

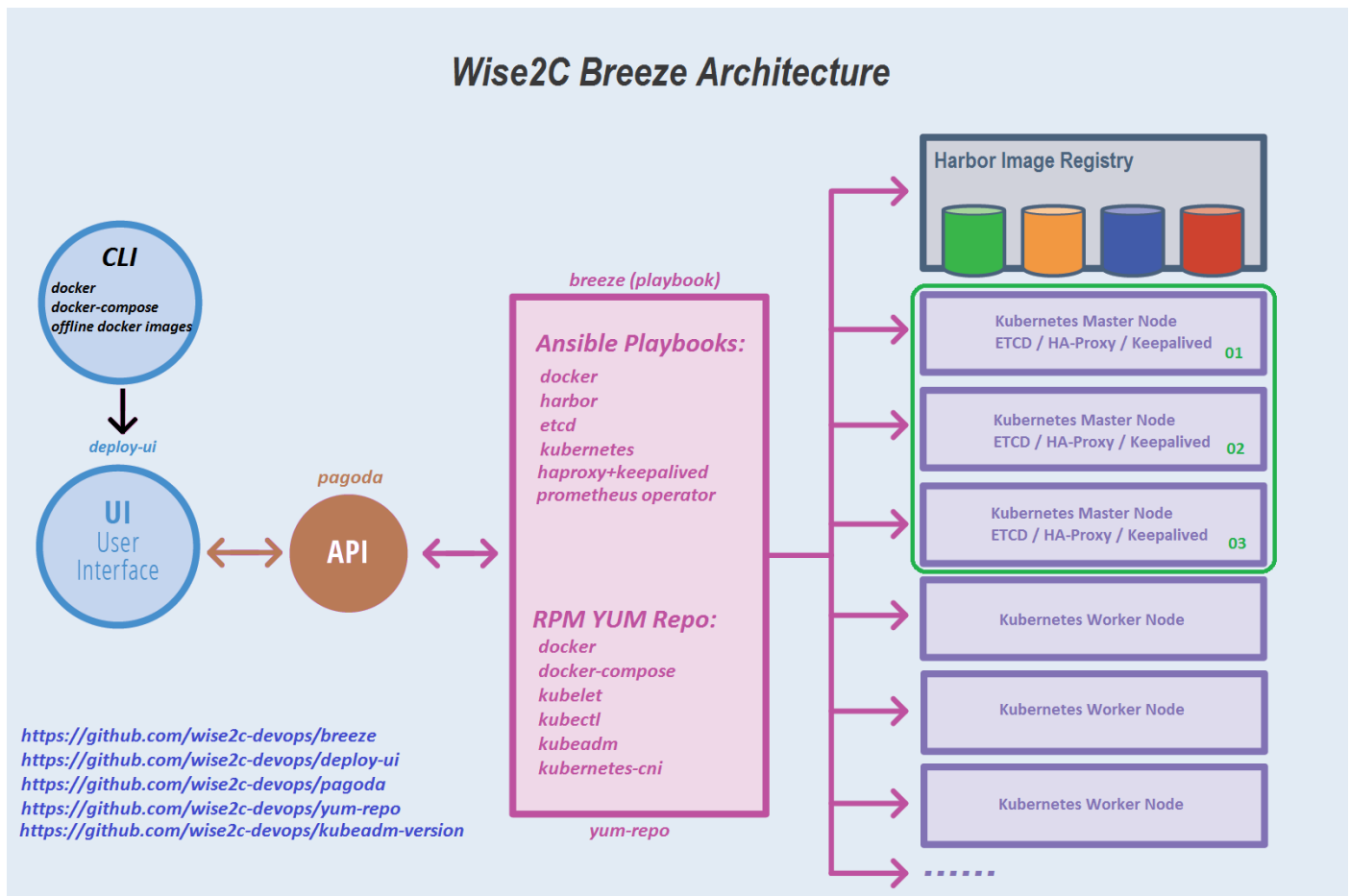
Breeze - Deploy a Production Ready Kubernetes Cluster with graphical interface

Project Breeze is an open source trusted solution allow you to create Kubernetes clusters on your internal, secure, cloud network with graphical user interface. (<https://github.com/wise2c-devops/breeze>)

Features

- * **Easy to run:** Breeze combines all resources you need such as kubernetes components images, ansible playbooks for the deployment of kubernetes clusters into a single docker image (wise2c/playbook). It also works as a local yum repository server. You just need a linux server with docker and docker-compose installed to run Breeze.
- * **Simplified the process of kubernetes clusters deployment:** With a few simple commands, you can get Breeze running, and then finish all the other deployment processes by the graphical interface.
- * **Support offline deployment:** After 4 images (playbook, yum-repo, pagoda, deploy-ui) have been loaded on the deploy server, kubernetes clusters can be setup without internet access. Breeze works as a yum repository server and deploys a local Harbor registry and uses kubeadm to setup kubernetes clusters. All docker images will be pulled from the local Harbor registry.
- * **Support multi-cluster:** Breeze supports multiple kubernetes clusters deployment.
- * **Support high available architecture:** With Breeze, you can setup kubernetes clusters with 3 master servers and 3 etcd servers combined with haproxy and keepalived. All worker nodes will use the virtual floating ip address to communicate with the master servers.

Architecture



You just need a linux server with docker and docker-compose installed to run Breeze.

For offline deployment, just download those 4 images listed in the file docker-compose.yml.

Below is the server list in our test environment:

Hostname	IP Address	Role	OS	Components
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
harbor	192.168.9.20	Harbor	CentOS 7.6 x64	Harbor 1.7.0
	192.168.9.30	VIP		HA virtual IP address

Steps:

1. Prepare the deploy server (deploy / 192.168.9.10)

(1) Install CentOS 7.6-1810 (7.5 and 7.4 are also supported) with Minimal mode and execute commands as below:

```
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) Install docker-compose

```
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) Install docker

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

(4) ssh login to other servers without password

a) ssh keygen:

```
ssh-keygen
```

b) execute the ssh-copy-id command:

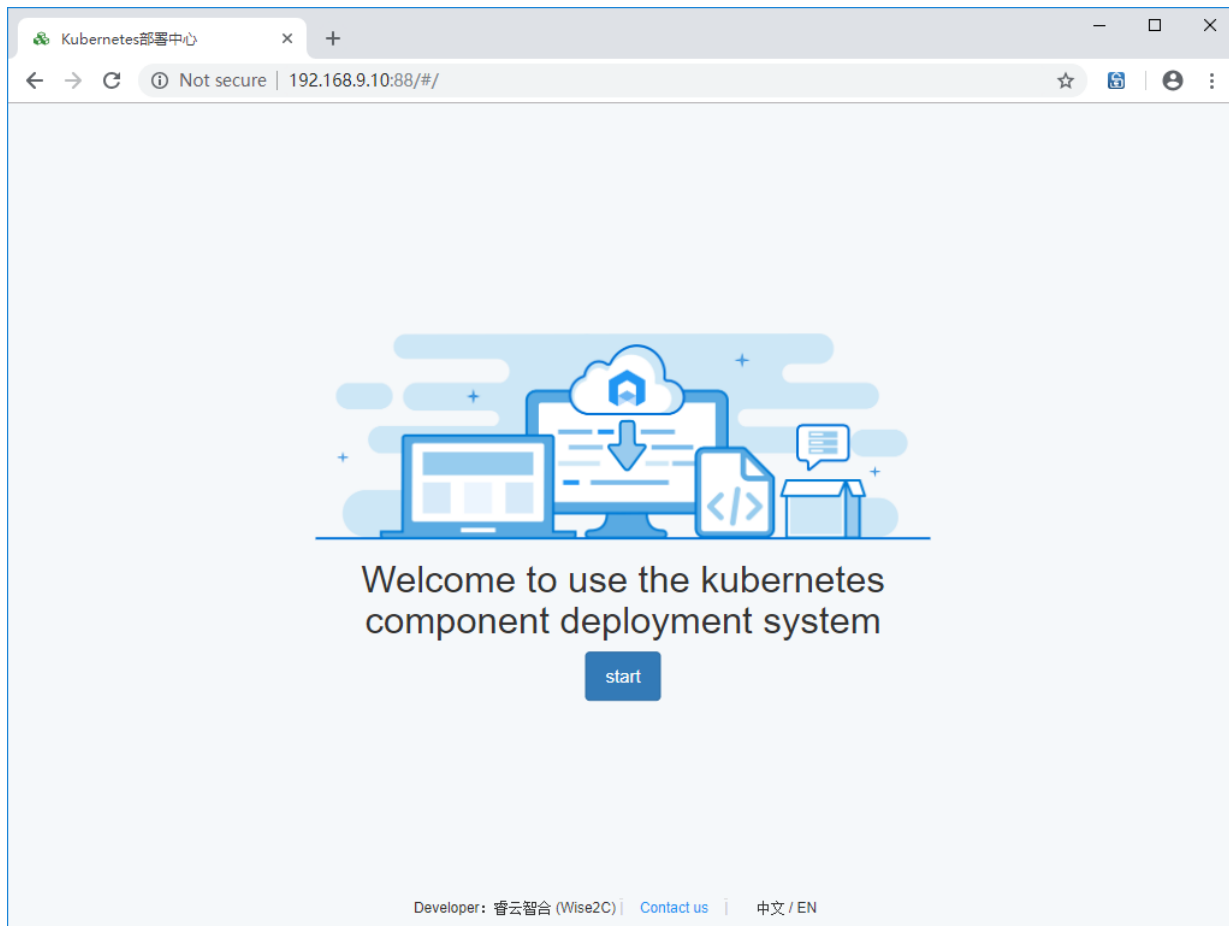
```
ssh-copy-id 192.168.9.11
ssh-copy-id 192.168.9.12
ssh-copy-id 192.168.9.13
ssh-copy-id 192.168.9.20
ssh-copy-id 192.168.9.21
```

