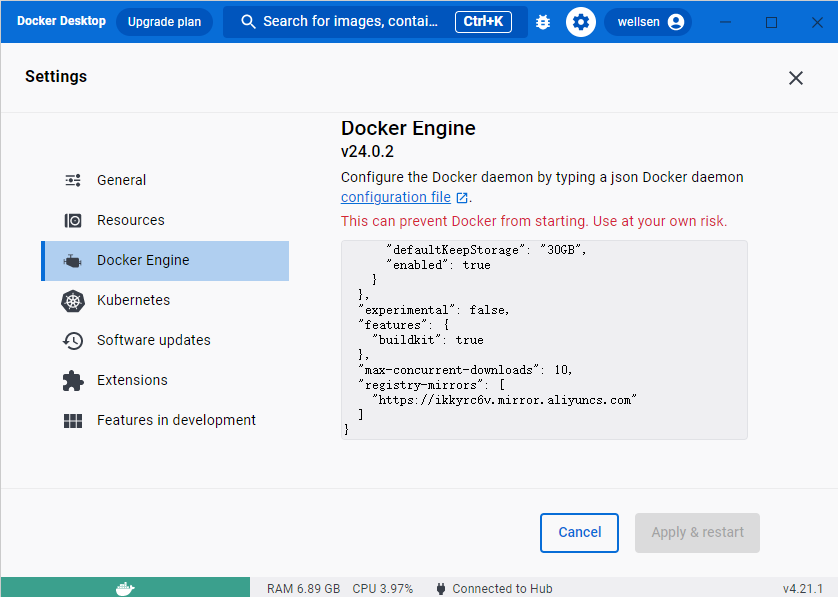
k8s本地开发minikube环境搭建步骤

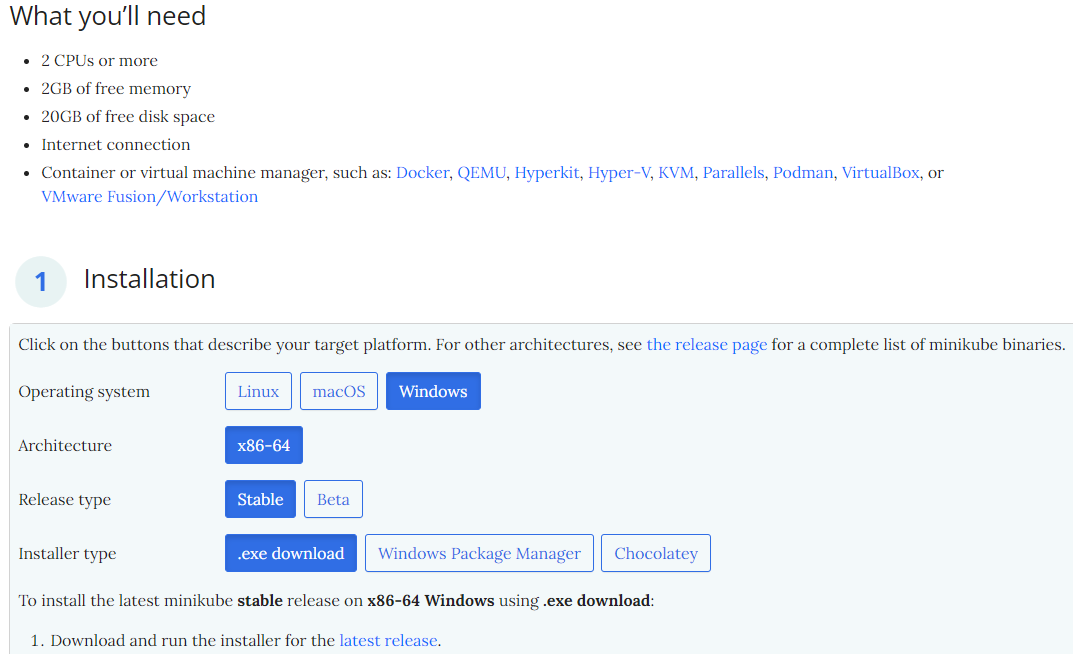
根据k8s官网[Kubernetes Documentation/Getting started](https://kubernetes.io/docs/setup/)文档Learning environment小结介绍，存在多种k8s本地测试环境搭建工具，例如Kind、Minikube、K3S等，这里我们采用Minikube工具来搭建测试环境。

## 搭建环境前配置要求

1.1安装docker desktop应用，并配置好"registry-mirrors"，这里我们使用阿里云镜像站



1.2 从官网[minikube start](https://minikube.sigs.k8s.io/docs/start/)文档中下载适用于Windows版本的Minikube工具minikube-installer.exe



下载完成后，运行安装文件minikube-installer.exe，安装成功后默认安装路径为C:\Program Files\Kubernetes\Minikube。将上述路径加入到系统环境变量中，方便后续执行minikube命令

