# ###监控安装逻辑###

内部通过商店设置中的商店下载对应的Prometheus Chart，然后通过helm实现安装。

集群中下载Chart是通过local中的rancher Pod下载，务必要保证rancher与商店设置中的下载源互通。

如果是持久化安装，首先需要先安装local-path-provisioner容器，使其可以动态的生成并创建对应的pv提供给Prometheus使用。

部署完成后可以在集群主机点击grafana查看到创建监控后监控的内容，因为需要在不同的节点上创建node-exporter所以会出现数据延迟的情况，耐心等待即可。

# 连接到外网安装

## 操作步骤

工具🡪监控🡪确定安装的信息🡪开启监控



按照上述步骤安装即可，安装完成后可以通过cattle-prometheus名称空间查看到prometheus创建的Pod及相应的工作资源。

# 内网安装

## 操作步骤

Clone chart到本地 🡪 创建gitlab 🡪 商店设置 🡪 修改system-library的chart源 🡪 指定私有仓库

## Clone chart到本地，创建gitlab

Rancher官方gitee chart源：<https://gitee.com/rancher/system-charts>

在本地主机使用

git clone -b release-v2.3 <https://gitee.com/rancher/system-charts> 将代码 clone 到本地，然后再通过一定手段将代码项目分支传入到机器中。

指定chart仓库，并且只clone release-v2.3版本的charts。

创建 gitlab，启动gitlab容器

mkdir /app/gitlab/{config,data,logs} -p

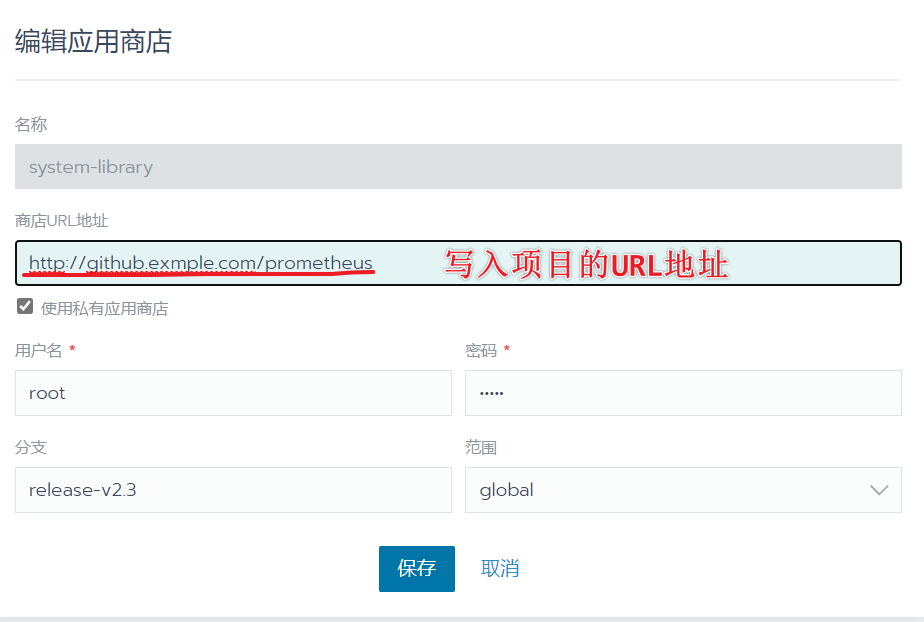
export GITLAB\_HOME=/srv/gitlab

sudo docker run –detach --hostname gitlab.example.com --publish 443:443 --publish 80:80 –publish 222:22 –name gitlab --restart always --volume $GITLAB\_HOME/config:/app/gitlab/config/ --volume $GITLAB\_HOME/logs:/app/gitlab/logs --volume $GITLAB\_HOME/data:/app/gitlab/data gitlab/gitlab-ee:latest

打开gitlab地址对项目进行创建，创建对应的用户名账号。

## 商店设置

节点配置gilab的用户名密码，之后将代码push到gitlab分支中，修改商店设置chart源，刷新商店设置源，刷新源后如果没有问题，即可开始安装Prometheus。



