not keep history
Storage period
90d
* 趋势存储时间
Do not keep trends
Storage period
365d
查看值
不变
展示值映射
新的应用集
应用集
-无-
填入主机资产纪录栏位
-无-
描述
已启用
添加
取消

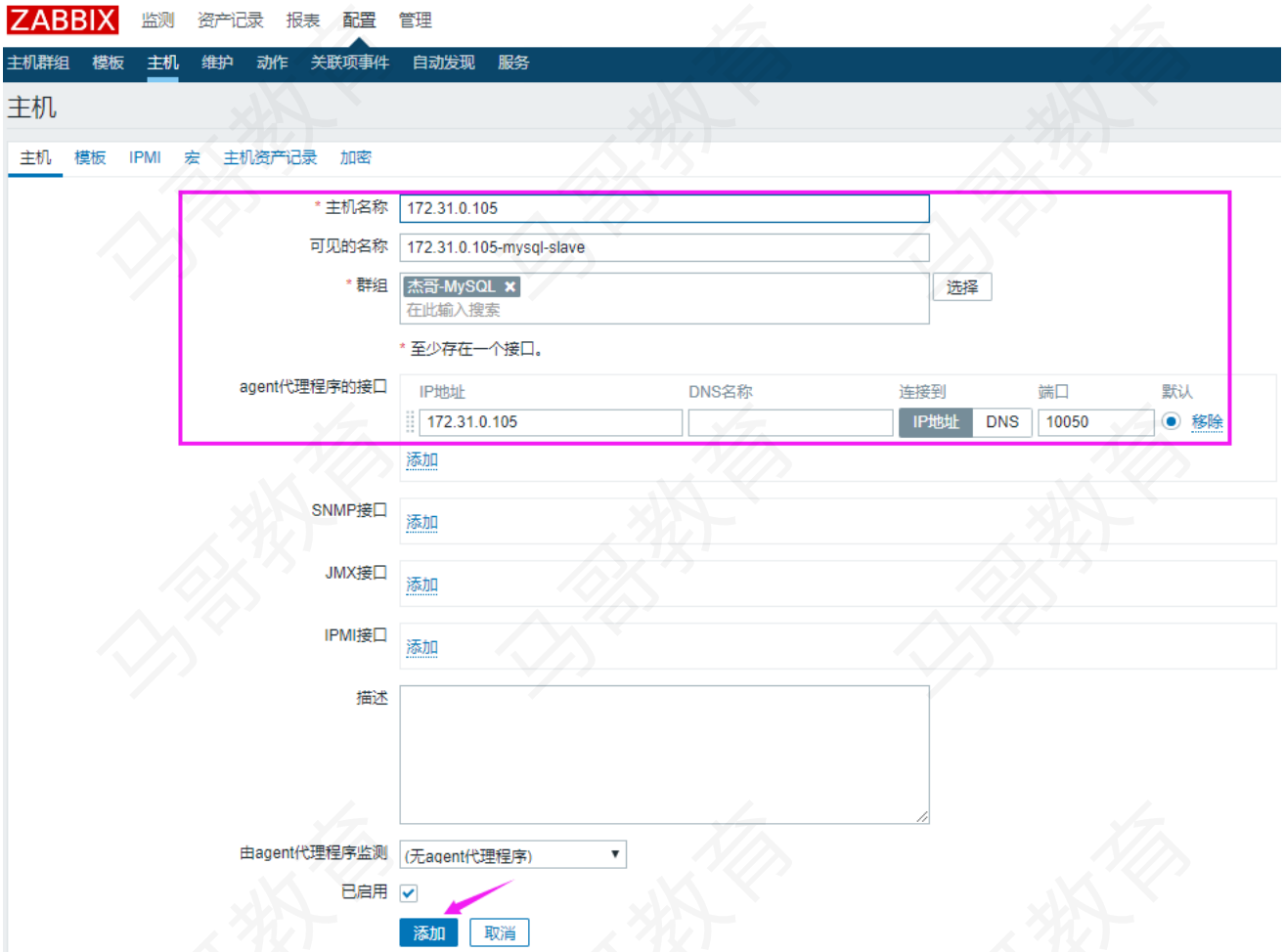
5.6.3.3.3：添加触发器：


ZABBIX
监测
资产记录
报表
配置
管理
主机群组
模板
主机
维护
动作
关联项事件
自动发现
服务
触发器
所有模板 / mysql-magedu-jiege
应用集
监控项 1
触发器
图形
聚合图形
自动发现规则
Web 场景
触发器
依赖关系
* 名称
mysql主从同步检查
严重性
未分类
信息
警告
一般严重
严重
灾难
* 表达式
{mysql-magedu-jiege:mysql_monitor[master_slave_check].last()}>55
添加
表达式构造器
事件成功迭代
表达式
恢复表达式
无
问题事件生成模式
单个
多重
事件成功关闭
所有问题
所有问题如果标签值匹配
标记
标记
值
移除
添加
允许手动关闭
URL
描述
已启用
添加
取消

5.6.3.3.4：添加图形：

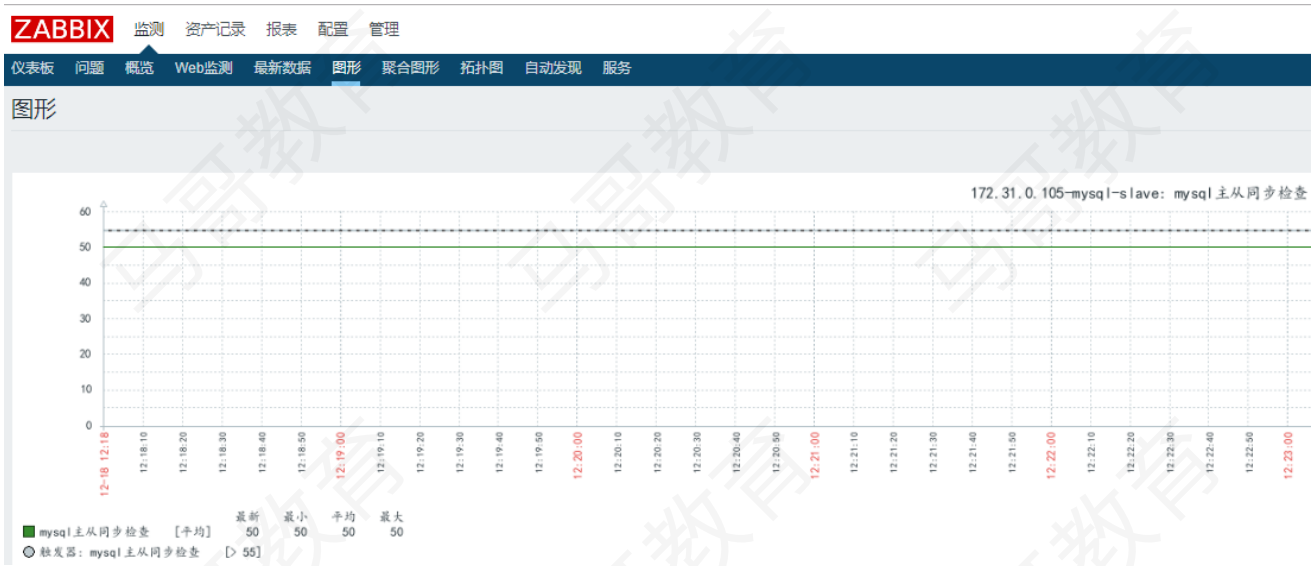
5.6.3.4：模板关联至主机：

5.6.3.4.1：添加主机：



5.6.3.4.2：关联模板：

5.6.3.5：验证监控数据：



5.7：自定义端口和进程监控：

略

5.8：故障自治愈功能：

当zabbix 监控到指定的监控项异常的时候，通过指定的操作使故障自动恢复，通常是重启服务等一些简单的操作，也可以调用脚本执行比较复杂的操作。

设置监控项和触发器，新建动作，在触发条件里面添加操作，在远程主机通过zabbix 客户端执行命令

```
1.开启zabbix sudu权限
2.配置允许允许特殊字符
3.配置远程命令
4.验证和测试
```

5.8.1：zabbix agent需要开启远程命令执行：

```
root@zabbix-node4:~# vim /etc/zabbix/zabbix_agentd.conf
73 EnableRemoteCommands=1 #开启远程执行命令
287 UnsafeUserParameters=1 #允许远程执行命令的时候使用不安全的参数(特殊字符串)
root@zabbix-node4:~# systemctl  restart zabbix-agent
```

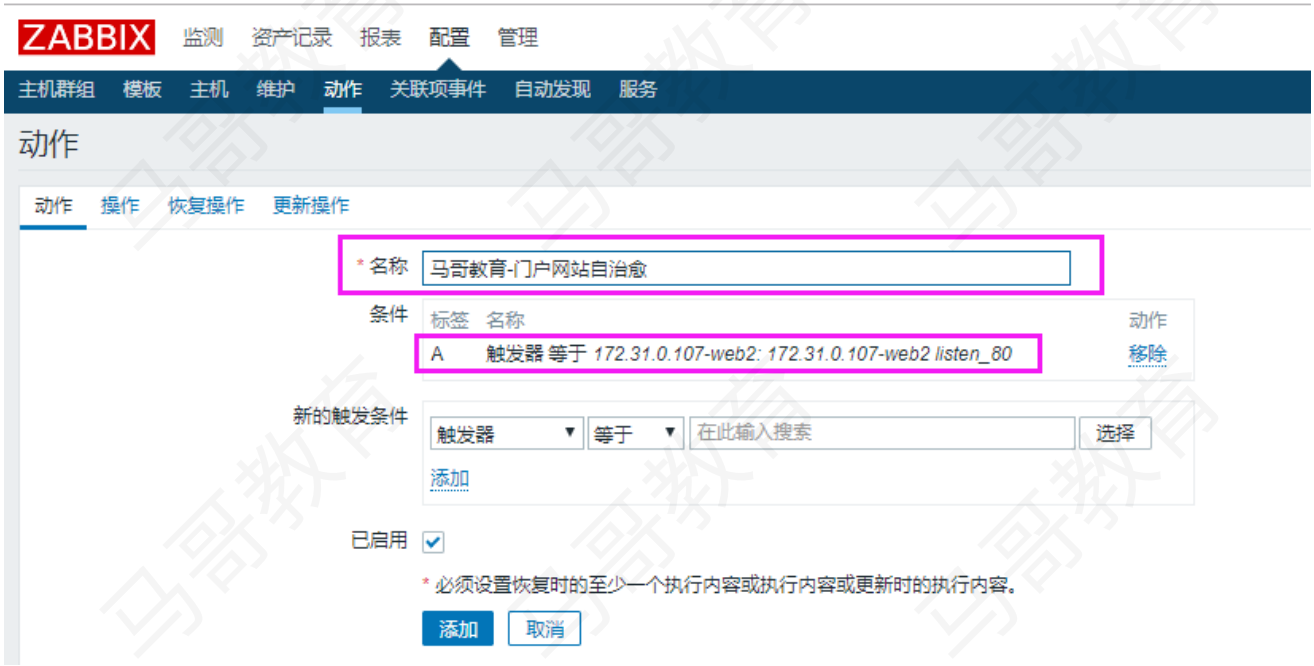
5.8.2：zabbix用户授权：

如果zabbix agent是使用zabbix用户启动的，那么要在zabbix 用户授权使用特权命令，负责有些命令zabbix没有权限执行，会导致定义好的自治愈策略因为权限拒绝为执行失败。

```
root@zabbix-node4:~# vim /etc/sudoers
55 # Defaults    !visiblepw #不强制使用tty
93 zabbix  ALL = NOPASSWD: ALL #授权指定用户执行特殊命令不再需要密码，比如sudo等
```

