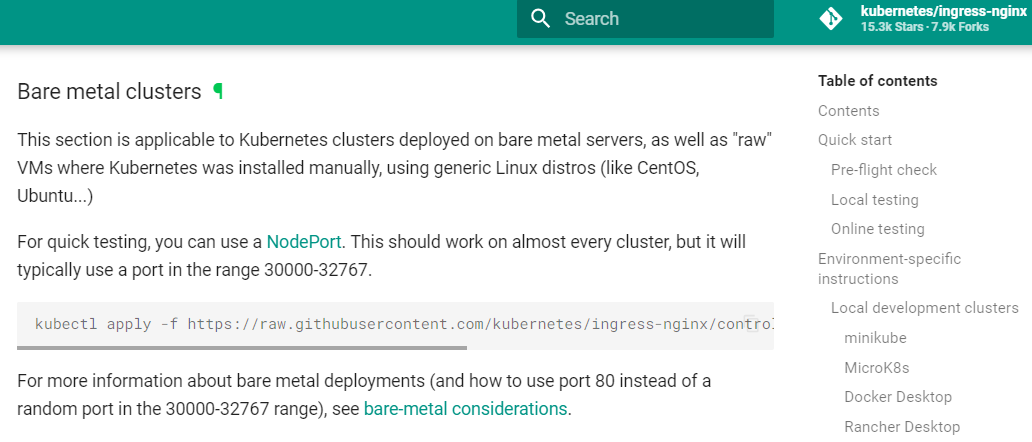
# docker单容器本地部署的rancher中安装ingress controller

2023.07.25

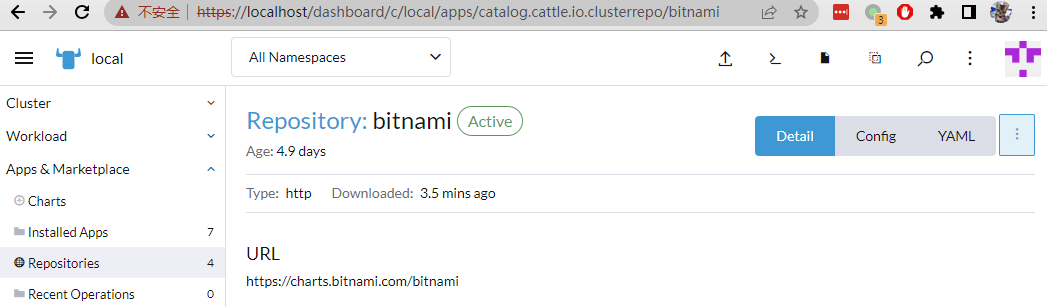
本地使用docker desktop直接运行rancher镜像虚拟机出的k8s默认并未提供ingress controller。

我们可以自行安装[kubernetes/ingress-nginx](https://kubernetes.github.io/ingress-nginx/deploy/#bare-metal-clusters)，本地测试集群安装可以参考文档中Bare metal clusters小结

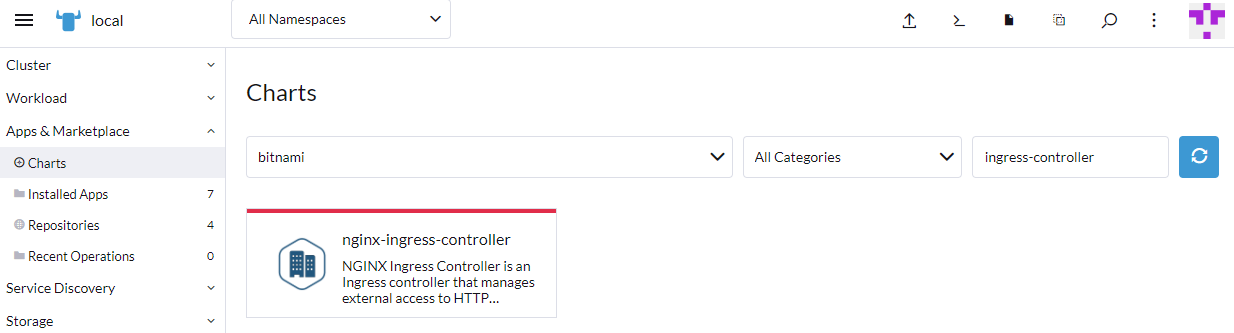


这里我们并不采用上述文档中描述方式安装，而是直接采用使用bitnami提供的helm chart（CHART NAME: nginx-ingress-controller）来完成[nginx-ingress-controller](https://kubernetes.github.io/ingress-nginx/deploy/)的安装。

