> 本文由 简悦 SimpRead 转码， 原文地址 kaiwu.lagou.com

本课时我们来学习：Prometheus，并结合组件 Grafana，整体介绍一套监控方案。

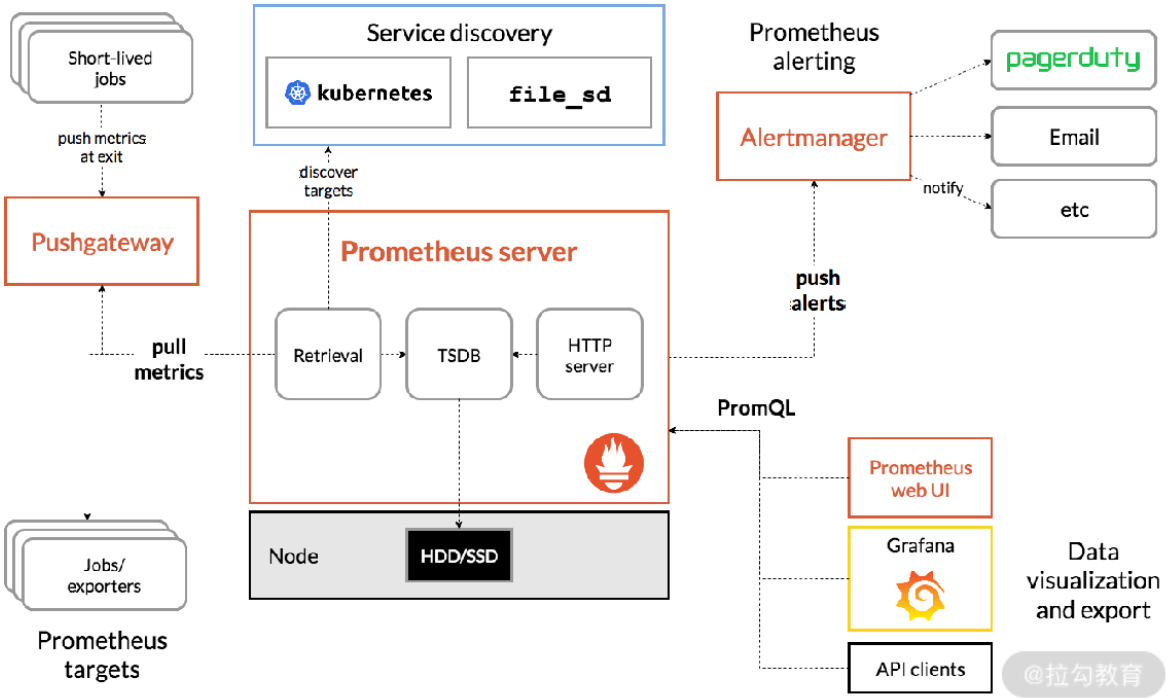
Prometheus 介绍

首先来介绍 Prometheus，可能你对它的了解相比 Zabbix 会更陌生一些，Prometheus 是一套由 GO 语言开发的开源监控系统。它是继 Kubernetes 之后的第二个 CNCF 托管项目，近些年被广泛使用在基于 K8s 或 Swarm 这种容器编排的整体服务平台中作为监控系统使用。它的官方网站是 https://prometheus.io/（如果想具体了解的话，你可以登录官方网站，去详细看它的帮助使用手册）。

Prometheus 是由一套组件所组成，接下来我们具体来介绍一下它的核心组件：

- 第 1 个组件是 Prometheus server（主要组件），它是一个核心组件，用于拉取监控数据并存储到时序数据库，相当于 Prometheus 整个监控系统里面的心脏。
- 第 2 个就是客户端 SDK，用于植入监控的应用程序中，完成数据的采集。
- 第 3 个组件就是 Push Gateway，它是一个支持客户端使用主动推送监控数据的中间网关。刚讲到 Prometheus server 默认以拉取监控数据的方式来抓取客户端数据。当我们的客户端需要通过 push 主动推送时就需要 Push Gateway 组件来作中间转化。
- 第 4 个组件是 Exporter，它是 Prometheus 的一类数据采集组件的总称，负责从目标节点处搜集数据，并将其转化为 Prometheus 支持的抓取的格式，这提供了一个统一形式的接口供服务端抓取。
- 第 5 个组件是 Alertmanager，警告管理器，在监控系统中负责报警模块，所有 Prometheus 发出的报警都是通过它进行处理和触发告警消息的。