## 启动K8S测试集群

Minikube安装完成后，可以直接使用下面命令启动一个新的集群

minikube start

如果一切顺利，那么你可以得到一个默认名为minikube的集群，命令执行结果如下：

C:\Users>minikube start  
\* Microsoft Windows 10 Enterprise 10.0.19045.3208 Build 19045.3208 上的 minikube v1.30.1  
\* 自动选择 docker 驱动  
\* Using Docker Desktop driver with root privileges  
\* Starting control plane node minikube in cluster minikube  
\* Pulling base image ...  
\* Downloading Kubernetes v1.26.3 preload ...  
> preloaded-images-k8s-v18-v1...: 397.02 MiB / 397.02 MiB 100.00% 4.43 Mi  
> index.docker.io/kicbase/sta...: 373.53 MiB / 373.53 MiB 100.00% 3.44 Mi  
! minikube was unable to download gcr.io/k8s-minikube/kicbase:v0.0.39, but successfully downloaded docker.io/kicbase/stable:v0.0.39 as a fallback image  
\* Creating docker container (CPUs=2, Memory=4000MB) ...  
\* 正在 Docker 23.0.2 中准备 Kubernetes v1.26.3…  
- Generating certificates and keys ...  
- Booting up control plane ...  
- 配置 RBAC 规则 ...  
\* Configuring bridge CNI (Container Networking Interface) ...  
- Using image gcr.io/k8s-minikube/storage-provisioner:v5  
\* Verifying Kubernetes components...  
\* Enabled addons: storage-provisioner, default-storageclass  
\* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

当然，你可以可以指定集群名称和K8S的版本，样例命令如下：

minikube start -p clusterName --kubernetes-version=v1.20.1

如果，你的网络下载集群启动所需镜像失败的话，可以参考minikube文档[Proxies and VPNs](https://minikube.sigs.k8s.io/docs/handbook/vpn_and_proxy/)中设置代理的相关介绍，如下截图所示：

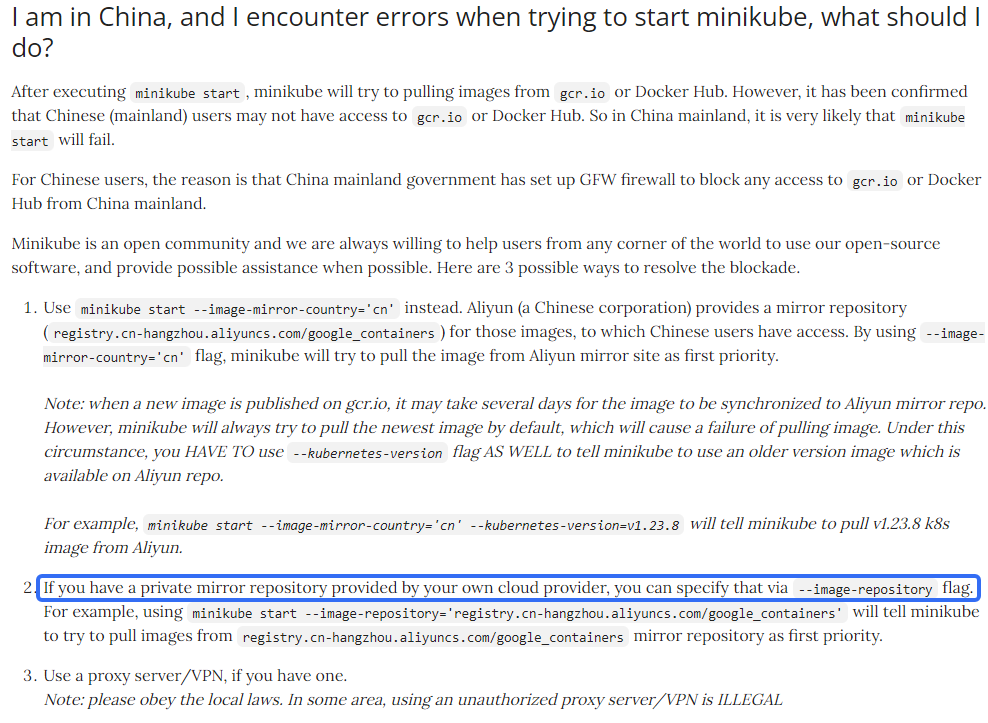


最后，如果没有可用的网络代理，且minikube下载启动集群所依赖资源持续失败，那么你可以尝试使用minikub FQA文档中[I am in China, and I encounter errors when trying to start minikube, what should I do?](https://minikube.sigs.k8s.io/docs/faq/) 小结介绍的方法命令

minikube start --image-mirror-country='cn'

或命令

minikube start --image-repository='registry.cn-hangzhou.aliyuncs.com/google\_containers'



