Zabbix监控

1. Zabbix服务的监控，当Zabbix服务安装启动，web管理工具配置完毕，会自动生成Zabbix的主机，模版为Template App Zabbix Server和Template OS Linux，主要监控Zabbix agent、Zabbix server、CPU、文件系统、内存、网卡流量、进程、系统信息等。
2. 对于阿里云的SLB监控，SLB并不存在ECS，无法安装客户端，通过阿里提供的SDK编写脚本进行外部检查实现对SLB的监控，监控主机上需要安装阿里提供的SDK，在/usr/local/share/Zabbix/externalscripts目录增加slbmon.py脚本，脚本内容如下：

#!/usr/bin/env python

#coding=utf-8

import json

import sys

from aliyunsdkcore import client

import time

from aliyunsdkcms.request.v20180308 import QueryMetricListRequest

class SlbMon(object):

def \_\_init\_\_(self,slbid='lb-bp13nnhz6ou18t9qu3me5',slbip='120.55.236.226'):

self.slbid = slbid

self.slbip = slbip

#SLB最大并发数

def MaxConn(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceMaxConnection')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

#SLB当前活跃连接数

def ActiveConn(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceActiveConnection')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'port': '443', 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

#SLB实例每秒新建连接数

def NewConn(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceNewConnection')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

#实例每秒丢失连接数

def DropConn(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceDropConnection')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'port': '443', 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

# 实例每秒入bit数

def TrafficRX(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceTrafficRX')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'port': '443', 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

# 实例每秒出bit数

def TrafficTX(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceTrafficTX')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'port': '443', 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

#实例每秒非活跃连接数

def InactiveConn(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceInactiveConnection')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'port': '443', 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

#实例每秒丢失入bit数

def DropTrafficRX(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceDropTrafficRX')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'port': '443', 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

#实例每秒丢失出bit数

def DropTrafficTX(self):

clt = client.AcsClient('LTAIrt7Yy3uzbTQI', '5152BKyAB2lLaBd3hE7NRF680IUCxp', 'cn-hangzhou')

request = QueryMetricListRequest.QueryMetricListRequest()

request.set\_accept\_format('json')

# 两个参数按照预设值设定，参见参考资料描述

request.set\_Project('acs\_slb\_dashboard')

request.set\_Metric('InstanceDropTrafficTX')

# 只获取1小时内的数据

start\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time() - 3600))

timestamp\_start = int(time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M:%S"))) \* 1000

request.set\_StartTime(timestamp\_start)

# 从系统输入参数获取IP地址和SLB-ID

request.set\_Dimensions({'instanceId': self.slbid, 'port': '443', 'vip': self.slbip})

request.set\_Period('60')

# result = clt.do\_action(request)

result = clt.do\_action\_with\_exception(request)

# print result

# json结果处理

datadict = json.loads(result)

# 取倒数第二条数据，因为倒数第一条不太准

res = eval(datadict['Datapoints'])[-2]['Average']

print(res)

if \_\_name\_\_ == '\_\_main\_\_':

slb = SlbMon()

options = sys.argv[1:]

if not options or 'MaxConn' in options:

slb.MaxConn()

if not options or 'ActiveConn' in options:

slb.ActiveConn()

if not options or 'DropConn' in options:

slb.DropConn()

if not options or 'NewConn' in options:

slb.NewConn()

if not options or 'TrafficRX' in options:

slb.TrafficRX()

if not options or 'TrafficTX' in options:

slb.TrafficTX()

if not options or 'InactiveConn' in options:

slb.InactiveConn()

if not options or 'DropTrafficRX' in options:

slb.DropTrafficRX()

if not options or 'DropTrafficTX' in options:

slb.DropTrafficTX()

1. Linux 服务器常规监控，使用Zabbix提供的Template OS Linux模版进行监控，主要监控CPU、文件系统、内存、网卡流量、进程、系统信息等。
2. 对java应用的监控，应用启动命令增加jmx配置，Zabbix通过jmx 接口进行监控，监控项如下：

Java程序启动增加jmx配置：

-Dcom.sun.management.jmxremote

-Dcom.sun.management.jmxremote.authenticate=false

-Dcom.sun.management.jmxremote.ssl=false

-Dcom.sun.management.jmxremote.port=12346

-Djava.rmi.server.hostname=192.168.0.225

创建JAVA\_JAR模版

创建应用集class和监控项（JMX agent代理程序）：

[类-已加载](http://192.168.0.227:9888/zabbix/items.php?form=update&hostid=10255&itemid=28316)：jmx["java.lang:type=ClassLoading","LoadedClassCount"]

类-已卸载：jmx["java.lang:type=ClassLoading","UnloadedClassCount"]

类-总计：jmx["java.lang:type=ClassLoading","TotalLoadedClassCount"]

创建应用集HeapStatistics和监控项（JMX agent代理程序）：

堆内存已使用：jmx["java.lang:type=Memory","HeapMemoryUsage.used"]

堆内存已提交：jmx["java.lang:type=Memory","HeapMemoryUsage.committed"]

堆内存最大：jmx["java.lang:type=Memory","HeapMemoryUsage.max"]

创建应用集Sessions和监控项（JMX agent代理程序）：

Sessions-Number of sessions we rejected due to maxActive beeing reached：

jmx["Catalina:type=Manager,context=/,host=localhost",rejectedSessions]

Sessions-The maximum number of active Sessions allowed, or -1 for no limit：

jmx["Catalina:type=Manager,context=/,host=localhost",maxActiveSessions]

Sessions-会话数：

jmx["Catalina:type=Manager,context=/,host=localhost",sessionCounter]

Sessions-当前活动会话数：

jmx["Catalina:type=Manager,context=/,host=localhost",activeSessions]

Sessions-最大活动会话数：

jmx["Catalina:type=Manager,context=/,host=localhost",maxActive]

创建应用集ThreadStatistics和监控项（JMX agent代理程序）：

活动线程：jmx["java.lang:type=Threading","ThreadCount"]

线程峰值：jmx["java.lang:type=Threading","PeakThreadCount"]

线程总计：jmx["java.lang:type=Threading","TotalStartedThreadCount"]

