

## 七载耕耘，全面盘点：Zabbix 实战文章精华大全分享

——整理 by 乐维社区

# 基础知识

## 一、Zabbix\_sender 介绍及简单使用

### 1. zabbix\_sender 是什么?有什么作用?

zabbix 获取 key 值有超时时间，如果自定义的 key 脚本一般需要执行很长时间，这根本没法去做监控，那怎么办呢？这时候就需要使用 zabbix 监控类型 zabbix trapper，配合 zabbix\_sender 给它传递数据。所以说 zabbix\_sender 是更新 items 值最快的方式

### 2. zabbix\_sender 命令解析

方法一:

```
zabbix_sender -z server -s host -k key -o value
```

方法二:

```
zabbix_sender -c config-file -k key -o value
```

方法三:

```
zabbix_sender -z server -i file
```

-c --config                   zabbix\_agent 配置文件绝对路径

-z --zabbix-server           zabbix server 的 IP 地址

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

- p --port           zabbix server 端口.默认 10051
- s --host           主机名, 与 zabbix\_server web 上主机的 hostname
- l --source-address    源 IP
- k --key             监控项的 key
- o --value            key 值
- i --input-file        从文件里面读取 hostname、key、value 一行为一条数据, 使用空格作为分隔符, 如果主机名带空格, 那么请使用双引号包起来
- r --real-time         将数据实时提交给服务器
- v --verbose          详细模式, -vv 更详细

### 3. 案例

创建监控项

所有主机 / test 已启用 ZBX SNMP JMX IPMI 应用集 监控项 触发器 图形 自动发现规则 Web 场景

监控项 进程

\* 名称 test **名称随便写**

类型 Zabbix采集器 **类型需要选采集器**

\* 键值 key.key **自定义的key**

信息类型 数字 (无正负)

单位

\* 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

允许的主机

新的应用集 **新的应用集**

应用集 -无-

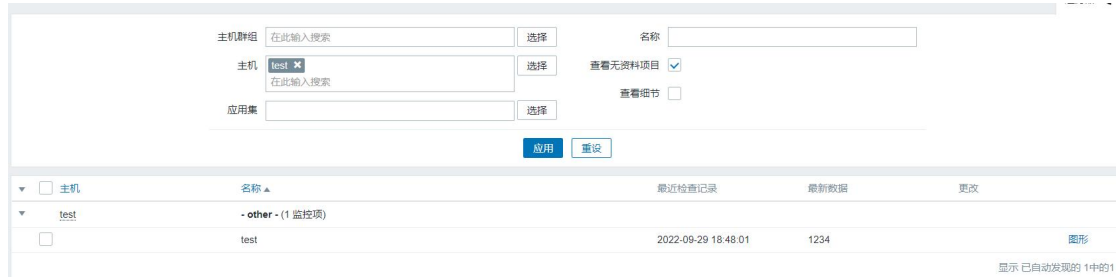
填入主机资产记录栏位 -无-

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

测试发送数据

```
zabbix_sender -z 192.168.1.12 -s test -k key.key -o 1234
```

查看是否可以正常获取数据



测试可以正常获取到数据

## 二、浅谈 Zabbix 的四大监控方式

### 一、Agent

顾名思义，也就是需要在被监控的操作系统安装 agent，通过 agent 和 server 端通讯传输数据。

优点：

- 1、占用系统资源少，每个系统以 200 个监控项计算，平均占用资源不到 0.5%，网络带宽不到 10k/s；
- 2、支持被动、主动的方式传输数据；
- 3、支持接收远程执行命令，可配置关联触发器执行某些进程、服务重启操作；
- 4、支持自定义监控键值，根据监控需求自定义键值、脚本获取某些监控数据；
- 5、保留监控日志，如监控报错可通日志排查。

缺点：

需要逐台安装配置，解决办法，可已下载免编译版本或者 rpm 安装包，编写安装脚本，一般安装方法是使用命令安装后，修改对应的参数，通常只需要修改 Server、ServerActive、Hostname，如需要自动注册则另外配置 Hostmetadata=system.name；后续可在 zabbix server 安装 nginx，使用 nginx 做文件服务器，直接在 linux 上使用命令安装即可，nginx 参考配置文章 利用 Nginx 实现免上传安装 zabbix agent\_乐维\_lwops 的博客-CSDN 博客。

## 二、SNMP

SNMP 也就是简单网络管理协议 (Simple Network Management Protocol, 是专门设计用于在 IP 网络管理网络节点 (服务器、工作站、路由器、交换机及 HUBS 等) 的一种标准协议，它是一种应用层协议)；通常需要结合设备的 MIB(Management Information Base)文件使用。在网络设备，如交换机、路由器、防火墙、行为管理器、AP、AC、加密设备、防毒墙等，以及物理服务器、存储等支持 snmp 功能的都可以通过开启 snmp 相关的服务和配置 (不同设备的 snmp 开启方法，建议参考帮助文档或者官方文档操作)，用 zabbix 进行简单的数据监控。

当然，操作系统也可以使用 snmp 方式监控，但只能是监控到系统层面的运行情况，如 cpu、内存、系统分区、网络流量，支持自定义 oid，但是配置麻烦。

SNMP 优点：

- 1、适用性广，主要应用在物理设备，snmp 使用的设备性能可以忽略不计；
- 2、配置简单，在管理页面操作几下或者执行 6 条左右命令即可配置；



3、数据简洁，snmp 功能的数据可读性比较简洁，通常一个 oid 对应一个数据。

缺点：

- 1、设备太多，需要根据厂家的帮助手册来配置；
- 2、每个厂家的 mib 文件大多私有，部分厂家不向外开放；
- 3、需要根据设备定制化 snmp 模板；
- 4、可监控性比较基础，基本上出厂内置，不易拓展性。

### 三、IPMI

IPMI（智能平台管理接口），Intelligent Platform Management Interface 的缩写。原本是一种 Intel 架构的企业系统的周边设备所采用的一种工业标准。通常是在物理服务器、存储设备最常见，用户可以通过 IPMI 进行设备的常规配置及管理，例如修改管理口 IP、修改管理员账号信息、重启设备等操作，在 zabbix 还可以使用 IPMI 获取设备组件的运行状态，如主板、电源、风扇、传感器的运行状态、运行温度等，一般是在只支持 IPMI 的情况下使用，而且指标比较少，同一厂家不同型号之间的命令也不通用，用的比较少。

### 四、JMX

JMX(Java Management Extensions)是一个为应用程序植入管理功能的框架，从 Java5.0 开始引入到标准 Java 技术平台中。也就是只适用于 java 语言开发的中间件，像 Tomact、Jboss、Resin、Weblogic、IBM WAS、ActiveMQ 等，国产的像中创、金蝶、东方通等

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

都支持使用 jmx 获取运行数据，非国产的中间件多数可以在网络上找得到 jmx 的开启方式，国产的一般需要厂家的支持。JMX 做为 java 类应用的一种监控方式，能通过开启 jmx 功能支持获取中间件的 jvm 运行状态、jvm 的内存池、线程池、老年代青年代的垃圾回收、节点运行状态、会话数等信息。另外，JMX 配置都有固定的格式，同时也支持加密方式，可以参考网上的案例或者官网文档自行配置。

### 三、zabbix 历史数据与趋势数据

历史数据 (history) 和趋势数据 (trends) 是 Zabbix 中存储收集到的数据的两种方式。

历史数据：

- 1: 每分钟收集到的监控数据
- 2: 在正常项目中，一般不建议保留过长的历史数据，历史数据包含了太多信息，对数据库会造成较大的负载压力。
- 3: 可以选择保留较长的趋势数据，来替代历史数据。

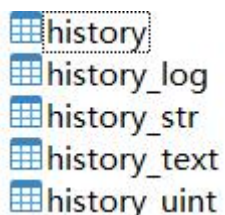
趋势数据：

- 1: 按小时统计计算后的平均数据
- 2: 趋势数据是一种 zabbix 内建的历史数据压缩机制，可以用来存储数字类型监控项的每小时的最小值、最大值、平均值和记录数量。
- 3: 趋势数据设置的留存时间应当比历史数据留存时间设置的长。

历史数据表

从 history\_uint 表里面可以查询到设备监控项目的最大，最小和平均值，即存储监控数据的

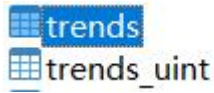
原始数据。

history  
history\_log  
history\_str  
history\_text  
history\_uint

趋势数据表：

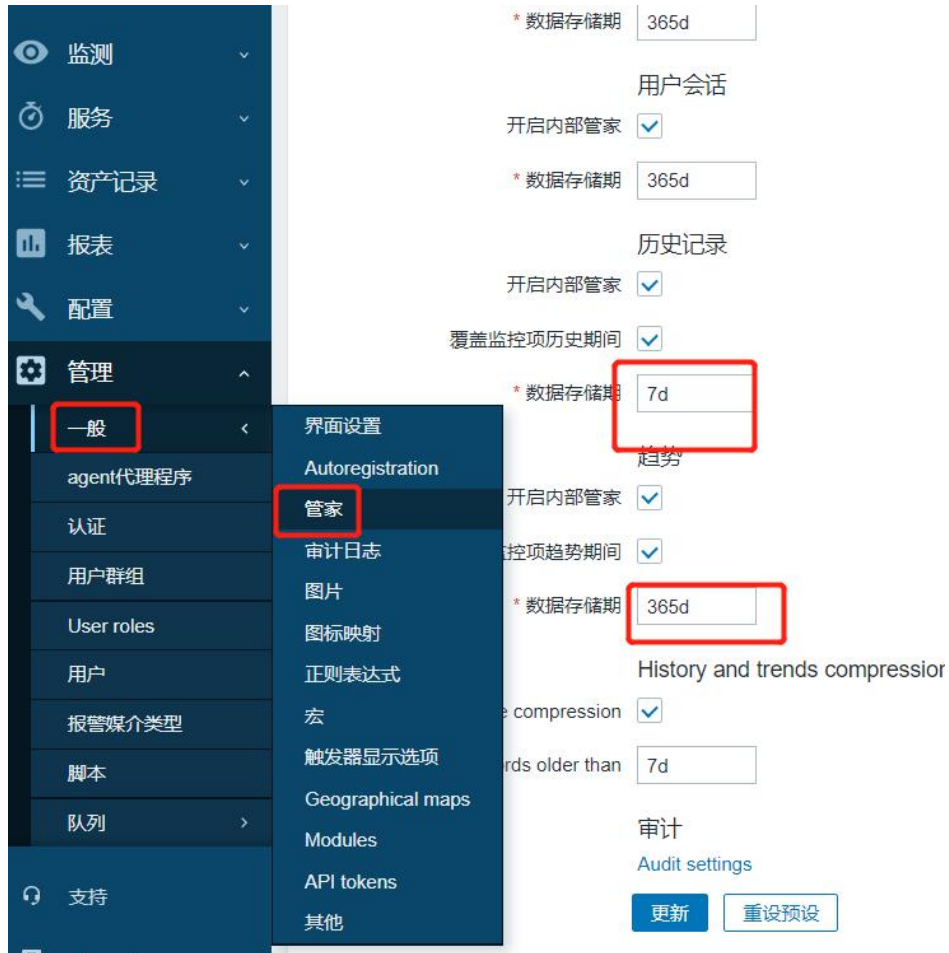
趋势数据是经过 Zabbix 计算的数据，数据是从 history\_uint 里面汇总的，从 trends\_uint 可以查看到监控数据每小时最大，最小和平均值，即存储监控数据的汇总数据。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



历史数据、趋势数据修改:

可通过前端界面更改配置，具体如下:



## 四、更多.....

# 安全

## 十九、如何实现 kafka 与 zookeeper 的 SSL 认证

作者 乐维社区 (forum.lwops.cn/) 乐乐

在构建现代的分布式系统时，确保数据传输的安全性是至关重要的。Apache Kafka 和 Zookeeper 作为流行的分布式消息队列和协调服务，提供了 SSL (Secure Sockets Layer) 认证机制，以增强数据传输过程中的安全性。本文将详细探讨“如何实现 Kafka 与 Zookeeper 的 SSL 认证”，从生成 SSL 证书到配置服务端和客户端的全过程，确保你的数据在传输过程中得到充分的保护。无论你是系统管理员还是开发人员，了解并实施 SSL 认证都是保障系统安全的关键步骤。接下来，我们将一步步引导你完成 Kafka 与 Zookeeper 的 SSL 认证配置，确保你的集群通信安全无忧。

### 一、配置 Kafka 账号密码：

**1、首先，需要修改 kafka 配置文件：vim /asop/kafka/kafka\_2.11-2.1.0/config/server.properties**

```
broker.id=0
```

```
listeners=SASL_PLAINTEXT://:9092
```

```
advertised.listeners=SASL_PLAINTEXT://10.176.31.137:9092
```

```
num.network.threads=3
```

```
num.io.threads=8
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

socket.send.buffer.bytes=102400

socket.receive.buffer.bytes=102400

socket.request.max.bytes=104857600

log.dirs=/asop/kafka/logs

num.partitions=1

num.recovery.threads.per.data.dir=1

offsets.topic.replication.factor=1

transaction.state.log.replication.factor=1

transaction.state.log.min.isr=1

log.retention.hours=168

log.segment.bytes=1073741824

log.retention.check.interval.ms=300000

zookeeper.connect=localhost:2181

zookeeper.connection.timeout.ms=6000

group.initial.rebalance.delay.ms=0

#使用的认证协议

security.inter.broker.protocol=SASL\_PLAINTEXT

#SASL 机制

ssl.enabled.mechanisms=PLAIN

ssl.mechanism.inter.broker.protocol=PLAIN

#完成身份验证的类

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

```
authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer
```

```
#如果没有找到 ACL (访问控制列表) 配置, 则允许任何操作。
```

```
allow.everyone.if.no.acl.found=false
```

```
#需要开启设置超级管理员,设置 visitor 用户为超级管理员
```

```
super.users=User:visitor
```

2、其次, 为 server 创建登录验证文件,可以根据自己爱好命名文件, 如 vim /asop/kafka/kafka\_2.11-2.1.0/config/kafka\_server\_jaas.conf , 文件内容如下

```
KafkaServer {  
    org.apache.kafka.common.security.plain.PlainLoginModule required  
        username="visitor"  
        password="qaz@123"  
        user_visitor="qaz@123";  
};
```

3、然后修改 kafka 安装目录 vim

/asop/kafka/kafka\_2.11-2.1.0/bin/kafka-server-start.sh, 在文件最上面添加变量

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
export KAFKA_OPTS="
```

```
-Djava.security.auth.login.config=/asop/kafka/kafka_2.11-2.1.0/config/kafka_server_jaas.conf"
```

```
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
export KAFKA_OPTS="-Djava.security.auth.login.config=/asop/kafka/kafka_2.11-2.1.0/config/kafka_server_jaas.conf"
if [ $# -lt 1 ]; then
    echo "USAGE: $0 [-daemon] server.properties [--override property=value]"
    exit 1
fi
base_dir=$(dirname $0)
if [ "x$KAFKA_LOG4J_OPTS" = "x" ]; then
    export KAFKA_LOG4J_OPTS="-Dlog4j.configuration=file:$base_dir/./config/log4j.properties"
fi
if [ "x$KAFKA_HEAP_OPTS" = "x" ]; then
    export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G"
fi
```

4、接下来为 consumer 和 producer 创建登录验证文件，可以根据爱好命名文件，如 kafka\_client\_jaas.conf，文件内容如下 (如果是程序访问，如 **springboot** 访问，可以不配置)

```
vim /asop/kafka/kafka_2.11-2.1.0/config/kafka_client_jaas.conf
```

```
KafkaClient {
```

```
org.apache.kafka.common.security.plain.PlainLoginModule required
```

```
    username="visitor"
```

```
    password="qaz@123";
```

```
};
```

5、在 consumer.properties 和 producer.properties 里分别加上如下配置：

```
vim /asop/kafka/kafka_2.11-2.1.0/config/consumer.properties
```

```
vim /asop/kafka/kafka_2.11-2.1.0/config/producer.properties
```



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
security.protocol=SASL_PLAINTEXT
```

```
sasl.mechanism=PLAIN
```

## 6、修改 kafka 安装目录 bin/kafka-console-producer.sh 和

bin/kafka-console-consumer.sh，在文件最上面添加变量

```
vim /asop/kafka/kafka_2.11-2.1.0/bin/kafka-console-producer.sh
```

```
vim /asop/kafka/kafka_2.11-2.1.0/bin/kafka-console-consumer.sh
```

```
export KAFKA_OPTS="
```

```
-Djava.security.auth.login.config=/asop/kafka/kafka_2.11-2.1.0/config/k
```

```
afka_client_jaas.conf"
```

```
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
export KAFKA_OPTS="-Djava.security.auth.login.config=/asop/kafka/kafka_2.11-2.1.0/config/kafka_client_jaas.conf"  
if [ "$KAFKA_HEAP_OPTS" = "x" ]; then  
    export KAFKA_HEAP_OPTS="-Xmx512M"  
fi  
  
exec $(dirname $0)/kafka-run-class.sh kafka.tools.ConsoleConsumer "$@"  
~  
~  
~
```

## 7、分别启动 zookeeper 和 kafka,至此服务端 kafka 用户登录验证配置完成(先

关闭 kafka 后关闭 zookeeper)

关闭服务 kafka

```
# /asop/kafka/kafka_2.11-2.1.0/bin/kafka-server-stop.sh -daemon
```

```
/asop/kafka/kafka_2.11-2.1.0/config/server.properties
```

启动服务 kafka

```
#!/asop/kafka/kafka_2.11-2.1.0/bin/kafka-server-start.sh -daemon  
  
/asop/kafka/kafka_2.11-2.1.0/config/server.properties
```

关闭服务 zookeeper-3.4.13

```
/asop/zk/zookeeper-3.4.13/bin/zkServer.sh stop  
  
/asop/zk/zookeeper-3.4.13/conf/zoo.cfg
```

启动服务 zookeeper-3.4.13

```
/asop/zk/zookeeper-3.4.13/bin/zkServer.sh start  
  
/asop/zk/zookeeper-3.4.13/conf/zoo.cfg
```

## 8、创建及查看主题

```
/asop/kafka/kafka_2.11-2.1.0/bin/kafka-console-producer.sh --broker-list  
10.176.31.137:9092 --topic cmdb --producer-property  
security.protocol=SASL_PLAINTEXT --producer-property sasl.mechanism=PLAIN
```

```
[root@ZZCXCSCSY176X31137 ~]# /asop/kafka/kafka_2.11-2.1.0/bin/kafka-console-producer.sh --broker-list 10.176.31.137:9092 --topic cmdb  
>test  
>abcd  
>
```

接收消息

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
/asop/kafka/kafka_2.11-2.1.0/bin/kafka-console-consumer.sh --bootstrap-server  
10.176.31.137:9092 --topic cmdb --from-beginning --consumer-property  
security.protocol=SASL_PLAINTEXT --consumer-property sasl.mechanism=PLAIN
```

```
[root@ZZCXCSY176X31137 ~]# /asop/kafka/kafka_2.11-2.1.0/bin/kafka-console-consumer.sh --bootstrap-server 10.176.31.137:9092 --topic cmdb  
test  
abcd
```

## 二、zk 和 kafka 配置 sasl 账号密码：

### 1. Zookeeper 配置 SASL

#### 1.1 新建 zoo\_jaas.conf 文件

zoo\_jaas.conf 文件名、文件所在路径没有特殊要求，一般放置在  
\${ZOOKEEPER\_HOME}/conf 目录下 vim  
/asop/zk/zookeeper-3.4.13/conf/zoo\_jaas.conf

```
Server {  
  
    org.apache.kafka.common.security.plain.PlainLoginModule required  
  
    username="admin"  
  
    password="admin@12"  
  
    user_kafka="kafka@123";  
  
};
```

Server.username、Server.password 为 Zookeeper 内部通信的用户名和密码，

因此保证每个 zk 节点该属性一致即可

Server.user\_xxx 中 xxx 为自定义用户名, 用于 zkClient 连接所使用的用户名和密码, 即为 kafka 创建的用户名

## 1.2 配置 /asop/zk/zookeeper-3.4.13/conf/zoo.cfg 文件

```
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
```

```
requireClientAuthScheme=sasl
```

```
jaasLoginRenew=3600000
```

```
zookeeper.sasl.client=true
```

zookeeper.sasl.client 设置为 true, 开启客户端身份验证, 否则 zoo\_jaas.conf 中配置的用户名将不起作用, 客户端仍然可以无 jaas 文件连接, 只是带有 WARNING 而已

## 1.3 导入依赖包

因为使用的权限验证类为 :  
org.apache.kafka.common.security.plain.PlainLoginModule, 所以需要 kafka 相关 jar 包, 新建文件夹 zk\_sasl\_lib, 如下: 从 kafka/lib 目录下复制以下几个 jar 包到 zookeeper 的 lib 和新建的 zk\_sasl\_lib 目录下:

```
kafka-clients-2.4.1.jar
```

```
lz4-java-1.6.0.jar
```

```
slf4j-api-1.7.28.jar
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
slf4j-log4j12-1.7.28.jar
```

```
snappy-java-1.1.7.3.jar
```

```
mkdir /asop/zk/zookeeper-3.4.13/zk_sasl_lib
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/kafka-clients-2.1.0.jar
```

```
/asop/zk/zookeeper-3.4.13/lib/
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/lz4-java-1.5.0.jar
```

```
/asop/zk/zookeeper-3.4.13/lib/
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/slf4j-api-1.7.25.jar
```

```
/asop/zk/zookeeper-3.4.13/lib/
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/slf4j-log4j12-1.7.25.jar
```

```
/asop/zk/zookeeper-3.4.13/lib/
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/snappy-java-1.1.7.2.jar
```

```
/asop/zk/zookeeper-3.4.13/lib/
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/kafka-clients-2.1.0.jar
```

```
/asop/zk/zookeeper-3.4.13/zk_sasl_lib
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/lz4-java-1.5.0.jar
```

```
/asop/zk/zookeeper-3.4.13/zk_sasl_lib
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/slf4j-api-1.7.25.jar
```

```
/asop/zk/zookeeper-3.4.13/zk_sasl_lib
```

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/slf4j-log4j12-1.7.25.jar
```

```
/asop/zk/zookeeper-3.4.13/zk_sasl_lib
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
cp /asop/kafka/kafka_2.11-2.1.0/libs/snappy-java-1.1.7.2.jar
```

```
/asop/zk/zookeeper-3.4.13/zk_sasl_lib
```

```
chmod 755 -R /asop/zk/zookeeper-3.4.13/zk_sasl_lib/
```

```
chmod 755 -R /asop/zk/zookeeper-3.4.13/zk_sasl_lib/
```

#### 1.4 修改 zkEnv.sh 文件/asop/zk/zookeeper-3.4.13/bin/zkEnv.sh

修改前：如果没有就直接加

```
export SERVER_JVMFLAGS="-Xmx${ZK_SERVER_HEAP}m $SERVER_JVMFLAGS"
```

修改后：

```
for jar in /asop/zk/zookeeper-3.4.13/zk_sasl_lib/*.jar;
```

```
do
```

```
    CLASSPATH="$jar:$CLASSPATH"
```

```
done
```

```
export SERVER_JVMFLAGS="
```

```
-Djava.security.auth.login.config=/asop/zk/zookeeper-3.4.13/conf/zoo_jaas.conf "
```

重启 Zookeeper 服务即可

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

关闭服务 zookeeper-3.4.13

```
/asop/zk/zookeeper-3.4.13/bin/zkServer.sh stop
```

```
/asop/zk/zookeeper-3.4.13/conf/zoo.cfg
```

启动服务 zookeeper-3.4.13

```
/asop/zk/zookeeper-3.4.13/bin/zkServer.sh start
```

```
/asop/zk/zookeeper-3.4.13/conf/zoo.cfg
```

## 2. Kafka 配置 SASL

### 2.1 新建 kafka\_server\_jaas.conf 文件

kafka\_server\_jaas.conf 文件名和存放路径没有要求，一般放置在

`${KAFKA_HOME}/config` 目录下

```
/asop/kafka/kafka_2.11-2.1.0/config/kafka_server_jaas.conf
```

```
KafkaServer {
```

```
    org.apache.kafka.common.security.plain.PlainLoginModule required
```

```
        username="visitor"
```

```
        password="qaz@123"
```

```
        user_visitor="qaz@123";
```

```
};
```

```
Client{
```

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

```
org.apache.kafka.common.security.plain.PlainLoginModule required  
  
    username="kafka"  
  
    password="kafka@123";  
  
};
```

KafkaServer.username、KafkaServer.password 为 broker 内部通信的用户名  
密码, 同上

KafkaServer.user\_xxx 其中 xxx 必须和 KafkaServer.username 配置的用户名一致, 密码也一致

KafkaServer.user\_producer、KafkaServer.user\_consumer 为了之后的 ACL 做准备, 达到消费者生产者使用不同账号且消费者账号只能消费数据, 生产者账号只能生产数据

Client.username、Client.password 填写 Zookeeper 中注册的账号密码, 用于 broker 与 zk 的通信 (若 zk 没有配置 SASL 可以忽略、若 zookeeper.sasl.client 为 false 也可以忽略只是带有, 日志如下)

```
[2021-06-29 17:14:30,204] WARN SASL configuration failed:
```

```
javax.security.auth.login.LoginException: No JAAS configuration section named
```

```
'Client' was found in specified JAAS configuration file:
```

```
'/Users/wjun/env/kafka/config/kafka_server_jaas.conf'. Will continue connection to  
Zookeeper server without SASL authentication, if Zookeeper server allows it.
```

```
(org.apache.zookeeper.ClientCnxn)
```



## 2.2 修改 server.properties 文件

broker.id=0

listeners=SASL\_PLAINTEXT://:9092

advertised.listeners=SASL\_PLAINTEXT://192.168.157.198:9092

num.network.threads=3

num.io.threads=8

socket.send.buffer.bytes=102400

socket.receive.buffer.bytes=102400

socket.request.max.bytes=104857600

log.dirs=/asop/kafka/logs

num.partitions=1

num.recovery.threads.per.data.dir=1

offsets.topic.replication.factor=1

transaction.state.log.replication.factor=1

transaction.state.log.min.isr=1

log.retention.hours=168

log.segment.bytes=1073741824

log.retention.check.interval.ms=300000

zookeeper.connect=127.0.0.1:2181

zookeeper.connection.timeout.ms=6000

group.initial.rebalance.delay.ms=0

#使用的认证协议

security.inter.broker.protocol=SASL\_PLAINTEXT

#SASL 机制

sasl.enabled.mechanisms=PLAIN

sasl.mechanism.inter.broker.protocol=PLAIN

#完成身份验证的类

authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer

#如果没有找到 ACL (访问控制列表) 配置，则允许任何操作。

allow.everyone.if.no.acl.found=false

#需要开启设置超级管理员,设置 visitor 用户为超级管理员

super.users=User:visitor

其中 localhost 需要修改成 IP 地址

super.users 配置超级用户，该用户不受之后的 ACL 配置影响

## 2.3 修改启动脚本

修改 kafka-server-start.sh 文件，使之加载到 kafka\_server\_jaas.conf 文件

/asop/kafka/kafka\_2.11-2.1.0/bin/kafka-server-start.sh

修改前:

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
if [ "x$KAFKA_HEAP_OPTS" = "x" ]; then  
    export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G"  
fi
```

修改后:

```
(先在首行加这一行,如果有了就不用加了) export KAFKA_OPTS="  
-Djava.security.auth.login.config=/asop/kafka/kafka_2.11-2.1.0/config/kafka_server  
_jaas.conf"  
if [ "x$KAFKA_HEAP_OPTS" = "x" ]; then  
    export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G  
-Djava.security.auth.login.config=/asop/kafka/kafka_2.11-2.1.0/config/kafka_server  
_jaas.conf"  
fi
```

设置 zookeeper 的 ACL 规则

```
/asop/zk/zookeeper-3.4.13/bin/zkCli.sh    #进入 zk 的命令行模式  
  
addauth digest admin:admin@12    #切换登陆用户 (超级管理员是在 zk 的配置文  
件/asop/zk/zookeeper-3.4.13/conf/zoo_jaas.conf 里面)
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
setAcl / ip:127.0.0.1:cdrwa,auth:kafka:kafka@123:cdrwa    #(设置可以登陆的 IP 和用  
户账号密码， admin 是上面的 zk 的配置文件里面定义的管理员， Kafka 用户是  
/asop/kafka/kafka_2.11-2.1.0/config/kafka_server_jaas.conf 文件里面的定义的 kafka  
连接 zk 的用户 (Client 下面的) )
```

```
addauth digest kafka:kafka@123    #再切换为 kafka 用户再设置一次 acl
```

```
setAcl / ip:127.0.0.1:cdrwa,auth:kafka:kafka@123:cdrwa
```

**注意: 如果要加白名单 IP 或者用户要在原来的基础上加，不然会覆盖**

```
setAcl / ip:127.0.0.1:cdrwa,auth:kafka:kafka@123:cdrwa,auth:admin:admin@12:cdrwa,ip:1.1.  
1.1
```

需要恢复权限，不设置 acl 的话就运行

```
setAcl / world:anyone:cdrwa
```

重启 kafka 服务即可

关闭服务 kafka

```
# /asop/kafka/kafka_2.11-2.1.0/bin/kafka-server-stop.sh -daemon  
  
/asop/kafka/kafka_2.11-2.1.0/config/server.properties
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

启动服务 kafka

```
#!/asop/kafka/kafka_2.11-2.1.0/bin/kafka-server-start.sh -daemon
```

```
/asop/kafka/kafka_2.11-2.1.0/config/server.properties
```

至此，完成 kafka 与 zookeeper 配置 ssl 认证。大家好，我是乐乐专注 IT 运维技术研究与分享，更多运维技巧欢迎关注乐维社区，更多运维问题也欢迎留言提问。

## 二十、如何在 Weblogic 环境中启动认证方式对接 Zabbix 监控

在 WebLogic Server 中，启动认证可用于确保只有经过授权的用户和系统能够访问 WebLogic Server 及其应用程序，通过合理配置认证提供者和安全领域，管理员可以有效管理和控制用户访问。

本文将详细介绍如何在 Weblogic 环境中配置启动认证并对接 Zabbix 监控，通过设置 Weblogic 的相关配置文件、创建认证文件并配置监控部分，确保只有授权用户才能启动 Weblogic 服务器，并实现对其运行状态的实时监控。

这一实践将会增强 Weblogic 服务器的安全性并提高系统的运维效率。

### 1.配置及创建 weblogic 相关配置文件

#### 1.1 环境

##### 1.1.1 配置 weblogic 配置文件相关位置为 domain 下的 setDomainEnv.sh

```
[weblogic@zabbix-server bin]$ ls
nodemanager          setDomainEnv.sh      stopManagedWebLogic.sh
server_migration     startManagedWebLogic.sh  stopWebLogic.sh
service_migration    startWebLogic.sh
[weblogic@zabbix-server bin]$ pwd
/home/weblogic/Oracle/Middleware/user_projects/domains/base_domain/
bin
[weblogic@zabbix-server bin]$
```

相关修改如下：

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Djava.rmi.server.hostname=172.17.10.100"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS}
```

```
-Djavax.management.builder.initial=weblogic.management.jmx.mbeanserver.WLS
```

```
MBeanServerBuilder"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote=true"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS}
```

```
-Dcom.sun.management.jmxremote.port=12345"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS}
```

```
-Dcom.sun.management.jmxremote.rmi.port=12345"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.ssl=false"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS}
```

```
-Dcom.sun.management.jmxremote.authenticate=true"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS}
```

```
-Dcom.sun.management.jmxremote.access.file=/home/weblogic/Oracle/Middlewa  
re/user_projects/domains/base_domain/jmxremote.access"
```

```
JAVA_OPTIONS="${JAVA_OPTIONS}
```

```
-Dcom.sun.management.jmxremote.password.file=/home/weblogic/Oracle/Middl  
eware/user_projects/domains/base_domain/jmxremote.password"
```

```
export JAVA_OPTIONS
```

注: /home/weblogic/Oracle/Middleware/user\_projects/domains/base\_domain/位置  
为自己定义文件的位置或文件所在位置, **jmxremote.access**、**jmxremote.password** 文  
件有则使用, 没有则创建。

### 1.1.2 创建认证文件 (weblogic 账号、密码)，修改相应权限。

```
[weblogic@zabbix-server base_domain]$ cat jmxremote.access
weblogic readonly
[weblogic@zabbix-server base_domain]$ cat jmxremote.password
weblogic weblogic@123
```

Chmod 600 jmxremote.\*

```
rw-r----- 1 weblogic weblogic 18 4月 24 10:11 jmxremote.access
rw-r----- 1 weblogic weblogic 22 4月 24 10:11 jmxremote.password
```

**注：access 文件前面的 weblogic 为用户，设为只读，password 文件前面为用户后面为密码，权限修改必做，否则可能会发生错误无法启动!!!**

### 1.1.3 启动 weblogic (我之前是停止的，按实际需求操作)

```
[weblogic@zabbix-server base_domain]$ ls
autodeploy  fileRealm.properties  lib  startWebLogic.sh
bin         init-info             security
config     jmxremote.access      servers
console-ext jmxremote.password    shutdown.py
[weblogic@zabbix-server base_domain]$ ./startWebLogic.sh
```

```
Starting WLS with line:
/usr/local/java/bin/java -server -Xms256m -Xmx512m -XX:MaxPermSi
ze=256m -Dweblogic.Name=AdminServer -Djava.security.policy=/home/we
blogic/Oracle/Middleware/wlserver_10.3/server/lib/weblogic.policy
-Dweblogic.ProductionModeEnabled=true -da -Dplatform.home=/home/w
eblogic/Oracle/Middleware/wlserver_10.3 -Dwls.home=/home/weblogi
c/Oracle/Middleware/wlserver_10.3/server -Dweblogic.home=/home/weblogi
c/Oracle/Middleware/wlserver_10.3/server -Dweblogic.management.di
scover=true -Dwls.iterativeDev=false -Dwls.testConsole=false -Dwls
.logErrorsToConsole=false -Dweblogic.ext.dirs=/home/weblogic/Oracle
/Middleware/patch_wls1036/profiles/default/sysexm_manifest_classpat
h:/home/weblogic/Oracle/Middleware/patch_ocp371/profiles/default/sy
sexm_manifest_classpath -Djava.rmi.server.hostname=172.17.10.100 -D
javax.management.builder.initial=weblogic.management.jmx.mbeanserve
r.WLSMBeanServerBuilder -Dcom.sun.management.jmxremote=true -Dcom.s
un.management.jmxremote.port=12345 -Dcom.sun.management.jmxremote.r
mi.port=12345 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.ma
nagement.jmxremote.authenticate=true -Dcom.sun.management.jmxremote
.access.file=/home/weblogic/Oracle/Middleware/user_projects/domains
/base_domain/jmxremote.access -Dcom.sun.management.jmxremote.passwo
rd.file=/home/weblogic/Oracle/Middleware/user_projects/domains/base
_domain/jmxremote.password weblogic.Server
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermS
ize=256m; support was removed in 8.0 这里提示认证开启成功，并开始加载配置文件
```



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
Using root identity from user: root
Enter username to boot WebLogic server:weblogic
Enter password to boot WebLogic server:
2024-11-21 10:12:04.197 UTC+08:00: WebLogicServer [130]
360> <Server started in RUNNING mode>
```

输入用户名密码  
启动成功

## 2. 监控部分配置

### 2.1 环境

2.1.1 如果已经监控过并触发更新数据，先执行清除操作，如果没有见 2.1.2

模板  
Inherited items 29 Not inherited items 22

有触发器

名称	触发器	键值	间隔	历史记录	模板	类型	状态	标记
获取数据库执行的指令: 数据库buildInfo指令每小时执行次数	mongodb[buildInfo.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库buildInfo指令每小时执行的失败次数	mongodb[buildInfo.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库delete指令每小时执行次数	mongodb[delete.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库delete指令每小时执行的失败次数	mongodb[delete.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库endSessions指令每小时执行次数	mongodb[endSessions.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库endSessions指令每小时执行的失败次数	mongodb[endSessions.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库find指令每小时执行次数	mongodb[find.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库find指令每小时执行的失败次数	mongodb[find.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库ismaster指令每小时执行次数	mongodb[ismaster.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库ismaster指令每小时执行的失败次数	mongodb[ismaster.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库listIndexes指令每小时执行次数	mongodb[listIndexes.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库listIndexes指令每小时执行的失败次数	mongodb[listIndexes.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库repSetGetStatus指令每小时执行次数	mongodb[repSetGetStatus.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库repSetGetStatus指令每小时执行的失败次数	mongodb[repSetGetStatus.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库sasContinue指令每小时执行次数	mongodb[sasContinue.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库sasContinue指令每小时执行的失败次数	mongodb[sasContinue.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库sasStart指令每小时执行次数	mongodb[sasStart.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库sasStart指令每小时执行的失败次数	mongodb[sasStart.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库serverStatus指令每小时执行次数	mongodb[serverStatus.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库serverStatus指令每小时执行的失败次数	mongodb[serverStatus.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库whatsmyuri指令每小时执行次数	mongodb[whatsmyuri.total]	7d	365d	Zabbix采集器	已启用	Application: 数据库		
获取数据库执行的指令: 数据库whatsmyuri指令每小时执行的失败次数	mongodb[whatsmyuri.failed]	7d	365d	Zabbix采集器	已启用	Application: 数据库		

22 选择 应用 应用 Execute now 清除历史 复制 批量更新 清除

显示 已自动发现的 22 个

2.1.2 配置账号密码以修改模板宏方式配置 (账号密码为真实账号密码)

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

## 模板

所有模板 / 中间件Java JMX环境模板[不迁移...] / 监控项 31 / 触发器 8 / 图形 / 仪表盘 / 自动发现规则 2 / Web 场景

模板 标记 宏 值映射

\* 模板名称

可见的名称

模板

\* 群组

描述

## 链接模板

所有模板 / 中间件Java JMX环境模板克隆[不...] / 监控项 31 / 触发器 8 / 图形 / 仪表盘 / 自动发现规则 5 / Web 场景

模板 标记 宏 2 值映射

\* 模板名称

可见的名称

模板	名称	动作
	中间件Weblogic模板[通用JMX]	<a href="#">取消链接</a> <a href="#">取消链接并清理</a>

\* 群组

描述

相关的自动发现规则中 username、password 改为相应宏值

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

所有模板 / 中间件Java JMX环境模板克隆(不... 自动发现清单 / JMX Garbage Collector(垃圾收集器) 监控项原型 3 触发器类型 图形原型

自动发现规则 进程 LLD macros 过滤器 覆盖

\* 名称 JMX Garbage Collector(垃圾收集器)

类型 JMX agent代理程序

\* 键值 jmx.discovery[beans,"\*:type=GarbageCollector,name=\*"]

\* JMX 端点 service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi

用户名称 {\$JMX\_USERNAME}

密码 {\$JMX\_PASSWORD}

\* 更新间隔 3600s

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

\* 资源周期不足 30d

描述

已启用

## 监控项原型也要改

### 监控项原型

所有模板 / 中间件Java JMX环境模板克隆(不... 自动发现清单 / JMX Garbage Collector(垃圾收集器) 监控项原型 3 触发器类型 图形原型 主机模板

监控项原型 标记 1 进程

\* 名称 GC (#JMXNAME)垃圾收集器可用状态

类型 JMX agent代理程序

\* 键值 jmx{#JMXOBJ}.Valid 选择

信息类型 字符

\* JMX 端点 service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi

用户名称 {\$JMX\_USERNAME}

密码 {\$JMX\_PASSWORD}

\* 更新间隔 180

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

值映射 在此输入搜索 选择

## 相关监控项批量修改

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

<input checked="" type="checkbox"/>	... 已提交的非堆内存	触发器 1	jmx["java.lang.type=Memory",NonHeapMemoryUsage.committed]	180	90d	365d	JMX agent代理程序	已启用	Application: 堆内存
<input checked="" type="checkbox"/>	... 当前JIT编译器的名称	触发器 1	jmx["java.lang.type=Compilation",Name]	180	90d		JMX agent代理程序	已启用	Application: 编译器
<input checked="" type="checkbox"/>	... 总加载类计数		jmx["java.lang.type=ClassLoading",TotalLoadedClassCount]	180	90d	365d	JMX agent代理程序	已启用	Application: 类计数
<input checked="" type="checkbox"/>	... 总物理内存大小		jmx["java.lang.type=OperatingSystem",TotalPhysicalMemorySize]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 打开文件描述符计数	触发器 1	jmx["java.lang.type=OperatingSystem",OpenFileDescriptorCount]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 提交虚拟内存大小		jmx["java.lang.type=OperatingSystem",CommittedVirtualMemorySize]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 最大文件描述符计数	触发器 1	jmx["java.lang.type=OperatingSystem",MaxFileDescriptorCount]	3600	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 活动守护线程的当前数目		jmx["java.lang.type=Threading",DaemonThreadCount]	180	90d	365d	JMX agent代理程序	已启用	Application: 线程
<input checked="" type="checkbox"/>	... 活动线程的当前数目, 包括守护线程和非守护线程		jmx["java.lang.type=Threading",ThreadCount]	180	90d	365d	JMX agent代理程序	已启用	Application: 线程
<input checked="" type="checkbox"/>	... 空闲物理内存大小		jmx["java.lang.type=OperatingSystem",FreePhysicalMemorySize]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 系统CPU负载		jmx["java.lang.type=OperatingSystem",SystemCpuLoad]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 系统名称		jmx["java.lang.type=OperatingSystem",Name]	3600	90d		JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 系统平均负载		jmx["java.lang.type=OperatingSystem",SystemLoadAverage]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 编译花费的总时间		jmx["java.lang.type=Compilation",TotalCompilationTime]	180	90d	365d	JMX agent代理程序	已启用	Application: 编译器
<input checked="" type="checkbox"/>	... 自由交换空间大小		jmx["java.lang.type=OperatingSystem",FreeSwapSpaceSize]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 进程CPU负载		jmx["java.lang.type=OperatingSystem",ProcessCpuLoad]	180	90d	365d	JMX agent代理程序	已启用	Application: 系统
<input checked="" type="checkbox"/>	... 非堆内存最大值	触发器 2	jmx["java.lang.type=Memory",NonHeapMemoryUsage.max]	180	90d	365d	JMX agent代理程序	已启用	Application: 堆内存

显示 已自动发现的 31中的31

31 选择

### 批量更新

监控项 标记 进程

- 单位  原始的
- 认证方法  原始的
- 用户名称
- 公钥文件  原始的
- 私钥文件  原始的
- 密码
- 更新间隔  原始的
- 历史数据保留时长  原始的
- 趋势存储时间  原始的
- 状态  原始的
- 日志时间格式  原始的
- 值映射  原始的
- 启用trapping  原始的
- 允许的主机  原始的

### 模板

模板 标记  2 值映射

宏	值	描述	
<input checked="" type="checkbox"/> {JMX_USERNAME}	<input type="text" value="值"/>	<input type="text" value="The user of webllogic"/>	<input type="button" value="移除"/>
<input checked="" type="checkbox"/> {JMX_PASSWORD}	<input type="text" value="值"/>	<input type="text" value="The password of webllogic"/>	<input type="button" value="移除"/>

所有主机  webllogic 已启用  JMX 自动发现清单 / JMX Garbage Collector(垃圾收集器) 监控项原型 3 触发器类型 图形原型 主机模板

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

主机

主机 IPMI 标记 宏 2 资产记录 加密 值映射

\* 主机名称

可见的名称

模板 名称  动作

\* 群组

Interfaces	类型	IP地址	DNS名称	连接到	端口	默认
JMX		<input type="text" value="172.17.10.100"/>	<input type="text"/>	<input checked="" type="radio"/> IP <input type="radio"/> DNS	<input type="text" value="12345"/>	<input checked="" type="radio"/> 移除

描述

主机

主机 IPMI 标记 宏 2 资产记录 加密 值映射

\* 主机名称

可见的名称

模板

\* 群组

Interfaces	类型	IP地址	DNS名称	连接到	端口	默认
JMX		<input type="text" value="172.17.10.100"/>	<input type="text"/>	<input checked="" type="radio"/> IP <input type="radio"/> DNS	<input type="text" value="12345"/>	<input checked="" type="radio"/> 移除

描述

## 修改监控资源

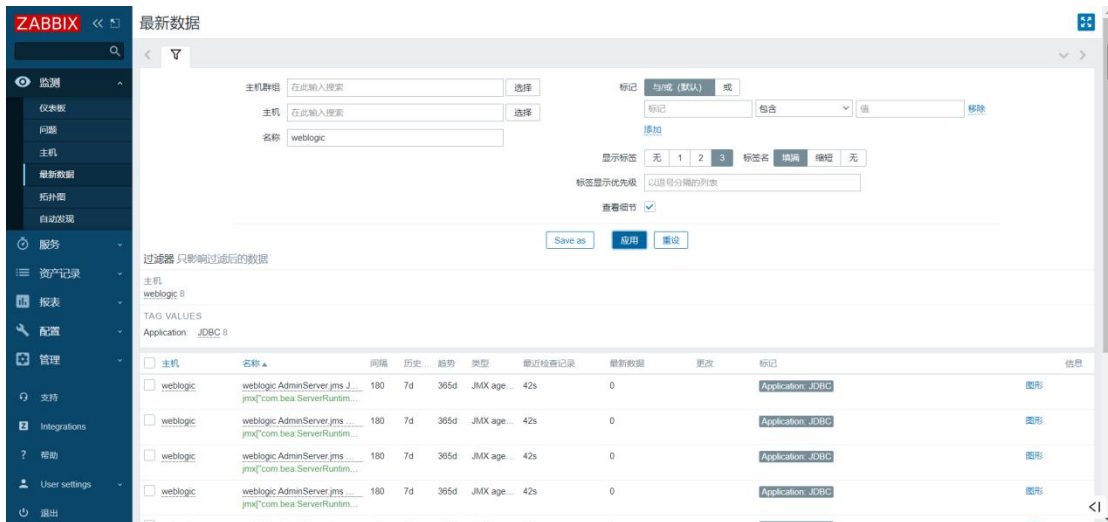
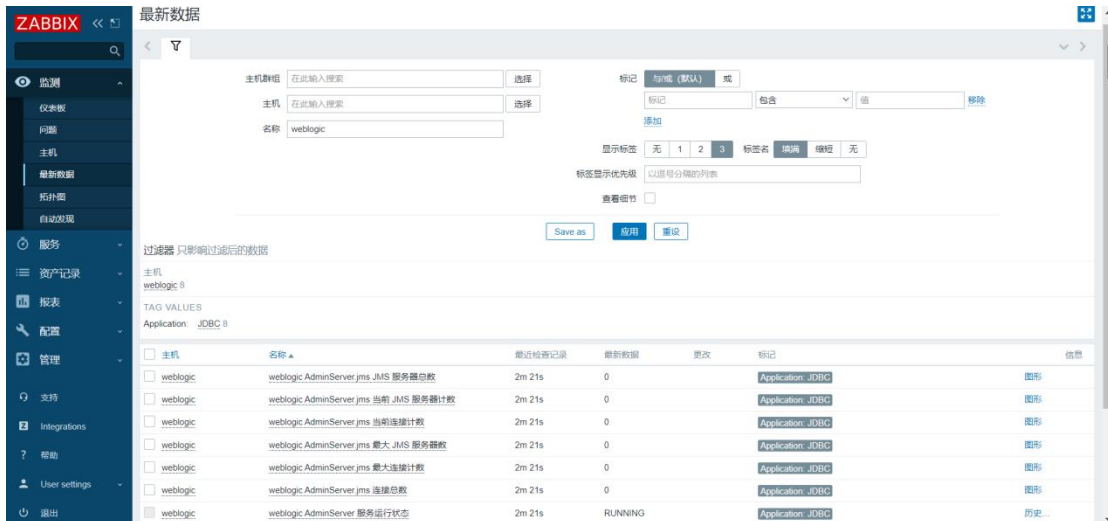
所有主机 /  已启用  监控项 31 触发器 8 图形 自动发现规则 2 Web 场景

## 修改相应值为真实值

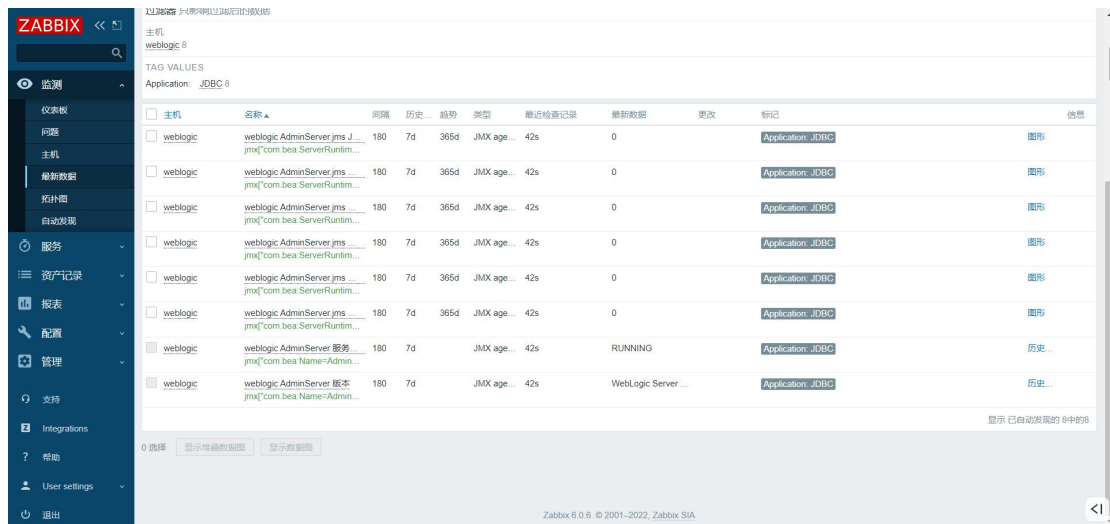
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



等待数据采集，并查看监控结果：



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



到此，全部配置完成。大家好，我是乐乐，专注 IT 运维技术研究与分享，更多运维技巧及运维问题欢迎关注乐维社区并留言~



## 二十一、如何通过 Zabbix Docker 配置 HTTPS 访问系统?

### 概述

前面文章曾介绍过[如果使用 docker-compose 快速部署一个 Zabbix 系统](#)，但是部署的 Zabbix 系统是使用 http 协议进行访问的。有时候为了保证安全。我们需要配置使用 https 协议进行访问。

下面就讲述如何使用自签名的 ssl 证书配置 https 访问。（注：若是有签发的证书，也可使用配置，无须自己生成自签名证书。）

### 前提条件

本文主要讲述 Zabbix 官方镜像如何配置 https 访问，需为 Zabbix 官方提供的 zabbix docker 镜像来部署的 Zabbix 监控系统。

### 生成自签名证书

首先，先生成自签名证书。

这里提供一个快速生成证书脚本，执行脚本需要输入一个 IP 的参数，然后会在脚本所在目录下面生成名为 ssl.crt 的证书和 ssl.key 的密钥。



```
[root@localhost ssl]# ls
ssl.sh
[root@localhost ssl]#
[root@localhost ssl]# cat ssl.sh
#!/bin/bash
baseshell=$(cd $(dirname $0);pwd)
# 变量定义
IP=$1

# 生成证书配置文件
cat > ${baseshell}/ssl.cnf << EOF
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no
[req_distinguished_name]
C = CN
ST = GD
L = GZ
O = EDGE
OU = BASE
CN = $IP
[v3_req]
keyUsage = critical, digitalSignature, keyAgreement
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
[ alt_names ]
DNS.1 = $IP
DNS.2 = 8.8.8.8
IP.1 = $IP
EOF
# 生成crt和key证书
openssl req -x509 -nodes -days 9999 -newkey rsa:2048 -keyout ${baseshell}/ssl.key -out ${baseshell}/ssl.crt -config ${baseshell}/ssl.cnf -sha256
[root@localhost ssl]#
[root@localhost ssl]# bash ssl.sh 192.168.1.1
Generating a 2048 bit RSA private key
.....+++
writing new private key to '/tmp/ssl/ssl.key'
.....
[root@localhost ssl]# ls
ssl.cnf  ssl.crt  ssl.key  ssl.sh
[root@localhost ssl]#
```

```
#!/bin/bash
baseshell=$(cd $(dirname $0);pwd)
# 变量定义
IP=$1
# 生成证书配置文件
cat > ${baseshell}/ssl.cnf << EOF
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no
[req_distinguished_name]
C = CN
ST = GD
L = GZ
O = EDGE
OU = BASE
CN = $IP
[v3_req]
keyUsage = critical, digitalSignature, keyAgreement
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
[ alt_names ]
DNS.1 = $IP
DNS.2 = 8.8.8.8
IP.1 = $IP
EOF
```



```
0 docker-compose.yml nginx.com ssl
[root@localhost zabbix]# cat docker-compose.yml
version: '3.7'
services:
  postgresql-6:
    container_name: zabbix_postgresql_6_test
    image: postgres:14.4
    restart: always
    environment:
      POSTGRES_DB: zabbix
      POSTGRES_PASSWORD: zabbix@2022
      POSTGRES_USER: zabbix
    ports:
      - "45432:5432"
    volumes:
      - ./6/data:/var/lib/postgresql/data
      - ./6/opt:/opt
  zabbix-server-6:
    container_name: zabbix_server_6_test
    image: zabbix/zabbix-server-pgsql:6.0.6-centos
    restart: always
    environment:
      POSTGRES_DB: zabbix
      POSTGRES_PASSWORD: zabbix@2022
      POSTGRES_USER: zabbix
      DB_SERVER_HOST: 192.168.3.141
      DB_SERVER_PORT: 45432
    ports:
      - "40051:10051"
    volumes:
      - ./6/zabbix/alertscripts:/usr/lib/zabbix/alertscripts
      - ./6/zabbix/externalscripts:/usr/lib/zabbix/externalscripts
  zabbix-web-6:
    container_name: zabbix_web_6_test
    image: zabbix/zabbix-web-nginx-pgsql:6.0.6-ubuntu
    restart: always
    environment:
      POSTGRES_DB: zabbix
      POSTGRES_PASSWORD: zabbix@2022
      POSTGRES_USER: zabbix
      DB_SERVER_HOST: 192.168.3.141
      DB_SERVER_PORT: 45432
      ZBX_SERVER_HOST: 192.168.3.141
      ZBX_SERVER_PORT: 40051
    ports:
      - "48080:8080"
      - "48443:8443"
    volumes:
      - ./ssl:/etc/ssl/nginx"
[root@localhost zabbix]#
```

将证书挂载进web镜像的目录

修改完后，直接使用命令：

`docker-compose up -d`

重新创建容器即可生效

- 若是 zabbix 系统直接使用 docker 命令启动（可参考文章：[如何使用 docker 快速部署 zabbix 监控系统](#)），则需重新创建 ZABBIX WEB 容器，如：

先删除旧容器，删除命令：

`docker rm -f 容器名`

然后重新创建容器命令：

```
docker run --name zabbix-web -e ZBX_SERVER_HOST=192.168.75.31 -e ZBX_SERVER_PORT=40051 -e DB_SERVER_HOST=192.168.75.31 -e DB_SERVER_PORT=3306
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
-e MYSQL_DATABASE=zabbix -e MYSQL_USER=root -e MYSQL_PASSWORD=zabbix  
-p 48080:8080 -p 48443:8443 -v /data/zabbix/ssl:/etc/ssl/nginx -d zabbix/zabbix-  
web-nginx-mysql:latest
```

挂载容器参数解释：-v ./ssl:/etc/ssl/nginx

-v —— 指启用卷挂载，格式：宿主机目录(文件) /容器目录(文件)

/data/zabbix/ssl:/etc/ssl/nginx —— 指，将宿主机/data/zabbix/ssl 目录挂载到容器的  
/etc/ssl/nginx 目录

## 访问测试

配置好，重新创建容器后，查看容器日志：

docker logs -f 容器名 (容器 ID)

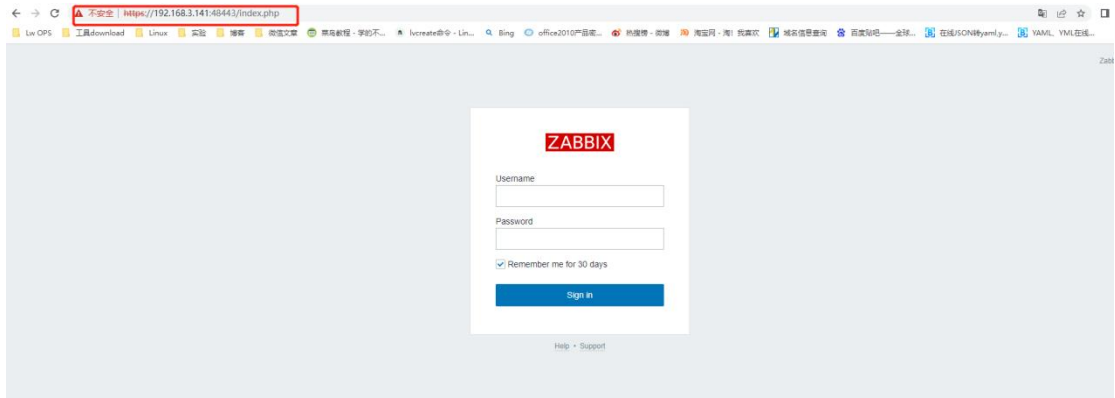
可以看到，已经启用了 SSL

```
[root@localhost zabbix]# docker-compose up -d  
[+] Running 3/3  
# Container zabbix_postgresql_6_test Running  
# Container zabbix_server_6_test Running  
# Container zabbix_web_6_test Started  
[root@localhost zabbix]# docker logs -f zabbix_web_6_test  
** Deploying Zabbix web-interface (Nginx) with PostgreSQL database  
** Using POSTGRES_USER variable from ENV  
** Using POSTGRES_PASSWORD variable from ENV  
*****  
* DB_SERVER_HOST: 192.168.3.141  
* DB_SERVER_PORT: 45432  
* DB_SERVER_DBNAME: zabbix  
* DB_SERVER_SCHEMA: public  
*****  
** Adding Zabbix virtual host (HTTP)  
** Enable SSL support for Nginx  
** Preparing Zabbix frontend configuration file  
#####  
** Executing supervisord  
2023-01-11 10:02:53,694 INFO Included extra file "/etc/supervisor/conf.d/supervisord_zabbix.conf" during parsing  
2023-01-11 10:02:53,694 INFO Included extra file "/etc/supervisor/conf.d/supervisord_zabbix.conf" during parsing  
2023-01-11 10:02:53,697 INFO RPC interface 'supervisor' initialized  
2023-01-11 10:02:53,697 INFO RPC interface 'supervisor' initialized  
2023-01-11 10:02:53,697 INFO supervisord started with pid 1  
2023-01-11 10:02:53,697 INFO supervisord started with pid 1
```

最后到浏览器进行访问测试：

https://IP:PORT

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



如图，配置已完成。

大家好，我是乐乐，专注运维技术研究与分享，关注我，了解更多 Zabbix 知识。如有 Zabbix 问题也可以到[乐维社区](#)留言提问，共同探讨解决方案。

## 二十二、更多.....

# 安装

## 二十七、龙蜥操作系统(Anolis OS) 安装 Zabbix 7.0 LTS 教程

龙蜥操作系统 (Anolis OS) 作为龙蜥社区发行的开源 Linux 发行版，以其稳定、高性能、安全、可靠和 100%兼容 CentOS 8 软件生态的特点，成为众多企业和开发者的首选操作系统。它不仅支持多计算架构，如 X86、ARM、RISC-V 等，还针对云端场景进行了优化，为云上典型场景带来显著的性能提升和故障率降低。

本文将详细介绍如何在龙蜥操作系统 (Anolis OS) 上安装 Zabbix 7.0 LTS，帮助用户搭建起一套高效、稳定的监控系统，实现对系统、网络、应用等各个方面的全面监控，为企业业务的顺利运行提供有力保障。

### 概述

基于 yum 安装 Zabbix 7.0 LTS 版本，本次部署环境：

操作系统：Anolis OS 8.8 (x86 架构)

数据库：PostgreSQL 16.4

中间件：Nginx 1.14.1、PHP 8.0.30

Zabbix：zabbix server 7.0.3 、 zabbix agent 7.0.3

部署时，需要联网访问 yum 源。

附：Anolis 操作系统 下载 地址：

[https://mirrors.aliyun.com/anolis/8.8/isos/GA/x86\\_64/AnolisOS-8.8-x86\\_64-dvd.iso](https://mirrors.aliyun.com/anolis/8.8/isos/GA/x86_64/AnolisOS-8.8-x86_64-dvd.iso)

## 安装 Zabbix 7.0 LTS

安装 zabbix 的 yum 源：（如果是 arm 架构，需要修改 zabbix 的源）

附：自行查找对应操作系统版本 yum 源的方式为：

- 1、通过浏览器访问 <https://repo.zabbix.com/zabbix/7.0/>
- 2、逐层下钻查看对应操作系统版本，名字带 zabbix-release 的文件为 yum 源文件

```
curl -o zabbix-release-latest.el8.noarch.rpm -k https://repo.zabbix.com/zabbix/7.0/rhel/8/x86_64/zabbix-release-latest.el8.noarch.rpm
rpm -Uvh zabbix-release-latest.el8.noarch.rpm
dnf clean all
dnf repolist
```

```
[root@localhost ~]# curl -o zabbix-release-latest.el8.noarch.rpm -k https://repo.zabbix.com/zabbix/7.0/rhel/8/x86_64/zabbix-release-latest.el8.noarch.rpm
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 32504 100 32504 0 0 18426 0 0:00:01 0:00:01 --:--:-- 18436
[root@localhost ~]# ls
anaconda-ks.cfg zabbix-release-latest.el8.noarch.rpm
[root@localhost ~]# rpm -Uvh zabbix-release-latest.el8.noarch.rpm
警告: zabbix-release-latest.el8.noarch.rpm: 头V4 RSA/SHA512 Signature, 密钥 ID b5333005: NOKEY
Verifying... ##### [100%]
准备中... ##### [100%]
正在升级/安装...
1:zabbix-release-7.0-5.el8 ##### [100%]
[root@localhost ~]# dnf clean all
51 个文件已删除
[root@localhost ~]# dnf repolist
仓库名称
AppStream AnolisOS-8 - AppStream
BaseOS AnolisOS-8 - BaseOS
DDE AnolisOS-8 - DDE
Extras AnolisOS-8 - Extras
HighAvailability AnolisOS-8 - HighAvailability
Plus AnolisOS-8 - Plus
PowerTools AnolisOS-8 - PowerTools
kernel-5.10 AnolisOS-8 - Kernel 5.10
zabbix Zabbix Official Repository - x86_64
zabbix-non-supported Zabbix Official Repository (non-supported) - x86_64
zabbix-tools Zabbix Official Repository (tools) - x86_64
[root@localhost ~]#
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