了解了几个核心组件后，我们来看一下 Prometheus 这些组件所构成的 Promethues 整体监控架构。



在这张图里我们可以看到，中间的位置是 Prometheus server 核心组件，这个核心组件串联 Prometheus 整体监控流程。在 Prometheus server 这个大框里面有几部分：一个是收集模块（Retreval），另外一部分就是 TSDB，这是一个时序的数据库，HTTP server 是一个提供给外部访问的服务接口。

我们先来看收集模块，这个模块上面的 “discover targets”- 代表去发现监控节点目标， Prometheus 一个非常重要的特性就是完美支持 K8s 对于 Docker 容器编排的服务架构，它能够动态地发现 K8s 集群里面目标节点 pod ，发现并提取所有节点的监控信息，通过 pull metrics 针对性地提取具体每一个节点 pod 上面的监控数据。

接下来我们看到图中的左侧，它展示了发现目标节点信息以后，具体提取这些节点上面的监控数据的方式。我们看到第一种方式就是通过 exporters 组件去进行节点数据采集，并标准化成一个通用的 metric 接口，给到 Prometheus server 的服务端来进行抓取。

第二种方式是：Prometheus server 抓取一个 Push gateway（推送网关），客户端推送信息传递到推送网关，通过推送网关来进行转换，然后服务端再去拉取网关的数据，这样相当于 Push gateway 做了一层转换。

收取监控数据以后，Prometheus server 会存到自己的时序数据库里。存取到数据库以后，通过 Prometheus server 端的运算、转换，逻辑处理，最后提供给对外部 HTTP 接口进行调用访问。

如果触发了一些报警的规则，则需要把这些消息的报警规则推送给 Alertmanager 进行处理，可以对外发送邮件，提醒信息等行为，这些行为都是由 Alertmanager 组件负责实现的。

另外就是可视化展示了，可视化的展示需要集成一些可视化插件。刚刚讲到的 Grafana 就属于可视化的插件，可以从 Prometheus server 里进行数据提取和展示。除了 Grafana 以外，还有一个是 Prometheus web UI，这是 Prometheus 默认自带的一个可视化管理后台。

以上就是基于 K8s 服务架构下，Prometheus 组件的整体架构。

了解了这些组件贯穿的架构图示之后，我们接下来会重点讲解 Prometheus server、exporters、Grafana 这三个组件，为你演示从数据的采集到 Prometheus 服务端抓取 Grafana，也就是数据的监控信息展示报表，这个流程架构是如何实现的。

Prometheus 和 Zabbix 功能、使用差异

在介绍整体架构之前，想必你对我们整个模块里讲到的两个监控系统：Zabbix 和 Prometheus ，那它们有什么区别？在这里我列了一张表格，分别从开发语言成熟度、性能、社区活跃度以及容器 K8s 微服务的支持，还有部署复杂度以及监控配置的复杂度等维度来为你做了一个对比。

	开发语言	成熟度	性能	社区活跃度	容器、k8s、微服务	部署复杂度	监控配置复杂度
zabbix	php	高	中	中	低	中	中
Prometheus	go	中	高	高	高	低	高

它们之间的不同主要体现在以下几个方面：

首先 Zabbix 是由 PHP 开发的，而 Prometheus 是 GO 语言开发的。其次，从开发时间来看，Prometheus 比 Zabbix 晚出来很多年，没有太多复杂多余的逻辑，再次存储数据库的方式也不同（后面具体介绍），这些因素使得 Prometheus 在性能上会比 Zabbix 更优。

