实验 09-Kubernetes Monitor

一、实验目的

1、了解 Prometheus 和 HertzBeat；

2、掌握 Prometheus 的部署；

3、实现 Prometheus 监控 K8s 集群；

4、实现 HertzBeat 监控 K8s 集群。

二、实验学时

2 学时

三、实验类型

设计性

四、实验任务

1、完成 Prometheus 和 Grafana 的部署；

2、完成 Prometheus 监控 K8s 主机；

3、完成 Prometheus 监控 K8s 集群；

4、完成 HertzBeat 监控 K8s 集群；

5、完成监控数据可视化。

五、实验环境

1、硬件

本实验基于实验教学中心网络运维实验室服务器集群开展，每个实验小组分配集群中的 1 台物理服务器作为实验基础平台，提供云计算资源。每个人配备计算机 1 台。（学生可根据自身情况使用个人计算机）。

2、软件

Windows 操作系统，或 MacOS 操作系统。

安装最新版本的浏览器，建议使用 Edge、Chrome 等。

3、网络

计算机使用无线网络接入局域网，能够访问实验教学中心网络运维实验室服务器集群，并支持对互联网的访问。

4、工具

无。

六、实验内容步骤

本实验需要 VM 1 台，配置信息如表 8-1 所示。

表 8-1 虚拟机配置规划表

序号	虚拟机配置	操作系统配置
1	虚拟机名称：Cloud-K8s-Monitor CPU：2 核 内存：2GB 硬盘：40GB（系统盘） + 100GB（数据存储） 网卡：Cloud-Platform-VM-Network	主机名：Cloud-K8s-Monitor 操作系统：openEuler 24.03 LT IP 地址：172.16.125.107 子网掩码：255.255.255.0 网关：172.16.125.1 DNS：172.16.125.3

1、部署 Prometheus、Grafana 和 Alertmanager

步骤 1：系统环境准备。创建数据目录并挂载、安装 Docker 环境。

Shell

```
#查看磁盘结构
[root@Cloud-K8s-Monitor ~]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                   8:0    0   40G  0 disk
├─sda1                8:1    0    1M  0 part
├─sda2                8:2    0    1G  0 part /boot
└─sda3                8:3    0   19G  0 part
  ├─openeuler-root 253:0    0   17G  0 lvm  /
  └─openeuler-swap 253:1    0    2G  0 lvm  [SWAP]
sdb                   8:16   0  100G  0 disk
sr0                  11:0    1 1024M  0 rom
#格式化/dev/sdb
[root@Cloud-K8s-Monitor ~]# mkfs.ext4 /dev/sdb
mke2fs 1.47.0 (5-Feb-2023)
丢弃设备块：完成
创建含有 26214400 个块（每块 4k）和 6553600 个 inode 的文件系统
文件系统 UUID：1522f744-18d2-450c-a555-40bc8c8ce7e3
超级块的备份存储于下列块：
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

正在分配组表：完成
正在写入 inode表：完成
创建日志（131072 个块）：完成
写入超级块和文件系统账户统计信息：已完成
#创建数据目录并赋予权限
mkdir /data
chmod 777 /data
#进行挂载
mount /dev/sdb /data
#开机自动挂载
echo '/dev/sdb    /data    ext4    defaults    0    0' >> /etc/fstab
#安装docker
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
sed -i 's/\$releasever/8/g' /etc/yum.repos.d/docker-ce.repo
yum install -y docker-ce docker-ce-cli containerd.io
cat > /etc/docker/daemon.json <<EOF
{
  "registry-mirrors": [
    "https://registry.cn-hangzhou.aliyuncs.com",
```

```
    "https://hub.xdark.top",
    "https://hub.littlediary.cn",
    "https://dockerpull.org",
    "https://hub.crdz.gq",
    "https://docker.1panel.live",
    "https://docker.mirrors.ustc.edu.cn",
    "https://docker.m.daocloud.io",
    "https://noohub.ru",
    "https://huecker.io",
    "https://dockerhub.timeweb.cloud",
    "https://docker.1panel.dev",
    "https://docker.unsee.tech",
    "https://docker.1panel.live"
  ]
}
EOF
systemctl start docker
systemctl enable docker
```

步骤 2：编写 yaml 文件，部署 Prometheus、Grafana、Alertmanager

（1）分别创建 `/data/prometheus`、`/data/alertmanager`、`/data/grafana` 目录作为 `prometheus`、`alertmanager`、`grafana` 服务的数据目录，并分别设置目录 `/data/prometheus` 和 `/data/grafana` 的权限为 777。

Shell

```
mkdir -p /data/{prometheus,alertmanager,grafana}
chmod 777 /data/prometheus
chmod 777 /data/grafana
```

（2）分别创建 `/etc/prometheus`、`/etc/alertmanager`、`/etc/grafana` 目录，并在 `/etc/prometheus` 目录下创建 `prometheus` 服务的配置文件 `prometheus.yml`，在 `/etc/alertmanager` 目录下创建 `alertmanager` 服务的配置文件 `config.yml`，在 `/etc/grafana` 目录下创建 `grafana` 服务的配置文件 `config.monitoring`。

Shell

```
mkdir -p /etc/{prometheus,alertmanager,grafana}
vi /etc/prometheus/prometheus.yml
----------/etc/prometheus/prometheus.yml----------
# 全局参数
global:
  scrape_interval: 15s     # 设定抓取数据的周期，默认为1min
  scrape_timeout: 15s      # 设定抓取数据的超时时间，默认为10s
  evaluation_interval: 30s # 设定更新rules文件的周期，默认为1min
  external_labels:         # 额外的属性，会添加到拉取得数据并存到数据库中
    monitor: 'codelab-monitor'

scrape_configs:
  # 监控自身
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  # 监控alertmanager
  - job_name: "alertmanager"
    static_configs:
      - targets: ["172.16.125.107:9093"]
----------/etc/prometheus/prometheus.yml----------

vi /etc/alertmanager/config.yml
----------/etc/alertmanager/config.yml----------
# 全局配置，配置邮件服务器
global:
  smtp_from: '' # 配置发件人邮箱地址 可配置自己的邮箱
  smtp_smarthost: 'smtp.163.com:25' # SMTP 服务器地址与端口，也可配置qq邮箱。
  smtp_auth_username: '' # 配置发件人邮箱地址 可配置自己的邮箱
  smtp_auth_password: '' # 配置发件人邮箱密码 可配置自己邮箱的密码
  smtp_require_tls: false
  smtp_hello: '163.com'

# 路由配置
route:
  group_by: ['alertname']     # 根据告警名称分组
  group_wait: 5s              # 等待时间，等待同一组告警的时间
  group_interval: 5s          # 每组告警之间的间隔时间
  repeat_interval: 2m         # 重复发送间隔时间
  receiver: 'email'           # 默认接收者设置为'email'，即下方定义的'emai
l'接收者

```

```
# 接收者配置
receivers:
- name: 'email'                # 接收者名称
  email_configs:
  - to: '3183748324@qq.com'  # 收件人邮箱地址
    send_resolved: true        # 是否发送已解决的告警
----------/etc/alertmanager/config.yml----------

vi /etc/grafana/config.monitoring
----------/etc/grafana/config.monitoring----------
# grafana管理界面的登录用户密码，用户名是admin
GF_SECURITY_ADMIN_PASSWORD=admin#123456
# grafana管理界面是否允许注册，默认不允许
GF_USERS_ALLOW_SIGN_UP=false
----------/etc/grafana/config.monitoring----------
```

（3）编写 docker compose 配置文件，创建 Prometheus +Grafana + Alertmanager 容器，docker compose 文件内容如下：