切换 PHP 版本为 8.0:

```
dnf module switch-to php:8.0
```

如出现以下提示:

```
[root@localhost ~]# dnf module switch-to php:8.0
AnolisOS-8 - AppStream                3.0 MB/s | 12 MB  00:03
AnolisOS-8 - BaseOS                  5.2 MB/s | 7.5 MB 00:01
AnolisOS-8 - DSE                     1.9 MB/s | 977 kB 00:08
AnolisOS-8 - Extras                   5.4 kB/s | 2.3 kB 00:00
AnolisOS-8 - HighAvailability         1.3 MB/s | 774 kB 00:00
AnolisOS-8 - Plus                    3.9 MB/s | 3.9 MB 00:00
AnolisOS-8 - PowerTools              2.0 MB/s | 1.6 MB 00:00
AnolisOS-8 - Kernel 5.10             4.4 MB/s | 5.2 MB 00:01
Zabbix Official Repository - x86_64  0.0 B/s | 0 B    00:01
Errors during downloading metadata for repository 'zabbix':
 - Curl error (60): Peer certificate cannot be authenticated with given CA certificates for https://repo.zabbix.com/zabbix/7.0/rhel/8/x86_64/repodata/repomd.xml [SSL certificate problem: certificate is not yet valid]
错误: 为仓库 'zabbix' 下载元数据失败: Cannot download repomd.xml: Cannot download repodata/repomd.xml: All mirrors were tried
```

执行:

```
vi /etc/yum.repos.d/zabbix.repo
```

```
vi /etc/yum.repos.d/zabbix-tools.repo
```

给每个仓库源加上 sslverify=0

**可选**把 gpgcheck=1 改完 gpgcheck=0, 安装时如提示验证不通过, 则可修改

然后继续执行即可

```
[root@localhost ~]# dnf module switch-to php:8.0
Zabbix Official Repository - x86_64 29 kB/s | 74 kB 00:02
Zabbix Official Repository (non-supported) - x86_64 790 B/s | 1.4 kB 00:01
Zabbix Official Repository (tools) - x86_64 905 B/s | 1.6 kB 00:01
你确定要安装吗?

软件包                                架构      版本      仓库      大小
---
启用模块流:
httpd                                2.4
nginx                                 1.14
php                                   8.0

事务概要
-----
确定吗? [y/N]: y
完毕!
[root@localhost ~]#
```

安装 Zabbix 服务端, Web 前端, 客户端

```
dnf install zabbix-server-pgsql zabbix-web-pgsql zabbix-nginx-conf
zabbix-sql-scripts zabbix-selinux-policy zabbix-agent
```



```
验证 : zabbix-web-deps-7.0.3-release1.el8.noarch 48/50
验证 : zabbix-web-pgsql-7.0.3-release1.el8.noarch 49/50
验证 : fping-5.1-1.el8.x86_64 50/50

已安装:
dlopen-libs-2.0.32-3.0.1.an8.x86_64
dejavu-fonts-2.35-7.an8.noarch
fontpackages-filesystem-1.44-22.el8.noarch
gd-2.2.5-7.an8.x86_64
libbrotli-libs-2.1-10.0.1.an8.x86_64
libX11-common-1.7.0-9.an8.noarch
libXpm-3.5.13-10.0.1.an8.x86_64
libjpeg-turbo-2.0.90-7.0.1.an8.x86_64
libtiff-4.4.0-12.0.2.an8.x86_64
libwebp-1.2.0-8.0.1.an8.x86_64
libxml-1.1.32-6.0.1.an8.x86_64
net-snmp-libs-1.5.8-30.0.1.an8.x86_64
nginx-all-modules-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.noarch
nginx-mod-http-image-filter-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.x86_64
nginx-mod-http-xslt-filter-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.x86_64
nginx-mod-stream-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.x86_64
php-bcmath-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
php-fpm-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
php-ldap-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
php-pdo-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
php-xml-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
zabbix-agent-7.0.3-release1.el8.x86_64
zabbix-selinux-policy-7.0.3-release1.el8.x86_64
zabbix-sql-scripts-7.0.3-release1.el8.noarch
zabbix-web-deps-7.0.3-release1.el8.noarch
zabbix-web-pgsql-7.0.3-release1.el8.noarch
zabbix-web-pgsql-7.0.3-release1.el8.noarch

djvuru-fonts-common-2.35-7.an8.noarch
fontconfig-2.13.1-4.an8.x86_64
fping-5.1-1.el8.x86_64
httpd-filesystem-2.4.37-65.0.1.module+an8.9.0+11234+b7617093.2.noarch
libX11-1.7.0-9.an8.x86_64
libXau-1.0.9-8.an8.x86_64
libevent-2.1.8-5.el8.x86_64
libpq-12.11-1.0.1.an8.x86_64
libtool-1.5.26-2.4.6-25.0.3.an8.x86_64
libxcb-1.13.1-1.el8.x86_64
logrotate-3.14.0-0.1.an8.x86_64
nginx-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.x86_64
nginx-filesystem-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.noarch
nginx-mod-http-perk-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.x86_64
nginx-mod-mail-1.1.14.1-9.0.2.module+an8.7.0+10880+b751b443.x86_64
oniguruma-6.8.2-3.0.1.an8.x86_64
php-common-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
php-gd-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
php-mbstring-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
php-pgsql-8.0.30-1.0.1.module+an8.8.0+11122+325ae2b8.x86_64
unixODBC-2.3.7-1.0.1.an8.x86_64
zabbix-nginx-conf-7.0.3-release1.el8.noarch
zabbix-server-pgsql-7.0.3-release1.el8.x86_64
zabbix-web-7.0.3-release1.el8.noarch
zabbix-web-pgsql-7.0.3-release1.el8.noarch

完毕!
[root@localhost ~]#
```

## 安装 postgresql 数据库

参考官网安装步骤：<https://www.postgresql.org/download/linux/redhat/>

先安装 postgresql 源：（如果是 arm 架构，需要修改 zabbix 的源）

```
dnf install -y
https://download.postgresql.org/pub/repos/yum/reporepms/EL-8-x86_64/pgdg-re
dhat-repo-latest.noarch.rpm
```

如果提示 SSL certificate problem: certificate is not yet valid 。同步本地的时间即可。

禁用系统内置 yum 源的 PostgreSQL 安装模块

```
dnf -qy module disable postgresql
```

```
[root@localhost ~]# dnf -qy module disable postgresql
导入 GPG 公钥 0x08B40D20:
Userid: "PostgreSQL RPM Repository <pgsql-pkg-yum@lists.postgresql.org>"
指纹: D4BF 08AE 67A0 B4C7 A1DB CCD2 40BC A2B4 08B4 0D20
来自: /etc/pki/rpm-gpg/PGDG-RPM-GPG-KEY-RHEL
导入 GPG 公钥 0x08B40D20:
Userid: "PostgreSQL RPM Repository <pgsql-pkg-yum@lists.postgresql.org>"
指纹: D4BF 08AE 67A0 B4C7 A1DB CCD2 40BC A2B4 08B4 0D20
来自: /etc/pki/rpm-gpg/PGDG-RPM-GPG-KEY-RHEL
导入 GPG 公钥 0x08B40D20:
Userid: "PostgreSQL RPM Repository <pgsql-pkg-yum@lists.postgresql.org>"
指纹: D4BF 08AE 67A0 B4C7 A1DB CCD2 40BC A2B4 08B4 0D20
来自: /etc/pki/rpm-gpg/PGDG-RPM-GPG-KEY-RHEL
导入 GPG 公钥 0x08B40D20:
Userid: "PostgreSQL RPM Repository <pgsql-pkg-yum@lists.postgresql.org>"
指纹: D4BF 08AE 67A0 B4C7 A1DB CCD2 40BC A2B4 08B4 0D20
来自: /etc/pki/rpm-gpg/PGDG-RPM-GPG-KEY-RHEL
导入 GPG 公钥 0x08B40D20:
Userid: "PostgreSQL RPM Repository <pgsql-pkg-yum@lists.postgresql.org>"
指纹: D4BF 08AE 67A0 B4C7 A1DB CCD2 40BC A2B4 08B4 0D20
来自: /etc/pki/rpm-gpg/PGDG-RPM-GPG-KEY-RHEL
导入 GPG 公钥 0x08B40D20:
Userid: "PostgreSQL RPM Repository <pgsql-pkg-yum@lists.postgresql.org>"
指纹: D4BF 08AE 67A0 B4C7 A1DB CCD2 40BC A2B4 08B4 0D20
来自: /etc/pki/rpm-gpg/PGDG-RPM-GPG-KEY-RHEL
[root@localhost ~]#
```

## 安装 postgresql

```
dnf install -y postgresql16-server
```

```
总计 2.6 MB/s | 18 MB 08:06
导入 GPG 公钥 0x08B40D20: 2.4 MB/s | 2.4 kB 08:08
Userid: "PostgreSQL RPM Repository <pgsql-pkg-yum@lists.postgresql.org>"
指纹: D4BF 08AE 67A0 B4C7 A1DB CCD2 40BC A2B4 08B4 0D20
来自: /etc/pki/rpm-gpg/PGDG-RPM-GPG-KEY-RHEL
导入公钥成功
运行事务检查
事务检查成功
运行事务测试
事务测试成功
运行事务
准备中:
安装: postgresql16-libs-16.4-1PGDG.rhel8.x86_64 1/1
运行脚本: postgresql16-libs-16.4-1PGDG.rhel8.x86_64 1/4
安装: libicu-69.3-2.0.2.amB.x86_64 1/4
运行脚本: libicu-69.3-2.0.2.amB.x86_64 2/4
安装: postgresql16-16.4-1PGDG.rhel8.x86_64 2/4
运行脚本: postgresql16-16.4-1PGDG.rhel8.x86_64 3/4
安装: postgresql16-server-16.4-1PGDG.rhel8.x86_64 4/4
运行脚本: postgresql16-server-16.4-1PGDG.rhel8.x86_64 4/4
安装: postgresql16-server-16.4-1PGDG.rhel8.x86_64 4/4
运行脚本: postgresql16-server-16.4-1PGDG.rhel8.x86_64 4/4
验证: libicu-69.3-2.0.2.amB.x86_64 1/4
验证: postgresql16-16.4-1PGDG.rhel8.x86_64 2/4
验证: postgresql16-libs-16.4-1PGDG.rhel8.x86_64 3/4
验证: postgresql16-server-16.4-1PGDG.rhel8.x86_64 4/4
已安装:
libicu-69.3-2.0.2.amB.x86_64 postgresql16-16.4-1PGDG.rhel8.x86_64 postgresql16-libs-16.4-1PGDG.rhel8.x86_64 postgresql16-server-16.4-1PGDG.rhel8.x86_64
完成!
[root@localhost ~]#
```

## 初始化数据库并启动

```
/usr/pgsql-16/bin/postgresql-16-setup initdb
systemctl enable postgresql-16
systemctl start postgresql-16
```

ss -tnl #可看到 5432 端口

```
[root@localhost ~]# /usr/pgsql-16/bin/postgresql-16-setup initdb
Initializing database ... OK
[root@localhost ~]# systemctl enable postgresql-16
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql-16.service → /usr/lib/systemd/system/postgresql-16.service.
[root@localhost ~]# systemctl start postgresql-16
[root@localhost ~]# ss -tnl
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port      Process
LISTEN     0            128         0.0.0.0:22               0.0.0.0:*
LISTEN     0            288         127.0.0.1:5432           0.0.0.0:*
LISTEN     0            128         [::]:22                  [::]:*
LISTEN     0            288         [::]:5432                [::]:*
```

创建数据库：（创建数据库用户时，自行输入密码并保存记录）

sudo -u postgres createuser --pwprompt zabbix

sudo -u postgres createdb -O zabbix zabbix

```
[root@localhost ~]# sudo -u postgres createuser --pwprompt zabbix
为新角色输入的口令：
再输入一遍：
[root@localhost ~]# sudo -u postgres createdb -O zabbix zabbix
[root@localhost ~]#
```

导入初始架构和数据

zcat /usr/share/zabbix-sql-scripts/postgresql/server.sql.gz | sudo -u zabbix psql

zabbix

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
DELETE 84865
COMMIT
[root@localhost ~]#
```

为 Zabbix server 配置数据库

vi /etc/zabbix/zabbix\_server.conf

DBPassword=创建数据库用户时，自行输入的密码

修改后，结果如下：

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
[root@localhost ~]# grep -Ev "^#|^$" /etc/zabbix/zabbix_server.conf
LogFile=/var/log/zabbix/zabbix_server.log
LogFileSize=0
PidFile=/run/zabbix/zabbix_server.pid
SocketDir=/run/zabbix
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
SNMPTrapperFile=/var/log/snmptrap/snmptrap.log
Timeout=4
LogSlowQueries=3000
StatsAllowedIP=127.0.0.1
EnableGlobalScripts=0
[root@localhost ~]#
```

为 Zabbix 前端配置 PHP

```
vi /etc/nginx/conf.d/zabbix.conf
```

```
#取消注释第 2 行和第 3 行
```

```
server {
    listen      8080;
    server_name example.com;
```

## 防火墙 和 SELinux

防火墙放通端口：（其他端口可以按需放通）

```
firewall-cmd --add-port=8080/tcp --permanent
```

```
firewall-cmd -reload
```

关闭 SELinux：

```
setenforce 0
```

```
vi /etc/sysconfig/selinux    #开机不启动的设置
```

```
修改为 SELINUX=disabled
```

## 启动网页和 zabbix 服务

启动 nginx 和 php 服务:

```
systemctl start php-fpm nginx
```

```
systemctl status php-fpm nginx
```

```
ss -tnl #能看到 8080 端口
```

```
● php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/php-fpm.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-09-03 16:41:49 CST; 11s ago
     Main PID: 20373 (php-fpm)
    Status: "Processes active: 0, idle: 10, Requests: 0, slow: 0, Traffic: 0req/sec"
       Tasks: 11 (limit: 5710)
      Memory: 35.5M
     CGroup: /system.slice/php-fpm.service
            └─20373 php-fpm: master process (/etc/php-fpm.conf)
              └─20379 php-fpm: pool www
                └─20380 php-fpm: pool www
                  └─20381 php-fpm: pool www
                    └─20382 php-fpm: pool www
                      └─20383 php-fpm: pool www
                        └─20384 php-fpm: pool zabbix
                          └─20385 php-fpm: pool zabbix
                            └─20387 php-fpm: pool zabbix
                              └─20388 php-fpm: pool zabbix
                                └─20389 php-fpm: pool zabbix

9月 03 16:41:49 localhost systemd[1]: Starting The PHP FastCGI Process Manager...
9月 03 16:41:49 localhost systemd[1]: Started The PHP FastCGI Process Manager.

● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
  Drop-In: /usr/lib/systemd/system/nginx.service.d
           └─php-fpm.conf
   Active: active (running) since Tue 2024-09-03 16:41:49 CST; 11s ago
     Process: 20377 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
     Process: 20376 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 20374 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Main PID: 20386 (nginx)
       Tasks: 3 (limit: 5710)
```

```
[root@localhost ~]# ss -tnl
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port      Process
LISTEN    0            128         0.0.0.0:22              0.0.0.0:*
LISTEN    0            2048        127.0.0.1:5432         0.0.0.0:*
LISTEN    0            511         0.0.0.0:80             0.0.0.0:*
LISTEN    0            511         0.0.0.0:8080           0.0.0.0:*
LISTEN    0            128         ::::22                 ::::*
LISTEN    0            2048        ::::5432                ::::*
LISTEN    0            511         ::::80                  ::::*
```

启动 zabbix 服务:

```
systemctl start zabbix-server zabbix-agent
```

```
systemctl status zabbix-server zabbix-agent
```



ss -tnl #能看到 10051、10050 端口

```
[root@localhost ~]# systemctl status zabbix-server zabbix-agent
● zabbix-server.service - Zabbix Server
   Loaded: loaded (/usr/lib/systemd/system/zabbix-server.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-09-03 16:46:37 CST; 3s ago
   Process: 20485 ExecStart=/usr/sbin/zabbix_server -c $CONFFILE (code=exited, status=0/SUCCESS)
   Main PID: 20502 (zabbix_server)
     Tasks: 64 (limit: 5710)
    Memory: 80.3M
   CGroup: /system.slice/zabbix-server.service
           └─20502 /usr/sbin/zabbix_server -c /etc/zabbix/zabbix_server.conf
             └─20507 /usr/sbin/zabbix_server: ha manager
               └─20509 /usr/sbin/zabbix_server: service manager #1 started
                 └─20510 /usr/sbin/zabbix_server: configuration syncer [synced configuration in 0.120403 sec, idle 10 sec]
                   └─20514 /usr/sbin/zabbix_server: alert manager #1 started
                     └─20515 /usr/sbin/zabbix_server: alerter #1 started
                       └─20516 /usr/sbin/zabbix_server: alerter #2 started
                         └─20517 /usr/sbin/zabbix_server: alerter #3 started
                           └─20518 /usr/sbin/zabbix_server: preprocessing manager #1 started
                             └─20519 /usr/sbin/zabbix_server: lld manager #1 started
                               └─20520 /usr/sbin/zabbix_server: lld worker #1 started
                                 └─20521 /usr/sbin/zabbix_server: lld worker #2 started
                                   └─20522 /usr/sbin/zabbix_server: housekeeper [startup idle for 30 minutes]
                                     └─20523 /usr/sbin/zabbix_server: timer #1 [updated 0 hosts, suppressed 0 events in 0.001973 sec, idle 22 sec]
                                       └─20524 /usr/sbin/zabbix_server: http poller #1 [got 0 values in 0.000071 sec, idle 5 sec]
                                         └─20525 /usr/sbin/zabbix_server: browser poller #1 [got 0 values in 0.000168 sec, idle 5 sec]
                                           └─20526 /usr/sbin/zabbix_server: discovery manager #1 [processing 0 rules, 0 unsaved checks]
                                             └─20527 /usr/sbin/zabbix_server: history syncer #1 [processed 2 values, 2 triggers in 0.001071 sec, idle 1 sec]
```

```
● zabbix-agent.service - Zabbix Agent
   Loaded: loaded (/usr/lib/systemd/system/zabbix-agent.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-09-03 16:46:37 CST; 3s ago
   Process: 20486 ExecStart=/usr/sbin/zabbix_agentd -c $CONFFILE (code=exited, status=0/SUCCESS)
   Main PID: 20488 (zabbix_agentd)
     Tasks: 13 (limit: 5710)
    Memory: 7.1M
   CGroup: /system.slice/zabbix-agent.service
           └─20488 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
             └─20489 /usr/sbin/zabbix_agentd: collector [idle 1 sec]
               └─20490 /usr/sbin/zabbix_agentd: listener #1 [waiting for connection]
                 └─20491 /usr/sbin/zabbix_agentd: listener #2 [waiting for connection]
                   └─20492 /usr/sbin/zabbix_agentd: listener #3 [waiting for connection]
                     └─20493 /usr/sbin/zabbix_agentd: listener #4 [waiting for connection]
                       └─20494 /usr/sbin/zabbix_agentd: listener #5 [waiting for connection]
                         └─20495 /usr/sbin/zabbix_agentd: listener #6 [waiting for connection]
                           └─20496 /usr/sbin/zabbix_agentd: listener #7 [waiting for connection]
                             └─20497 /usr/sbin/zabbix_agentd: listener #8 [waiting for connection]
                               └─20498 /usr/sbin/zabbix_agentd: listener #9 [waiting for connection]
                                 └─20499 /usr/sbin/zabbix_agentd: listener #10 [waiting for connection]
                                   └─20500 /usr/sbin/zabbix_agentd: active checks #1 [idle 1 sec]
```

```
[root@localhost ~]# ss -tnl
State      Recv-Q      Send-Q           Local Address:Port       Peer Address:Port       Process
LISTEN    0            128              0.0.0.0:22                0.0.0.0:*
LISTEN    0            200              127.0.0.1:5432            0.0.0.0:*
LISTEN    0            128              0.0.0.0:10050            0.0.0.0:*
LISTEN    0            128              0.0.0.0:10051            0.0.0.0:*
LISTEN    0            511              0.0.0.0:80                0.0.0.0:*
LISTEN    0            128              0.0.0.0:8080              0.0.0.0:*
LISTEN    0            128              [::]:22                   [::]:*
LISTEN    0            200              [::]:5432                  [::]:*
LISTEN    0            128              [::]:10050                 [::]:*
LISTEN    0            128              [::]:10051                 [::]:*
LISTEN    0            511              [::]:80                    [::]:*
```

可选：查看 zabbix-server 启动后，日志是否有报错

tail -n 100 /var/log/zabbix/zabbix\_server.log

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

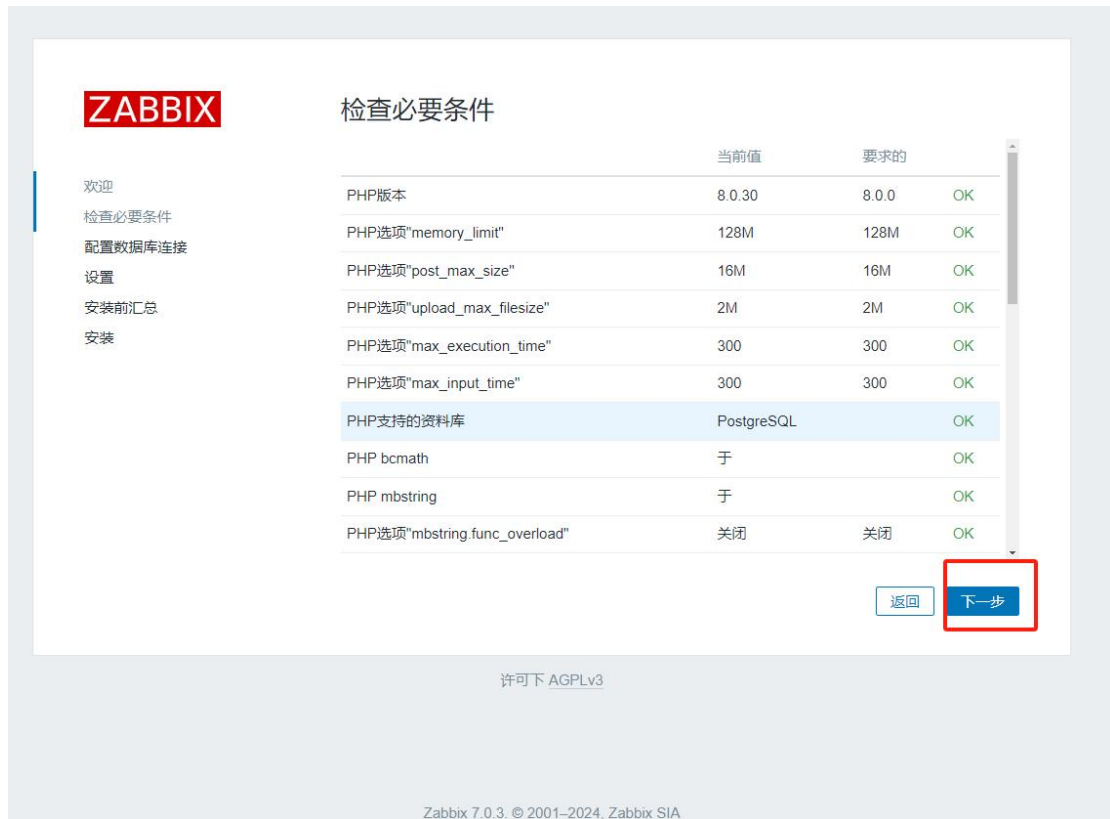
```
[root@localhost ~]# tail -n 100 /var/log/zabbix/zabbix_server.log
20502:20240903:164637.214 Starting Zabbix Server. Zabbix 7.0.3 (revision d93ce022627).
20502:20240903:164637.214 ***** Enabled features *****
20502:20240903:164637.214 SNMP monitoring: YES
20502:20240903:164637.214 IPMI monitoring: YES
20502:20240903:164637.214 Web monitoring: YES
20502:20240903:164637.214 VMware monitoring: YES
20502:20240903:164637.214 SMTP authentication: YES
20502:20240903:164637.214 ODBC: YES
20502:20240903:164637.214 SSH support: YES
20502:20240903:164637.214 IPv6 support: YES
20502:20240903:164637.214 TLS support: YES
20502:20240903:164637.214 *****
20502:20240903:164637.214 using configuration file: /etc/zabbix/zabbix_server.conf
20502:20240903:164637.331 current database version (mandatory/optional): 07000000/07000003
20502:20240903:164637.331 required mandatory version: 07000000
20507:20240903:164637.336 starting HA manager
20507:20240903:164637.364 HA manager started in active mode
20502:20240903:164637.366 server #0 started [main process]
20509:20240903:164637.366 server #1 started [service manager #1]
20510:20240903:164637.367 server #2 started [configuration syncer #1]
20514:20240903:164637.563 server #3 started [alert manager #1]
20515:20240903:164637.567 server #4 started [alerter #1]
20518:20240903:164637.567 server #7 started [preprocessing manager #1]
20520:20240903:164637.568 server #9 started [lld worker #1]
20522:20240903:164637.569 server #11 started [housekeeper #1]
20524:20240903:164637.575 server #13 started [http poller #1]
20526:20240903:164637.580 server #15 started [discovery manager #1]
20528:20240903:164637.580 server #17 started [history syncer #2]
```

## 初始化 zabbix 页面

通过浏览器访问 <http://IP:8080>

根据页面提示的步骤对 zabbix 进行初始化。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



根据实际情况修改需要连接的数据库信息：（数据库主机如果写成 IP 地址可能会导致连接数据库失败，需要对连接数据库的用户进行远程访问授权）

整理 by 乐维社区 (<https://forum.lwops.cn>)



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

**ZABBIX**

### 配置数据库连接

数据库类型: PostgreSQL

数据库主机: localhost

数据库端口: 5432 0 - 使用默认端口

数据库名称: zabbix

数据库概要:

将凭据存储在: 文本 HashiCorp Vault CyberArk Vault

用户: zabbix

密码: .....

数据库 TLS 加密:

验证数据库证书:

返回 下一步

许可下 AGPLv3

Zabbix 7.0.3. © 2001–2024, Zabbix SIA

Zabbix 主机名称 可为空，默认时区选择 UTC+8

**ZABBIX**

### 设置

Zabbix主机名称:

默认时区: (UTC+08:00) Asia/Shanghai

默认主题: 蓝

返回 下一步

许可下 AGPLv3

Zabbix 7.0.3. © 2001–2024, Zabbix SIA

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



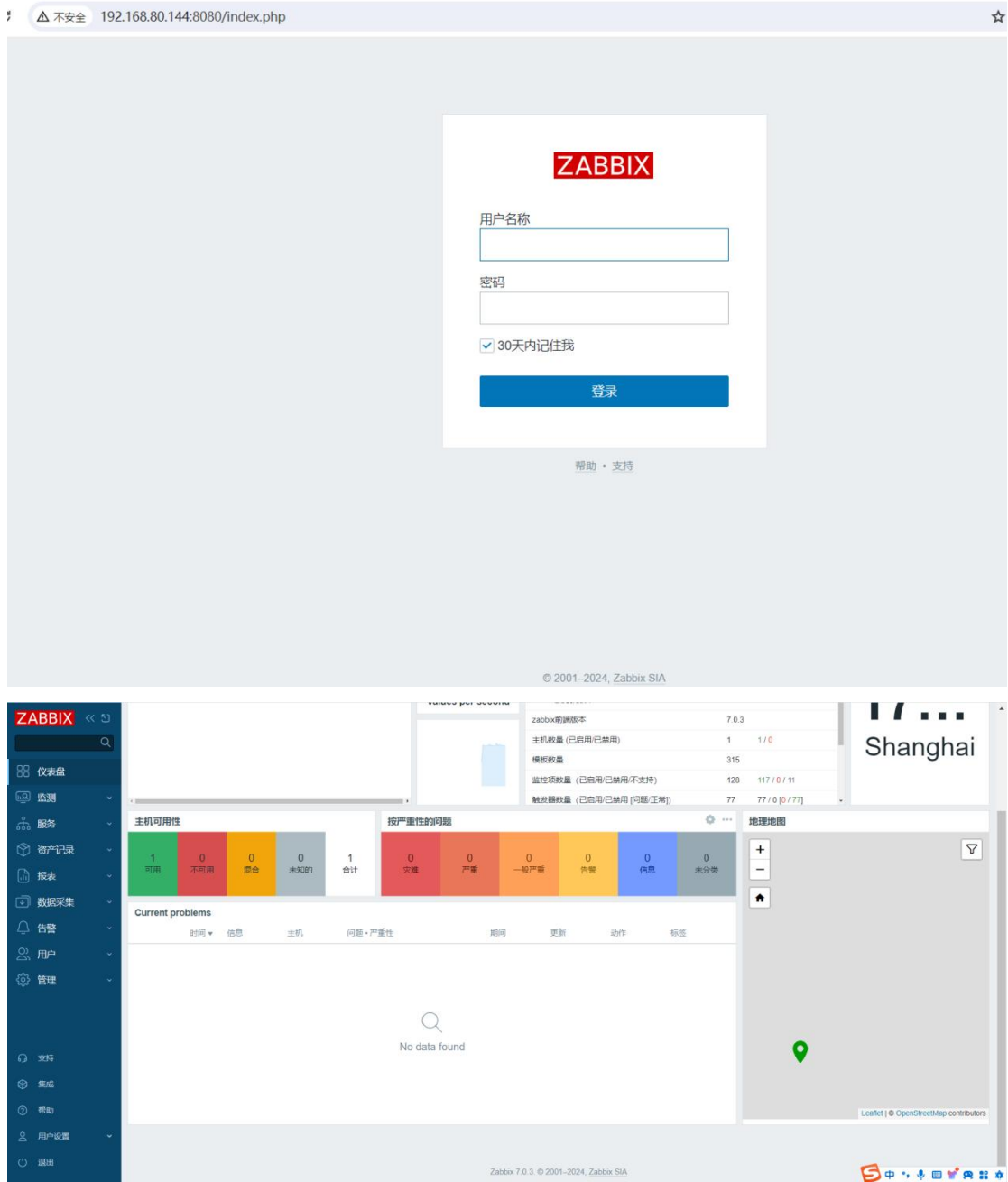
提示的前端配置文件较为重要，如后续需要调整前端连接的数据库信息等，可在改配置文件中进行调整。



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

默认用户名: Admin

默认密码: zabbix



整理 by 乐维社区 (<https://forum.lwops.cn>)

## 二十八、轻松上手 | RockyLinux 9.4 安装及 Zabbix 7.0 部署教程详解

2024 年 6 月 30 日, CentOS 停止更新和维护, 不少企业用户开始寻求新的替代方案。

RockyLinux, 一个开源、社区拥有和管理、免费的企业 Linux 发行版, 提供强大的生产级平台, 可作为 CentOS 停止维护 (改为滚动更新的 Stream 版) 后, RHEL 的下游 Linux 操作系统替代方案, 并继承了原 CentOS 的开源免费特点。

本文介绍了 RockyLinux 9.4 安装以及在该环境中部署 Zabbix 7.0 的详细教程, 可帮助原有的 CentOS 与 Zabbix 用户轻松上手并快速完成系统切换。以下是详细介绍:

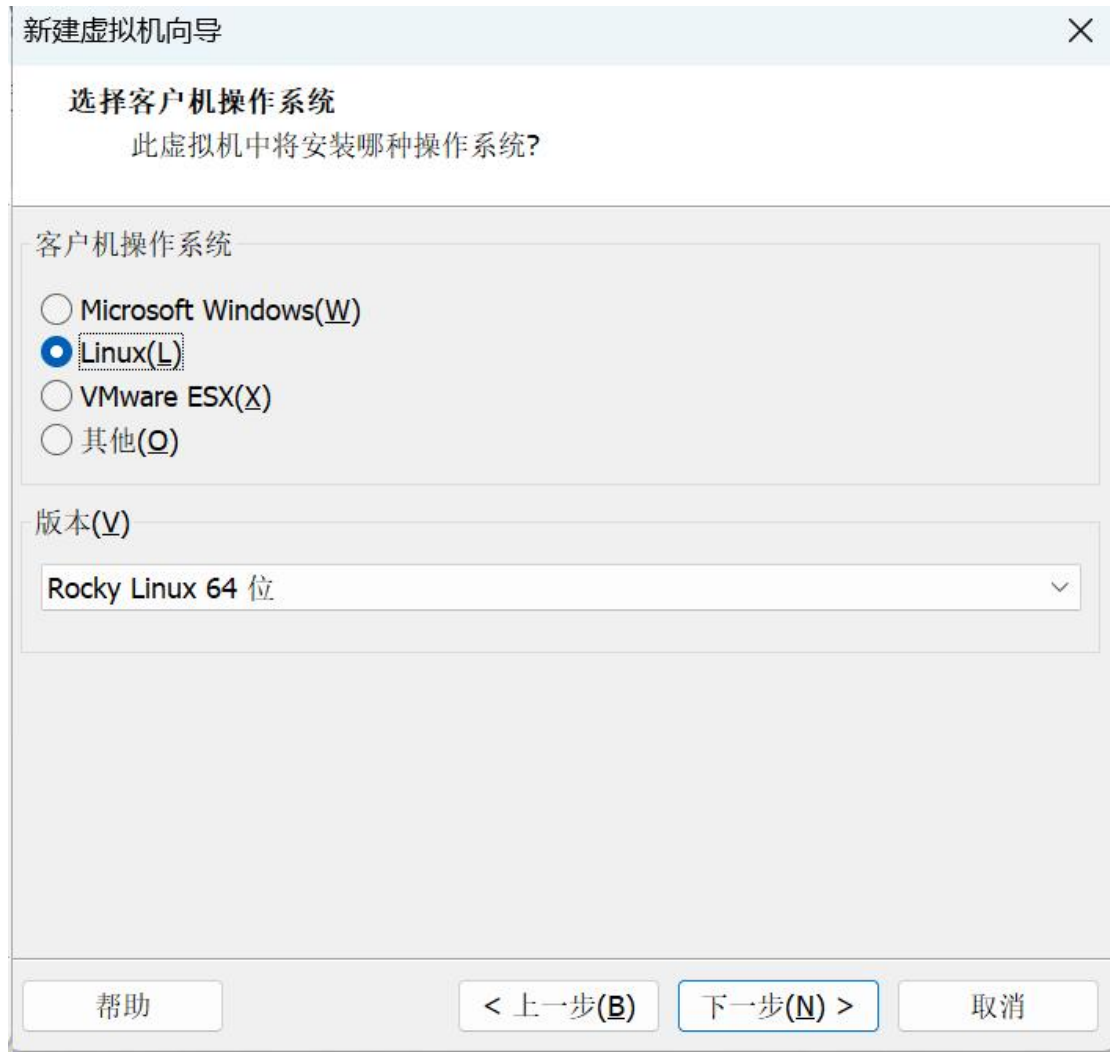
RockyLinux 下载地址: <https://rockylinux.org/zh-CN/download>

版本	版本释义
DVD	完整版
Boot	自定义
Minimal	最小化

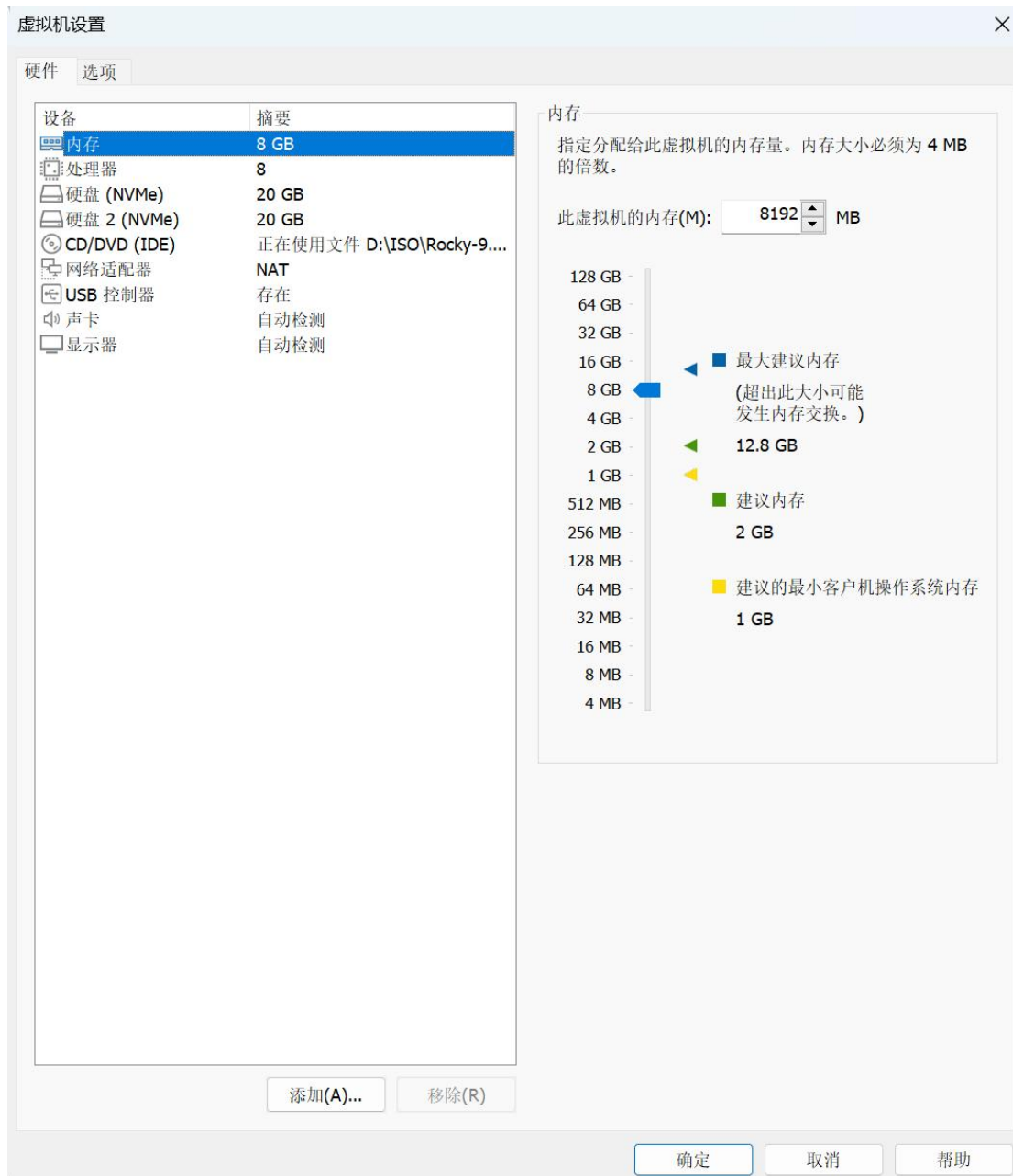
## 一、安装 RockyLinux9.4，选择典型或自定义都可以。



(1) 选择虚拟机操作系统。

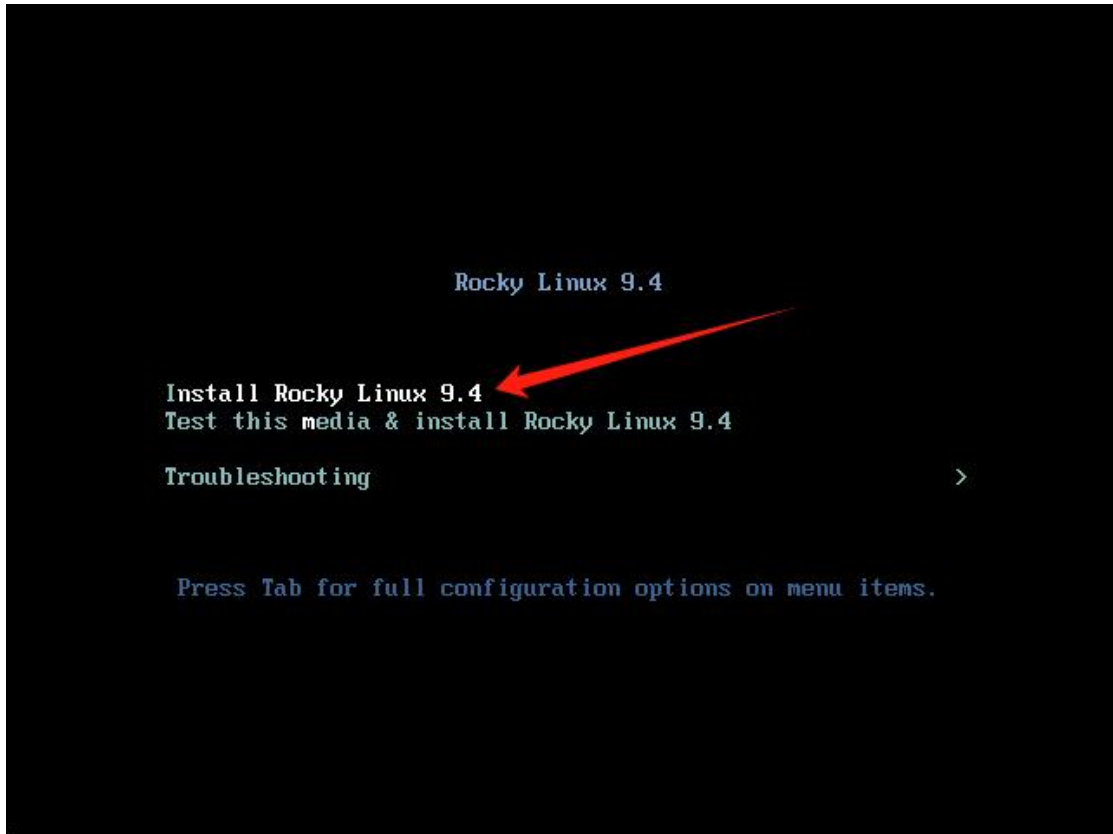


(2) 根据电脑自身配置自行选择。

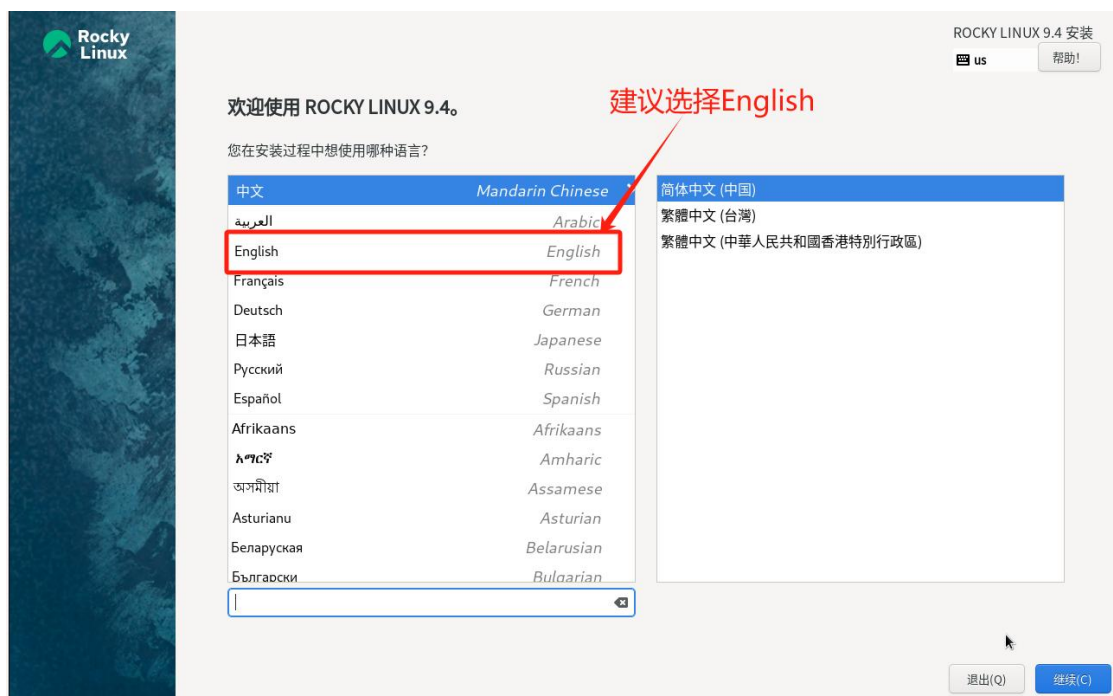


(3) 选择第一个安装 rocky Linux。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



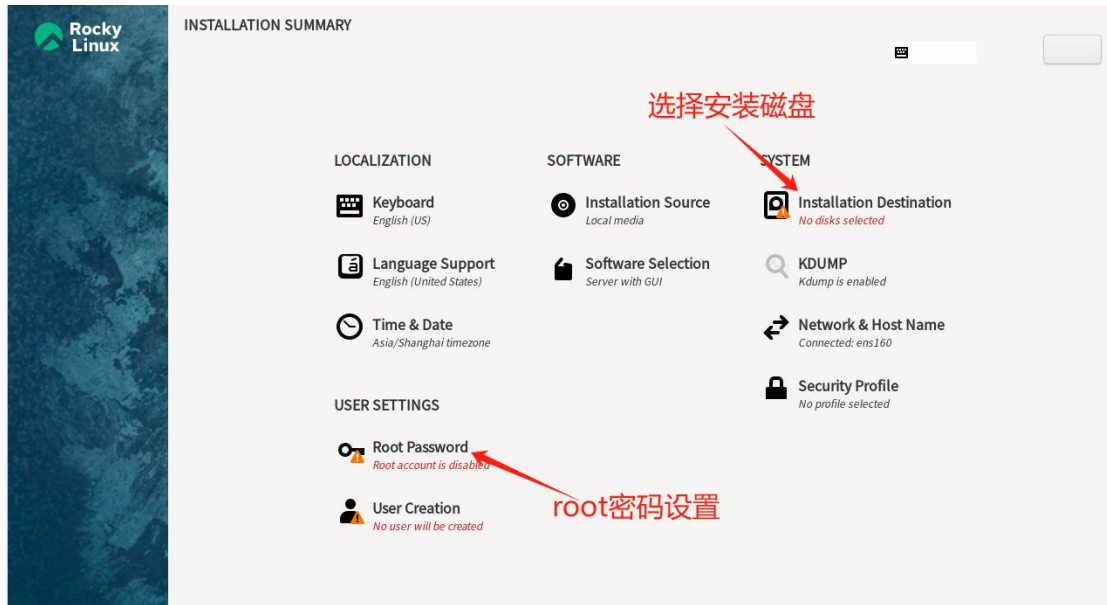
(4) 建议选择英语，一方面可以锻炼英语能力，另一方面会提高软件的兼容性，之后点击 continue。



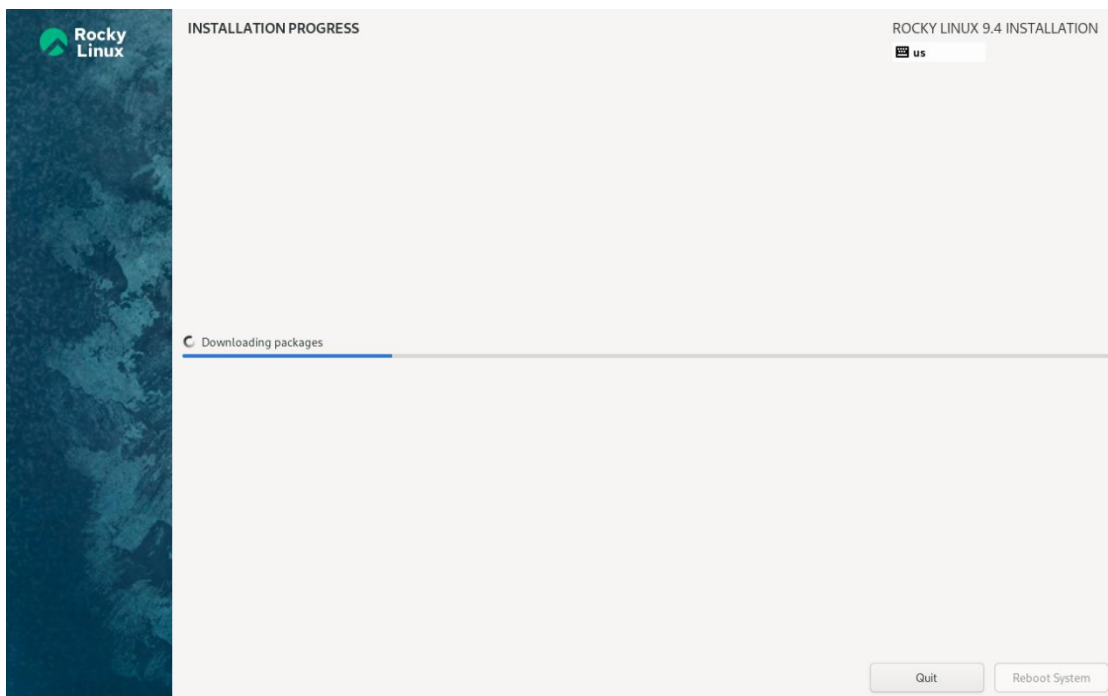
(5) 设置完之后点击 begin installation.



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



(6) 至此系统安装完成。



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



## 二、配置 IP 地址，子网，网关信息：

**注**：以往的 Redhat/Centos 配置 Ip 等信息都是在 `/etc/sysconfig/network-scripts/ifcfg-eth33` 或者是 `ifcfg-eth0`，Rocky Linux 不同在 `/etc/sysconfig/network-scripts` 只有个 `readme-ifcfg-rh.txt` 文件里面会指引配置 Ip 地址的路径信息。

```
NetworkManager stores new network profiles in keyfile format in the
/etc/NetworkManager/system-connections/ ←----- Rocky Linux配置IP地址路径信息

Previously, NetworkManager stored network profiles in ifcfg format
in this directory (/etc/sysconfig/network-scripts/). However, the ifcfg
format is deprecated. By default, NetworkManager no longer creates
new profiles in this format.

Connection profiles in keyfile format have many benefits. For example,
this format is INI file-based and can easily be parsed and generated.

Each section in NetworkManager keyfiles corresponds to a NetworkManager
setting name as described in the nm-settings(5) and nm-settings-keyfile(5)
man pages. Each key-value-pair in a section is one of the properties
listed in the settings specification of the man page.

If you still use network profiles in ifcfg format, consider migrating
them to keyfile format. To migrate all profiles at once, enter:

# nmcli connection migrate

This command migrates all profiles from ifcfg format to keyfile
format and stores them in /etc/NetworkManager/system-connections/.

Alternatively, to migrate only a specific profile, enter:

# nmcli connection migrate <profile_name>:UUID:Bus_path

For further details, see:
* nm-settings-keyfile(5)
* nmcli(1)
...
...
"/etc/sysconfig/network-scripts/readme-ifcfg-ph.txt" 31L, 1244B
1.1 611
```

(1) 根据指引来到 /etc/NetworkManager/system-connections 下面有一个 ens160.nmconnection 文件，这里就是配置 IP 地址等信息的位置。

```
[root@localhost system-connections]# pwd
/etc/NetworkManager/system-connections
[root@localhost system-connections]#
[root@localhost system-connections]#
[root@localhost system-connections]#
[root@localhost system-connections]# ls
ens160.nmconnection
[root@localhost system-connections]#
[root@localhost system-connections]#
[root@localhost system-connections]# _
```

(2) 根据情况自行修改。

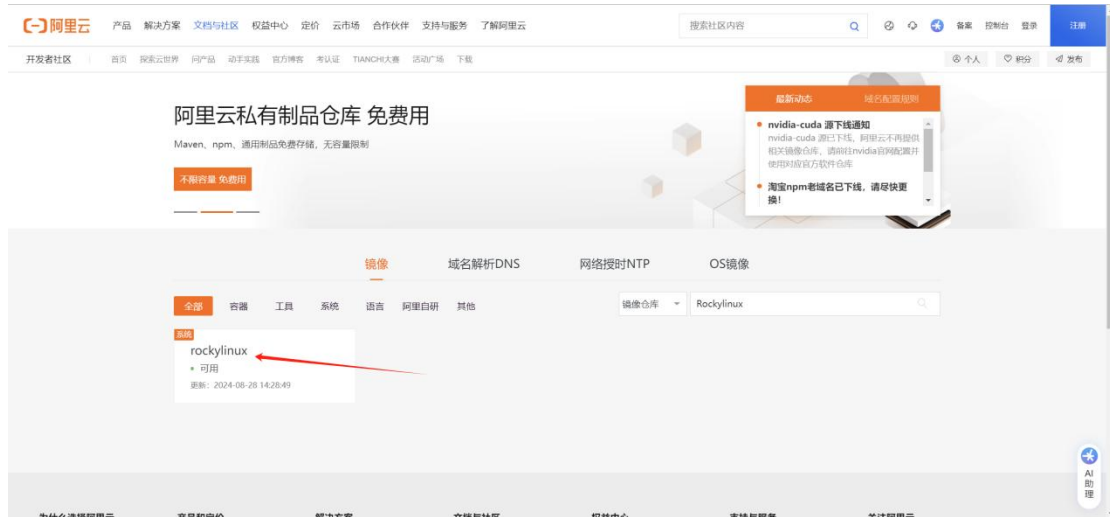
初始配置	修改后配置	备注
[ipv4] method=auto	[ipv4] method=manual address=192.168.10.94/24,192.168.10.2 dns=223.5.5.5	method=auto 是 DHCP 自动获取 IP 地址方式。

	<p>may-fail=false</p>	<p>\$#method=manual 是手动配置 IP 地址方式。</p> <p>\$#Address=Ip/ 子网, 【英文逗号】网关地址</p> <p>\$#dns= 阿里的 dns 地址【114 或 8 都可以】</p> <p>\$# may-fail=false 如果遇到任何错误或失败，系统不会忽略这些错误，而是会停止配置过程并报告错误。</p>
--	-----------------------	---





本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



**注：**修改前备份源文件，图中修改的文件格式为{Rocky-\*.repo}不对需要改为小写，并且杠也要去掉，不然执行修改命令后源不生效。【正确格式参考如下】

yum源修改命令.tx  
t

```
$#sed -e 's|^mirrorlist=|#mirrorlist=g' \  
  
-e  
  
's|^#baseurl=http://dl.rockylinux.org/$contentdir|baseurl=https://mirrors.aliyun.co  
m/rockylinux|g' \  

```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
-i.bak \  
  
/etc/yum.repos.d/rocky*.repo  
  
$#dnf makecache
```

至此系统基础配置都以完成，开始部署 zabbix7.0~

### 三、部署 Zabbix7.0:

【Zabbix 基本概念: <https://www.zabbix.com/documentation/current/zh/>】

(1) 使用 Rpm 安装对应版本的 zabbix 仓库:

```
$# rpm -Uvh  
  
https://repo.zabbix.com/zabbix/7.0/rocky/9/x86\_64/zabbix-release-7.0-2.el9.noarc  
  
h.rpm
```

```
[root@localhost ~]# ls /etc/yum.repos.d/  
rocky-addons.repo rocky-devel.repo rocky-extras.repo rocky.repo zabbix.repo  
[root@localhost ~]#
```

注: 默认是 Zabbix 官方仓库如网络问题无法下载，可更换阿里仓库。





### (3) 安装数据库、初始化:

安装数据库: `$# yum install -y mariadb mariadb-server` “图-1”

初始化: `$# mysql_secure_installation` “图-2”

图-1

```
验证 : perl-parent-1.0.238-460.el9.noarch 80/80
已安装:
checkpolicy-3.6-1.el9.x86_64
mariadb-common-3.10.5.22-1.el9_2.x86_64
mariadb-server-3.10.5.22-1.el9_2.x86_64
perl-AutoLoader-5.74-481.el9.noarch
perl-Class-Struct-0.66-481.el9.noarch
perl-Data-Dumper-2.174-462.el9.x86_64
perl-DynaLoader-1.47-481.el9.x86_64
perl-Exporter-5.74-461.el9.noarch
perl-File-Copy-2.34-481.el9.noarch
perl-File-Stat-1.09-481.el9.noarch
perl-Getopt-Std-1.12-481.el9.noarch
perl-IO-Socket-IP-0.41-5.el9.noarch
perl-MIME-Base64-3.16-4.el9.x86_64
perl-Mozilla-CA-20200520-6.el9.noarch
perl-POSIX-1.94-481.el9.x86_64
perl-Pod-Perldoc-3.28.01-461.el9.noarch
perl-Scalar-List-Utils-4.1.56-461.el9.x86_64
perl-Storable-1.3.21-460.el9.x86_64
perl-Term-ANSIColor-5.01-461.el9.noarch
perl-Text-Tabs+Wrap-2013.0523-460.el9.noarch
perl-base-2.27-481.el9.noarch
perl-interpreter-4.5.32.1-481.el9.x86_64
perl-mro-1.23-481.el9.x86_64
perl-parent-1.0.238-460.el9.noarch
perl-vars-1.05-481.el9.noarch
python3-distro-1.5.0-7.el9.noarch
python3-setuptools-4.4.4-1.el9.x86_64
mariadb-3.10.5.22-1.el9_2.x86_64
mariadb-errmsg-3.10.5.22-1.el9_2.x86_64
mariadb-server-utils-3.10.5.22-1.el9_2.x86_64
perl-B-1.80-481.el9.x86_64
perl-DBD-MariaDB-1.21-16.el9_0.x86_64
perl-Digest-1.19-4.el9.noarch
perl-Encode-4.3.08-462.el9.x86_64
perl-FontT-1.13-481.el9.x86_64
perl-File-Path-2.18-4.el9.noarch
perl-FileHandle-2.03-481.el9.noarch
perl-HTTP-Tiny-0.076-462.el9.noarch
perl-IO-Socket-SSL-2.073-1.el9.noarch
perl-Math-BigInt-1.1.9998.18-460.el9.noarch
perl-NDBM_File-1.15-481.el9.x86_64
perl-PathTools-3.78-461.el9.x86_64
perl-Pod-Simple-1.3.42-4.el9.noarch
perl-SelectSaver-1.02-481.el9.noarch
perl-Symbol-1.08-481.el9.noarch
perl-Term-Cap-1.17-460.el9.noarch
perl-Time-Local-2.1.300-7.el9.noarch
perl-constant-1.33-461.el9.noarch
perl-libnet-3.13-4.el9.noarch
perl-overload-1.31-481.el9.noarch
perl-podlators-1.4.14-460.el9.noarch
policycoreutils-python-utils-3.6-2.1.el9.noarch
python3-libsemaphore-3.6-1.el9.x86_64
python3-setuptools-53.0.0-12.el9_4.1.noarch
mariadb-backup-3.10.5.22-1.el9_2.x86_64
mariadb-gssapi-server-3.10.5.22-1.el9_2.x86_64
mysql-selinux-1.0.10-1.el9.noarch
perl-Carp-1.50-460.el9.noarch
perl-DBI-1.643-9.el9.x86_64
perl-Digest-MD5-2.58-4.el9.x86_64
perl-Errno-1.30-481.el9.x86_64
perl-File-Basename-2.85-481.el9.noarch
perl-File-Temp-1.0.231.100-4.el9.noarch
perl-Getopt-Long-1.2.52-4.el9.noarch
perl-IO-1.43-481.el9.x86_64
perl-IPC-Open3-1.21-481.el9.noarch
perl-Math-Complex-1.59-481.el9.noarch
perl-Net-SSLeay-1.92-2.el9.x86_64
perl-Pod-Escapes-1.1.07-460.el9.noarch
perl-Pod-Usage-4.2.01-4.el9.noarch
perl-Socket-4.2.031-4.el9.x86_64
perl-Sys-Hostname-1.23-481.el9.x86_64
perl-Text-ParseWords-3.30-460.el9.noarch
perl-URI-5.09-3.el9.noarch
perl-if-0.60.800-481.el9.noarch
perl-libs-4.5.32.1-481.el9.x86_64
perl-overloading-0.02-481.el9.noarch
perl-subst-1.03-481.el9.noarch
python3-audit-3.1.2-2.el9.x86_64
python3-policycoreutils-3.6-2.1.el9.noarch
```

图-2 注: 初始化遇到找不到 sock 文件问题, 手动启动数据库服务即可解决。

```
[root@localhost ~]# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2)
Enter current password for root (enter for none):
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2)
Enter current password for root (enter for none):
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2)
Enter current password for root (enter for none):
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2)
Enter current password for root (enter for none):
```

初始化选项:	选择 Y/n
Enter current password for root (enter for	直接回车

none): #: 输入 root 的当前密码 (输入表示无)	
Switch to unix_socket authentication [Y/n] #: 切换到 unix_socket 身份验证[是/否]	y
Change the root password? [Y/n] #: 更改 root 密码? [是/否]	y
Remove anonymous users? [Y/n] #: 删除匿名用户? [是/否]	y
Disallow root login remotely? [Y/n] #: 不允许远程 root 登录? [是/否]	n
Remove test database and access to it? [Y/n] #: 删除测试数据库并访问它? [是/否]	y
Reload privilege tables now? [Y/n] #: 现在重新加载权限表吗? [是/否]	y

```
Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
... skipping.

