Kubernetes和ku8manager

安装、运维手册

目录

[前期准备（for docker） 3](#_Toc450726727)

[一data和metadata 准备 3](#_Toc450726728)

[二 规划服务器用途 3](#_Toc450726729)

[第一部分 手动安装kubernetes 5](#_Toc450726730)

[一 安装etcd 5](#_Toc450726731)

[二 安装docker 6](#_Toc450726732)

[三 安装私库registry 9](#_Toc450726733)

[四 安装kubernetes master 10](#_Toc450726734)

[五 安装 kubernetes nodes 14](#_Toc450726735)

[六 安装skydns（前提：Kubernetes集群安装完毕） 16](#_Toc450726736)

[七 安装heapster（前提：Kubernetes集群安装完毕） 22](#_Toc450726737)

[第二部分 ansible-playbook 命令安装kubernetes 29](#_Toc450726738)

[一 安装ku8manager 29](#_Toc450726739)

[二 kubernetes\_cluster\_setup/hosts操作 31](#_Toc450726740)

[三 kubernetes\_cluster\_setup/pre-setup操作 32](#_Toc450726741)

[四 kubernetes\_cluster\_setup/group\_vars 操作 33](#_Toc450726742)

[五 kubernetes\_cluster\_setup/setup.yml 操作 34](#_Toc450726743)

[六kubernetes\_cluster\_setup/roles 操作 34](#_Toc450726744)

[七 在kubernetes\_cluster\_setup目录下执行 ansible-playbook 完成集群的安装 35](#_Toc450726745)

[八 检查kubernetes集群安装情况 45](#_Toc450726746)

[九 kubernetes\_cluster\_uninstall为卸载脚本 46](#_Toc450726747)

[第三部分Ku8manager 维护手册 49](#_Toc450726748)

[一 initsql.sql 维护 49](#_Toc450726749)

[二ku8manager运行脚本维护 52](#_Toc450726750)

[第四部分 kubernetes网络 53](#_Toc450726751)

[一 georce\_route\_quagga工具 53](#_Toc450726752)

[二 手工配置 54](#_Toc450726753)

[第五部分Kubernetes日志收集和查询 56](#_Toc450726754)

[第六部分 ku8manager多集群安装和维护 57](#_Toc450726755)

[一ku8manager多集群安装 57](#_Toc450726756)

[二ku8manager多集群维护 59](#_Toc450726757)

[编制历史 60](#_Toc450726758)

## 前期准备（for docker）

### 一data和metadata 准备

由于docker需要用到裸盘的两个LV：data和metadata。所以在操作之前先确认LV是否存在。

系统组进行初始Linux安装时，创建docker需要的2个LV。（可以向系统组提要求）。

检查lv,命令lvs/lvdisplay,查看docker的LV 是否存在。如果没有LV ，需要安装人员手动创建，创建步骤如下：

1 df /df -h 找一块大的磁盘，umount 卸载挂载磁盘（如果卸载不成功，lsof 找到进程，干掉再重新卸载。），当然，如果有空余的磁盘可以用，采用新建一个VG,，在创建的VG上创建LV（例：有剩余两块盘sdj和sdk , 创建vg ：vgcreate vg-docker /dev/sdk dev/sdj）。

