

实现跨平台实时可视化监控
(nimon+InfluxDB+Prometheus+Grafana)

V1.0

浪潮商用机器有限公司
— 技术支持部 李松青
2021.06

目录

1	概念简介	4
1.1	nmon.....	4
1.2	njmon,nimon	4
1.3	InfluxDB.....	4
1.4	Telegraf	5
1.5	Prometheus.....	6
1.6	Grafana.....	8
2	可视化监控方案	9
2.1	InfluxDB 数据源方案.....	9
2.2	Prometheus 数据源方案.....	9
3	实验部署环境	10
3.1	实践环境.....	10
3.2	管理节点安装 Redhat 7.6 系统与参数调整.....	12
4	InfluxDB 数据源方案实现	14
4.1	安装配置 InfluxDB.....	14
4.1.1	安装 InfluxDB	14
4.1.2	初始化 Influxdb 数据库	15
4.1.3	创建 njmon 库.....	16
4.1.4	添加管理员用户	16
4.1.5	启用认证	17
4.2	配置 nimon 监控数据入库.....	18

4.2.1	K1 Power Linux 平台监控采集入库	18
4.2.2	K1 Power/AIX 平台监控采集入库	22
4.3	安装部署 Grafana.....	24
4.3.1	安装并启动 Grafana	24
4.3.2	安装 InfluxDB 插件	25
4.3.3	配置 InfluxDB 数据源	26
4.3.4	添加 Dashboard.....	29
4.4	验证监控.....	31
5	Prometheus 数据源方案实现	32
5.1	安装部署 Prometheus.....	32
5.1.1	安装 Prometheus	32
5.2	安装部署 telegraf.....	34
5.2.1	安装 telegraf.....	34
5.2.2	配置 telegraf.....	35
5.2.3	启动 telegraf.....	35
5.2.4	验证 prometheus 中 telegraf target 状态	36
5.3	安装配置 nimon 监控数据入库.....	36
5.3.1	验证 nimon 监控数据.....	37
5.4	安装部署 Grafana.....	37
5.4.1	安装并启动 Grafana	37
5.4.2	添加 prometheus-demo(telegraf)数据源.....	38
5.4.3	添加 Dashboard.....	38
5.4.4	Grafana Alert 功能	40
6	附录	45

1 概念简介

1.1 nmon

nmon 是 **N**igel's **p**erformance **M**onitor 的缩写，它是 IBM 员工 Nigel Griffiths 开发的一款计算机性能监控和数据收集工具。这个工具已有将近 20 年历史，早在 2003 年，nmon 工具已经成为用于 AIX 4.1.5、4.2.0、4.3.2 和 4.3.3 的成熟监控工具，后来不断为 AIX 5.1,5.2,5.3,6.1,7.1,7.2 迭代新的版本，并逐步拓展 Power/AIX 之外的 Linux 系统和其它平台。现在它可以监控 AIX 和 Linux 系统，支持 Power, X86, Mainframe 乃至 ARM (Raspberry Pi)平台，是一个标准的支持跨多平台、多系统的监控工具。

1.2 njmon,nimon

njmon 是 nmon 的新一代监控工具，它实现了 nmon 的类似功能，性能指标输出为 JSON 格式。可以将 njmon 性能统计监控数据输出直接 push 到 InfluxDB 数据库中，Grafana 配置 InfluxDB 数据源实时读取监控数据，来实现性能指标实时可视化监控。

nimon 功能与 njmon 相似，所不同的是它将性能指标输出为 InfluxDB Line 协议格式。

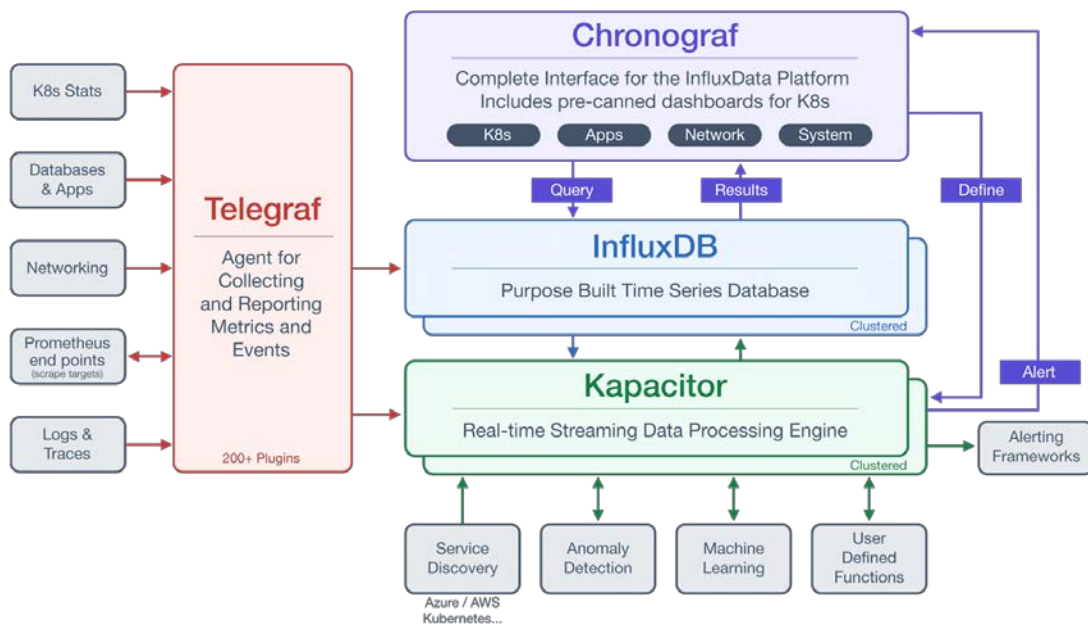
1.3 InfluxDB

InfluxDB 是由 InfluxData 开发的开源时序型数据库。它使用 go 语言编写，致力于高效地存储和查询时序型数据。InfluxDB 被广泛应用于存储系统性能指标监控数据。

InfluxDB 语法是类 SQL 的，增删改查与 MySQL 基本相同。InfluxDB 默认端口为 8086，它的 measurement 对应关系型数据库中的 table 概念。

1.4 Telegraf

Telegraf 是一个用 Golang 写的开源数据收集 Agent，基于插件驱动。Telegraf 是 influxdata 公司的时间序列平台 TICK 技术栈中的 “T”，主要用于收集时间序列型数据，比如服务器 CPU 指标、内存指标、各种 IoT 设备产生的数据等等。



Telegraf 工作原理：定时去执行输入插件收集数据，数据经过处理插件和聚合插件，批量输出到数据存储。

• 数据指标 (Metrics)

- 指标名 (Measurement name)：指标描述和命名。
- 标签集合 (Tags)：Key/Value 键值对，可以类比为关系型数据库的键值，常用于快速索引和唯一标识。标签在设计的时候，尽量避免各种数值型，尽量使用有限集合。

- 字段集合 (Fields) : Key/Value 键值对, 包含指标描述的数据类型和值。
- 时间戳 (Timestamp) : 此条指标数据的时间戳。

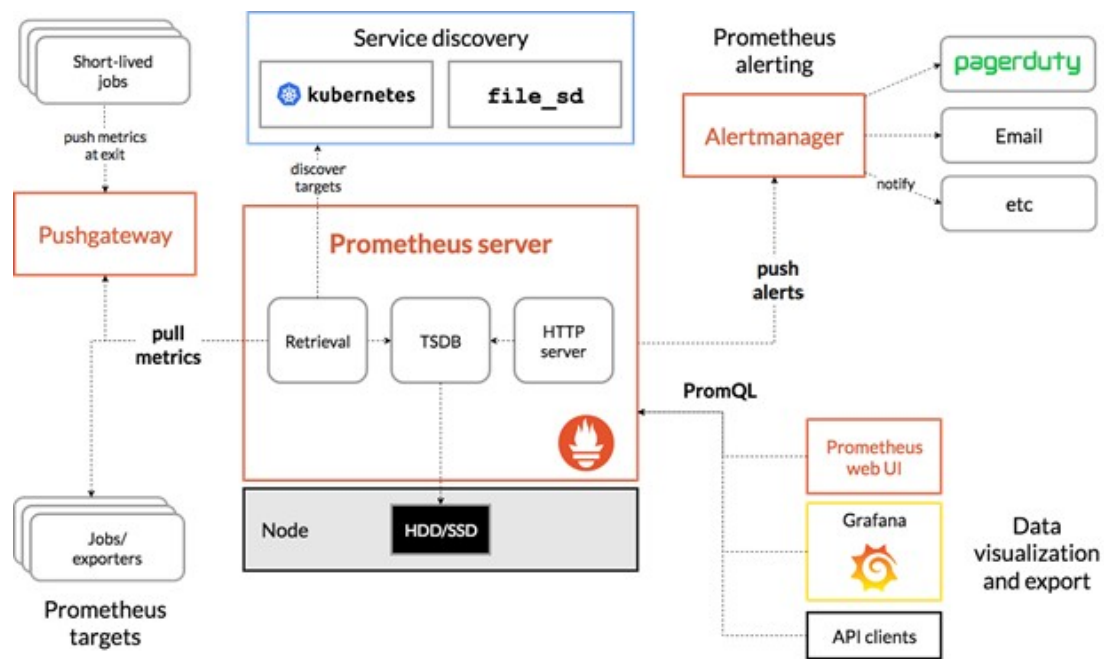
- **插件 (Plugins)** Telegraf 有四种类型的插件:

- 输入插件 (Inputs) : 收集各种时间序列性指标, 包含各种系统信息和应用信息的插件。
- 处理插件 (Process) : 当收集到的指标数据流要进行一些简单处理时, 比如给所有指标添加、删除、修改一个 Tag。只是针对当前的指标数据进行。
- 聚合插件 (Aggregate) : 聚合插件有别于处理插件, 就在于它要处理的对象是某段时间流经该插件的所有数据 (所以, 每个聚合插件都有一个 period 设置, 只会处理 now()-period 时间段内的数据), 比如取最大值、最小值、平均值等操作。
- 输出插件 (Outputs) : 收集到的数据, 经过处理和聚合后, 输出到数据存储系统, 可以是各种地方, 如: 文件、InfluxDB、各种消息队列服务等等。

本文示例中我们将用到 Inputs 和 Outputs 插件。

1.5 Prometheus

Prometheus (普罗米修斯) 是一套开源的监控&报警&时间序列数据库的组合, 是由 SoundCloud 公司开发的。目前 Redhat Openshift 与 Kubernetes 都在用 Prometheus, 越来越多公司和组织接受采用 Prometheus, 开源社会也十分活跃。Prometheus 及相关生态系统组件的整体架构如下图:



Prometheus daemon 负责定时间隔去目标上 Pull 抓取监控指标(metrics) 数据，由各被抓取目标系统或应用对外暴露 http 服务接口(exporter)给 Prometheus。

Prometheus：支持通过配置文件、文本文件、zookeeper、Consul、DNS SRV lookup 等方式指定抓取目标。支持很多方式的图表可视化，例如十分精美的 Grafana，自带的 Promdash，以及自身提供的模版引擎等等，还可以提供 HTTP API 的查询方式，自定义输出。

Alertmanager：是独立于 Prometheus 的组件，支持 Prometheus 的查询语句，提供十分灵活的报警方式。

PushGateway：这个组件支持 Client 主动推送 metrics 到 PushGateway，Prometheus 则定时去 Gateway 上 Pull 抓取数据。

它有点类似于 statsd，但 statsd 是直接发送给服务器端，而 Prometheus 主要是 daemon 进程主动去抓取。

Prometheus 组件大多数用 Go 编写的，它们可以轻松地构建和部署为静态二进制文件。<https://prometheus.io> 上有完整的文档，示例指南。

1.6 Grafana

Grafana 是一个跨平台的开源的度量分析和可视化工具，可以通过将采集的数据查询然后可视化的展示，并及时通知。它主要特点如下：

- 1、展示方式：快速灵活的仪表盘图表，面板插件有许多不同方式的可视化指标和日志，官方库中具有丰富的仪表盘插件，比如热图、折线图、图表等多种展示方式；
- 2、数据源，可以是 Prometheus, Graphite, InfluxDB, OpenTSDB, Elasticsearch, CloudWatch 和 KairosDB 等；
- 3、告警通知：以可视方式定义最重要指标的警报规则，Grafana 将不断计算并发送通知，在数据达到阈值时通过 Slack、PagerDuty 等获得通知；
- 4、混合展示：在同一图表中混合使用不同的数据源，可以基于每个查询指定数据源，甚至自定义数据源；
- 5、注释：使用来自不同数据源的丰富事件注释图表，将鼠标悬停在事件上会显示完整的事件元数据和标记；
- 6、过滤器：Ad-hoc 过滤器允许动态创建新的键/值过滤器，这些过滤器会自动应用于使用该数据源的所有查询。

2 可视化监控方案

2.1 InfluxDB 数据源方案

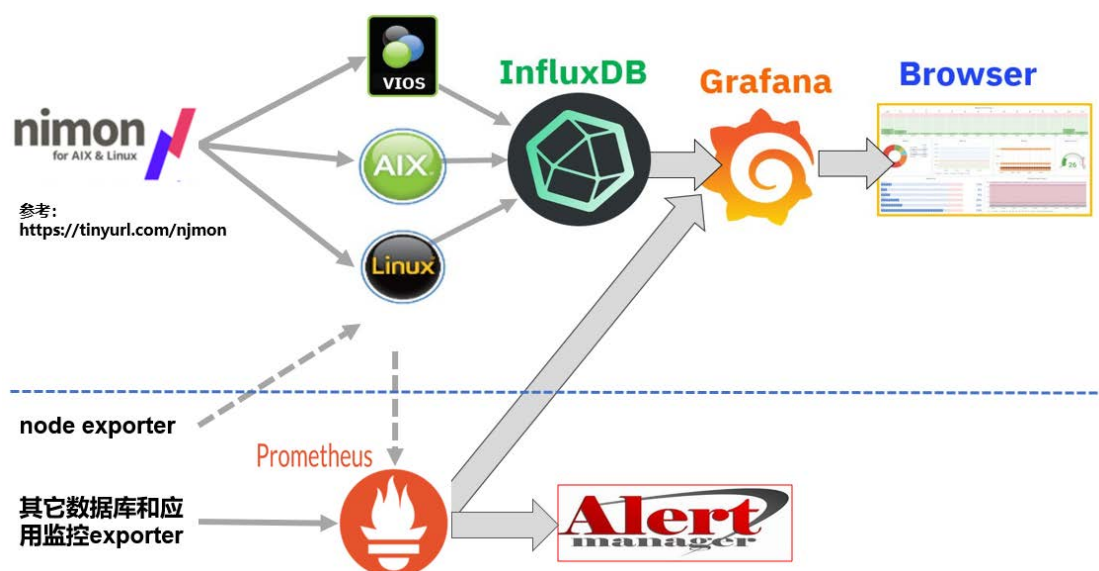
采用 nimon 采集 AIX, VIOS, Linux 性能指标数据, 输出直接 push 到 InfluxDB 数据库中,