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
[root@localhost ~]#
```

(4) 导入初始架构和数据:

- 导入前准备工作: 先创建用户与数据库。

<pre>create database zabbix character set utf8mb4 collate utf8mb4_bin;</pre>	创建名为 zabbix 的数据库,并设置字符集为 utf8mb4。
<pre>create user zabbix@localhost identified by 'pwd123';</pre>	创建名为 zabbix 的用户, 并设置其密码为 pwd123。
<pre>grant all privileges on zabbix.* to zabbix@localhost;</pre>	授予 zabbix 用户在 zabbix 数据库上的所有权限。

```
MariaDB [(none)]> create database zabbix character set utf8mb4 collate utf8mb4_bin;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user zabbix@localhost identified by 'pwd123';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> grant all privileges on zabbix.* to zabbix@localhost;
Query OK, 0 rows affected (0.001 sec)
```

- 导入数据: `$# zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql --default-character-set=utf8mb4 -uzabbix -p zabbix .`

- 查看库大小: `SELECT table_schema AS "zabbix",`

```
ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Size (MB)"
```

```
FROM information_schema.TABLES
```

```
GROUP BY table_schema;
```

```
MariaDB [zabbix]> SELECT table_schema AS "zabbix",
-> ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Size (MB)"
-> FROM information_schema.TABLES
-> GROUP BY table_schema;
+-----+-----+
| zabbix | Size (MB) |
+-----+-----+
| information_schema | 0.20 |
| mysql | 3.19 |
| performance_schema | 0.00 |
| zabbix | 60.69 |
+-----+-----+
4 rows in set (0.020 sec)
MariaDB [zabbix]>
```

(5) 接下来到最后配置 Server 和 Nginx:

- Nginx 配置:

`vim /etc/nginx/conf.d/zabbix.conf`

```
server {
#   listen      8080;
#   server_name example.com; 取消注释

    root       /usr/share/zabbix;
    index      index.php;

    location = /favicon.ico {
        log_not_found off;
    }

    location / {
        try_files $uri $uri/ =404;
    }

    location /assets {
        access_log off;
        expires    10d;
    }

    location ~ /\.ht {
        deny       all;
    }

    location ~ /(api|conf|include|locale) {
        deny       all;
        return     404;
    }

    location /vendor {
        deny       all;
    }
}
"/etc/nginx/conf.d/zabbix.conf" 61L, 1978B
```

2, 1 顶端

- Server 配置:

`vim /etc/zabbix/zabbix_server.conf`

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
## Option: DBUser
# Database user.
#
# Mandatory: no
# Default:
# DBUser=

DBUser=zabbix

## Option: DBPassword
# Database password.
# Comment this line if no password is used.
#
# Mandatory: no
# Default:
DBPassword=pwd123

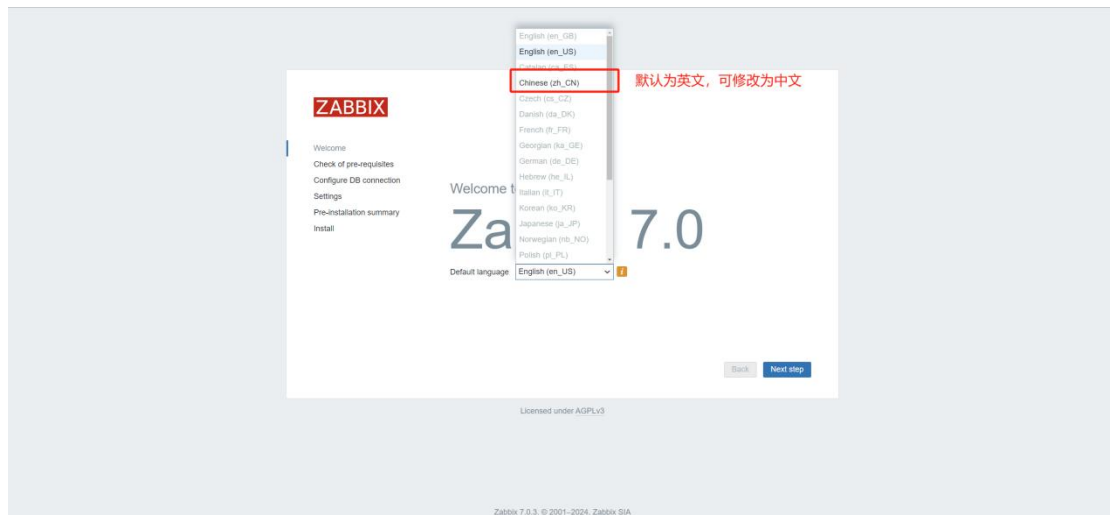
## Option: DBSocket
# Path to MySQL socket.
#
# Mandatory: no
# Default:
# DBSocket=

## Option: DBPort
# Database port when not using local socket.
# If the Net Service Name connection method is used to connect to Oracle database, the port number from the
# tnsnames.ora file will be used. The port number set here will be ignored.
#
# Mandatory: no
# Range: 1024-65535
# Default:
"/etc/zabbix/zabbix_server.conf" 1126L, 29842B
```

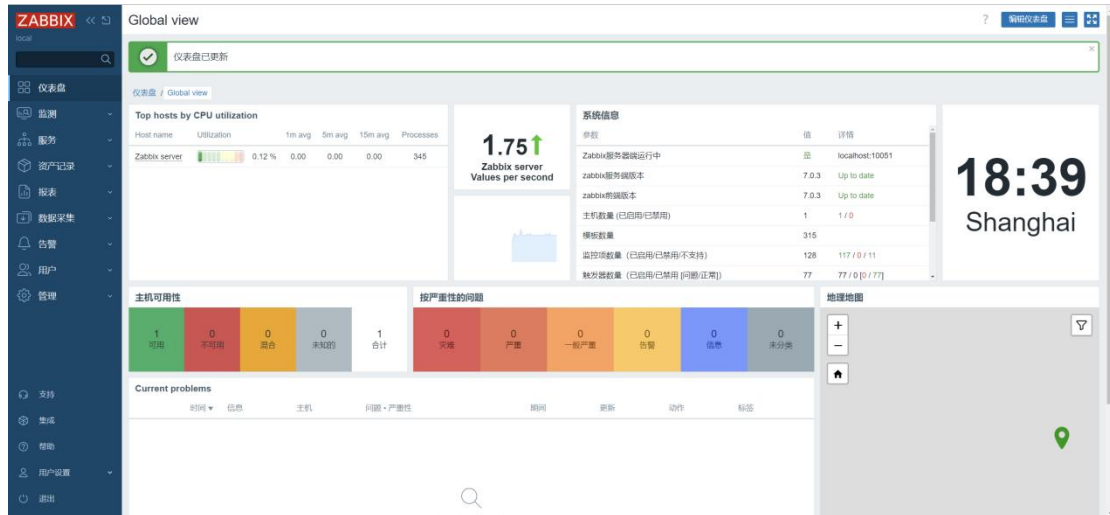
- 启动服务：Server、nginx、Agents（前面已启动过数据库）：

```
[root@localhost ~]# systemctl start zabbix-server.service
[root@localhost ~]# systemctl start zabbix-agent.service
[root@localhost ~]# systemctl start nginx.service
[root@localhost ~]# ss -ltn
State      Recv-Q    Send-Q    Local Address:Port    Peer Address:Port    Process
LISTEN    0          4096      0.0.0.0:10050          0.0.0.0:*
LISTEN    0          4096      0.0.0.0:10051          0.0.0.0:*
LISTEN    0          511       0.0.0.0:80            0.0.0.0:*
LISTEN    0          128       0.0.0.0:22            0.0.0.0:*
LISTEN    0          511       0.0.0.0:8080          0.0.0.0:*
LISTEN    0          4096      [::]:10050            [::]:*
LISTEN    0          4096      [::]:10051            [::]:*
LISTEN    0          511       [::]:80               [::]:*
LISTEN    0          128       [::]:22               [::]:*
LISTEN    0          80        *:3306                 *:*
```

- 访问页面：<http://IP:8080>，默认账密：Admin/zabbix。



本文档为样章，完整版文档请添加乐乐（lerwee）获取



以上就是本期的全部内容。大家好，我是乐乐专注 IT 运维技术研究与分享，更多运维技巧  
欢迎关注乐维社区，更多运维问题也欢迎到乐维社区留言提问。

## 二十九、如何在离线环境中编译安装 Zabbix

### 说明

有时候为了安全，公司的内网环境是不连接外网的，然后又需要针对性的重新编译一个特殊功能的 Zabbix 监控系统，但是相关的依赖的安装是个比较麻烦的问题，要么制作一个本地 yum 源进行依赖安装，要么做一个网络映射，让内网机器可以访问外网。如果既不想搭建 yum 源，又不想做网络映射，那该怎么办？

为解决这个问题。本文将介绍使用 centos 系统来下载编译所需的相关依赖包，然后将依赖包和 Zabbix 源码包一起上传到内网机器，从而离线环境中编译安装 Zabbix。

### 前提条件

确定好内网机器的操作系统版本

系统要能执行 yum 命令

### 下载 rpm 依赖包和 zabbix 源码包

### 确保系统一致性

首先确定好内网机器的操作系统版本。这里我的环境机器系统版本是 centos7.9

```
[root@docke ~]# cat /etc/redhat-release
CentOS Linux release 7.9.2009 (Core)
[root@docke ~]#
```

然后在外网找一台操作系统一致的机器（一定要确保操作系统一致）



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
[root@general-test ~]# cat /etc/redhat-release  
CentOS Linux release 7.9.2009 (Core)  
[root@general-test ~]#
```

## 下载全量依赖包

安装下载命令

```
yum -y install yum-utils
```

下载全量依赖包

文章编译 zabbix 所需的依赖有 gcc gcc-c++ net-snmp-devel libevent-devel OpenIPMI

OpenIPMI-devel openssl-libs mariadb-devel device-mapper rpm libaio\*

执行下载命令：

```
repotrack gcc gcc-c++ net-snmp-devel libevent-devel OpenIPMI OpenIPMI-devel
```

```
openssl-libs mariadb-devel device-mapper rpm libaio*
```

等待下载完成

```
[root@general-test rpm]#  
[root@general-test rpm]# ls | wc -l  
317  
[root@general-test rpm]# cd ..  
[root@general-test ~]# zip -r rpm.zip rpm/
```

然后打包

## 下载 zabbix 源码包

[https://www.zabbix.com/download\\_sources](https://www.zabbix.com/download_sources)

到官网下载需要的版本

## 上传依赖包和源码包到离线服务器

然后将打包的 rpm.zip 包和源码包上传到离线服务器

```
[root@docke zabbix]# ls
rpm.zip zabbix-6.2.6.tar.gz
[root@docke zabbix]#
```

## 安装依赖包

解压依赖包

```
[root@docke zabbix]# ls
rpm.zip zabbix-6.2.6.tar.gz
[root@docke zabbix]# unzip rpm.zip
```

yum -y localinstall ./rpm/\*.rpm

若是出现安安装失败, 或者缺少依赖, 可忽略依赖。命令: yum -y localinstall ./rpm/\*.rpm

--skip-broken

亦或是在下载依赖时补全缺少的依赖。

```
[root@docke zabbix]# ls
rpm rpm.zip zabbix-6.2.6.tar.gz
[root@docke zabbix]# yum -y localinstall rpm/*.rpm
```

安装完成

```
Verifying : x86_64 python-2.7.5-99.el7.x86_64
Verifying : x86_64 python-2.7.5-99.el7.x86_64
Installed:
audit-libs.x86_64 0:4.8.5-4.el7          cpp.x86_64 0:4.8.5-44.el7              crackmapexec.x86_64 0:1.0.0-11.el7      gcc.x86_64 0:4.8.5-44.el7              gcc-c++.x86_64 0:4.8.5-44.el7
gdbm.x86_64 0:1.10-8.el7                 gdbm-devel.x86_64 0:1.10-8.el7          kernel-headers.x86_64 0:3.10.0-1150.81.1.el7  glibc-devel.x86_64 0:2.17-326.el7_9    glibc-devel.x86_64 0:2.17-326.el7_9
glibc-headers.x86_64 0:2.17-326.el7_9    gmp.x86_64 0:6.18-8.el7                libb2-devel.x86_64 0:1.5.3-21-25.el7    keyutils-libs.x86_64 0:1.5.8-3.el7      libccp.x86_64 0:4.8.5-44.el7
libb2.x86_64 0:1.5.3-21-25.el7           libb2-devel.x86_64 0:1.5.3-21-25.el7    libffi.x86_64 0:3.0.13-19.el7          libcurl.x86_64 0:7.54.0-15.el7          libgcc.x86_64 0:4.8.5-44.el7
libbrotli.x86_64 0:1.0.9-1.el7            libbrotli-devel.x86_64 0:1.0.9-1.el7     libidn.x86_64 0:2.32-15.el7            libedit.x86_64 0:3.114-10.el7           libltdl.x86_64 0:2.0.31-14.el7
libbrotli1.x86_64 0:1.0.9-1.el7          libbrotli1-devel.x86_64 0:1.0.9-1.el7   libidn2.x86_64 0:2.32-15.el7           libffi-devel.x86_64 0:3.0.13-19.el7     libltdl-devel.x86_64 0:2.0.31-14.el7
ncurses-libs.x86_64 0:5.9-14.20130511.el7_4  perl-ExtUtils-Install.noarch 0:1.58-299.el7_9  perl-devel.x86_64 4:5.16.3-209.el7_9    perl-ExtUtils-MakeMaker.noarch 0:6.66-3.el7  perl-ExtUtils-Manifest.noarch 0:1.61-244.el7
perl.x86_64 0:5.22.17.el7                perl-devel.noarch 0:5.16.3-209.el7_9    perl-ExtUtils-MakeMaker.noarch 0:6.66-3.el7  pyparsing.noarch 0:1.5.6-9.el7
sqlite.x86_64 0:3.17.1-8.el7_7.1         systemtap-sdt-devel.x86_64 0:4.0-13.el7  systemtap-sdt-devel.x86_64 0:4.0-13.el7  pyparsing.noarch 0:1.5.6-9.el7
readline.x86_64 0:6.2-11.el7

Updated:
bash.x86_64 0:4.2.46-35.el7_9             binutils.x86_64 0:2.27-44.base.el7_9.1  ca-certificates.noarch 0:2022.2.54-74.el7_9  corosync.x86_64 0:6.22-24.el7_9.2      expat.x86_64 0:2.1.0-15.el7_9          expat.x86_64 0:2.1.0-15.el7_9
glibc.x86_64 0:2.17-326.el7_9             glibc-common.x86_64 0:2.17-326.el7_9     gcrp.x86_64 0:1.5-11.el7_9              kpartx.x86_64 0:0.4.9-136.el7_9         nspr.x86_64 0:4.34.0-3.1.el7_9
nss.x86_64 0:3.79.0-4.el7_9               nss.x86_64 0:3.79.0-4.el7_9              nss-softoken.x86_64 0:3.79.0-4.el7_9     nss-softoken.x86_64 0:3.79.0-4.el7_9    nss-softoken-freebl.x86_64 0:3.79.0-4.el7_9
nss-softoken-freebl.x86_64 0:3.79.0-4.el7_9  nss-sysinit.x86_64 0:3.79.0-4.el7_9      nss-tools.x86_64 0:3.79.0-4.el7_9       python.x86_64 0:2.7.5-92.el7_9          python.x86_64 0:2.7.5-92.el7_9
python-libs.x86_64 0:2.7.5-92.el7_9        tzdata.noarch 0:2022g-1.el7              xz.x86_64 0:5.2.2-2.el7_9                xz-libs.x86_64 0:5.2.2-2.el7_9          xz-libs.x86_64 0:5.2.2-2.el7_9
zlib.x86_64 0:1.2.7-20.el7_9
```

## 开始编译 zabbix

### 编译检查

解压 zabbix 源码包, 并开始编译检查

```
./configure --prefix=/data/zabbix --enable-server --enable-agent --with-mysql
--with-net-snmp --with-libcurl --with-libxml2 --with-openipmi --enable-ipv6
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
[root@docker zabbix]# ls
rpm rpm.zip zabbix-6.2.6.tar.gz
[root@docker zabbix]# tar -zxf zabbix-6.2.6.tar.gz
[root@docker zabbix]#
[root@docker zabbix]# ls
rpm rpm.zip zabbix-6.2.6 zabbix-6.2.6.tar.gz
[root@docker zabbix]# cd zabbix-6.2.6/
[root@docker zabbix-6.2.6]# ls
aclocal.m4 bin ChangeLog conf config.sub configure.ac database include install-sh Makefile.am man missing README src
AUTHORS build compile config.guess configure COPYING decomp INSTALL m4 Makefile.in misc NEWS sass ui
[root@docker zabbix-6.2.6]# ./configure --prefix=/data/zabbix --enable-server --enable-agent --with-mysql --with-net-snmp --with-libcurl --with-libxml2 --with-openssl --enable-ipv6
```

结果无问题，开始编译安装

make -j2 && make install

```
Modules:                               no
Linker flags:                           -rdynamic
Libraries:                               -lz -lpthread -lcurl -lm -ldl -lresolv -lpcrc
Configuration file:                     /data/zabbix/etc/zabbix_agentd.conf
Modules:                                 /data/zabbix/lib/modules

Enable agent 2:                          no
Enable web service:                      no

Enable Java gateway:                    no

LDAP support:                            no
IPv6 support:                             yes

*****
*                Now run 'make install'                *
*                                                                 *
*                Thank you for using Zabbix!            *
*                <http://www.zabbix.com>                *
*****

[root@lwops zabbix-6.2.6]# make -j2 && make install
```

## 编译完成

```
make[1]: Entering directory `/tmp/zabbix-6.2.6/man'
make[2]: Entering directory `/tmp/zabbix-6.2.6/man'
make[2]: Nothing to be done for `install-exec-am'.
/usr/bin/mkdir -p /data/zabbix/share/man/man1
/usr/bin/install -c -m 644 'zabbix_get.man' '/data/zabbix/share/man/man1/zabbix_get.1'
/usr/bin/install -c -m 644 'zabbix_sender.man' '/data/zabbix/share/man/man1/zabbix_sender.1'
/usr/bin/mkdir -p /data/zabbix/share/man/man8
/usr/bin/install -c -m 644 'zabbix_agentd.man' '/data/zabbix/share/man/man8/zabbix_agentd.8'
/usr/bin/install -c -m 644 'zabbix_server.man' '/data/zabbix/share/man/man8/zabbix_server.8'
make[2]: Leaving directory `/tmp/zabbix-6.2.6/man'
make[1]: Leaving directory `/tmp/zabbix-6.2.6/man'
Making install in misc
make[1]: Entering directory `/tmp/zabbix-6.2.6/misc'
make[2]: Entering directory `/tmp/zabbix-6.2.6/misc'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/tmp/zabbix-6.2.6/misc'
make[1]: Leaving directory `/tmp/zabbix-6.2.6/misc'
make[1]: Entering directory `/tmp/zabbix-6.2.6'
make[2]: Entering directory `/tmp/zabbix-6.2.6'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/tmp/zabbix-6.2.6'
make[1]: Leaving directory `/tmp/zabbix-6.2.6'
[root@lwops zabbix-6.2.6]#
```

## 添加系统命令

复制启动脚本:

```
cp -ra ./misc/init.d/fedora/core/* /etc/init.d/
```

修改脚本路径:

```
vim /etc/init.d/zabbix_server
```

```
vim /etc/init.d/zabbix_agentd
```

将里面的:

```
BASEDIR=/usr/local
```

修改为:

```
BASEDIR=/data/zabbix
```

完成后便可使用系统命令来启动或停止 zabbix

```
service zabbix-server start/stop/status/restart
```

```
service zabbix-agentd start/stop/status/restrat
```

或

```
systemctl start/stop/status/restart zabbix_server
```

```
systemctl start/stop/status/restart zabbix_agentd
```

## 三十、更多.....

# 告警

## 三十七、Zabbix 事件告警监控：如何实现对相同部件触发器告警及恢复的强关联

有一定 Zabbix 使用经验的小伙伴可能会发现，接收告警事件时，其中可能包含着大量不同的部件名，同一部件的事件在逻辑上具有很强关联性，理论上应保持一致的告警/恢复状态，但 Zabbix 默认并未对它们进行关联，直接后果是运维人员只能进行大量重复操作，进而对部件的状态进行校正。

实际上，虽然 Zabbix 默认未对相同部件进行关联，但却可以通过手动配置实现关联操作。本文将深入探讨 Zabbix 在处理相同部件触发器的告警和恢复过程中的强关联机制。通过这种机制，我们可以确保一旦触发器状态发生变化，相关的告警能够被准确触发，并且在问题解决后，告警状态能够及时恢复，从而避免无效告警的干扰和资源的浪费。

例如，当前监控中有对硬件设备事件采集监控项（数据如下），要对其配置触发器告警，但是每个事件中包含告警对应的部件名，希望对相同部件的告警实现告警-恢复事件的自动关联。

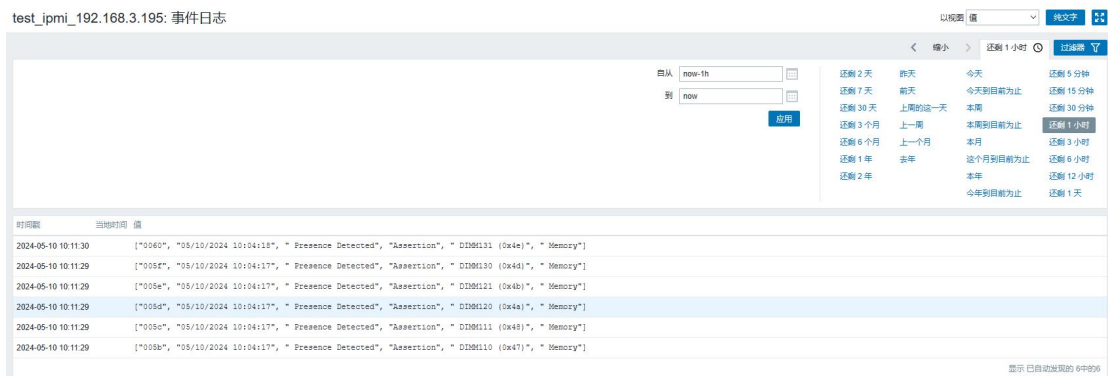


图 1

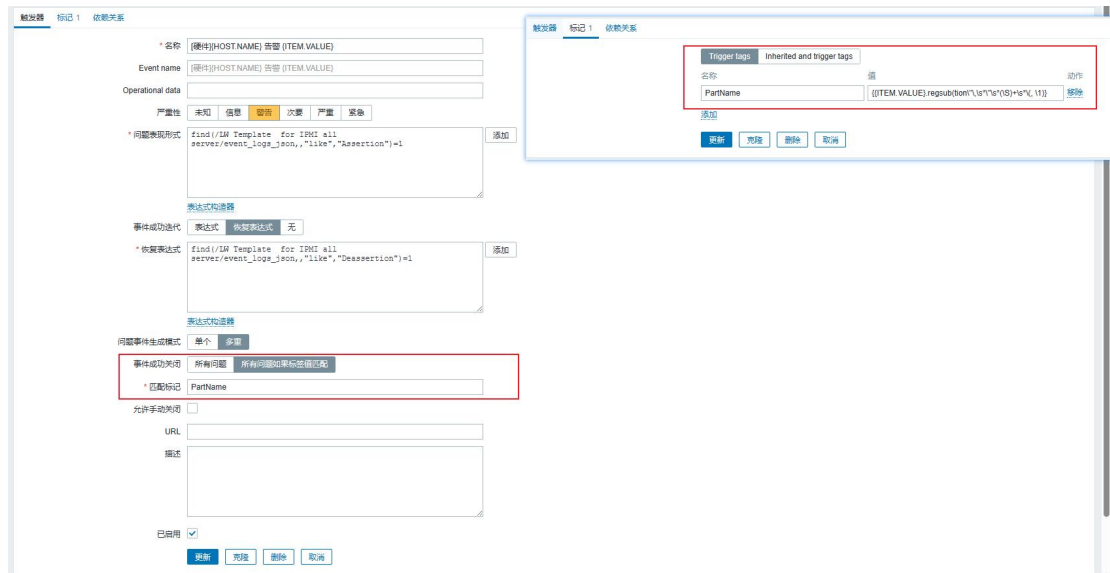
数据字段解析：

监控取值中，第 4 位为告警状态，"Assertion"表示告警产生，"Deassertion"表示告警恢复。

## 配置思路：

### 方法一(建议)：

参考方法一中基础配置，额外补充事件匹配-标签



标记中“值”参数支持写法如下

`{{ITEM.VALUE}.regex(pattern, output)}`

`{{ITEM.VALUE}.iregex(pattern, output)}`

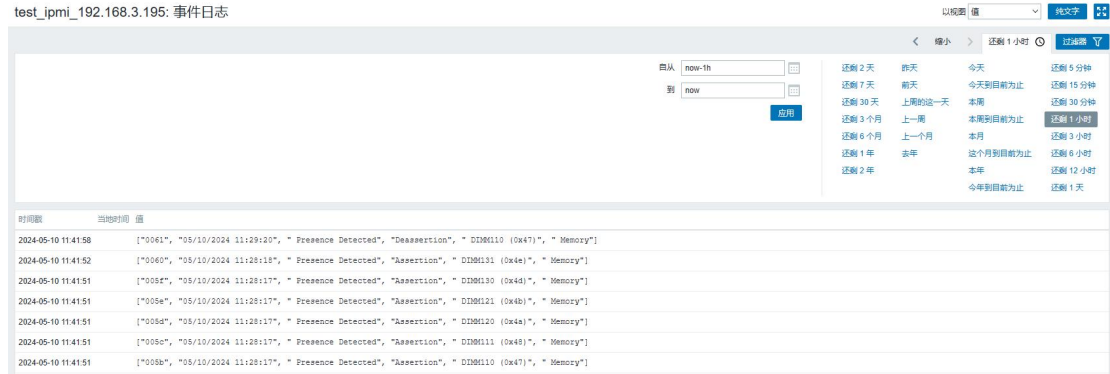
`{{#LLDMACRO}.regex(pattern, output)}`

`{{#LLDMACRO}.iregex(pattern, output)}`

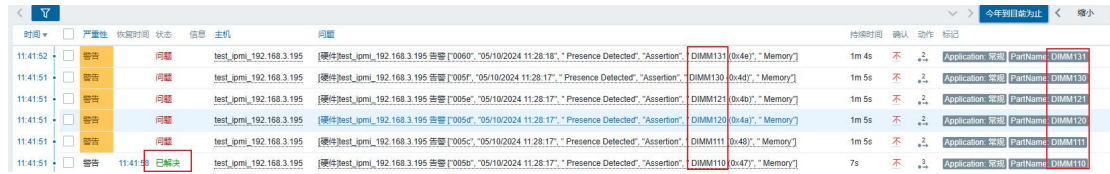
图中示例写法：

```
{{ITEM.VALUE}.regsub("tion\\\\"s*"s*(\S+)\s*(\1))
```

测试数据如下：



测试结果：



从结果中可以看到，部件名称被正则截取到标记中。

同时，只有 DIMM110 是既存在“ Assertion”记录，又存在“ Deassertion”记录的，所以只有 DIMM110 部件的告警是恢复了的。

## 优点：

1. 配置简单，仅配置一条即可
2. 告警事件不遗漏，多个部件告警信息，则产生多个告警事件
3. 可以实现单个部件的告警、恢复记录的关联，不会因为其他此部件的恢复记录，触发其他部件告警的恢复操作

## 缺点:

1. 配置逻辑较复杂，涉及标记、正则、内置宏等多方面

## 方法二:

配置单个触发器，如下图 2

- ① 将告警最新值带进告警标题中，区分具体告警部件等信息。
- ② 检测到关键字"Assertion"则告警
- ③ 检测到关键字"Deassertion"则恢复
- ④ 问题事件生成模式：多重：触发器未恢复的情况下可以多次产生告警；单个：多次规则

匹配，如果已经产生的告警为恢复的情况下，不会重新产生新告警。效果如下图 3



## 触发器

所有模板 / 硬件IPMI[通用] 监控项 9 触发器 3 图形 1 仪表盘 自动发现规则 7 Web 场景

触发器 标记 依赖关系

\* 名称 [硬件]{HOST.NAME} 告警 {ITEM.VALUE} ①

Event name [硬件]{HOST.NAME} 告警 {ITEM.VALUE}

Operational data

严重性 未知 信息 警告 次要 严重 紧急

\* 问题表现形式 find(/LW Template for IPMI all server/event\_logs\_json,, "like" "Assertion")=1 ② 添加

表达式构造器

事件成功迭代 表达式 恢复表达式 无

\* 恢复表达式 find(/LW Template for IPMI all server/event\_logs\_json,, "like" "Deassertion")=1 ③ 添加

表达式构造器

问题事件生成模式 单个 多重 ④

事件成功关闭 所有问题 所有问题如果标签值匹配

允许手动关闭

URL

描述

已启用

更新 克隆 删除 取消

图 2

时间	严重性	依赖时间	状态	信息	主机	问题	持续时间	确认	动作	标记
10:20:26	警告		问题		test_ipmi_192.168.3.195	[硬件]test_ipmi_192.168.3.195 警告 [0069], "05/10/2024 10:04:18"; "Presence Detected", "Assertion", "DIMM131 (0x44)", "Memory"]	2m 45s	否	🔍	Application 帮助
10:20:25	警告		问题		test_ipmi_192.168.3.195	[硬件]test_ipmi_192.168.3.195 警告 [0059], "05/10/2024 10:04:17"; "Presence Detected", "Assertion", "DIMM130 (0x4d)", "Memory"]	2m 46s	否	🔍	Application 帮助
10:20:25	警告		问题		test_ipmi_192.168.3.195	[硬件]test_ipmi_192.168.3.195 警告 [005e], "05/10/2024 10:04:17"; "Presence Detected", "Assertion", "DIMM121 (0x4b)", "Memory"]	2m 46s	否	🔍	Application 帮助
10:20:25	警告		问题		test_ipmi_192.168.3.195	[硬件]test_ipmi_192.168.3.195 警告 [005d], "05/10/2024 10:04:17"; "Presence Detected", "Assertion", "DIMM120 (0x4a)", "Memory"]	2m 46s	否	🔍	Application 帮助
10:20:25	警告		问题		test_ipmi_192.168.3.195	[硬件]test_ipmi_192.168.3.195 警告 [005c], "05/10/2024 10:04:17"; "Presence Detected", "Assertion", "DIMM111 (0x49)", "Memory"]	2m 46s	否	🔍	Application 帮助
10:11:29	警告		问题		test_ipmi_192.168.3.195	[硬件]test_ipmi_192.168.3.195 警告 [005b], "05/10/2024 10:04:17"; "Presence Detected", "Assertion", "DIMM110 (0x47)", "Memory"]	11m 42s	否	🔍	Application 帮助

图 3

## 优点:

1. 配置简单，仅配置一条即可
2. 告警事件不遗漏，多个部件告警信息，则产生多个告警事件

## 缺点:

1. 一个部件告警恢复，则其他部件告警一并恢复，如图 4 —— 仅 DIMM131 部件恢复，其他部件的告警也被恢复了，其实并没有

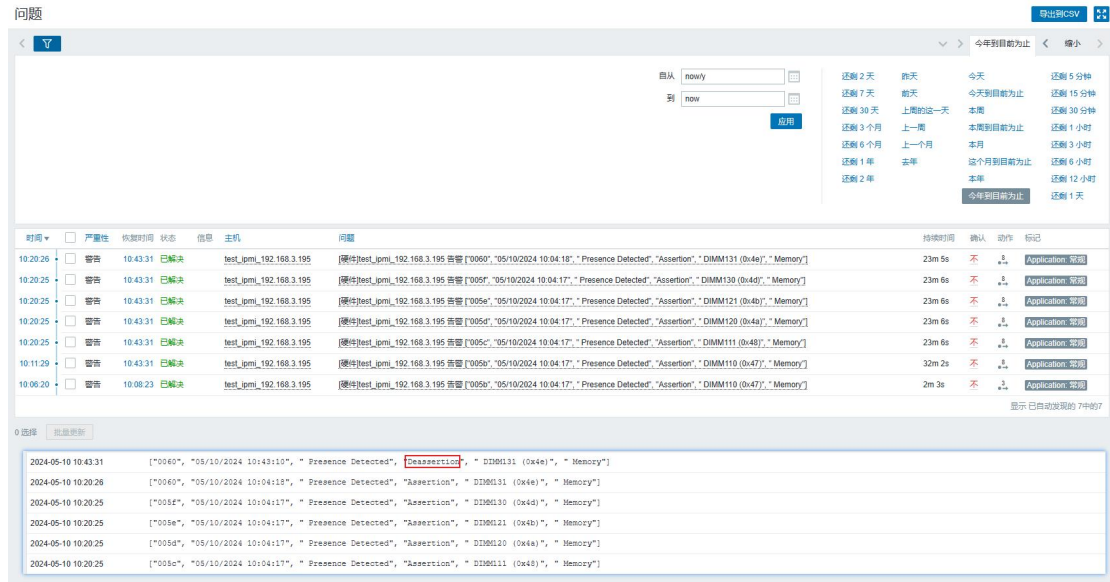


图 4

## 方法三:

对每个部件添加一个触发器 (如下图 2) :

DIMM110 部件告警触发器:

触发表达式: 监控值包含 DIMM110 且 包含关键字 "Assertion";

恢复表达式: 监控值包含 DIMM110 且 包含关键字 "Deassertion";

反复操作, 添加 DIMM111、DIMM120、DIMM121 等多个触发器。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

触发器 标记 依赖关系

\* 名称: {硬件}{HOST.NAME} DIMM110 部件告警

Event name: {硬件}{HOST.NAME} DIMM110 部件告警

Operational data:

严重性: 未知 信息 警告 次要 严重 紧急

\* 问题表现形式: find(/LW Template for IPMI all server/event\_logs\_json,, "like", "DIMM110")=1 and find(/LW Template for IPMI all server/event\_logs\_json,, "like", "Assertion")=1 添加

表达式构造器

事件成功迭代: 表达式 恢复表达式 无

\* 恢复表达式: find(/LW Template for IPMI all server/event\_logs\_json,, "like", "DIMM110")=1 and find(/LW Template for IPMI all server/event\_logs\_json,, "like", "Deassertion")=1 添加

表达式构造器

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

允许手动关闭:

URL:

描述:

已启用:

更新 克隆 删除 取消

图 2

## 优点:

1. 可以实现单个部件的告警、恢复记录的关联，不会因为其他此部件的恢复记录，触发其他部件告警的恢复操作

## 缺点:

1. 配置工作量巨大，每个部件都需要定义一个对应触发器
2. 可能会丢失、遗漏告警，因为可能部件关键字未加入触发器中

## 结论:

上述三种方法可以看出，逻辑上方法二、方法三更加简单明了，但是皆有不满足场景需求的情况；方法一则更贴合场景需求，且善用触发器的标记功能，也更利于监控平台的维护管理。

参考该第一方法，可延伸较多场景，如日志事件告警恢复 ID 关联、snmptrap 端口 up\down 数据告警关联、硬件事件相同部件名告警恢复关联、远程登入登出记录关联等。

以上就是这一期 Zabbix 技术分享。大家好，我是乐乐，专注运维技术研究与分享，关注我学习更多 Zabbix 使用技巧，更多问题也欢迎到乐维社区进行留言。

## 三十八、zabbix 监控进阶：如何分时段设置不同告警阈值 (多阈值告警)

在生产环境中，企业的业务系统状态并不是一成不变的。在业务高峰时段，如节假日、促销活动或特定时间段，系统负载和用户访问量会大幅增加，此时可能需要设置更高的告警阈值来适应更高的负载，反之，低谷期则要将告警阈值调低。

实践中，针对不同的业务状态调高或调低告警阈值，可以对 zabbix 配置多个触发器，设定在不同的时间段生效来实现。本文将以配置两个时段为例，讲解如何分时段设置不同告警阈值。

### 1. 创建触发器

触发器

所有主机 / 监控系统-Server 已启用 [ZBX] 监控项 140 触发器 61 图形 19 自动发现规则 4 Web 场景

创建触发器

主机群组 在此输入搜索 选择

主机 监控系统-Server 在此输入搜索 选择

名称

严重性  未知  警告  严重  
 信息  次要  紧急

状态  所有  正常  未知的

状态  所有  已启用  停用的

值  所有  正常  问题

标记 与或 (默认) 或

标记 包含 值 移除

添加

Inherited  所有  是  不

已发现的  所有  是  不

With dependencies  所有  是  不

应用 重设

<input type="checkbox"/>	严重性	值	名称	Operational data	表达式	状态	信息	标记
<input type="checkbox"/>	次要	正常	log-test		<code>nodata(/tops-server/logctl/tmp/test-*,...,skip,1m)=0</code>	已启用		
<input type="checkbox"/>	警告	正常	IO[主动]:[主机][HOSTNAME]——持续10分钟磁盘写速度达到500M B/s		<code>问题: count(/tops-server/vfs.dev.write[_sps]#10,"ge","500000000")=10 恢复: count(/tops-server/vfs.dev.write[_sps]#10,"ge","500000000")=0</code>	已启用	defaultTag	nzx
<input type="checkbox"/>	警告	正常	IO[主动]:[主机][HOSTNAME]——持续10分钟磁盘读速度达到500M B/s		<code>问题: count(/tops-server/vfs.dev.read[_sps]#10,"ge","500000000")=10 恢复: count(/tops-server/vfs.dev.read[_sps]#10,"ge","500000000")=0</code>	已启用	defaultTag	nzx
<input type="checkbox"/>	信息	正常	监控系统模板[主采集]:[操作系统][HOSTNAME]/etc/passwd密码文件发生变更		<code>(last(/tops-server/vfs.file.cksum[etc/passwd]#1)-&gt;last(/tops-server/vfs.file.cksum[etc/passwd]#2))&gt;0</code>	已启用	defaultTag	
<input type="checkbox"/>	严重	正常	监控系统模板[主采集]:[操作系统][HOSTNAME]CPU使用率持续10分钟大于95%		<code>问题: min(/tops-server/system.cpu.util[Usage],10m)&gt;95 恢复: max(/tops-server/system.cpu.util[Usage],10m)&lt;95</code>	已启用	defaultTag	
<input type="checkbox"/>	警告	正常	监控系统模板[主采集]:[操作系统][HOSTNAME]swap使用空间大于85%		<code>last(/tops-server/system.swap.size[used])&gt;85</code>	已启用	defaultTag	

## 2. 触发器配置

### 2.1. 配置第一个触发器

假定 0-8 点为企业业务低谷，期间业务系统 CPU 使用率一般不超过 75%，可以将触发器配置为：当 CPU 使用率超过 75% 时即触发告警。

触发器

所有主机 / 监控系统-Server 已启用 ZBX 监控项 140 触发器 61 图形 19 自动发现规则 4 Web 场景

触发器 标记 依赖关系

\* 名称: {HOST.NAME} CPU使用率在0-8点期间大于75%

Event name: {HOST.NAME} CPU使用率在0-8点期间大于75%

Operational data:

严重性: 未知 信息 警告 次要 **严重** 紧急

\* 表达式: `now()>=000000 and now()<=080000 and last(/itops-server/system.cpu.util[Usage])>75` 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

允许手动关闭:

URL:

描述:

已启用:

添加 取消

表达式:

```
now()>=000000 and now()<=080000 and last(/itops-server/system.cpu.util[Usage])>75
```

表达式含义:

`now()>=000000` #当前时间大于 0 点时触发 (时间格式为 HHMMSS)

`now()<=080000` #当前时间小于 8 点时触发 (时间格式为 HHMMSS)

`last(/itops-server/system.cpu.util[Usage])>75` #主机 CPU 使用率指标最新取值大于 75

时触发

**注：**表达式用 and 将各个函数串联，所有条件满足时触发器才会触发告警

## 2.2. 配置第二个触发器

假定 8-24 点为企业业务高峰，期间业务系统 CPU 使用率可能接近或短暂超过 90%，可以将触发器配置为：当 CPU 使用率超过 90%时触发告警。该触发器与第一个类似，只需修改时段、触发阈值。

触发器

所有主机 / 监控系统-Server 已启用 ZBX 监控项 140 触发器 63 图形 19 自动发现规则 4 Web 场景

触发器 标记 依赖关系

\* 名称 {HOST.NAME} CPU使用率在8-24点期间大于90%

Event name {HOST.NAME} CPU使用率在8-24点期间大于90%

Operational data

严重性 未知 信息 警告 次要 严重 紧急

\* 表达式 `time()>080000 and time()<=235959 and last(/itops-server/system.cpu.util[Usage])>90` 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

允许手动关闭

URL

描述

已启用

更新 克隆 删除 取消

表达式：

```
time()>080000 and time()<=235959 and last(/itops-server/system.cpu.util[Usage])>90
```

当完成以上配置后，主机 CPU 使用率在 0-8 点期间大于 75%时告警，在 8-24 点期间大于

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

**90%时告警，至此完成分时段不同告警阈值的设置。**

以上就是本期的全部内容。大家好，我是乐乐专注 IT 运维技术研究与分享，更多 zabbix 等开源监控工具使用技巧欢迎关注乐维社区，更多运维问题也欢迎留言提问。



## 三十九、如何生成好看的 zabbix 告警报表并发送邮件

### 一、场景模拟

小东是一名资深的 IT 运维人员，其直属领导想要了解公司业务系统的健康状态以及小东日常的工作情况等，要求小东每周统计系统告警情况并发邮件给他。小东所在公司搭建了一套 zabbix 开源监控，于是小东利用自己的专业知识，实现了 zabbix 告警统计，并生成漂亮的告警报表发送给自己的领导，得到了领导的称赞。

Zabbix 开源监控是 IT 监控领域的佼佼者，拥有强大的告警统计与报表生成能力，但对于刚上手的小白来说可能还有些难度，本文将详细介绍 zabbix 告警报表的生成过程及发送邮件的操作步骤。

### 二、实现原理

环境说明：本人部署的 zabbix 版本为 6.0 加 postgresql14.4 数据库，如果用 mysql 的话，查询语句可能不一致

#### 1、数据来源

基于 python 脚本实现，安装 psqcopg2 库，查询 pg 数据库的数据，主要是统计出最近一个月内告警出现最多的触发器，返回 10-20 条数据即可。统计最近一个月内哪些对象出现的告警次数比较多，我个人的话，返回的数据是 1000 条

#### 2、数据清理

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

把主要是统计出最近一个月内告警出现最多的触发器的数据整理成一个列表数据, 数据格式

如下

```
[[{"告警标签"}, {"触发器名称"}, {"告警次数"}, {"告警颜色"}]]
```

统计最近一个周内哪些对象出现的告警次数比较多(本人加了限制, 就是一个对象同一个触

发器起码一周内出现超过 5 次才统计), 我个人的话, 返回的数据是 1000 条

```
[[{"对象名称"}, {"对象 ip"}, {"触发器名称"}, {"告警出现次数"}]]
```

### 3、脚本实现过程

通过 python 脚本把最近一个月告警出现最多的触发器动态封装成 html 发到邮件正文。把

一周内哪些对象出现的告警次数多的填充到 excel 表, 并作为附件发送到邮件

## 三、实现步骤

需要安装一下 python 库

```
import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import psycopg2

import openpyxl

from openpyxl.styles import Alignment, Font

from openpyxl.utils import get_column_letter

import smtplib
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

import os
```

```
#!/usr/bin/env python3
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import psycopg2
import openpyxl
from openpyxl.styles import Alignment, Font
from openpyxl.utils import get_column_letter
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
import os
```

部分代码

Postgresql 数据库连接代码

```
config = {

    'host': '数据库 ip',

    'port': 5432, # PostgreSQL 默认端口是 5432

    'user': '用户名',

    'password': '密码',

    'database': '数据库名'
```

```
}

try:

    connection = psycopg2.connect(**config)

    with connection.cursor() as cursor:

        sql_query="""SELECT  t.description  AS  trigger_name,  t.priority  AS
trigger_level, COUNT(e.eventid) AS occurrence_count FROM events e JOIN triggers
t ON e.objectid = t.triggerid WHERE e.source = 0 AND e.object = 0 AND e.clock >=
EXTRACT(EPOCH FROM (NOW() - INTERVAL '1 WEEK')) GROUP BY t.description,
t.priority ORDER BY occurrence_count DESC LIMIT 100;"""

        cursor.execute(sql_query)

        results = cursor.fetchall()

        for row in results:

            excel_data.append([row[0],row[1],row[2],row[3]])

except Exception as e:

    Print(e)
```

```
#查询sql2
sql2="""WITH trigger_alerts AS (SELECT t.triggerid, i.hostid, h.host AS alert_ip, hi.ip AS alert_ip_address, C
OUNT(e.eventid) AS alert_count FROM triggers t JOIN functions f ON t.triggerid = f.triggerid JOIN items i ON f.itemid =
i.itemid JOIN events e ON t.triggerid = e.objectid AND e.object = 0 JOIN hosts h ON i.hostid = h.hostid JOIN interface
hi ON h.hostid = hi.hostid WHERE e.clock >= EXTRACT(EPOCH FROM DATE_TRUNC('week', CURRENT_DATE)) AND e.clock < EXTRACT
(EPOCH FROM (DATE_TRUNC('week', CURRENT_DATE) + INTERVAL '1 week')) GROUP BY t.triggerid, i.hostid, h.host, hi.ip HAVIN
G COUNT(e.eventid) > 5) SELECT ta.alert_ip, ta.alert_ip_address, REPLACE(t.description, '{HOST.NAME}', ta.alert_ip) AS
trigger_function, ta.alert_count FROM trigger_alerts ta JOIN triggers t ON ta.triggerid = t.triggerid ORDER BY ta.alert
_count DESC;
"""
excel_data=[]
cursor.execute(sql2)

# 打印查询结果
results = cursor.fetchall()
for row in results:
    excel_data.append([row[0], row[1], row[2], row[3]])
```

数据清洗封装动态 html 部分代码(有些 css 样式在邮件会不生效，所以可能需要用内嵌 style)

```
table_rows = ""

for trigger in trigger_stats:

    table_rows += "<tr style='width:10%;height:10px'><td

style='width:8%;border: 1px solid #999;text-align:center;padding: 5px 0;'><span

style='background-color: {}; padding: 2px 6px; border-radius: 3px;

color:white;'>{}</span></td><td style='width:10%;border: 1px solid

#999;text-align:center;padding: 5px 0;color: #f40;'>{}</td><td

style='width:10%;border: 1px solid #999;text-align:center;padding: 5px

0;'>{}</td></tr>".format(trigger['color'],trigger['level'],trigger["name"],

trigger["count"])
```

```
table_rows = ""
for trigger in trigger_stats:
    table_rows += "<tr style='width:10%;height:10px'><td style='width:8%;border: 1px solid #999;text-align:center;p
adding: 5px 0;'><span style='background-color: {}; padding: 2px 6px; border-radius: 3px; color:white;'>{}</span></td><t
d style='width:10%;border: 1px solid #999;text-align:center;padding: 5px 0;color: #f40;'>{}</td><td style='width:10%;bo
rder: 1px solid #999;text-align:center;padding: 5px 0;'>{}</td></tr>".format(trigger['color'],trigger['level'],trigger[
"name"], trigger["count"])

# 设置邮件内容
subject = '一个月告警统计报表'
html_content = ''
<html>
<head>
<style>
table {{
width: 60%;
border-collapse: collapse;
}}
```

Excel 数据填充部分代码(data 为清理好的数据)

```
def generate_excel(data):  
  
    # 创建一个新的工作簿  
  
    wb = openpyxl.Workbook()  
  
    sheet = wb.active  
  
    sheet.title = '告警报表'  
  
  
    # 定义表头  
  
    headers = ['业务名称', 'IP', '告警信息', '告警次数']  
  
  
    # 写入表头  
  
    for col_idx, header in enumerate(headers, start=1):  
  
        cell = sheet.cell(row=1, column=col_idx)  
  
        cell.value = header  
  
        cell.font = Font(bold=True)  
  
        cell.alignment = Alignment(horizontal='center')  
  
  
    # 填充数据  
  
    for row_idx, row_data in enumerate(data, start=2):  
  
        for col_idx, value in enumerate(row_data, start=1):
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
cell = sheet.cell(row=row_idx, column=col_idx)

cell.value = value

print(cell.value)

# 保存工作簿

excel_file = './alert_data.xlsx'

wb.save(os.path.basename(excel_file))
```

```
def generate_excel(data):
    # 创建一个新的工作簿
    wb = openpyxl.Workbook()
    sheet = wb.active
    sheet.title = '告警报表'

    # 定义表头
    headers = ['业务名称', 'IP', '告警信息', '告警次数']

    # 写入表头
    for col_idx, header in enumerate(headers, start=1):
        cell = sheet.cell(row=1, column=col_idx)
        cell.value = header
        cell.font = Font(bold=True)
        cell.alignment = Alignment(horizontal='center')

    # 填充数据
    for row_idx, row_data in enumerate(data, start=2):
        for col_idx, value in enumerate(row_data, start=1):
            cell = sheet.cell(row=row_idx, column=col_idx)
            cell.value = value
            print(cell.value)

    # 保存工作簿
    excel_file = './alert_data.xlsx'
    wb.save(os.path.basename(excel_file))
```

邮件发送部分代码

```
# 创建 MIMEMultipart 对象

msg = MIMEMultipart('alternative')

msg['From'] = sender_email

msg['To'] = receiver_email

msg['Subject'] = subject

# 附加 HTML 内容

msg.attach(MIMEText(html_content, 'html'))

generate_excel(excel_data)

with open('./alert_data.xlsx', 'rb') as attachment:

    part = MIMEBase('application', 'octet-stream')

    part.set_payload(attachment.read())

    encoders.encode_base64(part)

    part.add_header('Content-Disposition', 'attachment;
filename="{0}".format('./alert_data.xlsx'))

msg.attach(part)

try:

    # 连接到 QQ 邮箱的 SMTP 服务器

    server = smtplib.SMTP_SSL('smtp.qq.com', 465)

    server.login(sender_email, sender_password)
```



```
server.sendmail(sender_email, receiver_email, msg.as_string())

print('邮件发送成功')

except Exception as e:

    print('邮件发送失败:', e)

finally:

    server.quit()
```

```
# 创建MIMEMultipart对象
msg = MIMEMultipart('alternative')
msg['From'] = sender_email
msg['To'] = receiver_email
msg['Subject'] = subject

# 附加HTML内容
msg.attach(MIMEText(html_content, 'html'))
generate_excel(excel_data)
with open('./alert_data.xlsx', 'rb') as attachment:
    part = MIMEBase('application', 'octet-stream')
    part.set_payload(attachment.read())
encoders.encode_base64(part)
part.add_header('Content-Disposition', 'attachment; filename="{0}"'.format('./alert_data.xlsx'))
msg.attach(part)

try:
    # 连接到QQ邮箱的SMTP服务器
    server = smtplib.SMTP_SSL('smtp.qq.com', 465)
    server.login(sender_email, sender_password)
    server.sendmail(sender_email, receiver_email, msg.as_string())
    print('邮件发送成功')
except Exception as e:
    print('邮件发送失败:', e)
finally:
    server.quit()
```

## 四、成果演示

### 1、邮件一个月的告警统计报表

收件人: [redacted]@qq.com>  
附件: 1 个 (alert\_data.xlsx)

### 告警统计

以下是最近一个月告警的统计信息:

告警等级	告警名称	统计次数
严重	[操作系统]{[redacted]}的主动监控模式无法及时采集到数据	2039
严重	[网络设备]{[redacted]}:[{[redacted]}]CPU使用率平均3分钟高于90%	1085
严重	[网络链路]{HOST.NAME}设备端口{[redacted]}无法采集到带宽数据	451
警告	[网络设备]{[redacted]}板卡{[redacted]}CPU使用率平均3分钟高于80%	308
严重	[网络设备]{[redacted]}SNMP采集中断	129
严重	[网络设备]{[redacted]}失联,持续3分钟未响应	62
严重	[网络链路]{[redacted]}设备端口{[redacted]}带宽接收利用率连续3次取值大于0.01%	8
严重	[redacted]	8
严重	[硬件设备]{[redacted]}失联,持续3分钟未响应,设备可能宕机	2

## 2、告警统计 excel 邮件附件(部分宏值需要额外替换成具体值)

业务名称	IP	告警信息	告警次数
	192.168.38	[操作系统]192.38的主动监控模式无法及时采集到数据	665
	192.168.202	[网络设备]192.202_1板卡[=#SNMPVALUE]CPU使用率平均3分钟高于90%	380
	192.168.200	[网络链路]192.200-GigabitEthernet0-0-11设备端口{PORT}带宽接收利用率连续3次取值大于0.01%	139
	192.168.202	[网络设备]192.202_1板卡[=#SNMPVALUE]CPU使用率平均3分钟高于80%	90
	192.168.200	[网络设备]192.200-AC失联,持续3分钟未响应	48
	192.168.200	[网络设备]192.200-ACSNMP采集中断	32
	192.168.203	[网络设备]192.203-H3C失联,持续3分钟未响应	16
	192.168.206	[网络设备]192.206失联,持续3分钟未响应	16
	192.168.208	[网络设备]192.208失联,持续3分钟未响应	8
	192.168.201	[硬件设备]192.201失联,持续3分钟未响应,设备可能宕机	8
	192.168.202	[网络链路]192.202-Ethernet0-0-3设备端口{PORT}无法采集到带宽数据	8
	192.168.38	[操作系统]192.38失联,持续3分钟未响应,系统可能宕机	8
	192.168.202	[网络设备]192.202_1失联,持续3分钟未响应	8
	192.168.201	[网络设备]192.201-AC失联,持续3分钟未响应	8
	192.168.201	[网络设备]192.201-AC7失联,持续3分钟未响应	8
	192.168.202	[网络链路]192.202-Ethernet0-0-20设备端口{PORT}无法采集到带宽数据	8

## 四十、更多.....

# 监控配置

## 四十三、Zabbix 监控 Spark 中间件配置教程

本文将介绍以 JMX 方式监控 Spark 中间件。JMX 具有跨平台、灵活性强、监控能力强、易于集成与扩展、图形化界面支持以及安全性与可配置性等多方面的优势，是监控 Spark 等复杂 Java 应用程序的重要工具之一。

Apache Spark 是一个开源的大数据处理框架，它提供了快速、通用和可扩展的数据处理能力，适用于执行大规模的数据处理和分析任务，特别是在批处理、实时流处理、机器学习和图计算等领域。

JMX (Java Management Extensions) 作为 Java 平台标准的一部分，提供了一种标准化的机制，用于监控和管理应用程序、系统对象、设备和服务。JMX 技术可以跨越不同的操作系统、体系结构和网络传输协议，灵活地开发无缝集成的系统、网络和服务管理应用。

JMX 可以被用于多种管理任务，包括：

- **系统监控**：监控系统的 CPU 使用率、内存消耗、线程数等指标。
- **性能调优**：获取应用程序的性能数据，如方法执行时间、请求响应时间等，帮助开发者找出性能瓶颈并进行优化。
- **故障排查**：当系统出现故障时，通过 JMX 快速定位问题所在，如查看日志、监控线程状态等。
- **安全管理**：实现系统的安全控制，如用户认证、访问控制等。
- **服务管理**：对于分布式系统，监控服务状态、管理服务实例，提高系统的可用性和可维护性。

## 二、Spark 配置开启 JMX 服务

### 1、编写 Spark 配置文件：

在安装的 spark 服务里找到名称为 “spark-env.sh” 的配置文件，进行编辑修改。

### 2、在 Spark 配置文件里启用 JMX：

输入命令：vi spark-env.sh 打开配置文件，并在文件中添加以下参数来开启 JMX 并设置 JMX 的监听端口。

```
export SPARK_DAEMON_JAVA_OPTS="$SPARK_DAEMON_JAVA_OPTS -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=7099 -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false"
```

解析：开启 JMX 服务并设置端口为 7099。

### 3、重启 Spark 服务：

根据自身所搭建的 Spark 模式来重启 Spark，以便 Spark 服务应用添加这些参数的更改

## 三、下载测试工具 cmdline-jmxclient-0.10.3.jar 包进行连通性测试

### 1、下载测试工具 cmdline-jmxclient-0.10.3.jar 包

cmdline-jmxclient-0.10.3.jar 为一个测试工具，可用来测试 JMX 是否配置正确，下载 cmdline-jmxclient-0.10.3.jar(下载到任意目录)。

```
wget http://crawler.archive.org/cmdline-jmxclient/cmdline-jmxclient-0.10.3.jar
```

### 2、测试 JAR 包，注意 IP 地址与 JMX 端口需与配置文件一致

```
java -jar cmdline-jmxclient-0.10.3.jar - 192.168.3.34:7099
```

### 3、测试成功效果图

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
[root@localhost tmp]# java -jar cmdline-jmxclient-0.10.3.jar - 192.168.3.1
java.lang:type=OperatingSystem
java.lang:name=Metaspace Manager,type=MemoryManager
java.lang:name=Metaspace,type=MemoryPool
JMXImplementation:type=MBeanServerDelegate
java.lang:name=PS Old Gen,type=MemoryPool
java.lang:type=ClassLoading
java.lang:type=Runtime
java.lang:name=PS Scavenge,type=GarbageCollector
com.sun.management:type=HotSpotDiagnostic
java.lang:type=Threading
java.lang:name=CodeCacheManager,type=MemoryManager
java.lang:name=PS Eden Space,type=MemoryPool
java.nio:name=mapped,type=BufferPool
com.oracle.management:type=ResourcePressureMBean
java.lang:name=Code Cache,type=MemoryPool
java.lang:name=Compressed Class Space,type=MemoryPool
java.nio:name=direct,type=BufferPool
java.lang:name=PS Survivor Space,type=MemoryPool
java.util.logging:type=Logging
com.sun.management:type=DiagnosticCommand
java.lang:name=PS MarkSweep,type=GarbageCollector
java.lang:type=Memory
java.lang:type=Compilation
[root@localhost tmp]#
```

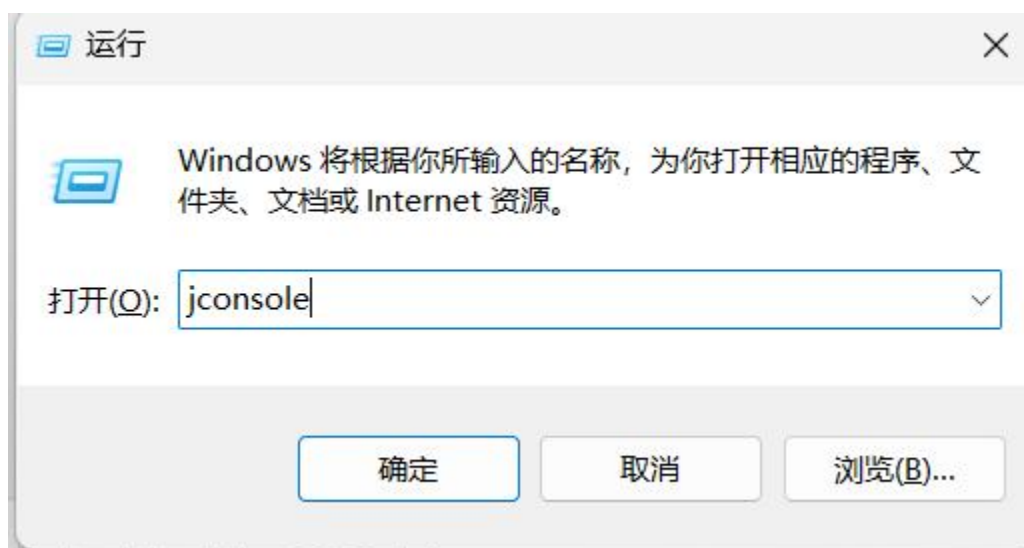
测试成功则证明可以成功连接并且返回数据。

#### 四、如何进行监控项添加监控

1、需要使用 windows 的 jconsole 控制台制作监控项，安装方式可以自行去搜索对应教程进行安装。

2、运用方式：

(1) 【win+r】 --- 【输入 jconsole】



(2) 选择远程进程，输入 IP:JMX 端口来进行登录



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



## 四十四、如何配置 SNMPTrap 监控

SNMP 是一种网络管理协议，SNMPTrap 则是基于此协议的一种数据传输方式。在 Zabbix 监控实践中，SNMP 的应用非常广泛，下面我们来看一下如何进行 SNMPTrap 配置监控。

当我们获取到设备发送过来的 trap 的时候，需要是从 trap 信息中获取到 Trap OID，如下图，这里我们会用到一个工具叫 MIB Browser。

以华为 RH5885V3 为例：

### 1、定位 TrapOID

首先看到 snmpTrapOID (横线部分)，对应的 value，就是我们所需要的 TrapOID 了 (方框部分)。

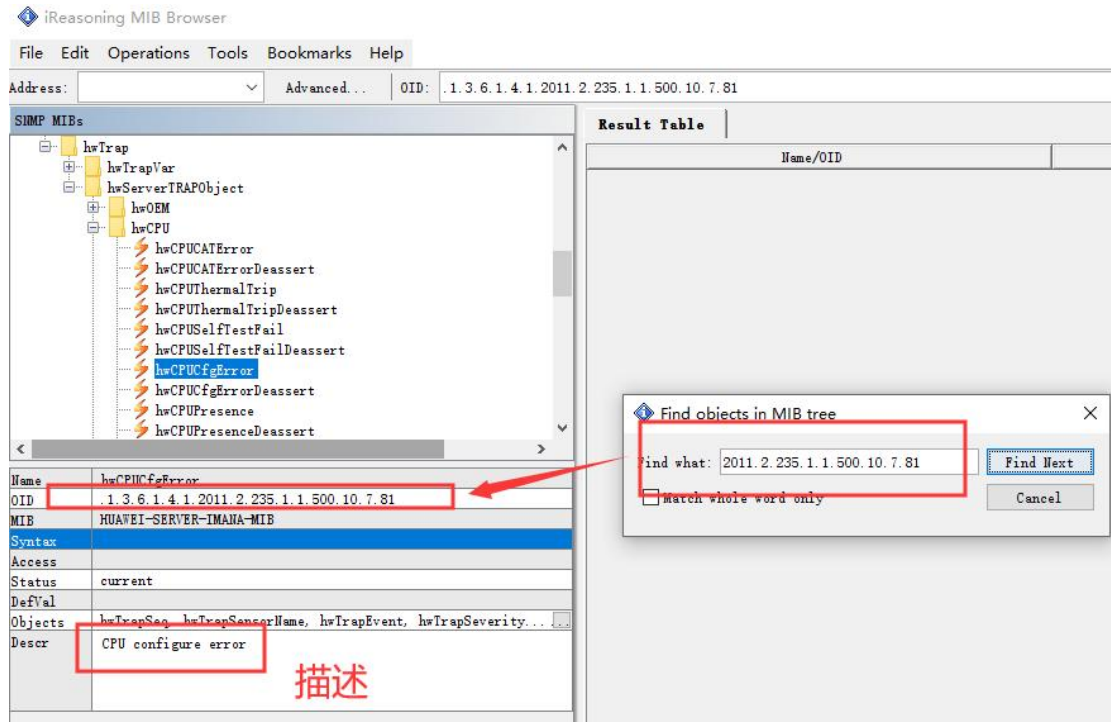
```
14:40:29 2019/11/20 ZBXTRAP 192.168.22.19
PDU INFO:
notificationtype      TRAP
version               1
receivedfrom          UDP: [192.168.22.19]:52791->[10.142.88.81]
errorstatus           0
messageid             0
community             TrapHuawei12#$
transactionid         2478220
errorindex            0
requestid             2019432088
VARBINDS:
DISMAN-EVENT-MIB::sysUpTimeInstance type=67 value=Timeticks: (225886099) 26 days 3:27:40.99
SNMPv2-MIB::snmpTrapOID.0 type=6 value=OID: SNMPv2-SMI::enterprises.2011.2.235.1.1.500.10.7.81
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.1 type=2 value=INTEGER: 9628
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.2 type=4 value=STRING: "CPU2 Status"
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.3 type=4 value=STRING: "Configuration error"
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.4 type=2 value=INTEGER: 4
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.5 type=4 value=STRING: "0x0705ffff"
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.6 type=2 value=INTEGER: 255
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.7 type=2 value=INTEGER: 255
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.8 type=4 value=STRING: "022CUU10F1000361"
SNMPv2-SMI::enterprises.2011.2.235.1.1.500.1.9 type=4 value=""
```

### 2、TrapOID 解析

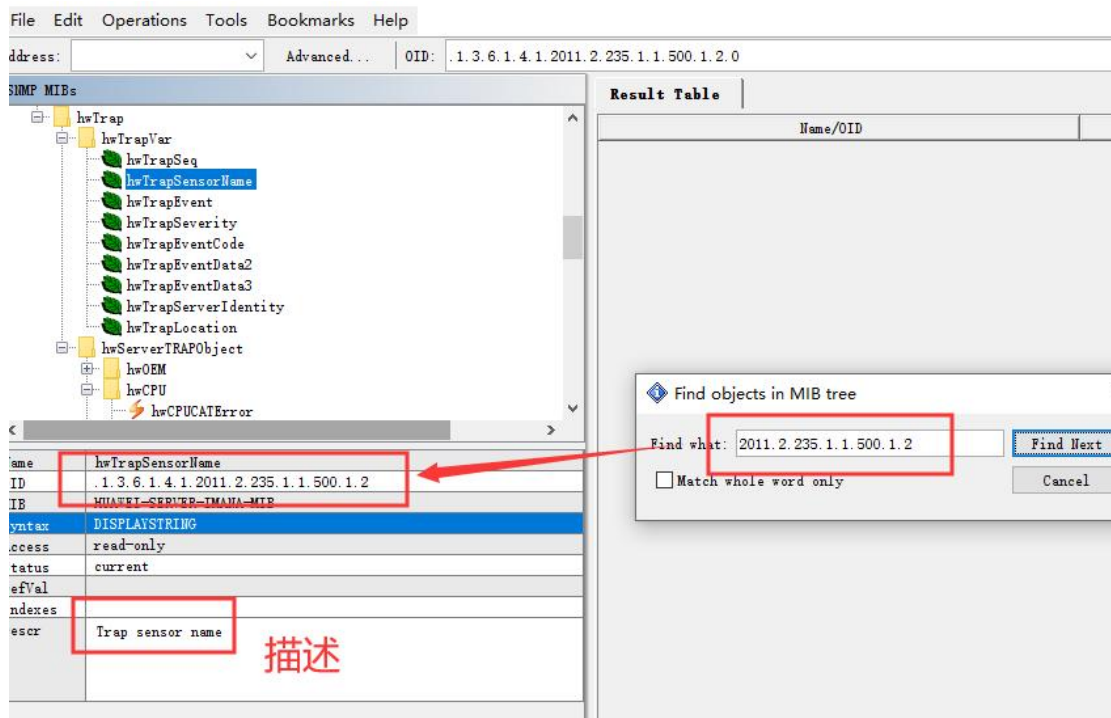
打开 MIB Browser 浏览器，加载华为 RH5885V3 的 MIB 文件，并复制 TrapOID 数字部分，即：2011.2.235.1.1.500.10.7.81，然后到 MIB Browser 浏览器上 ctrl + F 进行搜索，



就可以知道是什么部件发生了告警，如下图可得知 CPU 出现故障。



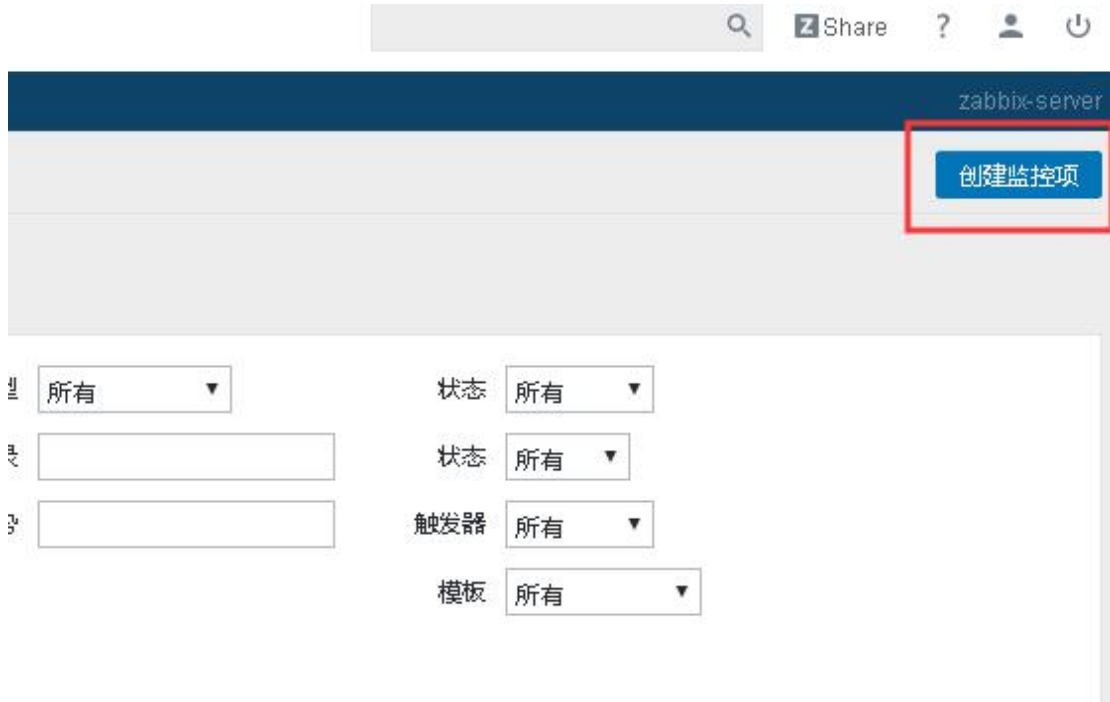
如果我们想知道更详细的信息，可以用同样的方式，复制后面的数字 OID 到 MIB Browser 浏览器里面进行搜索。例如我不知道 'CPU2 Status' 代表的是什么信息，可以复制下 OID，然后搜索一下，可得知 'CPU2 Status' 是传感器名，如下图所示。



### 3、Zabbix 创建 Trap 监控项

到 zabbix 上的模板创建监控项，如下图：

进入模板，点击右上角创建监控项



填好对应信息后，最下方点击 '添加' 即可。

名称 ▲	触发器	键值	间隔	历史记录	趋势	类型
CPU故障		snmptrap[2011.2.235.1.1.500.10.7.81]		90d		SNMP trap

## 4、Zabbix 创建 Trap 触发器

创建完监控项之后，我们开始创建触发器。

点击模板 ‘触发器’ —— ‘创建触发器’



点击 ‘添加’，触发器名称、严重性、问题事件生成模式可自定义。

A screenshot of the Zabbix 'Create Trigger' form. The form includes fields for '名称' (Name) with the value 'CPU故障', '严重性' (Severity) with '严重' selected, and a large '表达式' (Expression) text area. A '添加' (Add) button is highlighted in a red box. Below the expression field is a '表达式构造器' (Expression Builder) section with options for '事件成功迭代' (Event Success Iteration) and '事件生成模式' (Event Generation Mode). At the bottom, there is a '标记' (Tag) section with input fields for '标记' and '值', and a '移除' (Remove) button.

