

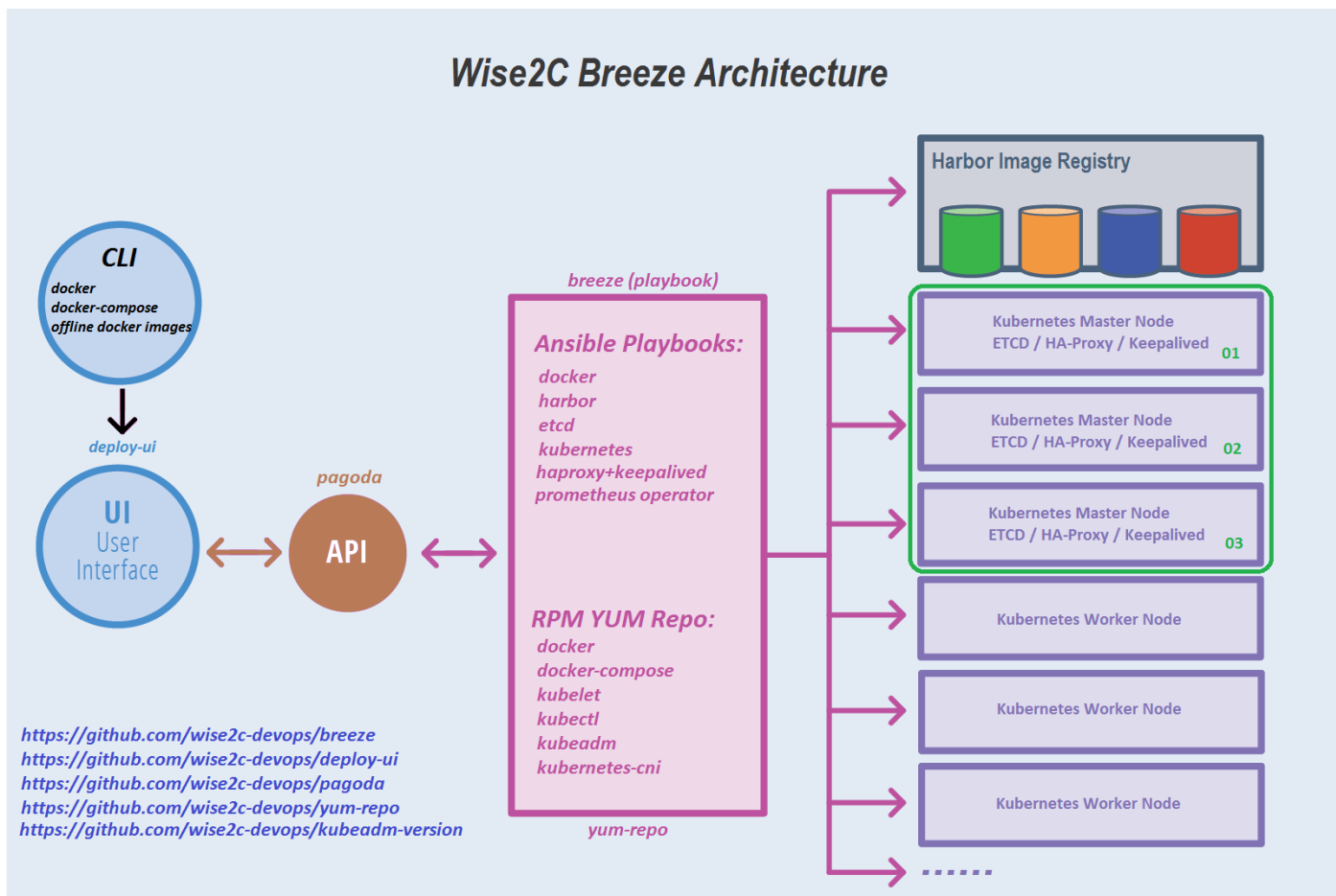
## 使用睿云智合开源 Breeze 工具部署 Kubernetes 高可用集群

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、kubectrl、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
master01	192.168.9.11	K8S Master Node	CentOS 7.6 x64	K8S Master / etcd / HAProxy / Keepalived
master02	192.168.9.12	K8S Master Node	CentOS 7.6 x64	K8S Master / etcd / HAProxy / Keepalived
master03	192.168.9.13	K8S Master Node	CentOS 7.6 x64	K8S Master / etcd / HAProxy / Keepalived
worker01	192.168.9.21	K8S Worker Node	CentOS 7.6 x64	K8S Worker / Prometheus
harbor	192.168.9.20	Harbor	CentOS 7.6 x64	Harbor 1.7.0
	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
systemctl enable docker && systemctl 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) 生成密钥，执行：

```
ssh-keygen
```

b) 针对目标服务器做 ssh 免密登录，依次执行：

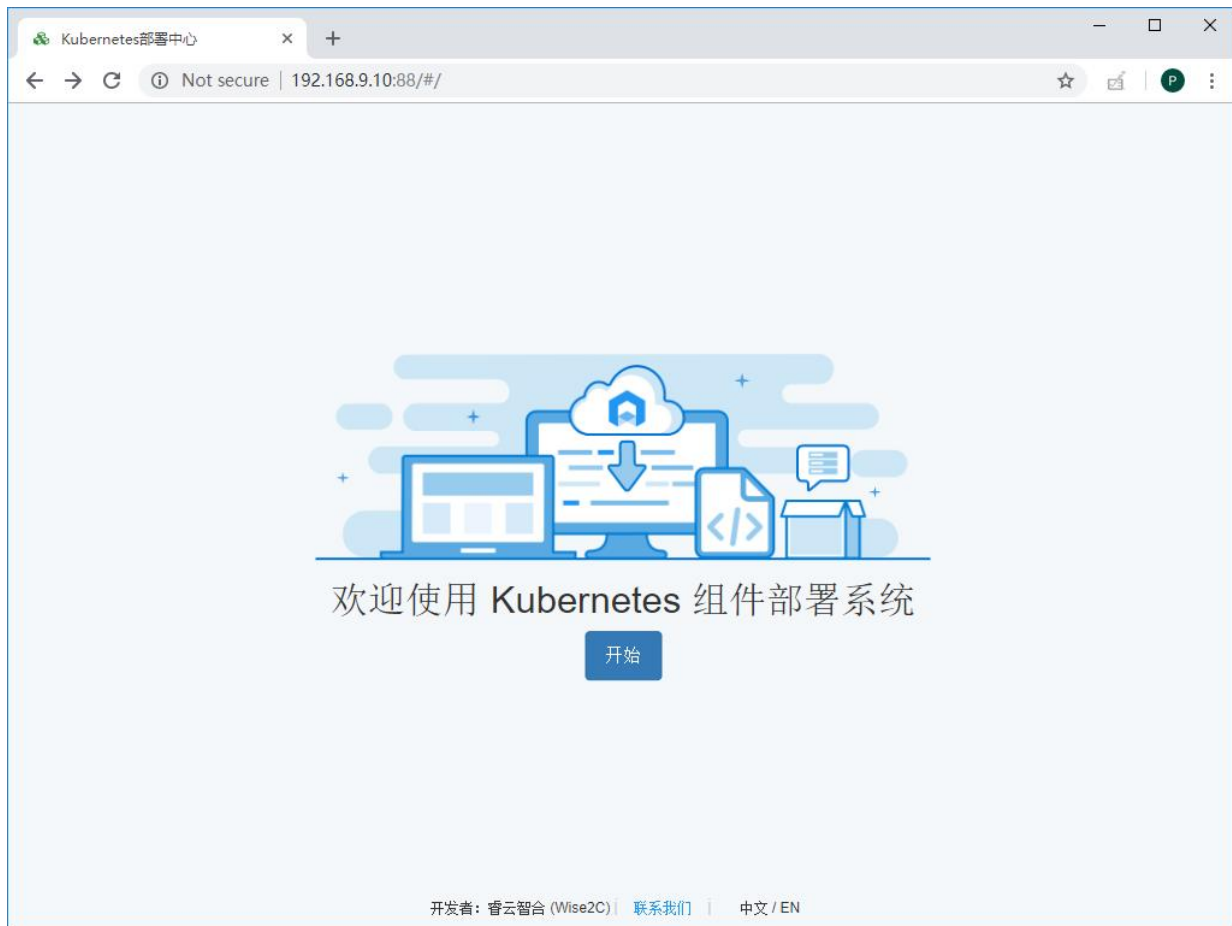
```
ssh-copy-id 192.168.9.11
ssh-copy-id 192.168.9.12
ssh-copy-id 192.168.9.13
ssh-copy-id 192.168.9.20
ssh-copy-id 192.168.9.21
```

