**DASHBOARD FOR YOURSELF**

Contents

[**一．概要** 3](#_Toc520191745)

[1.1简介 3](#_Toc520191746)

[1.2 架构 3](#_Toc520191747)

[**二．Collector实现** 3](#_Toc520191748)

[2.1 准备环境 3](#_Toc520191749)

[2.2 调用wsgi接口 3](#_Toc520191750)

[2.3 增加cpu使用率数据采集 4](#_Toc520191751)

[**三．Prometheus配置** 6](#_Toc520191752)

[3.1 Prometheus安装 6](#_Toc520191753)

[3.1.1下载相应的Prometheus二进制包： 6](#_Toc520191754)

[3.1.2解压启动 7](#_Toc520191755)

[3.1.3 浏览器查询 7](#_Toc520191756)

[3.2 Prometheus静态配置 8](#_Toc520191757)

[3.2.1 静态配置 8](#_Toc520191758)

[3.2.2 浏览器查询 9](#_Toc520191759)

[3.3 Prometheus基于文件配置 10](#_Toc520191760)

[**四．Grafana配置** 11](#_Toc520191761)

[4.1 Grafana安装 11](#_Toc520191762)

[4.1.1下载 11](#_Toc520191763)

[4.1.2 配置启动 11](#_Toc520191764)

[4.2数据源配置 11](#_Toc520191765)

[4.3 dashboard配置 13](#_Toc520191766)

[**五．备注** 15](#_Toc520191767)

[5.1参考资料 15](#_Toc520191768)

***\* 版本修订记录 \****

|  |  |  |
| --- | --- | --- |
| ***版本号*** | ***修订时间*** | ***修订内容*** |
| *v1.0* | *2018-07-24* | *初版修订* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***\* Release Copyleft©free \****

**一．概要**

1.1简介

Dashboard for youserlf，这是一种图形化的方式，用于监控服务器或是服务器上服务，以及各种系统的日志输出，当然，该文档只值针对Unix-Like系统及其运行在其之上的服务进行监控。

1.2 架构

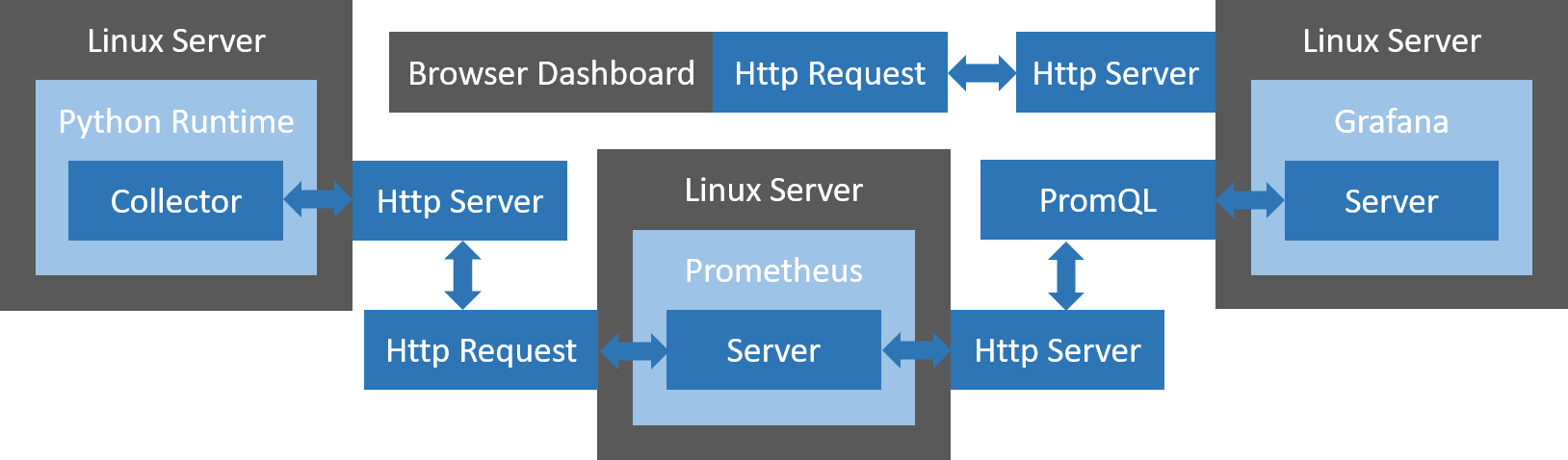


Figure 系统架构图

如上图所示，是相应的系统的架构图，主要由三个部分组成：

* 数据收集器Collector,主要负责目标服务器上的指标数据的收集和处理，是使用Prometheus Python API编写的客户端；
* Prometheus主要通过http请求将目标服务器上的指标数据存储在时间序列数据库中；
* Grafana通过配置Prometheus为数据源并从Prometheus获取数据生成需要显示的指标数据图表；

**二．Collector实现**

2.1 准备环境

|  |
| --- |
| # 首先确定安装了python环境，然后下载prometheus\_client  pip install prometheus\_client |

2.2 调用wsgi接口

|  |
| --- |
| #!/usr/bin/python  #-\*-coding: utf-8 -\*-  from prometheus\_client import make\_wsgi\_app  from wsgiref.simple\_server import make\_server  def main():  app = make\_wsgi\_app()  httpd = make\_server('0.0.0.0', 8000, app)  httpd.serve\_forever()  if \_\_name\_\_ == '\_\_main\_\_':  main() |

