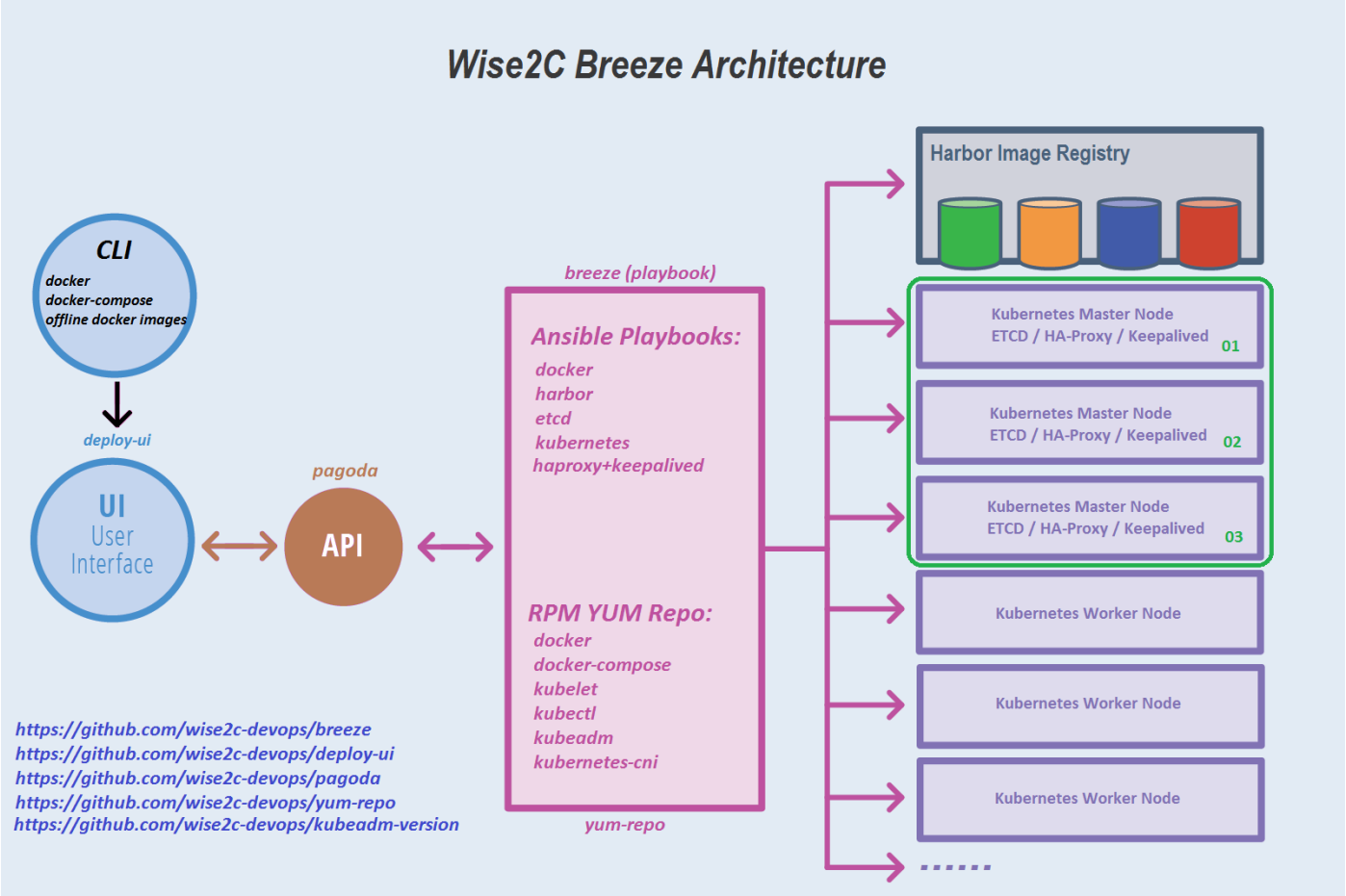
使用睿云智合开源 Breeze 工具部署 Kubernetes 高可用集群

Breeze 项目是深圳睿云智合所开源的 Kubernetes 图形化部署工具，大大简化了 Kubernetes 部署的步骤，其最大亮点在于支持全离线环境的部署，且不需要翻墙获取 Google 的相应资源包，尤其适合某些不便访问互联网的服务器场景。（项目地址 https://github.com/wise2c-devops/breeze ）

Breeze 开源工具由以下子项目构成：

1. **playbook（breeze）** 该项目由不同的 ansible playbook 构成，分别是 docker、etcd、registry、loadbalance、kubernetes
2. **yum-repo** 该项目用于为安装过程中提供离线的 yum repo 源，包含了 docker、kubelet、kubectl、kubeadm、kubernetes-cni、docker-compose 等 rpm 包库，除此之外我们还包括了可能会用到的 ceph 及 nfs 相关 rpm
3. **deploy-ui** 用户前端 UI，采用 vue.js 框架实现
4. **pagoda** 实现了对 ansible 脚本调用的 API 集
5. **kubeadm-version** 输出 kubernetes 组件镜像版本信息

Breeze 软件架构简图：



用户通过 Breeze 工具，只需要一台安装有 Docker 及 docker-compose 命令的服务器，连接互联网下载一个对应 Kubernetes 版本的 docker-compose.yaml 文件即可将部署程序运行出来，对部署机而已，只需能有普通访问互联网的能力即可，无需翻墙，因为我们已经将所有 Kubernetes 所需要的 docker 镜像以及 rpm 包内置于 docker image 里了。

如果需要离线安装，也是极其容易的，只需要将 docker-compose.yaml 文件里涉及的 docker 镜像保存下来，到了无网环境预先使用 docker load 命令载入，再运行 docker-compose up -d 命令即可无网运行部署程序。所有被部署的集群角色服务器，完全无需连入互联网。

该项目开源，用户可以很方便的 fork 到自己的 git 账号结合 travis 自动构建出任意 Kubernetes 版本的安装工具。

在我们的实验环境中准备了六台服务器，配置与角色如下（如果需要增加 Minion/Worker 节点请自行准备即可）：

主机名	IP 地址	角色	OS	组件
deploy	192.168.9.10	Breeze Deploy	CentOS 7.6 x64	docker / docker-compose / Breeze
k8s01	192.168.9.11	K8S Master	CentOS 7.6 x64	K8S Master / etcd / HAProxy / Keepalived
k8s02	192.168.9.12	K8S Master	CentOS 7.6 x64	K8S Master / etcd / HAProxy / Keepalived
k8s03	192.168.9.13	K8S Master	CentOS 7.6 x64	K8S Master / etcd / HAProxy / Keepalived
k8s04	192.168.9.14	K8S Minion Node	CentOS 7.6 x64	K8S Worker
registry	192.168.9.20	Harbor	CentOS 7.6 x64	Harbor 1.6.2
	192.168.9.30	VIP		HA 虚 IP 地址在 3 台 K8S Master 浮动

步骤：

一、准备部署主机（deploy / 192.168.9.10）

（1）以标准 Minimal 方式安装 CentOS 7.6 (1810) x64 之后(7.4 和 7.5 也支持)，登录 shell 环境，执行以下命令开放防火墙：

```
setenforce 0
sed --follow-symlinks -i "s/SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config
firewall-cmd --set-default-zone=trusted
firewall-cmd --complete-reload
```

（2）安装 docker-compose 命令

```
curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose
```

