基于Kubernetes的PaaS平台应用的设计与实现

1. 安装要求

- 操作系统：Cnetos7.8 最小化安装
- 数量：
- 硬盘：20G
- 内存：2G
- 处理器：1 核 2 个
- yum：安装阿里云在线yum源、Epel源

主机名	节点IP	解析
manager	192.168.2.50	manager.example.com
registry	192.168.2.51	registry.example.com
master	192.168.2.52	master.example.com
work0	192.168.2.53	work0.example.com
work1	192.168.2.54	work1.example.com

以下操作在一台主机上完成，其余直接克隆即可

1.1 基础配置

```
# 进入/etc/yum.repos.d 查看目录下文件

cd /etc/yum.repos.d/
ll

# 将所有文件备份到新建目录repo_bak下
mkdir repo_bak
mv *.repo repo_bak/
mv *.repo.bak repo_bak/
ll

# 下载阿里的CentOS-Base.repo 到/etc/yum.repos.d/
curl -o /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo


# 运行yum clean all 清除缓存，运行 yum makecache 生成新的缓存
yum install -y wget vim bash*

# epel源
wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-
7.repo

```

1.2 关闭防火墙、SElinux、swap分区

```
sed -i 's/enforcing/disabled/' /etc/selinux/config
setenforce 0
systemctl stop firewalld
systemctl disable firewalld
iptables -F
swapoff -a
sed -ri 's/.*swap.*/#&/' /etc/fstab
```

1.3 域名解析和改主机名

```
# 改主机名
hostnamectl set-hostname manager.example.com

# 域名解析:
[root@manager ~]# vim /etc/hosts
192.168.2.50   manager.example.com      manager
192.168.2.51   registry.example.com     registry
192.168.2.52   master.example.com       master
192.168.2.53   work0.example.com        work0
192.168.2.54   work1.example.com        work1

# 配置网络
[root@manager ~]# nmcli connection modify ens33 ipv4.addresses
192.168.2.50/24 ipv4.dns 8.8.8.8 ipv4.gateway 192.168.2.2 ipv4.method manual
autoconnect yes

# 重启网卡
[root@manager ~]# nmcli connection up ens33
```

1.4 克隆四台

```
# 修改主机名
hostnamectl set-hostname manager.example.com
hostnamectl set-hostname registry.example.com
hostnamectl set-hostname master.example.com
hostnamectl set-hostname work0.example.com
hostnamectl set-hostname work1.example.com

# 修改IP地址
nmcli con mod ens33 ipv4.addresses 192.168.2.51/24
nmcli con mod ens33 ipv4.addresses 192.168.2.52/24
nmcli con mod ens33 ipv4.addresses 192.168.2.53/24
nmcli con mod ens33 ipv4.addresses 192.168.2.54/24

```

2. Ansible节点的配置

以下过程均在`manager`主机上执行

2.1 安装Ansible

```
[root@manager ~]# yum install ansible -y
```

```
Installed:
  ansible.noarch 0:2.9.27-1.el7

Dependency Installed:
  PyYAML.x86_64 0:3.10-11.el7              libyaml.x86_64 0:0.1.4-11.el7_0
  python-backports.x86_64 0:1.0-8.el7      python-backports-ssl_match_hostname.noarch 0:
  python-enum34.noarch 0:1.0.4-1.el7       python-idna.noarch 0:2.4-1.el7
  python-jinja2.noarch 0:2.7.2-4.el7       python-markupsafe.x86_64 0:0.11-10.el7
  python-ply.noarch 0:3.4-11.el7           python-pycparser.noarch 0:2.14-1.el7
  python-six.noarch 0:1.9.0-2.el7          python2-cryptography.x86_64 0:1.7.2-2.el7
  python2-jmespath.noarch 0:0.9.4-2.el7    python2-pyasn1.noarch 0:0.1.9-7.el7

Complete!
[root@manager ~]#
```

2.2 编写主机清单

```
# 创建目录
[root@manager ~]# mkdir manager
[root@manager ~]# cd manager/

[root@manager manager]# vim ansible.cfg
[defaults]
inventory         = inventory
host_key_checking = False
remote_user       = root
[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False

# 编写清单文件
[root@manager manager]# vim inventory
[harbor]
registry.example.com  ansible_host=192.168.2.51

[master]
master0.example.com   ansible_host=192.168.2.52

[works]
work0.example.com     ansible_host=192.168.2.53
work1.example.com     ansible_host=192.168.2.54

[k8s-cluster:children]
master
works

[all:vars]
ansible_password  = centos

```

2.2 验证

```
[root@manager manager]# ansible all -m ping
```

```
[root@manager manager]# ansible all -m ping
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
work0.example.com | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.122.53 port 22: Connection timed out"
,
    "unreachable": true
}
work1.example.com | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.122.54 port 22: Connection timed out"
,
    "unreachable": true
}
master0.example.com | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.122.52 port 22: Connection timed out"
,
    "unreachable": true
}
registry.example.com | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.122.51 port 22: Connection timed out"
,
    "unreachable": true
}
[root@manager manager]#
```

2.3 配置SSH免密

```
[root@manager manager]# vim 01-set-authorized_key.yml

---
- name: create ssh key
  hosts: localhost
  tasks:
    - name: Create a 2048-bit SSH key for user jsmith in ~jsmith/.ssh/id_rsa
      user:
        name: root
        generate_ssh_key: yes

- name: ssh copy id
  hosts: all
  tasks:
    - name: Set authorized key taken from file
      authorized_key:
        user: root
        state: present
        key: "{{ lookup('file', '/root/.ssh/id_rsa.pub') }}"
```

执行

```
[root@manager manager]# ansible-playbook 01-set-authorized_key.yml

```

```
[root@manager manager]# ansible-playbook 01-set-authorized_key.yml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [create ssh key] **********************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [localhost]

TASK [Create a 2048-bit SSH key for user jsmith in ~jsmith/.ssh/id_rsa] ********************************************************
ok: [localhost]

PLAY [ssh copy id] *************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [registry.example.com]
ok: [work1.example.com]
ok: [master0.example.com]
ok: [work0.example.com]

TASK [Set authorized key taken from file] **************************************************************************************
changed: [work0.example.com]
changed: [master0.example.com]
changed: [registry.example.com]
changed: [work1.example.com]

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
master0.example.com        : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
registry.example.com       : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
work0.example.com          : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
work1.example.com          : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[root@manager manager]#
```

验证

```
[root@manager manager]# ssh work1
The authenticity of host 'work1 (192.168.2.54)' can't be established.
ECDSA key fingerprint is SHA256:2kSMrsjf/l+8z+AOpT1qt/dGREmP2p7PpZkK3nYzoyg.
ECDSA key fingerprint is MD5:bd:16:0d:bf:e0:de:dc:04:4c:19:e4:01:21:9e:42:78.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'work1' (ECDSA) to the list of known hosts.
Last login: Fri Apr 29 15:50:58 2022 from manager.example.com
[root@work1 ~]#
```

2.4 一键设计所有主机名

```
[root@manager manager]# vim 02-set-hostname.yml

---
- name: set hostname
  hosts: all
  tasks:
    - name: set host name
      hostname:
        name: "{{ inventory_hostname }}"
```

执行

```
[root@manager manager]# ansible-playbook 02-set-hostname.yml
```

查看

```
[root@manager manager]# ansible all -m shell -a "hostname"
```

```
[root@manager manager]# ansible all -m shell -a "hostname"
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
work1.example.com | CHANGED | rc=0 >>
work1.example.com
master0.example.com | CHANGED | rc=0 >>
master0.example.com
work0.example.com | CHANGED | rc=0 >>
work0.example.com
registry.example.com | CHANGED | rc=0 >>
registry.example.com
[root@manager manager]#
```

把所有的主机名对应的IP地址都已经正常

3. Harbor仓库配置

以下过程均在`registry`主机上执行

3.1 安装Docker

```
# step 1: 安装必要的一些系统工具
sudo yum install -y yum-utils device-mapper-persistent-data lvm2

# Step 2: 添加软件源信息
sudo yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

# Step 3
sudo sed -i 's+download.docker.com+mirrors.aliyun.com/docker-ce+' /etc/yum.repos.d/docker-ce.repo

# Step 4: 更新并安装Docker-CE
sudo yum makecache fast
sudo yum -y install docker-ce

# Step 5: 开启Docker服务
sudo systemctl enable --now docker
```

因为每台主机都需要安装Docker，所以我们直接在Ansible主机上编写一个playbook，直接每台受管主机上执行

以下操作在`manager`主机上执行

```
[root@manager manager]# vim 03-configure-docker.yml

---
- name: configure docker for k8s cluster
  hosts: k8s-cluster
  tasks:
    - name: install some packages
      yum:
        state: present
        name:
          - yum-utils
          - device-mapper-persistent-data
          - lvm2

    - name: add docker-ce repository 1
      shell: yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

```
17
18      - name: add docker-ce repository 2
19        shell: sed -i 's+download.docker.com+mirrors.aliyun.com/docker-ce+'
    /etc/yum.repos.d/docker-ce.repo
20
21      - name: install docker-ce
22        yum:
23          name: docker-ce
24          state: present
25
26      - name: start and enabled docker service\
27        service:
28          name: docker
29          enabled: yes
30          state: started
```

执行

```
1  [root@manager manager]# ansible-playbook 03-configure-docker.yml
```

到了这一步，除了`manager`主机，其他均已安装好docker

```
[root@registry ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-29 16:06:37 CST; 19min ago
     Docs: https://docs.docker.com
 Main PID: 2867 (dockerd)
    Tasks: 8
   Memory: 33.8M
   CGroup: /system.slice/docker.service
           └─2867 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.185678758+08:00"
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.185696510+08:00"
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.185703557+08:00"
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.310281373+08:00"
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.408023743+08:00"
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.439865222+08:00"
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.457865923+08:00"
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.457931639+08:00"
Apr 29 16:06:37 registry.example.com systemd[1]: Started Docker Application Container Engine.
Apr 29 16:06:37 registry.example.com dockerd[2867]: time="2022-04-29T16:06:37.470703922+08:00"
Hint: Some lines were ellipsized, use -l to show in full.
[root@registry ~]#
```

```
[root@master0 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-29 16:26:18 CST; 8s ago
     Docs: https://docs.docker.com
 Main PID: 1807 (dockerd)
    Tasks: 8
   Memory: 32.0M
   CGroup: /system.slice/docker.service
           └─1807 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 29 16:26:17 master0.example.com dockerd[1807]: time="2022-04-29T16:26:17.971838783+08:00"
Apr 29 16:26:17 master0.example.com dockerd[1807]: time="2022-04-29T16:26:17.971849457+08:00"
Apr 29 16:26:17 master0.example.com dockerd[1807]: time="2022-04-29T16:26:17.971856263+08:00"
Apr 29 16:26:18 master0.example.com dockerd[1807]: time="2022-04-29T16:26:18.068945346+08:00"
Apr 29 16:26:18 master0.example.com dockerd[1807]: time="2022-04-29T16:26:18.166953254+08:00"
Apr 29 16:26:18 master0.example.com dockerd[1807]: time="2022-04-29T16:26:18.200353472+08:00"
Apr 29 16:26:18 master0.example.com dockerd[1807]: time="2022-04-29T16:26:18.237825410+08:00"
Apr 29 16:26:18 master0.example.com dockerd[1807]: time="2022-04-29T16:26:18.237945625+08:00"
Apr 29 16:26:18 master0.example.com systemd[1]: Started Docker Application Container Engine.
Apr 29 16:26:18 master0.example.com dockerd[1807]: time="2022-04-29T16:26:18.250329642+08:00"
Hint: Some lines were ellipsized, use -l to show in full.
[root@master0 ~]#
```

