

一、简介

Rancher简介

来源官方：<https://www.cnrancher.com/>

Rancher是一个开源的企业级容器管理平台。通过Rancher，企业再也不必自己使用一系列的开源软件去从头搭建容器服务平台。Rancher提供了在生产环境中使用的管理Docker和Kubernetes的全栈化容器部署与管理平台。

Rancher由以下四个部分组成:

1.1、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。Rancher仅需要主机有CPU，内存，本地磁盘和网络资源。从Rancher的角度来说，一台云厂商提供的云主机和一台自己的物理机是一样的。

Rancher为运行容器化的应用实现了一层灵活的**基础设施服务**。Rancher的基础设施服务包括**网络**，**存储**，**负载均衡**，**DNS**和安全模块。Rancher的基础设施服务也是通过容器部署的，所以同样Rancher的基础设施服务可以运行在任何Linux主机上。

1.2、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。

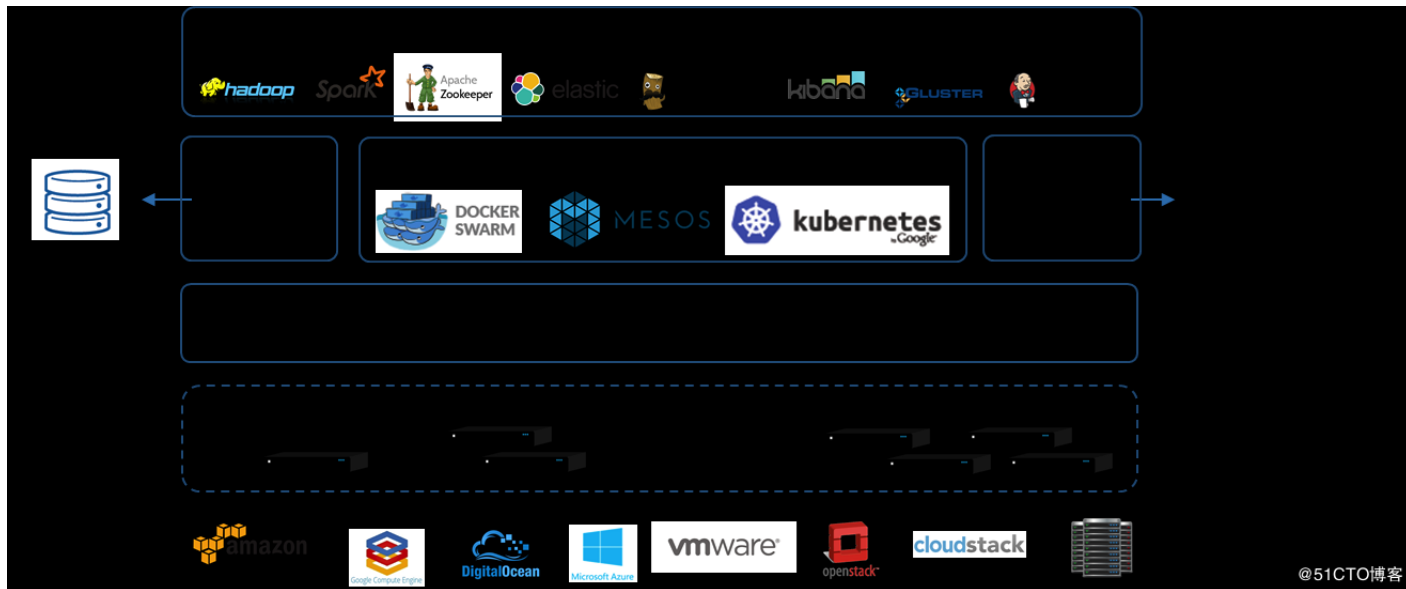
1.3、应用商店

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。

1.4、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](#)。 Rancher支持在[环境级别](#)的基于角色的访问控制 (RBAC)， 可以通过角色来配置某个用户或者用户组对开发环境或者生产环境的访问权限。

下图展示了Rancher的主要组件和功能:



@51CTO博客

Kubernetes简介

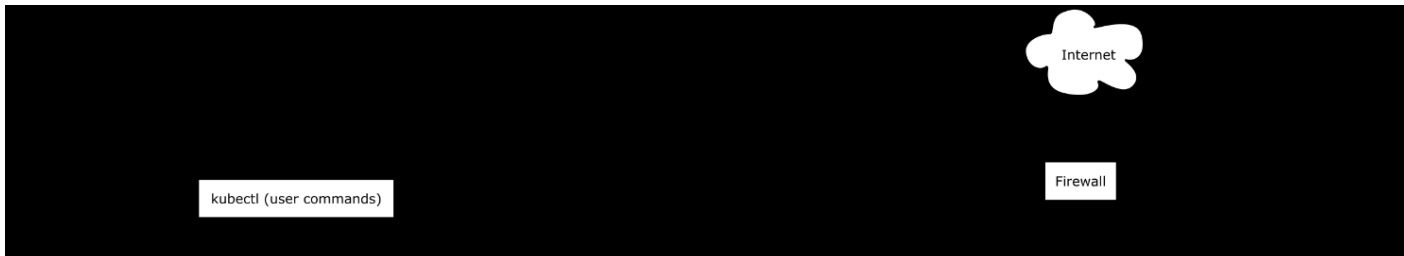
1.1 基础概念

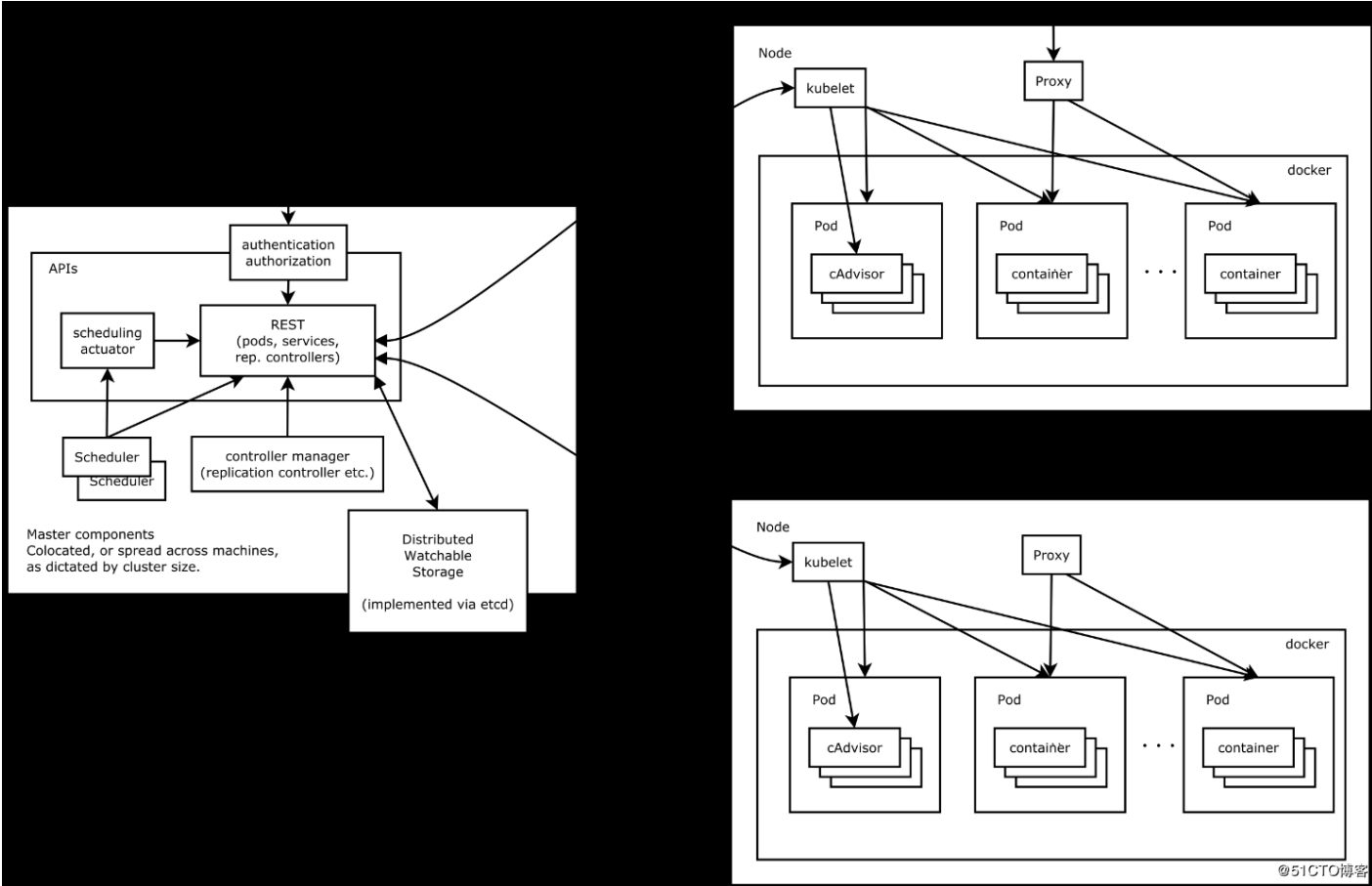
Kubernetes（通常写成“k8s”）Kubernetes是Google开源的容器集群管理系统。其设计目标是在主机集群之间提供一个能够自动化部署、可拓展、应用容器可运营的平台。Kubernetes通常结合docker容器工具工作，并且整合多个运行着docker容器的主机集群，Kubernetes不仅仅支持Docker，还支持Rocket，这是另一种容器技术。