5.8.3：创建动作：

配置--动作--创建动作：



5.8.4：执行远程操作：

执行命令或者执行脚本

```
sudo  /path/to/command
sudo /path/to/script.sh
sudo  /apps/nginx/sbin/nginx
```

5.8.5：验证自治愈功能：

将被测试的服务手动停止运行，验证能否自动启动或重启，更多操作可以远程执行脚本。

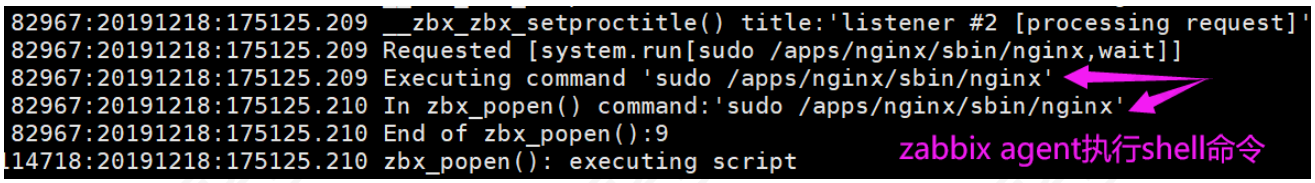
如下：

手动将Nginx、Tomcat等web服务停止后，验证zabbix agent能否自动启动或重启：

5.8.5.1：验证自治愈：

```
root@zabbix-node4:~# /apps/nginx/sbin/nginx
root@zabbix-node4:~#
root@zabbix-node4:~#
root@zabbix-node4:~#
root@zabbix-node4:~# ps -ef | grep nginx
root     110103      1  0 17:43 ?        00:00:00 nginx: master process /apps/nginx/sbin/nginx
nobody   110104 110103  0 17:43 ?        00:00:00 nginx: worker process
root     113743  78233  0 17:49 pts/0    00:00:00 grep --color=auto nginx
root@zabbix-node4:~#
```

5.8.5.2：zabbix 自治愈日志：



5.9：grafana图形展示：

5.9.1：安装grafana服务：

```
# dpkg -i grafana_6.4.2_amd64.deb
```

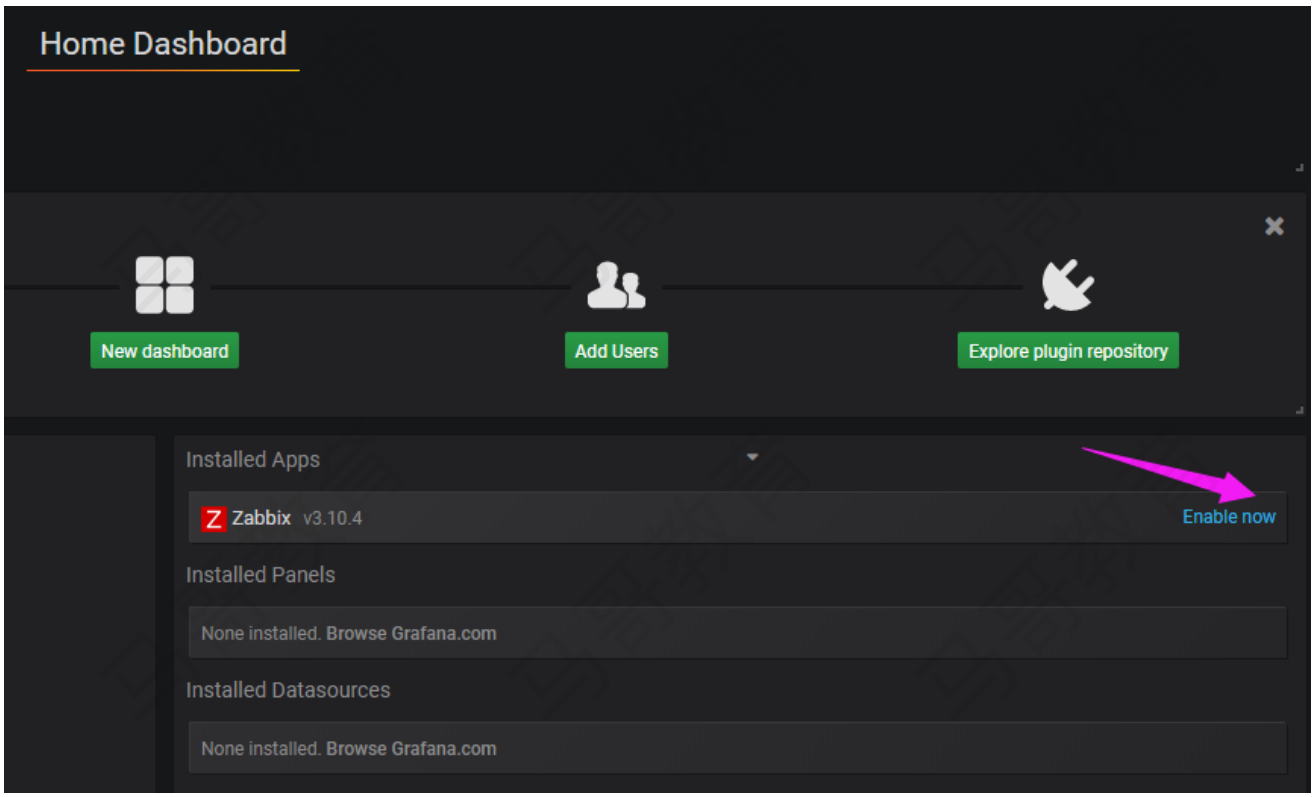
5.9.2：grafana安装并启用zabbix插件：

5.9.2.1：安装zabbix插件：

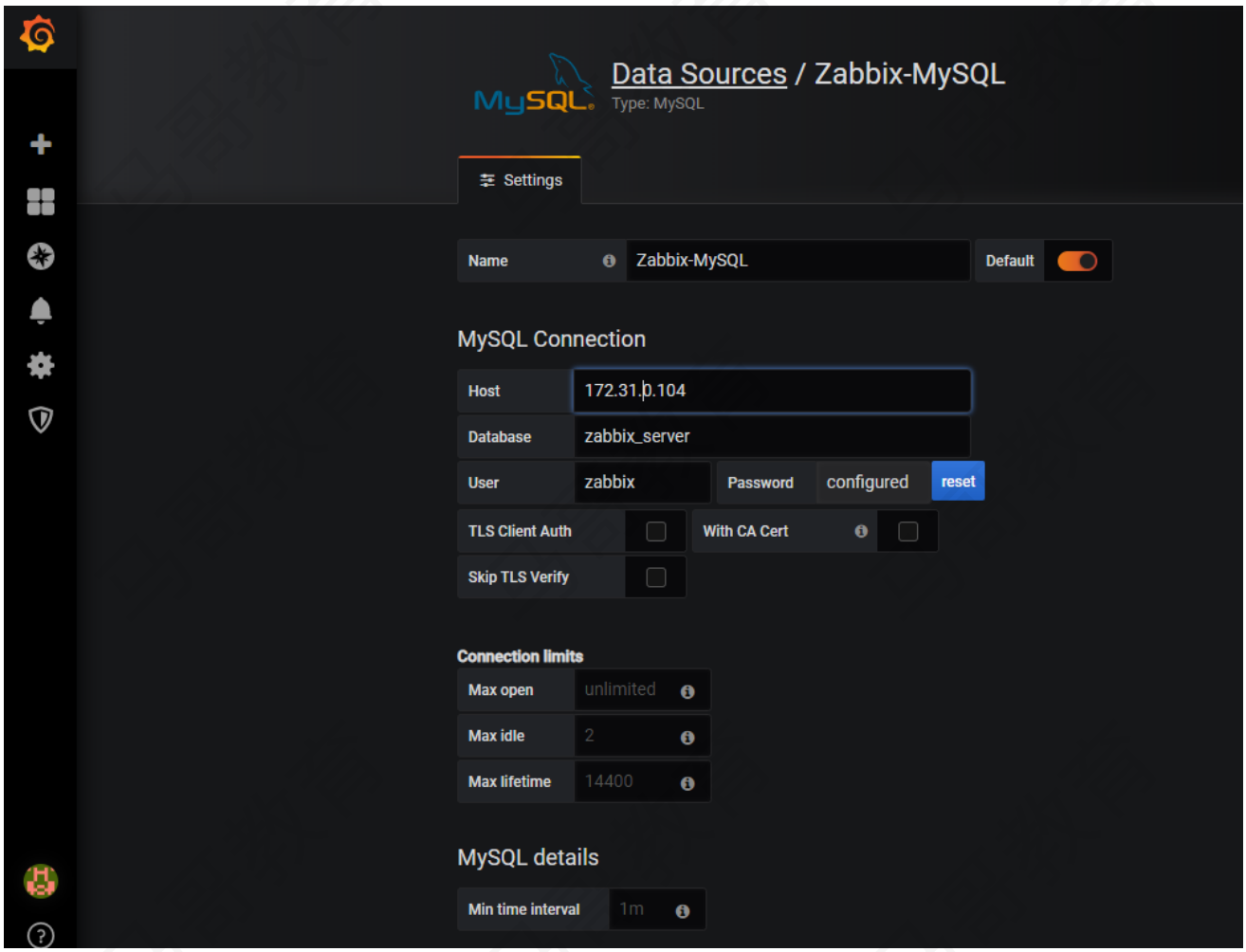
https://grafana.com/grafana/plugins/alexanderzobnin-zabbix-app

```
 # grafana-cli plugins install alexanderzobnin-zabbix-app
 installing alexanderzobnin-zabbix-app @ 3.10.4
from url: https://grafana.com/api/plugins/alexanderzobnin-zabbix-
app/versions/3.10.4/download
into: /var/lib/grafana/plugins

✓ Installed alexanderzobnin-zabbix-app successfully

Restart grafana after installing plugins . <service grafana-server restart>

# systemctl  restart grafana-server
```