```
[root@work0 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-29 16:13:52 CST; 12min ago
     Docs: https://docs.docker.com
 Main PID: 2962 (dockerd)
    Tasks: 8
   Memory: 36.3M
   CGroup: /system.slice/docker.service
           └─2962 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 29 16:13:51 work0.example.com dockerd[2962]: time="2022-04-29T16:13:51.963483843+08:00" l
Apr 29 16:13:51 work0.example.com dockerd[2962]: time="2022-04-29T16:13:51.963504927+08:00" l
Apr 29 16:13:51 work0.example.com dockerd[2962]: time="2022-04-29T16:13:51.963513157+08:00" l
Apr 29 16:13:52 work0.example.com dockerd[2962]: time="2022-04-29T16:13:52.152669984+08:00" l
Apr 29 16:13:52 work0.example.com dockerd[2962]: time="2022-04-29T16:13:52.291581377+08:00" l
Apr 29 16:13:52 work0.example.com dockerd[2962]: time="2022-04-29T16:13:52.356559251+08:00" l
Apr 29 16:13:52 work0.example.com dockerd[2962]: time="2022-04-29T16:13:52.372366741+08:00" l
Apr 29 16:13:52 work0.example.com dockerd[2962]: time="2022-04-29T16:13:52.372482607+08:00" l
Apr 29 16:13:52 work0.example.com systemd[1]: Started Docker Application Container Engine.
Apr 29 16:13:52 work0.example.com dockerd[2962]: time="2022-04-29T16:13:52.409845102+08:00" l
Hint: Some lines were ellipsized, use -l to show in full.
[root@work0 ~]#
```

```
[root@work1 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-04-29 16:13:51 CST; 12min ago
     Docs: https://docs.docker.com
 Main PID: 2974 (dockerd)
    Tasks: 8
   Memory: 33.8M
   CGroup: /system.slice/docker.service
           └─2974 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.407988961+08:00" l
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.408011889+08:00" l
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.408022478+08:00" l
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.611855944+08:00" l
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.753561530+08:00" l
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.818223708+08:00" l
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.832867235+08:00" l
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.832983463+08:00" l
Apr 29 16:13:51 work1.example.com systemd[1]: Started Docker Application Container Engine.
Apr 29 16:13:51 work1.example.com dockerd[2974]: time="2022-04-29T16:13:51.864488692+08:00" l
Hint: Some lines were ellipsized, use -l to show in full.
[root@work1 ~]#
```

3.2 安装Harbor

- 官网：Harbor (goharbor.io)
- 下载：Releases · goharbor/harbor (github.com)

以下操作在 `registry` 主机上执行

先把tar包从window是拷贝的 `regisrty` 主机下

解压

```
[root@registry apps]# ls
harbor-offline-installer-v2.4.1-rc2.tgz
[root@registry apps]# tar xzvf harbor-offline-installer-v2.4.1-rc2.tgz
```

配置yml文件

```
[root@registry harbor]# vim harbor.yml
hostname: registry.example.com

http:
  port: 80

#https:
#  port: 443
#  certificate: /apps/harbor/certs/harbor-ca.crt
#  private_key: /apps/harbor/certs/harbor-ca.key

harbor_admin_password: Harbor12345

database:
  password: root123
  max_idle_conns: 100
  max_open_conns: 900

data_volume: /data
... output omitted ...
```

安装 docker-compose

```
[root@registry harbor]# yum install docker-compose -y
```

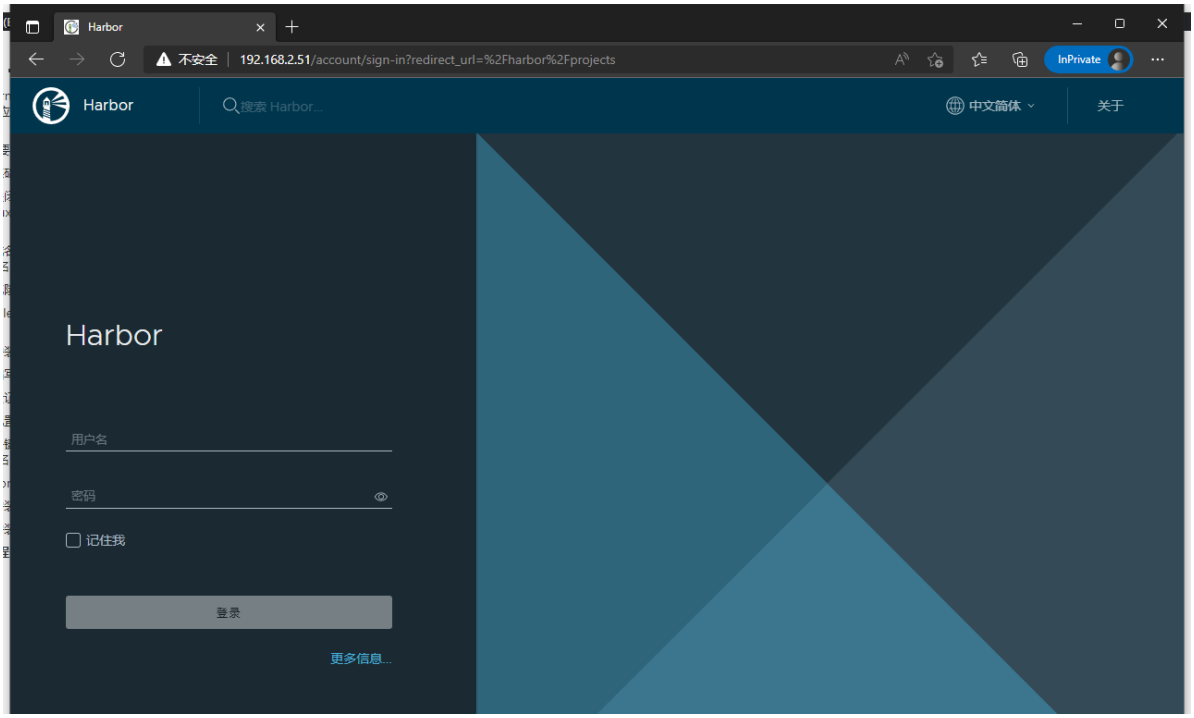
运行

```
[root@registry harbor]# ./install.sh --with-trivy
```

```
Creating redis ... done
Creating registry ... done
Creating harbor-core ... done
Creating network "harbor_harbor" with the default driver
Creating nginx ... done
Creating registry ...
Creating redis ...
Creating registryctl ...
Creating harbor-portal ...
Creating harbor-db ...
Creating trivy-adapter ...
Creating harbor-core ...
Creating nginx ...
Creating harbor-jobservice ...
✔ ----Harbor has been installed and started successfully.----
[root@registry harbor]#
```

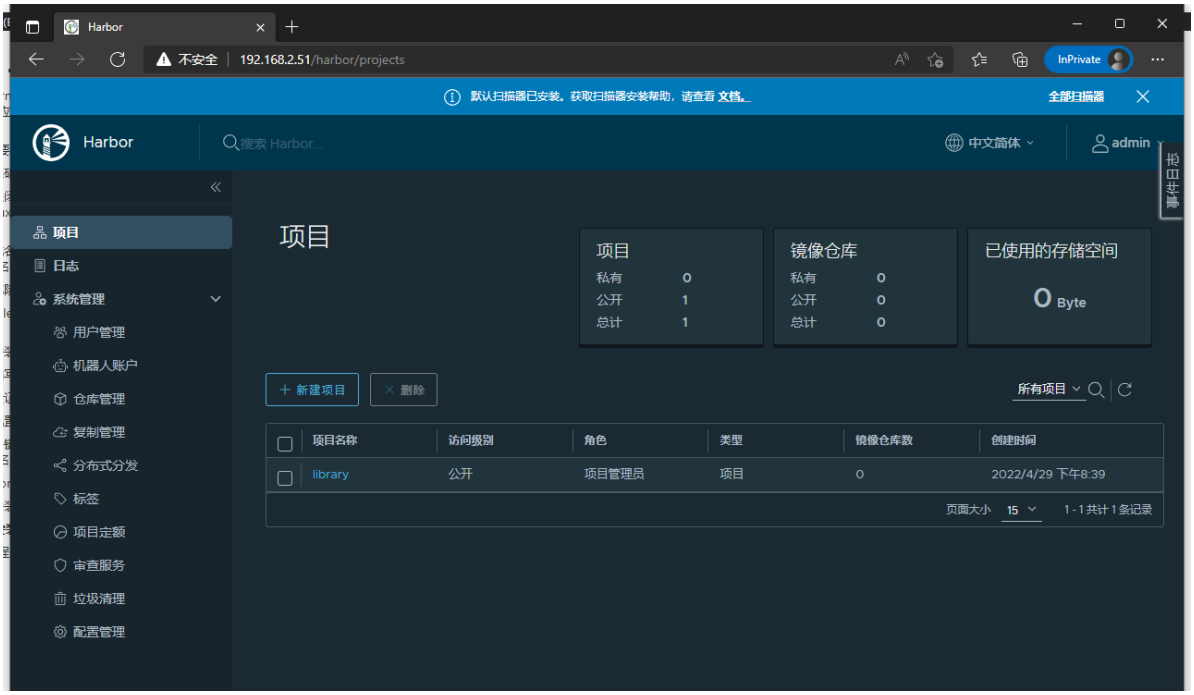
访问harbor

在浏览器输入：`192.168.2.51`



用户名：`admin`

密码：`Harbor12345`



3.3 配置docker文件

```
[root@registry harbor]# vim /etc/docker/daemon.json
{
    "registry-mirrors": ["https://registry.example.com"],
    "insecure-registries": ["http://registry.example.com",
                            "http://192.168.2.51"
                           ],
    "live-restore": true
}
```

3.4 重启

```
[root@registry harbor]# systemctl daemon-reload
[root@registry harbor]# systemctl restart docker
[root@registry harbor]# docker-compose up -d
```

3.4 登录harbor

```
[root@registry harbor]# docker login http://registry.example.com -u admin -p Harbor12345
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@registry harbor]#
```

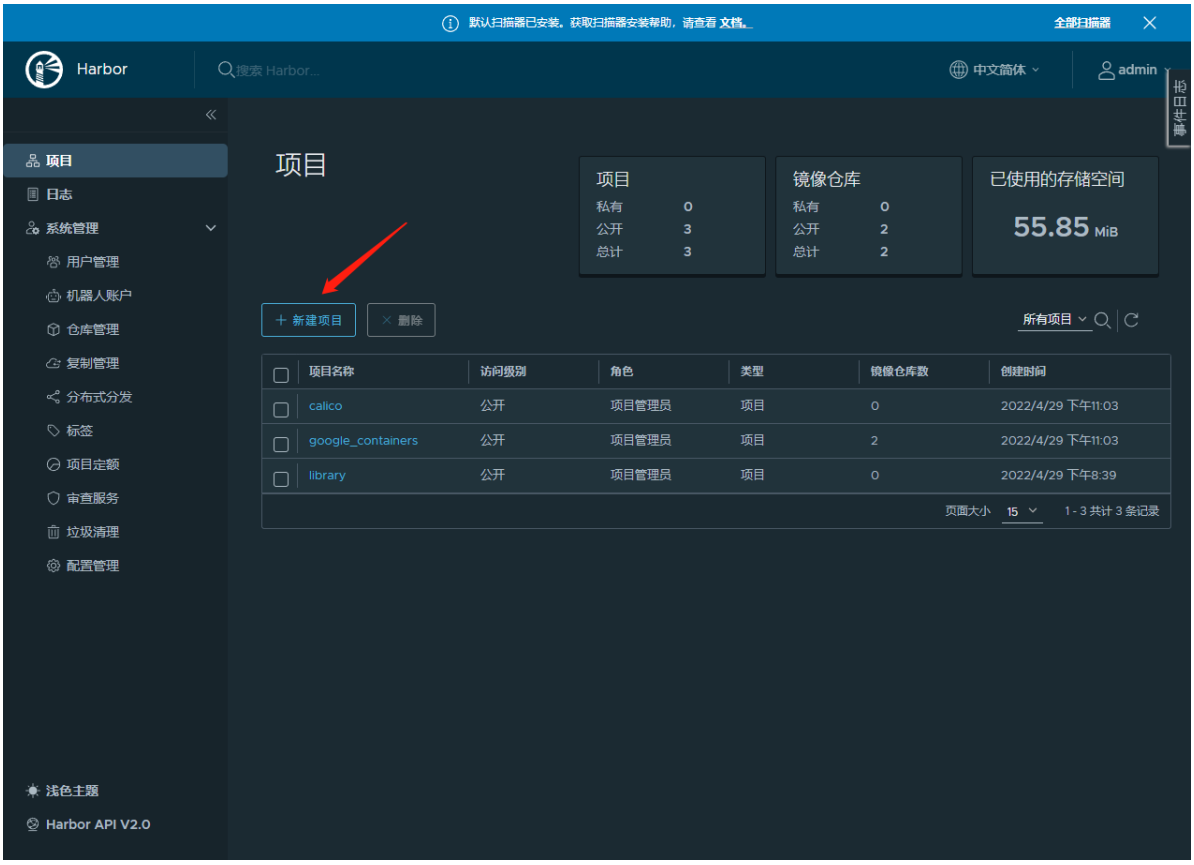
登录成功!

3.5 准备安装k8s集群的镜像

在`registry`主机上执行