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

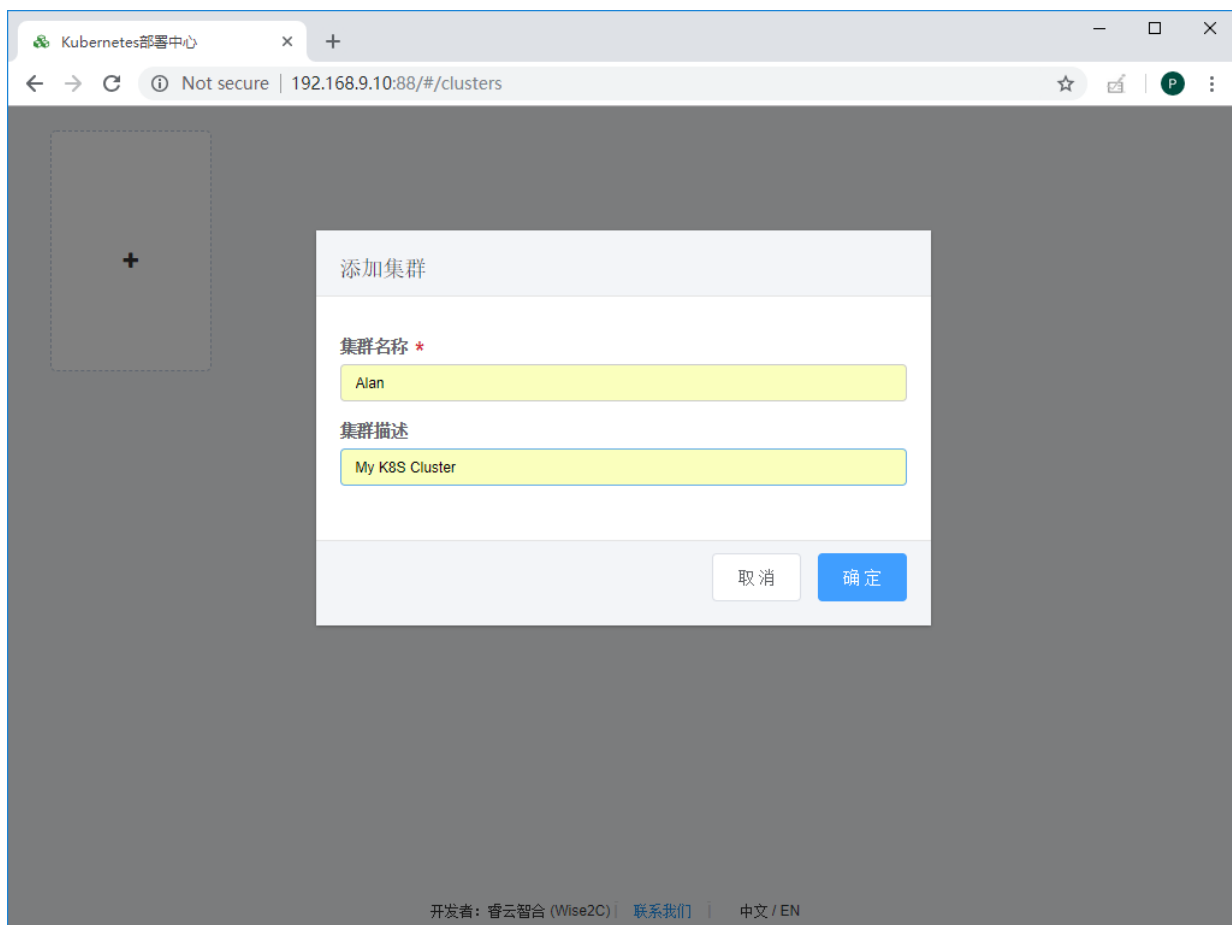
```
curl -L https://raw.githubusercontent.com/wise2c-devops/breeze/v1.13.1/docker-compose.yml -
o docker-compose.yml
docker-compose up -d
```