Shell

```
vi docker-compose.yml
-----------------docker-compose.yml-----------------
version: '3.3'
services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    restart: always
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /etc/prometheus/:/etc/prometheus/
      - /data/prometheus:/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      - '--web.console.libraries=/usr/share/prometheus/console_librarie
s'
      - '--web.console.templates=/usr/share/prometheus/consoles'
      #热加载配置
      - '--web.enable-lifecycle'
      #api配置
      #- '--web.enable-admin-api'
      #历史数据最大保留时间，默认15天
      - '--storage.tsdb.retention.time=720d'
    networks:
      net:
        ipv4_address: 172.20.210.10
    links:
      - alertmanager
      - cadvisor
    expose:
      - '9090'
    ports:
      - 9090:9090
    depends_on:
      - cadvisor

  alertmanager:
    image: prom/alertmanager:v0.25.0
    container_name: alertmanager
    restart: always
    volumes:
```

```
42      - /etc/localtime:/etc/localtime:ro
43      - /etc/alertmanager/:/etc/alertmanager/
44      - /data/alertmanager:/alertmanager
45    command:
46      - '--config.file=/etc/alertmanager/config.yml'
47      - '--storage.path=/alertmanager'
48    networks:
49      net:
50        ipv4_address: 172.20.210.11
51    expose:
52      - '9093'
53    ports:
54      - 9093:9093
55
56  cadvisor:
57    image: google/cadvisor:latest
58    container_name: cadvisor
59    restart: always
60    volumes:
61      - /etc/localtime:/etc/localtime:ro
62      - /:/rootfs:ro
63      - /var/run:/var/run:rw
64      - /sys:/sys:ro
65      - /var/lib/docker/:/var/lib/docker:ro
66    networks:
67      net:
68        ipv4_address: 172.20.210.12
69    expose:
70      - '8080'
71    ports:
72      - '8080:8080'
73
74  grafana:
75    image: grafana/grafana:latest
76    container_name: grafana
77    restart: always
78    volumes:
79      - /etc/localtime:/etc/localtime:ro
80      - /data/grafana:/var/lib/grafana
81      - /etc/grafana/provisioning/:/etc/grafana/provisioning/
82    env_file:
83      - /etc/grafana/config.monitoring
84    networks:
85      net:
```

```
 86         ipv4_address: 172.20.210.13
 87     links:
 88       - prometheus
 89     ports:
 90       - 3000:3000
 91     depends_on:
 92       - prometheus
 93
 94 networks:
 95  net:
 96   driver: bridge
 97   ipam:
 98    config:
 99     - subnet: 172.20.210.0/24
100 -----------------docker-compose.yml-----------------
```

（4）执行 docker compose，创建 Prometheus +Grafana + Alertmanager 容器服务。

Shell

```
1 docker compose -f docker-compose.yml up -d
```

安装完成后，用浏览器分别访问 Prometheus：http://IP:9090，和 Grafana：http://IP:3000，如图 1-1 和 1-2 所示。