```
[root@registry harbor]# vim /etc/docker/daemon.json
{
    "registry-mirrors": ["https://gbnzolz3.mirror.aliyuncs.com"],
    "insecure-registries": ["http://registry.example.com",
                            "http://192.168.2.51"
                           ],
    "live-restore": true
}

[root@registry harbor]# systemctl daemon-reload
```

在Harbor web界面上建项目



google_containers

calico

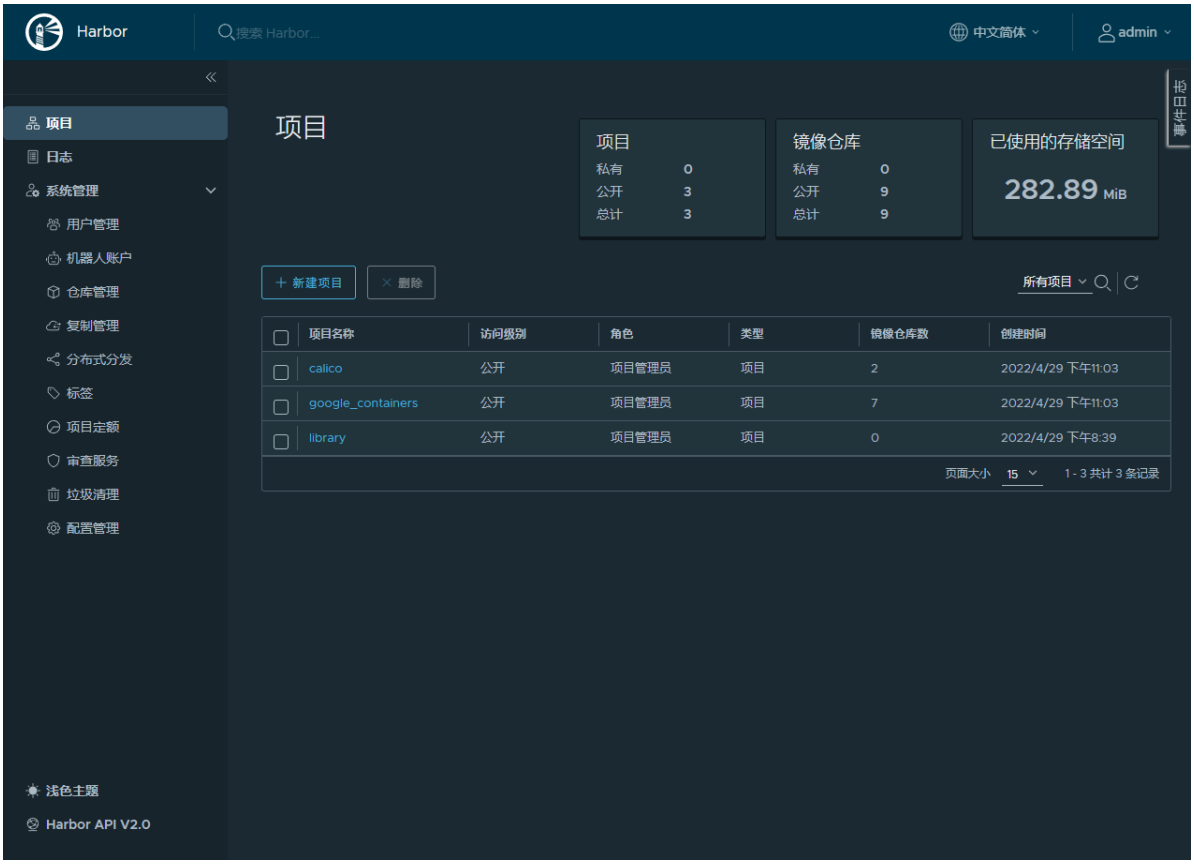
编写脚本，下载镜像、对镜像打标签、将镜像上传到自己的 Harbor 服务器

```
[root@registry harbor]# vim configure-k8s-setup-images.sh
#!/bin/bash
images=(
  kube-apiserver:v1.21.12
  kube-controller-manager:v1.21.12
```

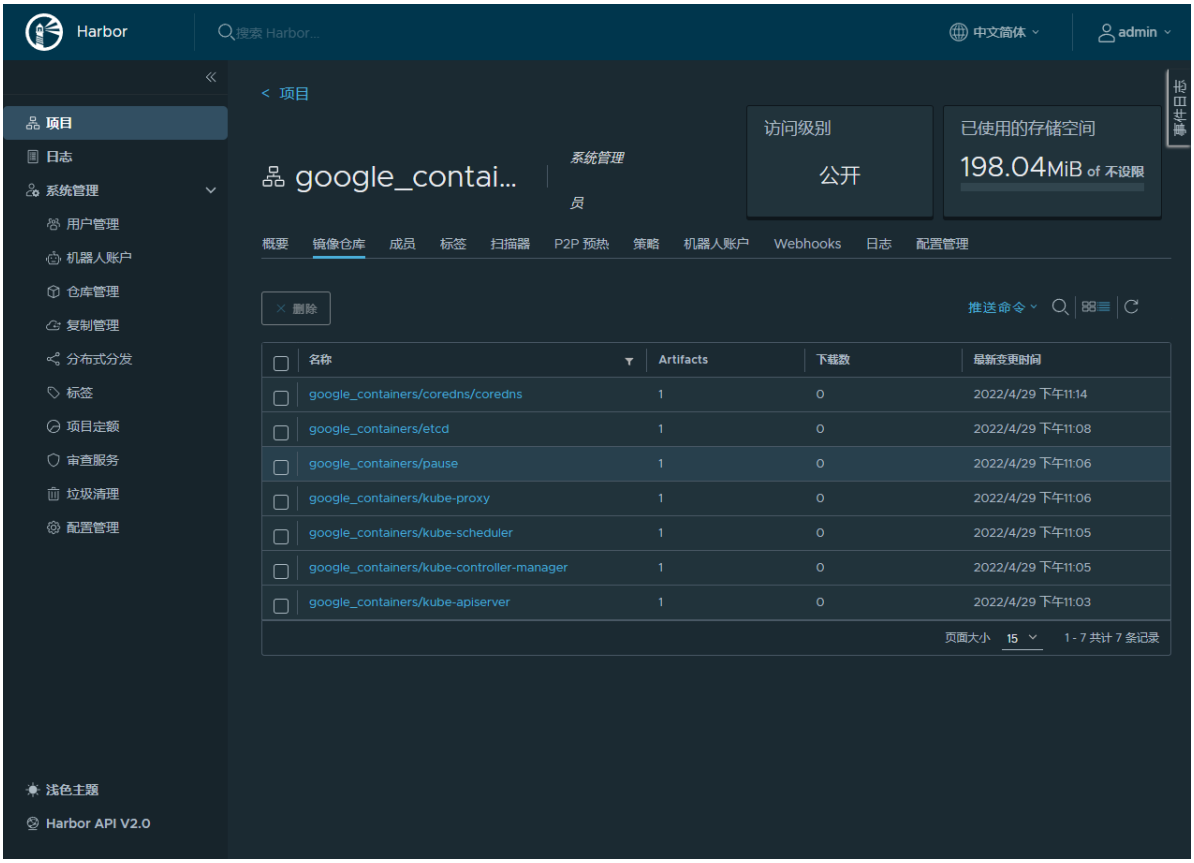
```
  kube-scheduler:v1.21.12
  kube-proxy:v1.21.12
  pause:3.4.1
  etcd:3.4.13-0
  coredns/coredns:v1.8.0
)
for imageName in ${images[@]}
do
  docker pull registry.aliyuncs.com/google_containers/${imageName}
  docker tag registry.aliyuncs.com/google_containers/${imageName}
registry.example.com/google_containers/${imageName}
  docker rmi registry.aliyuncs.com/google_containers/${imageName}
  docker login http://registry.example.com -u admin -p Harbor12345
  docker push registry.example.com/google_containers/${imageName}
done

[root@registry harbor]# chmod +x configure-k8s-setup-images.sh

[root@registry harbor]# ./configure-k8s-setup-images.sh

```

编写脚本，将镜像下载到本地、重新打标签、上传至 Harbor 服务器

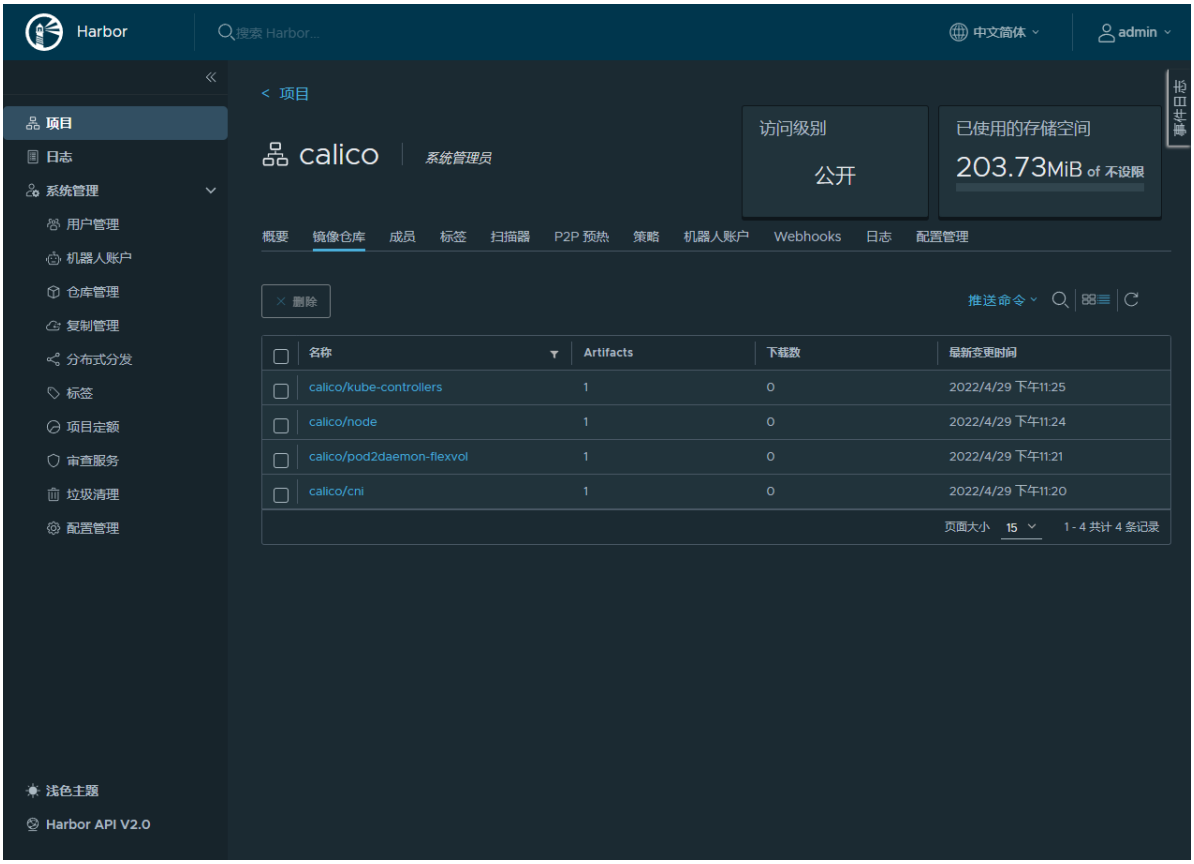
```
[root@registry harbor]# vim configure-calico-images.sh
#!/bin/bash
images=(
  cni:v3.22.2
  pod2daemon-flexvol:v3.22.2
  node:v3.22.2
  kube-controllers:v3.22.2
)
for imageName in ${images[@]}
do
  docker pull docker.io/calico/${imageName}
  docker tag docker.io/calico/${imageName}
registry.example.com/calico/${imageName}
  docker rmi docker.io/calico/${imageName}
  docker push registry.example.com/calico/${imageName}
done