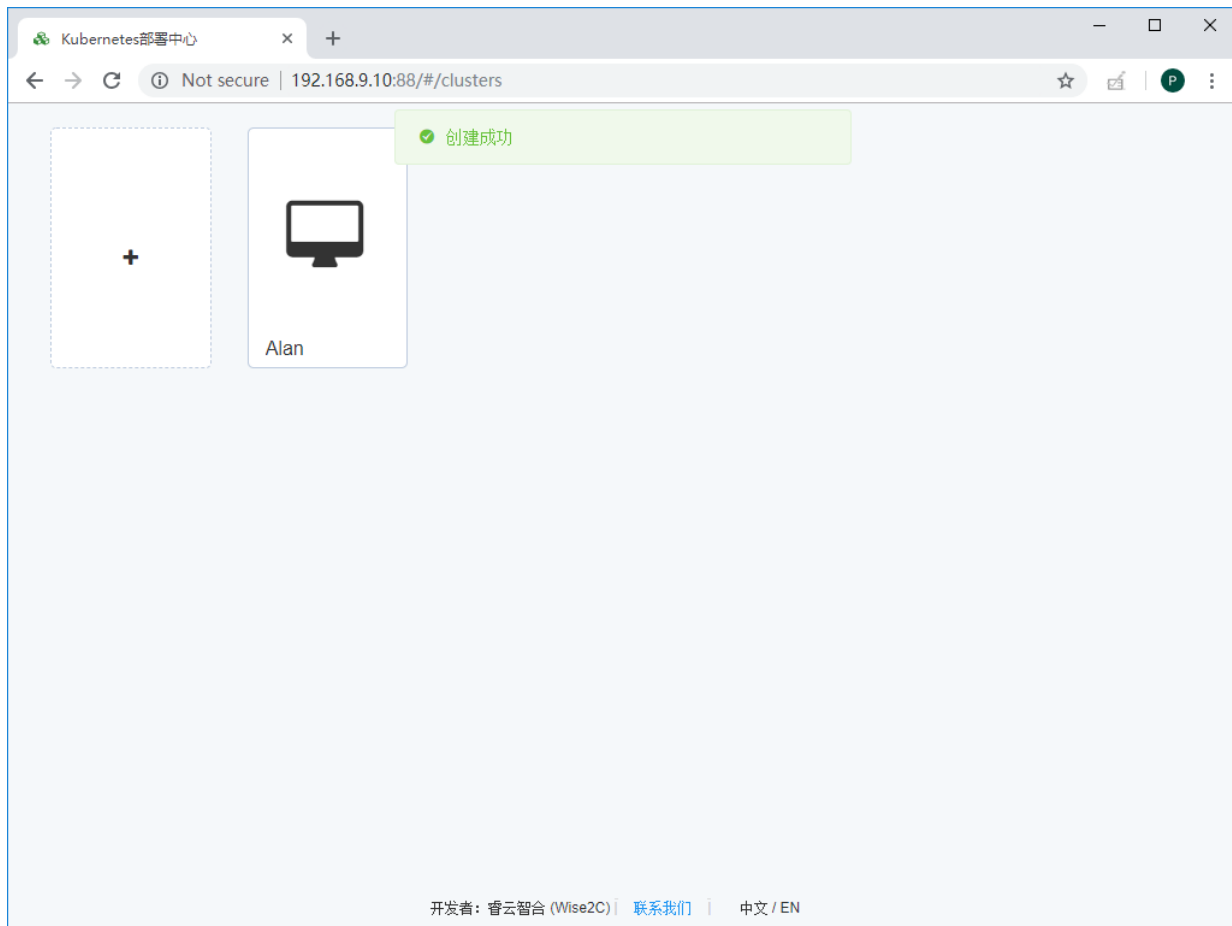
三、访问部署工具的浏览器页面(部署机 IP 及端口 88)，开始部署工作

<http://192.168.9.10:88>

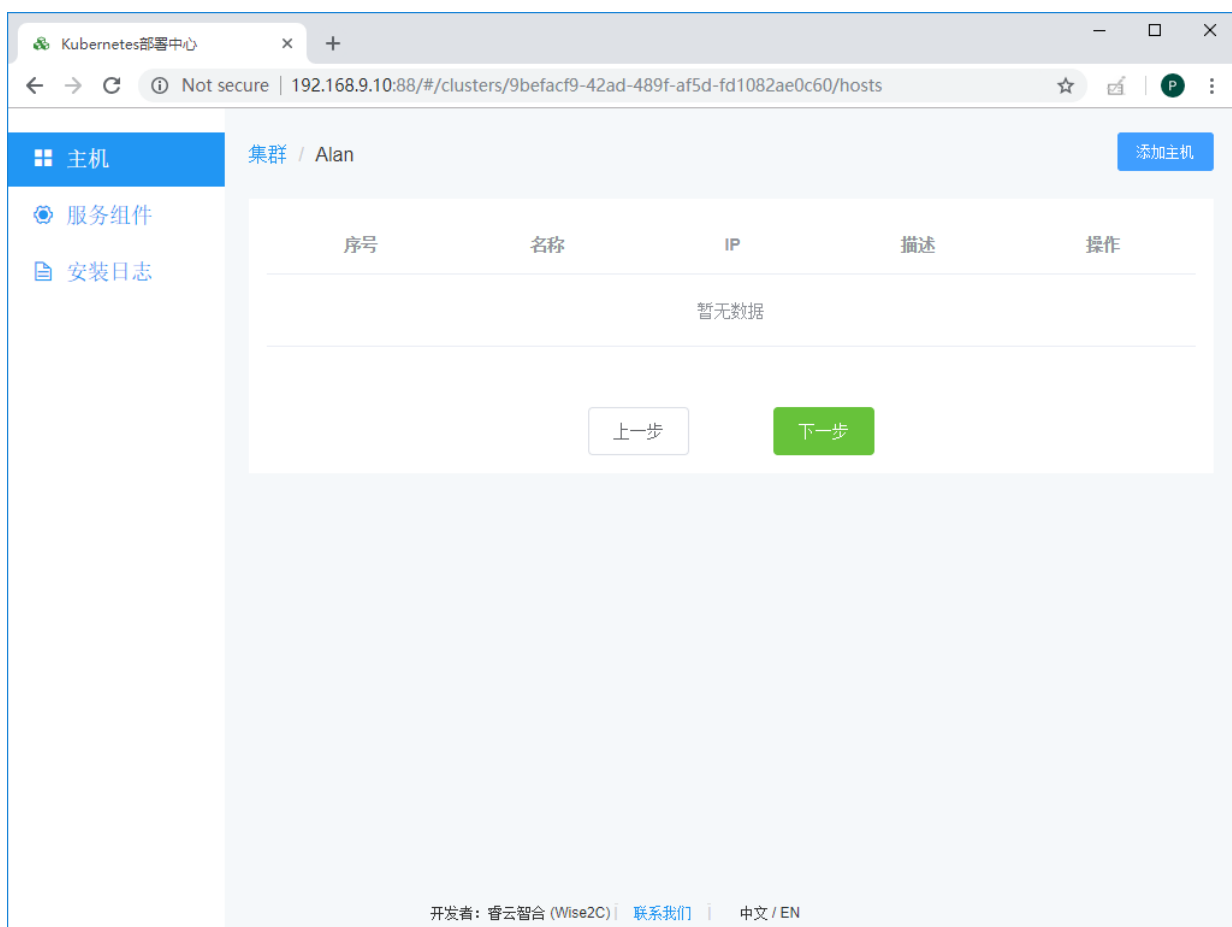


(1) 点击开始按钮后，点击+图标开始添加一个集群：

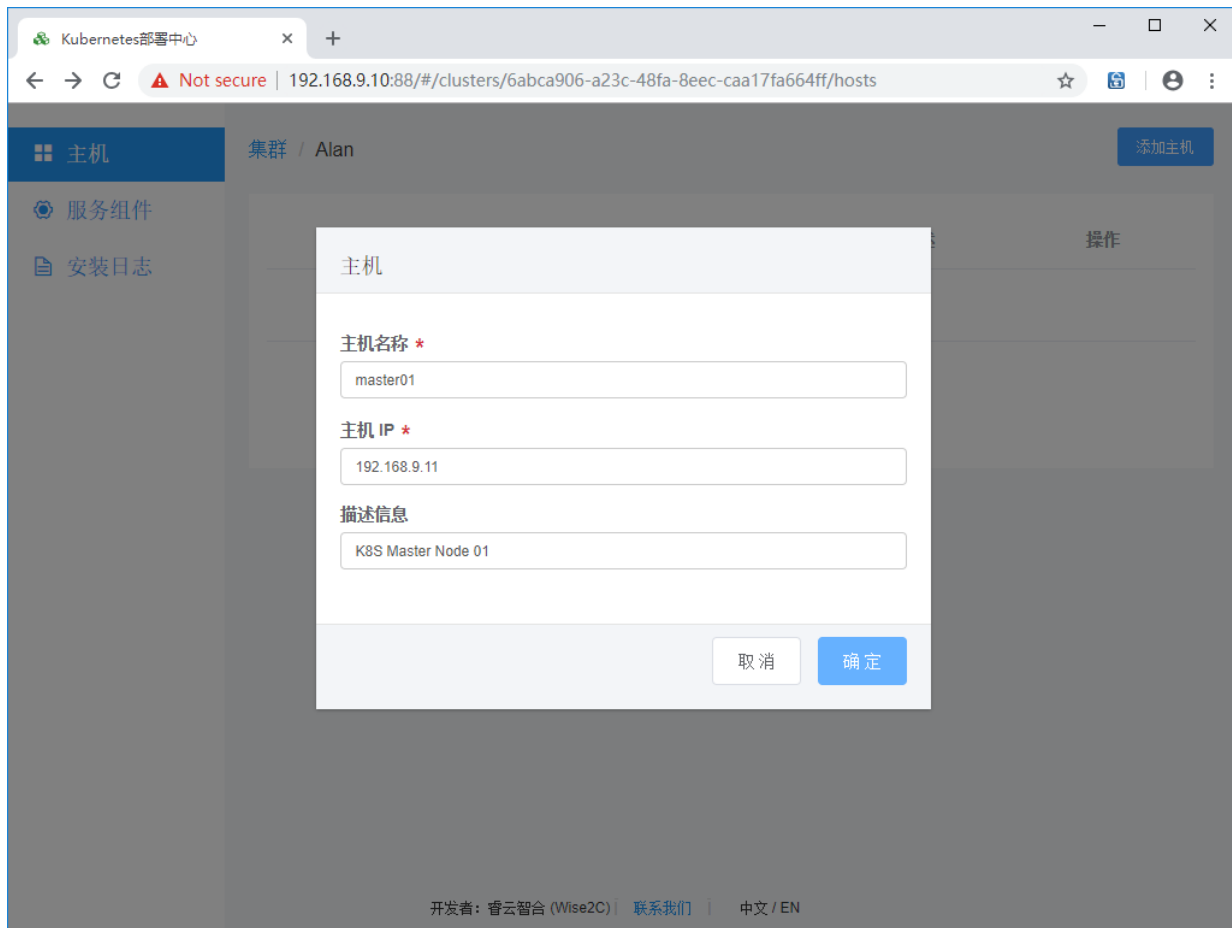




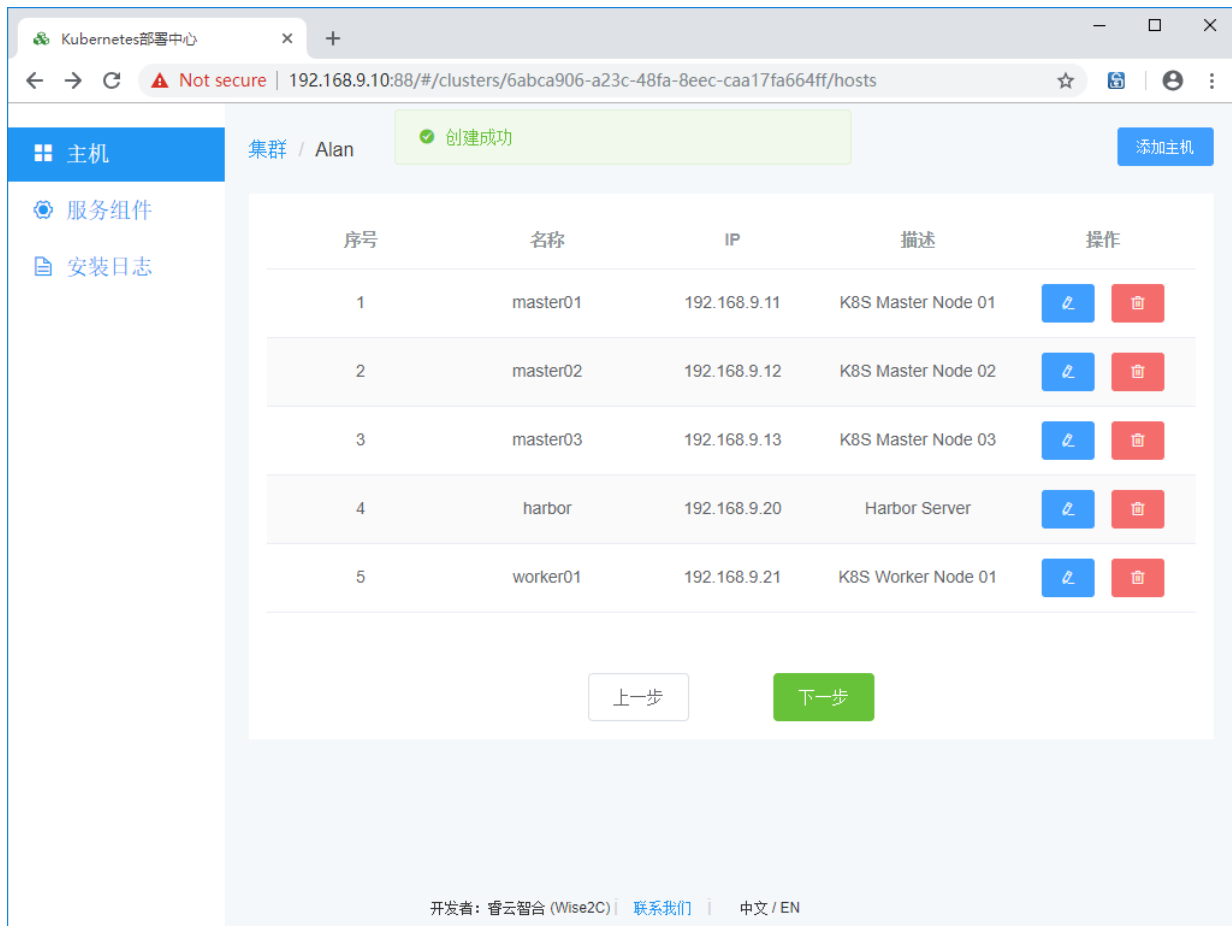
(2) 点击该集群图标进入添加主机界面：



点击右上角“添加主机按钮”：

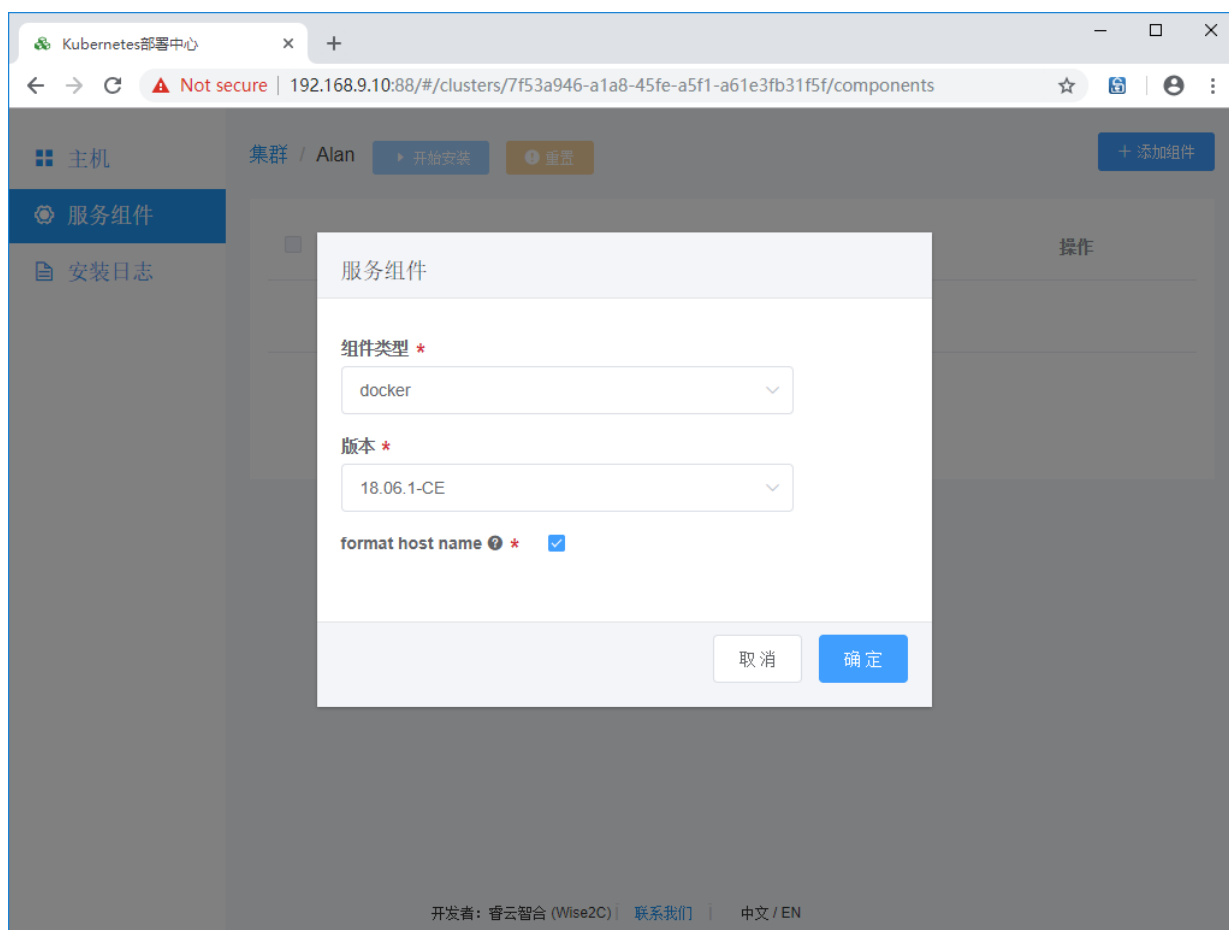


反复依次添加完整整个集群的 5 台服务器：

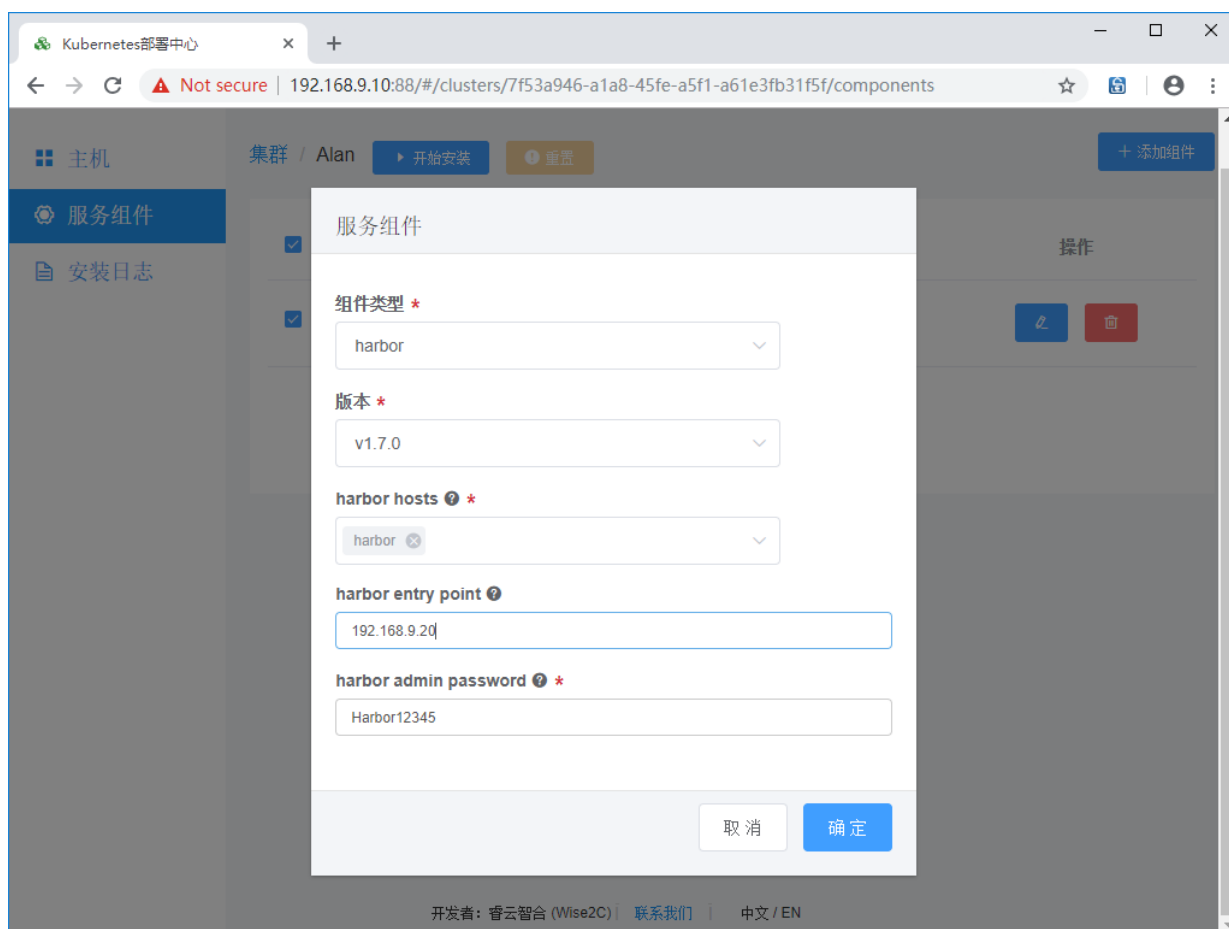


点击下一步进行服务组件定义

(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：harbor 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

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。

The screenshot shows the 'Kubernetes部署中心' (Kubernetes Deployment Center) interface. A modal window titled '服务组件' (Service Component) is open, displaying configuration options for a 'loadbalancer' component. The background shows a cluster named 'Alan' with a '开始安装' (Start Installation) button and a '+ 添加组件' (Add Component) button.