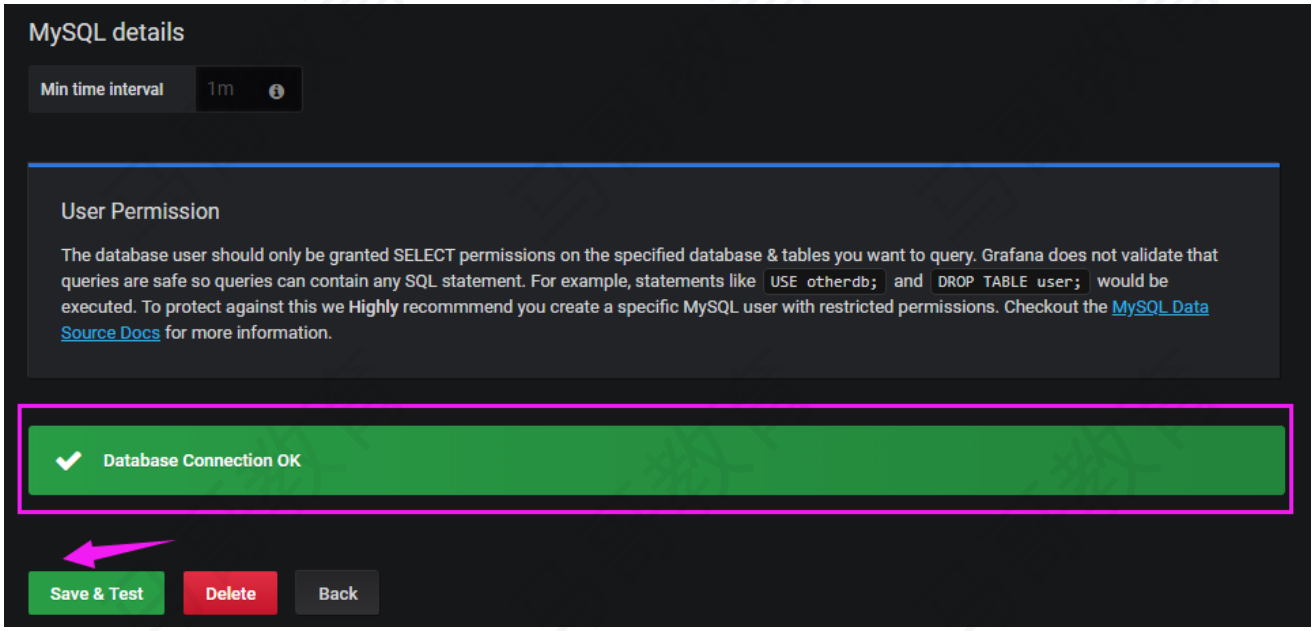
5.9.2.2：启动zabbix插件：



5.9.2.3：添加MySQL数据源：



5.9.2.4：测试MySQL数据源并保存：

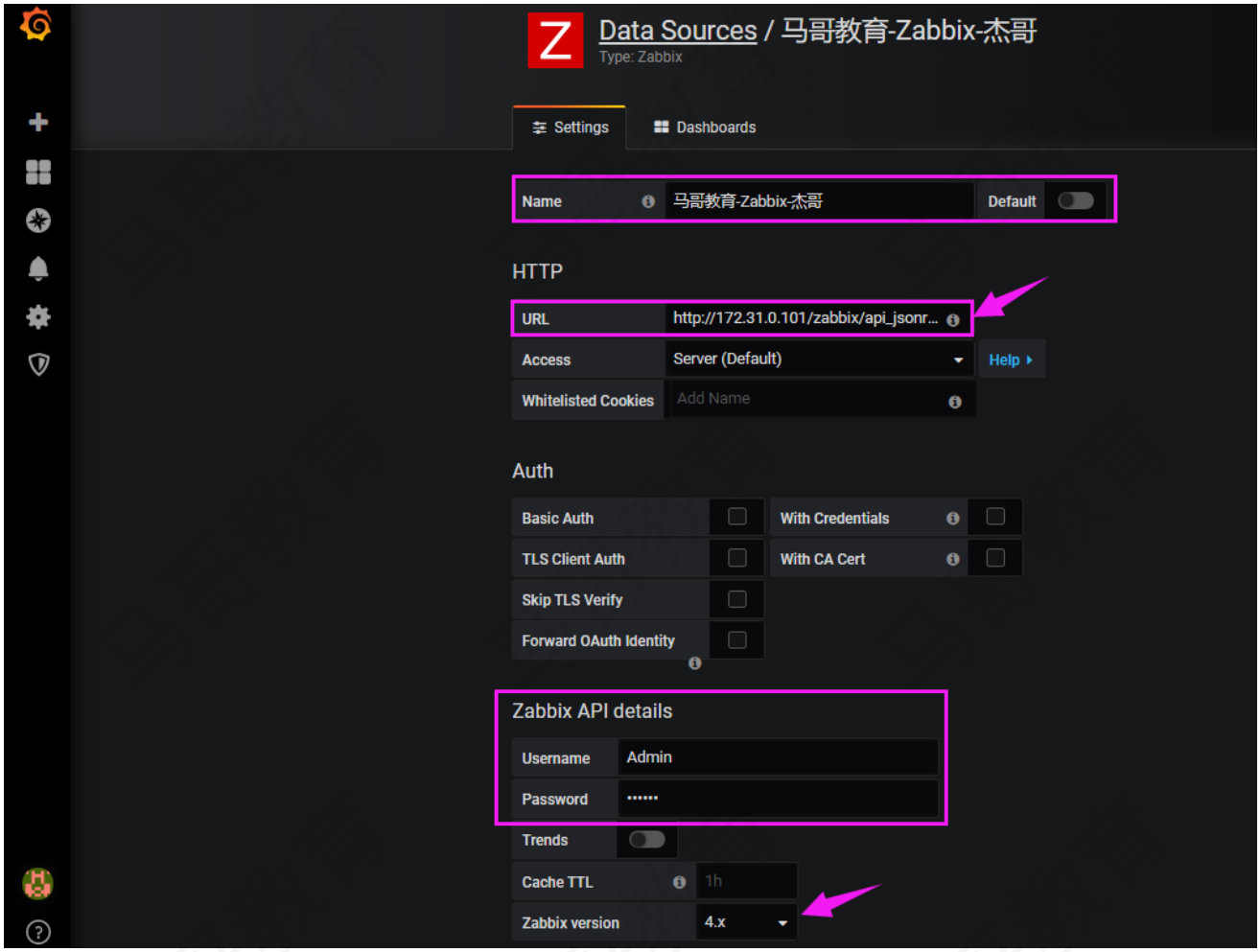


5.9.2.5：添加zabbix 数据源：

设置--data source--add data source：

在Others点击zabbix，即添加一个zabbix类型的数据源：

```
http://172.31.0.101/zabbix/api_jsonrpc.php
```



5.2.9.6：保存并添加数据源：

5.2.5.9.7：添加Dashboard：

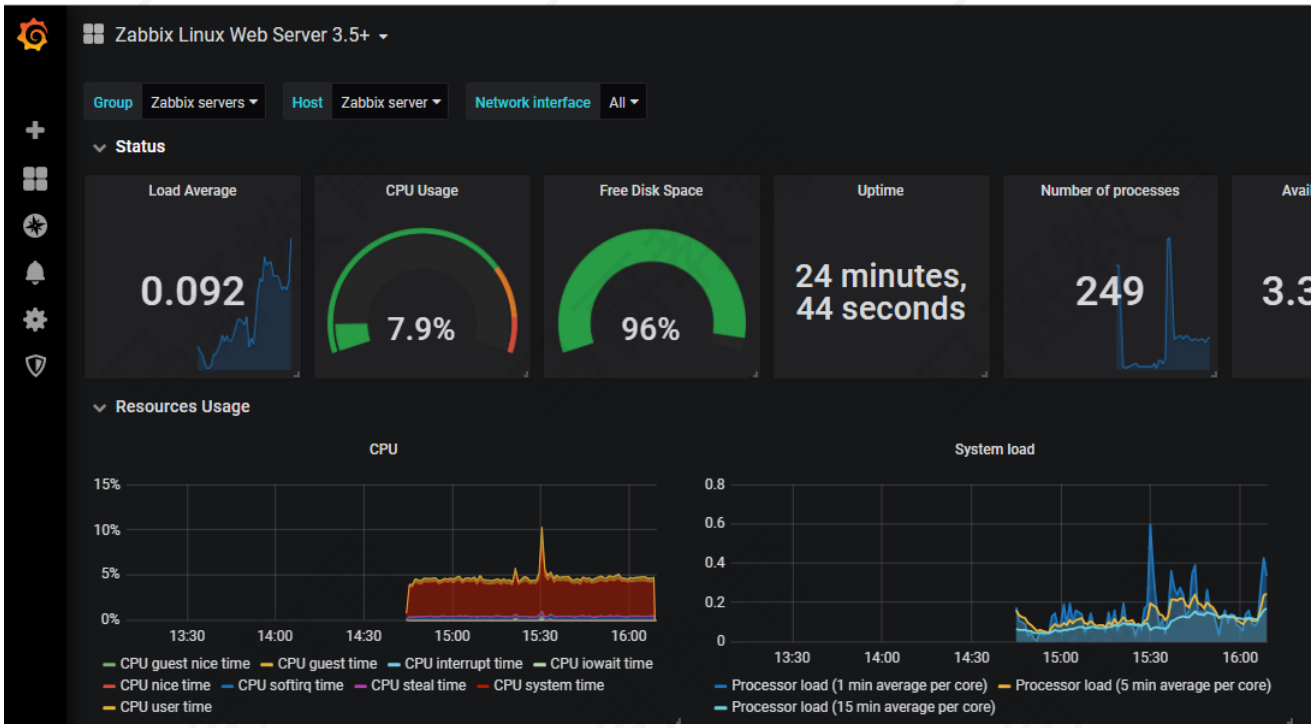
Home-New dashboard

5.2.5.9.7.1：单独添加图形：

略~

5.2.5.9.7.2：导入模板：

```
7877
```



5.10：自定义基础监控模板：

略

5.11：结合pyhton脚本监控案例：

5.11.1：kubernetes 集群状态监控脚本：

```
[root@kubernetes-master1 ~]# cat /etc/zabbix/zabbix_agentd.d/k8s_monitor.py
#!/usr/bin/env python
#coding:utf-8
#Author ZhangShiJie

import  subprocess
success_list = []
error_list= []
def get_status():
    obj = subprocess.Popen(("curl -sXGET
http://10.20.15.209:8080/api/v1/nodes"),shell=True, stdout=subprocess.PIPE)
    data =  obj.stdout.read()
    data1 = eval(data)
```

```
    data2 = data1.get('items')
    #print data2
    for i in data2:
        data3 = i.get('status')
        for i in data3.get('conditions'):
            if i.get('reason') == 'KubeletReady':
                if i.get('type') == "Ready":
                    if i.get('status') == 'True':
                        success_list.append(i.get('status'))
                    elif i.get('status') == 'False':
                        error_list.append(i.get('status'))
                    else:
                        break
                else:
                    error_list.append(i.get('status'))
                    #pass
            else:
                error_list.append(i.get('status'))
def count_status():
    if len(error_list) == 0:
        print 50
    else:
        print 100

def main():
    get_status()
    count_status()

if __name__ == "__main__":
    main()

# chmod a+x  /etc/zabbix/zabbix_agentd.d/k8s_monitor.py
```

5.11.2：监控MongodbDB复制集状态：

```
#cat /etc/zabbix/zabbix_agentd.d/mongodb_cluster_monitor.py
#!/bin/env  python
#coding:utf-8
#Author: ZhangShiJie

import  subprocess
success_list = []
error_list= []

def get_mongodb_status():
    obj = subprocess.Popen(("echo 'rs.status()' | /usr/local/mongodb/bin/mongo  -u user
-p wswd  --authenticationDatabase admin \
| grep health | awk -F':' '{print $2}' | awk -F',' '{print $1}'"),shell=True,
stdout=subprocess.PIPE)
    restful =  obj.stdout.read()
    data = restful.split()
    for i in data:
```

```
        if i == "1":
            success_list.append(i)
        else:
            error_list.append(i)

def count_status():
    if len(error_list) > 0:
        print 100
    else:
        print 50

def main():
    get_mongodb_status()
    count_status()

if __name__ == "__main__":
    main()

#chmod a+x /etc/zabbix/zabbix_agentd.d/redis_llen.py
```