[root@registry harbor]# chmod +x configure-calico-images.sh

[root@registry harbor]# ./configure-calico-images.sh
```


Harbor
搜索 Harbor...
中文简体
admin
项目
日志
系统管理
用户管理
机器人账户
仓库管理
复制管理
分布式分发
标签
项目定额
审查服务
垃圾清理
配置管理
浅色主题
Harbor API V2.0
项目
项目
私有 0
公开 3
总计 3
镜像仓库
私有 0
公开 9
总计 9
已使用的存储空间
282.89 MiB
+ 新建项目
× 删除
所有项目
项目名称 访问级别 角色 类型 镜像仓库数 创建时间
calico 公开 项目管理员 项目 2 2022/4/29 下午11:03
google_containers 公开 项目管理员 项目 7 2022/4/29 下午11:03
library 公开 项目管理员 项目 0 2022/4/29 下午8:39
页面大小 15 1 - 3 共计 3 条记录

验证


Harbor
搜索 Harbor...
中文简体
admin
项目
日志
系统管理
用户管理
机器人账户
仓库管理
复制管理
分布式分发
标签
项目定额
审查服务
垃圾清理
配置管理
浅色主题
Harbor API V2.0
< 项目
google_contai...
系统管理员
访问级别
公开
已使用的存储空间
198.04MiB of 不设限
概要 镜像仓库 成员 标签 扫描器 P2P 预热 策略 机器人账户 Webhooks 日志 配置管理
× 删除
推送命令
名称 Artifacts 下载数 最新变更时间
google_containers/coredns/coredns 1 0 2022/4/29 下午11:14
google_containers/etcd 1 0 2022/4/29 下午11:08
google_containers/pause 1 0 2022/4/29 下午11:06
google_containers/kube-proxy 1 0 2022/4/29 下午11:06
google_containers/kube-scheduler 1 0 2022/4/29 下午11:05
google_containers/kube-controller-manager 1 0 2022/4/29 下午11:05
google_containers/kube-apiserver 1 0 2022/4/29 下午11:03
页面大小 15 1 - 7 共计 7 条记录

2



4. Kubernetes配置

4.1 配置

在3.1步骤中，已经在master0、work0、work1上安装好了docker

以下步骤在`manager`主机上执行

```
[root@manager manager]# vim daemon.json
{
    "registry-mirrors": ["https://gbnzolz3.mirror.aliyuncs.com"],
    "insecure-registries": ["http://registry.example.com",
                            "http://192.168.2.51"
                           ],
    "live-restore": true
}
```

编写一个playbook，在master、work0、work1上面都配置好上面的json文件

```
[root@manager manager]# vim 04-config-docker.yml
---
- name: /etc/docker/daemon.json
  hosts: k8s-cluster
  tasks:
    - name: create directory
      file:
        path: /etc/docker/
        state: directory
```

```
    - name: copy file to /etc/docker
      copy:
        src: daemon.json
        dest: /etc/docker/daemon.json

    - name: restart docker service
      systemd:
        daemon_reload: yes
        name: docker
        state: restarted
[root@manager manager]#
```

4.2 配置k8s Yum源

以下操作在`master0`、`work0`、`work1`上运行

```
cat > /etc/yum.repos.d/kubernetes.repo << EOF
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
EOF
```

4.3 配置网络转发

以下操作在`master0`、`work0`、`work1`上运行

```
# 配置内核参数
cat > /etc/sysctl.d/k8s.conf << EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
vm.swappiness = 0
net.ipv4.ip_forward = 1
EOF

# 刷新生效
sysctl -p /etc/sysctl.d/k8s.conf

# 配置 ipvs
cat > /etc/sysconfig/modules/ipvs.modules << EOF
modprobe -- br_netfilter
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4
EOF

chmod 755 /etc/sysconfig/modules/ipvs.modules && \
bash /etc/sysconfig/modules/ipvs.modules
```

4.4 安装k8s

以下操作在`master0`、`work0`、`work1`上执行

安装并加入开机自启

```
[root@master0 ~]# yum install kubelet-1.21.11 kubeadm-1.21.11 kubectl-1.21.11 -y

[root@master0 ~]# systemctl enable --now kubelet

[root@master0 ~]# systemctl restart kubelet.service
```

4.5 安装 k8s master node

```
[root@master0 ~]# kubeadm init \
  --apiserver-advertise-address=192.168.2.52 \
  --image-repository registry.example.com/google_containers \
  --kubernetes-version v1.21.12 \
  --service-cidr=10.96.0.0/12 \
  --pod-network-cidr=10.244.0.0/16 \
  --ignore-preflight-errors=all \
  --v=5

[root@master0 ~]#  mkdir -p $HOME/.kube
[root@master0 ~]#  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@master0 ~]#  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

在`work1`、`work1`执行以下操作

```
kubeadm join 192.168.2.52:6443 --token i42a0m.53sfsomivcdo5eh2 \
    --discovery-token-ca-cert-hash sha256:73318c56d5b348cced0f2f3f9c3ea260622d7e2e7a14225d3955153950a4c468
```

在`master0`上查看

```
[root@master0 ~]# kubectl get node
NAME                 STATUS     ROLES                  AGE    VERSION
master0.example.com  NotReady   control-plane,master   3m4s   v1.21.11
work0.example.com    NotReady   <none>                 98s    v1.21.11
work1.example.com    NotReady   <none>                 95s    v1.21.11
```

当前的状态为`NotReady`，所以需要安装`calico`网络组件

```
[root@master0 ~]# vim calico.yaml
```

把`calico.yaml`文件里面的`docker.io`全部替换成`registry.example.com`

```
[root@master0 ~]# kubectl apply -f calico.yaml
```

验证

```
[root@master0 ~]#  kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-7d7b59d5b7-8fktt   1/1     Running   0          4m21s
calico-node-hx4hw                          1/1     Running   0          4m21s
calico-node-mnnr6                          1/1     Running   0          4m21s
calico-node-vk2bj                          1/1     Running   0          4m21s
coredns-5c7c5d548-2frlv                    1/1     Running   0          23s
coredns-5c7c5d548-dpt2k                    1/1     Running   0          47s
etcd-master0.example.com                   1/1     Running   1          20m
kube-apiserver-master0.example.com         1/1     Running   1          20m
kube-controller-manager-master0.example.com   1/1     Running   1          20m
kube-proxy-j82m6                           1/1     Running   1          20m
kube-proxy-rtgh5                           1/1     Running   0          19m
kube-proxy-wwz6l                           1/1     Running   0          19m
kube-scheduler-master0.example.com         1/1     Running   1          20m
[root@master0 ~]# kubectl get node
NAME                 STATUS   ROLES                  AGE   VERSION
master0.example.com   Ready    control-plane,master   21m   v1.21.11
work0.example.com     Ready    <none>                 19m   v1.21.11
work1.example.com     Ready    <none>                 19m   v1.21.11
[root@master0 ~]#
```

至此，节点集群搭建成功

4.6 创建永久Token

以下步骤在`master0`主机上执行

```
[root@master0 ~]# kubeadm token create --ttl 0 --print-join-command
kubeadm join 192.168.2.52:6443 --token 1ip1tv.bfyi5ljm6tfad158 --discovery-token-ca-cert-hash sha256:73318c56d5b348cced0f2f3f9c3ea260622d7e2e7a14225d3955153950a4c468
```

5. Prometheus配置

5.1 安装prometheus

在`manager`主机上执行

```
[root@manager /]# mkdir apps
[root@manager /]# cd apps
[root@manager apps]# tar xzvf prometheus-2.34.0.linux-amd64.tar.gz
[root@manager apps]# mkdir packages
[root@manager apps]# mv prometheus-2.34.0.linux-amd64.tar.gz packages/

[root@manager apps]# mv prometheus-2.34.0.linux-amd64 prometheus

[root@manager apps]# ll
total 85580
-rw-r--r-- 1 root root 87630471 Apr 30 00:05 grafana-enterprise-8.5.0.linux-amd64.tar.gz
drwxr-xr-x 2 root root       50 Apr 30 00:07 packages
drwxr-xr-x 4 3434 3434      132 Mar 15 23:33 prometheus


```

```
[root@manager ~]# vim /usr/lib/systemd/system/prometheus.service
```

```
2
3  [Unit]
4  Descripition=prometheus
5
6  [Service]
7  ExecStart=/apps/prometheus/prometheus --
   config.file=/apps/prometheus/prometheus.yml
8  ExecReload=/bin/kill -HUP $MAINPID
9  KillMode=process
10 Restart=on-failure
11
12 [Install]
13 WantedBy=multi-user.target
14
15
16 [root@manager ~]# systemctl start prometheus.service
17 [root@manager ~]# systemctl enable prometheus.service
```

`manager`

```
1  [root@manager prometheus]# vim prometheus.yml
2  添加下面的
3
4      static_configs:
5        - targets: ["localhost:9090"]
6
7    - job_name: "system-base-information"
8      static_configs:
9        - targets:
   ["192.168.2.52:9100","192.168.2.53:9100","192.168.2.54:9100"]
10
```

以下操作在`manager0`、`work0`、`work1`上执行

```
1  [root@master0 apps]# tar xzvf node_exporter-1.3.1.linux-amd64.tar.gz
2
3  [root@master0 apps]# mkdir package
4  [root@master0 apps]# mv node_exporter-1.3.1.linux-amd64.tar.gz package/
5  [root@master0 apps]# mv node_exporter-1.3.1.linux-amd64 node_exporter
6  [root@master0 apps]# cd node_exporter/
7  [root@master0 node_exporter]# ll
8
9  [root@master0 node_exporter]# vim
   /usr/lib/systemd/system/node_exporter.service
10 [Unit]
11 Description=node_exporter
12
13 [Service]
14 ExecStart=/apps/node_exporter/node_exporter
15 ExecReload=/bin/kill -HUP $MAINPID
16 KillMode=process
17 Restart=on-failure
18
```

```
[Install]
WantedBy=multi-user.target

[root@master0 node_exporter]# systemctl daemon-reload
[root@master0 node_exporter]# systemctl restart node_exporter.service
[root@master0 node_exporter]# systemctl enable node_exporter.service
```