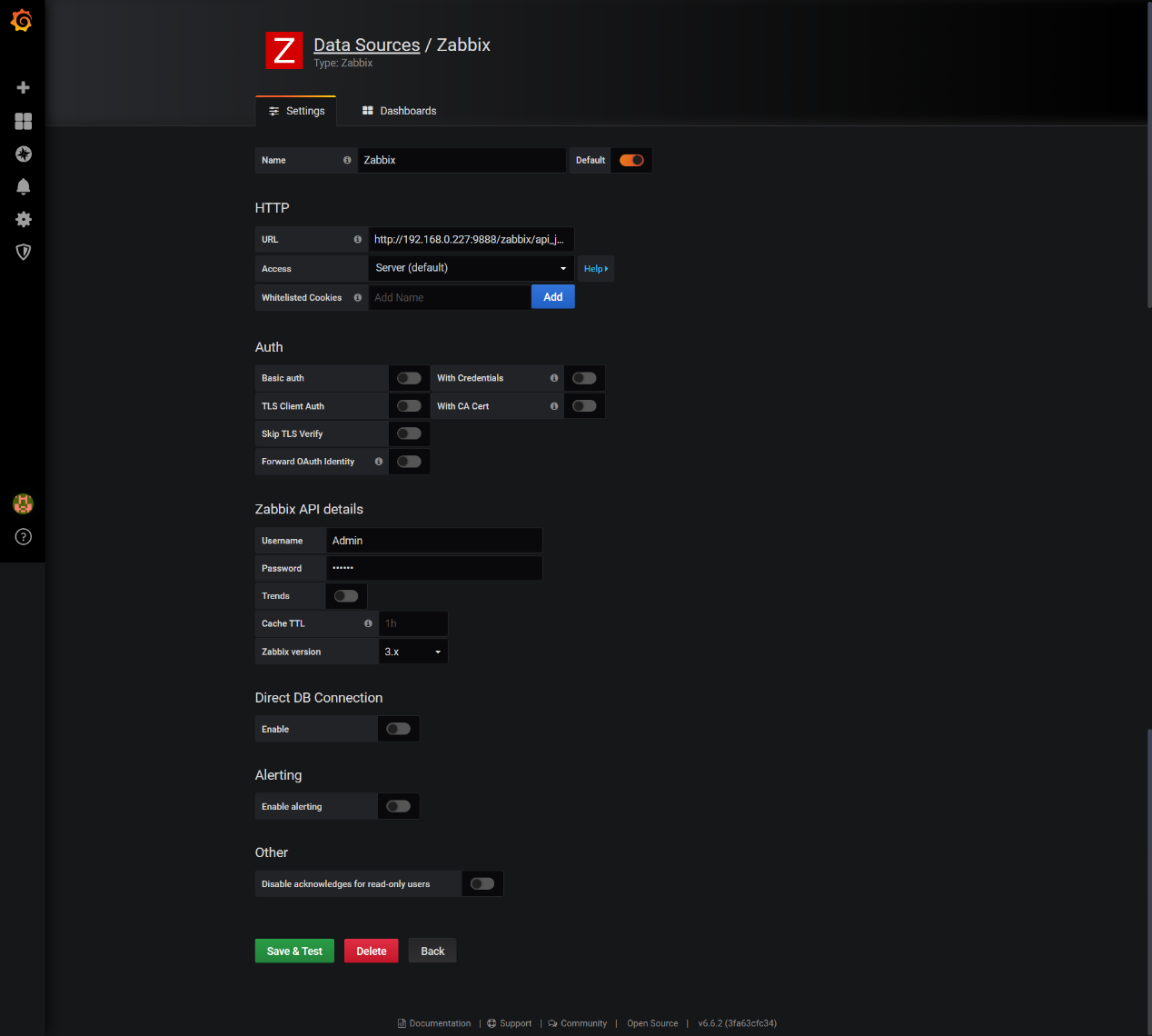
创建触发器：

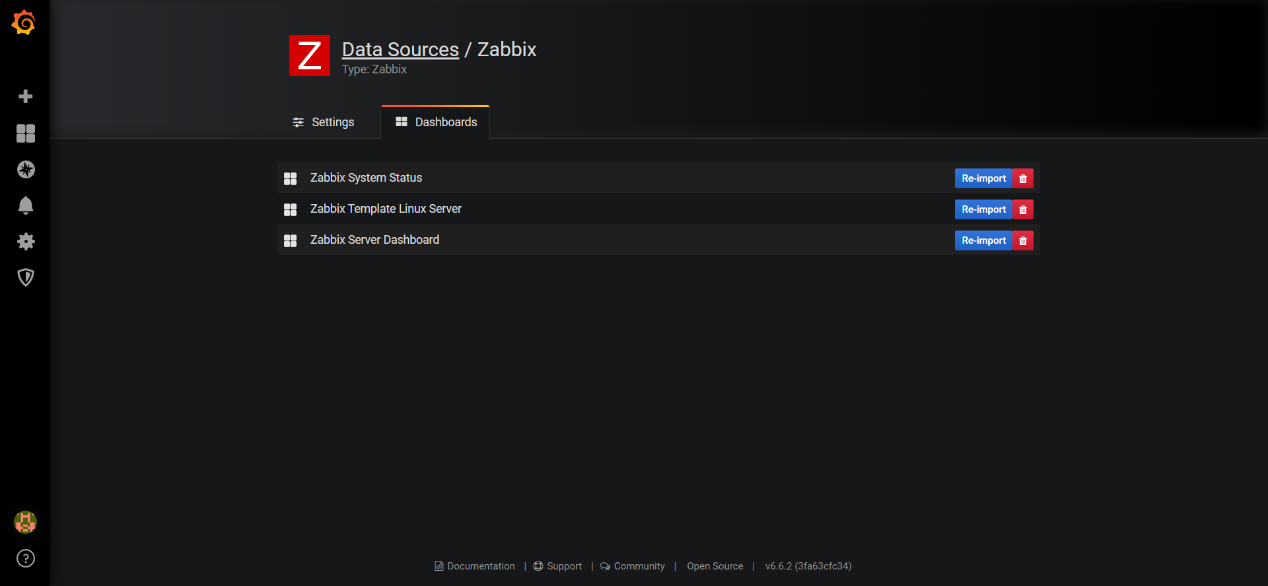
堆内存已使用（对应监控“堆内存已使用”）：

{JAVA\_JAR:jmx["java.lang:type=Memory","HeapMemoryUsage.used"].last(1m)}>1200M

Grafana集成Zabbix

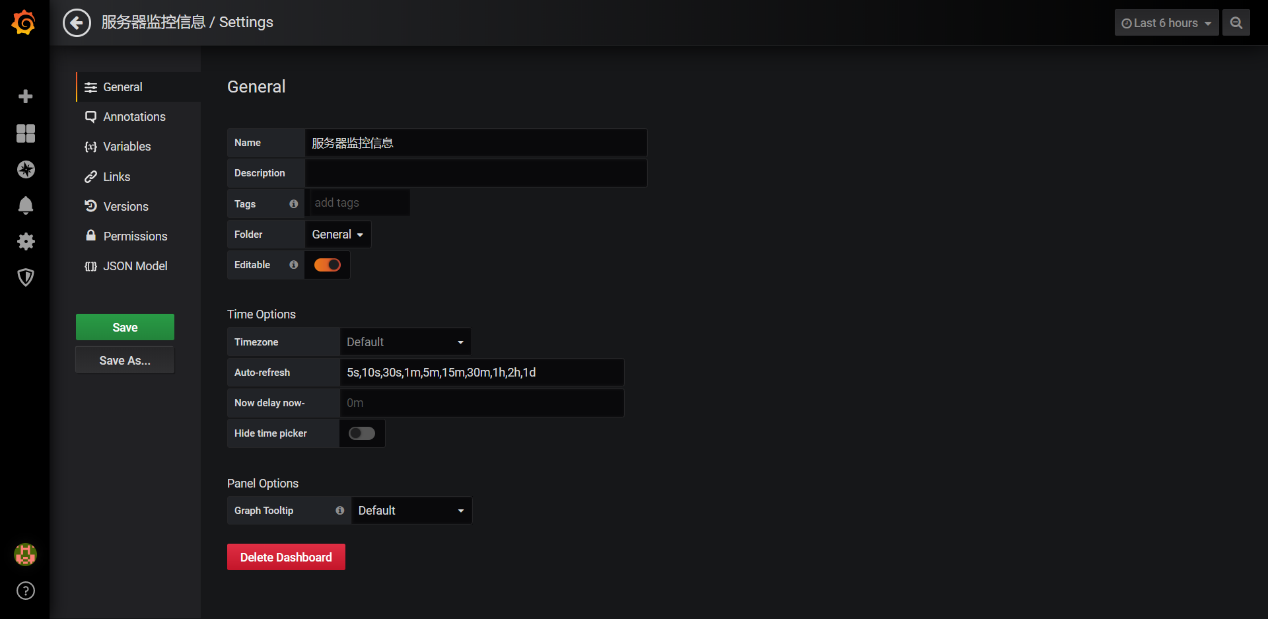
1. 添加数据源（zabbix）

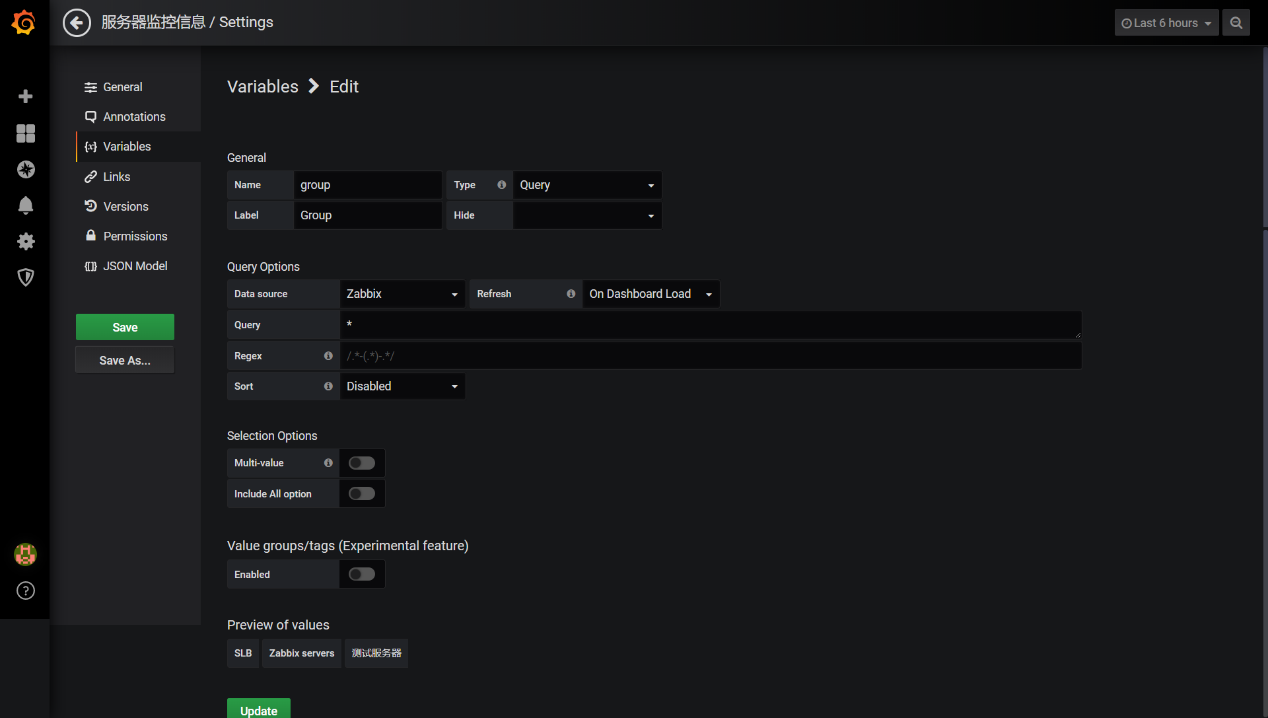
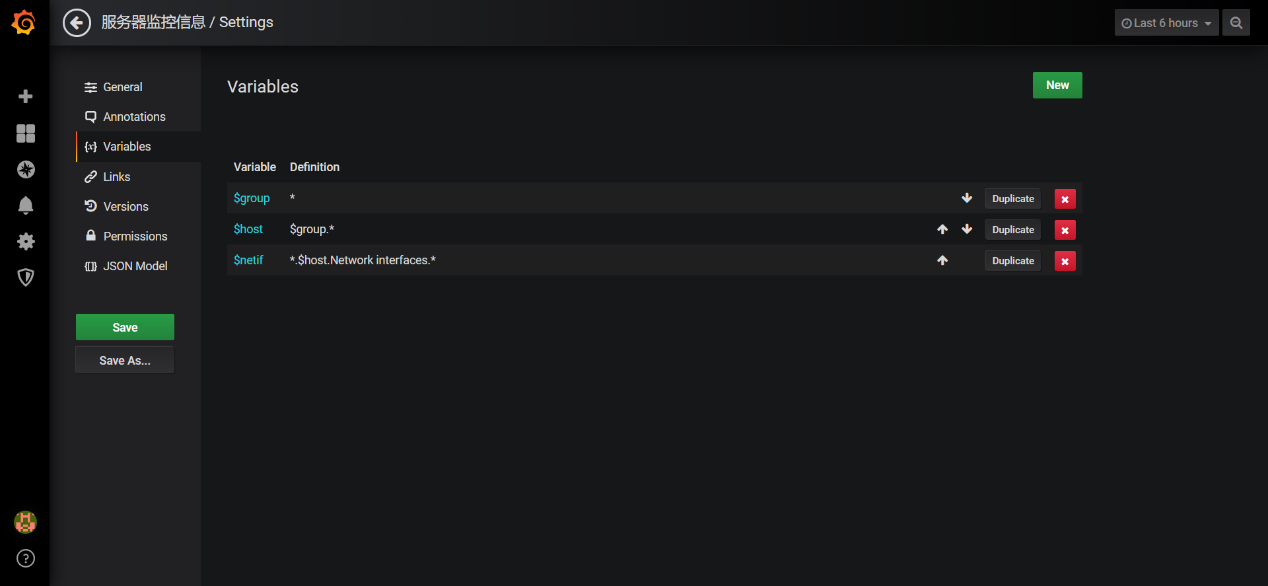