5.11.3：监控Redis列表长度：

```
[root@redis ~]#cat  /etc/zabbix/zabbix_agentd.d/redis_llen.py
#!/usr/bin/env python
#coding:utf-8
#Author ZhangShijie
import redis
def redis_conn():
    pool=redis.ConnectionPool(host="10.20.0.252",port=6379,db=0)
    conn = redis.Redis(connection_pool=pool)
    data = conn.llen('api4-nginx-accesslog')
    print(data)
redis_conn()

[root@redis ~]# chmod a+x  /etc/zabbix/zabbix_agentd.d/redis_llen.py
```

5.11.4：监控ELK集群状态：

```
[root@elk-s1 ~]# vim /etc/zabbix/zabbix_agentd.d/els_status.py

#!/usr/bin/env python
#coding:utf-8
#Author ZhangShijie

import subprocess
false="false"
obj = subprocess.Popen(("curl -sXGET http://10.20.3.128:9200/_cluster/health?
pretty=true"),shell=True, stdout=subprocess.PIPE)
data =  obj.stdout.read()
data1 = eval(data)
status = data1.get("status")
```

```
if status == "green":
    print "100"
else:
    print "50"

[root@elk-s1 ~]# chmod  a+x /etc/zabbix/zabbix_agentd.d/els_status.py
```

5.11.5：监控RabbitMQ集群节点状态：

```
#cat /etc/zabbix/zabbix_agentd.d/rabbit_cluster_monitor.py
#!/bin/env  python
#coding:utf-8
#Author: ZhangShiJie

import  subprocess
running_list = []
error_list = []
false="false"
true="true"
def get_status():
    obj = subprocess.Popen(("curl  -sXGET  -u guest:guest
http://10.20.3.171:15671/api/nodes"),shell=True, stdout=subprocess.PIPE)
    data =  obj.stdout.read()
    data1 = eval(data)
    for i in data1:
        if i.get("running") == "true":
            running_list.append(i.get("name"))
        else:
            error_list.append(i.get("name"))
def count_server():
    if len(running_list) < 3: #可以判断错误列表大于0或者运行列表小于3，3未总计的节点数量
        print 100 #100就是集群内有节点运行不正常了
    else:
        print 50  #50为所有节点全部运行正常
def main():
    get_status()
    count_server()
if __name__ == "__main__":
    main()

#chmoa a+x /etc/zabbix/zabbix_agentd.d/rabbit_cluster_monitor.py
```

六：Zabbix 事件通知机制：

出现故障报警的时候，可以通过不同方式通知管理员进行故障处理，尽快恢复业务

```
[root@s1 ~]# vim /etc/zabbix/zabbix_server.conf
AlertScriptsPath=/usr/lib/zabbix/alertscripts #报警脚本路径
ExternalScripts=/usr/lib/zabbix/externalscripts #外部脚本路径
```

6.1：邮件通知：

通过企业邮箱、第三方服务商邮箱发送报警邮件通知运维工程师。

6.1.1：邮箱开启SMTP：

如果是QQ邮箱需要单独开启SMTP，其他邮箱具体联系服务商。

确认是已经打开状态，如果是未开启状态只要点击开启并根据提示进行相关验证即可。

6.1.2：生成授权码：

QQ邮箱在第三方平台发送邮件不能直接使用QQ邮箱的登录密码，需要使用单独提供的授权码才可以登录，具体生成方式如下：

6.1.2.1：生成授权码：

6.1.2.2：发送验证码：

安装提示使用绑定的手机发送**配置邮件客户端**到1069070069，然后点击**我已发送。**

6.1.2.3：保存授权码：

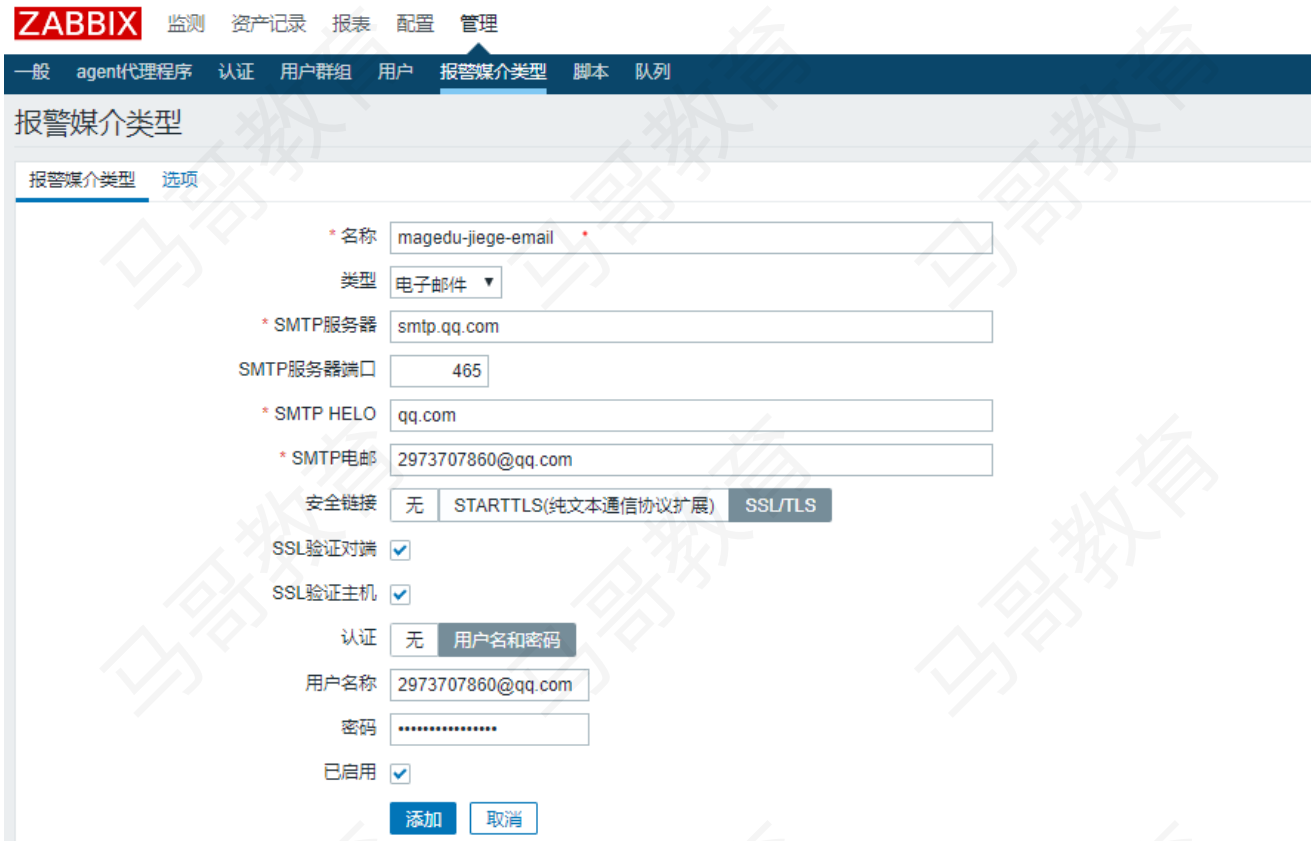
将页面生成后返回的授权码妥善保存好，后期会使用此授权码进行登录验证。

6.1.2.4：Zabbix Web创建报警媒介类型：

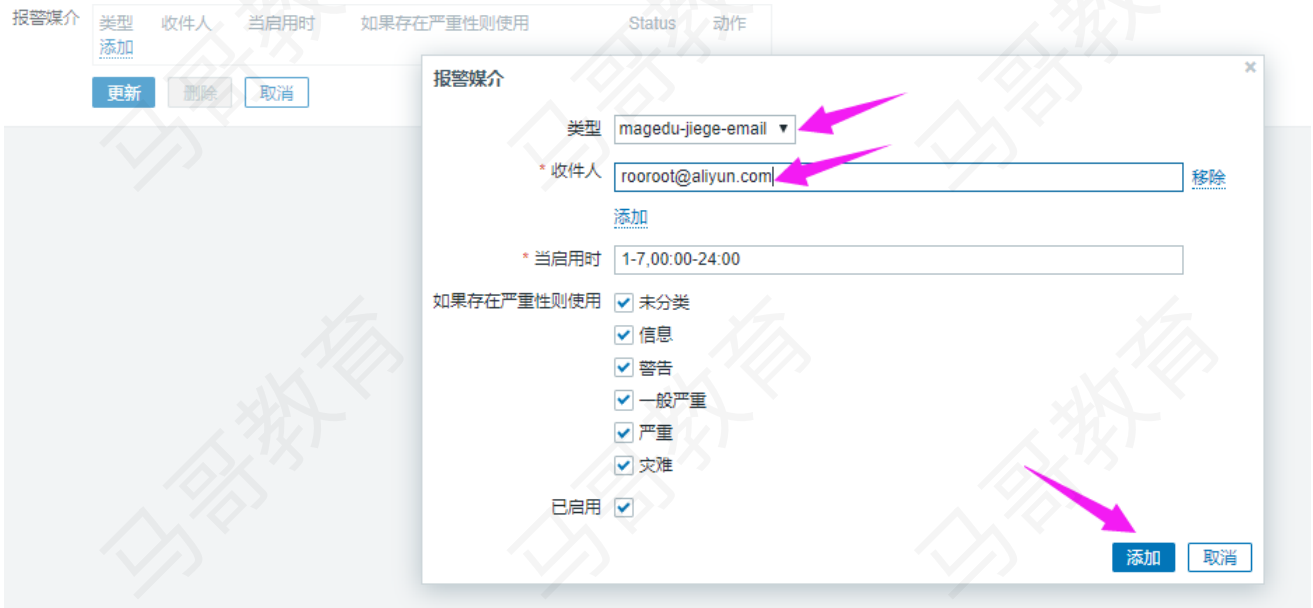
报警媒介类型是一种给运维工程师发送消息通知的渠道，即当zabbix 的触发器触发一个事件后，怎么才能把这个事件通过某些方式通知给运维工程师呢？那么媒介类型就起到这样的作用，媒介类型创建好之后，需要在每个账户里面添加相应的收件配置，比如邮件类型的媒介类型要给zabbix账户添加邮箱，如果是微信类型的媒介类型那么就要在zabbix账户设置微信号，同样的到来，短信类型的媒介类型那就得给zabbix账户设置手机号用于接收报警消息内容。

管理-->报警媒介类型-->创建报警媒介类型：

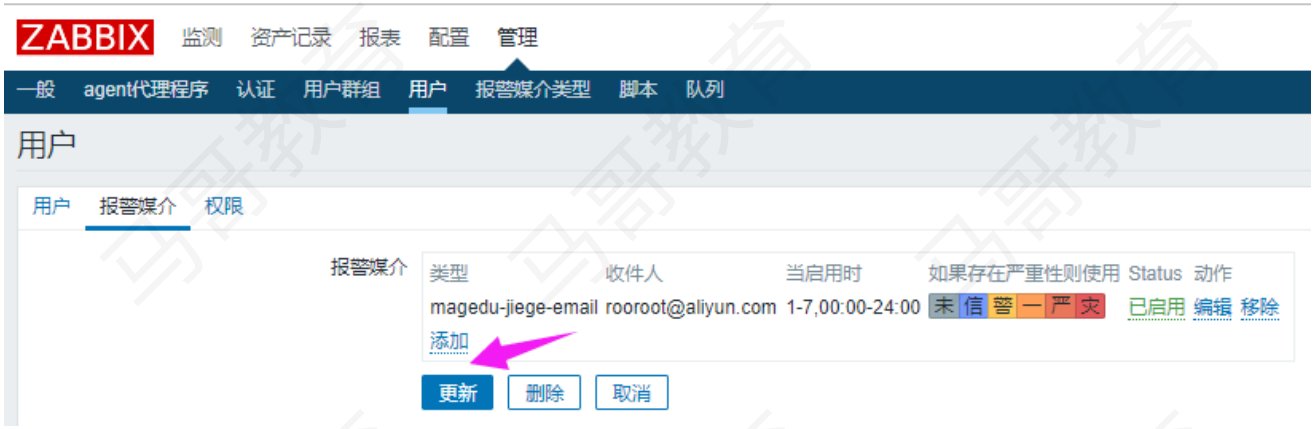
https://service.mail.qq.com/cgi-bin/help?subtype=1&&id=28&&no=371 #设置参考