（3）安装 docker

```
yum install docker
yum enable docker && yum start docker
```

（4）建立部署主机到其它所有服务器的 ssh 免密登录途径

a）生成秘钥，执行：

```
ssh-keygen
```

b）针对目标服务器做 ssh 免密登录，依次执行：

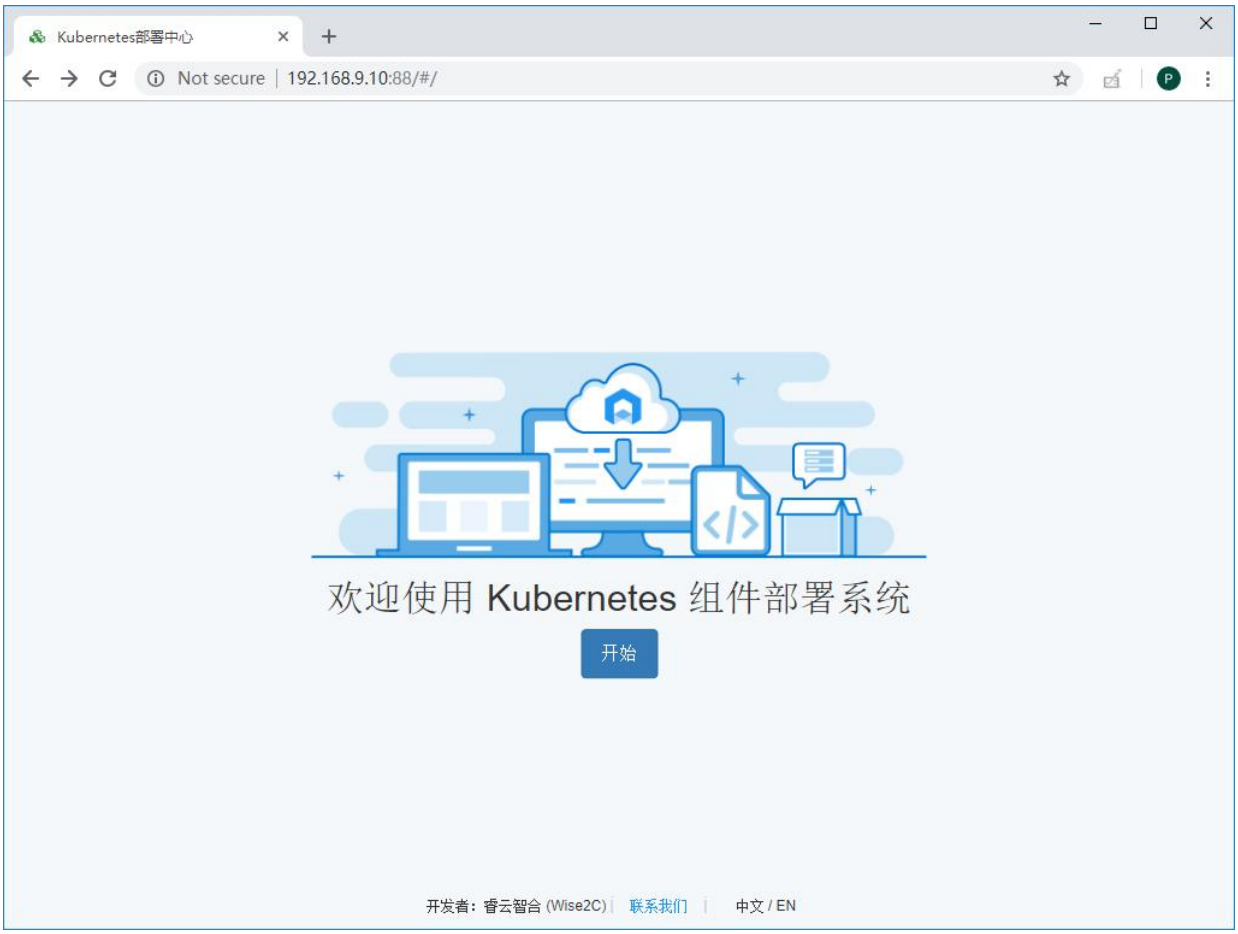
```
ssh-copy-id 192.168.9.11
ssh-copy-id 192.168.9.12
ssh-copy-id 192.168.9.13
ssh-copy-id 192.168.9.14
ssh-copy-id 192.168.9.20
```

二、获取针对 K8S 某个具体版本的 Breeze 资源文件并启动部署工具，例如此次实验针对刚刚发布的 K8S v1.13.1

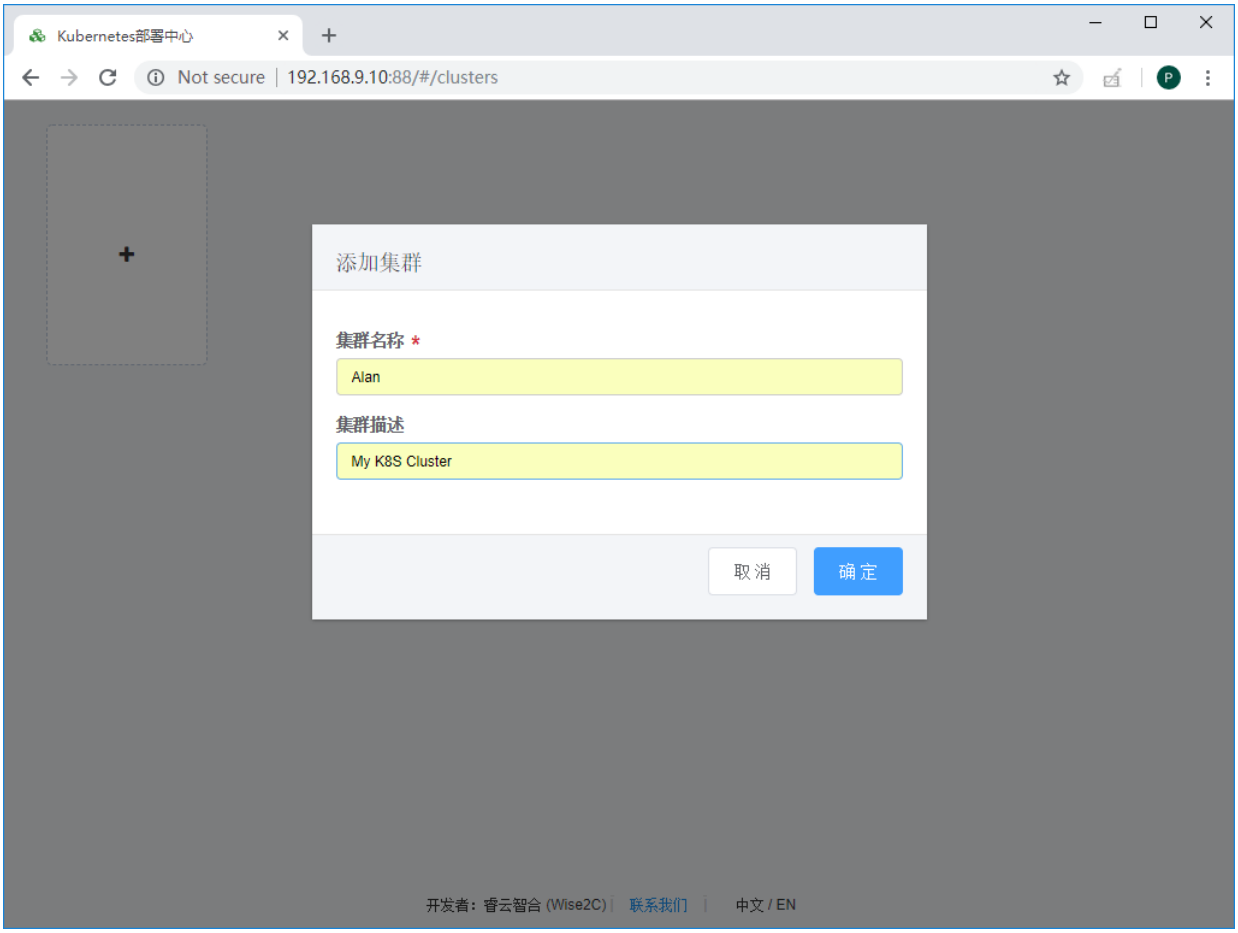
```
curl -L https://raw.githubusercontent.com/wise2c-devops/breeze/v1.13.1/docker-compose.yml -o docker-compose.yml
docker-compose up -d
```