功能特性：

- 自动化容器部署与复制
- 随时扩展或收缩容器规模
- 组织容器成组，提供容器间的负载均衡
- 快速更新及回滚容器版本
- 提供弹性伸缩，如果某个容器失效就进行替换

1.2 架构图





1.3 组件

1.3.1 Master

Master节点上面主要由四个模块组成：APIServer、scheduler、controller manager、etcd

- APIServer:APIServer负责对外提供RESTful的Kubernetes API服务，它是系统管理指令的统一入口，任何对资源进行增删改查的操作都要交给APIServer处理后再提交给etcd。如架构图中所示，kubectl（Kubernetes提供的客户端工具，该工具内部就是对Kubernetes API的调用）是直接和APIServer交互的。
- scheduler:scheduler的职责很明确，就是负责调度pod到合适的Node上。如果把scheduler看成一个黑匣子，那么它的输入是pod和由多个Node组成的列表，输出是Pod和一个Node的绑定，即将这个pod部署到这个Node上。Kubernetes目前提供了调度算法，但是同样也保留了接口，用户可以根据自己的需求定义自己的调度算法。
- controller manager:如果说APIServer做的是“前台”的工作的话，那controller manager就是负责“后台”的。每个资源一般都对应有一个控制器，而controller manager就是负责管理这些控制器的。比如我们通过APIServer创建一个pod，当这个pod创建成功后，APIServer的任务就算完成了。而后面保证Pod的状态始终和我们预期的一样的重任就由controller manager去保证了。

- etcd:etcd是一个高可用的键值存储系统，Kubernetes使用它来存储各个资源的状态，从而实现了Restful的API。

1.3.2 Node

每个Node节点主要由三个模块组成：kubelet、kube-proxy、runtime。

runtime指的是容器运行环境，目前Kubernetes支持docker和rkt两种容器。

- kube-proxy:该模块实现了Kubernetes中的服务发现和反向代理功能。反向代理方面：kube-proxy支持TCP和UDP连接转发，默认基于Round Robin算法将客户端流量转发到与service对应的一组后端pod。服务发现方面，kube-proxy使用etcd的watch机制，监控集群中service和endpoint对象数据的动态变化，并且维护一个service到endpoint的映射关系，从而保证了后端pod的IP变化不会对访问者造成影响。另外kube-proxy还支持session affinity。
- kubelet:Kubelet是Master在每个Node节点上面的agent，是Node节点上面最重要的模块，它负责维护和管理该Node上面的所有容器，但是如果容器不是通过Kubernetes创建的，它并不会管理。本质上，它负责使Pod得运行状态与期望的状态一致。

1.3.3 Pod

Pod是k8s进行资源调度的最小单位，每个Pod中运行着一个或多个密切相关的业务容器，这些业务容器共享这个Pause容器的IP和Volume，我们以这个不易死亡的Pause容器作为Pod的根容器，以它的状态表示整个容器组的状态。一个Pod一旦被创建就会放到Etcd中存储，然后由Master调度到一个Node绑定，由这个Node上的Kubelet进行实例化。

每个Pod会被分配一个单独的Pod IP，Pod IP + ContainerPort 组成了一个Endpoint。

1.3.4 Service

Service其功能使应用暴露，Pods 是有生命周期的，也有独立的 IP 地址，随着 Pods 的创建与销毁，一个必不可少的工作就是保证各个应用能够感知这种变化。这就要提到 Service 了，Service 是 YAML 或 JSON 定义的由 Pods 通过某种策略的逻辑组合。更重要的是，Pods 的独立 IP 需要通过 Service 暴露到网络中。

二、准备工作

2.1、系统环境

主机名	系统	IP	作用

master	CentOS 7.4	192.168.56.129	主控制节点
slave1	CentOS 7.4	192.168.56.130	业务节点

下面两个节点都要配置

2.2、检查hosts--配置后检查是否能解析外网

192.168.56.129 master

192.168.56.130 slave1

2.3、暂时关闭防火墙和seLinux

2.4、开启IPV4转发

在[/etc/sysctl.conf](#)新添加如下参数

```
net.ipv4.ip_forward = 1
```

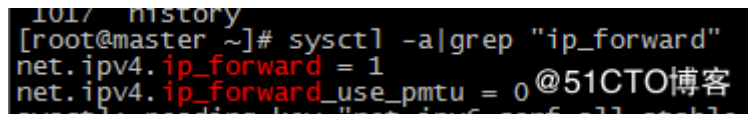
```
net.ipv4.ip_forward_use_pmtu = 0
```

生效命令：

```
[root@master ~]# sysctl -p
```

查看

```
[root@master ~]# sysctl -a|grep "ip_forward"
```



```
1017 ~ history
[root@master ~]# sysctl -a|grep "ip_forward"
net.ipv4.ip_forward = 1
net.ipv4.ip_forward_use_pmtu = 0 @51CTO博客
sysctl: reading key "net.ipv6.conf.all.stable"
```

