

## 第 12 章 通过 Rancher 管理 k8s 集群

本节所讲内容:

- 12.1 Rancher 简介
- 12.2 安装 rancher
- 12.3 通过 Rancher 管理已存在的 k8s 集群
- 12.4 使用 Rancher 中自带的监控功能查看 k8s 集群运行状态
- 12.5 通过 Rancher 仪表盘管理 k8s: 部署 web 站点
- 12.6 实战-分布式 LNMP 环境部署电商网站
- 12.7 实战-ingress-对外发布服务

实验环境: 需要有一套已经存在的 k8s 集群, 用我们之前实验用到的 k8s 即可, 确定 k8s 集群正常:

```
[root@xuegod63 ~]# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
xuegod63	Ready	control-plane,master	10d	v1.20.4
xuegod64	Ready	<none>	10d	v1.20.4

**k8s 的控制节点**

ip: 192.168.1.63

主机名: xuegod63

**k8s 的工作节点:**

ip: 192.168.1.64

主机名: xuegod64

### 12.1 Rancher 简介

#### 12.1.1 Rancher 简介

Rancher 是一套容器管理平台, 它可以帮助组织在生产环境中轻松快捷的部署和管理容器。  
Rancher 可以轻松地管理各种环境的 Kubernetes, 满足 IT 需求并为 DevOps 团队提供支持。

Kubernetes 不仅已经成为容器编排标准, 它也正在迅速成为各类云和虚拟化厂商提供的标准基础架构。Rancher 用户可以选择使用 Rancher Kubernetes Engine(RKE)创建 Kubernetes 集群, 也可以使用 GKE, AKS 和 EKS 等云 Kubernetes 服务。Rancher 用户还可以导入和管理现有的 Kubernetes 集群。

官网: <https://rancher.com/> ; <https://www.rancher.cn/>

**Rancher** ['rɑ:ntʃə(r)] 牧场主

扩展: GKE: Google Kubernetes Engine, Google 的 k8s 托管服务

AKS: Azure Kubernetes 服务 (AKS), 微软的 k8s 托管服务

EKS: Amazon Elastic Container Service for Kubernetes , Amazon (亚马逊) 的 K8S 托管服务

Rancher 为 DevOps 工程师提供了一个直观的用户界面来管理他们的服务容器, 用户不需要深入了解 Kubernetes 概念就可以开始使用 Rancher。 Rancher 包含应用商店, 支持一键式部署 Compose 模板。

扩展: docker-compose 是 Docker 容器进行编排的工具, 定义和运行多容器的应用, 可以一条命令启动多个容器, 使用 docker-compose 不再需要使用 shell 脚本来启动容器。

Compose [kəm'pəʊz] 组成

docker-compose 默认的模板文件是 docker-compose.yml, 其中定义的每个服务都必须通过 image 指令指定镜像或 build 指令 (需要 Dockerfile) 来自动构建。

### 12.1.2 Rancher 四个组成部分

#### 1、基础设施编排

Rancher 可以使用任何公有云或者私有云的 Linux 主机资源。Linux 主机可以是虚拟机, 也可以是物理机。

#### 2、容器编排与调度

很多用户都会选择使用容器编排调度框架来运行容器化应用。Rancher 包含了当前全部主流的编排调度引擎, 例如 Docker Swarm, Kubernetes, 和 Mesos。同一个用户可以创建 Swarm 或者 Kubernetes 集群。并且可以使用原生的 Swarm 或者 Kubernetes 工具管理应用。

除了 Swarm, Kubernetes 和 Mesos 之外, Rancher 还支持自己的 Cattle 容器编排调度引擎。Cattle 被广泛用于编排 Rancher 自己的基础设施服务以及用于 Swarm 集群, Kubernetes 集群和 Mesos 集群的配置, 管理与升级。

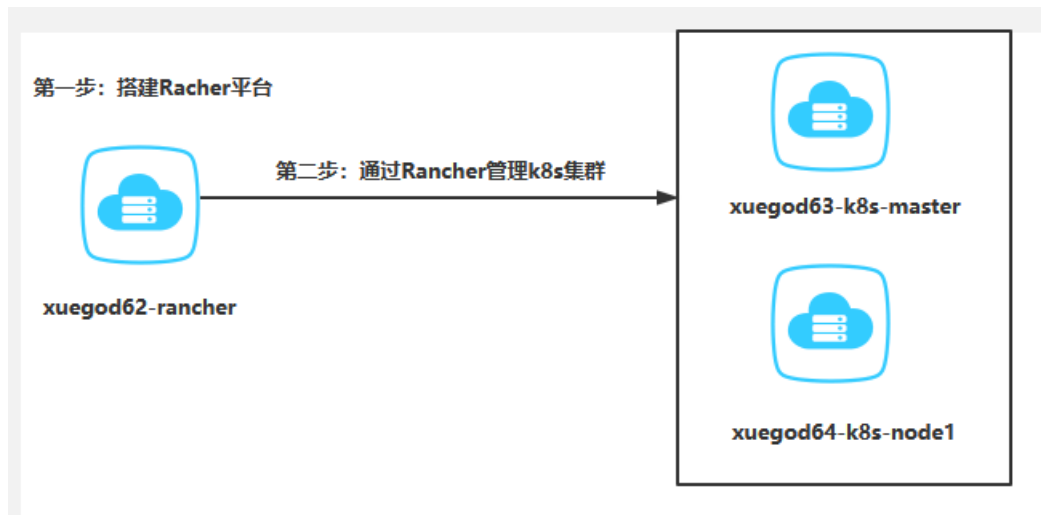
#### 3、应用商店

Rancher 的用户可以在应用商店里**一键部署由多个容器组成的应用**。用户可以管理这个部署的应用, 并且可以在这个应用有新的可用版本时进行自动化的升级。Rancher 提供了一个由 Rancher 社区维护的应用商店, 其中包括了一系列的流行应用。Rancher 的用户也可以创建自己的私有应用商店。

#### 4、企业级权限管理

Rancher 支持灵活的插件式的用户认证。支持 Active Directory, LDAP, Github 等 认证方式。

### 1.2.3 本章拓扑



## 12.2 安装 rancher

在 xuegod62 上操作

### 12.2.1 初始化实验环境

新创建一台虚拟机

**环境说明 (centos7.6):**

IP	主机名	内存	cpu
192.168.1.138	xuegod62	6G	4vCPU

**配置主机名:**

在 192.168.1.138 上执行如下:

```
hostnamectl set-hostname xuegod62
```

**#配置 hosts 文件:**

xuegod62、xuegod63、xuegod64 三个主机 hosts 文件要保持一致:

打开/etc/hosts 文件, 新增加如下几行:

```
192.168.1.63 xuegod63
192.168.1.64 xuegod64
192.168.1.138 xuegod62
```

**配置 rancher 到 k8s 主机互信**

生成 ssh 密钥对

```
[root@xuegod62 ~]# ssh-keygen #一路回车, 不输入密码
```

#把本地的 ssh 公钥文件安装到远程主机对应的账户

```
[root@xuegod62 ~]# ssh-copy-id xuegod63
```

```
[root@xuegod62 ~]# ssh-copy-id xuegod62
```

```
[root@xuegod62 ~]# ssh-copy-id xuegod64
```

**关闭防火墙**

```
[root@xuegod62 ~]# systemctl stop firewalld ; systemctl disable firewalld
```

**关闭 selinux**

```
[root@xuegod62 ~]# setenforce 0
[root@xuegod62 ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/selinux/config
```

注意: 修改 selinux 配置文件之后, 重启机器, selinux 才能永久生效

### 关闭 swap 分区。

#临时关闭

```
[root@xuegod62 ~]# swapoff -a
[root@xuegod62 ~]# free -m #可以看到 swap 分区的大小, 已经变为 0
```

	total	used	free	shared	buff/cache	available
Mem:	4876	501	516	20	3858	4068
Swap:	0	0	0			

### 永久关闭

```
[root@xuegod62 ~]# vim /etc/fstab
```

#	dev/mapper/centos-swap	swap	swap	defaults	0	0
---	------------------------	------	------	----------	---	---

```
#注释掉 swap 这行
```

注: 如果是克隆主机请删除网卡中的 UUID 并重启网络服务。

内核参数修改: br\_netfilter 模块用于将桥接流量转发至 iptables 链, br\_netfilter 内核参数需要开启转发。

```
[root@xuegod62 ~]# modprobe br_netfilter
[root@xuegod62 ~]# echo "modprobe br_netfilter" >> /etc/profile
[root@xuegod62 ~]# cat > /etc/sysctl.d/k8s.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
[root@xuegod62 ~]# sysctl -p /etc/sysctl.d/k8s.conf
```

### 在 xuegod62 上配置 docker 镜像源:

选择一种安装 docker 的方式: 1. 在线安装 或 2. 离线安装

#### 1. 在线安装

安装 docker-ce 环境依赖

```
[root@xuegod63 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
```

配置 docker-ce 国内 yum 源 (阿里云)

```
[root@xuegod63 ~]# yum-config-manager --add-repo
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

#### 2. 配置 docker-ce 的离线 yum 源:

下面需要的 k8s-docker.tar.gz 压缩包在课件里

```
[root@xuegod63 ~]# tar xf k8s-docker.tar.gz -C /opt/
[root@xuegod63 ~]# tee /etc/yum.repos.d/k8s-docker.repo << 'EOF'
```

```
[k8s-docker]
name=k8s-docker
baseurl=file:///opt/k8s-docker
enable=1
pgpcheck=0
EOF
```

在 xuegod62 上安装 docker-ce。我们已经配置了 docker 本地源，直接安装 docker-ce 服务。

```
[root@xuegod62 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
wget net-tools nfs-utils lrzsz gcc gcc-c++ make cmake libxml2-devel openssl-devel curl
curl-devel unzip sudo ntp libaio-devel wget vim ncurses-devel autoconf automake zlib-
devel python-devel epel-release openssh-server socat ipvsadm conntrack ntpdate
telnet
```

安装 docker-ce

```
[root@xuegod62 ~]# yum install docker-ce docker-ce-cli containerd.io -y
[root@xuegod62 ~]# systemctl start docker && systemctl enable docker.service
```

修改 docker 配置文件，配置镜像加速器

```
[root@xuegod62 ~]# tee /etc/docker/daemon.json << 'EOF'
{
"registry-mirrors":["https://rsbud4vc.mirror.aliyuncs.com","https://registry.docker-
cn.com","https://docker.mirrors.ustc.edu.cn","https://dockerhub.azk8s.cn","http://hub
-mirror.c.163.com","http://qtid6917.mirror.aliyuncs.com",
"https://rncxm540.mirror.aliyuncs.com"],
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
[root@xuegod62 ~]# systemctl daemon-reload
[root@xuegod62 ~]# systemctl restart docker
[root@xuegod62 ~]# systemctl status docker
```