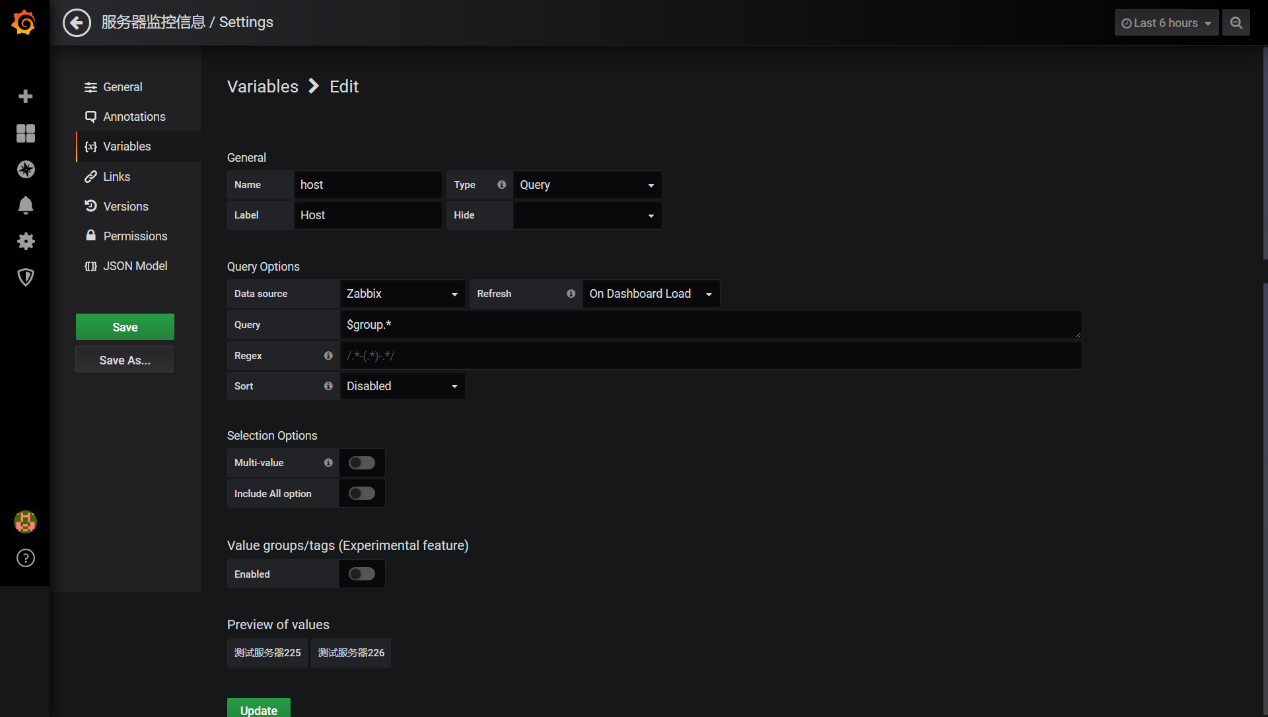


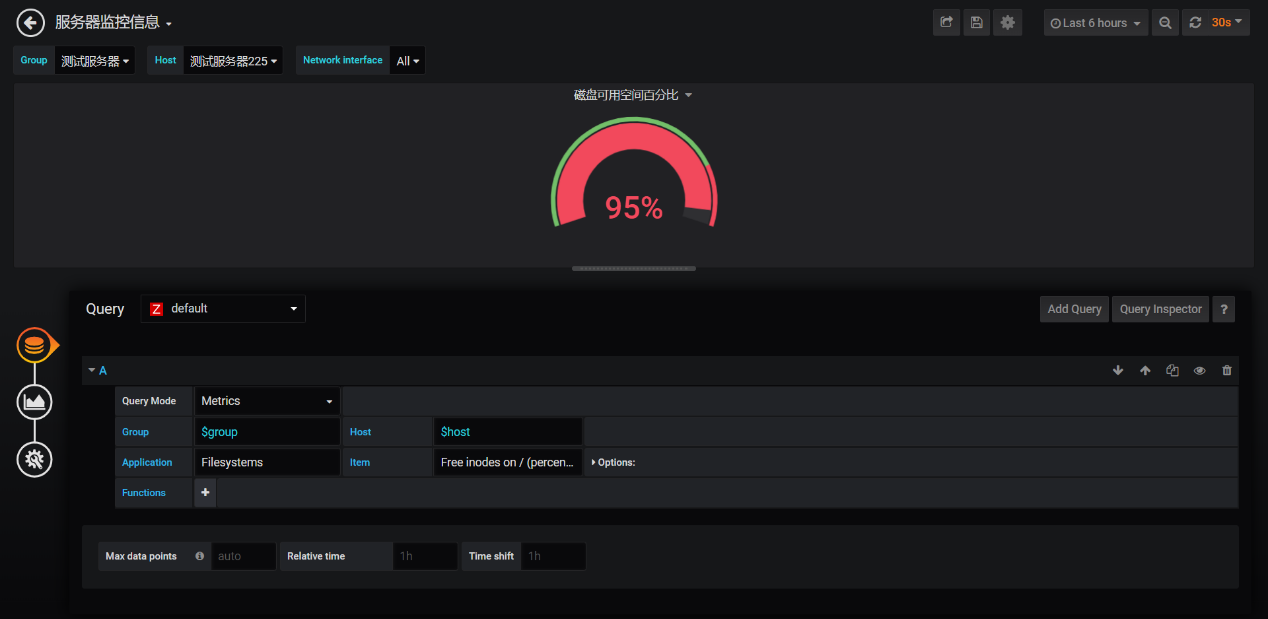
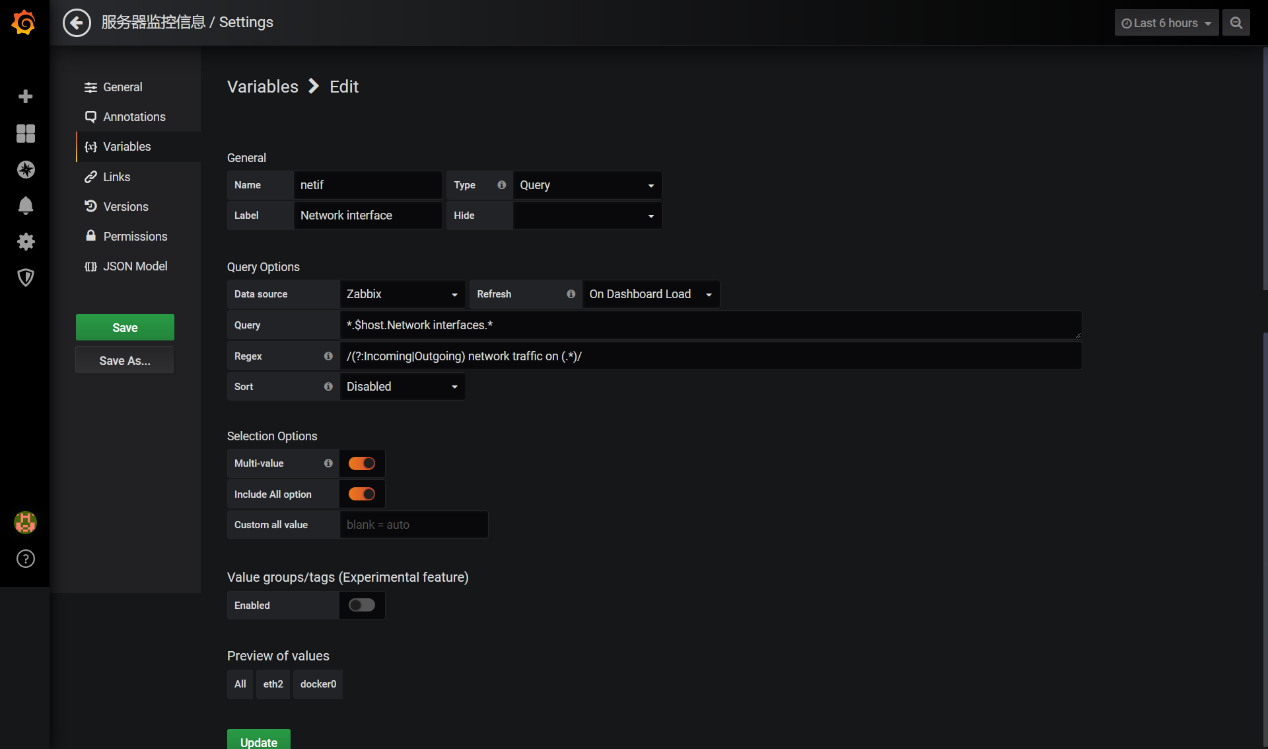
1. 添加仪表盘

