kubeadm的基础使用

- 本篇文章在张馆长的基础之上使用了Debian来模拟测试。
- 市面上很多kubeadm的文章都是错误示范或者不够详细，大多数都没写写系统设置之类的就直接kubeadm init导致很多跟着做的人会报错

我期望看到本文的读者最少具备以下知识:

- Linux一些目录规范和systemd
- 学过一点docker
- 懂dns和/etc/hosts、curl互相结合来测试一些web的接口响应状态
- 不要求github有自己项目，至少会浏览github

本教学将以下列节点数与规格来进行部署Kubernetes集群,系统CentOS 7.6+，有条件7.7，不要使用centos7.4以及以下，**这里使用Debian 10.2，内核4.19**，容器技术依赖于内核技术，低版本系统部署和运行后问题会非常多

IP	Hostname	role	CPU	Memory
10.10.10.91	K8S-M1	master	2	8G
10.10.10.92	K8S-M2	master	2	8G
10.10.10.93	K8S-M3	master	2	8G
10.10.10.94	K8S-N1	node	2	8G

vim /etc/hosts

more /etc/hosts

10.10.10.91 k8s-m1

10.10.10.92 k8s-m2

10.10.10.93 k8s-m3

10.10.10.94 k8s-n1

vim /etc/hostname

more /etc/hostname

K8S-M1

```
root@localhost:~# vim /etc/hosts
root@localhost:~# more /etc/hosts
127.0.0.1       localhost
127.0.1.1       localhost

10.10.10.91 k8s-m1
10.10.10.92 k8s-m2
10.10.10.93 k8s-m3
10.10.10.94 k8s-n1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
root@localhost:~# vim /etc/hostname
root@localhost:~# more /etc/hostname
K8S-M1
root@localhost:~#
```

```
root@localhost:~# vim /etc/hosts
root@localhost:~# more /etc/hosts
127.0.0.1       localhost
127.0.1.1       localhost

10.10.10.91 k8s-m1
10.10.10.92 k8s-m2
10.10.10.93 k8s-m3
10.10.10.94 k8s-n1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
root@localhost:~# vim /etc/hostname
root@localhost:~# more /etc/hostname
K8S-M2
root@localhost:~#
```

```
root@localhost:~# vim /etc/hosts
root@localhost:~# more /etc/hosts
127.0.0.1       localhost
127.0.1.1       localhost

10.10.10.91 k8s-m1
10.10.10.92 k8s-m2
10.10.10.93 k8s-m3
10.10.10.94 k8s-n1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
root@localhost:~# vim /etc/hostname
root@localhost:~# more /etc/hostname
K8S-M3
root@localhost:~#
```

```
root@localhost:~# vim /etc/hosts
root@localhost:~# more /etc/hosts
127.0.0.1       localhost
127.0.1.1       localhost

10.10.10.91 k8s-m1
10.10.10.92 k8s-m2
10.10.10.93 k8s-m3
10.10.10.94 k8s-n1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
root@localhost:~# vim /etc/hostname
root@localhost:~# more /etc/hostname
K8S-N1
root@localhost:~#
```

所有操作全部用root使用者进行，系统盘尽量大点，不然到时候镜像多了例如到了85%会被gc回收镜像

- 高可用一般建议大于等于3台的奇数台,我使用3台master来做高可用
- 一台也可以，但是差距不大，差异性我会在文章中注明的

事前准备

系统层面设置

假设系统是刚用官方iso安装完成未作任何配置(网络和dns自行去配置)

- 所有防火墙与SELinux 已关闭。如CentOS：

否则后续 K8S 挂载目录时可能报错 Permission denied，有些云厂商的ip是被NetworkManager纳管的(例如青云)，停了它会网络不通，可以不停。

systemctl disable --now firewalld NetworkManager

setenforce 0

sed -ri '/^[^#]*SELINUX=/s#=.+$#=disabled#' /etc/selinux/config

ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

Debian没有安装selinux，所以这一步略过。

- 关闭 dnsmasq (可选)

linux 系统开启了 dnsmasq 后(如 GUI 环境)，将系统 DNS Server 设置为 127.0.0.1，这会导致 docker 容器无法解析域名，需要关闭它

systemctl disable --now dnsmasq

由于我Debian没有安装dns服务，所以这一步也略过。

- Kubernetes 建议关闭系统Swap,在所有机器使用以下指令关闭swap并注释掉/etc/fstab中swap的行，不想关闭可以不执行，后面会应对的配置选项：每台都执行。

swapoff -a && sysctl -w vm.swappiness=0

sed -ri '/^[^#]*swap/s@^@#@' /etc/fstab

more /etc/fstab

```
root@K8S-M1:~# swapoff -a && sysctl -w vm.swappiness=0
vm.swappiness = 0
root@K8S-M1:~# sed -ri '/^[^#]*swap/s@^@#@' /etc/fstab
root@K8S-M1:~# more /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
/dev/mapper/VolGroup-lv_root /               xfs     defaults        0       0
# /boot was on /dev/sda1 during installation
UUID=a009ea15-a676-4838-91ac-c558ae14b838 /boot           xfs     defaults        0       0
#/dev/mapper/VolGroup-lv_swap none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
root@K8S-M1:~#
```

```
root@K8S-M2:~# swapoff -a && sysctl -w vm.swappiness=0
vm.swappiness = 0
root@K8S-M2:~# sed -ri '/^[^#]*swap/s@^@#@' /etc/fstab
root@K8S-M2:~# more /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
/dev/mapper/VolGroup-lv_root /               xfs     defaults        0       0
# /boot was on /dev/sda1 during installation
UUID=a009ea15-a676-4838-91ac-c558ae14b838 /boot           xfs     defaults        0       0
#/dev/mapper/VolGroup-lv_swap none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
root@K8S-M2:~#
```

```
root@K8S-M3:~# swapoff -a && sysctl -w vm.swappiness=0
vm.swappiness = 0
root@K8S-M3:~# sed -ri '/^[^#]*swap/s@^@#@' /etc/fstab
root@K8S-M3:~# more /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
/dev/mapper/VolGroup-lv_root /               xfs     defaults        0       0
# /boot was on /dev/sda1 during installation
UUID=a009ea15-a676-4838-91ac-c558ae14b838 /boot           xfs     defaults        0       0
#/dev/mapper/VolGroup-lv_swap none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
root@K8S-M3:~#
```

```
root@K8S-N1:~# swapoff -a && sysctl -w vm.swappiness=0
vm.swappiness = 0
root@K8S-N1:~# sed -ri '/^[^#]*swap/s@^@#@' /etc/fstab
root@K8S-N1:~# more /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
/dev/mapper/VolGroup-lv_root /               xfs     defaults        0       0
# /boot was on /dev/sda1 during installation
UUID=a009ea15-a676-4838-91ac-c558ae14b838 /boot           xfs     defaults        0       0
#/dev/mapper/VolGroup-lv_swap none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
root@K8S-N1:~#
```

安装一些基础依赖和工具，每台机器都安装。

centos-7版本：

```
yum install epel-release -y
yum install -y wget \
    git \
    conntrack-tools \
    psmisc \
    nfs-utils \
    jq \
    socat \
    bash-completion \
    ipset \
    ipvsadm \
    conntrack \
    libseccomp \
    net-tools \
    crontabs \
    sysstat \
    unzip \
    bind-utils \
    tcpdump \
    telnet \
    lsof \
    htop
```

Debian 10版本执行：

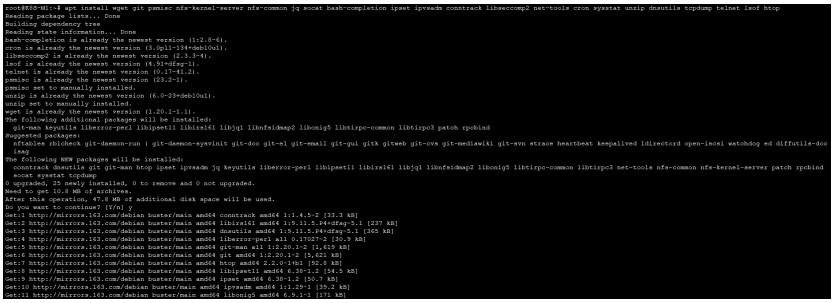
```
# apt update
```

```
root@K8S-M1:~# apt update
Hit:1 http://mirrors.163.com/debian buster InRelease
Get:2 http://mirrors.163.com/debian-security stretch/updates InRelease [94.3 kB]
Get:3 http://mirrors.163.com/debian buster-updates InRelease [49.3 kB]
Get:4 http://mirrors.163.com/debian buster-backports InRelease [46.7 kB]
Get:5 http://mirrors.163.com/debian-security stretch/updates/main Sources [201 kB]
Get:6 http://mirrors.163.com/debian-security stretch/updates/main amd64 Packages [508 kB]
Get:7 http://mirrors.163.com/debian buster-backports/main Sources.diff/Index [27.8 kB]
Get:8 http://mirrors.163.com/debian buster-backports/main amd64 Packages.diff/Index [27.8 kB]
Get:9 http://mirrors.163.com/debian buster-backports/main Sources 2019-11-28-0813.49.pdiff [645 B]
Get:10 http://mirrors.163.com/debian buster-backports/main Sources 2019-11-28-1422.41.pdiff [195 B]
Get:11 http://mirrors.163.com/debian buster-backports/main Sources 2019-11-28-2019.45.pdiff [622 B]
Get:11 http://mirrors.163.com/debian buster-backports/main Sources 2019-11-28-2019.45.pdiff [622 B]
Get:12 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-11-28-1422.41.pdiff [343 B]
Get:13 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-11-28-2019.45.pdiff [336 B]
Get:13 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-11-28-2019.45.pdiff [336 B]
Fetched 957 kB in 1s (688 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
```

apt install wget git psmisc nfs-kernel-server nfs-common jq socat bash-completion ipset ipvsadm conntrack libseccomp2 net-tools cron sysstat unzip dnsutils tcpdump telnet lsof htop curl

或者直接安装

apt install -y wget git psmisc nfs-kernel-server nfs-common jq socat bash-completion ipset ipvsadm conntrack libseccomp2 net-tools cron sysstat unzip dnsutils tcpdump telnet lsof htop curl




```
Creating config file /etc/default/sysstat with new version
update-alternatives: using /usr/bin/sar.sysstat to provide /usr/bin/sar (sar) in auto mode
Created symlink /etc/systemd/system/multi-user.target.wants/sysstat.service → /lib/systemd/system/sysstat.service.
Setting up libonig5:amd64 (6.9.1-1) ...
Setting up libtirpc3:amd64 (1.1.4-0.4) ...
Setting up libjq1:amd64 (1.5+dfsg-2+b1) ...
Setting up rpcbind (1.2.5-0.3+deb10u1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/rpcbind.service → /lib/systemd/system/rpcbind.service.
Created symlink /etc/systemd/system/sockets.target.wants/rpcbind.socket → /lib/systemd/system/rpcbind.socket.
Setting up nfs-common (1:1.3.4-2.5) ...

Creating config file /etc/idmapd.conf with new version
Adding system user `statd' (UID 118) ...
Adding new user `statd' (UID 118) with group `nogroup' ...
Not creating home directory `/var/lib/nfs'.
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target.
Created symlink /etc/systemd/system/remote-fs.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target.
nfs-utils.service is a disabled or a static unit, not starting it.
Setting up git (1:2.20.1-2) ...
Setting up jq (1.5+dfsg-2+b1) ...
Setting up nfs-kernel-server (1:1.3.4-2.5) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /lib/systemd/system/nfs-server.service.
Job for nfs-server.service canceled.

Creating config file /etc/exports with new version

Creating config file /etc/default/nfs-kernel-server with new version
Processing triggers for mime-support (3.62) ...
Processing triggers for gnome-menus (3.31.4-3) ...
Processing triggers for libc-bin (2.28-10) ...
Processing triggers for systemd (241-7~deb10u2) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for desktop-file-utils (0.23-4) ...
```

如果集群kube-proxy想使用ipvs模式的话需要开机加载下列模块儿，按照规范使用systemd-modules-load来加载而不是在/etc/rc.local里写modprobe

每个节点做法一样。

:> /etc/modules-load.d/ipvs.conf

module=(

> ip_vs

> ip_vs_rr

> ip_vs_wrr

> ip_vs_sh

> nf_conntrack

> br_netfilter

>)

for kernel_module in ${module[@]};do

> /sbin/modinfo -F filename $kernel_module |& grep -qv ERROR && echo $kernel_module >> /etc/modules-load.d/ipvs.conf || :

> done

more /etc/modules-load.d/ipvs.conf

```
root@K8S-M2:~# :> /etc/modules-load.d/ipvs.conf
root@K8S-M2:~# module=(
> ip_vs
> ip_vs_rr
> ip_vs_wrr
> ip_vs_sh
> nf_conntrack
> br_netfilter
>   )
root@K8S-M2:~# for kernel_module in ${module[@]};do
>     /sbin/modinfo -F filename $kernel_module |& grep -qv ERROR && echo $kernel_module >> /etc/modules-load.d/ipvs.conf || :
> done
root@K8S-M2:~# more /etc/modules-load.d/ipvs.conf
ip_vs
ip_vs_rr
ip_vs_wrr
ip_vs_sh
nf_conntrack
br_netfilter
root@K8S-M2:~#
```