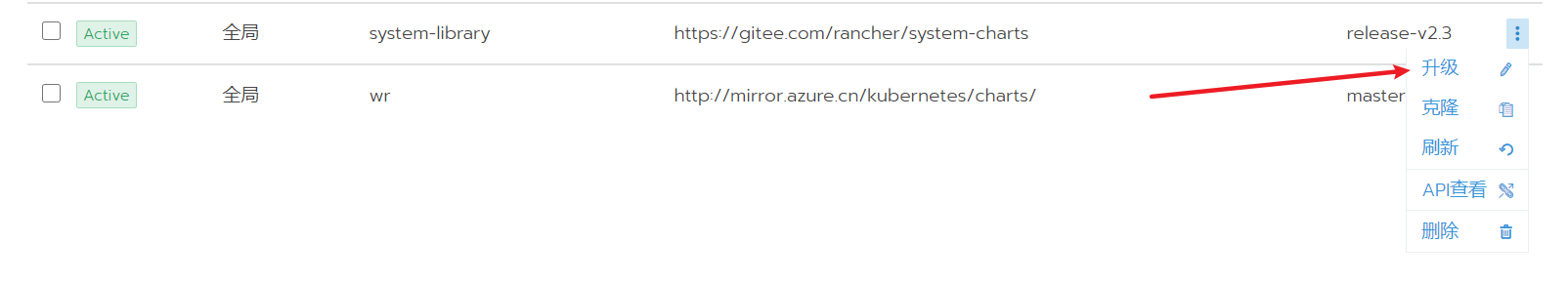
# 联网安装

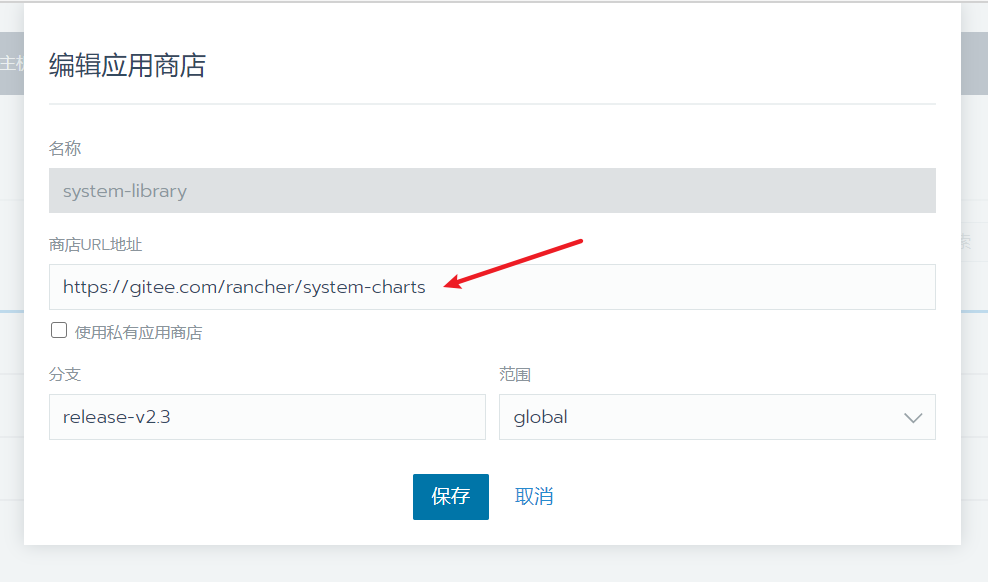
## 操作步骤

修改chart源 🡪 刷新源 🡪 安装prometheus

## 修改chart源

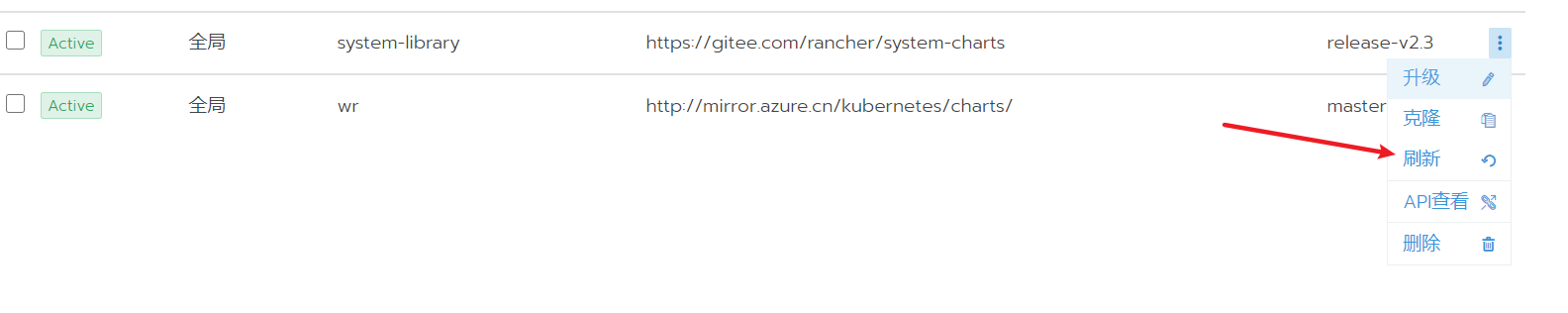
因为默认的github可能会因为某些原因连接超时，可以更换为rancher官方的gitee源。