选择要做触发器的监控项与触发器功能：



填写好对应信息后，点击‘插入’，恢复表达式同理。



勾选‘允许手动关闭’，填写描述，最后勾选‘启用’，点击‘添加’即可。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

名称

严重性  未分类  信息  警告  一般严重  严重  灾难

表达式

[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

[添加](#)

允许手动关闭

URL

描述

完成后如下：

<input type="checkbox"/>	严重性	名称 ▲	表达式
<input type="checkbox"/>	严重	CPU故障	{test-allen:snmptrap[2011.2.235.1.1.500.10.7.81].regexp(2011.2.235.1.1.500.10.7.81)}=1

## 5、snmptrap.fallback

此监控项可以接收所有设备发送过来的 snmptrap 信息。也就是一个监控项对应多个触发器。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

---

名称	<input type="text" value="设备发生故障"/>
类型	<input type="text" value="SNMP trap"/>
键值	<input type="text" value="snmptrap.fallback"/> <input type="button" value="选择"/>
信息类型	<input type="text" value="字符"/>
历史数据保留时长	<input type="text" value="90d"/>
查看值	<input type="text" value="不变"/> <input type="button" value="展示值映射"/>
新的应用集	<input type="text"/>

以上就是这一期的Zabbix技术分享。

大家好，我是乐乐，关注我，学习更多 Zabbix 使用小技巧，如在 Zabbix 使用过程中碰到问题，还可以到[乐维社区](#)进行留言提问。

SNMP trap 官方参考文档：

<https://www.zabbix.com/documentation/3.4/zh/manual/config/items/itemtypes/snmptrap>

## 四十五、使用 Zabbix SNMP 添加自定义 OID

### 1、编写脚本（脚本内容如下）

```
#!/bin/sh
```

```
php_conn=`/bin/ps aux | /bin/grep nginx | egrep -v 'grep' | wc -l`
```

```
echo $php_conn
```

```
[root@itim_6 ~]# cat /root/php_online.sh
#!/bin/sh
php_conn=`/bin/ps aux | /bin/grep nginx | egrep -v 'grep' | wc -l`
echo $php_conn
[root@itim_6 ~]#
```

### 2、确认 oid 是否被系统占用，如.1.3.6.1.4.1.2023.6900

```
[root@itim_6 ~]# snmpwalk -v 2c -c public 192.168.46.188 .1.3.6.1.4.1.2023.6900
```

```
0
```

```
[root@itim_6 ~]# snmpwalk -v 2c -c public 192.168.46.188 .1.3.6.1.4.1.2023.6900
SNMPv2-SMI::enterprises.2023.6900 = No Such Object available on this agent at this OID
[root@itim_6 ~]#
```

通过上图可知 oid 不存在

### 3、配置 snmpd

```
[root@itim_6 ~]# vim /etc/snmp/snmpd.conf
```

在 snmpd.conf 配置文件末尾插入如下内容：



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
extend .1.3.6.1.4.1.2023.6900 php /bin/sh /root/php_online.sh
```

```
#####  
# Further Information  
#  
# See the snmpd.conf manual page, and the output of "snmpd -H".  
extend .1.3.6.1.4.1.2023.6900 php /bin/sh /root/php_online.sh
```

未被系统占用oid      名称自定义      脚本绝对路径

保存配置文件并退出，重新启动 snmp 服务

```
[root@itim_6 ~]# systemctl restart snmpd.service
```

## 4、测试自定义 OID

```
[root@itim_6 ~]# snmpwalk -v 2c -c public 192.168.46.188 .1.3.6.1.4.1.2023.6900
```

```
0
```

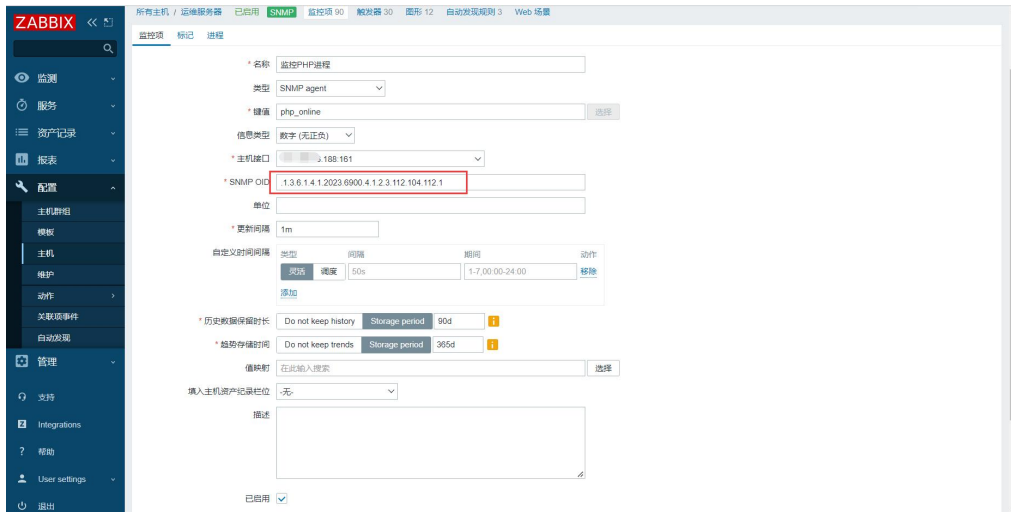
```
[root@itim_6 ~]# snmpwalk -v 2c -c public 192.168.46.188 .1.3.6.1.4.1.2023.6900  
SNMPv2-SMI::enterprises.2023.6900.1.0 = INTEGER: 1  
SNMPv2-SMI::enterprises.2023.6900.2.1.2.3.112.104.112 = STRING: "/bin/sh"  
SNMPv2-SMI::enterprises.2023.6900.2.1.3.3.112.104.112 = STRING: "/root/php_online.sh"  
SNMPv2-SMI::enterprises.2023.6900.2.1.4.3.112.104.112 = ""  
SNMPv2-SMI::enterprises.2023.6900.2.1.5.3.112.104.112 = INTEGER: 5  
SNMPv2-SMI::enterprises.2023.6900.2.1.6.3.112.104.112 = INTEGER: 1  
SNMPv2-SMI::enterprises.2023.6900.2.1.7.3.112.104.112 = INTEGER: 1  
SNMPv2-SMI::enterprises.2023.6900.2.1.20.3.112.104.112 = INTEGER: 4  
SNMPv2-SMI::enterprises.2023.6900.2.1.21.3.112.104.112 = INTEGER: 1  
SNMPv2-SMI::enterprises.2023.6900.3.1.1.3.112.104.112 = STRING: "11"  
SNMPv2-SMI::enterprises.2023.6900.3.1.2.3.112.104.112 = STRING: "11"  
SNMPv2-SMI::enterprises.2023.6900.3.1.3.3.112.104.112 = INTEGER: 1  
SNMPv2-SMI::enterprises.2023.6900.3.1.4.3.112.104.112 = INTEGER: 0  
SNMPv2-SMI::enterprises.2023.6900.4.1.2.3.112.104.112.1 = STRING: "11"  
[root@itim_6 ~]#
```

以上为获取自定义 oid 的所有数据，最后一行是我们需要获取的数据，在 zabbix web 界面

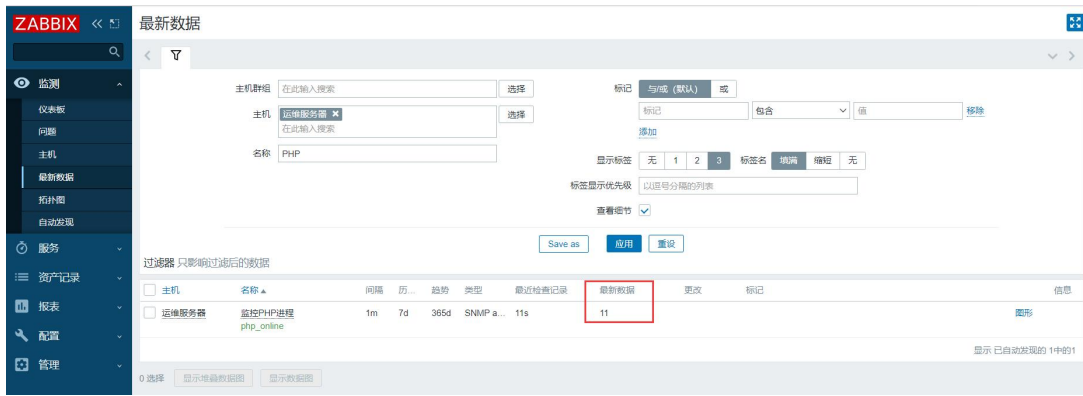
中填写的 oid 就是：.1.3.6.1.4.1.2023.6900.4.1.2.3.112.104.112.1



## 5、创建 snmp agent 类型监控项



## 6、验证是否正常获取数据



至此，Zabbix SNMP 添加自定义 OID 已完成

## 四十六、更多.....

# 数据库

## 五十、如何使用 Zabbix 监控 TiDB 数据库？

### 概述

TiDB 数据库是一个常见的开源分布式关系型数据库，通过使用分布式事务、分布式 SQL 引擎和分布式存储引擎来实现高可用性和横向扩展性。而 Docker 则是一个开源的容器化平台，它可以帮助开发者在不同的环境中轻松地部署和运行应用程序。

本文将介绍如何使用 Docker 快速安装和配置 TiDB，并使用 Zabbix 监控 TiDB。

### 安装步骤

1、安装 docker 并拉取镜像。

```
yum install docker
```

```
systemctl start docker
```

```
docker search pingcap/tidb # 搜索镜像, 如果搜索不到需要设置 docker 仓库源
```

```
[root@cenos7 ~]# docker search pingcap/tidb
INDEX          NAME                                DESCRIPTION                                STARS    OFFICIAL  AUTOMATED
docker.io     docker.io/pingcap/tidb              TiDB docker for PingCAP.                 165      [OK]
docker.io     docker.io/pingcap/tidb-operator      5
docker.io     docker.io/pingcap/tidb-binlog        4
docker.io     docker.io/pingcap/tidb-arm64         1
docker.io     docker.io/pingcap/tidb-backup-manager 1
docker.io     docker.io/pingcap/tidb-backup-manager-arm64 0
docker.io     docker.io/pingcap/tidb-binlog-arm64  0
docker.io     docker.io/pingcap/tidb-binlog-enterprise 0
docker.io     docker.io/pingcap/tidb-cloud-backup  0
docker.io     docker.io/pingcap/tidb-control        0
docker.io     docker.io/pingcap/tidb-dashboard-installer 0
docker.io     docker.io/pingcap/tidb-debug          0
docker.io     docker.io/pingcap/tidb-enterprise     0
docker.io     docker.io/pingcap/tidb-enterprise-tools 0
docker.io     docker.io/pingcap/tidb-lightning      0
docker.io     docker.io/pingcap/tidb-lightning-arm64 0
docker.io     docker.io/pingcap/tidb-lightning-enterprise 0
docker.io     docker.io/pingcap/tidb-monitor-initializer 0
docker.io     docker.io/pingcap/tidb-monitor-initializer-arm64 0
docker.io     docker.io/pingcap/tidb-monitor-initializer-enterprise 0
docker.io     docker.io/pingcap/tidb-monitor-reloader 0
docker.io     docker.io/pingcap/tidb-operator-arm64 0
docker.io     docker.io/pingcap/tidb-patched-jira  0
docker.io     docker.io/pingcap/tidb-tools          0
docker.io     docker.io/pingcap/tidb-vision         TiDB cluster visualization tool          0
[root@cenos7 ~]#
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

设置仓库源如：

```
vi /etc/docker/daemon.json
```

```
{
  "registry-mirrors": [
    " https://docker.anyhub.us.kg"
  ]
}
```

```
systemctl restart docker
```

```
docker pull pingcap/tidb # 拉取镜像
```

```
docker pull pingcap/tikv # 拉取镜像，单机运行可不拉取
```

```
docker pull pingcap/pd # 拉取镜像，单机运行可不拉取
```

如果要从国内仓库源上拉取则执行：

```
docker pull docker.anyhub.us.kg/pingcap/tidb
```

```
docker pull docker.anyhub.us.kg/pingcap/tikv #单机运行可不拉取
```

```
docker pull docker.anyhub.us.kg/pingcap/pd #单机运行可不拉取
```

```
docker images # 查看拉取的镜像
```

```
[root@cenos7 ~]# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
docker.anyhub.us.kg/pingcap/pd   latest      a6ed80ff7339    4 months ago    404 MB
docker.anyhub.us.kg/pingcap/tikv  latest      173ec5e96e96    4 months ago    773 MB
docker.anyhub.us.kg/pingcap/tidb  latest      bd253ee807c6    4 months ago    393 MB
[root@cenos7 ~]#
```

## 2、初始化容器。

```
mkdir /tidb
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
docker run --privileged=true -d --name tidb-server -p 4000:4000 -p 10080:10080 docker.anyhub.us.kg/pingcap/tidb #创建并运行容器, 将会在后台启动一个名为 tidb-server 的容器, 并将容器的 4000 端口映射到宿主机的 4000 端口、容器的 10080 端口映射到宿主机的 10080 端口。
```

```
docker ps -a #查看容器运行状态
```

```
[root@cenos7 ~]# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
3a1ad4764d8e   docker.anyhub.us.kg/pingcap/tidb   "/tidb-server"         About a minute Up About a minute   0.0.0.0:4000->4000/tcp, 0.0.0.0:10080->10080/tcp   tidb-server
[root@cenos7 ~]#
```

```
mysql -h 127.0.0.1 -P 4000 -u root -D test #使用 mysql 客户端命令尝试登录 tidb
```

```
[root@cenos7 ~]# mysql -h 127.0.0.1 -P 4000 -u root -D test
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 2097166
Server version: 8.0.11-TiDB-v7.5.1 TiDB Server (Apache License 2.0) Community Edition, MySQL 8.0 compatible

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

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

MySQL [test]> show databases;
+-----+
| Database |
+-----+
| INFORMATION_SCHEMA |
| METRICS_SCHEMA |
| PERFORMANCE_SCHEMA |
| mysql |
| test |
+-----+
5 rows in set (0.00 sec)

MySQL [test]>
```

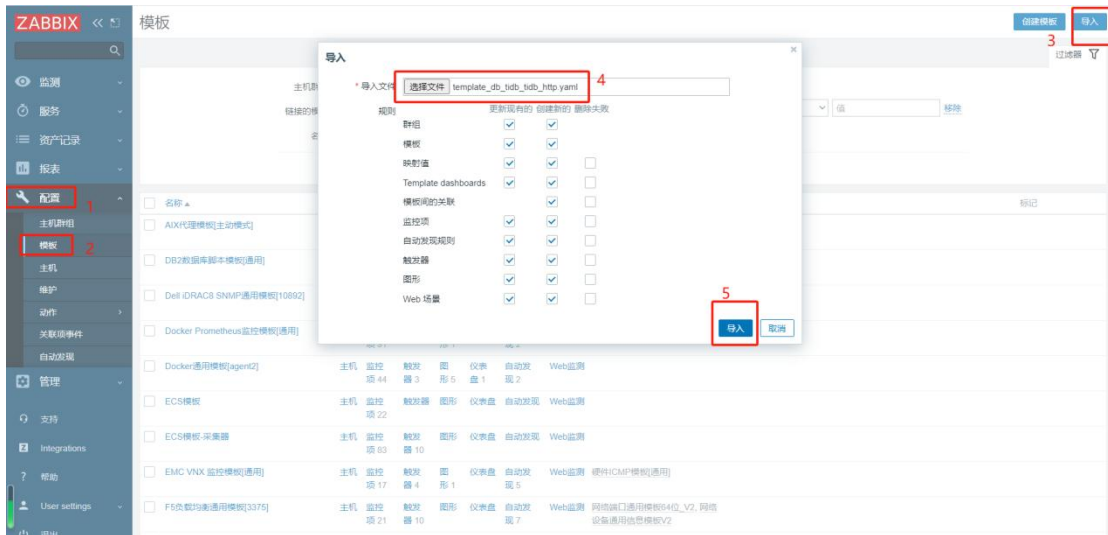
## 监控 TiDB

### 1、导入监控模板

使用 Zabbix 官方提供的监控模板：

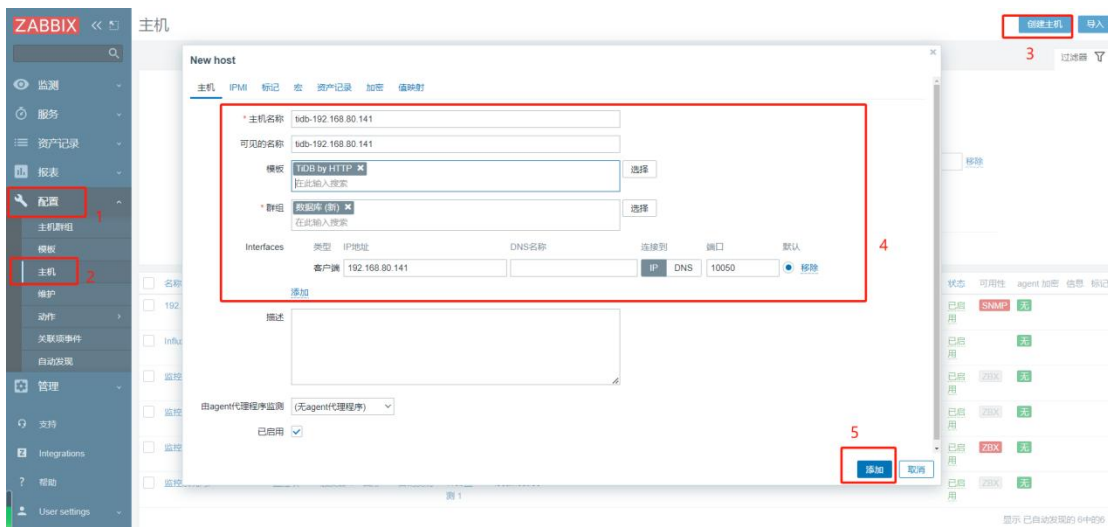
<https://www.zabbix.com/cn/integrations/tidb>

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



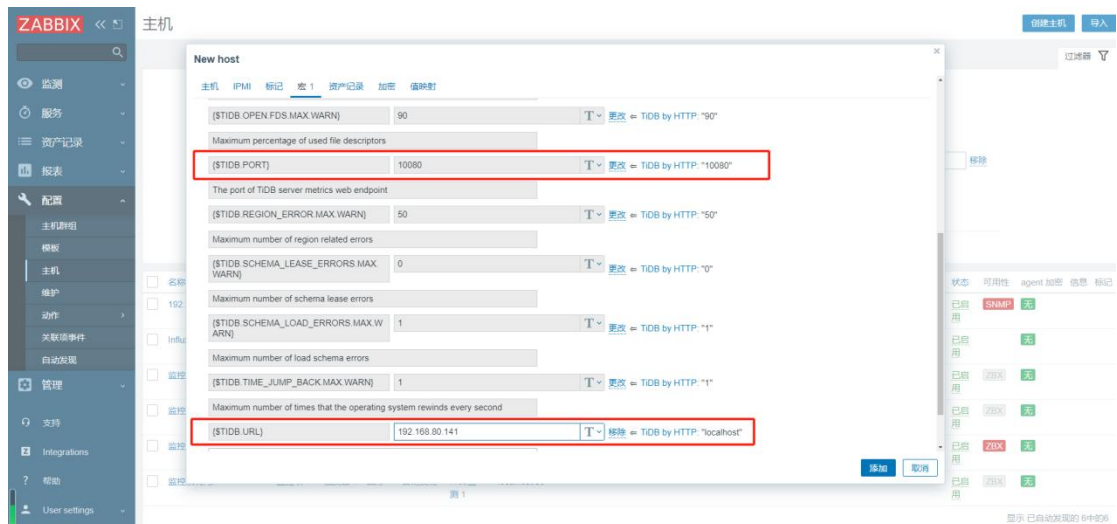
## 2、添加 TiDB 监控

点击配置->主机->创建主机，填写主机名称，选择刚刚导入的 TiDB 监控模板，设置一个群组。



点击宏，点击“继承以及主机 宏”，填写相关信息：

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



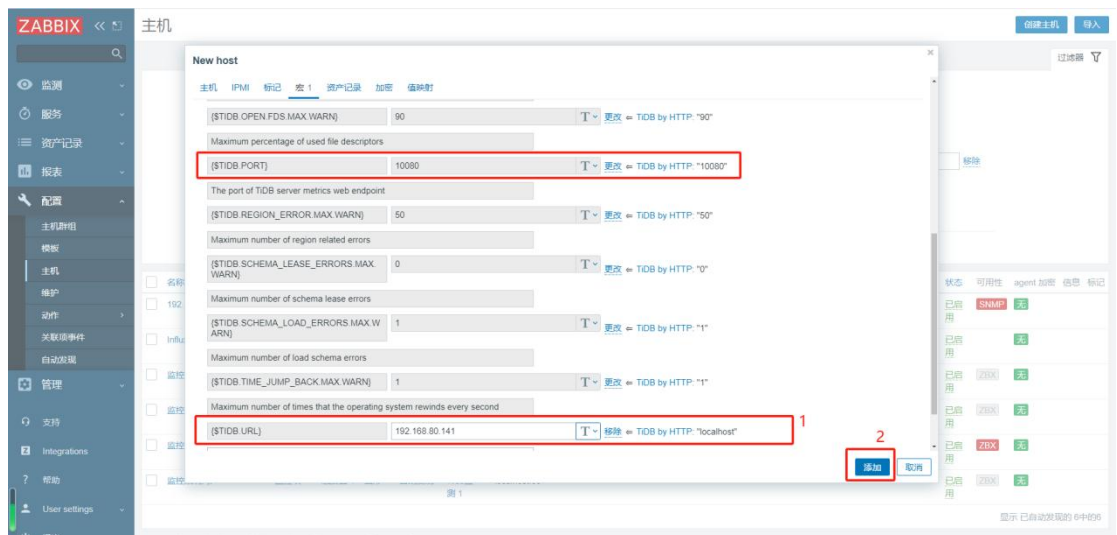
宏说明:

{TIDB.PORT}: 填写 TiDB 的接口端口, 不是连接用的端口, 默认为 10080。

{TIDB.URL}: 填写 TiDB 的 IP 地址。

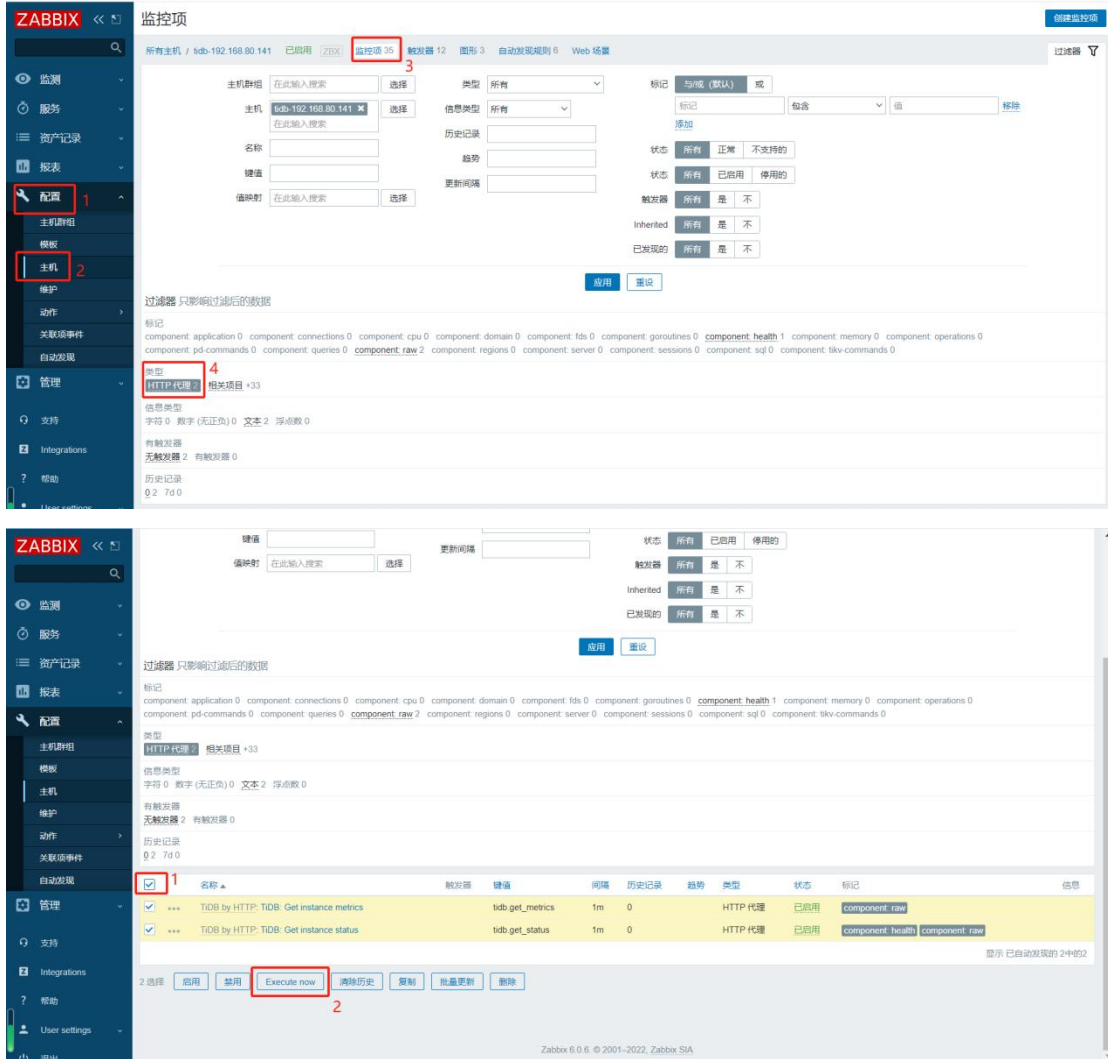
其他宏可使用默认值。

信息填写完成后, 点击添加按钮即可完成监控主机的添加。

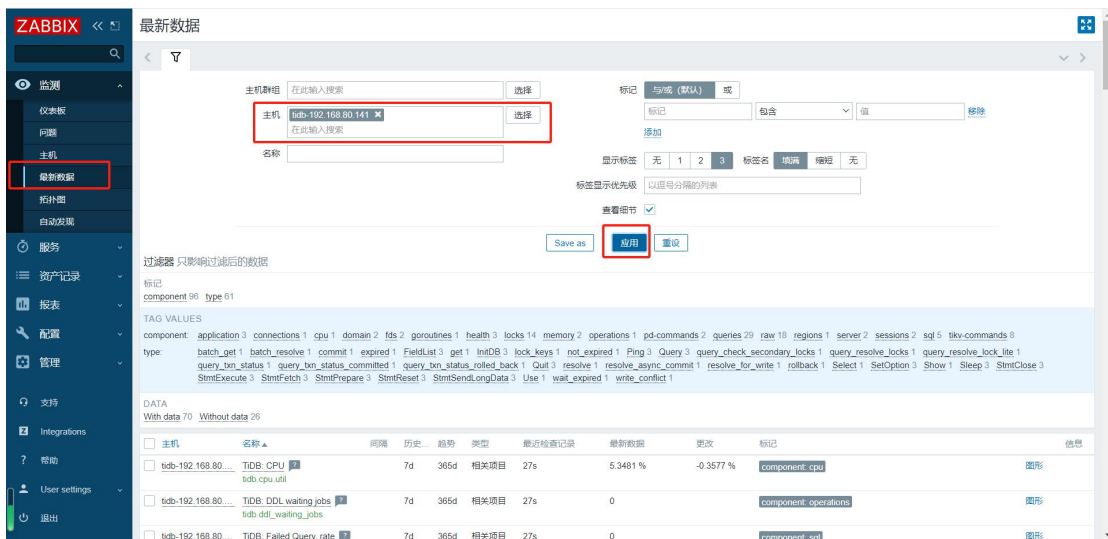


添加完成后, 可在主机管理界面, 对 http 代理类型的监控项触发立即执行, 使其快速获取数据和创建自动发现的监控项。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



到此，监控配置完成。查看监控数据：





本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

Host	Item	Unit	Value	Change	Component	Type	Link	
tdb-192.168.80...	TiDB: Get total server qu...		7d	相关项目	component: raw		历史...	
tdb-192.168.80...	TiDB: Goroutine count	7d 365d	相关项目	27s	140	+26	component: goroutines	图形
tdb-192.168.80...	TiDB: Heap memory usa...	7d 365d	相关项目	27s	376.03 MB	+18.25 MB	component: memory	图形
tdb-192.168.80...	TiDB: Keep alive rate	7d 365d	相关项目				component: health	图形
tdb-192.168.80...	TiDB: KV backlog rate	7d 365d	相关项目				component: tikv-com	图形
tdb-192.168.80...	TiDB: KV Commands b...	7d 365d	相关项目	27s	0.7754 Ops	+0.03918 Ops	component: tikv-com	图形
tdb-192.168.80...	TiDB: KV Commands c...	7d 365d	相关项目	27s	0 Ops		component: tikv-com type: batch_get	图形
tdb-192.168.80...	TiDB: KV Commands g...	7d 365d	相关项目	27s	0 Ops		component: tikv-com type: commit	图形
tdb-192.168.80...	TiDB: KV Commands lo...	7d 365d	相关项目	27s	0 Ops		component: tikv-com type: lock_keys	图形
tdb-192.168.80...	TiDB: KV Commands ro...	7d 365d	相关项目	27s	0 Ops		component: tikv-com type: rollback	图形
tdb-192.168.80...	TiDB: Load schema faile...	7d 365d	相关项目				component: domain	图形
tdb-192.168.80...	TiDB: Load schema total...	7d 365d	相关项目	27s	0.04561	+0.001439	component: domain	图形
tdb-192.168.80...	TiDB: Lock resolves rate	7d 365d	相关项目	27s	0 Ops		component: tikv-com	图形
tdb-192.168.80...	TiDB: Lock resolves: bat...	7d 365d	相关项目	27s	0 Ops		component: locks type: batch_resolve	图形
tdb-192.168.80...	TiDB: Lock resolves: ex...	7d 365d	相关项目	27s	0 Ops		component: locks type: expired	图形
tdb-192.168.80...	TiDB: Lock resolves: not...	7d 365d	相关项目	27s	0 Ops		component: locks type: not_expired	图形
tdb-192.168.80...	TiDB: Lock resolves: qu...	7d 365d	相关项目	27s	0 Ops		component: locks type: query_check_s...	图形
tdb-192.168.80...	TiDB: Lock resolves: qu...	7d 365d	相关项目	27s	0 Ops		component: locks type: query_resolve_I...	图形
tdb-192.168.80...	TiDB: Lock resolves: qu...	7d 365d	相关项目	27s	0 Ops		component: locks type: query_resolve_I...	图形
tdb-192.168.80...	TiDB: Lock resolves: qu...	7d 365d	相关项目	27s	0 Ops		component: locks type: query_resolve_I...	图形
tdb-192.168.80...	TiDB: Lock resolves: qu...	7d 365d	相关项目	27s	0 Ops		component: locks type: query_resolve_I...	图形
tdb-192.168.80...	TiDB: Lock resolves: qu...	7d 365d	相关项目	27s	0 Ops		component: locks type: query_resolve_I...	图形
tdb-192.168.80...	TiDB: Lock resolves: qu...	7d 365d	相关项目	27s	0 Ops		component: locks type: query_resolve_I...	图形
tdb-192.168.80...	TiDB: Lock resolves: res...	7d 365d	相关项目	27s	0 Ops		component: locks type: resolve	图形
tdb-192.168.80...	TiDB: Lock resolves: res...	7d 365d	相关项目	27s	0 Ops		component: locks type: resolve_async...	图形
tdb-192.168.80...	TiDB: Lock resolves: res...	7d 365d	相关项目	27s	0 Ops		component: locks type: resolve_for_writ...	图形
tdb-192.168.80...	TiDB: Lock resolves: wait...	7d 365d	相关项目	27s	0 Ops		component: locks type: wait_expired	图形
tdb-192.168.80...	TiDB: Lock resolves: writ...	7d 365d	相关项目	27s	0 Ops		component: locks type: write_conflict	图形
tdb-192.168.80...	TiDB: Open file descriptors	7d 365d	相关项目	27s	17		component: fds	图形
tdb-192.168.80...	TiDB: Open file descript...	7d 365d	相关项目	27s	1048576		component: fds	图形
tdb-192.168.80...	MySQL: Server query 'OK'	7d 365d	相关项目	27s	0 u/s		component: queries type: StmtReset	图形
tdb-192.168.80...	TiDB: Server query 'OK'	7d 365d	相关项目	27s	0 Ops		component: queries type: StmtSendLong	图形
tdb-192.168.80...	TiDB: SQL statements r...	7d 365d	相关项目	27s	0		component: sq	图形
tdb-192.168.80...	TiDB: SQL statements r...	7d 365d	相关项目	27s	0		component: sq type: Select	图形
tdb-192.168.80...	TiDB: SQL statements r...	7d 365d	相关项目	27s	0		component: sq type: Show	图形
tdb-192.168.80...	TiDB: SQL statements r...	7d 365d	相关项目	27s	0		component: sq type: Use	图形
tdb-192.168.80...	TiDB: Status	7d	相关项目	27s	Up (1)		component: health	历史...
tdb-192.168.80...	TiDB: TiClient region err...	7d 365d	相关项目	27s	0 Ops		component: regions	图形
tdb-192.168.80...	TiDB: Time jump back r...	7d 365d	相关项目	27s	0 Ops		component: application	图形
tdb-192.168.80...	TiDB: Total 'terror' server...	7d 365d	相关项目	27s	0 Ops		component: queries	图形
tdb-192.168.80...	TiDB: Total 'ok' server q...	7d 365d	相关项目	27s	0 Ops		component: queries	图形
tdb-192.168.80...	TiDB: Total server query...	7d 365d	相关项目	27s	0 Ops		component: queries	图形
tdb-192.168.80...	TiDB: Uptime	7d 365d	相关项目	27s	00:30:01	+00:01:28	component: application	图形
tdb-192.168.80...	TiDB: Version	7d	相关项目	27s	8.0.11-TiDB-v7.5.1		component: application	历史...

以上就是本期的全部内容。大家好，我是乐乐专注 IT 运维技术与分享，更多运维技巧

欢迎关注乐维社区，更多运维问题也欢迎到乐维社区留言提问。

整理 by 乐维社区 (<https://forum.lwops.cn>)



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

## 五十一、如何使用 Zabbix 监控神通数据库？

神通数据库，即神舟通用数据库（ShenTong Database），是我国自主研发的一款关系型数据库管理系统。它在国内市场有一定的应用，尤其是在一些对数据安全、独立性有较高要求的领域，如政府、金融、电信、能源、医疗等行业。

Zabbix 是一款被广泛应用的开源监控工具，常常被用于企业机构等 IT 基础架构的监控。

本文将介绍如何使用 Docker 快速安装和配置神舟通用数据库，并使用 Zabbix 监控神舟通用数据库。

安装步骤参考：

[https://blog.csdn.net/weixin\\_46603727/article/details/131300046](https://blog.csdn.net/weixin_46603727/article/details/131300046)

docker 镜像下载：[https://pan.baidu.com/s/1-W\\_tuGk4waewNhr6C8Z00g](https://pan.baidu.com/s/1-W_tuGk4waewNhr6C8Z00g)

提取码：9572

### 1. 安装步骤

1.1. 安装 docker，下载神舟通用数据库镜像 shentong\_342\_163\_x86\_64bit.zip 并上传至服务器。

```
yum install docker
```

```
systemctl start docker
```

```
unzip shentong_342_163_x86_64bit.zip
```

```
docker load -i shentong_342_163_x86_64bit
```

导入完成后，可以使用 `docker images` 查看导入的镜像。结果显示如下：

```
[root@cenos7 shentong]# ls
shentong_342_163_x86_64bit.zip
[root@cenos7 shentong]# unzip shentong_342_163_x86_64bit.zip
Archive: shentong_342_163_x86_64bit.zip
  inflating: shentong_342_163_x86_64bit
[root@cenos7 shentong]# docker load -i shentong_342_163_x86_64bit
174f56854903: Loading layer [=====>] 211.7 MB/211.7 MB
6d1a884f63aa: Loading layer [=====>] 1.451 GB/1.451 GB
41567f6995e1: Loading layer [=====>] 6.656 kB/6.656 kB
04299a68c8bb: Loading layer [=====>] 4.608 kB/4.608 kB
c937bcb63a5: Loading layer [=====>] 2.56 kB/2.56 kB
ce384df97985: Loading layer [=====>] 11.78 kB/11.78 kB
17f04db55620: Loading layer [=====>] 10.24 kB/10.24 kB
b2309a771903: Loading layer [=====>] 1.451 GB/1.451 GB
Loaded image: shentong_342_163_x86_64bit:latest
[root@cenos7 shentong]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker.io/pingcap/tidb   latest             bd253ee807c6       6 weeks ago        393 MB
shentong_342_163_x86_64bit  latest             38f7dd5ab67b       11 months ago      3.1 GB
[root@cenos7 shentong]# ls
shentong_342_163_x86_64bit
[root@cenos7 shentong]#
```

## 1.2. 初始化容器。

`date -s "2023/06/20" #该镜像有试用授权期限，因此需要调整日期`

`docker run -d -p 2003:2003 --name Oscar shentong_342_163_x86_64bit`

#创建并运行容器，将会在后台启动一个名为 oscar 的容器，并将容器的 2003 端口映射到宿主机的 2003 端口，将数据目录映射到/opt/ShenTong/。

`docker ps -a #查看容器运行状态`

```
[root@cenos7 shentong]# docker run -d -p 2003:2003 --name oscar shentong_342_163_x86_64bit
8bcc5e27d74f85789c7cf6816df9ce710c1cd0b0b1e2c013aab55910359a0b48
[root@cenos7 shentong]# docker ps -a
CONTAINER ID        IMAGE                COMMAND             CREATED             STATUS              PORTS               NAMES
8bcc5e27d74f      shentong_342_163_x86_64bit  "bash /opt/s.sh"   5 seconds ago      Up 3 seconds       0.0.0.0:2003->2003/tcp, 5555/tcp  oscar
[root@cenos7 shentong]#
```

正常启动后能看到容器里启动了 /opt/ShenTong/bin/oscar 和 /opt/ShenTong/bin/oscaragent 进程。如果没有/opt/ShenTong/bin/osca 进程，可能是授权到期了导致启动失败，需要调整本机时间。

```
[root@cenos7 ~]# docker ps -a
CONTAINER ID        IMAGE                COMMAND             CREATED             STATUS              PORTS               NAMES
1c001e022f37      shentong_342_163_x86_64bit  "bash /opt/s.sh"   Less than a second ago  Up 2 minutes       0.0.0.0:2003->2003/tcp, 5555/tcp  Oscar
[root@cenos7 ~]# docker exec -it Oscar bash
[root@1c001e022f37 /]# ps -ef
UID          PID    PPID    C   STIME TTY          TIME CMD
root         1      0   0  16:01 ?        00:00:00 bash /opt/s.sh
root         9      1   1  16:01 ?        00:00:01 /opt/ShenTong/bin/oscar -o nor
root        46     1   0  16:01 ?        00:00:00 /opt/ShenTong/bin/oscaragent -
root        48     1   0  16:01 ?        00:00:00 tail -f /dev/null
root        60     0   0  16:02 ?        00:00:00 bash
root        77     0   0  16:04 ?        00:00:00 bash
root        90     77   0  16:04 ?        00:00:00 ps -ef
[root@1c001e022f37 /]#
```

默认用户名: SYSDBA, 密码: szoscar55, 库: OSRDB, schema: SYSDBA

容器内可使用/opt/ShenTong/bin/isql -U SYSDBA/szoscar55 -d osrdb 命令连接数

数据库。

```
[root@cenos7 opt]# docker exec -it Oscar bash
[root@1c001e022f37 /]# /opt/ShenTong/bin/isql -U SYSDBA/szoscar55 -d osrdb
conversion between SQL_ASCII and UTF8 is not supported
Welcome to isql 7.1.20220902 the ShenTongDB interactive terminal.
Type:  COPYRIGHT for distribution terms
       HELP for help with SQL commands
       ? for help on internal commands
       ! to run system commands
       EXIT to quit

SQL> exit
[root@1c001e022f37 /]# exit
exit
[root@cenos7 opt]# █
```

测试连接并执行命令：（需要指定连接客户端的字符集为 UTF-8）

```
export LANG=en_US.UTF-8
```

```
export LC_ALL=en_US.UTF-8
```

```
/opt/ShenTong/bin/isql -U SYSDBA/szoscar55 -d osrdb sysdba
```

```
select version;
```

```
[root@1c001e022f37 /]# export LANG=en_US.UTF-8
[root@1c001e022f37 /]# export LC_ALL=en_US.UTF-8
[root@1c001e022f37 /]#
[root@1c001e022f37 /]# /opt/ShenTong/bin/isql -U SYSDBA/szoscar55 -d osrdb sysdba
isql: warning: extra command-line argument "sysdba" ignored
Welcome to isql 7.1.20220902 the ShenTongDB interactive terminal.
Type:  COPYRIGHT for distribution terms
       HELP for help with SQL commands
       ? for help on internal commands
       ! to run system commands
       EXIT to quit

SQL> select version;

                                VERSION
-----
神通数据库7.0非正式授权版7.0.8.191204 for Linux(x86 64bit) (65535 connections) (license invalid after 133 days)
(1 row)

SQL> █
```

## 2. 准备监控神舟通用数据库

2.1. 将神舟通用数据库提供的 python 模块及需要的 lib 文件复制到容器外的/opt 目录下。

```
docker cp Oscar:/opt/ShenTong/STPython/ /opt
```

```
docker cp Oscar:/opt/ShenTong/bin/libaci.so /lib64/
```

```
ldconfig -v | grep libaci #加载 lib
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
[root@cenos7 shentong]# docker cp 0scar:/opt/ShenTong/STPython/ /opt
[root@cenos7 shentong]# cd /opt/
[root@cenos7 opt]# ls
ShenTong STPython
[root@cenos7 opt]# ls STPython/
Python27 Python27-ucs4 Python35 Python36 Python38
[root@cenos7 opt]# ls STPython/Python35
STPython-2.0.5-cp35-cp35m-linux_x86_64.whl STPython-2.0.7-cp35-cp35m-linux_x86_64.whl
[root@cenos7 opt]#
```

2.2. 安装 python3.5 用于监控神舟通用数据库。

安装步骤略

2.3. 安装连接神舟通用数据库用的 python 模块。

```
python3 -m pip install --no-index --find-links=/opt/STPython/Python35
```

STPython

```
[root@cenos7 opt]# python3 -m pip install --no-index --find-links=/opt/STPython/Python35 STPython
Looking in links: /opt/STPython/Python35
Processing ./STPython/Python35/STPython-2.0.7-cp35-cp35m-linux_x86_64.whl
Installing collected packages: STPython
Successfully installed STPython-2.0.7
[root@cenos7 opt]#
```

2.4. 创建监控用的脚本。

在 zabbix 的外部检查目录下创建 2 个监控用的脚本，我使用的是 /itops/zabbix/share/zabbix/externalscripts/目录，需要根据实际情况进行修改。

```
vi /itops/zabbix/share/zabbix/externalscripts/pyshentongdb
```

```
#!/usr/bin/bash

/usr/bin/python3 /itops/zabbix/share/zabbix/externalscripts/pyshentongdb.py

--username $1 --password $2 --address $3 --port $4 --database $5 $6 $7 $8
```

```
vi /itops/zabbix/share/zabbix/externalscripts/pyshentongdb.py
```

```
#!/usr/bin/env python3

# coding: utf-8
```

```
# author: cxh

import argparse

import STPython

import inspect

import json

import re

version = 0.2

class Checks(object):

    def version(self):

        """查数据库版本"""

        sql = "select version"

        self.cur.execute(sql)

        res = self.cur.fetchall()

        for i in res:

            print(i[0])

class Main(Checks):
```

```
def __init__(self):

    parser = argparse.ArgumentParser()

    parser.add_argument('--username')

    parser.add_argument('--password')

    parser.add_argument('--address')

    parser.add_argument('--port')

    parser.add_argument('--database')

    subparsers = parser.add_subparsers()

    for name in dir(self):

        if not name.startswith("_"):

            p = subparsers.add_parser(name)

            method = getattr(self, name)

            argnames = inspect.getargspec(method).args[1:]

            for argname in argnames:

                p.add_argument(argname)

            p.set_defaults(func=method, argnames=argnames)

    self.args = parser.parse_args()

def db_connect(self):

    a = self.args

    username = a.username
```

```
password = a.password

address = a.address

port = a.port

database = a.database

self.db = STPython.Connection(user=username,password=password,dsn=str(address)+'/'+str(port)+'/'+str(database))

self.cur = self.db.cursor()

def db_close(self):

    self.db.close()

def __call__(self):

    try:

        a = self.args

        callargs = [getattr(a, name) for name in a.argnames]

        self.db_connect()

        try:

            return self.args.func(*callargs)

        finally:

            self.db_close()

    except Exception as err:
```



```
print("0")

print(str(err))

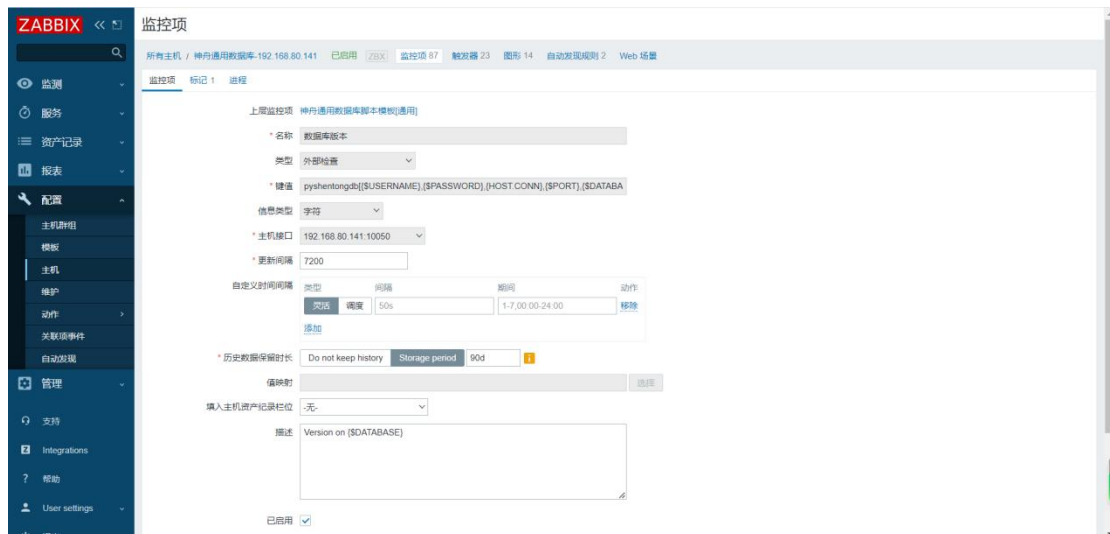
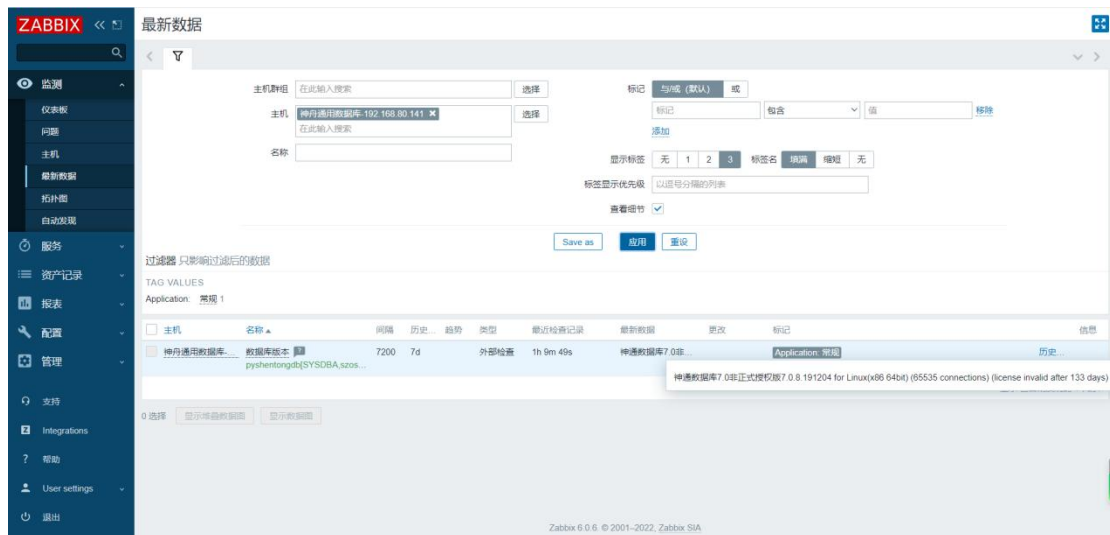
if __name__ == "__main__":

    main = Main()

    main()
```

因时间关系，本文章仅展示了神舟通用数据库的一个监控指标添加例子，后续需要神舟通用数据库的 DBA 提供常用的监控相关的 SQL 语句进行监控指标的扩充。

监控结果如下：



附：

数据库版本监控指标配置示例：

名称：数据库版本

类型：外部检查

键值：

```
pyshentongdb[{$USERNAME},{PASSWORD},{HOST.CONN},{PORT},{DATABASE},version]
```

更新间隔：7200 （按需进行设置）

需要在模板上增加以下宏：

{USERNAME}：填写数据库的连接用户名

{PASSWORD}：填写数据库的连接密码

{PORT}：填写数据库的监听端口

{DATABASE}：填写数据库的实例名，本文是 osrdb

以上就是本期文章的全部内容。大家好，我是乐乐，专注运维技术研究与分享，想要了解更多 zabbix 开源监控使用技巧，欢迎关注[乐维社区](#)，同时 zabbix 使用问题也欢迎到乐维社区留言提问~

## 五十二、如何使用 Zabbix 监控 InfluxDB 数据库？

### 一、简介

InfluxDB 是一个时序数据库，旨在处理时间戳数据的高写入和查询负载。它是用 Go 编程语言编写的开源数据库，专门用于存储和查询时间序列数据，如指标、事件和日志。

InfluxDB 通常用于监控和可观测性、物联网应用和实时分析。它支持类似 SQL 的查询语言，并与各种数据可视化和监控工具集成。

作为一款流行的开源时序数据库，InfluxDB 常年在 DB-Engines TSDB rank 中位居首位，可见，InfluxDB 还是非常受欢迎的。

include secondary database models 42 systems in ranking, January 2024

Rank			DBMS	Database Model	Score		
Jan 2024	Dec 2023	Jan 2023			Jan 2024	Dec 2023	Jan 2023
1.	1.	1.	InfluxDB <span style="color: orange;">+</span>	Time Series, Multi-model <span style="font-size: small;">i</span>	27.56	-0.56	-1.80
2.	2.	<span style="color: green;">↑</span> 3.	Prometheus	Time Series	8.95	+0.61	+2.19
3.	3.	<span style="color: red;">↓</span> 2.	Kdb <span style="color: orange;">+</span>	Multi-model <span style="font-size: small;">i</span>	7.97	-0.31	+1.08
4.	4.	4.	Graphite	Time Series	5.27	-0.19	-1.44
5.	5.	5.	TimescaleDB	Time Series, Multi-model <span style="font-size: small;">i</span>	5.24	-0.05	+0.70
6.	6.	6.	DolphinDB	Time Series, Multi-model <span style="font-size: small;">i</span>	4.46	+0.55	+1.66
7.	7.	<span style="color: green;">↑</span> 9.	Apache Druid	Multi-model <span style="font-size: small;">i</span>	3.58	+0.24	+1.26
8.	8.	<span style="color: green;">↑</span> 10.	TDengine <span style="color: orange;">+</span>	Time Series, Multi-model <span style="font-size: small;">i</span>	3.43	+0.22	+1.33
9.	9.	<span style="color: red;">↓</span> 7.	RRDtool	Time Series	2.80	-0.01	+0.09
10.	10.	<span style="color: green;">↑</span> 12.	QuestDB <span style="color: orange;">+</span>	Time Series, Multi-model <span style="font-size: small;">i</span>	2.72	+0.42	+0.83
11.	<span style="color: green;">↑</span> 12.	<span style="color: red;">↓</span> 8.	OpenTSDB	Time Series	2.21	+0.16	-0.36
12.	<span style="color: red;">↓</span> 11.	<span style="color: red;">↓</span> 11.	GridDB <span style="color: orange;">+</span>	Time Series, Multi-model <span style="font-size: small;">i</span>	2.17	-0.06	+0.13
13.	13.	13.	Fauna	Multi-model <span style="font-size: small;">i</span>	1.96	+0.20	+0.19
14.	14.	<span style="color: green;">↑</span> 15.	VictoriaMetrics	Time Series	1.57	+0.11	+0.50

本教程将介绍基于 CentOS 7.5, docker 进行 InfluxDB 数据库的安装，并使用 Zabbix 对 InfluxDB 数据库进行监控。其中，Zabbix 版本为 6.0.6。

## 二、安装步骤

1、安装 docker 并拉取镜像。

```
yum install docker
```

```
systemctl start docker
```

```
docker search influxdb # 搜索镜像，如果搜索不到需要设置 docker 仓库源
```

```
[root@cenos7 ~]# docker search influxdb
INDEX      NAME                DESCRIPTION          STARS  OFFICIAL  AUTOMATED
docker.io  docker.io/influxdb  InfluxDB is an open source time series dat... 1839   [OK]
docker.io  docker.io/telegraf  Telegraf is an agent for collecting metric... 649   [OK]
docker.io  docker.io/chronograf Chronograf is a visualization tool for tim... 360   [OK]
docker.io  docker.io/tutum/influxdb InfluxDB image - DEPRECATED. See https://d... 223   [OK]
docker.io  docker.io/arm32v7/influxdb InfluxDB is an open source time series dat... 28
docker.io  docker.io/bitnami/influxdb InfluxDB is an open source time series dat... 13
docker.io  docker.io/arm64v8/influxdb InfluxDB is an open source time series dat... 10
docker.io  docker.io/prom/influxdb-exporter A server that accepts InfluxDB metrics via... 8
docker.io  docker.io/rapidfort/influxdb RapidFort optimized, hardened image for In... 8
docker.io  docker.io/influxdb/influxdb InfluxDB is an open source time series dat... 7
docker.io  docker.io/sillydong/influxdb-ui web ui for influxdb query 7
docker.io  docker.io/matisq/influxdb TIG Stack - InfluxDB 1 [OK]
docker.io  docker.io/amd64/influxdb InfluxDB is an open source time series dat... 0
docker.io  docker.io/bitnami/influxdb-relay-archived A copy of the container images of the depr... 0
docker.io  docker.io/bitnamicharts/influxdb InfluxDB is an open source time series dat... 0
docker.io  docker.io/forestsctr/influxdb-docker-collect collect docker stats from mesos and transm... 0 [OK]
docker.io  docker.io/hephy/influxdb InfluxDB is an open source time series dat... 0
docker.io  docker.io/illusiveman/influxdb-to-s3 Backups influxdb to s3 0
docker.io  docker.io/map3/influxdb InfluxDB is an open source time series dat... 0
docker.io  docker.io/monasca/influxdb-init InfluxDB is an open source time series dat... 0
docker.io  docker.io/objectscale/influxdb InfluxDB is an open source time series dat... 0
docker.io  docker.io/objectscale/influxdb-operator InfluxDB is an open source time series dat... 0
docker.io  docker.io/simonsobs/ocs-influxdb-publisher-agent InfluxDB Publisher Agent 0
docker.io  docker.io/ustclug/influxdb influxdb used by USTC LUG 0 [OK]
docker.io  docker.io/vulhub/influxdb InfluxDB is an open source time series dat... 0
[root@cenos7 ~]#
```

```
docker pull influxdb # 拉取镜像
```

```
docker images # 查看拉取的镜像
```

```
[root@cenos7 ~]# docker pull influxdb
Using default tag: latest
Trying to pull repository docker.io/library/influxdb ...
latest: Pulling from docker.io/library/influxdb
2f44b7a888fa: Pull complete
60b879f7f0b9: Pull complete
391f2a951491: Pull complete
bc0f8d17c9bc: Pull complete
1eea01495215: Pull complete
c74718bd32e2: Pull complete
ab287bfba4b0: Pull complete
30f5135083ba: Pull complete
8acfea7e99e3: Pull complete
0c3fd5840e75: Pull complete
Digest: sha256:f3765ab8ae6c9fddebe36b95ce1a85097c92146573d1a809dfe7bed60920a2d2
Status: Downloaded newer image for docker.io/influxdb:latest
[root@cenos7 ~]# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
docker.io/influxdb  latest      7a1b63100937     2 weeks ago     378 MB
[root@cenos7 ~]#
```

2、初始化容器。

```
mkdir -p /data/influxdb #创建宿主机路径
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
docker run -dit --name influxdb --restart always -p 8086:8086 -v /data/influxdb:/var/lib/influxdb influxdb #创建并运行容器，映射 8086 端口，映射宿主机的/data/influxdb 到容器里/var/lib/influxdb influxdb 目录。
```