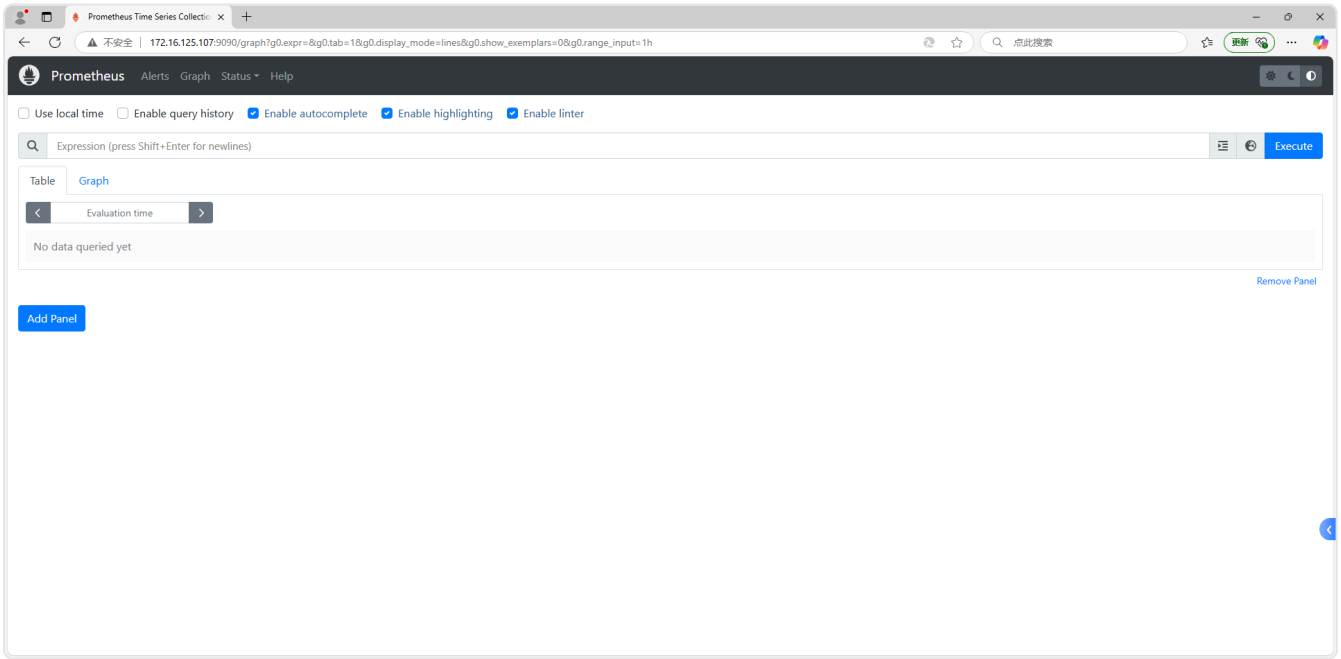

图 1-1 访问Prometheus

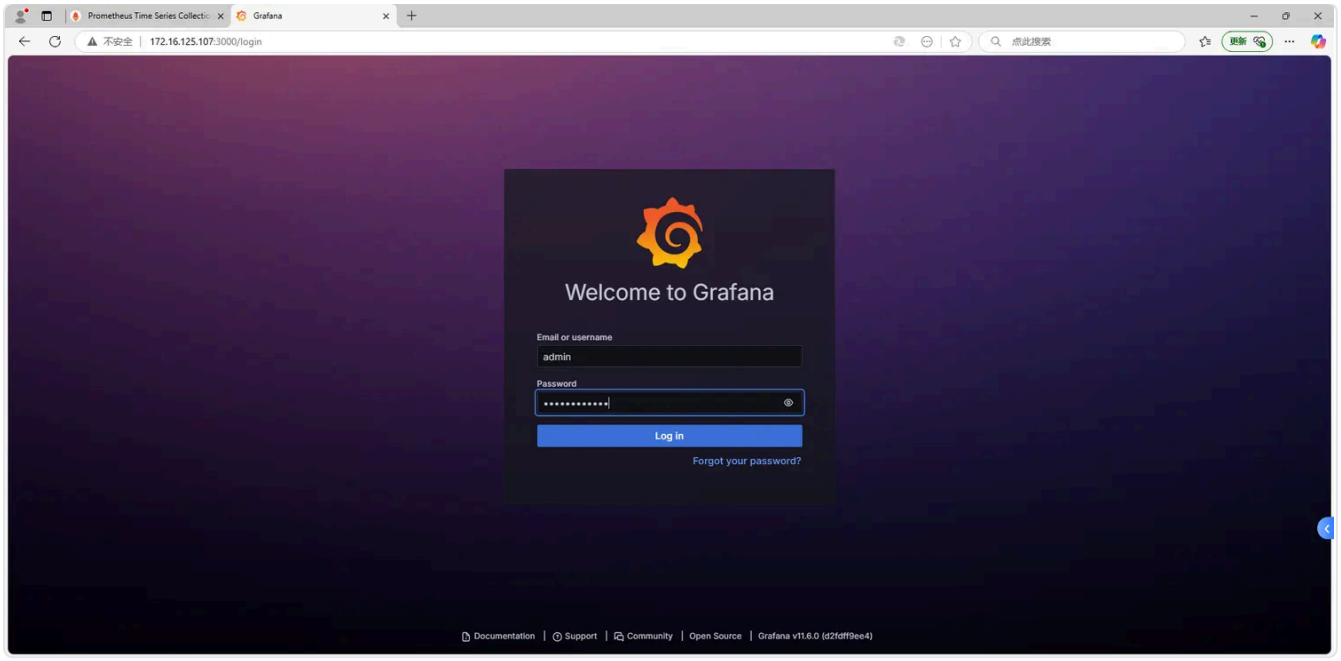

图 1-2 访问 Grafana

2、在 K8s 上部署采集器

（1）部署 node-exporter

Shell

```
#创建命名空间devops
[root@k8s-master ~]# kubectl create namespace devops
#编写yaml文件，部署node_export
[root@k8s-master ~]# vi node_export.yaml
---------------node_export.yaml-----------------
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: node-exporter
  namespace: monitoring
  labels:
    k8s-app: node-exporter
spec:
  selector:
    matchLabels:
        k8s-app: node-exporter
  template:
    metadata:
      labels:
        k8s-app: node-exporter
    spec:
      tolerations:
      - key: node-role.kubernetes.io/control-plane
        operator: Exists
        effect: NoSchedule
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      containers:
      - image: bitnami/node-exporter:1.7.0
        imagePullPolicy: IfNotPresent
        name: prometheus-node-exporter
        ports:
        - containerPort: 9100
          hostPort: 9100
          protocol: TCP
          name: metrics
        volumeMounts:
        - mountPath: /host/proc
          name: proc
        - mountPath: /host/sys
          name: sys
```

```
        - mountPath: /host
          name: rootfs
        args:
        - --path.procfs=/host/proc
        - --path.sysfs=/host/sys
        - --path.rootfs=/host
      volumes:
        - name: proc
          hostPath:
            path: /proc
        - name: sys
          hostPath:
            path: /sys
        - name: rootfs
          hostPath:
            path: /
      hostNetwork: true # 使用宿主机网络和PID
      hostPID: true
---------------node_export.yaml-----------------
[root@k8s-master ~]# kubectl apply -f node_export.yaml

```

（2）部署 kube-state-metrics

Shell

```
vi kube-state-metrics.yaml
----------kube-state-metrics.yaml------------
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kube-state-metrics
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kube-state-metrics
  template:
    metadata:
      labels:
        app: kube-state-metrics
    spec:
      serviceAccountName: kube-state-metrics
      containers:
      - name: kube-state-metrics
        image: registry.cn-hangzhou.aliyuncs.com/zhangshijie/kube-state-metrics:v2.6.0
        ports:
        - containerPort: 8080

---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: kube-state-metrics
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kube-state-metrics
rules:
- apiGroups: [""]