## 刷新源

刷新过程中rancher会主动的尝试连接chart源，连接成功则继续显示为active。



## 安装Prometheus

进入监控界面后选择启用即可。



# 持久化监控数据（指定主机/不指定）

**Prometheus持久化：**会将Prometheus保存到时序数据库中的监控数据保存，如果没有持久化则默认只保存15天的监控数据，超过15天后则就会被删除。

**Grafana持久化：**保存Grafana下载安装的pannel，如果没有备份，那么当你重启Grafana时，pannel就会丢失（Rancher基于chart安装的Grafana会自动安装Pannel所以不需要持久化）。

注意：

1. 在创建监控定义LabelSelector时，其中的键名与键值要与添加的label相同。

## 操作步骤

创建local-path-provisioner 🡪 修改chart源 🡪 添加节点label 🡪 安装监控

## 创建local-path-provisioner

创建local-path-provisioner，local-path-provisioner的作用就是在prometheus在想要创建persistentvolume时local-path-provisioner会自动的创建一个单独的Pod来通过busybox创建出对应的persistentvolume，最终绑定到prometheus之上。（创建busybox的主机取决于label selector或者是affinity调度）

1. **Wget到本地**

wget <https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml>

1. **修改镜像源**

因为是内网的服务需要搭建，但是docker拉取不到外网的镜像，所以要进行修改。

修改项：

export IMAGEURL=[镜像库地址] #例如南京的就是jxknj.yw.dzqd.cn

export IMAGEURL=jxknj.yw.dzqd.cn

sed -Ei -e "s@(.\*image:) (busybox)@\1 rancher/\2@" -e "s@(.\*image:) (.\*)@\1 ${IMAGEURL}/\2@" local-path-storage.yaml