Grafana 配置 InfluxDB 数据源实时读取监控数据, 来实现性能指标实时可视化监控。

其它各个数据库和应用, 启动监控 exporter, pull 到 Prometheus 中, Grafana 读

Prometheus 数据源做可视化展示。

如下图示:

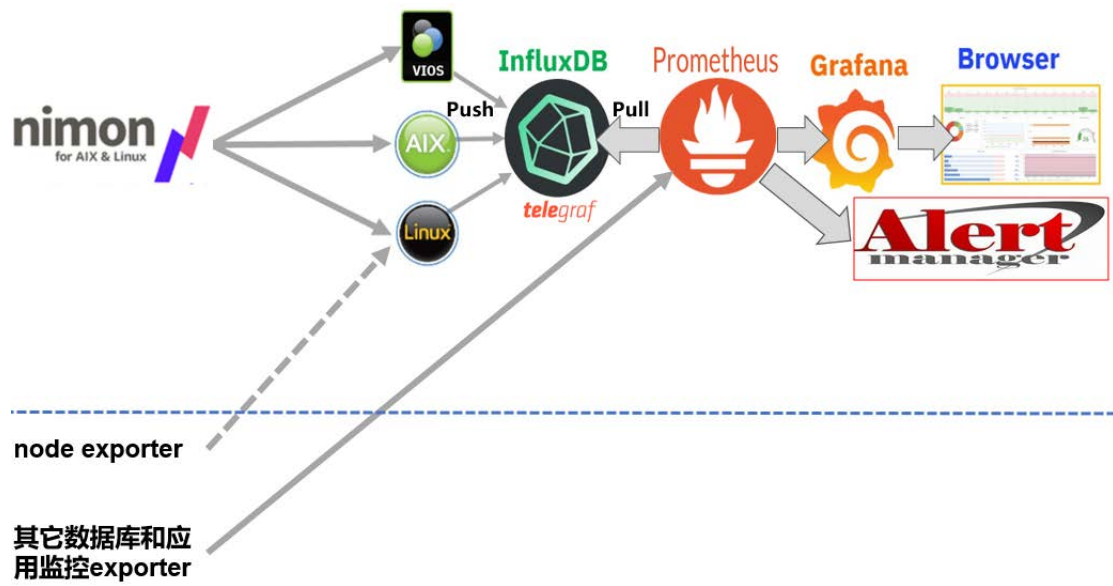


2.2 Prometheus 数据源方案

目前 Prometheus 较为流行, 很多上层数据库和应用都围绕 Prometheus 开发了监控 exporter agent, 此方案将可视化监控数据源统一为 Prometheus。

我们可以将 nimon 采集 AIX, VIOS, Linux 性能指标数据 push 到 Influxdata (含 InfluxDB+Telegraf) 中的监控数据, Prometheus 经 Telegraf pull 监控数据。

然后 Grafana 配置 Prometheus 数据源实时读取监控数据，来实现性能指标实时可视化监控。并通过 Prometheus 的 Alert Manager 组件实现告警。如下图示：



3 实验部署环境

3.1 实践环境

环境项	软/硬件名称	描述
管理节点	K1 Power Linux	配置：CPU 4 核 32 线程（SMT-8） 内存 512GB 1*300GB Disk 1*10Gb 网卡

		<p>注：考虑到该节点要存放 InfluxDB 数据库，配足数据库内存配置大一些（建议$\geq 128\text{GB}$）和硬盘容量根据实际保存性能指标监控节点数和保存数据周期来定（建议$\geq 200\text{GB}$）。</p> <p>系统：RedHat 7.6</p> <p>内核 4.14.0-115.8.1.el7a.ppc64le</p>
监控节点 AIX	K1 Power AIX 7.2	<p>配置：CPU 1 核 8 线程（SMT-8）</p> <p>内存 256GB</p> <p>1*100GB Disk</p> <p>1*10Gb 网卡</p> <p>系统：AIX 7200-05-01-2038</p>
监 控 节 点 Linux	K1 Power Linux RedHat 7.6	<p>配置：CPU 4 核 32 线程（SMT-8）</p> <p>内存 512GB</p> <p>1*100GB Disk</p> <p>1*10Gb 网卡</p> <p>系统：RedHat 7.6</p> <p>内核 4.14.0-115.8.1.el7a.ppc64le</p>
编译工具	devtoolset 7 for linux	RedHat devtoolset 7, gcc version 7.3.1 20180303
influxdb	1.8.4	influxdb 1.8.4 for ppc64le(Redhat 7)
telegraf	1.18.1	telegraf 1.18.1 for ppc64le
Prometheus	Prometheus	Prometheus v2.24.1 for ppc64le

	v2.24.1	
njmon/nimon	Linux v71 AIX v73	njmon_linux_binaries_v71, njmon_aix_binaries_v73
Grafana	Grafana v7.3.7	

3.2 管理节点安装 Redhat 7.6 系统与参数调整

系统安装过程略，版本与内核如下：

```
# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.6 (Maipo)
# cat /proc/version
Linux version 4.14.0-115.8.1.el7a.ppc64le (mockbuild@ppc64le-02.bsys.centos.org) (gcc version 4.8.5
20150623 (Red Hat 4.8.5-36) (GCC)) #1 SMP Wed Jun 5 15:02:21 UTC 2019
```

停用防火墙，tuned 自动优化

```
# systemctl stop tuned; systemctl disable tuned
# systemctl stop firewalld; systemctl disable firewalld
# systemctl stop iptables; systemctl disable iptables
# systemctl stop ip6tables; systemctl disable ip6tables
```

停用 selinux

```
# setenforce 0
# vi /etc/selinux/config
...
SELINUX=disabled
...
```

系统参数

```
# cat /etc/sysctl.conf
kernel.sched_migration_cost = 500000
kernel.sysrq = 0
kernel.core_uses_pid = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
```

kernel.shmmax = 549755813888
kernel.shmall = 6536870912
kernel.shmmni = 4096
kernel.sem = 1024 320000 1000 1280
kernel.sched_latency_ns=80000000
kernel.sched_wakeup_granularity_ns=40000000
kernel.sched_min_granularity_ns=40000000

#VM

vm.min_free_kbytes = 3355443
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10
vm.overcommit_memory = 0
vm.overcommit_ratio = 50
vm.zone_reclaim_mode = 1
vm.swappiness=0
vm.nr_hugepages=1
#vm.hugetlb_shm_group=26

#FS

fs.file-max = 327680
fs.aio-max-nr = 3145728

#NET

net.core.wmem_default = 16777216
net.core.wmem_max = 16777216
net.core.rmem_default = 16777216
net.core.rmem_max = 16777216
net.core.optmem_max = 524288
net.core.netdev_max_backlog = 250000
net.core.somaxconn=81920
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_mem = 16777216 16777216 16777216
net.ipv4.tcp_rmem = 16777216 16777216 16777216
net.ipv4.tcp_wmem = 16777216 16777216 16777216
net.ipv4.tcp_low_latency = 1
net.ipv4.tcp_max_syn_backlog=81920
net.ipv4.tcp_sack = 1
net.ipv4.neigh.default.gc_thresh1 = 2000
net.ipv4.neigh.default.gc_thresh2 = 2000
net.ipv4.neigh.default.gc_thresh3 = 2000
net.ipv4.ip_local_port_range = 1024 65000

```
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.all.arp_filter = 1
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_fin_timeout = 10
net.ipv4.tcp_retries2 = 2
net.ipv4.tcp_fack=1
net.ipv4.tcp_keepalive_intvl=15
net.ipv4.tcp_keepalive_probes=3
net.ipv4.tcp_keepalive_time=120
```

Limit 限制增加如下行

```
# cat /etc/security/limits.conf
*      soft    fsize unlimited
*      hard    fsize unlimited
*      soft    memlock unlimited
*      hard    memlock unlimited
*      soft    nofile 819200
*      hard    nofile 819200
*      soft    nproc 655360
*      hard    nproc 655360
*      soft    stack unlimited
*      hard    stack unlimited
```

4 InfluxDB 数据源方案实现

4.1 安装配置 InfluxDB

4.1.1 安装 InfluxDB

influxdb ppc64le 版本下载链接: <https://www.power-devops.com/influxdb>

直接 rpm 安装即可:

```
# rpm -Uvh influxdb-1.8.4-1.el7.ppc64le.rpm
```

```
Preparing... ##### [100%]
Updating / installing...
 1:influxdb-1.8.4-1.el7 ##### [100%]
```

4.1.2 初始化 Influxdb 数据库

4.1.2.1 启动并验证 influxdb 服务

```
# systemctl start influxdb
```

```
# systemctl status -l influxdb
```

```
influxdb.service - InfluxDB is an open-source, distributed, time series database
Loaded: loaded (/usr/lib/systemd/system/influxdb.service; disabled; vendor preset: disabled)
Active: active (running) since Thu 2021-03-18 13:30:49 CST; 6s ago
Docs: https://docs.influxdata.com/influxdb/
Main PID: 20695 (influxd)
CGroup: /system.slice/influxdb.service
        └─20695 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```

```
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715510Z lvl=info msg="Starting
precreation service" log_id=0Sxxt5xG000 service=shard-precreation check_interval=10m
advance_period=30m
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715516Z lvl=info msg="Starting
snapshot service" log_id=0Sxxt5xG000 service=snapshot
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715524Z lvl=info msg="Starting
continuous query service" log_id=0Sxxt5xG000 service=continuous_querier
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715530Z lvl=info msg="Starting HTTP
service" log_id=0Sxxt5xG000 service=httpd authentication=false
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715534Z lvl=info msg="opened HTTP
access log" log_id=0Sxxt5xG000 service=httpd path=stderr
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715531Z lvl=info msg="Storing
statistics" log_id=0Sxxt5xG000 service=monitor db_instance=_internal db_rp=monitor interval=10s
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715575Z lvl=info msg="Listening on
HTTP" log_id=0Sxxt5xG000 service=httpd addr=[::]:8086 https=false
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715624Z lvl=info msg="Starting
retention policy enforcement service" log_id=0Sxxt5xG000 service=retention check_interval=30m
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715678Z lvl=info msg="Sending
usage statistics to usage.influxdata.com" log_id=0Sxxt5xG000
Mar 18 13:30:49 ansiblemaster influxd[20695]: ts=2021-03-18T05:30:49.715695Z lvl=info msg="Listening for
signals" log_id=0Sxxt5xG000
```

```
# influx -version
```

```
InfluxDB shell version: 1.8.4
```

4.1.3 创建 njmon 库

```
# influx
```

```
Connected to http://localhost:8086 version 1.8.4
```

```
InfluxDB shell version: 1.8.4
```

```
> create database njmon
```

```
> show databases
```

```
name: databases
```

```
name
```

```
----
```

```
_internal
```

```
njmon
```

```
> exit
```

4.1.4 添加管理员用户

```
# influx
```

```
Connected to http://localhost:8086 version 1.8.4
```

```
InfluxDB shell version: 1.8.4
```

```
> show users
```

```
user admin
```

> **create user "admin" with password 'admin' with all privileges**

> **show users**

user admin

admin true

> **exit**

influx -host localhost -port 8086 -username admin -password admin

Connected to http://localhost:8086 version 1.8.4

InfluxDB shell version: 1.8.4

> show databases

name: databases

name

_internal

njmon

4.1.5 启用认证

设置 **auth-enabled = true**

vi /etc/influxdb/influxdb.conf

[http]

Determines whether HTTP endpoint is enabled.

enabled = true

Determines whether the Flux query endpoint is enabled.

flux-enabled = false

Determines whether the Flux query logging is enabled.