  resources: ["nodes", "pods", "services", "resourcequotas", "replicationcontrollers", "limitranges", "persistentvolumeclaims", "persistentvolumes", "namespaces", "endpoints"]
  verbs: ["list", "watch"]
```

```
40 - apiGroups: ["extensions"]
41   resources: ["daemonsets", "deployments", "replicasets"]
42   verbs: ["list", "watch"]
43 - apiGroups: ["apps"]
44   resources: ["statefulsets"]
45   verbs: ["list", "watch"]
46 - apiGroups: ["batch"]
47   resources: ["cronjobs", "jobs"]
48   verbs: ["list", "watch"]
49 - apiGroups: ["autoscaling"]
50   resources: ["horizontalpodautoscalers"]
51   verbs: ["list", "watch"]
52 ---
53 apiVersion: rbac.authorization.k8s.io/v1
54 kind: ClusterRoleBinding
55 metadata:
56   name: kube-state-metrics
57 roleRef:
58   apiGroup: rbac.authorization.k8s.io
59   kind: ClusterRole
60   name: kube-state-metrics
61 subjects:
62 - kind: ServiceAccount
63   name: kube-state-metrics
64   namespace: kube-system
65
66 ---
67 apiVersion: v1
68 kind: Service
69 metadata:
70   annotations:
71     prometheus.io/scrape: 'true'
72   name: kube-state-metrics
73   namespace: kube-system
74   labels:
75     app: kube-state-metrics
76 spec:
77   type: NodePort
78   ports:
79   - name: kube-state-metrics
80     port: 8080
81     targetPort: 8080
82     nodePort: 31888
83     protocol: TCP
```

```
  selector:
    app: kube-state-metrics
----------kube-state-metrics.yaml------------
[root@k8s-master ~]# kubectl apply -f kube-state-metrics.yaml
```

3、修改 Prometheus 配置文件，配置监控对象

Shell

```
#在配置文件prometheus.yml后添加监控任务
vi /etc/prometheus/prometheus.yml
---------------prometheus.yml-----------------
  # 监控主机
  - job_name: "K8sHost"
    static_configs:
      - targets: ["172.16.125.101:9100","172.16.125.102:9100","172.16.125.103:9100","172.16.125.104:9100","172.16.125.105:9100"]
  # 监控K8s
  - job_name: "K8s-Cluster"
    honor_timestamps: true
    metrics_path: /metrics
    scheme: http
    static_configs:
      - targets: ["172.16.125.101:31888"]
    metric_relabel_configs:
      - target_label: cluster
        replacement: K8s-Cluster
------------------------------------------------
#重启prometheus容器
docker restart prometheus
```

使用浏览器访问 Prometheus 查看监控对象。

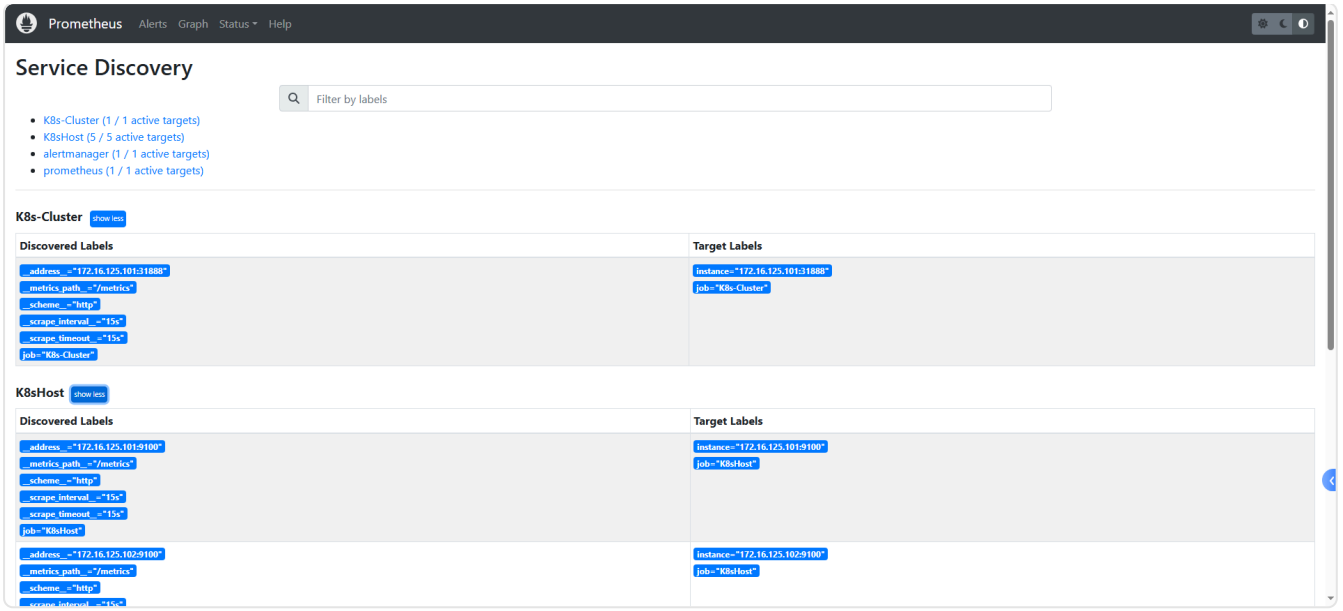

图 9-1 prometheus监控数据

4、在 Grafana 进行数据可视化

步骤 1：添加 Prometheus 数据源

选择左侧菜单“Connections”->“Data sources”，点击右上角【add new data source】按钮，选择 prometheus 类型的数据源进行配置，在这里配置 prometheus 服务器的访问路径：http://172.16.125.107:9090，如图 9-2 所示，其他设置保存默认，最后测试并保存，如图 9-3 所示。