## 使用kubectl与k8s集群交互

如果你的Windows本地安装过kubectl工具，那么可以直接使用kubectl与之前搭建的集群交互，例如，如下显示kubectl与集群版本号命令

kubectl version  
WARNING: This version information is deprecated and will be replaced with the output from kubectl version --short. Use --output=yaml|json to get the full version.  
Client Version: version.Info{Major:"1", Minor:"27", GitVersion:"v1.27.2", GitCommit:"7f6f68fdabc4df88cfea2dcf9a19b2b830f1e647", GitTreeState:"clean", BuildDate:"2023-05-17T14:20:07Z", GoVersion:"go1.20.4", Compiler:"gc", Platform:"windows/amd64"}  
Kustomize Version: v5.0.1  
Server Version: version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.3", GitCommit:"9e644106593f3f4aa98f8a84b23db5fa378900bd", GitTreeState:"clean", BuildDate:"2023-03-15T13:33:12Z", GoVersion:"go1.19.7", Compiler:"gc", Platform:"linux/amd64"}

例如，获取集群中所有的Pod信息命令

kubectl get po -A  
NAMESPACE NAME READY STATUS RESTARTS AGE  
kube-system coredns-787d4945fb-xjdrm 1/1 Running 1 (155m ago) 163m  
kube-system etcd-minikube 1/1 Running 1 (155m ago) 163m  
kube-system kube-apiserver-minikube 1/1 Running 2 (155m ago) 163m  
kube-system kube-controller-manager-minikube 1/1 Running 2 (155m ago) 163m  
kube-system kube-proxy-k47s6 1/1 Running 1 (155m ago) 163m  
kube-system kube-scheduler-minikube 1/1 Running 2 (155m ago) 163m  
kube-system storage-provisioner 1/1 Running 2 (155m ago) 163m

如果你本地未安装kubectl，还可以直接使用minikube提供的kubectl（minikube can download the appropriate version of kubectl and you should be able to use it like this），使用命令如下：

minikube kubectl -- get po -A

为了方便使用，你可以在bash中设置如下别名，简化需要输入的命令长度

alias kubectl="minikube kubectl --"

现在测试集群已经搭建完成，等待所有的pod初始化完成，即可。

## 安装Kubernetes Dashboard组件

Minikube默认不安装Kubernetes Dashboard组件，可以自行使用如下命令进行安装：

Initially, some services such as the storage-provisioner, may not yet be in a Running state. This is a normal condition during cluster bring-up, and will resolve itself momentarily. For additional insight into your cluster state, minikube bundles the Kubernetes Dashboard, allowing you to get easily acclimated to your new environment:

minikube dashboard  
\* Enabling dashboard ...  
- Using image docker.io/kubernetesui/dashboard:v2.7.0  
- Using image docker.io/kubernetesui/metrics-scraper:v1.0.8  
\* Some dashboard features require the metrics-server addon. To enable all features please run:  
  
minikube addons enable metrics-server  
  
  
\* Verifying dashboard health ...  
\* Launching proxy ...  
\* Verifying proxy health ...  
\* Opening http://127.0.0.1:13271/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...

如果不出意外的话，该命令会阻塞当前bash会话并打开浏览器展示kubernetes-dashboard网页。从上述日志可以看出，因为未启用metrics-server组件，部分特性功能无法使用。metric-server是集群核心监控数据的聚合器，主要用于收集群中各节点的监控数据、容器CPU和内存使用率等等，并且提供了Metrics API以供分析和使用。这些度量可以由用户直接访问（例如，通过使用kubectl top命令），或者由集群中的控制器（例如，Horizontal Pod Autoscaler）使用来进行决策。

AWS 对Metrics Server的介绍页面如下：



需要注意的是：

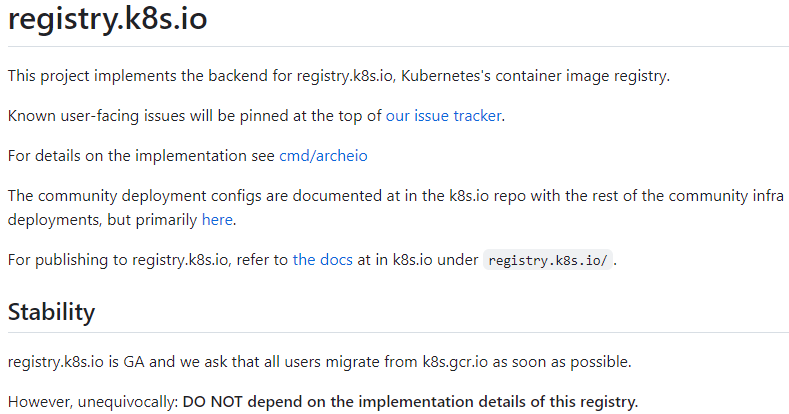
* metric-server提供的是实时的指标（实际是最近一次采集的数据，保存在内存中），并没有数据库来存储
* 这些数据指标并非由metric-server本身采集，而是由每个节点上的cadvisor采集，metric-server只是发请求给cadvisor并将metric格式的数据转换成aggregate api

