Prometheus 监控服务

最佳实践

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

文档目录

最佳实践
- 自建 Prometheus 迁入
- 云服务器场景下自定义接入
- 容器场景监控
- TKE Serverless 集群如何放通外网
- Prometheus 监控服务如何接入本地 Grafana

最佳实践

自建 Prometheus 迁入

最近更新时间：2024-01-29 15:55:14

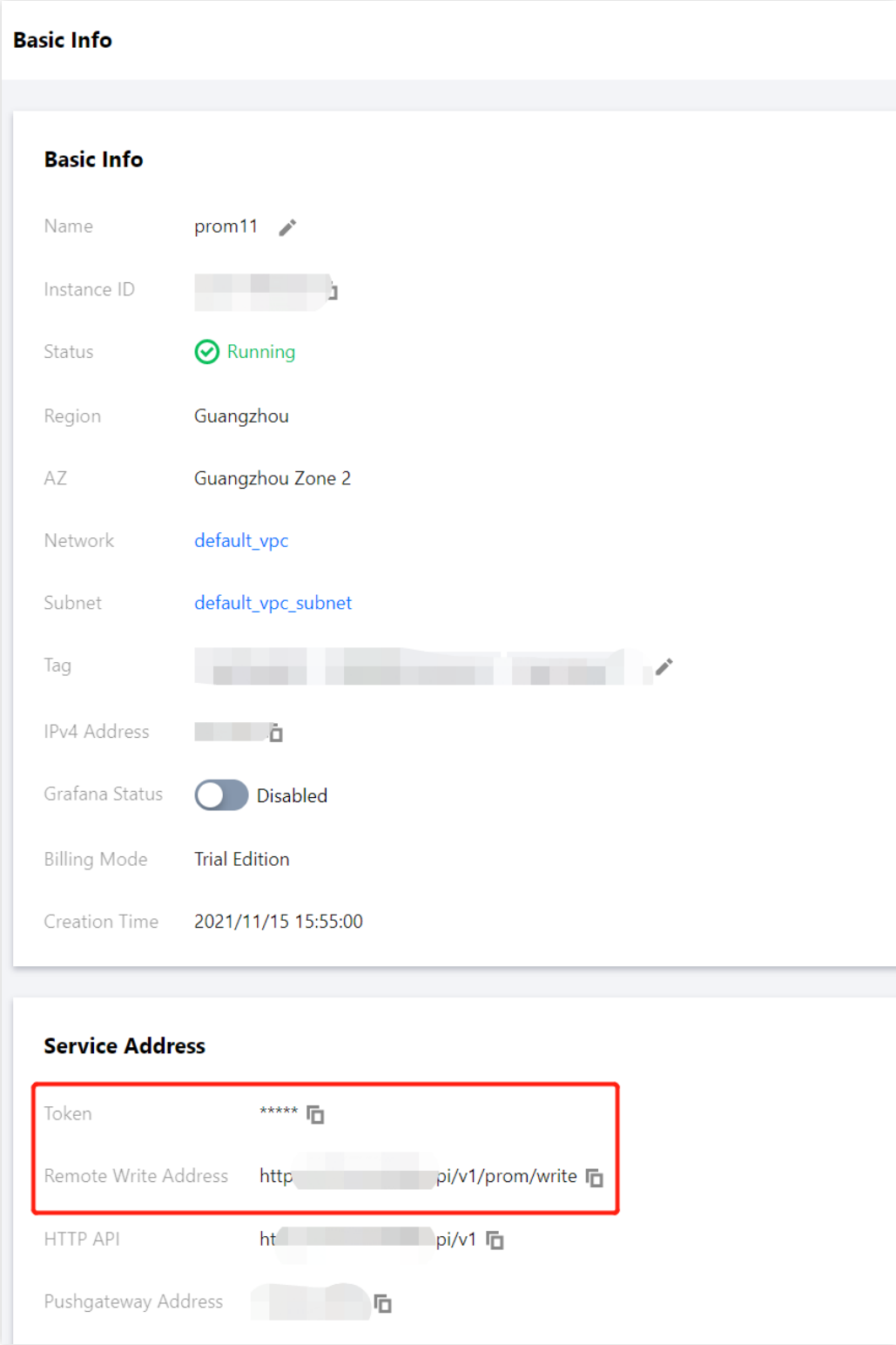
操作场景

已经有自建 Prometheus 需快速迁移到 Prometheus 监控服务。

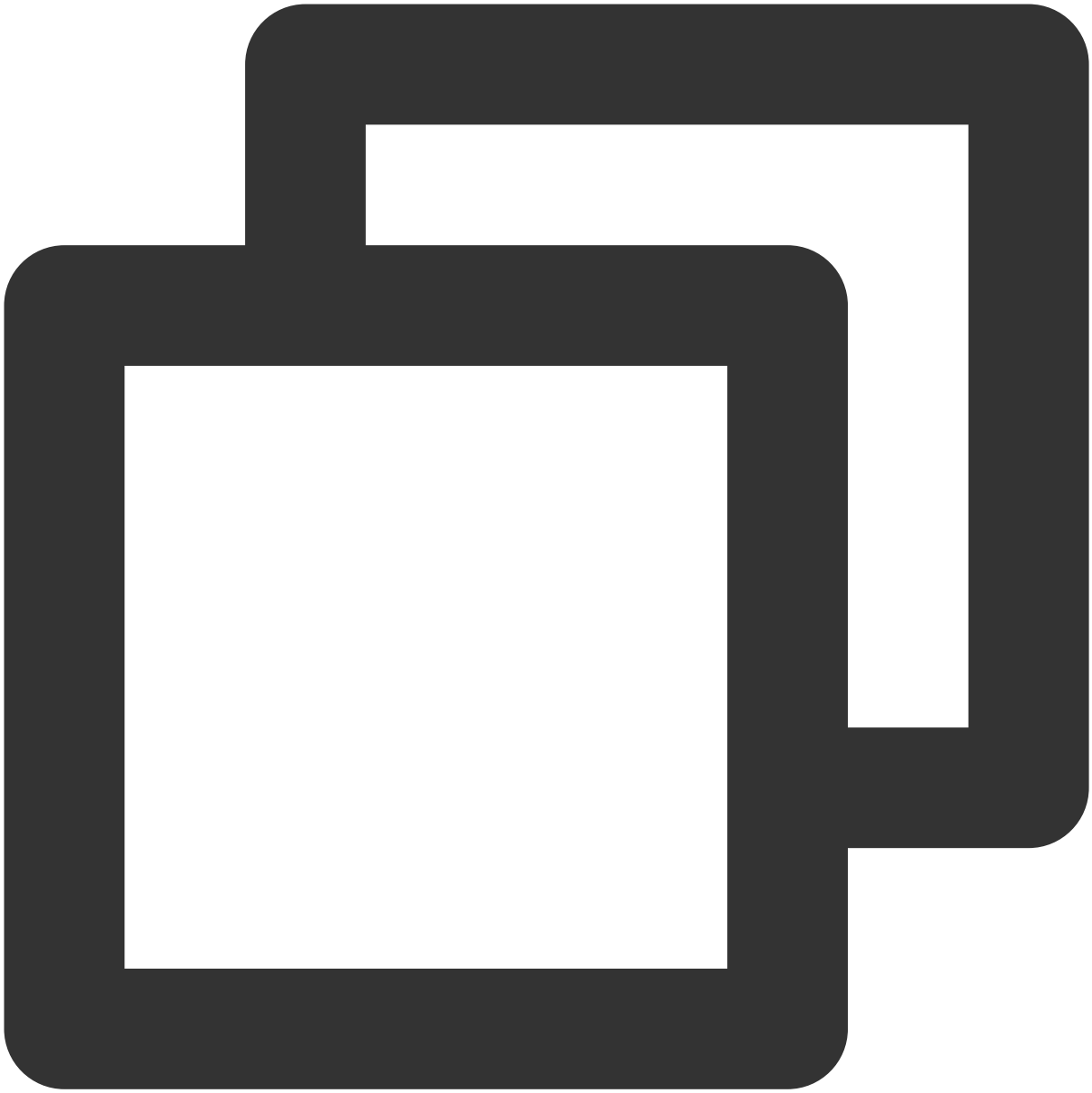
操作步骤

Prometheus 本身支持 Remote Write 到一个外部存储，因此沿用这个思想，在自建 Prometheus 的配置文件中加一个 Remote Write 配置指向到 Prometheus 监控服务即可。具体操作步骤如下：

1. 通过实例基本信息获取 Prometheus监控服务的 Remote Write 地址及 Token，如下：

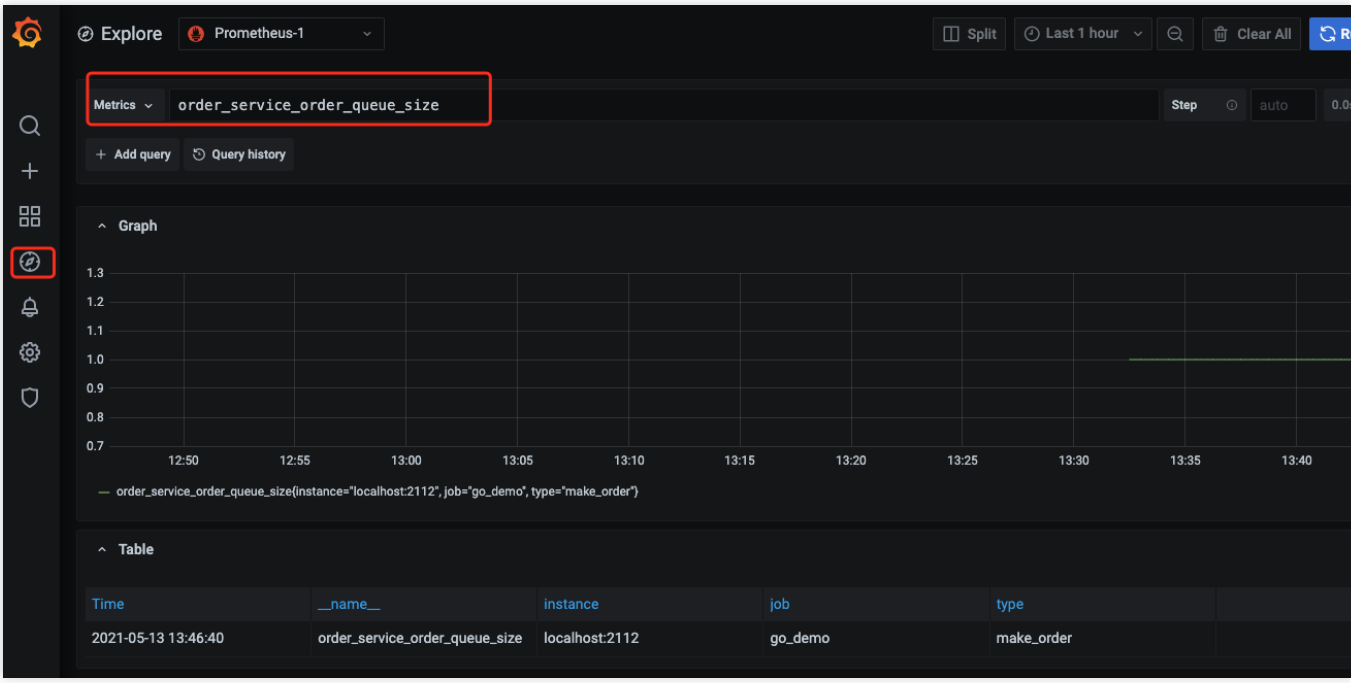


2. 修改 `prometheus.yml` ，修改完成 重启 `Prometheus` ，具体配置如下，如需了解更多 Remote Write 配置参数，请参考 remote_write 。



```
remote_write:

  - name: cm_prometheus # Remote write 的名称
    url: http://ip:port/api/v1/prom/write  # 从 Prometheus 基本信息中获取 Remote Write
    remote_timeout: 30s # 根据实际情况设置
    bearer_token: k32*****trR # 从 Prometheus 基本信息中获取 Token 信息
```

3. 打开 Prometheus 监控服务自带的 Grafana，通过 `Explore` 来验证数据是否写入成功，如下图，也可以 自定义 Grafana 监控大盘。