2.5、关闭Swap交换分区

2.6、安装Docker1.12.6版本

什么版本的Docker才能适配Rancher和Kubernetes

请参考：<http://rancher.com/docs/rancher/v1.6/zh/hosts/#docker>

DOCKER版本适用对比

版本	Rancher适用？	K8S适用？	安装脚本
----	------------	--------	------

1.9.x 和更低的版本	No		
1.10.0 - 1.10.2	No		
1.10.3 (和更高的版本)	No (Yes v1.6.5以及更低版本中)	No	<code>curl https://releases.rancher.com/install-docker/1.10.sh sh</code>
1.11.x	No		<code>curl https://releases.rancher.com/install-docker/1.11.sh sh</code>
1.12.0 - 1.12.2	No		
1.12.3 (和更高的版本)	Yes	Yes	<code>curl https://releases.rancher.com/install-docker/1.12.sh sh</code>
1.13.x	Yes	No	<code>curl https://releases.rancher.com/install-docker/1.13.sh sh</code>
17.03.x-ce	Yes	No	<code>curl https://releases.rancher.com/install-docker/17.03.sh sh</code>
17.03.x-ee	Yes	No	n/a
17.04.x-ce	No		<code>curl https://releases.rancher.com/install-docker/17.04.sh sh</code>
17.05.x-ce	No	No	<code>curl https://releases.rancher.com/install-docker/17.05.sh sh</code>
17.06.x-ce	Yes (v1.6.3以及更高版本)	No	<code>curl https://releases.rancher.com/install-docker/17.06.sh sh</code>
17.06.x-ee	Yes (v1.6.3以及更高版本)	No	n/a

@51CTO博客

1) 执行命令:

```
[root@master ~]# mkdir -p ~/_src
```

```
[root@master ~]# cd ~/_src/
```

```
[root@master _src]# wget http://yum.dockerproject.org/repo/main/centos/7/Packages/docker-engine-selinux-1.12.6-1.el7.centos.noarch.rpm
```

```
[root@master _src]# wget http://yum.dockerproject.org/repo/main/centos/7/Packages/docker-engine-1.12.6-1.el7.centos.x86_64.rpm
```

```
[root@master _src]# wget http://yum.dockerproject.org/repo/main/centos/7/Packages/docker-engine-debuginfo-1.12.6-1.el7.centos.x86_64.rpm
```

安装

```
[root@master _src]# yum localinstall -y docker-engine-selinux-1.12.6-1.el7.centos.noarch.rpm docker-engine-1.12.6-1.el7.centos.x86_64.rpm docker-engine-debuginfo-1.12.6-1.el7.centos.x86_64.rpm
```

```
Installed:
  docker-engine.x86_64 0:1.12.6-1.el7.centos
  docker-engine-selinux.noarch 0:1.12.6-1.el7.centos
  docker-engine-debuginfo.x86_64 0:1.12.6-1.el7.centos
@51CTO博客
```

2) 启动

```
[root@master ~]# systemctl enable docker
```

```
[root@master ~]# systemctl start docker
```

3) 查看版本

```
[root@master ~]# docker version
```

```

[root@master ~]# docker version
Client:
 Version:      1.12.6
 API version:  1.24
 Go version:   go1.6.4
 Git commit:   78d1802
 Built:        Tue Jan 10 20:20:19 2017
 OS/Arch:      linux/amd64
 Experimental: true

Server:
 Version:      1.12.6
 API version:  1.24
 Go version:   go1.6.4
 Git commit:   78d1802
 Built:        Tue Jan 10 20:20:19 2017
 OS/Arch:      linux/amd64
 Experimental: true
@51CTO博客
```

2.7、设置Docker镜像加速

此时如果用docker pull命令下载镜像，本地会连接hub.docker.com网站去下载，耗时较长，因此我们可以设置docker镜像加速，使得本地连接去国内镜像仓库下载，镜像加速的设置有很多种，本章以阿里云的设置为例，步骤如下：

1) 创建目录：

```
[root@master ~]# mkdir /etc/docker
```

2) 设置镜像仓库地址:

```
tee /etc/docker/daemon.json <<-'EOF'{ "registry-mirrors": ["https://xwx6wxd1.mirror.aliyuncs.com"]
}EOF
```

3) 重新加载配置:

```
[root@master ~]# systemctl daemon-reload
```

4) 重启服务

```
[root@master ~]# systemctl restart docker.service
```

备注: slave1节点操作一致

三、安装rancher

官方安装文档: <https://www.cnrancher.com/docs/rancher/v2.x/cn/overview/>

3.1、在master机器执行以下命令, 即可安装rancher:

```
[root@master ~]# docker run -d --restart always --name rancher-server -p 8080:8080 rancher/serve
r:v1.6.11-rc3 && docker logs -f rancher-server
```

报错如下：

```
8787404fc79c: Pull complete
Digest: sha256:7a83e6519b81c2b6987d5cb8deec088d8a4b407ac4a032701cadecf9ba83b599
Status: Downloaded newer image for rancher/server:v1.6.11-rc3
58311c95aa338373c647564d6bdfb8621b3ffface89ab51d687f9a4a6d36237b
docker: Error response from daemon: driver failed programming external connectivity on endpoint rancher-server
(2a0c320043f8dac49054af684ecfe7aa0bba39c6b44647f0e14e308ccdb4b603): iptables failed: iptables --wait -t nat -A
DOCKER -p tcp -d 0/0 --dport 8080 -j DNAT --to-destination 172.17.0.2:8080 ! -i docker0: iptables: No chain/target
get/match by that name.
(exit status 1).
```

@51CTO博客

参考地址：https://blog.csdn.net/shida_csdn/article/details/79376761

[root@master ~]# pkill docker #终止进程

[root@master ~]# iptables -t nat -F #清空nat表的所有链

[root@master ~]# ifconfig docker0 down #停止docker默认网桥

[root@master ~]# brctl delbr docker0 #删除网桥

[root@master ~]# systemctl restart docker #重启docker

查看即可

3.2、在浏览器访问http://192.168.56.129:8080，可以看到初始页面，在页面的右下角选择“简体中文”后，页面如下所示：



至此，rancher安装成功，接下来就是kubernetes的搭建工作。

3.3、配置环境模板

1)、环境配置---“Default”选择“环境管理”，如图



2) 添加环境模块

 Default 应用 应用商店 基础架构 系统管理 API

 在添加第一个服务或容器之前，必须至少添加一台安装了支持的Docker版本的Linux主机。添加主机

环境

添加环境

Rancher 支持将资源分组归理到多个环境。每个环境具有自己独立的基础架构资源及服务，并由一个或多个用户、团队或组织所管理。

例如，您可以创建独立的“开发”、“测试”及“生产”环境以确保环境之间的安全隔离，将“开发”环境的访问权限赋予全部人员，但限制“生产”环境的访问权限给一个小的团队。