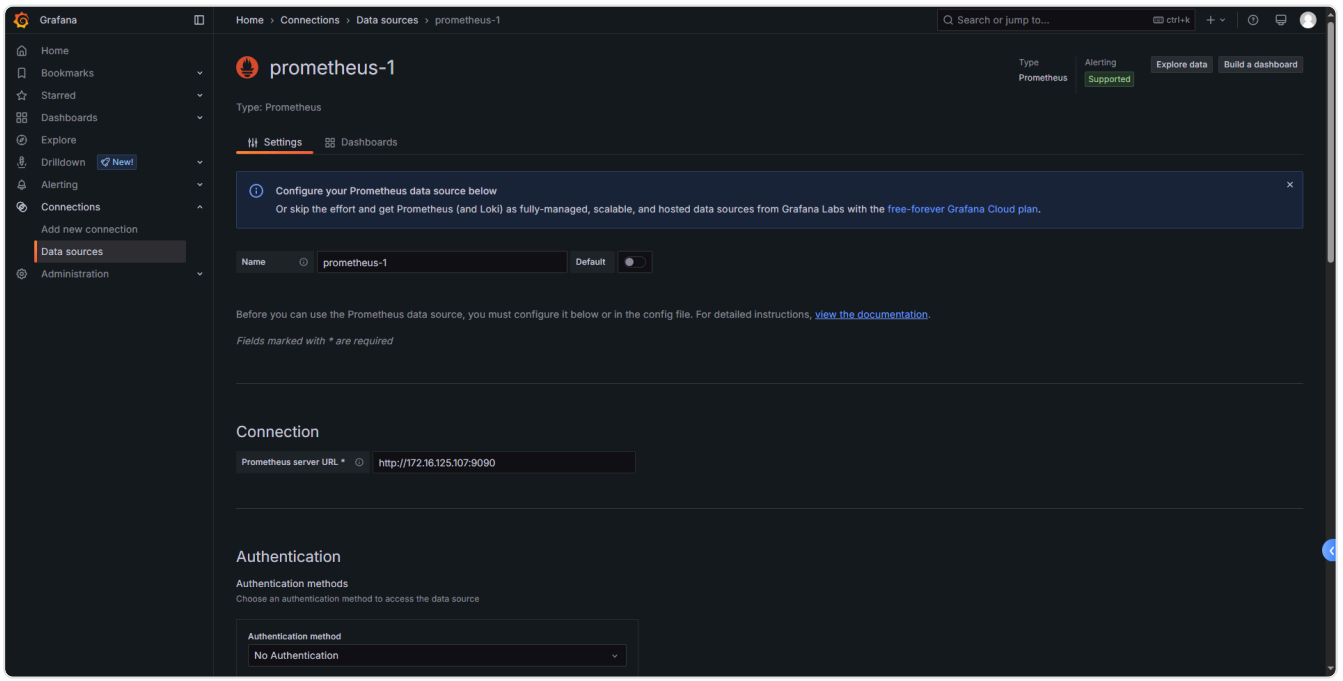

图 9-2 配置数据源

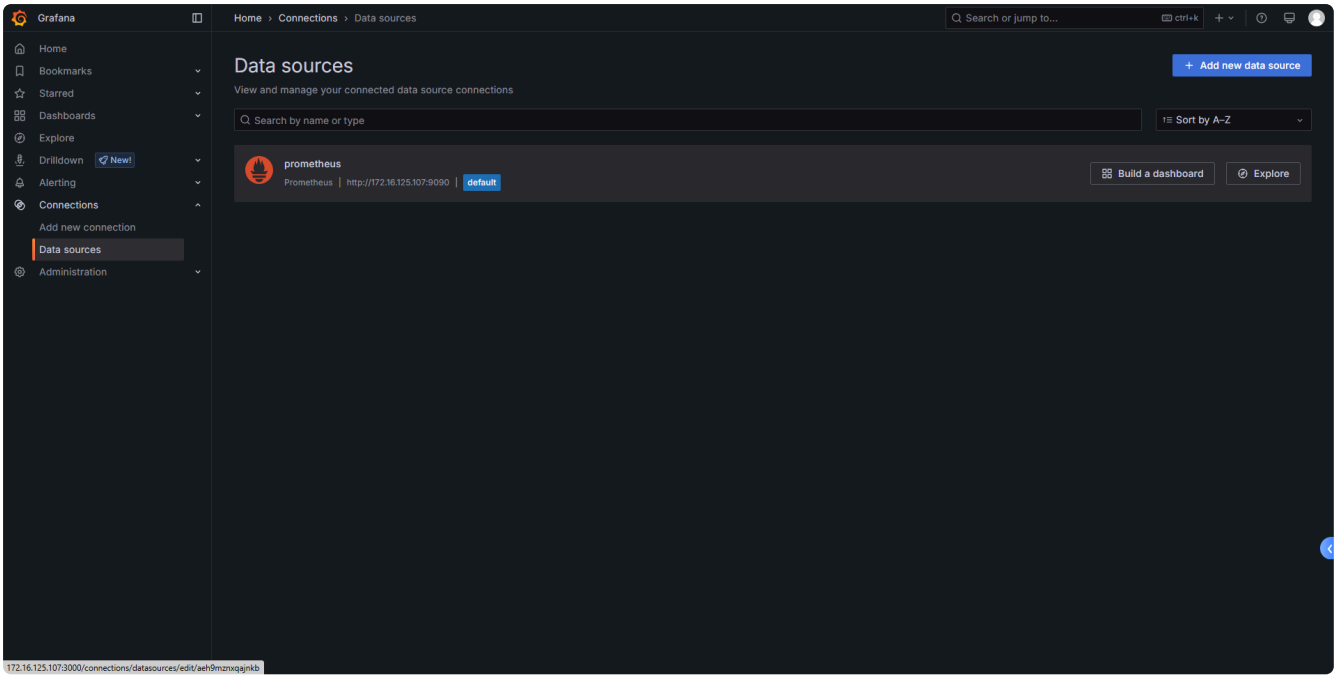

图 9-3 添加 prometheus 数据源

步骤 2：添加仪表盘

（1）选择左侧菜单“Dashboards”->“New”->“Import”，输入仪表盘 ID 为 11074 并载入，如图 9-4、图 9-5 所示。

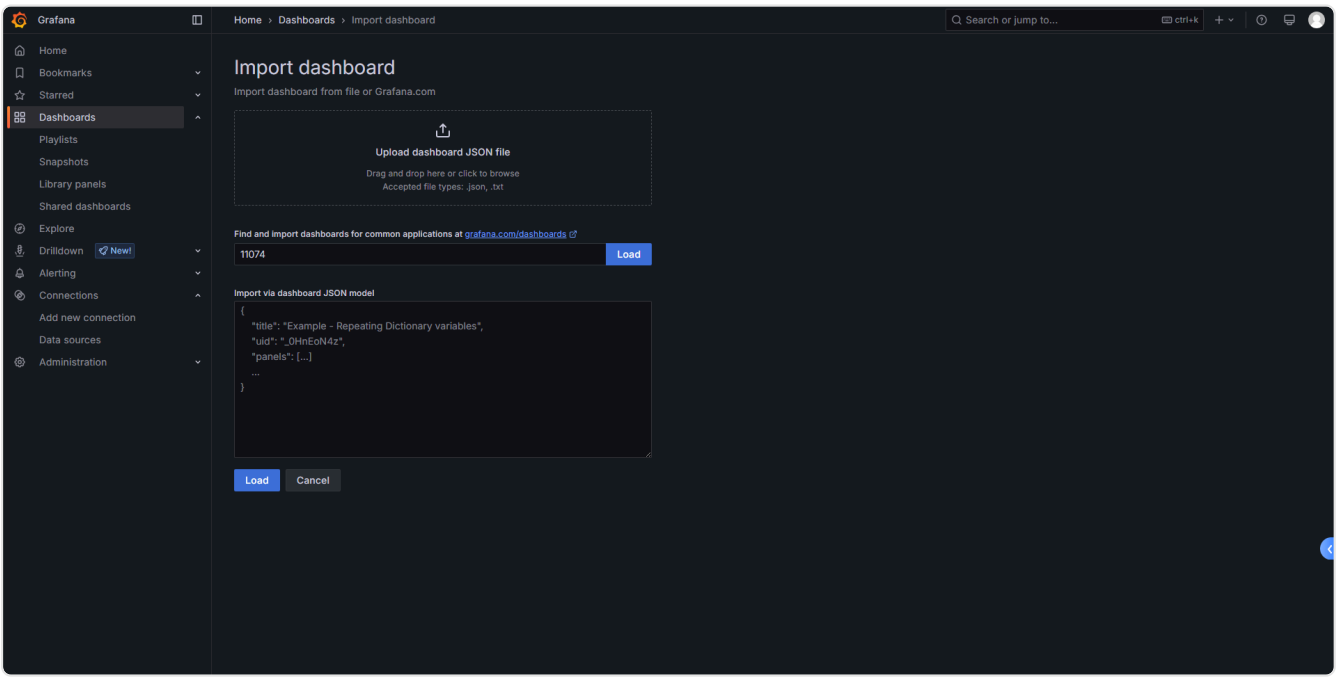

图 9-4 输入仪表盘ID

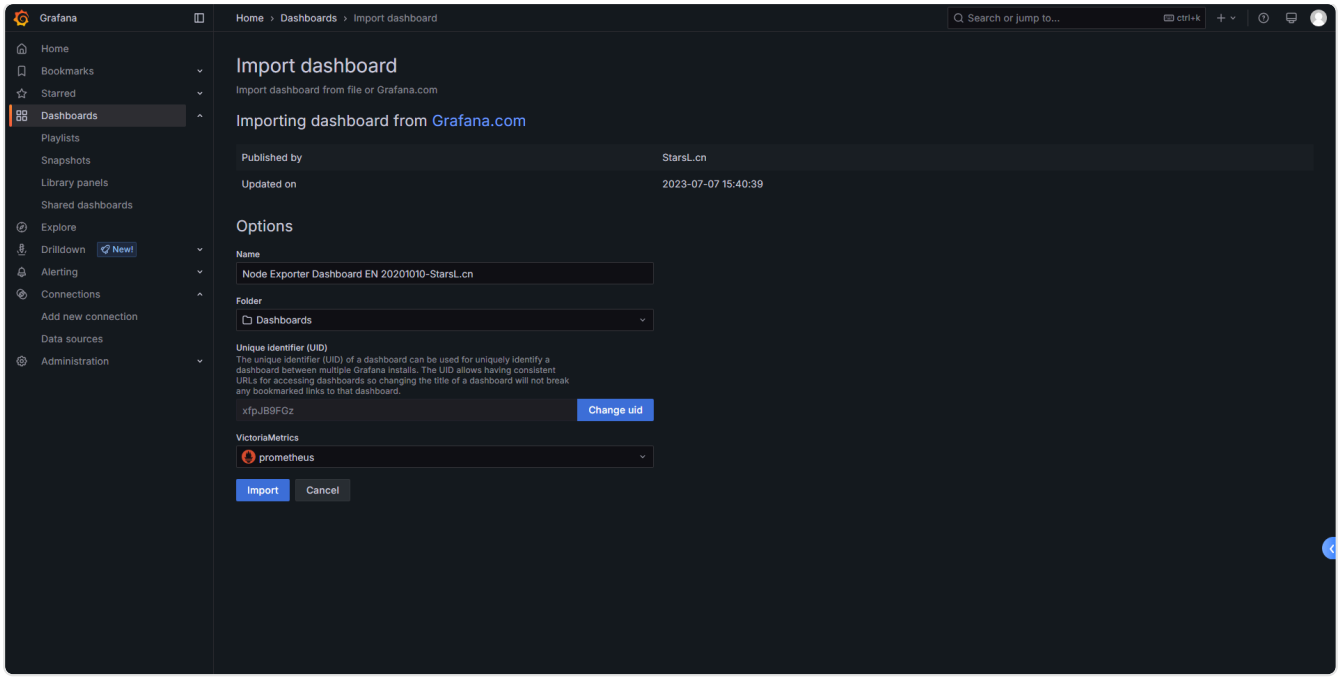

图 9-5添加仪表盘

（2）最后导入该仪表盘，数据展示如图 9-6 所示。

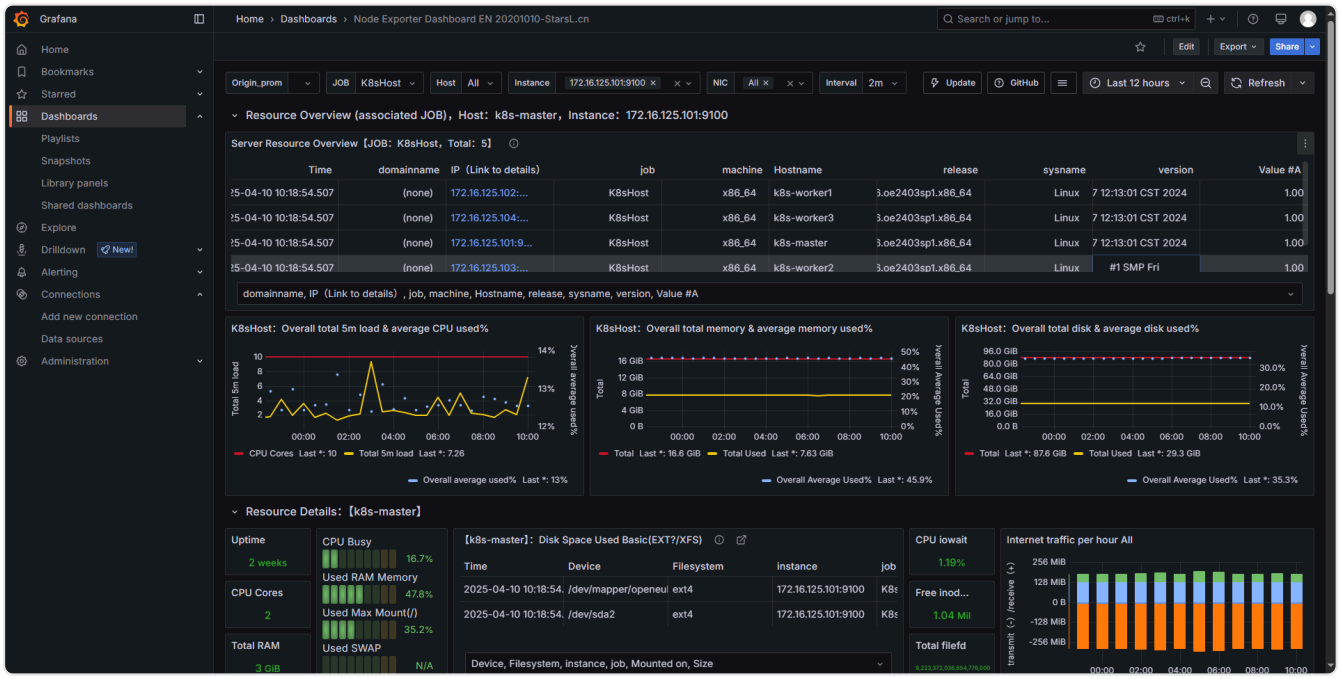

图 9-6 主机监控仪表盘

（3）按照上述步骤，添加仪表盘 ID 为 15661 并载入，最后导入该仪表盘，数据展示如图 9-7 所示。