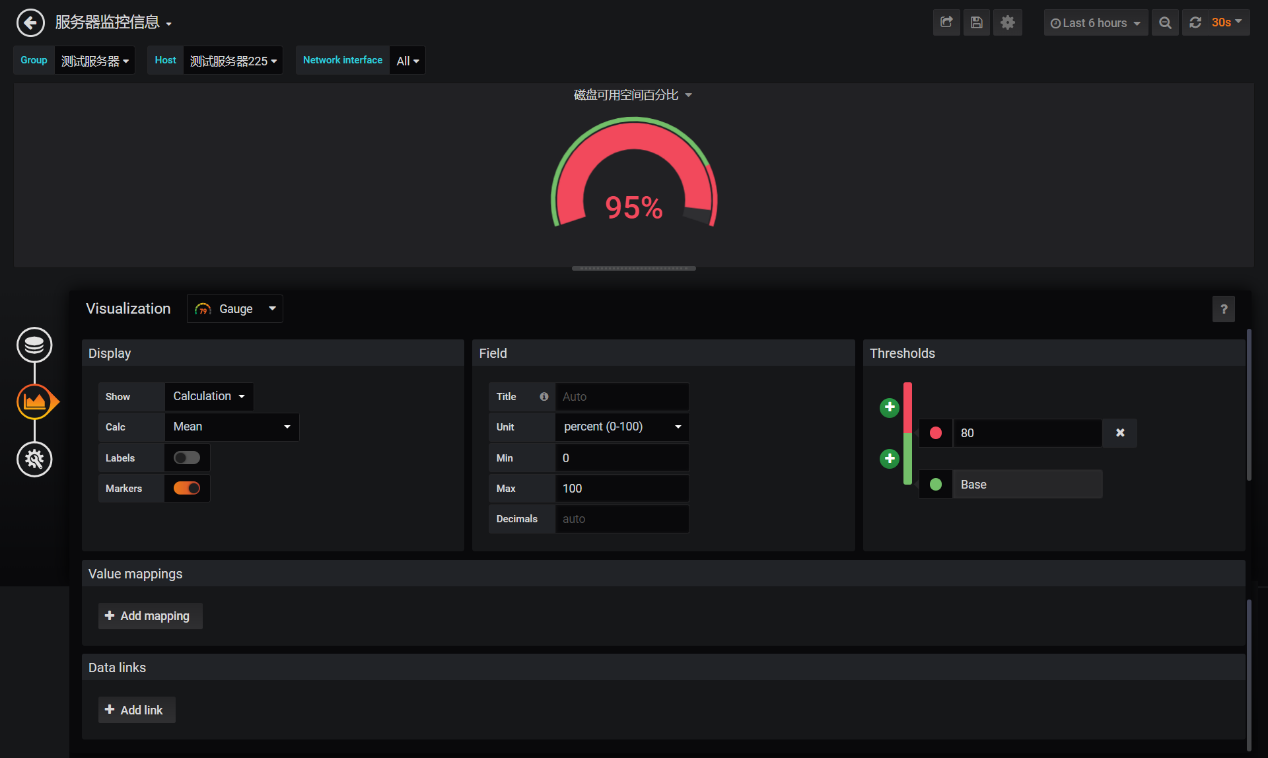
服务器监控信息

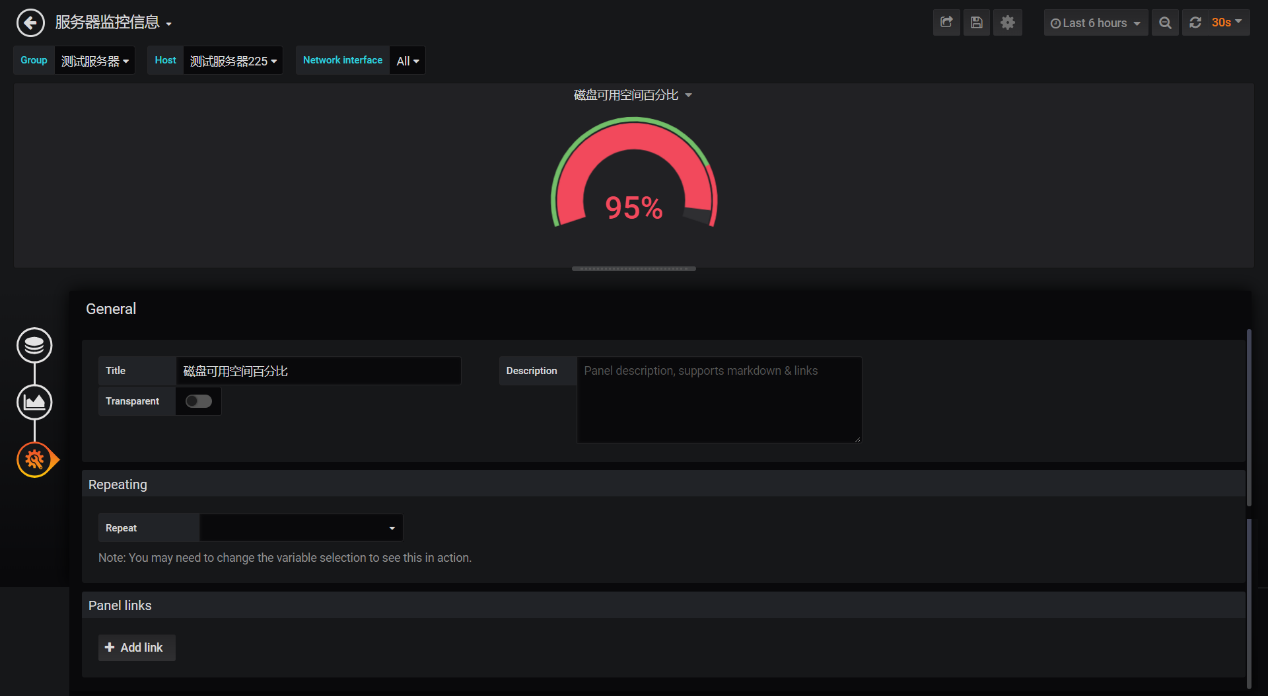


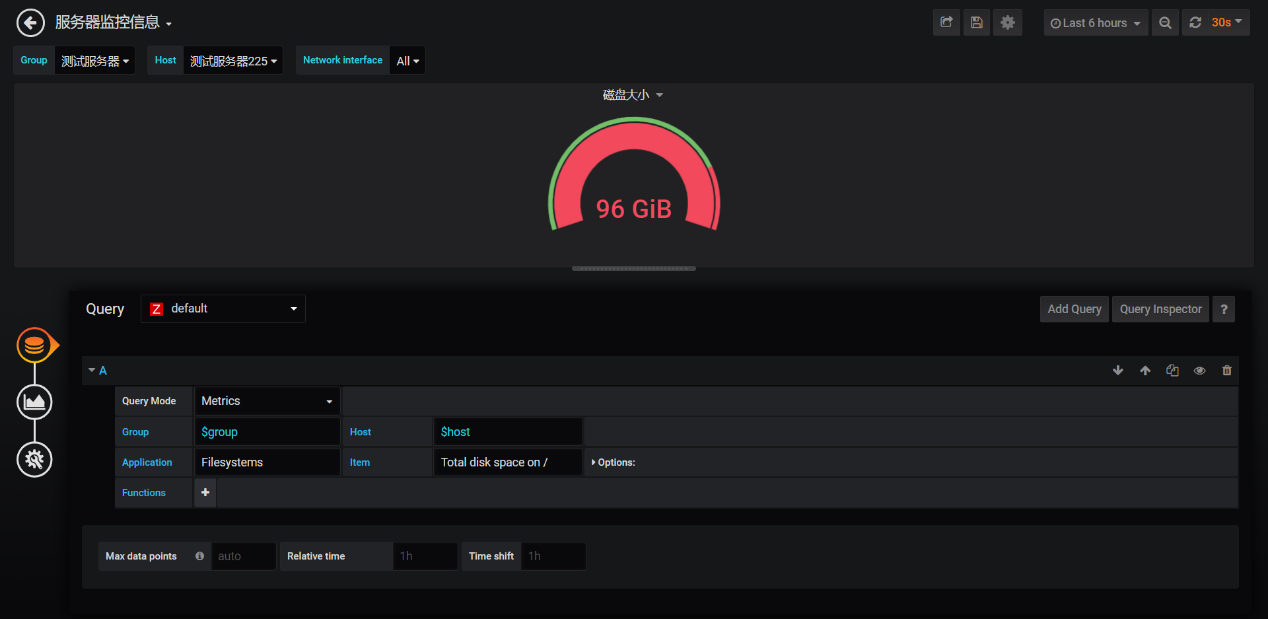