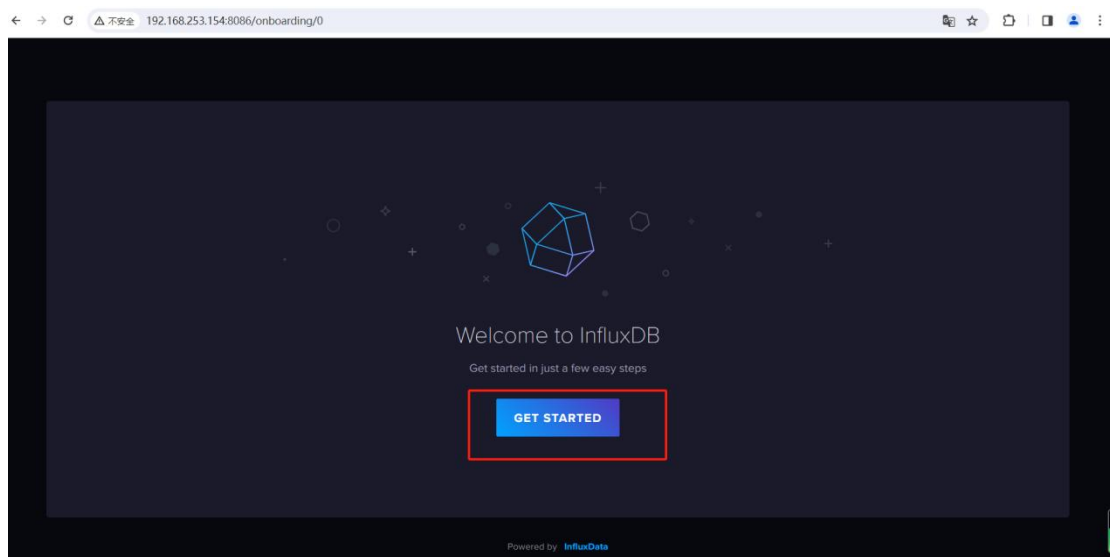
```
docker ps -a #查看容器运行状态
```

```
[root@cenos7 ~]# docker run -dit --name influxdb --restart always -p 8086:8086 -v /data/influxdb:/var/lib/influxdb influxdb
f32f6b8edb480741dc8a003890001339e1e0383067dad162b88c3c0588dc6d28
[root@cenos7 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS                               NAMES
f32f6b8edb48   influxdb "/entrypoint.sh in..." 8 seconds ago    Up 5 seconds    0.0.0.0:8086->8086/tcp    influxdb
[root@cenos7 ~]#
```

3、为 influxdb 创建一个只读权限的 token，参考

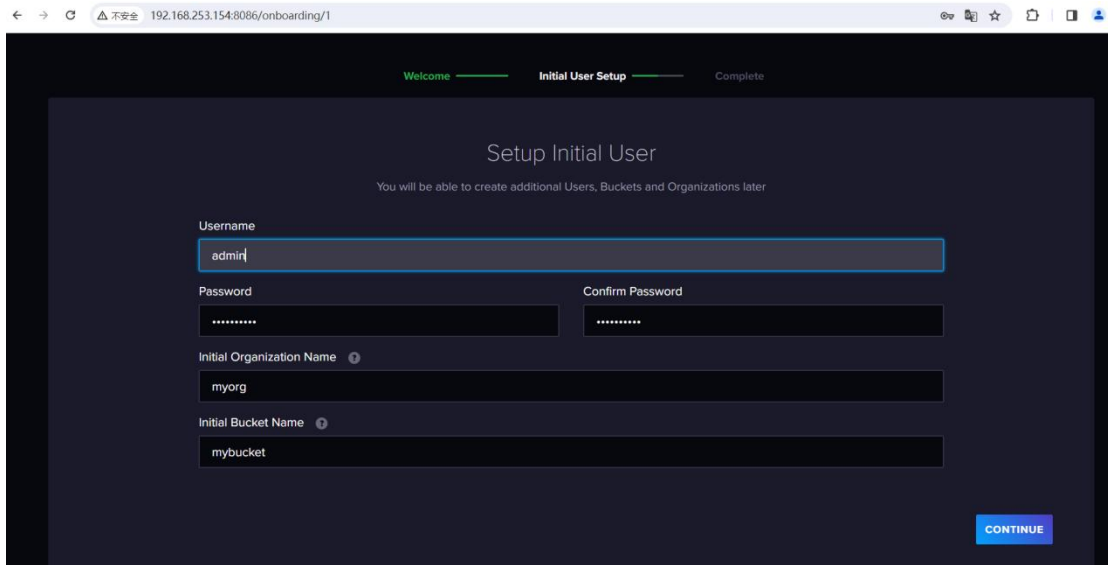
<https://docs.influxdata.com/influxdb/v2/admin/tokens/create-token/>

本次采用的操作方法是：使用浏览器访问 influxdb 的 8086 端口。按以下步骤进行操作：

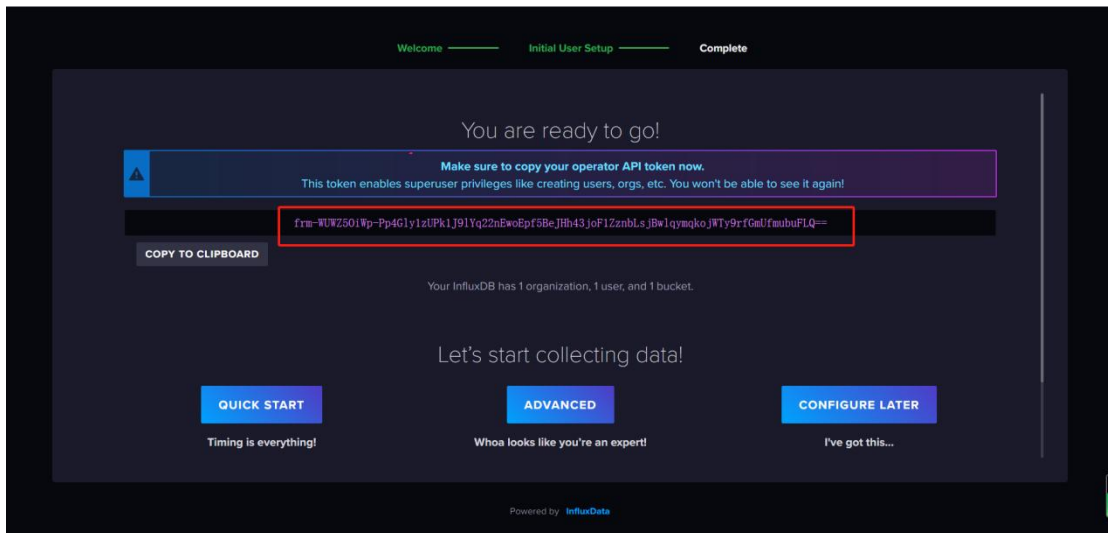


根据提示填写初始化信息：

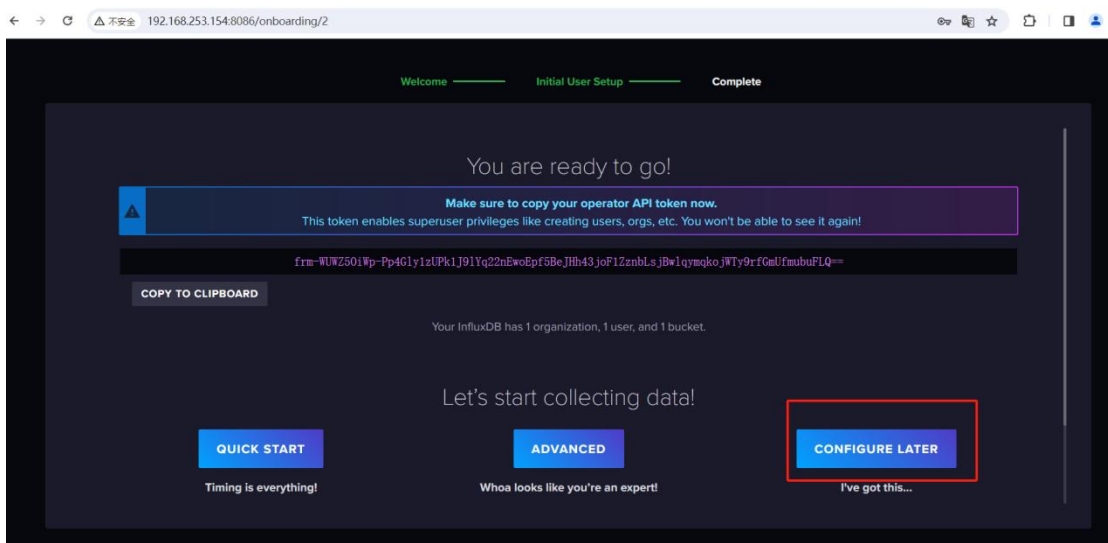
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



初始化后会提供管理员权限的 token，可直接使用或再创建一个只读 token：

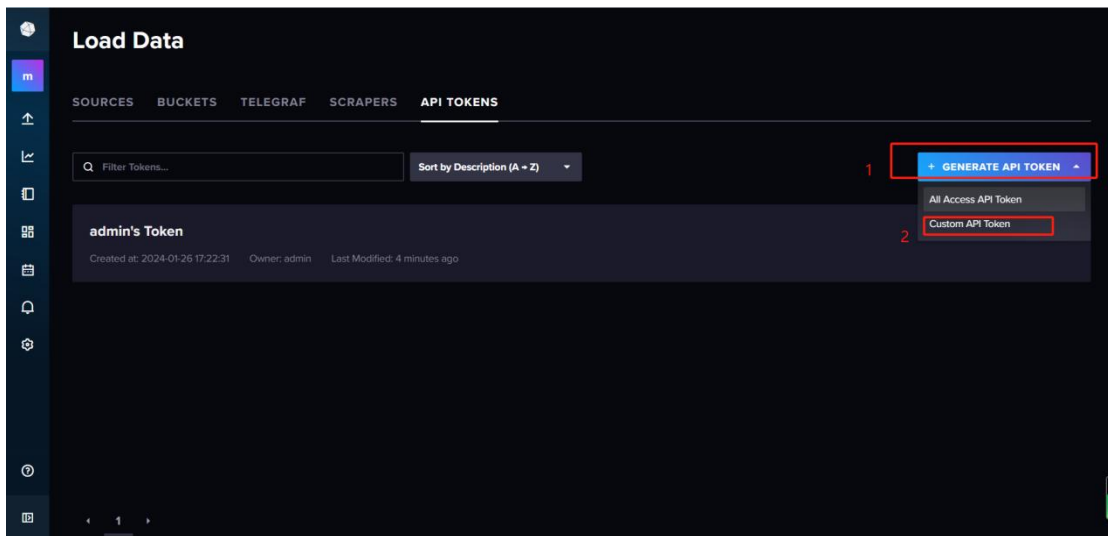
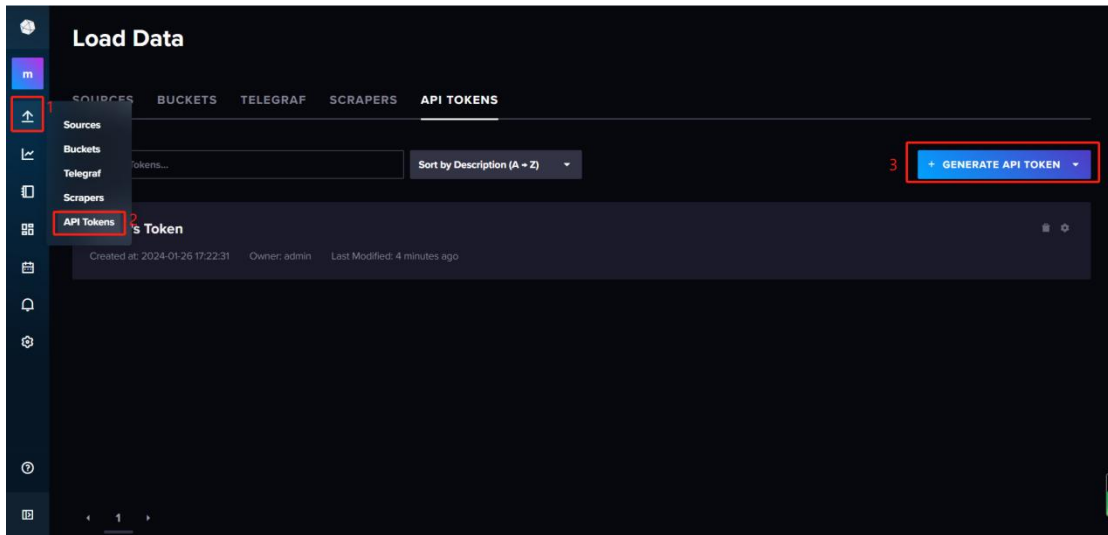


继续创建只读 token：

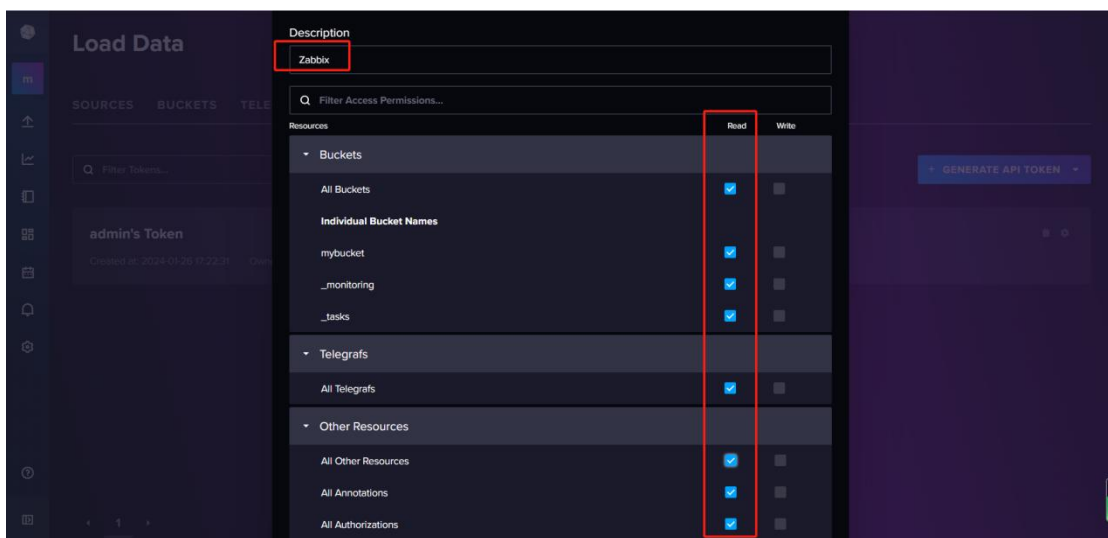


整理 by 乐维社区 (<https://forum.lwops.cn>)

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

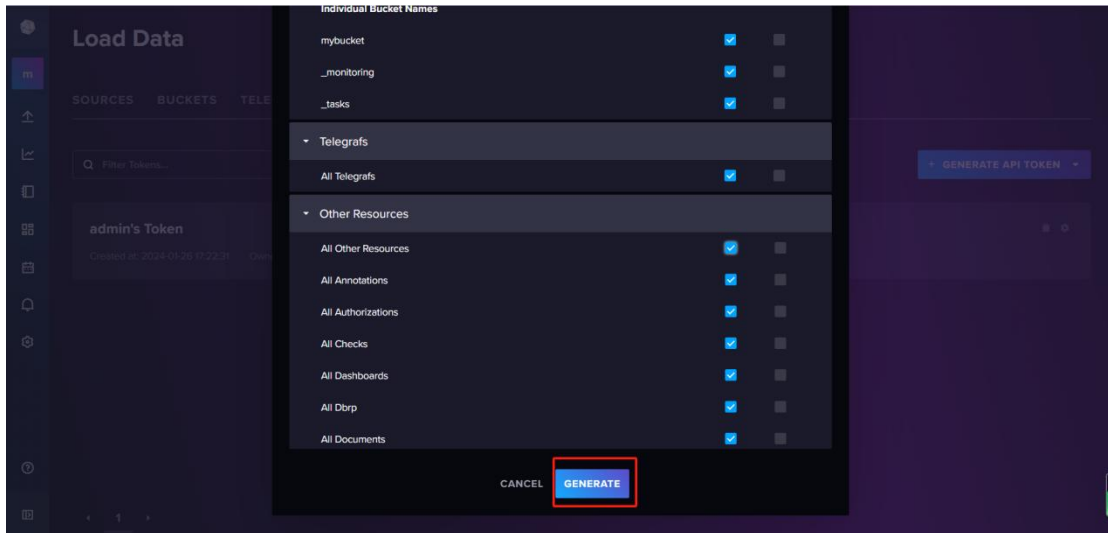


填写名字并选择只读权限：

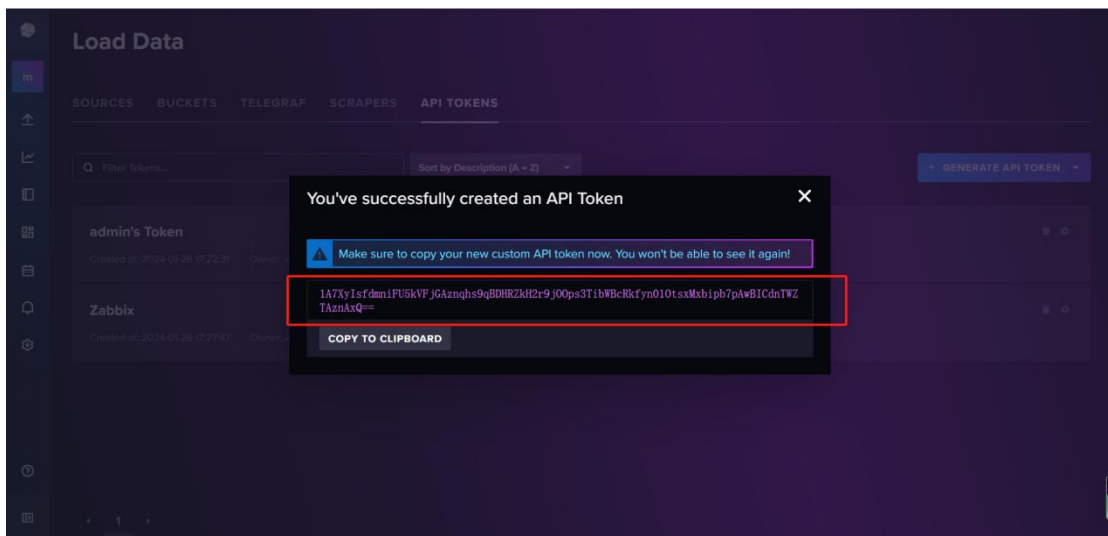


点击确定：

本文档为样章，完整版文档请添加乐乐（lerwee）获取



复制生成的 token:



### 三、监控 InfluxDB

#### 1、导入监控模板

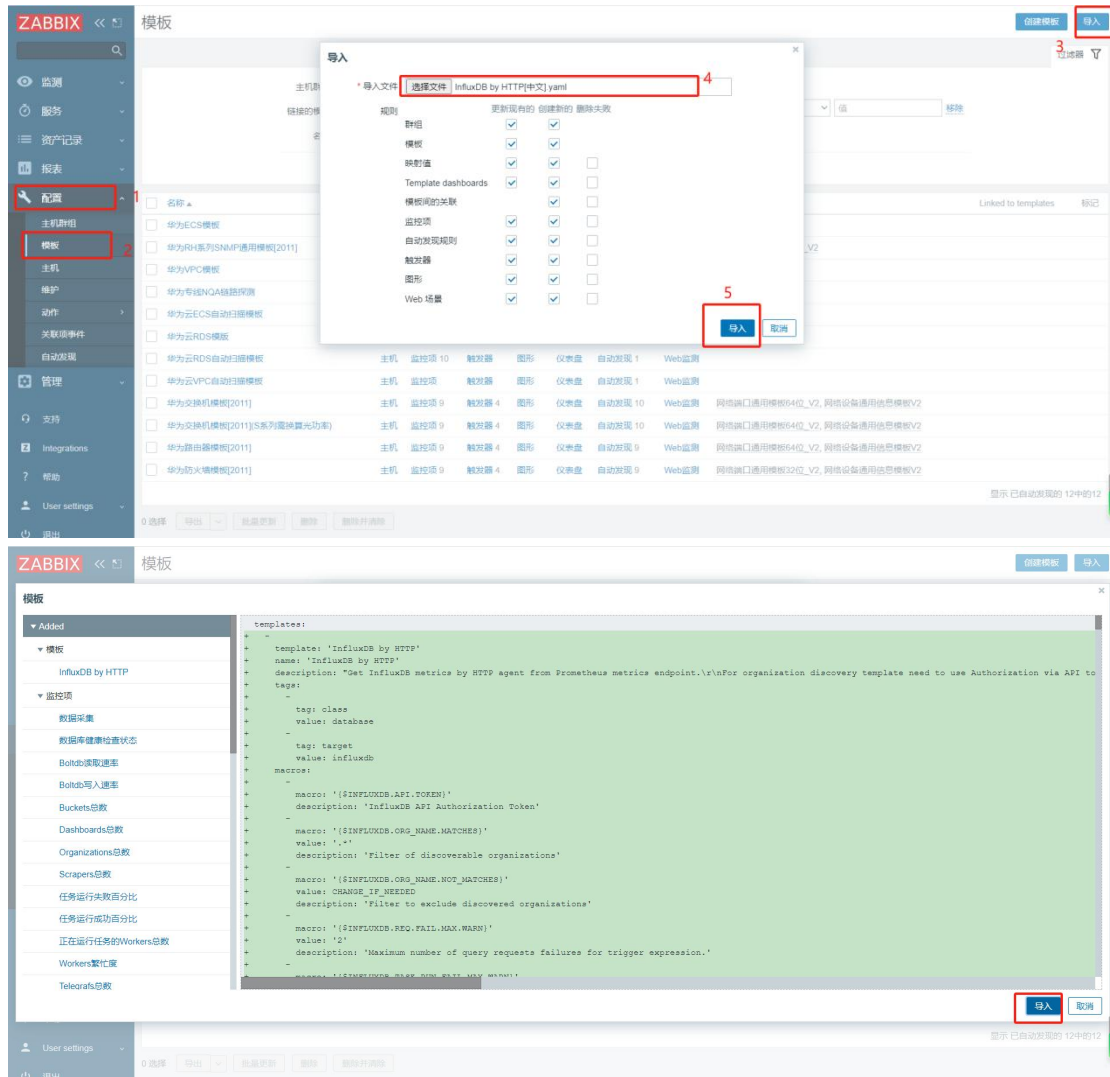
[InfluxDB by HTTP\[中文\].yaml](#)

也可使用 Zabbix 官方提供的监控模板:

<https://www.zabbix.com/cn/integrations/influxdb>



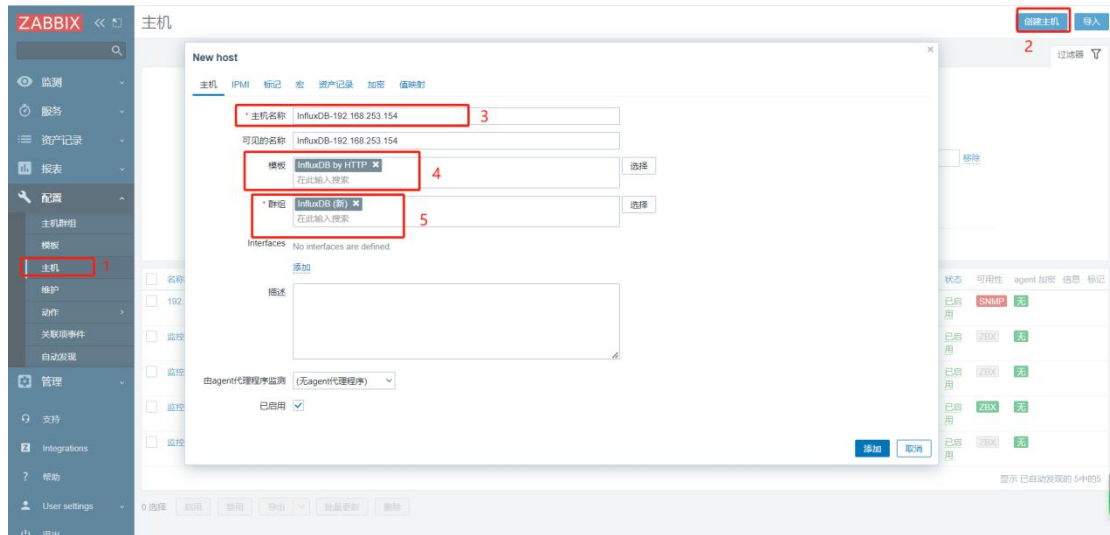
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



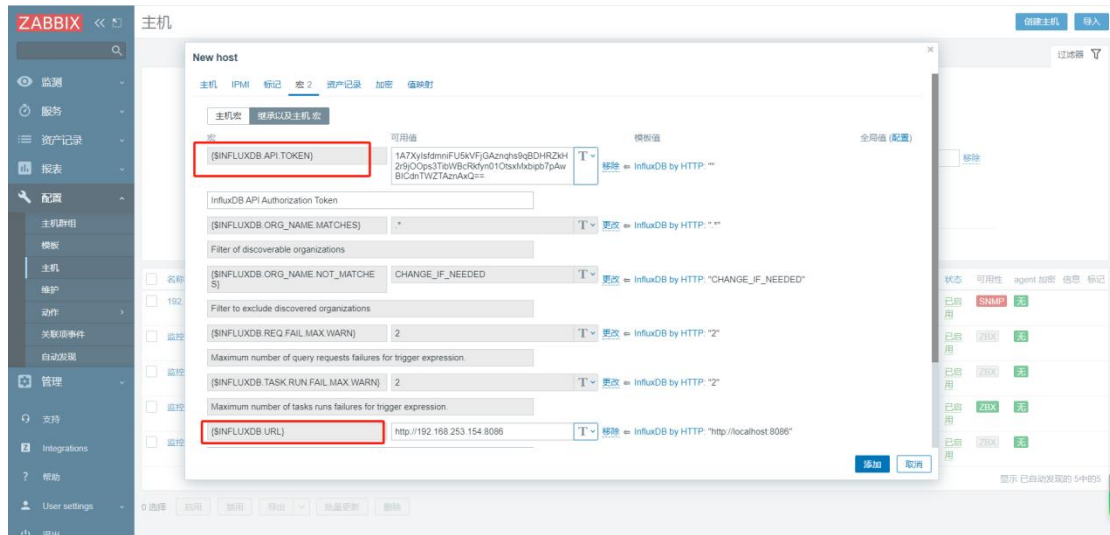
## 2、添加 InfluxDB 监控

点击配置->主机->创建主机，填写主机名称，选择刚刚导入的 InfluxDB 监控模板，设置一个群组。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



点击宏，点击“继承以及主机宏”，填写相关信息：



宏说明：

{\$(INFLUXDB.API.TOKEN)}: 填写 InfluxDB 的 token。

{\$(INFLUXDB.ORG\_NAME.MATCHES)}: 表示需要监控的 ORG 名称，默认.\*表示监控全部 ORG。

{\$(INFLUXDB.ORG\_NAME.NOT\_MATCHES)}: 表示不需要监控的 ORG 名称，默认 CHANGE\_IF\_NEEDED。

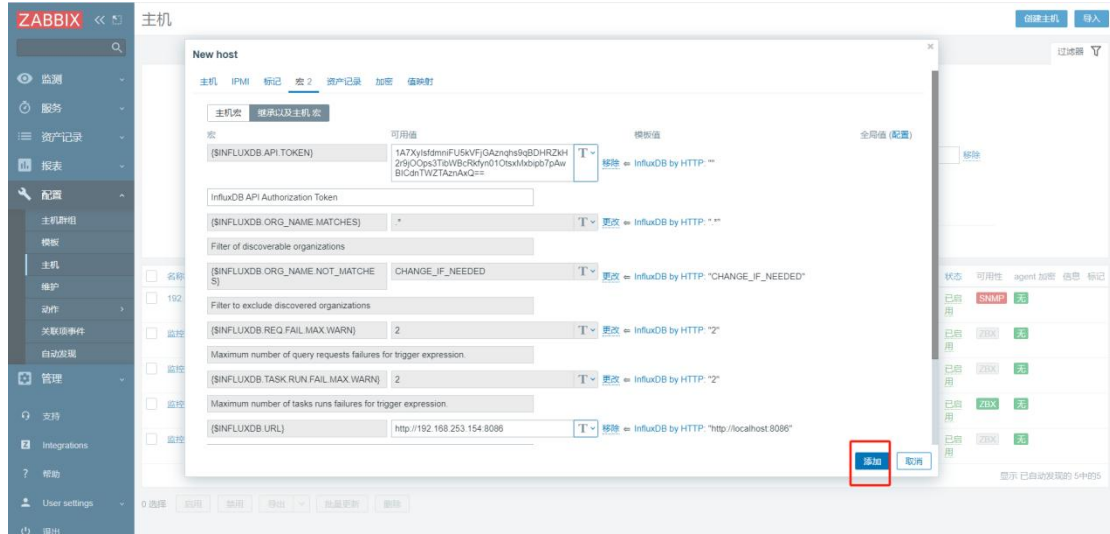
{\$(INFLUXDB.REQ.FAIL.MAX.WARN)}: 设置请求失败数量告警阈值，默认是 2。

{\$(INFLUXDB.TASK.RUN.FAIL.MAX.WARN)}: 设置任务失败数量告警阈值，默认是 2。

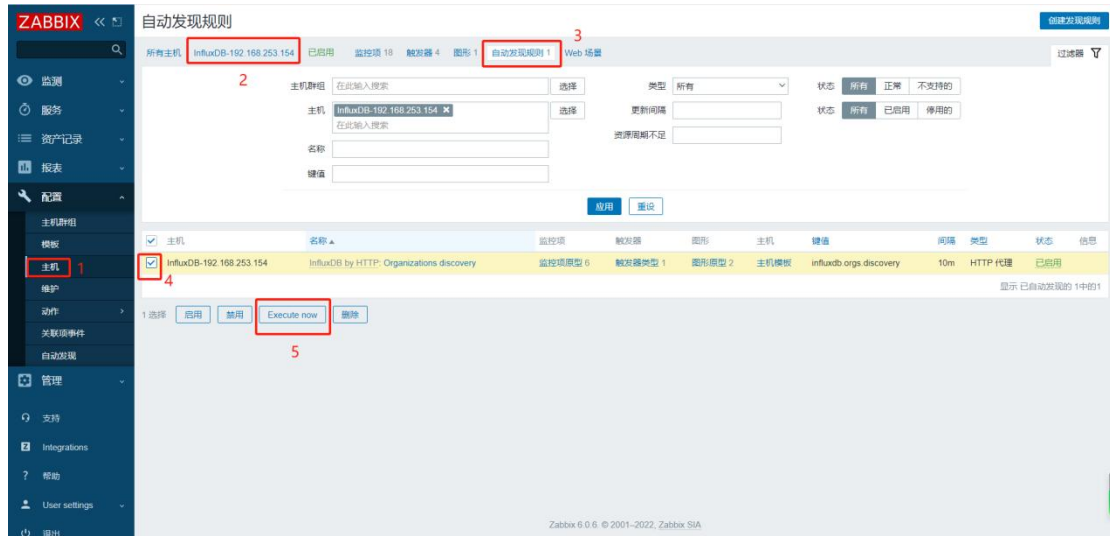
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

{\${INFLUXDB.URL}}: 设置 InfluxDB 的 URL，默认: http://IP 地址:8086

信息填写完成后，点击添加按钮即可完成监控主机的添加。



添加完成后，可在主机管理界面，对自动发现规则触发立即执行，使其快速创建自动发现的监控项。



查看监控数据，监控完成。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

The screenshot shows the Zabbix '最新数据' (Latest Data) page. The left sidebar contains navigation options like '仪表盘', '问题', '主机', '最新数据', '拓扑图', '自动发现', '服务', '资产记录', '报表', '配置', '管理', '支持', 'Integrations', '帮助', 'User settings', and '退出'. The main content area is titled '最新数据' and includes a search bar, filter options, and a table of metrics. A red box highlights the '应用' (Apply) button. The table lists various metrics for the host 'Inf fluxDB-192.168.253.154', including 'Boots写入速率', 'Boots读取速率', 'Buckets总数', 'Dashboards总数', 'Organizations总数', 'Scrapers总数', 'Telegraf plugins总数', 'Telegrafs总数', 'Tokens总数', 'Users总数', 'Workers繁忙度', and '正在运行任务的Workers总数'. The table also shows the last check time, latest data value, change, and tags for each metric.

主机	名称	最近检查记录	最新数据	更改	标记	信息
Inf fluxDB-192.168.253.1...	Boots写入速率	1m 14s	0		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Boots读取速率	1m 14s	0.1746		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Buckets总数	3m 14s	3		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Dashboards总数	3m 14s	0		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Organizations总数	3m 14s	1		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Scrapers总数	3m 14s	0		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Telegraf plugins总数				Application: 常规	图形
Inf fluxDB-192.168.253.1...	Dashboards总数	27m 12s	0		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Organizations总数	27m 12s	1		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Scrapers总数	27m 12s	0		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Telegraf plugins总数				Application: 常规	图形
Inf fluxDB-192.168.253.1...	Telegrafs总数	27m 12s	0		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Tokens总数	27m 12s	2		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Users总数	27m 12s	1		Application: 常规	图形
Inf fluxDB-192.168.253.1...	Workers繁忙度	1m 12s	0 %		Application: 常规	图形
Inf fluxDB-192.168.253.1...	[myorg] 每秒查询响应失败的字节大小				Application: 查询	图形
Inf fluxDB-192.168.253.1...	[myorg] 每秒查询响应成功的字节大小				Application: 查询	图形
Inf fluxDB-192.168.253.1...	[myorg] 每秒查询失败的请求数				Application: 查询	图形
Inf fluxDB-192.168.253.1...	[myorg] 每秒查询成功的请求数				Application: 查询	图形
Inf fluxDB-192.168.253.1...	[myorg] 每秒查询请求失败的字节大小				Application: 查询	图形
Inf fluxDB-192.168.253.1...	[myorg] 每秒查询请求成功的字节大小				Application: 查询	图形
Inf fluxDB-192.168.253.1...	任务运行失败百分比				Application: 常规	图形
Inf fluxDB-192.168.253.1...	任务运行成功百分比				Application: 常规	图形
Inf fluxDB-192.168.253.1...	数据库健康检查状态	27m 11s	Ok (1)		Application: 健康状态	图形
Inf fluxDB-192.168.253.1...	数据库版本	27m 12s	v2.7.5		Application: 常规	历史...
Inf fluxDB-192.168.253.1...	数据库集				Application: 数据库	历史...
Inf fluxDB-192.168.253.1...	正在运行任务的Workers总数	1m 12s	0		Application: 常规	图形
Inf fluxDB-192.168.253.1...	运行时长	1m 12s	1h 9m 36s	+2m 217.06...	Application: 常规	图形

## 五十三、更多.....

# 云监控

## 七十七、通过 zabbix 监控华为云 RDS

通过 zabbix 监控华为云 RDS 思路通过华为云 RDS 实例列表接口获取 RDS 实例，用于资源的自动发现通过华为云 CES 获取监控数据准备工用于访问华为云的 AK/SKpython3 环境用于调用华为云 api 的相关模块依...

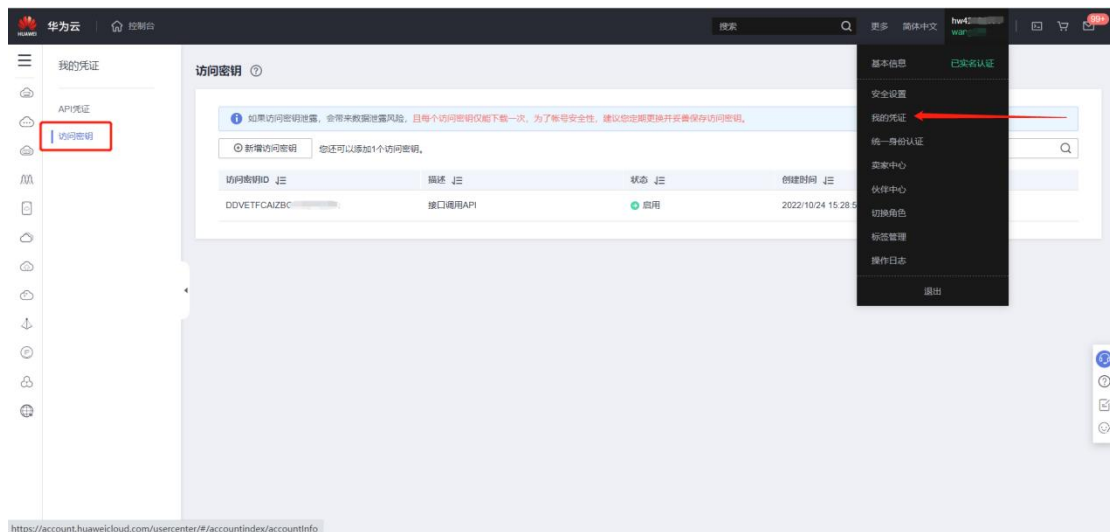
### 一、思路

通过华为云 RDS 实例列表接口获取 RDS 实例，用于资源的自动发现

通过华为云 CES 获取监控数据准

### 二、准备工作

用于访问华为云的 AK/SK



python3 环境

用于调用华为云 api 的相关模块依赖

<https://github.com/huaweicloud/huaweicloud-sdk-python-v3>

### 三、编写实例自动发现脚本

```
#!/bin/python3

# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials

from huaweicloudsdrds.v3.region.rds_region import RdsRegion

from huaweicloudsdkcore.exceptions import exceptions

from huaweicloudsdrds.v3 import *

import json

if __name__ == "__main__":

    #这里填访问密钥

    ak = ""

    sk = ""

    credentials = BasicCredentials(ak, sk) \

    client = RdsClient.new_builder() \

        .with_credentials(credentials) \

        .with_region(RdsRegion.value_of("cn-north-4")) \

        .build()
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
try:

    request = ListInstancesRequest()

    response = client.list_instances(request)

    response = json.loads(str(response))

    #拼接成用于自动发现的 json 数据, id 为 RDS 的实例 ID, name 为 RDS 的名称

    r = []

    for i in response['instances']:

        buf = {'#INSTANCE':i['id'],'#INSTANCE_NAME':i['name']}

        r.append(buf)

    print(json.dumps({"data":r}))

except exceptions.ClientRequestException as e:

    print(e.status_code)

    print(e.request_id)

    print(e.error_code)

    print(e.error_msg)
```

## 四、编写监控数据获取脚本

```
#!/bin/python3

# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials

from huaweicloudsdkces.v1.region.ces_region import CesRegion
```

```
from huaweicloudsdkcore.exceptions import exceptions

from huaweicloudsdkces.v1 import *

import json

import time

import sys

#传入的监控项

key = sys.argv[1]

#请求时间戳

from_time = int(round(time.time() * 1000)-300000)

to_time = int(round(time.time() * 1000))

if __name__ == "__main__":

    #这里填访问密钥

    ak = ""

    sk = ""

    credentials = BasicCredentials(ak, sk) \

    client = CesClient.new_builder() \

        .with_credentials(credentials) \

        .with_region(CesRegion.value_of("cn-north-4")) \

        .build()
```



try:

```
request = BatchListMetricDataRequest()

listDimensionsMetrics = [

    MetricsDimension(

        name="rds_cluster_id",

        #这里是实例 ID, 可改为变量, 模板宏即{HOST.HOST}

        value=""

    )

]

listMetricsbody = [

    MetricInfo(

        namespace="SYS.RDS",

        metric_name=key,

        dimensions=listDimensionsMetrics

    )

]

request.body = BatchListMetricDataRequestBody(

    to=to_time,

    _from=from_time,

    filter="max",

    period="1",

    metrics=listMetricsbody
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
)  
  
response = client.batch_list_metric_data(request)  
  
res = json.loads(str(response))  
  
instance = json.dumps(res)  
  
instances = json.loads(instance)['metrics']  
  
for i in instances:  
  
    #取最新的一个值  
  
    print(i['datapoints'][0]['max'])  
  
    #print(i)  
  
except exceptions.ClientRequestException as e:  
  
    print(0)  
  
    #print(e.request_id)  
  
    #print(e.error_code)  
  
    #print(e.error_msg)
```

## 五、新增模板

新增自动发现规则，用于实例自动发现

## 自动发现规则

所有模板 / 华为云RDS自动扫描模板 自动发现清单 / 实例名称自动发现 监控项原型 触发器类型 图形原型 主机模板 1

自动发现规则 进程 LLD macros 过滤器 覆盖

\* 名称

类型

\* 键值

\* 更新间隔

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

[添加](#)

\* 资源周期不足

描述

已启用

[更新](#) [克隆](#) [测试](#) [删除](#) [取消](#)

## 自动发现规则，链接监控数据获取模板

### 主机模板

所有模板 / 华为云RDS自动扫描模板 自动发现清单 / 实例名称自动发现 监控项原型 触发器类型 图形原型 主机模板 1

主机 标记 宏 资产记录 加密

\* 主机名称

可见的名称

模板 名称 动作

华为云RDS模板 [取消链接](#)

[选择](#)

\* 群组  [选择](#)

组模板  [移除](#)

[添加](#)

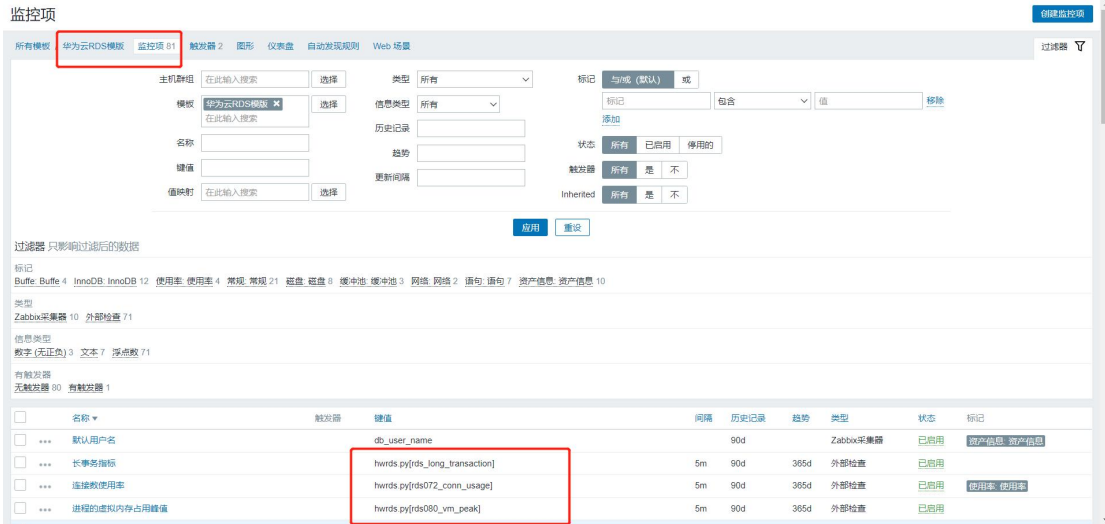
Interfaces  Inherit  习惯

启用新的

Discover

[更新](#) [克隆](#) [删除](#) [取消](#)

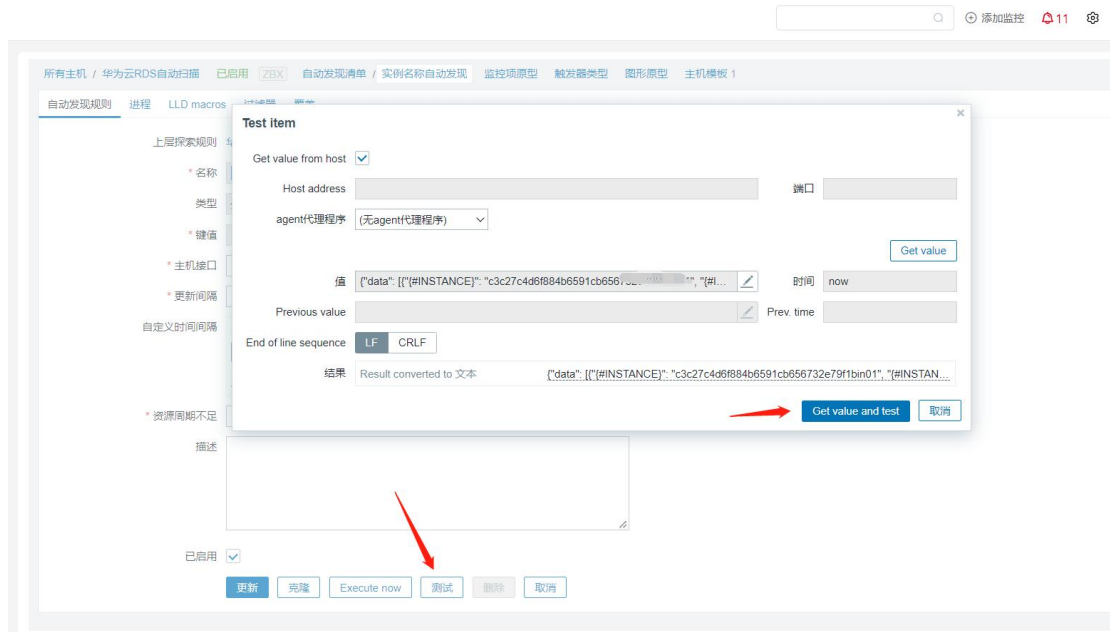
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



## 六、验证

验证是否可以正常发现实例

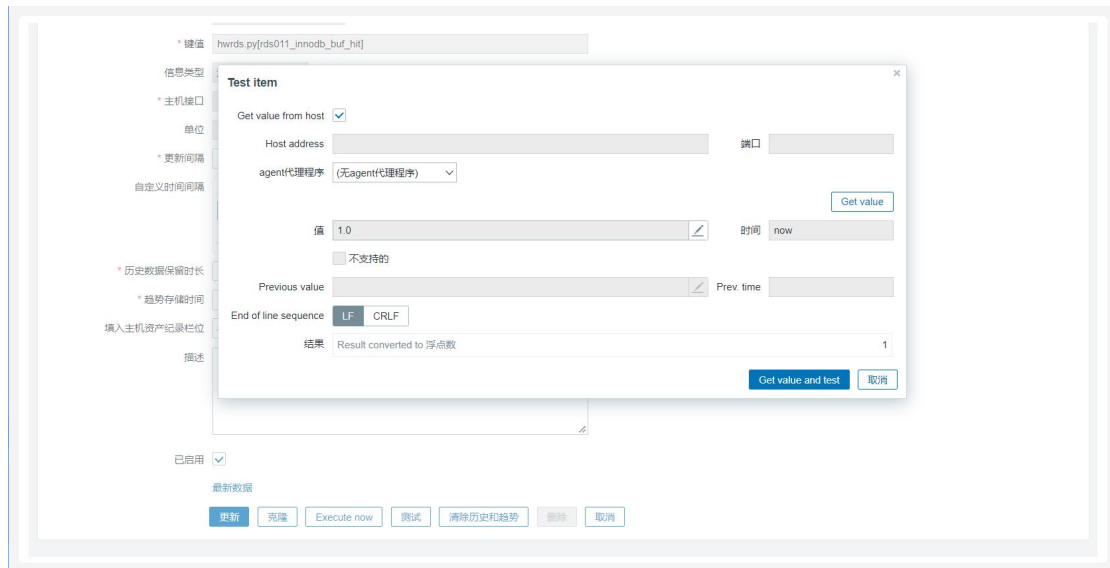
新增主机，链接华为云 RDS 自动扫描模板后，点自动发现规则，可以正常获取到数据



验证监控数据是否获取正常

点击发现出来实例，测试监控项可以正常获取到数据

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



## 七十八、腾讯云 NAT 网关监控

### 1. NAT 网关介绍

NAT 网关 (NAT Gateway) 是一种支持 IP 地址转换服务，提供网络地址转换能力，主要包括 SNAT (Source Network Address Translation, 源网络地址转换) 和 DNAT (Destination Network Address Translation, 目的网络地址转换) 能力，可为私有网络 (VPC) 内的资源提供安全、高性能的 Internet 访问服务。

NAT 网关分为公网 NAT 网关和私网 NAT 网关。公网 NAT 网关提供公网地址转换服务，而私网 NAT 网关提供私网地址转换服务。

NAT 网关支持高达 99.99% 的高可用性、5Gbps 的带宽以及 1000 万以上的并发连接数，其典型应用场景如下：

1. 大带宽、高可用的公网出口服务，例如：网络爬虫，访问 Internet 公共服务等。
2. 安全的公网出口服务，例如：云服务器需要与公网通信，但出于安全性考虑，不希望云服务器绑定公网 IP。

### 2. 监控前准备

本次监控方式通过 zabbix 外部检查方式自动发现实例，监控项数据通过 zabbix 采集器方式生成。将下面脚本上传至外部检查路径下。

部署路径有差异，根据实际环境的外部检查路径放至脚本。

```
#!/usr/bin/python3
import json
```

```
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.vpc.v20170312 import vpc_client, models
import sys
from os import popen
from zabbix import ZabbixMetric, ZabbixSender
akey = sys.argv[1]
ckey = sys.argv[2]
region = sys.argv[3]
ZABBIX_SEND_ADDR = '/itops/zabbix/bin/zabbix_sender'
ZABBIX_SERVER_IP = '127.0.0.1'
ZABBIX_PORT = '10051'
def get_data(offset):
    params = {
        "Limit": 100,
        "Offset": offset
    }
    req.from_json_string(json.dumps(params))
    # 返回的 resp 是一个 DescribeNatGatewaysResponse 的实例，与请求对象对应
    resp = client.DescribeNatGateways(req)
    # 输出 json 格式的字符串回包
    res = resp.to_json_string()
    data = json.loads(res)
    disklist = data['NatGatewaySet']
    # 获取记录总条目数
    totalcount = int(data['TotalCount'])
    return totalcount,disklist
try:
    # 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey
    # 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
    cred = credential.Credential(akey,ckey)
    # 实例化一个 http 选项，可选的，没有特殊需求可以跳过
    httpProfile = HttpProfile()
    httpProfile.endpoint = "vpc.tencentcloudapi.com"

    # 实例化一个 client 选项，可选的，没有特殊需求可以跳过
    clientProfile = ClientProfile()
    clientProfile.httpProfile = httpProfile
    # 实例化要请求产品的 client 对象,clientProfile 是可选的
    client = vpc_client.VpcClient(cred, region, clientProfile)
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
# 实例化一个请求对象,每个接口都会对应一个 request 对象
req = models.DescribeNatGatewaysRequest()

totalcount,disklist = get_data(0) ## 获取 100 条以内的 disk 数据
if totalcount > 100: ## 如果 disk 记录大于 100, 则将 100 往后的数据拼接至 list
    if totalcount % 100 > 0:
        num = totalcount // 100 + 1
    else:
        num = totalcount // 100
    for i in range(num):
        if i == 0:
            continue
        else:
            disklist = disklist + get_data(i*100)[1]

list_disk = []
for i in disklist:
    nat_id = i['NatGatewayId']
    nat_name = i['NatGatewayName']
    nat_name = nat_name+nat_id
    list_disk.append({'#NATID':nat_id,'#NATNAME':nat_name})
    keys_list = list(i.keys())
    for key in keys_list:
        result = ZabbixSender(ZABBIX_SERVER_IP).send([ZabbixMetric(nat_id,key,str(i[key]))])
print(json.dumps(list_disk))

except TencentCloudSDKException as err:
    print(err)
```

注意以下三处的值根据实际环境调整

ZABBIX\_SEND\_ADDR: zabbix\_sender 路径

ZABBIX\_SERVER\_IP: server 服务器 IP 地址

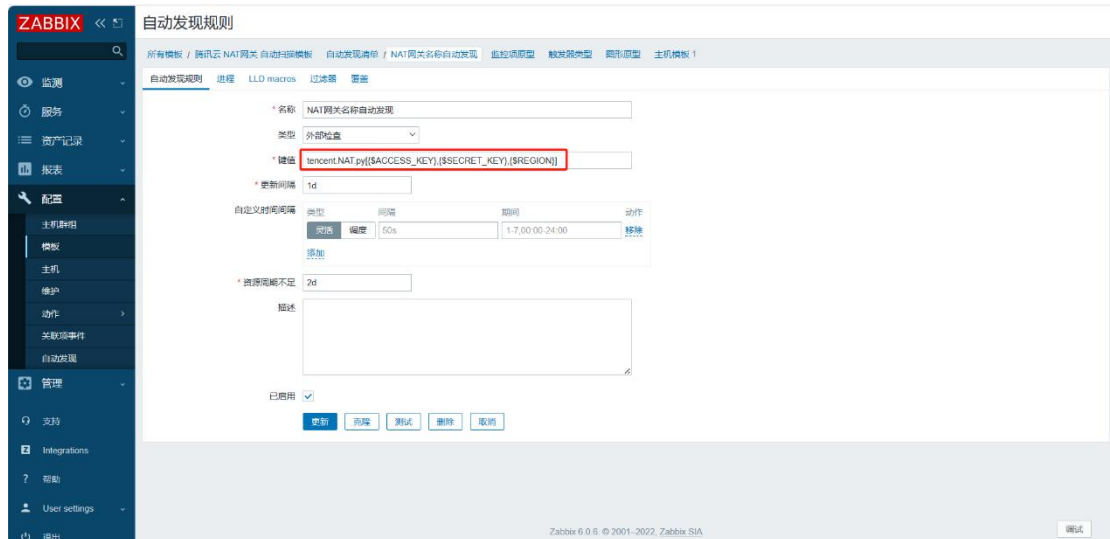
ZABBIX\_PORT: server 服务器端口

```
10 from zbxsend import ZabbixMetric, ZabbixSender
11 akey = sys.argv[1]
12 ckey = sys.argv[2]
13 region = sys.argv[3]
14 ZABBIX_SEND_ADDR = '/itops/zabbix/bin/zabbix_sender'
15 ZABBIX_SERVER_IP = '127.0.0.1'
16 ZABBIX_PORT = '10051'
```



## 3. 监控模板制作

### 3.1. 新增自动发现规则用于自动发现 NAT 网关实例



键值解释：

tencent.NAT.py：脚本名称，可自定义脚本名称，需要与外部检查路径下的脚本名称一致

`\${ACCESS\_KEY}`： SecretId

`\${SECRET\_KEY}`： SecretKey

可前往官网控制台 <https://console.cloud.tencent.com/cam/capi> 获取 SecretId 和 SecretKey

`\${REGION}`：资源地域，表示操作的资源所属的地域

### 3.2. 新增监控数据接收的监控模板

监控项名称入口：



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

大家好，我是乐乐，专注运维技术研究与分享，关注我学习更多 Zabbix 等使用技巧，更多运维问题还可以到乐维社区留言提问。乐维社区是运维监控领域的垂直社区，专注打造 Zabbix 等技术栈的线上交流与共享平台，每周三下午，社区关联答疑群（如下图）还提供免费专家在线答疑，欢迎小伙伴们加入。



## 七十九、使用 Zabbix 监控 openstack 的系统资源

### 概述

OpenStack 是一个开源的云计算管理平台项目，是一系列软件开源项目的组合。由 NASA(美国国家航空航天局)和 Rackspace 合作研发并发起，以 Apache 许可证 (Apache 软件基金会发布的一个自由软件许可证) 授权。

OpenStack 为私有云和公有云提供可扩展的弹性的云计算服务。项目目标是提供实施简单、可大规模扩展、丰富、标准统一的云计算管理平台。

对于 Openstack 运维人员来说，需要掌握 Openstack 云平台系统资源的整体运行情况,包括域(AZ)的cpu/mememory等资源使用情况;本文将介绍使用 zabbix 结合 openstack python-api 对 openstack 系统资源进行监控，详情如下：

### 使用控制台获取域(AZ)监控信息

使用如下命令可以获取当前 openstack 的所有可用域。

```
nova availability-zone-list
```

```
+-----+-----+
| Name           | Status                               |
+-----+-----+
| internal       | available                             | | |
||- computer03  |                                         |
|||- nova-storage | enabled :-) 2018-10-18T10:56:07.000000 |
```

```
| |- computer04          | | |
| | |- nova-storage     | enabled :-) 2018-10-18T10:56:08.000000 |
| |- computer05          |
| | |- nova-storage     | enabled :-) 2018-10-18T10:56:13.000000 |
| |- computer06          |
| | |- nova-storage     | enabled :-) 2018-10-18T10:56:05.000000 |
| |- computer07          |
| | |- nova-storage     | enabled :-) 2018-10-18T10:56:05.000000 |
| |- computer08          |
| | |- nova-storage     | enabled :-) 2018-10-18T10:56:07.000000 |
| |- controler02         |
| | |- nova-conductor   | enabled XXX 2017-10-09T09:46:27.000000 |
| | |- nova-consoleauth | enabled XXX 2017-10-09T09:46:37.000000 |
| | |- nova-monitor     | enabled :-) 2018-10-18T10:56:11.000000 |
| | |- nova-scheduler   | enabled XXX 2017-10-09T09:46:37.000000 |
| | |- nova-cert        | enabled XXX 2017-10-09T09:46:37.000000 |
| |- controller01        |
| | |- nova-conductor   | enabled :-) 2018-10-18T10:56:11.000000 |
| | |- nova-consoleauth | enabled :-) 2018-10-18T10:56:04.000000 |
| | |- nova-monitor     | enabled :-) 2018-10-18T10:56:07.000000 |
| | |- nova-scheduler   | enabled :-) 2018-10-18T10:56:09.000000 |
| | |- nova-cert        | enabled :-) 2018-10-18T10:56:12.000000 |
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
| IMS                | available                | | |
|- computer03      |                          |
|||- nova-compute  | enabled :-) 2018-10-18T10:56:07.000000 |
|- computer04      |                          |
|||- nova-compute  | enabled :-) 2018-10-18T10:56:05.000000 |
|- computer08      |                          |
|||- nova-compute  | enabled :-) 2018-10-18T10:56:13.000000 |
| paas              | available                |
|- computer05      |                          |
|||- nova-compute  | enabled :-) 2018-10-18T10:56:07.000000 |
|- computer06      |                          |
|||- nova-compute  | enabled :-) 2018-10-18T10:56:13.000000 |
|- computer07      |                          |
|||- nova-compute  | enabled :-) 2018-10-18T10:56:04.000000 |
+-----+-----+-----+
```

由结果可以看出，该系统含有两个域： IMS 和 paas 。

但是该命令结果不太直观，不容易提取，我们也可以使用如下命令：

```
nova aggregate-list

+----+-----+-----+
| Id | Name   | Availability Zone |
+----+-----+-----+
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
| 1 | IMS+RCS | IMS |
| 4 | paas | paas |
+----+-----+-----+

```

根据 aggregate-list ，我们可以查看域对应的计算节点：

```
nova aggregate-details paas

+----+-----+-----+-----+
-----+
| Id | Name | Availability Zone | Hosts |
Metadata |
+----+-----+-----+-----+
-----+
| 4 | paas | paas | 'computer07', 'computer05', 'computer06' |
'availability_zone=paas' |
+----+-----+-----+-----+
-----+

```

由结果可知，paas 域，一共包含三个 computer 节点，对于每个节点的使用情况，我们一样可以通过命令获取：

```
nova hypervisor-show computer07

+-----+-----+
| Property | Value |

```

```
+-----+-----+
| free_disk_gb      | 55      |
| free_ram_mb       | 126156  |
| host_ip           | 193.2.0.37 |
| hypervisor_hostname | computer07 |
| hypervisor_type   | QEMU    |
| id                | 9       |
| local_gb          | 445     |
| local_gb_used     | 390     |
| memory_mb         | 257740  |
| memory_mb_used    | 131584  |
| npt_ept           | ept     |
| pci_pools         | -       |
| running_vms       | 8       |
| service_disabled_reason | -       |
| service_host      | computer07 |
| service_id        | 27      |
| state             | up      |
| status            | enabled  |
| vcpus             | 48      |
| vcpus_used        | 32      |
+-----+-----+
```



```
#截取部分信息
```

我们可以得知，该节点的 vcpu 总数及其使用情况, memory\_mb 总数及其使用情况。

将每个节点的使用情况均获取后，经过计算，就可以得出域的资源整体使用情况。对于云平台的整体使用情况，我们也可以通过命令行获取：

```
nova hypervisor-stats

+-----+-----+
| Property          | Value  |
+-----+-----+
| count             | 6      |
| current_workload  | 0      |
| disk_available_least | 304   |
| free_disk_gb      | 518    |
| free_ram_mb       | 961736 |
| local_gb          | 2670   |
| local_gb_used     | 2152   |
| memory_mb         | 1546440 |
| memory_mb_used    | 584704 |
| running_vms       | 42     |
| vcpus             | 288    |
```

```
| vcpus_used          | 172    |  
+-----+-----+
```

上面我们讨论了使用命令行进行相关信息的获取，下面我们讨论使用 python-api 进行相关信息的获取与计算。

## 使用 openstack python-api 获取域监控信息

程序代码如下：

```
#!/usr/bin/python  
# -*- coding: utf-8 -*-  
  
#imports  
import json  
from optparse import OptionParser  
from novaclient import client as noclient  
from novaclient import utils  
  
#登录及授权  
keystone = {}  
keystone['os_username']='admin'  
keystone['os_password']='keystone'  
keystone['os_auth_url']='http://lb-vip:5000/v2.0/'  
keystone['os_tenant_name']='admin'
```

```
nova_client = noclient.Client(2, keystone['os_username'],
keystone['os_password'], keystone['os_tenant_name'], keystone['os_auth_url'])

def main():

    #获取云平台整体信息并打印

    total_info = nova_client.hypervisor_stats.statistics()._info.copy()

    print "total_info_vcpus:", total_info["vcpus"]

    print "total_info_vcpus_used:", total_info["vcpus_used"]

    print "total_info_memory_mb:", total_info["memory_mb"]

    print "total_info_memory_mb_used:", total_info["memory_mb_used"]

    print "total_info_running_vms:", total_info["running_vms"]

    #获取域列表信息

    aggregates = nova_client.aggregates.list()

    for aggregate in aggregates:

        #初始化每个域的资源统计变量

        vcpus = 0

        vcpus_used = 0

        memory_mb = 0

        memory_mb_used = 0

        running_vms = 0
```

```
#获取每个 aggregate 信息，并保存对应的 hostscomputer 节点列表

aggregate_info = aggregate_info.copy()

print          aggregate_info["id"],          aggregate_info["name"],
aggregate_info["availability_zone"], aggregate_info["hosts"]

aggregate_hosts = aggregate_info["hosts"]

#循环计算节点，保存相关资源信息

for aggregate_host in aggregate_hosts:

    hypervisor_info    =    utils.find_resource(nova_client.hypervisors,
aggregate_host)._info

    vcpus = vcpus + hypervisor_info["vcpus"]

    vcpus_used = vcpus_used + hypervisor_info["vcpus_used"]

    memory_mb = memory_mb + hypervisor_info["memory_mb"]

    memory_mb_used    =    memory_mb_used    +
hypervisor_info["memory_mb_used"]

    running_vms = running_vms + hypervisor_info["running_vms"]

#打印域资源信息

print "vcpus:", vcpus

print "vcpus_used:", vcpus_used

print "memory_mb:", memory_mb
```

```
print "memory_mb_used:", memory_mb_used

print "running_vms:", running_vms

if __name__ == "__main__":

    main()
```

执行该程序后, 可以获取各个域节点的信息:

```
total_info_vcpus: 288

total_info_vcpus_used: 172

total_info_memory_mb: 1546440

total_info_memory_mb_used: 584704

total_info_running_vms: 42

1 IMS+RCS IMS [u'computer04', u'computer03', u'computer08']

vcpus: 144

vcpus_used: 84

memory_mb: 773220

memory_mb_used: 243200

running_vms: 20

4 paas paas [u'computer07', u'computer05', u'computer06']

vcpus: 144

vcpus_used: 88
```

```
memory_mb: 773220  
  
memory_mb_used: 341504  
  
running_vms: 22
```

经过适当的计算，我们就可以获取各个域分配及使用比例等信息。

上面我们就使用 python-api 打印出了所有需要的信息，但是对于监控来说，我们需要提取的是各个监控项的信息，这样才能方便的搜索和做图表展示；下面我们讨论结合 zabbix 进行相关信息的监控。