2 注释已卸载的磁盘在fstab中的开机自动mount点配置项。

vi /etc/fstab

3 如果需要，用lvremove命令删除步骤1卸载的磁盘上的LV

lvremove /dev/rhel/opt

4 创建 lv

创建的时候，data和metadata磁盘空间比列为10:1，metadata 最多分配50G。建议vg剩余小部分空间，不要全部分配给lv（安全策略）。

lvcreate -L 1500G -n data vg-docker

lvcreate -L 30G -n metadata vg-docker

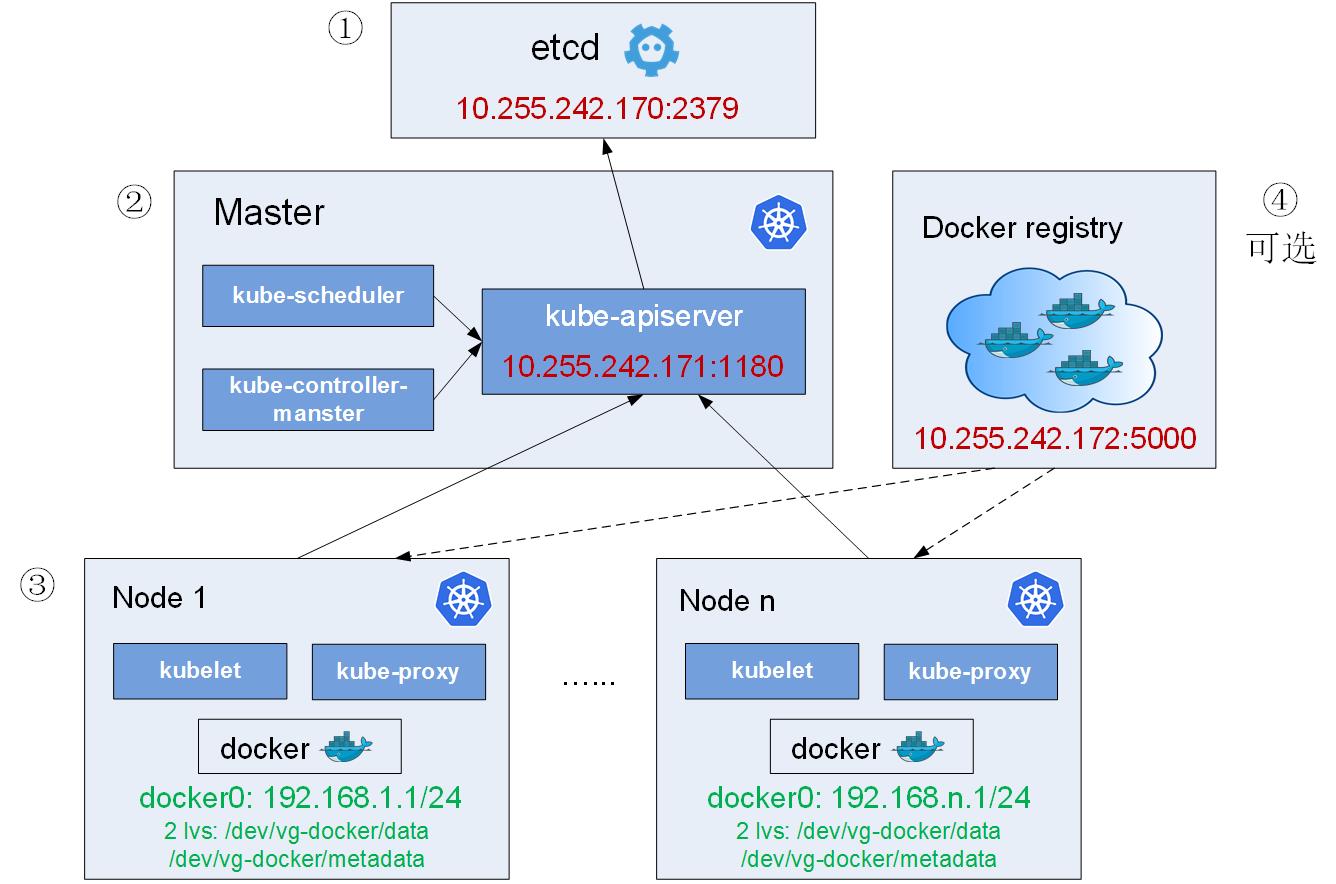
5 创建完成后，检查创建的LV是否已存在。

fdisk -l(查看磁盘) /lvdisplay (查看逻辑卷) /vgdisplay （卷组）

### 二 规划服务器用途

安装kubernetes之前，在准备好的服务器上规划etcd、nodes、master、registry

。哪些服务器安装nodes，哪些服务器安装etcd等，它们之间的一个关系如图所示。



以10.255.242.170集群机器为例：

|  |  |
| --- | --- |
| 主机 | 用途 |
| 10.255.242.170 | etcd1,master |
| 10.255.242.171 | etcd2,registry,ku8manager |
| 10.255.242.172 | etcd3,node |
| 10.255.242.173 | node |
| 10.255.242.174 | node |
| …. | node |

安装步骤概述：

1. etcd：Kubernetes的主数据库

2. master：包括kube-apiserver, kube-controller-manager, kube-scheduler 3个服务

3. nodes：包括kubelet, kube-proxy和docker 3个服务

4. docker registry：docker私库（可选安装），在内网默认需要安装

注：kubernetes的安装有多种方法，下面提供手动安装和使用ansible-playbook 两种方式，安装部署人员，任选其一进行kubernetes安装即可。需要注意的是使用ansib-playbook安装，需要先安装一个docker。

## 第一部分 ansible-playbook 命令安装kubernetes

使用ansible-playbook命令的前提，是ku8manager 已经安装完成并启动后，进入到容器中去执行ansible-playbook 命令。

先安装ku8manager。

### 一 安装ku8manager

安装ku8manager前提，docker已安装。

Ku8manager 项目需要用到三个文件夹，两个镜像，两个文件，一个jar包。下面Ku8manager运行脚本里有详细介绍。

#### 1 cp 文件

将ku8manager相关文件拷贝打要安装物理机。

（scp -rp 10.255.242.170:/home/ecip/ku8eye /root）

#### 2镜像操作

Dokcer load –i busybox.tar.gz

Docker tag b175bcb79023 busybox

docker load -i ku8manager-env.tar.gz

docker images

docker tag 5d9ca3091b03 ku8manager

docker images

#docker images

#### 3 vi ku8manager和busybox运行脚本

查看ku8 运行脚本，-v 物理机路径和容器路径，以及映射文件个数。

170上run\_ku8manager.sh

#cat run\_ku8manager.sh

docker run -tid --name ku8manager \

-p 3306:3306 --volumes-from=mariadb-data \

-v /var/run/docker.sock:/var/run/docker.sock \

-v ${PWD}/ku8eye-web-0.0.2-SNAPSHOT.jar:/ku8manager/ku8manager.jar \

-v ${PWD}/initsql.sql:/ku8manager/db\_scripts/initsql.sql \

-v ${PWD}/ku8\_ext\_files:/ku8\_ext\_files \

-v ${PWD}/kubernetes\_cluster\_setup:/ku8manager/kubernetes\_cluster\_setup \

-v ${PWD}/kubernetes\_cluster\_uninstall:/ku8manager/kubernetes\_cluster\_uninstall \

-p 1199:8080 ku8manager

参数说明：

docker run -tid --name ku8manager

-p 3306:3306 --volumes-from=mariadb-data：参见“如何持久化mysql数据到volume”

-v /var/run/docker.sock:/var/run/docker.sock #映射docker.sock文件

-v ${PWD}/ku8eye-web-0.0.2-SNAPSHOT.jar:/ku8manager/ku8manager.jar #映射jar包

-v ${PWD}/initsql.sql:/ku8manager/db\_scripts/initsql.sql #映射sql

-v ${PWD}/ku8\_ext\_files:/ku8\_ext\_files # 映射文件

-v ${PWD}/kubernetes\_cluster\_setup:/ku8manager/kubernetes\_cluster\_setup

-v ${PWD}/kubernetes\_cluster\_uninstall:/ku8manager/kubernetes\_cluster\_uninstall

-p 1199:8080 ku8manager # 启动端口1199

#cat **run\_busybox.sh**

docker run --name=mariadb-data -e TZ="Asia/Shanghai" -v /var/lib/mysql busybox true

busybox 脚本不需要做修改

#### 4 启动ku8manager

ku8manager启动时，mysql 数据库init 数据要持久化存放，所以首先要启动busybox容器。

#./run\_busybox.sh

# ./run\_ku8manager.sh

#### 5 检查ku8manager

# docker ps –a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

cd67d7508bea ku8manager "/ku8manager/docker-e" 3 hours ago Up 3 hours 0.0.0.0:3306->3306/tcp, 0.0.0.0:1199->8080/tcp ku8manager

a92a9180f7cf busybox "true" 6 days ago Exited (0) 6 days ago mariadb-data

52cd37353e43 774242a00f13 "/bin/registry /etc/d" 3 weeks ago Up 3 weeks 0.0.0.0:5000->5000/tcp registry使

#### 6 验证ku8manager

进入ku8manager容器进行验证。

# docker exec -ti ku8manager bash

进入容器后，查看tomcat.log 日志，看到有“ku8 app started completed”等字样后，打开谷歌浏览器，输入ku8 的ip:port 回车，进入ku8manager登录界面，输入用户名和密码，点击登录。如果能正常登陆，到这里，说明ku8manager安装完成。

### 二 kubernetes\_cluster\_setup/hosts操作

Ku8manager安装完成后，我们知道kubernetes\_cluster\_setup 已经挂载到容器内，接下来操作都是在容器内。

进入ku8manager容器，kubernetes\_cluster\_setup目录。修改各个文件。

Hosts文件用于存各待安装Kubernetes集群的服务器的分组与IP地址。

# machines list in group

[docker-registry]

10.255.242.171 ansible\_ssh\_user=root ansible\_ssh\_pass=7F%Ah@6fM

[etcd]

192.168.110.135 ansible\_ssh\_user=root ansible\_ssh\_pass=password etcd\_name=etcd1 etcd\_data\_dir="/var/etcd\_data" etcd\_server\_ip=http://192.168.110.135 etcd\_initial\_cluster="etcd1=http://192.168.110.135:2380,etcd2=http://192.168.110.136:2380,etcd3=http://192.168.110.137:2380"

[kube-master]

10.255.242.171 ansible\_ssh\_user=root ansible\_ssh\_pass=7F%Ah@6fM

# for docker, different docker0's IP on different node

# and kubelet, kube-proxy

[kube-node]