6. grafana配置

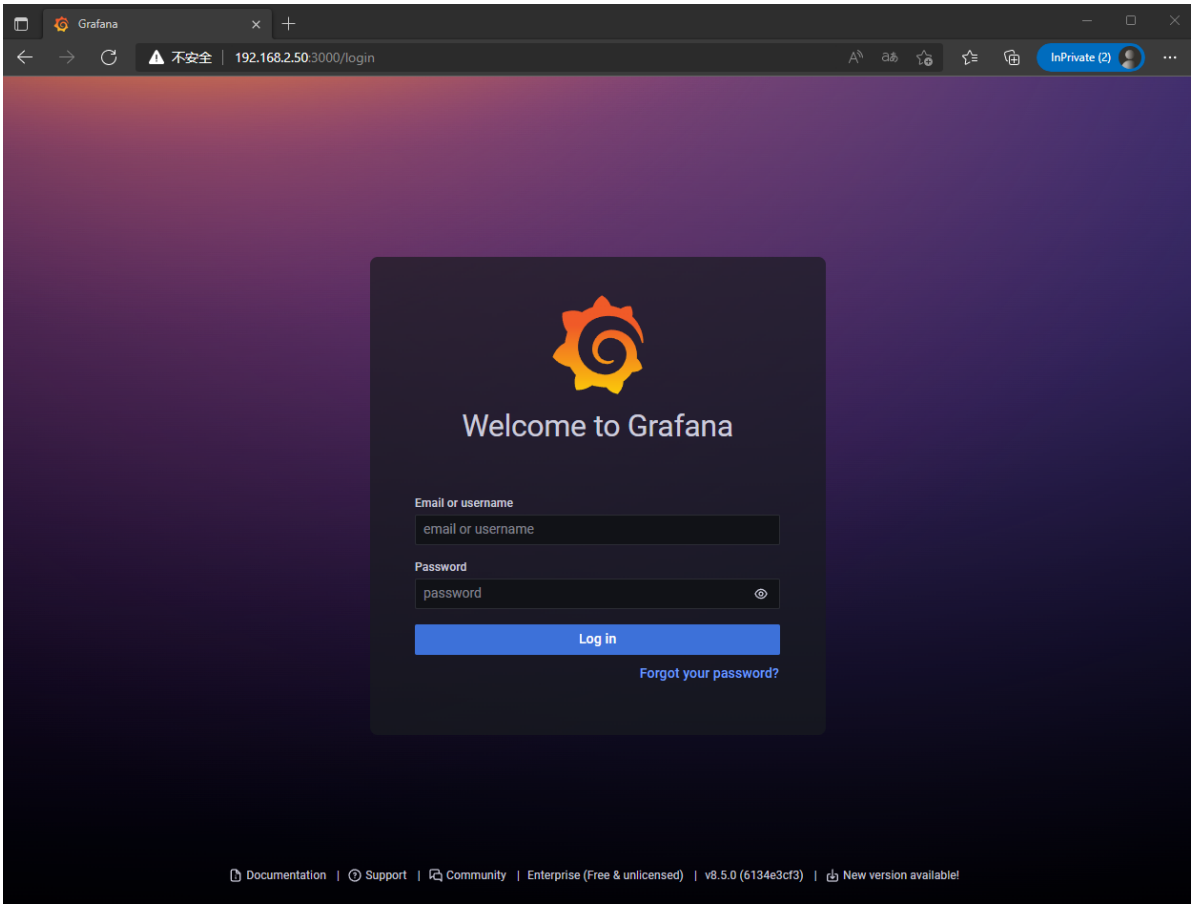
以下操作在`manager`主机上执行

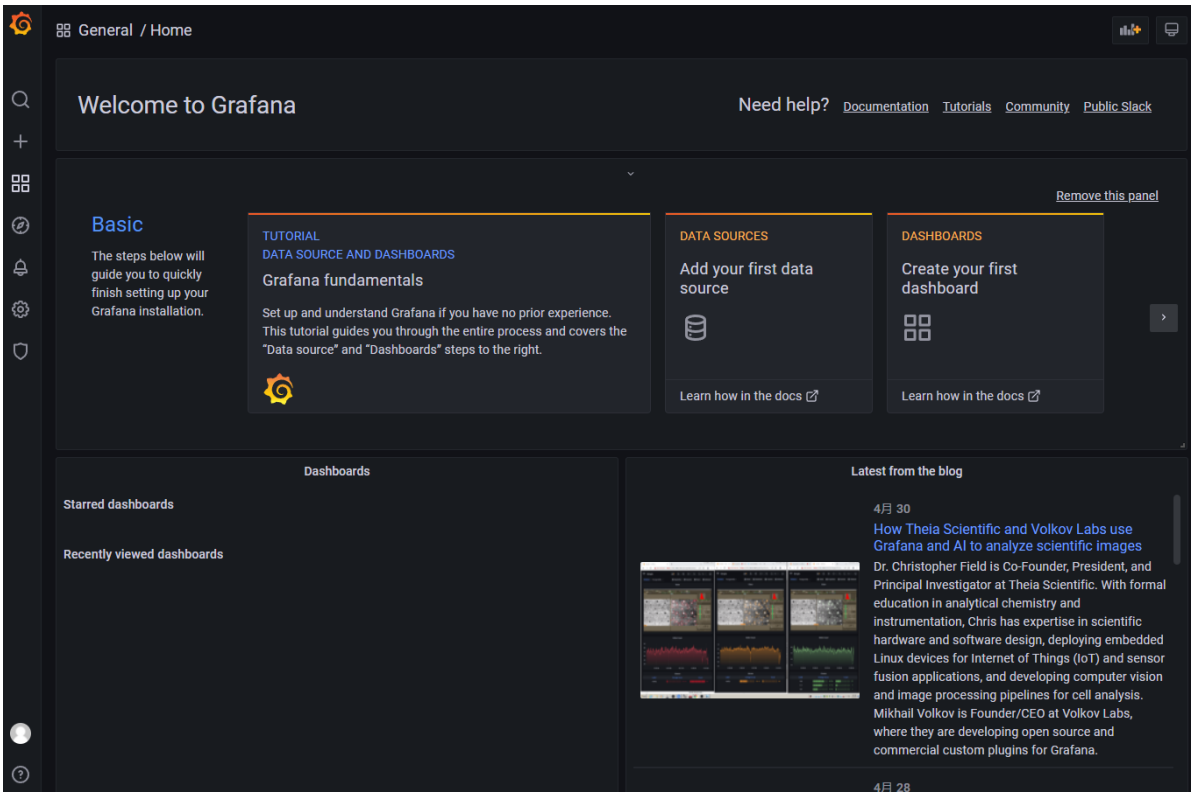
```
tar xzvf grafana-enterprise-8.5.0.linux-amd64.tar.gz
mv grafana-enterprise-8.5.0.linux-amd64.tar.gz packages/
mv grafana-8.5.0 grafana

# vim /usr/lib/systemd/system/grafana.service
[Unit]
Description=grafana

[Service]
ExecStart=/apps/grafana/bin/grafana-server -homepath=/apps/grafana
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
wantedBy=multi-user.target

[root@manager ~]# systemctl daemon-reload
[root@manager ~]# systemctl restart grafana.service
[root@manager ~]# systemctl enable grafana.service