直接启动设置会有警告

systemctl enable --now systemd-modules-load.service

```
root@K8S-M2:~# systemctl enable --now systemd-modules-load.service
The unit files have no installation config (WantedBy=, RequiredBy=, Also=,
Alias= settings in the [Install] section, and DefaultInstance= for template
units). This means they are not meant to be enabled using systemctl.

Possible reasons for having this kind of units are:
• A unit may be statically enabled by being symlinked from another unit's
  .wants/ or .requires/ directory.
• A unit's purpose may be to act as a helper for some other unit which has
  a requirement dependency on it.
• A unit may be started when needed via activation (socket, path, timer,
  D-Bus, udev, scripted systemctl call, ...).
• In case of template units, the unit is meant to be enabled with some
  instance name specified.
```

vim /lib/systemd/system/systemd-modules-load.service

增加install

[Install]

WantedBy=multi-user.target

```
root@K8S-M2:~# vim /lib/systemd/system/systemd-modules-load.service

#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Load Kernel Modules
Documentation=man:systemd-modules-load.service(8) man:modules-load.d(5)
DefaultDependencies=no
Conflicts=shutdown.target
Before=sysinit.target shutdown.target
ConditionCapability=CAP_SYS_MODULE
ConditionDirectoryNotEmpty=|/lib/modules-load.d
ConditionDirectoryNotEmpty=|/usr/lib/modules-load.d
ConditionDirectoryNotEmpty=|/usr/local/lib/modules-load.d
ConditionDirectoryNotEmpty=|/etc/modules-load.d
ConditionDirectoryNotEmpty=|/run/modules-load.d
ConditionKernelCommandLine=|modules-load
ConditionKernelCommandLine=|rd.modules-load

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/lib/systemd/systemd-modules-load
TimeoutSec=90s

[Install]
WantedBy=multi-user.target
```

```
root@K8S-M2:~# systemctl cat systemd-modules-load.service
# Warning: systemd-modules-load.service changed on disk, the version systemd has loaded is outdated.
# This output shows the current version of the unit's original fragment and drop-in files.
# If fragments or drop-ins were added or removed, they are not properly reflected in this output.
# Run 'systemctl daemon-reload' to reload units.
# /lib/systemd/system/systemd-modules-load.service
#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Load Kernel Modules
Documentation=man:systemd-modules-load.service(8) man:modules-load.d(5)
DefaultDependencies=no
Conflicts=shutdown.target
Before=sysinit.target shutdown.target
ConditionCapability=CAP_SYS_MODULE
ConditionDirectoryNotEmpty=|/lib/modules-load.d
ConditionDirectoryNotEmpty=|/usr/lib/modules-load.d
ConditionDirectoryNotEmpty=|/usr/local/lib/modules-load.d
ConditionDirectoryNotEmpty=|/etc/modules-load.d
ConditionDirectoryNotEmpty=|/run/modules-load.d
ConditionKernelCommandLine=|modules-load
ConditionKernelCommandLine=|rd.modules-load

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/lib/systemd/systemd-modules-load
TimeoutSec=90s

[Install]
WantedBy=multi-user.target
root@K8S-M2:~#
```

systemctl daemon-reload

systemctl enable --now systemd-modules-load.service

```
root@K8S-M2:~# systemctl daemon-reload
root@K8S-M2:~# systemctl enable --now systemd-modules-load.service
Created symlink /etc/systemd/system/multi-user.target.wants/systemd-modules-load.service → /lib/systemd/system/systemd-modules-load.service.
root@K8S-M2:~#
```

重启完服务发现已经加载了新的模块。

systemctl stop systemd-modules-load.service

systemctl start systemd-modules-load.service

systemctl status systemd-modules-load.service

```
root@K8S-M2:~# systemctl stop systemd-modules-load.service
root@K8S-M2:~# systemctl start systemd-modules-load.service
root@K8S-M2:~# systemctl status systemd-modules-load.service
● systemd-modules-load.service - Load Kernel Modules
   Loaded: loaded (/lib/systemd/system/systemd-modules-load.service; enabled; vendor preset: enabled)
   Active: active (exited) since Fri 2019-11-29 15:47:21 CST; 2s ago
     Docs: man:systemd-modules-load.service(8)
           man:modules-load.d(5)
  Process: 3548 ExecStart=/lib/systemd/systemd-modules-load (code=exited, status=0/SUCCESS)
 Main PID: 3548 (code=exited, status=0/SUCCESS)

Nov 29 15:47:21 K8S-M2 systemd[1]: Starting Load Kernel Modules...
Nov 29 15:47:21 K8S-M2 systemd-modules-load[3548]: Inserted module 'ip_vs'
Nov 29 15:47:21 K8S-M2 systemd-modules-load[3548]: Inserted module 'ip_vs_rr'
Nov 29 15:47:21 K8S-M2 systemd-modules-load[3548]: Inserted module 'ip_vs_wrr'
Nov 29 15:47:21 K8S-M2 systemd-modules-load[3548]: Inserted module 'ip_vs_sh'
Nov 29 15:47:21 K8S-M2 systemd-modules-load[3548]: Inserted module 'br_netfilter'
Nov 29 15:47:21 K8S-M2 systemd[1]: Started Load Kernel Modules.
root@K8S-M2:~#
```

这里查看已经有该模块了。

lsmod |grep ip_vs

```
root@K8S-M2:~# lsmod |grep ip_vs
ip_vs_sh               16384  0
ip_vs_wrr              16384  0
ip_vs_rr               16384  0
ip_vs                 172032  6 ip_vs_rr,ip_vs_sh,ip_vs_wrr
nf_conntrack          172032  1 ip_vs
nf_defrag_ipv6         20480  2 nf_conntrack,ip_vs
libcrc32c              16384  3 nf_conntrack,xfs,ip_vs
root@K8S-M2:~#
```

上面如果systemctl enable命令报错可以systemctl status -l systemd-modules-load.service看看哪个内核模块加载不了，在/etc/modules-load.d/ipvs.conf里注释掉它再enable试试

- 所有机器需要设定/etc/sysctl.d/k8s.conf的系统参数，目前对ipv6支持不怎么好，所以里面也关闭ipv6了。

cat <<EOF > /etc/sysctl.d/k8s.conf

net.ipv6.conf.all.disable_ipv6 = 1

net.ipv6.conf.default.disable_ipv6 = 1

net.ipv6.conf.lo.disable_ipv6 = 1

net.ipv4.neigh.default.gc_stale_time = 120

net.ipv4.conf.all.rp_filter = 0

net.ipv4.conf.default.rp_filter = 0

net.ipv4.conf.default.arp_announce = 2

net.ipv4.conf.lo.arp_announce = 2

net.ipv4.conf.all.arp_announce = 2

net.ipv4.ip_forward = 1
net.ipv4.tcp_max_tw_buckets = 5000
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_synack_retries = 2
# 要求iptables不对bridge的数据进行处理
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-arptables = 1
net.netfilter.nf_conntrack_max = 2310720
fs.inotify.max_user_watches=89100
fs.may_detach_mounts = 1
fs.file-max = 52706963
fs.nr_open = 52706963
vm.overcommit_memory=1
vm.panic_on_oom=0
EOF

```
root@K8S-M1:~# cat <<EOF > /etc/sysctl.d/k8s.conf
> net.ipv6.conf.all.disable_ipv6 = 1
> net.ipv6.conf.default.disable_ipv6 = 1
> net.ipv6.conf.lo.disable_ipv6 = 1
> net.ipv4.neigh.default.gc_stale_time = 120
> net.ipv4.conf.all.rp_filter = 0
> net.ipv4.conf.default.rp_filter = 0
> net.ipv4.conf.default.arp_announce = 2
> net.ipv4.conf.lo.arp_announce = 2
> net.ipv4.conf.all.arp_announce = 2
> net.ipv4.ip_forward = 1
> net.ipv4.tcp_max_tw_buckets = 5000
> net.ipv4.tcp_syncookies = 1
> net.ipv4.tcp_max_syn_backlog = 1024
> net.ipv4.tcp_synack_retries = 2
> # 要求iptables不对bridge的数据进行处理
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> net.bridge.bridge-nf-call-arptables = 1
> net.netfilter.nf_conntrack_max = 2310720
> fs.inotify.max_user_watches=89100
> fs.may_detach_mounts = 1
> fs.file-max = 52706963
> fs.nr_open = 52706963
> vm.overcommit_memory=1
> vm.panic_on_oom=0
> EOF
```

more /etc/sysctl.d/k8s.conf

```
root@K8S-M1:~# more /etc/sysctl.d/k8s.conf
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
net.ipv4.neigh.default.gc_stale_time = 120
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_announce = 2
net.ipv4.ip_forward = 1
net.ipv4.tcp_max_tw_buckets = 5000
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_synack_retries = 2
# 要求iptables不对bridge的数据进行处理
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-arptables = 1
net.netfilter.nf_conntrack_max = 2310720
fs.inotify.max_user_watches=89100
fs.may_detach_mounts = 1
fs.file-max = 52706963
fs.nr_open = 52706963
vm.overcommit_memory=1
vm.panic_on_oom=0
root@K8S-M1:~#
```

sysctl --system

```
root@K8S-M1:~# sysctl --system
* Applying /usr/lib/sysctl.d/30-tracker.conf ...
fs.inotify.max_user_watches = 65536
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
net.ipv4.neigh.default.gc_stale_time = 120
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_announce = 2
net.ipv4.ip_forward = 1
net.ipv4.tcp_max_tw_buckets = 5000
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_synack_retries = 2
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-arptables = 1
net.netfilter.nf_conntrack_max = 2310720
fs.inotify.max_user_watches = 89100
fs.file-max = 52706963
fs.nr_open = 52706963
vm.overcommit_memory = 1
vm.panic_on_oom = 0
* Applying /etc/sysctl.d/protect-links.conf ...
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
* Applying /etc/sysctl.conf ...
root@K8S-M1:~#
```

如果选择关闭swap也要在内核里关闭，不关闭可以不执行

echo 'vm.swappiness = 0' >> /etc/sysctl.d/k8s.conf

```
root@K8S-M1:~# echo 'vm.swappiness = 0' >> /etc/sysctl.d/k8s.conf
root@K8S-M1:~#
```

如果kube-proxy使用ipvs的话为了防止timeout需要设置下tcp参数

cat <<EOF >> /etc/sysctl.d/k8s.conf

https://github.com/moby/moby/issues/31208

ipvsadm -l --timout

修复ipvs模式下长连接timeout问题 小于900即可

net.ipv4.tcp_keepalive_time = 600

net.ipv4.tcp_keepalive_intvl = 30

net.ipv4.tcp_keepalive_probes = 10

EOF

```
root@K8S-M1:~# cat <<EOF >> /etc/sysctl.d/k8s.conf
> # https://github.com/moby/moby/issues/31208
> # ipvsadm -l --timout
> # 修复ipvs模式下长连接timeout问题 小于900即可
> net.ipv4.tcp_keepalive_time = 600
> net.ipv4.tcp_keepalive_intvl = 30
> net.ipv4.tcp_keepalive_probes = 10
> EOF
root@K8S-M1:~# more /etc/sysctl.d/k8s.conf
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
net.ipv4.neigh.default.gc_stale_time = 120
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_announce = 2
net.ipv4.ip_forward = 1
net.ipv4.tcp_max_tw_buckets = 5000
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_synack_retries = 2
# 要求iptables不对bridge的数据进行处理
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-arptables = 1
net.netfilter.nf_conntrack_max = 2310720
fs.inotify.max_user_watches=89100
fs.may_detach_mounts = 1
fs.file-max = 52706963
fs.nr_open = 52706963
vm.overcommit_memory=1
vm.panic_on_oom=0
vm.swappiness = 0
# https://github.com/moby/moby/issues/31208
# ipvsadm -l --timout
# 修复ipvs模式下长连接timeout问题 小于900即可
net.ipv4.tcp_keepalive_time = 600
net.ipv4.tcp_keepalive_intvl = 30
net.ipv4.tcp_keepalive_probes = 10
root@K8S-M1:~#
```

sysctl --system

```
root@K8S-M1:~# sysctl --system
* Applying /usr/lib/sysctl.d/30-tracker.conf ...
fs.inotify.max_user_watches = 65536
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
net.ipv4.neigh.default.gc_stale_time = 120
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_announce = 2
net.ipv4.ip_forward = 1
net.ipv4.tcp_max_tw_buckets = 5000
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_synack_retries = 2
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-arptables = 1
net.netfilter.nf_conntrack_max = 2310720
fs.inotify.max_user_watches = 89100
fs.file-max = 52706963
fs.nr_open = 52706963
vm.overcommit_memory = 1
vm.panic_on_oom = 0
vm.swappiness = 0
net.ipv4.tcp_keepalive_time = 600
net.ipv4.tcp_keepalive_intvl = 30
net.ipv4.tcp_keepalive_probes = 10
* Applying /etc/sysctl.d/protect-links.conf ...
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
* Applying /etc/sysctl.conf ...
root@K8S-M1:~#
```