10.255.242.173 ansible\_ssh\_user=root ansible\_ssh\_pass=7F%Ah@6fM docker0\_ip=193.168.1.1/24 kubelet\_hostname\_override=10.255.242.173

install\_quagga\_router=true

10.255.242.174 ansible\_ssh\_user=root ansible\_ssh\_pass=7F%Ah@6fM docker0\_ip=193.168.2.1/24 kubelet\_hostname\_override=10.255.242.174

install\_quagga\_router=true

10.255.242.175 ansible\_ssh\_user=root ansible\_ssh\_pass=7F%Ah@6fM docker0\_ip=193.168.3.1/24 kubelet\_hostname\_override=10.255.242.175

install\_quagga\_router=true

[kube-node] 参数说明：

docker0\_ip=193.168.16.1/24 docker0网桥的IP地址，每个node上设置为不同的IP地址。

kubelet\_hostname\_override=192.168.1.202 kubelet主机名 。

install\_quagga\_router=true 是否安装Quagga路由器（docker容器）

188和189主机 可以看到多了docker\_dm\_datadev 内容。如果将docker\_dm\_datadev 路径写在这，就以这的写入为准。如果没有加docker\_dm\_datadev，则一全局的\*//roles/kube-node/defaults/main.xml 为准。main.xml 下面再说。

注：docker0 写法讲究

1.对于安装registry和ku8manager的机器 docker0命名为192.168.254.1/24和192.168.253.1/24 为降序排。

2.对于安装node的机器 ，docker0 命名192.168.1.1/24，192.168.2.1/24......升序排

3.对于第二次重装kubernet 不成功的原因是 docker0的IP 被第一次安装占用，不能再次写入固定的docker0，这时候，需要重启机器（reboot），重启完之后的多次重装，就都可以成功。

### 三 kubernetes\_cluster\_setup/pre-setup操作

这里可以通过ansible-playbook 关闭机器防火墙，也可以手动关闭。

disablefirewalld.yml文件内容如下：

- hosts: all

tasks:

- name: stop firewalld service

command: systemctl stop firewalld

- name: disable firewalld service

command: systemctl disable firewalld

### 四 kubernetes\_cluster\_setup/group\_vars 操作

在/kubernetes\_cluster\_setup目录中创建group\_vars子目录存放全局变量

默认文件为all.yml，内容如下：

# global variables

dns\_setup: false

######## IMPORTANT - dns IP must be in kube-apiserver --service-cluster-ip-range RANGE ########

# these 2 variables will be used in kubelet config file and skydns-rc.yaml file

cluster\_domain\_name: cluster.local

cluster\_dns\_ip: 169.169.0.100

# docker private registry server name and IP address

docker\_registry\_server\_name: 10.255.242.171

docker\_registry\_server\_ip: 10.255.242.171

docker\_registry\_host\_port: 5000

docker\_registry\_url: "{{docker\_registry\_server\_name}}:{{ docker\_registry\_host\_port }}"

# push google\_container/pause image to docker private registry

push\_pause\_image: true

kubernetes\_pause\_image\_id: 8950680a606c

kubernetes\_pause\_image\_tag: "{{docker\_registry\_url}}/google\_containers/pause:2.0"

说明：

cluster\_domain\_name: cluster.local kube-dns #服务设置的domain名

cluster\_dns\_ip: 169.169.0.100 kube-dns #服务IP地址（需在kube-apiserver的apiserver\_service\_cluster\_ip\_range范围内）

docker\_registry\_server\_name: yourcompany.com docker registry#主机IP地址

docker\_registry\_server\_ip: 10.255.242.171 docker registry #主机IP地址

push\_pause\_image: true #是否将 Kubernetes pause 镜像push到 docker registry

kubernetes\_pause\_image\_id: 8950680a606c pause#镜像ID需要根据实际情况填写

kubernetes\_pause\_image\_tag: "{{docker\_registry\_url}}/google\_containers/pause" #pause镜像在 docker registry 的URL

### 五 kubernetes\_cluster\_setup/setup.yml 操作

在/kubernetes\_cluster\_setup目录中创建setup.yml，内容为ansible-playbook在各hosts安装role的配置：

setup.yml内容如下：

---

hosts: docker-registry

roles:

- docker-registry

- hosts: etcd

roles:

- etcd

- hosts: kube-master

roles:

- kube-

- hosts: kube-node

roles:

- kube-node

注：setup.yml按次序执行安装registry，etcd，master，node，如果只安装其中的某一部分，可以将其余部分通过注释的办法不予安装。

### 六kubernetes\_cluster\_setup/roles 操作

/kubernetes\_cluster\_setup/roles 下面会有四个目录，依次是：docker-registry，etcd，kube-master和kube-node。每个目录下面又会有四个文件夹，分别是：defaults， files，tasks，templates。

这个几个文件结构之间的运作关系：

defaults：存放变量的值，目录下main.yml文件将被ansible默认读取。

files：存放需安装（复制）的源文件。

tasks：ansible-playbook执行的任务脚本，目录下main.yml文件将被ansible默认读取。 templates：需修改参数的配置文件，其中参数由defaults目录中的值进行替换。

每次操作roles,只需要修改defaults\main.yml文件

下面是对几个相应的main.yml文件说明：

1） roles\docker-registry\defaults\main.yml

docker\_registry\_root\_dir: /hadoop1/docker\_registry docker registry 运行目录

docker\_registry\_image\_id: 774242a00f13 docker registry 镜像ID

docker\_registry\_image\_tag: registry:2.2.0 docker registry 镜像tag

2）roles\etcd\defaults\main.yml

peer\_ip: 10.255.242.171 etcd所在主机的IP地址（cluster配置时使用）

etcd\_data\_dir: /var/lib/etcd/etcd\_data etcd数据存储目录

3） roles\kube-master\defaults\main.yml

# for kube-apiserver

etcd\_servers: http://10.255.242.171:2379,http://10.255.242.170:2379,http://10.255.242.172:2379 #kube-apiserver所需etcd服务的URL

apiserver\_insecure\_port: 1180 #kube-apiserver监听的非安全端口号

apiserver\_service\_cluster\_ip\_range: 169.169.0.0/16 #Kubernetes Services可分配IP地址池

apiserver\_service\_node\_port\_range: 1000-65535 #NodePort 类型的 Service 可用端口范围，含两端

# for kube-controller-manager, kube-scheduler

kube\_master\_url: http://10.255.242.171:1180 #kube-apiserver服务URL

kube\_node\_sync\_period: 10s #master与node信息同步时间间隔

# to generate ssh keys on master server

ca\_crt\_CN: ecip.hp.com #master ssh key内CA证书中CN参数

server\_key\_CN: 10.255.242.171 #master ssh key内CN参数

4）roles\kube-node\defaults\main.yml

- for kubelet, kube-proxy

kube\_master\_url: http://http://10.255.242.171:1180 kube-apiserver服务URL

-- quagga router docker image info

quagga\_router\_image\_id: f96cfe685533 quagga router 镜像ID

quagga\_router\_image\_tag: index.alauda.cn/georce/router quagga router 镜像tag

### 七 在kubernetes\_cluster\_setup目录下执行 ansible-playbook 完成集群的安装

$ cd kubernetes\_cluster\_setup

在所有配置都修改完成后，执行ansible-playbook 命令开始安装：

$ ansible-playbook -i hosts setup.yml

192.168.1.200安装日志：

PLAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.200]