```

6.1 登录grafana

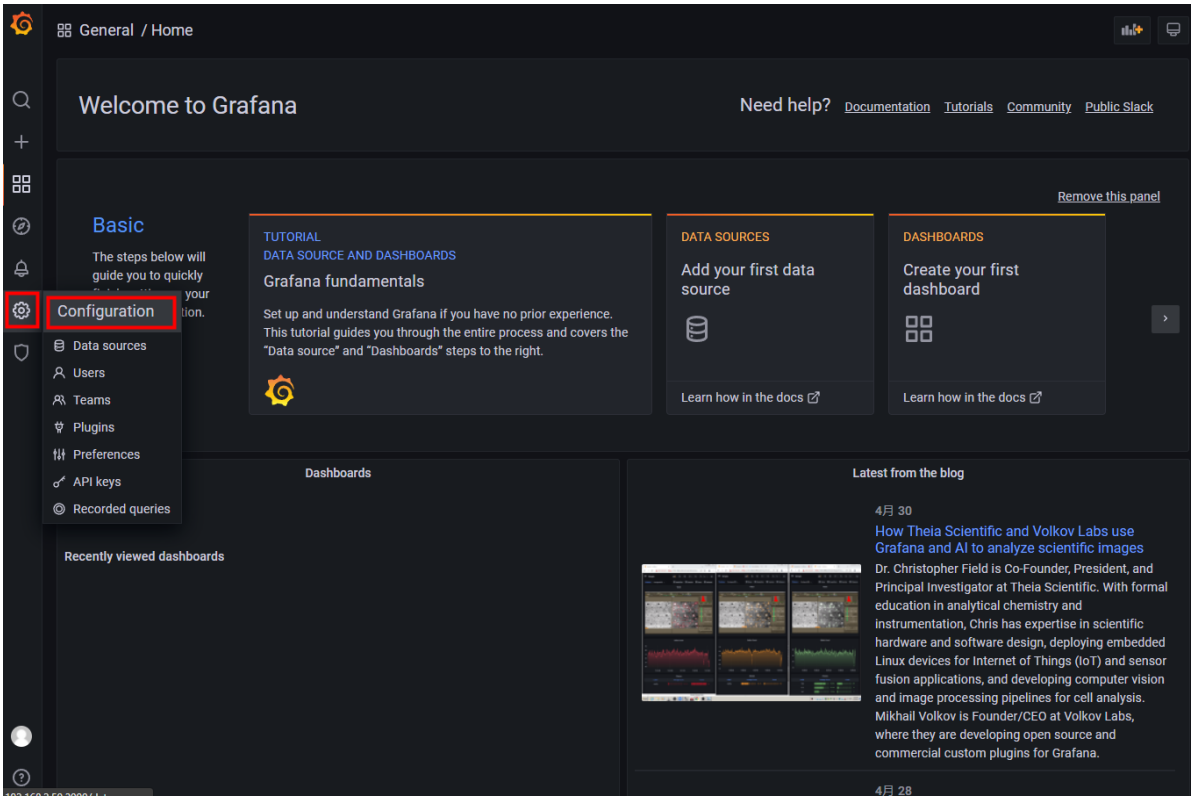
在浏览器输入：`192.168.2.50:3000`

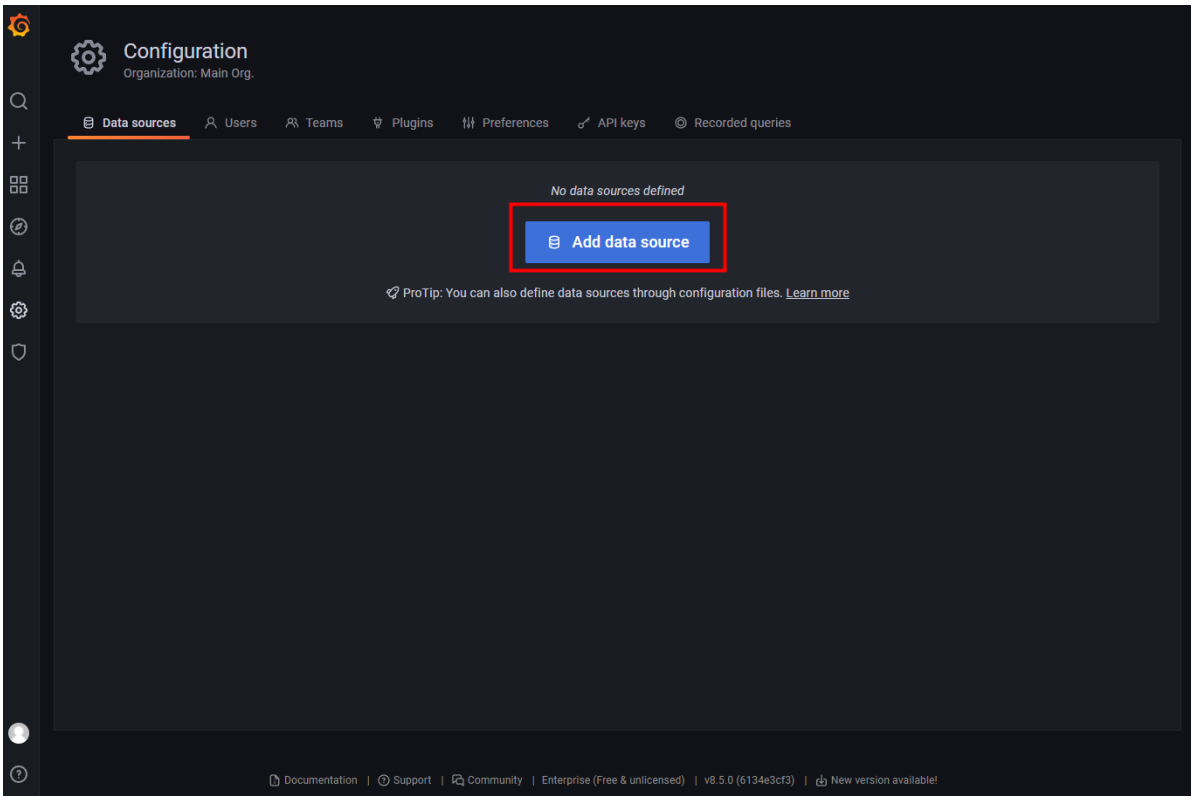
用户名/密码：admin/admin123

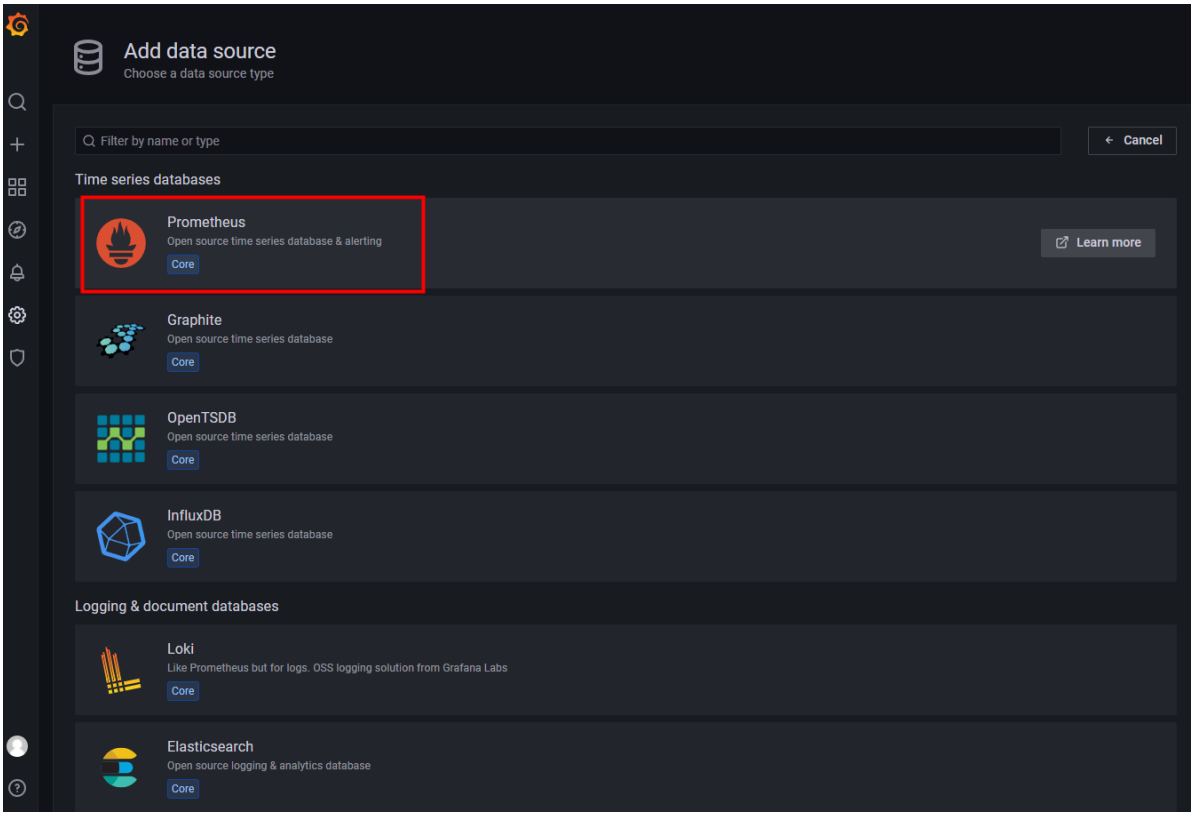


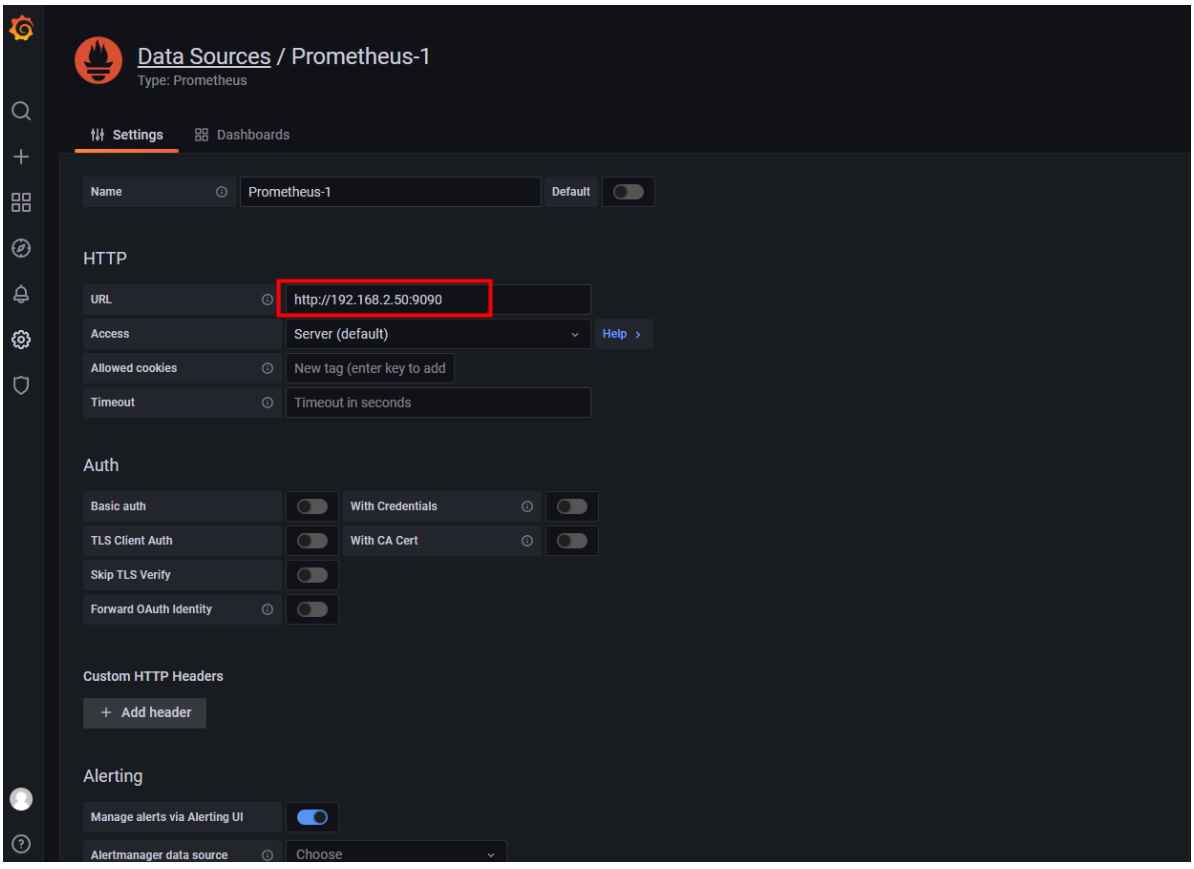


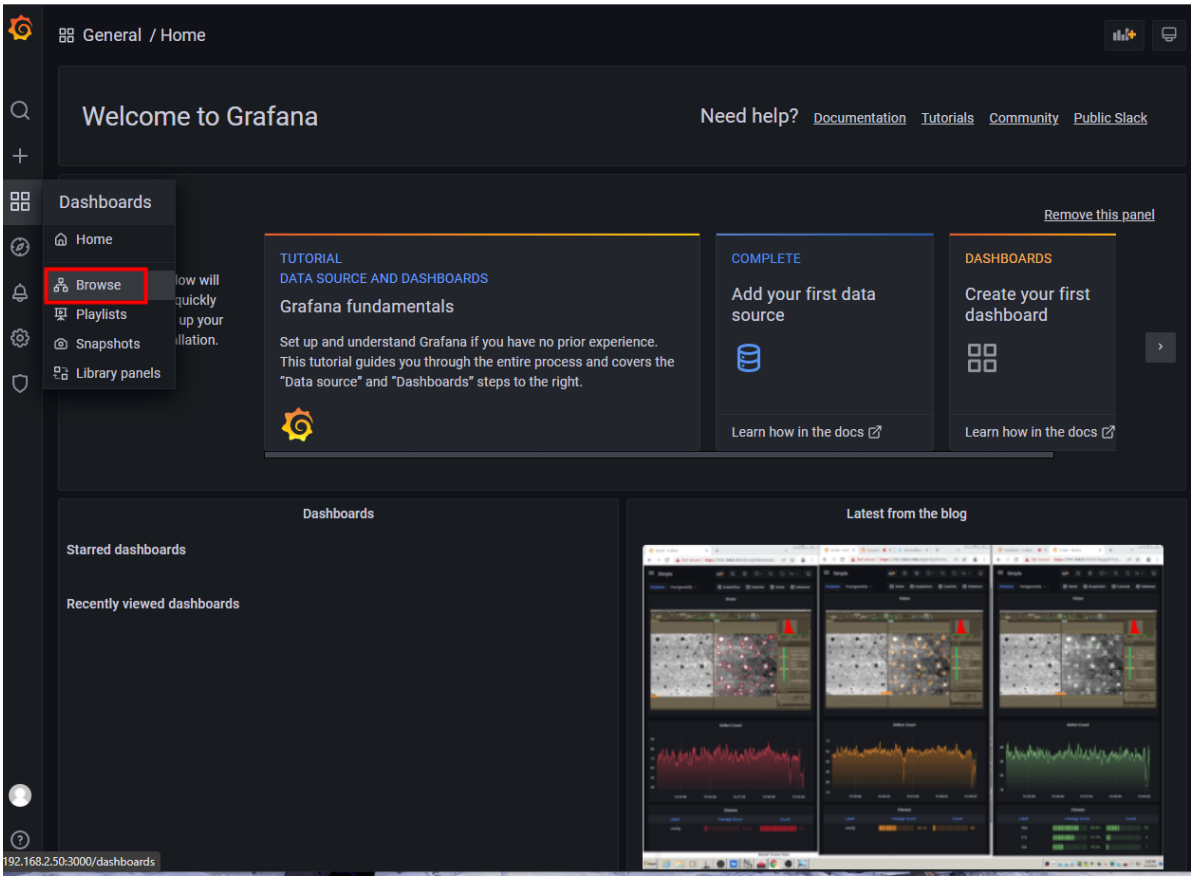
6.2 添加Prometheus

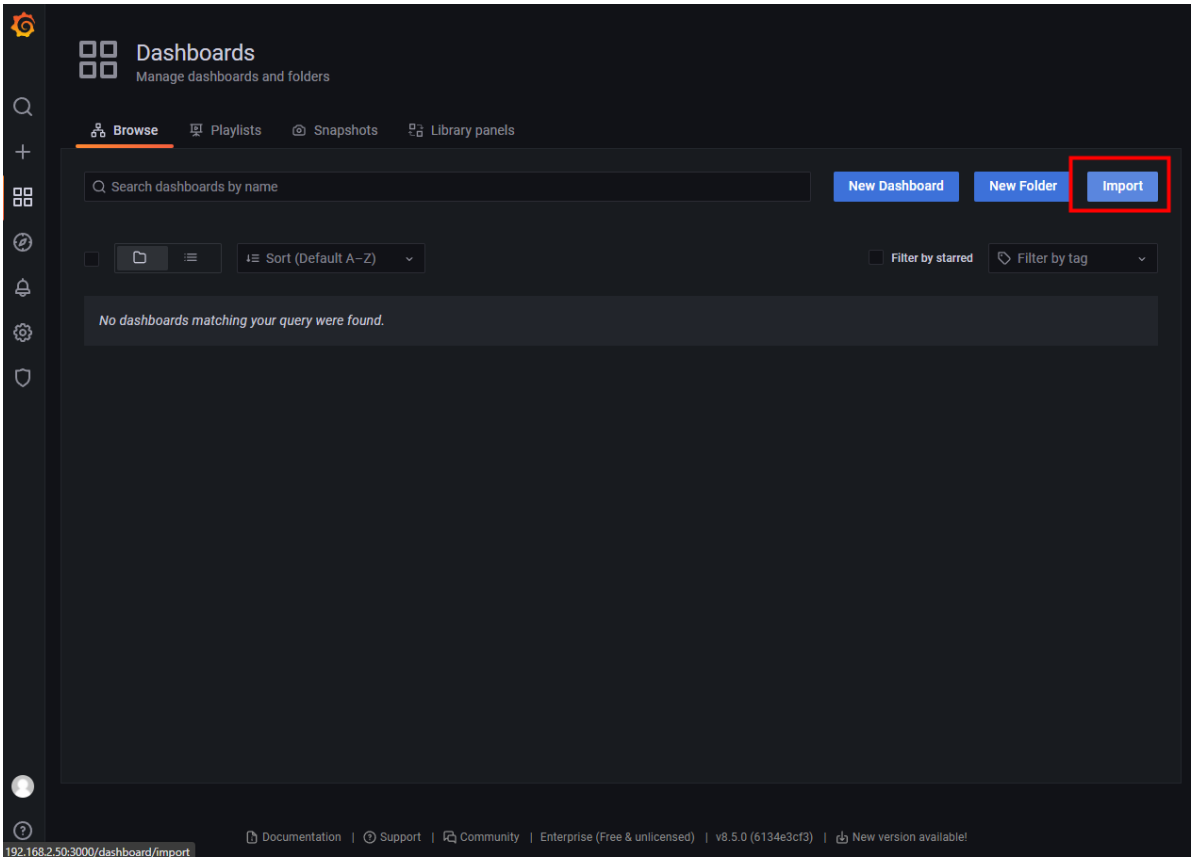

General / Home
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Remove this panel
Basic
The steps below will guide you to quickly
Configuration
Data sources
Users
Teams
Plugins
Preferences
API keys
Recorded queries
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
DATA SOURCES
Add your first data source
Learn how in the docs
DASHBOARDS
Create your first dashboard
Learn how in the docs
Dashboards
Recently viewed dashboards
Latest from the blog
4月 30
How Theia Scientific and Volkov Labs use Grafana and AI to analyze scientific images
Dr. Christopher Field is Co-Founder, President, and Principal Investigator at Theia Scientific. With formal education in analytical chemistry and instrumentation, Chris has expertise in scientific hardware and software design, deploying embedded Linux devices for Internet of Things (IoT) and sensor fusion applications, and developing computer vision and image processing pipelines for cell analysis. Mikhail Volkov is Founder/CEO at Volkov Labs, where they are developing open source and commercial custom plugins for Grafana.