```
# flux-log-enabled = false
# The bind address used by the HTTP service.
# bind-address = ":8086"
# Determines whether user authentication is enabled over HTTP/HTTPS.
auth-enabled = true
```

重启生效

4.2 配置 nimon 监控数据入库

分别讲述 K1 Power/AIX、K1 Power Linux 平台下采集性能指标监控数据入库。我们采用 nimon 采集并直接入 influxdb 数据库方式。

4.2.1 K1 Power Linux 平台监控采集入库

4.2.1.1 安装 njmon 和 nimon

从 <https://sourceforge.net/projects/nmon/files/> 下载 njmon_linux_binaries_v71.zip,

解压并安装 njmon & nimon:

```
# unzip njmon_linux_binaries_v71.zip
```

```
# cd njmon_linux_binaries
```

```
# ls
```

nimon.1	nimon_SLES12_ppc64le_v71
nimon_Ubuntu20_ppc64le_v71	njmon_RHEL7_x86_64_v71
njmon_SLES15_ppc64le_v71	
nimon_RHEL7_ppc64le_v71	nimon_SLES12_x86_64_v71
nimon_Ubuntu20_x86_64_v71	njmon_RHEL8_ppc64le_v71

njmon_Ubuntu18_ppc64le_v71

nimon_RHEL7_x86_64_v71 nimon_SLES15_ppc64le_v71 **ninstall**

njmon_RHEL8_x86_64_v71 njmon_Ubuntu18_x86_64_v71

nimon_RHEL8_ppc64le_v71 nimon_Ubuntu18_ppc64le_v71 njmon.1

njmon_SLES12_ppc64le_v71 njmon_Ubuntu20_ppc64le_v71

nimon_RHEL8_x86_64_v71 nimon_Ubuntu18_x86_64_v71

njmon_RHEL7_ppc64le_v71 njmon_SLES12_x86_64_v71

njmon_Ubuntu20_x86_64_v71

sh -x ninstall

type njmon

njmon is /usr/local/bin/njmon

type nimon

nimon is /usr/local/bin/nimon

4.2.1.2 验证 nimon 可用性

nimon -s 1 -c 1 |head -2

4755

timestamp,host=ansiblemaster,os=RHEL,architecture=ppc64le,serial_no=78264A8,mtm=9080-M9S

datetime="2021-03-19T15:07:57",UTC="2021-03-

19T07:07:57",snapshot_seconds=1i,snapshot_maxloops=1i,snapshot_loop=0i

启动 nimon 短时间监控，验证能否正常入库：

nimon -s 2 -c 2 -k -P -b -r -n -H -i 172.16.xxx.xxx -p 8086 -x njmon -y admin -z admin

-i 172.16.xxx.xxx : influxdb 主机 IP

-p 8086 : influxdb 数据库服务端口

-x njmon : influxdb 数据库 database 名字

-y admin -z admin : influxdb 数据库用户和密码

进入 influxdb 数据库查询数据验证:

```
# influx -username admin -password admin -database njmon -host 172.16.xxx.xxx -port 8086
```

Connected to http://172.16.xxx.xxx:8086 version 1.8.4

InfluxDB shell version: 1.8.4

```
> show databases;
```

name: databases

name

_internal

njmon

```
> use njmon
```

Using database njmon

```
> show measurements;
```

name: measurements

name

cpu_total

cpuinfo

cpuinfo_power

cpus

disks

filesystems

identity

lscpu

networks

os_release

ppc64_lparcfg

proc_meminfo

proc_version

proc_vmstat

processes

stat_counters

sys_dev_sys_cpu

timestamp

uptime

```
> select * from uptime;
```

name: uptime

time architecture days host

hours minutes mtm

os serial_no

users

```

-----
1616136876557256396 ppc64le      2    ansiblemaster 23    55    9080-M9S RHEL
78264A8 7
1616136878595584599 ppc64le      2    ansiblemaster 23    55    9080-M9S RHEL
78264A8 7
> select * from lscpu;
name: lscpu
time          architecture architecture_1 byte_order    cores_per_socket cpus host
model         model_name                               mtm          numa_nodes
online_cpu_list os  serial_no sockets threads_per_core
-----
-----
1616136876557256396 ppc64le      ppc64le      Little Endian 1              32
ansiblemaster 1.2 (pvr 004e 2102) POWER9 (architected), altivec supported 9080-M9S 2
0-31          RHEL 78264A8 4        8
1616136878595584599 ppc64le      ppc64le      Little Endian 1              32
ansiblemaster 1.2 (pvr 004e 2102) POWER9 (architected), altivec supported 9080-M9S 2
0-31          RHEL 78264A8 4        8
> select * from os_release;
name: os_release
time          architecture host          mtm      name          os
pretty_name          serial_no version    version_id
-----
-----
1616136876557256396 ppc64le      ansiblemaster 9080-M9S Red Hat Enterprise Linux Server
RHEL Red Hat Enterprise Linux Server 7.6 (Maipo) 78264A8 7.6 (Maipo) 7.6
1616136878595584599 ppc64le      ansiblemaster 9080-M9S Red Hat Enterprise Linux Server
RHEL Red Hat Enterprise Linux Server 7.6 (Maipo) 78264A8 7.6 (Maipo) 7.6

```

4.2.1.3 启用 nimon crontab 持续监控

```
# crontab -e
```

```
# crontab -l
```

```

1 0 * * * /usr/local/bin/nimon -s 5 -c 17280 -k -P -b -r -n -H -i 172.16. xxx.xxx -p 8086 -x njmon
-y admin -z admin

```

4.2.2 K1 Power/AIX 平台监控采集入库

4.2.2.1 安装 njmon 和 nimon

从 <https://sourceforge.net/projects/nmon/files/> 下载 njmon_aix_binaries_v73.zip, 解

压并安装 njmon & nimon:

```
# unzip njmon_aix_binaries_v73.zip
```

```
# ls n*
```

nimon.1	nimon_aix724_v73	njmon.1
njmon_aix724_v73		
nimon_aix61_v73	nimon_vios2_v73	njmon_aix61_v73
njmon_aix_binaries_v73.zip		
nimon_aix71_v73	nimon_vios3_v73	njmon_aix71_v73
njmon_vios2_v73		
nimon_aix722_v73	ninstall	njmon_aix722_v73
njmon_vios3_v73		

```
# sh -x ninstall
```

```
# export PATH=$PATH:/usr/lbin
```

```
# type njmon
```

```
njmon is /usr/lbin/njmon
```

```
# type nimon
```

```
nimon is /usr/lbin/nimon
```

4.2.2.2 验证 nimon 可用性

```
# nimon -s 1 -c 1 |head -2
```

```
20119972
```

```
identity,host=racnode1,os=AIX,architecture=POWER9,serial_no=7874C50,mtm=IBM-9009-42A
```

```
hostname="racnode1",fullhostname="racnode1",njmon_command="nimon -s 1 -c
```

```
1",njmon_version="nimon4AIX724-v73-16/02/2021",username="root",userid=0i,cookie="0xdeadbeef"
```

启动 nimon 短时间监控，验证能否正常入库：

```
# nimon -s 2 -c 2 -k -P -b -r -n -H -i 172.16.xxx.xxx -p 8086 -x njmon -y admin -z admin
```

-i 172.16.xxx.xxx : influxdb 主机 IP

-p 8086 : influxdb 数据库服务端口

-x njmon : influxdb 数据库 database 名字

-y admin -z admin : influxdb 数据库用户和密码

进入 influxdb 数据库查询数据验证：

```
# influx -username admin -password admin -database njmon -host 172.16.xxx.xxx -port 8086
```

Connected to http://172.16.xxx.xxx:8086 version 1.8.4

InfluxDB shell version: 1.8.4

```
> show databases;
```

name: databases

name

_internal

njmon

```
> use njmon
```

Using database njmon

```
> show measurements;
```

name: measurements

name

cpu_total

cpuinfo

cpuinfo_power

cpus

disks

filesystems

identity

lscpu

networks

os_release

ppc64_lparcfg

proc_meminfo

proc_version

proc_vmstat

processes

stat_counters

sys_dev_sys_cpu

timestamp

uptime

> select * from uptime where host='racnode1';

name: uptime

time	architecture	days	host	hours	minutes	mtm	os	serial_no
users								
----	-----	-----				--	-----	
1616552859150068103	POWER9		19	racnode1	20	57	IBM-9009-42A	AIX
7874C50 1								
1616552861586267045	POWER9		19	racnode1	20	57	IBM-9009-42A	AIX
7874C50 1								

4.2.2.3 启用 nimon crontab 持续监控

crontab -e

crontab -l

1 0 * * * /usr/sbin/nimon -s 5 -c 17280 -k -P -b -r -n -H -i 172.16.xxx.xxx -p 8086 -x njmon -y
admin -z admin

4.3 安装部署 Grafana

4.3.1 安装并启动 Grafana

下载 grafana

wget -c <https://downloads.power-devops.com/grafana-7.3.7-1.el7.ppc64le.rpm>

安装 grafana

rpm -Uvh grafana-7.3.7-1.el7.ppc64le.rpm

启动 grafana 服务

systemctl daemon-reload

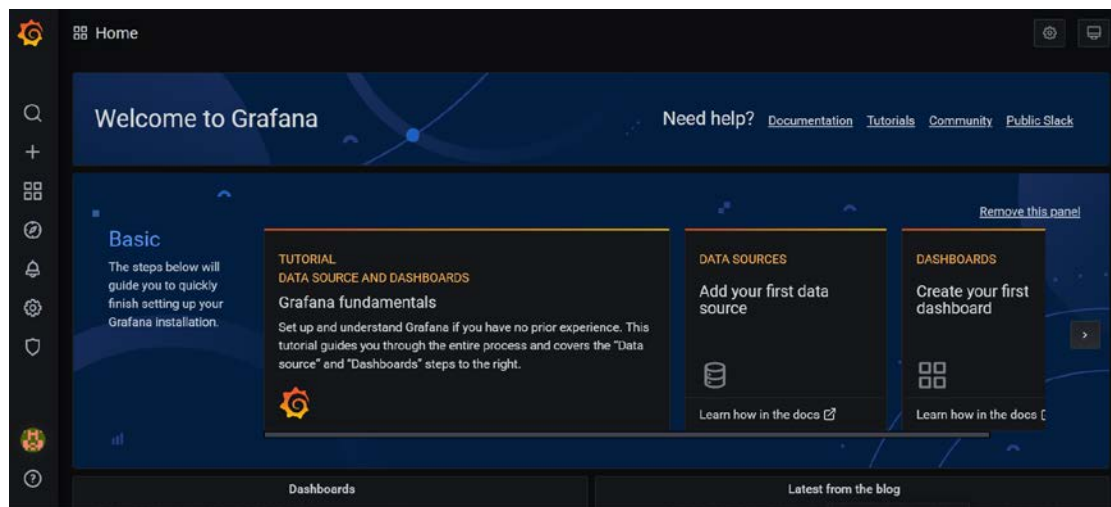
systemctl start grafana-server

systemctl status -l grafana-server

登录 grafana, 缺省用户/密码 admin/admin

<http://172.16.xxx.xxx:3000>

登录后界面如下:



4.3.2 安装 InfluxDB 插件

```
# grafana-cli plugins install grafana-clock-panel
```

installing grafana-clock-panel @ 1.0.3

from: <https://grafana.com/api/plugins/grafana-clock-panel/versions/1.0.3/download>

into: /var/lib/grafana/plugins

✓ Installed grafana-clock-panel successfully

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

```
# grafana-cli plugins install grafana-piechart-panel
```

installing grafana-piechart-panel @ 1.6.1

from: <https://grafana.com/api/plugins/grafana-piechart-panel/versions/1.6.1/download>

into: /var/lib/grafana/plugins

✓ Installed grafana-piechart-panel successfully

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