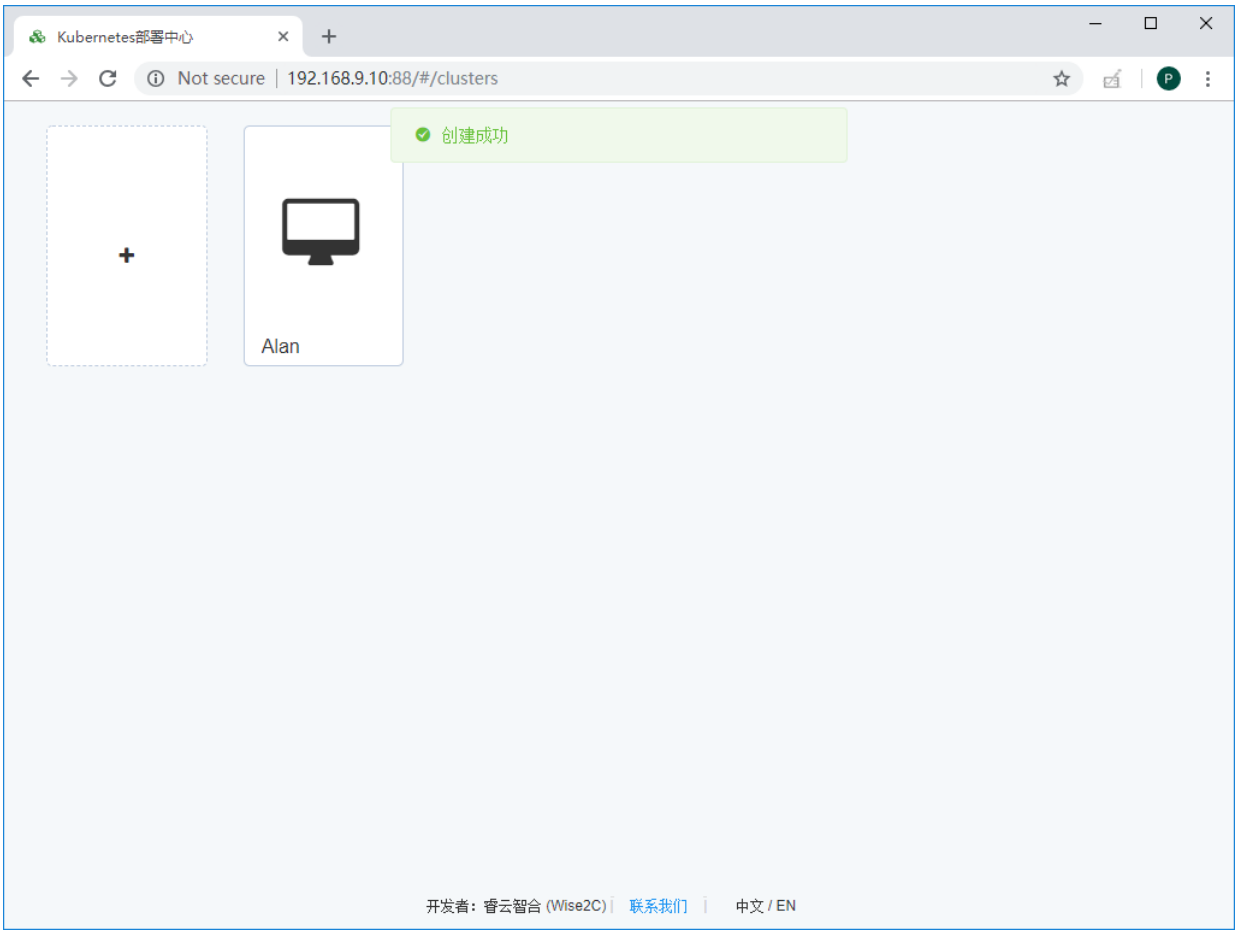
三、访问部署工具的浏览器页面(部署机 IP 及端口 88)，开始部署工作

http://192.168.9.10:88

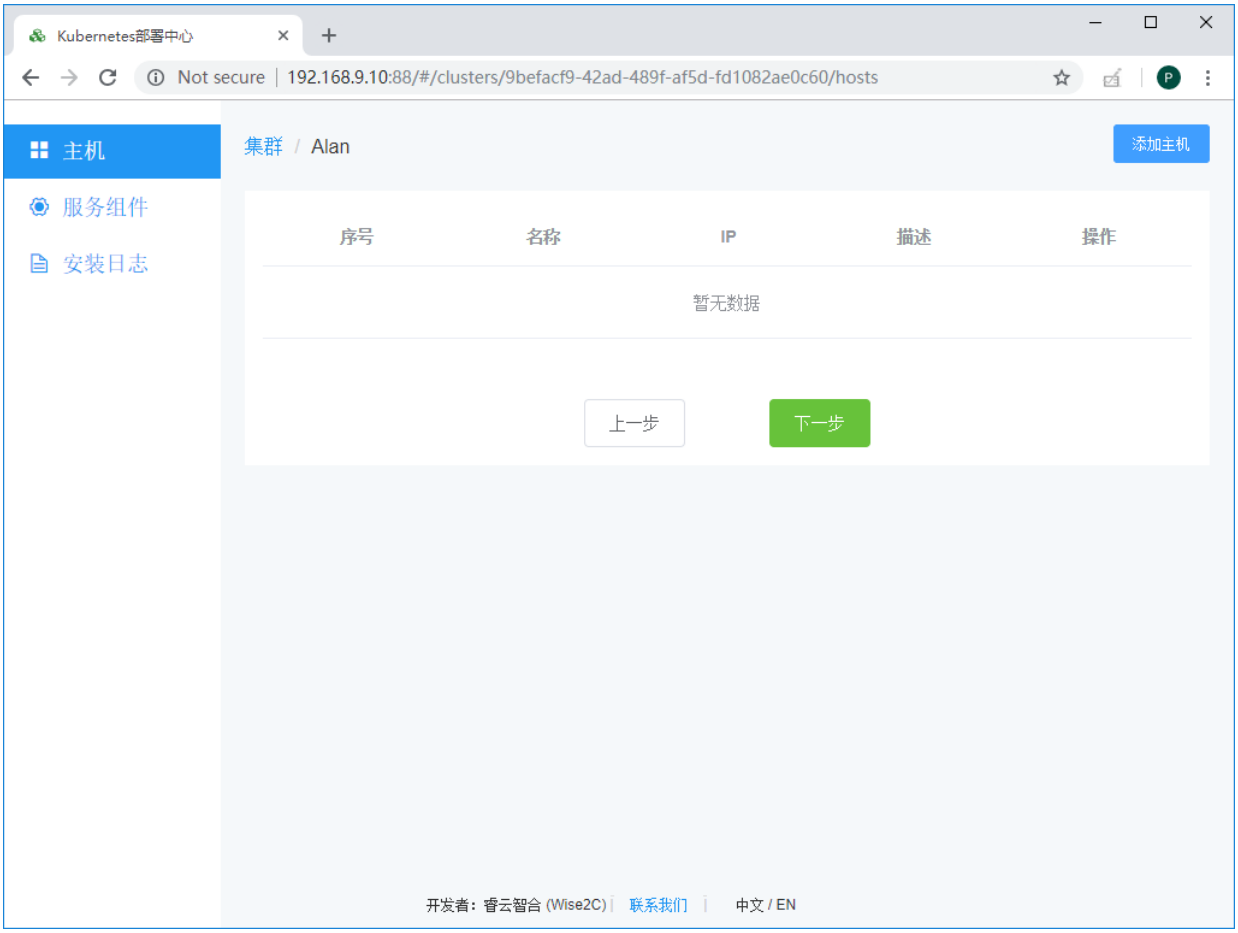


（1）点击开始按钮后，点击+图标开始添加一个集群：