4月 28

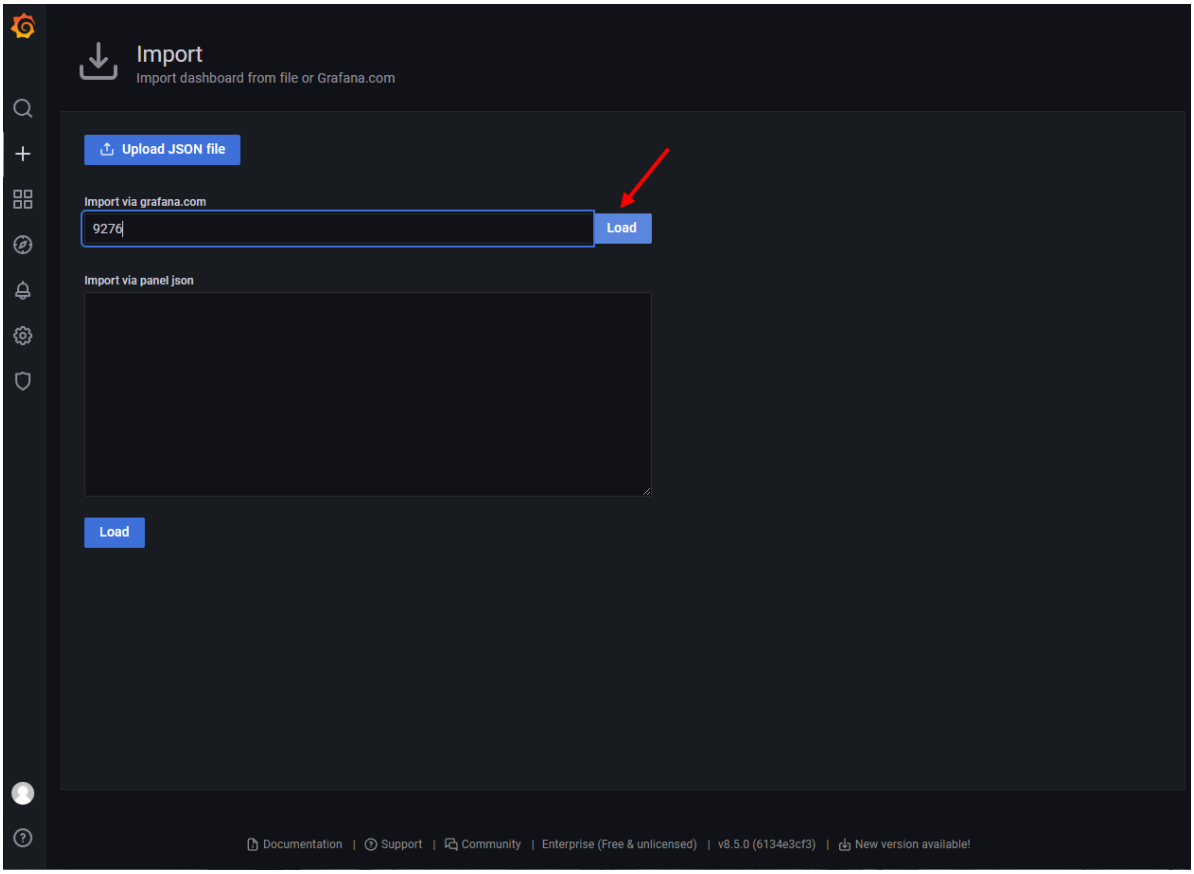

Configuration
Organization: Main Org.
Data sources
Users
Teams
Plugins
Preferences
API keys
Recorded queries
No data sources defined
Add data source
ProTip: You can also define data sources through configuration files. Learn more
Documentation | Support | Community | Enterprise (Free & unlicensed) | v8.5.0 (6134e3cf3) | New version available!

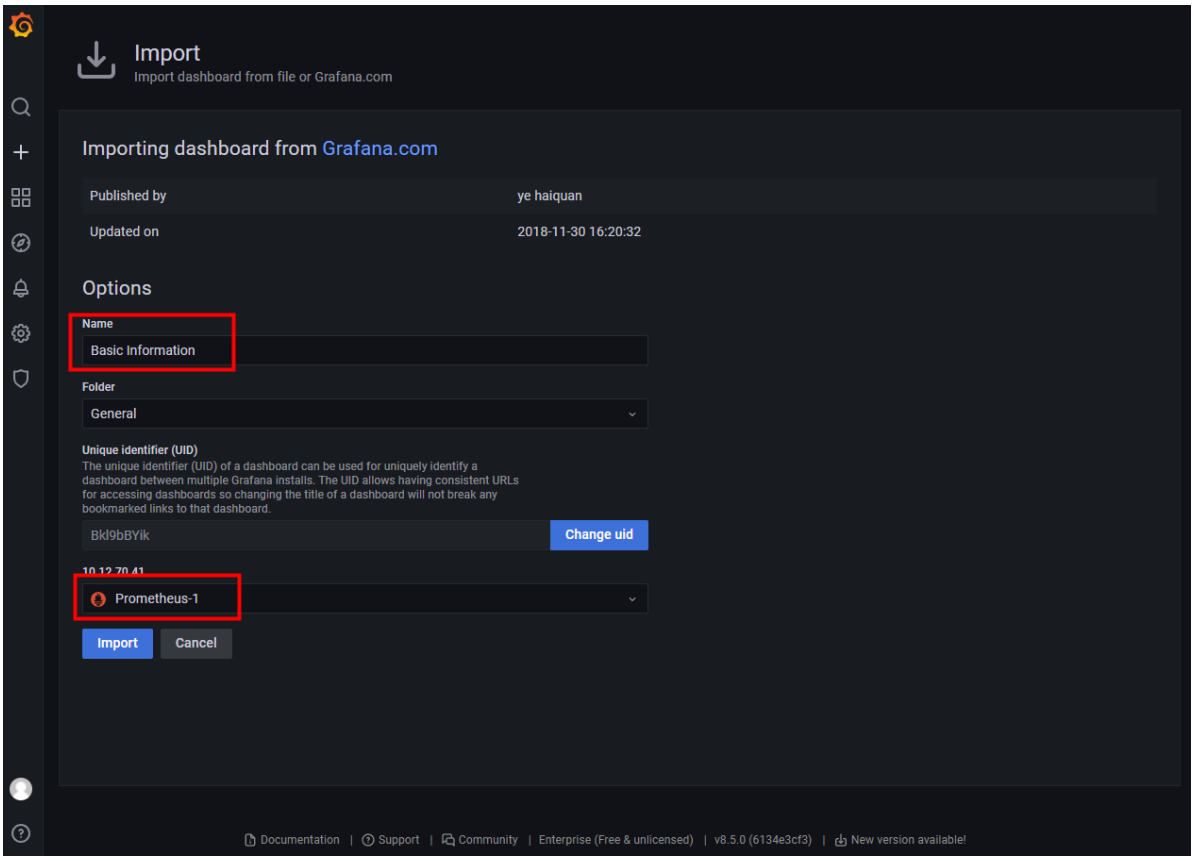

Add data source
Choose a data source type
Filter by name or type
Cancel
Time series databases
Prometheus
Open source time series database & alerting
Core
Learn more
Graphite
Open source time series database
Core
OpenTSDB
Open source time series database
Core
InfluxDB
Open source time series database
Core
Logging & document databases
Loki
Like Prometheus but for logs. OSS logging solution from Grafana Labs
Core
Elasticsearch
Open source logging & analytics database
Core

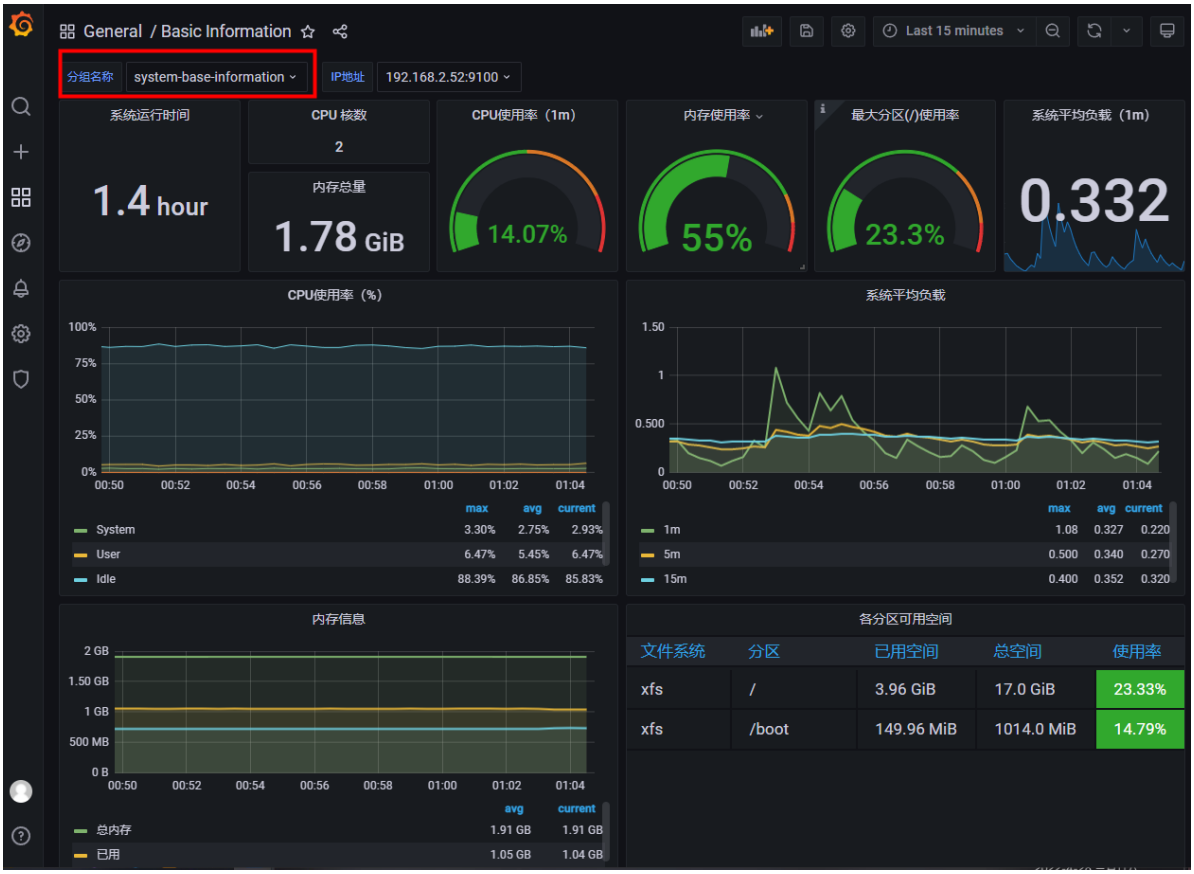

Data Sources / Prometheus-1
Type: Prometheus
Settings
Dashboards
Name
Prometheus-1
Default
HTTP
URL
http://192.168.2.50:9090
Access
Server (default)
Help
Allowed cookies
New tag (enter key to add
Timeout
Timeout in seconds
Auth
Basic auth
With Credentials
TLS Client Auth
With CA Cert
Skip TLS Verify
Forward OAuth Identity
Custom HTTP Headers
Add header
Alerting
Manage alerts via Alerting UI
Alertmanager data source
Choose