```
# grafana-cli plugins install grafana-influxdb-flux-datasource
```

installing grafana-influxdb-flux-datasource @ 5.4.1

from: <https://grafana.com/api/plugins/grafana-influxdb-flux-datasource/versions/5.4.1/download>

into: /var/lib/grafana/plugins

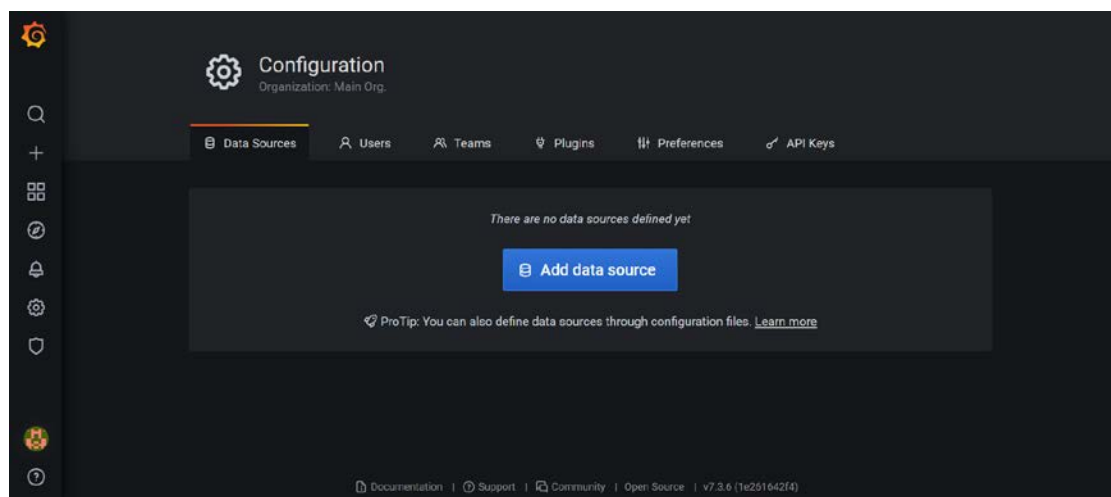
✓ Installed grafana-influxdb-flux-datasource successfully

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

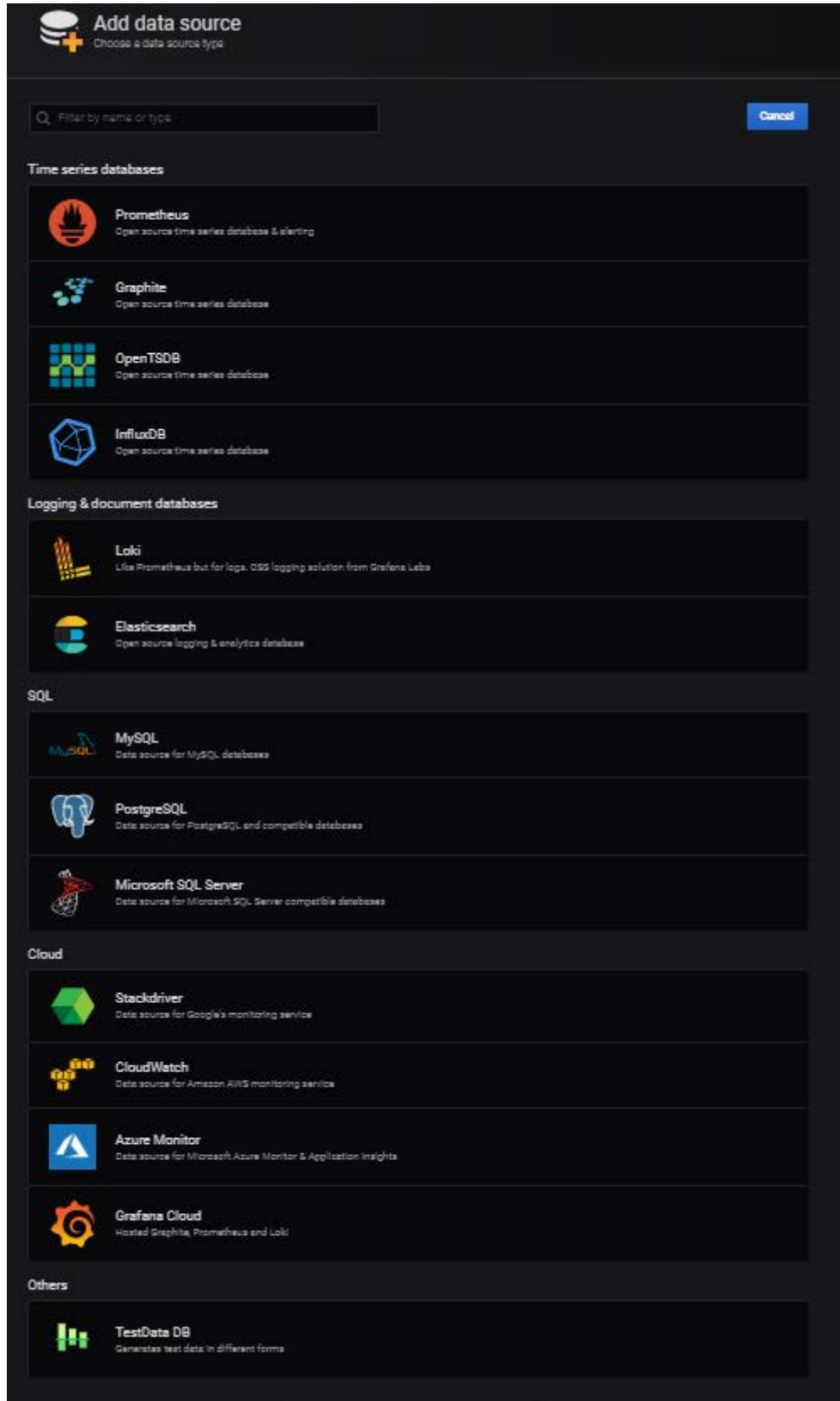
```
# systemctl restart grafana-server
```

4.3.3 配置 InfluxDB 数据源

首先配置 InfluxDB 数据源，Configuraion -> Add data soure




选择 InfluxDB:



输入 InfluxDB 数据源，URL: <http://172.16.xxx.xxx:8086>, Access 方式缺省为

Server(default), database: njmon, User: admin, Password: admin, 然后点 Save & Test,

保存并验证:

 **Data Sources / InfluxDB**
Type: InfluxDB

Settings

Name

InfluxDB

Default

☐

HTTP

URL

http://172.16.102.111:8086

Access

Server (Default)

Help

Whitelisted Cookies

Add Name

Auth

Basic Auth

☐

With Credentials

☐

TLS Client Auth

☐

With CA Cert

☐

Skip TLS Verify

☐

Forward OAuth Identity

☐

InfluxDB Details

Database

njmon

User

admin

Password

HTTP Method

GET

Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".. "database" LIMIT 10`

To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.


Min time interval

10s

Save & Test

Delete

Back

 **Data Sources / InfluxDB**
Type: InfluxDB

Settings

Name

InfluxDB

Default

☐

HTTP

URL

http://172.16.102.111:8086

Access

Server (Default)

Whitelisted Cookies

Add Name

Help

Auth

Basic Auth

☐

With Credentials

☐

TLS Client Auth

☐

With CA Cert

☐

Skip TLS Verify

☐

Forward OAuth Identity

☐

InfluxDB Details

Database

njimon

User

admin

Password

configured

reset

HTTP Method

GET

Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` OR `SELECT * FROM "_internal".."database" LIMIT 10`
To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Min time interval

10s

✓ Data source is working

提示 Data source is working 表示数据源正常。

4.3.4 添加 Dashboard

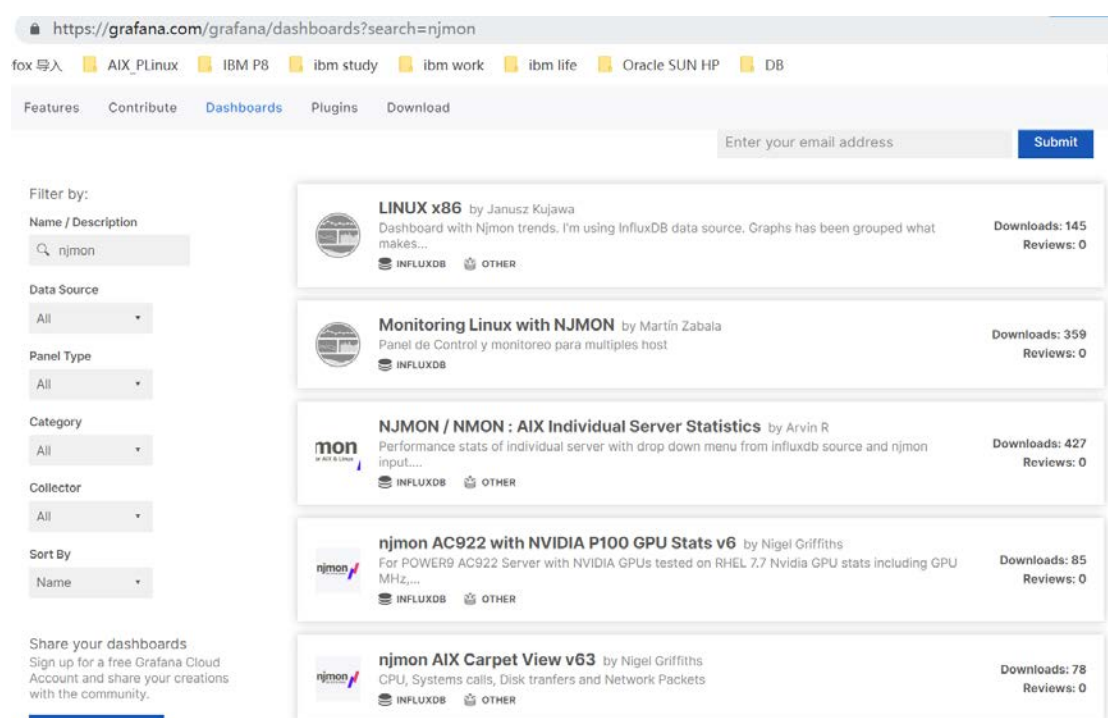
Grafana 提供了非常多 Dashboard 模板，有三种方式 Import Dashboard：

1. 下载 Dashboard 模板 json 文件，然后从本地 Upload json file:
2. 如果 Grafana 服务器能连外网，则直接从 grafana.com 输入 Dashboard 模板编号，

load Dashboard

3. 将 Dashboard 模板 json 文件内容粘贴过来, load Dashboard

在 <https://grafana.com/grafana/dashboards?search=njmon> 中查找合适的 njmon Dashboard, 并下载:



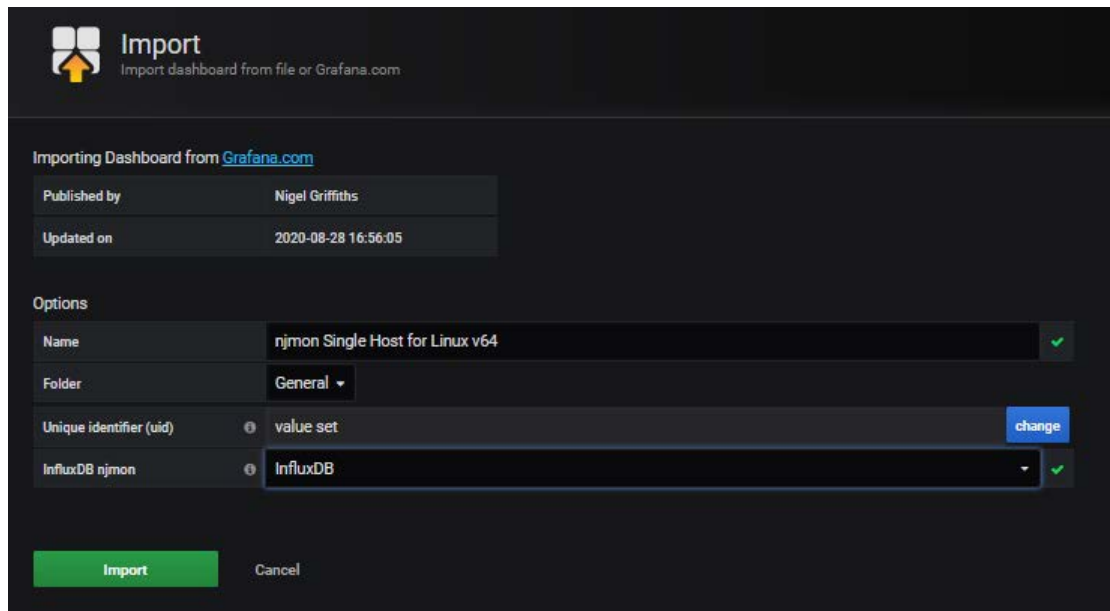
本次使用如下两个 Dashboard, 不能连外网的 Grafana 服务器可以将 Dashboard JSON 下载下来使用:

10844: njmon Single Host for Linux v64

10891: njmon for AIX Simple Six v64

本次实验 Grafana 服务器能连外网, 可以直接用 njmon Single Host for Linux v64

Dashboard 编号 10844 和 njmon for AIX Simple Six v64 编号 10891 分别 import:



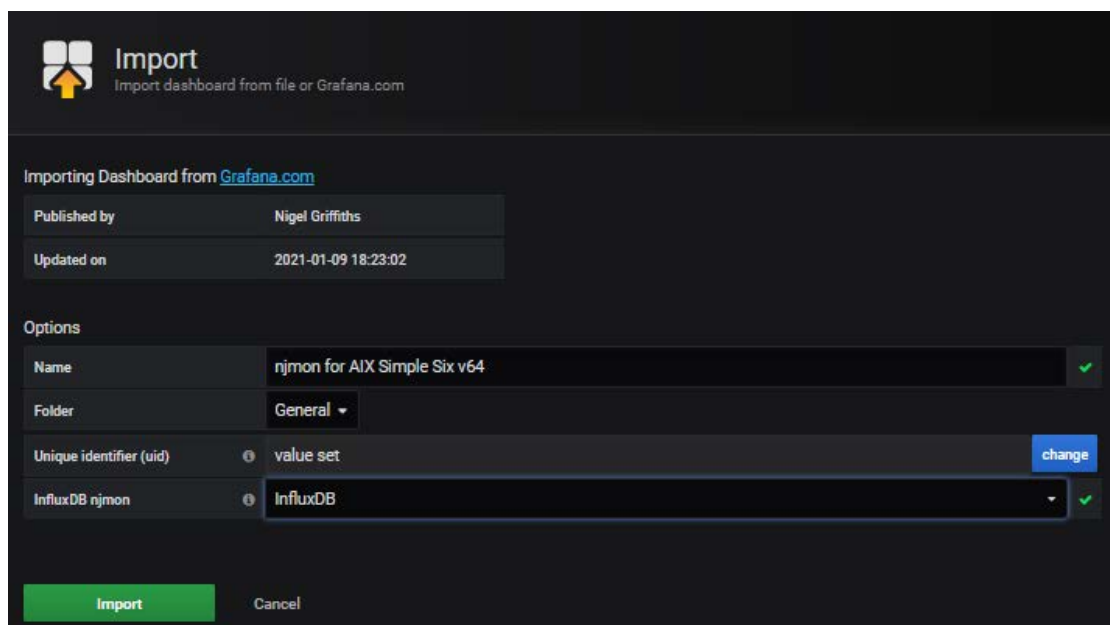
The screenshot shows the Grafana 'Import' interface. At the top, it says 'Import dashboard from file or Grafana.com'. Below this, it indicates the dashboard is being imported from 'Grafana.com'. A table shows the dashboard was published by 'Nigel Griffiths' and updated on '2020-08-28 16:56:05'. The 'Options' section contains four fields: 'Name' is 'njmon Single Host for Linux v64', 'Folder' is 'General', 'Unique identifier (uid)' is 'value set' with a 'change' button, and 'InfluxDB njmon' is 'InfluxDB'. At the bottom, there are 'Import' and 'Cancel' buttons.

Importing Dashboard from Grafana.com	
Published by	Nigel Griffiths
Updated on	2020-08-28 16:56:05

Options

Name	njmon Single Host for Linux v64		
Folder	General		
Unique identifier (uid)	value set	change	
InfluxDB njmon	InfluxDB		

Import Cancel



This screenshot is similar to the first one, showing the Grafana 'Import' interface. The dashboard is 'njmon for AIX Simple Six v64', published by 'Nigel Griffiths' and updated on '2021-01-09 18:23:02'. The 'Options' section shows 'Name' as 'njmon for AIX Simple Six v64', 'Folder' as 'General', 'Unique identifier (uid)' as 'value set' with a 'change' button, and 'InfluxDB njmon' as 'InfluxDB'. The 'Import' and 'Cancel' buttons are at the bottom.

Importing Dashboard from Grafana.com	
Published by	Nigel Griffiths
Updated on	2021-01-09 18:23:02

Options

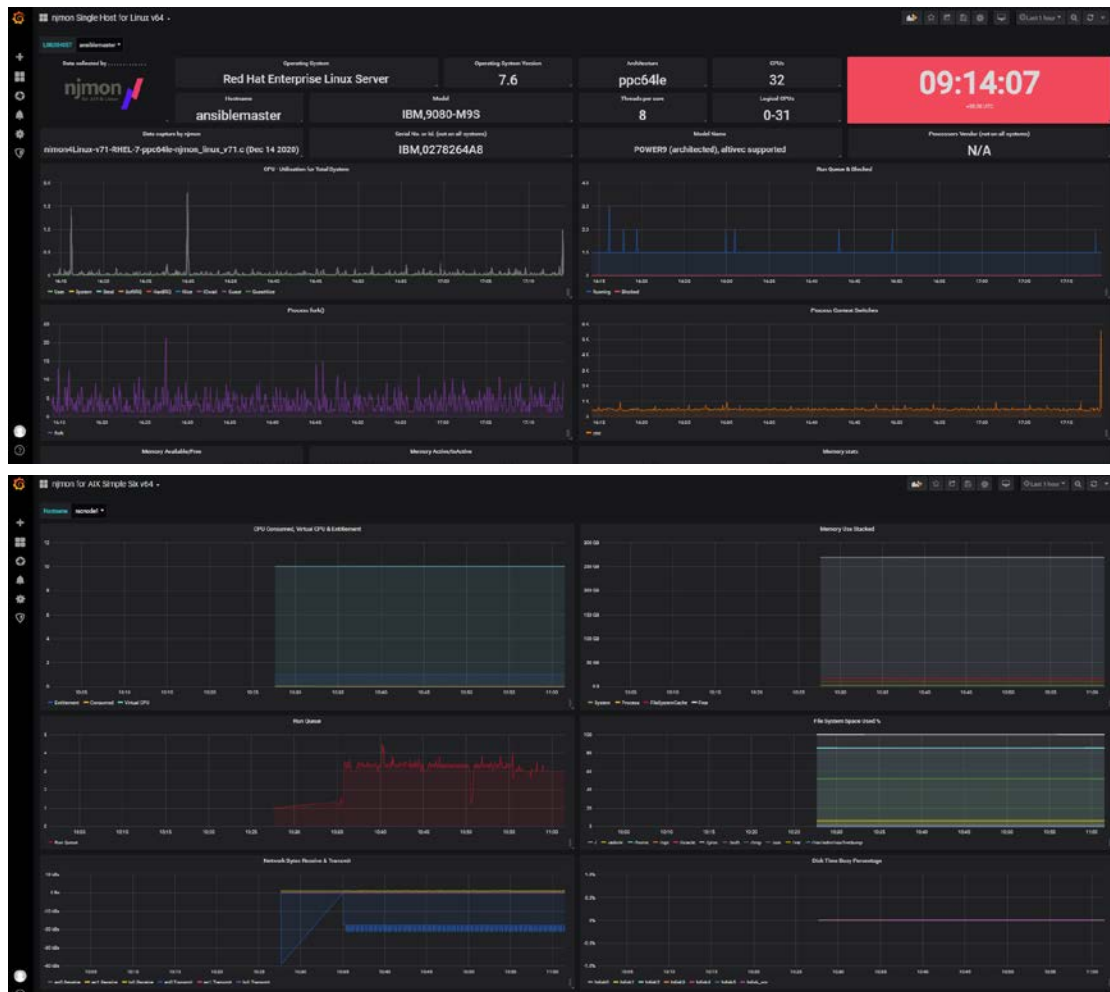
Name	njmon for AIX Simple Six v64		
Folder	General		
Unique identifier (uid)	value set	change	
InfluxDB njmon	InfluxDB		

Import Cancel

点击 Import, 添加完成。

4.4 验证监控

在完成上述安装部署后, 就可以从 Grafana Dashboard: njmon Single Host for Linux v64 和 njmon for AIX Simple Six v64 中分别看到 Linux 和 AIX 节点的监控数据, 如下图所示:



5 Prometheus 数据源方案实现

5.1 安装部署 Prometheus

5.1.1 安装 Prometheus

从 github 下载 Prometheus 及相关组件