第二，正是由于 Zabbix 问世的时间更早，所以它的代码成熟度会略高，而 Prometheus 由于出来的更晚，所以它可能会有代码需要加固。

第三，Prometheus 相对 Zabbix 而言，可能存在的另外一个劣势就是配置的复杂度更高，我们在上节课里面有讲到 Zabbix 的自动发现和自动上报，Zabbix 的监控配置在控制台相对完善，而 Prometheus 则需要去进行很多配置项的和数据规则这样的运算配置，所以它的复杂度相对 Zabbix 而言会更高一些。

于其他维度分析，我们看到 Prometheus 均是优于 Zabbix，尤其是对于 K8s + 容器的支持，而 Zabbix 到了很晚才去兼容支持 K8s 这种动态发现、编排容器的架构模式，所以当前很多 K8s 部署场景里的监控系统都会基于 Prometheus 来实现。

此外，再总结一些更多 Prometheus 的几个特性：

1. Prometheus 采用了一种 Pull（拉）且 HTTP 的方式获取数据降低客户端的复杂度，服务端可以更加方便地水平扩展。
2. Prometheus 存储使用时序数据库 Zabbix 采用关系数据库保存，这极大限制了 Zabbix 采集的性能， Prometheus 自研一套高性能的时序数据库， 在 V3 版本可以达到每秒千万级别的数据存储。

Prometheus+grafana 结合

接下来我们就拿 Prometheus + Grafana 结合来为你做一个搭建演示，来讲解这套结构应该如何配置。

这里分 4 大块，一个是安装和配置 Prometheus；第二块就是安装或者配置 Grafana UI 的外围控制台展示；第三是配置 Grafana 数据源，因为 Grafana 是需要通过 Prometheus 去抓取的，我们需要配置的 Grafana 所采集的 Prometheus 数据；第四就是我们需要配置客户端 Expprters 作数据采集.

首先我们来讲解 Prometheus 安装，它的安装比较容易。

Prometheus 它可以支持源码编译的安装，也可以支持容器的安装，这里我们来介绍容器的安装方式。

如果安装容器的话，我们通过安装 Docker 这样的容器组件，通过 yum install docker 先安装好 Docker 这个服务，并且启动通过 systemctl start docker，这种 CentOS7 的服务管理启动方式，启动 Docker 的服务，然后我们执行 docker pull prom/prometheus ，这样就从远端拉取了一个官方的 Prometheus 镜像。

```
docker run -d -p 9090:9090 \

    -v ~/docker/prometheus/:/etc/prometheus/ \

    prom/prometheu
```

然后我们就可以 docker run 了，直接以服务的方式运行，然后做一个端口映射，把本地的 9090 端口映射到容器的 9090 端口里面去，并且做一个数据卷的挂载。这个就是把 Prometheus 的配置文件挂载到了本地的一个目录（~/docker/prometheus/）下去，因为我们需要去修改 Prometheus 的配置文件，这样的话会更加方便。 后面加的就是拉取的镜像名称，这样就可以完成这个容器的启动，Prometheus 服务就已经安装完成。