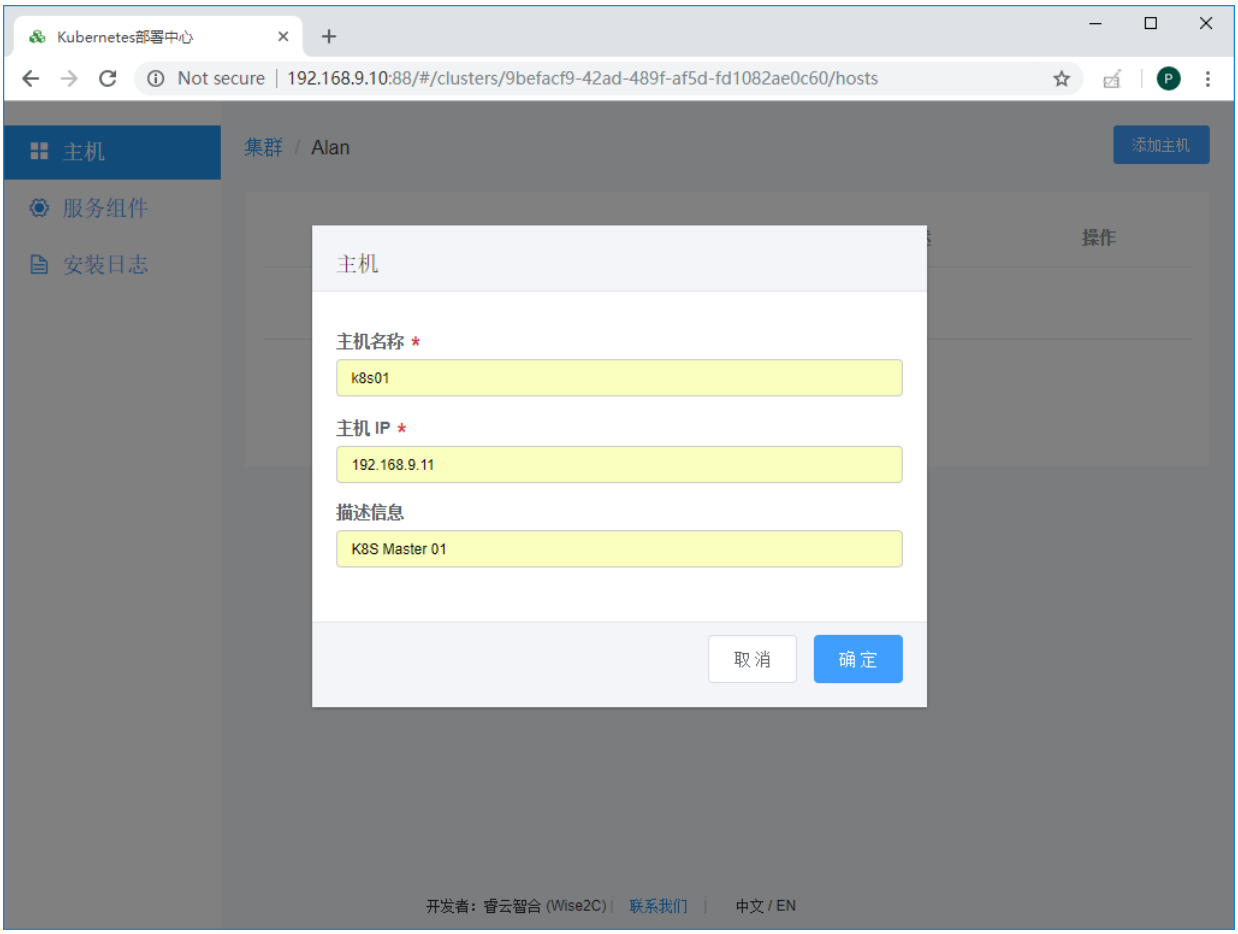




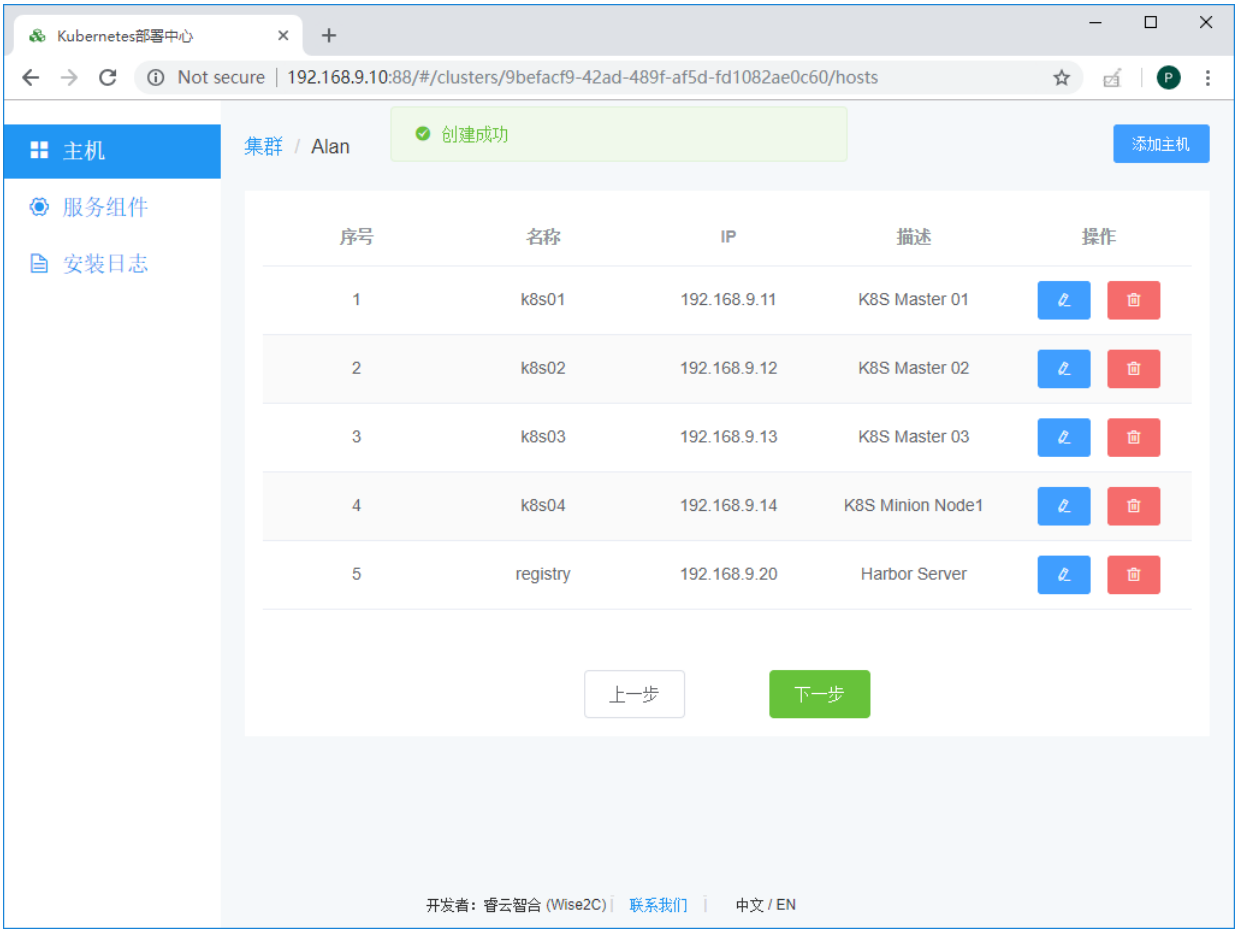
（2）点击该集群图标进入添加主机界面：



点击右上角“添加主机按钮”：

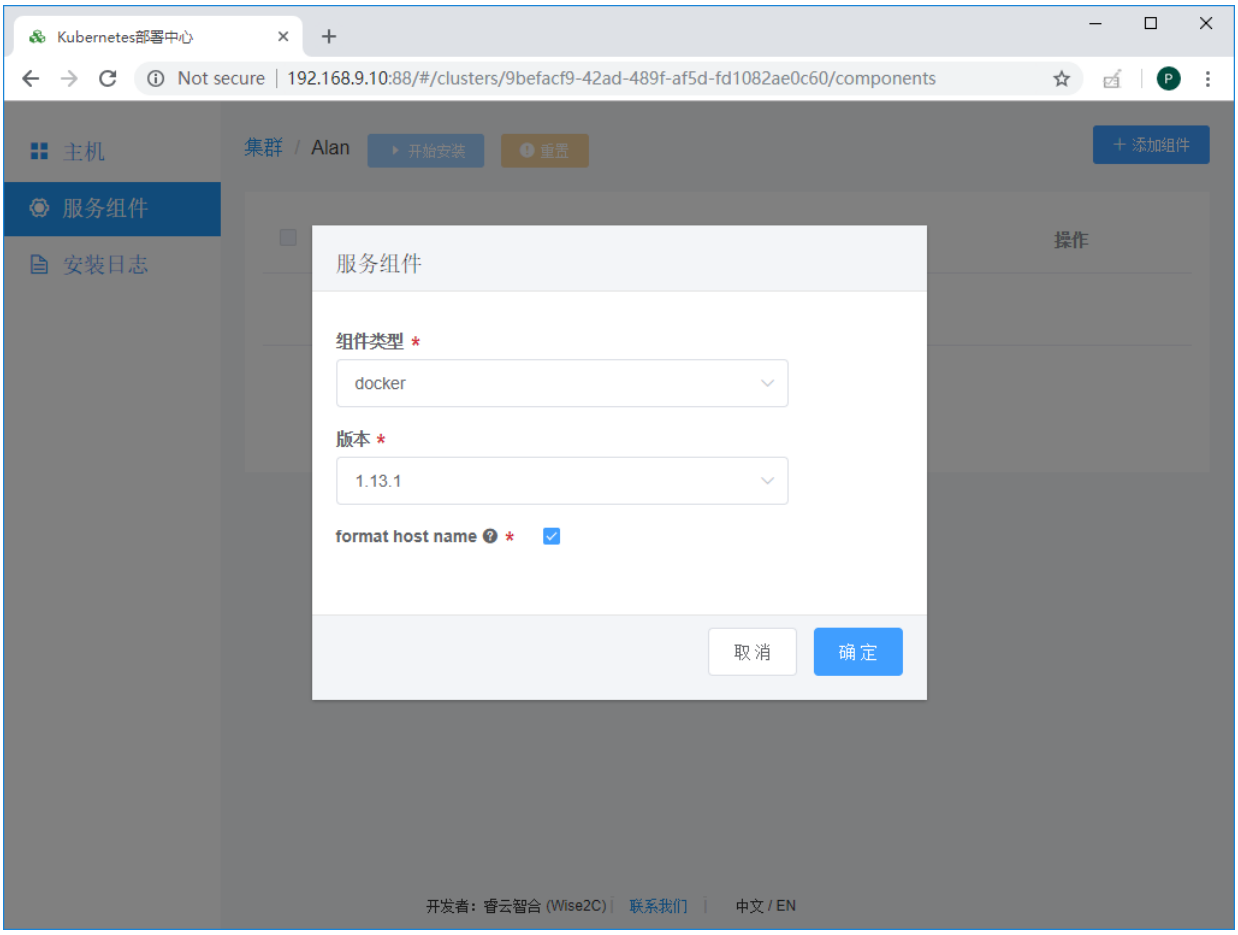


反复依次添加完整个集群的 5 台服务器：