```
wget -c https://github.com/prometheus/prometheus/releases/download/v2.24.1/prometheus-2.24.1.linux-ppc64le.tar.gz
```

```
wget -c https://github.com/prometheus/alertmanager/releases/download/v0.21.0/alertmanager-0.21.0.linux-ppc64le.tar.gz
```

```
wget -c https://github.com/prometheus/node_exporter/releases/download/v1.0.1/node_exporter-1.0.1.linux-ppc64le.tar.gz
```



```
# tar xzf prometheus-2.24.1.linux-ppc64le.tar.gz
```

```
# mv prometheus-2.24.1.linux-ppc64le prometheus; cd prometheus
```

更改 prometheus.yml

```
# vi prometheus.yml
```

```
global:
```

```
  scrape_interval:      15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
```

```
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
```

```
  # scrape_timeout is set to the global default (10s).
```

```
# Alertmanager configuration
```

```
alerting:
```

```
  alertmanagers:
```

```
    - static_configs:
```

```
      - targets:
```

```
        # - alertmanager:9093
```

```
# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
```

```
rule_files:
```

```
  # - "first_rules.yml"
```

```
  # - "second_rules.yml"
```

```
# A scrape configuration containing exactly one endpoint to scrape:
```

```
# Here it's Prometheus itself.
```

```
scrape_configs:
```

```
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
```

```
  - job_name: 'prometheus'
```

```
    # metrics_path defaults to '/metrics'
```

```
    # scheme defaults to 'http'.
```

```
    static_configs:
```

```
      - targets: ['172.16.xxx.xxx:9090']
```

```
  - job_name: 'telegraf'
```

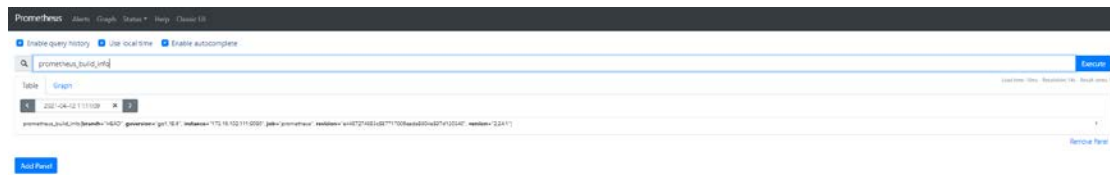
```
    static_configs:
```

```
      - targets: ['172.16.xxx.xxx:8099']
```

启动 prometheus:

```
nohup ./prometheus --config.file=prometheus.yml &
```

浏览器登录 <http://172.16.xxx.xxx:9090>，正常会出现如下窗口：



此时从 Prometheus 界面上 Status -> targets 查看，Prometheus 已启动 (up)：



5.2 安装部署 telegraf

5.2.1 安装 telegraf

Github 上可以下载 telegraf for ppc64le(K1 Power Linux)版本：

<https://github.com/influxdata/telegraf/releases>

power-devops 站点可以下载 AIX 版本：<https://www.power-devops.com/telegraf>

本次我们从 github 上下载 telegraf for ppc64le v1.18.1 版本：

https://dl.influxdata.com/telegraf/releases/telegraf-1.18.1_linux_ppc64le.tar.gz

```
# tar xzvf telegraf-1.18.1_linux_ppc64le.tar.gz
```

```
# cd telegraf-1.18.1/
```

```
# find . -type f
```

```
./usr/bin/telegraf
```

```
./usr/lib/telegraf/scripts/telegraf.service
```

```
./usr/lib/telegraf/scripts/init.sh
```

```
./etc/logrotate.d/telegraf
```

```
./etc/telegraf/telegraf.conf
```

5.2.2 配置 telegraf

```
# cp etc/telegraf/telegraf.conf etc/telegraf/telegraf.conf.bak
```

```
# cat >etc/telegraf/telegraf.conf <<- EOF
```

```
[[outputs.prometheus_client]]
```

```
listen = "172.16.xxx.xxx:8099"
```

```
metric_version = 2
```

```
path = "/metrics"
```

```
expiration_interval = "120s"
```

```
string_as_label = false
```

```
[[inputs.socket_listener]]
```

```
service_address = "tcp://172.16.xxx.xxx:8888"
```

```
data_format = "influx"
```

```
read_buffer_size = "256KiB"
```

```
read_timeout = "2s"
```

```
EOF
```

5.2.3 启动 telegraf

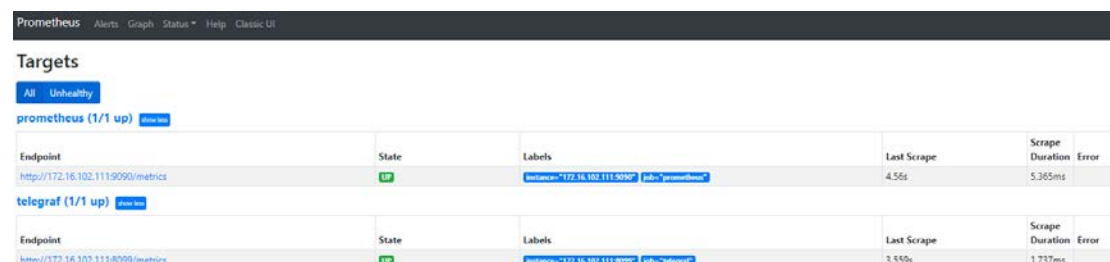
```
# nohup usr/bin/telegraf --config etc/telegraf/telegraf.conf --debug &
```

输出示例：

```
2021-04-12T03:51:23Z !! Starting Telegraf 1.18.1
2021-04-12T03:51:23Z !! Loaded inputs: socket_listener
2021-04-12T03:51:23Z !! Loaded aggregators:
2021-04-12T03:51:23Z !! Loaded processors:
2021-04-12T03:51:23Z !! Loaded outputs: prometheus_client
2021-04-12T03:51:23Z !! Tags enabled: host=ansiblemaster
2021-04-12T03:51:23Z !! [agent] Config: Interval:10s, Quiet:false, Hostname:"ansiblemaster",
Flush Interval:10s
2021-04-12T03:51:23Z D! [agent] Initializing plugins
2021-04-12T03:51:23Z D! [agent] Connecting outputs
2021-04-12T03:51:23Z D! [agent] Attempting connection to [outputs.prometheus_client]
2021-04-12T03:51:23Z !! [outputs.prometheus_client] Listening on
http://172.16.xxx.xxx:8099/metrics
2021-04-12T03:51:23Z D! [agent] Successfully connected to outputs.prometheus_client
2021-04-12T03:51:23Z D! [agent] Starting service inputs
2021-04-12T03:51:23Z !! [inputs.socket_listener] Listening on tcp://172.16.xxx.xxx:8888
2021-04-12T03:51:33Z D! [outputs.prometheus_client] Buffer fullness: 0 / 10000 metrics
```

5.2.4 验证 prometheus 中 telegraf target 状态

刷新 prometheus 验证 telegraf target 显示为 UP 状态:



The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', 'Help', and 'Classic UI'. Below this, the 'Targets' section is active. It shows two target groups: 'prometheus (1/1 up)' and 'telegraf (1/1 up)'. Each group has a table of targets. The 'telegraf' target is shown with a green 'UP' status, a last scrape time of 3.550s, and a scrape duration of 1.737ms.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.16.102.111:9090/metrics	UP	instance="172.16.102.111:9090" job="prometheus"	4.56s	5.365ms	
telegraf (1/1 up)					
http://172.16.102.111:8099/metrics	UP	instance="172.16.102.111:8099" job="telegraf"	3.550s	1.737ms	

5.3 安装配置 nimon 监控数据入库

[安装 nimon on AIX](#) 和 [安装 nimon on Linux ppc64le](#) 的方法与前面相同，不再赘述。

在 AIX 上 crontab 启动 nimon:

```
# crontab -e
# crontab -l
12 4 * * * /usr/lbin/nimon -s 5 -c 17280 -k -P -b -r -n -H -w -i 172.16.xxx.xxx -p 8888
```

在 Linux 上 crontab 启动 nimon:

```
# crontab -e
```

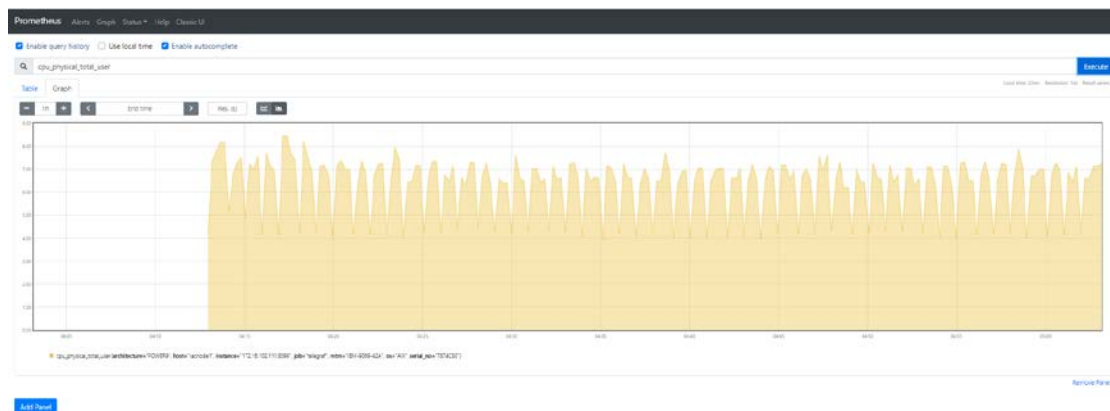
```
# crontab -l
```

```
13 4 * * * /usr/local/bin/nimon -s 5 -c 17280 -k -P -b -r -n -H -v -W -w -p 8888
```

5.3.1 验证 nimon 监控数据

从 Prometheus 界面 Graph 首页可以查看监控指标数据图表,如 cpu_physical_total_user

(User 开销比例)图如下:



5.4 安装部署 Grafana

5.4.1 安装并启动 Grafana

[安装 Grafana](#)与前面相同, 不再赘述。

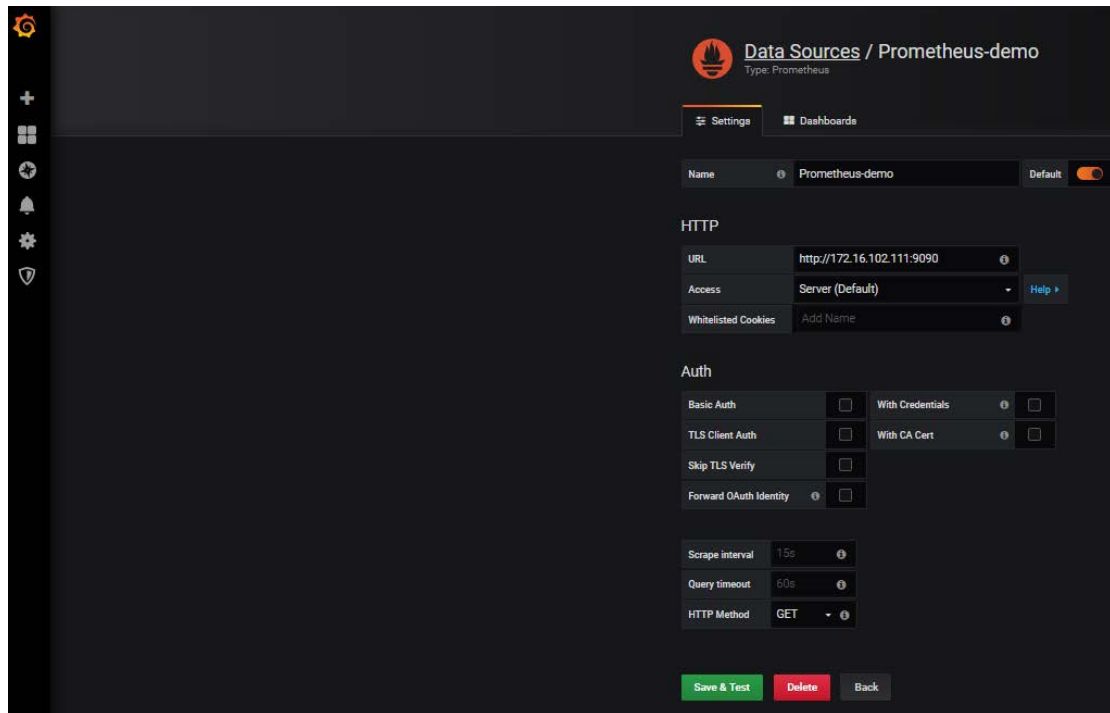
启动 grafana 服务