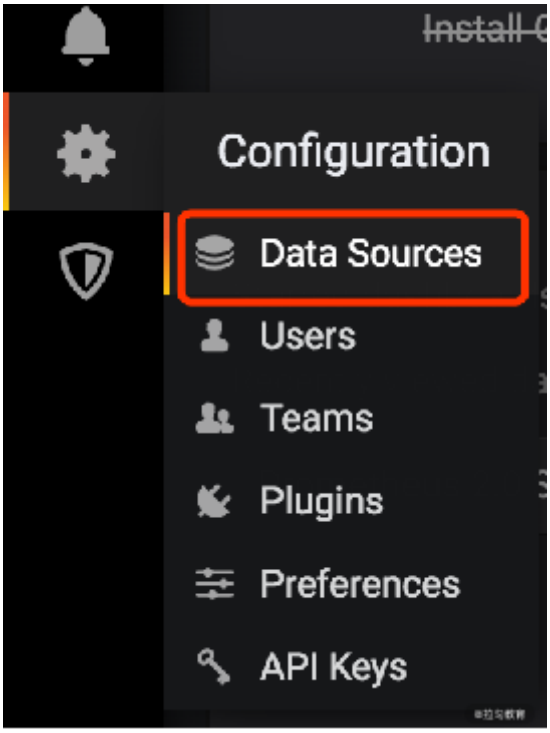
安装完成以后，接下来我们需要修改配置文件，它的配置文件是在我们本地目录路径 /root/docker/prometheus/prometheus.yml （主配置文件）。这个文件里我们首先要来看的有两大块，一个是 global 块，这里面定义的是 Prometheus 的全局新配置，还有 scrape_config 块，这里定义了 Prometheus 要抓取的目标，这块的配置是我们需要修改的，我们可以等到把客户端整体搭建起来以后，再来修改配置文件。

整个启用完成以后，我们可以在浏览器里面访问你的本机 IP 地址和 Prometheus 对外暴露的 9090 端口，这样就可以登录到 Prometheus UI 的控制台。

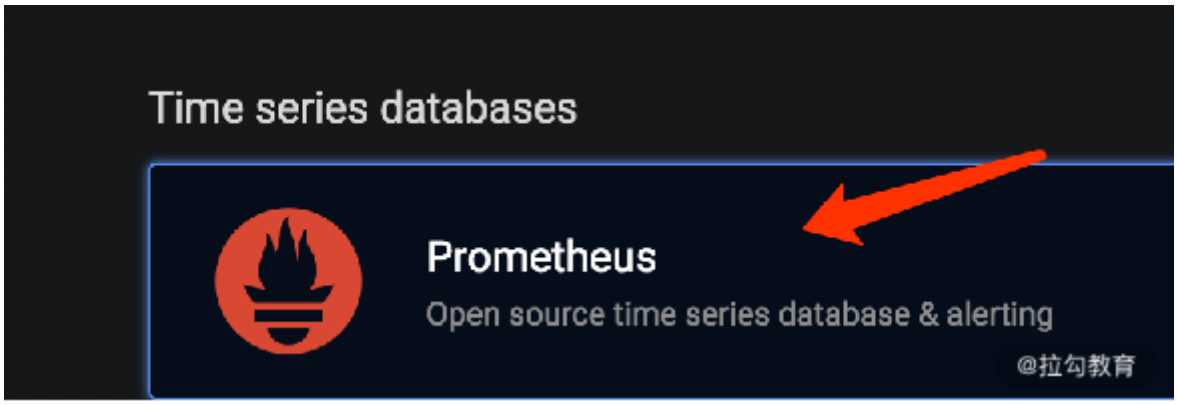
```
docker run -d -p 3000:3000 grafana/grafana
```

安装好 Prometheus 以后，接下来需要安装的就是 Grafana，它是一个 Web 界面化的监控数据的展示平台，我们同样可以通过 docker run 的方式来完成一键化安装，并且 3000 的端口对外服务。完成安装以后，同样可以在浏览器里访问 {ip}:3000 的端口，就可以访问到 Grafana 控制台的界面，它有一个默认的用户密码，用户是 admin，密码也是 admin，我们登录进去以后，就完成了 Grafana 的安装。