TASK [docker-registry : copy docker to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.200]

TASK [docker-registry : copy config file docker to /etc/sysconfig] \*\*\*\*\*\*\*

ok: [192.168.1.200]

TASK [docker-registry : make docker runtime root directory /hadoop1/docker

ok: [192.168.1.200]

TASK [docker-registry : copy docker.socket to /usr/lib/systemd/system] \*\*\*

ok: [192.168.1.200]

TASK [docker-registry : copy docker.service to /usr/lib/systemd/system] \*\*

ok: [192.168.1.200]

TASK [docker-registry : systemctl daemon-reload] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : restart docker.socket service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : enable docker service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.200]

TASK [docker-registry : restart docker service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : check docker.socket service started ok] \*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : check docker service started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : check if registry image exists] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : copy registry image file to /tmp] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

TASK [docker-registry : load registry image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

TASK [docker-registry : tag registry image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

TASK [docker-registry : check if registry container exists] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : start registry container] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : create running script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

TASK [docker-registry : start docker registry container] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

TASK [docker-registry : check docker registry started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.200]

TASK [docker-registry : copy pause image file to /tmp] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.200]

TASK [docker-registry : check if pause image exists] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.200]

TASK [docker-registry : load pause image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

TASK [docker-registry : tag pause image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

TASK [docker-registry : push pause image to private registry] \*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.200]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.1.200 : ok=18 changed=9 unreachable=0 failed=0

PLAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : copy etcd to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : copy etcdctl to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : make dir /etc/etcd] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : copy config file etcd.conf to /etc/etcd] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : copy etcd.service to /usr/lib/systemd/system] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : make dir /var/lib/etcd/ for etcd service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : systemctl daemon-reload] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [etcd : enable etcd service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [etcd : start etcd service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [etcd : test etcd started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

PLAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy kube-apiserver to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy kube-controller-manager to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy kube-scheduler to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy kubectl to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy hyperkube to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : make dir /var/run/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : create keys - ca.key] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : create keys - ca.crt] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : create keys - server.key] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : create keys - server.csr] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : create keys - server.crt] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : make dir /etc/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy config file apiserver to /etc/kubernetes] \*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy config file config to /etc/kubernetes] \*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy config file controller-manager to /etc/kubernetes] \*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy config file scheduler to /etc/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy kube-apiserver.service to /usr/lib/systemd/system] \*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy kube-controller-manager.service to /usr/lib/systemd/system] \*\*\*

ok: [192.168.1.201]

TASK [kube-master : copy kube-scheduler.service to /usr/lib/systemd/system] \*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : systemctl daemon-reload] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : make dir /var/log/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : enable kube-apiserver service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : start kube-apiserver service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : enable kube-controller-manager service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : start kube-controller-manager service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : enable kube-scheduler service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.201]

TASK [kube-master : start kube-scheduler service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : check kube-apiserver service started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : check kube-controller-manager service started ok] \*\*\*\*\*

changed: [192.168.1.201]

TASK [kube-master : check kube-scheduler service started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.201]

PLAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy docker to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.202]

ok: [192.168.1.203]

TASK [kube-node : copy config file docker to /etc/sysconfig] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : make docker runtime root directory /hadoop1/docker] \*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy docker.socket to /usr/lib/systemd/system] \*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy docker.service to /usr/lib/systemd/system] \*\*\*\*\*\*\*\*\*

ok: [192.168.1.202]

ok: [192.168.1.203]

TASK [kube-node : systemctl daemon-reload] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

TASK [kube-node : start docker.socket service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

TASK [kube-node : enable docker service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : start docker service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.202]

changed: [192.168.1.203]

TASK [kube-node : check docker.socket service started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.202]

changed: [192.168.1.203]

TASK [kube-node : check docker service started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

TASK [kube-node : install\_quagga\_router - delete all docker containers] \*\*\*

skipping: [192.168.1.202]

skipping: [192.168.1.203]

TASK [kube-node : install\_quagga\_router - delete all docker images] \*\*\*\*\*\*\*

skipping: [192.168.1.202]

skipping: [192.168.1.203]

TASK [kube-node : install\_quagga\_router - copy Quagga image to /tmp] \*\*\*\*\*\*

skipping: [192.168.1.202]

skipping: [192.168.1.203]

TASK [kube-node : install\_quagga\_router - load Quagga image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.202]

skipping: [192.168.1.203]

TASK [kube-node : install\_quagga\_router - tag Quagga image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [192.168.1.202]

skipping: [192.168.1.203]

TASK [kube-node : install\_quagga\_router - start Quagga container] \*\*\*\*\*\*\*\*\*

skipping: [192.168.1.202]

skipping: [192.168.1.203]

TASK [kube-node : copy kubelet to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy kube-proxy to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy kubectl to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.202]

ok: [192.168.1.203]

TASK [kube-node : copy hyperkube to /usr/bin] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : make dir /var/run/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.202]

ok: [192.168.1.203]

TASK [kube-node : make dir /etc/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy config file kubelet to /etc/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy config file config to /etc/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.202]

ok: [192.168.1.203]

TASK [kube-node : copy config file proxy to /etc/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : copy kubelet.service to /usr/lib/systemd/system] \*\*\*\*\*\*\*\*

ok: [192.168.1.202]

ok: [192.168.1.203]

TASK [kube-node : copy kube-proxy.service to /usr/lib/systemd/system] \*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : make dir /var/lib/kubelet] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : systemctl daemon-reload] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

TASK [kube-node : make dir /var/log/kubernetes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : enable kubelet service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.203]

ok: [192.168.1.202]

TASK [kube-node : start kubelet service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

TASK [kube-node : enable kube-proxy service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.1.202]

ok: [192.168.1.203]

TASK [kube-node : start kube-proxy service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

TASK [kube-node : check kubelet service started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

TASK [kube-node : check kube-proxy service started ok] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.1.203]

changed: [192.168.1.202]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.1.201 : ok=42 changed=15 unreachable=0 failed=0