```
systemctl daemon-reload
```

```
systemctl start grafana-server
```

```
systemctl status -l grafana-server
```

5.4.2 添加 prometheus-demo(telegraf)数据源

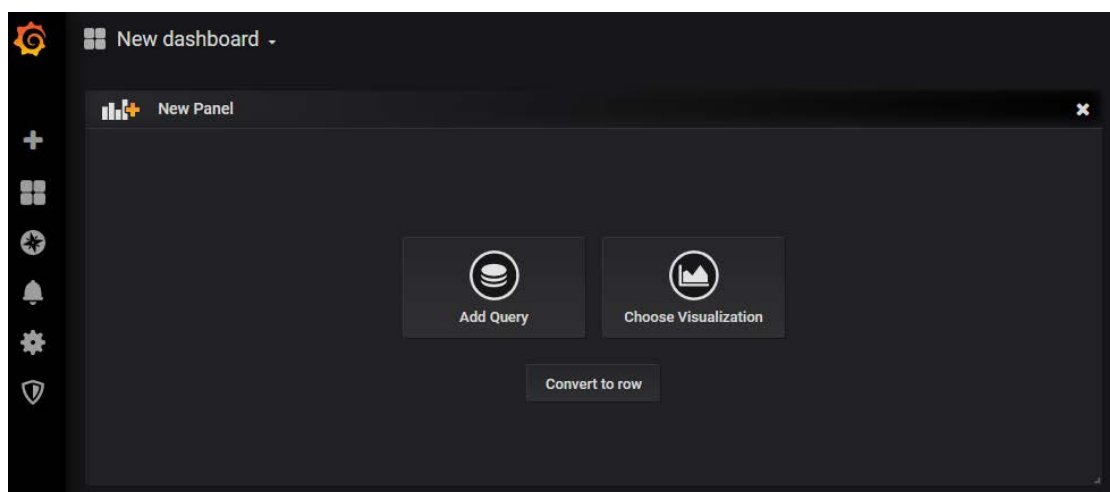


5.4.3 添加 Dashboard

在 <https://grafana.com/grafana/dashboards?search=njmon> 官网中现成的 njmon

Dashboard 是基于 InfluxDB 数据源的。因此这里我们先展示手工添加 Dashboard，选择

+New dashboard, Dashboard 名: Telegraf-AIX-demo



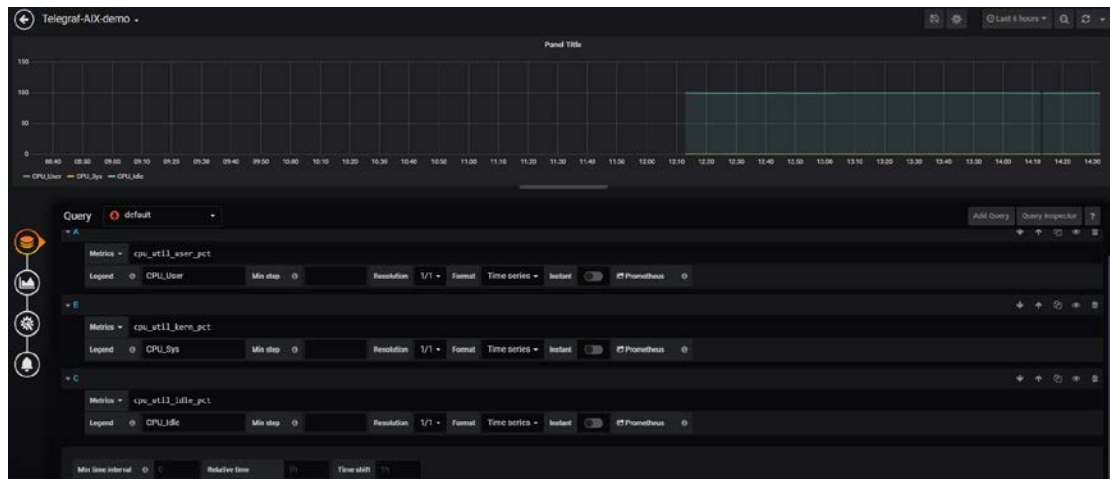
点 Add Query, Query 数据源选择 Prometheus-demo。再 Add Query 两次，共三个

Metrics, 分别输入:

cpu_util_user_pct, cpu_util_kern_pct, cpu_util_idle_pct

Legend 标签分别输入: **CPU_User, CPU_Sys, CPU_Idle**

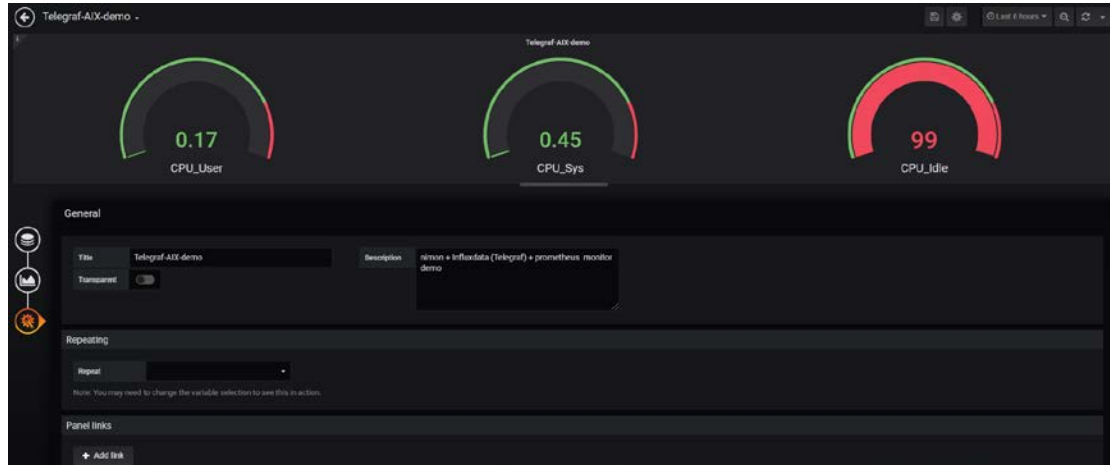
DashBoard 图表如下:



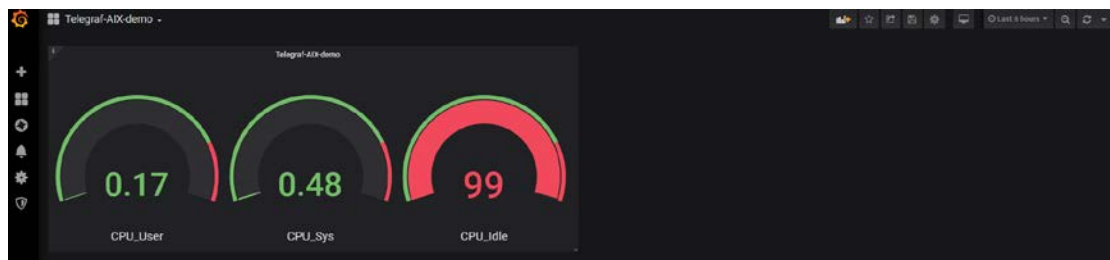
可以变更可视化窗格图表类型, 由于 CPU_Idle 的百分比值远大于 CPU_Sys,CPU_User,

Graph 显示不清楚, 可以将 Graph 变更为 Gauge 仪表盘, 图形如下:





保存 Dashboard 退出。然后从 Dashboard home 选择 Telegraf-AIX-demo 可以查看这个这个简单示例的效果如下图：



5.4.4 Grafana Alert 功能

5.4.4.1 配置 grafana SMTP 邮件服务

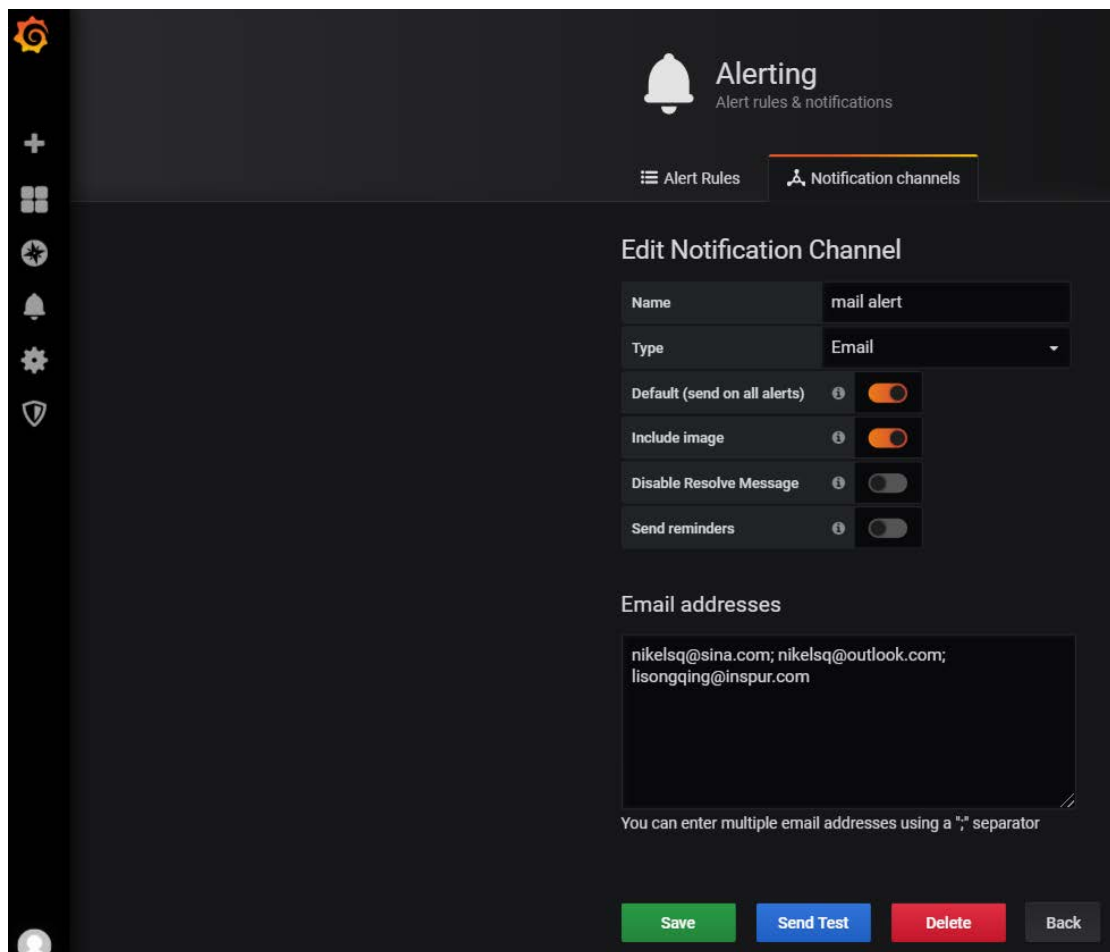
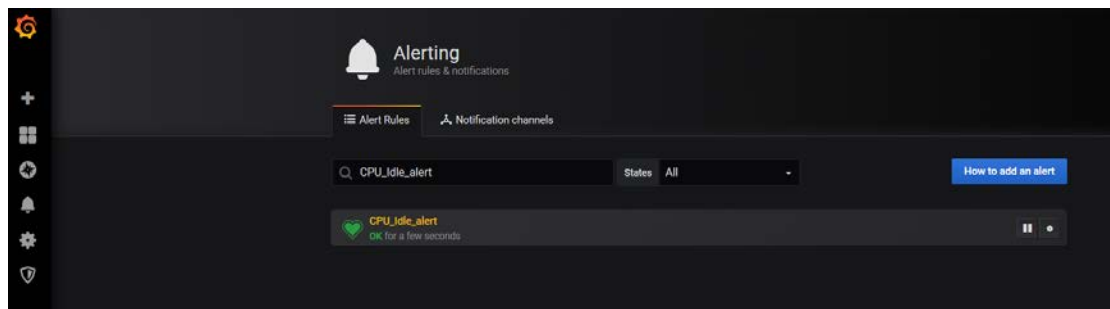
vi /etc/grafana/grafana.ini

```
##### SMTP / Emailing #####
[smtp]
enabled = true
host = smtp.live.com:587
user = nikelsq@outlook.com
# If the password contains # or ; you have to wrap it with trippel quotes. Ex ""#password;""
password =xxxxxxx
;cert_file =
;key_file =
skip_verify = true
from_address = nikelsq@outlook.com
from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com