6.1.2.5：给用户添加报警报警媒介：



6.1.2.6：更新报警媒介：



6.1.2.7：创建动作：

动作是对zabbix 触发器触发后生成的事件的具体处理操作，可以是远程执行命令，也可以是发送通知给指定的管理员进行故障处理，发送命令是调用的上一步骤创建好的报警媒介类型。

配置-->动作-->创建动作：

6.1.2.8：配置动作信息：

6.1.2.9：验证动作当前状态：

验证当前动作信息，并点击恢复操作，配置故障恢复的具体动作内容

6.1.2.10：配置故障恢复信息：

6.1.2.11：添加动作：

6.1.2.12：验证动作：

将某个被监控的服务手动停止，验证能否收到zabbix 发送的报警通知。

6.1.2.12.1：验证事件状态：

在zabbix web界面，验证当事件触发后邮件通知也没有 发送成功。

6.1.2.12.2：邮箱验证是否收到邮件：

到收件箱验证是否收到zabbix 发送的通知邮件

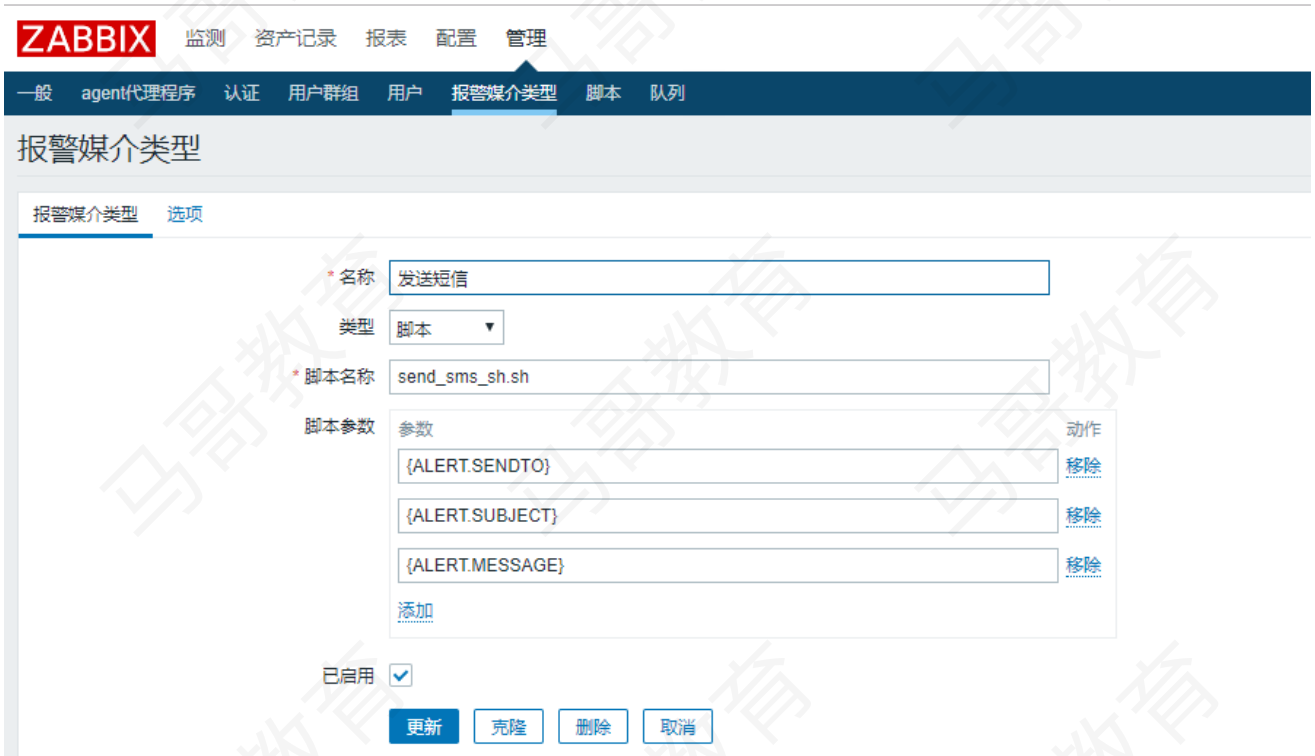
6.2：短信通知：

通知脚本见 api 及通知脚本/send_sms_sh.sh

服务端配置略，不同厂商配置不一样

6.2.1：添加短信报警媒介类型：

```
{ALERT.SENDTO}--收件人媒介
{ALERT.SUBJECT}--通知主题
{ALERT.MESSAGE}--通知内容

#zabbix 宏变量使用及场景
https://www.zabbix.com/documentation/4.0/zh/manual/appendix/macros/supported_by_location
```