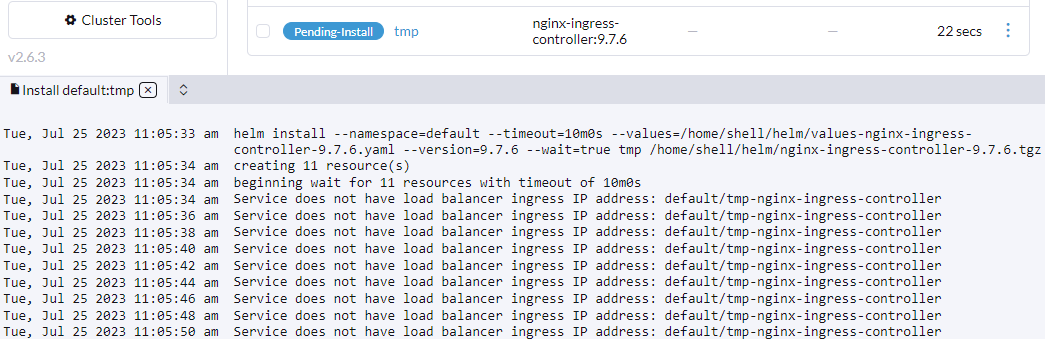
首先，在rancher中添加bitnami helm chart的仓库地址，如下所示：



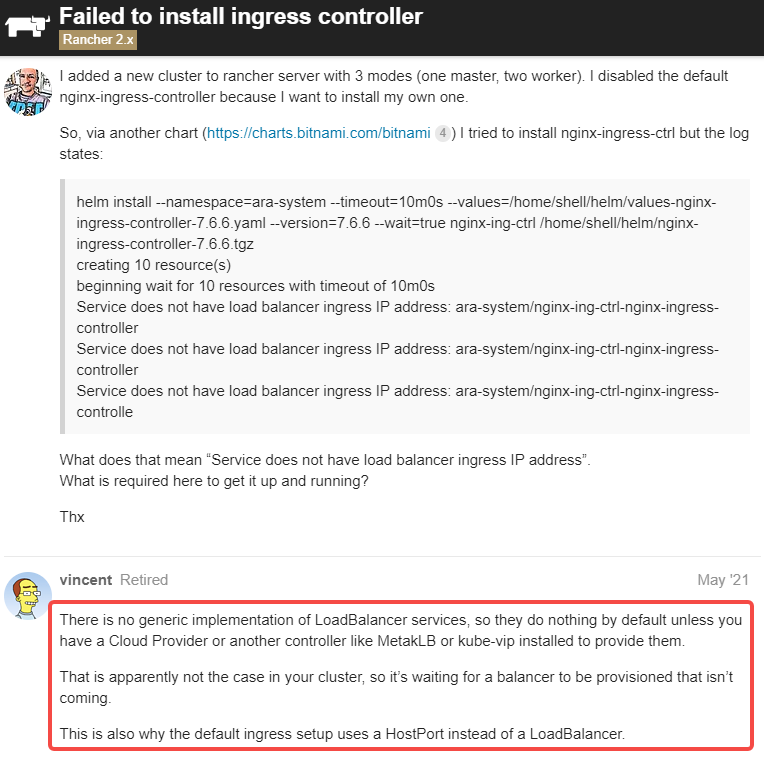
添加完bitnami仓库后，在Apps&Marketplace页面选择nginx-ingress-controller chart，如下所示：



直接使用默认参数安装的话，该helm chart会给nginx-ingress-controller分配LoadBalancer，而我们本地测试集群默认也没提供LoadBalancer服务，导致安装失败，如下图所示：