192.168.1.202 : ok=32 changed=10 unreachable=0 failed=0

192.168.1.203 : ok=32 changed=10 unreachable=0 failed=0

### 八 检查kubernetes集群安装情况

检查kubernetes集群安装情况，从master机器上，查看所有的 nodes

#kubectl -s 10.255.242.171:1180 get nodes

NAME LABELS STATUS AGE

10.255.242.173 group=ummp,kubernetes.io/hostname=10.255.242.173 Ready 12d

10.255.242.174 group=ummp,kubernetes.io/hostname=10.255.242.174 Ready 12d

10.255.242.175 group=ummp,kubernetes.io/hostname=10.255.242.175 Ready 12d

10.255.242.176 group=ummp,kubernetes.io/hostname=10.255.242.176 Ready 12d

10.255.242.177 group=ummp,kubernetes.io/hostname=10.255.242.177 Ready 12d

10.255.242.178 group=ummp,kubernetes.io/hostname=10.255.242.178 Ready 12d

10.255.242.179 group=ummp,kubernetes.io/hostname=10.255.242.179 Ready 12d

10.255.242.180 group=ummp,kubernetes.io/hostname=10.255.242.180 Ready 12d

10.255.242.181 group=ummp,kubernetes.io/hostname=10.255.242.181 Ready 12d

10.255.242.182 group=ummp,kubernetes.io/hostname=10.255.242.182 Ready 12d

以上第二部分使用ansible-playbook 命令安装kubernet集群的操作，可以参考

<https://github.com/bestcloud/ku8eye/blob/master/doc/ku8eye-web-dev-env.md>

### 九 kubernetes\_cluster\_uninstall为卸载脚本

同样使用ansible-playbook完成卸载操作。

$ cd kubernetes\_cluster\_uninstall

$ ansible-playbook -i hosts setup.yml

这里以卸载kube-node为例（卸载四川44机器）

kubernetes\_cluster\_uninstall]# ansible-playbook -i hosts setup.yml

PLAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [10.254.90.44]

TASK [kube-node : stat] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [10.254.90.44]

TASK [kube-node : stop and disable docker service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : delete docker0 bridge] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : stop and disable kubelet service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : stop and disable kube-proxy service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [10.254.90.44]

TASK [kube-node : file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fatal: [10.254.90.44]: FAILED! => {"changed": false, "failed": true, "msg": "rmtree failed: [Errno 16] Device or resource busy: '/var/lib/kubelet/pods/9263ca44-d5f2-11e5-91e3-d0431ef9f6fe/volumes/kubernetes.io~secret/default-token-siwpd'"}

...ignoring

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10.254.90.44 : ok=19 changed=16 unreachable=0 failed=0

[root@psn280d2s ~]# kk get nodes

看到44机器 status 变为NotReady

NAME LABELS STATUS AGE

10.254.90.32 kubernetes.io/hostname=10.254.90.32 Ready 20d

10.254.90.33 kubernetes.io/hostname=10.254.90.33 Ready 21d

10.254.90.34 kubernetes.io/hostname=10.254.90.34 Ready 21d

10.254.90.35 kubernetes.io/hostname=10.254.90.35 Ready 21d

10.254.90.36 kubernetes.io/hostname=10.254.90.36 Ready 21d

10.254.90.37 kubernetes.io/hostname=10.254.90.37 Ready 21d

10.254.90.38 kubernetes.io/hostname=10.254.90.38 Ready 21d

10.254.90.39 kubernetes.io/hostname=10.254.90.39 Ready 21d

10.254.90.40 kubernetes.io/hostname=10.254.90.40 Ready 21d

10.254.90.41 kubernetes.io/hostname=10.254.90.41 Ready 21d

10.254.90.42 kubernetes.io/hostname=10.254.90.42 Ready 21d

10.254.90.43 kubernetes.io/hostname=10.254.90.43 Ready 21d

10.254.90.44 kubernetes.io/hostname=10.254.90.44 NotReady 21d

## 第二部分Ku8manager 维护手册

前提：ku8manager容器正确启动。

打开Chrome浏览器（目前只支持Chrome浏览器），输入主机ip和容器端口

http://10.255.242.170:1199/ 打开页面，登录即可。

目前ku8manger用到的11张表如下所示。

|  |  |
| --- | --- |
| 表名 | 说明 |
| Host | 主机列表 |
| Ku8\_project | 创建的应用的数据 |
| ku8\_cluster | 集群列表 |
| ku8\_res\_partion | 资源分区 |
| ku8\_service | 服务列表 |
| ku8\_service\_template | 服务模板 |
| ku8s\_srv\_endpoint | 终端，存放master，infuxDB,register |
| docker\_image | imager 信息列表 |
| tenant | 租户信息 |
| user | 用户信息表 |
| user\_group | 用户组表 |
| zone | 地域信息表 |

### 一 initsql.sql 维护

Ku8manager目前用到的表有docker\_imager,host,ku8\_cluster,ku8\_project,ku8\_res\_partion,

ku8\_service,ku8\_service\_template,ku8s\_srv\_endpoint,tenant,user,user\_group,zone，共12张表。

mysql数据库初始化的时，需要host，ku8s\_srv\_endpoint，user,user\_group这四张表有初始数据。docker\_imager表不需要初始数据。所以，从工具svn上 down下来的initsql.sql 需要对这几张张表做维护。

#### 1 host表

Host表用于存储主机信息，表结构如下：

`ID` int(11) NOT NULL AUTO\_INCREMENT COMMENT 'primary key',

`ZONE\_ID` int(11) DEFAULT NULL COMMENT 'belongs which zone ',

`HOST\_NAME` char(32) NOT NULL COMMENT 'host name ',

`IP` char(32) NOT NULL COMMENT 'ip addr',

`ROOT\_PASSWD` char(16) DEFAULT NULL COMMENT 'root password ',

`LOCATION` varchar(128) DEFAULT NULL COMMENT 'host location ,etc room ',

`NOTE` varchar(256) DEFAULT NULL COMMENT 'note for this record',

`LAST\_UPDATED` timestamp NULL DEFAULT NULL ON UPDATE CURRENT\_TIMESTAMP COMMENT 'last updated time',

`CORES` smallint(6) DEFAULT NULL,

`MEMORY` mediumint(6) DEFAULT NULL,

`USAGE\_FLAG` tinyint(4) DEFAULT NULL,

`SSH\_LOGIN` tinyint(4) DEFAULT NULL,

`CLUSTER\_ID` int(11) DEFAULT NULL,

Inser操作的时候，需要注意的字段有ID，为主键 ,HOST\_NAME，填写主机IP，IP，ROOT\_PASSWD，填写root用户的 password；

INSERT INTO `host` VALUES ('1', '1', '10.254.90.31', '10.254.90.31', 'calvin', 'sichuan', null, '2015-11-24 15:20:04', '4', '524288', '0', '0', '0');

INSERT INTO `host` VALUES ('2', '1', '10.254.90.32', '10.254.90.32', 'calvin', 'sichuan', null, '2015-11-24 15:20:53', '8', '524288', '0', '0', '0');

INSERT INTO `host` VALUES ('3', '1', '10.254.90.33', '10.254.90.33', 'calvin', 'sichuan', null, '2015-11-24 15:20:18', '16', '1048576', '0', '0', '0');

INSERT INTO `host` VALUES ('4', '1', '10.254.90.34', '10.254.90.34', 'calvin', 'sichuan', null, '2015-11-24 15:20:04', '4', '524288', '0', '0', '0');

INSERT INTO `host` VALUES ('5', '1', '10.254.90.35', '10.254.90.35', 'calvin', 'sichuan', null, '2015-11-24 15:20:53', '8', '524288', '0', '0', '0');