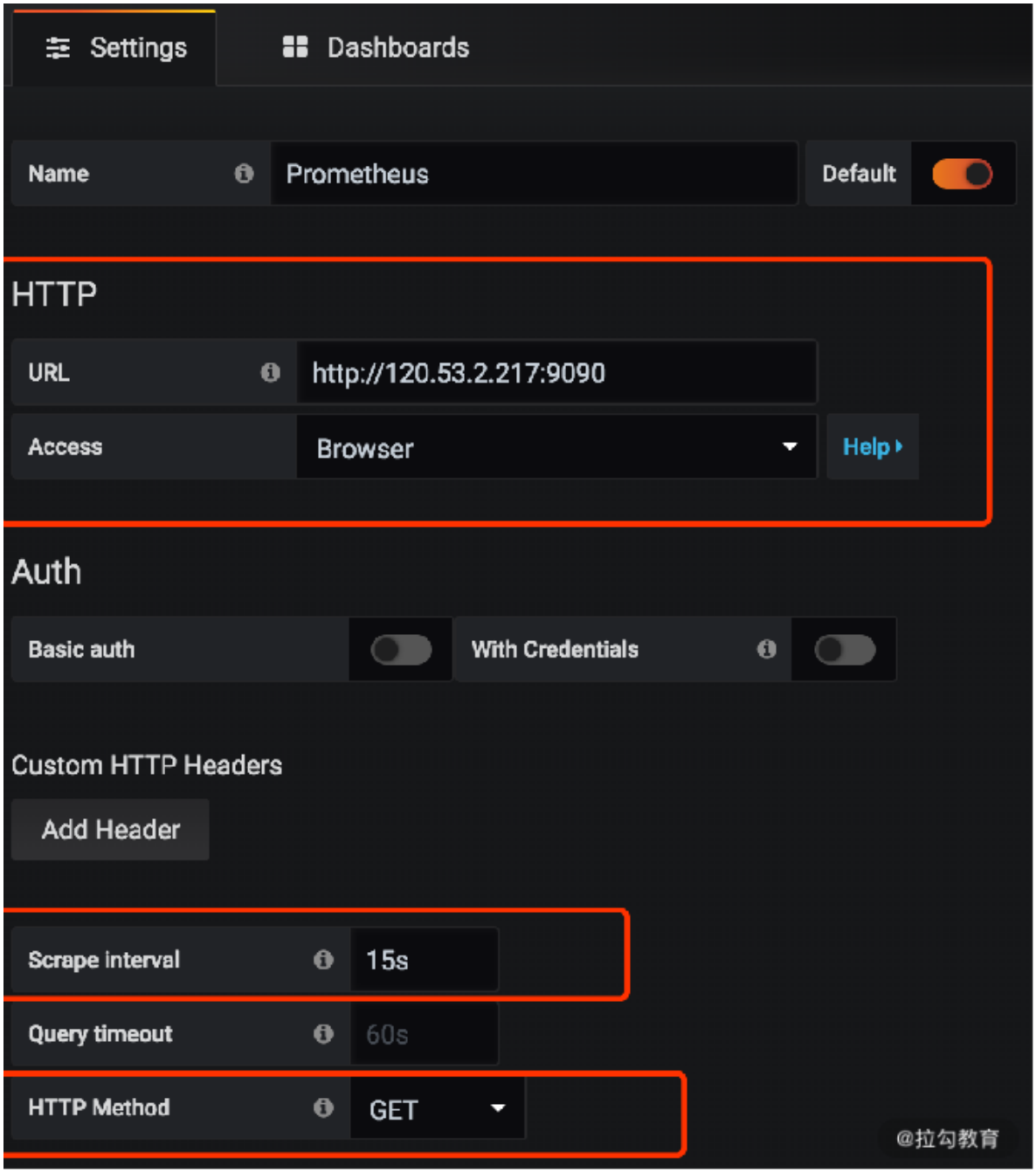
接下来需要配置的就是 Grafana 的采集数据源要从哪个地方采集，所以我们需要把 Grafana 的数据源配置成从 Prometheus 去采集，我们在登录到 Grafana 界面以后，在浏览器里面点击 Data Source 这个按钮：



然后点击 add data source，它里面有一个默认的配置模板，我们可以点击 Prometheus：



然后我们就可以进入 Grafana 的控制台，针对 Prometheus 这样的一个数据源，它的具体的配置界面。这里需要配置的有如下几大块配置：



一个是 HTTP，也就是 Prometheus 对外服务的接口的地址，我们填写的是 Prometheus 的服务地址和它对应的服务端口，并且设置它的权限是浏览的权限。另外 Scrape interval 配置的是采集间隔，也就是每 15 秒去做一次采集。HTTP Method 代表是以 GET 的方式去请求 Prometheus 服务。这里就完成了整个对于 Prometheus 数据源的采集。