显示如下，说明 docker 安装成功了

- docker.service - Docker Application Container Engine  
Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)  
Active: active (**running**) since Wed 2021-03-17 12:39:06 CST

### 12.2.2 安装 Rancher

Rancher2.5.7 支持 k8s1.20.4，所以我们安装 rancher2.5.7 版本，  
# rancher\_257\_v1.tar.gz 压缩包在课件，把课件上传到 xuegod62、xuegod63、xuegod64 上  
在 xuegod62、xuegod63、xuegod64 上操作如下命令，解压镜像：  
[root@xuegod62 ~]# docker load -i rancher\_257\_v1.tar.gz  
[root@xuegod62 ~]# docker run -d --restart=unless-stopped -p 80:80 -p 443:443 --

privileged rancher/rancher:latest

注: unless-stopped, 在容器退出时总是重启容器, 但是不考虑在 Docker 守护进程启动时就已经停止了容器

验证 rancher 是否启动:

```
[root@xuegod62 ~]# docker ps | grep rancher
```

显示如下, 说明启动成功:

```
f9321083e186 rancher/rancher:v2.5.7 "entrypoint.sh" About a minute ago Up
About a minute 0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp loving_johnson
```

### 12.2.3 登录 Rancher 平台

在浏览器访问 xuegod62 的 ip 地址:



选择高级



接受风险并继续

## Welcome to Rancher

Set password for the default `admin` user

- ☒ Set a specific password to use:  
☐ Use a new randomly generated password:

New Password \* 设置一个新的密码

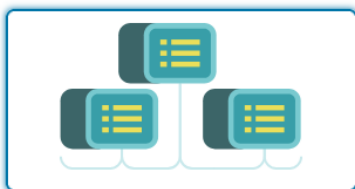
•••••

Confirm Password \*

•••••

确认密码

Set Default View \*



☒ I want to create or manage multiple clusters



☐ I'm only going to use the cluster Rancher was installed on

- ☒ Allow collection of anonymous statistics [Learn More](#)  
☒ I agree to the [Terms and Conditions](#) for using Rancher \*

设置密码之后点击继续即可

Continue



## Rancher Server URL

What URL should be used for this Rancher installation? All the nodes in your clusters will need to be able to reach this.

URL

https:// 192.168.40.138



Are you sure all the hosts you will create will be able to reach 192.168.40.138 ?  
It looks like a private IP or local network.

Save URL

点击 Save URL

### What's New in 2.5

**Cluster Explorer:** New dashboard to provide a deeper look into clusters under management.

- Manage all Kubernetes cluster resources including custom resources from the Kubernetes operator ecosystem
- Deploy and manage Helm charts from our new Apps & Marketplace
- View logs and interact with kubectl shell in a new IDE-like viewer

**Monitoring and Alerting powered by Prometheus:** Allows management of custom Grafana dashboards and provide customization to AlertManager

**Logging powered by Banzai Cloud:** Customize FluentBit and Fluentd configurations and ship logs to a remote data store

**CIS Scans powered by kube-bench:** Extended support to perform CIS scans tailored for EKS and GKE platforms and perform a generic scan on any Kubernetes distribution

**Istio 1.7:** Allows users to deploy multiple ingress and egress gateways

**Rancher Continuous Delivery powered by Fleet:** Fleet is a built-in deployment tool for delivering applications and configurations from a Git source repository across multiple clusters.

- Deploy any Kubernetes resource defined by manifests, kustomize, or Helm
- Scale deployments to any number of clusters using a staged checkout and pull-based update model
- Organize clusters into groups for easier management
- Map Git source repositories to cluster group targets

**Enhanced EKS Lifecycle Management:**

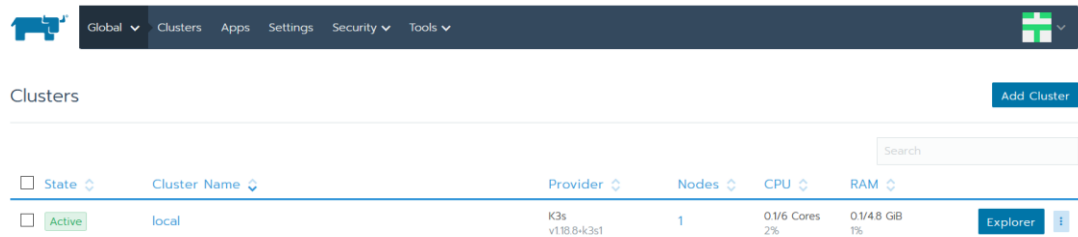
- Provisioning has been enhanced to support managed node groups, private access, and control plane logging
- Registering existing EKS clusters allow management of upgrades and configuration

**Rancher Server Backups:**

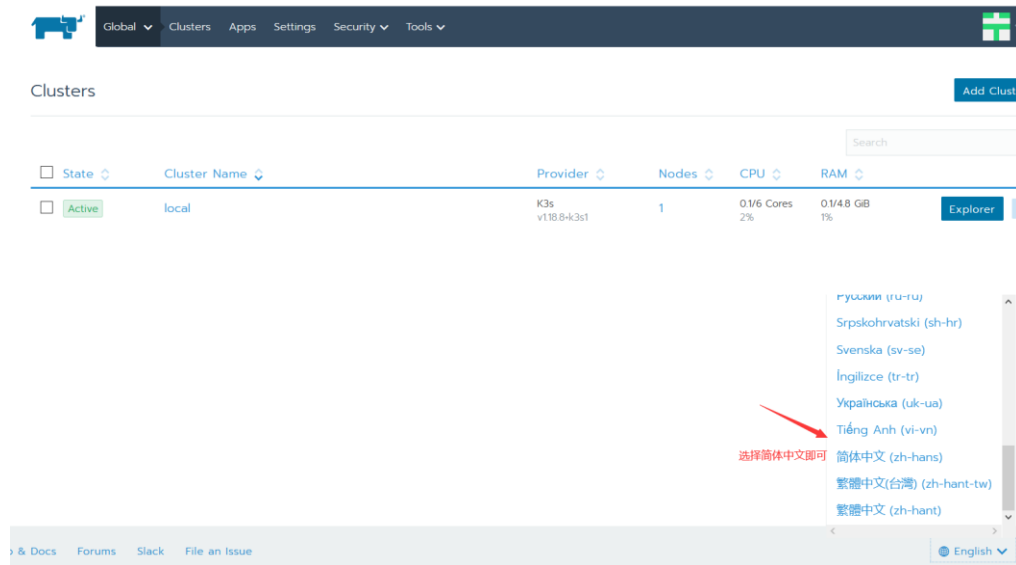
- Back up Rancher server without access to the etcd database
- Restore data into any Kubernetes cluster

Close

点击 Close, 出现如下:



如果大家用英文不方便, 可以把页面字体变成中文, 方法如下:



变成如下中文界面:



## 12.3 通过 Rancher 管理已存在的 k8s 集群

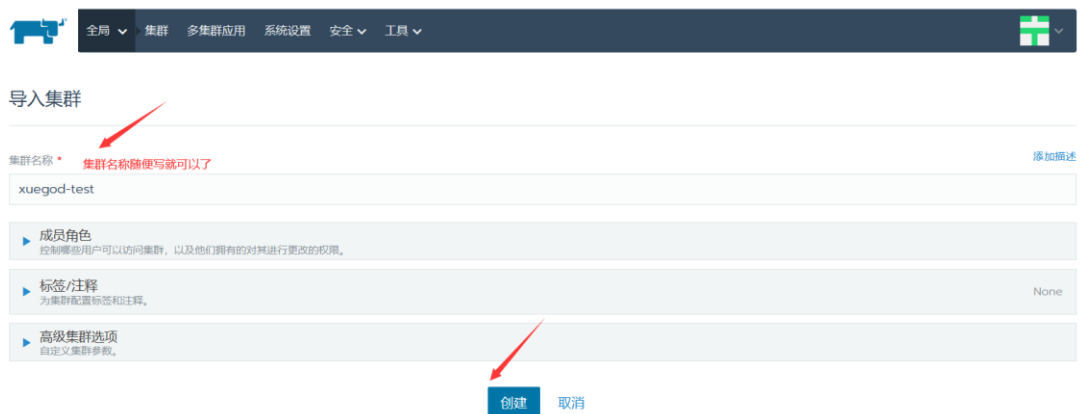
## #把已经存在的 k8s 集群导入到 rancher 了



### 选择添加集群，出现如下：



### 选择导入，出现如下：



### 点击创建之后出现如下提示：



全局 集群 多集群应用 系统设置 安全 工具

导入集群

注意: 如果想要导入 Google Kubernetes Engine(GKE) 集群 (或一些不提供绑定集群管理角色 cluster-admin 的 kubectl 配置文件的集群), 需要通过以下命令来绑定集群管理角色 cluster-admin。用您的谷歌帐户地址替换 [USER\_ACCOUNT] (您可以使用 `gcloud config get-value account` 检索这个地址)。如果您不是导入谷歌 Kubernetes 集群, 那么用 kubectl 配置文件中配置的执行用户替换 [USER\_ACCOUNT]。

```
kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user [USER_ACCOUNT]
```

在现有的受支持的 Kubernetes 集群上运行下面的 kubectl 命令, 将其导入 Rancher:

```
kubectl apply -f https://192.168.40.138/v3/import/tc6wrckn4n7zg4prlqk4jfdjn4qhsss9jn599zh84d2kbjtcxsw9p4_c-t7p7t.yaml
```

如果由于您的 Rancher 安装使用不受信任/自签名的 SSL 证书而出现 由未知权限签名的证书 错误, 请运行下面的命令以绕过证书检查

```
curl --insecure -sL https://192.168.40.138/v3/import/tc6wrckn4n7zg4prlqk4jfdjn4qhsss9jn599zh84d2kbjtcxsw9p4_c-t7p7t.yaml | kubectl apply -f -
```

在 k8s 控制节点执行如下内容

完成

在 k8s 控制节点 xuegod63 上执行上面箭头所指的命令, 如下所示, 执行两次:

`curl --insecure -sL`

`https://192.168.1.138/v3/import/tc6wrckn4n7zg4prlqk4jfdjn4qhsss9jn599zh84d2kbjtcxsw9p4_c-t7p7t.yaml | kubectl apply -f -`

`curl --insecure -sL`

`https://192.168.1.138/v3/import/n72f7fvs5b2ctd99hw8wt6bkm7zw65vskzk9pbvqddknr6cntlnl59_c-27sgj.yaml | kubectl apply -f -`

在回到主页面可以看到 rancher 已经成功导入 k8s1.20.4 版本了



删除	状态	集群名称	供应商	主机数	处理器	内存	仪表盘
<input type="checkbox"/>	Active	local	K3s v1.18.8-k3s1	1	0.1/6 Cores 2%	0.1/4.8 GiB 1%	仪表盘
<input type="checkbox"/>	Active	xuegod-test <span>⚠</span>	Imported v1.20.4	2	0.3/6 Cores 4%	0.5/6 GiB 0%	仪表盘