4. 也可以通过Prometheus API 进行自建可视化，详情请参考 监控数据查询。

云服务器场景下自定义接入

最近更新时间：2024-01-29 15:55:14

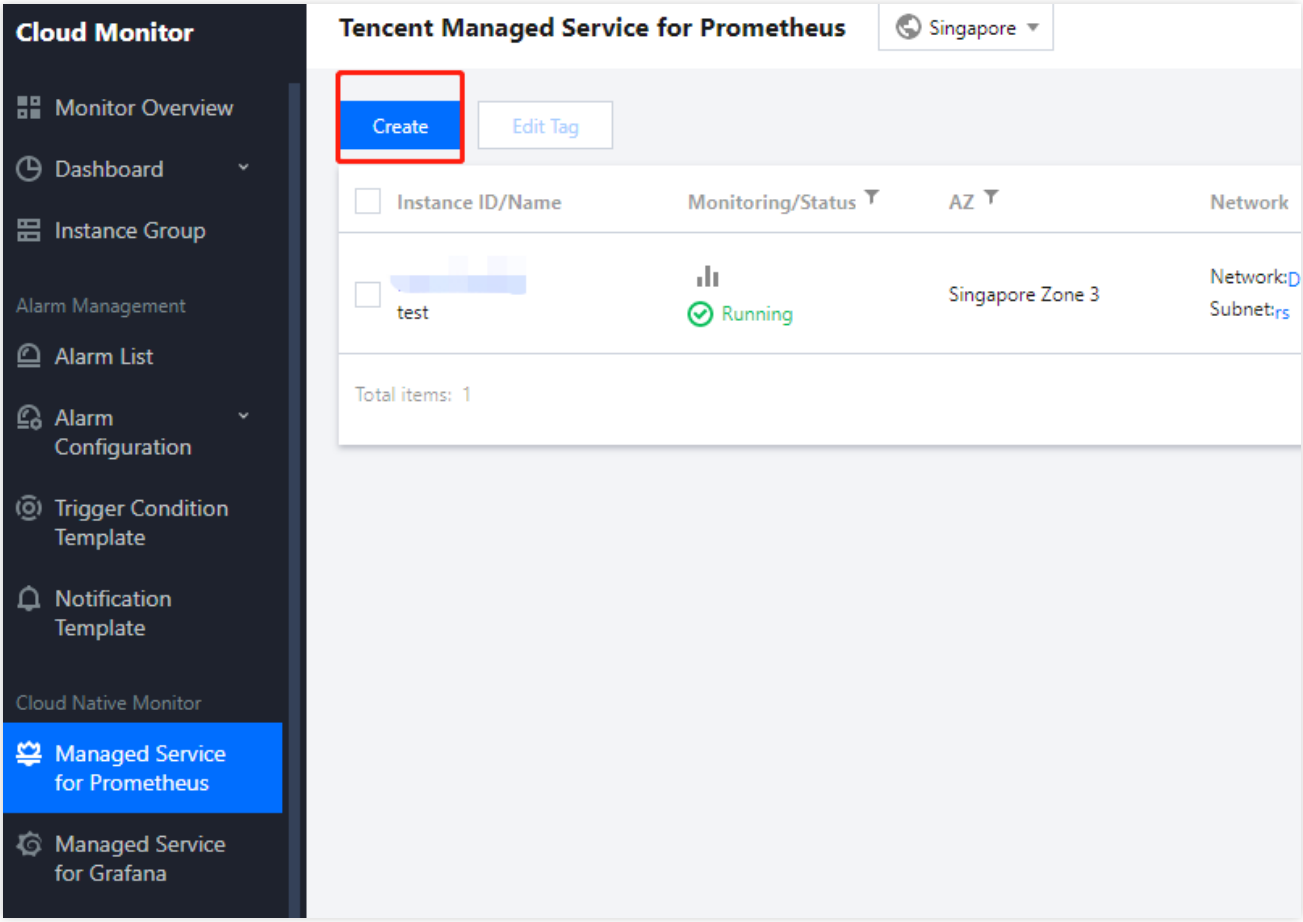
本文将详细为您介绍腾讯云云服务器（CVM）如何从0开始接入 Prometheus 监控服务。

购买 Prometheus

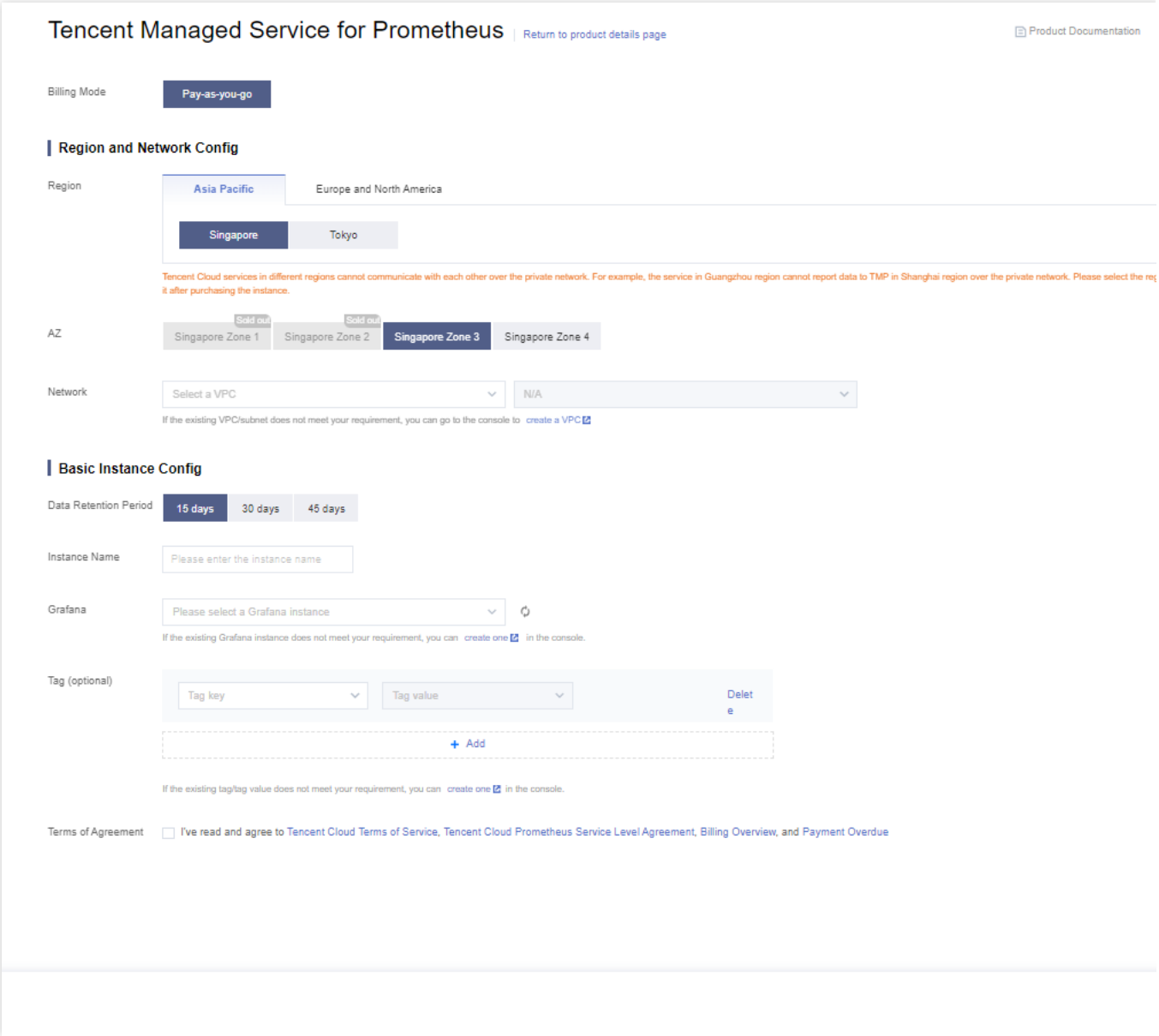
说明：

购买的 Prometheus 实例需跟监控的云服务器同一个 vpc 下，能实现网络互通。

1. 登录 Prometheus 监控服务控制台，单击**新建**购买 Prometheus 实例 。



2. 在购买页中，选择合适的实例规格、网络。选择相同 vpc 网段，保证 Prometheus 能与需要采集的云服务器网段相同，才能采集到数据。实例规格，可根据自己的业务上报量进行选择。

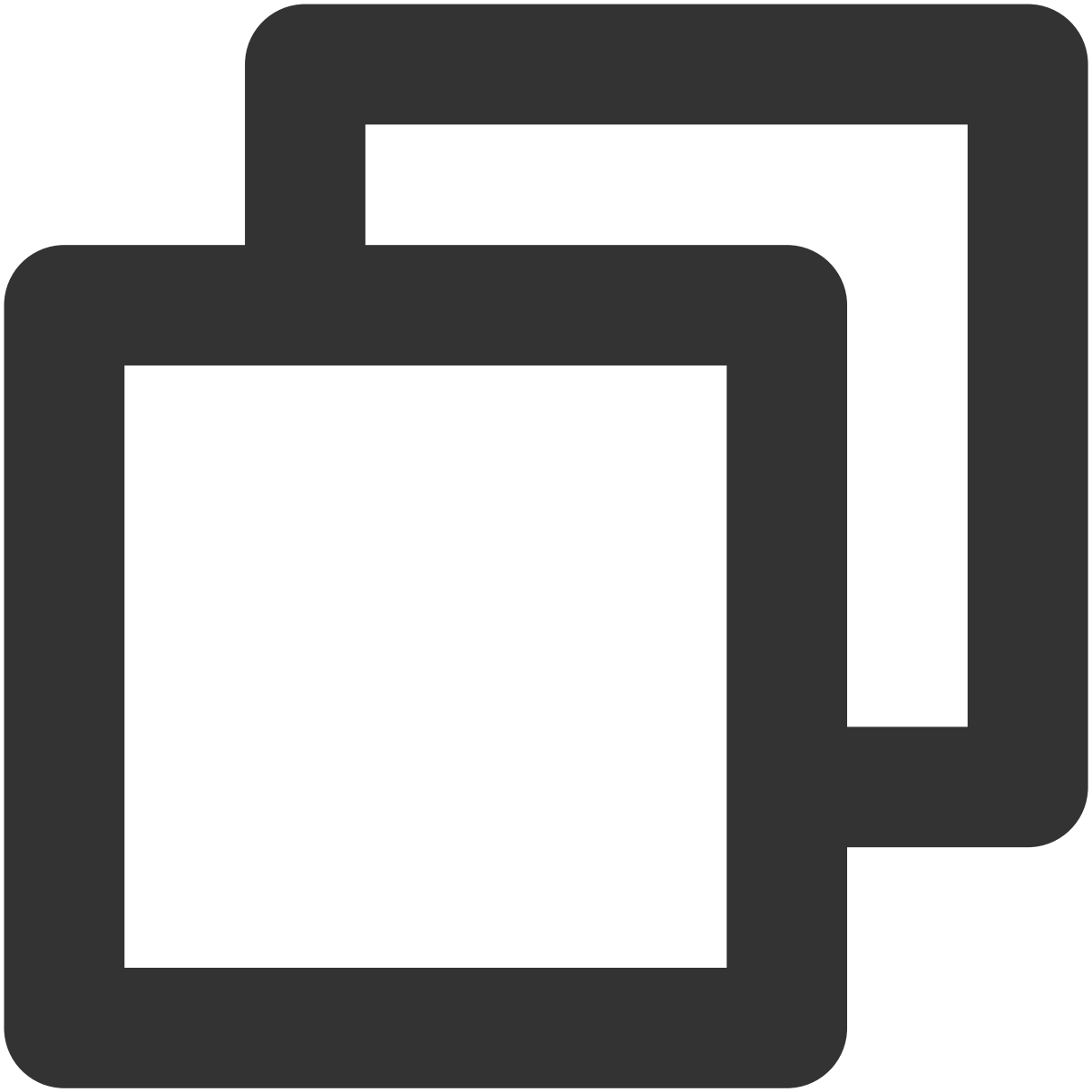


3. 选择完后，单击立即购买并支付即可。

接入云服务器基础指标

1. 下载安装 node_exporter：

在需要上报的云服务器上，下载并安装 node_exporter（采集基础指标数据的 exporter），您可以单击进入 Prometheus 开源官网下载地址 node_exporter，也可以直接执行下列命令，下载解压：



```
wget https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_expo
```

文件目录如下:

```
[root@VM-0-7-centos node_exporter-1.2.2.linux-amd64]# ll
total 18080
-rw-r--r-- 1 3434 3434    11357 Aug  6  2021 LICENSE
-rwxr-xr-x 1 3434 3434 18494215 Aug  6  2021 node_exporter
-rw-r--r-- 1 3434 3434      463 Aug  6  2021 NOTICE
[root@VM-0-7-centos node_exporter-1.2.2.linux-amd64]# ./node_exporter
```