状态	名称	描述	模板
 Unhealthy	Default	无描述	Cattle

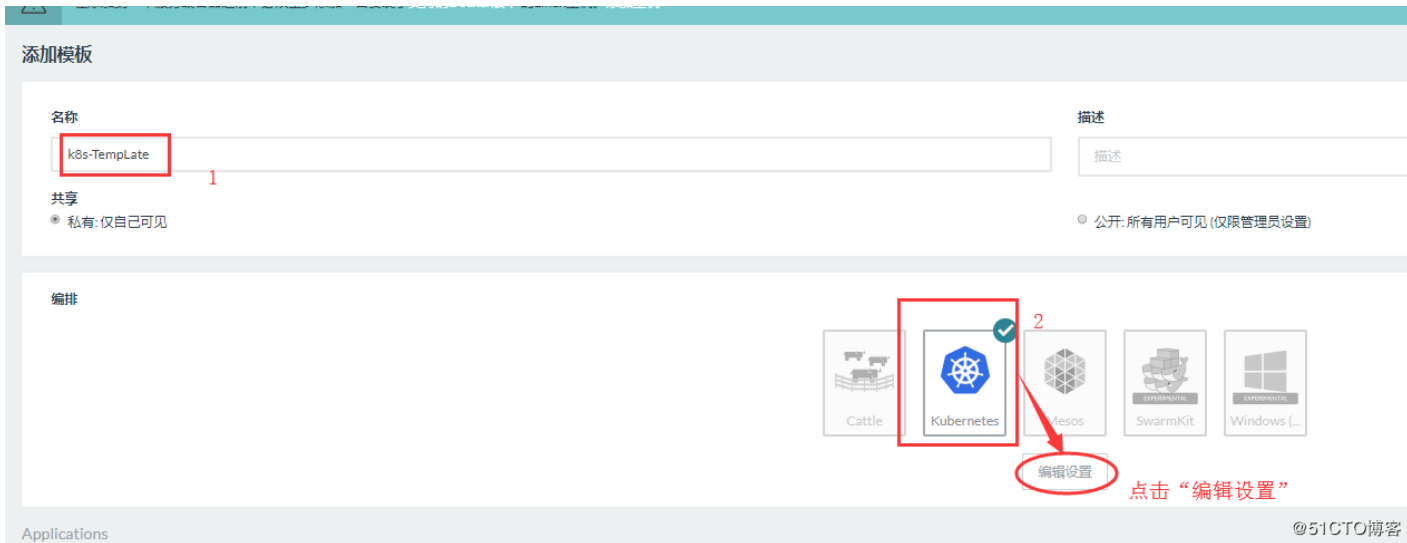
环境模板

添加环境模板

环境模板允许用户定义一个需要部署的基础设施服务的组合。

基础设施服务包括但不限于容器编排（例如：cattle、kubernetes、mesos、swarm、网络）或rancher服务（例如：健康检查、DNS、元数据、调度、服务发现和存储）。

输入项目名：k8s-TempLate



如下图，下拉菜单只有一个选择，请选中



应用商店: Library
类别: Orchestration
支持: 官方认证

模板版本

v1.8.3-rancher3

选择一个模板版本设置

Kubernetes v1.8.3

Software Versions

- Kubernetes v1.8.3
- Etcd v2.3.7

Upgrading to this Version

Warning: The existing template version *must be* **v1.2.4-rancher9** or later. Ignoring th

Changelog for Kubernetes v1.8.3

- Added configurable value to service cluster IP cidr.
- Added Azure cloud provider support.
- Added configurable log verbosity levels for add-ons.
- Fixed add-ons for RBAC enabled setups

Required Open Ports on hosts

The following TCP ports are required to be open for **kubectl**: **10250** and **10255**. To a

Plane Isolation

If you want to separate the planes for resiliency by labeling your hosts to separate out th
etcd=true, are required on your hosts in order for Kubernetes to successfully launch. f

KubeDNS

KubeDNS is enabled for name resolution as described in the [Kubernetes DNS docs](#). The

@51CTO博客

Private Registry for Add-Ons and Pod Infra Container Image

registry.cn-shenzhen.aliyuncs.com

The registry to pull addon images and the optional pod infra container image. This should be set to the private registry from which the images for addon-services (i.e. helm, heapster, dashboard, dns) and the pod infra container image should be pulled. (Default: gcr.io).

Image namespace for kubernetes-helm Image

rancher_cn

Image namespace for kubernetes-helm (Default: kubernetes-helm).

Enable Rancher Ingress Controller*

☒ True ☐ False

Deploy the Rancher ingress controller which automatically provisions Rancher load balancers in response to Kubernetes ingress creation events.

Enable Kubernetes Add-ons*

☒ True ☐ False

Setting this to "false" will disable Dashboard, Helm, SkyDNS, and other Kubernetes add-on services. This is meant for advanced users who want to deploy custom versions of these add-ons.

Pod Infra Container Image

rancher_cn/pause-amd64:3.0

The image whose network/ipc namespaces containers in each pod will use. It uses kubelet default if left empty. Do not add the private registry to this field, update the "Private Registry for Add-Ons and Pod Infra Container Image" field with the registry information.

Service Cluster IP CIDR*

10.43.0.0/16

A CIDR notation IP range from which to assign service cluster IPs. This must not overlap with any IP ranges assigned to the cluster pod IP CIDR (10.42.0.0/16).

Image namespace for Add-Ons and Pod Infra Container Image

rancher_cn

Image namespace for Add-Ons and Pod Infra Container Image (Default: gcr.io).

Sky DNS service scale

1

Number of replicas for SKY DNS service. Each replica would run on

Fail on swap*

false

Kubelet will fail if swap is enabled on the host.

Log verbosity level for Kubernetes Add-ons

2

Log verbosity level for the Kubernetes Add-ons. Numbers 0-4 are d

Audit Logs

☐ True ☒ False

Enable Audit logs to stdout for the kube-apiserver

KubeDNS cluster IP*

10.43.0.10

The IP address assigned to kubedns service. This must not overlap with any IP ranges assigned to the cluster pod IP CIDR (10.42.0.0/16).

上图四个红框填入的内容如下表所示：

名称	值
Private Registry for Add-Ons and Pod Infra Container Image	registry.cn-shenzhen.aliyuncs.com
Image namespace for kubernetes-helm Image	rancher_cn
Pod Infra Container Image	rancher_cn/pause-amd64:3.0
Service Cluster IP CIDR*	10.43.0.0/16

Private Registry for Add-Ons and Pod Infra Container Image	registry.cn-shenzhen.aliyuncs.com
Image namespace for Add-Ons and Pod Infra Container Image	rancher_cn
Image namespace for kubernetes-helm Image	rancher_cn
Pod Infra Container Image	rancher_cn/pause-amd64:3.0

3) 将页面拖动到最底部，点击“设置”按钮，如下图：

Etc heartbeat Interval*

500

Time (in milliseconds) of a heartbeat interval.

Rancher LB name separator

rancherlb

Separator used in Rancher LB names generated by ingress controller

no_proxy to be used by Kubernetes services

rancher.internal,cluster.local,rancher-metadata,rancher-kubernetes-auth,kubernetes,169.254.169.254,169.254.169.250,10.42.0.0/16,10.43.0.0/16

Comma separated string used to be excluded from the proxy

Etc Election Timeout*

5000

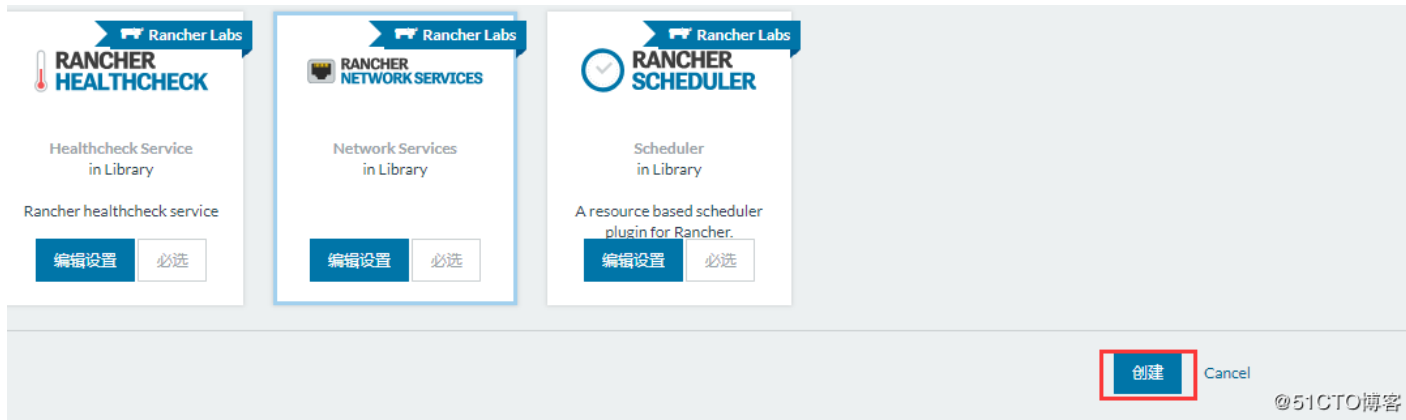
Time (in milliseconds) for an election to timeout.