## 12.4 使用 Rancher 中自带的监控功能查看 k8s 集群运行状态

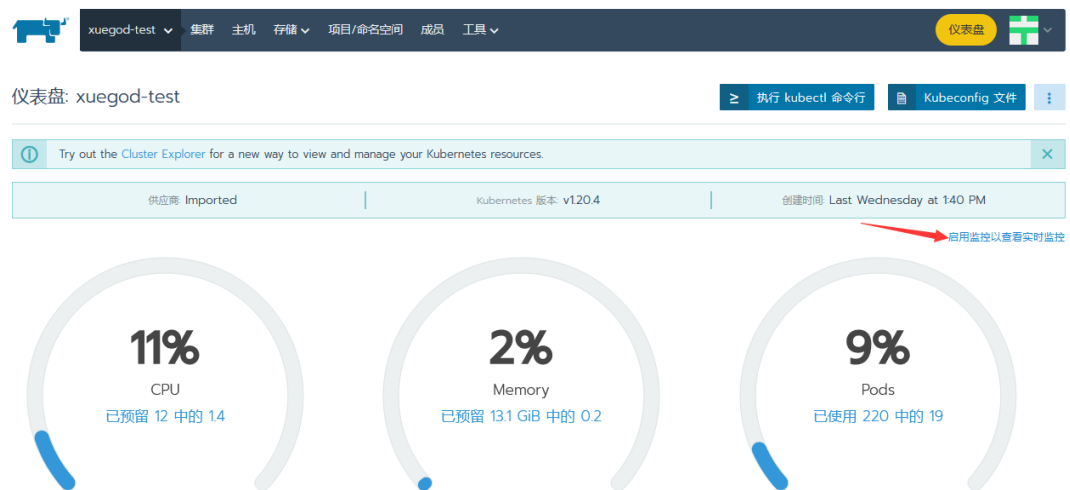
### 12.4.1 启用 Rancher 集群级别监控

启动监控时间可能比较长, 需要等 10-20 分钟

在 rancher 主页面, 点击集群名称 xuegod-test

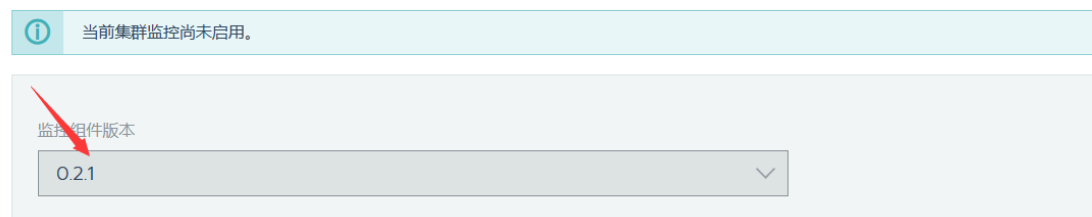


### 启动监控并查看实时监控指标



### 集群监控配置

启用集群监视时，需要确保工作节点和 Prometheus pod 具有足够的资源。请访问 [Rancher docs](#) 了解建议的资源限制。



组件版本用 0.2.1，其他默认就可以了，点最后：启用监控

选择部署监控相关的工作负载所运行的节点。

[+ 添加选择器](#)

PrometheusPod的容忍

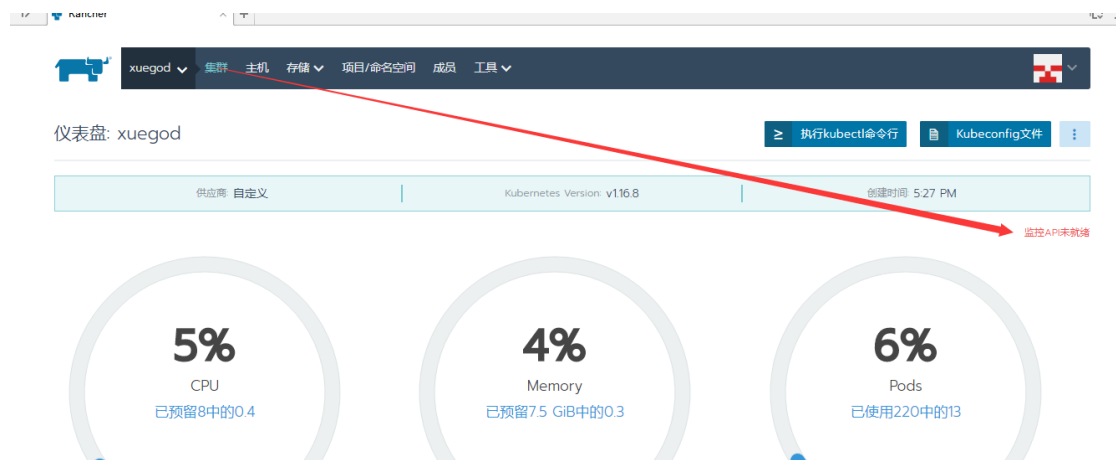
[+ 添加调度容忍](#)

为Grafana启用持久化存储

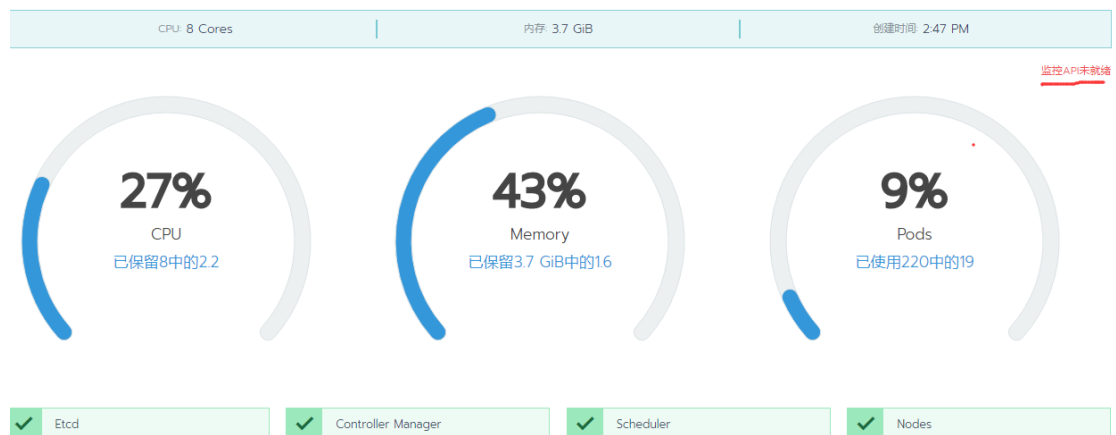
☐ 是 ☒ 否

[启用监控](#)

## 点：集群

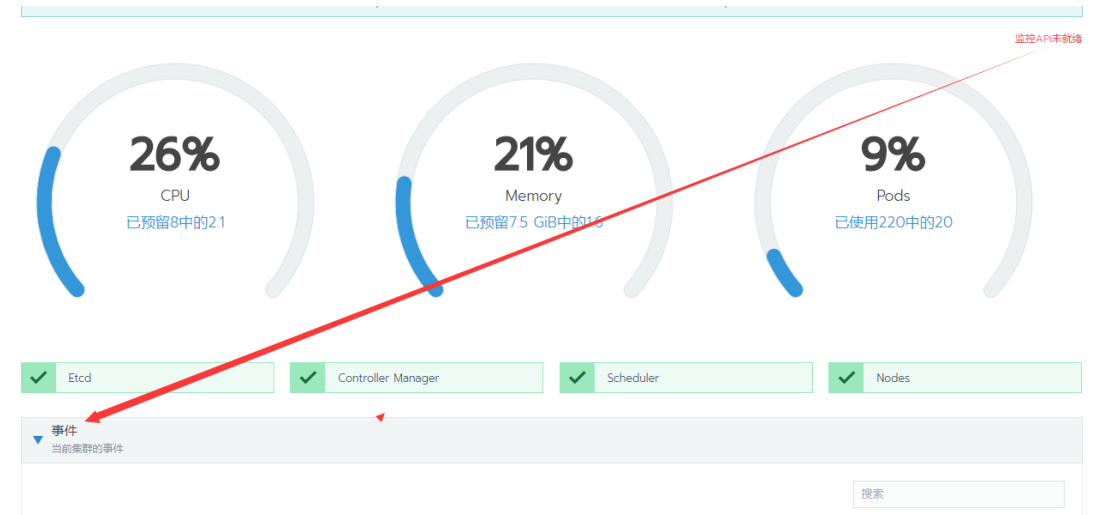


要等待监控 API 就绪，才可以显示出来



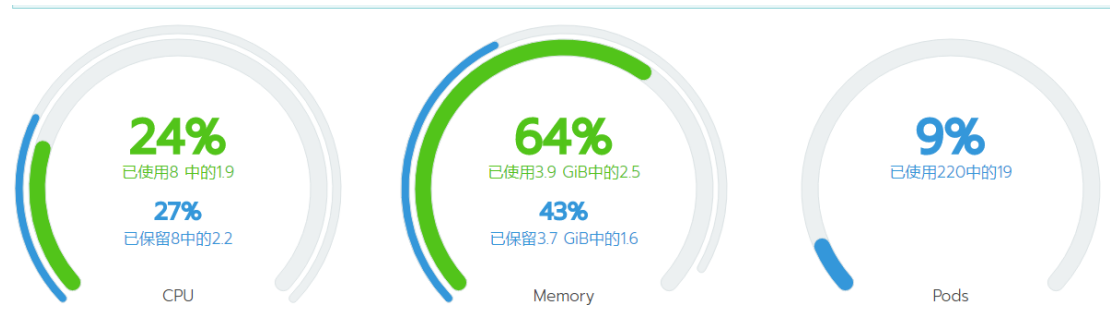
此处，需要等待 20 分钟，等待集群把监控 API 安装上。

在这个里可以查看过安装的过程：



具体事件如下：

命名空间	类型	事件原因	对象	事件信息	最后更新
cattle-prometheus	Normal	Started	prometheus-cluster-monitoring-0	Started container prometheus-proxy	a few seconds ago
cattle-prometheus	Normal	Pulling	prometheus-cluster-monitoring-0	Pulling image "rancher/prometheus-auth:v0.2.0"	a few seconds ago
cattle-prometheus	Normal	Pulled	prometheus-cluster-monitoring-0	Successfully pulled image "rancher/nginx:1.17.4-alpine"	a few seconds ago
cattle-prometheus	Normal	Created	prometheus-cluster-monitoring-0	Created container prometheus-proxy	a few seconds ago