## minikube配置启用metrics-server插件

minikube工具管理的测试集群启用metrics-server插件非常简单，执行如下命令即可：

minikube addons enable metrics-server  
\* metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.  
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS  
- Using image registry.k8s.io/metrics-server/metrics-server:v0.6.3  
\* The 'metrics-server' addon is enabled

不过，因为我的网络不能直接访问registry.k8s.io域名，导致Pod迟迟无法正常启动。百度了很久也未能找到registry.k8s.io可用的镜像站，无法解决这个网络不能访问的问题。其中使用的registry.k8s.io是k8s官方镜像仓库的新地址，用于替换旧的k8s.gcr.io，具体可以参考[Github registry.k8s.io](https://github.com/kubernetes/registry.k8s.io)仓库如下描述



好在，minikube提供了指定插件替换镜像的命令行参数。首先执行如下命令，查看metrics-server插件依赖的所有镜像及版本

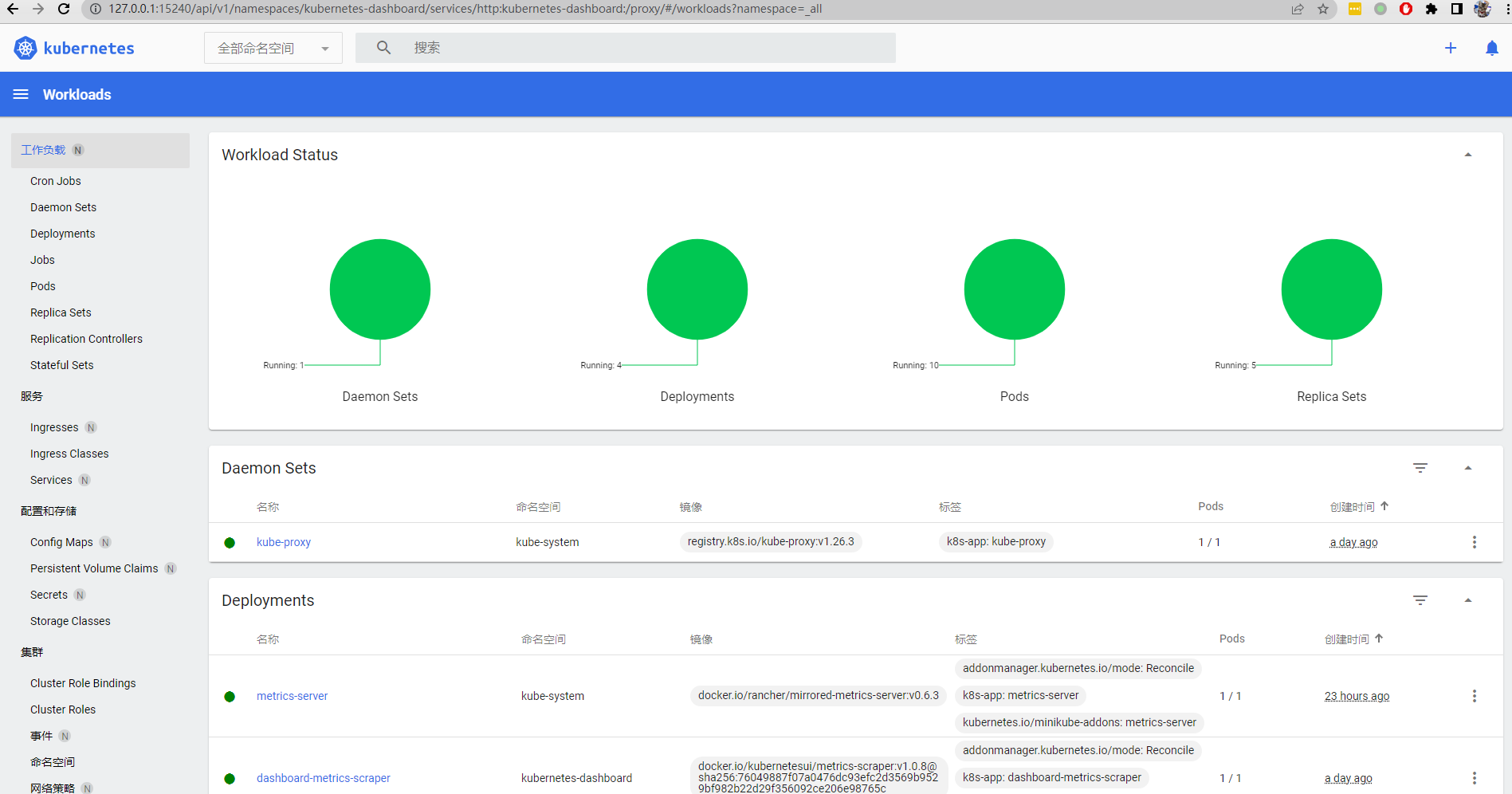
minikube addons images metrics-server  
- metrics-server has following images:  
|---------------|--------------------------------------------------------------------------------------------------------------|------------------|  
| IMAGE NAME | DEFAULT IMAGE | DEFAULT REGISTRY |  
|---------------|--------------------------------------------------------------------------------------------------------------|------------------|  
| MetricsServer | metrics-server/metrics-server:v0.6.3@sha256:c60778fa1c44d0c5a0c4530ebe83f9243ee6fc02f4c3dc59226c201931350b10 | registry.k8s.io |  
|---------------|--------------------------------------------------------------------------------------------------------------|------------------|