HTTP proxy to be used by Kubernetes services

To be used when Kubernetes services are deployed on host(s) behind a proxy, i.e.: http(s)://<fqdn>:<port>



4) 再将页面拖动到最底部，点击“创建”按钮，如下图：



这样我们就完成了环境模板的配置，这里面的参数帮助rancher寻找国内的镜像仓库，从而避免了无法从google仓库下载镜像的问题，在以往这个问题是通过科学上网来解决的；

3.4、创建Kubernetes

1) 点击“创建环境”按钮，如下图红框：



在添加第一个服务或容器之前，必须至少添加一台安装了支持的Docker版本的Linux主机。添加主机

环境

添加环境

Rancher 支持将资源分组归属到多个环境。每个环境具有自己独立的基础架构资源及服务，并由一个或多个用户、团队或组织所管理。

例如，您可以创建独立的“开发”、“测试”及“生产”环境以确保环境之间的安全隔离，将“开发”环境的访问权限赋予全部人员，但限制“生产”环境的访问权限。

状态	名称	描述
Unhealthy	Default	无描述

@51CTO博客

2) 在创建环境的页面中，输入新的环境的名称：master-k8s，选择我们刚才创建的环境模板，在点击底部的“创建”按钮，如下图：

添加环境

名称

master-k8s1

描述

例如: 开发试验环境

环境模板

Cattle

k8s-Templ...
Orchestration: Kubernetes
Framework: Healthcheck Service, Network Services, Scheduler
Networking: Rancher IPsec

Kubernetes

Mesos

Swarm

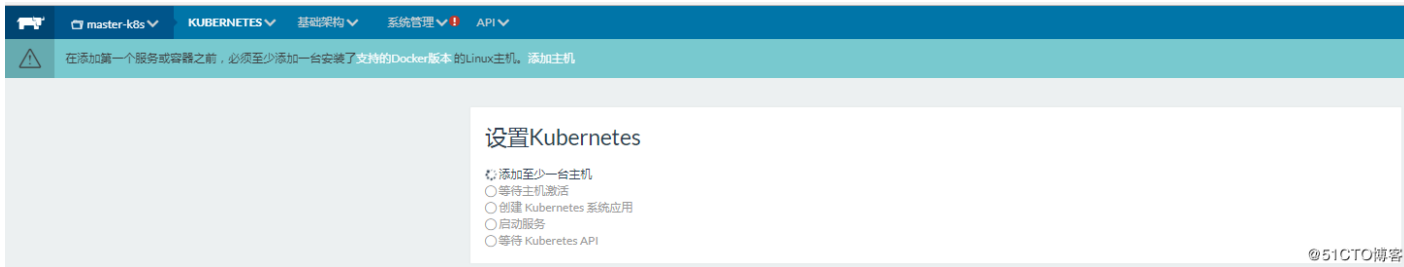
访问控制

未启用访问控制。
任何访问API/UI的人都具有管理员权限，并能够使用任意环境。

3 创建 Cancel

@51CTO博客

3) 如下图红框所示，在左上角位置选择刚刚创建的环境，可以看到目前环境已经OK，正在等待node的加入：



至此，Kubernetes的master已经搭建完毕！！！！

3.5、添加节点——将机器加入到K8S环境

1) master机器的IP是192.168.56.129，所以在浏览器打开地址192.168.56.129:8080，左上角选择新增的环境，可以看到如下图的页面，点击红框中的“添加主机”：



2)如下图，在页面上确认红框中的IP地址是不是你的master机器对外暴露的地址(多网卡的机器要关注)，确认无误后点击“保存”。

主机: 添加主机

主机注册地址

主机连接Rancher API的Base URL是？

● 当前站点地址:

http://[redacted].8080

● 其他地址:

例如:http://example.com:8080

不要包含 /v1 或任何其他路径，但如果你设置了 SSL Termination(SSL终止) 在 Rancher 前面，请确保使用 https://。

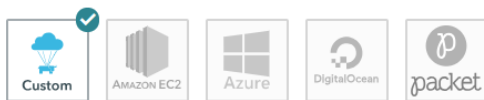


确定要创建的所有主机都能够连接 http://[redacted].8080 ?
当前地址似乎是私有IP或内部网络。

保存

@51CTO博客

3)如下图，点击红框按钮，会将此按钮左侧的文本信息复制下来：



管理docker-machine驱动

1. Start up a Linux machine somewhere and install a [supported version of Docker](#) on it.
 2. 确认安全组或防火墙允许以下通讯:
 - 与其他所有主机之间的 UDP 端口 500 和 4500 (用于IPsec网络)
 3. 可选项: 在主机上增加标签

⊕ 添加标签
 4. 指定用于注册这台主机的公网IP。如果留空, Rancher会自动检测IP注册。通常在主机有唯一公网IP的情况下这是可以的。如果主机位于防火墙/NAT设备之后, 或者主机同时也是运行 **rancher/server** 容器的主机时, 则必须设置此IP。

例如: 1.2.3.4
 5. 将下列脚本拷贝到每一台主机上运行以注册 Rancher:


```
sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.7-rc2 http://192.168.1.100:8080/v1/scr  
ipts/192.168.1.100:8080:CG76Bdn54MkQHLvbf5TK1x1y6Ek
```
 6. 点击下面的关闭按钮, 新的主机注册后会显示在 **主机** 页面。

关闭

@51CTO博客

4)登录slave1节点, 执行上面复制下来的命令, 该命令会先下载docker镜像, 然后启动容器去加入到K8S环境, 此时再去刷新管理页面, 见到如下图所示, 已经感知到机器的加入, 开始接下来的一系

列操作，此时请耐心等待（等待时间比较，喝杯茶再回来）：



有可能节点获取不到东西，建议检查一下安全规则（防火墙、转发、selinux），配置后**重启**即可

```
[root@slave1 ~]# docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.7-rc2 http://192.168.56.129:8080/v1/scripts/77946269E3BA0BCE619B:15100000:CG76Bdn54MkQHLVbfstKlxiy6Ek
```

```
INFO: Running Agent Registration Process, CATTLE_URL=http://192.168.56.129:8080/v1
INFO: Attempting to connect to: http://192.168.56.129:8080/v1
ERROR: http://192.168.56.129:8080/v1 is not accessible (Connection timed out after 15002 milliseconds)
ERROR: http://192.168.56.129:8080/v1 is not accessible (Connection timed out after 15001 milliseconds)
ERROR: http://192.168.56.129:8080/v1 is not accessible (Connection timed out after 15001 milliseconds)
ERROR: http://192.168.56.129:8080/v1 is not accessible (Connection timed out after 15000 milliseconds)
ERROR: http://192.168.56.129:8080/v1 is not accessible (Connection timed out after 15001 milliseconds)
```