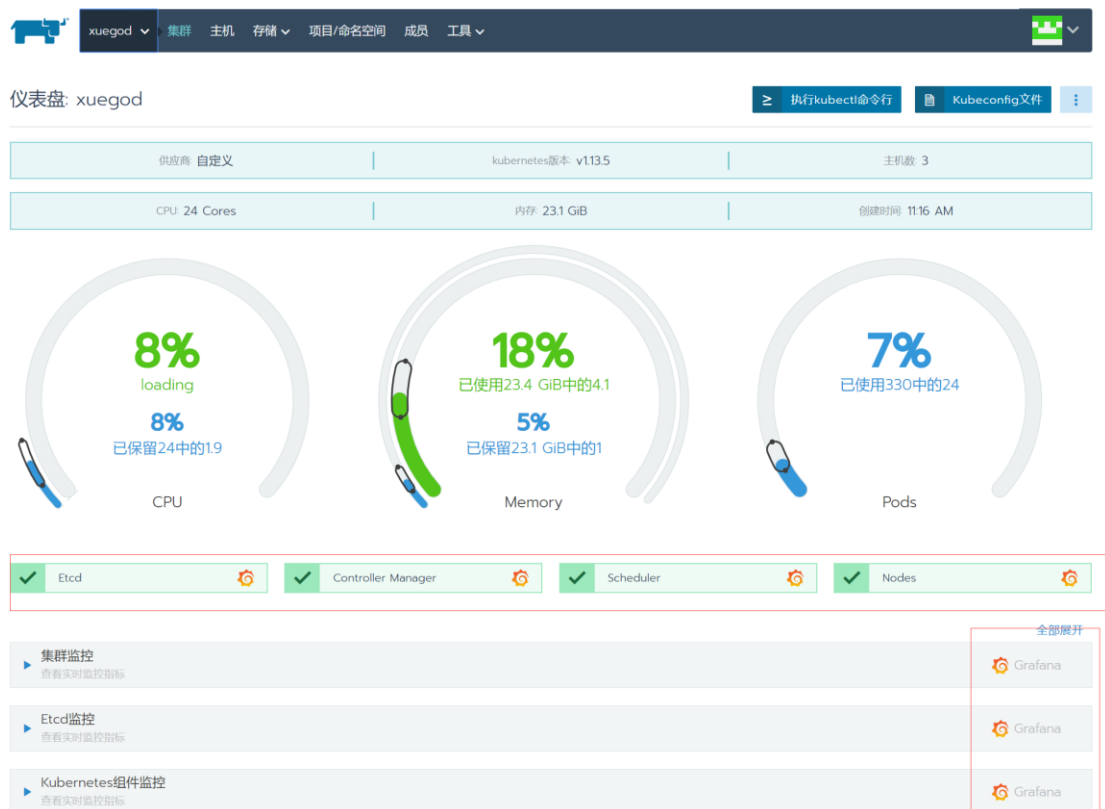


看到这个界面后，再等待 15 分钟，让集群把 grafana 组件安装上。查看事件：

命名空间	类型	事件原因	对象	事件信息	最后更新
cattle-prometheus	Warning	Unhealthy	grafana-cluster-monitoring-65cbc8749-76pwp	Readiness probe failed: Get http://10.42.15.3000/api/health: dial tcp 10.42.15.3000: connect: connection refused	32 minutes ago
cattle-prometheus	Normal	Started	grafana-cluster-monitoring-65cbc8749-76pwp	Started container grafana-proxy	33 minutes ago
cattle-..	Normal	Created	grafana-cluster-monitoring-..	Created container grafana-proxv	33 minutes ago

可以看到已经开始在安装 grafana 相关组件了。

刷新当前页面，监控部署完成后就可以看到我们的监控信息了。



我们点开集群监控，选择时间为五分钟，因为默认是一小时，而我们刚刚开启监控图标还没有足够的数据。



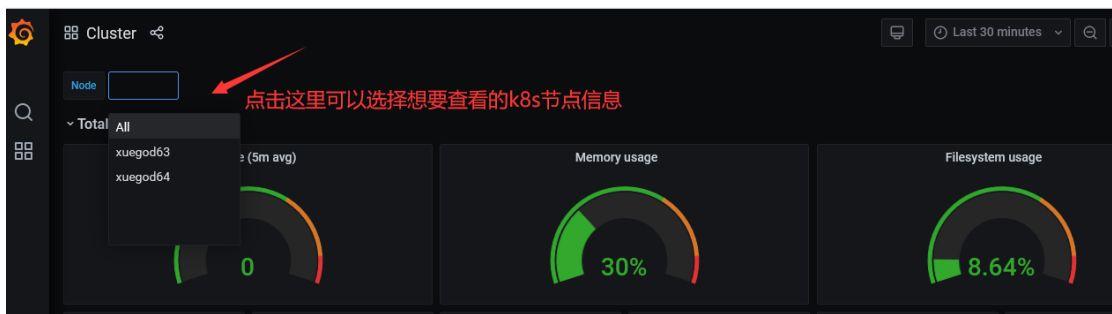
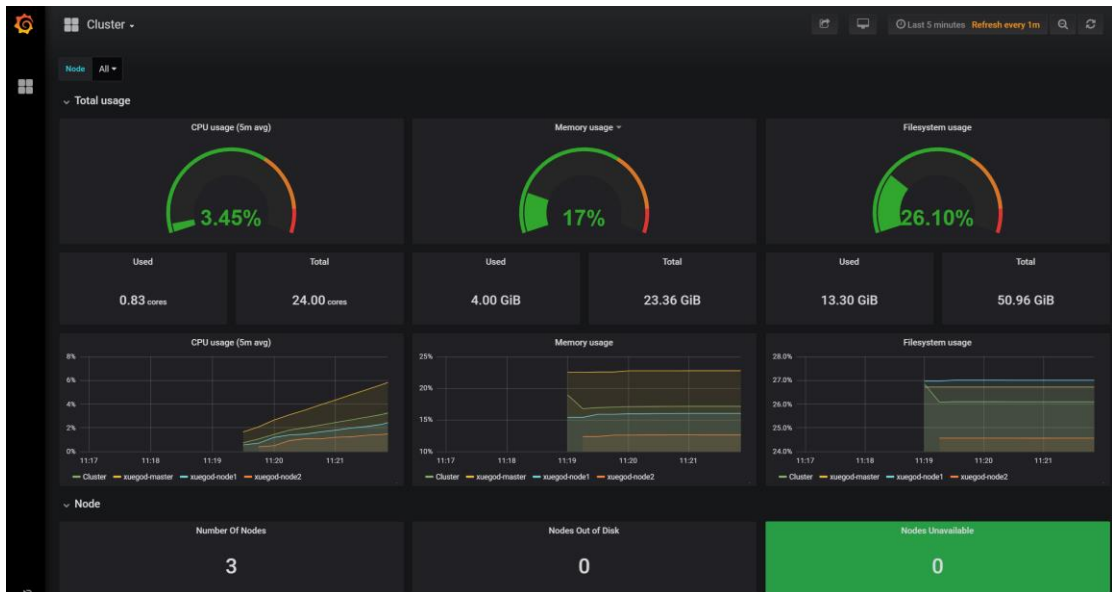
#### 12.4.2 查看 Grafana 监控

我们可以点击每个项目上的 Grafana 图标即可跳转到 Grafana 监控页面



点击后我们可以跳转到 Grafana 监控界面





## 12.5 通过 Rancher 仪表盘管理 k8s: 部署 web 站点

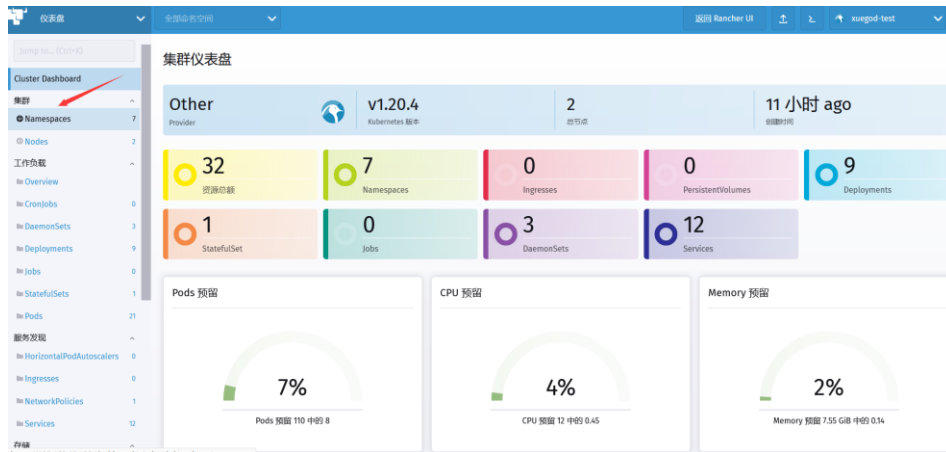
打开 rancher 主页面

### 1、创建名称空间

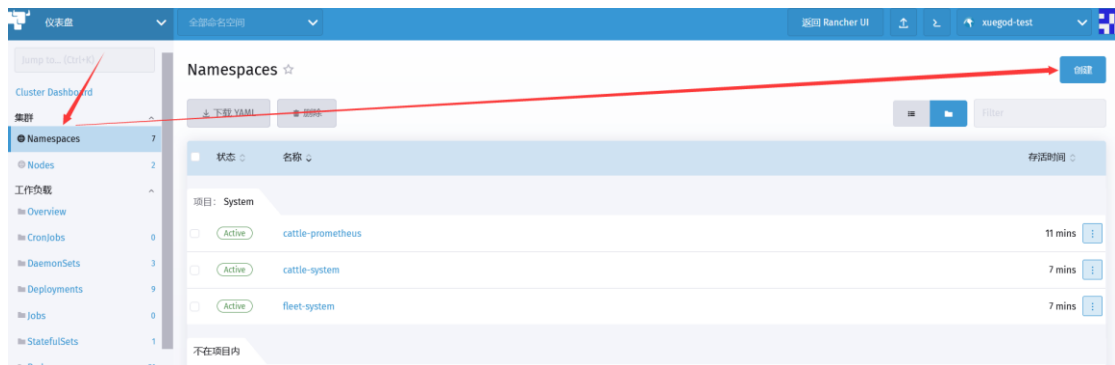
The image shows the Rancher dashboard with a table of clusters. The table has columns for status, cluster name, provider, host count, processor, and memory. There are two clusters listed: 'local' and 'xuegod-test'. A red arrow points to the '仪表盘' (Dashboard) button for the 'xuegod-test' cluster.