INSERT INTO `host` VALUES ('6', '1', '10.254.90.36', '10.254.90.36', 'calvin', 'sichuan', null, '2015-11-24 15:20:18', '16', '1048576', '0', '0', '0');

#### 2.ku8s\_srv\_endpoint 表

ku8s\_srv\_endpoint表存放master、influxDB和registry信息，表结构如下：

`ID` int(11) NOT NULL AUTO\_INCREMENT COMMENT 'primary key',

`NODE\_ROLE` tinyint(4) DEFAULT NULL COMMENT 'kubernetes node role(etcd,master ,node,docker registry ) ',

`SERVICE\_TYPE` tinyint(4) DEFAULT NULL COMMENT 'service type ',

`CLUSTER\_ID` int(11) DEFAULT NULL COMMENT 'belong to which cluster ',

`HOST\_ID` int(11) DEFAULT NULL COMMENT ' at witch host ',

`SERVICE\_URL` varchar(64) DEFAULT NULL COMMENT 'service url',

`SERVICE\_STATUS` tinyint(4) DEFAULT NULL COMMENT 'service status (ok ,bad)',

`NOTE` varchar(256) DEFAULT NULL COMMENT 'note for this record',

`SSH\_PORT` int(5) DEFAULT NULL,

`SSH\_HOST` varchar(20) DEFAULT NULL,

`SSH\_PASS` varchar(50) DEFAULT NULL,

`SSH\_USER` varchar(50) DEFAULT NULL,

`LAST\_UPDATED` timestamp NULL DEFAULT NULL ON UPDATE CURRENT\_TIMESTAMP COMMENT 'last updated time',

目前规定，字段CLUSTER\_ID=’0’,存master，端口固定为1180，SSH\_PORT为22；字段CLUSTER\_ID=’0’ 并且 SERVICE\_TYPE=’3’ ，存的是registry信息；influxDB 端口为1192。

---master

INSERT INTO `ku8s\_srv\_endpoint` VALUES (0, 1, 1, 0, '10.254.90.32', 'http://10.254.90.32:1180', 1, NULL,’22’, '10.254.110.31', '123456', 'root', NULL);

--register

INSERT INTO `ku8s\_srv\_endpoint` VALUES (0, 1, 3, 0, '10.254.90.32', 'http://10.254.90.31:5000, 1, NULL, NULL, '10.254.110.31', '123456', 'root', NULL);

--influxDB

INSERT INTO `ku8s\_srv\_endpoint` VALUES (0, 1, 1, 1, '10.254.90.32', 'http://10.254.90.33:1192, 1, NULL, NULL, '10.254.110.31', '123456', 'root', NULL);

#### 3 user表

User表存用户信息，表结构如下：

`USER\_ID` varchar(16) NOT NULL COMMENT 'id of user',

`ALIAS` varchar(64) DEFAULT NULL COMMENT 'user alias',

`PASSWORD` char(8) NOT NULL,

`USER\_TYPE` tinyint(4) DEFAULT NULL COMMENT 'tenent user or admin user',

`user\_group` int(11) NOT NULL,

`STATUS` tinyint(4) DEFAULT NULL COMMENT 'account status ,used for control login',

`TENANT\_ID` int(11) DEFAULT NULL COMMENT 'belongs witch tenant',

`NOTE` varchar(256) DEFAULT NULL COMMENT 'note for this record',

`LAST\_UPDATED` timestamp NULL DEFAULT NULL ON UPDATE CURRENT\_TIMESTAMP COMMENT 'last updated time',

User表关联user\_group表权限的问题，字段user\_group默认从svn取下来值不做修改。

一般user表有guest、hpcms和admin 三个用户。

INSERT INTO `user` VALUES ('guest', 'guest', '123456', '1', '0', '0', '2', null, '2016-01-21 14:54:51');

INSERT INTO `user` VALUES ('hpcms', 'hpcms', '123456', '1', '0', '0', '1', null, '2016-01-21 14:54:50');

注：如果是非中心机器，这里只需要admin一个用户

INSERT INTO `user` VALUES (admin, 'admin', 'admin', '1', '0', '0', '1', null, '2016-01-21 14:54:50');

#### 4 docker\_images

docker\_images表，用于存放images,初始化时候不存放数据，从SVN获取时候，去掉insert。

注：1 . 对于mysql 数据库序列，默认是从1 开始。即使insert 写的是0，添加入数据库也会变成1。如果就要是用0 ，可以进入数据库update修改。

#### 5其他表

除了以上五张表以外的7张表，暂时不需要做变动，从SVN上获取后，不做修改。

### 二ku8manager运行脚本维护

ku8eye-web使用端口1199 作为唯一访问端口。使用 –v 将文件映射到容器内。

run\_ku8manager.sh 内容如下：

docker run -tid --name ku8manager \

-p 3306:3306 --volumes-from=mariadb-data \ #映射mariadb-data容器

-v /var/run/docker.sock:/var/run/docker.sock \#映射docker.sock文件

-v ${PWD}/ku8eye-web-0.0.2-SNAPSHOT.jar:/ku8manager/ku8manager.jar \ #映射jar包

-v ${PWD}/initsql.sql:/ku8manager/db\_scripts/initsql.sql \ #映射mysql的sql

-v ${PWD}/ku8\_ext\_files:/ku8\_ext\_files \ #映射文件

-v ${PWD}/kubernetes\_cluster\_setup:/ku8manager/kubernetes\_cluster\_setup \

-v ${PWD}/kubernetes\_cluster\_uninstall:/ku8manager/kubernetes\_cluster\_uninstall \

-p 1199:8080 ku8manager

每次更新的jar和不同环境下的initsql.sql 文件都需要实时更新到容器内。

## 第三部分 kubernetes网络

这里说的网络是指，kubernetes集群中，各个nodes之间的网络。所以前提是nodes 已经安装完成。

配置网络，我们采用georce\_route\_quagga工具和手工配置两种方式。一般建议使用quagga工具。但使用quagga工具方式，有个大的缺陷是，在遇到到物理机存在多网卡时，不能指定网卡，比如现在省机器多数都是多网卡。所以在安装网络前，先检查物理网络。route 命令查看，如果为多网卡，只能采用手工配置方式。

### 一 georce\_route\_quagga工具

把georce\_route\_quagga工具做成镜像。因为nodes 安装有手动和使用ansible-playbook命令安装方式，所以，这里又分为这两种方式。

#### 1 手动安装georce\_route\_quagga镜像

cp georce\_route\_quagga.tar 镜像和start\_quagga\_router.sh到物理机任目录。

docker load –i georce\_route\_quagga.tar

docker tag f96cfe685533 index.alauda.cn/georce/router

]# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

index.alauda.cn/georce/router latest f96cfe685533 9 months ago 139.1 MB

s ~]# cat start\_quagga\_router.sh

docker run -itd --name=router --privileged --net=host --restart=always index.alauda.cn/georce/router

s ~]#./start\_quagga\_router.sh

~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

5c5ded94f962 index.alauda.cn/georce/router "/opt/run.sh" 4 weeks ago Up 4 days

#### 2 ansible-playbook命令安装

在使用ansible-playbook命令安装nodes 的过程中，第一步安装的是docker，第二步安装的就是quagga。在这就不做具体说明，大家可以查看使用ansible-playbook部分。