## 结合 zabbix 获取域相关监控信息

### 获取可用域信息列表

上面我们已经获取了所有的可用域信息，但对于 zabbix 来说，我们还需要返回固定格式的数据，供 zabbix 进行解析：

```
#!/usr/bin/python  
  
# -*- coding: utf-8 -*-  
  
#imports  
  
import json  
  
from optparse import OptionParser  
  
from novaclient import client as noclient  
  
from novaclient import utils
```

```
#登录及授权

keystone = {}

keystone['os_username']='admin'

keystone['os_password']='keystone'

keystone['os_auth_url']='http://lb-vip:5000/v2.0/'

keystone['os_tenant_name']='admin'

nova_client = noclient.Client(2, keystone['os_username'],
keystone['os_password'], keystone['os_tenant_name'], keystone['os_auth_url'])

def main():

    r = {"data":[]}

    aggregates = nova_client.aggregates.list()

    for aggregate in aggregates:

        aggregate_info = aggregate._info.copy()

        r['data'].append( {"#NAME"}:aggregate_info["name"])

    print(json.dumps(r, indent=2, sort_keys=True, encoding="utf-8"))

if __name__ == "__main__":

    main()
```

执行后, 返回的结果如下,可以通过供 zabbix 自动发现模版使用:

```
{
  "data": [
    {
      "#{AVAILABLE_ZONE}": "IMS",
      "#{NAME}": "IMS+RCS"
    },
    {
      "#{AVAILABLE_ZONE}": "paas",
      "#{NAME}": "paas"
    }
  ]
}
```

## zabbix, Item 及自动发现 LLD 配置

云平台整体信息, Item 设置如下:

```
openstack.total[vcpus]
openstack.total[vcpus_used]
openstack.total[memory_mb]
openstack.total[memory_mb_used]
openstack.total[running_vms]
```

获取{#NAME}后就可以根据监控项获取对应的监控内容了, zabbix 自动发现自动发现

key 设置如下:



```
openstack.system.discovery
```

discovery 下, 设置监控 Item,key 设置如下:

```
openstack.zone[hosts,{#NAME}]  
openstack.zone[vcpus,{#NAME}]  
openstack.zone[vcpus_used,{#NAME}]  
openstack.zone[memory_mb,{#NAME}]  
openstack.zone[memory_mb_used,{#NAME}]  
openstack.zone[running_vms,{#NAME}]
```

其中{#NAME}为第一步的脚本中返回的可用域对应的 aggregate 相关信息。

## zabbix-agent 配置与对应的脚本

对应的 zabbix-agent.conf 配置如下:

```
# /etc/zabbix/zabbix_agentd.d/userparameter_openstack-system.conf  
  
UserParameter=openstack.system.discovery,python  
  
/etc/zabbix/zabbix_agentd.d/openstack-system.py --item discovery  
  
UserParameter=openstack.total[*],python  
  
/etc/zabbix/zabbix_agentd.d/openstack-system.py --item total --monitor $1  
  
UserParameter=openstack.zone[*],python  
  
/etc/zabbix/zabbix_agentd.d/openstack-system.py --item $1 --aggregate $2
```

对应的 zabbix 自动发现的监控脚本如下：

```
# /etc/zabbix/zabbix_agentd.d/openstack-system.py

#!/usr/bin/python

# -*- coding: utf-8 -*-

#imports

import json

from optparse import OptionParser

from novaclient import client as noclient

from novaclient import utils

#登录及授权

keystone = {}

keystone['os_username']='admin'

keystone['os_password']='keystone'

keystone['os_auth_url']='http://lb-vip:5000/v2.0/'

keystone['os_tenant_name']='admin'

nova_client = noclient.Client(2, keystone['os_username'],

keystone['os_password'], keystone['os_tenant_name'], keystone['os_auth_url'])

def main():
```

```
options = parse_args()

if options.item=="discovery":

    zone_list()

elif options.item=="total":

    total_moniter(options)

else:

    zone_moniter(options)

#判断入参合法性

def parse_args():

    parser = OptionParser()

    valid_item = ["discovery", "total", "hosts", "vcpus", "vcpus_used",

"memory_mb", "memory_mb_used", "running_vms"]

    parser.add_option("", "--item", dest="item", help="", action="store",

type="string", default=None)

    parser.add_option("", "--moniter", dest="moniter", help="", action="store",

type="string", default=None)

    parser.add_option("", "--aggregate", dest="aggregate", help="",

action="store", type="string", default=None)

    (options, args) = parser.parse_args()

    if options.item not in valid_item:

        parser.error("Item has to be one of: "+", ".join(valid_item))
```

```
return options

#获取可用域列表

def zone_list():

    r = {"data":[]}

    aggregates = nova_client.aggregates.list()

    for aggregate in aggregates:

        aggregate_info = aggregate._info.copy()

        r['data'].append(
            { "#NAME": aggregate_info["name"],
              "#AVAILABLE_ZONE": aggregate_info["availability_zone"] } )

    print(json.dumps(r, indent=2, sort_keys=True, encoding="utf-8"))

#获取云平台整体监控信息

def total_monitor(options):

    total_info = nova_client.hypervisor_stats.statistics()._info.copy()

    print (total_info[options.monitor])

#获取可用域对应的监控信息

def zone_monitor(options):

    aggregate = utils.find_resource(nova_client.aggregates, options.aggregate)

    aggregate_info = aggregate._info.copy()
```

```
aggregate_hosts = aggregate_info["hosts"]

if options.item=="hosts":

    print (aggregate_hosts)

else:

    monitor_data = 0

    for aggregate_host in aggregate_hosts:

        hypervisor_info = utils.find_resource(nova_client.hypervisors,
aggregate_host)._info

        monitor_data = monitor_data + hypervisor_info[options.item]

    print (monitor_data)

if __name__ == "__main__":

    main()
```

在 zabbix 上进行对应的配置后重启, 将模版应用于主机, 此时应当监控获取所有的可用域, 并监控对应的信息。

## 参考资料

1. Openstack 中的 zone ,aggregates 和 host 及其应用,<https://blog.csdn.net/ztejiagn/article/details/8948688>
2. nova 命令汇总四 —— 计算相关命令,<http://blog.51cto.com/13788458/2129157>
3. The novaclient Python

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

API, <https://docs.openstack.org/python-novaclient/latest/reference/api/index.html>

#### 4. GitHub

[larsks/openstack-api-samples, https://github.com/larsks/openstack-api-samples](https://github.com/larsks/openstack-api-samples)

以上就是这一期的 Zabbix 技术分享内容。

## 八十、更多.....

# Docker 容器

## 八十二、如何使用 docker 快速部署一个 zabbix 监控系统

使用 docker 拉取官方提供的 zabbix 镜像来快速部署一个 zabbix 监控系统

### 1.环境准备

Centos7 系统

服务器 IP: 192.168.75.31

### 2.安装 docker

#### 2.1. Yum 安装

```
yum -y install docker
```

#### 2.2. 二进制安装 (可离线)

Docker 二进制包下载,也可在 windows 下载后再上传至服务器

```
wget https://download.docker.com/linux/static/stable/x86_64/docker-20.10.14.tgz
```

Z

```
[root@localhost ~]# wget https://download.docker.com/linux/static/stable/x86_64/docker-20.10.14.tgz
--2022-09-21 17:06:08-- https://download.docker.com/linux/static/stable/x86_64/docker-20.10.14.tgz
Resolving download.docker.com (download.docker.com)... 54.230.21.59, 54.230.21.56, 54.230.21.60, ...
Connecting to download.docker.com (download.docker.com)|54.230.21.59|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 64273875 (61M) [Application/x-tar]
Saving to: 'docker-20.10.14.tgz'
3
docker-20.10.14.tgz 100%[----->] 61.39M 20.1MB/s in 3.2s
2022-09-21 17:06:13 (19.1 MB/s) - "docker-20.10.14.tgz" saved [64273875/64273875]
[root@localhost ~]# ls
docker-20.10.14.tgz
```

```
tar -zxvf docker-20.10.14.tgz
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
[System not activated][root@localhost ~]#  
[System not activated][root@localhost ~]#  
[System not activated][root@localhost ~]# tar -zxf docker-20.10.14.tgz  
[System not activated][root@localhost ~]# ls  
docker  docker-20.10.14.tgz  
[System not activated][root@localhost ~]# █
```

复制文件

```
cp -ra docker/* /usr/bin
```

```
docker docker-20.10.14.tgz  
[System not activated][root@localhost ~]# cp -ra docker/* /usr/bin/  
[System not activated][root@localhost ~]# cd docker/  
[System not activated][root@localhost docker]# ls  
containerd  containerd-shim  containerd-shim-runc-v2  ctr  docker  dockerd  docker-init  docker-proxy  runc
```

编写 system 文件

```
vim /etc/systemd/system/docker.service
```

```
[Unit]
```

```
Description=Docker Application Container Engine
```

```
Documentation=http://docs.docker.io
```

```
[Service]
```

```
OOMScoreAdjust=-1000
```

```
ExecStart=/usr/bin/dockerd
```

```
ExecStartPost=/sbin/iptables -I FORWARD -s 0.0.0.0/0 -j ACCEPT
```

```
ExecReload=/bin/kill -s HUP \${MAINPID}
```

```
Restart=on-failure
```

```
RestartSec=5
```



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

LimitNOFILE=infinity

LimitNPROC=infinity

LimitCORE=infinity

Delegate=yes

KillMode=process

[Install]

WantedBy=multi-user.target

启动 docker

systemctl daemon-reload

systemctl start docker

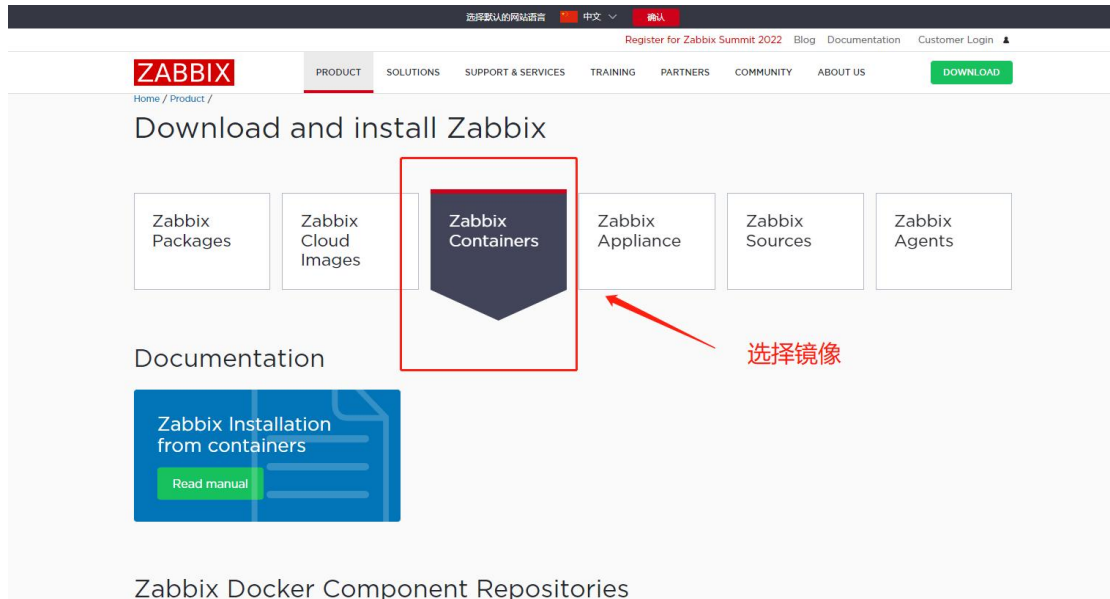
```
system not activated [root@localhost ~]# vim /etc/systemd/system/docker.service
system not activated [root@localhost ~]# systemctl daemon-reload
system not activated [root@localhost ~]# systemctl start docker
system not activated [root@localhost ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/etc/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2022-09-21 17:09:13 CST; 7s ago
     Docs: http://docs.docker.io
   Process: 1774 ExecStartPost=/sbin/iptables -I FORWARD -s 0.0.0.0/0 -j ACCEPT (code=exited, status=0/SUCCESS)
   Main PID: 1773 (dockerd)
     Tasks: 17
    Memory: 29.3M
    CGroup: /system.slice/docker.service
            └─1773 /usr/bin/dockerd
                └─1772 containerd -config /var/run/docker/containerd/containerd.toml --log-level info
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.322443736+08:00" level=info msg="parsed scheme: \"unix\" module=grpc
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.322482909+08:00" level=info msg="scheme \"unix\" not registered, fallback to default scheme" module=grpc
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.322499861+08:00" level=info msg="ccResolverWrapper: sending update to cc: [{unix:///var/run/docker/containerd/containerd.sock <nil> 0 <nil>}] <nil> <nil>"
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.322598049+08:00" level=info msg="clientConn switching balancer to \"pick_first\"" module=grpc
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.351659589+08:00" level=info msg="Loading containers: start."
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.420742328+08:00" level=info msg="Default bridge (dockero) is assigned with an IP address 172.17.0.0/16. Daemon option --bip can be used to set a preferred IP a
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.476288355+08:00" level=info msg="Loading containers: done."
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.565670955+08:00" level=info msg="Docker daemon" commit=67a90dc graphdriver(s)=overlay2 version=20.10.14
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.565869494+08:00" level=info msg="Daemon has completed initialization"
Sep 21 17:09:14 localhost.localdomain dockerd[1773]: time="2022-09-21T17:09:14.588839854+08:00" level=info msg="API listen on /var/run/docker.sock"
system not activated [root@localhost ~]#
```

### 3.拉取 docker 镜像

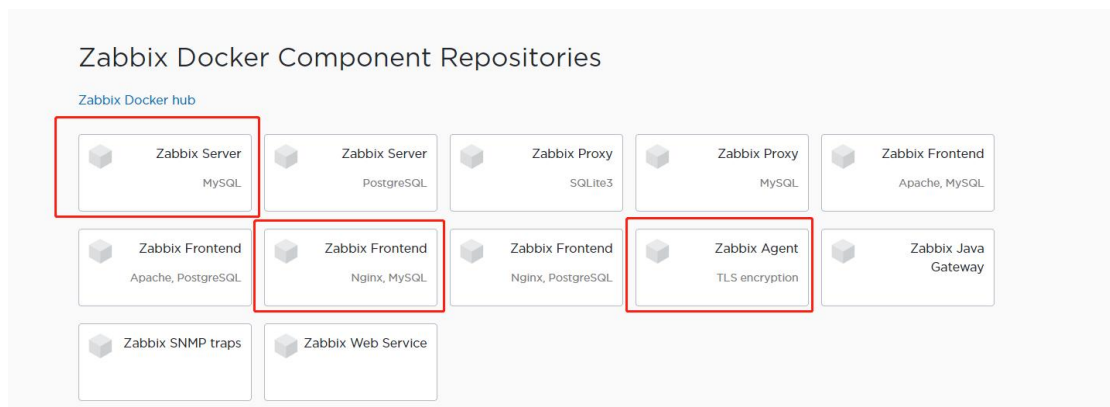
访问 zabbix 官方提高的 docker 镜像

[https://www.zabbix.com/container\\_images](https://www.zabbix.com/container_images)

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



这里选择使用 MySQL 数据库的 zabbix-server,zabbix-agent 以及 zabbix 界面的三个镜像，zabbix 版本为最新的 6.2



Zabbix6 版本以上的 MySQL 数据库需为 MySQL 8 以上，因此拉取 MySQL 官方提供的 MySQL 8 镜像。

拉取四个镜像命令：（linux 上执行）

```
docker pull mysql:latest
```

```
docker pull zabbix/zabbix-server-mysql:latest
```

```
docker pull zabbix/zabbix-web-nginx-mysql:latest
```

```
docker pull zabbix/zabbix-agent:latest
```

```
[System not activated][root@localhost ~]# docker pull mysql:latest
latest: Pulling from library/mysql
051f419db9dd: Pull complete
7627573fa82a: Pull complete
a44b358d7796: Pull complete
95753aff4b95: Pull complete
a1fa3bee53f4: Pull complete
f5227e0d612c: Pull complete
b4b4368b1983: Pull complete
f26212810c32: Pull complete
d803d4215f95: Pull complete
d5358a7f7d07: Pull complete
435e8908cd69: Pull complete
Digest: sha256:b9532b1edea72b6cee12d9f5a78547bd3812ea5db842566e17f8b33291ed2921
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
[System not activated][root@localhost ~]# docker pull zabbix/zabbix-server-mysql:latest
latest: Pulling from zabbix/zabbix-server-mysql
213ec9aee27d: Pull complete
9484476a42b3: Pull complete
ebbc2985f6d2: Pull complete
76ac7d2089c9: Pull complete
fe0137176975: Pull complete
c408cc9e8da5: Pull complete
1c55d56cfe7c: Pull complete
4f4fb700ef54: Pull complete
c731d2d69475: Pull complete
Digest: sha256:8e616db62b75daec9260d39f921d9ddf021eb613b32353a3b991ee17efcddd9
Status: Downloaded newer image for zabbix/zabbix-server-mysql:latest
docker.io/zabbix/zabbix-server-mysql:latest
[System not activated][root@localhost ~]# docker pull zabbix/zabbix-web-nginx-mysql:latest
latest: Pulling from zabbix/zabbix-web-nginx-mysql
213ec9aee27d: Already exists
3324412a7d2b: Pull complete
3721c33b0791: Pull complete
d790b14bec9: Pull complete
4f4fb700ef54: Pull complete
0c31154cbbbd: Pull complete
Digest: sha256:5c5c70d2529b34e6dfe0dac36f115da76f2ab3495fd0cea2193e8747c8916da1
Status: Downloaded newer image for zabbix/zabbix-web-nginx-mysql:latest
docker.io/zabbix/zabbix-web-nginx-mysql:latest
[System not activated][root@localhost ~]# docker pull zabbix/zabbix-agent:latest
latest: Pulling from zabbix/zabbix-agent
213ec9aee27d: Already exists
1b601b806eee: Pull complete
a20d3d7735e6: Pull complete
176f3b1a0a49: Pull complete
65f4014bf43c: Pull complete
35afe5502e3a: Pull complete
4f4fb700ef54: Pull complete
2635ad96ccfb: Pull complete
Digest: sha256:c2084d414ff30f92e69c65f13dfe153f83eddc7e99d030296f865347fdc85a2d
Status: Downloaded newer image for zabbix/zabbix-agent:latest
docker.io/zabbix/zabbix-agent:latest
[System not activated][root@localhost ~]#
```

命令执行完后查看镜像是否拉取成功

```
docker images
```

```
[root@docke ~]# docker images
REPOSITORY          TAG         IMAGE ID      CREATED      SIZE
mysql               latest     43fcfca0776d  3 days ago  449MB
zabbix/zabbix-web-nginx-mysql  latest     209092699757  9 days ago  211MB
zabbix/zabbix-server-mysql    latest     3e15d051ed63  9 days ago  82.2MB
zabbix/zabbix-agent          latest     a9975d793793  9 days ago  15.4MB
[root@docke ~]#
```

## 4.启动镜像

### 4.1. 首先要启动 MySQL 数据库

```
docker run --name zabbix-mysql -e MYSQL_DATABASE=zabbix -e MYSQL_ROOT_PASSWORD=zabbix -p 3306:3306 -d mysql:latest
```

参数解释:

--name zabbix-mysql #定义容器名称

-e MYSQL\_DATABASE=zabbix #初始数据库名 zabbix

-e MYSQL\_ROOT\_PASSWORD=zabbix #数据库 root 用户的密码 zabbix

-p 3306:3306 #将容器 3306 端口映射到主机 3306 端口

```
(System not activated)[root@localhost ~]# docker run --name zabbix-mysql -e MYSQL_DATABASE=zabbix -e MYSQL_ROOT_PASSWORD=zabbix -p 3306:3306 -d mysql:latest
11325861d606ef11463d8ce0a989e57dd8c066a259c45ade2dfbcffc39d8f7a2
(System not activated)[root@localhost ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
11325861d606   mysql:latest  "docker-entrypoint.s..."  2 seconds ago  Up 1 second   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp  zabbix-mysql
(System not activated)[root@localhost ~]#
```

### 4.2. 然后启动 zabbix-server

```
docker run --name zabbix-server -e MYSQL_DATABASE=zabbix -e DB_SERVER_HOST=192.168.75.31 -e DB_SERVER_PORT=3306 -e MYSQL_USER=root -e MYSQL_PASSWORD=zabbix -p 10051:10051 -d zabbix/zabbix-server-mysql:latest
```

参数解释:

--name zabbix-server #定义容器名称

-e DB\_SERVER\_HOST=192.168.75.31 #数据库连接地址

-e DB\_SERVER\_PORT=3306 #数据库连接端口

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

-e MYSQL\_DATABASE=zabbix #连接的数据库名 zabbix

-e MYSQL\_USER=root #数据库连接用户 root

-e MYSQL\_PASSWORD=zabbix #数据库连接密码 zabbix

-p 10051:10051 #将容器 10051 端口映射到主机 10051 端口

-d 后台运行

zabbix/zabbix-server-mysql:latest 启动容器的镜像

```
[root@localhost ~]# docker run --name zabbix-server -e MYSQL_DATABASE=zabbix -e DB_SERVER_HOST=192.168.75.31 -e DB_SERVER_PORT=3306 -e MYSQL_USER=root -e MYSQL_PASSWORD=zabbix -p 10051:10051 -d zabbix/zabbix-server-mysql:latest
3705ac21388c474e270e01016d91184d438536446095c720d4591458569d34
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
3705ac21388c   zabbix/zabbix-server-mysql:latest   "fsbin/rmi -- /brf..." 3 seconds ago  Up 2 seconds  0.0.0.0:10051->10051/tcp, :::10051->10051/tcp  zabbix-server
113258616606   mysql:latest                         "docker-entrypoint.s..." 23 seconds ago  Up 22 seconds  0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp  zabbix-mysql
```

### 4.3. 最后启动 zabbix 界面

```
docker run --name zabbix-web -e ZBX_SERVER_HOST=192.168.75.31 -e ZBX_SERVER_PORT=10051 -e DB_SERVER_HOST=192.168.75.31 -e DB_SERVER_PORT=3306 -e MYSQL_DATABASE=zabbix -e MYSQL_USER=root -e MYSQL_PASSWORD=zabbix -p 8080:8080 -p 8443:8443 -d zabbix/zabbix-web-nginx-mysql:latest
```

参数解释:

--name zabbix-web #定义容器名称

-e ZBX\_SERVER\_HOST=192.168.75.31 #web 连接 server 的地址

-e ZBX\_SERVER\_PORT=10051 #web 连接 server 的端口

-e DB\_SERVER\_HOST=192.168.75.31 #数据库连接地址

本文档为样章，完整版文档请添加乐乐（lerwee）获取

```
-e DB_SERVER_PORT=3306          #连接的数据库端口

-e MYSQL_DATABASE=zabbix       #连接的数据库名 zabbix

-e MYSQL_USER=root             #数据库连接用户

-e MYSQL_PASSWORD=zabbix       #数据库连接密码 zabbix

-p 8080:8080                   #将容器 8080 端口映射到主机 8080 端口

-d                               #后台运行

zabbix/zabbix-web-nginx-mysql:latest    #启动容器的镜像
```

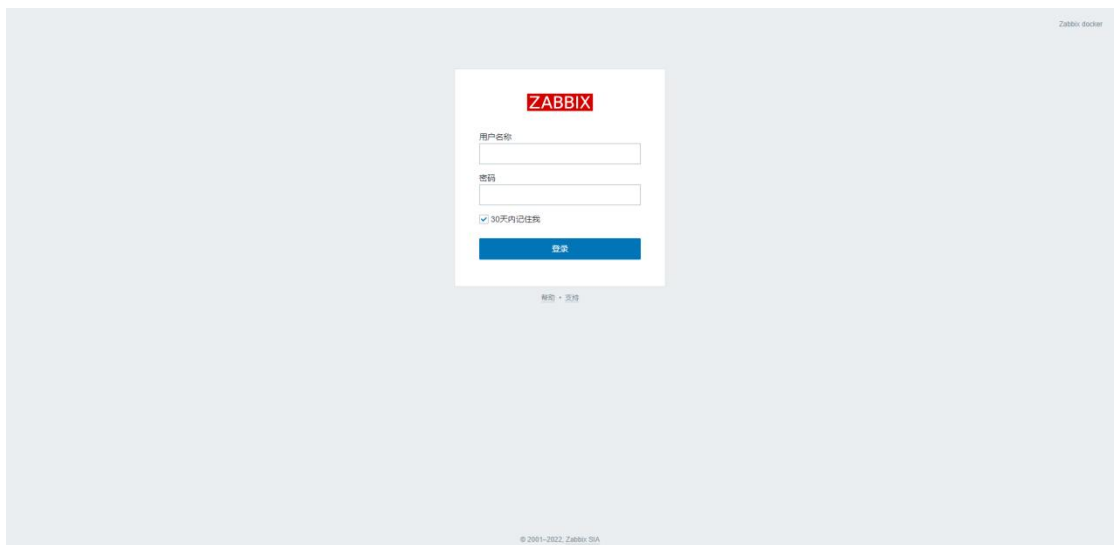
```
[root@localhost ~]# docker run --name zabbix-web -e ZBX_SERVER_HOST=192.168.75.31 -e ZBX_SERVER_PORT=10051 -e DB_SERVER_HOST=192.168.75.31 -e DB_SERVER_PORT=3306 -e MYSQL_DATABASE=zabbix -e MYSQL_USER=root -e MYSQL_PASSWORD=zabbix -e ZBX_SERVER_PORT=10051 zabbix/zabbix-web-nginx-mysql:latest
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                                                                 NAMES
8649f93b05e   zabbix/zabbix-web-nginx-mysql:late  "docker-entrypoint.sh"  3 seconds ago  Up 2 seconds  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:8443->8443/tcp, :::8443->8443/tcp  zabbix-web
0752ac21386c   zabbix/zabbix-server-mysql:latest   "/sbin/tini -- /usr/..." 26 seconds ago  Up 27 seconds  0.0.0.0:10051->10051/tcp, :::10051->10051/tcp  zabbix-server
11292804699c   mysql:latest                         "docker-entrypoint.sh"  48 seconds ago  Up 47 seconds  0.0.0.0:3306->3306/tcp, :::3306->3306/tcp  zabbix-mysql
```

## 5.访问 web

全部启动后等待 2-3 分钟，待数据库初始化完成。

然后访问 web 界面

<http://IP:8080>



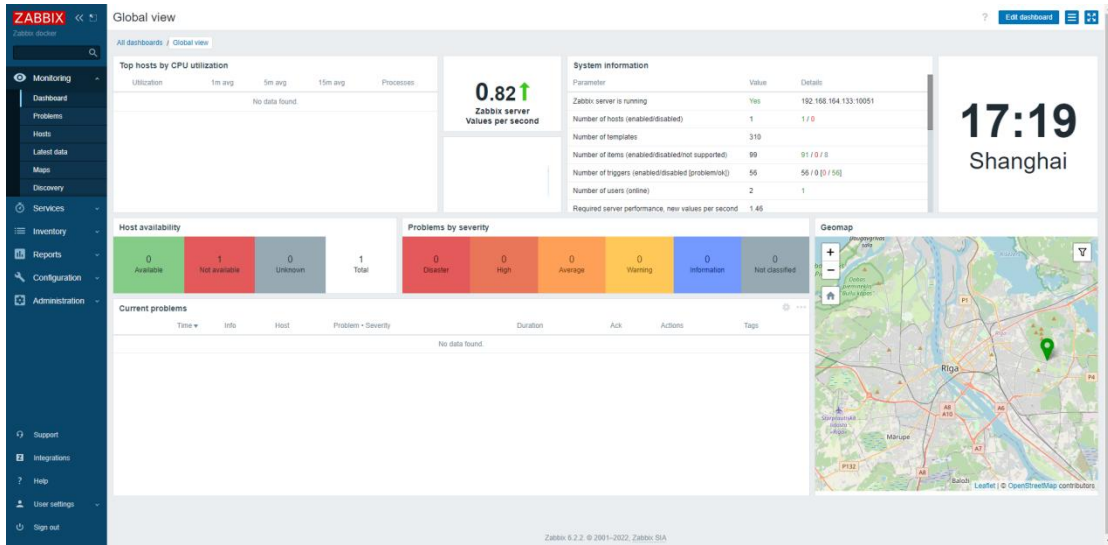
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

初始登录信息

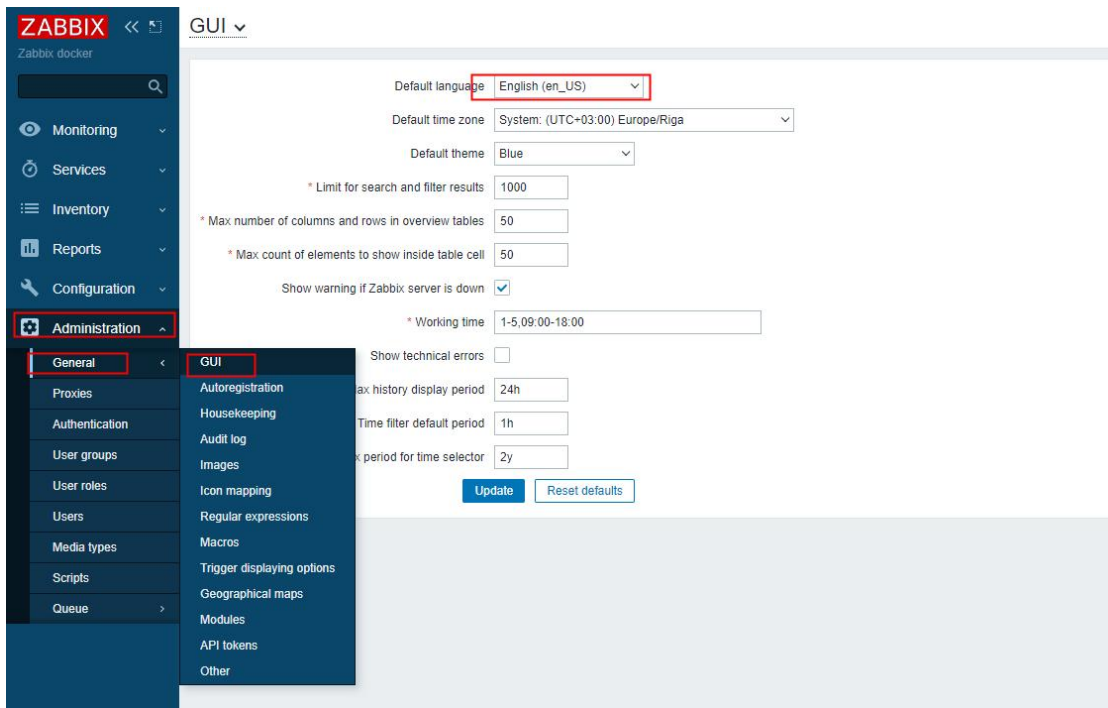
账户: Admin

密码: zabbix

登录后的界面



修改成中文显示



## 6.添加 agent 监控

启动 agent

```
docker run --name zabbix-agent --link zabbix-server -e ZBX_HOSTNAME=192.168.75.31 -e ZBX_SERVER_HOST=172.17.0.1 -e ZBX_SERVER_PORT=10051 -p 10050:10050 --privileged -d zabbix/zabbix-agent:latest
```

参数解释:

--name zabbix-agent #定义容器名称

--link zabbix-server #连接到 zabbix-server 容器

-e ZBX\_HOSTNAME=192.168.75.31 #zabbix-server 主机名

-e ZBX\_SERVER\_HOST=172.17.0.1 #zabbix-server 连接地址, 这里固定这个 IP

-p 10050:10050 #将容器 10050 端口映射到主机 10050 端口

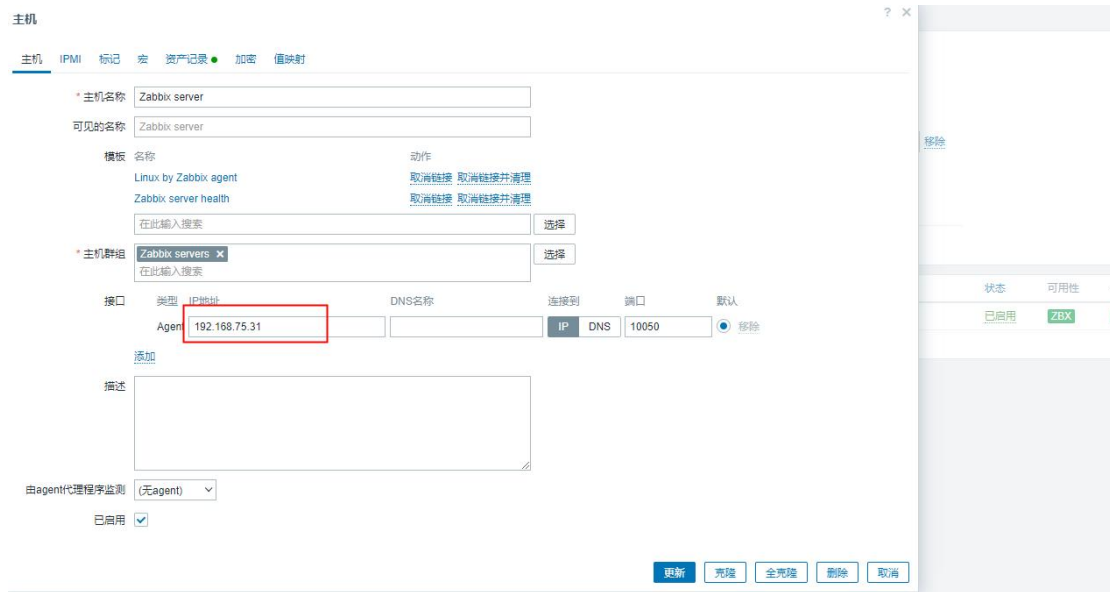
-d #后台运行

**PS:**这里 ZBX\_SERVER\_HOST 无须修改 IP

然后到 UI 界面将默认的 zabbix 监控修改一下 agent 的 IP 信息

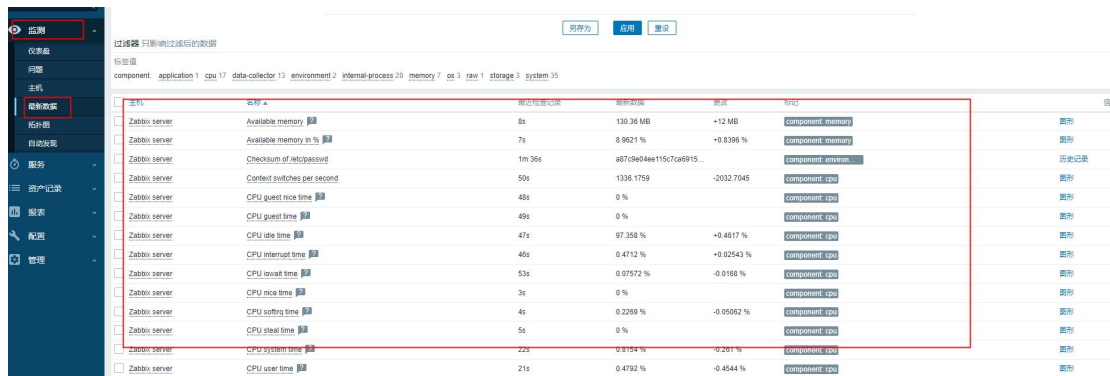


本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



即可完成 agent 监控的添加

在监测>最新数据可看到监控正常获取数据



至此，整个系统的搭建到此结束。

要监控其他的服务器，可在服务器安装部署一个 agent，然后在配置>主机添加对应的监控对象即可完成监控。

Agent 部署：将 Agent 打包成通用二进制部署包，以方便如何快速进行

Agent 客户端的安装，参考文章：[如何快速部署 zabbix-agent 客户端](https://forum.lwops.cn)

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

使用 docker-compose 工具进行部署 (更快!) : 比单纯使用 docker 部

署 zabbix 更快拉起一个监控系统, 而且管理起来更方便! 参考: [使用](#)

[docker-compose 快速部署 zabbix 监控系统](#)

## 八十三、zabbix6 监控 k8s

1、创建 yml 文件安装 metrics 监控组件，修改节点名称 (kubectl get nodes 可查看节点名)

```
vim metrics.yaml

apiVersion: v1

automountServiceAccountToken: false

kind: ServiceAccount

metadata:

  labels:

    app.kubernetes.io/component: exporter

    app.kubernetes.io/name: kube-state-metrics

    app.kubernetes.io/version: 2.4.2

  name: kube-state-metrics

  namespace: monitoring

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

  labels:

    app.kubernetes.io/component: exporter

    app.kubernetes.io/name: kube-state-metrics

    app.kubernetes.io/version: 2.4.2

  name: kube-state-metrics

rules:

- apiGroups:

  - ""

  resources:

  - configmaps

  - secrets

  - nodes
```

- pods
- services
- resourcequotas
- replicationcontrollers
- limitranges
- persistentvolumeclaims
- persistentvolumes
- namespaces
- endpoints
- verbs:
- list
- watch
- apiGroups:
- apps
- resources:
- statefulsets
- daemonsets
- deployments
- replicaset
- verbs:
- list
- watch
- apiGroups:
- batch
- resources:
- cronjobs
- jobs
- verbs:
- list
- watch
- apiGroups:
- autoscaling
- resources:

```
- horizontalpodautoscalers
verbs:
- list
- watch
- apiGroups:
- authentication.k8s.io
resources:
- tokenreviews
verbs:
- create
- apiGroups:
- authorization.k8s.io
resources:
- subjectaccessreviews
verbs:
- create
- apiGroups:
- policy
resources:
- poddisruptionbudgets
verbs:
- list
- watch
- apiGroups:
- certificates.k8s.io
resources:
- certificatesigningrequests
verbs:
- list
- watch
- apiGroups:
- storage.k8s.io
resources:
```

```
- storageclasses
- volumeattachments
verbs:
- list
- watch
- apiGroups:
- admissionregistration.k8s.io
resources:
- mutatingwebhookconfigurations
- validatingwebhookconfigurations
verbs:
- list
- watch
- apiGroups:
- networking.k8s.io
resources:
- networkpolicies
- ingresses
verbs:
- list
- watch
- apiGroups:
- coordination.k8s.io
resources:
- leases
verbs:
- list
- watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
labels:
```

```
  app.kubernetes.io/component: exporter
  app.kubernetes.io/name: kube-state-metrics
  app.kubernetes.io/version: 2.4.2
  name: kube-state-metrics
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kube-state-metrics
subjects:
- kind: ServiceAccount
  name: kube-state-metrics
  namespace: monitoring
---
apiVersion: v1
kind: Service
metadata:
  annotations:
    prometheus.io/scraped: "true" # 设置能被 prometheus 抓取到，因为不带这个 annotation prometheus-service-endpoints 不会去抓这个 metrics
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 2.4.2
    name: kube-state-metrics
    namespace: monitoring
spec:
  # clusterIP: None # 允许通过 svc 来进行访问
  ports:
  - name: http-metrics
    port: 8088
    targetPort: http-metrics
  - name: telemetry
    port: 8089
    targetPort: telemetry
```

```
selector:
  app.kubernetes.io/name: kube-state-metrics
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 2.4.2
  name: kube-state-metrics
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: kube-state-metrics
  template:
    metadata:
      labels:
        app.kubernetes.io/component: exporter
        app.kubernetes.io/name: kube-state-metrics
        app.kubernetes.io/version: 2.4.2
    spec:
      nodeName: k8s-master-1 # 设置在 k8s-master-1 上运行
      tolerations: # 设置能容忍在 master 节点运行
        - key: "node-role.kubernetes.io/master"
          operator: "Exists"
          effect: "NoSchedule"
      automountServiceAccountToken: true
      containers:
        # - image: k8s.gcr.io/kube-state-metrics/kube-state-metrics:v2.4.2
        - image: anjia0532/google-containers.kube-state-metrics.kube-state-metrics:v2.4.2
```



```
# livenessProbe:
# httpGet:
#   path: /healthz
#   port: 8088
#   initialDelaySeconds: 5
#   timeoutSeconds: 5
name: kube-state-metrics
ports:
- containerPort: 8088
  name: http-metrics
- containerPort: 8089
  name: telemetry
# readinessProbe:
# httpGet:
#   path: /
#   port: 8089
#   initialDelaySeconds: 5
#   timeoutSeconds: 5
securityContext:
  allowPrivilegeEscalation: false
capabilities:
  drop:
  - ALL
readOnlyRootFilesystem: true
runAsUser: 65534
runAsGroup: 65534
#fsGroup: 65534
serviceAccountName: kube-state-metrics
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
app.kubernetes.io/version: 2.4.2
spec:
  nodeName: k8s-master-1 # 设置在k8s-master-1上运行
  tolerations: # 设置能容忍在master节点运行
  - key: "node-role.kubernetes.io/master"
    operator: "Exists"
    effect: "NoSchedule"
  automountServiceAccountToken: true
  containers:
  - image: k8s.gcr.io/kube-state-metrics/kube-state-metrics:v2.4.2
  - image: anjia0532/google-containers.kube-state-metrics.kube-state-metrics:v2.4.2
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8088
      initialDelaySeconds: 5
      timeoutSeconds: 5
    name: kube-state-metrics
    ports:
    - containerPort: 8088
      name: http-metrics
    - containerPort: 8089
      name: telemetry
    readinessProbe:
      httpGet:
        path: /
        port: 8089
      initialDelaySeconds: 5
      timeoutSeconds: 5
    securityContext:
```

## 2、添加命名空间 monitoring

```
kubectl create namespace monitoring
```

## 3、执行 yamI 文件

```
kubectl apply -f metrics.yaml
```

注：若在启动容器时提示拒绝连接，则在 yamI 文件中注释掉对应健康探针

如下为正常启动示例

```
[root@node1 ~]# kubectl get pods -n kube-metrics
NAME                                READY   STATUS    RESTARTS   AGE
kube-state-metrics-6b96bdf5fb-zkwc4 1/1     Running   0           45s
```

## 4、下载解压 helm-v3.9.4-linux-amd64.tar.gz 包，对 helm 命令进行软链

```
tar xvf helm-v3.9.4-linux-amd64.tar.gz
```

## 5、官网下载配置文件、模板进行 k8s agent 安装

<https://git.zabbix.com/projects/ZT/repos/kubernetes-helm/browse?at=refs%2Fheads%2Frelease%2F6.0>

## 本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

(以下为官网配置步骤)

添加存储库:

```
helm repo add zabbix-chart-6.0 https://cdn.zabbix.com/zabbix/integrations/kubernetes-helm/6.0
```

导出图表到文件的默认值: helm-zabbix\$HOME/zabbix\_values.yaml

```
helm show values zabbix-chart-6.0/zabbix-helm-chrt > $HOME/zabbix_values.yaml
```

根据文件中的环境更改值。\$HOME/zabbix\_values.yaml

列出群集的命名空间。

```
kubectl get namespaces
```

创建命名空间 (如果群集中不存在) 。 monitoring

```
kubectl create namespace monitoring
```

在 Kubernetes 集群中部署 Zabbix。 (如有必要, 请更新 YAML 文件路径) 。

```
helm install zabbix zabbix-chart-6.0/zabbix-helm-chrt --dependency-update -f $HOME/zabbix_values.yaml -n monitoring
```

查看容器。

```
kubectl get pods -n monitoring
```

查看 Pod 的信息。

```
kubectl describe pods/POD_NAME -n monitoring
```

查看 Pod 的所有容器。

```
kubectl get pods POD_NAME -n monitoring -o jsonpath='{.spec.containers[*].name}'
```

查看 Pod 的日志容器。

```
kubectl logs -f pods/POD_NAME -c CONTAINER_NAME -n monitoring
```

容器的访问提示。

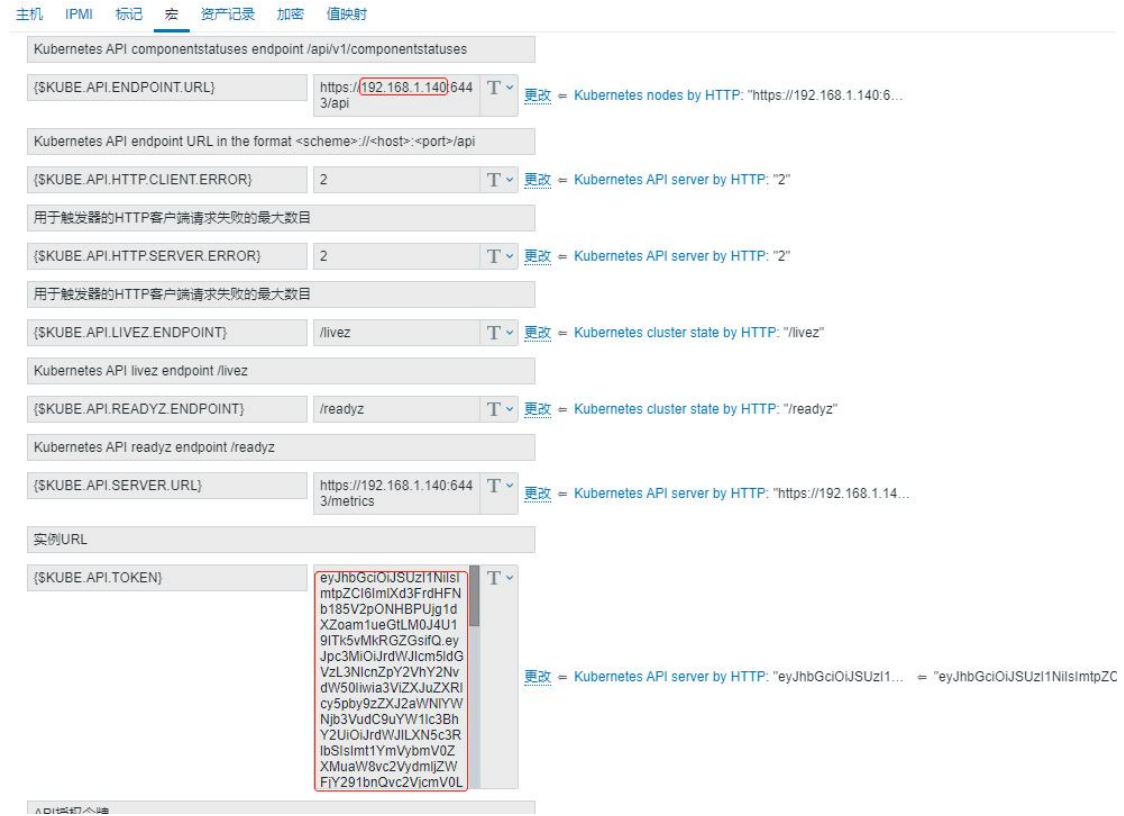
```
kubectl exec -it pods/POD_NAME -c CONTAINER_NAME -n monitoring -- sh
```

## 卸载

要卸载/删除部署, 请执行以下操作: zabbix

```
helm delete zabbix -n monitoring
```

## 6、修改模板所有与 IP、TOKEN 相关的宏 (IP 为 k8s 服务器 IP 地址，下附获取 TOKEN 方式)



TOKEN 获取方式:

k8s 服务器执行命令

```
kubectl describe secrets $(kubectl get secrets -n monitoring |grep zabbix-service-account | grep -v zabbix-service-account-token-hctrs | cut -f1 -d ' ') -n monitoring |grep -E '^token' |cut -f2 -d ':'|tr -d '\t'|tr -d ' '
```

注: 宏中调用的 API 请求地址为默认端口, 若接口无数据返回, 可执行 `kubectl config view |grep server|cut -f 2- -d ":" | tr -d " "` 获取当前环境的 API 地址

## 八十四、监控平台组件 docker 编译使用手册

### 一、docker 安装

#### 1. 安装 docker

```
curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyu
```

#### 2. 启动 docker 应用

```
systemctl start docker && systemctl enable docker
```

### 二、SQL 安装

MySQL 与 PostGreSQL 只选其一安装即可

#### 1、mysql 安装

使用方法

##### 1、先创建对应用户

```
useradd mysql -u20223 -s /sbin/nologin
```

##### 2、上传 mysql5.7.36.tar 与 data.tar 文件

mysql5.7.36.tar 为镜像文件； data.tar 为数据库文件和配置文件

##### 3、导入镜像

```
docker load -i mysql5.7.36.tar
```

### 3. 解压 data.tar

```
mkdir -p /itops/
```

```
tar -xf data.tar -C /itops/
```

```
chown mysql:mysql /itops/mysql -R
```

### 4. 运行

```
docker run -itd -p3306:3306 -v /itops/mysql/etc:/etc/mysql/conf.d
```

```
-v /itops/mysql/data:/var/lib/mysql lw_mysql:5.7.36
```

账号: root

密码: ITIM\_p@ssw0rd

账号: zabbix

密码: zabbix

## 2、postgresql 安装

### 使用方法

#### 1、先创建对应用户

```
useradd postgres -u20224 -s /sbin/nologin
```

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

## 2、上传 postgresql12.8.tar 与 data.tar 文件

postgresql12.8.tar 为镜像文件； data.tar 为数据库文件和配置文件

## 3、导入镜像

```
docker load -i postgresql12.8.tar
```

## 4、解压 data.tar

```
mkdir -p /itops/
```

```
tar -xf data.tar -C /itops/
```

```
chown postgres.postgres /itops/postgres -R
```

## 5、运行

```
docker run -itd -v/itops/postgres/data:/var/lib/postgresql/data -v
```

```
/itops/postgres/etc/postgres.conf:/etc/postgresql/postgresql.conf -p5432:5432
```

```
lw_postgres:12.8
```

# 三、server 安装

## 使用方法

### 1、先创建对应用户

```
useradd zabbix -u20222 -s /sbin/nologin
```

### 2、上传 zabbix\_server.tar 与 conf.tar 文件

zabbix\_server.tar 为镜像文件； conf.tar 为 server 配置文件和日志等目录

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

### 3、导入镜像

```
docker load -i zabbix_server.tar
```

### 4、解压 conf.tar

```
mkdir -p /itops/zabbix_server
```

```
tar -xf conf.tar -C /itops/zabbix_server
```

### 5、替换本机 IP 地址

```
sed -i 's/192.168.229.17/本机 ip/g' /itops/zabbix_server/etc/zabbix_server.conf
```

```
sed -i 's/192.168.229.17/本机 ip/g' /itops/zabbix_server/etc/zabbix_agentd.conf
```

### 6、修改目录属主

```
chown zabbix.zabbix /itops/zabbix_server -R
```

### 7、运行

```
docker run -itd -p10050:10050 -p10051:10051 -p10052:10052
```

```
-v/itops/zabbix_server:/itops/zabbix/ zabbix_server:5.0.16
```

## 四、Nginx 安装

### 使用方法

#### 1、上传压缩包 nginx-v1.3.tar.gz 与 itops\_v1\_4\_x86\_64.tar 文件

itops\_v1\_4\_x86\_64.tar 为镜像文件; nginx-v1.3.tar.gz 为 nginx 配置文件和日志等目录

#### 2、导入镜像



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
docker load -i itops_v1_4_x86_64.tar
```

### 3、解压 nginx-v1.3.tar.gz

```
mkdir -p /itops/
```

```
tar -xf nginx-v1.3.tar.gz -C /itops/
```

### 4、修改配置文件

```
sed -i 's/192.168.126.151/本机
```

```
ip/g' /itops/nginx/html/zabbix/conf/zabbix.conf.php
```

```
sed -i 's/192.168.126.151/本机
```

```
ip/g' /itops/nginx/html/lwjk_v3/web/z/conf/zabbix.conf.php
```

```
sed -i 's/192.168.126.151/本机 ip/g' /itops/nginx/html/lwjk_v3/config/db.php
```

### 5、运行

```
docker run -d -v /itops/nginx/etc:/itops/etc -v
```

```
/itops/nginx/html:/itops/nginx/html -p 80:80 -p 8081:8081 itops:v1.3
```

```
/itops/php/sbin/php-fpm --fpm-config /itops/etc/php/php-fpm.conf
```

```
/itops/nginx/sbin/nginx -c /itops/etc/nginx/nginx.conf -g "daemon off;"
```

## 五、agent 安装

### 使用方法

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

## 1、上传 zabbix\_agent.tar conf.tar

zabbix\_agent.tar 为镜像文件; conf.tar 为 agent 配置文件和日志目录

## 2、导入镜像

```
docker load -i zabbix_agent.tar
```

## 3、解压 conf.tar

```
mkdir -p /itops/zabbix_agent
```

```
tar -xf conf.tar -C /itops/zabbix_agent
```

```
chown zabbix.zabbix /itops/zabbix_agent -R
```

## 4、修改配置文件

```
sed -i 's/127.0.0.1/本地 ip/g' /itops/zabbix_agent/etc/zabbix_agentd.conf
```

## 5、运行

```
docker run -itd -p10050:10050 -v/itops/zabbix_agent/etc:/itops/zabbix/etc/
```