2. 运行 node_exporter 采集基础监控数据：

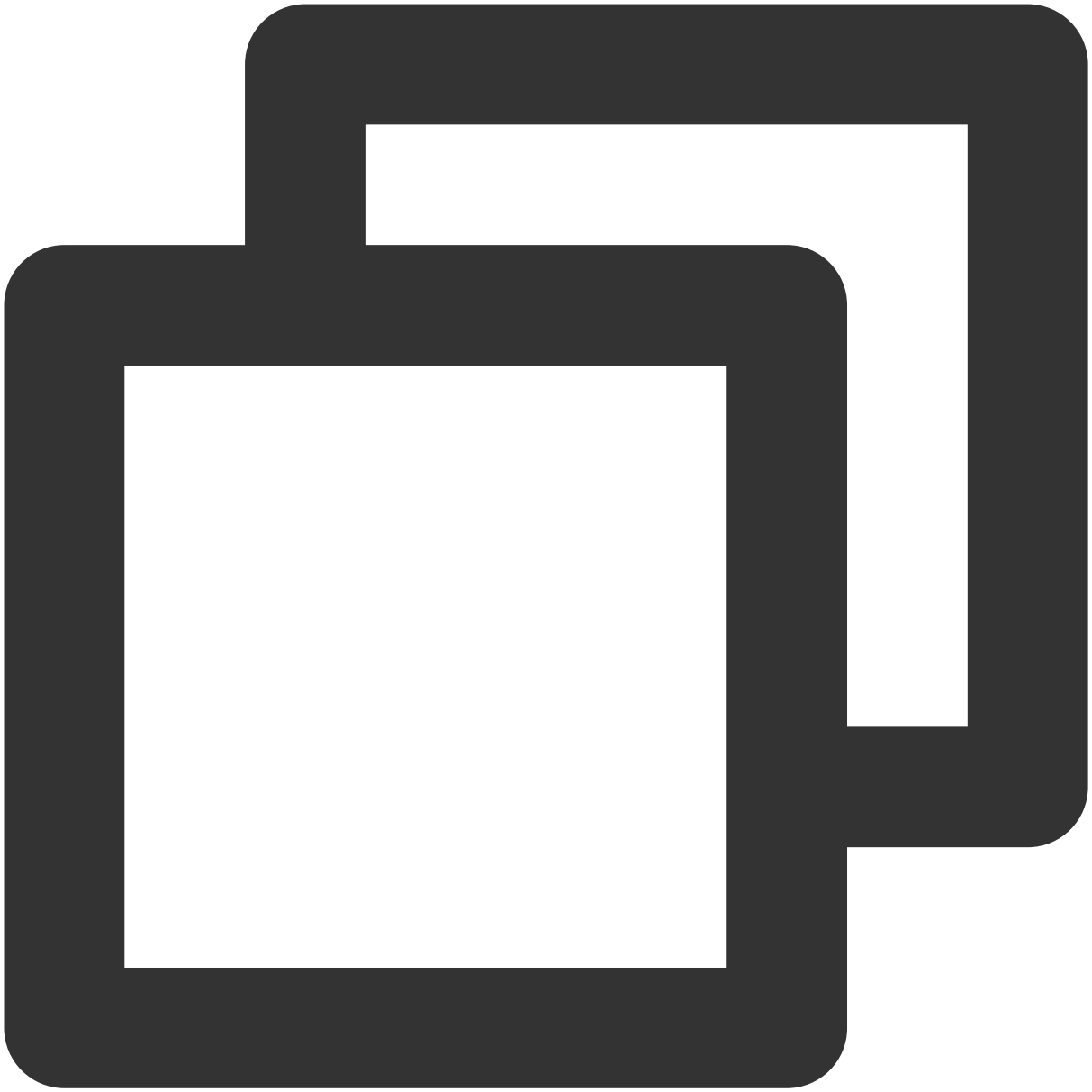
2.1 进入相关文件夹，执行 node_exporter。

```
cd node_exporter-1.3.1.linux-amd64
./node_exporter
```

如下图所示即为成功采集到了基础监控数据。

```
rw-r--r-- 1 3434 3434      463 Aug  6  2021 NOTICE
[root@VM-0-7-centos node_exporter-1.2.2.linux-amd64]# ./node_exporter
evel=info ts=2022-02-11T07:15:26.555Z caller=node_exporter.go:182 msg="Starting node_exporter" version
on=26645363b486e12be40af7ce4fc91e731a33104e)"
evel=info ts=2022-02-11T07:15:26.555Z caller=node_exporter.go:183 msg="Build context" build_context="(
date=20210806-13:44:18)"
evel=warn ts=2022-02-11T07:15:26.555Z caller=node_exporter.go:185 msg="Node Exporter is running as roo
run as unpriviledged user, root is not required."
evel=info ts=2022-02-11T07:15:26.555Z caller=filesystem_common.go:110 collector=filesystem msg="Parsed
ints-exclude" flag=^/(dev|proc|sys|var/lib/docker/.+)($|/)
evel=info ts=2022-02-11T07:15:26.555Z caller=filesystem_common.go:112 collector=filesystem msg="Parsed
exclude" flag=^(autofs|binfmt_misc|bpf|cgroup2?|configfs|debugfs|devpts|devtmpfs|fusectl|hugetlbfs|iso
ostore|rpc_pipefs|securityfs|selinuxfs|squashfs|sysfs|tracefs)$
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:108 msg="Enabled collectors"
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=arp
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=bcache
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=bonding
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=btrfs
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=conntrack
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=cpu
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=cpufreq
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=diskstats
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=edac
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=entropy
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=fibrechannel
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=filefd
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=filesystem
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=hwmon
evel=info ts=2022-02-11T07:15:26.556Z caller=node_exporter.go:115 collector=infiniband
```

2.2 可通过下列命令，将该基础监控数据暴露在9100 端口：



```
curl 127.0.0.1:9100/metrics
```

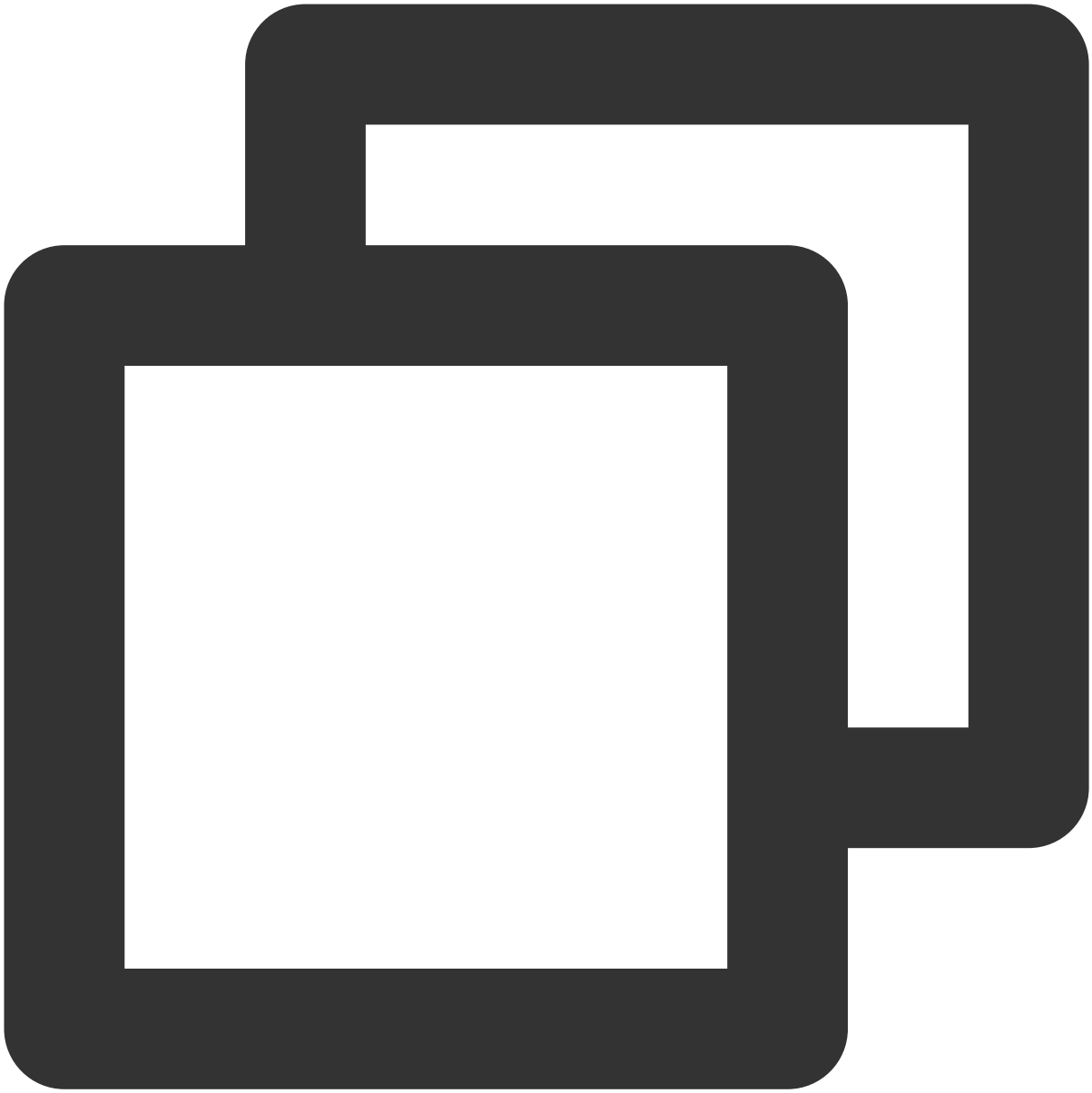
如下图为执行命令后看到的暴露出来的指标监控数据。

```
[root@VM-0-7-centos node_exporter-1.2.2.linux-amd64]# clear
[root@VM-0-7-centos node_exporter-1.2.2.linux-amd64]# curl 127.0.0.1:9100/metrics
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.16.7"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.344136e+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 2.344136e+06
# 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 4562
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 1362
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since
# TYPE go_memstats_gc_cpu_fraction gauge
```

3. 新增抓取任务：

登录 Prometheus 监控服务控制台，进入 **集成中心** > **选择云服务器**，在任务配置中根据页面提示进行配置。

抓取任务参考配置如下：



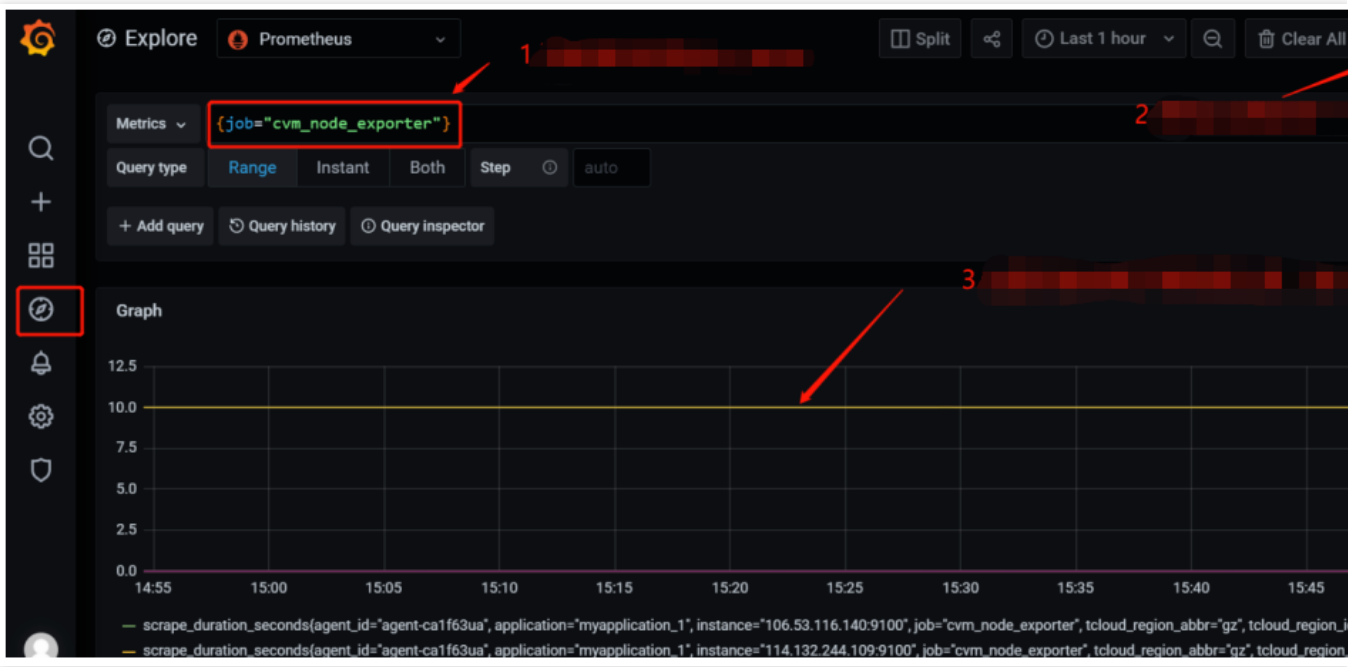
```
job_name: example-job-name
metrics_path: /metrics
cvm_sd_configs:
- region: ap-guangzhou
  ports:
  - 9100
  filters:
  - name: tag:示例标签键
    values:
    - 示例标签值
relabel_configs:
```

```
  - source_labels: [__meta_cvm_instance_state]
    regex: RUNNING
    action: keep
  - regex: __meta_cvm_tag_(.*)
    replacement: $1
    action: labelmap
  - source_labels: [__meta_cvm_region]
    target_label: region
    action: replace
```