启动wsgi服务器：

|  |
| --- |
| python wsgi\_cpu\_exporter.py |

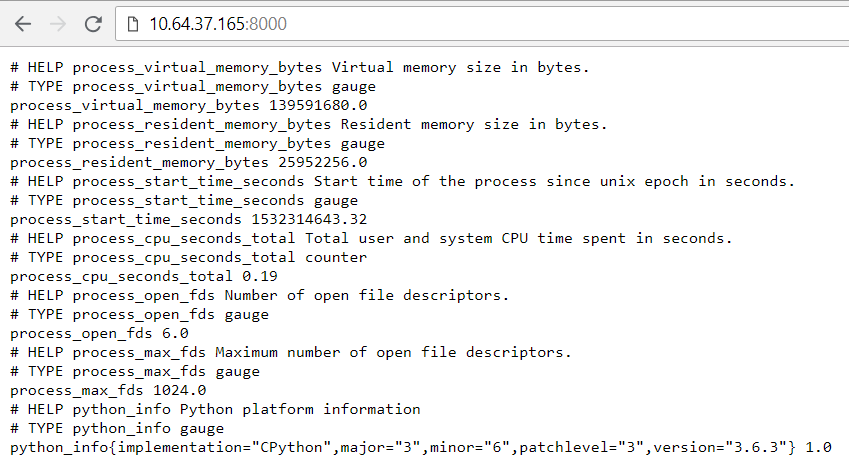
浏览器访问：

Figure exporter浏览器访问图

Prometheus在使用http请求指标数据的时默认的上下文为/metrics，以上启动的http server没有上下文限制，任何上下文都会返回如上图的数据，所以可以不配置相应的上下文；

2.3 增加cpu使用率数据采集

|  |
| --- |
| #!/usr/bin/python  #-\*-coding: utf-8 -\*-  from prometheus\_client import make\_wsgi\_app  from wsgiref.simple\_server import make\_server  from prometheus\_client.core import GaugeMetricFamily, REGISTRY  import subprocess  class CustomCpuCollector(object):  def collect(self):  p = subprocess.Popen(['iostat', '-c'], stdout = subprocess.PIPE)  o, error = p.communicate()  cpu\_list = o.decode().split('\n')[3].split()  user\_cpu = cpu\_list[0]  nice\_cpu = cpu\_list[1]  system\_cpu = cpu\_list[2]  iowait\_cpu = cpu\_list[3]  steal\_cpu = cpu\_list[4]  idle\_cpu = cpu\_list[5]  c = GaugeMetricFamily('cpu\_used\_ratio', 'cpu used ration', labels = ['type'])  c.add\_metric(['user'], float(user\_cpu))  c.add\_metric(['nice'], float(nice\_cpu))  c.add\_metric(['system'], float(system\_cpu))  c.add\_metric(['iowait'], float(iowait\_cpu))  c.add\_metric(['steal'], float(steal\_cpu))  c.add\_metric(['idle'], float(idle\_cpu))  yield c  def main():  REGISTRY.register(CustomCpuCollector())  app = make\_wsgi\_app()  httpd = make\_server('0.0.0.0', 8000, app)  httpd.serve\_forever()  if \_\_name\_\_ == '\_\_main\_\_':  main() |

在客制化的时候需要导入相应的指标类型包和REGISTRY包，其中指标类型包有一下四种：

* GaugeMetricFamily
* CounterMetricFamily
* SummaryMetricFamily
* HistogramMetricFamily