### 二 手工配置

在手动配置网络之前，检查主机网络。再次强调，一定是在nodes主机上操作。

添加路由表条数计算：nodes数 – 1（node主机到其他nodes主机的网络）

添加Destination和Genmask：参考每个node上docker0

选定网卡：使用route命令，

~]# route

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

default 10.254.106.1 0.0.0.0 UG 0 0 0 team0

10.254.106.0 0.0.0.0 255.255.255.128 U 0 0 0 team0

link-local 0.0.0.0 255.255.0.0 U 1004 0 0 em3

link-local 0.0.0.0 255.255.0.0 U 1006 0 0 p3p1

link-local 0.0.0.0 255.255.0.0 U 1007 0 0 p3p2

link-local 0.0.0.0 255.255.0.0 U 1010 0 0 team0

192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 em3

192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 docker0

192.168.130.0 0.0.0.0 255.255.255.0 U 0 0 0 p3p1

192.168.130.0 0.0.0.0 255.255.255.0 U 0 0 0 p3p2

route 看到default 使用的是team0 ，所以选定网卡为team0。

知道Destination 、Genmask和网卡后，开始添加 。

route add -net 192.168.1.0 gw 10.254.106.32 netmask 255.255.255.0 dev team0

route add -net 192.168.2.0 gw 10.254.106.33 netmask 255.255.255.0 dev team0

route add -net 192.168.3.0 gw 10.254.106.34 netmask 255.255.255.0 dev team0

route add -net 192.168.4.0 gw 10.254.106.35 netmask 255.255.255.0 dev team0

route add -net 192.168.5.0 gw 10.254.106.36 netmask 255.255.255.0 dev team0

# route

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

default 10.254.106.1 0.0.0.0 UG 0 0 0 team0

10.254.106.0 0.0.0.0 255.255.255.128 U 0 0 0 team0

link-local 0.0.0.0 255.255.0.0 U 1004 0 0 em3

link-local 0.0.0.0 255.255.0.0 U 1006 0 0 p3p1

link-local 0.0.0.0 255.255.0.0 U 1007 0 0 p3p2

link-local 0.0.0.0 255.255.0.0 U 1010 0 0 team0

192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 em3

192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 docker0

192.168.2.0 10.254.106.33 255.255.255.0 UG 0 0 0 team0

192.168.3.0 10.254.106.34 255.255.255.0 UG 0 0 0 team0

192.168.4.0 10.254.106.35 255.255.255.0 UG 0 0 0 team0

192.168.5.0 10.254.106.36 255.255.255.0 UG 0 0 0 team0

192.168.130.0 0.0.0.0 255.255.255.0 U 0 0 0 p3p1

192.168.130.0 0.0.0.0 255.255.255.0 U 0 0 0 p3p2

随机选一台node主机，进入该主机的任意容器，ping route add 添加的路由，查看网络情况。

# ping 192.168.1.1

PING 192.168.1.1 (192.168.1.1): 56 data bytes

64 bytes from 192.168.1.1: seq=0 ttl=63 time=0.237 ms

64 bytes from 192.168.1.1: seq=1 ttl=63 time=0.184 ms

64 bytes from 192.168.1.1: seq=2 ttl=63 time=0.150 ms

64 bytes from 192.168.1.1: seq=3 ttl=63 time=0.181 ms

为此，添加成功。

## 第四部分Kubernetes日志收集和查询

前提kubernetes集群安装完成。

Kubernetes日志收集和查询需要用到三个镜像文件：fluentd-elasticsearch.tar，elasticsearch\_1.8.tar，kibana\_1.3.tar；

六个yaml文：k8s-fluntd-daemonSet.yaml,es-service-account.yaml,es-service.yaml, es-controller.yaml, kibana-controller.yaml,kibana-service.yaml

两个配置文件：td-agent-master.conf，td-agent-node.conf，

由于文件多，操作步骤繁复，这里单独列出一个分册《Kubernetes安装运维手册--日志收集和查询安装文档1.2.docx》。

## 第五部分 ku8manager多集群安装和维护

由于总部需要对多个单个集群进行集中管理，开发了ku8manager多集群版本。

上面第二部分ku8manager安装以及第三部分讲述的都是单集群的安装和运维。这里单独把多集群的安装拿出来。

### 一ku8manager多集群安装

安装ku8manager前提，docker已安装。以生产环境测试集群10.255242.213为例:

#### 1 cp文件

多集群安装和单集群差不多，都有两个镜像，两个文件，一个jar包。唯一不同的是jar换成ku8manager-cluster-1.0.jar，sql 换成init\_multi.sql

将ku8manager相关文件cp 到物理机/opt

#### 2镜像操作

#Dokcer load –i busybox.tar.gz

#Docker tag b175bcb79023 busybox

#docker load -i ku8manager-env.tar.gz

#docker tag 5d9ca3091b03 ku8manager

#docker images

ku8manager latest 5d9ca3091b03 3 months ago 1.279 GB

busybox latest b175bcb79023 3 months ago 1.114 MB

#### 3 vi ku8manager和busybox运行脚本

#vi run\_busybox.sh

docker run --name=mariadb-data -e TZ="Asia/Shanghai" -v /var/lib/mysql busybox true

# vi run\_ku8manager.sh

docker run -tid --name ku8manager \

-p 3306:3306 --volumes-from=mariadb-data \

-v /var/run/docker.sock:/var/run/docker.sock \

-v ${PWD}/ku8manager-cluster-1.0.jar:/ku8manager/ku8manager.jar \

-v ${PWD}/init\_multi.sqll:/ku8manager/db\_scripts/initsql.sql \

-v ${PWD}/ku8\_ext\_files:/ku8\_ext\_files \

-v ${PWD}/kubernetes\_cluster\_setup:/ku8manager/kubernetes\_cluster\_setup \

-v ${PWD}/kubernetes\_cluster\_uninstall:/ku8manager/kubernetes\_cluster\_uninstall \

-p 1198:8088 ku8manager

为了和单集群区分，多集群我们使用1198:8088 端口。

#### 4启动并检查ku8manager容器

和单集群一样，在启动ku8manager之前，需要先启动busybox 来存储ku8manager的init 数据。

#./ run\_busybox.sh

# docker ps -a |grep busybox

a289eaae76bf busybox "true" 7 weeks ago Exited (0) 4 days ago mariadb-data

启动busybox 成功后，启动ku8manager

#./ run\_ku8manager.sh

#docker ps -a |grep ku8manager

4a3fbb7078ad ku8manager "/ku8manager/docker-e" 2 days ago Up 2 days 0.0.0.0:3306->3306/tcp, 0.0.0.0:1198->8088/tcp ku8manager

2274358caf74 ku8manager

#### 5验证ku8manager

Ku8manager容器启动成功后，进入其容器,查看tomcat.log 日志。看到有main] org.ku8eye.App : Started App in 10.011 seconds (JVM running for 11.439) 说明启动ku8eye.App启动成功。

# less tomcat.log

[class org.mybatis.spring.SqlSessionFactoryBean] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)

2016-05-06 16:14:09.890 INFO 113 --- [ main] trationDelegate$BeanPostProcessorChecker : Bean 'sqlSessionFactory' of type...skipping...