4. 查看数据是否上报成功：

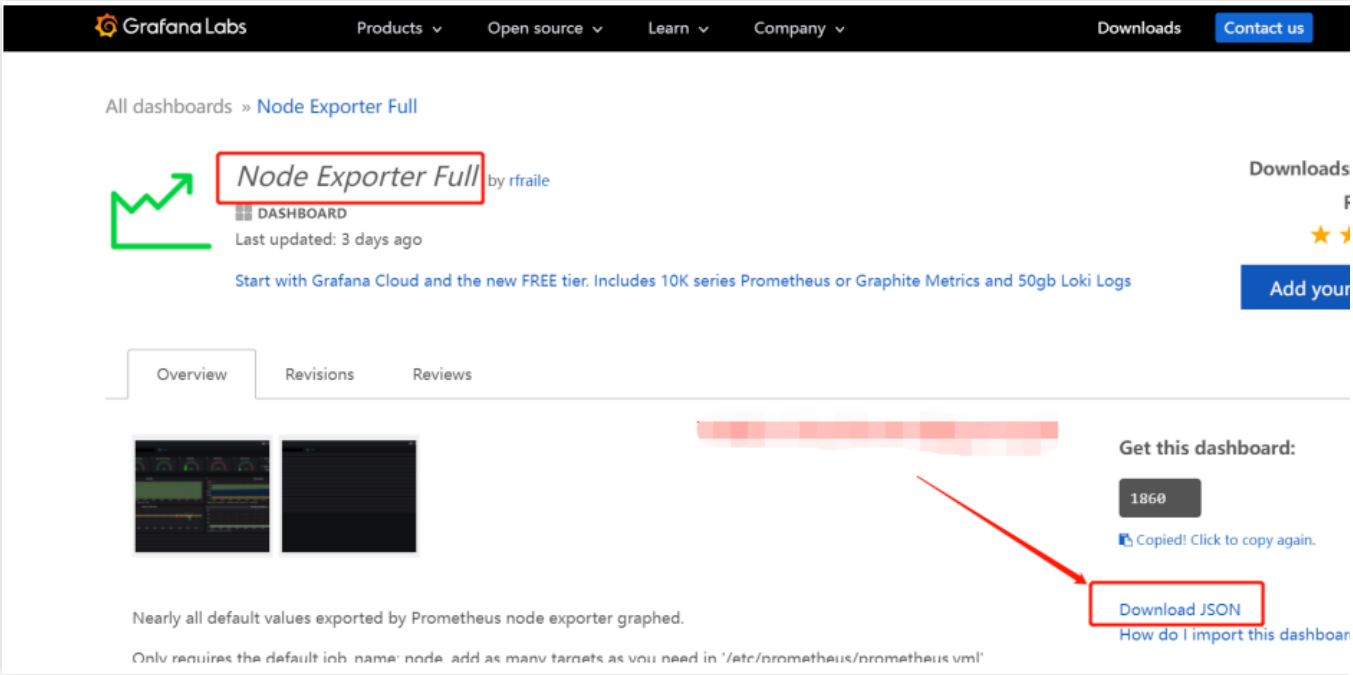
登录 Prometheus 监控服务控制台，单击 Grafana 图标，进入 Grafana。

如下图所示，到 Explore 搜索下 `{job="cvm_node_exporter"}` 查看是否有数据，若有数据，则表示上报成功。

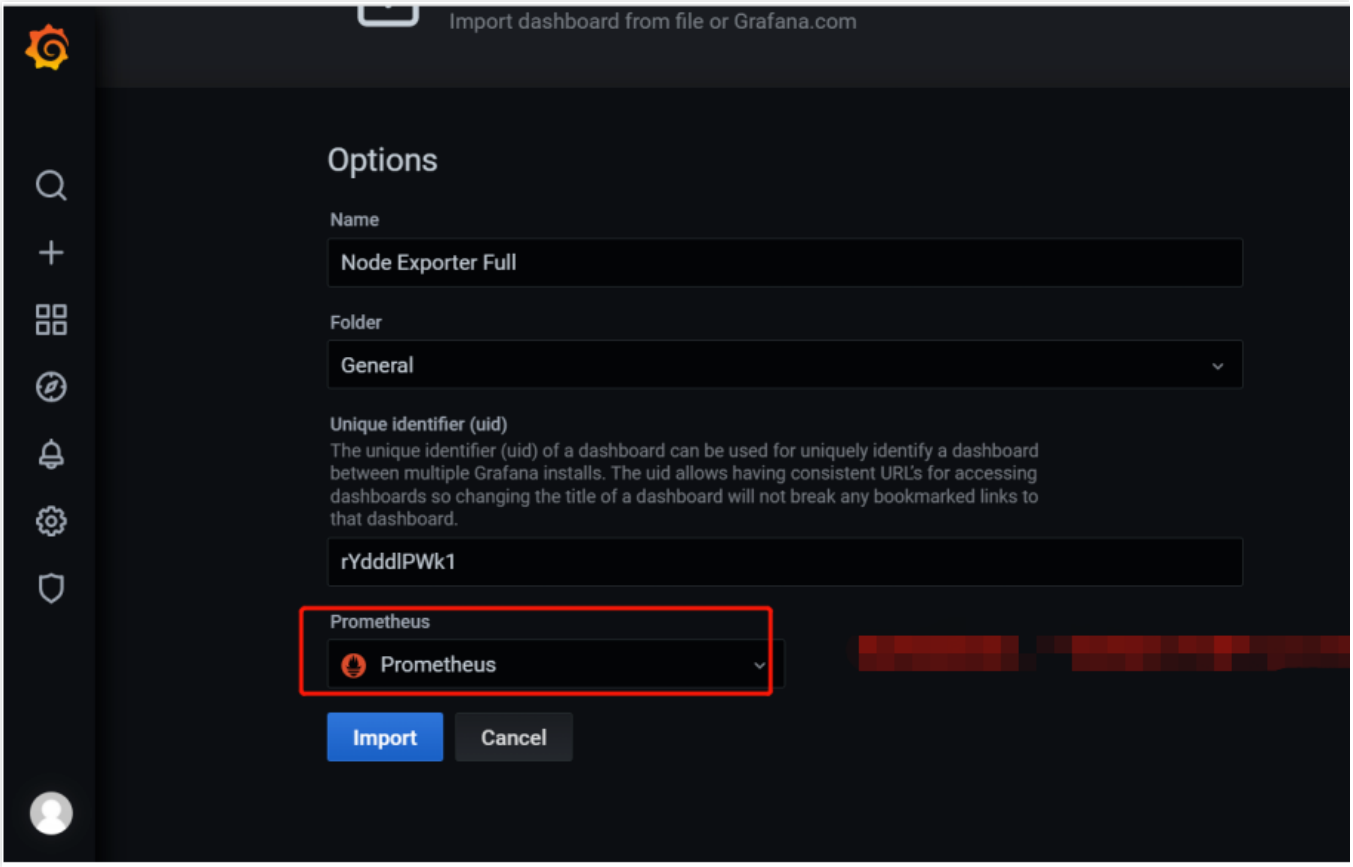


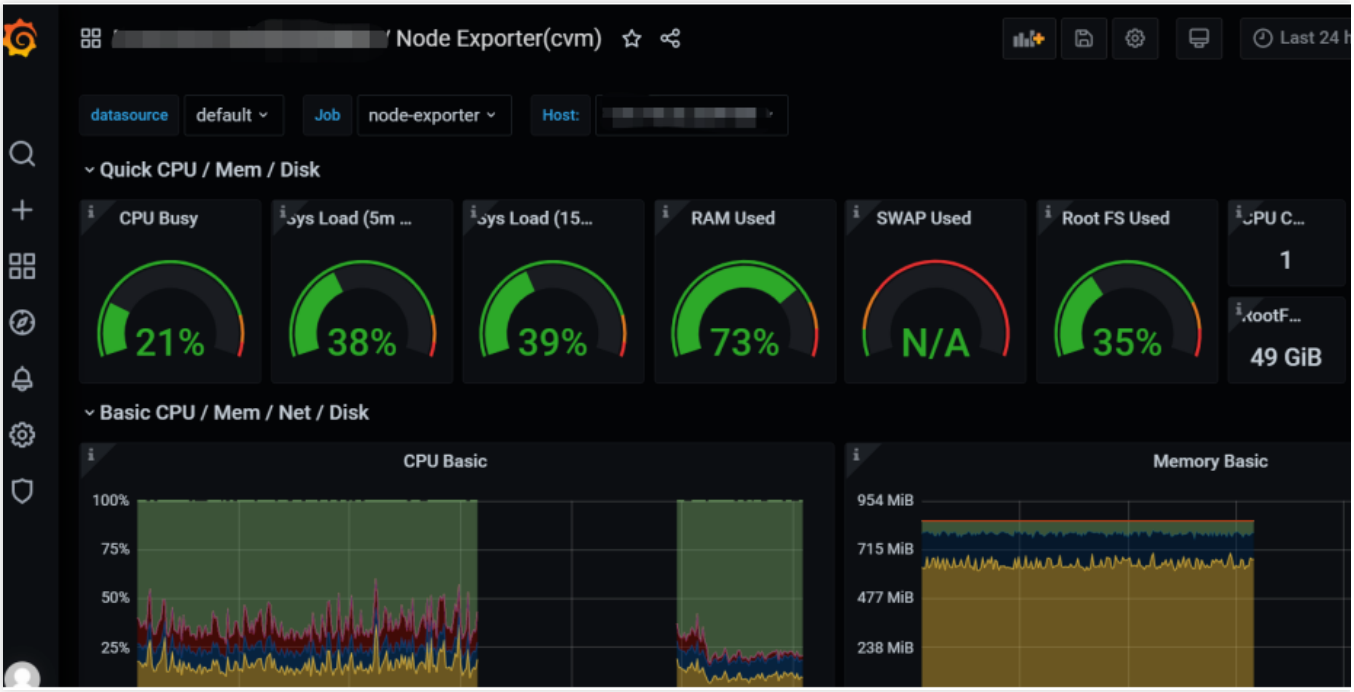
5. 配置 Dashboard 界面：Dashboard 界面每个产品都会有一些现成的 json 文件，可以直接导入。

5.1 **下载 Dashboard 文件**：登录 Dashboard 界面，单击搜索 node_exporter，选择最新的 Dashboard 并下载。



5.2 **导入 Dashboard 的 json 文件**：登录 Prometheus 监控服务控制台，进入**基本信息 > Grafana 地址**，单击进入 Grafana，在 Grafana 控制台 > Create > Import > 在 Upload JSON file 中上传 Dashboard 文件。





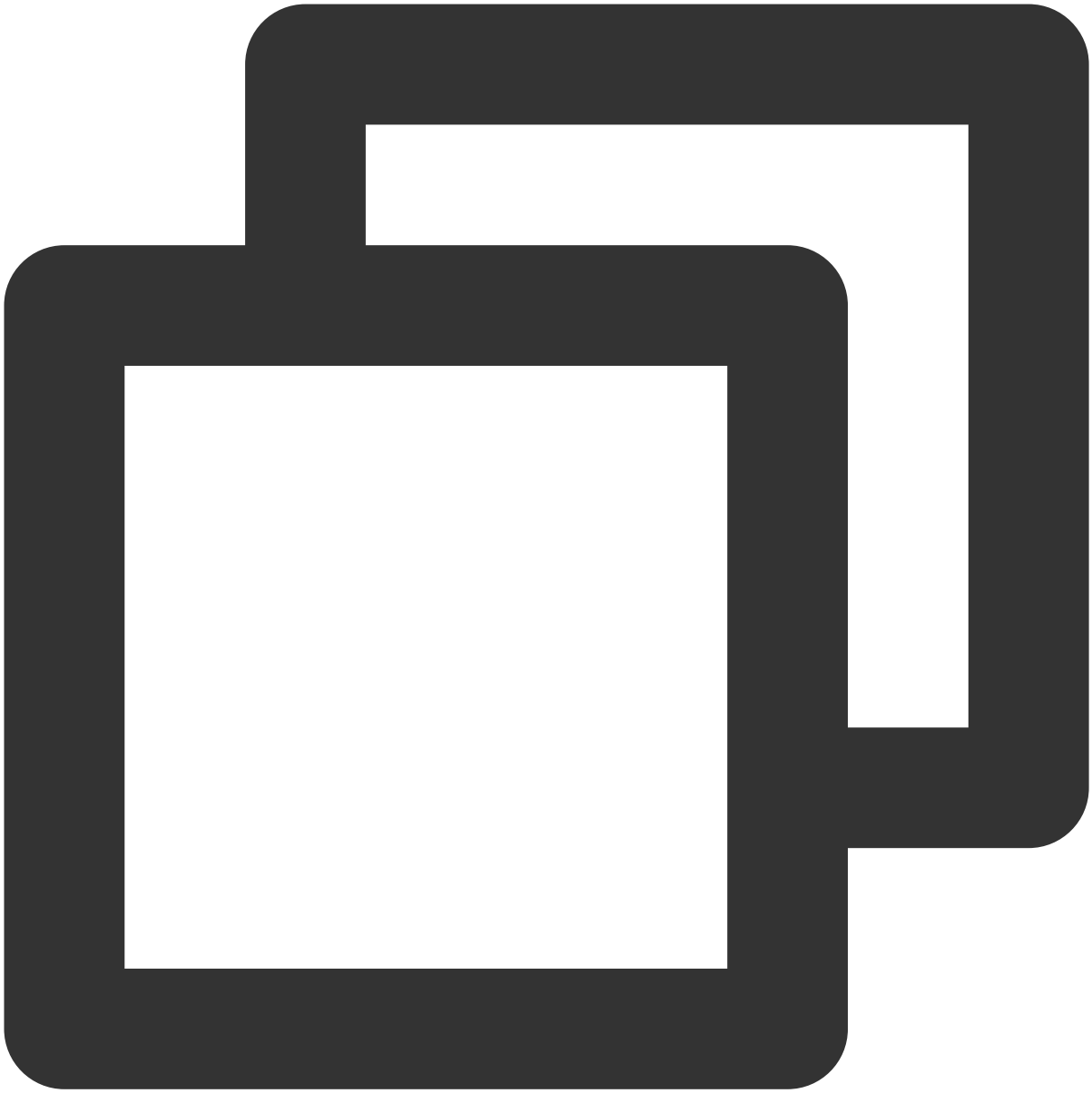
接入云服务器业务层指标

Prometheus 根据监控的不同场景提供了 Counter、Gauge、Historgram、Summary 四种指标类型。Prometheus 社区提供了多种开发语言的 SDK，每种语言的使用方法基本上类似，主要是开发语言语法上的区别，下面主要以 Go 作为例子，使用 Counter 指标类型如何上报自定义监控指标数据。其余指标类型请参见 自定义监控。