启动http服务器，浏览器查看是否有新增的指标数据：

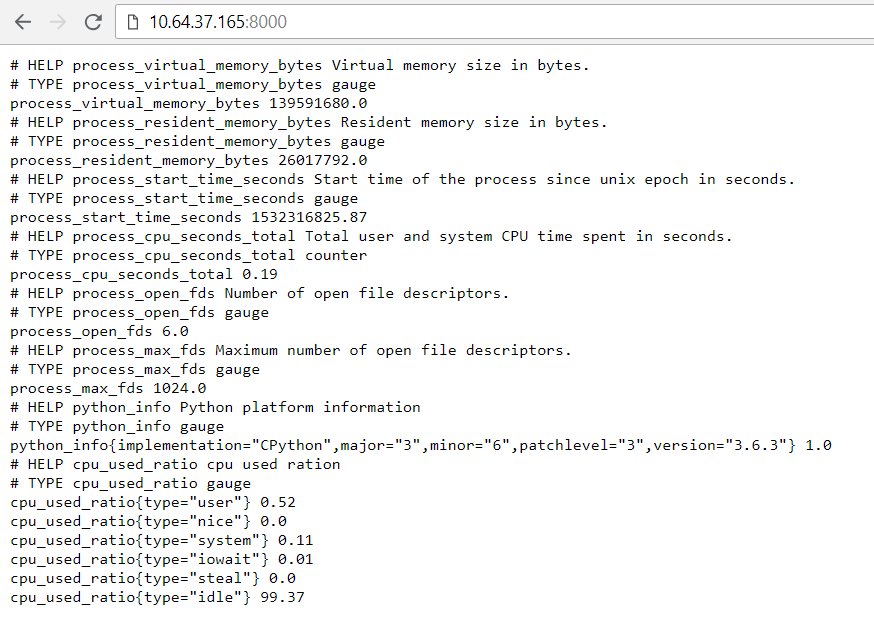


Figure 浏览器查看之自定义指标数据

从上图最下侧可以看出已经可以收集到服务器cpu的使用率了；到此exporter的代码就算完成了。

**三．Prometheus配置**

3.1 Prometheus安装

3.1.1下载相应的Prometheus二进制包：

<https://prometheus.io/download/>

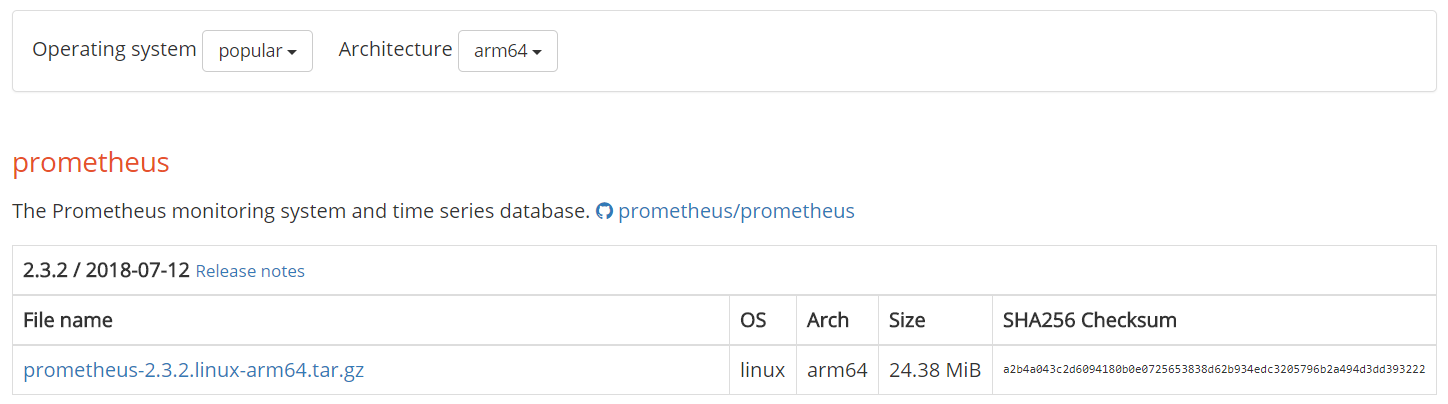


Figure prometheus下载

3.1.2解压启动

|  |
| --- |
| [dashboard@osd Prometheus]$ tar -zxvf prometheus-2.3.2.linux-arm64.tar.gz |

进入解压缩目录后即可启动,默认的启动端口为9090，可以使用命令行参数进行修改，另外，Prometheus会默认的监控自己的状态，在解压缩的目录的prometheus.yml的配置文件中可以设置监控的prometheus的端口，需要进行修改：

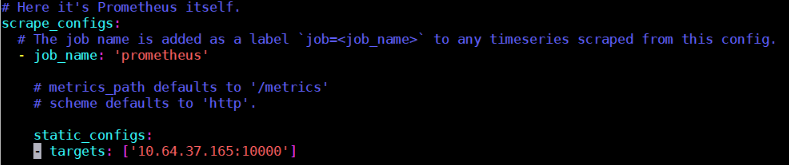


Figure 修改默认配置图

启动prometheus:

|  |
| --- |
| [dashboard@osd prometheus]$ ./prometheus --web.listen-address="0.0.0.0:10000" |