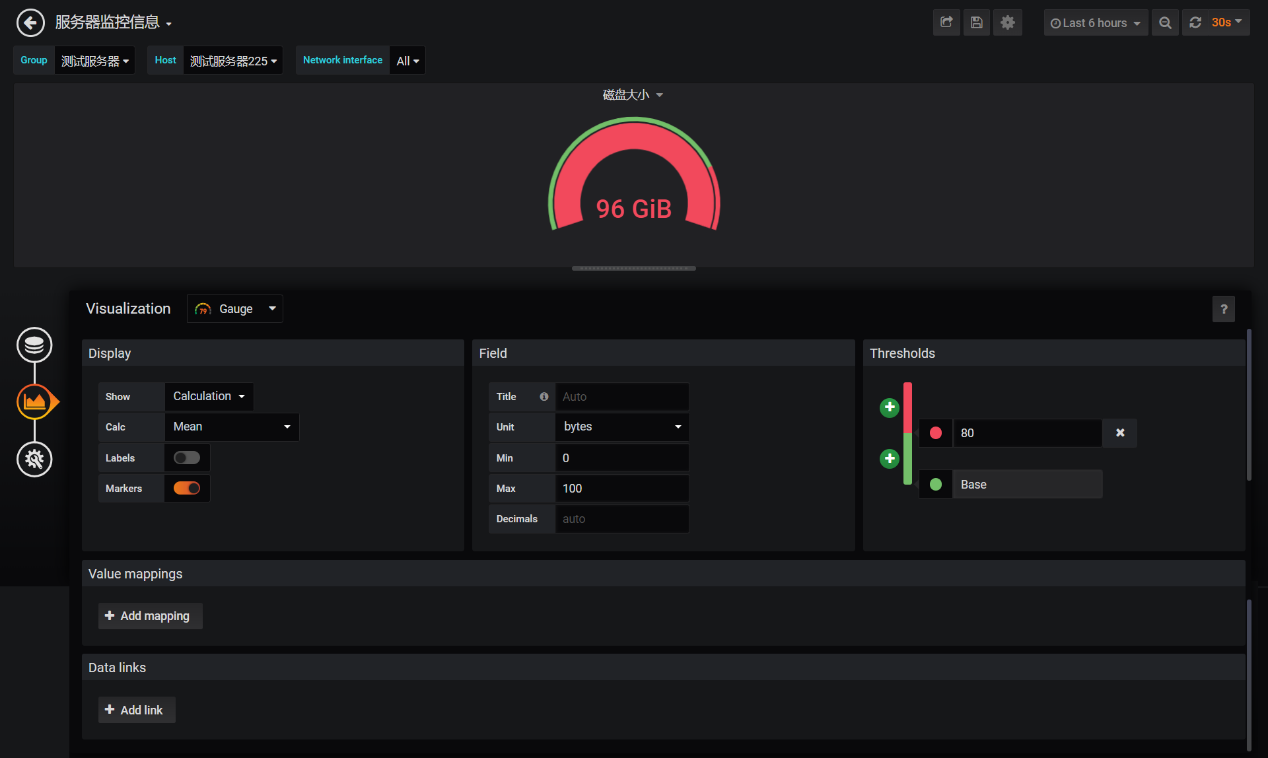


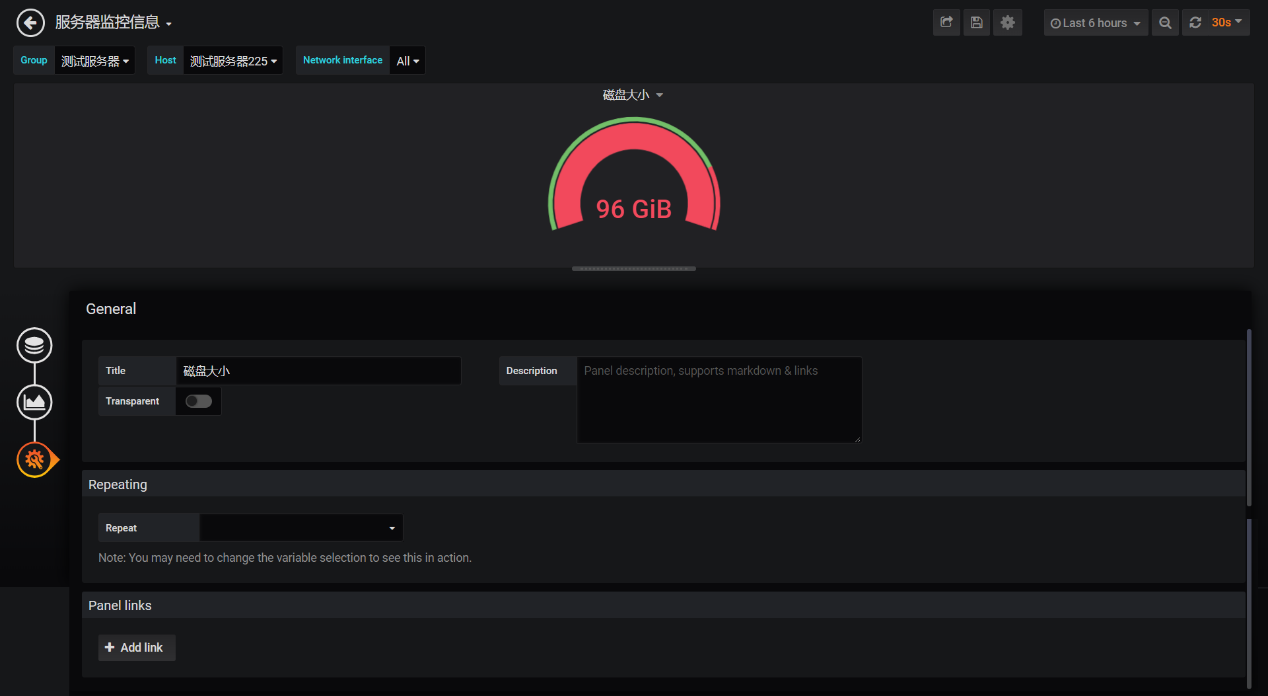


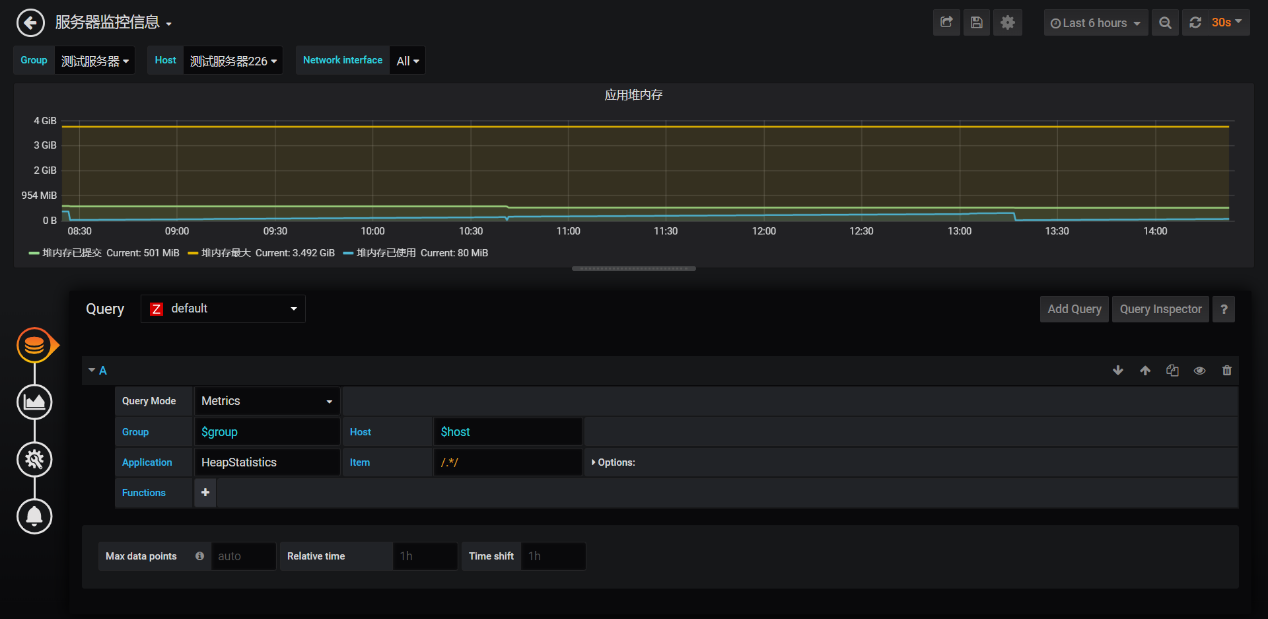


