k8s中所有节点的初始化配置与安装

保存集群信息到hosts文件中

```
#小的集群可以使用hosts文件代替域名解析，大的集群需搭建DNS服务器
vim /etc/hosts
#写入如下内容
10.41.95.99 k8s-master01
10.41.95.83 k8s-node01
10.41.95.138 k8s-node02
10.41.95.99 reg.swharbor.com
```

设置系统主机名以及 Host 文件的相互解析

```
hostnamectl set-hostname k8s-master01(主机名由具体节点名决定)
```

安装依赖包

```
yum install -y conntrack ntpdate ntp ipvsadm ipset jq iptables curl sysstat
libseccomp wget vim net-tools git
```

设置防火墙为 Iptables 并设置空规则

```
systemctl stop firewalld
systemctl disable firewalld
yum -y install iptables-services
systemctl start iptables
systemctl enable iptables
iptables -F
service iptables save
```

关闭 SELINUX

```
swapoff -a && sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab #注释/etc/fstab文件中
包含swap的行
setenforce 0 && sed -i 's/^SELINUX=.*/SELINUX=disabled/' /etc/selinux/config #设
置/etc/selinux/config中SELINUX为disabled
```

调整内核参数,对于 K8S

```
vim /etc/sysctl.d/kubernetes.conf
#写入以下内容
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
net.ipv4.ip_forward=1
net.ipv4.tcp_tw_recycle=0
vm.swappiness=0 # 禁止使用 swap 空间,只有当系统 OOM 时才允许使用它
vm.overcommit_memory=1 # 不检查物理内存是否够用
vm.panic_on_oom=0 # 开启 OOM
fs.inotify.max_user_instances=8192
fs.inotify.max_user_watches=1048576
fs.file-max=52706963
fs.nr_open=52706963
net.ipv6.conf.all.disable_ipv6=1
net.netfilter.nf_conntrack_max=2310720
#加载配置
sysctl -p /etc/sysctl.d/kubernetes.conf
```

设置 rsyslogd 和 systemd journald

```
mkdir /var/log/journal # 持久化保存日志的目录
mkdir /etc/systemd/journald.conf.d
cat > /etc/systemd/journald.conf.d/99-prophet.conf <<EOF
[Journal]
# 持久化保存到磁盘
Storage=persistent
# 压缩历史日志
Compress=yes
SyncIntervalSec=5m
RateLimitInterval=30s
RateLimitBurst=1000
# 最大占用空间 10G
SystemMaxUse=10G
# 单日志文件最大 200M
SystemMaxFileSize=200M
# 日志保存时间 2 周
MaxRetentionSec=2week
# 不将日志转发到 syslog
ForwardToSyslog=no
EOF
systemctl restart systemd-journald
```

升级系统内核为 4.44

```
#centos7中自带的3.10.x内核存在一些不稳定的因素，尽可能升级系统内核
#目前机房服务器存在升级内核之后无法开机的情况
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
# 安装完成后检查 /boot/grub2/grub.cfg 中对应内核 menuentry 中是否包含 initrd16 配置,如
果没有,再安装
一次!
yum --enablerepo=elrepo-kernel install -y kernel-lt
# 设置开机从新内核启动
grub2-set-default 'CentOS Linux (4.4.189-1.el7.elrepo.x86_64) 7 (Core)'
```

kube-proxy开启ipvs的前置条件

```
# kube-proxy支持 iptables 和 ipvs 两种模式
# 目前k8s最新的负载均衡模式 ipvs
# ipvs依赖于iptables
modprobe br_netfilter
cat > /etc/sysconfig/modules/ipvs.modules <<EOF
#!/bin/bash
```

```
modprobe -- ip_vs
modprobe -- ip_vs_rr
modprobe -- ip_vs_wrr
modprobe -- ip_vs_sh
modprobe -- nf_conntrack_ipv4
EOF
chmod 755 /etc/sysconfig/modules/ipvs.modules && bash
/etc/sysconfig/modules/ipvs.modules &&
lsmod | grep -e ip_vs -e nf_conntrack_ipv4
```

安装 Docker 软件

```
yum install -y yum-utils device-mapper-persistent-data lvm2

yum-config-manager \
--add-repo \
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

#yum update -y &&
yum -y install docker-ce-cli-19.03.4-3.el7
yum install -y docker-ce-19.03.4-3.el7
## 创建 /etc/docker 目录
mkdir /etc/docker
# 配置 daemon.
cat > /etc/docker/daemon.json <<EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2",
"storage-opts": [
"overlay2.override_kernel_check=true"
],
"insecure-registries": ["reg.swharbor.com"] #添加仓库名称，否则使用不了搭建的私有仓库
}
EOF
mkdir -p /etc/systemd/system/docker.service.d
# 重启docker服务 设置自启动是为了保证重启服务器之后环境还可以正常运行
systemctl daemon-reload && systemctl restart docker && systemctl enable docker
```