3.1.3 浏览器查询

如图所示，可以看到Prometheus已经启动成功；并且从图Figure 7中可以看到正在监控的目标服务。

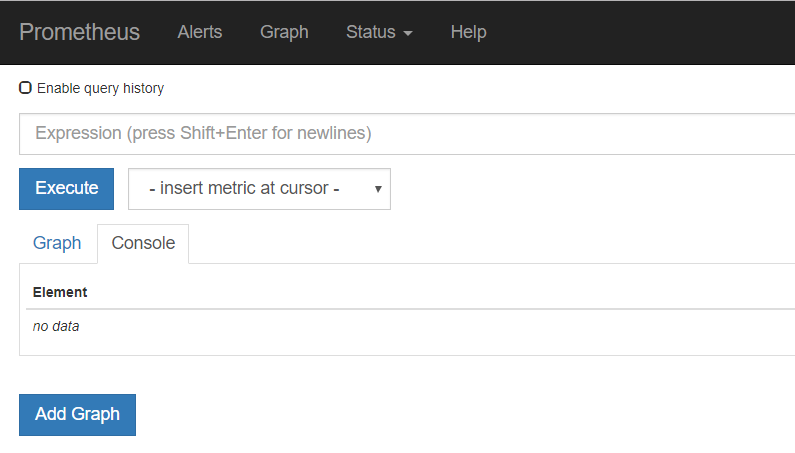


Figure prometheus web界面图

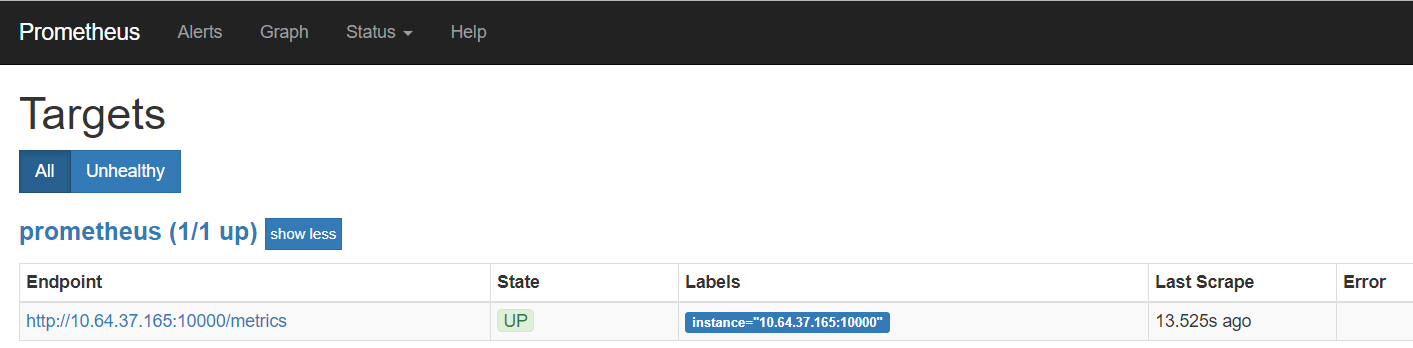


Figure prometheus target图

3.2 Prometheus静态配置

3.2.1 静态配置

为了能够收集自定义服务的数据，可以在prometheus.yml文件中进行静态配置，如图Figure8所示：

|  |
| --- |
| - job\_name: 'cpu\_test'  static\_configs:  - targets: ['10.64.37.165:8000'] |

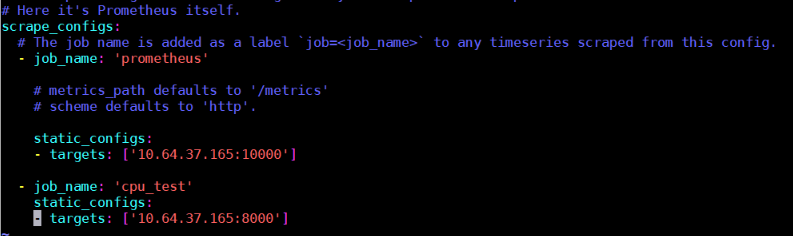


Figure 自定义collector监控配置

3.2.2 浏览器查询

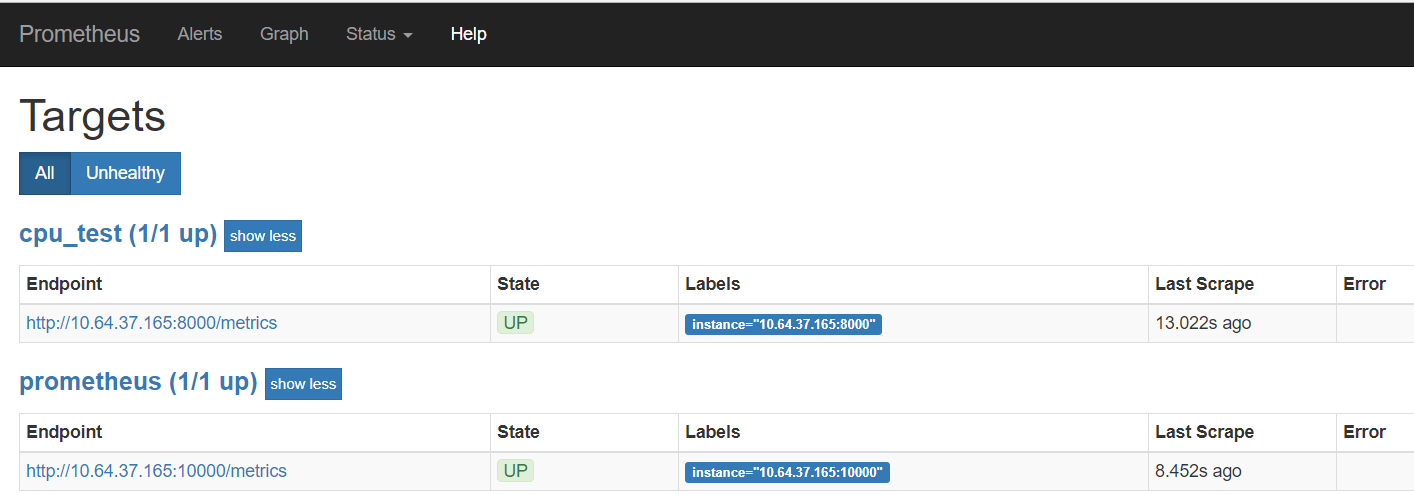


Figure custorm collector target

从上图可以看到自定义的collector已经Prometheus的监控列表中。点击Graph标签，下拉选中我们自定义的指标cpu\_used\_ration,点击Excute,可以看到相应lables的数据。选中Graph标签，可以看到如Figure11的图表；