Counter

计数类型，数据是单调递增的指标，服务重启之后会重置。可以用 Counter 来监控请求数、异常数、用户登录数和订单数等。

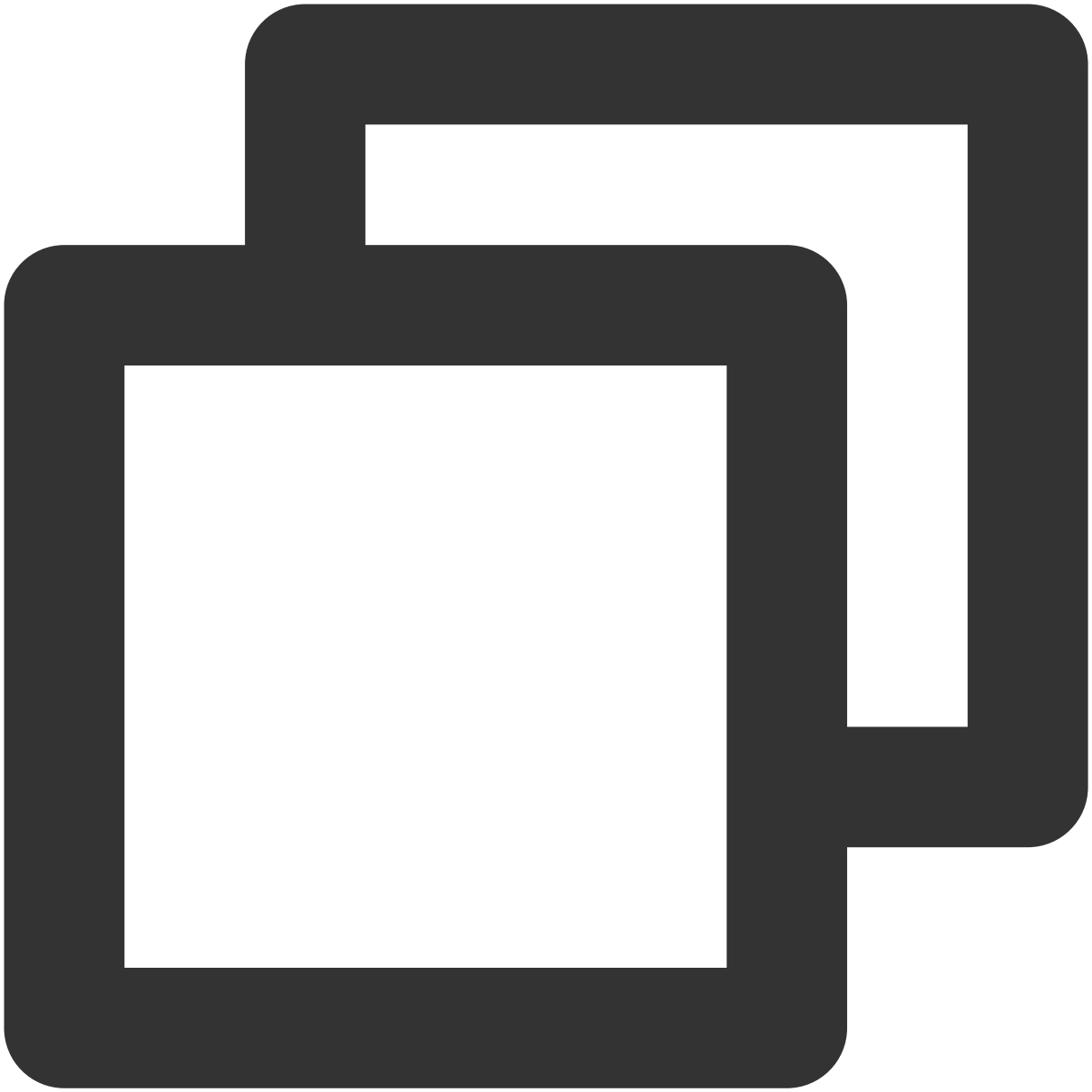
1. 如何通过 Counter 来监控订单数：



```
package order

import (
 "github.com/prometheus/client_golang/prometheus"
 "github.com/prometheus/client_golang/prometheus/promauto"
)

// 定义需要监控 Counter 类型对象
var (
 opsProcessed = promauto.NewCounterVec(prometheus.CounterOpts{
     Name: "order_service_processed_orders_total",
```

```
      Help: "The total number of processed orders",
  }, []string{"status"}) // 处理状态
)

// 订单处理
func makeOrder() {
  opsProcessed.WithLabelValues("success").Inc() // 成功状态
  // opsProcessed.WithLabelValues("fail").Inc() // 失败状态

  // 下单的业务逻辑
}
```

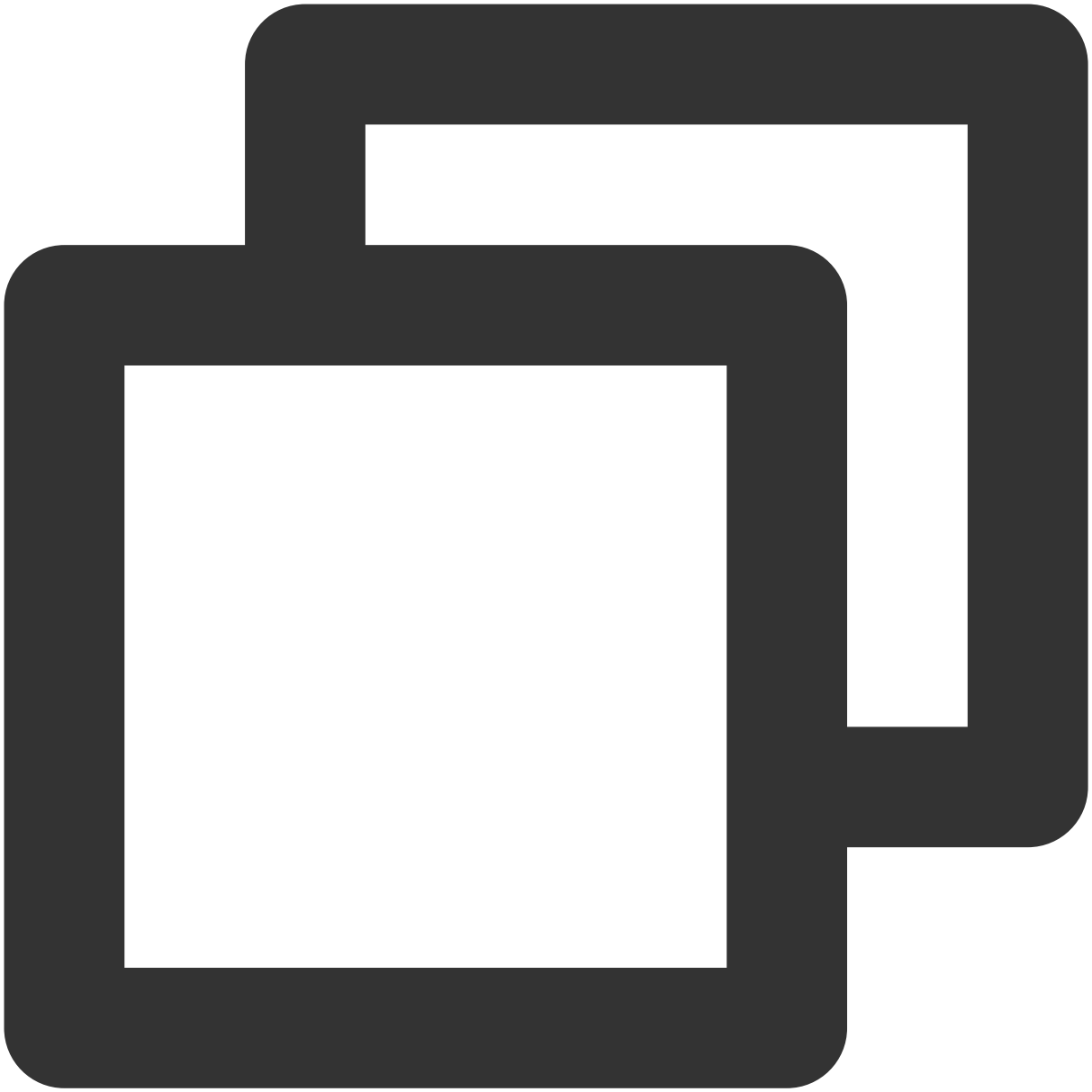
例如，通过 `rate()` 函数获取订单的增长率：



```
rate(order_service_processed_orders_total[5m])
```

2. 暴露 Prometheus 指标：

通过 `promhttp.Handler()` 把监控埋点数据暴露到 HTTP 服务上。

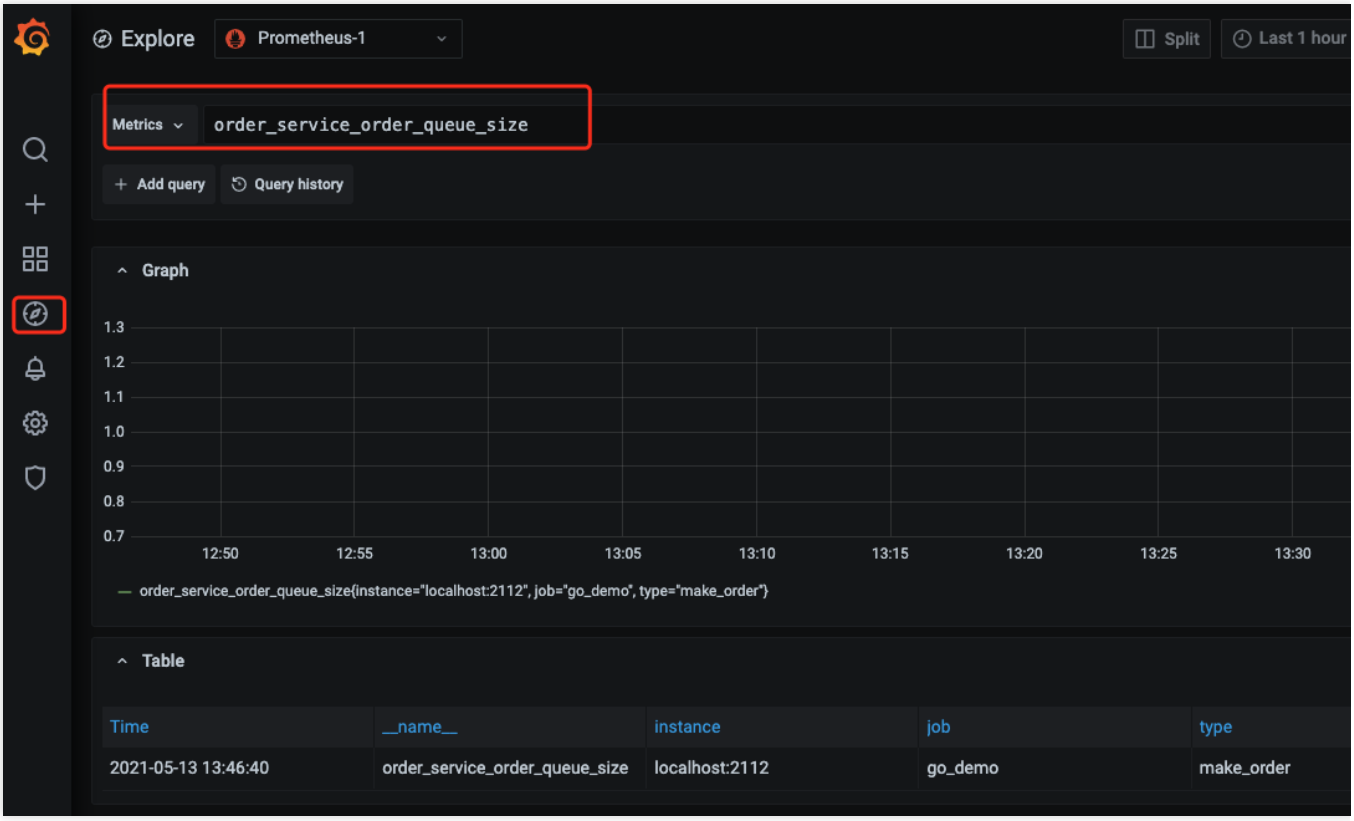


```
package main

import (
 "net/http"

 "github.com/prometheus/client_golang/prometheus/promhttp"
)

func main() {
     // 业务代码
```

```
        // 把 Prometheus 指标暴露在 HTTP 服务上
        http.Handle("/metrics", promhttp.Handler())

        // 业务代码
}
```

3. 采集数据：

完成相关业务自定义监控埋点之后，应用发布，即可通过 Prometheus 来抓取监控指标数据。采集完成后，等待数分钟，您即可在 Prometheus 监控服务集成的 Grafana 中查看业务指标监控数据。