2. Get the compose file (e.g. for Kubernetes v1.13.1)

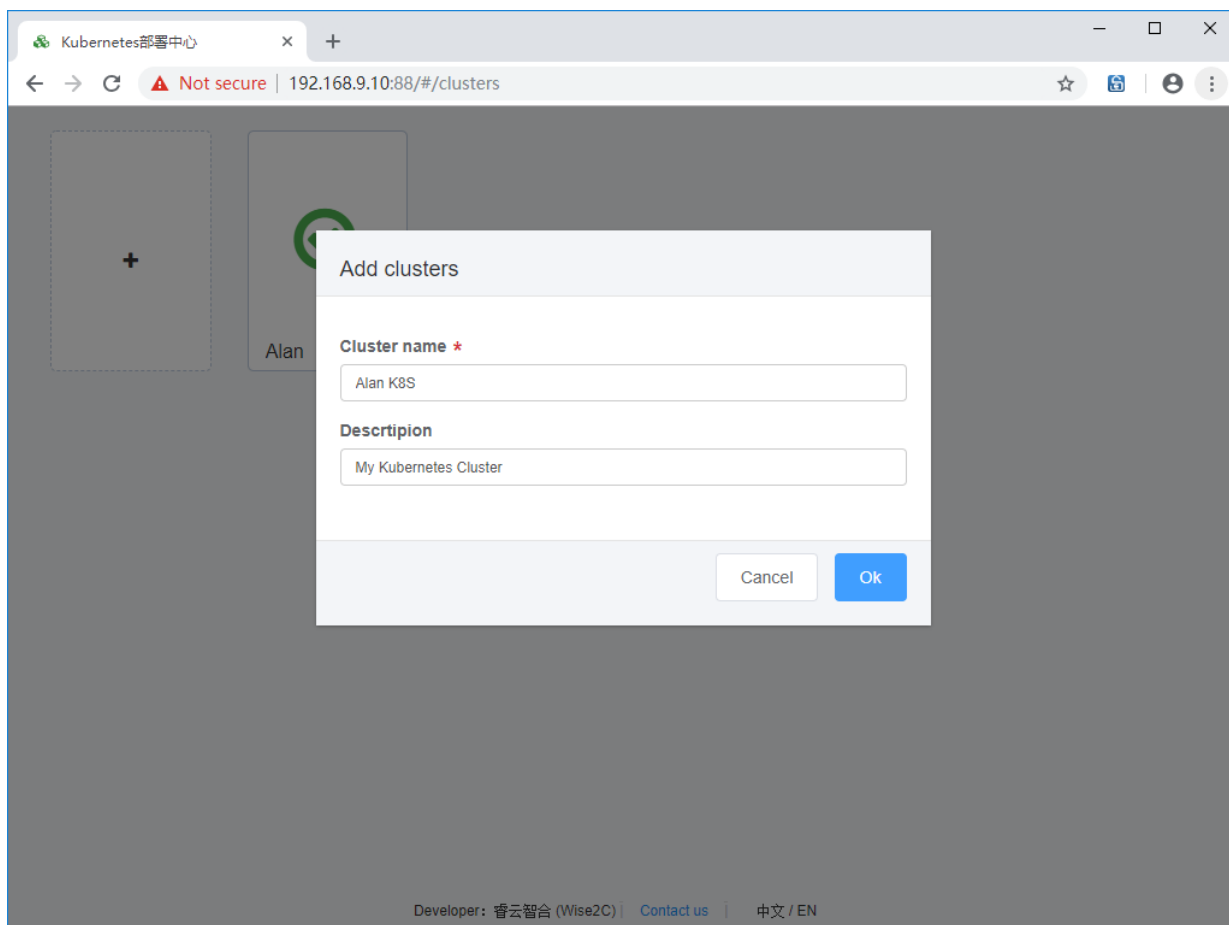
```
curl -L https://raw.githubusercontent.com/wise2c-devops/breeze/v1.13.1/docker-compose.yml -o docker-compose.yml
docker-compose up -d
```

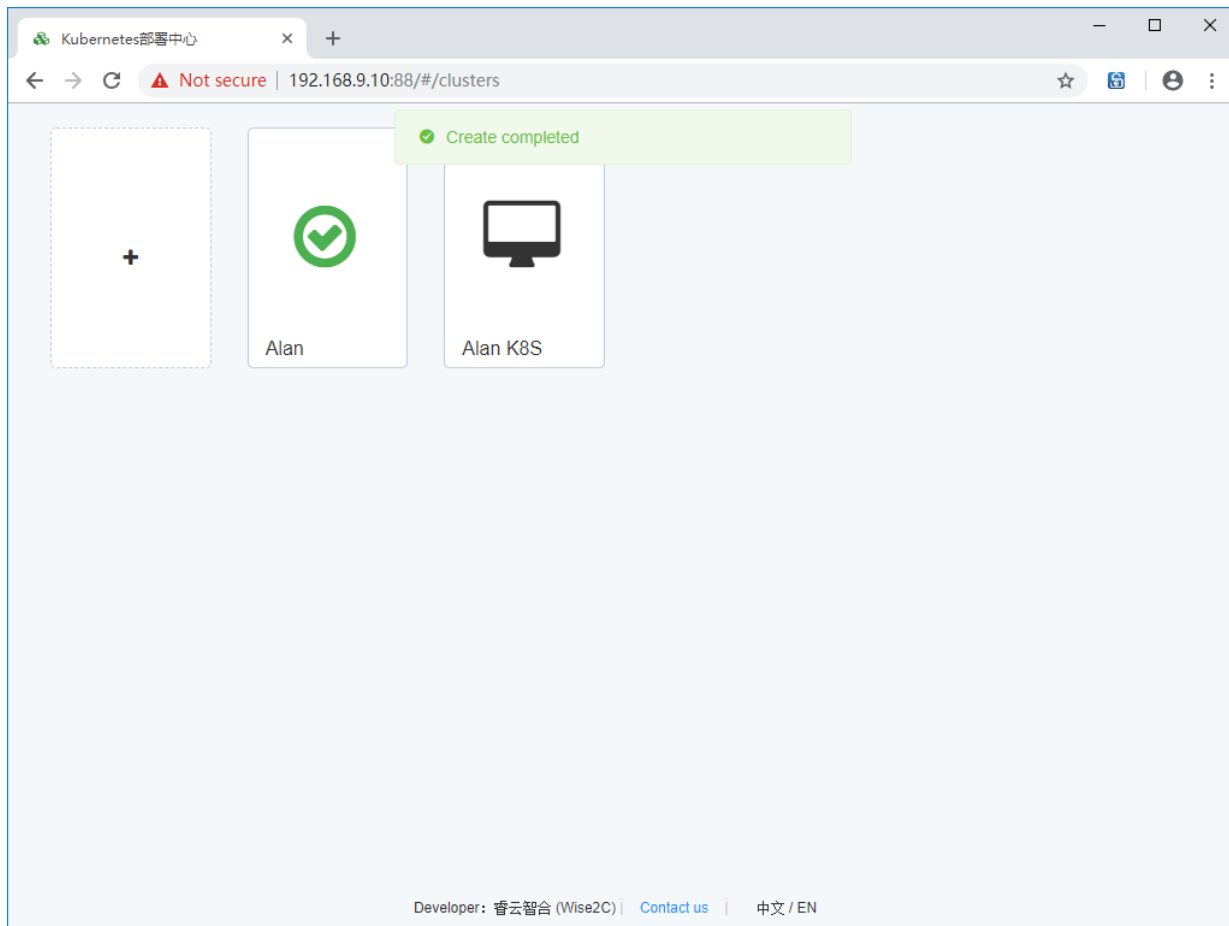
3. Access the Breeze web portal:

<http://192.168.9.10:88>

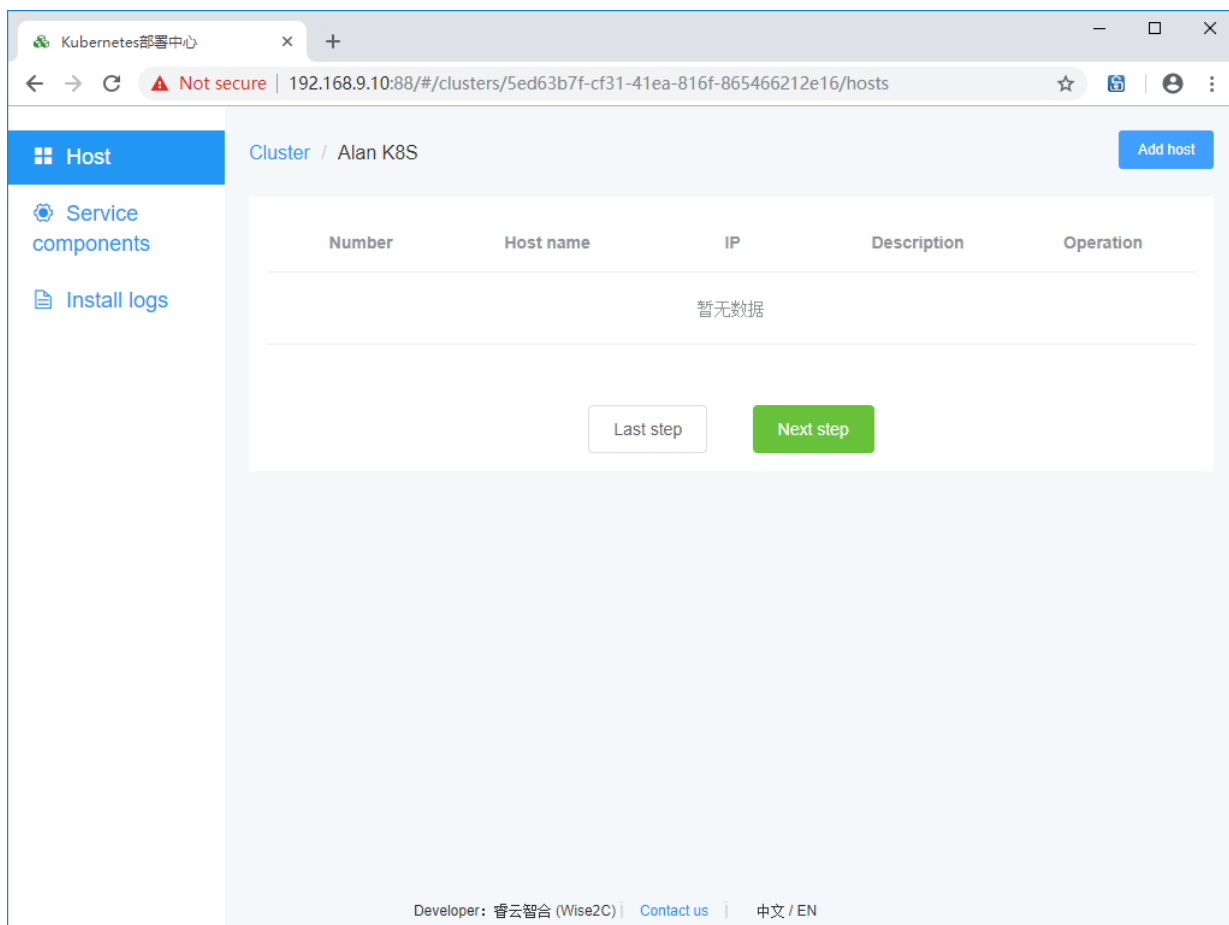


(1) Click “start” button and then click “+”