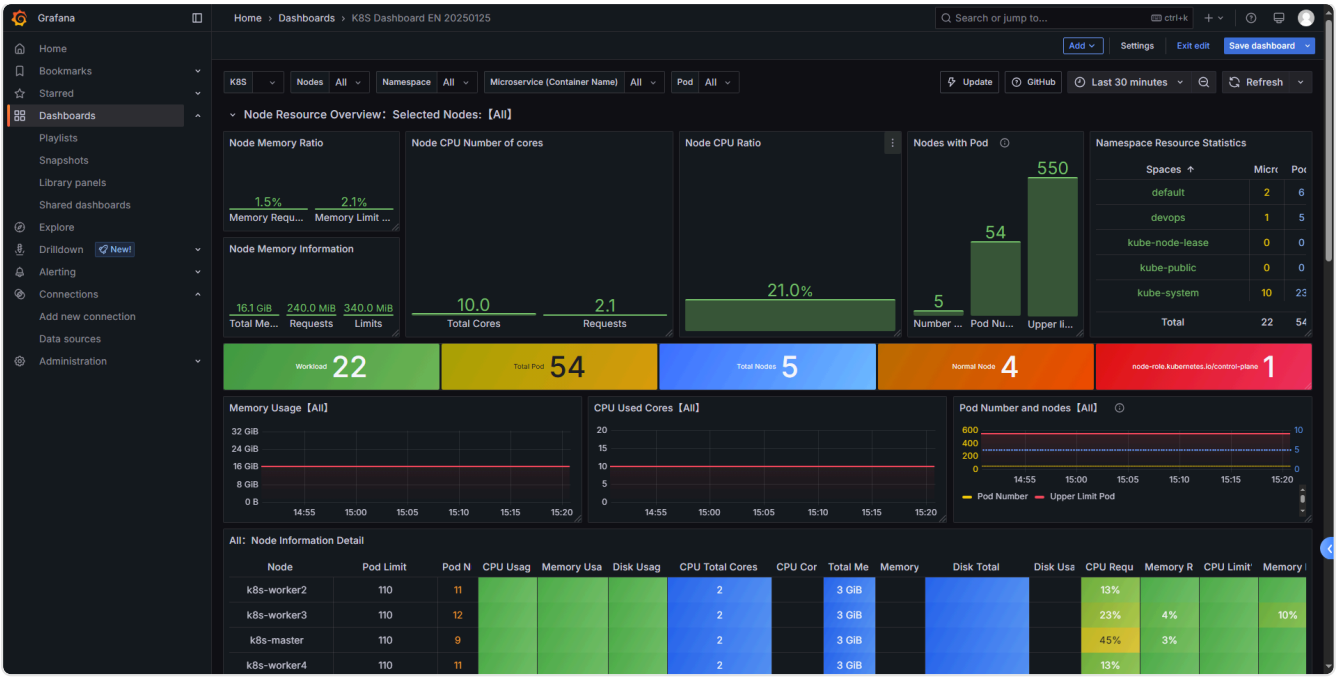

图 9-7 K8S集群监控仪表盘

> 💡 **提醒：**
>
> 可以到 Grafana 官网 https://grafana.com/grafana/dashboards/ 根据采集器的名称或服务名称选择相应的仪表盘。可以下载仪表盘的 json 文件或复制仪表盘的 ID 用于导入。

5、使用 HertzBeat 监控 K8s

步骤 1：部署 HertzBeat

Shell

```
docker run -d -p 1157:1157 -p 1158:1158 \
-v $(pwd)/data:/opt/hertzbeat/data \
-v $(pwd)/logs:/opt/hertzbeat/logs \
--restart=always \
--name hertzbeat apache/hertzbeat
```