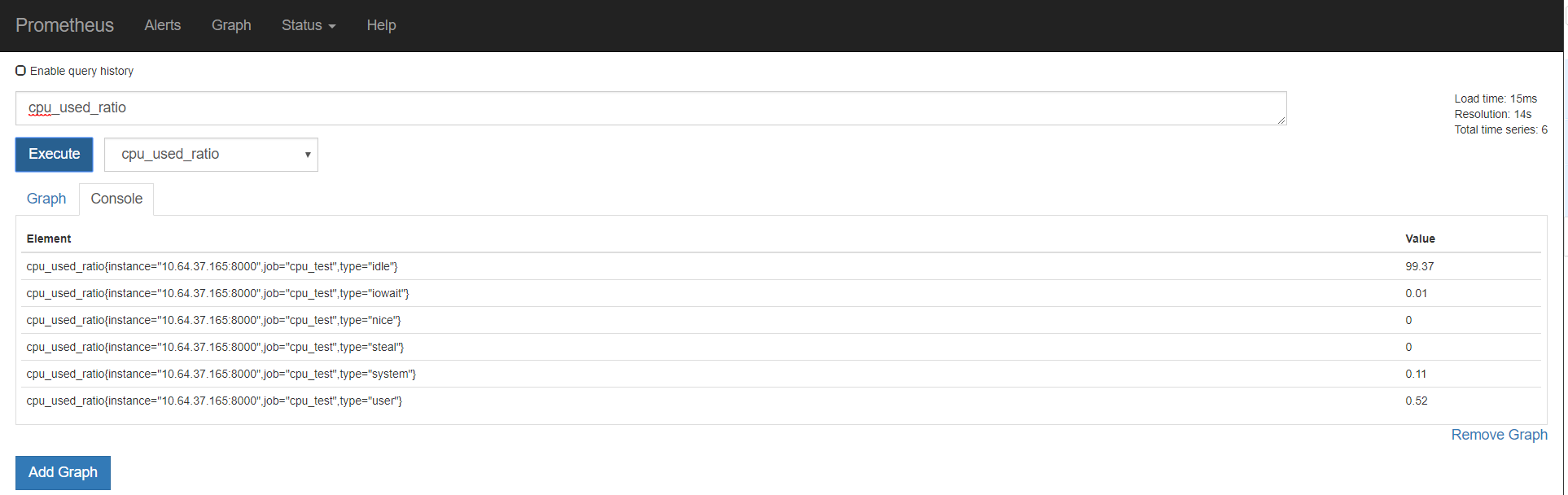


Figure 自定义指标数据查询

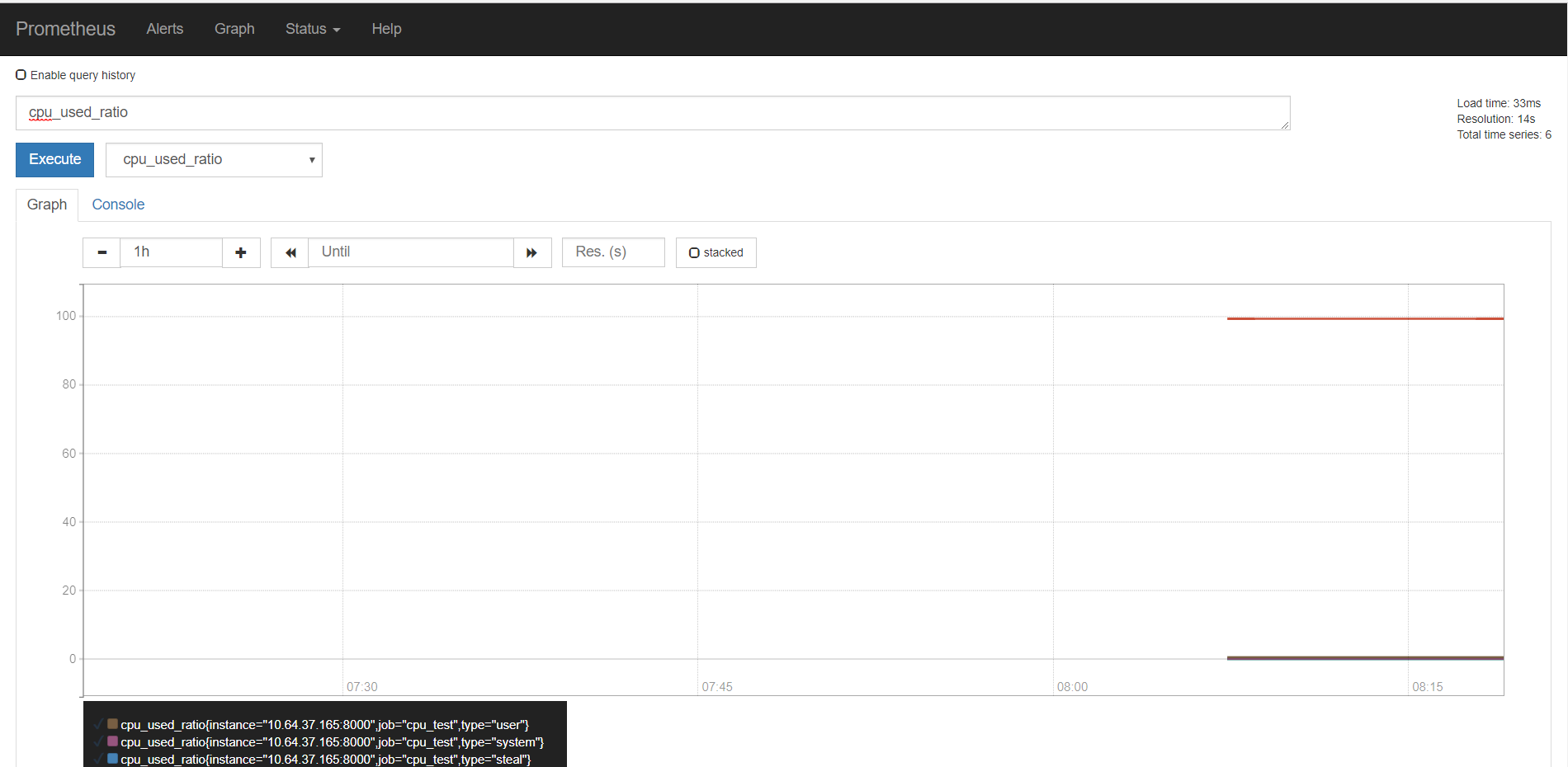


Figure 自定义指标图表

3.3 Prometheus基于文件配置

除了使用静态配置之外，可以使用基于文件的配置，Prometheus会定期的扫描配置文件，以此方式发现增加或减少的监控目标，这种配置方式无需重启Prometheus，而静态配置方式需要重启服务使监控目标生效；