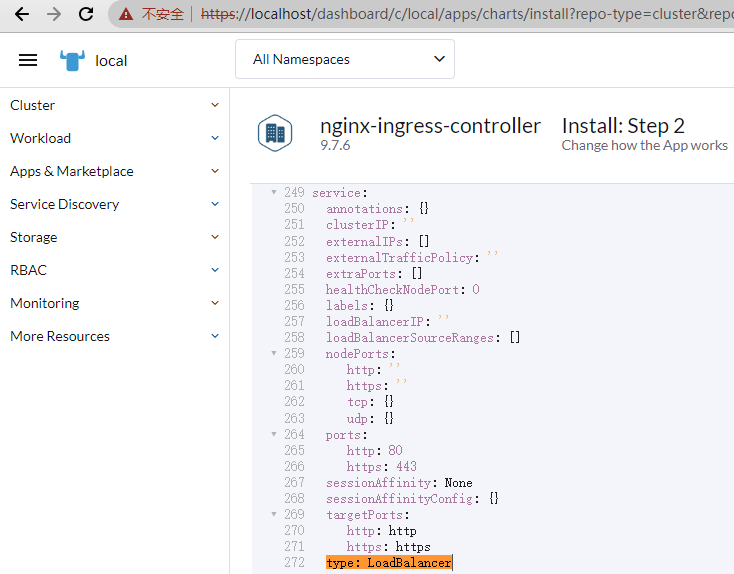


安装失败的原因上边也解释了，解决方案可以参考如下文档：



这里我们选择简单的使用NodePort的方式部署，而不是再安装一个loadbalaner服务。

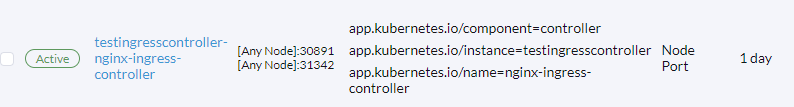
使用NodePort方式部署nginx-ingress-controller仅需修改安装Helm chart是如下YAML的service.type参数：