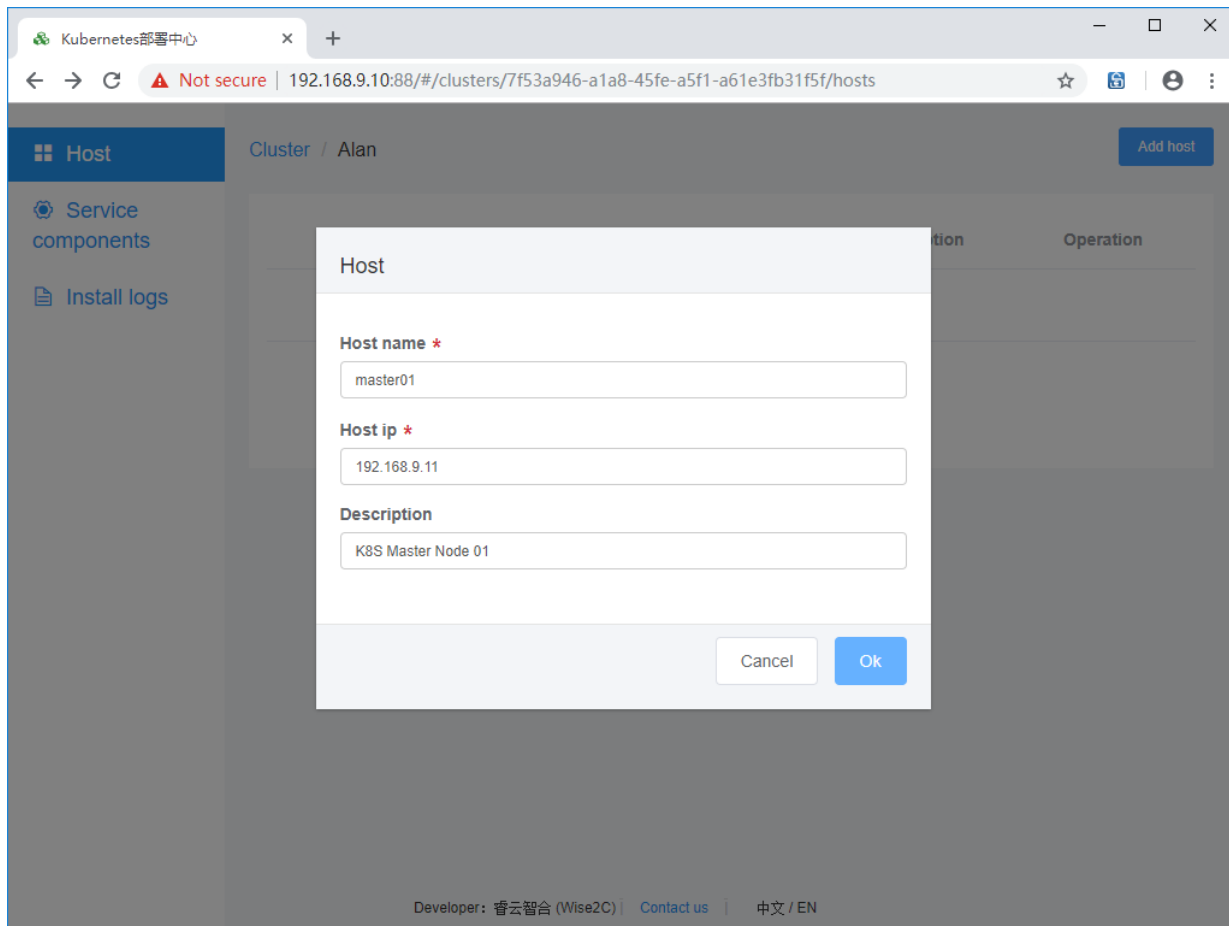




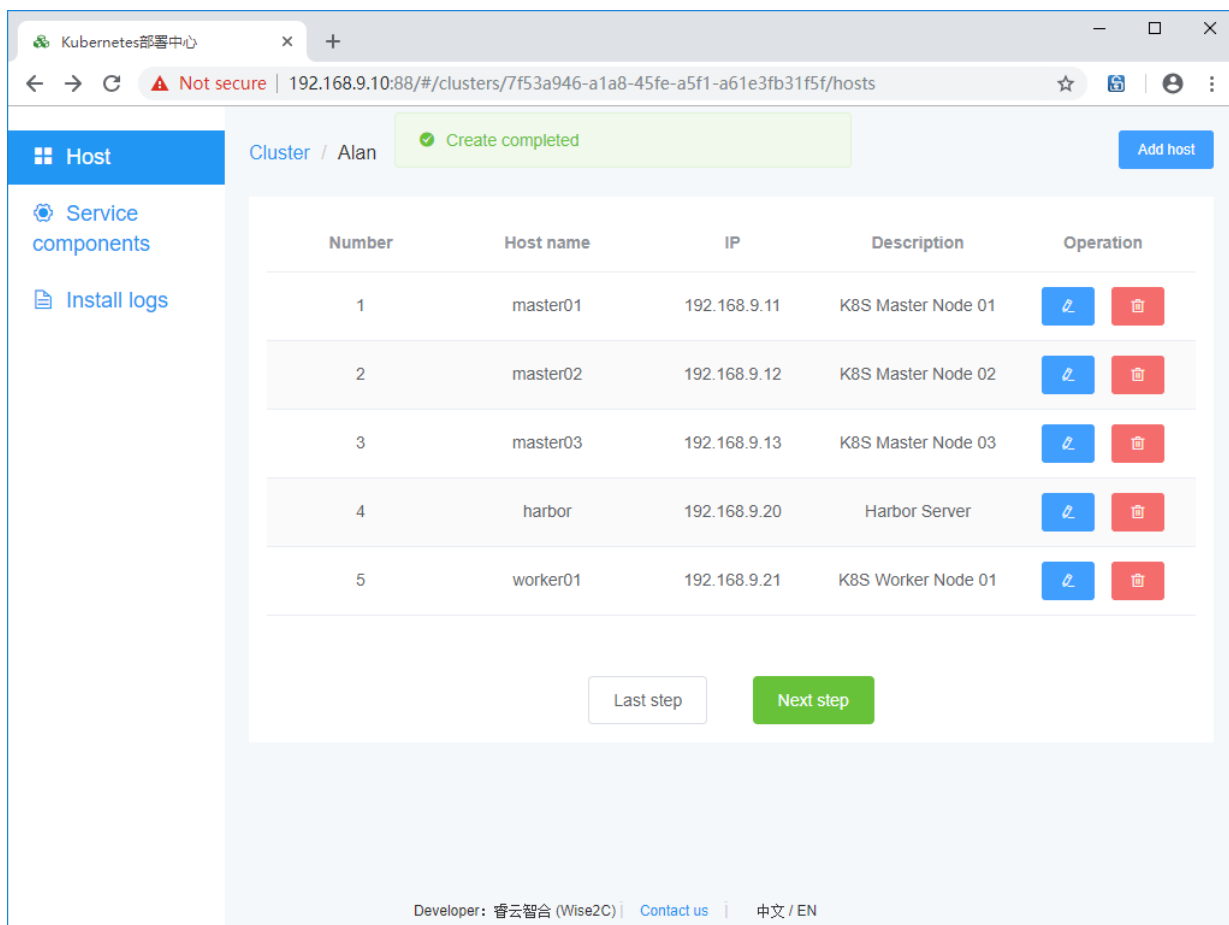
(2) Click this cluster icon to add servers:



Click the "Add host" button.

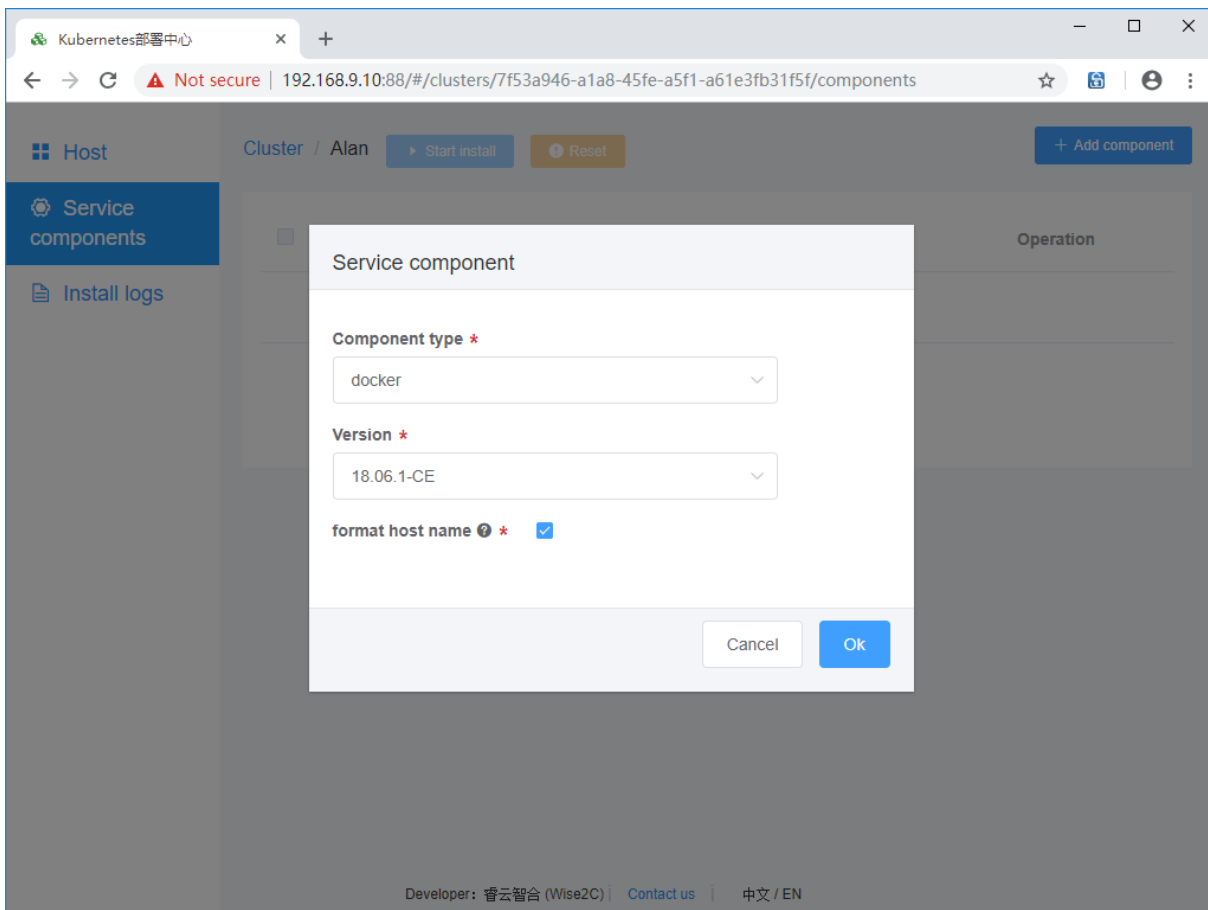


Repeatedly adding five servers to the cluster in turn:

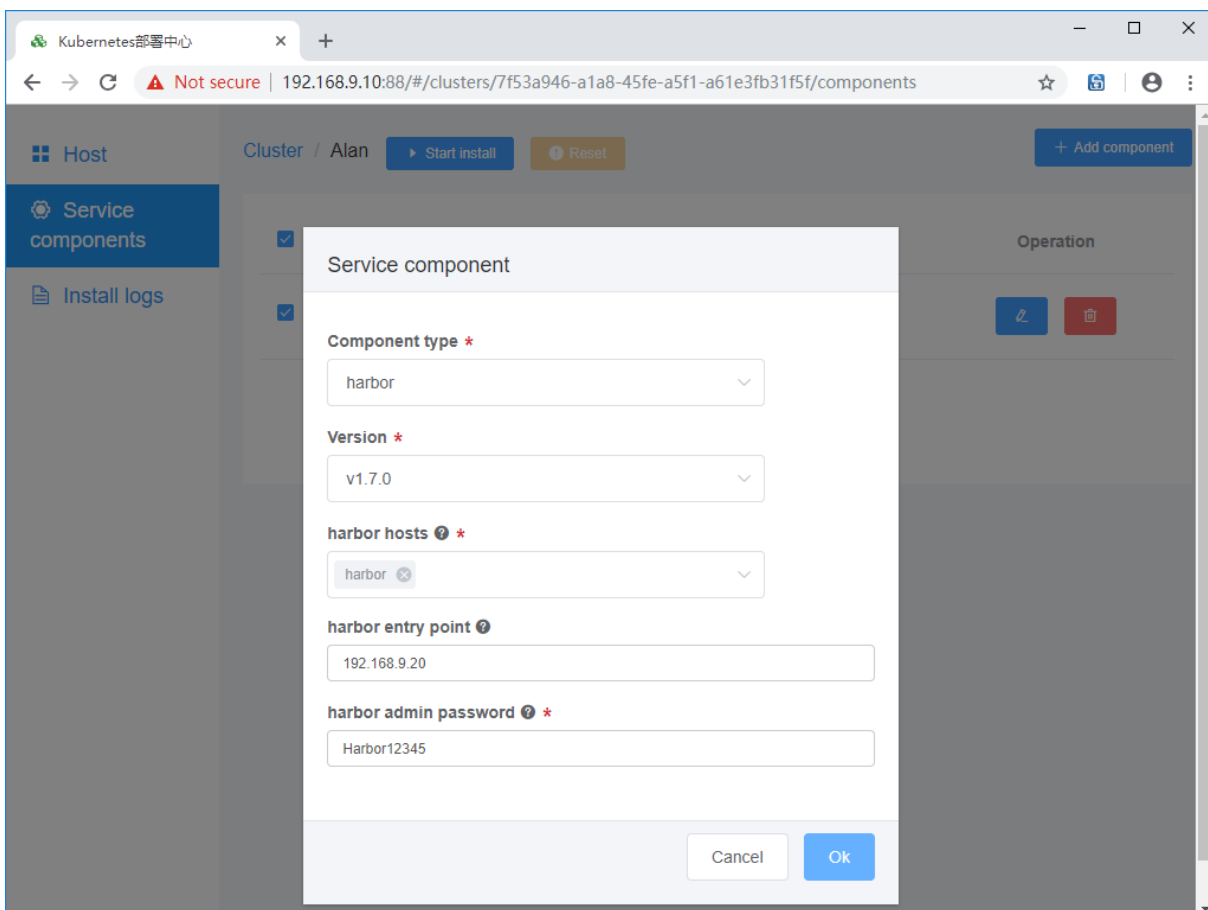


Click Next step for Service Component Definition

(3) Click on the "Add Components" button in the upper right corner to add service components and select docker, because all hosts need to be installed, so there is no need to select a server.



Adding harbor registry components. Note: registry entry point is the entry point for the Harbor server. You can use domain name or IP address.



Next is the HA components (haproxy+keepalived) :

vip for k8s master: Virtual floating IP address for master servers.

NIC name: network interface name in the linux OS. Please ensure all the interfaces have the same name.

Network mask: input the network mask bit number

router id: configuration in keepalived, do not use same value for multiple clusters

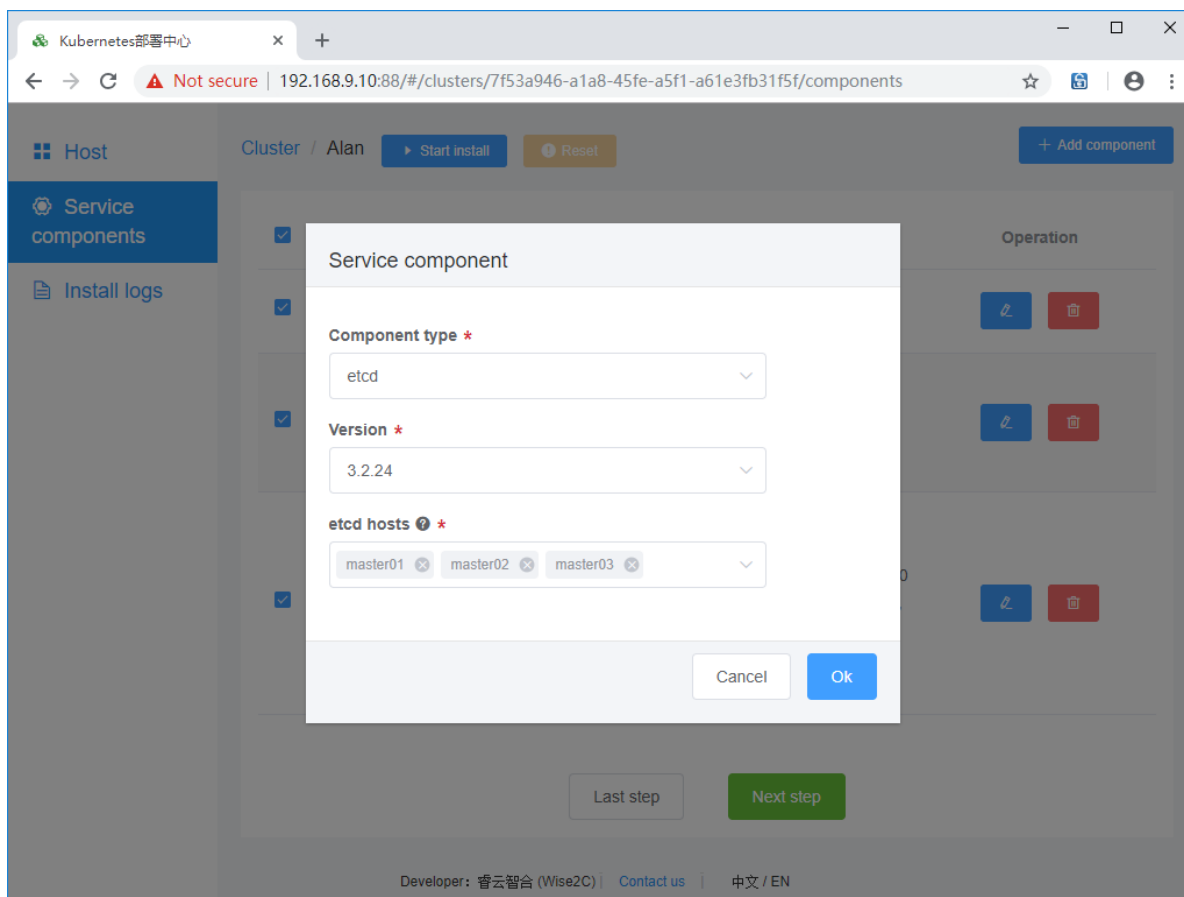
virtual router id: configuration in keepalived, do not use same value for multiple clusters

The screenshot shows a web interface for Kubernetes deployment. A modal window titled 'Service component' is open, allowing configuration for a new component. The background shows a cluster named 'Alan' with buttons for 'Start install', 'Reset', and 'Add component'. The left sidebar has links for 'Host', 'Service components', and 'Install logs'. The modal contains the following fields:

- Component type ***: A dropdown menu with 'loadbalancer' selected.
- Version ***: A dropdown menu with 'HAProxy-1.8.14_Keepalived-1.3.5' selected.
- haproxy hosts ? ***: A multi-select field containing 'master01', 'master02', and 'master03'.
- vip for k8s master ? ***: A text input field containing '192.168.9.30'.
- 网卡 ? *** (network interface name): A text input field containing 'ens33'.
- 网卡掩码 ? *** (network mask bits): A text input field containing '24'.
- router id ?**: A text input field containing '10'.
- virtual route id ?**: A text input field containing '160'.