**服务组件**

**组件类型 \***  
loadbalancer

**版本 \***  
HAProxy-1.8.14\_Keepalived-1.3.5

**haproxy hosts ? \***  
k8s01 x k8s02 x k8s03 x

**vip for k8s master ? \***  
192.168.9.30

**网卡 ? \***  
ens33

**网卡掩码 ? \***  
24

**router id ?**  
10

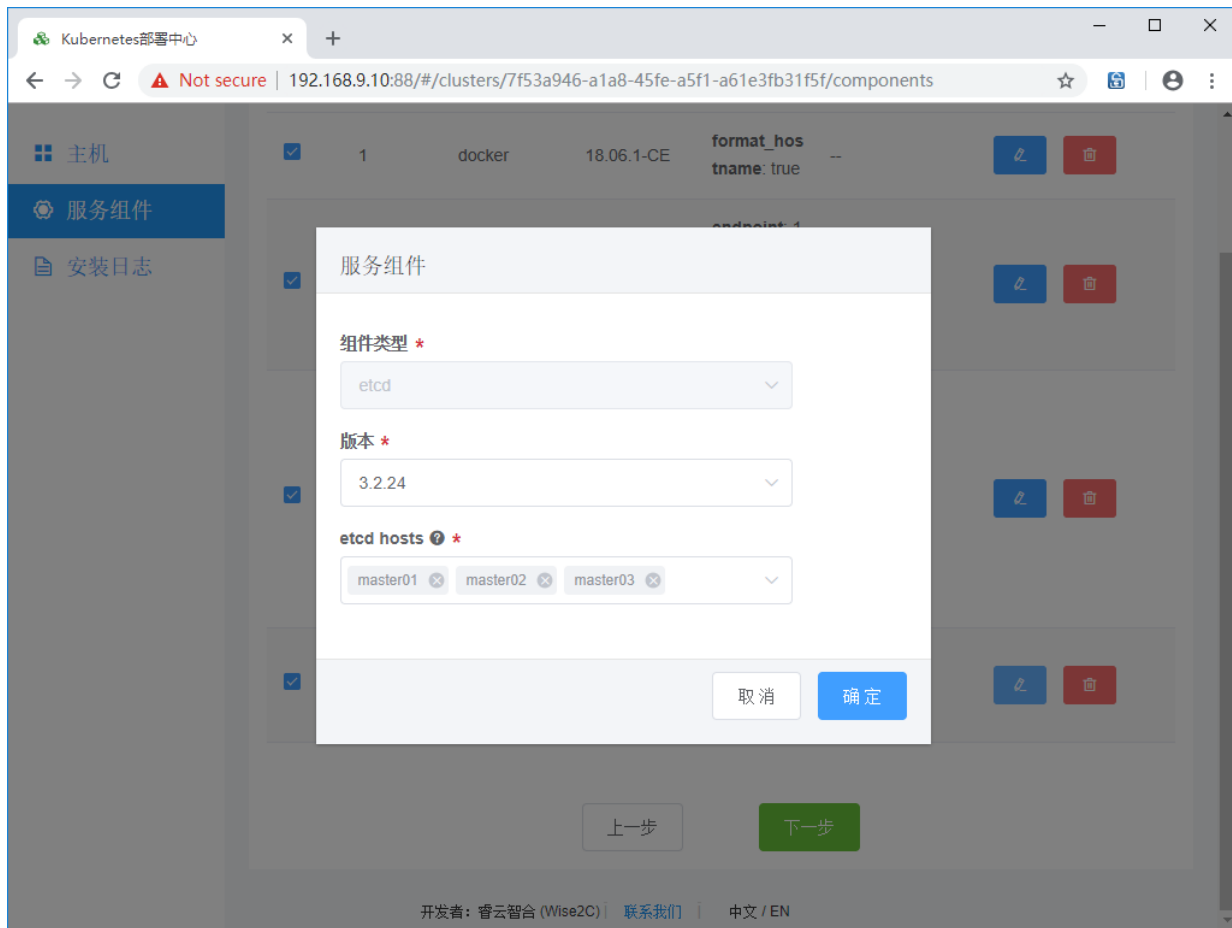
**virtual route id ?**  
160

**操作**  
[Edit] [Delete]

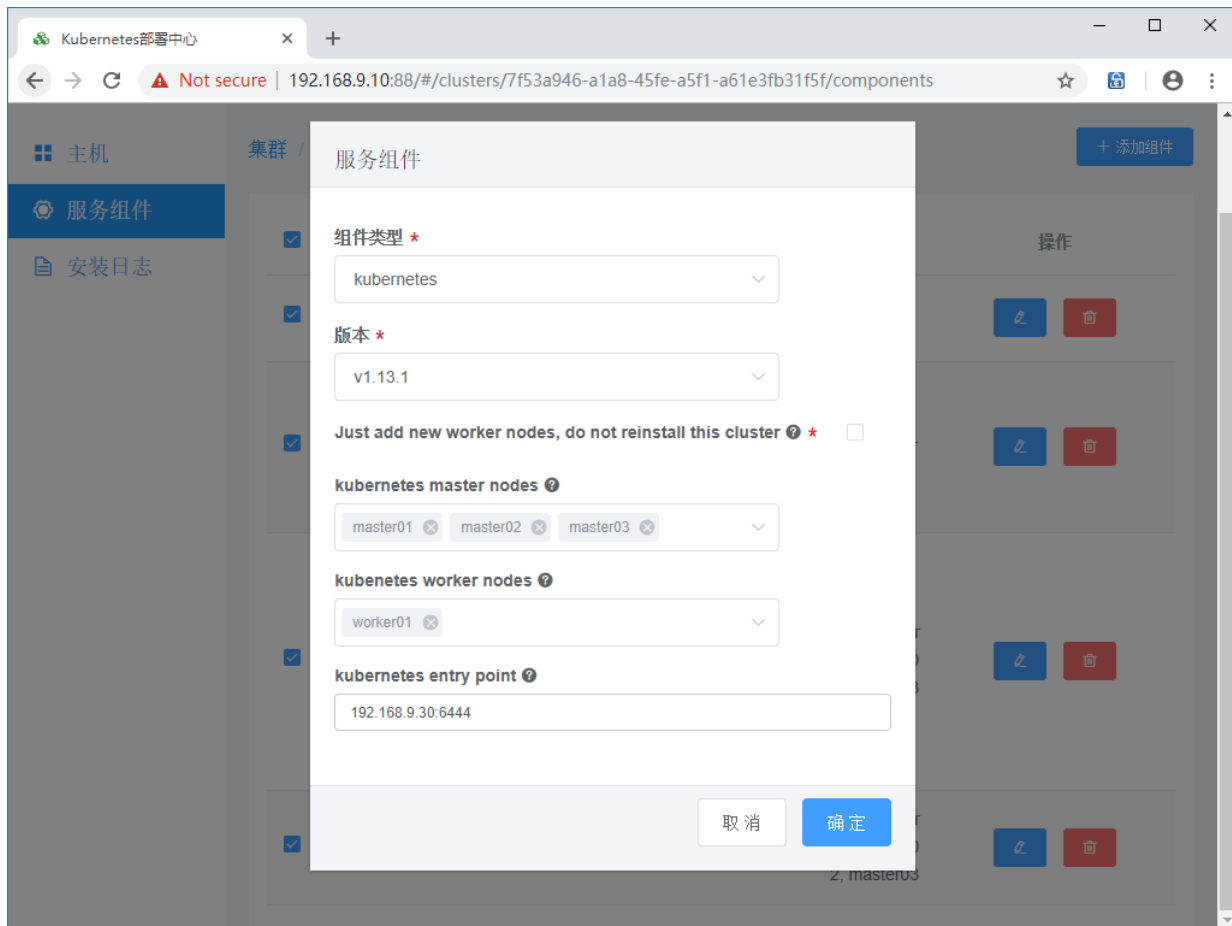
取消 确定

开发者：睿云智合 (Wise2C) | 联系我们 | 中文 / EN