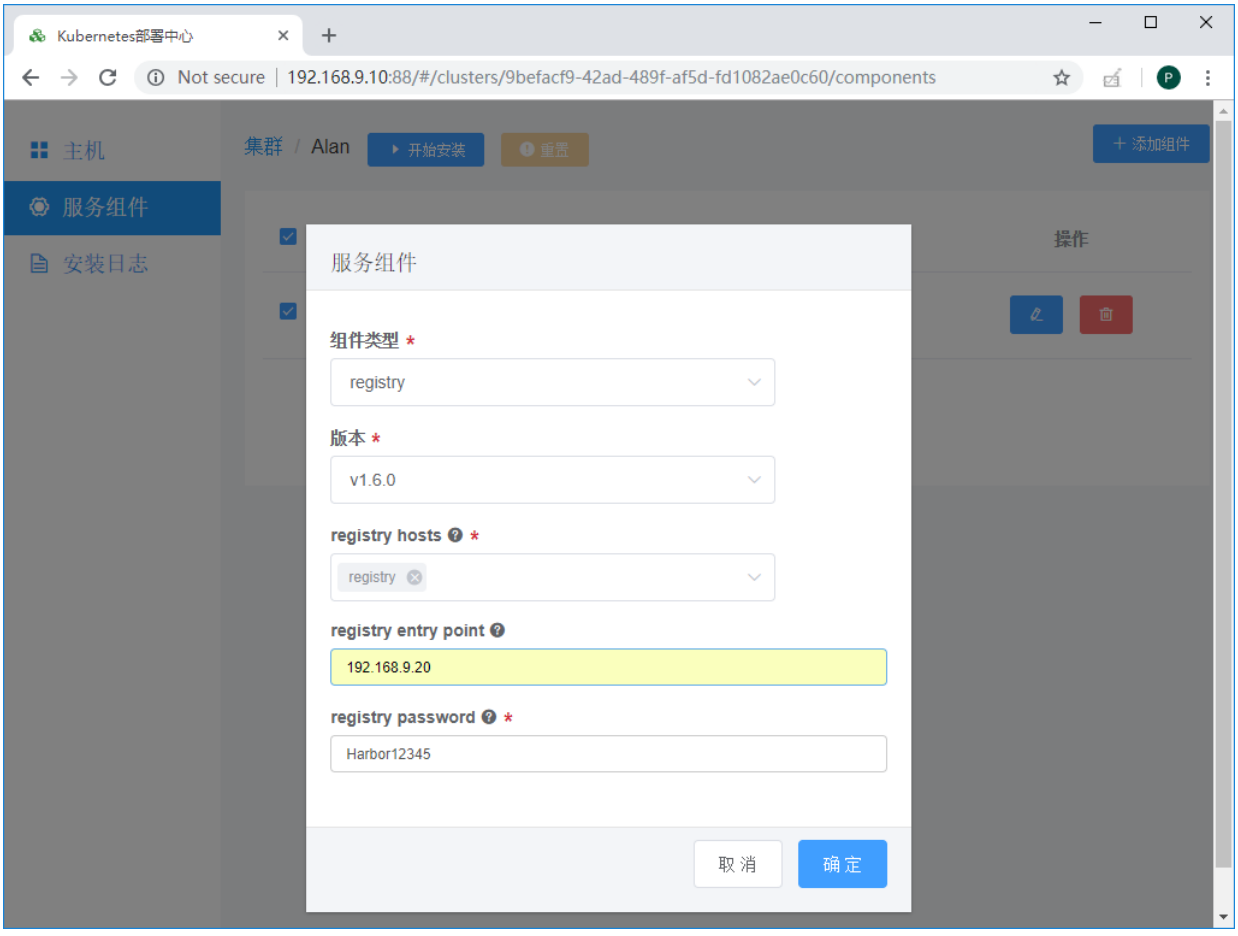


点击下一步进行服务组件定义

（3）点击右上角“添加组件”按钮添加服务组件，选择 docker，因为所有主机都需要安装，因此无需选择服务器：

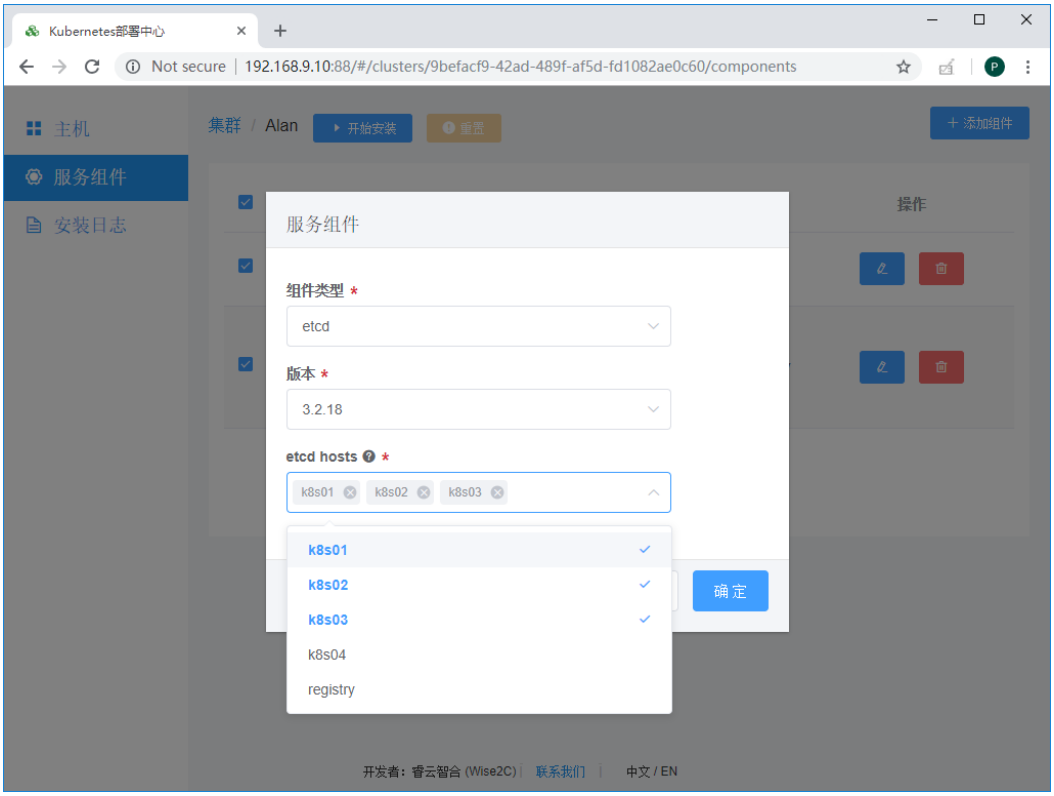


再添加镜像仓库组件