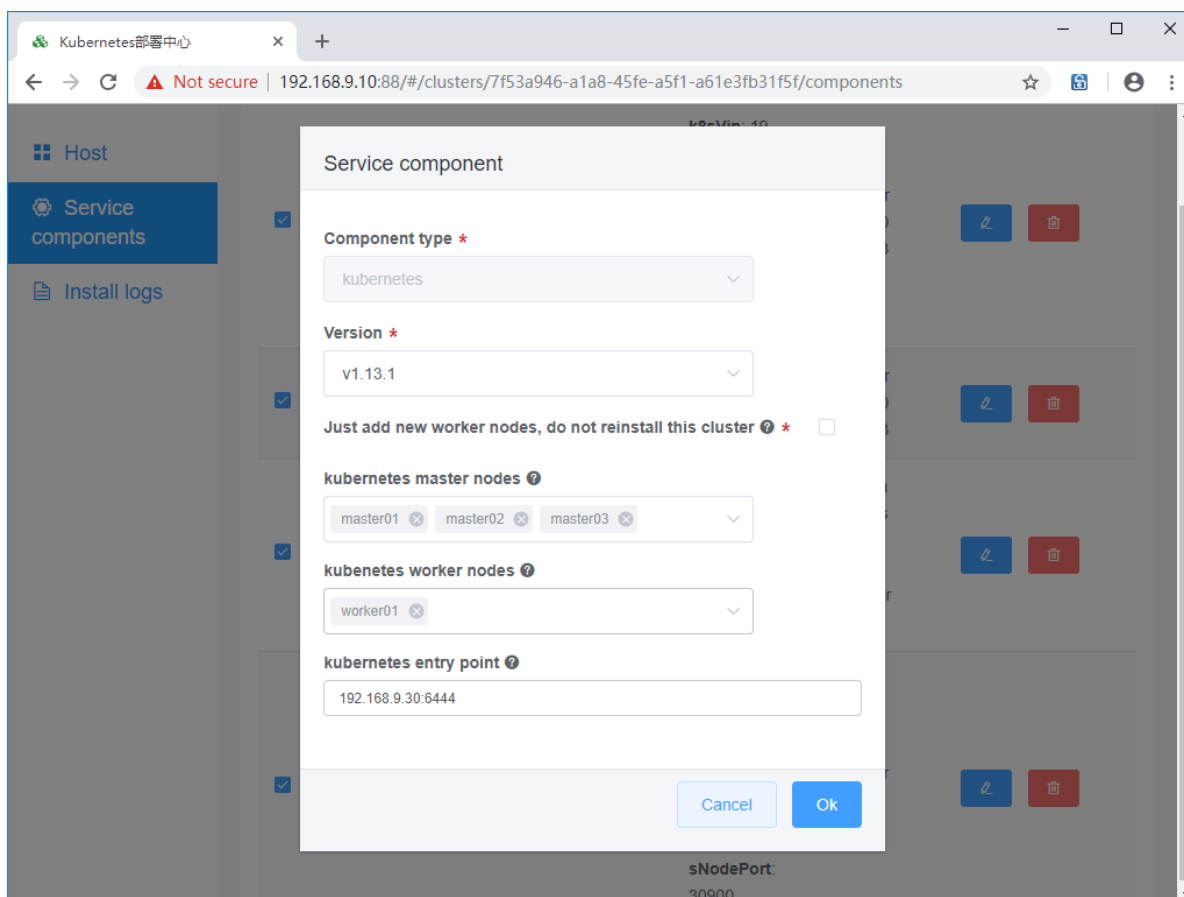
At the bottom of the modal are 'Cancel' and 'Ok' buttons. The footer of the page reads: 'Developer: 睿云智合 (Wise2C) | Contact us | 中文 / EN'.

Then add etcd servers:



Next, add the k8s components: master nodes and worker nodes. (for the first time to install a new cluster, do not check "Just add new worker nodes, do not reinstall this cluster")

Set the correct values for kubernetes entry point: VIP:6444



The last step is optional. If you want to deploy prometheus and alertmanager and grafana, just add this component:

Kubernetes部署中心

192.168.9.10:88/#/clusters/41e18964-d572-47fe-bd64-349963aa21d7/components

Host

Cluster / Alan

Start install

Reset

+ Add component

Service components

Install logs

Number	Service name	Version	Attribute	Host	Operation
1	docker	18.06.1-CE	format_hostname: true	--	<div>🔗</div> <div>🗑</div>
2	harbor	v1.7.0	endpoint: 192.168.9.20 password: Harbor12345	self: harbor	<div>🔗</div> <div>🗑</div>
3	loadbalancer	HAProxy-1.8.14 _Keepalived-1.3.5	k8sVip: 192.168.9.30 netMask: 24 nic: ens33 routerID: 10 vRouterID: 160	self: master01, master02, master03	<div>🔗</div> <div>🗑</div>
4	etcd	3.2.24	--	self: master01, master02, master03	<div>🔗</div> <div>🗑</div>
5	kubernetes	v1.13.1	AddWorkerNodesOnly: false endpoint: 192.168.9.30:6444	master: master01, master02, master03 worker: worker01	<div>🔗</div> <div>🗑</div>
6	prometheus	v2.5.0	AlertManagerNodePort: 30903 GrafanaNodePort: 30902 PrometheusNodePort: 30900	self: worker01	<div>🔗</div> <div>🗑</div>

Last step

Next step

Service component

Component type *

prometheus

Version *

v2.5.0

PrometheusOperator node *

worker01

NodePort for Prometheus *

30900

NodePort for AlertManager *

30903

NodePort for Grafana *

30902

Cancel

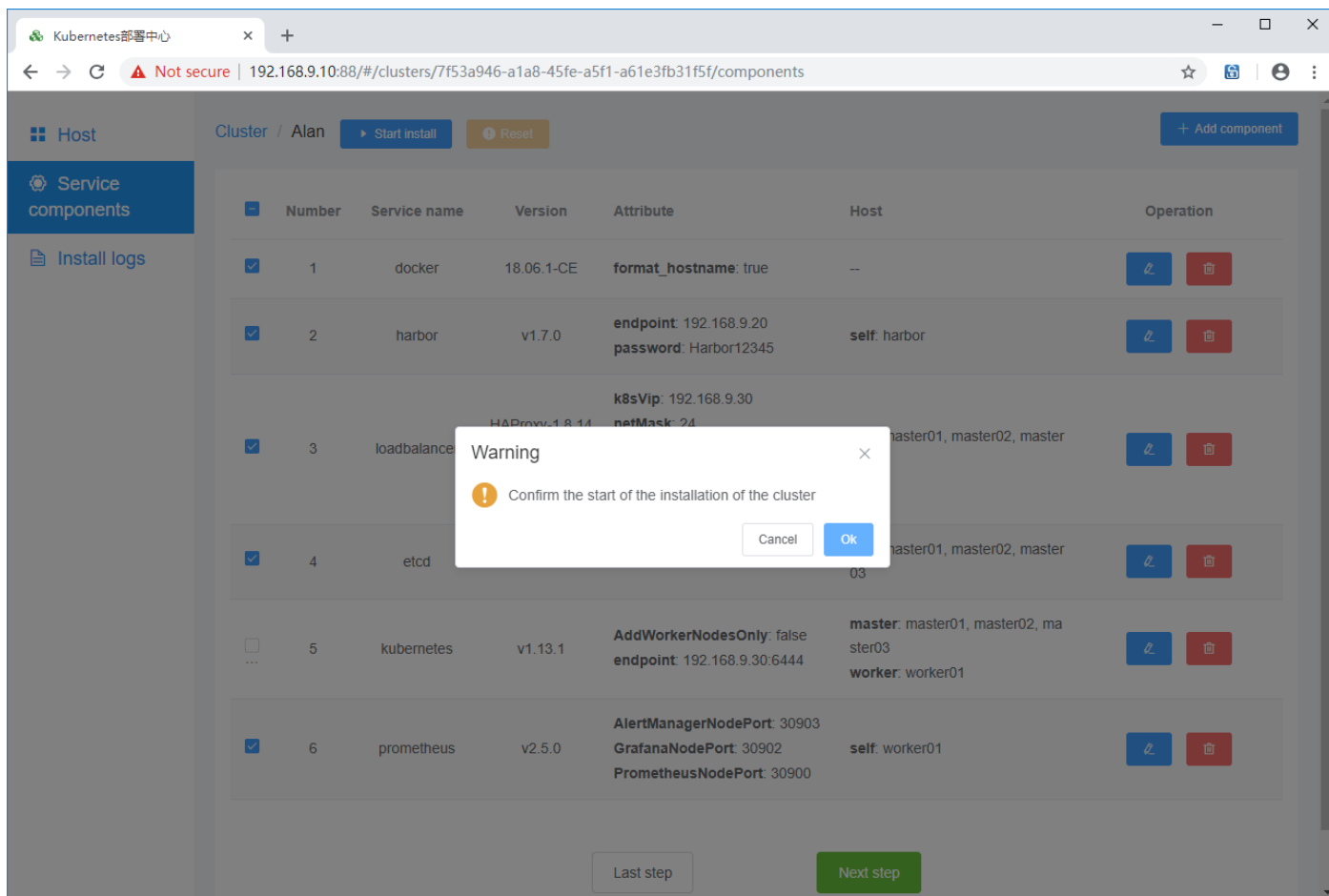
Ok

Developer: 睿云智合 (Wise2C)