1. **应用该yaml创建相应资源**

cat local-path-storage.yaml #手工复制内容保留

通过cat命令查看到该yaml文件内容复制到rancher的图形化界面导入即可。

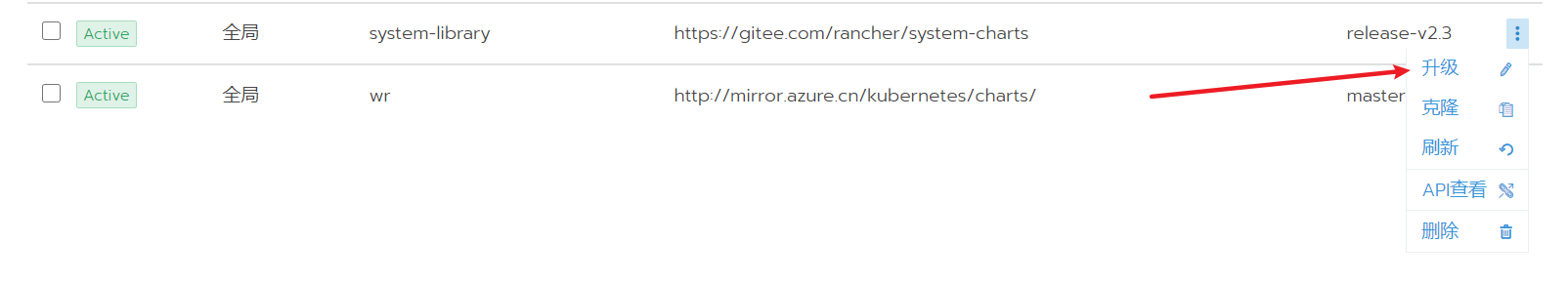


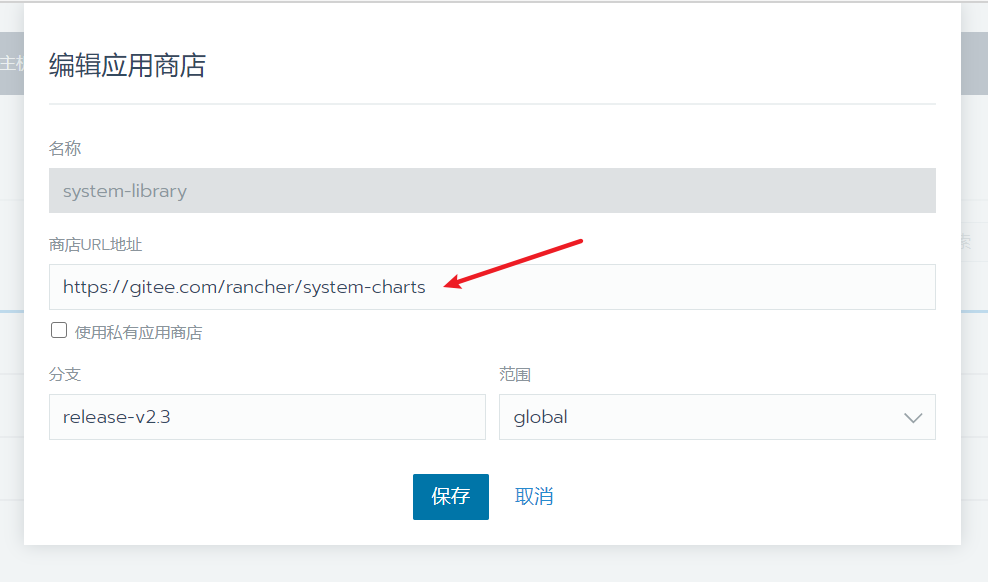


或配置kubectl命令进行创建。

## 修改chart源

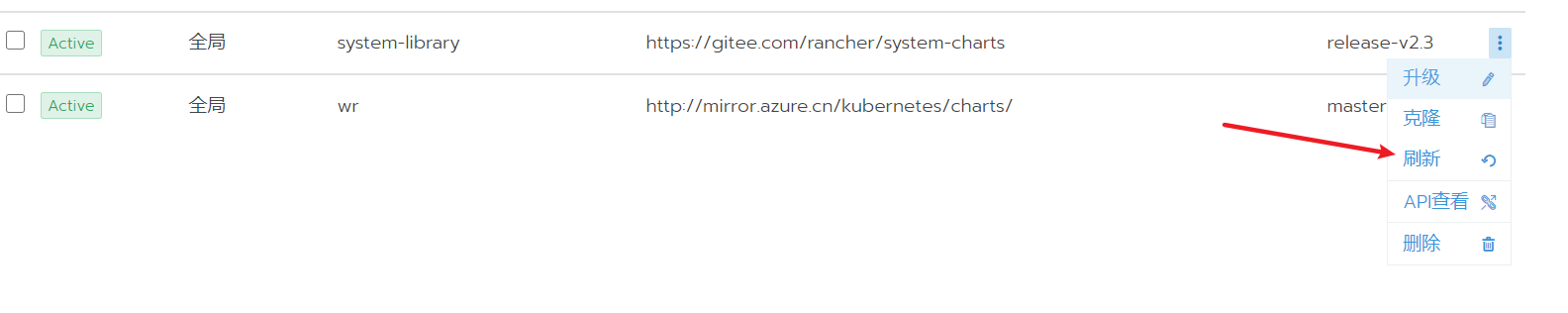
因为默认的github可能会因为某些原因连接超时，可以更换为rancher官方的gitee源。