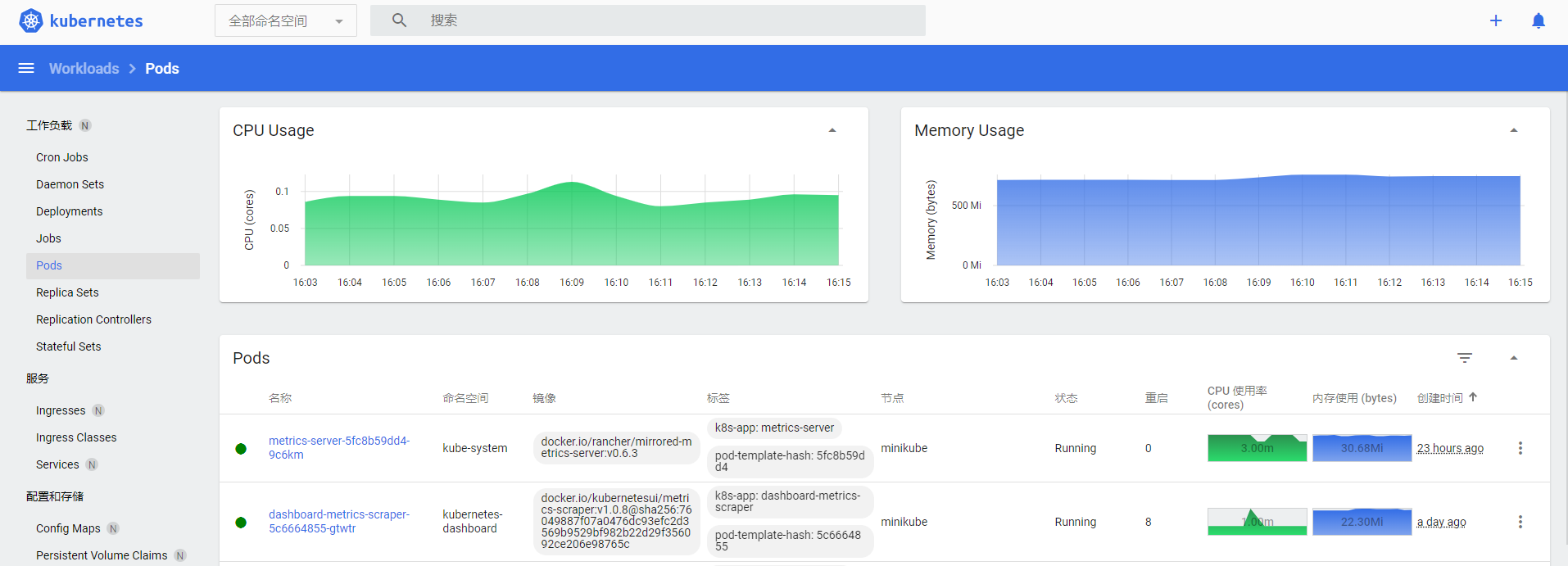
然后，在docker hub上搜索对应的其他人上传的镜像仓库，找到一个对应的[rancher/mirrored-metrics-server](https://hub.docker.com/r/rancher/mirrored-metrics-server/tags)，最后使用如下命令直接改用docker hub中的rancher/mirrored-metrics-server启用metrics-server插件。

minikube addons enable metrics-server --images='MetricsServer=rancher/mirrored-metrics-server:v0.6.3' --registries='MetricsServer=docker.io'  
\* metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.  
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS  
- Using image docker.io/rancher/mirrored-metrics-server:v0.6.3  
\* The 'metrics-server' addon is enabled