状态	集群名称	供应商	主机数	处理器	内存	操作
Active	local	K3s v1.18.8-k3s1	1	0.1/6 Cores 2%	0.1/5.7 GiB 1%	仪表盘
Active	xuegod-test	Imported v1.20.4	2	0.5/12 Cores 4%	0.1/7.6 GiB 2%	仪表盘

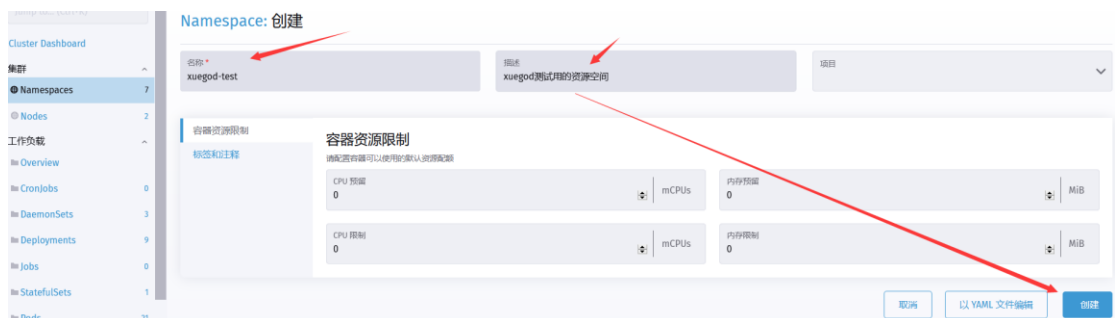
点击仪表盘



点击 Namespace:



点击创建:

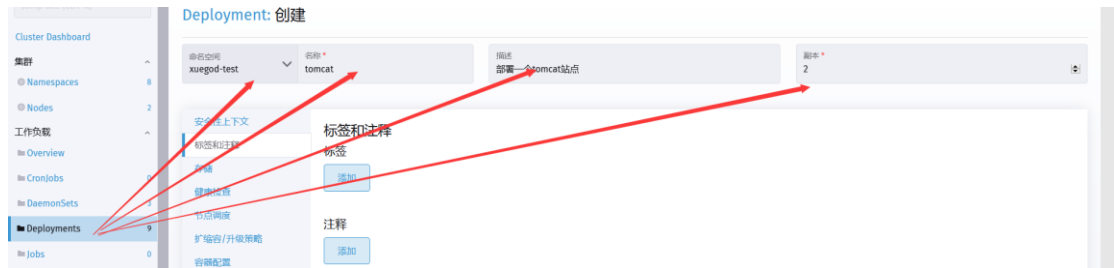


把 Namespace 名称写上, 其他默认即可, 点击创建

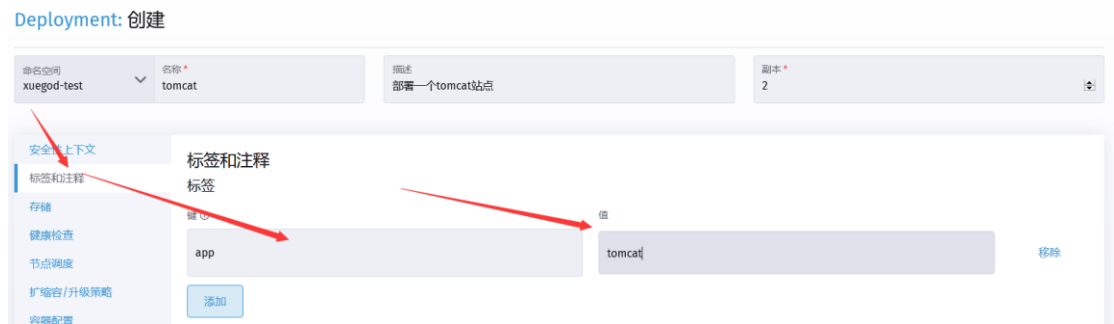
## 2、创建 Deployment 资源



点击创建:

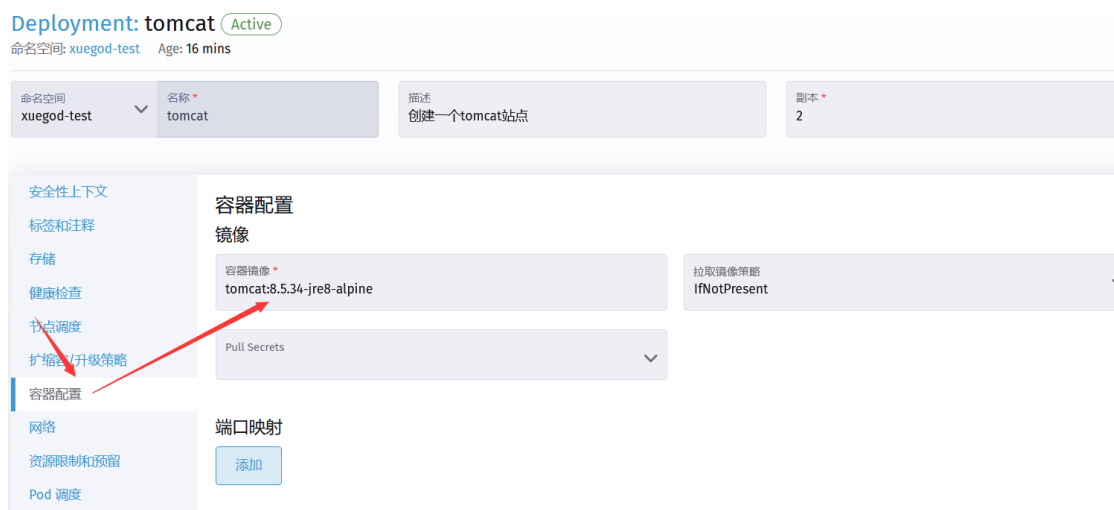


## 添加标签



## 容器配置

### #指定镜像



### #给 pod 打上标签

容器配置

网络

资源限制和配额

Pod 调度

端口映射

添加

命令

入口 (Entrypoint)  
e.g. /bin/sh

命令 (CMD)  
e.g. /usr/sbin/httpd -f httpd.conf

工作目录  
e.g. /myapp

标准输入  
No

环境变量

添加

Pod 标签

键	值	移除
workload.user.cattle.io/workloadselector	apps.deployment-xuegod-test-tomcat	移除
app	tomcat	移除

容器配置完成, 点击创建

容器配置

网络

资源限制和配额

Pod 调度

端口映射

添加

命令

入口 (Entrypoint)  
e.g. /bin/sh

命令 (CMD)  
e.g. /usr/sbin/httpd -f httpd.conf

工作目录  
e.g. /myapp

标准输入  
No

环境变量

添加

Pod 标签

添加

Pod 注释

添加

取消

以 YAML 文件编辑

创建

查看资源是否创建成功:

仪表盘

全部命名空间

返回 Rancher UI

2.

xuegod-test

命名空间: cattle-system

Active	cattle-cluster-agent	rancher/rancher-agent:v2.5.7	1/1	1	1	27 mins	...
--------	----------------------	------------------------------	-----	---	---	---------	-----

命名空间: fleet-system

Error	fleet-agent	rancher/fleet-agent:v0.3.4	0/1	1	0	27 mins	...
-------	-------------	----------------------------	-----	---	---	---------	-----

命名空间: kube-system

Active	calico-kube-controllers	calico/kube-controllers:v3.18.0	1/1	1	1	10 天	...
Active	coredns	registry.aliyuncs.com/google_containers/coredns:1.7.0	2/2	2	2	10 天	...
Active	default-http-backend	registry.cn-hangzhou.aliyuncs.com/hachikou/defaultbackend:1.0	1/1	1	1	2.5 天	...
Active	nginx-ingress-controller	registry.cn-hangzhou.aliyuncs.com/peter1009/nginx-ingress-controller:0.20.0	1/1	1	1	2.5 天	...

命名空间: xuegod-test

Active	tomcat	tomcat	2/2	2	2	24 secs	...
--------	--------	--------	-----	---	---	---------	-----

点击左侧的 deployment, 找到 xuegod-test 名称空间, 可以看到 tomcat

命名空间: xuegod-test

Active	tomcat	tomcat	2/2	2	2	1.7 mins	...
--------	--------	--------	-----	---	---	----------	-----

## 查看 tomcat 详细信息

**Deployment: tomcat** Active

命名空间: xuegod-test Age: 2.2 mins

描述: 创建一个tomcat站点  
Image: tomcat  
标签: [app: tomcat]  
注释: Show 1 annotation

**Pods by State**

0	Active
0	Transitioning
0	警告
0	错误

**Pods** 条件 最近事件 相关资源

状态	名称	节点	镜像
Running	tomcat-56bf8897-7wp4n	xuegod64	tomcat
Running	tomcat-56bf8897-rsxdc	xuegod64	tomcat

## 3、创建 Service 资源, 把 k8s 集群内部的 tomcat 暴露出来

仪表盘 全部命名空间 返回 Rancher UI

服务发现

HorizontalPodAutoscalers 0

Ingresses 0

NetworkPolicies 1

**Services 12**

创建

下载 YAML 删除

状态 名称 类型 目标端口 选择器 存活时间

## Service: 创建

**IP** 集群 IP [More Info](#)

在集群内部 IP 上公开服务。选择此值使服务只能从集群内部访问。这是默认类型。

**EN** 外部 DNS 服务名称 [More Info](#)

将服务与“externalName”字段的内容(如 foo.bar.example.com)进行映射, 返回一个带有其值的 CNAME 记录。没有设置任何形式的代理。

**H** Headless [More Info](#)

既没有定义集群 IP, 也没有定义负载均衡器。这些是用来与 Kubernetes 实现之外的其他服务发现机制对接的。没有分配集群 IP, kube-proxy 也不处理这些服务。

**NP** 节点端口 [More Info](#)

在每个节点的 IP 上以静态端口 (“NodePort”) 公开服务。您将能够通过请求: 从集群外部联系这种类型的服务。

**LB** 负载均衡器 [More Info](#)

使用云提供商的负载均衡器向外部暴露服务。

## 选择节点端口

## #定义服务端口

### Service: 创建 NodePort

命名空间: xuegod-test

名称: tomcat-service

描述: 请输入一些能更好地描述该资源的文字

定义服务端口

会话保持

监听 IP

选择器

标签和注释信息

定义服务端口

端口名称: tomcat

监听端口: 8080

目标端口: 8080

节点端口: 30180

添加

取消

以 YAML 文件编辑

创建

### #定义选择器

命名空间: xuegod-test

名称: tomcat-service

描述: 请输入一些能更好地描述该资源的文字

定义服务端口

会话保持

监听 IP

选择器

标签和注释信息

选择器

Matches 0 of 23 pods. If no selector is created, manual endpoints must be made.