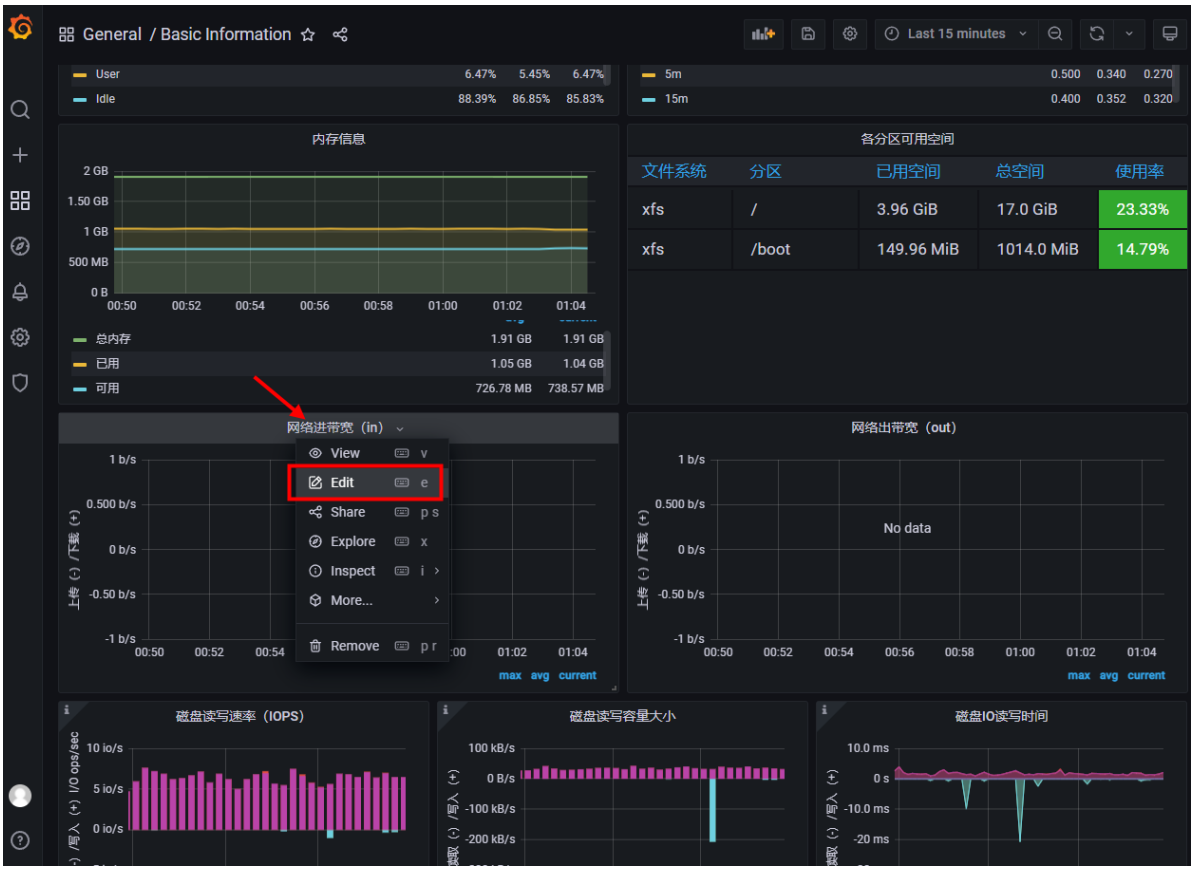

General / Home
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Dashboards
Home
Browse
Playlists
Snapshots
Library panels
Remove this panel
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
COMPLETE
Add your first data source
Learn how in the docs
DASHBOARDS
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Recently viewed dashboards
Latest from the blog
192.168.2.50:3000/dashboards

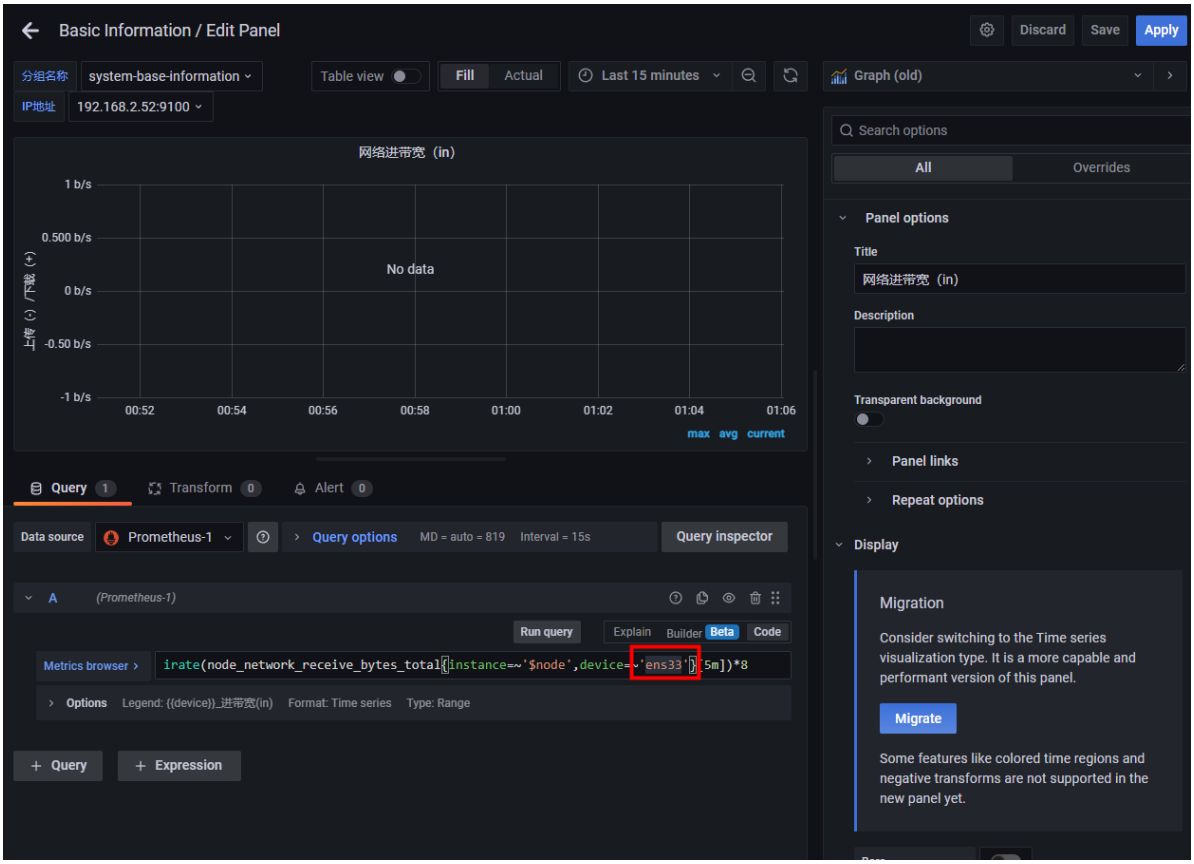

Dashboards
Manage dashboards and folders
Browse
Playlists
Snapshots
Library panels
Search dashboards by name
New Dashboard
New Folder
Import
Sort (Default A–Z)
Filter by starred
Filter by tag
No dashboards matching your query were found.
Documentation | Support | Community | Enterprise (Free & unlicensed) | v8.5.0 (6134e3cf3) | New version available!
192.168.2.50:3000/dashboard/import


Import
Import dashboard from file or Grafana.com
Upload JSON file
Import via grafana.com
9276
Load
Import via panel json
Load
Documentation | Support | Community | Enterprise (Free & unlicensed) | v8.5.0 (6134e3cf3) | New version available!


Import
Import dashboard from file or Grafana.com
Importing dashboard from Grafana.com
Published by
ye haiquan
Updated on
2018-11-30 16:20:32
Options
Name
Basic Information
Folder
General
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
Bkl9bBYik
Change uid
Prometheus-1
Import
Cancel
Documentation | Support | Community | Enterprise (Free & unlicensed) | v8.5.0 (6134e3cf3) | New version available!


General / Basic Information
Last 15 minutes
分组名称
system-base-information
IP地址
192.168.2.52:9100
系统运行时间
1.4 hour
CPU 核数
2
内存总量
1.78 GiB
CPU使用率（1m）
14.07%
内存使用率
55%
最大分区(/)使用率
23.3%
系统平均负载（1m）
0.332
CPU使用率（%）
System 3.30% 2.75% 2.93%
User 6.47% 5.45% 6.47%
Idle 88.39% 86.85% 85.83%
系统平均负载
1m 1.08 0.327 0.220
5m 0.500 0.340 0.270
15m 0.400 0.352 0.320
内存信息
总内存 1.91 GB 1.91 GB
已用 1.05 GB 1.04 GB
各分区可用空间
文件系统 分区 已用空间 总空间 使用率
xfs / 3.96 GiB 17.0 GiB 23.33%
xfs /boot 149.96 MiB 1014.0 MiB 14.79%


General / Basic Information
Last 15 minutes
User 6.47% 5.45% 6.47%
Idle 88.39% 86.85% 85.83%
5m 0.500 0.340 0.270
15m 0.400 0.352 0.320
内存信息
总内存 1.91 GB 1.91 GB
已用 1.05 GB 1.04 GB
可用 726.78 MB 738.57 MB
各分区可用空间
文件系统 分区 已用空间 总空间 使用率
xfs / 3.96 GiB 17.0 GiB 23.33%
xfs /boot 149.96 MiB 1014.0 MiB 14.79%
网络进带宽（in）
View v
Edit e
Share p s
Explore x
Inspect i
More...
Remove p r
网络出带宽（out）
No data
磁盘读写速率（IOPS）
磁盘读写容量大小
磁盘IO读写时间


Basic Information / Edit Panel
Discard
Save
Apply
分组名称
system-base-information
Table view
Fill
Actual
Last 15 minutes
Graph (old)
IP地址
192.168.2.52:9100
网络进带宽（in）
1 b/s
0.500 b/s
0 b/s
-0.50 b/s
-1 b/s
No data
上传（-）/下载（+）
00:52
00:54
00:56
00:58
01:00
01:02
01:04
01:06
max avg current
Query 1
Transform 0
Alert 0
Data source
Prometheus-1
Query options
MD = auto = 819
Interval = 15s
Query inspector
A
(Prometheus-1)
Run query
Explain
Builder
Beta
Code
Metrics browser
irate(node_network_receive_bytes_total{instance=~'$node',device=~'ens33'}[5m])*8
Options
Legend: {{device}}_进带宽(in)
Format: Time series
Type: Range
+ Query
+ Expression
Search options
All
Overrides
Panel options
Title
网络进带宽（in）
Description
Transparent background
Panel links
Repeat options
Display
Migration
Consider switching to the Time series visualization type. It is a more capable and performant version of this panel.
Migrate
Some features like colored time regions and negative transforms are not supported in the new panel yet.