```
-v/itops/zabbix_agent/logs:/itops/zabbix/logs/
```

```
-v/itops/zabbix_agent/scripts:/itops/zabbix/scripts/ zabbix_agent:5.0.16
```

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

6、登录 web 界面

http://本机 ip

账号: Admin

密码: zabbix

**八十五、更多.....**

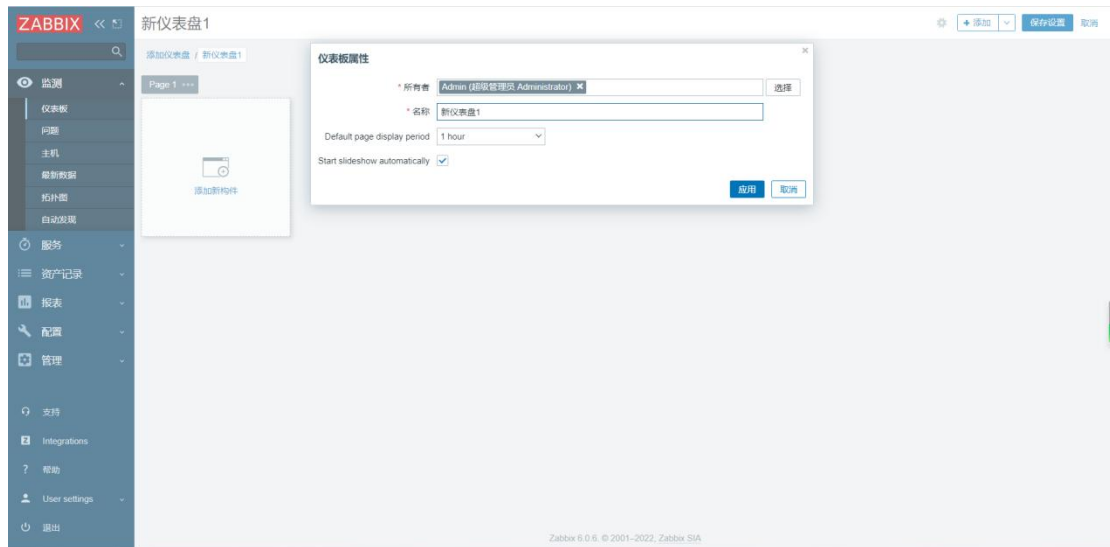
# 可视化

## 八十八、Zabbix 聚合图形配置指引

作为一款企业级监控工具，Zabbix 支持提供图形化的报表和图形展示功能。用户可以使用 Zabbix 聚合图形 (aggregate graph) 来汇总和展示多个数据项的图形，将多个监控指标放在同一个图形中，以便更好地展示整体趋势和变化。

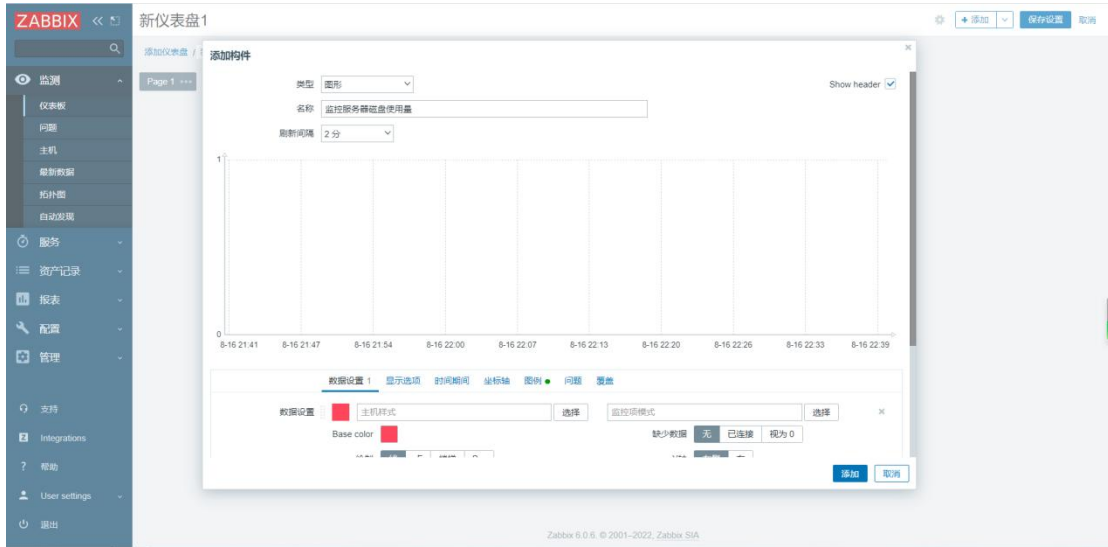
本文将详细介绍 Zabbix 6 版本聚合图形配置的过程：

1. 创建一个新的仪表盘：在 Zabbix 的 Web 界面中，选择“监测” > “仪表盘” > “添加仪表盘” > “创建仪表盘”，填写仪表盘名称后点击“应用”按钮即可。

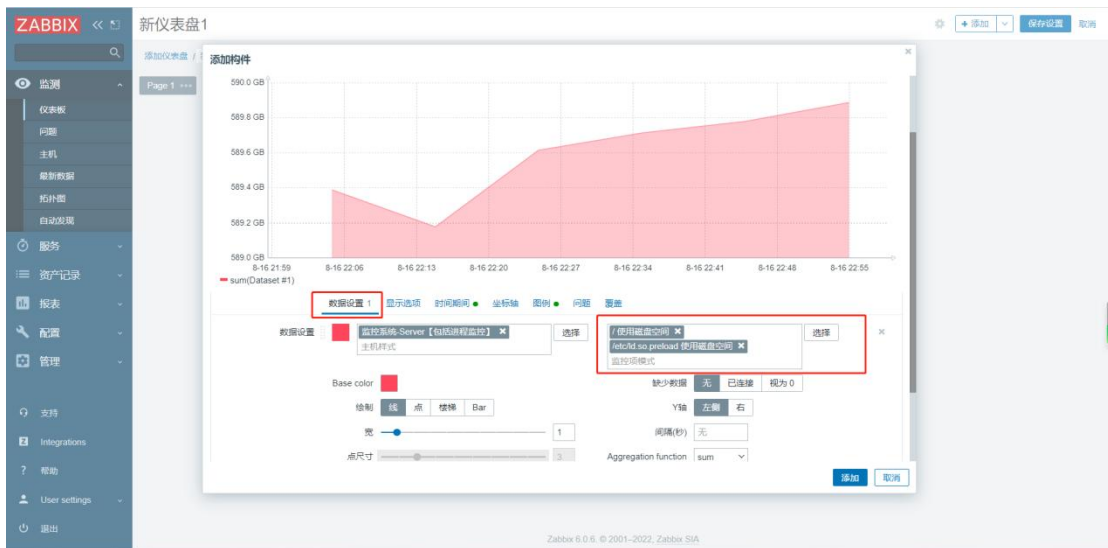


2. 创建一个新的聚合图形：点击空白位置，“类型”选择“图形”，填写构件“名称”。

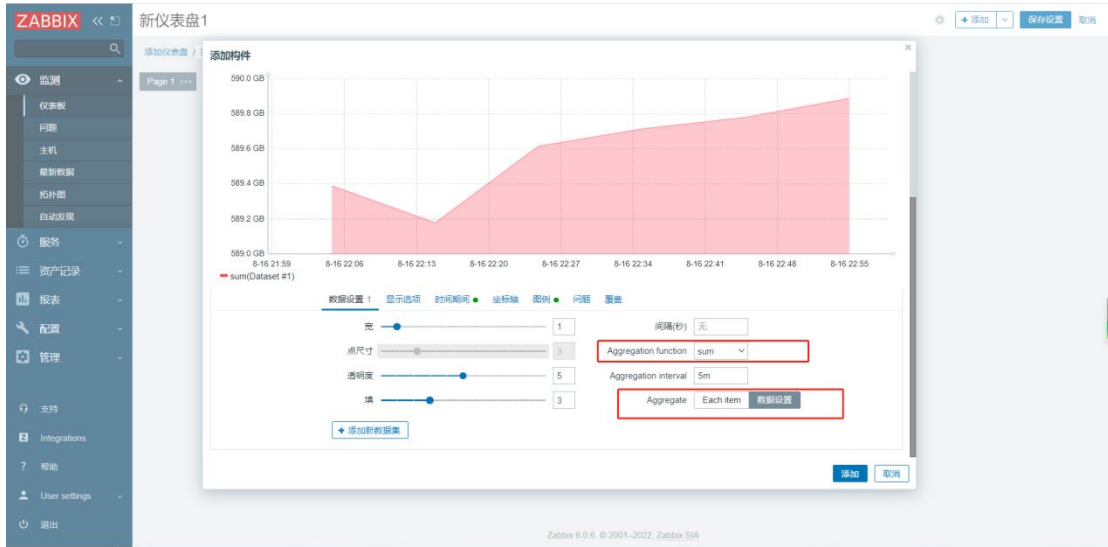
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



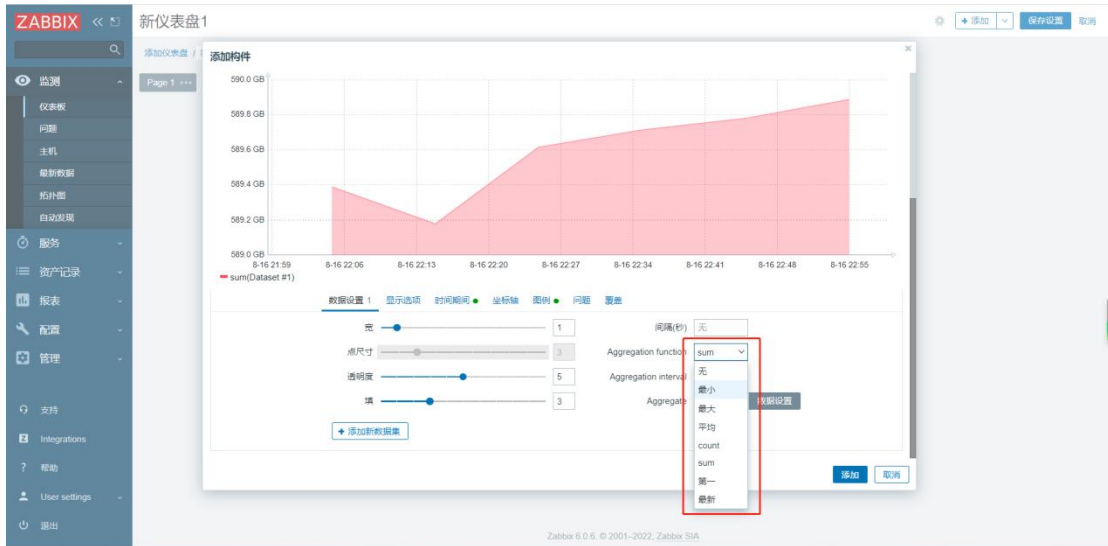
3.添加监控项：在“数据设置”标签页下，分别单击2个“选择”按钮，分别设置监控主机和监控项。可以选择多个监控项，并为数据集设置颜色。Aggregation function 设置为 sum 并且 Aggregate 设置为数据设置，即表示将监控项数据进行汇总计算。



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



也可根据需要，调整展示的数据方式，可选最大值、最小值、平均值、最新值等。

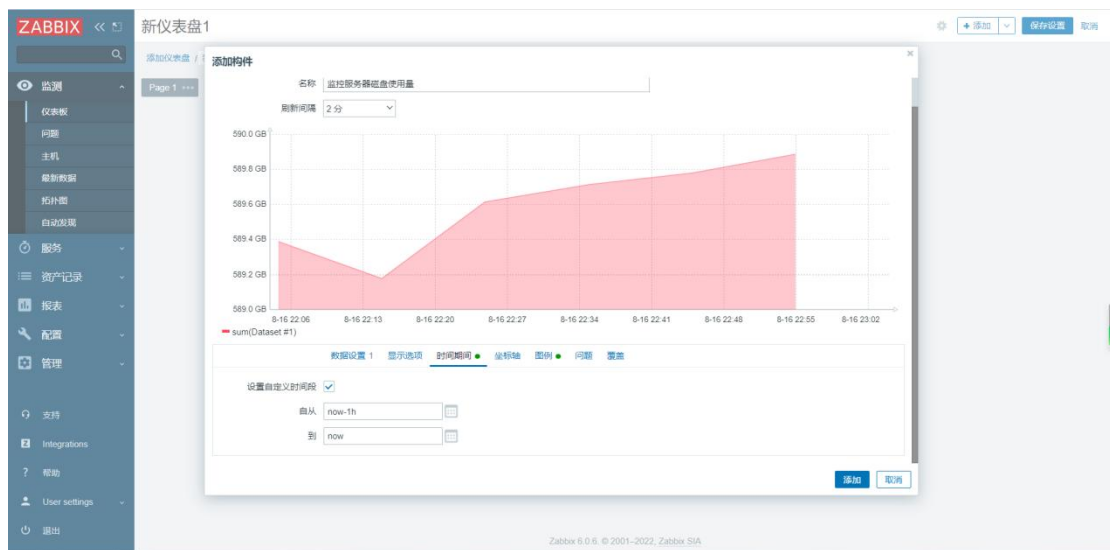


4.配置显示选项：在“显示选项”标签页下，默认设置“选择历史数据”为“自动”即可。“历史记录”表示每次采集到的实际数据，“趋势”表示监控项每1小时数据统计后的最大值、最小值、平均值，这一个小时的时间段是无法自定义的。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



5.配置时间范围：在“时间周期”标签页下，勾选“设置自定义时间段”，然后选择要显示的时间范围，默认是最近 1 小时。



6.配置坐标轴：在“时间周期”标签页下，可设置坐标轴的最大值、最小值，一般默认不做调整，由系统计算即可。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



7.配置图例：在“图例”标签页下，可设置是否展示图例，以及设置图例的行数。



配置完成后，点击添加即可。

通过以上步骤，用户可以轻松地配置 Zabbix 的聚合图形，以便更好地展示多个监控项的数据。

以上就是这一期的 Zabbix 技术知识分享。大家好，我是乐乐，专注运维技术研究与分享，关注我学习 Zabbix 等使用技巧，更多运维问题还可以到[乐维社区](https://forum.lwops.cn)留言提问哦~



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



wednesday!

每周三 pm.  
大神坐诊  
你问我答

QQ群：177428068  
617295020

Zabbix&Ansible主题日

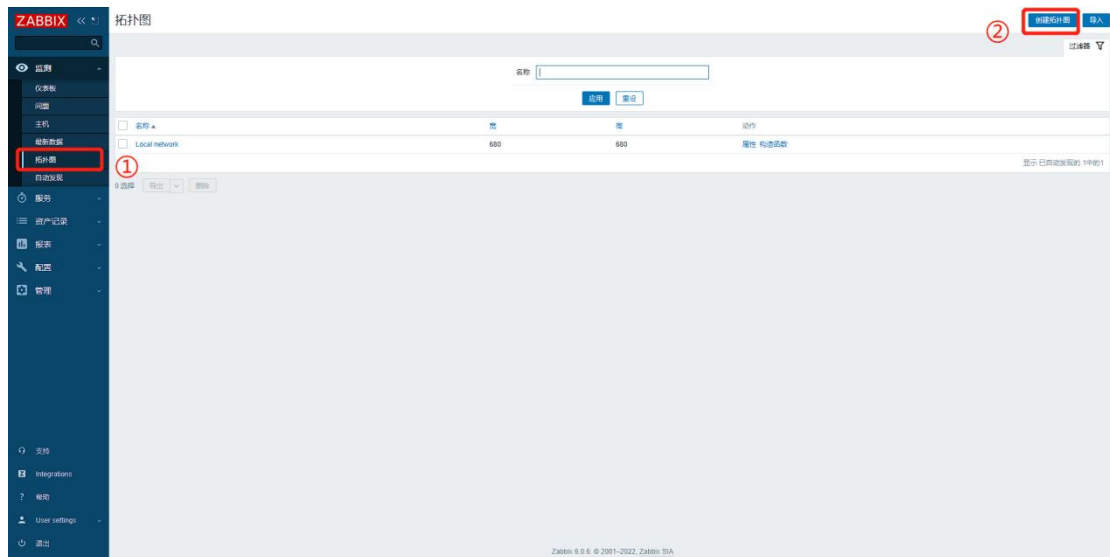
乐维  
www.lwops.cn

The banner features a dark blue background with a grid pattern. On the left, there is a large yellow ring with a red heart inside, and a small figure of a person sitting on the ring. A white speech bubble contains the QQ group numbers. On the right, the text 'Wednesday!' is in red and white, followed by '每周三 pm. 大神坐诊 你问我答' in white and yellow. A blue arrow points to the text 'Zabbix&Ansible主题日'. The bottom right corner has the '乐维' logo and website 'www.lwops.cn'.

## 八十九、Zabbix6.0 仪表盘配置简易网络拓扑教程

### 一、新建拓扑图

#### 1. 点击新建按钮



#### 2. 配置网络拓扑图信息 (PS:所有字段后续可以修改调整, 所以这边可以先随意配置)

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

## 网络拓扑图

拓扑图 分享

\*所有者 Admin (Zabbix Administrator) ✕ 选择

\*名称 简易网络拓扑

\*宽 800

\*高 800

背景图片 没有图片 ▾

自动的图标映射 <手册> ▾ 展示图标映射

图标高亮

触发器状态上的标记组件改变

显示问题 展开单一问题 问题数量 问题的数量和扩大最关键的一个

高级标签

拓扑元素标签类型 标签 ▾

拓扑元素标签位置 底部 ▾

问题显示 所有 ▾

最小的严重级别 未分类 信息 警告 一般严重 严重 灾难

显示处理的问题

URLs

名称	URL	组件	动作
添加			

添加 取消

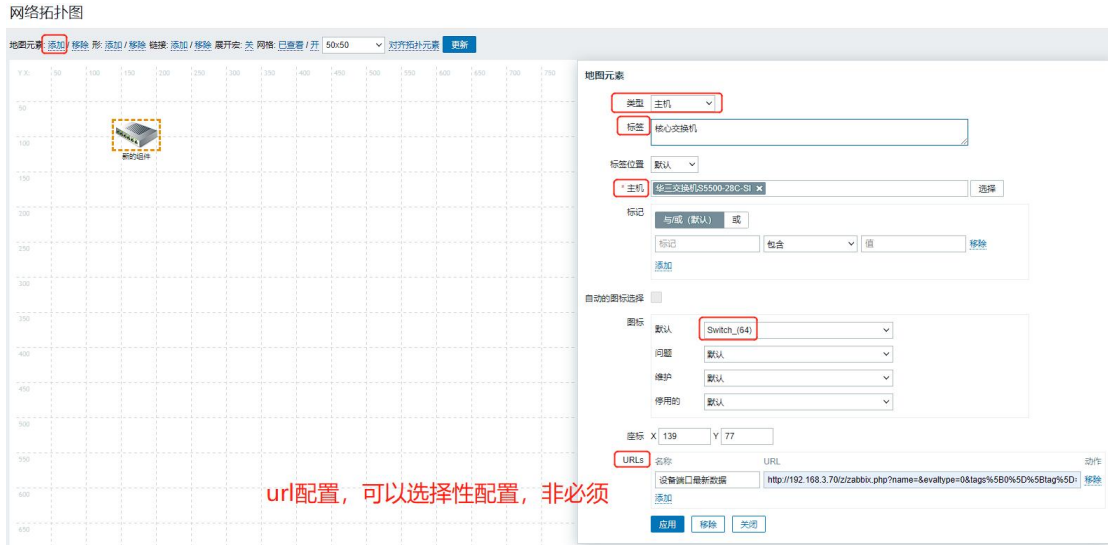
3. 添加完成后，“拓扑图”菜单中点击新建的拓扑名称进入拓扑页

然后点击右上角“编辑拓扑图”按钮，进入到拓扑图编辑界面

## 二、拓扑图绘制

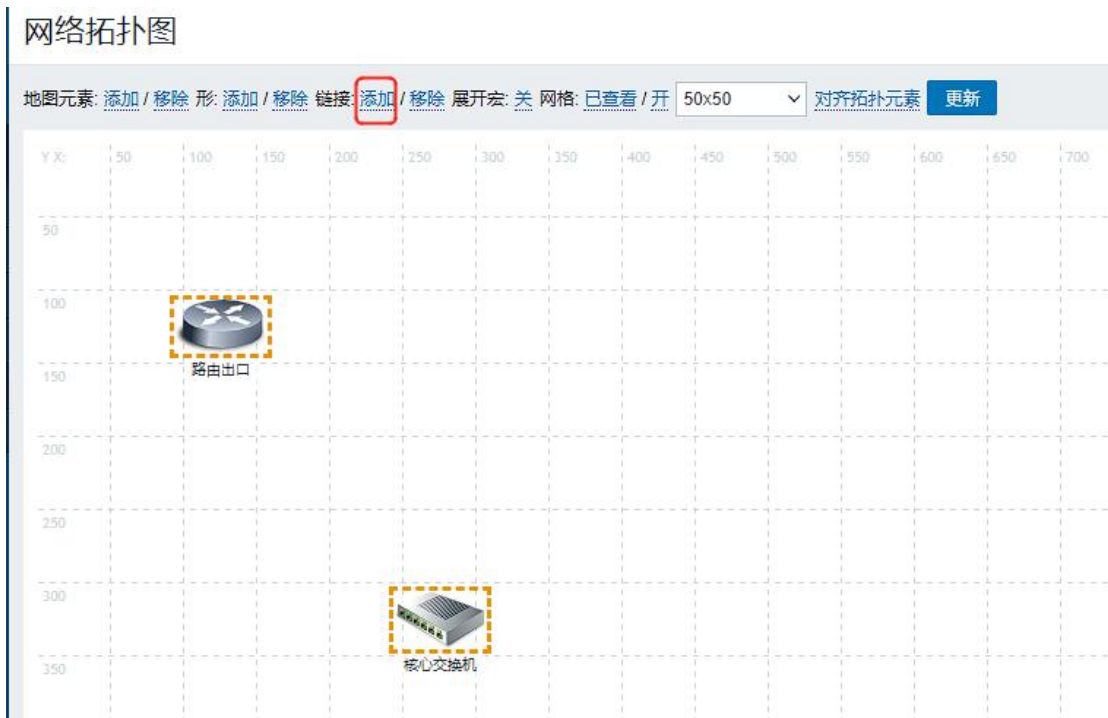
1.添加已经监控的网络设备主机

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



2. 如图操作添加多个设备后，设备间两两选择添加“链接”

a) 安装 ctrl 选择两台网络设备，然后点击链路“添加”按钮



b) 编辑链路配置绑定触发器及指标标签

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

**批量更新组件**

所选的组件

类型	名称
主机	华三SR8802-X-S
主机	华三交换机S5500-28C-SI

标签

标签位置 默认

自动的图标选择

图标(默认) Cloud\_(24)

图标(问题) 默认

图标(维修) 默认

图标(停用的) 默认

链接 自从 到 链接指示器 动作

华三交换机S5500-28C-SI 华三SR8802-X-S

链接 自从 到 链接指示器 动作

华三交换机S5500-28C-SI 华三SR8802-X-S

标签 带宽:1000Mbps  
上行速率:{?last(/S5500-28C-SI/ifHCOutOctetsPersecond[GigabitEthernet1/0/23])}  
下行速率:{?last(/S5500-28C-SI/ifHCInOctetsPersecond[GigabitEthernet1/0/23])}

类型(OK) 线

Color (OK) ■

链接指示器	触发器	类型	Color	动作
华三交换机S5500-28C-SI: [网络设备]华三交换机S5500-28C-SI设备端口GigabitEthernet1/0/23处于DOWN状态		线	<span style="color: red;">■</span>	<input type="button" value="移除"/>
华三交换机S5500-28C-SI: [网络设备]华三交换机S5500-28C-SI设备端口GigabitEthernet1/0/23带宽发送利用率大于90%		线	<span style="color: red;">■</span>	<input type="button" value="移除"/>
华三交换机S5500-28C-SI: [网络设备]华三交换机S5500-28C-SI设备端口GigabitEthernet1/0/23带宽接收利用率大于90%		线	<span style="color: red;">■</span>	<input type="button" value="移除"/>

标签使用监控项数据，格式参考 zabbix 官方手册

### 1 配置网络地图 (zabbix.com)

链接属性:

参数	说明
标签	将在链接顶部呈现的标签。 此字段支持表达式法，但仅限于 avg、last、min 和 max 函数，以时间为参数 (例如，{?avg (/host/key, 1h)})。

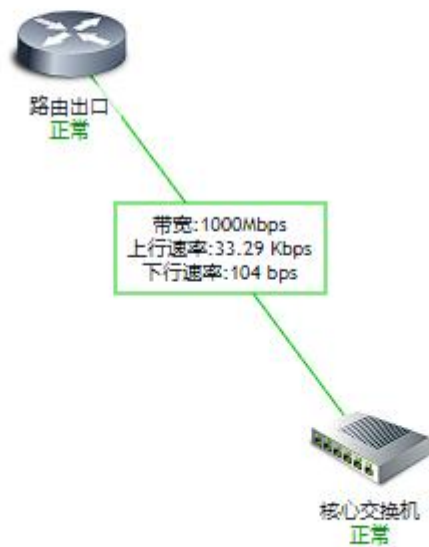
### 3.更新拓扑图，查看效果

整理 by 乐维社区 (<https://forum.lwops.cn>)

## 网络拓扑图

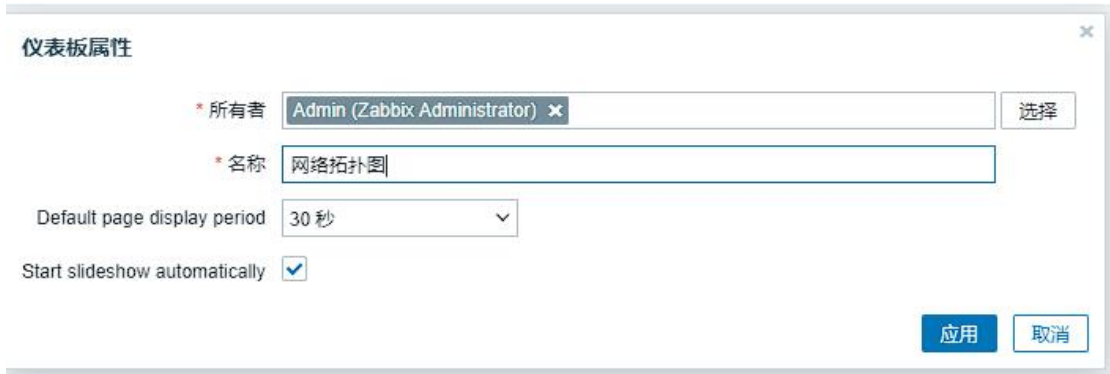
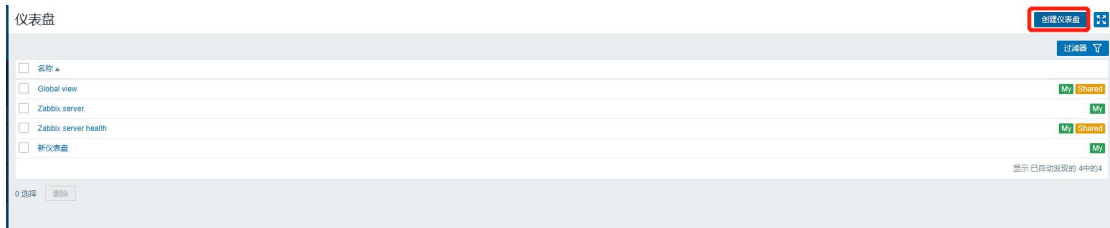
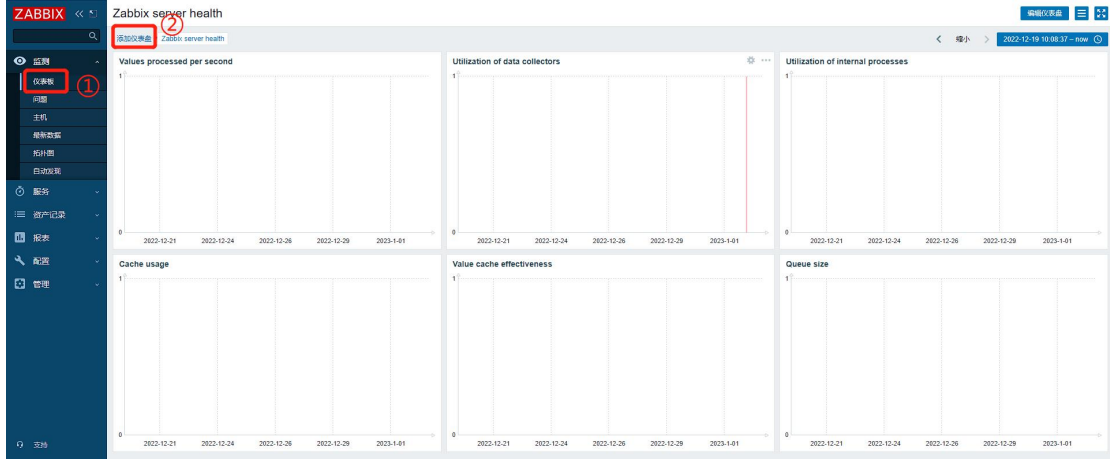


### 所有地图 / 简易网络拓扑



### 三、配置首页仪表盘使用此拓扑图

#### 1. 创建新仪表盘

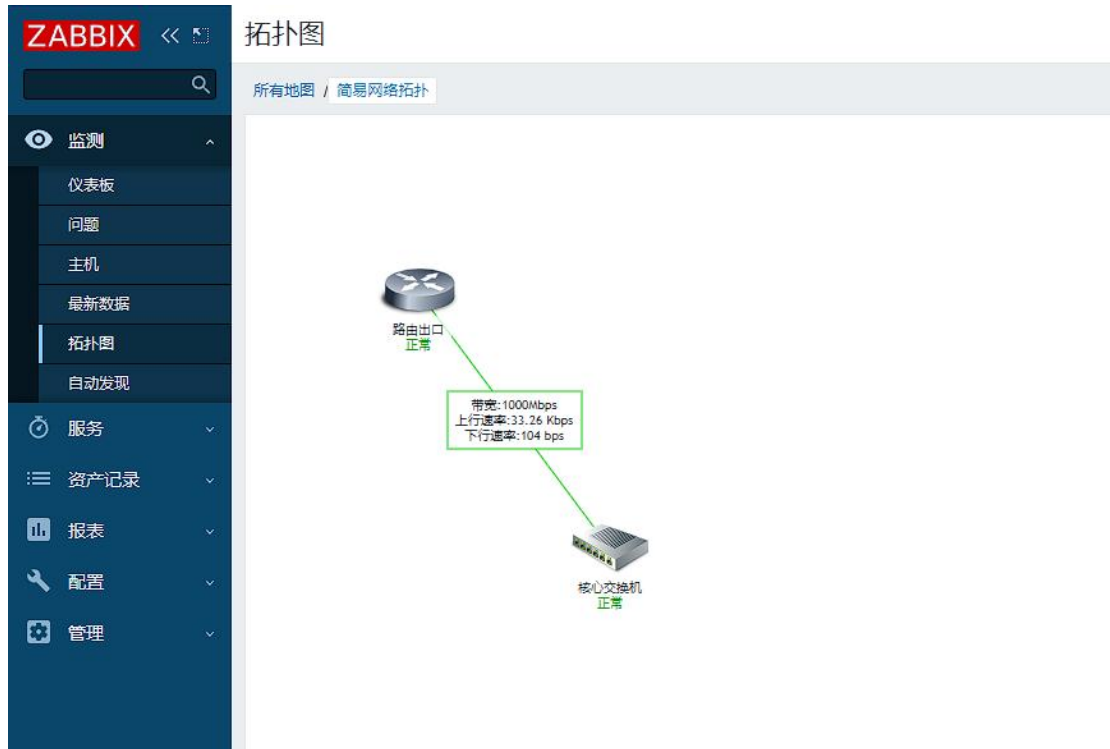


#### 2. 添加卡片组件，选择拓扑图



#### 3. 效果查看

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



更多.....



# 第三方平台对接

## 九十、Zabbix 与 Prometheus 区别

Zabbix 和 Prometheus 都是非常流行的监控系统。它们有许多相似之处，但也有一些不同之处。以下是 Zabbix 和 Prometheus 监控对比的一些关键点：

### 1、数据模型和查询语言

Prometheus 使用一个称为 PromQL 的查询语言来查询和处理时间序列数据。PromQL 支持许多数据模型和查询功能，包括度量标准、标签和聚合函数。Zabbix 使用自己的数据模型和查询语言，包括项、触发器和动作等概念。

### 2、存储方式

Prometheus 使用一种称为 TSDB 的时间序列数据库来存储时间序列数据。TSDB 使用一种称为 WAL 的写前日志，以确保数据的可靠性。Zabbix 使用关系型数据库来存储数据。

### 3、自动化和配置管理

Prometheus 具有自动化和自动配置的能力，它可以自动发现服务和指标，并对它们进行监控。Zabbix 也提供了类似的功能，但需要手动配置。

### 4、可视化和警报

Zabbix 和 Prometheus 都支持可视化和警报功能。Zabbix 提供了一个基于 Web 的前端界面，可以查看监控数据和设置警报。Prometheus 通常与 Grafana 等工具一起使用，以实现更高级的可视化和警报功能。

### 5、性能和扩展性

Prometheus 在性能和扩展性方面表现良好，能够处理大规模的时间序列数据。Zabbix

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

也具有良好的性能和扩展性，但在大规模监控方面可能需要更多的资源和配置。

综上所述，Zabbix 和 Prometheus 都是出色的监控系统，具有各自的优势和适用场景。

**zabbix** 更加适合用于 **本地计算机** 的监控，而 **Prometheus** 更适合在现在流行的 **云计算** 监控上使用。

大家好，我是乐乐，专注运维技术研究与分享，关注我，了解更多运维知识。如有问题也可以到乐维社区 (<https://forum.lwops.cn/>) 留言提问，与运维技术爱好者共同交流心得。

# 九十一、Zabbix 对接 Prometheus 实操——基于 Prometheus pattern 监控

## 概述

得益于对云原生和容器监控的友好支持，如今，Prometheus 监控受到越来越多企业的青睐。然而，对于已经部署了 Zabbix 监控系统的企业，想要用 Prometheus 完全替换 Zabbix，可能既无必要，短期也不现实。实际上，Zabbix 4.2 及后续版本增加了对 Prometheus 数据源的接入，已经能够实现用 Zabbix 来对接 Prometheus 监控。

Prometheus 通过 Exporters 组件来收集数据。Exporters 是一类数据采集组件的总称，它负责从目标处搜集数据，并将其转化为 Prometheus 支持的格式，并且暴露出一个 HTTP API 地址，等待 Prometheus Server 拉取数据并进行数据处理。

Zabbix 对接 Prometheus 也是通过 HTTP 代理，拉取 Exporters 提供的大量 Prometheus 指标数据，然后通过内置的 Prometheus pattern 进行数据的处理和筛选，从而获取监控值。

本文将介绍如何使用 Zabbix 的 Prometheus pattern 项来对接 Prometheus 数据源。以 Prometheus 官方提供的 Node\_exporter 采集器为例。

## Node\_exporter 部署

### 下载 Node\_exporter 部署包

Node\_exporter 部署包可以从 Prometheus 官网进行下载。

地址：<https://prometheus.io/download/>

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

## node\_exporter

Exporter for machine metrics [prometheus/node\\_exporter](#)

1.5.0 / 2022-11-29 <a href="#">Release notes</a>				
File name	OS	Arch	Size	SHA256 Checksum
<a href="#">node_exporter-1.5.0.darwin-amd64.tar.gz</a>	darwin	amd64	4.58 MiB	<a href="#">aa6742953f1b194e8093026e19ee2700f4fe6401ce4592b4c2894a32d7cbb</a>
<a href="#">node_exporter-1.5.0.linux-amd64.tar.gz</a>	linux	amd64	9.71 MiB	<a href="#">ef999f231ab54e23a34b9f0b10c28e9acee9e95a05a5e5edf3de85437db7aebb</a>

可以选择所需的版本，然后复制相关链接在服务器上面使用 `wget` 或 `curl` 进行下载，抑或  
直接下载后再上传到服务器。

## 解压安装

下载好后，直接进行解压：

```
[root@localhost prometheus]# ls
node_exporter-1.5.0.linux-amd64.tar.gz
node_exporter-1.5.0.linux-amd64
node_exporter-1.5.0.linux-amd64.tar.gz
[root@localhost prometheus]# pwd
/data/test/prometheus
[root@localhost prometheus]#
```

## 编写 systemd 启动服务

```
vim /usr/lib/systemd/system/node_exporter.service
```

将以下内容复制到文件中：

```
[Unit]
Description=node-exporter service
After=network.target

[Service]
User=prometheus
Group=prometheus
KillMode=control-group
Restart=on-failure
RestartSec=60
ExecStart=/data/test/prometheus/node_exporter-1.5.0.linux-amd64/node_exporter \
  --web.listen-address=:9100 \
  --collector.systemd \
  --collector.systemd.unit-whitelist=(sshd|nginx).service \
  --collector.processes \
  --collector.tcpstat

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

注意，启动命令的路径要根据实际的路径进行修改。

## 创建普通用户 prometheus

useradd prometheus

## 启动 Node\_exporter 服务

systemctl daemon-reload

systemctl start node\_exporter.service

systemctl status node\_exporter.service

```
[root@localhost ~]# systemctl status node_exporter.service
● node_exporter.service - node_exporter service
   Loaded: loaded (/usr/lib/systemd/system/node_exporter.service; disabled; vendor preset: disabled)
   Active: active (running) since 2023-02-22 14:50:53 CST; 4 days ago
     Main PID: 8345 (node_exporter)
        Tasks: 6
       Memory: 28.4M
      CGROUP: /system.slice/node_exporter.service
           └─8345 /data/test/prometheus/node_exporter-1.5.0.linux-amd64/node_exporter --web.listen-address=:9100 --collector.systemd --collector.systemd.unit-whitelist=(ssh|nginx).service --collector.processes --collector.tcp...

2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.068Z caller=node_exporter.go:117 level=info collector=thermal_zone
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.068Z caller=node_exporter.go:117 level=info collector=time
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=node_exporter.go:117 level=info collector=timex
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=node_exporter.go:117 level=info collector=udp_queues
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=node_exporter.go:117 level=info collector=uname
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=node_exporter.go:117 level=info collector=wstat
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=node_exporter.go:117 level=info collector=xfs
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=node_exporter.go:117 level=info collector=xfs
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=tls_config.go:232 level=info msg="Listening on" address=[::]:9100
2月 22 14:50:53 localhost.localdomain node_exporter[8345]: ts=2023-02-22T09:50:53.069Z caller=tls_config.go:235 level=info msg="TLS is disabled." http2=false address=[::]:9100
```

## 访问测试

然后访问 IP:9100 端口

```
← → C ▲ 不安全 | 192.168.3.141:9100/metrics

# HELP go_gc_duration_seconds & summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.8449e-05
go_gc_duration_seconds{quantile="0.25"} 2.8166e-05
go_gc_duration_seconds{quantile="0.5"} 3.3589e-05
go_gc_duration_seconds{quantile="0.75"} 3.9873e-05
go_gc_duration_seconds{quantile="1"} 7.5482e-05
go_gc_duration_seconds_sum 4.062974748
go_gc_duration_seconds_count 116839
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.19.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.802296e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 2.2381205188e+11
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 2.377195e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 4.087411389e+09
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 9.559824e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 2.802296e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 3.047424e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 4.849664e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 28285
```

如图所示表示部署成功，并且成功采集到数据。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

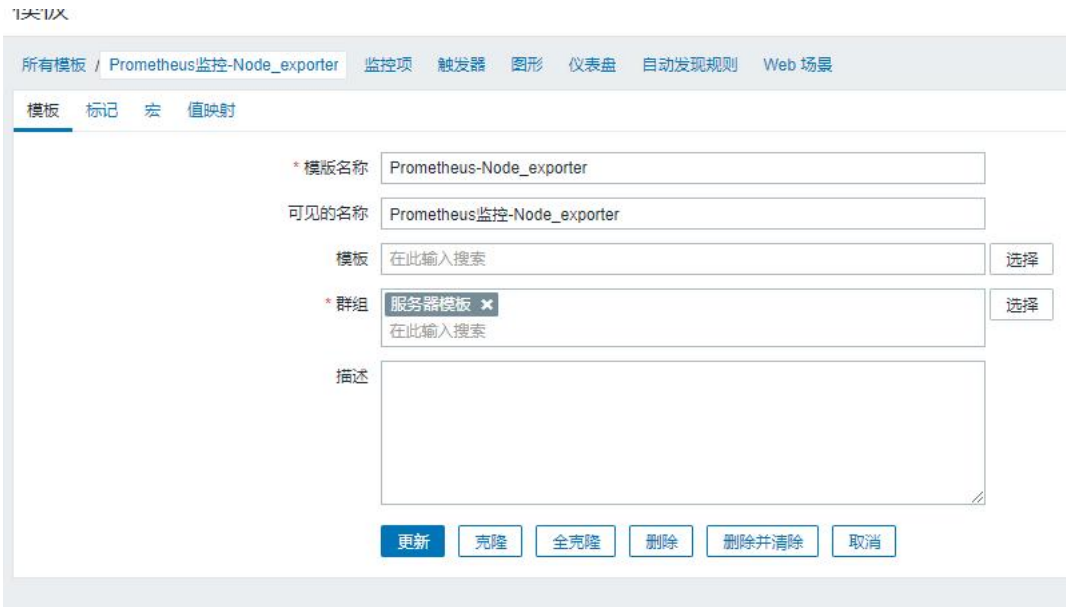
## Node\_exporter 相关指标说明

Node\_exporter 是 Prometheus 用于监控服务器的采集器，其相关的采集指标说明可以参考以下地址，然后按需进行监控：

参考指标地址：[https://blog.csdn.net/qq\\_33326449/article/details/126663517](https://blog.csdn.net/qq_33326449/article/details/126663517)

## Zabbix 对接 Node\_exporter

### 创建监控模板

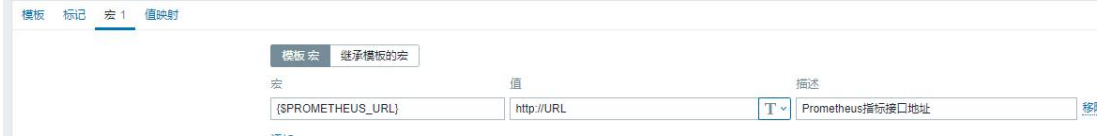


The screenshot shows the Zabbix web interface for creating a new template. The breadcrumb navigation is '所有模板 / Prometheus监控-Node\_exporter'. The current page is '模板' (Templates). The form fields are as follows:

- \* 模板名称: Prometheus-Node\_exporter
- 可见的名称: Prometheus监控-Node\_exporter
- 模板: 在此输入搜索 (with a '选择' button)
- \* 群组: 服务器模板 (with a dropdown icon and '选择' button)
- 描述: (empty text area)

At the bottom, there are buttons for '更新', '克隆', '全克隆', '删除', '删除并清除', and '取消'.

添加一个宏值：用以灵活监控多个主机



The screenshot shows the '宏 1 值映射' (Macro 1 Value Mapping) configuration page. It features a table with the following content:

宏	值	描述
{\$PROMETHEUS_URL}	http://URL	Prometheus指标接口地址

There is a '添加' (Add) button below the table and a '模板宏 继承模板的宏' (Template Macro Inherit Template Macro) checkbox.

### 创建 HTTP 代理监控项，获取大量 Prometheus 指标

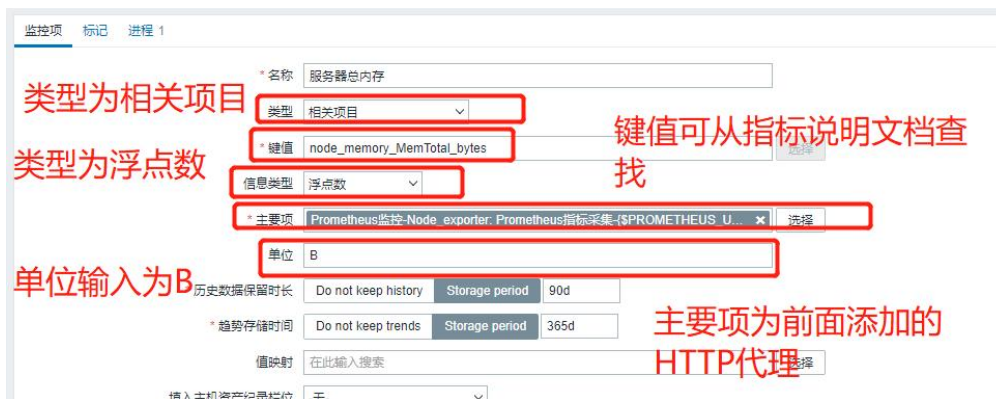
首先使用 HTTP 代理监控项，获取 Prometheus 的指标数据。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

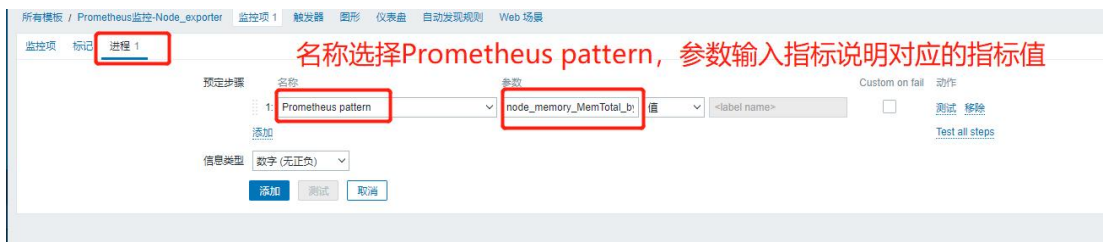


## 创建相关项目监控项，使用 Prometheus pattern 获取监控值

这里以监控服务器的内存大小为例：



然后添加处理步骤：



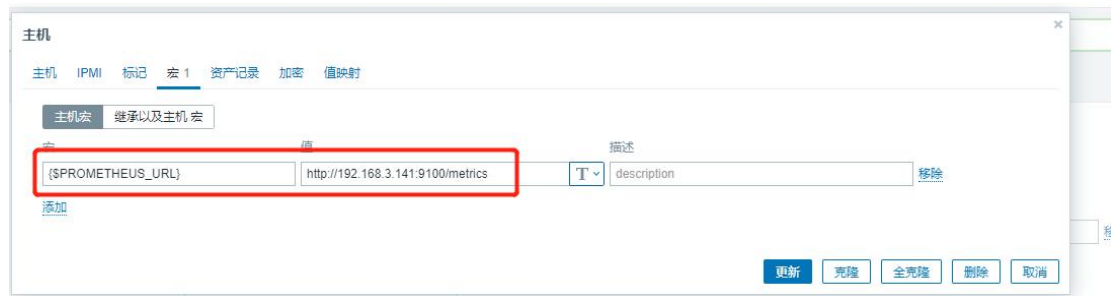
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

## 添加监控主机

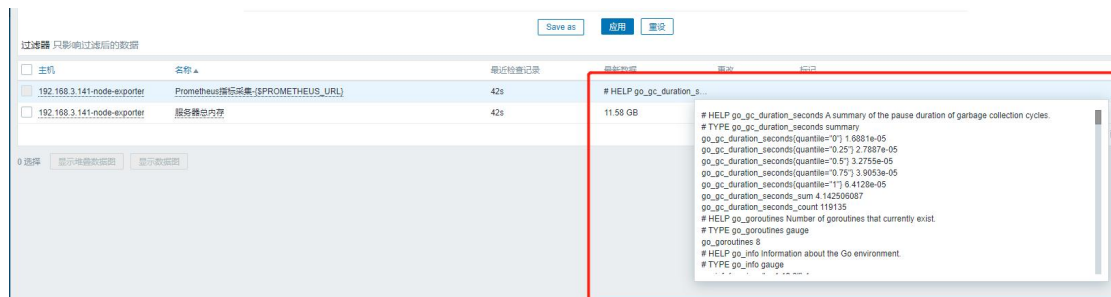
然后添加主机进行测试



修改宏值：



## 查看指标数据



如图所示，成功对接，并且获取到监控数据。

以上就是这一期的内容。大家好，我是乐乐，专注运维技术与分享，关注我，了解更多



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

运维知识。更多 Zabbix、Prometheus 等技术知识，还可关注[乐维社区](#)，如有问题也可以到社区进行留言提问，与广大运维技术爱好者一起探讨。

## 九十二、开源监控 zabbix 对接可视化工具 grafana 教程

今天要给大家介绍的是开源监控工具 zabbix 对接可视化工具 grafana 问题。



有一定运维经验的小伙伴大抵都或多或少使用过、至少也听说过开源监控工具 zabbix，更进一步的小伙伴可能知道 zabbix 在数据呈现方面有着明显的短板，因此需要搭配第三方的可视化工具使用。

grafana 就是这样一款第三方可视化工具，它主要用于大规模指标数据的可视化展现，是网络架构和应用分析中最流行的时序数据展示工具。grafana 官方库提供丰富的图表类型，如热图、直方图、折线图和地图等，以适应不同的数据展示需求。下面我们直接进入 zabbix 对接 grafana 的操作：

### 一、对接前提

#### 1.1 已经安装好 zabbix 且有监控数据

## 二、安装 grafana

### 2.1 安装 grafana

```
yum install -y https://dl.grafana.com/enterprise/release/grafana-enterprise-10.4.
```

```
1-1.x86_64.rpm
```

### 2.2 启动 grafana 服务

```
service grafana-server start
```

应用端口默认为 3000

用户名:admin

密码:admin

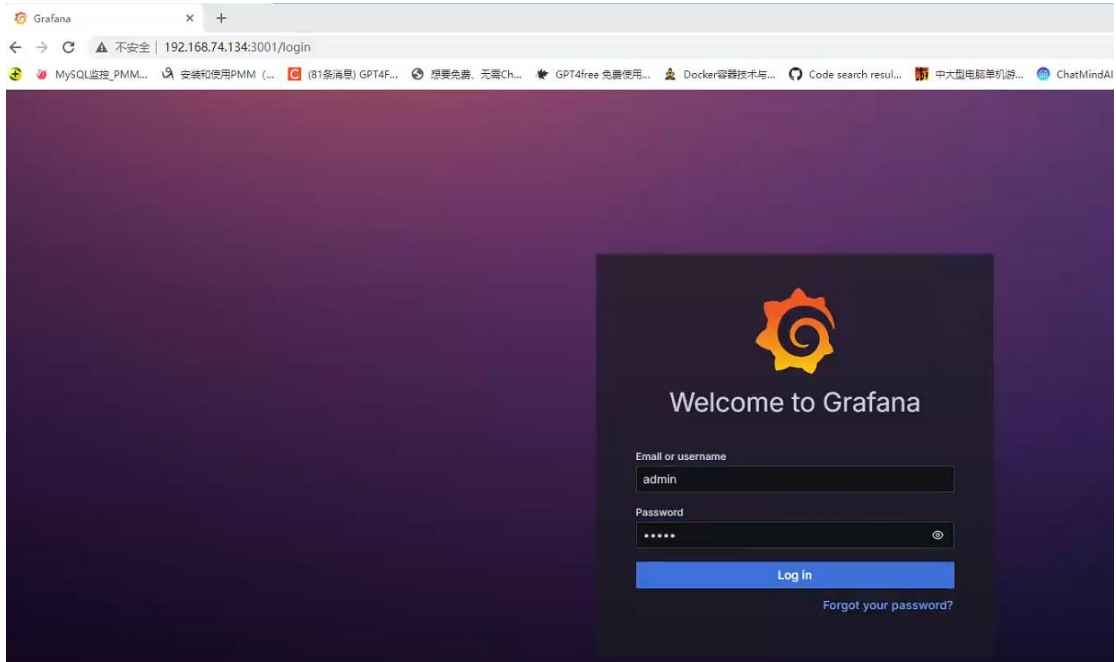
如果需要修改服务端口，修改/etc/grafana/grafana.ini 配置文件中的 http\_port

```
;http_addr =  
  
# The http port to use  
http_port = 3001
```

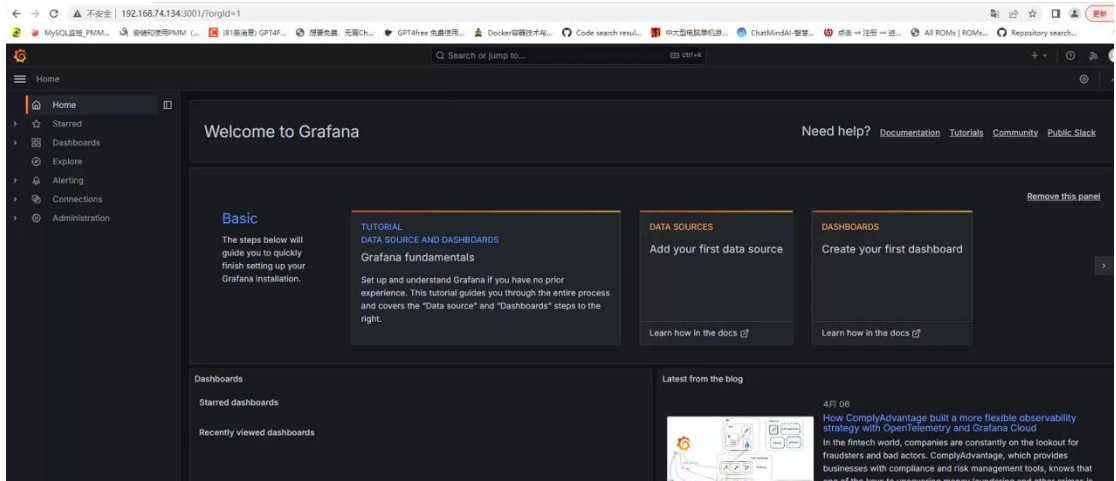
因本地服务端口冲突。本次实现修改 grafana 端口为 3001

用浏览器访问 <http://ip/3001>

本文档为样章，完整版文档请添加乐乐（lerwee）获取

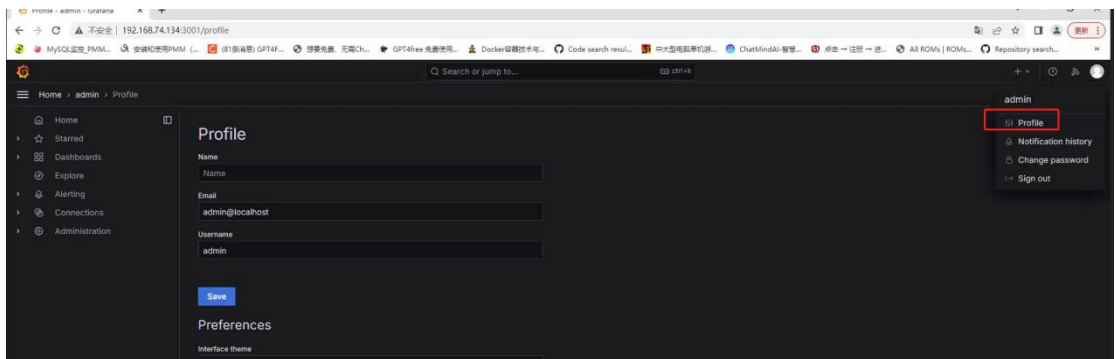


成功登录 grafana 的首页



2.3 汉化配置界面

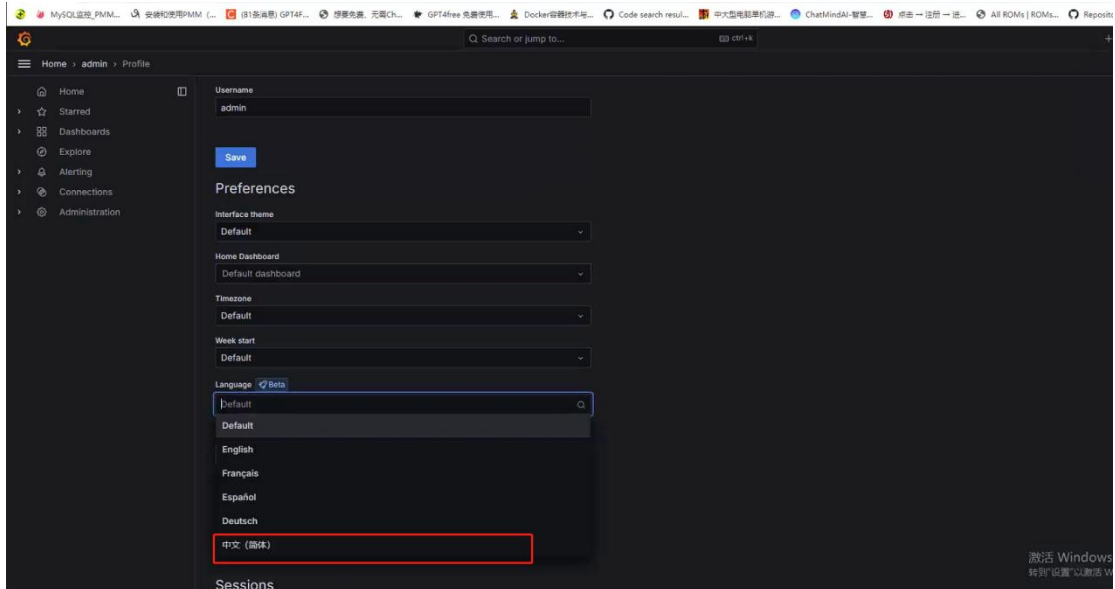
打开参数设置界面



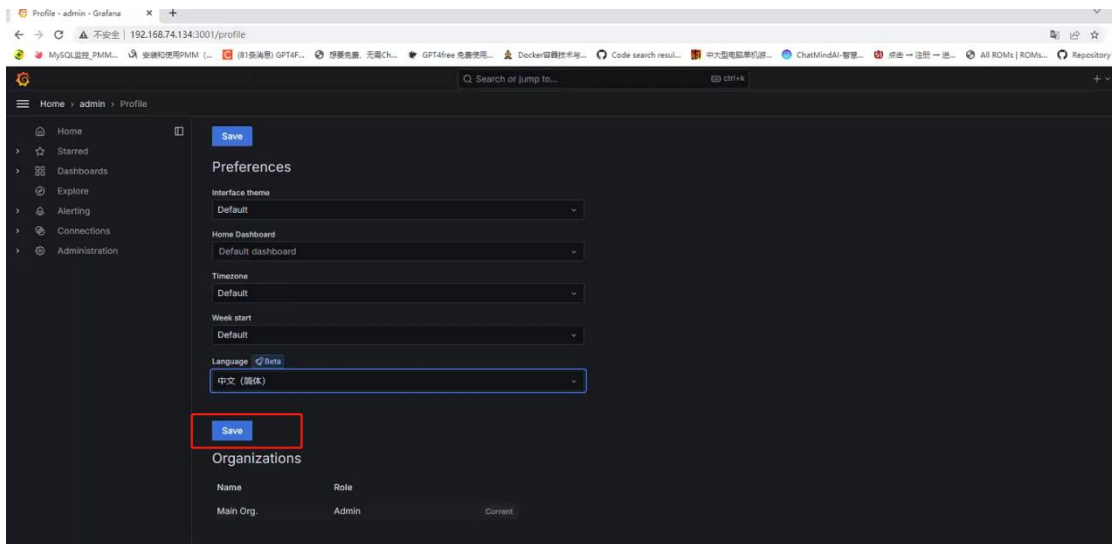
选择语言为中文(简体)

整理 by 乐维社区 (https://forum.lwops.cn)

本文档为样章，完整版文档请添加乐乐（lerwee）获取



点击保存



### 三、对接 zabbix 数据

#### 3.1 安装 zabbix 插件

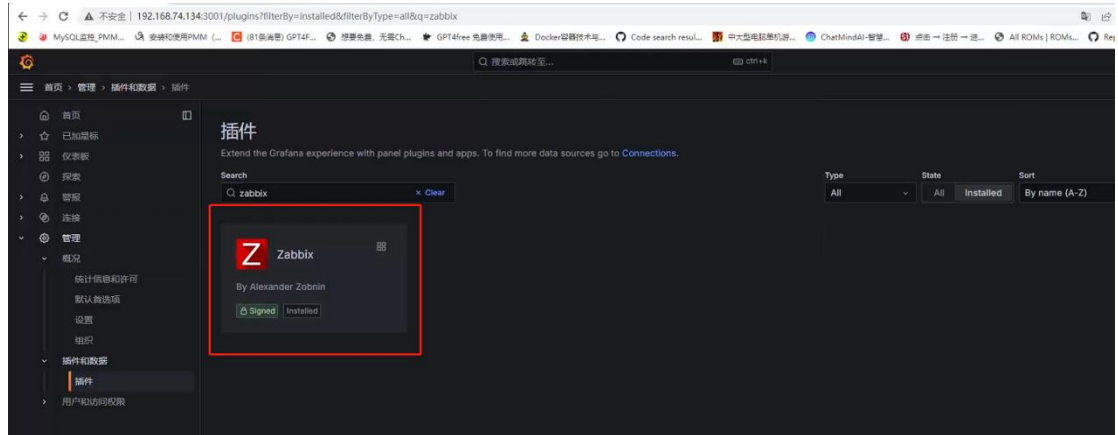
grafana-cli plugins install alexanderzobnir-zabbix-app

重启 grafana-server 服务

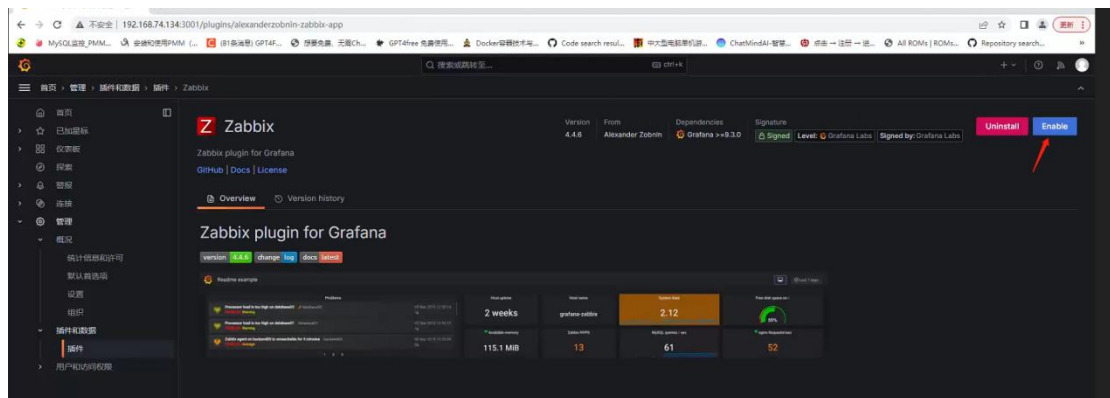
systemctl restart grafana-server.service

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

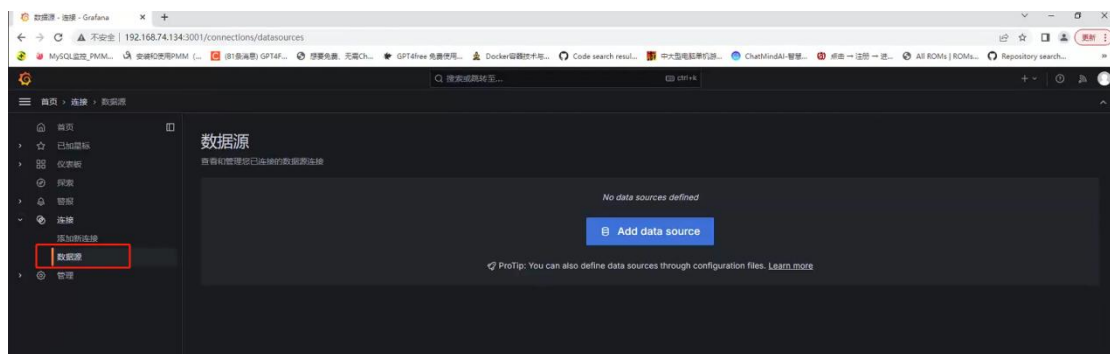
上述步骤操作后，可在图片中对应的菜单位置看到 zabbix 插件已经安装



启用 zabbix 插件

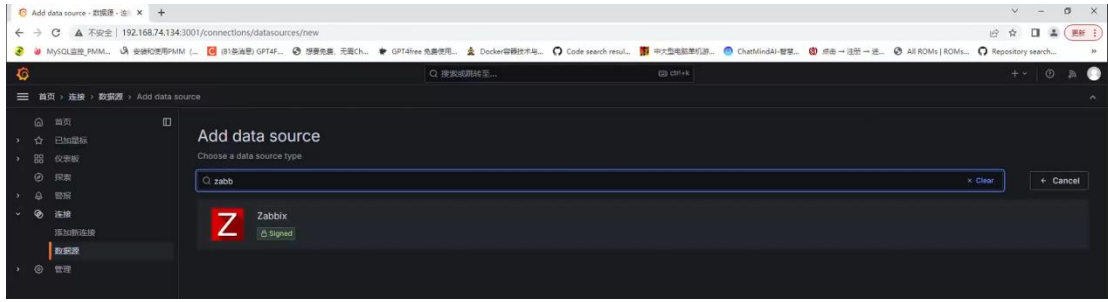


3.2 创建 zabbix 数据源

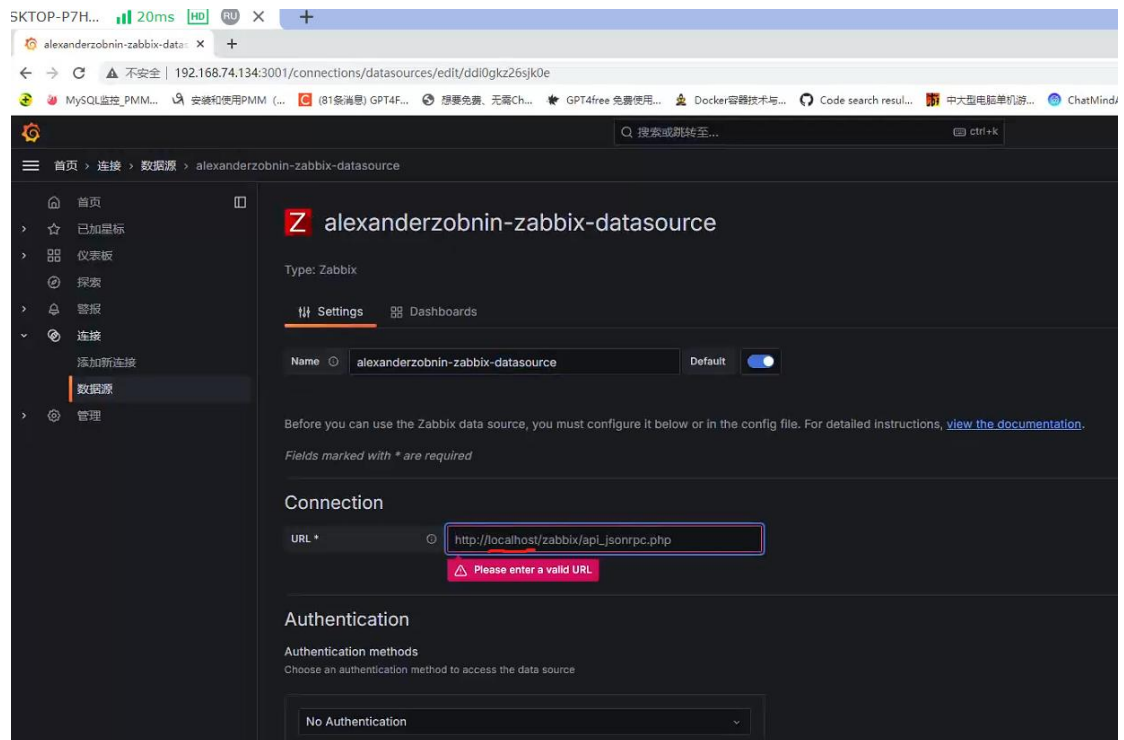


整理 by 乐维社区 (<https://forum.lwops.cn>)

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

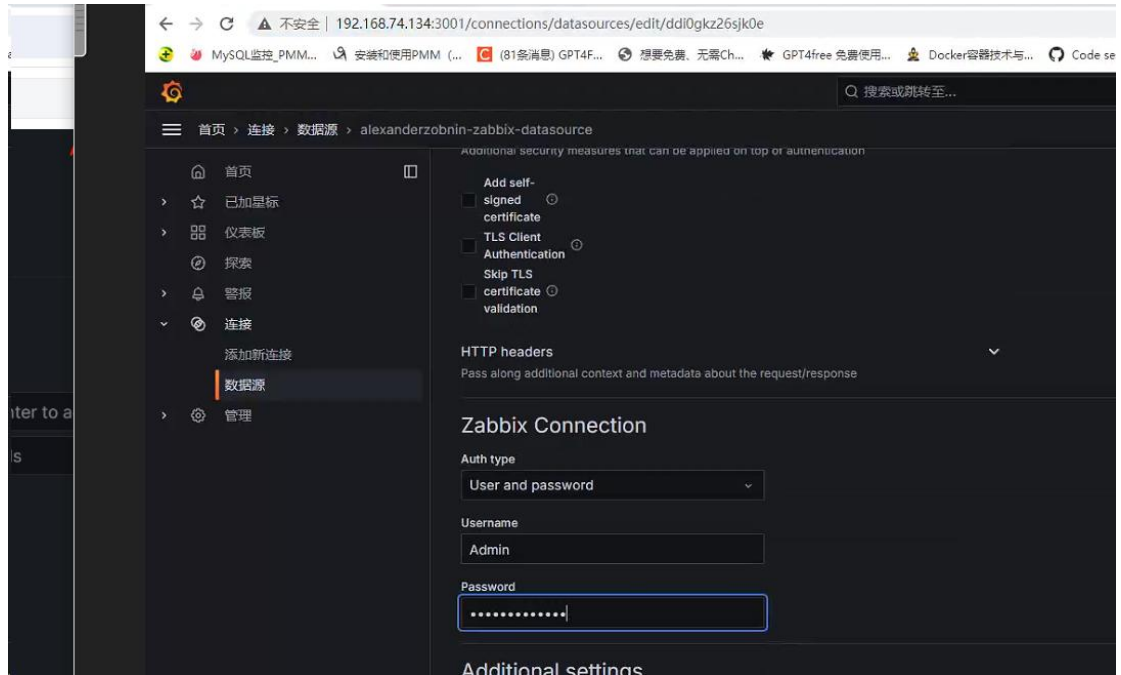


如图片的位置，填写 zabbix 服务端 ip

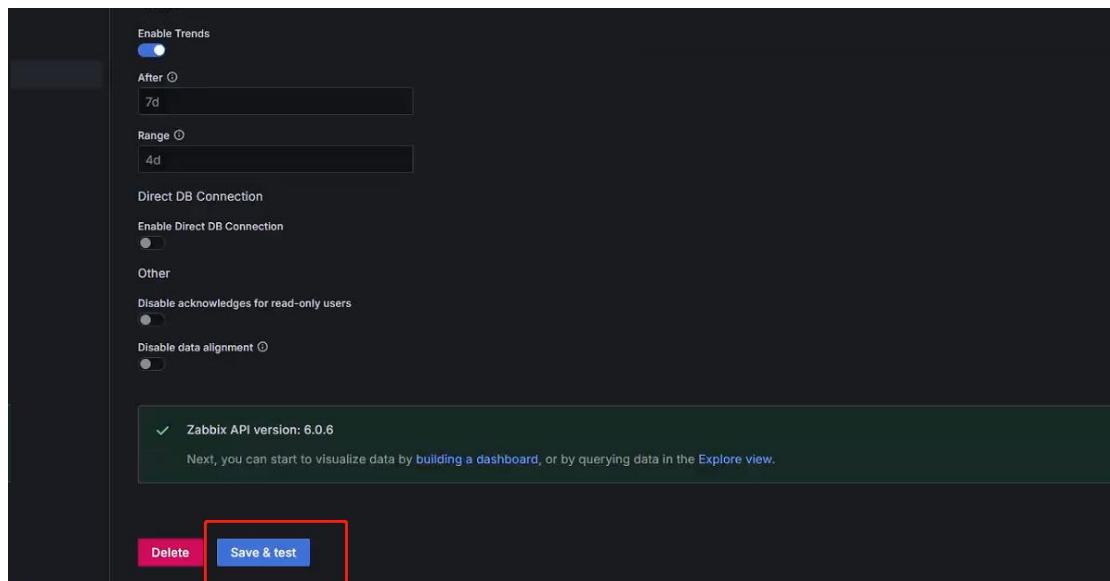


填写 zabbix\_server 的登陆账号和密码

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



点击保存测试，出现下面提示，说明已经连接成功

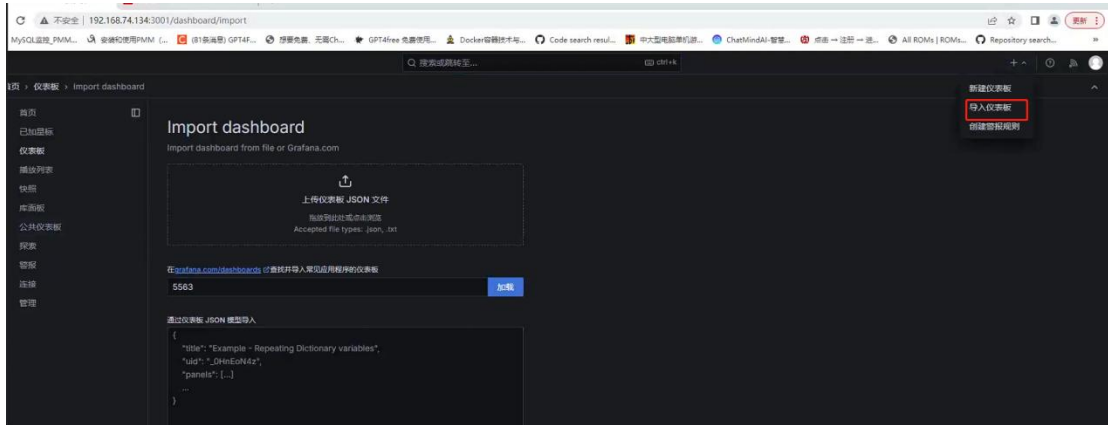


### 3.3 使用 grafana 官方的模版

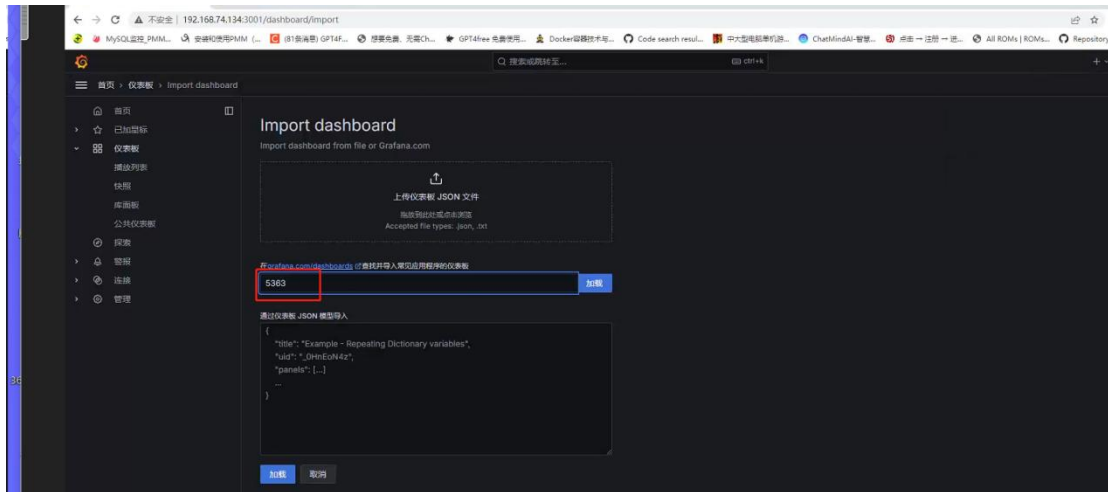
选择导入模版



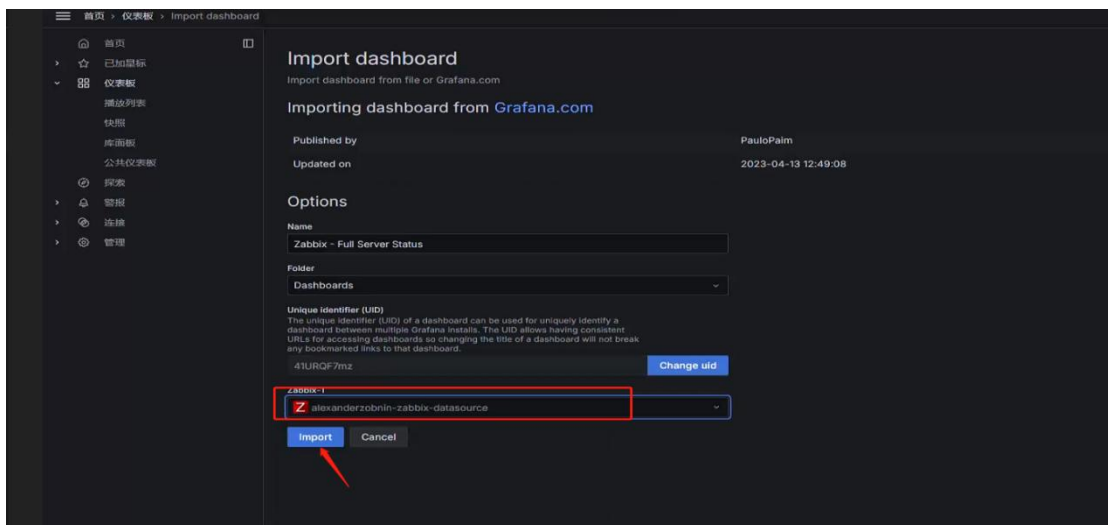
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