添加

从文件读取

移除

取消

以 YAML 文件编辑

创建

### 创建

### 查看 service 是否创建成功

Services13

存储

PersistentVolumes0

StorageClasses0

ConfigMaps

PersistentVolumeClaims1

Secrets59

RBAC

ClusterRoleBindings77

ClusterRoles91

RoleBindings41

命名空间: default

Active

kubernetes

集群 IP

10.96.0.1:443 / TCP > 6443

10 天

命名空间: kube-system

Active

default-http-backend

集群 IP

10.97.44.233:80 / TCP > 8080

k8s-app=default-http-backend

2.5 天

Active

kube-dns

集群 IP

10.96.0.10:53 / UDP > 53  
10.96.0.10:53 / TCP > 53

k8s-app=kube-dns

10 天

命名空间: xuegod-test

Active

tomcat-service

节点端口

30180/TCP

app=tomcat

8 mins

### 通过上面可以看到已经创建了 tomat-service


命名空间: xuegod-test	节点端口	30180/TCP	app=tomcat	8 mins
-------------------	------	-----------	------------	--------

点击节点端口 30180/TCP, 可以访问内部的 tomcat 了

## Apache Tomcat/8.5.34



If you're seeing this, you've successfully installed Tomcat. Congratulations!



**Recommended Reading:**  
[Security Considerations HOW-TO](#)  
[Manager Application HOW-TO](#)  
[Clustering/Session Replication HOW-TO](#)

[Server Status](#)  
[Manager App](#)  
[Host Manager](#)

**Developer Quick Start**

<a href="#">Tomcat Setup</a> <a href="#">First Web Application</a>	<a href="#">Realms &amp; AAA</a> <a href="#">JDBC DataSources</a>	<a href="#">Examples</a>	<a href="#">Servlet Specifications</a> <a href="#">Tomcat Versions</a>
---	--	--------------------------	---

## 12.6 实战-分布式 LNMP 环境部署电商网站

为了方便同学们理解 K8S 中集群服务是如何工作的, 究竟服务是以什么方式进行通讯的, 所以我们实践搭建 LNMP 环境来深入体验。

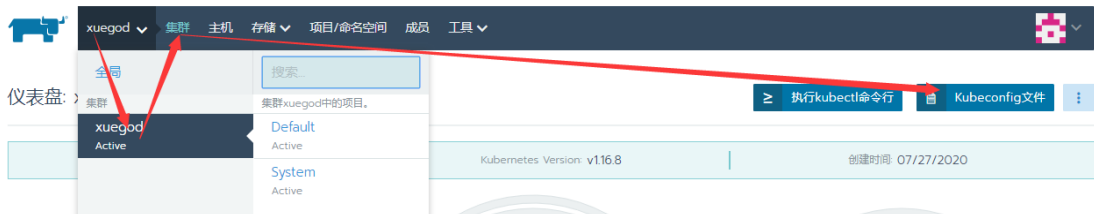
### 安装 kubectl

离线安装

上传课程资料中的 kubectl

```
[root@xuegod62 ~]# chmod +x kubectl && mv kubectl /usr/local/bin/kubectl
```

### 创建 kubectl 配置文件



### 复制配置文件到剪切板

```
- name: "xuegod-xuegod62"
  context:
    user: "xuegod"
    cluster: "xuegod-xuegod62"

current-context: "xuegod"
```

复制到剪贴板

然后 下载 kubectl (如有需要) 并运行。

关闭

```
[root@xuegod62 ~]# mkdir ~/.kube
```

粘贴配置文件, 注意粘贴时检查文件头是否缺少。

```
[root@xuegod62 ~]# vim ~/.kube/config
```

```
apiVersion: v1
kind: Config
clusters:
- name: "xuegod"
  cluster:
    server: "https://192.168.1.63/k8s/clusters/c-h2x54"
    certificate-authority-data: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSU0tLS0tCk1JSUM3akNDQ\
    WRhZ0F3SUJBZ0lCQURBTkNa3Foa2lH0XcwQkFRc0ZBREFvTVJJd0VBWURWUVRXdsMGFHVXQKY\
    21GdVkyZ3hFakFRQmd0VkJBTVRDV05oZEhSc1pTMWpZVEFlRncweU1EQTRNRE13T1RReE5UQmFGd\
    zB6TURBNAPNREV3T1RReE5UQmFNQ2d4RWpBUUJnTlZCQW9UQ1hSb1pTMXlZVzVqYURFU01CQUdBM\
    VVFQXhNSlkyRjBkR3hsCkxXTmhNSU1CSWpBTkNa3Foa2lH0XcwQkFRRUZBQU9DQVE4QU1JSUJDZ\
    0tDQVFFQXk5SUdCYVBmT0JHV1hYcVEKOWtBM2FBTHI2bldXZHRGUTBvcjR3VGtJ0Ww0eHlkQWhib\
    llaYnZKS1RsakdGNHpdU0s4M3BDRW4vbRyYcTU0awpuVW1kM3p5S1ljUTNGejhFUIIvY2FlTkplLQ\
    2xrRhdERlpmM2RiVVc1N2tHcm56d2NST3FHbmFr0EoxNUwUUVJkluZnkwNFRiSHFZMVE1aXVmt\
    ERjkb5NRFDf0EVnc2hHZE1QRkVTcHNEU0tVRXpt0XNuQ1RETmpqWk40ZzNocnoKK1dBQ0NYOWI4R\
    kLIYjF5aEFoNFM2OWIXaU85d0RuTnBwVnhKcTFRZjZobHNUTGtsQyYtiZmptUFBmWFhQTHdmZwpTZ\
    2sraGYvMkZ0YldTRENGaWVlRVFxaUtyZXRjbHjd2FKZnBTOTg4RHZoaanRS0DZZSXXJJehBIZXp2Z\
    XhXY3dUCksWRGpod0lEQVFBQm95TXdJVEFPQmd0VkhR0EJBZjhFQkFNQ0F0XUdEd1lEVlIwVEFRS\
    C9CQV3QXdfQ196QU4KQmdrcWhraUc5dzBCQVZFRkFBT0NBUEU0VjBdEjY1ozTlMya29vai9lWTArN\
    zhCck9oWk5yT3IzZHNxY1d6b0lLWgp1Tk4WwLLamtlSHE1RjFTbU1NVHdiRnJHbCswZUZ6cDZzR\
    UYxNE9ZVGRFVS9mMy9JK3J5b3RSc1V0RkRoUGxLCK1CVDFiMzNuUVBZVlc4WGZrVlZoaC9PQ1JTU\
    TJh0Eo2bmdNRHFqMFZHTDdWahZ0c0EYNTZHMxiYs1I1Uj16TE4KLy82MXg4T1F2eCt0WjB4RXNDQ\
    1BGY29wdk5kNEpZQTFKRUXyY3BnMWVvdjhLSmtkQTfKRMjYnM1mUmkyRk10awpsMFZIZlZnSitUV\
    VN5RTc3bHVQRnJxZEVFZGZRY2lwdHEzdWsxYjJ0Z0VSdjZBN05MMURFYTAzS2FjcWVldWNNc1BvT\
    VZoVE9wVGLCSTdlVVowSlRyZXc5TncxNXRYnNGSGpmSDFBMk50VXM0N0E9PQotLS0tLUVORCBDR\
    VJUSUZJQ0FURSU0tLS0t"
```

#### 测试 kubectl 命令

```
[root@xuegod62 ~]# kubectl get node
```

NAME	STATUS	ROLES	AGE	VERSION
xuegod63	Ready	control-plane,master	55d	v1.20.4
xuegod64	Ready	worker	55d	v1.20.4

### 12.6.1 部署 nfs 共享存储和 pv

实验背景说明: nginx 和 php 不在一台机器上, 怎么运行?

nginx pod 192.168.1.71 192.168.1.62/data/2.php -> php

php pod 192.168.1.72 找不到网站根下找不/data/2.php, 不能解析!

注: nginx 转给 php 服务是一个 php 脚本的路径, 不是脚本内容!

必须把 nginx 网站根共享给 php 所在的服务器, 而且路径要一样。

#### 创建数据卷

```
[root@xuegod63 ~]# mkdir -p /web/{html,data}
```

注: 生产环境可以使用 raid 来保证数据可靠性。

```
[root@xuegod63 ~]# yum -y install nfs-utils rpcbind #安装 nfs
```

```
[root@xuegod63]# systemctl start nfs
```

```
[root@xuegod63 ~]# vim /etc/exports
```

```
/web/html *(rw,no_root_squash)
```

```
/web/data *(rw,no_root_squash)
```

\*: 表示任何人都有权限连接, 当然也可以是一个网段, 一个 IP, 也可以是域名



rw: 读写的权限

sync: 表示文件同时写入硬盘和内存

no\_root\_squash: 当登录 NFS 主机使用共享目录的使用者是 root 时, 其权限将被转换为匿名使用者, 通常它的 UID 与 GID, 都会变成 nobody 身份

```
[root@xuegod63 ~]# exportfs -arv
```

扩展: /tmp 192.168.1.0/24(ro) 允许 192.168.1.0 网段访问

配置开机自启并立即启动 rpcbind

```
[root@xuegod63 ~]# systemctl enable --now rpcbind
```

配置开机自启并立即启动 nfs (注意启动顺序, 必须先启动 rpcbind)

```
[root@xuegod63 ~]# systemctl enable --now nfs
```

## 12.6.2 k8s 的 pv 与 pvc 持久化存储

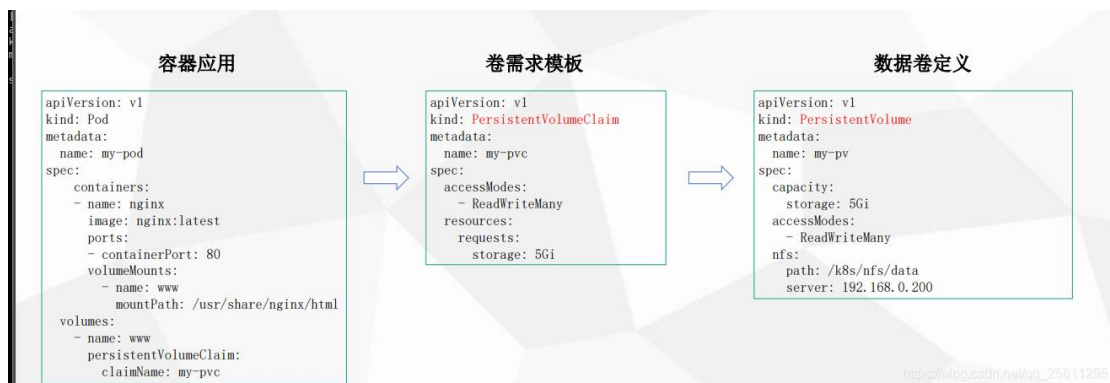
PersistentVolume (PV): 对存储资源创建和使用的抽象, 使得存储作为集群中的资源管理

PersistentVolumeClaim (PVC): 让用户不需要关心具体的 Volume 实现细节。

总的来说, **PV 是提供者, PVC 是消费者, 消费的过程就是绑定**。容器与 PV、PVC 之间的关系, 可以如下图所示:

PVC: 我需要 5GiB 的权限是读写的 pv

PV: 定义出一个 5GiB 权限是读写的 pv



解压资源文件

```
[root@xuegod63 ~]# tar xf lnmp-v9.tar.gz
```

```
[root@xuegod63 ~]# cd lnmp
```

```
[root@xuegod63 lnmp]# unzip ecshop.zip
```

```
[root@xuegod63 ~]# mv ecshop/* /web/html/
```

```
[root@xuegod63 lnmp]# vim /root/lnmp/pv.yaml #查看我准备好的 pv 文件
```