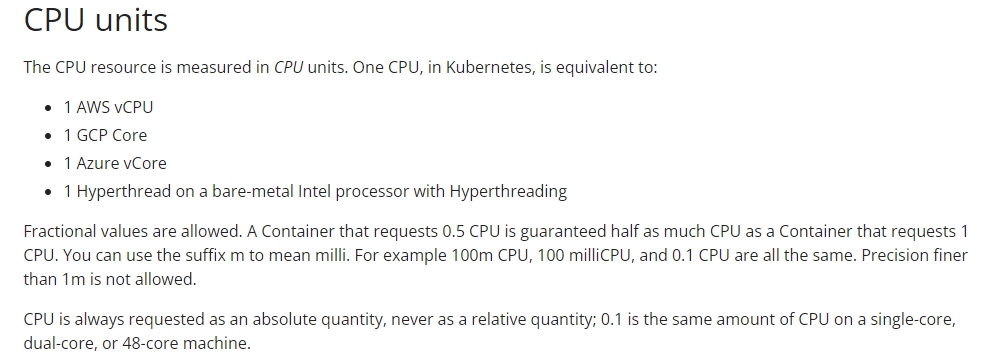
更过细节可以参考minikube工具的[Config the Addon to Use Custom Registries and Images](https://minikube.sigs.k8s.io/docs/handbook/addons/custom-images/)文档。

metrics-server插件正常运行几分钟后，就可以在Kubernetes Dashboard中查看Pod、Node等资源占用情况，显示效果如下：





其监控页面中CPU使用率单位可以参考k8s [Assign CPU Resources to Containers and Pods](https://kubernetes.io/docs/tasks/configure-pod-container/assign-cpu-resource/#cpu-units)文档如下小结



## minikube节点机器

使用如下命令可以列出minikube虚拟的所有节点

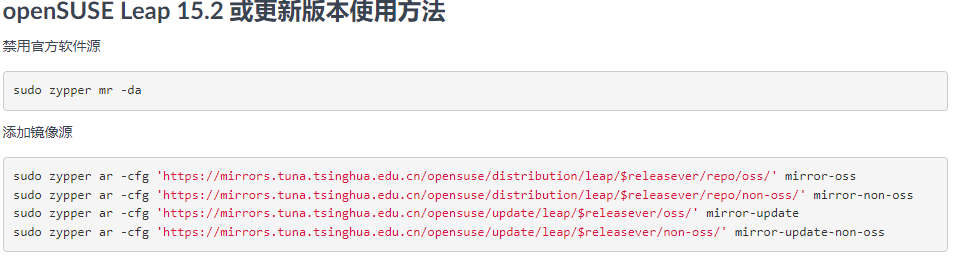
kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
minikube Ready control-plane 27h v1.26.3

透过ssh连接指定节点命令如下

minikube ssh -n minikube

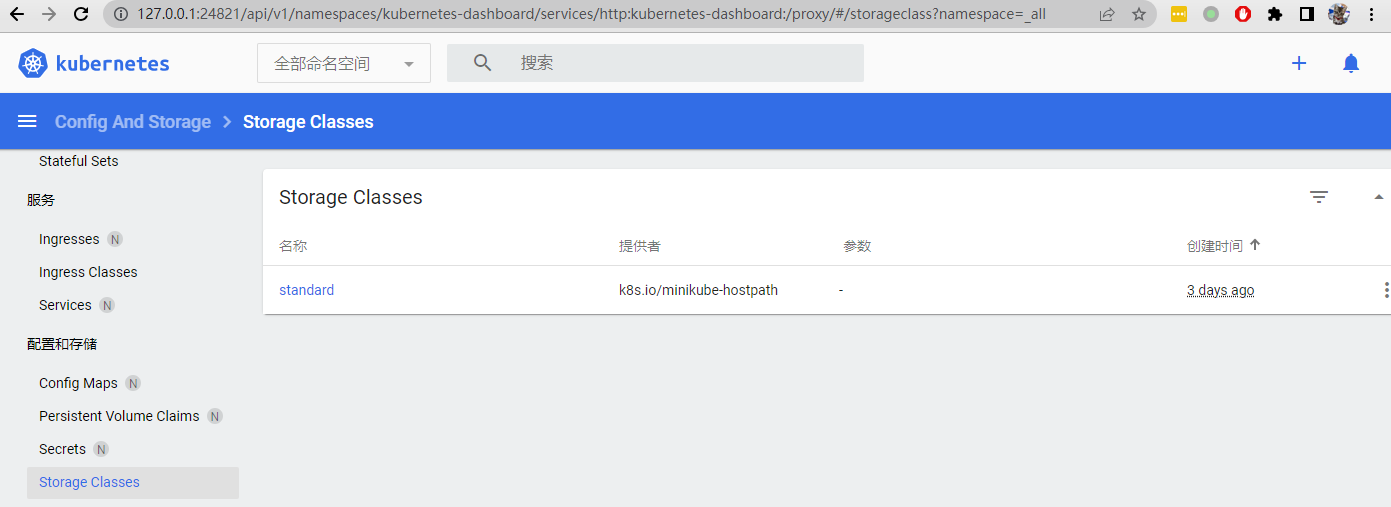
更过命令请参考minikube入门文档[minikube start](https://minikube.sigs.k8s.io/docs/start/)