容器场景监控

最近更新时间：2024-01-29 15:55:14

实践背景

众所周知，Prometheus 是容器场景的最佳监控工具，但自建 Prometheus 对于运维人力有限的中小型企业而言，成本太高；对于业务发展快速的大企业又容易出现性能瓶颈。因而使用云上托管 Prometheus 已成为越来越多上云企业的第一选择。那我们该如何使用托管 Prometheus 监控 腾讯云容器服务(TKE) 。

实践步骤

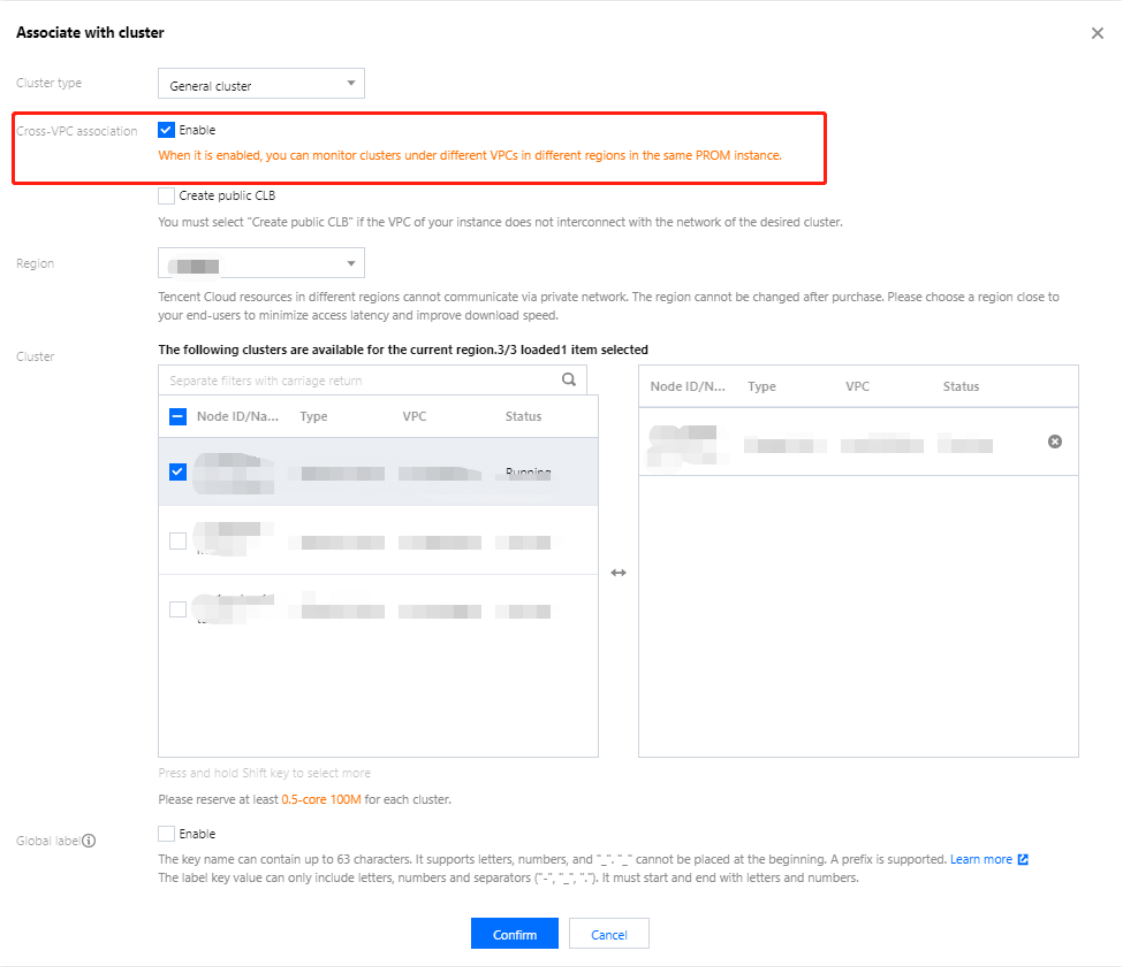
步骤1：购买实例

1. 登录 Prometheus 监控服务控制台。
2. 单击 **新建**，根据您的需求选择购买地域、存储时长并选择需要关联的 Grafana 实例，若无 Grafana 实例，可参见 相关操作指引 创建（需创建实例并完成购买）。
3. 完成后，单击**立即购买**即可。详细计费规则请查看 按量付费说明。

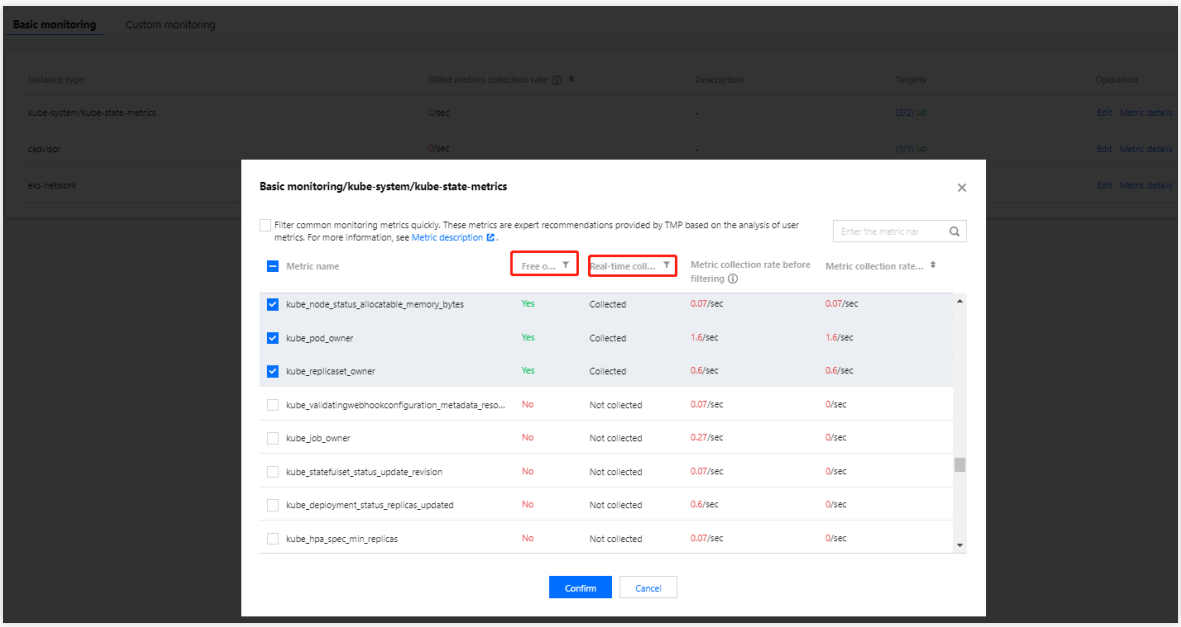
步骤2：集成容器服务

1. 新建完实例后，在实例列表中单击对应的实例 ID/名称，进入实例详情页面。
2. 在左侧菜单栏中单击 **集成容器服务 > 关联集群**。
3. 在弹窗中选择需要关联的集群，共支持接入4种集群类型（标准集群、弹性集群、注册集群、边缘集群），同时支持集群跨 VPC，不同 VPC 网络不互通的情况需要勾选创建公网 CLB。

版权所有：腾讯云计算（北京）有限责任公司

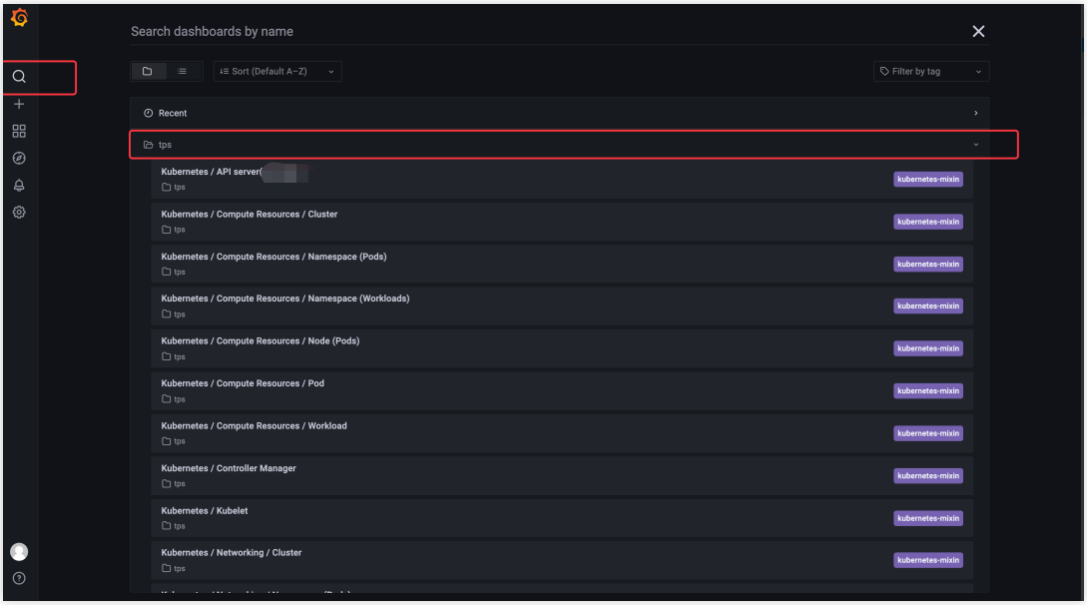


4. 关联集群后，可以在集群监控 > 数据采集配置页面手动配置指标采集，查看默认采集的免费基础指标，同时可以调整新增或减少采集指标。

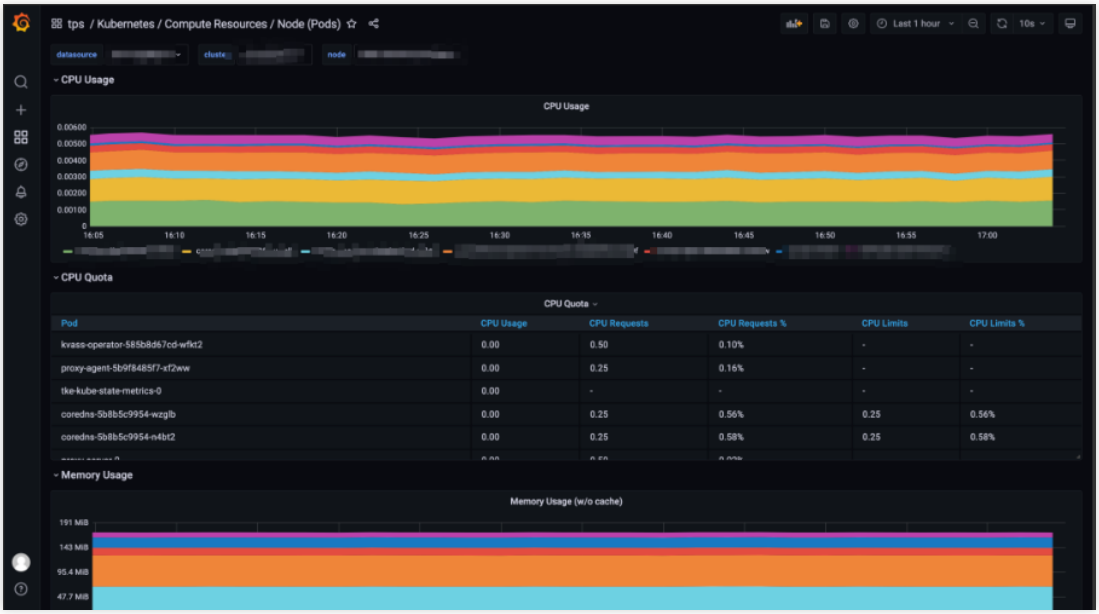


步骤3：Grafana 查看监控数据

1. 在实例列表中点击实例名称右侧的 Grafana 图标，进入 Grafana 服务平台。
2. 在 Grafana 服务平台 > Dashboard 搜索列表，默认预设了容器相关的监控面板，单击某个面板名称。