## 刷新源

刷新过程中rancher会主动的尝试连接chart源，连接成功则继续显示为active。



## 添加节点label

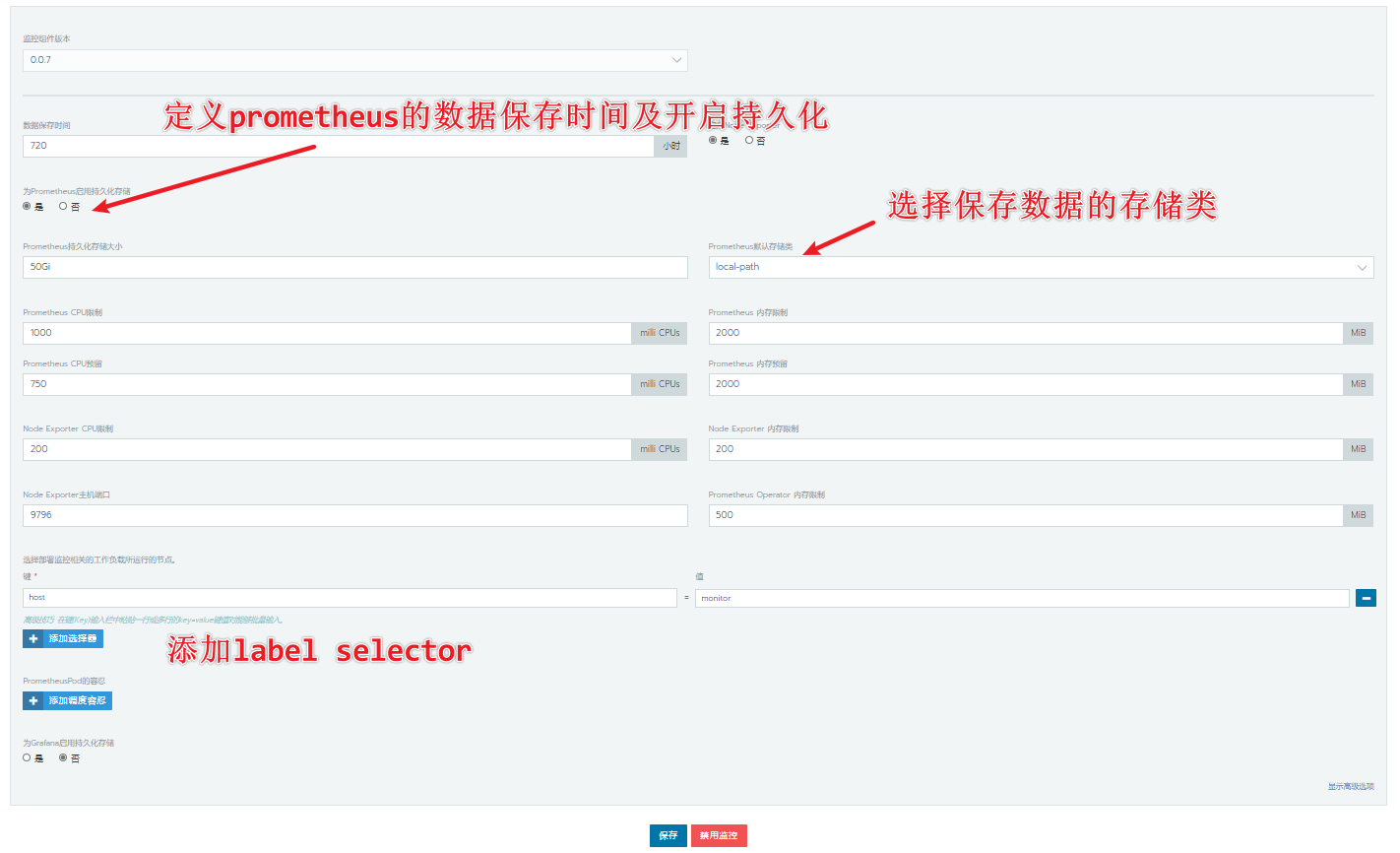
注意：该步骤只是在需要指定到指定节点保存持久化数据时使用。

使用命令添加label。

kubectl label node [NODE\_NAME] host=monitor

## 创建监控

注意：Label Selector只是在需要指定节点保存数据时使用，如果不需要指定节点，直接创建即可。



# 常见错误

## 部署监控拉取chart超时

排错思路：可以查看local集群中的rancher日志，商店功能是由rancher最终操作的，其中会有报错产生。如果是网络原因可以尝试登录到节点之上测试网络连通性。

通过应用商店可以看到正在安装的应用。

## 创建Prometheus后[prometheus-cluster-monitoring](https://rqynj.yw.dzqd.cn/p/c-zqspg:p-4c662/workloads/cattle-prometheus:prometheus-cluster-monitoring-0)容器报错

该错误主要是因为容器没有权限无法写入到创建的persistentvolumeclaim，修改statefulset即可。

caller=main.go:731 err=“opening storage failed: create dir: mkdir /prometheus/wal: permission denied”

由于Context权限导致该问题的产生，修改Context设置后即可。

[shadm1@dqynj139109 ~]$ ./kubectl edit statefulset prometheus-cluster-monitoring --kubeconfig=kubeconfig -n cattle-prometheus

securityContext:

fsGroup: 0

runAsNonRoot: false

runAsUser: 0