- 优化设置 journal 日志相关，避免日志重复搜集，浪费系统资源。修改systemctl启动的最小文件打开数量。关闭ssh方向dns解析

这两句是可以不用执行的。因为Debian默认的配置文件里没有。

sed -ri 's/^\$ModLoad imjournal/#&/' /etc/rsyslog.conf

sed -ri 's/^\$IMJournalStateFile/#&/' /etc/rsyslog.conf

```
root@K8S-M1:~# sed -ri 's/^\$ModLoad imjournal/#&/' /etc/rsyslog.conf
root@K8S-M1:~# sed -ri 's/^\$IMJournalStateFile/#&/' /etc/rsyslog.conf
root@K8S-M1:~#
```

sed -ri 's/^#(DefaultLimitCORE)=/\1=100000/' /etc/systemd/system.conf

sed -ri 's/^#(DefaultLimitNOFILE)=/\1=100000/' /etc/systemd/system.conf

```
root@K8S-M1:~# sed -ri 's/^#(DefaultLimitCORE)=/\1=100000/' /etc/systemd/system.conf
root@K8S-M1:~# sed -ri 's/^#(DefaultLimitNOFILE)=/\1=100000/' /etc/systemd/system.conf
```

sed -ri 's/^#(UseDNS)yes/\1no/' /etc/ssh/sshd_config

more /etc/ssh/sshd_config |grep Use

```
root@K8S-M1:~# sed -ri 's/^#(UseDNS )yes/\1no/' /etc/ssh/sshd_config
root@K8S-M1:~# more /etc/ssh/sshd_config |grep Use
#AuthorizedKeysCommandUser nobody
#IgnoreUserKnownHosts no
UsePAM yes
#X11UseLocalhost yes
#PermitUserEnvironment no
#UseDNS no
#Match User anoncvs
root@K8S-M1:~#
```

- 文件最大打开数，按照规范，在子配置文件写

cat>/etc/security/limits.d/kubernetes.conf<<EOF

* soft nproc 131072

* hard nproc 131072

* soft nofile 131072

* hard nofile 131072

root soft nproc 131072

root hard nproc 131072

root soft nofile 131072

root hard nofile 131072

EOF

```
root@K8S-M1:~# cat>/etc/security/limits.d/kubernetes.conf<<EOF
> *       soft    nproc   131072
> *       hard    nproc   131072
> *       soft    nofile  131072
> *       hard    nofile  131072
> root    soft    nproc   131072
> root    hard    nproc   131072
> root    soft    nofile  131072
> root    hard    nofile  131072
> EOF
root@K8S-M1:~# more /etc/security/limits.d/kubernetes.conf
*       soft    nproc   131072
*       hard    nproc   131072
*       soft    nofile  131072
*       hard    nofile  131072
root    soft    nproc   131072
root    hard    nproc   131072
root    soft    nofile  131072
root    hard    nofile  131072
root@K8S-M1:~#
```

集群的HA依赖于时间一致性，安装并配置chrony

apt install chrony

```
root@K8S-M1:~# apt install chrony
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  networkd-dispatcher
The following NEW packages will be installed:
  chrony
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 234 kB of archives.
After this operation, 510 kB of additional disk space will be used.
Get:1 http://mirrors.163.com/debian buster/main amd64 chrony amd64 3.4-4 [234 kB]
Fetched 234 kB in 0s (547 kB/s)
Selecting previously unselected package chrony.
(Reading database ... 136960 files and directories currently installed.)
Preparing to unpack .../chrony_3.4-4_amd64.deb ...
Unpacking chrony (3.4-4) ...
Setting up chrony (3.4-4) ...
Creating '_chrony' system user/group for the chronyd daemon…

Creating config file /etc/chrony/chrony.conf with new version

Creating config file /etc/chrony/chrony.keys with new version
Created symlink /etc/systemd/system/chronyd.service → /lib/systemd/system/chrony.service.
Created symlink /etc/systemd/system/multi-user.target.wants/chrony.service → /lib/systemd/system/chrony.service.
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for systemd (241-7~deb10u2) ...
root@K8S-M1:~#
```

这里安装完自动已经设置为开机启动了。

cat>/etc/chrony/chrony.conf<<EOF

Welcome to the chrony configuration file. See chrony.conf(5) for more

```
# information about usuable directives.

server cn.pool.ntp.org iburst minpoll 4 maxpoll 10
server s1b.time.edu.cn iburst minpoll 4 maxpoll 10

# This directive specify the location of the file containing ID/key pairs for
# NTP authentication.
keyfile /etc/chrony/chrony.keys

# This directive specify the file into which chronyd will store the rate
# information.
driftfile /var/lib/chrony/chrony.drift

# Uncomment the following line to turn logging on.
#log tracking measurements statistics

# Log files location.
logdir /var/log/chrony

# Stop bad estimates upsetting machine clock.
maxupdateskew 100.0

# This directive enables kernel synchronisation (every 11 minutes) of the
# real-time clock. Note that it can’ t be used along with the 'rtcfile' directive.
rtcsync

# Step the system clock instead of slewing it if the adjustment is larger than
# one second, but only in the first three clock updates.
makestep 1 3
EOF
```

重启服务让更新一下时间

```
# systemctl restart chronyd.service
# systemctl status chronyd.service
```

```
root@K8S-M1:~# systemctl restart chronyd.service
root@K8S-M1:~# systemctl status chronyd.service
● chrony.service - chrony, an NTP client/server
   Loaded: loaded (/lib/systemd/system/chrony.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-12-03 14:22:51 CST; 6s ago
     Docs: man:chronyd(8)
           man:chronyc(1)
           man:chrony.conf(5)
  Process: 2945 ExecStart=/usr/sbin/chronyd $DAEMON_OPTS (code=exited, status=0/SUCCESS)
  Process: 2950 ExecStartPost=/usr/lib/chrony/chrony-helper update-daemon (code=exited, status=0/SUCCESS)
 Main PID: 2947 (chronyd)
    Tasks: 2 (limit: 9497)
   Memory: 1.5M
   CGroup: /system.slice/chrony.service
           ├─2947 /usr/sbin/chronyd -F -1
           └─2948 /usr/sbin/chronyd -F -1

Dec 03 14:22:51 K8S-M1 systemd[1]: Starting chrony, an NTP client/server...
Dec 03 14:22:51 K8S-M1 chronyd[2947]: chronyd version 3.4 starting (+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +SC
Dec 03 14:22:51 K8S-M1 chronyd[2947]: Frequency -2.520 +/- 31.112 ppm read from /var/lib/chrony/chrony.drift
Dec 03 14:22:51 K8S-M1 chronyd[2947]: Loaded seccomp filter
Dec 03 14:22:51 K8S-M1 systemd[1]: Started chrony, an NTP client/server.
Dec 03 14:22:56 K8S-M1 chronyd[2947]: Selected source 139.199.215.251

root@K8S-M1:~#
```

- 修改hostname

kubelet和kube-proxy上报node信息默认是取hostname的，除非通过--hostname-override指定，这里自行设置hostname

```
# hostnamectl set-hostname xxx
```

由于前面已经修改了主机名文件，所以这里不用设置了。

- docker官方的内核检查脚本建议(RHEL7/CentOS7: User namespaces disabled; add ’user_namespace.enable=1’ to boot command line),使用下面命令开启

```
grubby --args="user_namespace.enable=1" --update-kernel="$(grubby --default-kernel)"
```

Debian 10 内核已经高于centos 7内核很多。所以不用理会这里

上面4台机器都这样处理。

重启系统

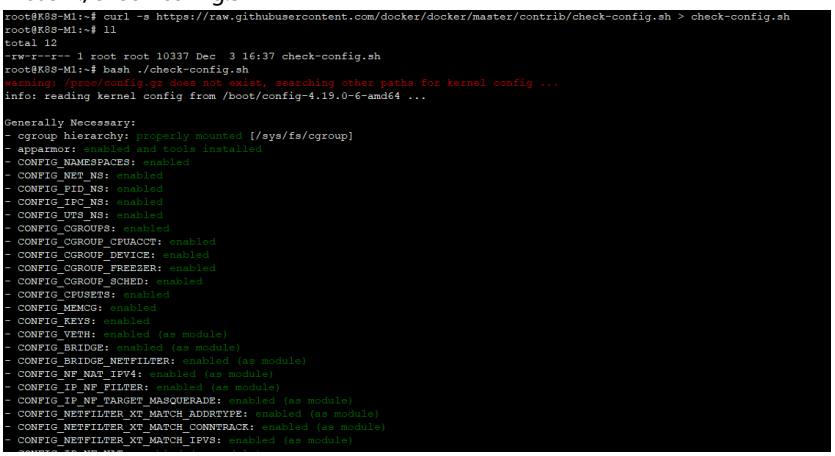
reboot

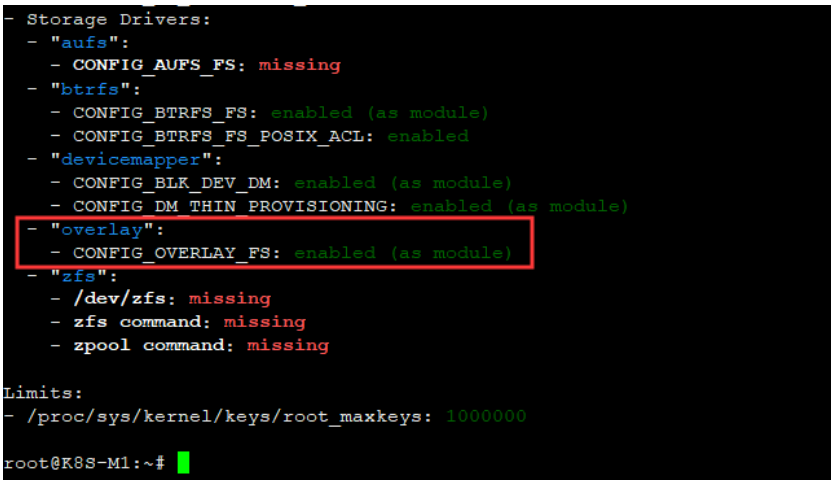
安装docker

- 检查系统内核和模块是否适合运行 docker (仅适用于 linux 系统)

curl -s https://raw.githubusercontent.com/docker/docker/master/contrib/check-config.sh > check-config.sh

bash ./check-config.sh



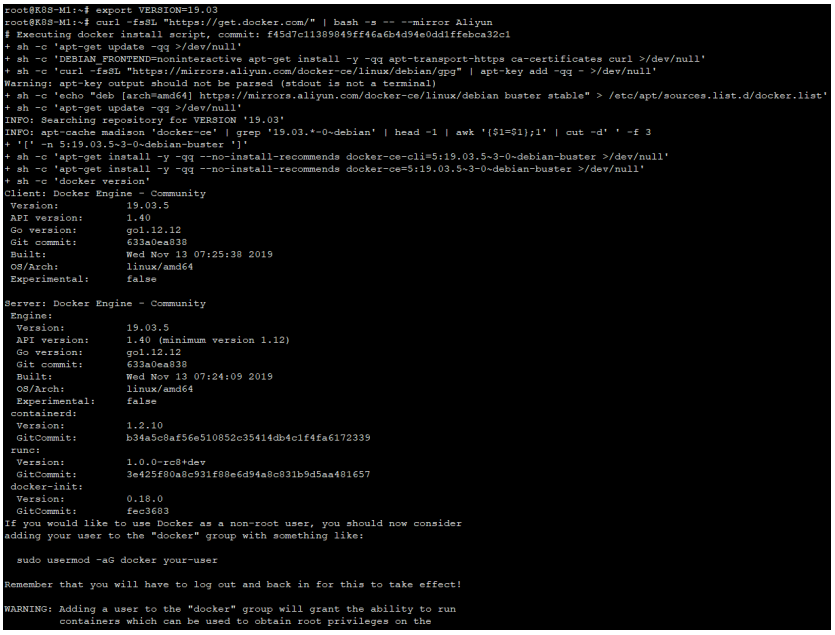


现在docker存储驱动都是使用的overlay2(不要使用devicemapper，这个坑非常多)，我们重点关注overlay2是否不是绿色

这里我们使用年份命名版本的docker-ce，假设我们要安装v1.16.3的k8s，我们去 https://github.com/kubernetes/kubernetes 里进对应版本的CHANGELOG-1.16.md里搜The list of validated docker versions remain查找支持的docker版本，docker版本不一定得在支持列表里，实际上19.03也能使用，这里我们使用docker官方的安装脚本安装docker(该脚本支持centos和ubuntu)

export VERSION=19.03

curl -fsSL "https://get.docker.com/" | bash -s -- --mirror Aliyun



所有机器配置加速源并配置docker的启动参数使用systemd,使用systemd是官方的建议,详见 https://kubernetes.io/docs/setup/cri/

```
# mkdir -p /etc/docker/
# cat>/etc/docker/daemon.json<<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "registry-mirrors": [
      "https://fz5yth0r.mirror.aliyuncs.com",
      "http://hub-mirror.c.163.com/",
      "https://docker.mirrors.ustc.edu.cn/",
      "https://registry.docker-cn.com"
  ],
  "insecure-registries": [
       "192.168.6.0/24",
       "192.168.7.0/24",
       "10.10.10.0/24"
  ],
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "3"
  }
}
EOF
# more /etc/docker/daemon.json
```

```
root@K8S-M1:~# mkdir -p /etc/docker/
root@K8S-M1:~# cat>/etc/docker/daemon.json<<EOF
> {
>   "exec-opts": ["native.cgroupdriver=systemd"],
>   "registry-mirrors": [
>       "https://fz5yth0r.mirror.aliyuncs.com",
>       "http://hub-mirror.c.163.com/",
>       "https://docker.mirrors.ustc.edu.cn/",
>       "https://registry.docker-cn.com"
>   ],
>   "storage-driver": "overlay2",
>   "storage-opts": [
>     "overlay2.override_kernel_check=true"
>   ],
>   "log-driver": "json-file",
>   "log-opts": {
>     "max-size": "100m",
>     "max-file": "3"
>   }
> }
> EOF
root@K8S-M1:~# more /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "registry-mirrors": [
      "https://fz5yth0r.mirror.aliyuncs.com",
      "http://hub-mirror.c.163.com/",
      "https://docker.mirrors.ustc.edu.cn/",
      "https://registry.docker-cn.com"
  ],
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "3"
  }
}
root@K8S-M1:~#
```