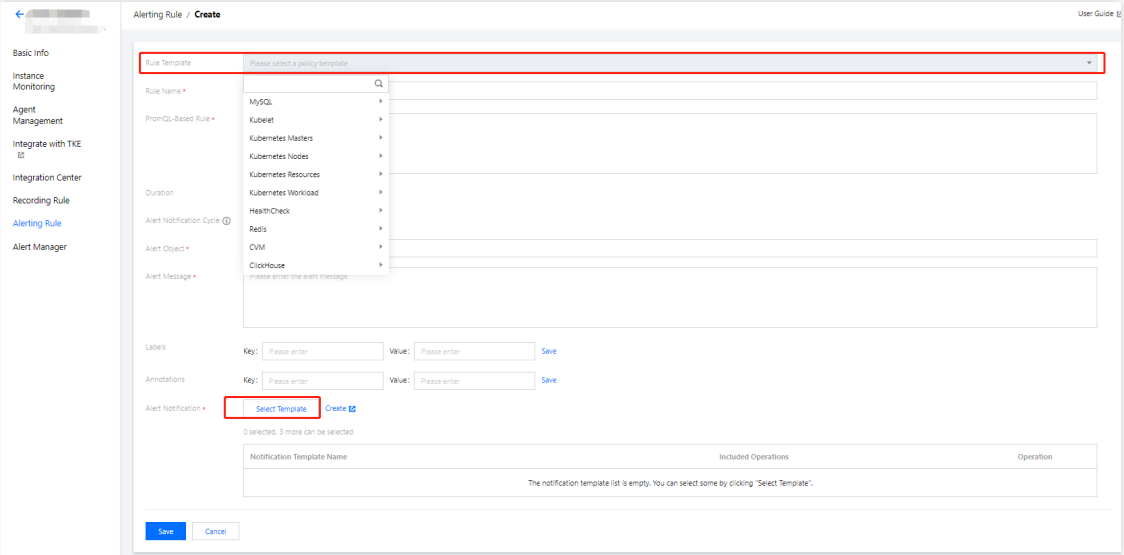


进入面板页面，可以查看预设好的监控数据图表。



步骤4：配置告警策略

在 Prometheus 实例详情页，点击告警策略，可以选择预设的模板类型，无需手动配置，告警通知可选择腾讯云可观测平台已有的通知模板，从而实现快速配置告警。


Alerting Rule / Create
User Guide
Basic Info
Instance Monitoring
Agent Management
Integrate with TKE
Integration Center
Recording Rule
Alerting Rule
Alert Manager
Rule Template
Please select a policy template
Rule Name *
PromQL-Based Rule *
Duration
Alert Notification Cycle
Alert Object *
Alert Message *
Please enter the alert message
MySQL
Kubelet
Kubernetes Masters
Kubernetes Nodes
Kubernetes Resources
Kubernetes Workload
HealthCheck
Redis
CVM
ClickHouse
Labels
Key:
Value:
Save
Annotations
Alert Notification *
Select Template
Create
0 selected. 3 more can be selected
Notification Template Name
Included Operations
Operation
The notification template list is empty. You can select some by clicking "Select Template".
Save
Cancel

TKE Serverless 集群如何放通外网

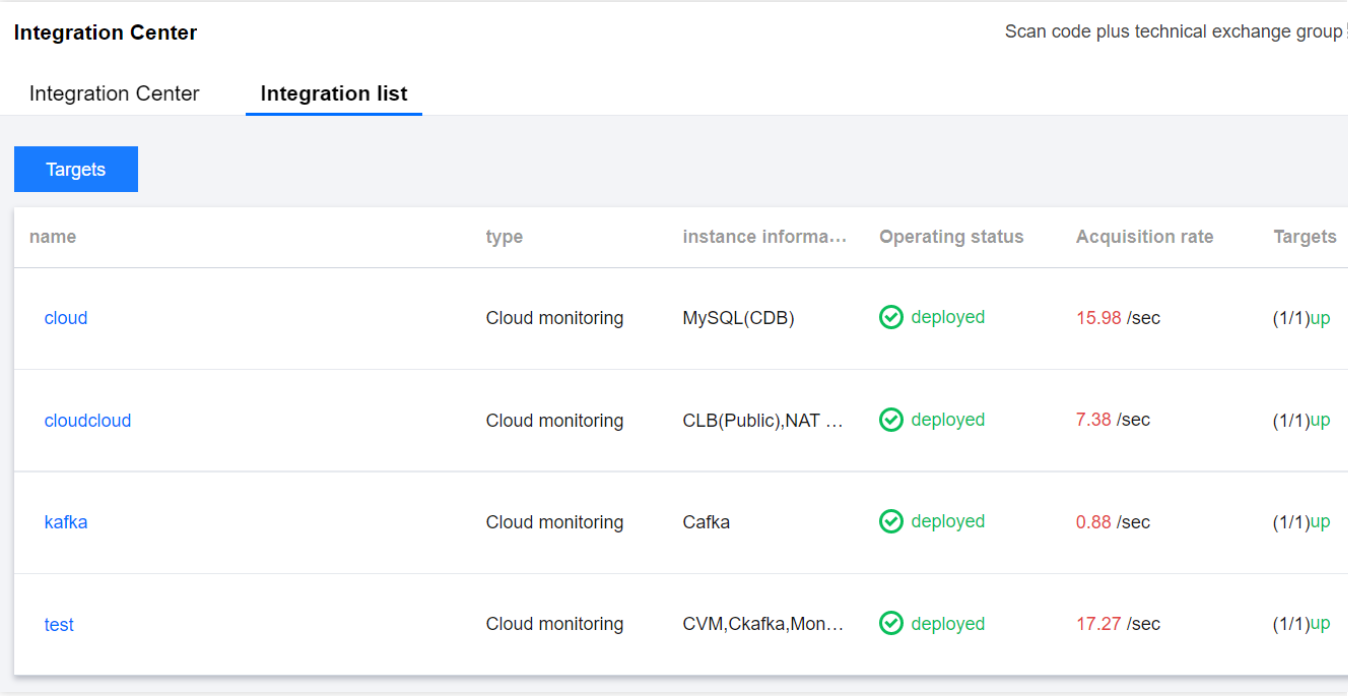
最近更新时间：2024-01-29 15:55:14

操作背景

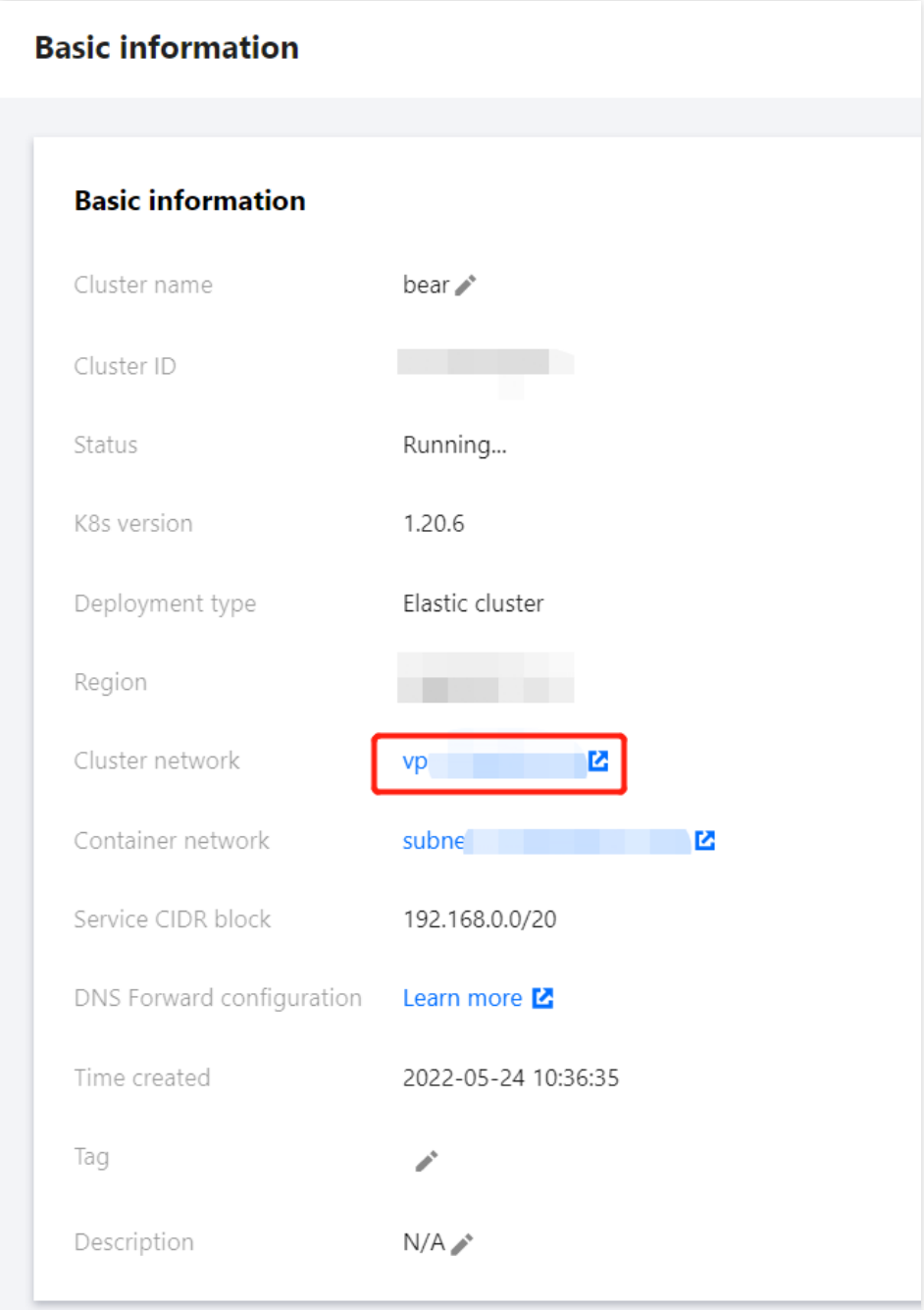
Prometheus 监控服务的集成了云监控服务，当您在新建集成时选择对象存储 COS，由于 COS 不支持内网访问，需要用户对集成云监控 Exporter 所在 Serverless 容器服务（TKE Serverless）开通外网访问。

操作步骤

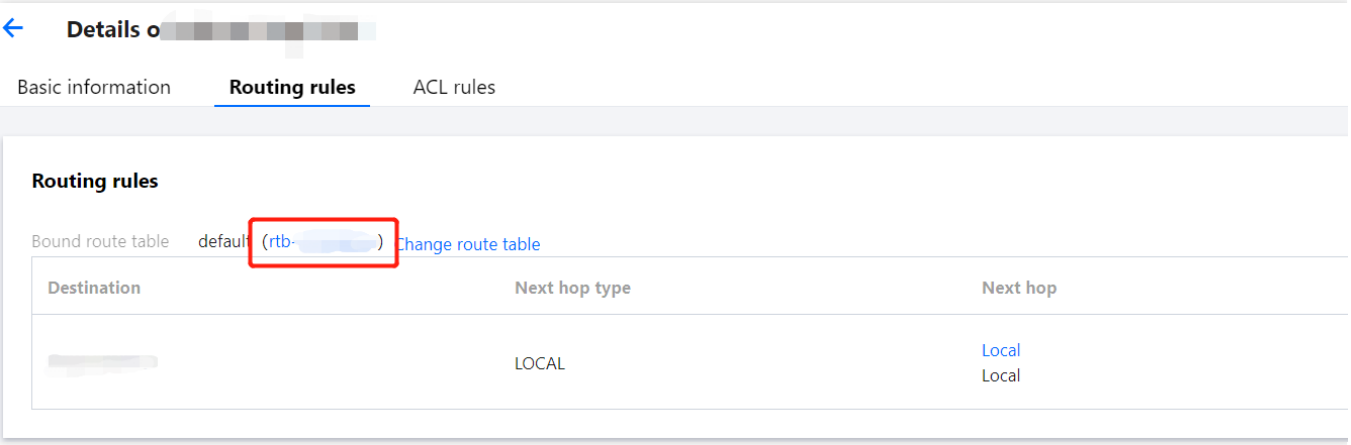
1. 登录 Prometheus 监控服务控制台。
2. 单击对应的实例，进入实例管理页，单击**集成中心** > **集成列表**。
3. 在**集成列表**的**操作**列中选择类型为云监控的一列，单击**日志**进入 EKS 集群。



4. 在顶部的菜单栏，切换到 Pod 管理界面，单击实例名称名字，进入集群界面。
5. 在**基本信息**页面，单击**容器网络**。



6. 在容器网络顶部菜单，切换为**路由策略**栏。单击路由表链接（列表上的 rtb-xxx）进入路由表界面。

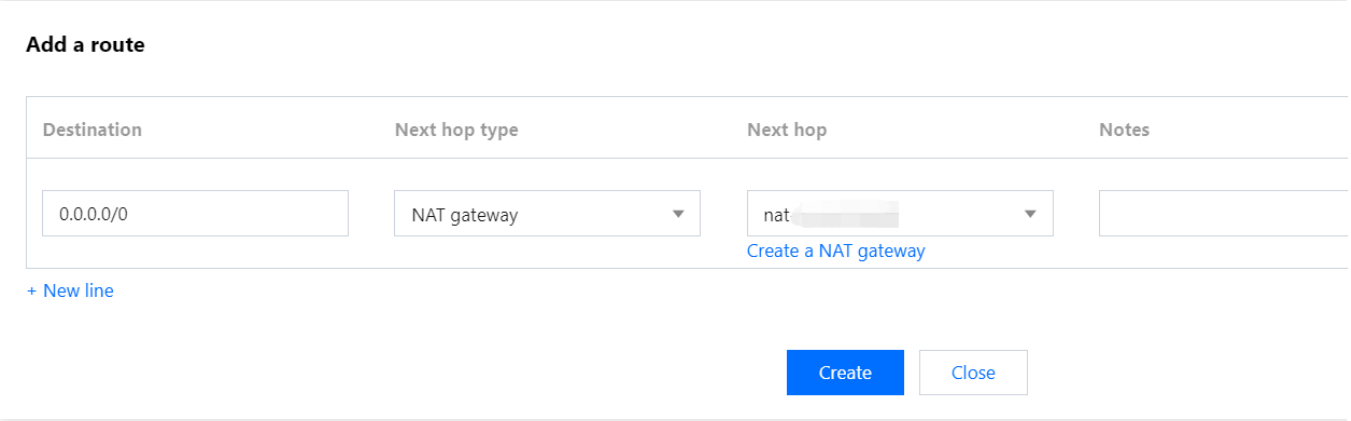


7. 在路由表界面单击**新增路由策略**。

目的端：填写 0.0.0.0/0。

下一跳类型：选择 NAT 网关。

下一跳：选择对应的网关，若无网关，可参见 NAT 网关 指引进行创建 。



8. 填写完后，单击**创建**。创建成功后表示 TKE Serverless 开通外网访问。

Prometheus 监控服务如何接入本地 Grafana

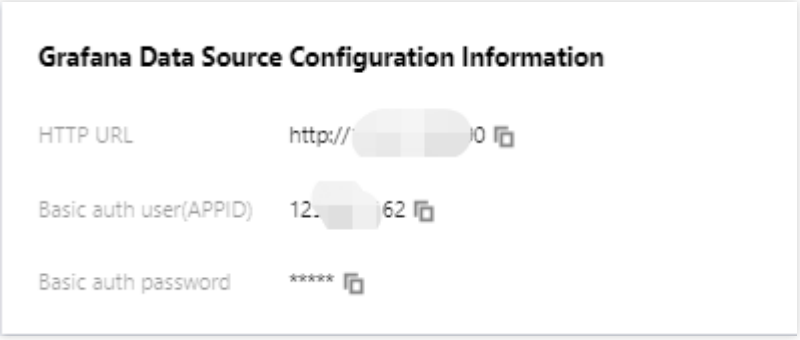
最近更新时间：2024-01-29 15:55:14

若您需要在本地的 Grafana 系统中查看 Prometheus 监控服务相关数据，可以利用 Prometheus 监控服务提供的 HTTP API 来实现。本文介绍如何将 Prometheus 监控服务数据接入本地 Grafana 的实现方法。