注: kind: PersistentVolume ; 种类: 持久卷 **Persistent** [pə'sɪstənt] 持久的

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv          pv name
  labels:
    apps: mysql-pv
spec:
  capacity:
    storage: 5Gi          PV大小5G
  accessModes:
    - ReadWriteMany      可读可写多客户端挂载
  nfs:
    path: /web/data       NFS路径
    server: 192.168.1.63  NFS服务器IP
```

创建 pv

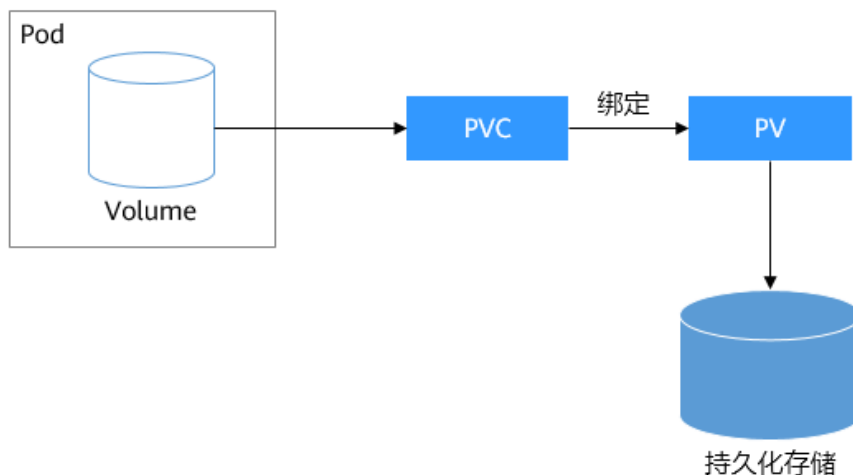
```
[root@xuegod63 lnmp]# kubectl apply -f pv.yaml
```

注: pv.yaml 一共创建了 2 个 pv 分别是 mysql-pv、web-pv

ReadWriteOnce (RWO): 读写权限, 但是只能被单个节点挂载

ReadOnlyMany (ROX): 只读权限, 可以被多个节点挂载

ReadWriteMany (RWX): 读写权限, 可以被多个节点挂载



PV 描述的是持久化存储卷, 主要定义的是一个持久化存储在宿主机上的目录, 比如一个 NFS 的挂载目录。

PVC 描述的是 Pod 所希望使用的持久化存储的属性, 比如, Volume 存储的大小、可读写权限等等。

使用过程是: nfs→pv→pvc→volume→volumeMount to path

### 12.6.3 部署 php 服务

```
[root@xuegod63 ~]# kubectl api-resources # "组/版本"的格式输出对象支持的 API 版本
```

```
[root@xuegod63 ~]# kubectl api-resources
NAME                                SHORTNAMES  APIVERSION  NAMESPACED  KIND
bindings                           v1          v1          true        Binding
componentstatuses                  cs          v1          false       ComponentStatus
configmaps                         cm          v1          true        ConfigMap
endpoints                          ep          v1          true        Endpoints
events                             ev          v1          true        Event
limitranges                       limits      v1          true        LimitRange
namespaces                         ns          v1          false       Namespace
nodes                             no          v1          false       Node
persistentvolumeclaims             pvc         v1          true        PersistentVolumeClaim
persistentvolumes                  pv          v1          false       PersistentVolume
pods                              po          v1          true        Pod
podtemplates                      v1          v1          true        PodTemplate
replicationcontrollers             rc          v1          true        ReplicationController
resourcequotas                     quota       v1          true        ResourceQuota
```

```
[root@xuegod63 lnmp]# vim php-deployment.yaml
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-server
  labels:
    name: php-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: php-server
  template:
    metadata:
      labels:
        app: php-server
    spec:
      containers:
        - name: php-server
          image: php:v1
          imagePullPolicy: IfNotPresent
          volumeMounts:
            - name: php-data
              mountPath: /var/www/html/
          ports:
            - containerPort: 9000
          lifecycle:
            postStart:
              exec:
                command: [ "/bin/bash", "-c", "chown apache:apache /var/www/html -R" ]
          volumes:
            - name: php-data
              persistentVolumeClaim:
                claimName: web-pvc
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: web-pvc
  labels:
    app: php-server
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      apps: web-pv
```

## 部署 php-deployment

```
[root@xuegod63 lnmp]# kubectl apply -f php-deployment.yaml
deployment.apps/php-server created
```

```
[root@xuegod63 lnmp]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
php-server-7688967d96-lx74j        1/1     Running   0           9s
```

部署 php-svc

```
[root@xuegod63 lnmp]# vim php-svc.yaml
```

---

apiVersion: v1

kind: Service

metadata:

name: php

spec:

ports:

- name: php

port: 9000

protocol: TCP

targetPort: 9000

selector:

app: php-server

```
[root@xuegod63 lnmp]# kubectl apply -f php-svc.yaml
```

service/php created

#### 12.6.4 部署 nginx 服务

配置文件注意事项, 该配置文件是通过 yum 安装 nginx 获取的, 课程内 nginx 镜像也是使用 centos 镜像通过 yum 的方式安装的 nginx, 同学们需要注意其它 Linux 发行版中的 Nginx 配置方式是有区别的, 课程资料中提供了 nginx 镜像制作的 Dockerfile 可自行查阅。

```
[root@xuegod63 lnmp]# vim /root/lnmp/conf/nginx.conf
```

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

http {

log\_format main '\$remote\_addr - \$remote\_user [\$time\_local] "\$request" '

'\$status \$body\_bytes\_sent "\$http\_referer" '

'"\$http\_user\_agent" "\$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

```
keepalive_timeout 65;
types_hash_max_size 2048;
include /etc/nginx/mime.types;
default_type application/octet-stream;
include /etc/nginx/conf.d/*.conf;
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    root /usr/share/nginx/html;
    include /etc/nginx/default.d/*.conf;
    location / {
        index index.php;
    }
    location ~ \.php$ {
        # root html;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME /var/www/html/$fastcgi_script_name;
        include fastcgi_params;
    }
    error_page 404 /404.html;
    location = /40x.html {
    }
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
}
```

注: nginx 的站点根目录是 `/usr/share/nginx/html` 而 php 的解析目录是 `/var/www/html` 两者仅仅是挂载到容器中的路径不一样, 但是同源。fastcgi\_pass 指定后端 php 解析地址时只需要指定 php-svc 的名称即可, 查看 svc 名称方式如下:

```
[root@xuegod63 lnmp]# kubectl get svc
```

```
[root@xuegod63 lnmp]# kubectl get svc
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes    ClusterIP     10.43.0.1       <none>           443/TCP          20h
php           ClusterIP     10.43.103.139   <none>           9000/TCP          14s
```

### ConfigMap 创建 nginx 配置文件

ConfigMap 简单介绍: ConfigMap 是 k8s 中非常重要的一个资源对象, 简称 cm, 常用于向容器中注入配置文件等操作, 不仅可以保存单个属性值, 也可以保存整个配置文件。

语法格式: `kubectl create configmap name --from-file=path`

```
[root@xuegod63 lnmp]# kubectl create configmap lnmp-nginx-config --from-
```

file=/root/lnmp/conf/nginx.conf

```
[root@xuegod63 lnmp]# kubectl get configmap #查看新生成的 ConfigMap
NAME          DATA  AGE
lnmp-nginx-config 1     131d
```

创建 nginx-deployment

```
[root@xuegod63 lnmp]# vim nginx-deployment.yaml
```

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-shop

spec:

selector:

matchLabels:

app: nginx-shop

replicas: 1

template:

metadata:

labels:

app: nginx-shop

spec:

containers:

- name: nginx-shop

image: nginx:v1

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

volumeMounts:

- name: nginx-data

mountPath: /usr/share/nginx/html

- name: nginx-conf

mountPath: /etc/nginx/nginx.conf

subPath: nginx.conf

volumes:

- name: nginx-data

persistentVolumeClaim:

claimName: web-pvc

- name: nginx-conf

configMap:

name: lnmp-nginx-config

部署 nginx

```
[root@xuegod63 lnmp]# kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-shop created
```

创建 nginx-svc

```
[root@xuegod63 lnmp]# vim nginx-svc.yaml
```

---

apiVersion: v1

kind: Service

metadata:

name: nginx-shop

spec:

type: NodePort

ports:

- name: nginx

port: 80

protocol: TCP

targetPort: 80

nodePort: 30010

selector:

app: nginx-shop

注: nginx-svc 增加了 nodePort 对外提供服务。

```
[root@xuegod63 lnmp]# kubectl apply -f nginx-svc.yaml
```

访问测试: <http://192.168.1.64:30010/>



可以看到安装界面则表示 php 解析正确没有问题。现在不要部署网站, 因为还没有安装数据库了。

#### 12.6.5 部署 mysql 服务

mysql 服务我们就不额外挂载数据卷和指定 node 运行了, 生产环境中还是建议单独挂载数据卷使用。

```
[root@xuegod63 lnmp]# vim mysql-deployment.yaml
```

---

apiVersion: apps/v1

kind: Deployment

metadata:

```
name: mysql
namespace: default
labels:
  k8s-app: mysql
spec:
  selector:
    matchLabels:
      k8s-app: mysql
  replicas: 1
  template:
    metadata:
      labels:
        k8s-app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:5.7
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 3306
              protocol: TCP
          volumeMounts:
            - name: mysql-data
              mountPath: /var/lib/mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: "123456"
      volumes:
        - name: mysql-data
          persistentVolumeClaim:
            claimName: mysql-pvc
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
  labels:
    k8s-app: mysql
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```



```
selector:
  matchLabels:
    apps: mysql-pv
```

mysql 容器中存在初始化脚本, 使用 **MYSQL\_ROOT\_PASSWORD** 变量配置 mysql 登录密码。

```
[root@xuegod63 lnmp]# kubectl apply -f mysql-deployment.yaml
deployment.apps/mysql created
```

创建 mysql-svc

```
[root@xuegod63 lnmp]# vim mysql-svc.yaml
```

---

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    k8s-app: mysql
spec:
  ports:
    - name: mysql
      port: 3306
      protocol: TCP
      targetPort: 3306
  selector:
    k8s-app: mysql
```