Live Restore Enabled这个千万别开，某些极端情况下容器Dead状态之类的必须重启docker daemon才能解决，开了就只能重启机器解决了

- 设置docker开机启动,CentOS安装完成后docker需要手动设置docker命令补全：

centos 版本：

yum install -y epel-release bash-completion && \

cp /usr/share/bash-completion/completions/docker /etc/bash_completion.d/

Debian 版本：

apt install bash-completion

vim /etc/bash.bashrc

取消下面的这段注释。

```
# enable bash completion in interactive shells
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
```

cp /usr/share/bash-completion/completions/docker /etc/bash_completion.d/

```
root@K8S-M1:~# apt install bash-completion
Reading package lists... Done
Building dependency tree
Reading state information... Done
bash-completion is already the newest version (1:2.8-6).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@K8S-M1:~# vim /etc/bash.bashrc
root@K8S-M1:~# ll /etc/bash_completion
bash_completion    bash_completion.d/
root@K8S-M1:~# ll /etc/bash_completion.d/
total 4
-rw-r--r-- 1 root root 439 Jan 22  2019 git-prompt
root@K8S-M1:~# cp /usr/share/bash-completion/completions/docker /etc/bash_completion.d/
root@K8S-M1:~# ll /etc/bash_completion.d/
total 120
-rw-r--r-- 1 root root 116282 Dec  3 17:27 docker
-rw-r--r-- 1 root root    439 Jan 22  2019 git-prompt
root@K8S-M1:~#
```

- 防止FORWARD的DROP策略影响转发,给docker daemon添加下列参数修正，当然暴力点也可以`iptables -P FORWARD ACCEPT`

mkdir -p /etc/systemd/system/docker.service.d/

cat>/etc/systemd/system/docker.service.d/10-docker.conf<<EOF

[Service]

ExecStartPost=/sbin/iptables -I FORWARD -s 0.0.0.0/0 -j ACCEPT

ExecStopPost=/bin/bash -c '/sbin/iptables -D FORWARD -s 0.0.0.0/0 -j ACCEPT &> /dev/null || :'

EOF

more /etc/systemd/system/docker.service.d/10-docker.conf

```
root@K8S-M1:~# mkdir -p /etc/systemd/system/docker.service.d/
root@K8S-M1:~# cat>/etc/systemd/system/docker.service.d/10-docker.conf<<EOF
> [Service]
> ExecStartPost=/sbin/iptables -I FORWARD -s 0.0.0.0/0 -j ACCEPT
> ExecStopPost=/bin/bash -c '/sbin/iptables -D FORWARD -s 0.0.0.0/0 -j ACCEPT &> /dev/null || :'
> EOF
root@K8S-M1:~# more /etc/systemd/system/docker.service.d/10-docker.conf
[Service]
ExecStartPost=/sbin/iptables -I FORWARD -s 0.0.0.0/0 -j ACCEPT
ExecStopPost=/bin/bash -c '/sbin/iptables -D FORWARD -s 0.0.0.0/0 -j ACCEPT &> /dev/null || :'
root@K8S-M1:~#
```

启动docker并看下信息是否正常

```
root@K8S-M1:~# systemctl enable --now docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
root@K8S-M1:~#
```

docker info

```
root@K8S-M1:~# docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.5
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
 runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
 init version: fec3683
 Security Options:
  apparmor
  seccomp
   Profile: default
 Kernel Version: 4.19.0-6-amd64
 Operating System: Debian GNU/Linux 10 (buster)
 OSType: linux
 Architecture: x86_64
 CPUs: 2
 Total Memory: 7.769GiB
 Name: K8S-M1
 ID: XUVQ:GSUJ:HRVR:AV5F:JDD2:GQO7:FUES:MNYN:AOHI:DGWI:MVNS:Q5F5
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

WARNING: No swap limit support
root@K8S-M1:~#
```

如果enable docker的时候报错开启debug，如何开见 https://github.com/zhangguanzhang/Kubernetes-ansible/wiki/systemctl-running-debug

kubeadm部署

安装kubeadm相关

默认源在国外会无法安装，我们使用国内的镜像源，所有机器都要操作

阿里云设置K8S源

CentOS / RHEL / Fedora

cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/

enabled=1

gpgcheck=1

repo_gpgcheck=1

gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-

package-key.gpg
EOF
setenforce 0
yum install -y kubelet kubeadm kubectl
systemctl enable kubelet && systemctl start kubelet

Debian / Ubuntu

apt-get update && apt-get install -y apt-transport-https
curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
EOF
more /etc/apt/sources.list.d/kubernetes.list
apt-get update
apt-get install -y kubelet kubeadm kubectl

```
root@K8S-M1:~# apt-get update && apt-get install -y apt-transport-https
Hit:1 https://mirrors.aliyun.com/docker-ce/linux/debian buster InRelease
Hit:2 http://mirrors.163.com/debian buster InRelease
Hit:3 http://mirrors.163.com/debian-security stretch/updates InRelease
Hit:4 http://mirrors.163.com/debian buster-updates InRelease
Hit:5 http://mirrors.163.com/debian buster-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
apt-transport-https is already the newest version (1.8.2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@K8S-M1:~# curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | apt-key add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   659  100   659    0     0    255      0  0:00:02  0:00:02 --:--:--   255
OK
```

```
root@K8S-M1:~# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
> deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
> EOF
root@K8S-M1:~# more /etc/apt/sources.list.d/kubernetes.list
deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
root@K8S-M1:~#
```

```
root@K8S-M1:~# apt-get update
Hit:1 https://mirrors.aliyun.com/docker-ce/linux/debian buster InRelease
Get:2 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial InRelease [8,993 B]
Ign:3 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages
Get:3 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 Packages [31.3 kB]
Hit:4 http://mirrors.163.com/debian buster InRelease
Get:5 http://mirrors.163.com/debian-security stretch/updates InRelease [94.3 kB]
Get:6 http://mirrors.163.com/debian buster-updates InRelease [49.3 kB]
Get:7 http://mirrors.163.com/debian buster-backports InRelease [46.7 kB]
Get:8 http://mirrors.163.com/debian buster-backports/main Sources.diff/Index [27.8 kB]
Get:9 http://mirrors.163.com/debian buster-backports/main amd64 Packages.diff/Index [27.8 kB]
Get:10 http://mirrors.163.com/debian buster-backports/main Sources 2019-12-03-0217.31.pdiff [40 B]
Get:11 http://mirrors.163.com/debian buster-backports/main Sources 2019-12-03-0817.40.pdiff [835 B]
Get:12 http://mirrors.163.com/debian buster-backports/main Sources 2019-12-03-1415.56.pdiff [706 B]
Get:13 http://mirrors.163.com/debian buster-backports/main Sources 2019-12-04-0217.42.pdiff [1,865 B]
Get:14 http://mirrors.163.com/debian buster-backports/main Sources 2019-12-04-2017.40.pdiff [629 B]
Get:15 http://mirrors.163.com/debian buster-backports/main Sources 2019-12-05-0216.51.pdiff [31 B]
Get:16 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-12-03-0217.31.pdiff [270 B]
Get:17 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-12-03-1415.56.pdiff [289 B]
Get:15 http://mirrors.163.com/debian buster-backports/main Sources 2019-12-05-0216.51.pdiff [31 B]
Get:18 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-12-04-0217.42.pdiff [189 B]
Get:19 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-12-04-0817.23.pdiff [1,465 B]
Get:20 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-12-05-0216.51.pdiff [299 B]
Get:20 http://mirrors.163.com/debian buster-backports/main amd64 Packages 2019-12-05-0216.51.pdiff [299 B]
Fetched 293 kB in 18s (15.9 kB/s)
Reading package lists... Done
```

该服务kubelet自动已经设置为开机启动了。

```
root@K8S-M1:~# apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cri-tools ebtables ethtool kubernetes-cni
The following NEW packages will be installed:
  cri-tools ebtables ethtool kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 54.2 MB of archives.
After this operation, 291 MB of additional disk space will be used.
Get:1 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 cri-tools amd64 1.13.0-00 [8,776 kB]
Get:2 http://mirrors.163.com/debian buster/main amd64 ebtables amd64 2.0.10.4+snapshot20181205-3 [86.0 kB]
Get:3 http://mirrors.163.com/debian buster/main amd64 ethtool amd64 1:4.19-1 [121 kB]
Get:4 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 kubernetes-cni amd64 0.7.5-00 [6,473 kB]
Get:5 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 kubelet amd64 1.16.3-00 [20.7 MB]
Get:6 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 kubectl amd64 1.16.3-00 [9,233 kB]
Get:7 https://mirrors.aliyun.com/kubernetes/apt kubernetes-xenial/main amd64 kubeadm amd64 1.16.3-00 [8,762 kB]
Fetched 54.2 MB in 5s (10.6 MB/s)
Selecting previously unselected package cri-tools.
(Reading database ... 137240 files and directories currently installed.)
Preparing to unpack .../0-cri-tools_1.13.0-00_amd64.deb ...
Unpacking cri-tools (1.13.0-00) ...
Selecting previously unselected package ebtables.
Preparing to unpack .../1-ebtables_2.0.10.4+snapshot20181205-3_amd64.deb ...
Unpacking ebtables (2.0.10.4+snapshot20181205-3) ...
Selecting previously unselected package ethtool.
Preparing to unpack .../2-ethtool_1%3a4.19-1_amd64.deb ...
Unpacking ethtool (1:4.19-1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../3-kubernetes-cni_0.7.5-00_amd64.deb ...
Unpacking kubernetes-cni (0.7.5-00) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../4-kubelet_1.16.3-00_amd64.deb ...
Unpacking kubelet (1.16.3-00) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../5-kubectl_1.16.3-00_amd64.deb ...
Unpacking kubectl (1.16.3-00) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../6-kubeadm_1.16.3-00_amd64.deb ...
Unpacking kubeadm (1.16.3-00) ...
Setting up kubectl (1.16.3-00) ...
Setting up ebtables (2.0.10.4+snapshot20181205-3) ...
Setting up cri-tools (1.13.0-00) ...
Setting up kubernetes-cni (0.7.5-00) ...
Setting up ethtool (1:4.19-1) ...
Setting up kubelet (1.16.3-00) ...
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /lib/systemd/system/kubelet.service.
Setting up kubeadm (1.16.3-00) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10) ...
```

```
root@K8S-M1:~# systemctl is-enabled kubelet.service
enabled
root@K8S-M1:~#
```

k8s的node就是kubelet+cri(一般是docker)，kubectl是一个agent读取kubeconfig去访问kube-apiserver来操作集群，kubeadm是部署，所以master节点需要安装三个，node一般不需要kubectl

master部分

安装相关软件

yum install -y \
  kubeadm-1.16.3 \
  kubectl-1.16.3 \
  kubelet-1.16.3 \
  --disableexcludes=kubernetes && \
  systemctl enable kubelet

前面已经安装了，这里跳过

配置kubelet(所有节点)

查看kubelet的systemd文件

systemctl cat kubelet.service

```
root@K8S-M1:~# systemctl cat kubelet.service
# /lib/systemd/system/kubelet.service
[Unit]
Description=kubelet: The Kubernetes Node Agent
Documentation=https://kubernetes.io/docs/home/

[Service]
ExecStart=/usr/bin/kubelet
Restart=always
StartLimitInterval=0
RestartSec=10

[Install]
WantedBy=multi-user.target

# /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the KUBELET_KUBEADM_ARGS variable dynamically
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. Preferably, the user should use
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_EXTRA_ARGS should be sourced from this file.
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
root@K8S-M1:~#
```

我们可以看到/etc/sysconfig/kubelet是EnvironmentFile，里面注释也写明了我们应该在该文件里写KUBELET_EXTRA_ARGS来给kubelet配置运行参数,下面是个例子，具体参数啥的可以kubelet –help看

centos 系列

cat >/etc/sysconfig/kubelet<<EOF

KUBELET_EXTRA_ARGS="--xxx=yyy --aaa=bbb"

EOF

Debian系列

cat >/etc/default/kubelet<<EOF
KUBELET_EXTRA_ARGS="--xxx=yyy --aaa=bbb"
EOF

配置HA,所以机器

关于HA我博客 https://zhangguanzhang.github.io/2019/03/11/k8s-ha/ 说得很清楚，这里我用local proxy来玩，因为localproxy是每台机器上的，可以不用SLB和vpc无法使用vip的限制,需要每个机器上运行nginx实现

每台机器配置hosts

cat >>/etc/hosts << EOF
127.0.0.1 apiserver.k8s.local
10.10.10.91 apiserver01.k8s.local
10.10.10.92 apiserver02.k8s.local
10.10.10.93 apiserver03.k8s.local
EOF

```
root@K8S-M1:~# cat >>/etc/hosts << EOF
> 127.0.0.1 apiserver.k8s.local
> 10.10.10.91 apiserver01.k8s.local
> 10.10.10.92 apiserver02.k8s.local
> 10.10.10.93 apiserver03.k8s.local
> EOF
root@K8S-M1:~# more /etc/hosts
127.0.0.1       localhost
127.0.1.1       localhost

10.10.10.91 k8s-m1
10.10.10.92 k8s-m2
10.10.10.93 k8s-m3
10.10.10.94 k8s-n1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1 apiserver.k8s.local
10.10.10.91 apiserver01.k8s.local
10.10.10.92 apiserver02.k8s.local
10.10.10.93 apiserver03.k8s.local
root@K8S-M1:~#
```