6.2.2：添加联系人报警媒介：

设置收件人的指定收件类型内容

6.2.3：创建短信通知动作：

定义动作细节

6.2.4：配置短信发送具体内容：

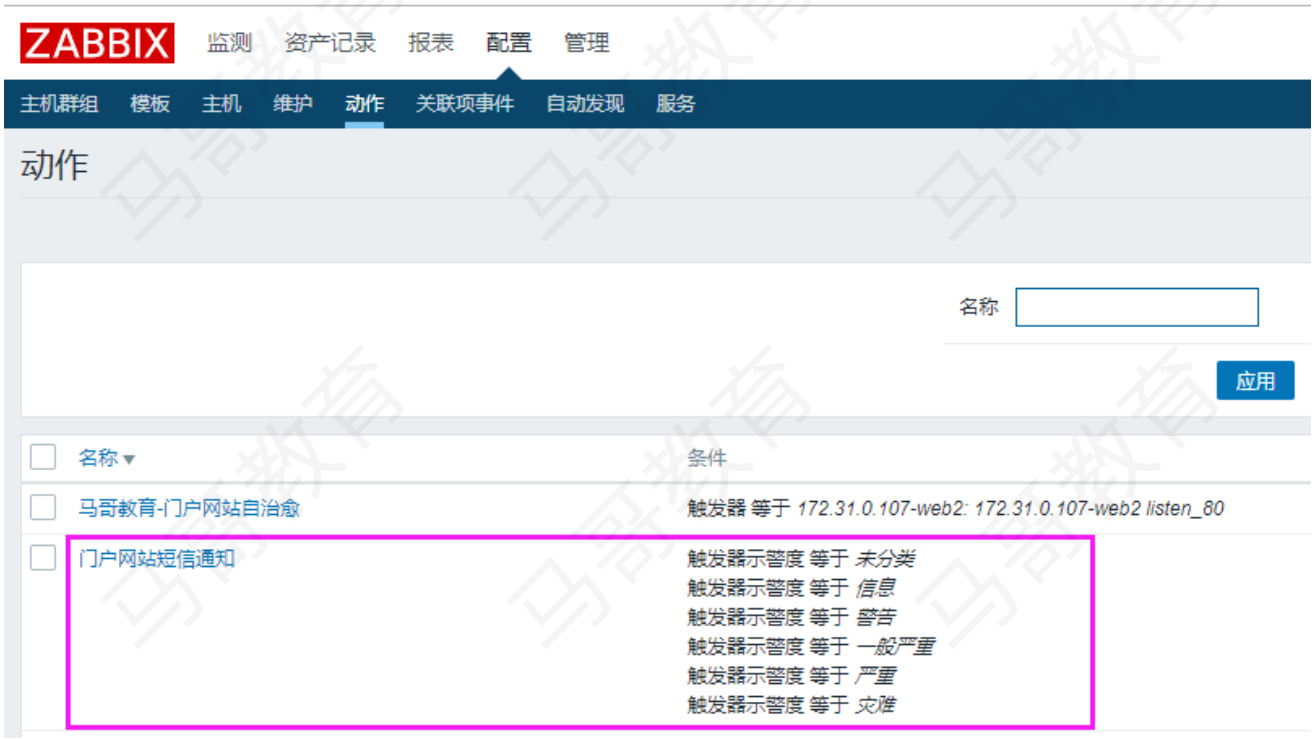
定义消息内容细节和收件人及发送方式

6.2.5：配置恢复操作：

配置恢复操作

6.2.6：验证动作状态：

验证内容



6.2.8：测试短信报警：

讲监控项手动停止，测试能否在出现故障后发送短信报警

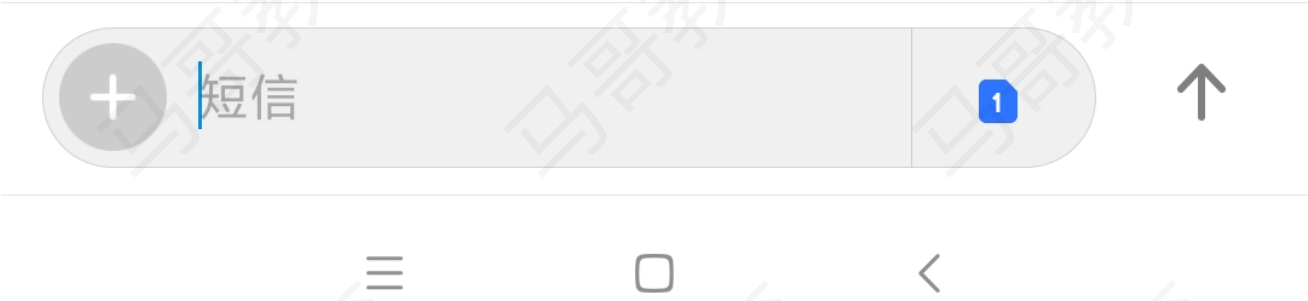
1 13:36

【马哥教育】北京业务故障服务器:172.18.0.101,IP:Nginx,详情:端口,down

1 14:42

【马哥教育】北京业务故障服务器:172.31.0.107-web2,IP:172.31.0.107,详情:nginx_listen_80,0

【马哥教育】北京业务恢复服务器:172.31.0.107-web2,IP:172.31.0.107,详情:nginx_listen_80,1



6.3：微信通知：

微信企业账号注册见 api **及通知脚本/weixin.py及文档微信报警**

6.3.1：企业微信注册及配置：

https://work.weixin.qq.com/

打开企业微信官网注册账号，使用自己的手机号进行注册。

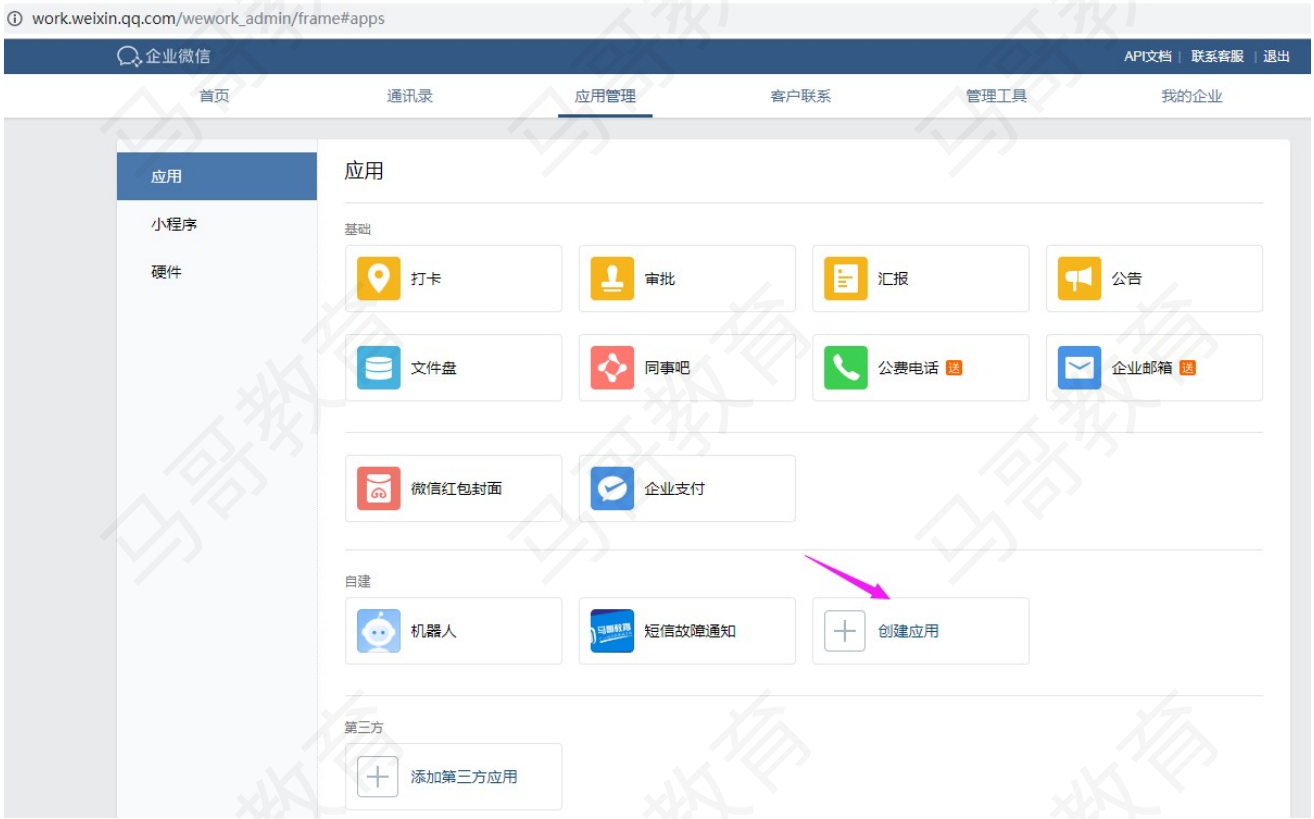
6.3.1.1：企业微信账号注册：

6.3.1.2：登录PC版：

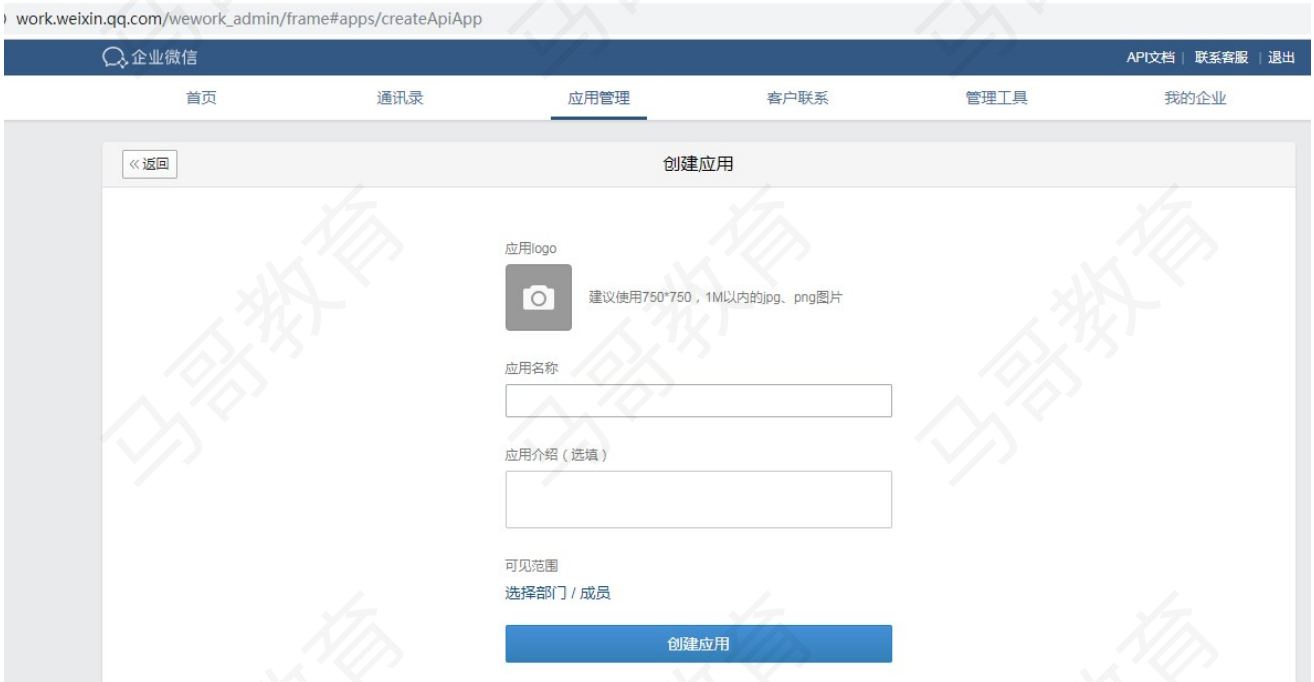
注册完成账号之后就可以扫码登录PC版web界面了，如下：

6.3.1.3：创建应用：

在web界面创建一个应用，用于微信报警通知。



6.3.1.4：填写应用信息：



6.3.1.5：注册完成：

AgentID和Secret会在发送微信报警信息的时候调用

6.3.1.6：创建微信账号：

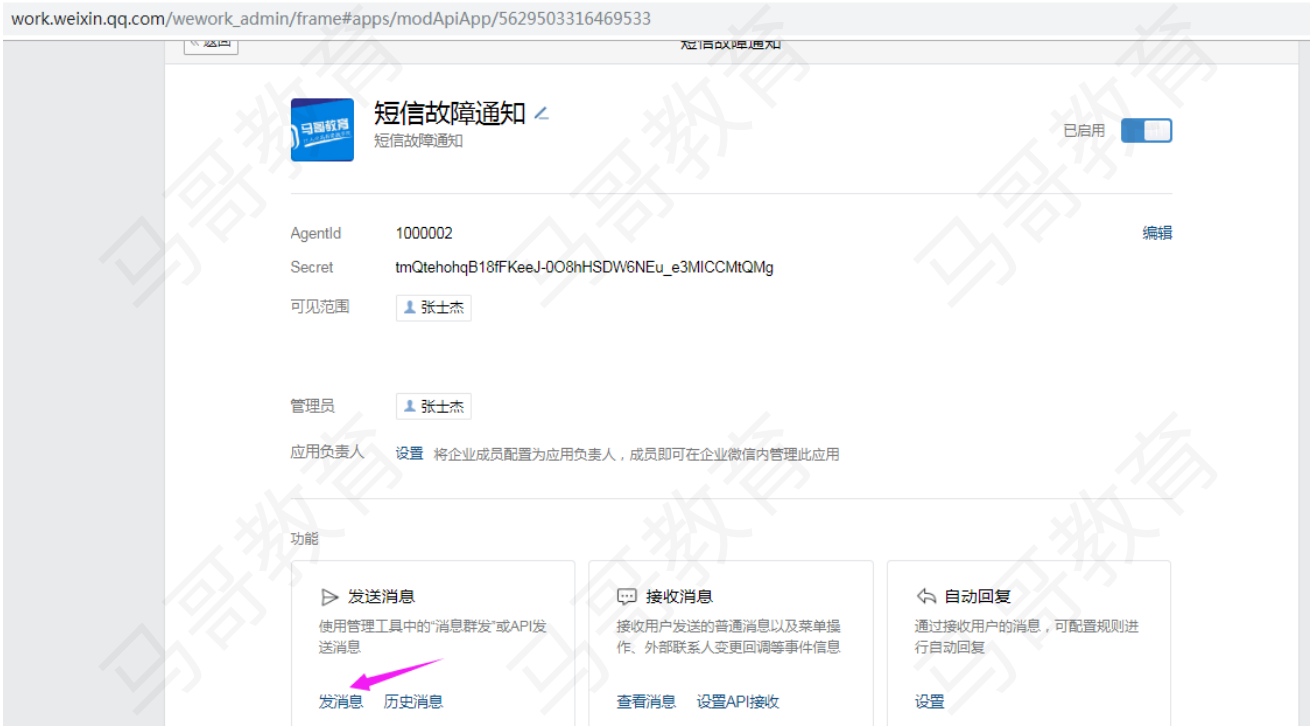
用户账户名称必须唯一，在发送微信报警信息的时候会调用

6.3.1.7：验证通讯录：

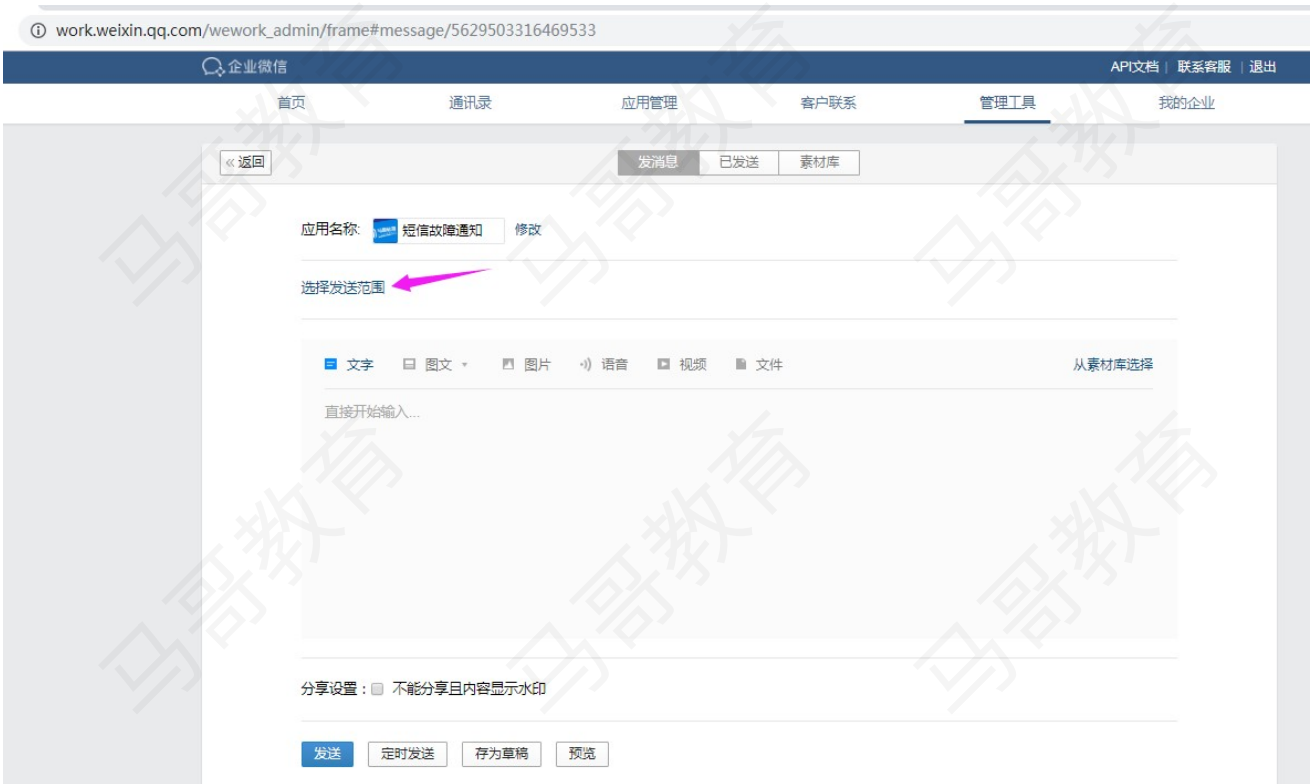
6.3.1.8：查看企业信息：

企业ID在发送微信报警信息的时候会调用

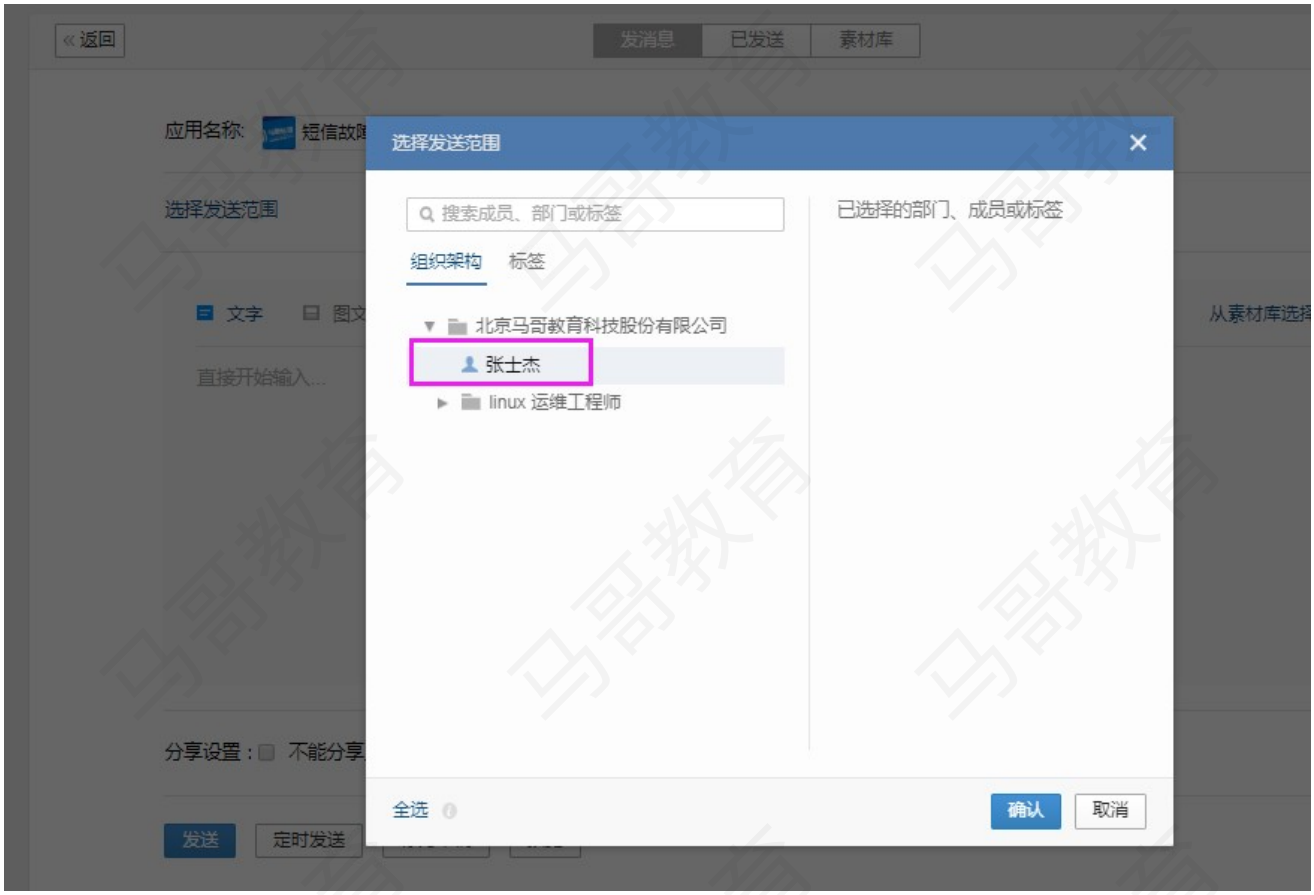
6.3.1.9：测试发送信息：



6.3.1.10：选择消息接收人：



6.3.1.11：确认消息接收人：



6.3.1.12：开始发送信息：

6.3.1.13：确认发送：

6.3.1.14：手机验证消息：

6.3.2：zabbix server配置：

zabbix server实现微信通知基于python调用脚本实现且需要安装requests模块：

https://work.weixin.qq.com/api/doc#90000/90003/90487 #简易教程

先获取token，token是通过corpid(企业ID)和corpsecret(应用 Secret)获取到的，然后通过API发送消息，根据官方文档可以看到，发送的消息其实就是一个post请求，请求方式如下：

```
请求方式：POST（HTTPS）
请求地址： https://qyapi.weixin.qq.com/cgi-bin/message/send?access_token=ACCESS_TOKEN