```
[root@xuegod63 lnmp]# kubectl apply -f mysql-svc.yaml
service/mysql created
```

## 12.7 实战-ingress-对外发布服务

发布服务是通过反向代理实现的, 也就是通过 nginx 反向代理我们 k8s 中的服务, 有些同学会想那 nginx 代理我们的服务了, 那 ingress 到底是做什么的? 原因很简单, **nginx 并不能实时监测我们后端 pod 的变化, 比如增加或减少后端服务需要手工修改 nginx 配置**, 而 ingress 有一个监听器可以通过监听 kube-apiserver 来感知后端的 service、pod 变化。然后根据 ingress 的配置将后端信息更新给我们的反向代理。

工作原理: 域名解析指向集群边缘节点, 然后 ingress 匹配域名规则将请求转发至对应的服务。

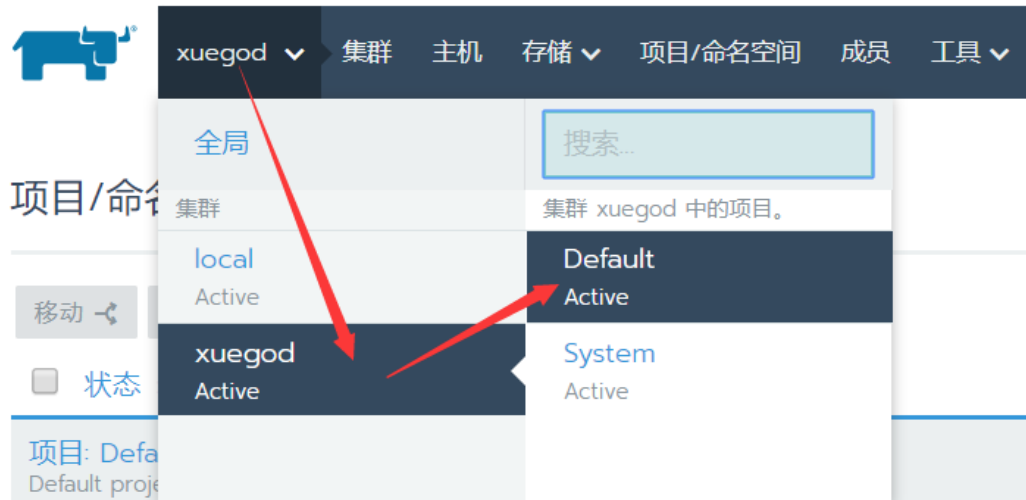
扩展: 边缘节点

边缘节点是指我们 k8s 集群中对外提供服务的节点, 因为 k8s 集群如果每一台主机都拥有公网 IP 的话其实是没有必要的, 通常我们只需要边缘节点能够对外提供服务就可以了。此时我们发布服务时只要把域名解析指向边缘节点的 ip 就可以访问 k8s 中的服务。

注: rancher 中自带了服务发布的功能, 是通过 nginx 实现的, 如果是自建的 k8s 集群则建议使用

Traefik。

### 创建 ingress 规则



### 创建名称 xuegod→自定义域名→xuegod.shop.com

#### 添加规则

名称 添加描述 命名空间 创建新的命名空间

xuegod default

规则

☐ 自动生成 xuegod.shop.com 后缀域名

☒ 自定义域名

☐ 默认后端

访问域名 xuegod.shop.com

### 删除默认的配置项，默认的是工作负载，我们需要添加服务。

① 如果目标后端是一个服务，您只能使用该服务所暴露的端口。您可以在服务发现页找到该服务，然后通过编辑 YAML 的方式添加所需要的端口。

目标后端 + 服务 + 工作负载

访问路径(如需使用后端重写功能, 请查看下方标签/注释) 服务/工作负载 容器端口

例如 /foo 选择服务... n/a

+ 添加规则

### 添加服务 (service):

规则

☐ 自动生成 `nginx` 后缀域名

☒ 自定义域名

☐ 默认后端

访问域名

xuegod.shop.com

① 如果目标后端是一个服务，您只可以使用该服务所暴露的端口。您可以在服务发现页找到该服务，然后通过编辑 YAML 的方式添加所需要的端口。

目标后端 + 服务 + 工作负载

访问路径(如需使用后端重写功能, 请查看下方标签/注释): 服务/工作负载 容器端口

例如: /foo nginx-shop nginx

服务选择 nginx-shop 端口选择 nginx 即可,

注: 端口名称 nginx 是 service ports name, 调用时可以使用端口也可以使用名称。

目标后端 + 服务 + 工作负载

访问路径(如需使用后端重写功能, 请查看下方标签/注释): 服务/工作负载 容器端口

例如: /foo nginx-shop nginx

+ 添加规则

填好配置请勿点击添加规则, 直接保存即可。

保存 取消

状态	名称	目标	创建时间
命名空间: default			
<input type="checkbox"/> Initializing	xuegod L7 Ingress	xuegod.shop.com > nginx-shop	11:20 PM

稍等 2 分钟:

命名空间: default			
<input type="checkbox"/> Active	xuegod L7 Ingress	xuegod.shop.com > nginx-shop	11:20 PM

添加本地 hosts 解析。

C:\Windows\System32\drivers\etc

名称	修改日期	类型	大小
HOSTS	2020/8/4 18:03	文件	2 KB

添加行:

192.168.1.64 xuegod.shop.com

浏览器访问: <http://xuegod.shop.com/>

界面语言

只需三步,即可完成安装

1 欢迎使用ECSHOP

2 检查环境

3 西

© 2005-2016 上海商派软件有限公司。保留所有权利。

感谢您选择 ECShop 网上商店管理系统。希望我们的努力能为您提供一个高效快速和强大的电子商务解决方案。ECShop 英文全称为 e-Commerce Shop, 中文全称为 ECShop 网上商店管理系统, 简称 ECShop。  
上海商派软件有限公司为 ECShop 产品的开发商, 依法独立拥有 ECShop 产品著作权(版权登记号: 2008SRBJ0394)。ECShop 官方网站为 <http://www.ECShop.com>。  
ECShop 著作权已在中华人民共和国国家版权局注册, 著作权受到法律和国际公约保护。使用者: 无论个人或组织、盈利与否、用途如何 (包括以学习和研究为目的), 均需仔细阅读本协议, 在理解、同意、并遵守本协议的全部条款后, 方可开始使用 ECShop 软件。  
本授权协议适用且仅适用于 ECShop 2.x.x 版本, 上海商派软件有限公司拥有对本授权协议的最终解释权。

I. 协议许可的权利

1. 您可以在完全遵守本最终用户授权协议的基础上, 将本软件应用于非商业用途(包括个人用户: 不具备法人资格的自然人, 以个人名义从事电子商务开设网店的; 非盈利性用途: 从事非盈利活动的商业机构及非盈利性组织, 将 ECShop 产品用且仅用于产品演示、展示及发布, 而并不是用来买卖及盈利的运营活动的)

2. 您可以在协议规定的约束和限制范围内修改 ECShop 源代码(如果被提供的活)或界面风格以适应您的网站要求。

3. 您拥有使用本软件构建的商店中全部会员资料、文章、商品及相关信息的所有权, 并独立承担与其内容的相关法律义务。

4. 获得商业授权之后, 您可以将本软件应用于商业用途, 同时依据所购买的授权类型中确定的技术支持期限、技术支持方式和技术支持内容, 自授权时刻起, 在技术支持期限内拥有通过指定的方式获得指定范围内的技术支持服务。商业授权用户享有反映和提出意见的权力, 相关意见将被作为首要考虑, 但没有一定被采纳的承诺或保证。

II. 协议规定的约束和限制

☒ 我已仔细阅读, 并同意上述条款中的所有内容

下一步: 配置安装环境

学神 IT 教育官方 QQ 群: 957231097 或唐老师 QQ: 3340273106 领取更多资料

只需三步,即可完成安装

1 欢迎使用ECSHOP

2 检查环境

3 配置系统

系统环境

操作系统.....	Linux
PHP 版本.....	5.4.16
是否支持MySQL.....	支持
GD 版本.....	2
是否支持 JPEG.....	不支持
是否支持 GIF.....	支持
是否支持 PNG.....	支持
重要文件是否完整.....	完整
服务器是否开启安全模式.....	关闭

目录权限检测

cert.....	可写
images.....	可写
images/upload.....	可写
images/upload/Image.....	可写
images/upload/File.....	可写
images/upload/Flash.....	可写
images/upload/Media.....	可写
data.....	可写
data/afficheimg.....	可写
data/brandlogo.....	可写
data/cardimg.....	可写
data/feedbackimg.....	可写
data/packimg.....	可写
data/sqldata.....	可写
temp.....	可写
temp/backup.....	可写
temp/caches.....	可写
temp/compiled.....	可写
temp/query_caches.....	可写
temp/static_caches.....	可写

模板可写性检查

所有模板,全部可写

上一步: 欢迎页

重新检查

下一步: 配置系统

注: 管理员 admin 的密码要大于或等于 8 位, 且最少要包括字母和数字。我的密码: xuegod123

只需三步,即可完成安装

1 欢迎使用ECSHOP

2 检查环境

3 数据库设置

#### 数据库帐号

数据库主机: mysql  
端口号: 3306  
用户名: root  
密码: .....  
数据库名: ecshop  
表前缀: ecs\_ (建议您修改表前缀)

主机: mysql 填写service name即可  
端口: service 端口  
用户名: root  
密码: 123456

已有数据库 ▾ 搜 数据库名称任意

#### 管理员帐号

管理员姓名: admin  
登录密码: .....  
密码强度: 弱 中 强  
密码确认: .....  
电子邮箱:

后台账户: admin  
密码: 注意密码复杂度。

#### 杂项

选择语言包: ☒ 简体中文 ☐ 繁体中文 ☐ English  
设置时区: 亚洲, 中国, 上海  
禁用验证码: ☐ (选择此项, 进入后台、发表评论无需验证)  
预选商品类型: ☐ 书 ☒ 音乐 ☒ 电影 ☒ 手机 ☒ 笔记本电脑  
数码相机 ☒ 数码摄像机 ☒ 化妆品 ☒ 精品手机  
安装测试数据: ☒ (选择此项, 将默认全选预选商品类型)

上一步: 检测系统环境

立即安装

恭喜您, ECSHOP 已经成功地安装完成。  
基于安全的考虑, 请在安装完成后删除 install 目录。

- 前往 ECSHOP 首页
- 前往 ECSHOP 后台管理中心

© 2005-2016 上海商派软件有限公司。保留所有权利。

<http://xuegod.shop.com/>



<http://shop.com/admin>



前后台页面均正常。

多个服务的发布: 多个域名解析到边缘节点, ingress 创建域名对应的后端服务规则即可。反向代理会先匹配域名规则再将请求调度到后端, 和基于域名的虚拟主机一样。

## 总结:

- 12.1 Rancher 简介
- 12.2 安装 rancher
- 12.3 通过 Rancher 管理已存在的 k8s 集群
- 12.4 使用 Rancher 中自带的监控功能查看 k8s 集群运行状态
- 12.5 通过 Rancher 仪表盘管理 k8s: 部署 web 站点
- 12.6 实战-分布式 LNMP 环境部署电商网站
- 12.7 实战-ingress-对外发布服务