操作步骤

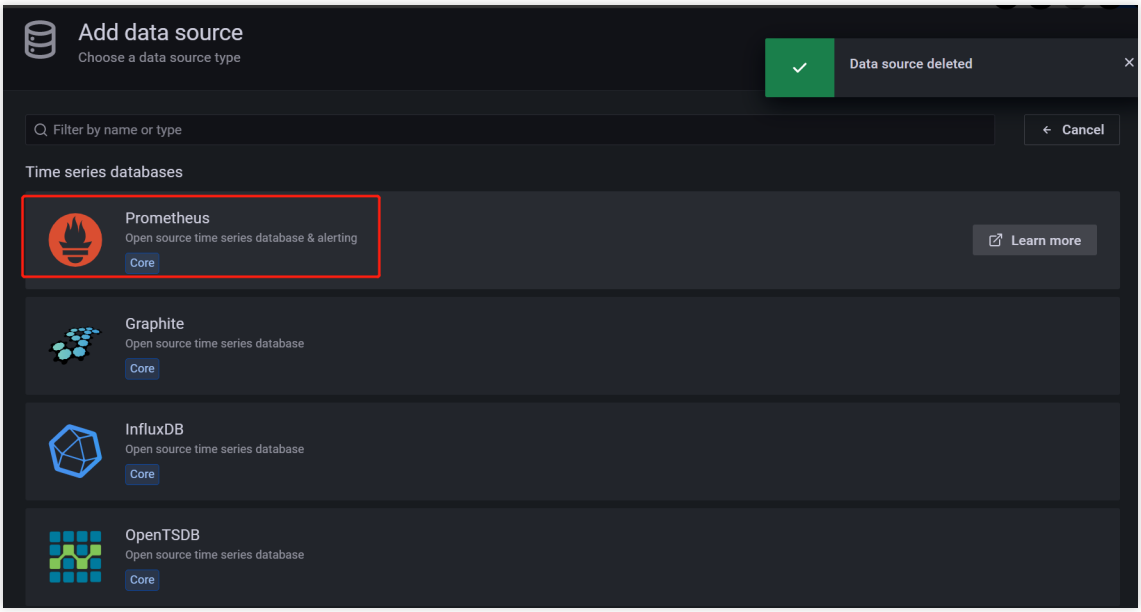
步骤1：获取 Prometheus 监控服务提供的 HTTP API

1. 登录 Prometheus 服务监控控制台
2. 单击对应的按量付费实例，进入 Prometheus 基本信息页。
3. 在服务地址模块获取 HTTP API 地址。若您需要提高 Grafana 数据读取的安全性，可获取 Prometheus 实例的鉴权 Token，按照步骤2指引填入即可。



步骤2：在本地 Grafana 添加数据源

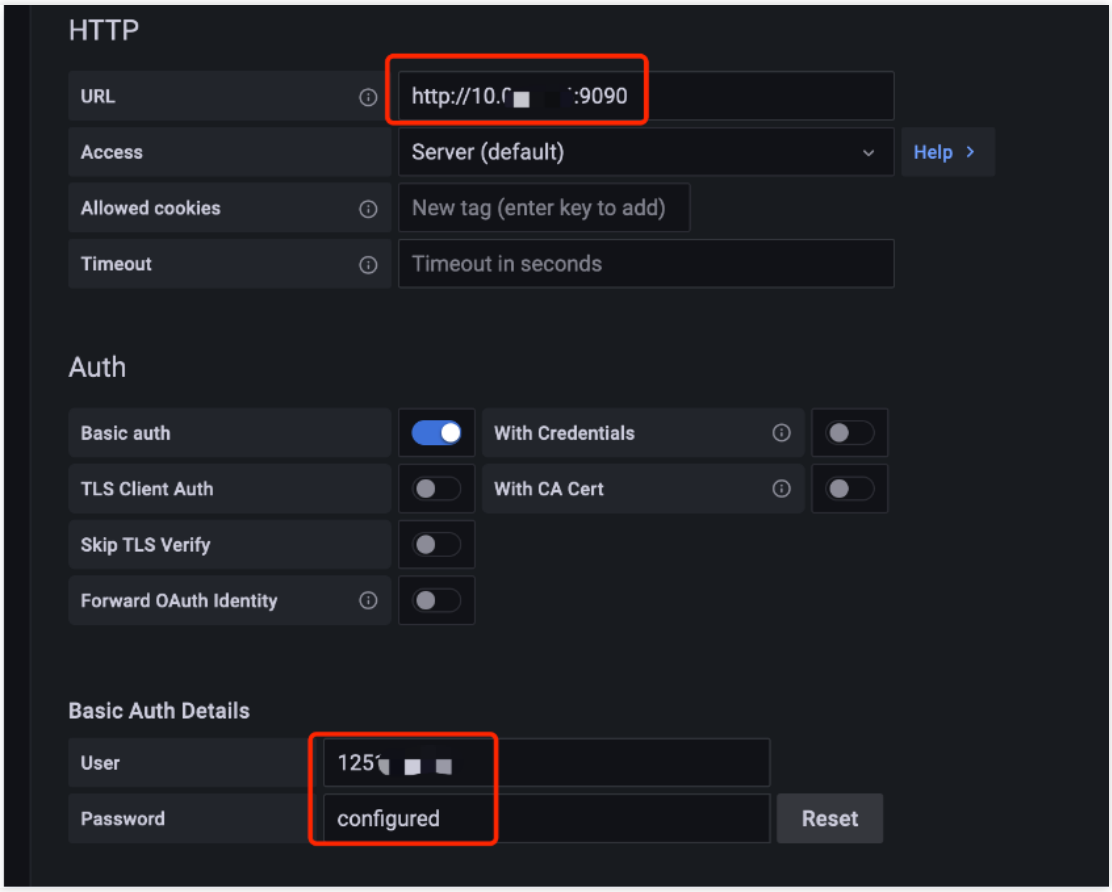
1. 以**管理员账号**登录本地 Grafana 系统。
2. 在左侧导航栏中选择 **Configuration > Data Sources**（非管理员无法查看该菜单）。
3. 在 Data Sources页中单击 **Add data source**。
4. 在 Add data source 页面中选择 **Prometheus**。



5. 在 Settings 页签的 Name 字段输入自定义的名称，在 URL 字段中粘贴步骤1：「Grafana 数据源配置信息」中的 **HTTP API**。

6. 在 Auth 区域开启 Basic Auth 开关，设置 User 为「Grafana 数据源配置信息」中的 **Basic auth user**，设置 Password 为「Grafana 数据源配置信息」中的 **Basic auth password**。

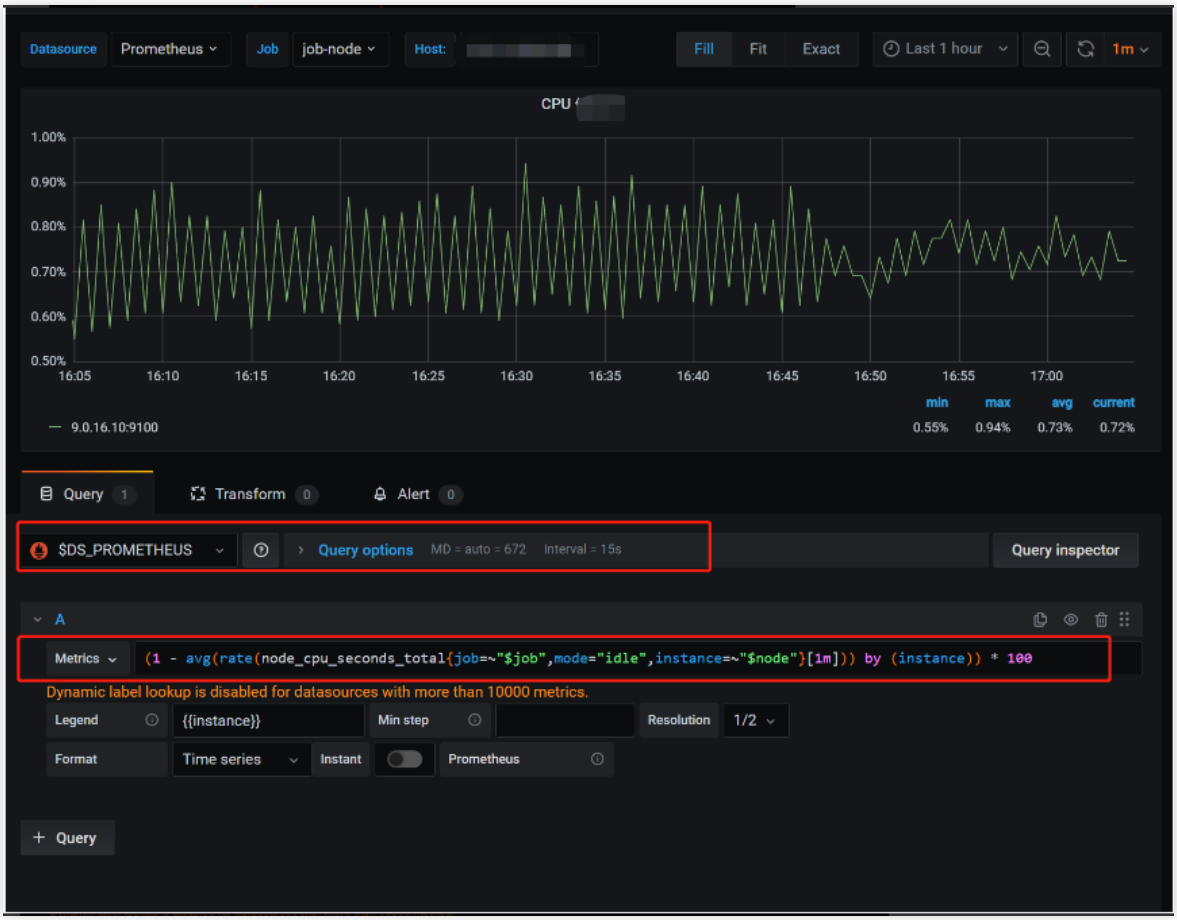
7. 完成后单击 **save & test** 即可。



步骤3：验证是否添加成功

请按照以下步骤验证 Prometheus 监控服务是否成功接入本地 Grafana：

1. 登录您本地 Grafana 系统。
2. 在左侧导航栏中选择**+ > Create**。
3. 在 New dashboard 页面中单击 **Add a new panel** 。
4. 在 Edit Panel 页面的 Query 页面的下拉框中，选择步骤2中所添加的数据源，在 A 区域的 Metrics 字段输入指标名称并按回车。
5. 若能显示出相应指标的图表，则说明操作成功。否则请检查填写的接口地址或 Token 是否正确，以及数据源是否有 Prometheus 监控服务数据。


Datasource
Prometheus
Job
job-node
Host:
Fill
Fit
Exact
Last 1 hour
1m
CPU
1.00%
0.90%
0.80%
0.70%
0.60%
0.50%
16:05
16:10
16:15
16:20
16:25
16:30
16:35
16:40
16:45
16:50
16:55
17:00
min
max
avg
current
9.0.16.10:9100
0.55%
0.94%
0.73%
0.72%
Query 1
Transform 0
Alert 0
$DS_PROMETHEUS
Query options
MD = auto = 672
Interval = 15s
Query inspector
A
Metrics
(1 - avg(rate(node_cpu_seconds_total{job=~"$job",mode="idle",instance=~"$node"}[1m])) by (instance)) * 100
Dynamic label lookup is disabled for datasources with more than 10000 metrics.
Legend
{{instance}}
Min step
Resolution
1/2
Format
Time series
Instant
Prometheus
+ Query