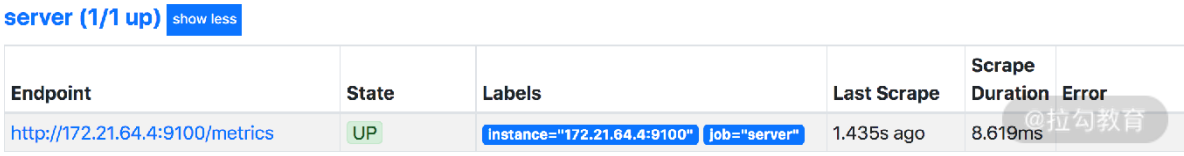
接下来我们需要拿一个客户端的服务器去采集监控数据，登录 Grafana 去查看，是否能看到 Prometheus 的这些监控数据，并且做报表化的展示。

接下来我们就来安装这个客户端，我们需要安装的是 node_exporter 组件，我们可以通过这个 GitHub 地址去下载源码包：
https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node_exporter-0.18.1.linux-amd64.tar.gz

下载完成以后直接解压，完成之后会直接到这个界面上面，通过 cd node_exporter-0.18.1.linux-amd64/ 目录下，并执行 ./node_exporter 二进制命令，就能启动客户端采集，它提供这样的一个 metric 数据接口给到服务端来进行拉取。

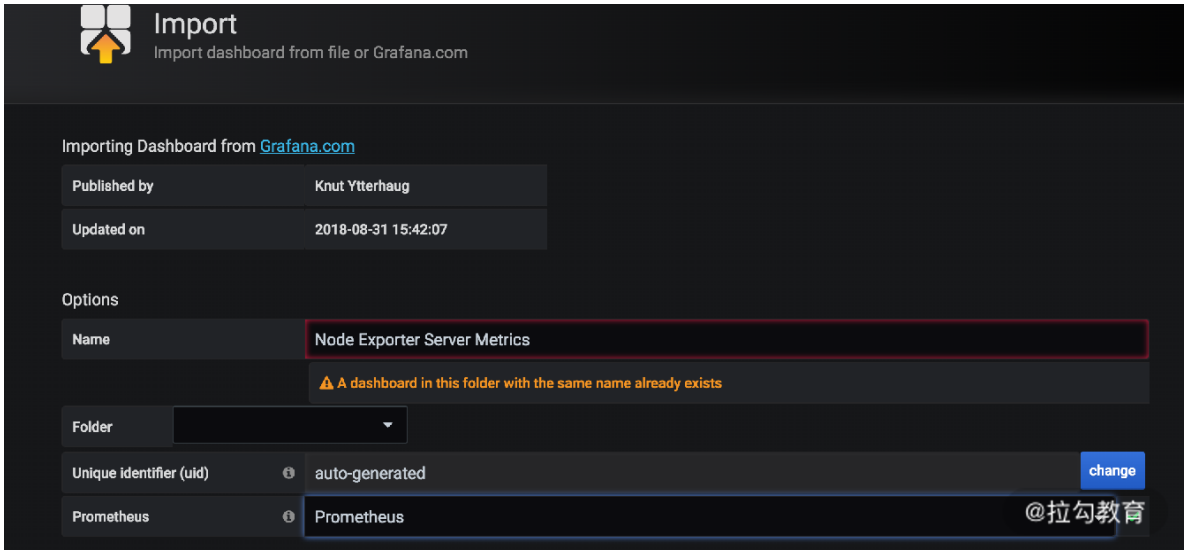
接下来 Prometheus 的服务端配置文件中配置客户端采集 metric 接口地址。