安装成功后，可以看到如下信息

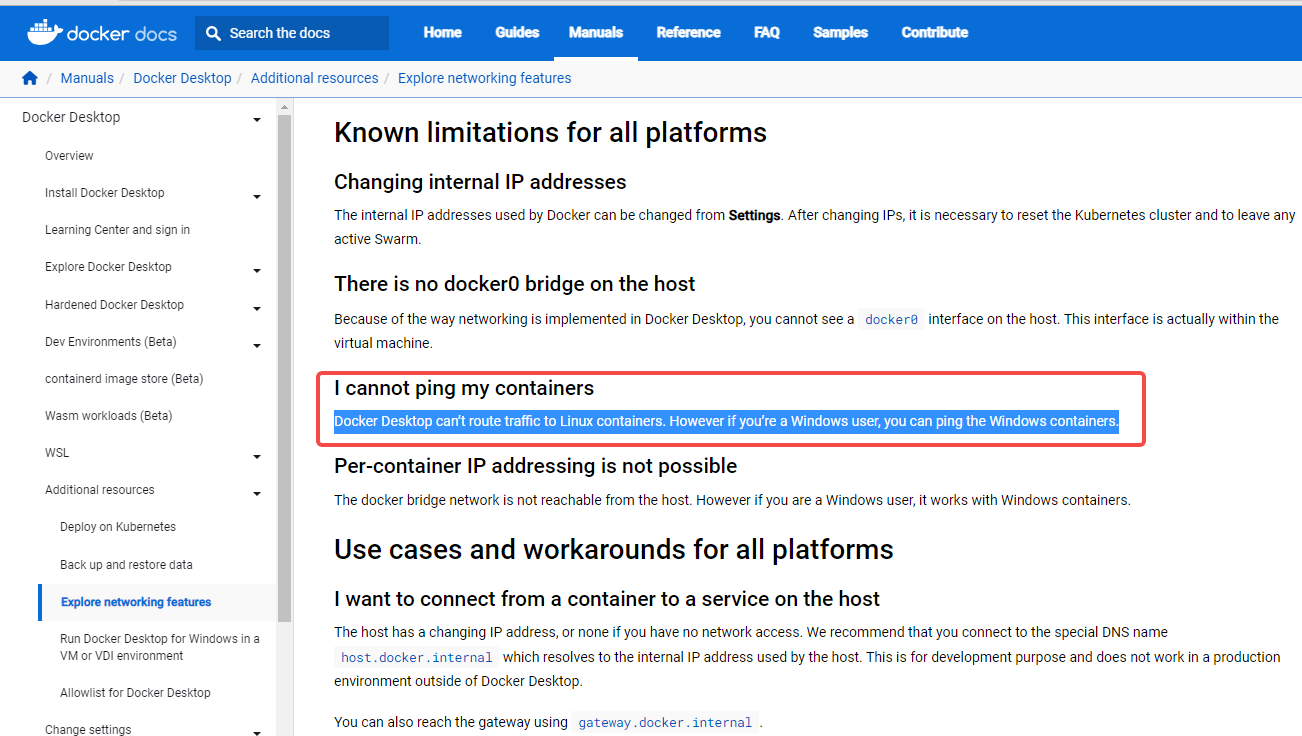
helm install --namespace=default --timeout=10m0s --values=/home/shell/helm/values-nginx-ingress-controller-9.7.6.yaml --version=9.7.6 --wait=true testingresscontroller /home/shell/helm/nginx-ingress-controller-9.7.6.tgz  
creating 11 resource(s)  
beginning wait for 11 resources with timeout of 10m0s  
Deployment is not ready: default/testingresscontroller-nginx-ingress-controller. 0 out of 1 expected pods are ready  
Deployment is not ready: default/testingresscontroller-nginx-ingress-controller. 0 out of 1 expected pods are ready  
NAME: testingresscontroller  
LAST DEPLOYED: Mon Jul 24 03:29:21 2023  
NAMESPACE: default  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None  
NOTES:  
CHART NAME: nginx-ingress-controller  
CHART VERSION: 9.7.6  
APP VERSION: 1.8.1  
\*\* Please be patient while the chart is being deployed \*\*  
The nginx-ingress controller has been installed.  
Get the application URL by running these commands:  
export HTTP\_NODE\_PORT=$(kubectl --namespace default get services -o jsonpath="{.spec.ports[0].nodePort}" testingresscontroller-nginx-ingress-controller)  
export HTTPS\_NODE\_PORT=$(kubectl --namespace default get services -o jsonpath="{.spec.ports[1].nodePort}" testingresscontroller-nginx-ingress-controller)  
export NODE\_IP=$(kubectl --namespace default get nodes -o jsonpath="{.items[0].status.addresses[1].address}")  
echo "Visit http://$NODE\_IP:$HTTP\_NODE\_PORT to access your application via HTTP."  
echo "Visit https://$NODE\_IP:$HTTPS\_NODE\_PORT to access your application via HTTPS."  
An example Ingress that makes use of the controller:  
apiVersion: networking.k8s.io/v1  
kind: Ingress  
metadata:  
name: example  
namespace: default  
spec:  
ingressClassName: nginx  
rules:  
- host: www.example.com  
http:  
paths:  
- backend:  
service:  
name: example-service  
port:  
number: 80  
path: /  
pathType: Prefix  
# This section is only required if TLS is to be enabled for the Ingress  
tls:  
- hosts:  
- www.example.com  
secretName: example-tls  
If TLS is enabled for the Ingress, a Secret containing the certificate and key must also be provided:  
apiVersion: v1  
kind: Secret  
metadata:  
name: example-tls  
namespace: default  
data:  
tls.crt: <base64 encoded cert>  
tls.key: <base64 encoded key>  
type: kubernetes.io/tls  
---------------------------------------------------------------------  
SUCCESS: helm install --namespace=default --timeout=10m0s --values=/home/shell/helm/values-nginx-ingress-controller-9.7.6.yaml --version=9.7.6 --wait=true testingresscontroller /home/shell/helm/nginx-ingress-controller-9.7.6.tgz  
---------------------------------------------------------------------

在server中可以找到对应的nodeport信息，如下图所示：



因为在Windows平台无法直接从windows访问docker container的IP，原因docker文档[Explore networking features](https://docs.docker.com/desktop/networking/)