每台机器生成配置文件，上面的三个hosts可以不写，写下面配置文件里域名写ip即可，但是这样更改ip需要重新加载，三台主机上都执行

```
# mkdir -p /etc/kubernetes
# cat > /etc/kubernetes/nginx.conf << EOF
user nginx nginx;
worker_processes auto;
events {
    worker_connections  20240;
    use epoll;
}
error_log /var/log/nginx_error.log info;

stream {
    upstream kube-servers {
        hash $remote_addr consistent;
        server apiserver01.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
        server apiserver02.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
        server apiserver03.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
    }

    server {
        listen 8443 reuseport;
        proxy_connect_timeout 3s;
        # 加大timeout
        proxy_timeout 3000s;
```

```
        proxy_pass kube-servers;
    }
}
EOF
```

```
root@K8S-M1:~# mkdir -p /etc/kubernetes
root@K8S-M1:~# cat > /etc/kubernetes/nginx.conf << EOF
> user nginx nginx;
> worker_processes auto;
> events {
>     worker_connections  20240;
>     use epoll;
> }
> error_log /var/log/nginx_error.log info;
>
> stream {
>     upstream kube-servers {
>         hash $remote_addr consistent;
>         server apiserver01.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
>         server apiserver02.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
>         server apiserver03.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
>     }
>
>     server {
>         listen 8443 reuseport;
>         proxy_connect_timeout 3s;
>         # 加大timeout
>         proxy_timeout 3000s;
>         proxy_pass kube-servers;
>     }
> }
> EOF
root@K8S-M1:~#
```

因为localproxy是每台机器上的，可以不用SLB和vpc无法使用vip的限制，这里我使用staticPod创建

```
# docker run --restart=always \
    -v /etc/kubernetes/nginx.conf:/etc/nginx/nginx.conf \
    -v /etc/localtime:/etc/localtime:ro \
    --name k8s \
    --net host \
    -d \
    nginx:alpine
# docker ps
```

```
root@K8S-M1:~# docker run --restart=always \
>     -v /etc/kubernetes/nginx.conf:/etc/nginx/nginx.conf \
>     -v /etc/localtime:/etc/localtime:ro \
>     --name k8s \
>     --net host \
>     -d \
>     nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
89d9c30c1d48: Pull complete
24f1c4f0b2f4: Pull complete
Digest: sha256:0e61b143db3110f3b8ae29a67f107d5536b71a7c1f10afb14d4228711fc65a13
Status: Downloaded newer image for nginx:alpine
40c53723f1774ef0a095a70842549449dcbc189e207b14ce5107d7fb11e910fb
root@K8S-M1:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
40c53723f177        nginx:alpine        "nginx -g 'daemon of…"   23 seconds ago      Up 22 seconds                           k8s
root@K8S-M1:~#
```

配置集群信息(第一个master上配置)

- 打印默认init的配置信息

```
# kubeadm config print init-defaults > initconfig.yaml
```

```
root@K8S-M1:~# kubeadm config print init-defaults > initconfig.yaml
```

我们看下默认init的集群参数

```
# more initconfig.yaml
apiVersion: kubeadm.k8s.io/v1beta2
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 1.2.3.4
```

```
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: k8s-m1
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta2
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: k8s.gcr.io
kind: ClusterConfiguration
kubernetesVersion: v1.16.0
networking:
  dnsDomain: cluster.local
  serviceSubnet: 10.96.0.0/12
scheduler: {}
```

```
root@K8S-M1:~# more initconfig.yaml
apiVersion: kubeadm.k8s.io/v1beta2
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 1.2.3.4
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: k8s-m1
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta2
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: k8s.gcr.io
kind: ClusterConfiguration
kubernetesVersion: v1.16.0
networking:
  dnsDomain: cluster.local
  serviceSubnet: 10.96.0.0/12
scheduler: {}
root@K8S-M1:~#
```

我们得修改下，可以参考下列的v1beta2文档,如果是低版本可能是v1beta1，某些字段和新的是不一样的，自行查找godoc看

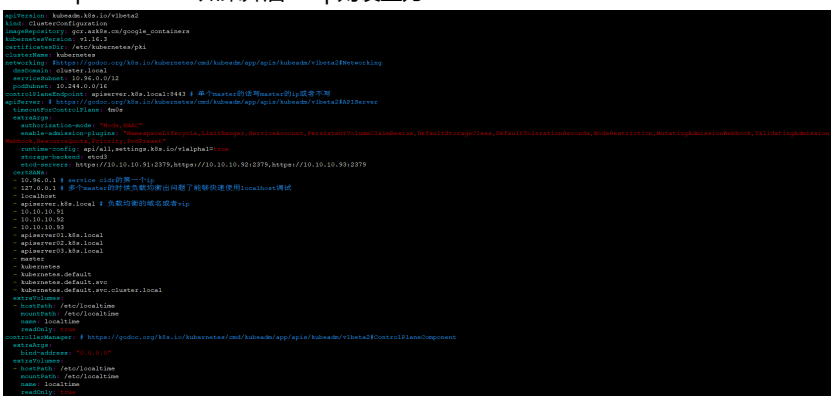
https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#hdr-Basics

https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2
https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#pkg-constants
https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#ClusterConfiguration
ip啥的自行更改成和自己的一致，cidr不懂咋计算就别乱改。controlPlaneEndpoint写域名(内网没dns所有机器写hosts也行)或者SLB，VIP，原因和注意事项见 https://zhangguanzhang.github.io/2019/03/11/k8s-ha/ 这个文章我把HA解释得很清楚了，不要再问我了

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
imageRepository: gcr.azk8s.cn/google_containers
kubernetesVersion: v1.16.3
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
networking: #https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#Networking
  dnsDomain: cluster.local
  serviceSubnet: 10.96.0.0/12
  podSubnet: 10.244.0.0/16
controlPlaneEndpoint: apiserver.k8s.local:8443 # 单个master的话写master的ip或者不写
apiServer: # https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#APIServer
  timeoutForControlPlane: 4m0s
  extraArgs:
    authorization-mode: "Node,RBAC"
    enable-admission-plugins:
"NamespaceLifecycle,LimitRanger,ServiceAccount,PersistentVolumeClaimResize,DefaultStorageClass,DefaultTolerationSeconds,N
    runtime-config: api/all,settings.k8s.io/v1alpha1=true
    storage-backend: etcd3
    etcd-servers: https://10.10.10.91:2379,https://10.10.10.92:2379,https://10.10.10.93:2379
  certSANs:
  - 10.96.0.1 # service cidr的第一个ip
  - 127.0.0.1 # 多个master的时候负载均衡出问题了能够快速使用localhost调试
  - localhost
  - apiserver.k8s.local # 负载均衡的域名或者vip
  - 10.10.10.91
  - 10.10.10.92
  - 10.10.10.93
  - apiserver01.k8s.local
  - apiserver02.k8s.local
  - apiserver03.k8s.local
  - master
  - kubernetes
  - kubernetes.default
  - kubernetes.default.svc
  - kubernetes.default.svc.cluster.local
  extraVolumes:
  - hostPath: /etc/localtime
    mountPath: /etc/localtime
    name: localtime
    readOnly: true
controllerManager: #
https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#ControlPlaneComponent
  extraArgs:
    bind-address: "0.0.0.0"
  extraVolumes:
  - hostPath: /etc/localtime
```

```
    mountPath: /etc/localtime
    name: localtime
    readOnly: true
scheduler:
  extraArgs:
    bind-address: "0.0.0.0"
  extraVolumes:
  - hostPath: /etc/localtime
    mountPath: /etc/localtime
    name: localtime
    readOnly: true
dns: # https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#DNS
  type: CoreDNS # or kube-dns
  # imageRepository: coredns  #这里可以用上面的仓库，不过就是1.6.2的。这里统一用几个仓库，好替换。
  imageRepository: gcr.azk8s.cn/google_containers
  imageTag: 1.6.2
etcd: # https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#Etcd
  local:
    imageRepository: gcr.azk8s.cn/google_containers
    imageTag: 3.3.17
    dataDir: /var/lib/etcd
    serverCertSANs:
    - master
    - localhost
    - 10.10.10.91
    - 10.10.10.92
    - 10.10.10.93
    - etcd01.k8s.local
    - etcd02.k8s.local
    - etcd03.k8s.local
    peerCertSANs:
    - master
    - localhost
    - 10.10.10.91
    - 10.10.10.92
    - 10.10.10.93
    - etcd01.k8s.local
    - etcd02.k8s.local
    - etcd03.k8s.local
    extraArgs: # 暂时没有extraVolumes
      auto-compaction-retention: "1h"
      max-request-bytes: "33554432"
      quota-backend-bytes: "8589934592"
      enable-v2: "false" # disable etcd v2 api
  # external: //外部etcd的时候这样配置 https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#Etcd
    # endpoints:
    # - "10.10.10.91:2379"
    # - "10.10.10.92:2379"
    # - "10.10.10.93:2379"
    # caFile: "/etc/kubernetes/pki/etcd/etcd-ca.crt"
    # certFile: "/etc/kubernetes/pki/etcd/etcd.crt"
    # keyFile: "/etc/kubernetes/pki/etcd/etcd.key"
```

```
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration # https://godoc.org/k8s.io/kube-proxy/config/v1alpha1#KubeProxyConfiguration
mode: ipvs # or iptables
ipvs:
  excludeCIDRs: null
  minSyncPeriod: 0s
  scheduler: ""
  strictARP: false
  syncPeriod: 15s
iptables:
  masqueradeAll: true
  masqueradeBit: 14
  minSyncPeriod: 0s
  syncPeriod: 30s
---
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration # https://godoc.org/k8s.io/kubelet/config/v1beta1#KubeletConfiguration
cgroupDriver: systemd
failSwapOn: true # 如果开启swap则设置为false
```



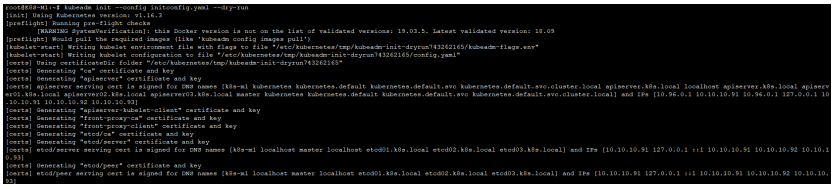
```
scheduler:
  extraArgs:
    bind-address: "0.0.0.0"
  extraVolumes:
  - hostPath: /etc/localtime
    mountPath: /etc/localtime
    name: localtime
    readOnly: true
dns: # https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#DNS
  type: CoreDNS # or kube-dns
  # imageRepository: coredns  #这里可以用上面的仓库，不过就是1.6.2的。这里统一用几个仓库，好替换。
  imageRepository: gcr.azk8s.cn/google_containers
  imageTag: 1.6.2
etcd: # https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#Etcd
  local:
    imageRepository: gcr.azk8s.cn/google_containers
    imageTag: 3.3.17
    dataDir: /var/lib/etcd
    serverCertSANs:
    - master
    - localhost
    - 10.10.10.91
    - 10.10.10.92
    - 10.10.10.93
    - etcd01.k8s.local
    - etcd02.k8s.local
    - etcd03.k8s.local
    peerCertSANs:
    - master
    - localhost
    - 10.10.10.91
    - 10.10.10.92
    - 10.10.10.93
    - etcd01.k8s.local
    - etcd02.k8s.local
    - etcd03.k8s.local
    extraArgs: # 暂时没有extraVolumes
      auto-compaction-retention: "1h"
      max-request-bytes: "33554432"
      quota-backend-bytes: "8589934592"
      enable-v2: "false" # disable etcd v2 api
  # external: //外部etcd的时候这样配置 https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2#Etcd
    # endpoints:
    # - "10.10.10.91:2379"
    # - "10.10.10.92:2379"
    # - "10.10.10.93:2379"
    # caFile: "/etc/kubernetes/pki/etcd/etcd-ca.crt"
    # certFile: "/etc/kubernetes/pki/etcd/etcd.crt"
    # keyFile: "/etc/kubernetes/pki/etcd/etcd.key"
```

```
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration # https://godoc.org/k8s.io/kube-proxy/config/v1alpha1#KubeProxyConfiguration
mode: ipvs # or iptables
ipvs:
  excludeCIDRs: null
  minSyncPeriod: 0s
  scheduler: ""
  strictARP: false
  syncPeriod: 15s
iptables:
  masqueradeAll: true
  masqueradeBit: 14
  minSyncPeriod: 0s
  syncPeriod: 30s
---
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration # https://godoc.org/k8s.io/kubelet/config/v1beta1#KubeletConfiguration
cgroupDriver: systemd
failSwapOn: true # 如果开启swap则设置为false
```

swap的话看最后一行，apiserver的exterArgs是为了开启podPreset

- 检查文件是否错误，忽略warning错误的话会抛出error，没错则会输出到包含字符串kubeadm join xxx啥的

kubeadm init --config initconfig.yaml --dry-run



- 检查镜像是否正确

kubeadm config images list --config initconfig.yaml