首先，修改prometheus.yml为文件配置方式：

|  |
| --- |
| - job\_name: 'cpu\_test'  file\_sd\_configs:  - files: ['conf/\*.json']  refresh\_interval: 20s |

在当前目录下创建conf目录，并在conf目录下新增.json配置文件：

|  |
| --- |
| [dashboard@osd prometheus]$ cat conf/cpu\_test.json  [  {  "targets":["10.64.37.165:8000"]  }  ] |

监听配置完成后可以在浏览器端查看是否有自定义的监控目标；另外，可以使用下面的命令检查配置文件是否正确：

|  |
| --- |
| [dashboard@osd prometheus]$ ./promtool check config prometheus.yml  Checking prometheus.yml  SUCCESS: 0 rule files found |

**四．Grafana配置**

4.1 Grafana安装

4.1.1下载

进入Grafana官网，下载相应系统的安装文件;

<https://grafana.com/grafana/download>

4.1.2 配置启动

如果下载的为二进制包，解压缩后可以修改conf文件下的配置文件来修改端口号，默认端口号为3000。Grafana默认使用的数据库为sqlite,无需安装，如果安装了mysql或其他的数据库也可以进行配置；配置完成后进入bin目录下直接执行二进制文件启动，首次启动时会在数据库中创建相应的表：