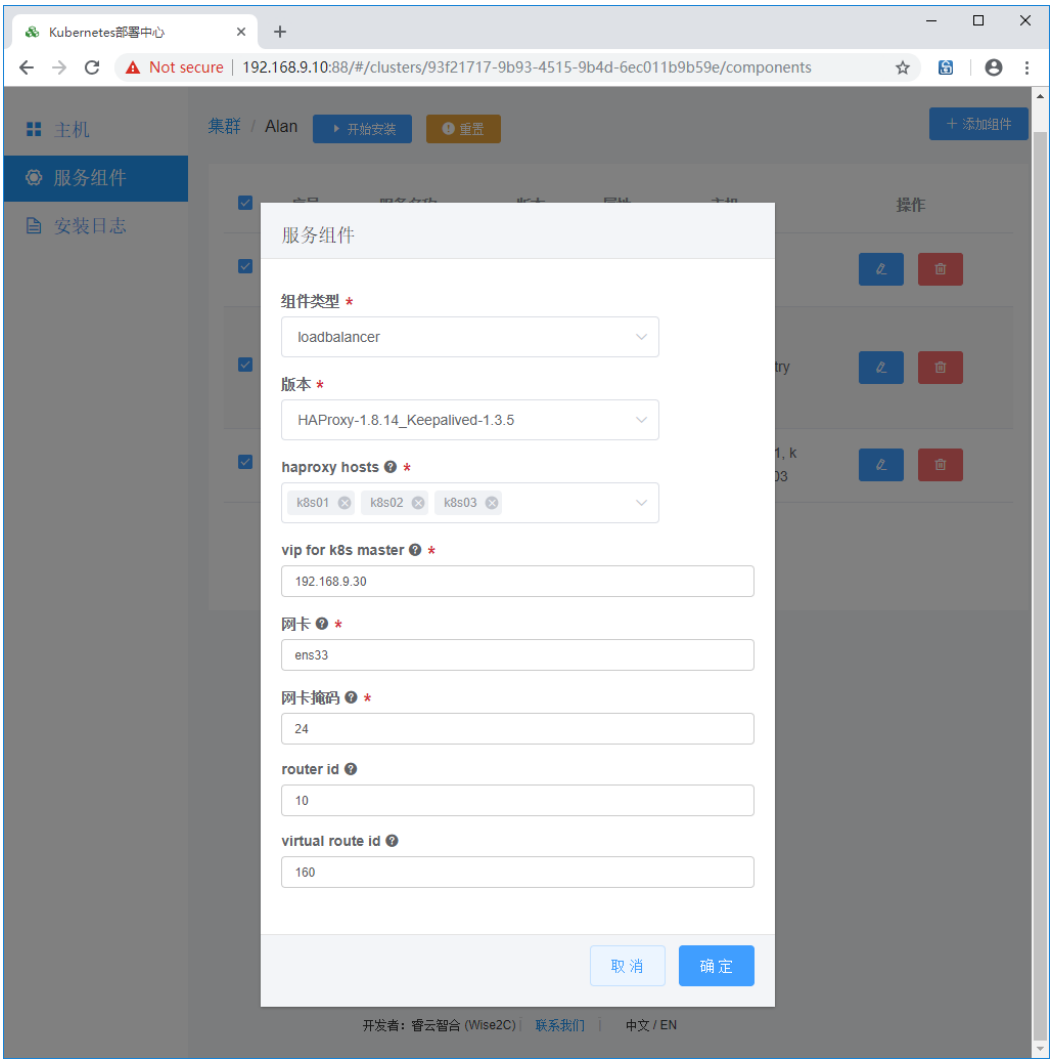
备注：registry entry point 默认就填写 Harbor 服务器的 IP 地址，有些环节可能使用域名则填写域名

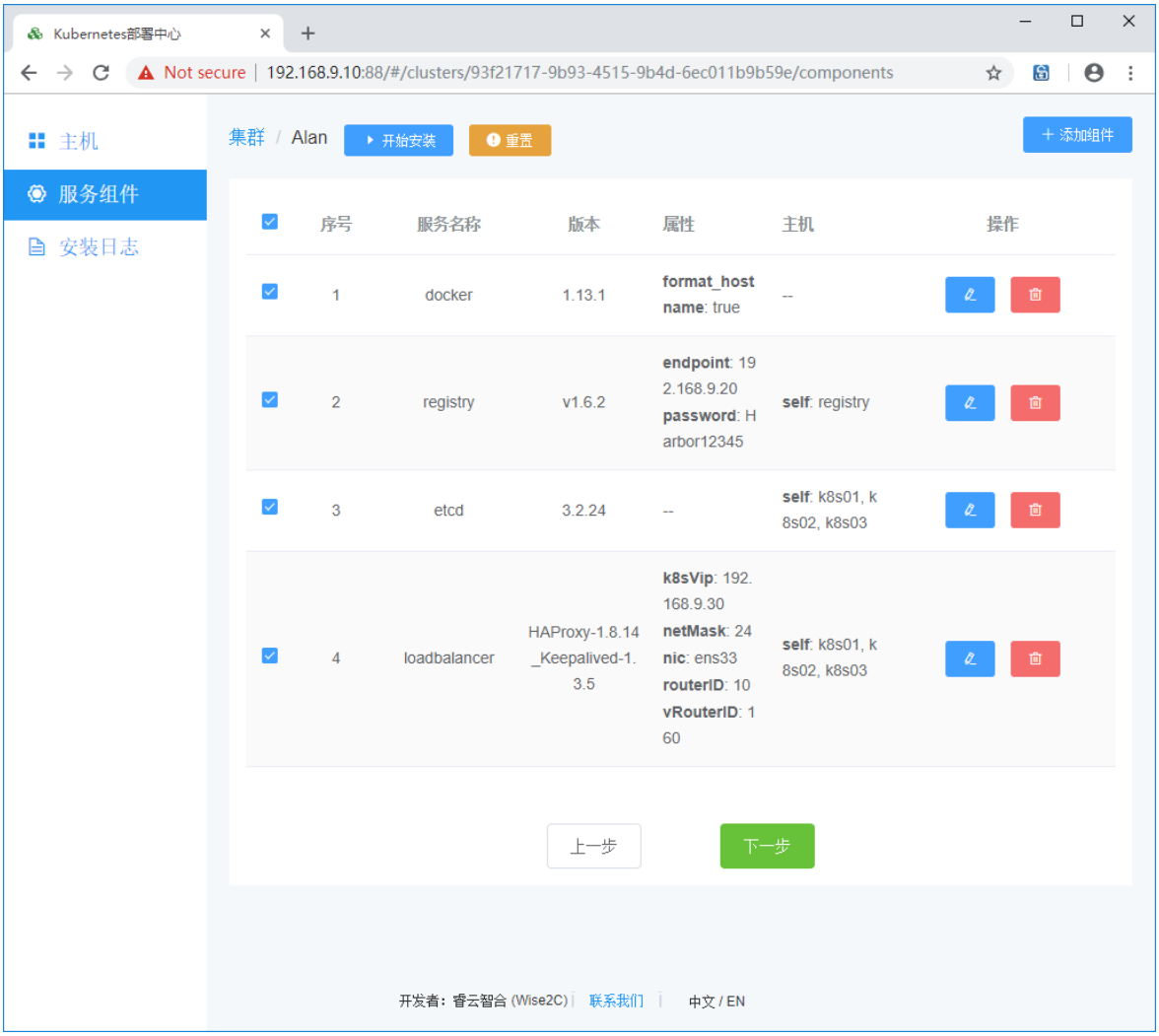
继续添加 etcd 组件，这里我们将其合并部署于 k8s master 节点，也可以挑选单独的主机进行部署：