```
root@K8S-M1:~# kubeadm config images list --config initconfig.yaml
gcr.azk8s.cn/google_containers/kube-apiserver:v1.16.3
gcr.azk8s.cn/google_containers/kube-controller-manager:v1.16.3
gcr.azk8s.cn/google_containers/kube-scheduler:v1.16.3
gcr.azk8s.cn/google_containers/kube-proxy:v1.16.3
gcr.azk8s.cn/google_containers/pause:3.1
gcr.azk8s.cn/google_containers/etcd:3.3.17
gcr.azk8s.cn/google_containers/coredns:1.6.2
root@K8S-M1:~#
```

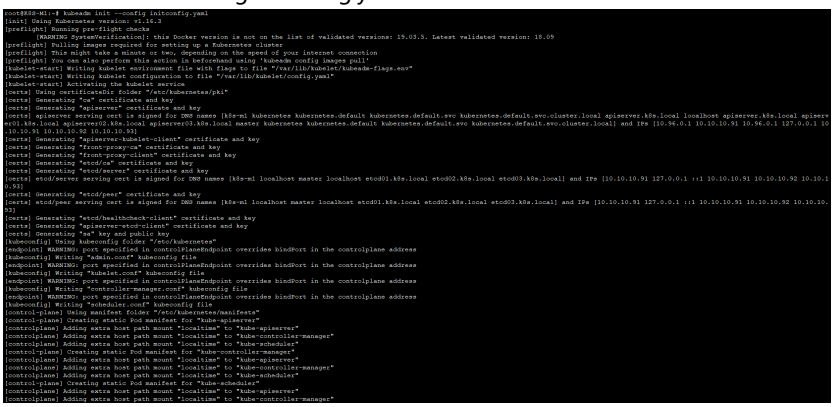
- 预先拉取镜像

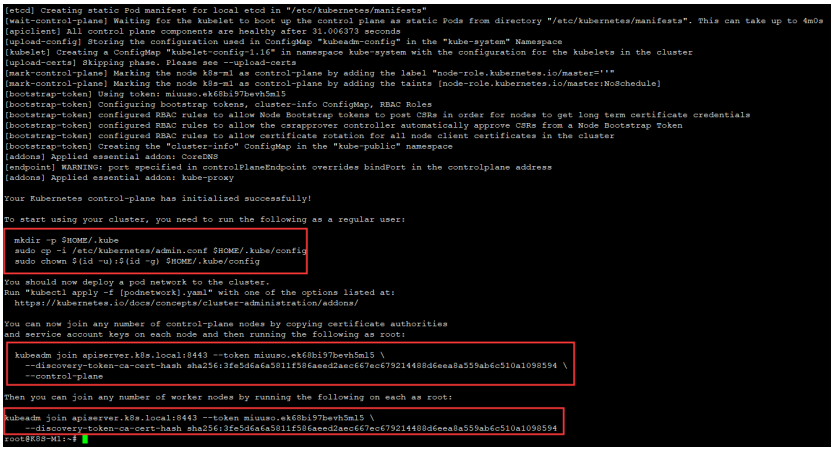
```
root@K8S-M1:~# kubeadm config images pull --config initconfig.yaml
[config/images] Pulled gcr.azk8s.cn/google_containers/kube-apiserver:v1.16.3
[config/images] Pulled gcr.azk8s.cn/google_containers/kube-controller-manager:v1.16.3
[config/images] Pulled gcr.azk8s.cn/google_containers/kube-scheduler:v1.16.3
[config/images] Pulled gcr.azk8s.cn/google_containers/kube-proxy:v1.16.3
[config/images] Pulled gcr.azk8s.cn/google_containers/pause:3.1
[config/images] Pulled gcr.azk8s.cn/google_containers/etcd:3.3.17
[config/images] Pulled gcr.azk8s.cn/google_containers/coredns:1.6.2
root@K8S-M1:~#
```

kubeadm init

下面init只在第一个master上面操作

kubeadm init --config initconfig.yaml





如果超时了看看是不是kubelet没起来，调试见 https://github.com/zhangguanzhang/Kubernetes-ansible/wiki/systemctl-running-debug

记住init后打印的token，复制kubectl的kubeconfig，kubectl的kubeconfig路径默认是~/.kube/config

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
root@K8S-M1:~# mkdir -p $HOME/.kube
root@K8S-M1:~# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@K8S-M1:~# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@K8S-M1:~# ll .kube/
total 8
-rw------- 1 root root 5459 Dec  6 14:45 config
root@K8S-M1:~#
```

如果单个master，也不想整其他的node，需要去掉master节点上的污点

kubectl taint nodes --all node-role.kubernetes.io/master-

下面是多个master的操作

拷贝ca证书到其他master节点上，因为交互输入密码，我们安装sshpass，zhangguanzhang是root密码

centos示例：

yum install sshpass -y

alias ssh='sshpass -p zhangguanzhang ssh -o StrictHostKeyChecking=no'

alias scp='sshpass -p zhangguanzhang scp -o StrictHostKeyChecking=no'

Debian 如下：每台机器都装一下。

apt install sshpass

```
root@K8S-M1:~# apt install sshpass
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  sshpass
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 11.3 kB of archives.
After this operation, 31.7 kB of additional disk space will be used.
Get:1 http://mirrors.163.com/debian buster/main amd64 sshpass amd64 1.06-1 [11.3 kB]
Fetched 11.3 kB in 1s (11.8 kB/s)
Selecting previously unselected package sshpass.
(Reading database ... 137298 files and directories currently installed.)
Preparing to unpack .../sshpass_1.06-1_amd64.deb ...
Unpacking sshpass (1.06-1) ...
Setting up sshpass (1.06-1) ...
Processing triggers for man-db (2.8.5-2) ...
root@K8S-M1:~#
```

alias ssh='sshpass -p hS1234.. ssh -o StrictHostKeyChecking=no'

alias scp='sshpass -p hS1234.. scp -o StrictHostKeyChecking=no'

```
root@K8S-M1:~# alias ssh='sshpass -p hS1234.. ssh -o StrictHostKeyChecking=no'
root@K8S-M1:~# alias scp='sshpass -p hS1234.. scp -o StrictHostKeyChecking=no'
```

复制ca证书到其他master节点

里面的IP是另外两个主节点的IP。

```
# for node in 10.10.10.92 10.10.10.93;do
    ssh $node 'mkdir -p /etc/kubernetes/pki/etcd'
    scp -r /etc/kubernetes/pki/ca.* $node:/etc/kubernetes/pki/
    scp -r /etc/kubernetes/pki/sa.* $node:/etc/kubernetes/pki/
    scp -r /etc/kubernetes/pki/front-proxy-ca.* $node:/etc/kubernetes/pki/
    scp -r /etc/kubernetes/pki/etcd/ca.* $node:/etc/kubernetes/pki/etcd/
done
```

```
root@K8S-M1:~# for node in 10.10.10.92 10.10.10.93;do
>     ssh $node 'mkdir -p /etc/kubernetes/pki/etcd'
>     scp -r /etc/kubernetes/pki/ca.* $node:/etc/kubernetes/pki/
>     scp -r /etc/kubernetes/pki/sa.* $node:/etc/kubernetes/pki/
>     scp -r /etc/kubernetes/pki/front-proxy-ca.* $node:/etc/kubernetes/pki/
>     scp -r /etc/kubernetes/pki/etcd/ca.* $node:/etc/kubernetes/pki/etcd/
> done
Warning: Permanently added '10.10.10.92' (ECDSA) to the list of known hosts.
Warning: Permanently added '10.10.10.93' (ECDSA) to the list of known hosts.
root@K8S-M1:~#
```

所有master配置kubectl

准备kubectl的kubeconfig，前面已经执行过了，不用执行了。

mkdir -p $HOME/.kube

sudo \cp /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

设置kubectl的补全脚本

kubectl completion bash > /etc/bash_completion.d/kubectl

```
root@K8S-M1:~# kubectl completion bash > /etc/bash_completion.d/kubectl
root@K8S-M1:~# ll /etc/bash_completion.d/
total 508
-rw-r--r-- 1 root root 116282 Dec  3 17:27 docker
-rw-r--r-- 1 root root    439 Jan 22  2019 git-prompt
-rw-r--r-- 1 root root 393965 Dec  6 15:02 kubectl
root@K8S-M1:~#
```

所有master配置etcdctl

复制出容器里的etcdctl

docker cp `docker ps -a | awk '/k8s_etcd/{print $1}'`:/usr/local/bin/etcdctl /usr/local/bin/etcdctl

ll /usr/local/bin/etcdctl -h

```
root@K8S-M1:~# docker cp `docker ps -a | awk '/k8s_etcd/{print $1}'`:/usr/local/bin/etcdctl /usr/local/bin/etcdctl
root@K8S-M1:~# ll /usr/local/bin/etcdctl -h
-r-xr-xr-x 1 root root 17M Oct 12 02:37 /usr/local/bin/etcdctl
root@K8S-M1:~#
```

1.13还是具体哪个版本后k8s默认使用v3 api的etcd，这里我们配置下etcdctl的参数

```
# cat >/etc/profile.d/etcd.sh<<'EOF'
ETCD_CERET_DIR=/etc/kubernetes/pki/etcd/
ETCD_CA_FILE=ca.crt
ETCD_KEY_FILE=healthcheck-client.key
ETCD_CERT_FILE=healthcheck-client.crt
ETCD_EP=https://10.10.10.91:2379,https://10.10.10.92:2379,https://10.10.10.93:2379

alias etcd_v2="etcdctl --cert-file ${ETCD_CERET_DIR}/${ETCD_CERT_FILE} \
          --key-file ${ETCD_CERET_DIR}/${ETCD_KEY_FILE}  \
          --ca-file ${ETCD_CERET_DIR}/${ETCD_CA_FILE}  \
          --endpoints $ETCD_EP"

alias etcd_v3="ETCDCTL_API=3 \
   etcdctl   \
  --cert ${ETCD_CERET_DIR}/${ETCD_CERT_FILE} \
  --key ${ETCD_CERET_DIR}/${ETCD_KEY_FILE} \
  --cacert ${ETCD_CERET_DIR}/${ETCD_CA_FILE} \
   --endpoints $ETCD_EP"
EOF
# more /etc/profile.d/etcd.sh
```

```
root@K8S-M1:~# cat >/etc/profile.d/etcd.sh<<'EOF'
> ETCD_CERET_DIR=/etc/kubernetes/pki/etcd/
> ETCD_CA_FILE=ca.crt
> ETCD_KEY_FILE=healthcheck-client.key
> ETCD_CERT_FILE=healthcheck-client.crt
> ETCD_EP=https://10.10.10.91:2379,https://10.10.10.92:2379,https://10.10.10.93:2379
>
> alias etcd_v2="etcdctl --cert-file ${ETCD_CERET_DIR}/${ETCD_CERT_FILE} \
>              --key-file ${ETCD_CERET_DIR}/${ETCD_KEY_FILE}  \
>              --ca-file ${ETCD_CERET_DIR}/${ETCD_CA_FILE}  \
>              --endpoints $ETCD_EP"
>
> alias etcd_v3="ETCDCTL_API=3 \
>     etcdctl   \
>    --cert ${ETCD_CERET_DIR}/${ETCD_CERT_FILE} \
>    --key ${ETCD_CERET_DIR}/${ETCD_KEY_FILE} \
>    --cacert ${ETCD_CERET_DIR}/${ETCD_CA_FILE} \
>     --endpoints $ETCD_EP"
> EOF
root@K8S-M1:~# more /etc/profile.d/etcd.sh
ETCD_CERET_DIR=/etc/kubernetes/pki/etcd/
ETCD_CA_FILE=ca.crt
ETCD_KEY_FILE=healthcheck-client.key
ETCD_CERT_FILE=healthcheck-client.crt
ETCD_EP=https://10.10.10.91:2379,https://10.10.10.92:2379,https://10.10.10.93:2379

alias etcd_v2="etcdctl --cert-file ${ETCD_CERET_DIR}/${ETCD_CERT_FILE} \
             --key-file ${ETCD_CERET_DIR}/${ETCD_KEY_FILE}  \
             --ca-file ${ETCD_CERET_DIR}/${ETCD_CA_FILE}  \
             --endpoints $ETCD_EP"

alias etcd_v3="ETCDCTL_API=3 \
    etcdctl   \
   --cert ${ETCD_CERET_DIR}/${ETCD_CERT_FILE} \
   --key ${ETCD_CERET_DIR}/${ETCD_KEY_FILE} \
   --cacert ${ETCD_CERET_DIR}/${ETCD_CA_FILE} \
    --endpoints $ETCD_EP"
root@K8S-M1:~#
```

重新ssh下或者手动加载下环境变量. `/etc/profile.d/etcd.sh`

source /etc/profile.d/etcd.sh

etcd_v3 endpoint status --write-out=table

```
root@K8S-M1:~# etcd_v3 endpoint status --write-out=table
+--------------------------+------------------+---------+---------+-----------+-----------+------------+
|         ENDPOINT         |        ID        | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
+--------------------------+------------------+---------+---------+-----------+-----------+------------+
| https://10.10.10.91:2379 | b184042d22d93475 |  3.3.17 |  1.6 MB |      true |        12 |      12488 |
| https://10.10.10.92:2379 | aa32f71742d65ad3 |  3.3.17 |  1.6 MB |     false |        12 |      12488 |
| https://10.10.10.93:2379 | be9dbcf2230849f4 |  3.3.17 |  1.6 MB |     false |        12 |      12488 |
+--------------------------+------------------+---------+---------+-----------+-----------+------------+
root@K8S-M1:~#
```