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：



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

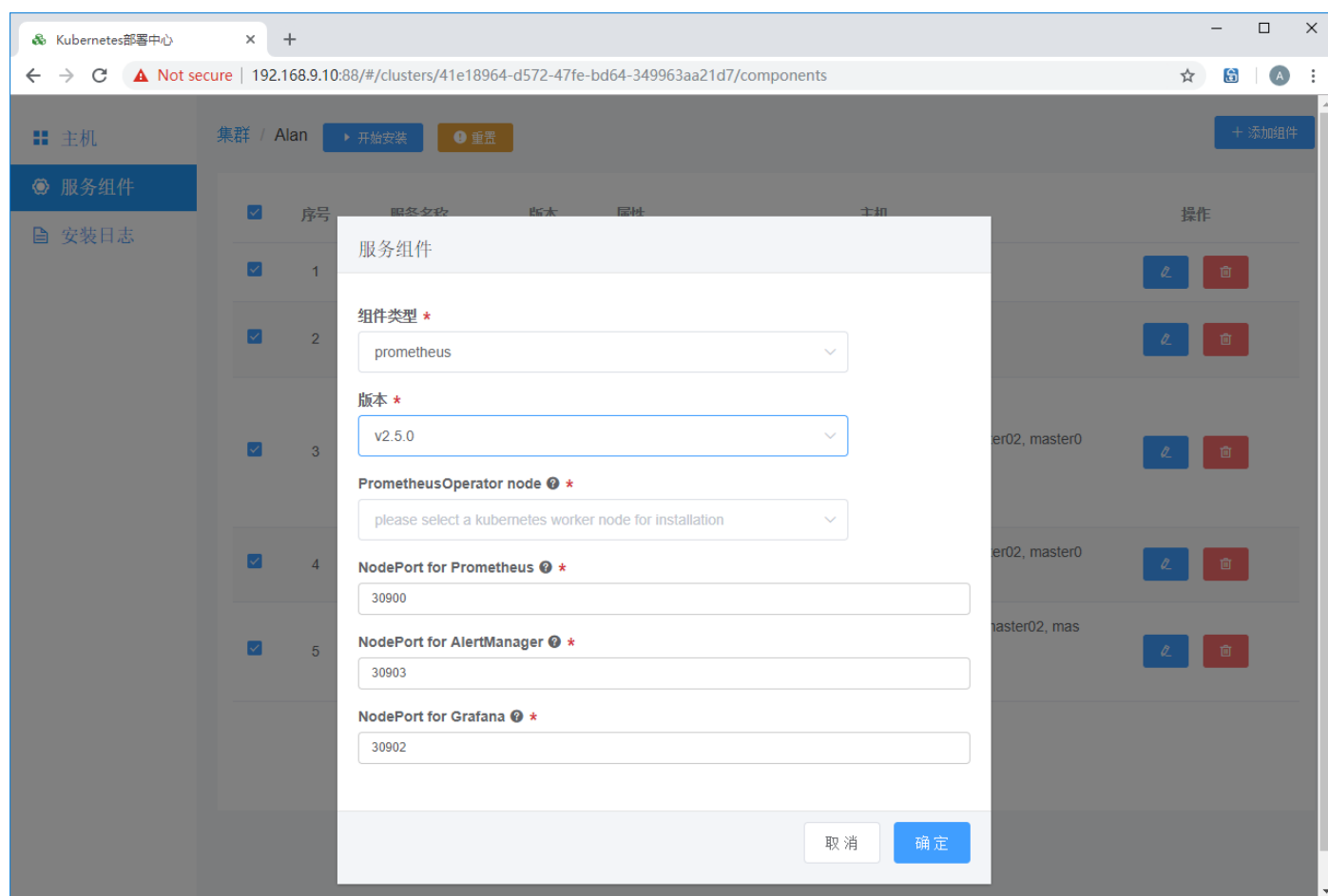


备注：这里 kubernetes entry point 是为了 HA 场景，比如此次试验我们在每一个 k8s master 节点同时各部署了



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

可选 PrometheusOperator+Kube-Prometheus 组件，只需要定义三个服务分别暴露的 NodePort 即可：



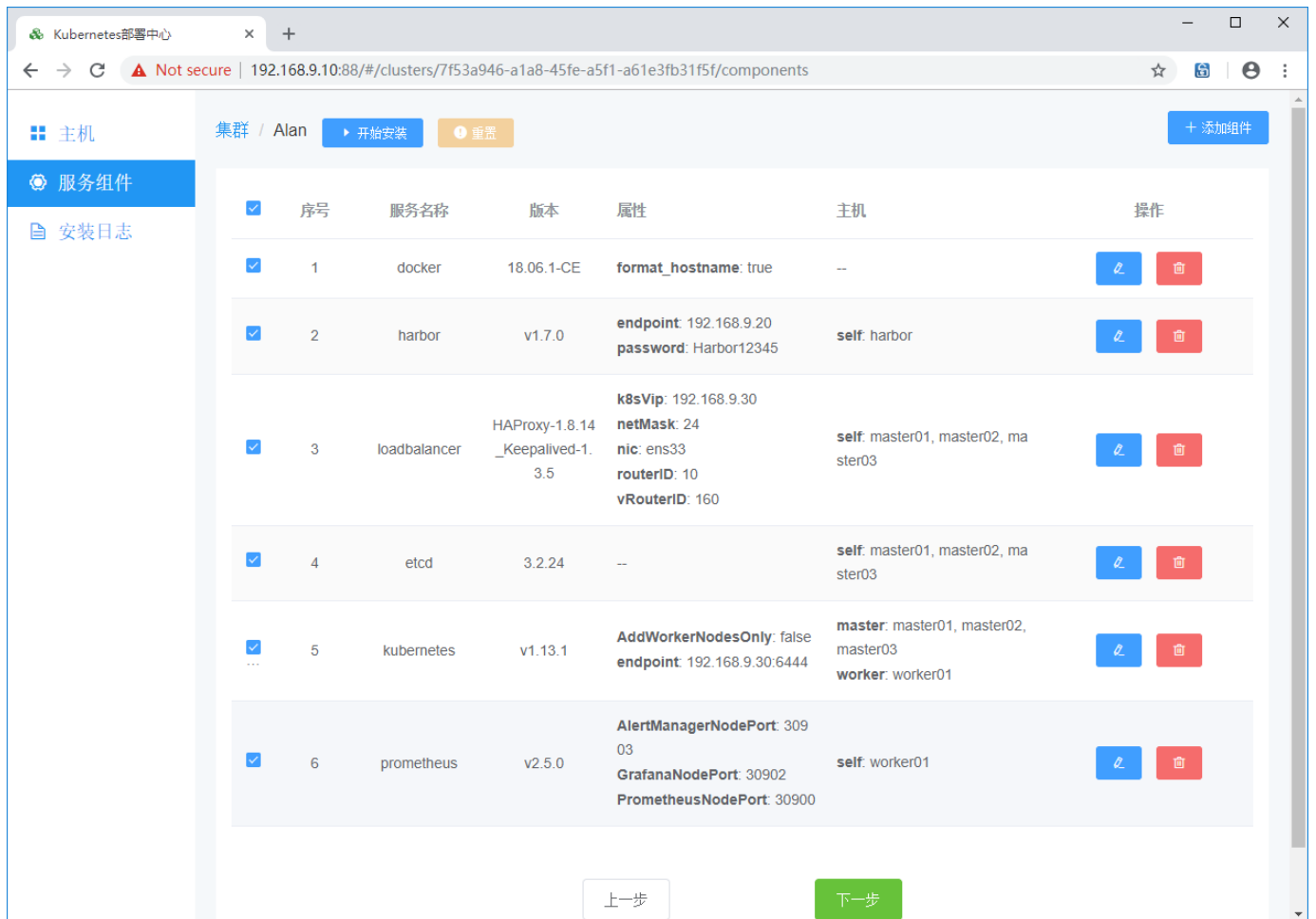
适用操作系统为RHEL 7.4/7.5/7.6 或 CentOS 7.4/7.5/7.6

Note:

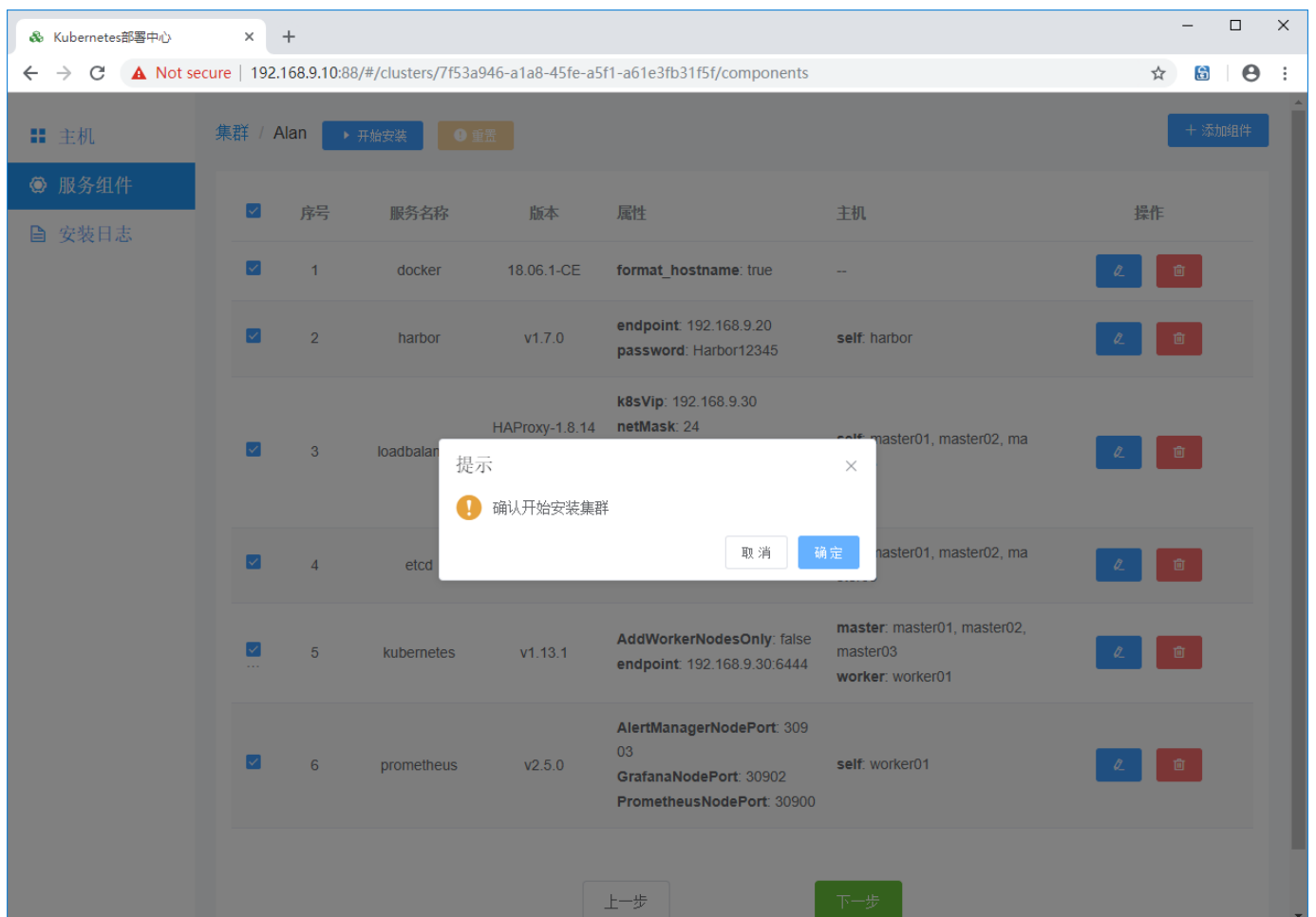
1. 请不要把Breeze所在的部署主机加入部署集群主机列表
2. 为了避免包冲突，请使用纯净的CentOS Minimal安装出来的OS来部署集群
3. PrometheusOperator + Kube-Prometheus项目为选装项，需要该功能的中国区用户请务必先对每台被部署机节点设置正确的时区，可参照以下命令：

```
timedatectl set-timezone Asia/Shanghai
```