浏览器访问 http://172.16.125.107:1157/，默认账户密码 admin/hertzbeat，如图 9-8 所示、9-9 所示。

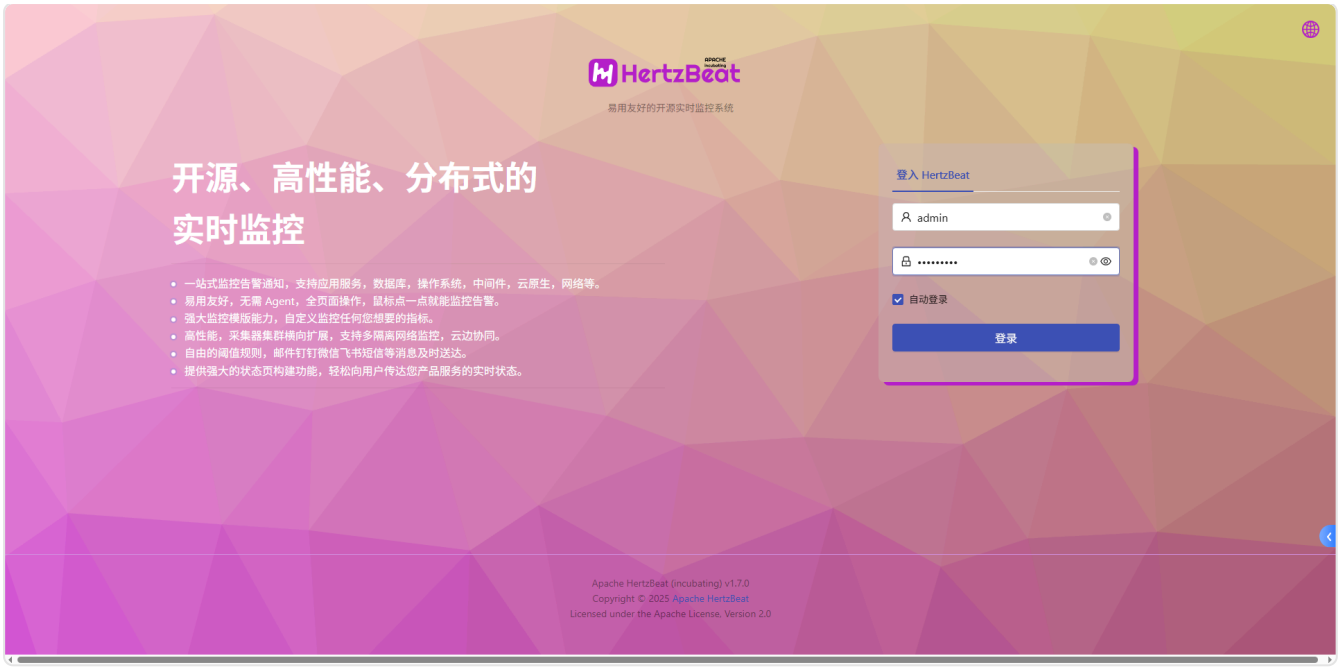

图 9-8 登录

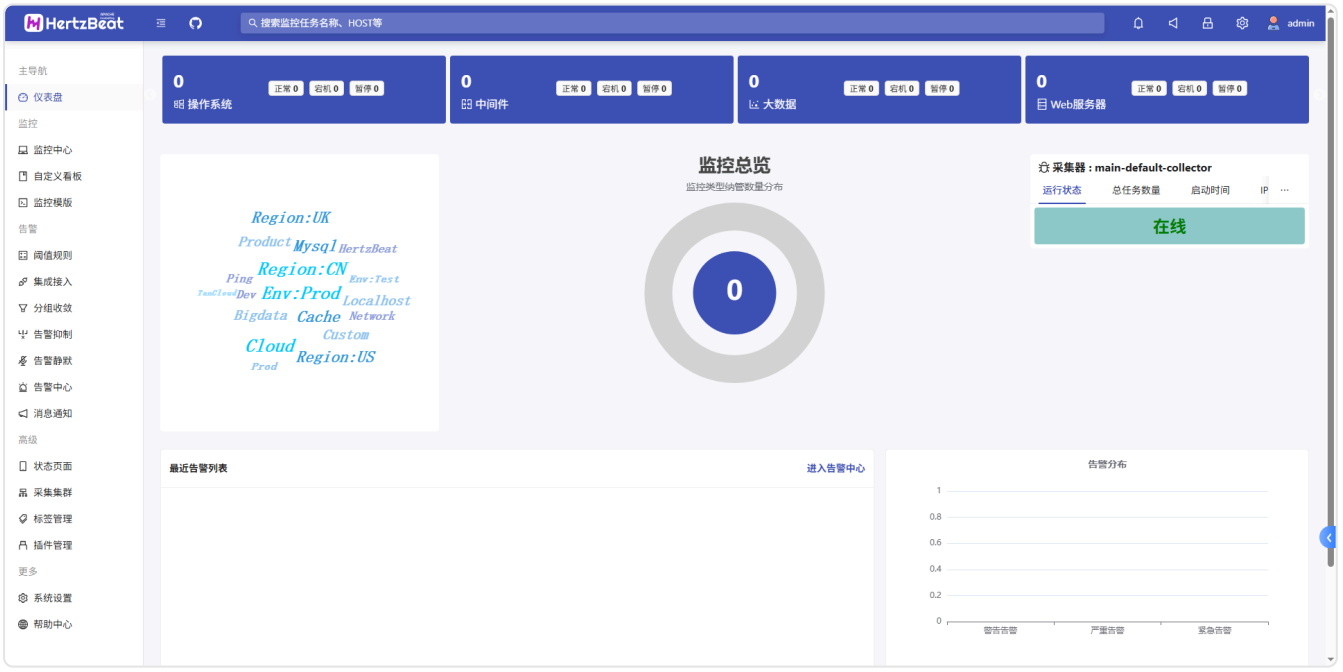

图 9-9 HertzBeat 界面

步骤 2：添加 K8s 监控

在左侧菜单栏选择“监控中心”->“新增监控”->“云原生监控”->“Kubernetes”，配置监控信息测试连接并添加，如图 9-10、图 9-11 所示。

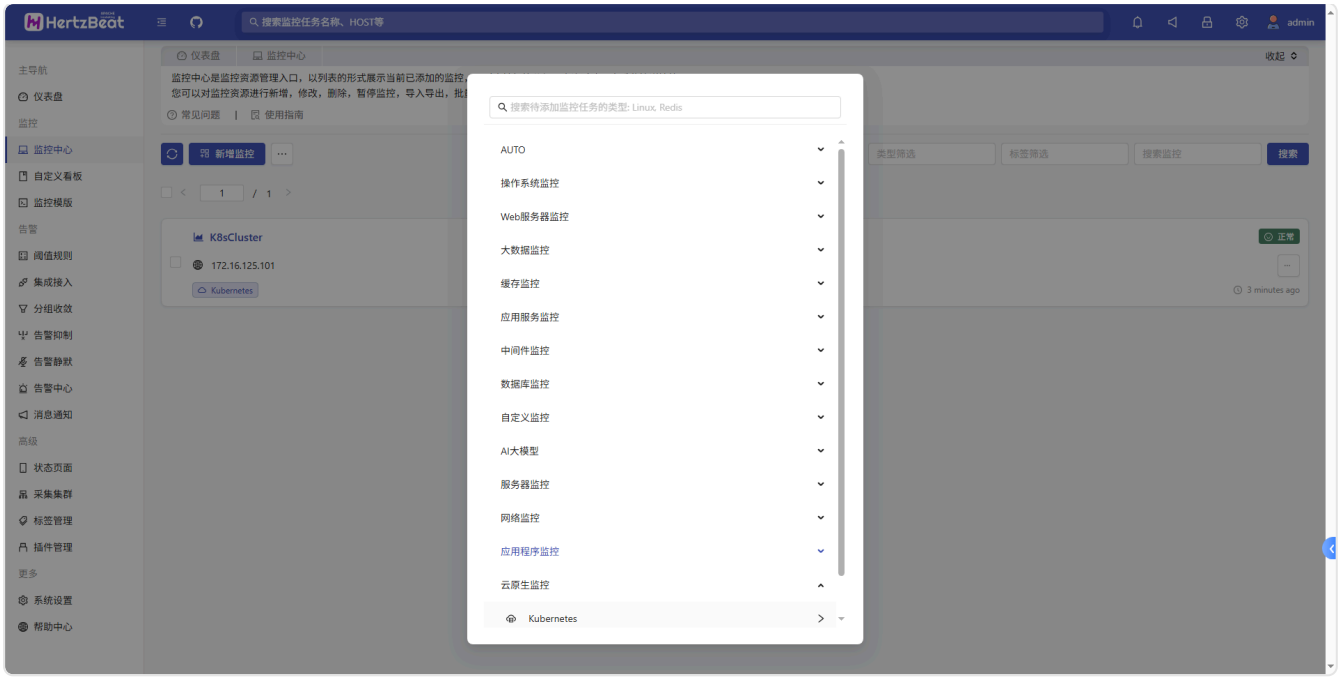

图 9-10 选择Kubernetes

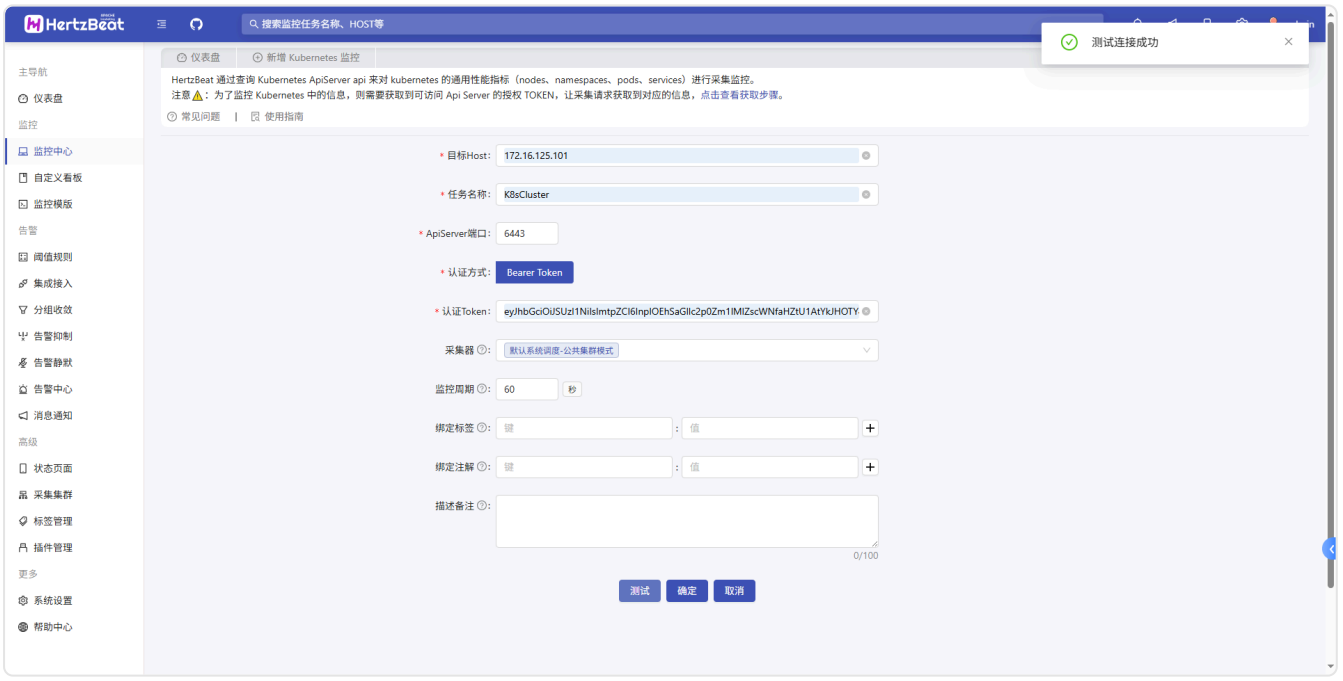

图 9-11 测试连接K8s集群

> 💡 提醒：
>
> 认证 Token 的获取方式可参考《实验 7-Kubernetes Cluster》使用 Token 添加已有 Kubernetes 集群到 Kuboard。复制脚本指令，直接粘贴到 kubernetes 集群的 master 节点的命令行窗口执行，以获得 Token。