|  |
| --- |
| [dashboard@osd bin]$ ./grafana-server |

启动完成后静茹浏览器进行访问，默认用户名和密码为admin/admin,并且会提示修改密码：

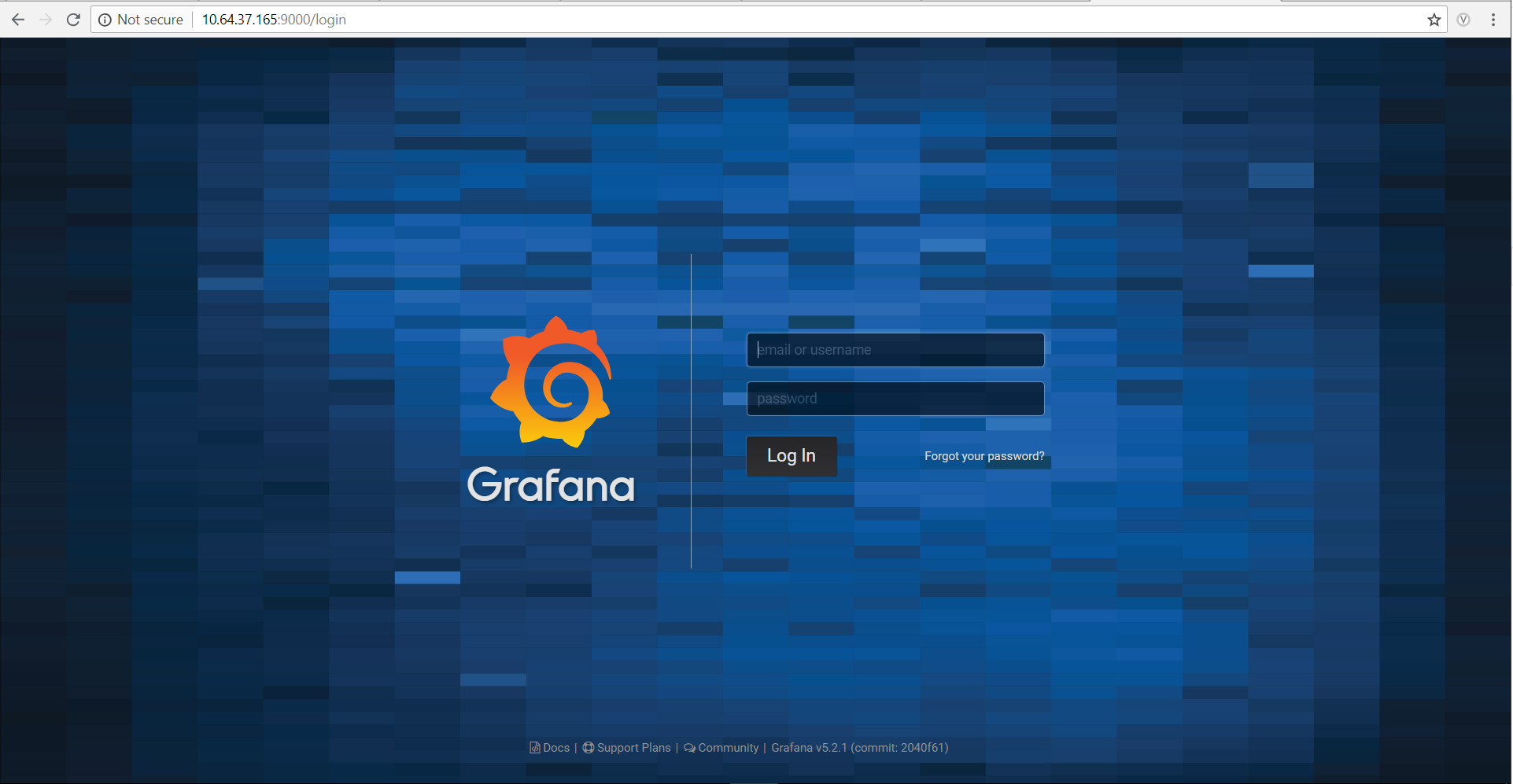


Figure Grafana首次登陆页面

4.2数据源配置

进入系统后，首先需要配置数据源：

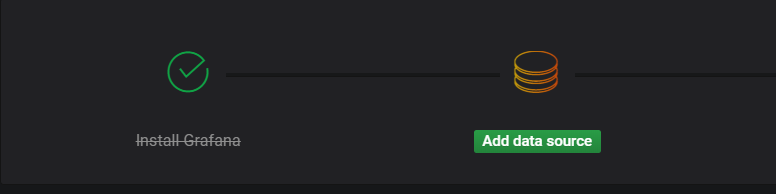


Figure 配置数据源入口

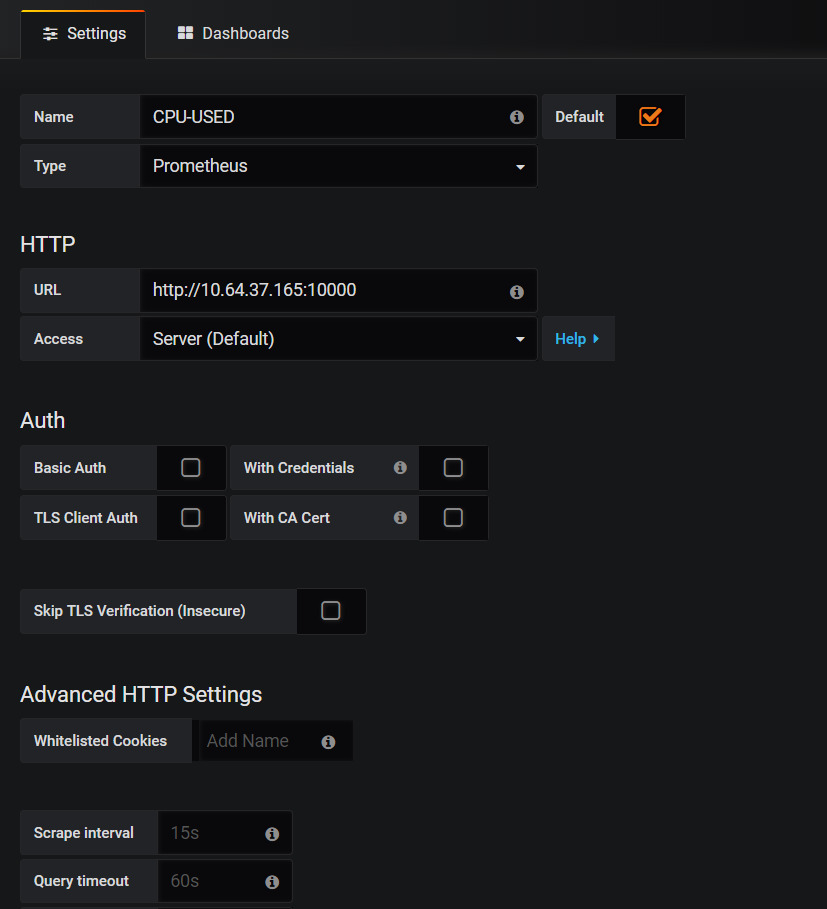


Figure 数据源配置图

如图所示，选择数据源类型为Prometheus，并且在地址栏填写Prometheus的地址信息，然后点击保存测试即可；Grafana会自动测试数据源是否正常运行；