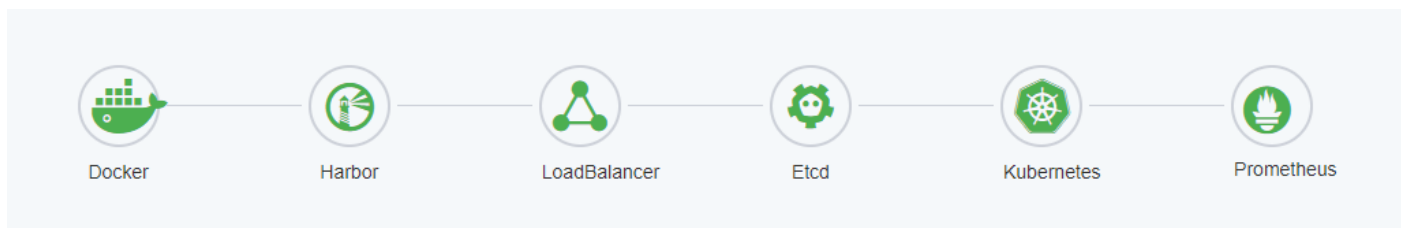
设置完成的界面如下：



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：

kubectl get cs

kubectl get csr

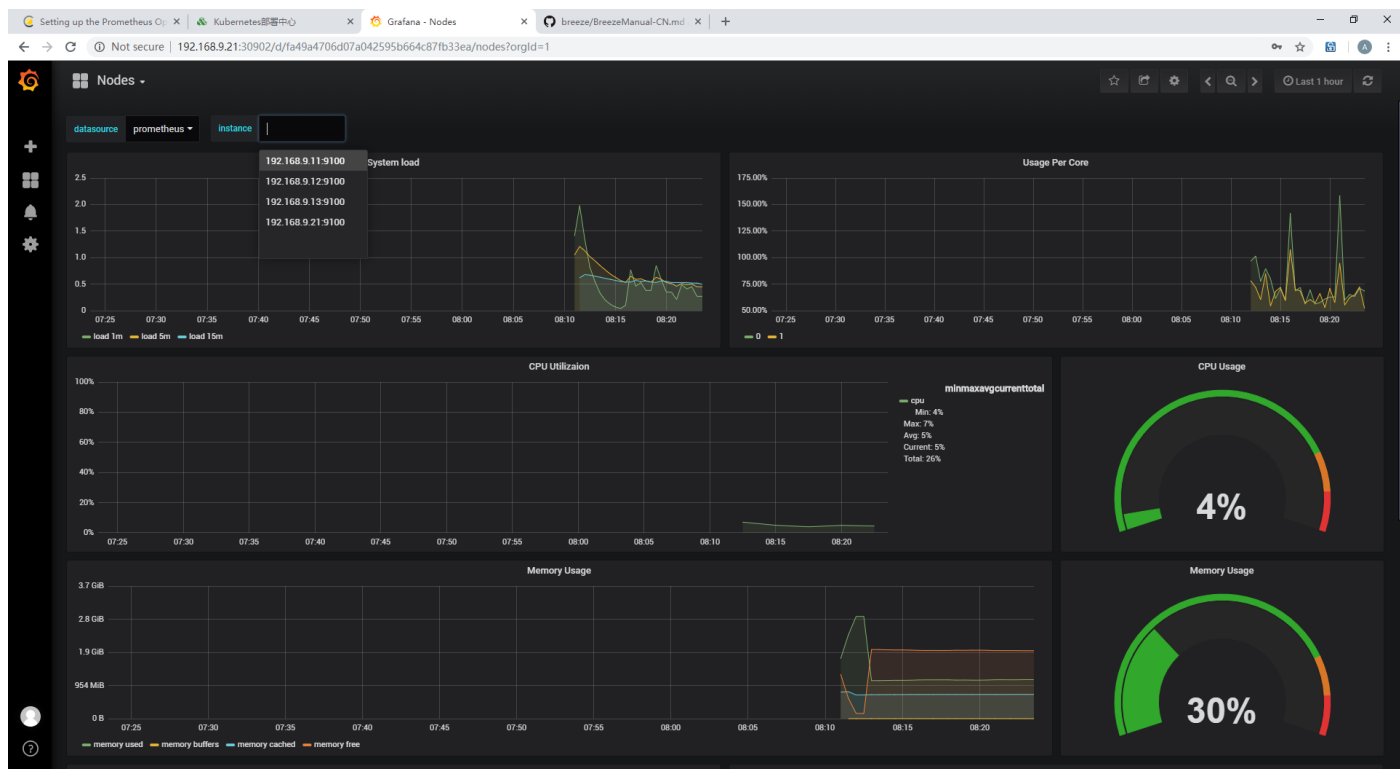
kubectl get nodes -o wide

kubectl -n kube-system get pods

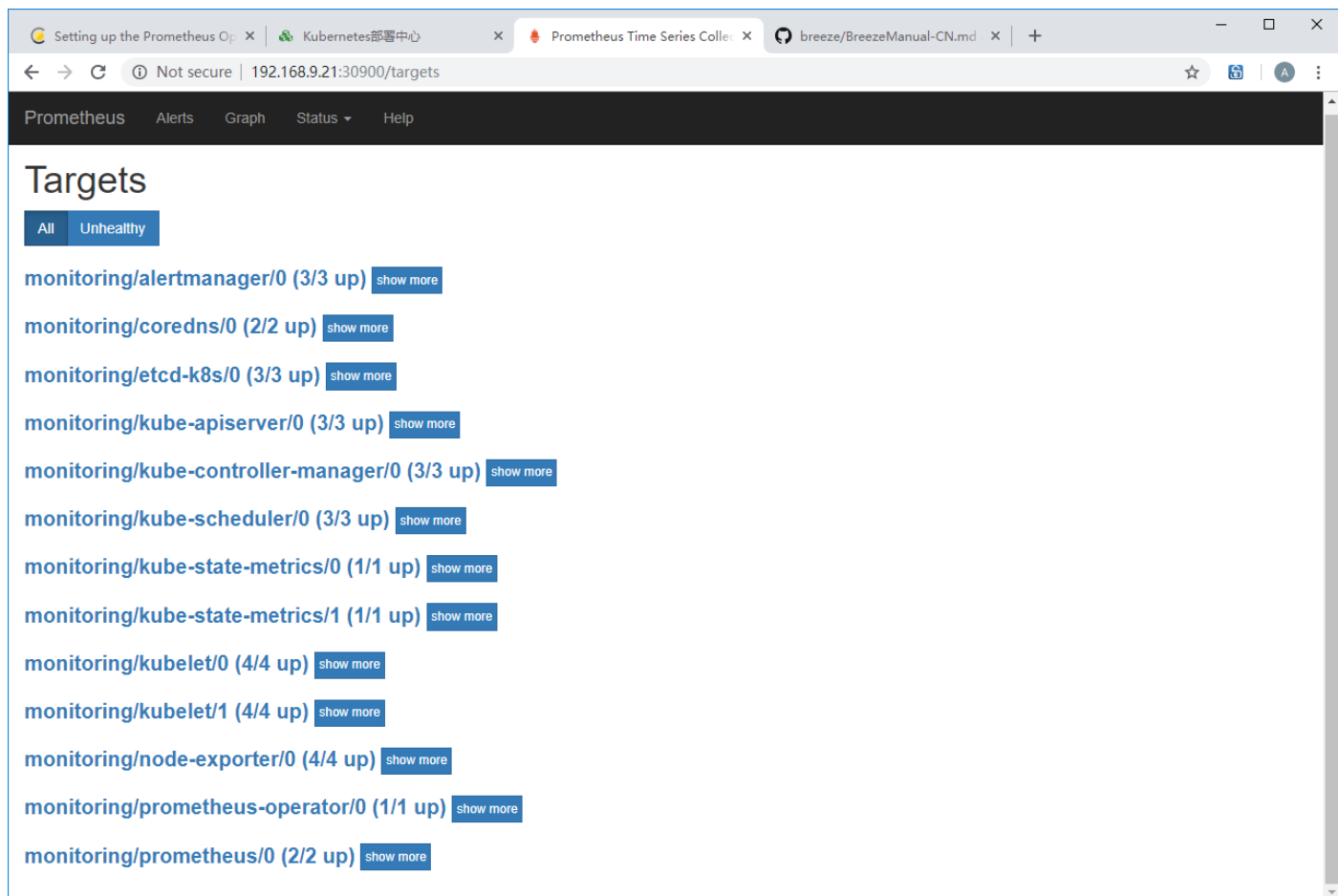
kubectl -n monitoring get pods

```
1 Deploy x 2 Harbor x 3 Master01 x 4 Master02 x 5 Master03 x 6 Worker01 x +
[root@worker01 ~]# kubectl get cs
NAME                STATUS    MESSAGE             ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-1               Healthy   {"health": "true"}
etcd-2               Healthy   {"health": "true"}
etcd-0               Healthy   {"health": "true"}
[root@worker01 ~]# kubectl get nodes -o wide
NAME                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
master01            Ready     master   83m   v1.13.1   192.168.9.11  <none>        CentOS Linux 7 (Core) 3.10.0-957.el7.x86_64 docker://18.6.1
master02            Ready     master   83m   v1.13.1   192.168.9.12  <none>        CentOS Linux 7 (Core) 3.10.0-957.el7.x86_64 docker://18.6.1
master03            Ready     master   82m   v1.13.1   192.168.9.13  <none>        CentOS Linux 7 (Core) 3.10.0-957.el7.x86_64 docker://18.6.1
worker01            Ready     <none>    81m   v1.13.1   192.168.9.21  <none>        CentOS Linux 7 (Core) 3.10.0-957.el7.x86_64 docker://18.6.1
[root@worker01 ~]# kubectl -n kube-system get pods
NAME                READY     STATUS    RESTARTS   AGE
coredns-9dbbc75f6-6d7l5  1/1      Running   0           82m
coredns-9dbbc75f6-v859q  1/1      Running   0           82m
kube-apiserver-master01  1/1      Running   0           82m
kube-apiserver-master02  1/1      Running   0           82m
kube-apiserver-master03  1/1      Running   0           82m
kube-controller-manager-master01  1/1      Running   0           82m
kube-controller-manager-master02  1/1      Running   0           82m
kube-controller-manager-master03  1/1      Running   0           82m
kube-flannel-ds-6nwt6    1/1      Running   0           82m
kube-flannel-ds-7xkd5    1/1      Running   0           81m
kube-flannel-ds-9wh8g    1/1      Running   0           82m
kube-flannel-ds-fpdqt    1/1      Running   0           82m
kube-proxy-hbgpt        1/1      Running   0           81m
kube-proxy-mfplf        1/1      Running   0           82m
kube-proxy-q9bqr        1/1      Running   0           82m
kube-proxy-wtpwz        1/1      Running   0           82m
kube-scheduler-master01  1/1      Running   0           82m
kube-scheduler-master02  1/1      Running   0           82m
kube-scheduler-master03  1/1      Running   0           82m
kubernetes-dashboard-84cf4d5bbd-7fc2m  1/1      Running   0           82m
[root@worker01 ~]# kubectl -n monitoring get pods
NAME                READY     STATUS    RESTARTS   AGE
alertmanager-main-0  2/2      Running   0           78m
alertmanager-main-1  2/2      Running   0           78m
alertmanager-main-2  2/2      Running   0           78m
grafana-654bbf9c67-cszg7  1/1      Running   0           79m
kube-state-metrics-57b57dd9d5-vf4vt  4/4      Running   0           78m
node-exporter-pmdb2    2/2      Running   0           79m
node-exporter-qkmd4    2/2      Running   0           79m
node-exporter-vg8zx    2/2      Running   0           79m
node-exporter-z9rzn    2/2      Running   0           79m
prometheus-adapter-748944fff8-nsv7d  1/1      Running   0           79m
prometheus-k8s-0       3/3      Running   1           78m
prometheus-k8s-1       3/3      Running   1           78m
prometheus-operator-7b69687684-dl8qd  1/1      Running   0           79m
[root@worker01 ~]#
```

Grafana: <http://node:30902>



Prometheus: <http://node:30900>



Alertmanager: <http://node:30903>

The screenshot shows the Alertmanager web interface in a browser. The address bar indicates the URL is `192.168.9.21:30903/#/alerts`. The interface has tabs for 'Alerts', 'Silences', and 'Status', with 'Alerts' being the active tab. A 'New Silence' button is in the top right. Below the tabs, there's a filter section with 'Filter' and 'Group' tabs, and a 'Receiver: All' dropdown. A search bar with a '+' button is present, with a hint 'Custom matcher, e.g. env="production"'. Below this, a filter box contains 'alertname="AlertmanagerMembersInconsistent"' with a '+' button. The main area displays three alert entries, each with a timestamp '00:16:18, 2018-12-26' and links for '+ Info', 'Source', and 'Silence'. Each alert entry has a set of labels: 'severity="critical"', 'service="alertmanager-main"', 'prometheus="monitoring/k8s"', 'pod="alertmanager-main-2"' (for the first), 'namespace="monitoring"', 'job="alertmanager-main"', 'instance="10.244.3.11:9093"', and 'endpoint="web"' (for the first). The second alert has 'pod="alertmanager-main-1"' and the third has 'pod="alertmanager-main-0"'. Each label has a '+' button to expand it.

Alan Peng  
2019 年 1 月