步骤 3：查看 K8s 监控数据

监控实时数据详情如图 9-12 所示，监控历史图表详情如图 9-13 所示。

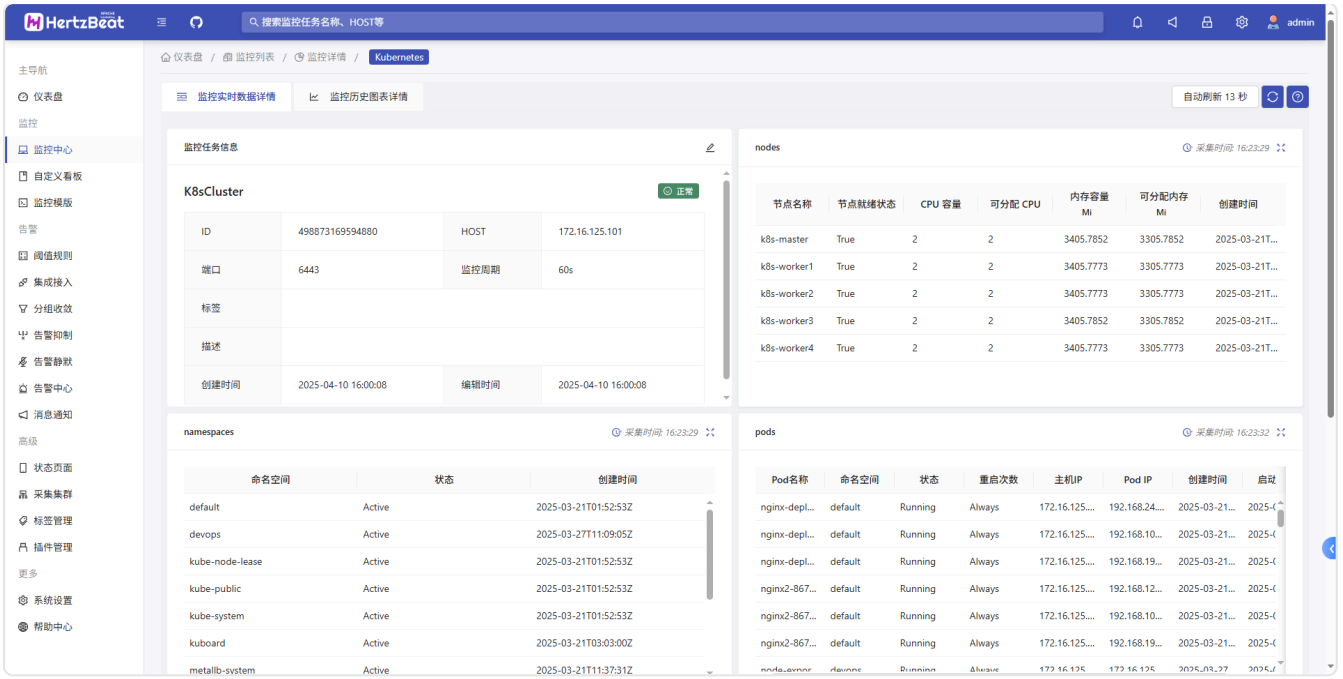

图 9-12 监控实时数据

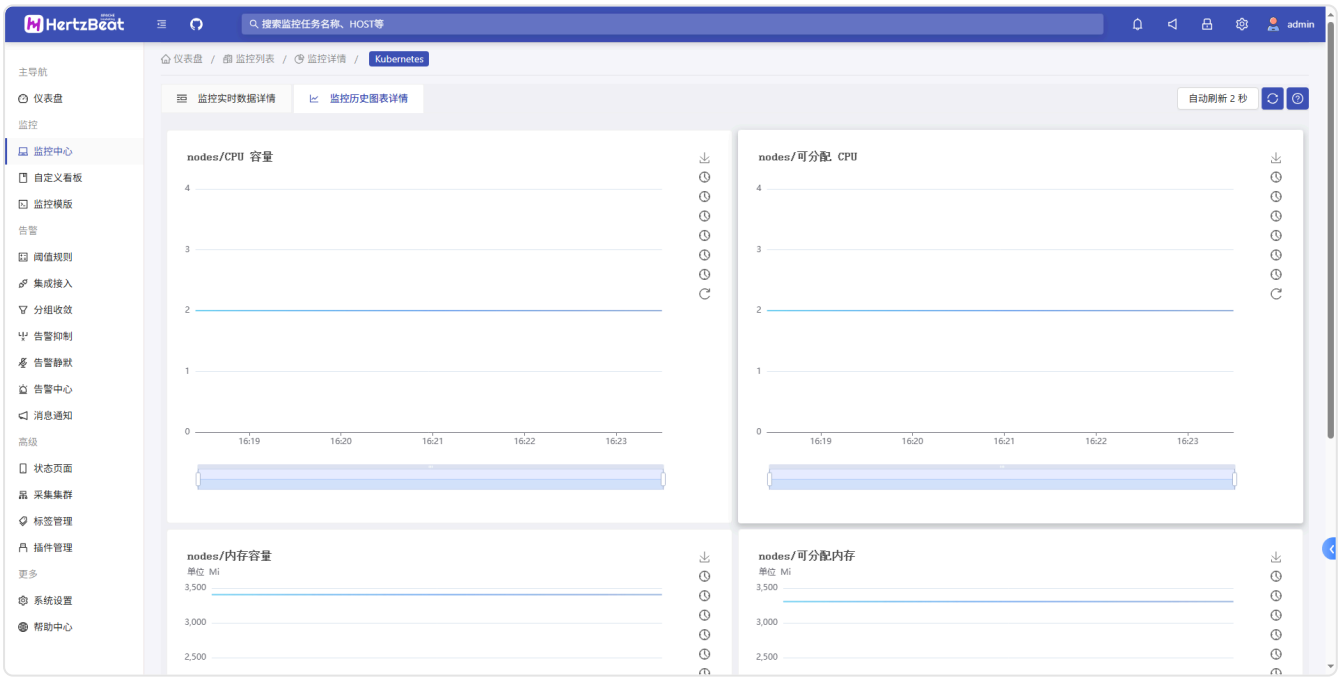

图 9-13 监控历史图表

七、实验讲解

本实验配置讲解视频，访问课程学习平台。

八、实验考核

实验考核为【实验随堂查】。

实验随堂查：每个实验设置 3-5 考核点，学生现场进行演示和汇报讲解。

1、考核点

考核点 1：完成 Prometheus 监控平台的部署。（30 分）

考核点 2：使用 Grafana 可视化监控数据。（40 分）

考核点 3：使用 HertzBeat 监控 K8s。（30 分）

2、考核方式

以实验小组为单位进行考核，每个小组由 1 位同学进行实验成果汇报，小组其他成员回答教师提问。根据汇报和答疑情况，对小组成员进行逐一打分。

由教师进行评分。