@51CTO博客

5) 节点加入成功后，页面如下图所示，点击红框中的按钮就进入了K8S的dashboard：



报Service unavailable错误：等待十分钟左右即可，启动接口有点慢

下文安装kubectl装好之后，在控制台用kubectl describe命令查看dashbroad的pod和service的执行进度，查看错误日志。

至此，我们已经完成了节点机器加入K8S环境的操作，接下来我们快速体验在K8S环境创建Pod和Service的操作；

体验K8S环境

1) 创建一个文件tomcat.yaml，内容如下：

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: tomcat001

spec:

replicas: 1

template:

metadata:

labels:

name: tomcat001

spec:

containers:

- name: tomcat001

image: tomcat:7.0.82-jre7

tty: true

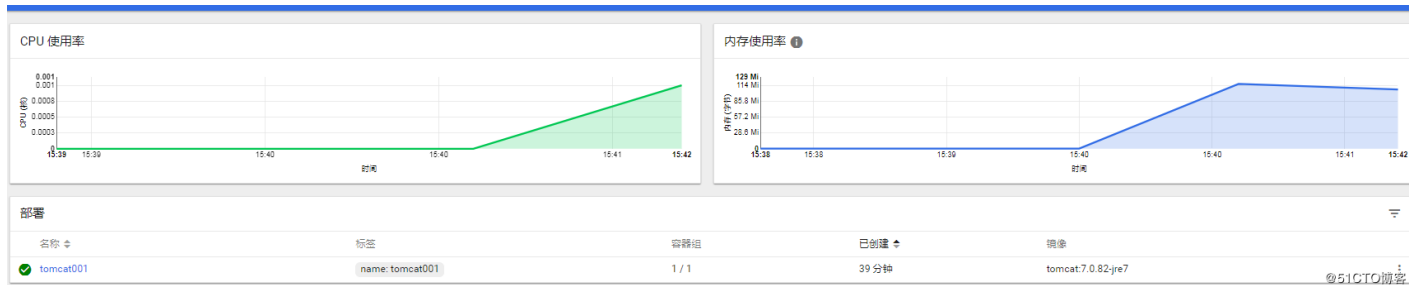
ports:

- containerPort: 8080

2) 在dashboard页面上上传这个tomcat.yaml文件，操作如下图所示：



3) 等镜像下载和容器创建成功后，在dashboard的部署页面可以看到tomcat001的部署情况，如下图



4) 创建一个文件tomcat-svc.yaml，内容如下：

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
  name: tomcat001
```

```
spec:
```

```
  type: NodePort
```

```
  ports:
```

- port: 8080

nodePort: 30018

selector:

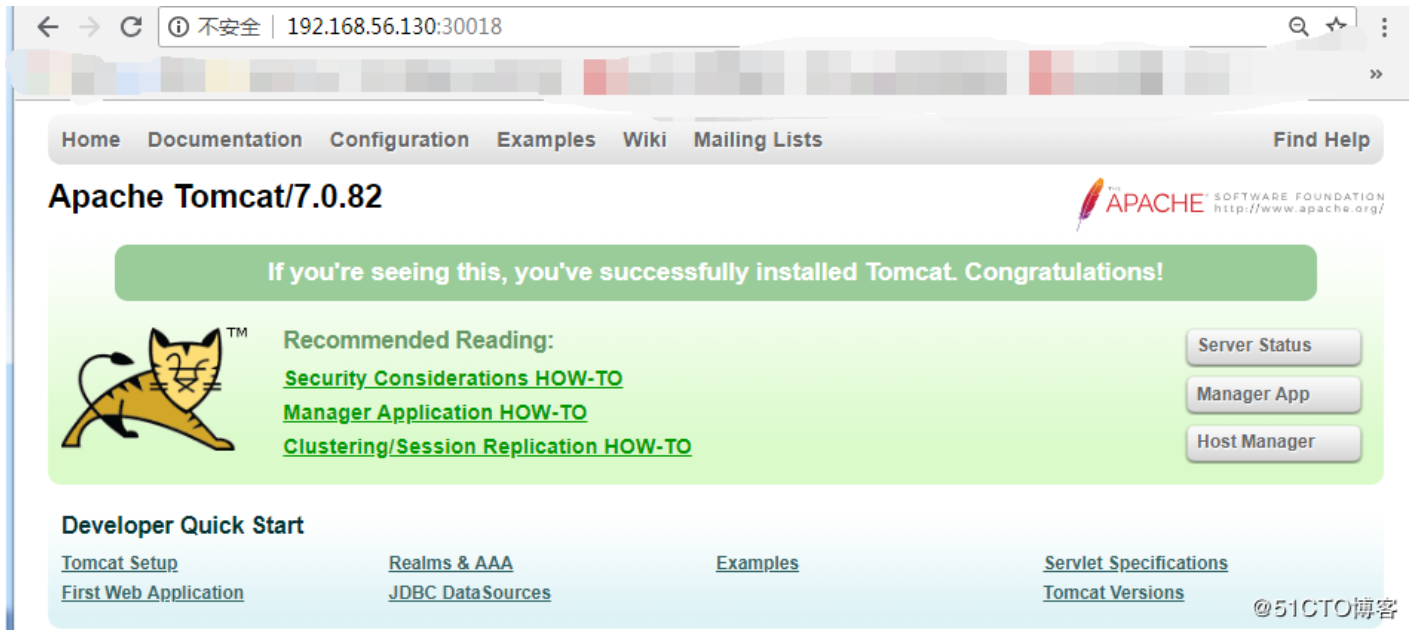
name: tomcat001

5) 如同上个步骤，上传后在dashboard的“服务”页面查看

服务			
名称	标签	集群 IP	内部入口
✓ tomcat001	-	10.43.148.6	tomcat001:8080 TCP tomcat001:30018 TCP
✓ kubernetes	component: apiserver provider: kubernetes	10.43.0.1	kubernetes:443 TCP kubernetes:0 TCP

6) 通过业务节点slave1的IP地址访问

<http://192.168.56.130:30018/>



3.5、安装kubectI工具

1) 下载kubectI工具

有两种下载方式，您可以选择其中任意一种：

(1) 在我的GitHub下载，地址是：https://github.com/zq2599/blog_demos/blob/master/k8s_tools/kubectrlinux/kubectrl.zip，在这个页面点击”download”按钮即可下载，下载后记得解压；

(2) 在linux机器上执行以下命令下载：

```
# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectrl
```

2) 设置工具

(1) kubectrl文件上传到linux机器后，授权

```
# chmod +x kubectrl
```

(2) 将kubectrl移动到可以全局访问的目录下

```
# mv ./kubectrl /usr/local/bin/
```

(3) 测试，任意目录执行一下语句

```
[root@master opt]# mv /usr/local/bin/  
[root@master opt]# kubectl  
kubectl controls the Kubernetes cluster manager.  
  
Find more information at https://github.com/kubernetes/kubernetes.  
  