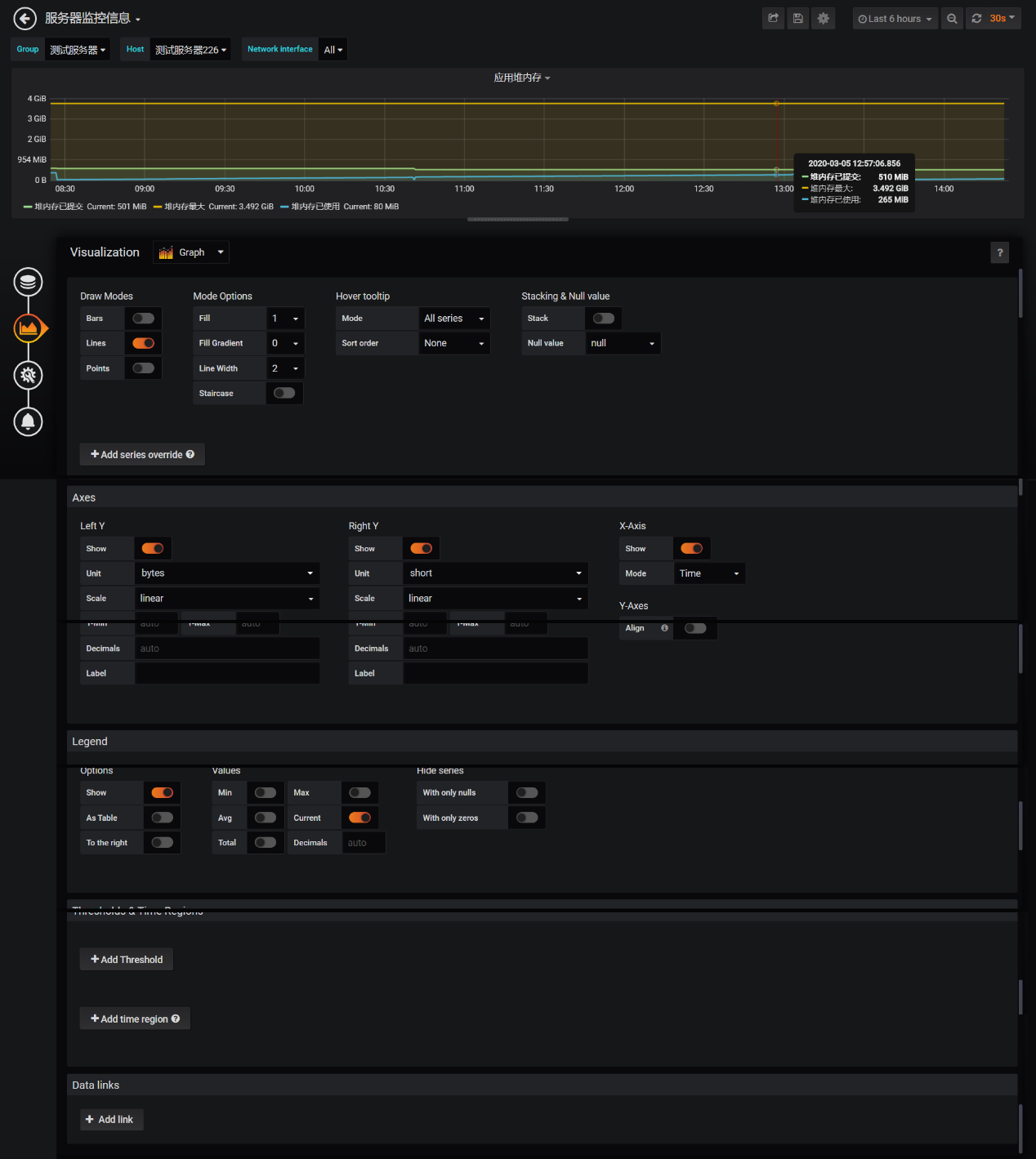


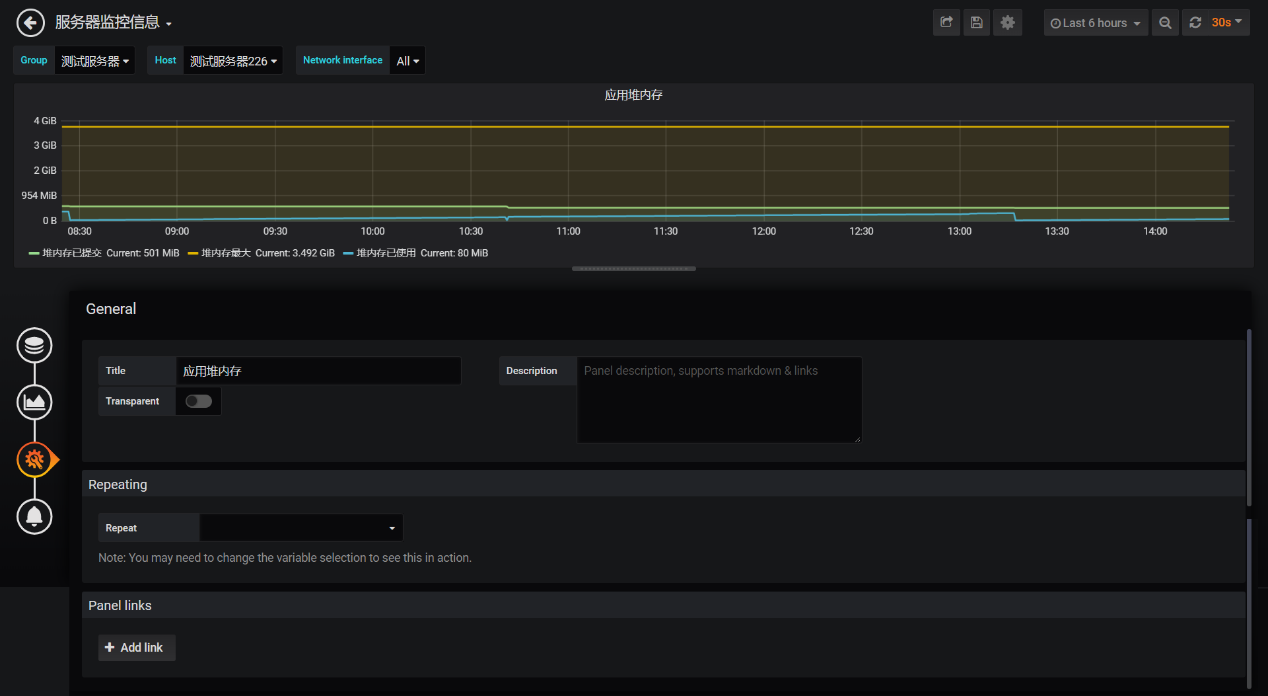


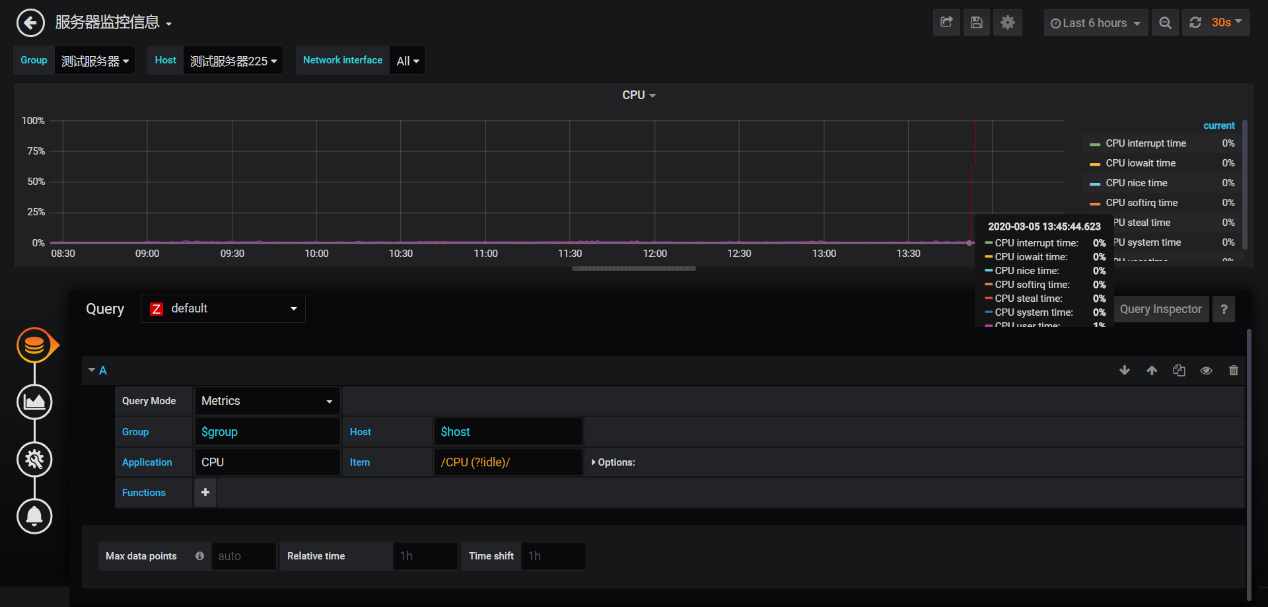


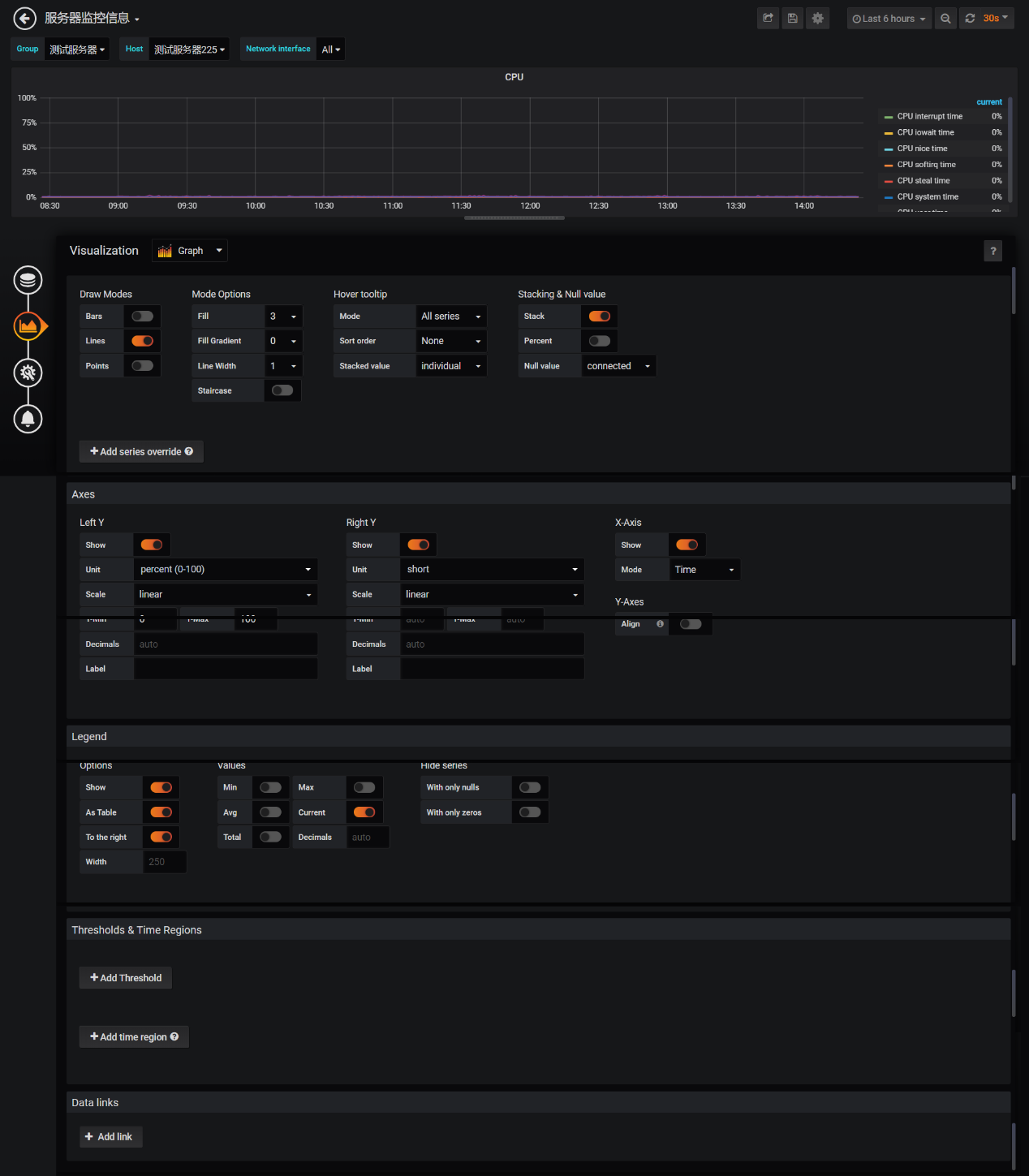