2016-05-06 16:14:14.883 INFO 113 --- [ main] org.ku8eye.App : Started App in 10.011 seconds (JVM running for 11.439)

打开google浏览器，地址栏输入<http://10.255.242.213:1198> 进入登录页面。输入用户名/密码：admin/123 如果能够正常进入，则多集群ku8manager安装完成。

### 二ku8manager多集群维护

#### 1 init\_multi.sql 维护

多集群目前共用到24张表，分别是docker\_image,host,ku8\_app\_deployment,

ku8\_app\_view,ku8\_application,ku8\_cluster,ku8\_config\_map,ku8\_job,ku8\_job\_deployment,ku8\_job\_view,ku8\_registry,ku8\_registry\_cluster,ku8\_res\_part\_cluster\_zone,ku8\_res\_partion,ku8\_service,ku8\_service\_deployment,ku8\_service\_view,ku8s\_srv\_endpoint,task\_log,tenant,user,user\_action,user\_group,zone。

需要维护的表有user\_group，user

1. user\_group表

user\_group 用于存放用户组，表结构如下：

`group\_id` int(11) NOT NULL AUTO\_INCREMENT,

`group\_name` varchar(50) NOT NULL,

`tenant\_id` int(11) NOT NULL,

`ku8\_parttion\_ids` varchar(500) DEFAULT NULL,

PRIMARY KEY (`group\_id`),

UNIQUE KEY `tenant\_id` (`tenant\_id`,`group\_name`) USING BTREE

user\_group 需要默认存放一条超级管理组。当group\_id=’1’ 为超级管理员

INSERT INTO `user\_group` VALUES ('1', 'SUPER\_ADMIN\_GROUP', '1', null);

1. user表

user表用户存放用户信息，表结构如下：

`USER\_ID` varchar(16) NOT NULL COMMENT 'id of user',

`ALIAS` varchar(64) DEFAULT NULL COMMENT 'user alias',

`PASSWORD` char(32) NOT NULL,

`USER\_TYPE` tinyint(4) DEFAULT NULL COMMENT 'tenent user or admin user',

`user\_group` int(11) NOT NULL,

`STATUS` tinyint(4) DEFAULT NULL COMMENT 'account status, used for control login',

`TENANT\_ID` int(11) DEFAULT NULL COMMENT 'belongs witch tenant',

`NOTE` varchar(256) DEFAULT NULL COMMENT 'note for this record',

`LAST\_UPDATED` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT 'last updated time',

PRIMARY KEY (`USER\_ID`)

user表 默认存在用户名为admin的超级管理员。当group\_id=’1’ 为超级管理员；group\_id=’3’为租户管理员；group\_id=’0为租户。

INSERT INTO `user` VALUES ('admin', 'admin', '202CB962AC59075B964B07152D234B70', '1', '1', null, '1', 'super admin', '2016-02-22 20:00:31');

1. user\_action 表

user\_action 表存放用户访问权限信息表，表结构如下：

`user\_type` varchar(8) NOT NULL,

`uri` varchar(100) NOT NULL,

`note` varchar(200) DEFAULT NULL,

`class` varchar(50) NOT NULL,

PRIMARY KEY (`uri`)

user\_type=’ 0,1,3’ ：任何用户都可以有执行权限。

user\_type=’ 1,3’ ：超级用户和租户管理员权限。

user\_type=’ 1’：超级用户权限

user\_type=’ 3’： 租户管理员权限。

user\_type=’ 0’： 租户权限。

user\_action 定义了用户访问关系。默认存放188条数据。由于有数据量大，这个里就不全贴insert 语句。

INSERT INTO `user\_action` VALUES ('0,1,3', '/application/deleteApplication', 'Delete Application by ID', 'ApplicationController');

INSERT INTO `user\_action` VALUES ('0,1,3', '/application/deletePod', 'Delete the current Pod', 'ApplicationController');

INSERT INTO `user\_action` VALUES ('0,1,3', '/application/deployApplication', 'Deploy application to Clusters', 'ApplicationController');

INSERT INTO `user\_action` VALUES ('0,1,3', '/application/getApplication', 'Get application by ID', 'ApplicationController');

INSERT INTO `user\_action` VALUES ('0,1,3', '/application/getApplications', 'Get all application', 'ApplicationController');

#### 2 ku8manager运行脚本维护

ku8manager使用端口1198 作为唯一访问端口。使用 –v 将文件映射到容器内。

run\_ku8manager.sh 内容如下：

docker run -tid --name ku8manager \

-p 3306:3306 --volumes-from=mariadb-data \ #映射mariadb-data容器

-v /var/run/docker.sock:/var/run/docker.sock \#映射docker.sock文件

-v ${PWD}/ ku8manager-cluster-1.0.jar:/ku8manager/ ku8manager-cluster-1.0.jar\ #映射jar包

-v ${PWD}/init\_multi.sql:/ku8manager/db\_scripts/initsql.sql \ #映射mysql的sql

-v ${PWD}/ku8\_ext\_files:/ku8\_ext\_files \ #映射文件

-v ${PWD}/kubernetes\_cluster\_setup:/ku8manager/kubernetes\_cluster\_setup \

-v ${PWD}/kubernetes\_cluster\_uninstall:/ku8manager/kubernetes\_cluster\_uninstall \

-p 1198:8088 ku8manager

每次更新的jar和不同环境下的init\_multi.sql 文件都需要实时更新到容器内。

## 编制历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 更新时间 | 主要内容或重大修改 | 执笔人 | 审核人 |
| V1.0.0 | 2016-12-8 | 初稿 | 陈建松、赵玲丽 | 龚正 |
| V1.0.1 | 2016-2-2 | 修正版 | 陈建松、赵玲丽 | 龚正 |
| V1.0.3 | 2016-2-18 | 增加第三部分 ku8manager维护手册 | 陈建松、赵玲丽 | 龚正 |
| V1.0.4 | 2016-3-23 | 1增加第四部分 kubernetes网络  2 私库镜像，添加删除本地镜像功能 | 陈建松、赵玲丽 | 龚正 |
| V1.0.5 | 2016-04-13 | 1 更新master安装步骤  2添加了kubernetes日志收集和查询功能 | 陈建松、赵玲丽 | 龚正 |
| V1.0.6 | 2016-05-06 | 1添加ku8manager多集群安装 | 陈建松、赵玲丽 | 龚正 |