Basic Commands (Beginner):  
  create      Create a resource from a file or from stdin.  
  expose       Take a replication controller, service, deployment or pod and expose it as a new  
Kubernetes Service  
  run          Run a particular image on the cluster  
  set          Set specific features on objects  
  run-container Run a particular image on the cluster. This command is deprecated, use "run"  
instead  
  
Basic Commands (Intermediate):  
  get          Display one or many resources  
  explain      Documentation of resources  
  edit         Edit a resource on the server  
  delete       Delete resources by filenames, stdin, resources and names, or by resources and  
label selector  
  
Deploy Commands:  
  rollout      Manage the rollout of a resource  
  rolling-update Perform a rolling update of the given ReplicationController  
  scale        Set a new size for a Deployment, ReplicaSet, Replication Controller, or Job  
  autoscale    Auto-scale a Deployment, ReplicaSet, or ReplicationController  
  
Cluster Management Commands:  
  certificate   Modify certificate resources.  
  cluster-info  Display cluster info  
  top           Display Resource (CPU/Memory/Storage) usage.  
  cordon        Mark node as unschedulable  
  uncordon      Mark node as schedulable  
  drain         Drain node in preparation for maintenance  
  taint         Update the taints on one or more nodes  
  
Troubleshooting and Debugging Commands:  
  describe      Show details of a specific resource or group of resources  
  logs          Print the logs for a container in a pod
```

@51CTO博客

工具已经准备好，接下来我们把配置做好，使得kubectl可以连接到K8S上执行命令；

3) 配置参数

(1) 在rancher的管理页面上，点击下图红框1中的”CLI”，在出现的页面中点击红框2中的”生成配置“：




(2) 如下图，点击红框中的”复制到剪切板“，将按钮上方的配置信息复制下来：

Kubernetes命令行

将设置保存到 `~/.kube/config:`

```
apiVersion: v1
kind: Config
clusters:
- cluster:
    api-version: v1
    insecure-skip-tls-verify: true
    server: "https://192.168.56.129:8080/r/projects/1a7/kubernetes"
    name: "master-k8s"
contexts:
- context:
    cluster: "master-k8s"
    user: "master-k8s"
    name: "master-k8s"
current-context: "master-k8s"
users:
- name: "master-k8s"
  user:
    username: "D35D32F08546D2EA1A5C"
```

```
password: "iPVVgYWo4rKKojwiiU9foyDQSTJgJgmYnZEUdLTD"
```

 复制到剪贴板

然后下载 (如果需要) 并运行 `kubect1`

@51CTO博客

(3) 创建文件，复制上面的参数

```
# mkdir ~/.kub
```

```

[root@master .kube]# cat config
apiVersion: v1
kind: Config
clusters:
- cluster:
    api-version: v1
    insecure-skip-tls-verify: true
    server: "https://192.168.56.129:8080/r/projects/1a7/kubernetes"
    name: "master-k8s"
contexts:
- context:
    cluster: "master-k8s"
    user: "master-k8s"
    name: "master-k8s"
current-context: "master-k8s"
users:
- name: "master-k8s"
  user:
    username: "D35D32F08546D2EA1A5C"
    password: "iPVVgYWo4rKKojwi iU9foydQSTjgJgmYnZEUDLTd"

```

(4) 查看进程服务

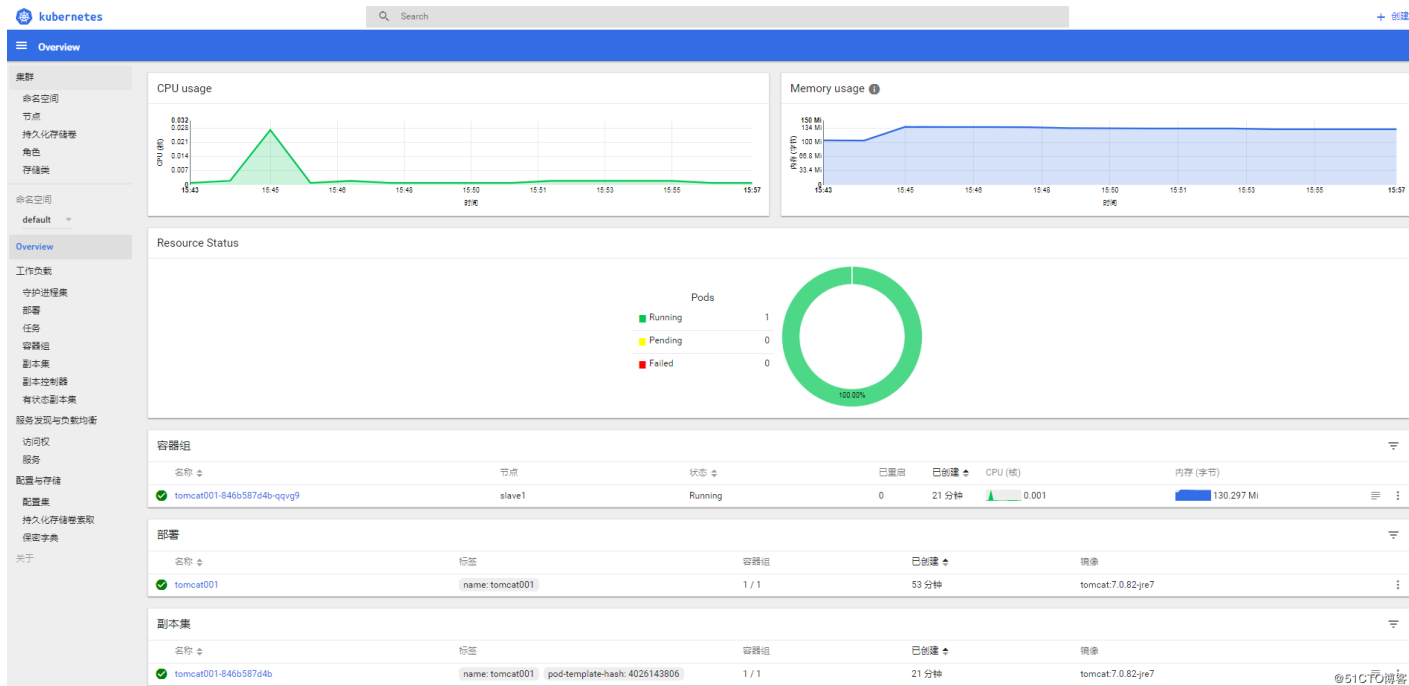
```
# kubectl get service -a -o wide --all-namespaces
```