配置etcd备份脚本

mkdir -p /opt/etcd

cat>/opt/etcd/etcd_cron.sh<<'EOF'

```
#!/bin/bash
set -e

export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

: ${bak_dir:=/root/} #缺省备份目录,可以修改成存在的目录
: ${cert_dir:=/etc/kubernetes/pki/etcd/}
: ${endpoints:=https://10.10.10.91:2379,https://10.10.10.92:2379,https://10.10.10.93:2379

bak_prefix='etcd-'
cmd_suffix='date +%Y-%m-%d-%H:%M'
bak_suffix='.db'

#将规范化后的命令行参数分配至位置参数（$1,$2,...)
temp=`getopt -n $0 -o c:d: -u -- "$@"`

[ $? != 0 ] && {
    echo '
Examples:
  # just save once
  bash $0 /tmp/etcd.db
  # save in contab and  keep 5
  bash $0 -c 5
    '
    exit 1
```

```
    }
set -- $temp


# -c 备份保留副本数量
# -d 指定备份存放目录
while true;do
    case "$1" in
        -c)
            [ -z "$bak_count" ] && bak_count=$2
            printf -v null %d "$bak_count" &>/dev/null || \
                { echo 'the value of the -c must be number';exit 1; }
            shift 2
            ;;
        -d)
            [ ! -d "$2" ] && mkdir -p $2
            bak_dir=$2
            shift 2
            ;;
         *)
            [[ -z "$1" || "$1" == '--' ]] && { shift;break; }
            echo "Internal error!"
            exit 1
            ;;
    esac
done


function etcd_v2(){

    etcdctl --cert-file $cert_dir/healthcheck-client.crt \
            --key-file  $cert_dir/healthcheck-client.key \
            --ca-file   $cert_dir/ca.crt \
        --endpoints $endpoints $@
}

function etcd_v3(){

    ETCDCTL_API=3 etcdctl   \
      --cert $cert_dir/healthcheck-client.crt \
      --key  $cert_dir/healthcheck-client.key \
      --cacert $cert_dir/ca.crt \
      --endpoints $endpoints $@
}

etcd::cron::save(){
    cd $bak_dir/
    etcd_v3 snapshot save  $bak_prefix$($cmd_suffix)$bak_suffix
    rm_files=`ls -t $bak_prefix*$bak_suffix | tail -n +$[bak_count+1]`
    if [ -n "$rm_files" ];then
        rm -f $rm_files
    fi
```

}

main(){
    [-n "$bak_count"] && etcd::cron::save || etcd_v3 snapshot save $@
}

main $@
EOF

```
root@K8S-M1:~# mkdir -p /opt/etcd
root@K8S-M1:~# cat>/opt/etcd/etcd_cron.sh<<'EOF'
> #!/bin/bash
> set -e
>
> export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
>
> :  ${bak_dir:=/root/} #缺省备份目录,可以修改成存在的目录
> :  ${cert_dir:=/etc/kubernetes/pki/etcd/}
> :  ${endpoints:=https://10.10.10.91:2379,https://10.10.10.92:2379,https://10.10.10.93:2379
>
> bak_prefix='etcd-'
> cmd_suffix='date +%Y-%m-%d-%H:%M'
> bak_suffix='.db'
>
> #将规范化后的命令行参数分配至位置参数（$1,$2,...)
> temp=`getopt -n $0 -o c:d: -u -- "$@"`
>
> [ $? != 0 ] && {
>     echo '
> Examples:
>   # just save once
>   bash $0 /tmp/etcd.db
>   # save in contab and  keep 5
>   bash $0 -c 5
>     '
>     exit 1
>     }
> set -- $temp
>
>
> # -c 备份保留副本数量
> # -d 指定备份存放目录
> while true;do
>     case "$1" in
>         -c)
>             [ -z "$bak_count" ] && bak_count=$2
>             printf -v null %d "$bak_count" &>/dev/null || \
>                 { echo 'the value of the -c must be number';exit 1; }
>             shift 2
>             ;;
>         -d)
>             [ ! -d "$2" ] && mkdir -p $2
>             bak_dir=$2
>             shift 2
>             ;;
>          *)
>             [[ -z "$1" || "$1" == '--' ]] && { shift;break; }
>             echo "Internal error!"
>             exit 1
>             ;;
```

```
>     esac
> done
>
>
> function etcd_v2(){
>
>     etcdctl --cert-file $cert_dir/healthcheck-client.crt \
>             --key-file  $cert_dir/healthcheck-client.key \
>             --ca-file   $cert_dir/ca.crt \
>         --endpoints $endpoints $@
> }
>
> function etcd_v3(){
>
>     ETCDCTL_API=3 etcdctl   \
>        --cert $cert_dir/healthcheck-client.crt \
>        --key  $cert_dir/healthcheck-client.key \
>        --cacert $cert_dir/ca.crt \
>        --endpoints $endpoints $@
> }
>
> etcd::cron::save(){
>     cd $bak_dir/
>     etcd_v3 snapshot save  $bak_prefix$($cmd_suffix)$bak_suffix
>     rm_files=`ls -t $bak_prefix*$bak_suffix | tail -n +$[bak_count+1]`
>     if [ -n "$rm_files" ];then
>         rm -f $rm_files
>     fi
> }
>
> main(){
>     [ -n "$bak_count" ] && etcd::cron::save || etcd_v3 snapshot save $@
> }
>
> main $@
> EOF
root@K8S-M1:~#
```

crontab -e添加下面内容自动保留四个备份副本,这里也可以放置在systemd的定时任务里面。

vim /etc/crontab

more /etc/crontab

* */1 * * * root bash /opt/etcd/etcd_cron.sh -c 4 -d /opt/etcd/ &>/dev/null

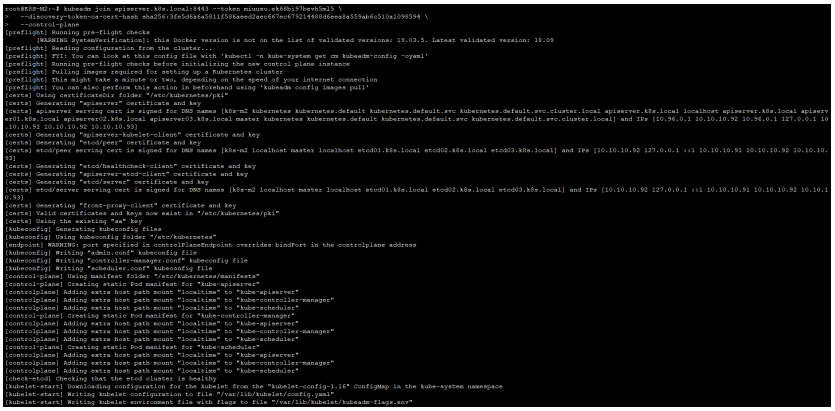
```
root@K8S-M1:~# vim /etc/crontab
root@K8S-M1:~# more /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* */1 * * *  root bash /opt/etcd/etcd_cron.sh  -c 4 -d /opt/etcd/ &>/dev/null
```

其他master join进来，master2和master3上执行同样的操作。

kubeadm join apiserver.k8s.local:8443 --token miuuso.ek68bi97bevh5ml5 \

  --discovery-token-ca-cert-hash sha256:3fe5d6a6a5811f586aeed2aec667ec679214488d6eea8a559ab6c510a1098594 \

  --control-plane



```
[kubelet-start] Activating the kubelet service
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
[etcd] Announced new etcd member joining to the existing etcd cluster
[etcd] Creating static Pod manifest for "etcd"
[etcd] Waiting for the new etcd member to join the cluster. This can take up to 40s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[mark-control-plane] Marking the node k8s-m2 as control-plane by adding the label "node-role.kubernetes.io/master=''"
[mark-control-plane] Marking the node k8s-m2 as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]

This node has joined the cluster and a new control plane instance was created:

* Certificate signing request was sent to apiserver and approval was received.
* The Kubelet was informed of the new secure connection details.
* Control plane (master) label and taint were applied to the new node.
* The Kubernetes control plane instances scaled up.
* A new etcd member was added to the local/stacked etcd cluster.

To start administering your cluster from this node, 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

Run 'kubectl get nodes' to see this node join the cluster.

root@K8S-M2:~#
```

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
root@K8S-M2:~# mkdir -p $HOME/.kube
root@K8S-M2:~# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@K8S-M2:~# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@K8S-M2:~# ll .kube/
total 8
-rw------- 1 root root 5459 Dec  6 15:09 config
root@K8S-M2:~#
```

token忘记的话可以kubeadm token list查看，可以通过kubeadm token create创建

sha256的值可以通过下列命令获取

openssl x509 -pubkey -in \

/etc/kubernetes/pki/ca.crt | \

openssl rsa -pubin -outform der 2>/dev/null | \

openssl dgst -sha256 -hex | sed 's/^.* //'

```
root@K8S-M1:~# openssl x509 -pubkey -in \
>     /etc/kubernetes/pki/ca.crt | \
>     openssl rsa -pubin -outform der 2>/dev/null | \
>     openssl dgst -sha256 -hex | sed 's/^.* //'
3fe5d6a6a5811f586aeed2aec667ec679214488d6eea8a559ab6c510a1098594
root@K8S-M1:~#
```

node

按照前面的做:

- 配置系统设置
- 设置hostname
- 安装docker-ce
- 设置hosts和nginx
- 配置软件源，安装kubeadm kubelet

每台机器配置hosts

cat >>/etc/hosts << EOF

127.0.0.1 apiserver.k8s.local

10.10.10.91 apiserver01.k8s.local

10.10.10.92 apiserver02.k8s.local

10.10.10.93 apiserver03.k8s.local

10.10.10.94 apiserver04.k8s.local

EOF

```
root@K8S-N1:~# cat >>/etc/hosts << EOF
> 127.0.0.1 apiserver.k8s.local
> 10.10.10.91 apiserver01.k8s.local
> 10.10.10.92 apiserver02.k8s.local
> 10.10.10.93 apiserver03.k8s.local
> 10.10.10.94 apiserver04.k8s.local
> EOF
root@K8S-N1:~# more /etc/hosts
127.0.0.1       localhost
127.0.1.1       localhost

10.10.10.91 k8s-m1
10.10.10.92 k8s-m2
10.10.10.93 k8s-m3
10.10.10.94 k8s-n1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1 apiserver.k8s.local
10.10.10.91 apiserver01.k8s.local
10.10.10.92 apiserver02.k8s.local
10.10.10.93 apiserver03.k8s.local
10.10.10.94 apiserver04.k8s.local
root@K8S-N1:~#
```

每台机器生成配置文件，上面的三个hosts可以不写，写下面配置文件里域名写ip即可，但是这样更改ip需要重新加载，三台主机上都执行

```
# mkdir -p /etc/kubernetes
# cat > /etc/kubernetes/nginx.conf << EOF
user nginx nginx;
worker_processes auto;
events {
    worker_connections  20240;
    use epoll;
}
error_log /var/log/nginx_error.log info;

stream {
    upstream kube-servers {
        hash $remote_addr consistent;
        server apiserver01.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
        server apiserver02.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
        server apiserver03.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
        server apiserver04.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
    }

    server {
        listen 8443 reuseport;
        proxy_connect_timeout 3s;
        # 加大timeout

        proxy_timeout 3000s;
        proxy_pass kube-servers;
    }
}
EOF
```

```
root@K8S-N1:~# mkdir -p /etc/kubernetes
root@K8S-N1:~# cat > /etc/kubernetes/nginx.conf << EOF
> user nginx nginx;
> worker_processes auto;
> events {
>     worker_connections  20240;
>     use epoll;
> }
> error_log /var/log/nginx_error.log info;
>
> stream {
>     upstream kube-servers {
>         hash $remote_addr consistent;
>         server apiserver01.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
>         server apiserver02.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
>         server apiserver03.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
>         server apiserver04.k8s.local:6443 weight=5 max_fails=1 fail_timeout=3s;
>     }
>
>     server {
>         listen 8443 reuseport;
>         proxy_connect_timeout 3s;
>         # 加大timeout
>
>         proxy_timeout 3000s;
>         proxy_pass kube-servers;
>     }
> }
> EOF
```

因为localproxy是每台机器上的，可以不用SLB和vpc无法使用vip的限制，这里我使用staticPod创建

docker run --restart=always \

-v /etc/kubernetes/nginx.conf:/etc/nginx/nginx.conf \

-v /etc/localtime:/etc/localtime:ro \

--name k8s \

--net host \

-d \

nginx:alpine

docker ps

```
root@K8S-N1:~# docker run --restart=always \
>   -v /etc/kubernetes/nginx.conf:/etc/nginx/nginx.conf \
>   -v /etc/localtime:/etc/localtime:ro \
>   --name k8s \
>   --net host \
>   -d \
>   nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
89d9c30c1d48: Pull complete
24f1c4f0b2f4: Pull complete
Digest: sha256:0e61b143db3110f3b8ae29a67f107d5536b71a7c1f10afb14d4228711fc65a13
Status: Downloaded newer image for nginx:alpine
e8970816b7c1e8333c450c21f1f93799b865c9c0a3ea23d95e26d57df8adcb8e
root@K8S-N1:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
e8970816b7c1        nginx:alpine        "nginx -g 'daemon of…"   16 seconds ago      Up 14 seconds                           k8s
root@K8S-N1:~#
```

和master的join一样，提前准备好环境和docker，然后join的时候不需要带--control-plane

kubeadm join apiserver.k8s.local:8443 --token miuuso.ek68bi97bevh5ml5 \

--discovery-token-ca-cert-hash sha256:3fe5d6a6a5811f586aeed2aec667ec679214488d6eea8a559ab6c510a1098594