刚刚讲到的 Prometheus 配置里面有一个 scrape_configs，我们来配置它所采集的目标服务器，填写好目标客户端的服务器 IP 地址和对应的 exporter 所监听的端口，默认是 9100，然后我们重启一下 Prometheus 的服务端容器。接下来我们在整体界面进行检查，看是否有对应的数据产生，我们先通过 Prometheus 的界面来检查，可以登录到 Prometheus 的管理界面里，点击 Status，会看到这里有一个目标节点，以及状态、时间。

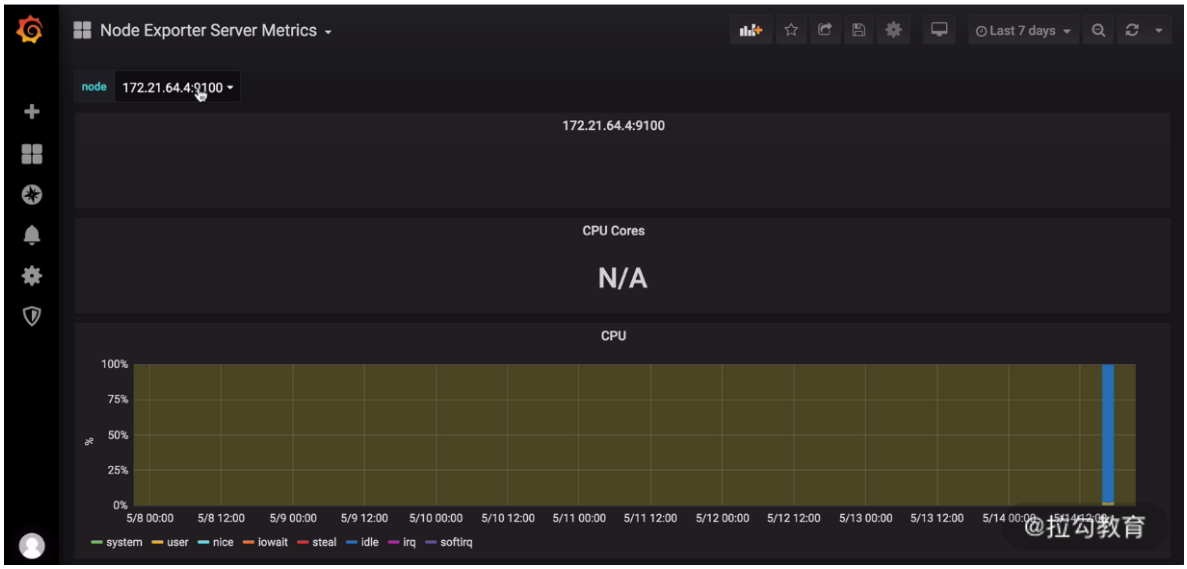


最后我们就可以登录到 Grafna 的控制台，然后来看一下客户端的这些数据是否在 Grafna 里进行了报表化的展示。接下来需要导入 Grafna 对于普罗修斯所默认携带的面板插件。

我来做一个演示，登录到我的浏览器打开 grafana 后台，然后点击 Dashboards 按钮，下面有一个 manager，然后点击 import，这里我搜索一个关键词 “405“，然后回车一下。这个时候我们点击完 load ，界面中有 Prometheus Metric 的插件，然后我们点击 input，这样就完成了面板插件导入。



最后我们点击主界面，这个时候我们在 Dashboard 的下面会看到一个新的 Dashboard 按钮（Node Exporter Server Metrics），我们点击进去，就会看到对应节点的 IP 地址和端口信息：



这个一个节点就是我们刚刚所启用的新节点，在这里我们可以下拉来查看它相关的数据，如第一排是 CPU 的相关监控数据，后面是内存的数据，负载的数据，磁盘的一些使用数据，还有相关的一些监控报表数据，它们都会以图表的形式进行界面化的展示，我们可以在图形界面的控制台里面，直观地看到这样一些监控数据。

到这里就完成了我们这节课所讲到的一个演示，从客户端 exporter 去做数据采集到服务端的数据拉取，以及 Grafna 最终的图表展示。