安装 Kubeadm

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
yum -y install kubeadm-1.16.2 kubectl-1.16.2 kubelet-1.16.2
systemctl enable kubelet.service
```

下载相关镜像至服务器

```
#目前相关镜像以保存至私有仓库 reg.swharbor.com 直接pull即可
docker pull reg.swharbor.com/k8s/kube-controller-manager:v1.16.2
docker pull reg.swharbor.com/k8s/kube-scheduler:v1.16.2
docker pull reg.swharbor.com/k8s/kube-proxy:v1.16.2
docker pull reg.swharbor.com/k8s/pause:3.1
docker pull reg.swharbor.com/k8s/etcd:3.3.15-0
docker pull reg.swharbor.com/k8s/coredns:1.6.2
docker pull reg.swharbor.com/k8s/flannel:v0.11.0-amd64

docker tag reg.swharbor.com/k8s/kube-apiserver:v1.16.2 k8s.gcr.io/kube-
apiserver:v1.16.2
docker tag reg.swharbor.com/k8s/kube-controller-manager:v1.16.2 k8s.gcr.io/kube-
controller-manager:v1.16.2
docker tag reg.swharbor.com/k8s/kube-scheduler:v1.16.2 k8s.gcr.io/kube-
scheduler:v1.16.2
docker tag reg.swharbor.com/k8s/kube-proxy:v1.16.2 k8s.gcr.io/kube-proxy:v1.16.2
docker tag reg.swharbor.com/k8s/pause:3.1 k8s.gcr.io/pause:3.1
docker tag reg.swharbor.com/k8s/etcd:3.3.15-0  k8s.gcr.io/etcd:3.3.15-0
docker tag reg.swharbor.com/k8s/coredns:1.6.2 k8s.gcr.io/coredns:1.6.2
docker tag reg.swharbor.com/k8s/flannel:v0.11.0-amd64
quay.io/coreos/flannel:v0.11.0-amd64
```

以上所有的步骤每个节点都需要运行

对于master节点而然

初始化主节点

```
kubeadm config print init-defaults > kubeadm-config.yaml
#需要根据实际情况修改如下设置
localAPIEndpoint:
  advertiseAddress: 10.41.95.99 #IP设置为master的物理ip地址
kubernetesVersion: v1.16.2 #设置当前安装的k8s版本
networking:
  dnsDomain: cluster.local
  podSubnet: 10.244.0.0/16 #添加这一行作为pod网络(flannel)
  serviceSubnet: 10.96.0.0/12
---                         #添加以下ipvs配置
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
```

```
featureGates:
SupportIPVSProxyMode: true
mode: ipvs

kubeadm init --config=kubeadm-config.yaml | tee kubeadm-init.log
```

初始化之后的准备操作

```
# 以下内容是master节点初始化成功之后的内容，执行以下三条命令之后，记得保存最后一条命令(加入集群的命令)
To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.41.95.99:6443 --token abcdef.0123456789abcdef \
    --discovery-token-ca-cert-hash
sha256:3eb9e5adac44bca1ed52334562397743e82c6cb31fda16c1c535cf4bd97d23e5
```

部署网络

```
# 如果k8s中扁平化网络没有搭建好，master节点不会ready
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
# 问题点，有时候网络会配置不成功导致master节点无法ready
journalctl -f -u kubelet.service #具体问题具体分析，可以重新安装k8s
```

对于node节点而然

加入集群命令

```
# 执行加入集群的命令即可，其会自动配置node的扁平化网络
kubeadm join 10.41.95.99:6443 --token abcdef.0123456789abcdef \
    --discovery-token-ca-cert-hash
sha256:3eb9e5adac44bca1ed52334562397743e82c6cb31fda16c1c535cf4bd97d23e5
```

master节点验证是否加入成功

```
kubectl get nodes
NAME           STATUS   ROLES    AGE   VERSION
k8s-master01   Ready    master   14h   v1.16.2
k8s-node01     Ready    <none>   13h   v1.16.2
```

网络上集群搭建有kubeadm与二进制安装方法，目前采用的是kubeadm安装方式(比较简单)