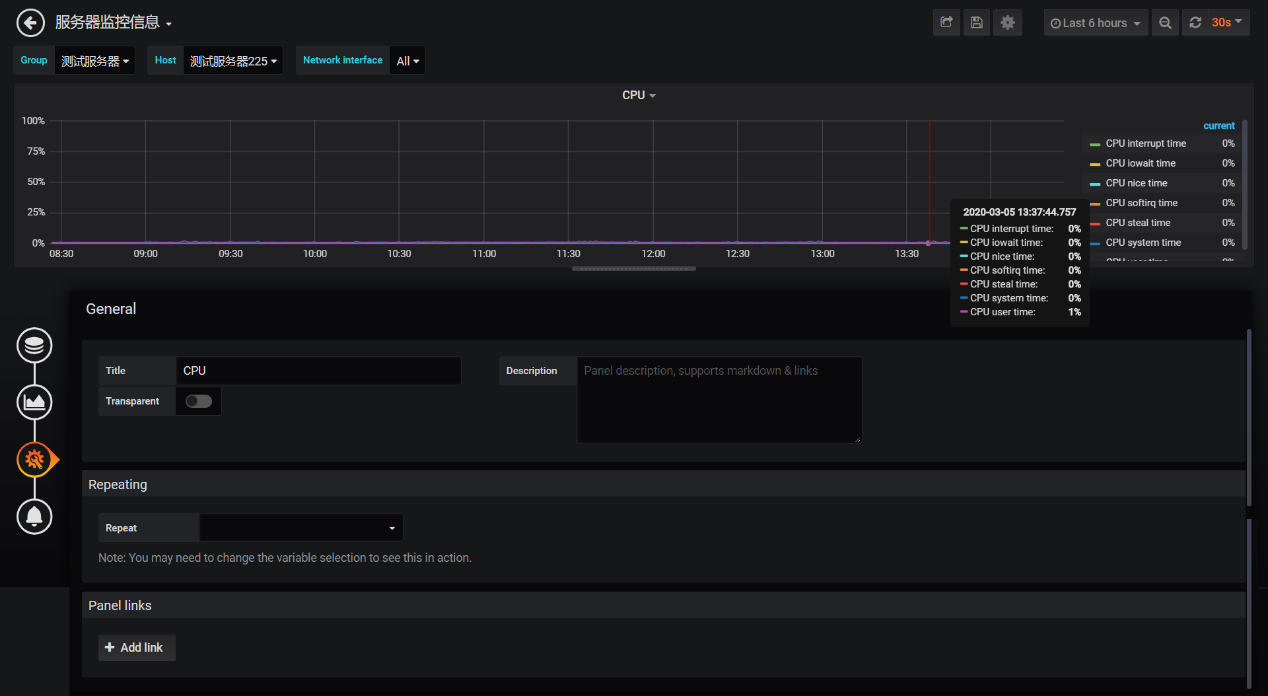


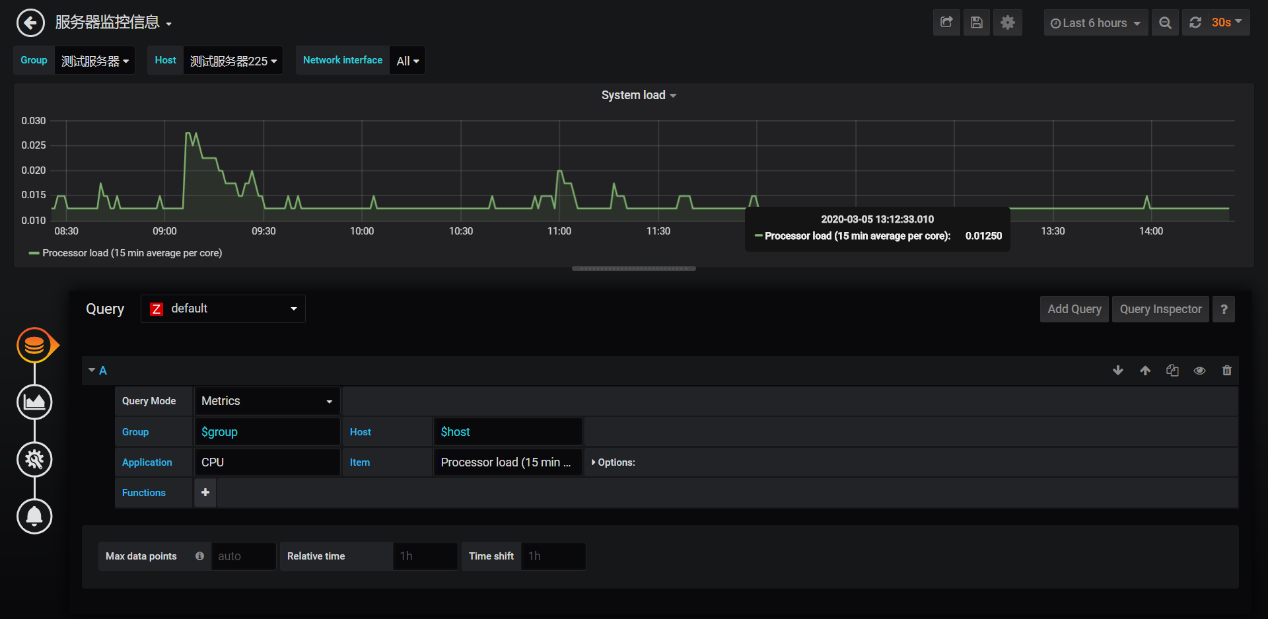


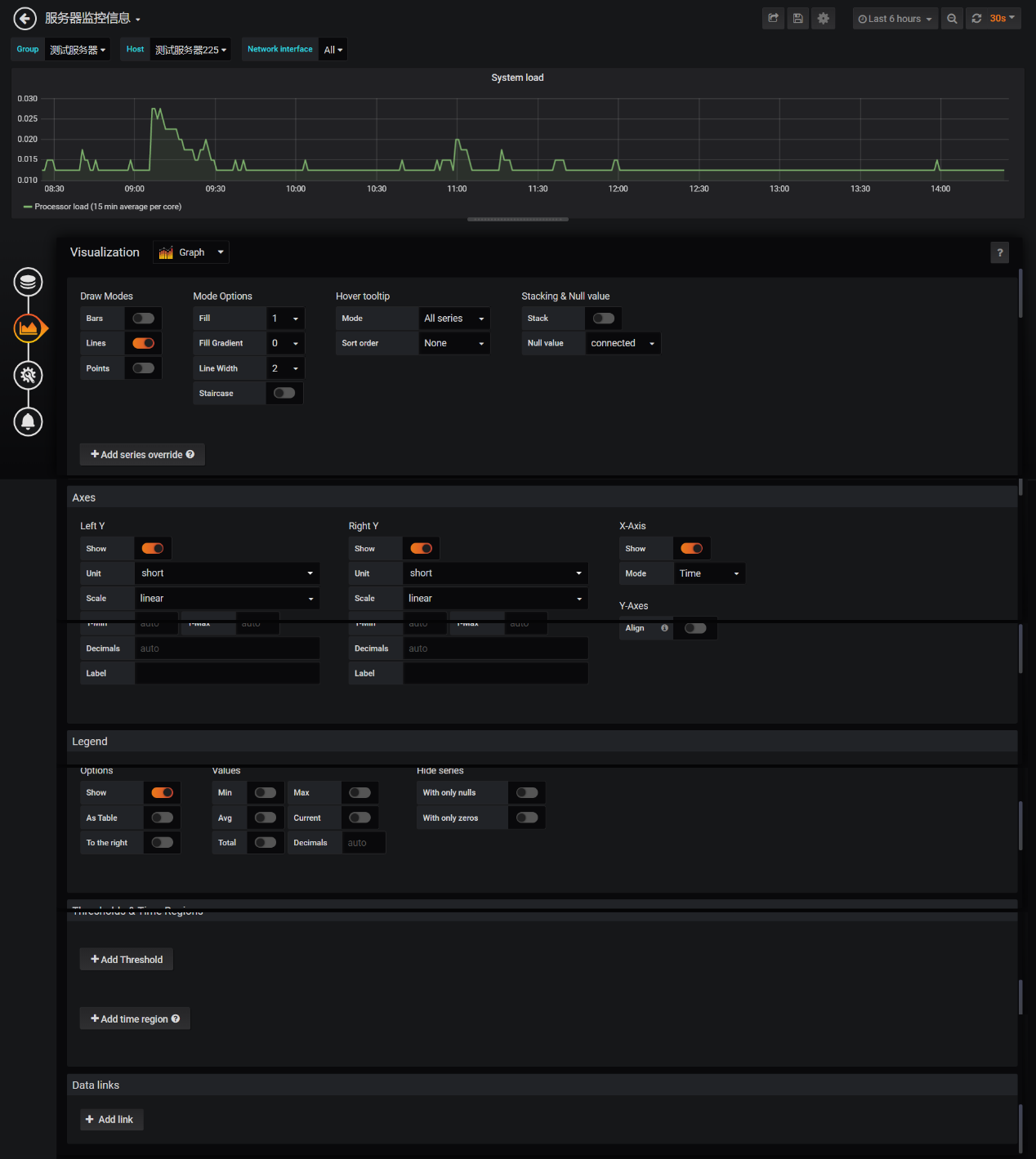