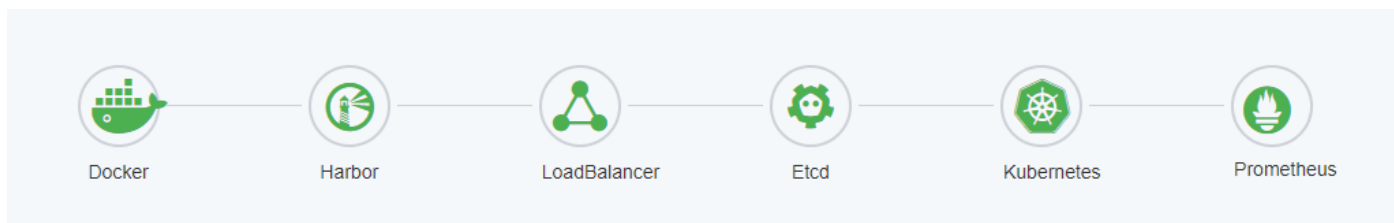
Contact us

中文 / EN

4. Click Next step to start deploy:



Patently waiting for all components icon to turn green which means the deployment is finished without problem.



Verify the cluster health status by commands:

```
kubectl get cs
```

```
kubectl get csr
```

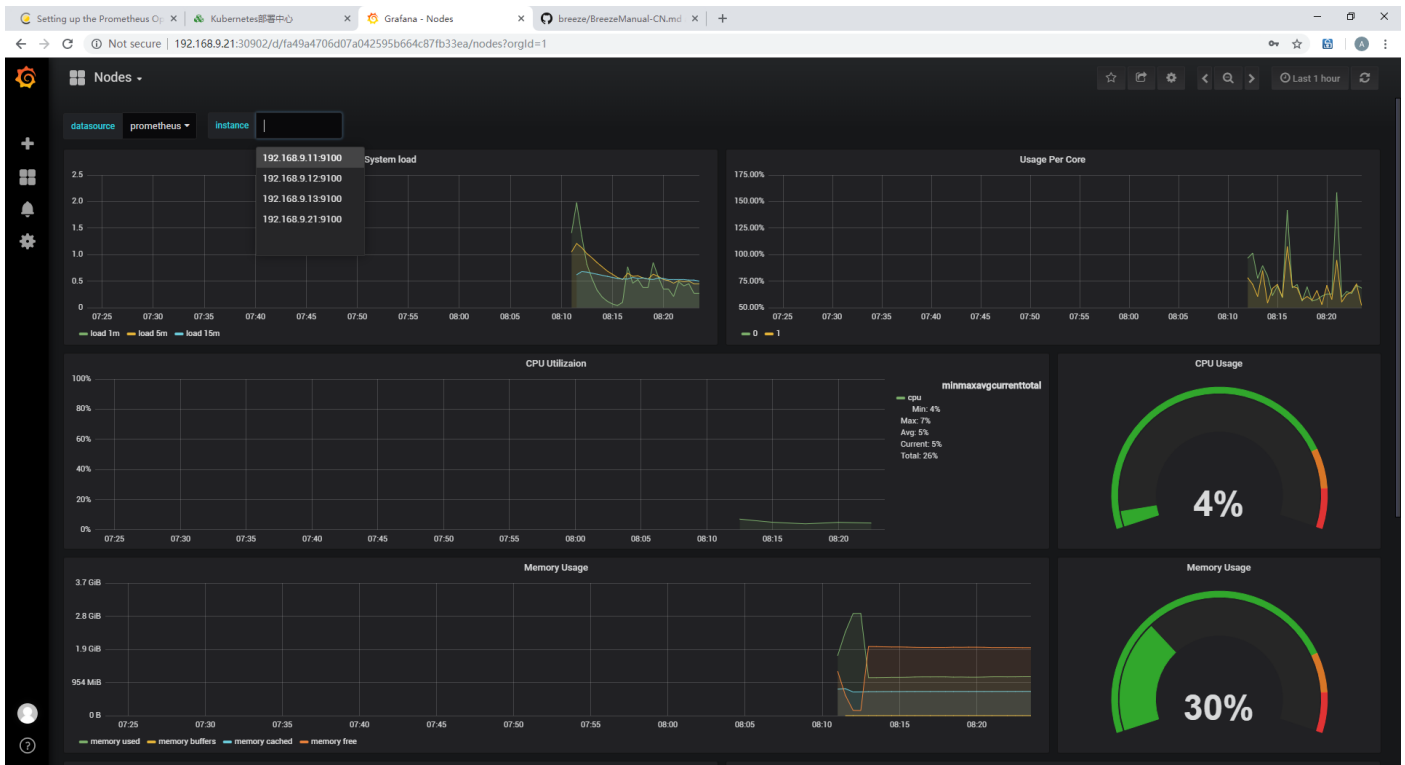
```
kubectl get nodes -o wide
```

```
kubectl -n kube-system get pods
```