post 参数为 access_token 和 消息体。
```

服务端API调用：

https://work.weixin.qq.com/api/doc#90000/90135/90664

6.3.2.1：编写python脚本：

在zabbix server安装基础模块并编写pyhton脚本，python脚本通过调用企业微信的API实现自动发送通知消息，具体内容如下：

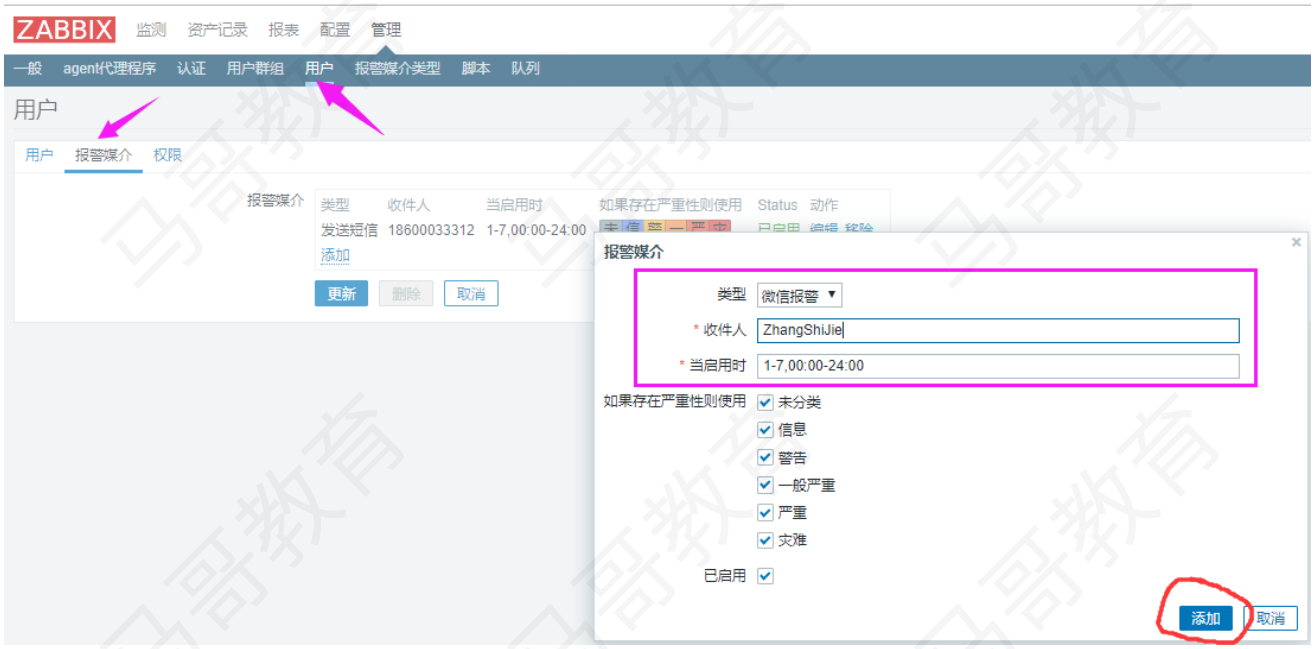
```
# apt install python-pip
# pip install requests

# pwd
/apps/zabbix_server/share/zabbix/alertscripts

# cat wx.py
#!/usr/bin/python3.6
#coding:utf-8
#Author:Zhang ShiJie
import requests
import sys
import os
```

```
import json
import logging


logging.basicConfig(level = logging.DEBUG, format = '%(asctime)s, %(filename)s, %
(levelname)s, %(message)s',
       datefmt = '%a, %d %b %Y %H:%M:%S',
       filename = os.path.join('/tmp','weixin.log'),
       filemode = 'a')

corpid='企业ID'
appsecret="秘钥"
agentid="AgentID"
token_url='https://qyapi.weixin.qq.com/cgi-bin/gettoken?corpid=' + corpid +
'&corpsecret=' + appsecret
req=requests.get(token_url)
accesstoken=req.json()['access_token']


msgsend_url='https://qyapi.weixin.qq.com/cgi-bin/message/send?access_token=' +
accesstoken
touser=sys.argv[1]
subject=sys.argv[2]
message=sys.argv[2] + "\n\n" +sys.argv[3]


params={
       "touser": touser,
       "msgtype": "text",
       "agentid": agentid,
       "text": {
           "content": message
           },
       "safe":0
       }


req=requests.post(msgsend_url, data=json.dumps(params))
logging.info('sendto:' + touser + ';;subject:' + subject + ';;message:' + message)

# python wx.py  ZhangShiJie "这是主题" "这是内容"
```

```
# python3.6 weixin.py  ZhangShiJie "这是主题" "这是内容"
#
```