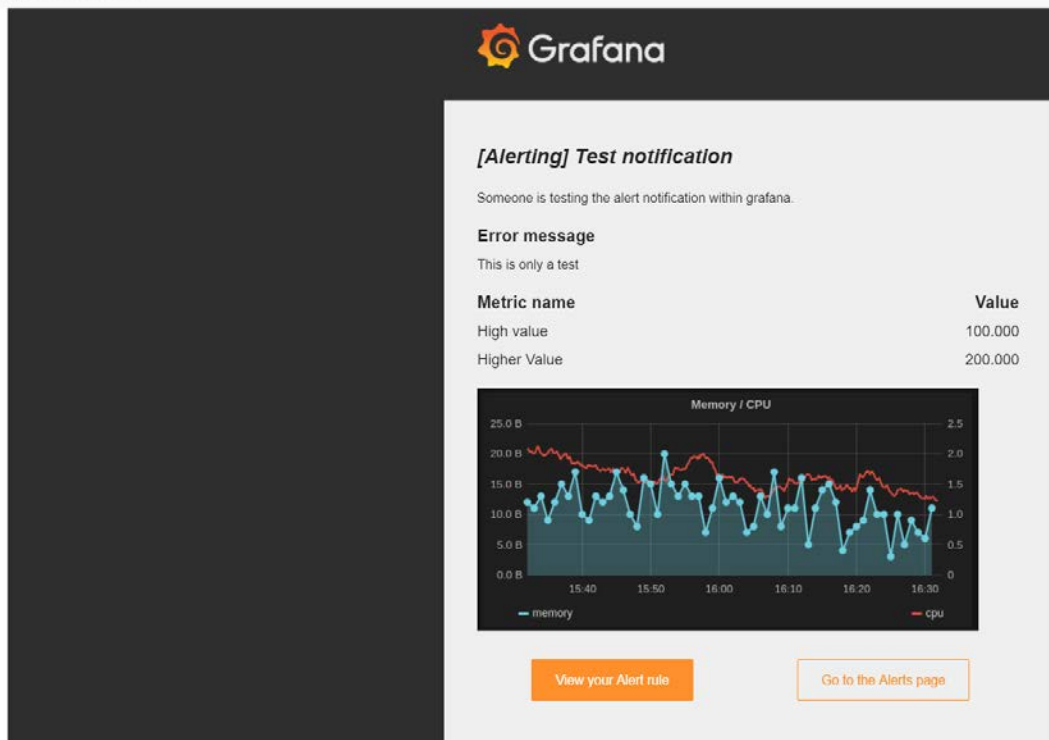
[emails]
;welcome_email_on_sign_up = false
```

重启 Grafan 服务 # systemctl restart grafana-server.service

在 Alerting - Notification Channel 添加邮件告警:

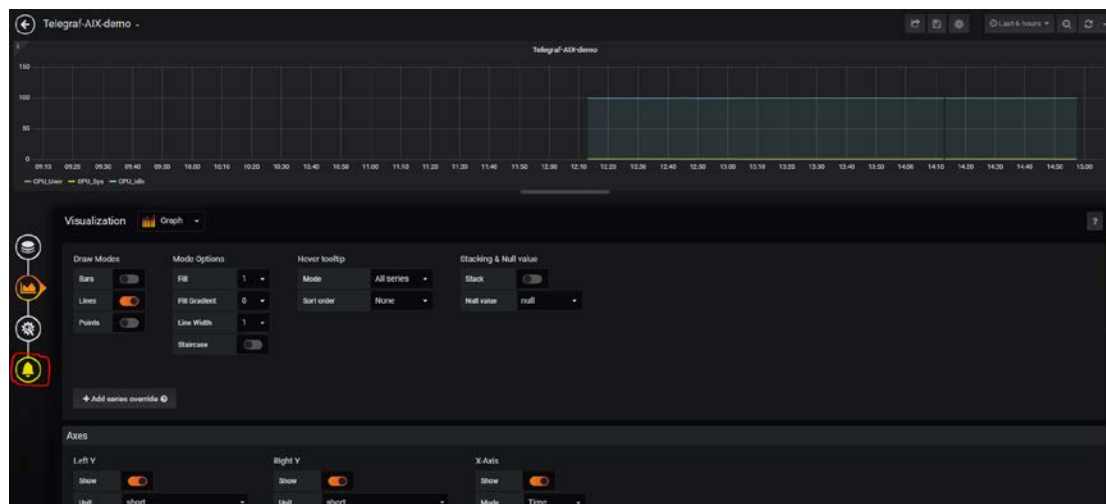


点击 Send Test, 各个邮箱将收到一个测试告警邮件:

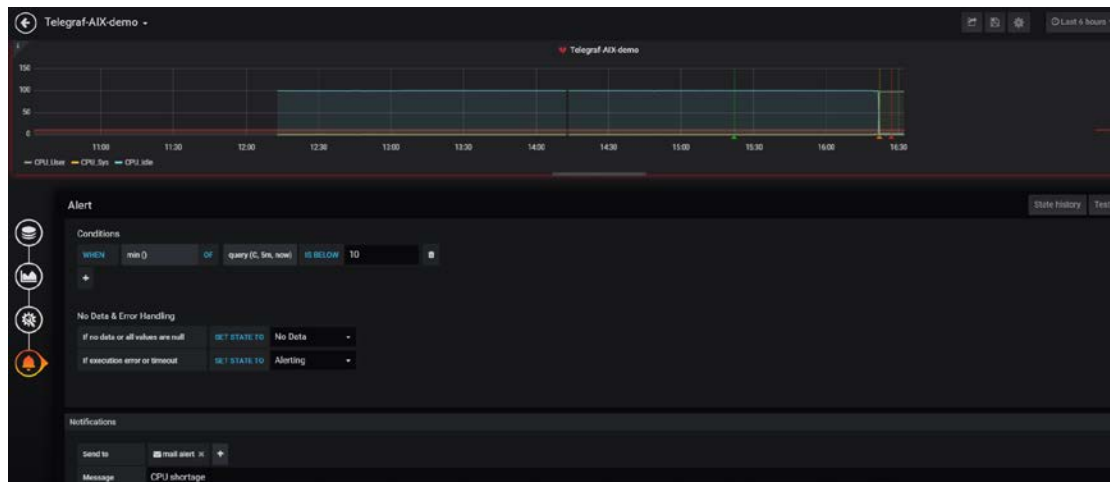


5.4.4.2 配置 Grafana Alert

Grafana 4.0 开始支持 Graph 图表支持 alert 告警功能。我们将前一节 **Dashboard Telegraf-AIX-demo** 改回 Graph 图表，显示如下图，左下角多出一个 alert 图表：



添加 CPU_Idle（监控指标 C） alerting,如果持续 5 分钟<10% (IS BELOW)则告警。



5.4.4.3 模拟 Alert

在 AIX 上运行 ncpu 持续 30 分钟压测，使得系统 CPU Idle%<10%，触发告警设置：

```
# ./ncpu -p 75 -z 0 -s 1800
```

```
./ncpu - processes=75 snooze=0% hibernate=0 secs for 600 seconds
```

Topas Monitor for host:racnode1							EVENTS/QUEUES		FILE/TTY	
Mon Apr 12 16:35:31 2021 Interval:1							Cswitch	280	Readch	2403
							Syscall	352	Writech	1348
CPU	User%	Kern%	Wait%	Idle%	Physc	Entc%	Reads	3	Rawin	0
Total	97.3	0.0	0.0	2.7	9.98	997.90	Writes	1	Ttyout	656
							Forks	0	Igets	0
Network	BPS	I-Pkts	O-Pkts	B-In	B-Out	Execs	0	Namei	4	
Total	1.79K	21.00	3.00	966.0	866.0	Runqueue	75.00	Dirblk	0	
							Waitqueue	0.0		
Disk	Busy%	BPS	TPS	B-Read	B-Writ				MEMORY	
Total	0.0	0	0	0	0	PAGING		Real, MB	262144	
							Faults	0	% Comp	4
FileSystem	BPS	TPS	B-Read	B-Writ		Steals	0	% Noncomp	3	
Total	2.35K	2.00	2.35K	0		PgspIn	0	% Client	3	
							PgspOut	0		
Name	PID	CPU%	PgSp	Owner		PageIn	0	PAGING SPACE		
ncpu	20185508	2.8	428K	root		PageOut	0	Size, MB	1024	
ncpu	19857778	2.3	180K	root		Sios	0	% Used	2	
ncpu	17236254	2.3	468K	root				% Free	98	
ncpu	10486242	1.2	426K	root						

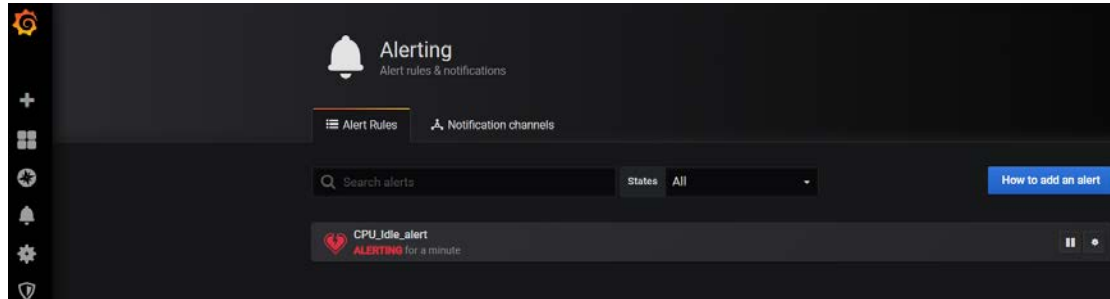
Grafana 运行日志/var/log/grafana.log 中将出现如下日志记录：