4.3 dashboard配置

数据源配置完成后，接下来即可配置dashboard,回到系统主页面，点击dashboard新增：

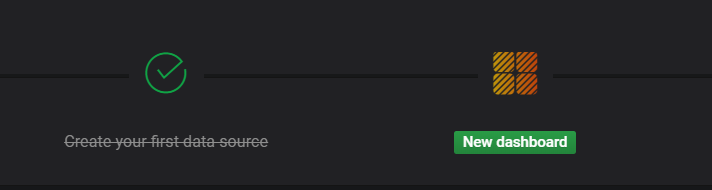


Figure Grafana新增dashboard入口图

以下为可以新增的dashboard类型，这里选择Graph,将自定义的CPU使用率以图表展示；

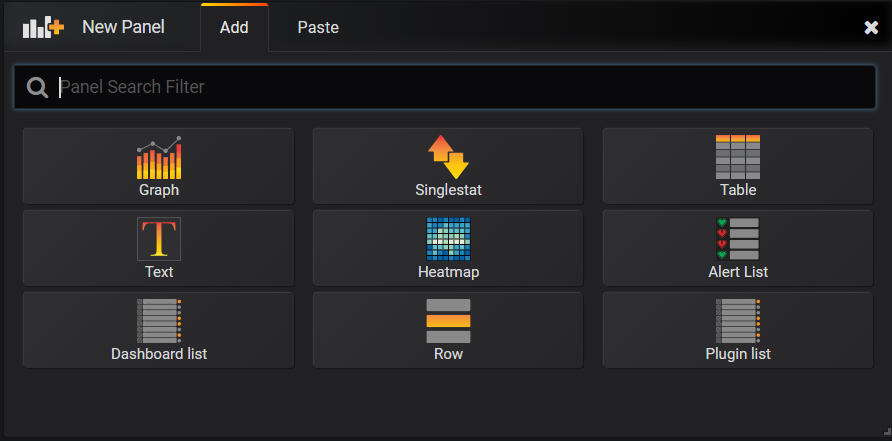


Figure Grafana dashboad类型图

新增的图表需要配置数据源指标后才能显示，如图所示编辑图表：

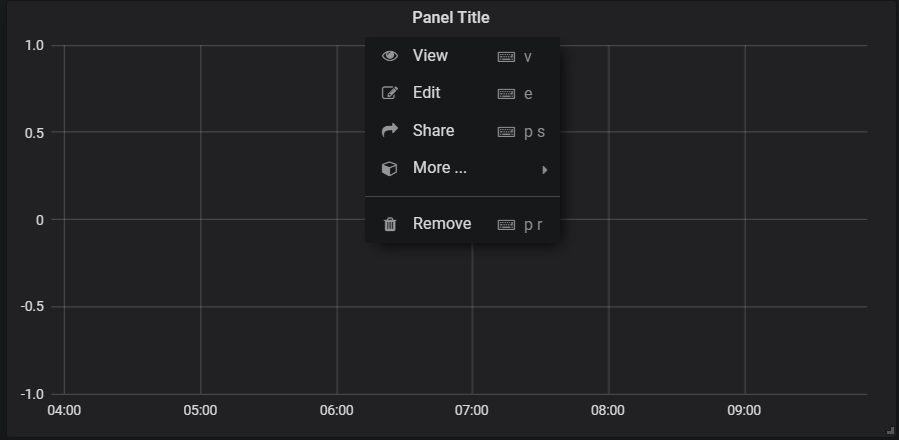


Figure 编辑图表

首先可以修改图表的标题：

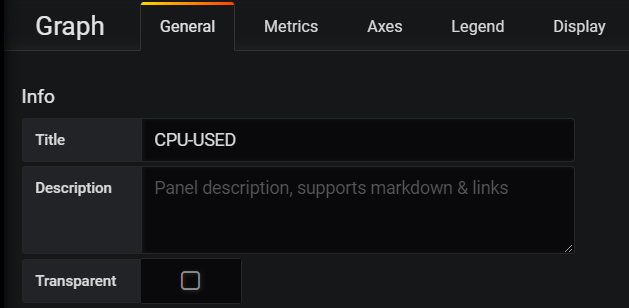


Figure 图表标题修改

为了显示数据，需要选择数据源中的监控指标，另外可以修改图表的lengend使用指标数据中的标签作为名称；

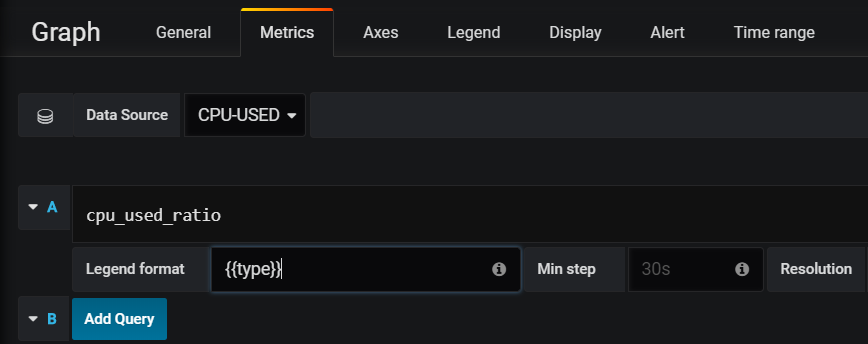


Figure 图表指标数据配置

保存后即可生成自定义的CPU使用情况的图表：

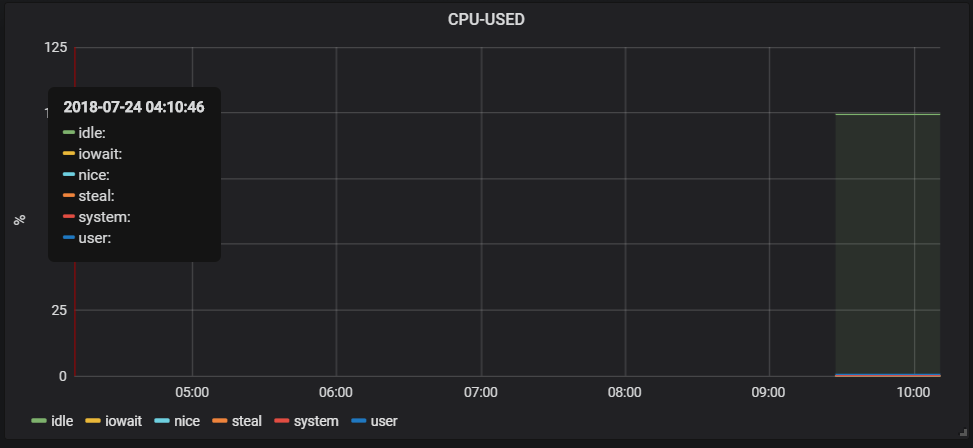


Figure 自定义CPU使用率

**五．备注**

5.1参考资料

【1】Grafana官网<https://grafana.com/grafana/download?platform=arm>

【2】Prometheus官网<https://prometheus.io/docs/introduction/overview/>

【3】Python\_exporter <https://github.com/prometheus/client_python>