如果使用Rancher提供的镜像直接使用docker-desktop部署一个k3s集群+rancher的话，那么就没有上述这么方便访问node节点集群的方法。需要直接使用docker-desktop打开container的终端即可。不过rancher是基于openSUSE的镜像，其使用zypper来管理包，用的很少，操作不是很顺畅。其可用清华源可以参考文档[openSUSE 软件仓库镜像使用帮助](https://mirrors.tuna.tsinghua.edu.cn/help/opensuse/)来配置使用。

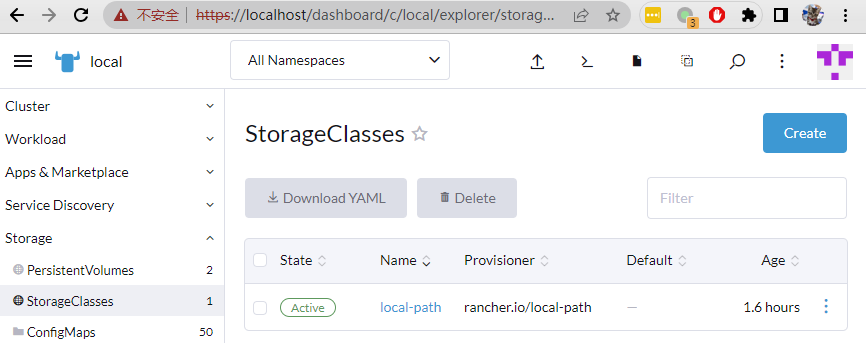


## K8s集群Storage Classes问题

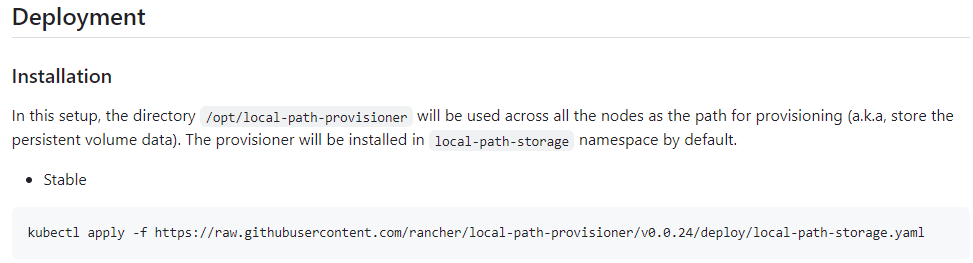
minikube创建的虚拟机集群默认配置了名为“standard“的Storage Classes，如下图所示：

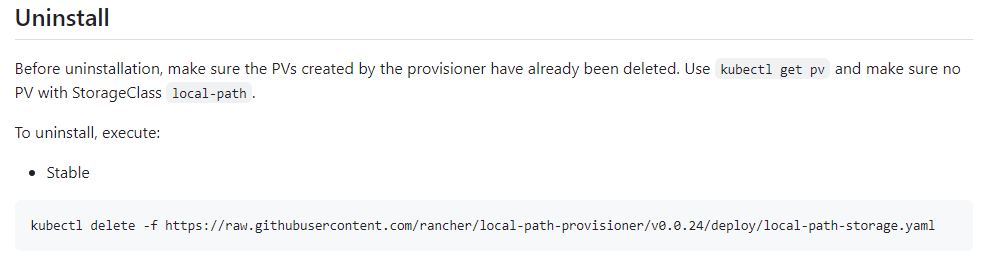


这样无需安装其它组件即可响应pvc中声明。而如果使用Rancher提供的镜像直接使用docker-desktop部署一个k3s集群+rancher的话，则默认没有安装配置任何Storage Classes。因为rancher仅虚拟一个node节点机器，所以可以直接安装使用[Local Path Provisioner](https://github.com/rancher/local-path-provisioner)，该镜像可以自动为rancher创建的虚拟机集群配置一个名为“local-path“的Storage Classes。



[local-path-provisioner](https://github.com/rancher/local-path-provisioner)未找到好用的helm chart，按<https://github.com/rancher/local-path-provisioner>仓库中的命令使用kubectl安装即可。

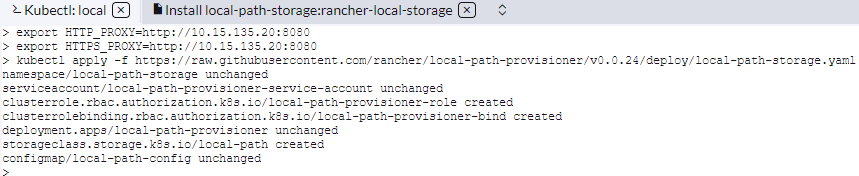




Rancer容器化临时部署的[local-path-provisioner](https://github.com/rancher/local-path-provisioner)有时候会在容器重启后，部署持续crash，需要再重新部署一下来解决部分资源丢失的问题，从而解决[local-path-provisioner](https://github.com/rancher/local-path-provisioner)部署crash的问题。