```

[root@master .kube]# kubectl get service -a -o wide --all-namespaces

```

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
default	kubernetes	ClusterIP	10.43.0.1	<none>	443/TCP	1h	<none>
default	tomcat001	NodePort	10.43.148.6	<none>	8080:30018/TCP	13m	name=tomcat001
kube-system	heapster	ClusterIP	10.43.201.213	<none>	80/TCP	19m	k8s-app=heapster
kube-system	kube-dns	ClusterIP	10.43.0.10	<none>	53/UDP,53/TCP	19m	k8s-app=kube-dns
kube-system	kubernetes-dashboard	ClusterIP	10.43.169.2	<none>	80/TCP	19m	k8s-app=kubernetes-dashboard
kube-system	monitoring-grafana	ClusterIP	10.43.89.109	<none>	80/TCP	19m	k8s-app=grafana
kube-system	monitoring-influxdb	ClusterIP	10.43.233.26	<none>	8086/TCP	19m	k8s-app=influxdb
kube-system	tiller-deploy	ClusterIP	10.43.23.99	<none>	44134/TCP	19m	app=helm,name=tiller



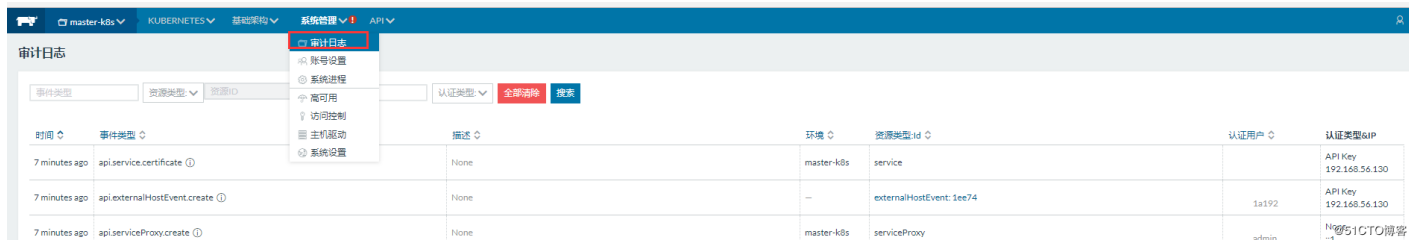
到此，部署完毕！！！！

问题来了，每次访问rancher直接进入了管理平台，一点安全性可言都没有，下来我们来设置“系统管理”

四、账号安全设置

4.1、日志审计

只有管理员用户有权限访问审计日志。审计日志在系统管理->审计日志。



The screenshot shows the Rancher UI interface for the Audit Log. The top navigation bar includes links for 'master-k8s', 'KUBERNETES', '基础架构', '系统管理', and 'API'. The '系统管理' link is highlighted, and a dropdown menu is open, showing options like '审计日志' (Audit Log), '账号设置' (Account Settings), '系统进程' (System Processes), '访问控制' (Access Control), '主机驱动' (Host Drivers), and '系统设置' (System Settings). The '审计日志' link is selected, and the page displays a table of audit events.

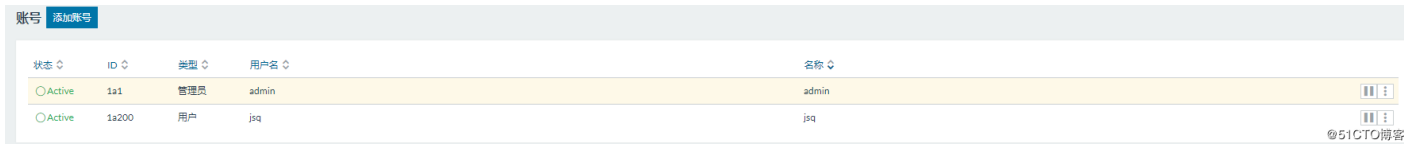
时间	事件类型	描述	环境	资源类型id	认证用户	认证类型&IP
7 minutes ago	api.service.certificate	None	master-k8s	service		API Key 192.168.56.130
7 minutes ago	api.externalHostEvent.create	None	-	externalHostEvent: 1ee74	1a192	API Key 192.168.56.130
7 minutes ago	api.serviceProxy.create	None	master-k8s	serviceProxy	admin	API Key 192.168.56.130

Rancher的审计日志是不同事件类型的集合：

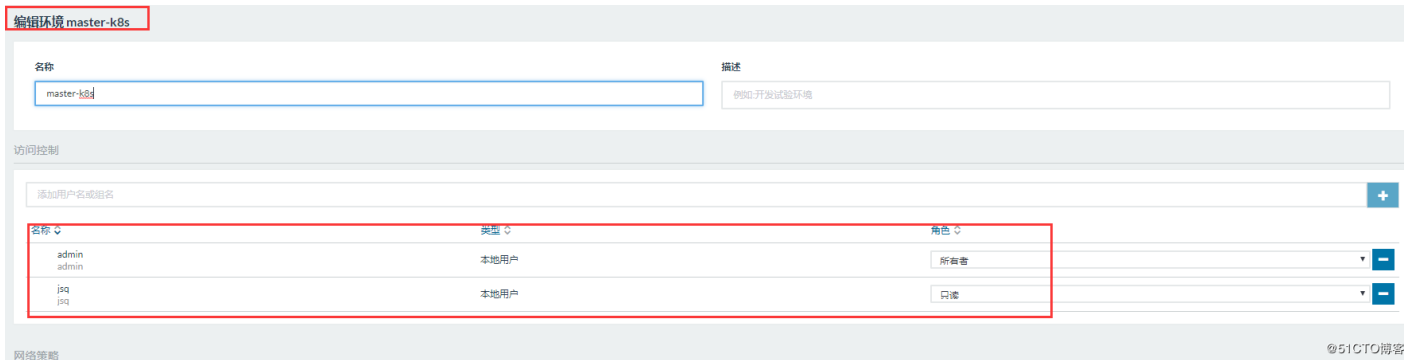
(1) 任何带有前缀api的事件是API的一次调用。事件类型将记录API操作，谁执行的操作以及API调用的方式(即通过UI，通过API密钥)。

(2) 何没有带api前缀的事件都是Rancher Server做的事情。例如，在协调服务的容器期间，在实例创建时会产生一个instance.create事件。

4.2、账号设置

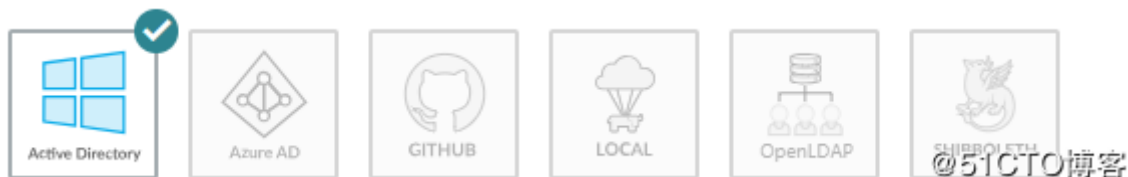


编辑“环境管理”---master-k8s



4.3、访问控制

用户在访问你的Rancher服务之前，需要进行身份认证。同时，只有拥有合法的API密钥才能使用Rancher API。



(1) 活动目录

选择活动目录图标。如果你想要通过TLS来使用活动目录，请确保你已经使用了相应的证书来启动Rancher Server。填写相关信息后，通过点击身份认证进行认证校验。当活动目录认证成功后，你将自动以已认证的用户名身份登录。并且把你的账号设置为了管理员权限。

(2) Azure AD 验证

选择Azure AD图标。填写相应信息并单击Azure认证进行认证校验。当认证成功后，你将自动以已认证的用户名身份登录。并且把你的账号设置为了管理员权限。

(3) GitHub

选择GitHub图标，并按照用户界面中的说明将Rancher注册为GitHub应用程序。点击使用GitHub进行身份认证后，当认证成功后，你将自动以已认证的Github账号登录。并且把你的账号设置为了管

理员权限。

(4) local

选择本地图标。通过提供登录用户名，全名和密码来创建管理员用户。点击启用本地认证来启用本地身份认证。通过单击此按钮，管理员用户将被创建并保存在数据库中。这时你将自动用刚刚创建的管理员帐户登录到Rancher服务。

1. 设置管理员用户

该账户将成为管理员并拥有对 Rancher 的完全控制权限

登录用户名*

admin

全名

例如: John Smith

密码*

.....

确认密码*

.....

启用访问控制

点击以开启访问控制并登陆

启用本地验证

@51CTO博客

(5)OpenLDAP

填写对应信息后，通过点击身份认证进行认证校验。当OpenLDAP认证成功后，你将自动以已认证的用户名身份登录。并且把你的账号设置为了管理员权限。

(6)Shibboleth

选择Shibboleth图标。填写Shibboleth帐户的配置信息，点击保存保存信息，然后点击测试来测试访问控制是否正常工作。

在使用Shibboleth时，你应该注意一些已知的问题：

(1) 不支持搜索或查找功能。在添加用户时，请确保输入的用户ID是准确的，这样才能保证用户被添加成功。

(2) 当添加用户到一个环境时，不支持组ID，除非管理员是该组的成员之一。

功能模块比较复杂，后续补充。。。。。