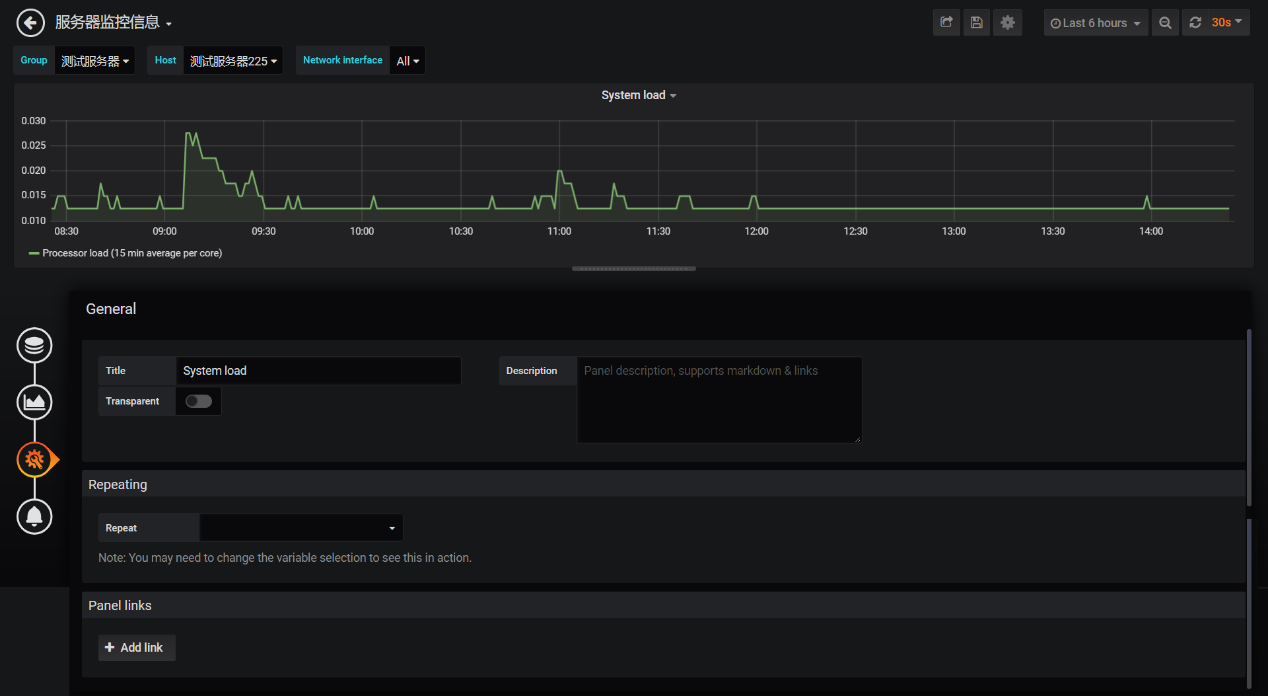


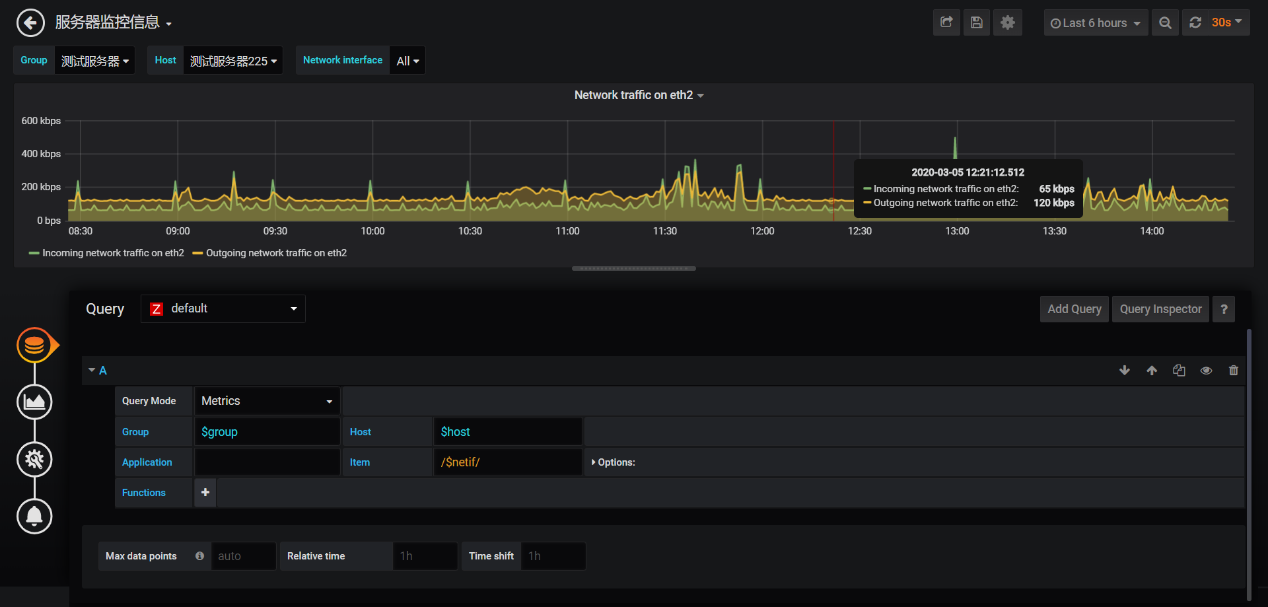


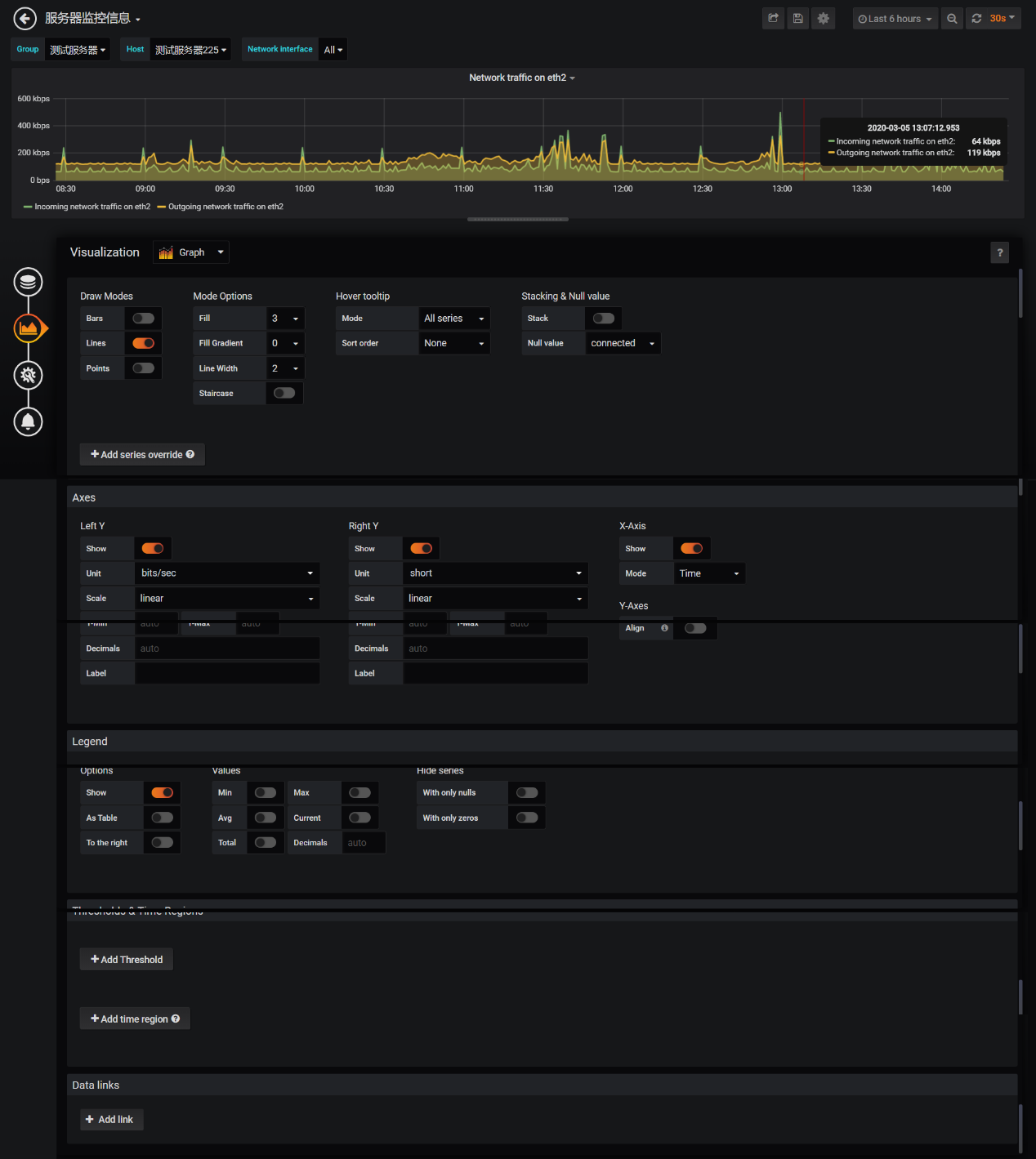














SLB监控