```
t=2021-04-12T16:35:01+0800 lvl=info msg="New state change" logger=alerting.resultHandler
ruleId=1 newState=pending prev state=ok
```

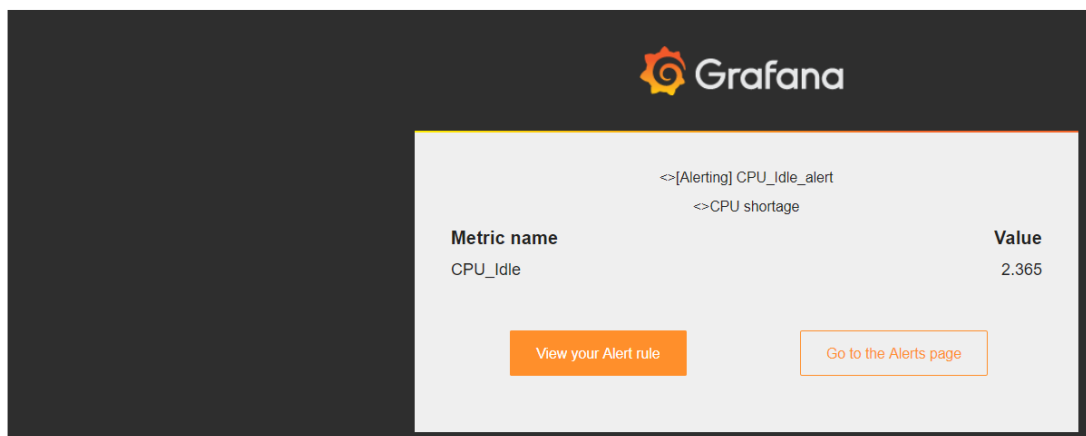
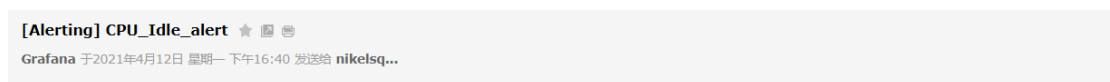
```
t=2021-04-12T16:40:01+0800 lvl=info msg="New state change" logger=alerting.resultHandler
ruleId=1 newState=alerting prev state=pending
```

t=2021-04-12T16:40:01+0800 lvl=info msg="Sending alert notification to"
logger=alerting.notifier.email addresses="[nikelsq@sina.com nikelsq@outlook.com
lisongqing@inspur.com]"

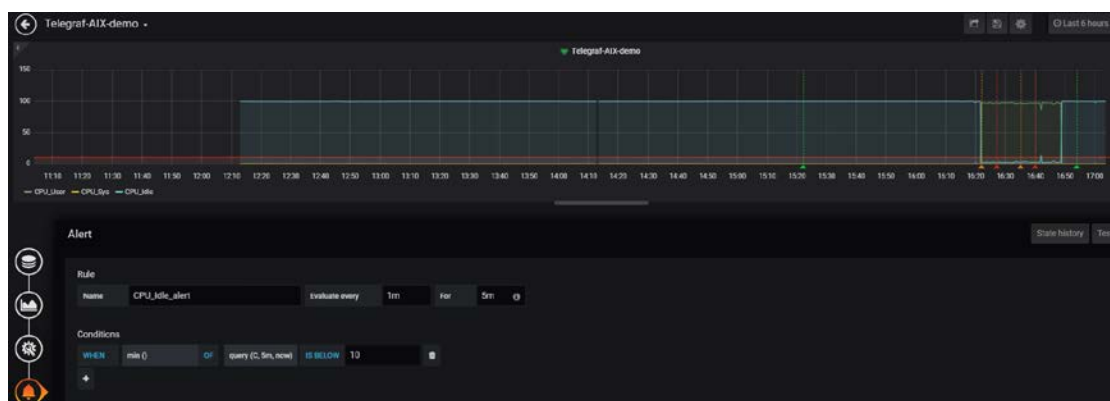
界面上将能看到 ALERTING 已触发:

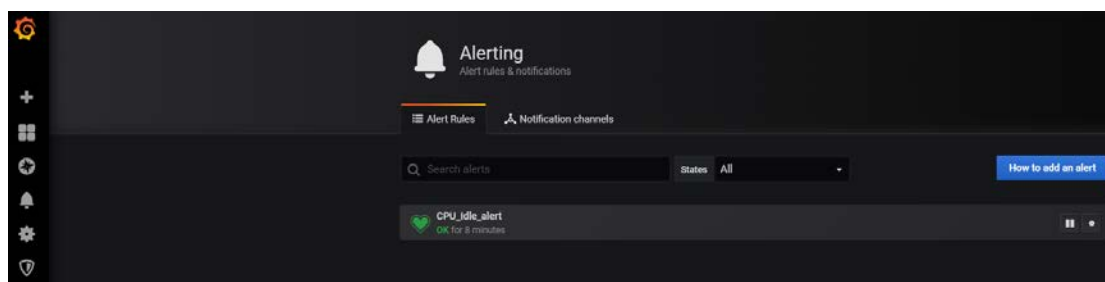


检查邮箱已收到 Alerting 告警邮件:



当负载降下来 5 分钟后, Alerting 状态清除:



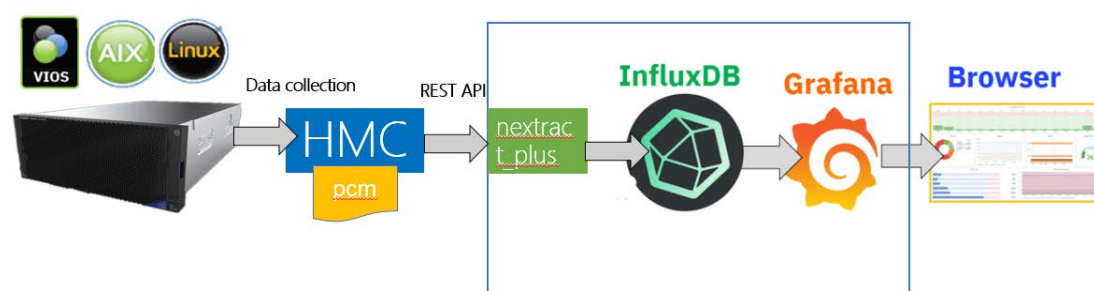


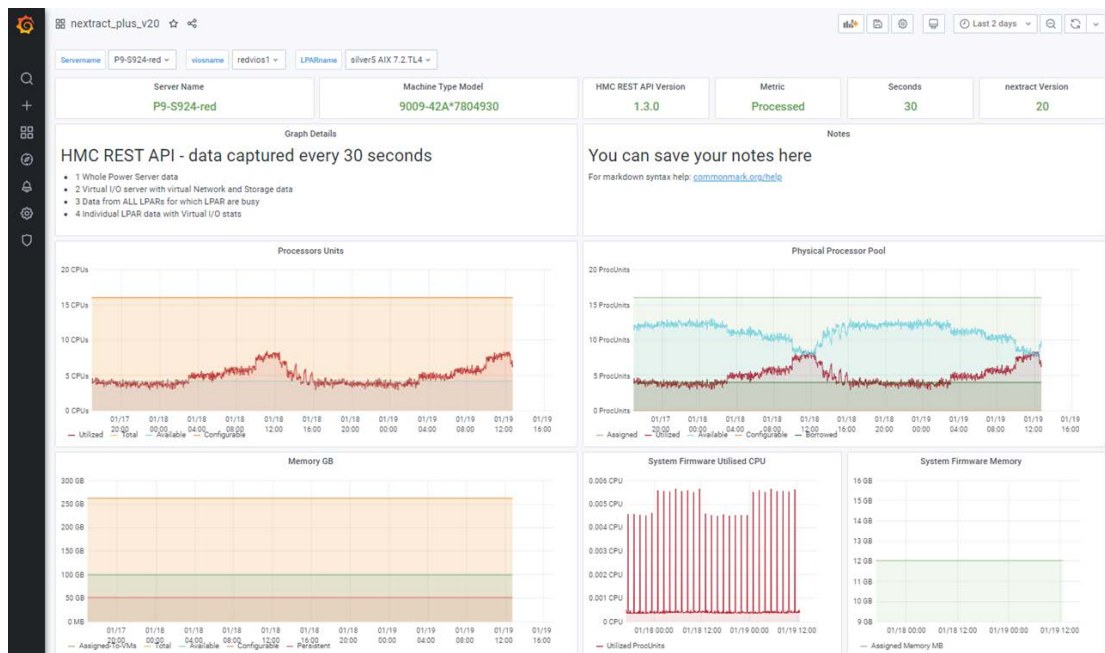
6 基 于 HMC 的 nextract_plus +InfluxDB +Grafana

基于 HMC 的监控解决方案，实现对 Power 系统和分区性能监控及展示功能。

HMC Performance and Capacity Monitoring (PCM)提供了一系列 JSON 格式的性能指标数据，可以通过 HMC REST API 获取。

Nigle arggriffiths 开发的 nextract_plus，就是通过 HMC REST API 获取 PCM 数据，经过计算和格式转换，发送到时间序列数据库 InfluxDB。存入 InfluxDB 的数据，在 Grafana 中查询并展示出来。

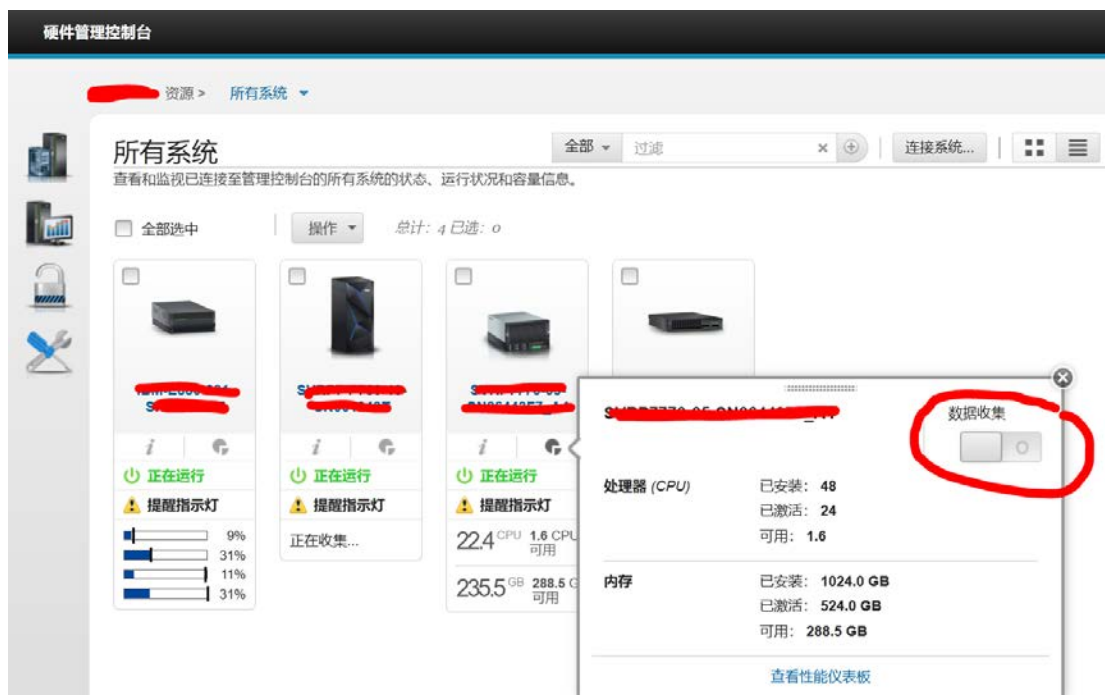




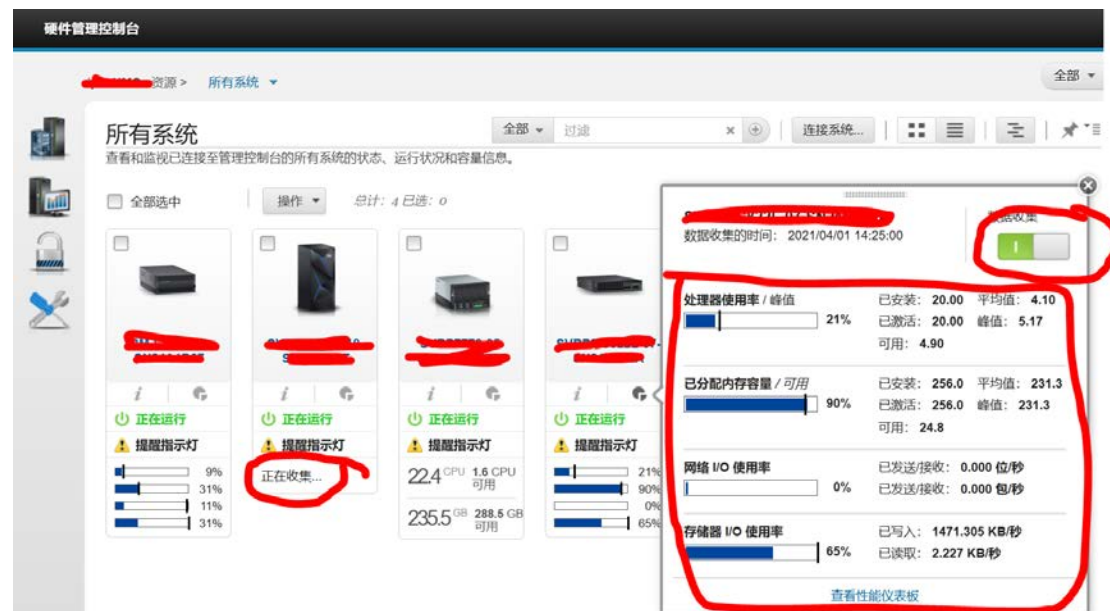
6.1 打开 HMC 的 Data Collection 功能

要想对服务器收集 PCM，需要先在 HMC 中打开服务器的 Data Collection 功能。

方法是登陆到 HMC，在 All system 视图中，在想要监控的服务器上点击“饼图”图标，然后打开 Data Collection 开关。如下图所示



打开后，需要等待一段时间，等 HMC 收集数据，在 All Systems 视图中，会看到该系统下显示“正在收集...”。已经收集完数据的服务器下面会显示使用率概要信息，再次点开“饼图”图标，可以看到较详细的使用率信息。如下图所示。



6.2 创建数据库

```
# influx -username admin -password admin -database njmon -host XXX.XXX.XXX.XXX -port 8086  
> create database hmc_nextractplus
```

下载 nextract_plus，解压。

```
mkdir nextract_plus
```

```
wget https://www.ibm.com/support/pages/system/files/inline-files/nextract\_plus34.zip
```

```
wget -c https://www.ibm.com/support/pages/system/files/inline-files/nextract\_plus35.zip
```

```
unzip nextract_plus34.zip
```

```
ls
```

```
example.conf      hmc_pcm.py      nextract_plus.py      nextract_plus_v35-  
1617464088040.json  README.txt
```

最新版本的 nextract_plus 在配置文件中配置 HMC 访问信息和 InfluxDB 访问信息。编辑配置文件如下：

```
cp example.conf hmc68.conf
```

```
vi hmc68.conf
```

```
{  
  
"hmc_hostname": "10.152.6.68",  
  
"hmc_username": "xxxxx",  
  
"hmc_password": "xxxxxxxxx",  
  
"ihostname": "localhost",  
  
"iport": 8086,  
  
"iusername": "",  
  
"ipassword": "",  
  
"idbname": "nextractplus"  
}
```

```
# python3 -m pip download influxdb
```



```
# python3 -m pip install influxdb
```

WARNING: Running pip install with root privileges is generally not a good idea. Try

```
`__main__.py install --user` instead.
```

Collecting influxdb

Downloading

<https://files.pythonhosted.org/packages/01/95/3a72ea5e19df828d27af0a50092f2b24114a5f89922efb4d8a0960bf13ef/influxdb-5.3.1-py2.py3-none-any.whl> (77kB)

100%  81kB 846kB/s

Collecting msgpack (from influxdb)

Downloading

<https://files.pythonhosted.org/packages/59/04/87fc6708659c2ed3b0b6d4954f270b6e931def707b227c4554f99bd5401e/msgpack-1.0.2.tar.gz> (123kB)

100%  133kB 1.1MB/s

Requirement already satisfied: python-dateutil>=2.6.0 in
/usr/local/lib/python3.6/site-packages (from influxdb)

Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/site-
packages (from influxdb)

Requirement already satisfied: pytz in /usr/local/lib/python3.6/site-packages (from
influxdb)

Collecting requests>=2.17.0 (from influxdb)

Downloading

<https://files.pythonhosted.org/packages/29/c1/24814557f1d22c56d50280771a173>

07e6bf87b70727d975fd6b2ce6b014a/requests-2.25.1-py2.py3-none-any.whl (61kB)

100%  61kB 1.4MB/s

Requirement already satisfied: urllib3<1.27,>=1.21.1 in

/usr/local/lib/python3.6/site-packages (from requests>=2.17.0->influxdb)

Collecting idna<3,>=2.5 (from requests>=2.17.0->influxdb)

Downloading

<https://files.pythonhosted.org/packages/a2/38/928ddce2273eaa564f6f50de91932>

7bf3a00f091b5baba8dfa9460f3a8a8/idna-2.10-py2.py3-none-any.whl (58kB)

100%  61kB 1.3MB/s

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/site-packages (from requests>=2.17.0->influxdb)

Collecting chardet<5,>=3.0.2 (from requests>=2.17.0->influxdb)

Downloading

<https://files.pythonhosted.org/packages/19/c7/fa589626997dd07bd87d9269342ccb74b1720384a4d739a1872bd84fbe68/chardet-4.0.0-py2.py3-none-any.whl> (178kB)

100%  184kB 1.1MB/s

Installing collected packages: msgpack, idna, chardet, requests, influxdb

Running setup.py install for msgpack ... done

Successfully installed chardet-4.0.0 idna-2.10 influxdb-5.3.1 msgpack-1.0.2 requests-2.25.1

python3 -m pip list

DEPRECATION: The default format will switch to columns in the future. You can use
--format=(legacy|columns) (or define a format=(legacy|columns) in your pip.conf
under the [list] section) to disable this warning.

certifi (2020.12.5)

chardet (4.0.0)

idna (2.10)

influxdb (5.3.1)

influxdb-client (1.15.0)

msgpack (1.0.2)

pip (9.0.3)

python-dateutil (2.8.1)

pytz (2021.1)

requests (2.25.1)

Rx (3.1.1)

setuptools (39.2.0)

six (1.15.0)

urllib3 (1.26.4)

然后就可以运行 nextract_plus 了。

```
# mkdir debug  
  
# python3 ./nextract_plus.py hmc_ips.conf
```

.....

```
Added 20793 records to InfluxDB for Server=SVRP8-S822L-07-SN2116AEA
```

```
Logging off the HMC - found 112157 measures
```

```
DEBUG:Logoff()
```

```
DEBUG:Successfully disconnected from the HMC (code=204)
```

整个运行时间有几分钟。完成后，屏幕会显示已从 HMC 注销，如上图所示。

注：因为 nextract_plus 是收集之前 2 小时的数据，完成任务后，程序退出。因此最好将 nextract_plus 放到 crontab 中，每小时运行一次。

nextract_plus 运行结束后，可以进入 InfluxDB 查看数据

```
influx
```

```
> use nextractplus
```

```
Using database nextractplus
```

```
> show measurements
```

```
name: measurements
```

```
name
```

```
----
```

```
lpar_memory
```

```
lpar_net_virtual
```

```
lpar_processor
```

```
lpar_storage_vFC
```

```
lpar_storage_virtual
```

```
server_details
```

```
server_memory
```

server_physicalProcessorPool

server_processor

server_sharedProcessorPool

server_sriov

vios_details

vios_memory

vios_network_generic

vios_network_lpars

vios_network_shared

vios_network_virtual

vios_processor

vios_storage_FC

vios_storage_lpars

vios_storage_physical

vios_storage_virtual

> select count(*) from server_details

name: server_details

time	count_APIVersion	count_assignedMem	count_frequency	count_metric
------	------------------	-------------------	-----------------	--------------

count_mtm	count_name	count_nextract	count_utilizedProcUnits
-----------	------------	----------------	-------------------------

0	601	601	601	601
---	-----	-----	-----	-----

601

601

601

```
create database server;
```

```
create database hmc_nextract;
```

7 附录

后续我们将陆续推出一些进阶主题，如:实时可视化监控方案高可用设计，监控和告警指标定制等。