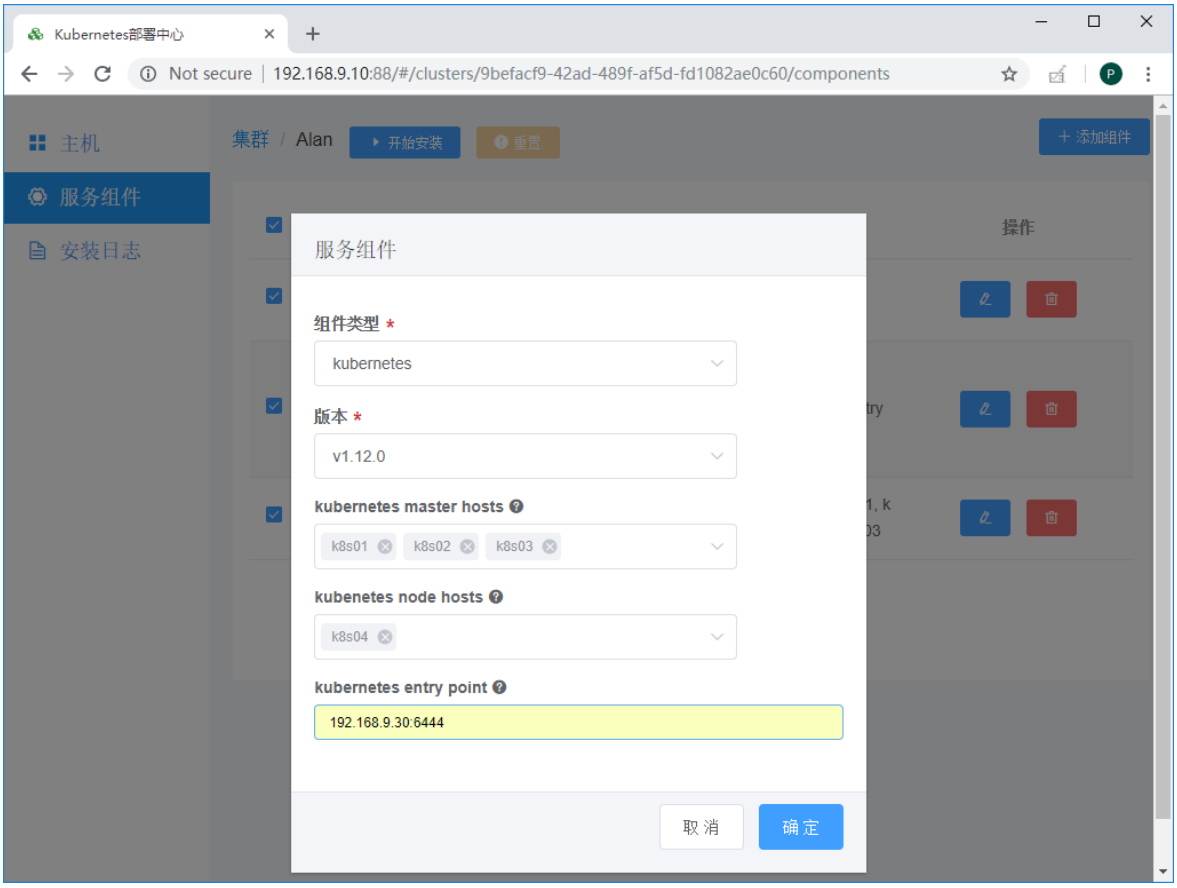
接下来是设置高可用组件（haproxy+keepalived）：

vip for k8s master 是指三个 k8s master 服务器的高可用虚拟浮动 IP 地址；网卡请填写实际操作系统下的网卡名，注意请保证 3 个节点网卡名一致；router id 和 virtual router id 请确保不同 k8s 集群使用不同的值。