6.3.2.2：添加微信报警媒介类型：

6.3.2.3：添加联系人报警媒介：



6.3.2.4：更新报警媒介：

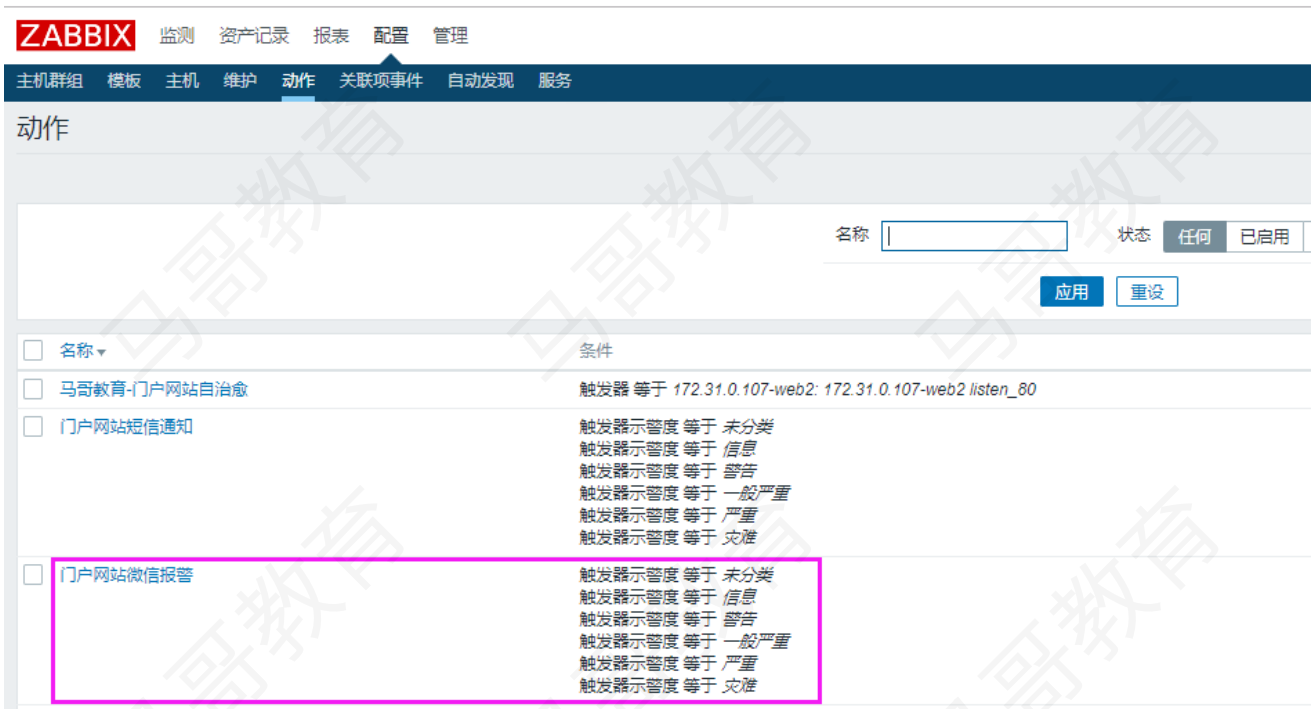
6.3.2.5：添加微信报警动作：

6.3.2.6：配置故障操作：


ZABBIX
监测
资产记录
报表
配置
管理
主机群组
模板
主机
维护
动作
关联项事件
自动发现
服务
动作
动作
操作
恢复操作
更新操作
* 默认操作步骤持续时间
60s
默认标题
门户业务报警
消息内容
【马哥教育】北京业务故障服务器:{HOST.NAME},IP:{HOSTNAME1},详情:{ITEM.NAME},{ITEM.VALUE}
暂停操作以制止问题
操作
步骤 细节
开始于 持续时间 动作
1 - 2 发送消息给用户: Admin (Zabbix Administrator) 通过 发送短信 立即地 60s 编辑 移除
操作细节
步骤
1 - 2 (0 - 无穷大)
步骤持续时间
60s (0 - 使用默认)
操作类型
发送消息
* 您必须至少选择一个用户或用户组。
发送到用户群组
用户群组
动作
添加
发送到用户
用户
动作
Admin (Zabbix Administrator)
移除
添加
仅送到
微信报警

6.3.2.7：配置恢复操作：

6.3.2.8：验证动作状态：



6.3.2.9：测试微信报警：


15:58
短信故障通知
【马哥教育】北京业务故障服务器:172.31.0.107-web2,IP:172.31.0.107,详情:nginx_listen_80,0
门户业务报警
【马哥教育】北京业务故障服务器:172.31.0.107-web2,IP:172.31.0.107,详情:nginx_listen_80,0
业务恢复
【马哥教育】北京业务恢复服务器:172.31.0.107-web2,IP:172.31.0.107,详情:nginx_listen_80,1

七：Zabbix 自动化运维：

通过脚本批量部署zabbix agent、通过API批量自动添加主机或者自动发现监控主机

7.1：Zabbix Agent批量部署：

通过脚本，实现数十上百台agent的自动化批量安装，可以通过源码或apt等方式安装。

7.1.1：源码编译安装：

官网下载源码包，通过自定义编译参数、自定义配置文件和监控脚本实现安装

见 **zabbix-agent-onekey-install-4.0.X**

```
#!/bin/bash
#Date:2016/08/20
#Author: Zhangshijie

DIR=`pwd`
grep "Ubuntu" /etc/issue &> /dev/null
if [ $? -eq 0 ];then
  apt update
```

```
  apt-get  -y install iproute2  ntpdate  tcpdump telnet traceroute nfs-kernel-server
nfs-common  lrzsz tree  openssl libssl-dev libpcre3 libpcre3-dev zlib1g-dev ntpdate
tcpdump telnet traceroute  gcc openssh-server lrzsz tree  openssl libssl-dev libpcre3
libpcre3-dev zlib1g-dev ntpdate tcpdump telnet traceroute iotop unzip zip make
fi

grep "Kernel" /etc/issue &> /dev/null
if [ $? -eq 0 ];then
  yum install  vim iotop bc gcc gcc-c++ glibc glibc-devel pcre  pcre-devel openssl
openssl-devel zip unzip zlib-devel  net-tools lrzsz tree ntpdate telnet lsof tcpdump
wget libevent libevent-devel bc  systemd-devel bash-completion traceroute -y
fi

tar xvf zabbix-4.0.X.tar.gz && cd zabbix-4.0.X && ./configure --
prefix=/apps/zabbix_agent --enable-agent && make && make install
useradd zabbix
mkdir  /apps/zabbix_agent/pid
mkdir  /apps/zabbix_agent/logs

\cp ${DIR}/zabbix-agent.service /lib/systemd/system/zabbix-agent.service
\cp ${DIR}/zabbix_agentd.conf /apps/zabbix_agent/etc/zabbix_agentd.conf
\cp ${DIR}/zabbix_agentd.conf.d/* /apps/zabbix_agent/etc/zabbix_agentd.conf.d/

HOST_IP=`ifconfig  eth0 | grep -w inet  | awk '{print $2}'`
sed -i "s/Hostname=/Hostname=${HOST_IP}/g" /apps/zabbix_agent/etc/zabbix_agentd.conf
chown zabbix.zabbix -R /apps/zabbix_agent/

systemctl daemon-reload && systemctl enable zabbix-agent && systemctl restart zabbix-
agent
```

7.1.2：apt/yum在线安装：

略

7.2：Zabbix API添加主机

https://www.zabbix.com/documentation/4.0/zh/manual/api

通过API，实现完全自动化添加删除agent、关联模板等操作

7.2.1：API使用基础：

```
见文本文档 Zabbix-API使用
1.获取token:
需要指定用的账户名 密码和id
# curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "USER",
        "password": "PASSWD"
```

```
    },
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool

#返回的数据
{
    "jsonrpc": "2.0",
    "result": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}


2.查看指定主机信息:
# curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "filter": {
            "host": [
                "172.31.0.107"
            ]
        }
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool


3.获取所有主机列表:
# curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["host"]
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool


4.获取所有用户:
# curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend"
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool
```

5.获取单个模板信息:

```
# curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "NGINX_Check_Statuc"
            ]
        }
    },
    "auth": "178bb976840186b6412461ca30d249d5",
    "id": 1
}'   http://192.168.7.101/zabbix/api_jsonrpc.php | python -m json.tool
```

6.获取多个模板信息:

```
# curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "tomcat-template-magedu-jiege",
                "mysql-magedu-jiege"
            ]
        }
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool
```

7.基于脚本获取token

```
# cat token.py
#!/usr/bin/env python
#Author: Zhangshijie
# -*- coding:utf-8 -*-

import requests
import json

url = 'http://172.31.0.101/zabbix/api_jsonrpc.php'
post_data = {
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
```

```
        "password": "zabbix"
    },
    "id": 1
}
post_header = {'Content-Type': 'application/json'}

ret = requests.post(url, data=json.dumps(post_data), headers=post_header)

zabbix_ret = json.loads(ret.text)
if not zabbix_ret.has_key('result'):
    print 'login error'
else:
    print zabbix_ret.get('result')
```

8.通过API添加主机命令格式:
API添加主机为预先知道要添加的主机IP、预先安装并配置好zabbix agent、预先知道要关联的模板ID/组ID等信息，然后同API提交请求添加

```
# curl -s -X POST -H 'Content-Type:application/json' -d '
 {
     "jsonrpc": "2.0",
     "method": "host.create", #定义方法，N多钟
     "params": {
         "host": "API Add Host Test", #自定义添加后的agent的名称
         "interfaces": [
             {
                 "type": 1, #类型为1表示agent，2是SNMP，3是IMPI,4是JMX
                 "main": 1, #主要接口
                 "useip": 1, #0是使用DNS，1是使用IP地址
                 "ip": "172.31.0.24", #添加的zabbix agent的IP地址
                 "dns": "",
                 "port": "10050" #agent端口
             }
         ],
         "groups": [
             {
                 "groupid": "2" #添加到的组的ID
             }
         ],
         "templates": [
             {
                 "templateid": "10001" #关联的模板的ID
             }
         ]
     },
     "auth": "977781251d1222ebead6f05da1a9ec4d",
     "id": 1
 }' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool
```

8.1: 通过API添加单台不带proxy的agent示例:

```
# curl -s -X POST -H 'Content-Type:application/json' -d '
 {
```

```
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "API Add Host Test",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "172.31.0.24",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "15"
            }
        ],
        "templates": [
            {
                "templateid": "10001"
            }
        ]
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool
```

8.2 通过API添加主机-带proxy模式：

```
# curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "172.18.0.109",
        "proxy_hostid": "10274",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "172.31.0.109",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "15"
            }
        ],
```

```
        "templates": [
            {
                "templateid": "10270"
            }
        ]
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool



9.验证主机信息:
#  curl -s -X POST -H 'Content-Type:application/json' -d '
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "filter": {
            "host": [
                "172.18.0.109"
            ]
        }
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool
```

7.2.2：API批量添加主机：

```
10.通过脚本调用API快速添加:
root@zabbix-server:~# cat zabbix-add-host.sh
#!/bin/bash

IP="
172.31.100.105
172.31.100.106
172.31.100.107
172.31.100.108
"

for node_ip in ${IP};do
 curl -s -X POST -H 'Content-Type:application/json' -d '
 {
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "'${node_ip}'",
        "name": "magedu-jiege-tomcat_'${node_ip}'",
        "proxy_hostid": "10274",
        "interfaces": [
            {
```

```
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "'${node_ip}'",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "15"
            }
        ],
        "templates": [
            {
                "templateid": "10270"
            }
        ]
    },
    "auth": "8b20c44282bc5ba5d70d24ffe2c0e467",
    "id": 1
}' http://172.31.0.101/zabbix/api_jsonrpc.php | python3 -m json.tool
done
```

7.3：Zabbix 动态发现主机

http://blogs.studylinux.net/?p=705

##