选择模版进行 5363 是一个 zabbix 的模版。输入模版 id，点击加载

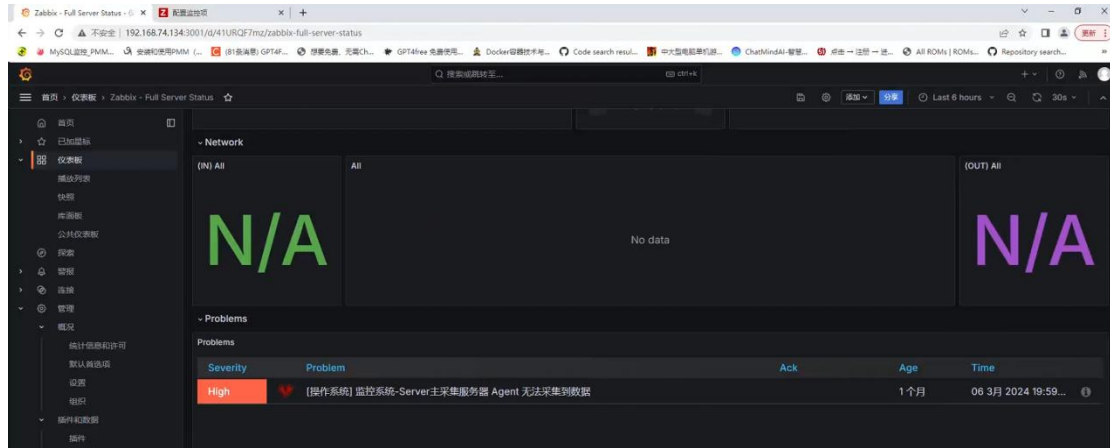


选择配置好的 zabbix 的数据源。点击 import 按钮进行模版导入



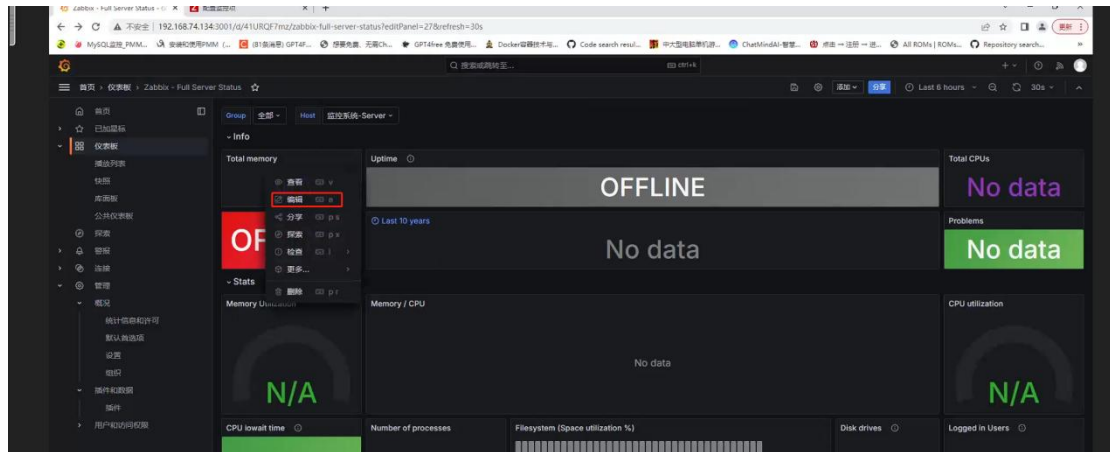
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

## 查看导入的模版

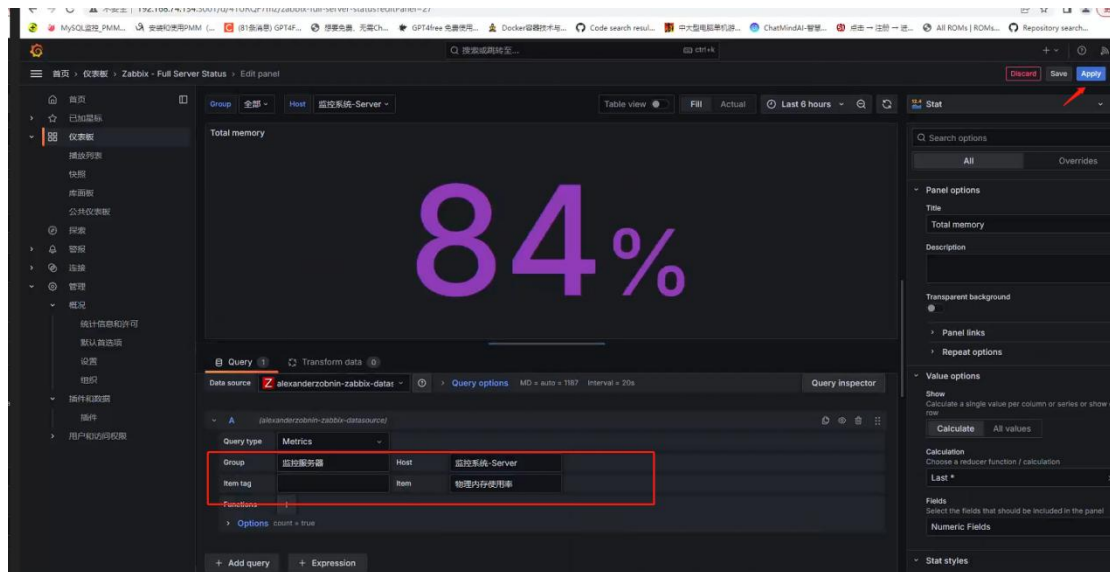


到这一步默认很多数据没有，这是因为导入的模版的 item 项和 zabbix 那边的无法对应，

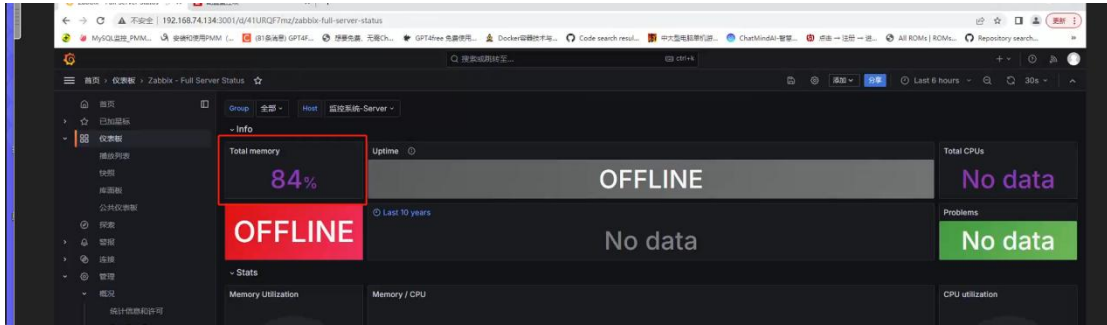
自行修改面板的监控项即可



修改仪表盘的监控项

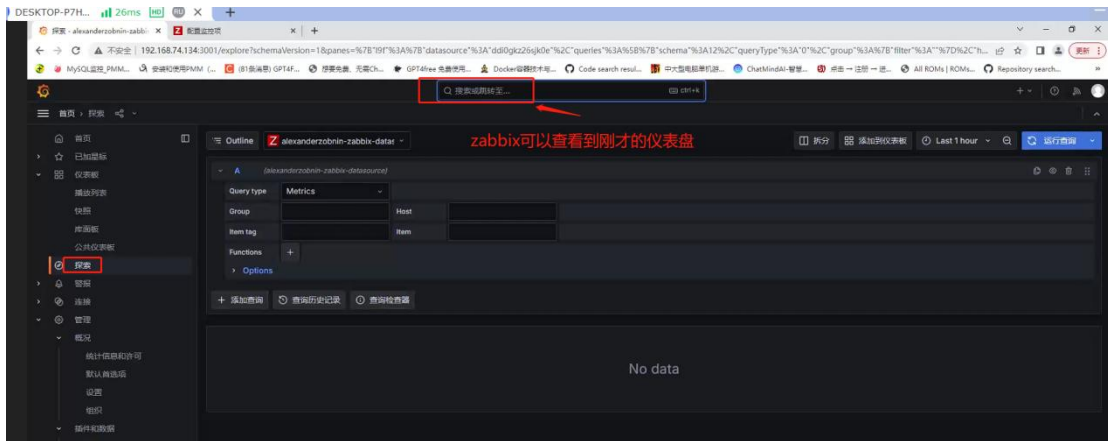


本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

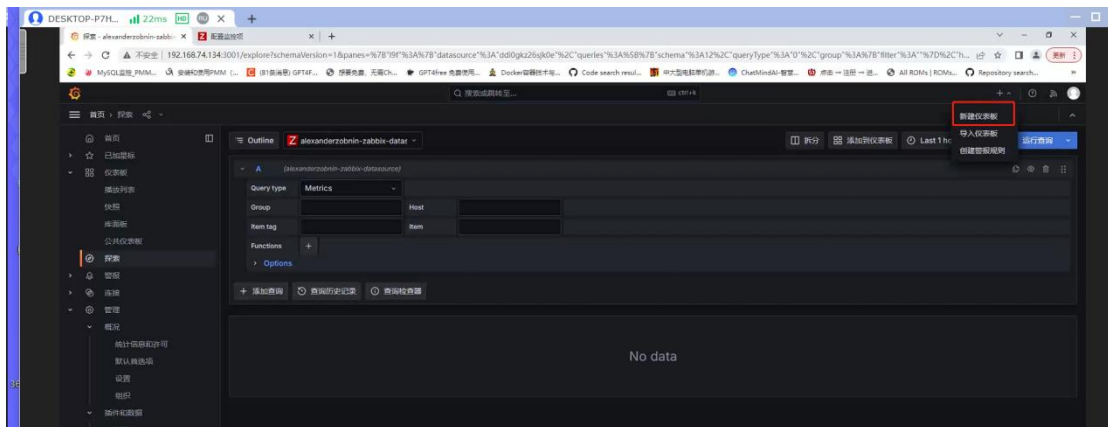


完成对接 zabbix，已经可以正常获取数据。

## 四、grafana 仪表盘配置

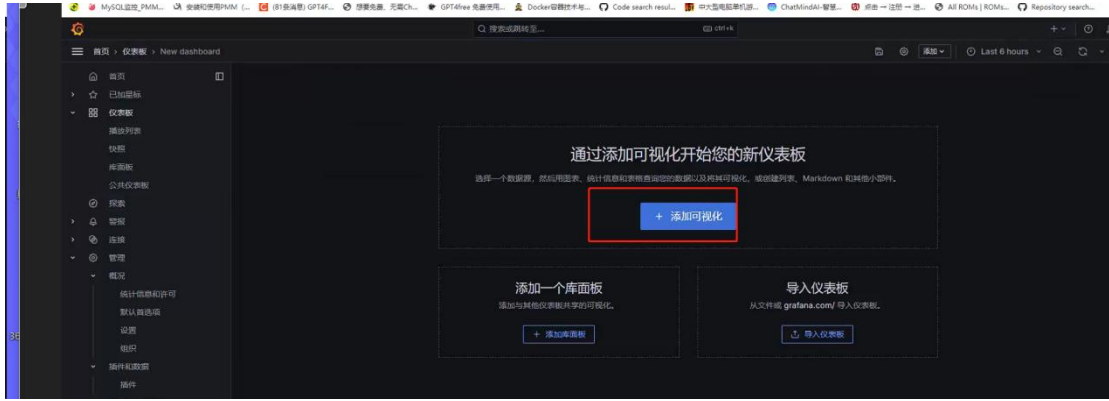


自定义面板

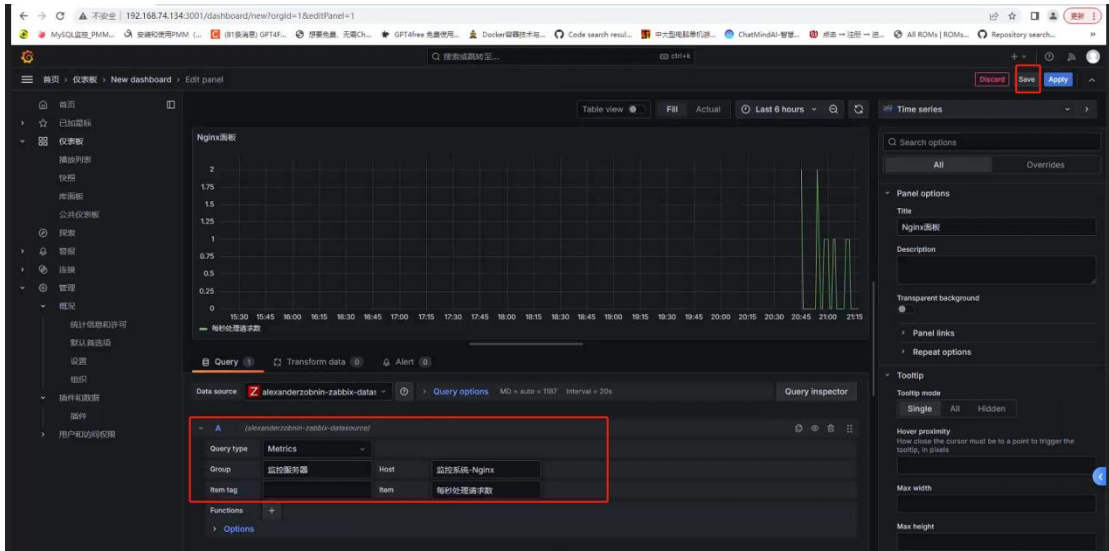


选择添加可视化

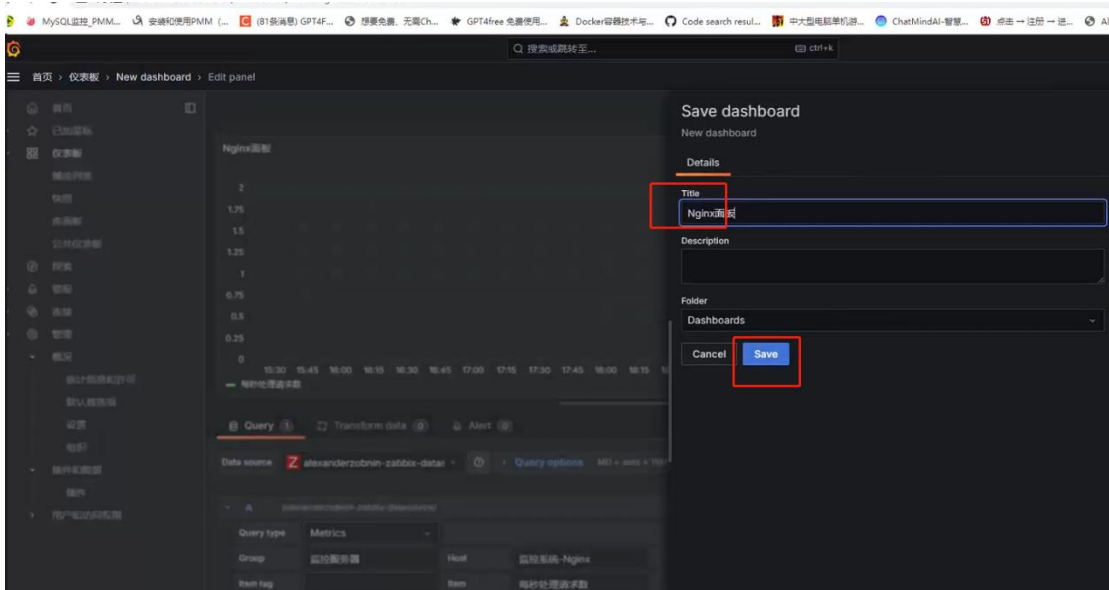
本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



### 选择要展示的指标

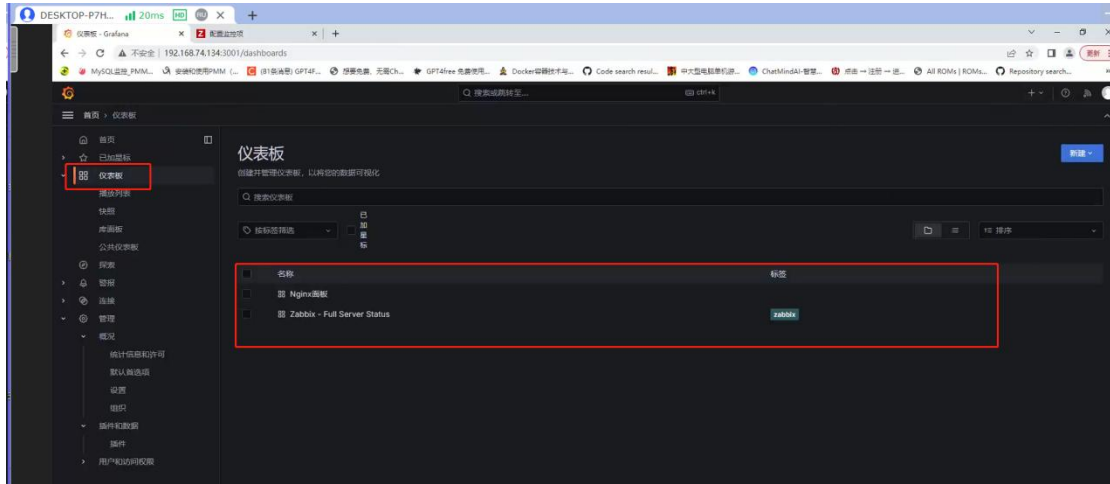


### 修改面板标题

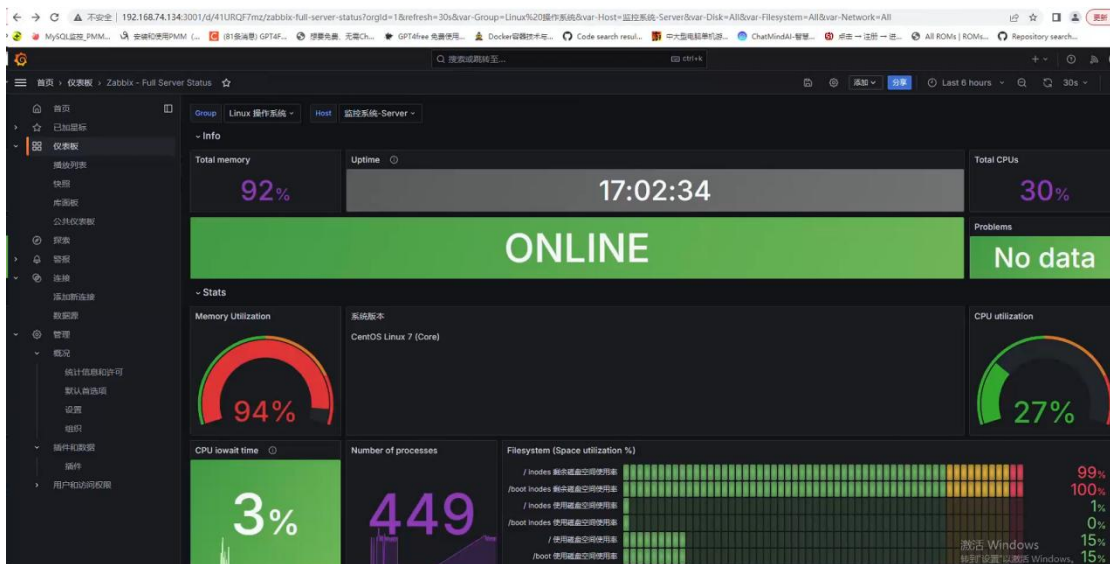


本文档为样章，完整版文档请添加乐乐（lerwee）获取

查看已存在的配置面板



修改后的效果图如下



至此，zabbix 对接 grafana 完成。

更多.....

# 常见问题

## 1、Zabbix 部署过程可能遇到的问题

这几天忙了一个项目，为顾客自定义安装一个他们公司专有的功能和界面，一堆的工具包，php 包，客户没有这么专业弄的。所以我的老同事就让我，弄一个可以为客户一键部署的 zabbix 安装包。

1, zabbix 界面忘记密码的操作，root 用户进入数据库，使用 zabbix 库 update users set passwd='\$2y\$10\$IepFhXA/cXywW4t9MHI9J.2kyI23m7WGceQcy2tN144weF.z/lnie' where alias="Admin"

这样登录的密码就是 Etx@2019 了，当然这个只是参考，原理就是进数据库改密码。

2, mysql\_config not found, configure: error: Please reinstall the mysql distribution 报错，这种的数据库问题，可以执行：

配置文件 --with-mysqli=/usr/local/mysql/bin/mysql\_config

具体对于自己 mysql 安装路径

3, 安装 sqlsrv ,pdo\_sqlsrv 的 php 扩展的时候报错，可下面这样做：

```
PHP日记 from /root/mssql/sqlsrv-4.3.0/shared/xplat_winhls.h:24, 首页 PHP Go Lang 前端 Lin
from /root/mssql/sqlsrv-4.3.0/shared/FormattedPrint.h:24,
from /root/mssql/sqlsrv-4.3.0/shared/core_sqlsrv.h:41,
from /root/mssql/sqlsrv-4.3.0/php_sqlsrv.h:25,
from /root/mssql/sqlsrv-4.3.0/conn.cpp:20:
/root/mssql/sqlsrv-4.3.0/shared/xplat.h:30:17: fatal error: sql.h: No such file or directory
#include <sql.h>
^
compilation terminated.
make: *** [conn.lo] Error 1*/

###重新安装 unixODBC-devel###
#yum -y install unixODBC-devel

# make install
Installing shared extensions: /usr/local/php/lib/php/extensions/no-debug-non-zts-20170718/

#php.ini 增加扩展
echo "extension=salrv.so" >> /usr/local/php/etc/php.ini
```



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

[size=12.0000pt]4, 还有就是安装完了以后, 界面报这种错, 状态码是 500 的, 这个报错我花了很多时间去解决的, 以为是配置文件的问题或者是 php-fpm 的问题, 搜索了很多方法, 最后是老同事说的授权的问题。。。。。

可以去到 html 文件夹下, 对 php 页面相关的所有文件, 执行 `chown -R zabbix:zabbix */nginx/html`, 具体看自己的 nginx 安装位置。



An internal server error occurred.

5, 部署过程出现 mysql 未找到命令, mysqladmin, python3 未找到命令的错误, 这些常见的就是软链接的问题了, 你可以直接再脚本上加上 `ln -s /usr/local/mysql/bin/mysql /usr/bin/mysql` 类似的命令, 具体看自己的软件安装位置。

6, nginx 启动过程中出现:

Nginx 错误: [emerg] getpwnam( "www" ) failed

这种情况是缺少了 www 用户, 属于脚本的用户添加失败, 这时也可以手动添加用户和用户组, 命令是:

```
/usr/sbin/groupadd -f www
```

```
/usr/sbin/useradd -g www www
```

或者, 也可以直接修改 nginx 的 nginx.conf 文件里的用户为目前已创建的用户, 如 zabbix。

本文档为样章, 完整版文档请添加乐乐 (lerwee) 获取

7, 在安装 mysql 还是啥的时候, 出了以下的错误:

```
./boost/python/detail/wrap_python.hpp:50:23: fatal error: pyconfig.h: No such file  
or directory
```

```
compilation terminated.
```

```
"g++" -ftemplate-depth-128 -O3 -finline-functions -Wno-inline -Wall -march=i686
```

```
-pthread -fPIC -m32 -DBOOST_ALL_NO_LIB=1 -DBOOST_PYTHON_SOURCE
```

```
-DNDEBUG -I"." -I"/usr/include/python2.7" -c -o
```

```
"bin.v2/libs/python/build/gcc-5.4.0/release/threading-multi/object/function_doc_s
```

```
ignature.o" "libs/python/src/object/function_doc_signature.cpp"
```

```
...failed gcc.compile.c++
```

```
bin.v2/libs/python/build/gcc-5.4.0/release/threading-multi/object/function_doc_si
```

```
gnature.o...
```

```
..failed updating 58 targets...
```

```
...skipped 12 targets...
```

```
...updated 11810 targets...
```

起初看到这么一大堆东西的时候很烦恼, 不知啥原因, 找了百度很多条, 才找到一条博客是解决问题的

解决方案是: `yum -y install python-dev`

8,



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

```
[root@VM-0-10-centos ~]# cd /itops/nginx/html/lwjk_v3/
[root@VM-0-10-centos lwjk_v3]# php run mm/1 freeway
Exception 'Error' with message 'Class 'app\components\ModuleObject' not found'
in /itops/nginx/html/lwjk_v3/modules/mm/console/Application.php:65

Stack trace:
#0 /itops/nginx/html/lwjk_v3/vendor/yiisoft/yii2/base/Module.php(522): app\modules\mm\console\Application->createController('i')
#1 /itops/nginx/html/lwjk_v3/vendor/yiisoft/yii2/console/Application.php(180): yii\base\Module->runAction('mm/1', Array)
#2 /itops/nginx/html/lwjk_v3/vendor/yiisoft/yii2/console/Application.php(147): yii\console\Application->runAction('mm/1', Array)
#3 /itops/nginx/html/lwjk_v3/vendor/yiisoft/yii2/base/Application.php(386): yii\console\Application->handleRequest(Object(yii\console\Request))
#4 /itops/nginx/html/lwjk_v3/run(21): yii\base\Application->run()
#5 {main}
[root@VM-0-10-centos lwjk_v3]#
```

Stack trace 类的报错

这类问题我也是找开发人员才可以知道怎么解决的。

这种报错可能就 php 某个模块或版本不对应的，这时要更新替换某个 php 文件了。

9, 使用 1G 内存 1 个核的 CPU 的腾讯云主机源码安装 mysql 时，在编译 make 命令时遇到错误。

如下：

```
c++: internal compiler error: Killed (program cc1plus)
```

Please submit a full bug report,

with preprocessed source if appropriate.

See <<http://bugzilla.redhat.com/bugzilla>> for instructions.

```
make[2]: *** [sql/CMakeFiles/sql.dir/item_geofunc.cc.o] Error 4
```

```
make[1]: *** [sql/CMakeFiles/sql.dir/all] Error 2
```

```
make: *** [all] Error 2
```

解决办法：

[size=12.0000pt]1、增加虚拟内存，也就是增加 swap。

[size=12.0000pt]2、或者把跑的线程核数降低，可修改为跟主机一样 cpu 核数

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

[size=12.0000pt]10，还有一些就是 shell 脚本编写的问题了，很多细节需要注意，多练脚本就可以减少出错了。修改一些配置文件的时候，可以使用 sed -i 的命令，来批量修改文件中某部分的内容了，非常省时

如：

sed -i 就是直接对文本文件进行操作的。

```
sed -i 's/原字符串/新字符串/' /home/1.txt
```

```
sed -i 's/原字符串/新字符串/g' /home/1.txt
```

## 2、Zabbix 定制-MIB 库与 MIB Browser 用法

对于一个 Zabbix 监控玩的很溜小伙伴来说，MIB 这个词肯定有所耳闻。本文主要讲解 MIB 的含义、作用以及 MIB 的查看方法。

先看下百度百科的说法：



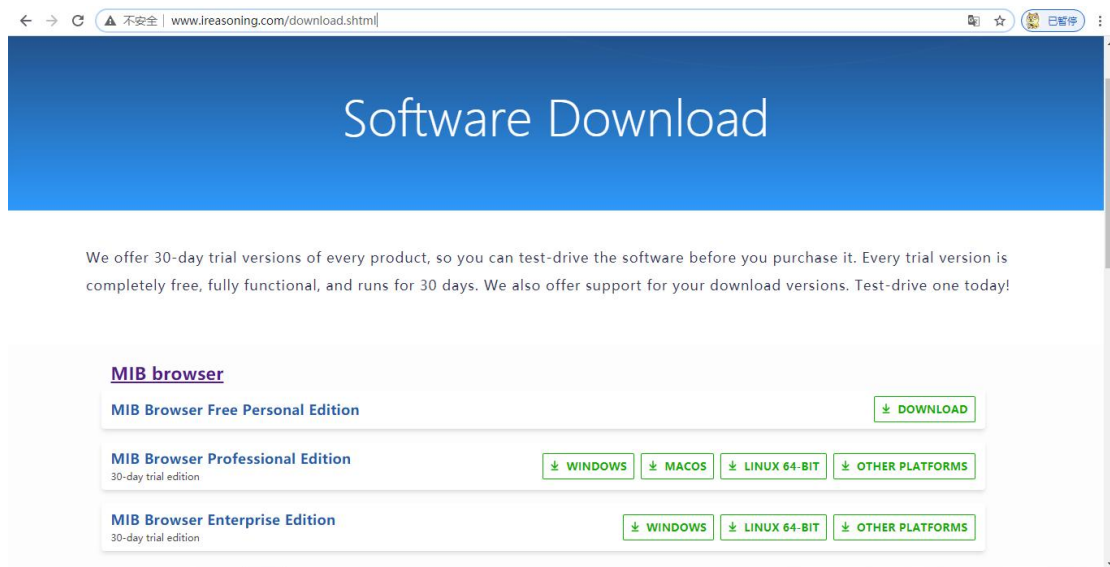
个人理解：含义总的来说 MIB 就是一个有结构的数据库，里面存储了一些 OID 数据记录，OID（对象标识符）。一般各大硬件、服务器厂商都会维护有自己产品专属的 MIB 库，里面存储了对应系列、型号硬件的监控指标，例如风扇、电源等相关部件的信息。MIB 库它实际上就是一个.MIB 或者.txt 结尾的文件，如下图：

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

名称	修改日期	类型	大小
STORAGE-MIB.mib	2020/11/6 16:55	MIB 文件	51 KB

上述图中的文件其实就是一个宏杉 MS3000 G2 型号存储设备 MIB 库文件，里面记录了该型号设备的一些硬件信息，如果我们想通过 Zabbix 或者其他类似工具的 SNMP 协议监控该存储设备，那就要对 MIB 库内容进行一个分析，这个地方我就为大家推荐一个好用的 MIB 库查看工具 MIB Browser，这是一款收费工具，有 30 天的使用期限，不过可以通过重复安装的方式实现永久使用，下载链

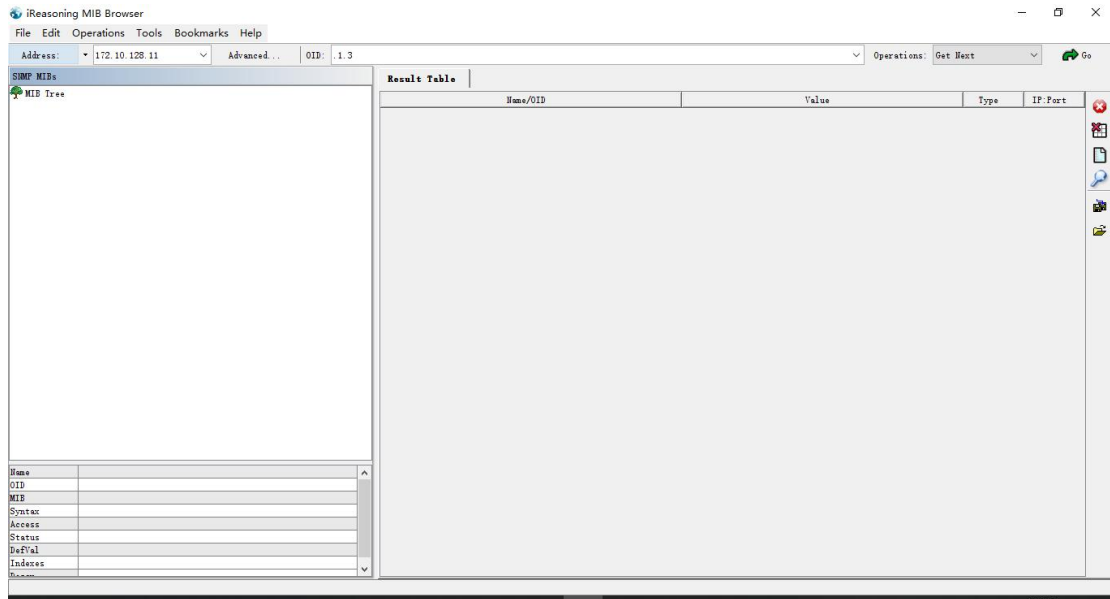
接 <http://www.ireasoning.com/download.shtml> 如下图：



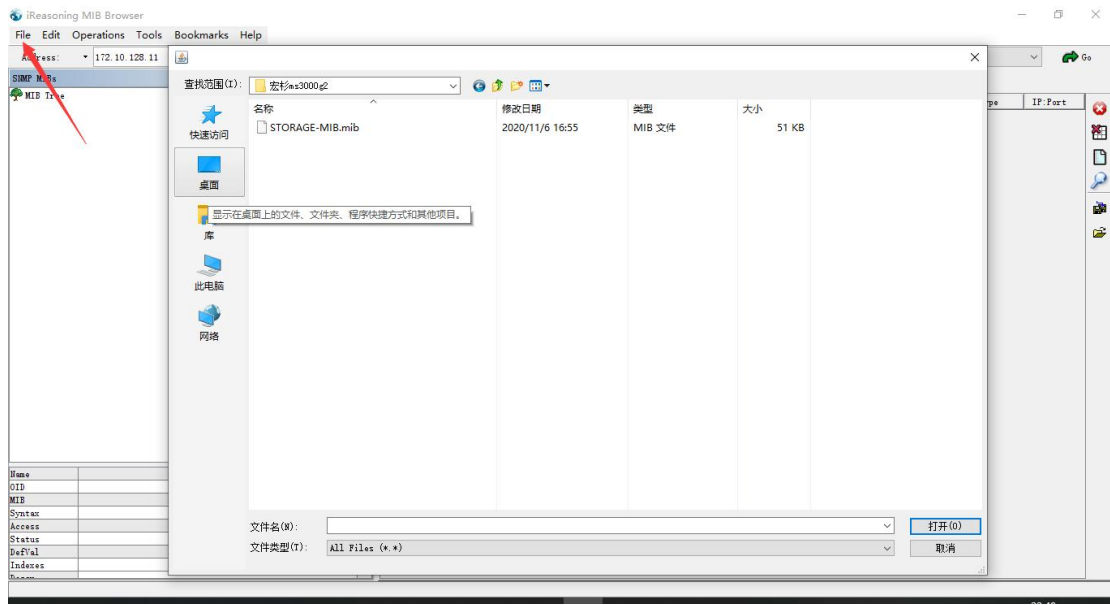
使用教程：

1.安装完成后打开 MIB Browser，如下图：

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

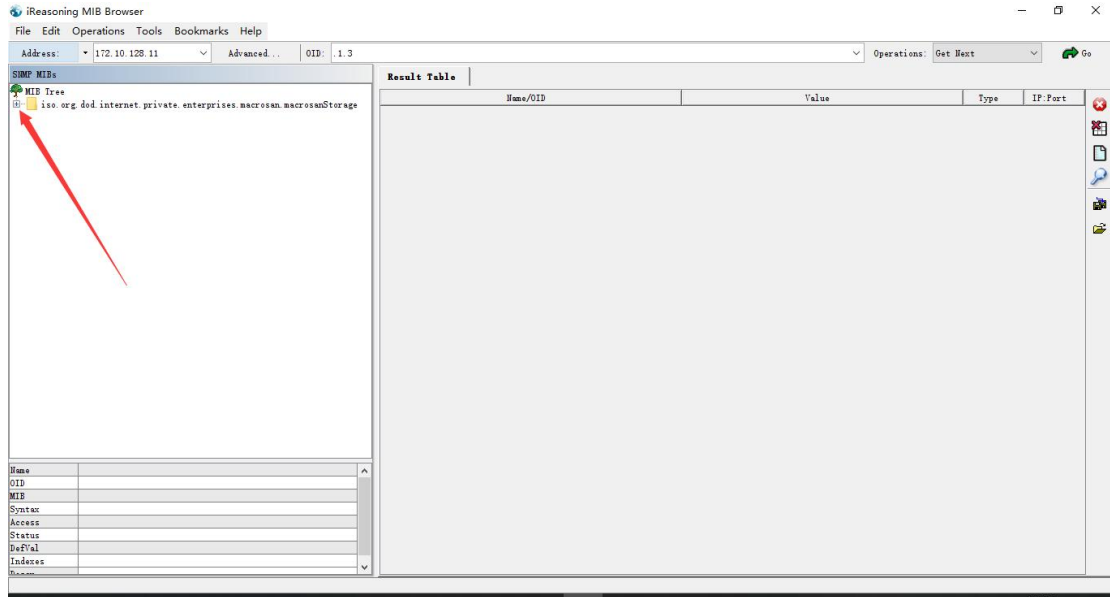


2. 点击左上角 File - Load MIBs 选取刚才的 MIB 库文件，加载 MIB 信息，如下图：



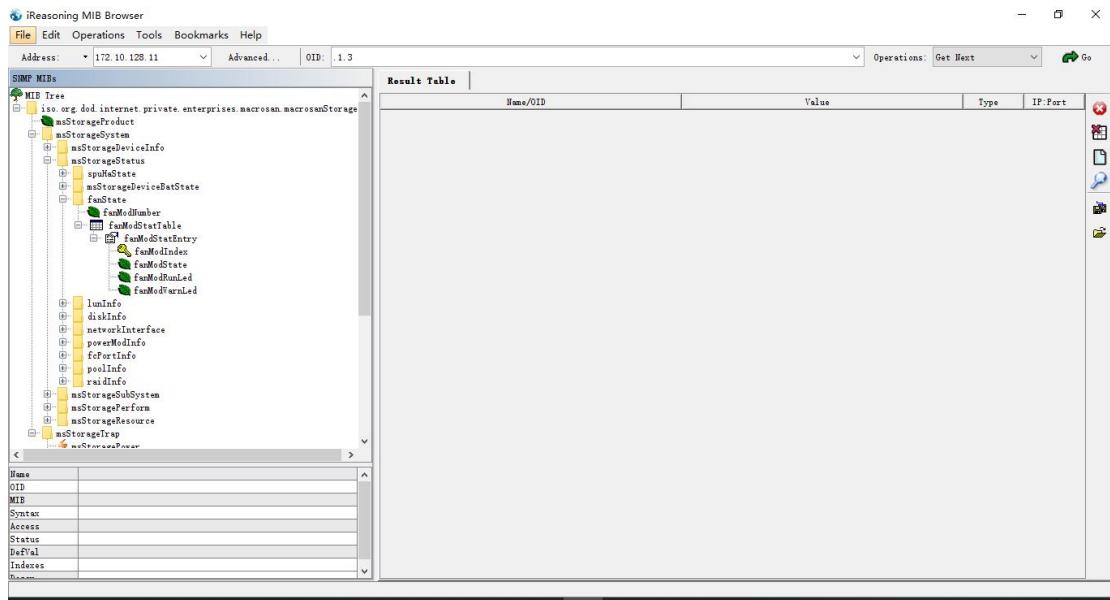
3. MIB 库加载成功后，可以看到 MIB Tree 可以展开了，如下图：

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



4.展开后可以看到宏杉 MS3000 G2 存储的 MIB 库按照 SNMP、SNMP Trap 协议分为两大块。

5.一级一级展开，可以看到里面记录下该型号存储的磁盘、风扇、Lun、FC 等信息，如下图：



6.这个地方我们随便点开一个 OID，例如风扇模块的信息,就可以看到该存储风扇的具体情况，结合监控工具和 SNMP 就可以监控上述设备的一些指标，如下图

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

The screenshot shows the iReasoning MIB Browser interface. The left pane displays a tree view of MIB objects under the 'SIMP MIBs' root. The right pane shows a 'Result Table' with columns for Name/OID, Value, Type, and IP:Port. A red box highlights the details for the 'fanModState' object.

Name/OID	Value	Type	IP:Port
fanModState			

Name	fanModState
OID	.1.3.6.1.4.1.35904.1.2.2.3.2.1.2
MIB	MACROSAN-STORAGE-MIB
Syntax	INTEGER (0..20)
Access	read-only
Status	mandatory
Final	
Indexes	fanModIndex

iso.org|dod|internet-private|enterprises|macroSAN|macroSANStorage.msStorageSystem.msStorageStatus.fanState.fanModStatTable.fanModStatEntry.fanModState

### 3、如何使用 Zabbix 进行 IPMI 监控?

使用 Zabbix 进行 IPMI 监控所需要包含的步骤, 包括安装相关软件包, 配置 Zabbix, 获取传感信息, 定义模板和配置 IPMI 用户密码等

#### 1. 安装 IPMItool 软件包

```
# yum -y install OpenIPMI OpenIPMI-devel ipmitoolfreeipmi
```

#### 2. 配置 Zabbix

服务器端配置 zabbix IPMI pollers

```
# cd /usr/local/zabbix/etc/
```

```
# sed -i '/# StartIPMIPollers=0/aStartIPMIPollers=5'zabbix_server.conf
```

```
# service zabbix-server restart
```

#### 3. 获取传感器信息

登录 Zabbix 服务器, 通过 ipmitool 远程访问服务器传感器信息



```
[root@localhost shell]# ipmitool -I lanplus -H 192.168.1.1 -U admin -P password sensor list
```

UID Light	0x0	discrete	0x0080	na	na	na	na	na	na	na
Int. Health LED	0x0	discrete	0x0080	na	na	na	na	na	na	na
Ext. Health LED	0x0	discrete	0x0080	na	na	na	na	na	na	na
Power Supply 1	35	Watts	ok	na	na	na	na	na	na	na
Power Supply 2	95	Watts	ok	na	na	na	na	na	na	na
Power Supply 3	150	Watts	ok	na	na	na	na	na	na	na
Power Supply 4	110	Watts	ok	na	na	na	na	na	na	na
Power Supplies	0x0	discrete	0x0180	na	na	na	na	na	na	na
Fan 1	50.568	percent	ok	na	na	na	na	na	na	na
Fan 2	50.568	percent	ok	na	na	na	na	na	na	na
Fan 3	50.568	percent	ok	na	na	na	na	na	na	na
Fan 4	50.568	percent	ok	na	na	na	na	na	na	na
Fans	0	percent	ok	na	na	na	na	na	na	na
Temp 1	25.000	degrees C	ok	na	na	na	na	42.000	46.000	
Temp 2	68.000	degrees C	ok	na	na	na	na	126.000	127.000	
Temp 3	40.000	degrees C	ok	na	na	na	na	126.000	127.000	
Temp 4	40.000	degrees C	ok	na	na	na	na	126.000	127.000	
Temp 5	40.000	degrees C	ok	na	na	na	na	126.000	127.000	
Temp 6	34.000	degrees C	ok	na	na	na	na	87.000	92.000	
Temp 7	na	na	na	na	na	na	na	87.000	92.000	
Temp 8	34.000	degrees C	ok	na	na	na	na	87.000	92.000	

```
# ipmitool -I lanplus -H 12.168.1.10 -U admin -P password-L user sensor list
```

```
# ipmitool -I lanplus -H 12.168.1.10 -U admin -P password-L user sensor get "
```

```
Power Supply 1"
```

```
[root@localhost ~]# ipmitool -I lanplus -H 192.168.1.1 -U admin -P password sensor get "Power Supply 1"
```

```
Locating sensor record..
```

```
Sensor ID      : Power Supply 1 (0x4) 填写的key值
```

```
Entity ID     : 10.1
```

```
Sensor Type (Discrete): Power Supply
```

```
Sensor Reading : 35 Watts 获取到的数据
```

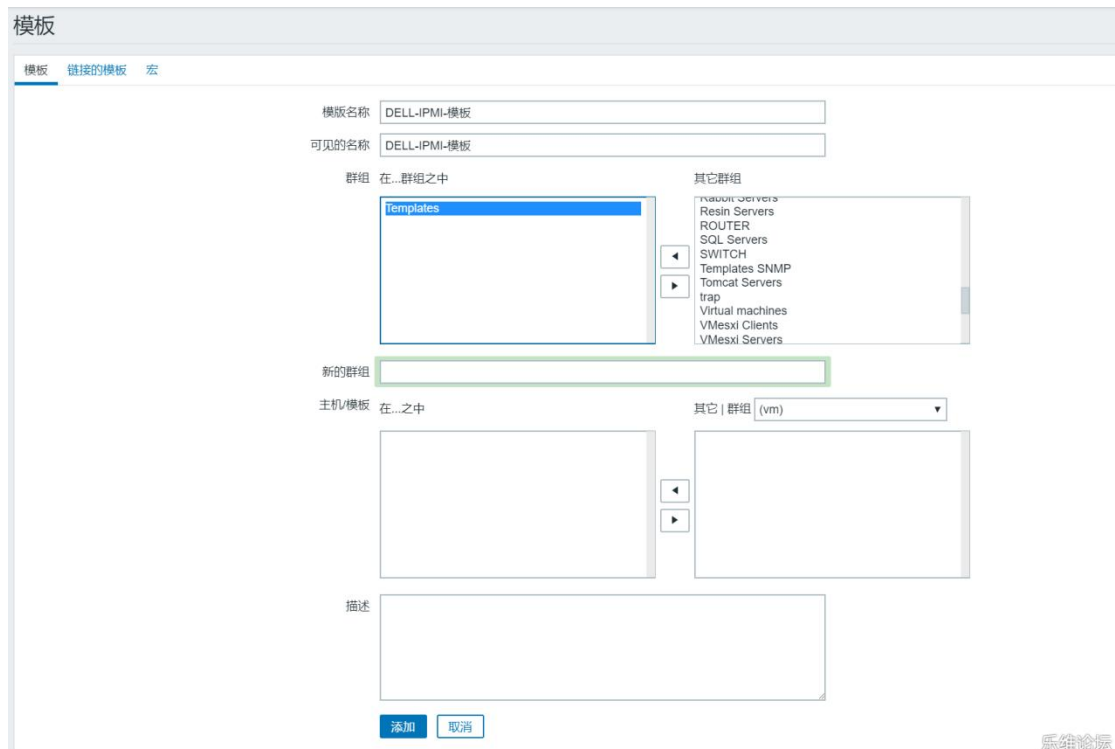
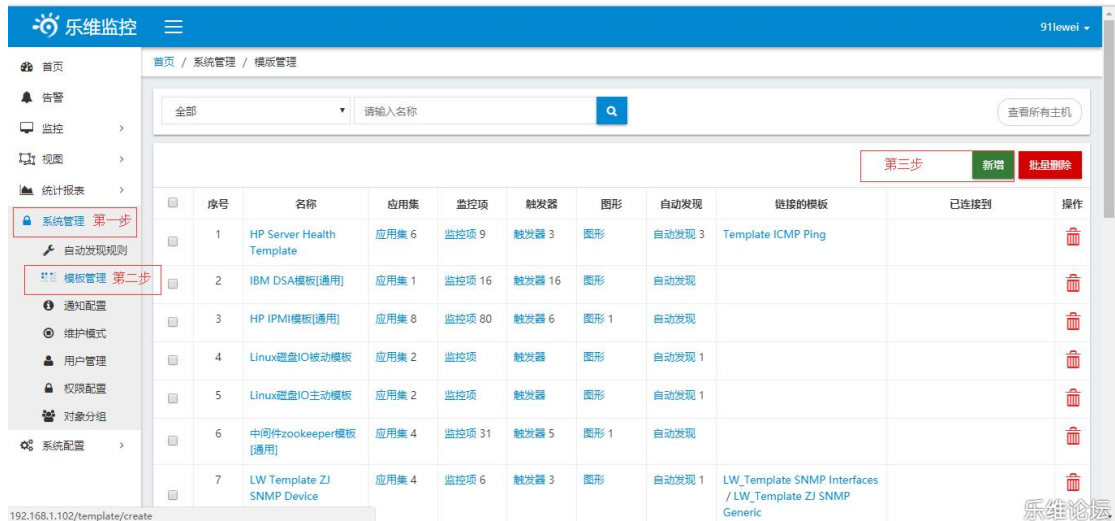
```
States Asserted : Power Supply
```

```
                [Presence detected]
```

#### 4. 定义模板

创建一个空模板

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



模板建好之后，我们来添加一个 key

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



我们先确定 key，我们通过 sensor 的方式进行监控，在命令行执行

获取 Power Supply 1 的具体信息

```
[root@localhost ~]# ipmitool -I lanplus -H 192.168.1.1 -U admin -P 1234567890 sensor get "Power Supply 1"
Locating sensor record...
Sensor ID       : Power Supply 1 (0x4) 填写的key值
Entity ID      : 10.1
Sensor Type (Discrete): Power Supply
Sensor Reading  : 35 Watts 获取到的数据
States Asserted : Power Supply
                 [Presence detected]
```

之后，我们编写 Key，有空格的地方，我们使用\_下划线连接，注意 IPMI sensor (IPMI 传感器) 不要加下划线，其他的就跟配置 agent 的方式一样就可以了。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

监控项

所有模板 / DELL-IPMI-模板 应用集 监控项 触发器 图形 聚合图形 自动发现规则 Web 场景

名称 Power Supply 1

类型 IPMI客户端

键值 ipmi\_power\_supply\_1 key值

IPMI传感器 Power Supply 1 要获取数据的参数

信息类型 数字(无正负)

数据类型 十进制数字

单位 Watts

使用自定义倍数  1

数据更新间隔(秒) 30

自定义时间间隔

类型	间隔	期间	动作
灵活	调度	50	1-7,00:00-24:00

历史数据保留时长(单位天) 90

趋势数据存储周期(单位天) 365

储存值 不变

查看值 不变 展示值映射

新的应用集 电源

应用集 无

填入主机资产纪录栏位 无

描述

已启用

乐维论坛

添加完成。

接着，添加一条乐维监控（用于前端显示）

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

监控项

所有模板 / Dell IPMI模板[通用] 应用集 4 监控项 10 触发器 10 图形 1 聚合图形 自动发现规则 Web 场景

名称 LW\_HARDWARE\_DELL 固定格式为 "LW\_(类型)\_(产品型号)"

类型 外部检查

键值 LW\_HARDWARE\_DELL 同上 选择

信息类型 数字 (无正负)

数据类型 十进制数字

单位

使用自定义倍数  1

数据更新间隔(秒) 30

自定义时间间隔

类型	间隔	期间	动作
灵活	调度	50	1-7,00:00-24:00 移除

添加

历史数据保留时长(单位天) 90

趋势数据存储周期(单位天) 365

储存值 不变

查看值 不变 展示值映射

新的应用集

应用集

- 无-
- Electric current / Voltage
- Fans Sensors
- Power Supply
- Temperature Sensors

填入主机资产纪录栏位 -无-

描述 {"agent":1}

已启用

更新 克隆 删除 取消

乐维论坛

模板制作完成。

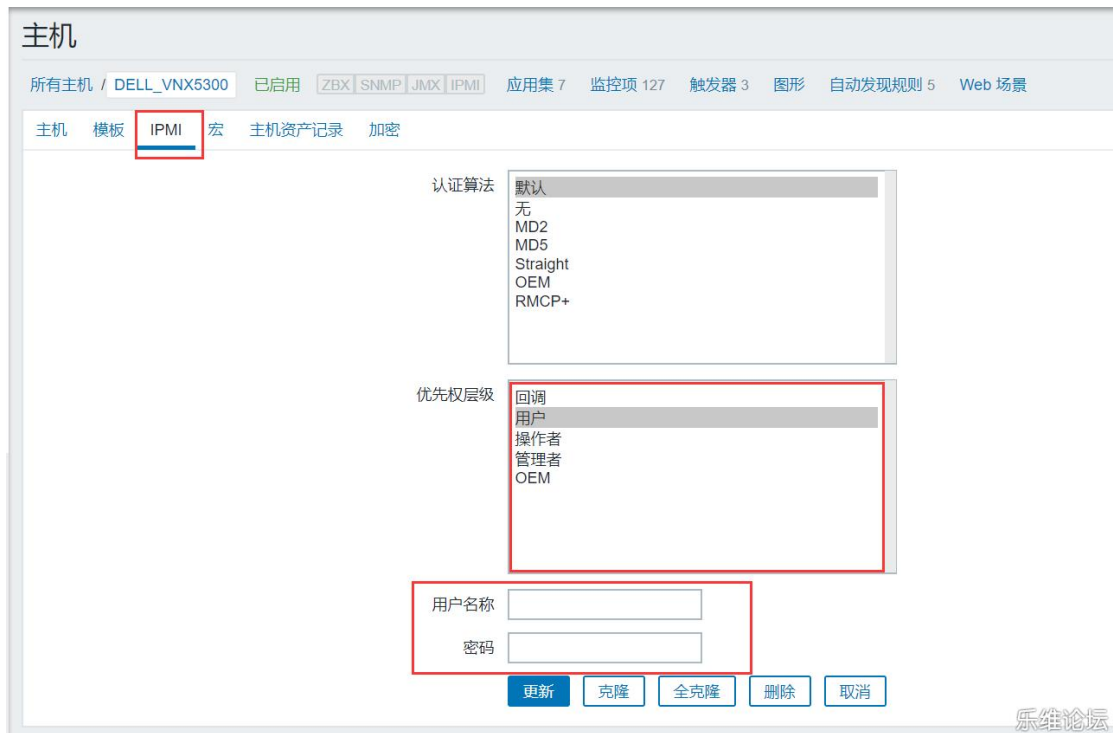
新增监控，过程略。

## 5. 配置 IPMI 用户密码

系统管理—模板管理

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

找到新增模板 “DELL-IPMI-模板” ——点击已连接到的主机



点击 IPMI，对优先权层级进行选择，填写用户名称和密码，点击更新。

主机配置完成。

稍等片刻，监控就会出现数据了。

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取



## 4、Zabbix alerter processes more than 75% busy 告警处理

### 一、告警现象

```
告警主机: 172.18.20.15
告警时间: 2021.03.08 02:23:43
告警等级: Average
告警信息: Zabbix alerter processes more than 75% busy
告警项目: zabbix[process,alerter,avg,busy]
问题详情: Zabbix busy alerter processes, in %:100 %
当前状态: PROBLEM:100 %
事件 ID: 408627698
```

### 二、排查思路

#### 1. 告警分析

告警显示的 zabbix 内部 alerter process 超过 75%，已经达到 100%。alerter process 是用于在触发告警时发送通知的进程，当产生大量的告警时，该进程繁忙就会告警。但结合我们自己的情况，每次出现告警时，伴随的其他告警颇多。所以先是查询了一番，有如下的回复，另外根据以下的回复，并没有看出异常。

(1) Zabbix 的后端数据库卡了

(2) Zabbix 服务器的 IO 卡顿

(3) Zabbix 进程分配到内存不足

2. 进一步在查询，看到可以通过修改 StartAlerters 的数量来调整，紧接在 zabbix\_server.conf 中修改。

```
### Option: StartAlerters

#   Number of pre-forked instances of alerters.

#   Alerters send the notifications created by action operations.

#

# Mandatory: no

# Range: 0-100

# Default:

StartAlerters=10
```

3. 然而在 zabbix\_server.conf 的配置文件中，并没有看到 StartAlerters 这个参数，于是手动加了以下配置，并重启 zabbix 服务，重启之后发现 zabbix 服务无法启动。

```
StartAlerters=10
```

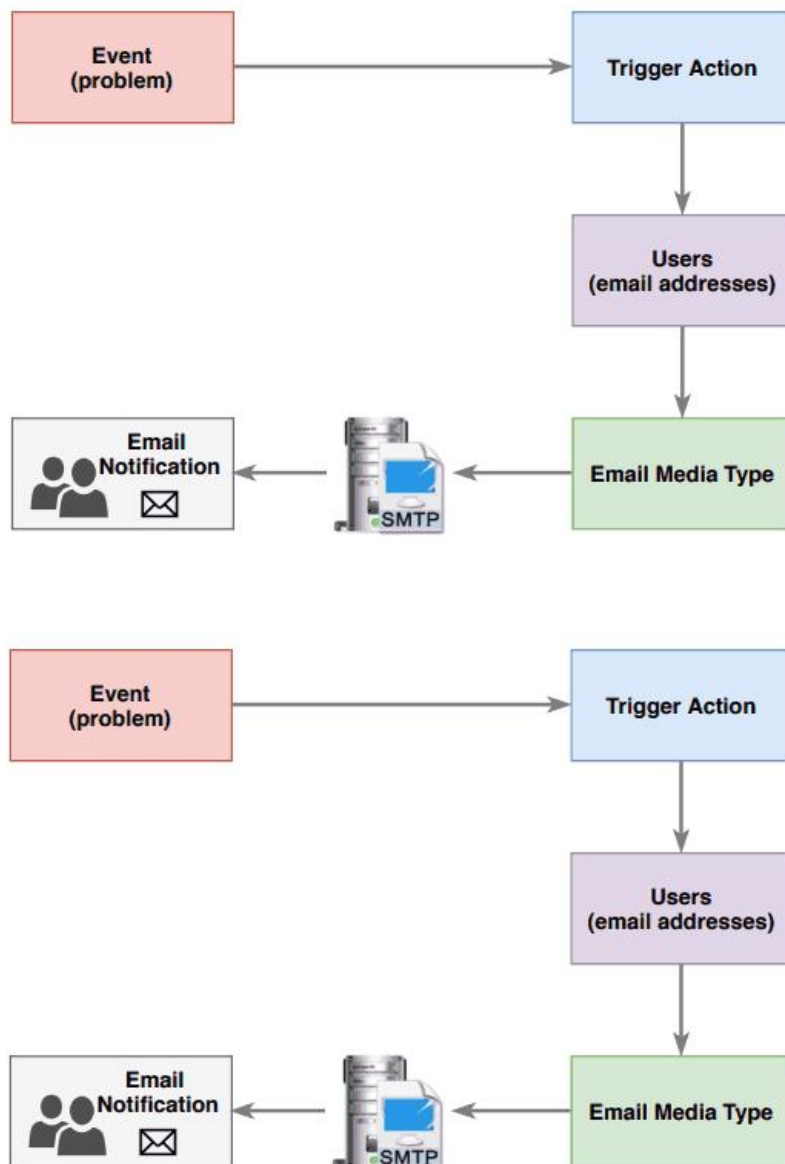


本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

4. 在 zabbix 上的官网上看到，只有 zabbix 4.0 以上的版本才支持修改 StartAlerters 的数量，当前使用的是 3.0 的版本

5. 到这一步，已经没有招了，难道要准备升级版本？这动作太大了，而且没有找到根本原因，根据以往对网络设备升级经验来看，升级版本很多时候并没有用，只是作为尝试的方法。所以升级版本暂时放弃，那到底是啥原因呢.....

6. 重新整理了一下告警思路，下图可做参考



本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

7. 周末在家继续排查，想起最近没有收到 zabbix 每天早上发送的定时邮件，所以判断是因为在出现告警时，大量的邮件无法发出，导致 alerter process 高。那么接下来就是查看发送邮件的脚本。

8. 通过查看发送邮件的脚本，才知道之前更换了告警邮箱，原邮箱支持免密登录，新的邮箱不支持免密登录，开启使用用户名和密码登录后，可以正常发邮件。

```
smtp = smtplib.SMTP()

smtp.connect(mail_host)

smtp.login(mail_user, mail_pwd)
```

9. 观察两天，未收到告警，判断修复

## 5、更多.....

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

# 监控公开模板 (lwjk\_template) 下载

乐维监控公开模板是乐维制作与推出的、面向市面主流厂商及型号设备的开源的监控模板（含部分脚本），任何个人、团队、或企业，都可以免费使用对应模板实现基础设施监控纳管，异常告警等。



[监控模板下载](#)

本文档为样章，完整版文档请添加乐乐 (lerwee) 获取