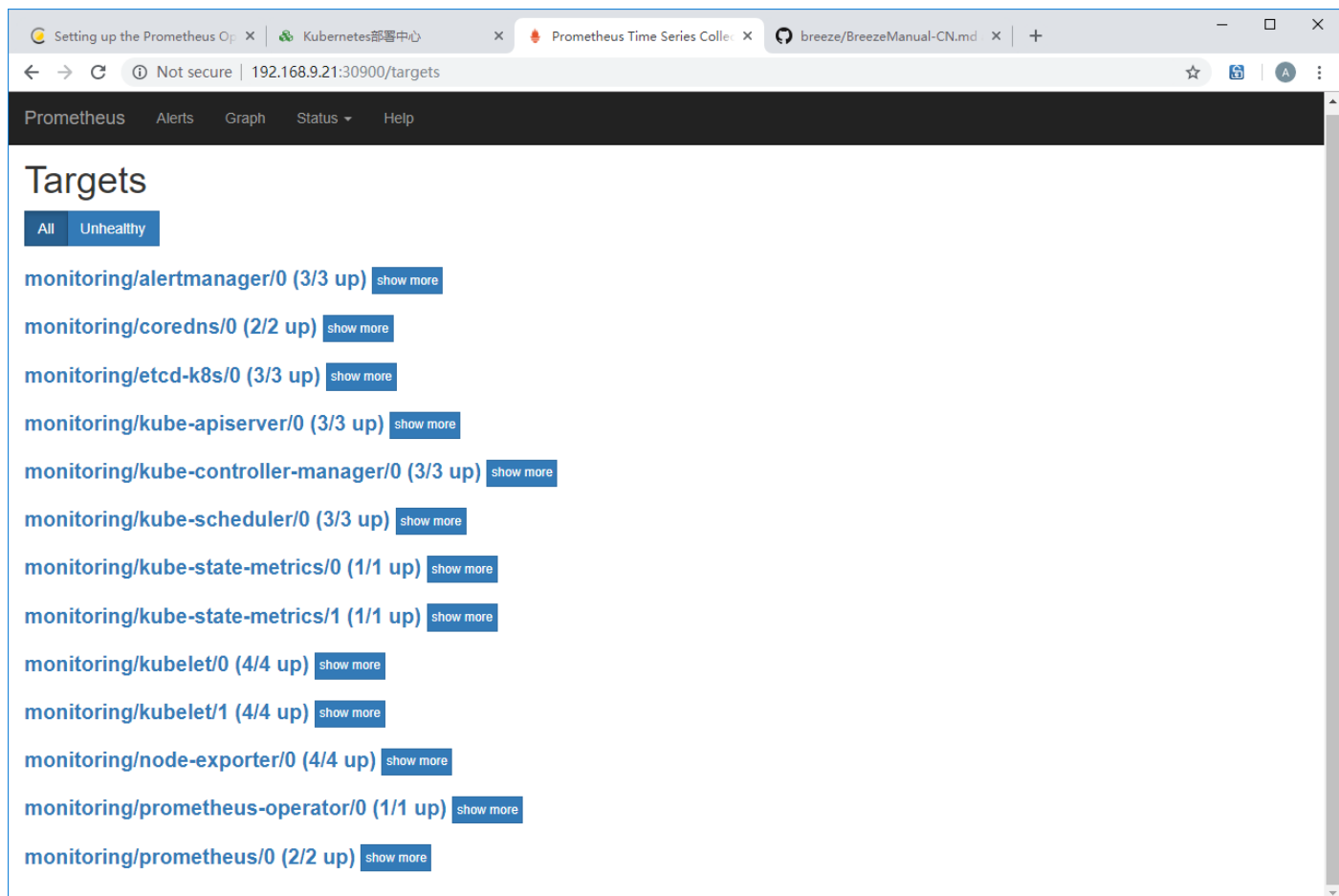
```
kubectl -n monitoring get pods
```

```
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 ~]#
[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 ~]#
[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 ~]#
[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>



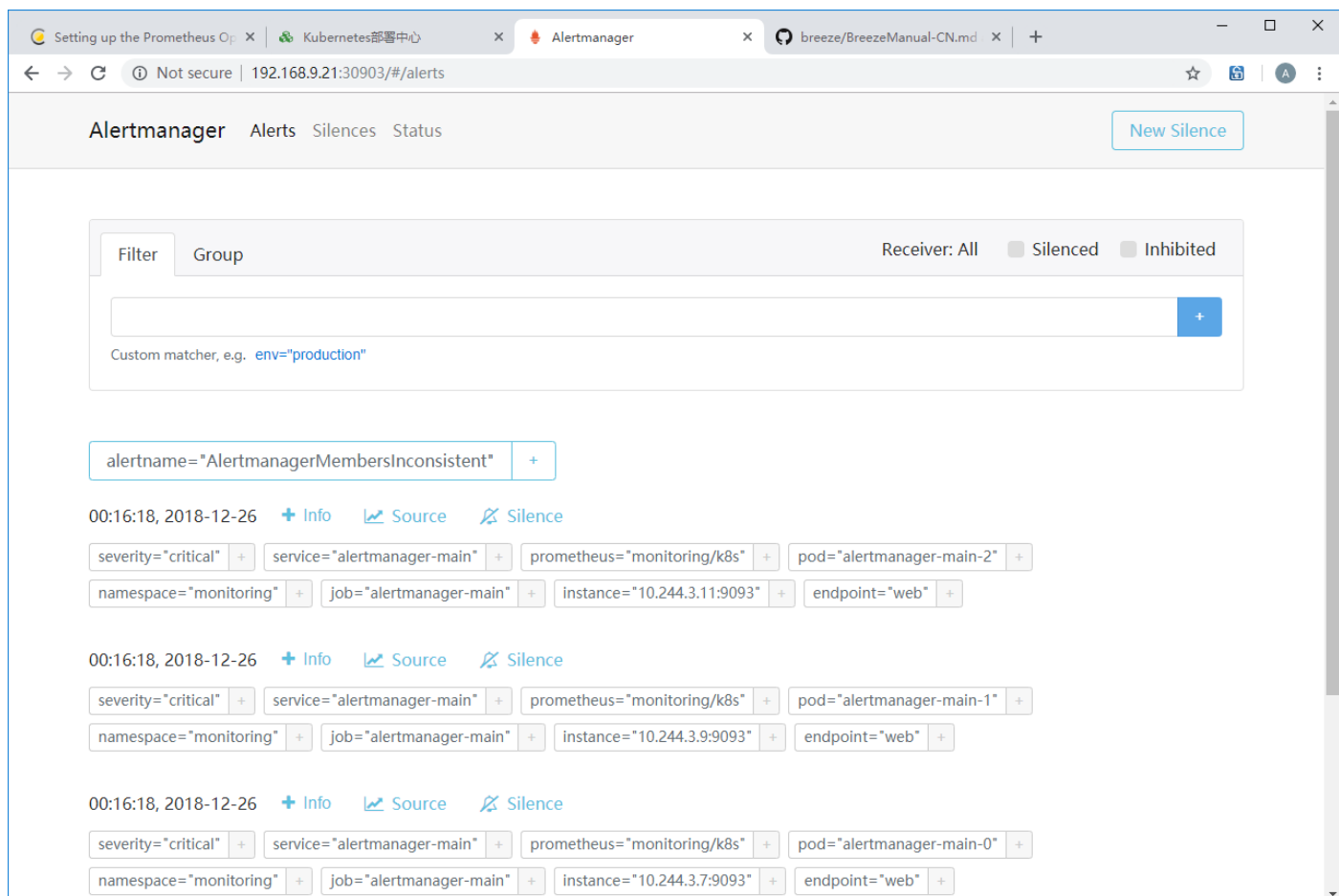
Prometheus: <http://node:30900>



The screenshot shows the Prometheus web interface at the 'Targets' page. The browser address bar indicates the URL is `192.168.9.21:30900/targets`. The page has a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation bar, there are two tabs: 'All' (selected) and 'Unhealthy'. The main content area lists 13 targets, each with a status indicator (e.g., '3/3 up') and a 'show more' link. The targets are:

- monitoring/alertmanager/0 (3/3 up) [show more](#)
- monitoring/coredns/0 (2/2 up) [show more](#)
- monitoring/etcd-k8s/0 (3/3 up) [show more](#)
- monitoring/kube-apiserver/0 (3/3 up) [show more](#)
- monitoring/kube-controller-manager/0 (3/3 up) [show more](#)
- monitoring/kube-scheduler/0 (3/3 up) [show more](#)
- monitoring/kube-state-metrics/0 (1/1 up) [show more](#)
- monitoring/kube-state-metrics/1 (1/1 up) [show more](#)
- monitoring/kubelet/0 (4/4 up) [show more](#)
- monitoring/kubelet/1 (4/4 up) [show more](#)
- monitoring/node-exporter/0 (4/4 up) [show more](#)
- monitoring/prometheus-operator/0 (1/1 up) [show more](#)
- monitoring/prometheus/0 (2/2 up) [show more](#)

Alertmanager: <http://node:30903>



The screenshot shows the Alertmanager web interface at the 'Alerts' page. The browser address bar indicates the URL is `192.168.9.21:30903/#/alerts`. The page has a navigation bar with 'Alertmanager', 'Alerts' (selected), 'Silences', and 'Status'. There is a 'New Silence' button in the top right. Below the navigation bar, there is a filter section with 'Filter' and 'Group' tabs. The 'Filter' tab is active, showing a search bar with a placeholder 'Custom matcher, e.g. env="production"'. Below the search bar, there is a list of alerts. Each alert entry includes a timestamp, a severity level, and a list of labels. The first alert is:

00:16:18, 2018-12-26 [+ Info](#) [Source](#) [Silence](#)

Labels: severity="critical", service="alertmanager-main", prometheus="monitoring/k8s", pod="alertmanager-main-2", namespace="monitoring", job="alertmanager-main", instance="10.244.3.11:9093", endpoint="web".

The second alert is:

00:16:18, 2018-12-26 [+ Info](#) [Source](#) [Silence](#)

Labels: severity="critical", service="alertmanager-main", prometheus="monitoring/k8s", pod="alertmanager-main-1", namespace="monitoring", job="alertmanager-main", instance="10.244.3.9:9093", endpoint="web".

The third alert is:

00:16:18, 2018-12-26 [+ Info](#) [Source](#) [Silence](#)

Labels: severity="critical", service="alertmanager-main", prometheus="monitoring/k8s", pod="alertmanager-main-0", namespace="monitoring", job="alertmanager-main", instance="10.244.3.7:9093", endpoint="web".