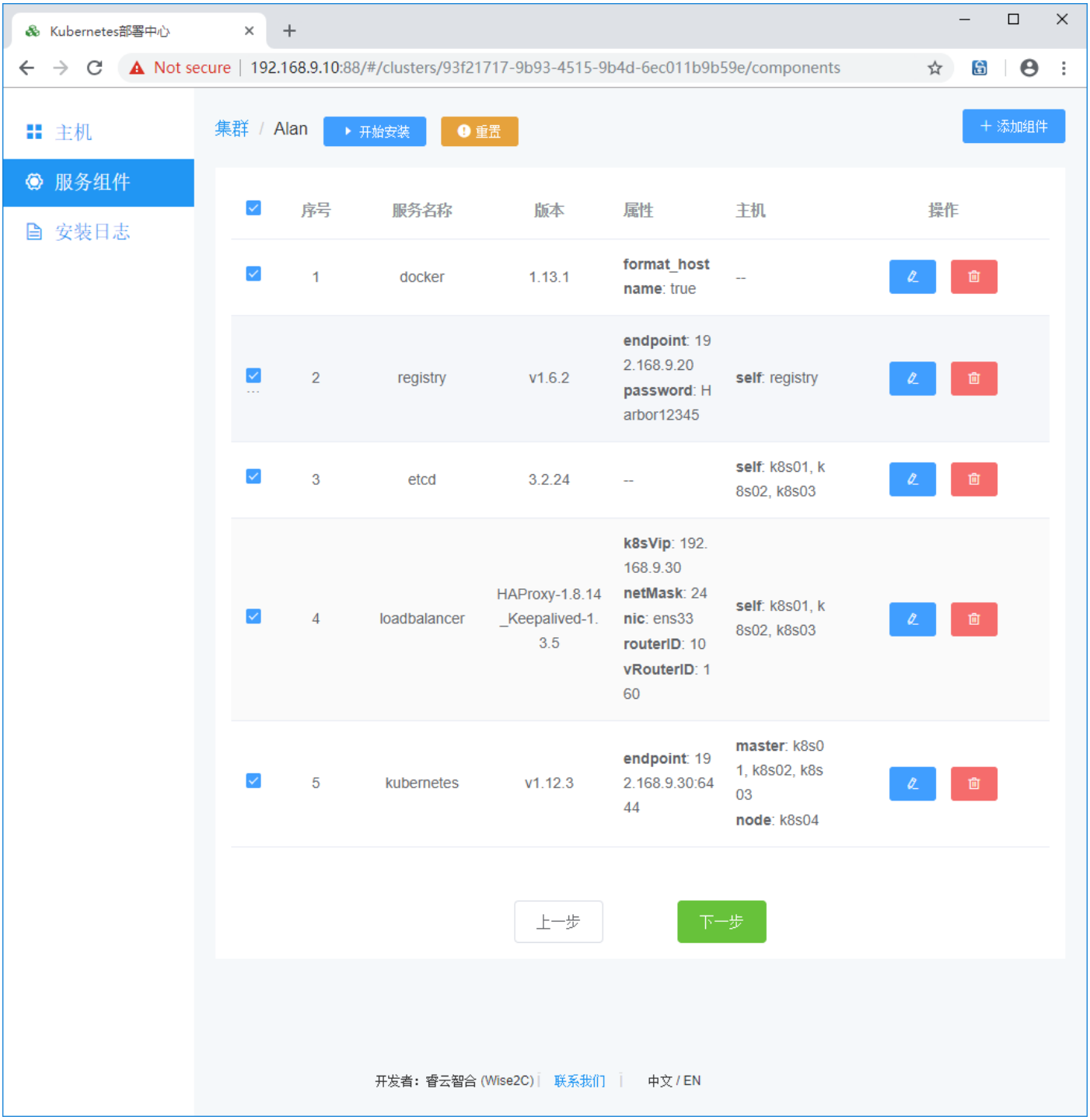


最后添加 k8s 组件，这里分为 master 和 minion nodes：

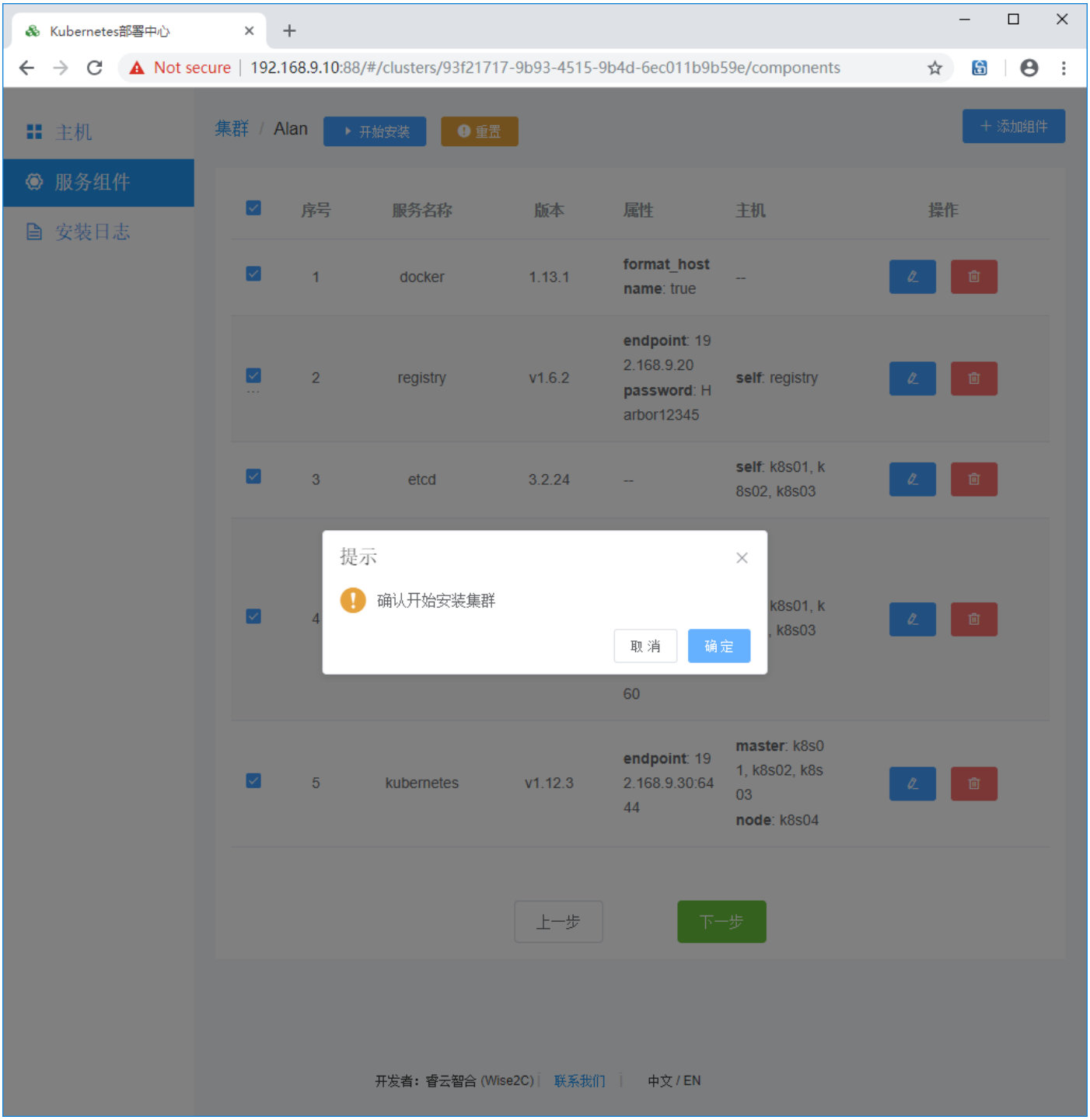


备注：这里 kubernetes entry point 是为了 HA 场景，比如此次试验我们在每一个 k8s master 节点同时各部署了 haproxy 和 keepalived 组件，其产生的虚 IP 是 192.168.9.30，端口是 6444，那么我们在这里应该填写为 192.168.9.30:6444，如果您只安装一个 master，那么可以填写为 master 的入口，例如 192.168.9.11:6443

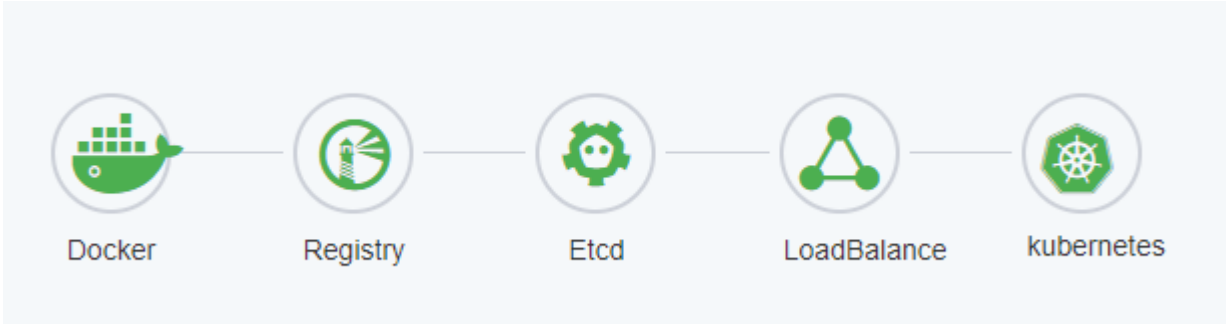
设置完成的界面如下：



四、点击下一步，执行部署流程：



在接下来的部署过程中，屏幕会有日志及图标颜色的动态变化，耐心等待最后部署界面所有组件颜色变为绿色即可结束 K8S 高可用集群的部署工作。



验证：

```
[root@k8s01 ~]# kubectl get cs
NAME                 STATUS    MESSAGE             ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health": "true"}
etcd-1               Healthy   {"health": "true"}
etcd-2               Healthy   {"health": "true"}
[root@k8s01 ~]#
[root@k8s01 ~]# kubectl -n kube-system get pods
NAME                                    READY   STATUS    RESTARTS   AGE
coredns-64fcf865cf-gnvtt                1/1     Running   2          47h
coredns-64fcf865cf-x9r7q                1/1     Running   2          47h
heapster-7c4668c895-rxcwp               1/1     Running   2          47h
kube-apiserver-k8s01                    1/1     Running   2          47h
kube-apiserver-k8s02                    1/1     Running   2          47h
kube-apiserver-k8s03                    1/1     Running   2          47h
kube-controller-manager-k8s01           1/1     Running   2          47h
kube-controller-manager-k8s02           1/1     Running   2          47h
kube-controller-manager-k8s03           1/1     Running   2          47h
kube-flannel-ds-2vn5m                   1/1     Running   2          47h
kube-flannel-ds-k5wt6                   1/1     Running   2          47h
kube-flannel-ds-mzbmx                   1/1     Running   3          47h
kube-flannel-ds-wzdt7                   1/1     Running   3          47h
kube-proxy-c6jwq                        1/1     Running   2          47h
kube-proxy-fslhg                        1/1     Running   2          47h
kube-proxy-rtsmz                        1/1     Running   2          47h
kube-proxy-xbcd9                        1/1     Running   2          47h
kube-scheduler-k8s01                    1/1     Running   2          47h
kube-scheduler-k8s02                    1/1     Running   2          47h
kube-scheduler-k8s03                    1/1     Running   2          47h
kubernetes-dashboard-5db77f9dfb-swspm   1/1     Running   2          47h
monitoring-grafana-5565445645-c66p8     1/1     Running   2          47h
monitoring-influxdb-7bd68f7d84-52wjr    1/1     Running   2          47h
[root@k8s01 ~]#
[root@k8s01 ~]# kubectl get nodes -o wide
NAME    STATUS   ROLES    AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION          CONTAINER-RUNTIME
k8s01   Ready    master   47h   v1.12.3   192.168.9.11   <none>        CentOS Linux 7 (Core)   3.10.0-862.el7.x86_64   docker://1.13.1
k8s02   Ready    master   47h   v1.12.3   192.168.9.12   <none>        CentOS Linux 7 (Core)   3.10.0-862.el7.x86_64   docker://1.13.1
k8s03   Ready    master   47h   v1.12.3   192.168.9.13   <none>        CentOS Linux 7 (Core)   3.10.0-862.el7.x86_64   docker://1.13.1
k8s04   Ready    <none>   47h   v1.12.3   192.168.9.14   <none>        CentOS Linux 7 (Core)   3.10.0-862.el7.x86_64   docker://1.13.1
[root@k8s01 ~]#
```

Alan Peng

2018 年 12 月 4 日