$ kubectl apply -f https://raw.githubusercontent.com/rancher/local-path-provisioner/v0.0.24/deploy/local-path-storage.yaml  
namespace/local-path-storage created  
serviceaccount/local-path-provisioner-service-account created  
clusterrole.rbac.authorization.k8s.io/local-path-provisioner-role created  
clusterrolebinding.rbac.authorization.k8s.io/local-path-provisioner-bind created  
deployment.apps/local-path-provisioner created  
storageclass.storage.k8s.io/local-path created  
configmap/local-path-config created

可以看出是rancher容器重启后部分custerrole都是导致的，重建既可以使服务临时恢复

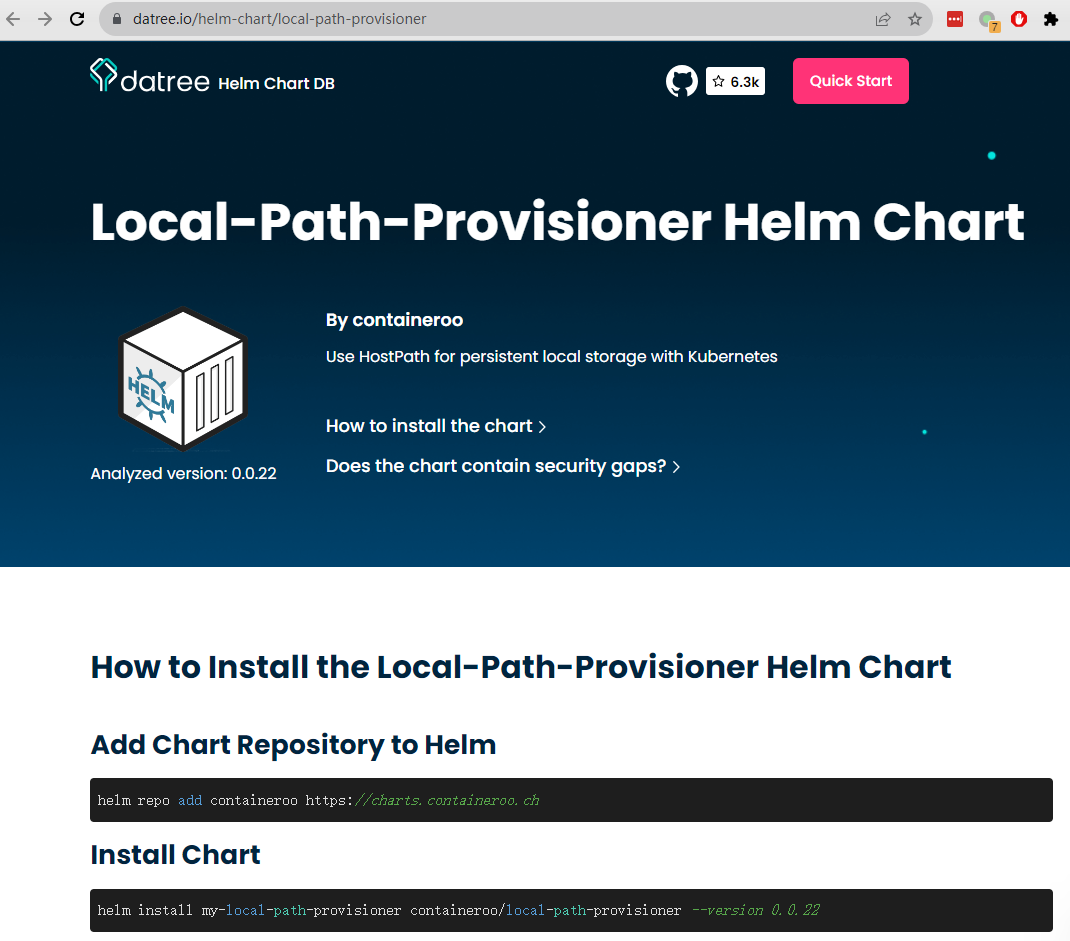


使用上述方案kubectl直接部署的重启后crash问题还挺烦人的，查找一圈，发现issue中也有人碰到类似问题，但是却并未有人给出解决方案。



### 搜索后找到一个local-path-provisioner第三方的helm chart

第三方local-path-provisioner的helm chart截图如下， 看起来还挺靠谱。不管怎样，用helm chart创建的local-path-provisioner资源至少不会出现重启后Pod Crash的现象，问题完美解决 ( ╯□╰ )



### Add Chart Repository to Helm

helm repo add containeroo https:*//charts.containeroo.ch*