```
root@K8S-N1:~# kubeadm join apiserver.k8s.local:8443 --token miuuso.ek68bi97bevh5ml5 \
>     --discovery-token-ca-cert-hash sha256:3fe5d6a6a5811f586aeed2aec667ec679214488d6eea8a559ab6c510a1098594
[preflight] Running pre-flight checks
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 19.03.5. Latest validated version: 18.09
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.16" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Activating the kubelet service
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@K8S-N1:~#
```

master1上执行。

kubectl get node -o wide

```
root@K8S-M1:~# kubectl get node -o wide
NAME     STATUS     ROLES    AGE    VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE                      KERNEL-VERSION   CONTAINER-RUNTIME
k8s-m1   NotReady   master   70m    v1.16.3   10.10.10.91    <none>        Debian GNU/Linux 10 (buster)   4.19.0-6-amd64   docker://19.3.5
k8s-m2   NotReady   master   34m    v1.16.3   10.10.10.92    <none>        Debian GNU/Linux 10 (buster)   4.19.0-6-amd64   docker://19.3.5
k8s-m3   NotReady   master   31m    v1.16.3   10.10.10.93    <none>        Debian GNU/Linux 10 (buster)   4.19.0-6-amd64   docker://19.3.5
k8s-n1   NotReady   <none>   117s   v1.16.3   10.10.10.94    <none>        Debian GNU/Linux 10 (buster)   4.19.0-6-amd64   docker://19.3.5
root@K8S-M1:~#
```

role只是一个label，可以打label，想显示啥就node-role.kubernetes.io/xxxx

kubectl label node k8s-n1 node-role.kubernetes.io/node=""

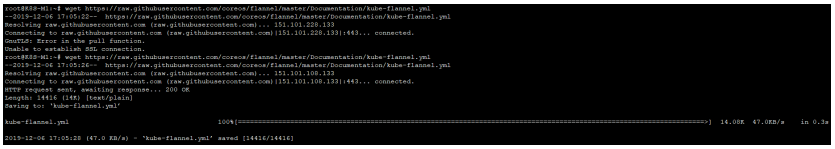
```
root@K8S-M1:~# kubectl label node k8s-n1 node-role.kubernetes.io/node=""
node/k8s-n1 labeled
```

addon

容器的网络还没处理好，coredns无法分配到ip会处于pending状态，这里我用flannel部署，如果你了解bgp可以使用calico

在master上操作，yaml文件来源与flannel官方github https://github.com/coreos/flannel/tree/master/Documentation

wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



备份一份，等下要改这个文件。

cp kube-flannel.yml{,.bak}

```
root@K8S-M1:~# cp kube-flannel.yml{,.bak}
root@K8S-M1:~# ll
total 40
-rw-r--r-- 1 root root  4118 Dec  6 14:27 initconfig.yaml
-rw-r--r-- 1 root root 14416 Dec  6 17:05 kube-flannel.yml
-rw-r--r-- 1 root root 14416 Dec  6 17:06 kube-flannel.yml.bak
root@K8S-M1:~#
```

修改

- 如果是在1.16之前使用psp，policy/v1beta1得修改成extensions/v1beta1

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

这里是1.16的不用修改。

- rbac的version改为下面，不要使用v1beta1了，使用下面命令修改

也可以手动编辑

apiVersion: rbac.authorization.k8s.io/v1beta1

替换为了下面的。

apiVersion: rbac.authorization.k8s.io/v1

sed -ri '/apiVersion: rbac/s#v1.+#v1#' kube-flannel.yml

```
root@K8S-M1:~# sed -ri '/apiVersion: rbac/s#v1.+#v1#' kube-flannel.yml
```

- 官方yaml自带了四种架构的daemonset，我们删掉除了amd64以外的，大概是227行到结尾

sed -ri '227,$d' kube-flannel.yml

```
root@K8S-M1:~# sed -ri '227,$d' kube-flannel.yml
```

- pod的cidr修改了的话这里也要修改，如果是在同一个二层，可以使用把vxlan改为host-gw模式,vxlan的话需要安全组放开8472端口的udp

原来

```
  net-conf.json: |
    {
      "Network": "10.244.0.0/16",
      "Backend": {
        "Type": "vxlan"
      }
    }
```

修改为

```
  net-conf.json: |
    {
      "Network": "10.244.0.0/16",
      "Backend": {
        "Type": "host-gw"
      }
    }
```

```
  net-conf.json: |
    {
      "Network": "10.244.0.0/16",
      "Backend": {
        "Type": "host-gw"
      }
    }
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-flannel-ds-amd64
  namespace: kube-system
  labels:
    tier: node
    app: flannel
```

- 使用下面命令修改镜像

sed -ri '/image/s#quay.io#quay.azk8s.cn#' kube-flannel.yml

```
root@K8S-M1:~# sed -ri '/image/s#quay.io#quay.azk8s.cn#' kube-flannel.yml
```

- 修改limits，需要大于request

修改后

```
limits:
  cpu: "200m"
  memory: "100Mi"
```

```
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
            cpu: "200m"
            memory: "100Mi"
        securityContext:
          privileged: false
          capabilities:
            add: ["NET_ADMIN"]
```

部署flannel

任意master上执行

1.16后node的cidr是数组，而不是单个了，flannel目前0.11和之前版本部署的话会有下列错误

Error registering network: failed to acquire lease: node "xxx" pod cidr not assigned

手动打patch，后续扩的node也记得打下

```
# nodes=`kubectl get node --no-headers | awk '{print $1}'`
# for node in $nodes;do
    cidr=`kubectl get node "$node" -o jsonpath='{.spec.podCIDRs[0]}'`
    [ -z "$(kubectl get node $node -o jsonpath='{.spec.podCIDR}')" ] && {
        kubectl patch node "$node" -p '{"spec":{"podCIDR":"'"$cidr"'"}}'
    }
done
```

```
root@K8S-M1:~# nodes=`kubectl get node --no-headers | awk '{print $1}'`
root@K8S-M1:~# for node in $nodes;do
>     cidr=`kubectl get node "$node" -o jsonpath='{.spec.podCIDRs[0]}'`
>     [ -z "$(kubectl get node $node -o jsonpath='{.spec.podCIDR}')" ] && {
>         kubectl patch node "$node" -p '{"spec":{"podCIDR":"'"$cidr"'"}}'
>     }
> done
root@K8S-M1:~#
```

kubectl apply -f kube-flannel.yml

```
root@K8S-M1:~# kubectl apply -f kube-flannel.yml
podsecuritypolicy.policy/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds-amd64 created
root@K8S-M1:~#
```

验证集群可用性

kubectl -n kube-system get pod -o wide

```
root@K8S-M1:~# kubectl -n kube-system get pod -o wide
NAME                              READY   STATUS    RESTARTS   AGE    IP              NODE     NOMINATED NODE   READINESS GATES
coredns-667f964f9b-rpj6b          1/1     Running   0          175m   10.244.2.2      k8s-m3   <none>           <none>
coredns-667f964f9b-zjz6l          1/1     Running   0          175m   10.244.1.2      k8s-m2   <none>           <none>
etcd-k8s-m1                       1/1     Running   0          175m   10.10.10.91     k8s-m1   <none>           <none>
etcd-k8s-m2                       1/1     Running   0          140m   10.10.10.92     k8s-m2   <none>           <none>
etcd-k8s-m3                       1/1     Running   0          137m   10.10.10.93     k8s-m3   <none>           <none>
kube-apiserver-k8s-m1             1/1     Running   0          175m   10.10.10.91     k8s-m1   <none>           <none>
kube-apiserver-k8s-m2             1/1     Running   0          140m   10.10.10.92     k8s-m2   <none>           <none>
kube-apiserver-k8s-m3             1/1     Running   0          137m   10.10.10.93     k8s-m3   <none>           <none>
kube-controller-manager-k8s-m1    1/1     Running   1          175m   10.10.10.91     k8s-m1   <none>           <none>
kube-controller-manager-k8s-m2    1/1     Running   0          140m   10.10.10.92     k8s-m2   <none>           <none>
kube-controller-manager-k8s-m3    1/1     Running   0          137m   10.10.10.93     k8s-m3   <none>           <none>
kube-flannel-ds-amd64-jm5qn       1/1     Running   0          47s    10.10.10.94     k8s-n1   <none>           <none>
kube-flannel-ds-amd64-pqrvt       1/1     Running   0          47s    10.10.10.93     k8s-m3   <none>           <none>
kube-flannel-ds-amd64-qcj79       1/1     Running   0          47s    10.10.10.92     k8s-m2   <none>           <none>
kube-flannel-ds-amd64-zxhwt       1/1     Running   0          47s    10.10.10.91     k8s-m1   <none>           <none>
kube-proxy-4snqp                  1/1     Running   0          140m   10.10.10.92     k8s-m2   <none>           <none>
kube-proxy-4zb8b                  1/1     Running   0          137m   10.10.10.93     k8s-m3   <none>           <none>
kube-proxy-q6x6s                  1/1     Running   0          107m   10.10.10.94     k8s-n1   <none>           <none>
kube-proxy-t2hcp                  1/1     Running   0          175m   10.10.10.91     k8s-m1   <none>           <none>
kube-scheduler-k8s-m1             1/1     Running   1          175m   10.10.10.91     k8s-m1   <none>           <none>
kube-scheduler-k8s-m2             1/1     Running   0          140m   10.10.10.92     k8s-m2   <none>           <none>
kube-scheduler-k8s-m3             1/1     Running   0          137m   10.10.10.93     k8s-m3   <none>           <none>
root@K8S-M1:~#
```

等待kube-system空间下的pod都是running后我们来测试下集群可用性

```
# cat<<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
```

```
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:alpine
        name: nginx
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
  - name: busybox
    image: busybox:1.28
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
EOF
```

```
root@K8S-M1:~# cat<<EOF | kubectl apply -f -
> apiVersion: apps/v1
> kind: Deployment
> metadata:
>   name: nginx
> spec:
>   selector:
>     matchLabels:
>       app: nginx
>   template:
>     metadata:
>       labels:
>         app: nginx
>     spec:
>       containers:
>       - image: nginx:alpine
>         name: nginx
>         ports:
>         - containerPort: 80
> ---
> apiVersion: v1
> kind: Service
> metadata:
>   name: nginx
> spec:
>   selector:
>     app: nginx
>   ports:
>     - protocol: TCP
>       port: 80
>       targetPort: 80
> ---
> apiVersion: v1
> kind: Pod
> metadata:
>   name: busybox
>   namespace: default
> spec:
>   containers:
>   - name: busybox
>     image: busybox:1.28
>     command:
>       - sleep
>       - "3600"
>     imagePullPolicy: IfNotPresent
>   restartPolicy: Always
> EOF
deployment.apps/nginx created
service/nginx created
pod/busybox created
root@K8S-M1:~#
```

等待pod running

```
root@K8S-M1:~# kubectl get po,svc -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
pod/busybox                 1/1     Running   0          88s   10.244.3.3   k8s-n1   <none>           <none>
pod/nginx-5c559d5697-vhm72  1/1     Running   0          88s   10.244.3.2   k8s-n1   <none>           <none>

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE   SELECTOR
service/kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP   3h    <none>
service/nginx        ClusterIP   10.98.88.73   <none>        80/TCP    88s   app=nginx
root@K8S-M1:~#
```

在master上curl nginx的svc的ip出现nginx的index内容即集群正常，例如我的nginx svc ip是10.98.88.73

curl -s 10.98.88.73

```
root@K8S-M1:~# curl -s 10.98.88.73
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@K8S-M1:~#
```

关于使用kubeadm的注意事项和个人建议

- 小白不要着急啥都往上部署，例如dashboard和什么helm，没这个必要，先把命令行的kubectl和一些基础学会了

- 默认证书是只有一年的，可以自己去修改源码更改
- 先去把官方文档的concept和tasks板块看完，市面上的书籍和教程实际上都是讲的这俩板块儿
- 不懂网络的话去找点CCIE的教程看下
- systemd和docker以及Linux基础都是挺重要的，–help找选项很多人居然都不会
- yaml，yaml的结构就是无非那些字符，数字，object，数组的混合，可以尝试大脑中把一段yaml转换成json，不然看不懂yaml的结构学不会k8s。很多层级实际上是遵循着逻辑的，例如一个pod有多个容器，所以pod.spec.containers就是一个obkect的数组，又因为pod共享network namepsace，所以hostNetwork这个属性肯定是containers同个级别的
- kubeconfig实际上就是存了三个信息，一个是host(集群)，用户认证信息，这俩都是可以写多个的，所以都是yaml的数组-开头，以及当前的context是哪个host搭配哪个认证信息。和web的jwt思想一样
- 互斥和污点都是基础知识，也在concept和tasks板块里，上生产的话多份pod肯定要互斥自己分散开来，就像没有容器技术的时代是每个节点跑一份服务，这样down了一个node业务不会挂

关于kubeadm过程和更多详细参数选项见下面文章

- https://www.jianshu.com/p/1e65610dd223