中如下图小结所示：



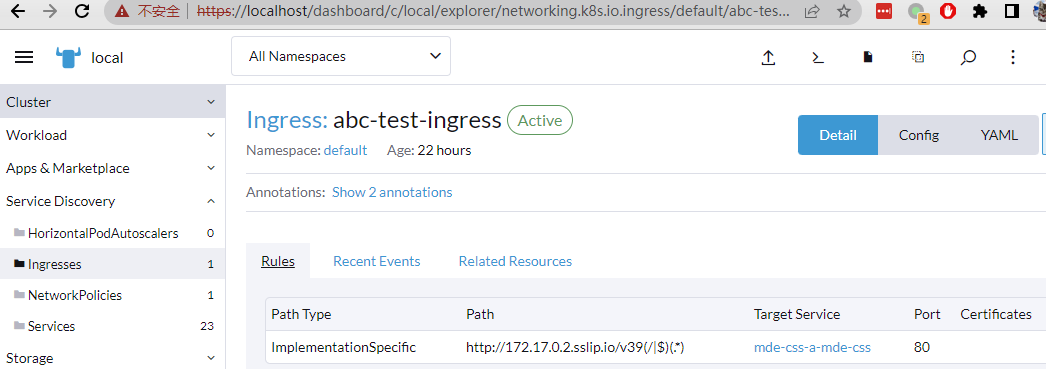
为了测试ingress controller，我们在docker desktop中rancher容器所在的同一bridge虚拟网络中，创建一个firefox容器，用于直接访问集群通过ingress暴漏出来的服务。Firefox容器启动命令如下：

docker run -d --name=firefox -p 5800:5800 -v d:\\tmp\firefox:/config:rw --shm-size=512m --hostname firefox wellsen/firefox

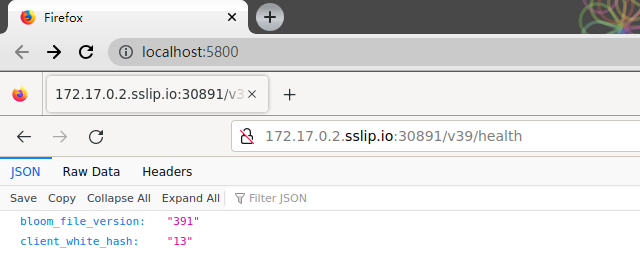
在ingress页面配置如下测试ingress，配置主要Yaml信息如下：

apiVersion: networking.k8s.io/v1  
kind: Ingress  
metadata:  
 annotations:  
 kubernetes.io/ingress.class: nginx  
 nginx.ingress.kubernetes.io/rewrite-target: /$2  
 name: abc-test-ingress  
 namespace: default  
spec:  
 rules:  
 - host: 172.17.0.2.sslip.io  
 http:  
 paths:  
 - backend:  
 service:  
 name: mde-css-a-mde-css  
 port:  
 number: 80  
 path: /v39(/|$)(.\*)  
 pathType: ImplementationSpecific

配置完成后，显示效果如下：



打开我们启动的firefox容器中的浏览器，输入对应URL，如下图所示：



其中，URL <http://172.17.0.2.sslip.io:30891/v39/health>的hostname为一种特殊的NDS服务，可以将IP转为合法的FQN并且无需用户真实注册。

如果不想安装firefox容器，使用service的端口转发命令，将service的端口转发到windows本地也可以，命令如下：

kubectl --kubeconfig .\local.yaml port-forward --namespace default svc/testingresscontroller-nginx-ingress-controller 9999:80

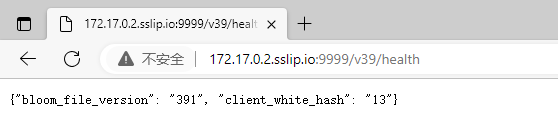
然后使用Curl命令测试服务，如下所示：

curl -H "HOST: 172.17.0.2.sslip.io" -v 127.0.0.1:9999/v39/health



当然，如果你非要在Windows本地的浏览器中访问，那么只能修改Windows的hosts文件了，如下所示在C:\Windows\System32\drivers\etc\hosts文件中添加如下一行记录

127.0.0.1 172.17.0.2.sslip.io



## 启用ingress controller的metrics

[kubernetes/ingress-nginx](https://kubernetes.github.io/ingress-nginx/deploy/#bare-metal-clusters)默认监控metrics配置并未启用，需要显示的enable相关配置。主要样例配置值如下：

metrics:  
 enabled: true  
 serviceMonitor:  
 enabled: true  
 namespace: ctl-kube-prometheus

若要启用metrics，必须保证集群中prometheus-operator服务正常，否则相关指标监控会出现问题。一切正常的话应存在一个名称包含metrics的services，prometheus通过servicemonitor来自动抓取相关指标。

