https://www.bilibili.com/video/BV1UF411s7Gs?spm_id_from=333.337.search-card.all.click&vd_source=6d1166f6ebf24940077bce753ef61d75

目前我的k8s集群还只是只有1台master的，并不是高可用的，因为资源不够，之前的视频里也没有专门搭建HA的k8s的内容，前面看了下sealos的高可用搭建方法，这里再看下另外一种方法，这里没有用到自动化部署方式，3台master和2台node都是手动搭建的

视频里用ubuntu的系统，我的是centos，有些命令和目录不太一样

视频里只用了1台nginx做负载均衡（ps：也可2台，对nginx做高可用）

这是搭建文档，包括ubunut和centos的：

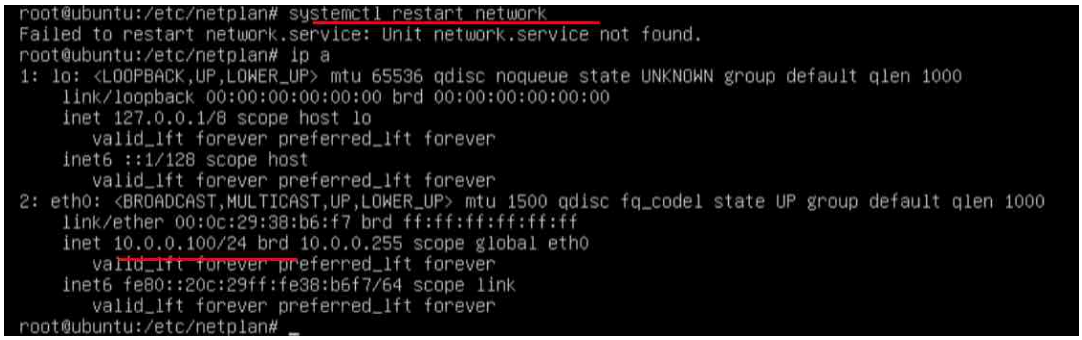
https://zahui.fan/posts/b86d9e9f/

为了避免链接失效，视频里的步骤和文档步骤都记录一下：

1、安装vm1（ingress和apiserver的负载均衡，nfs存储）

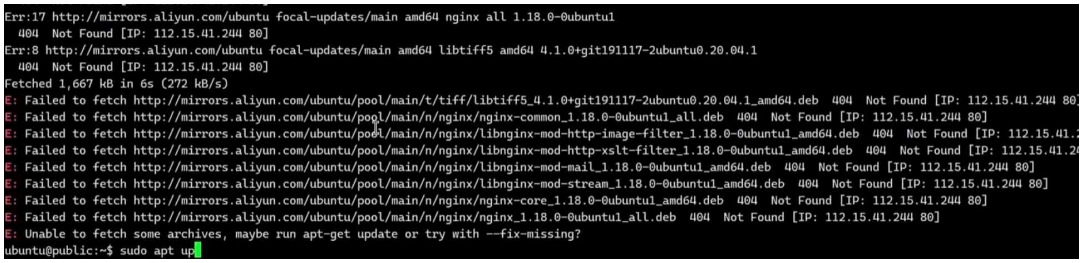
先安装一台vm，安装nginx，作为ingress和apiserver的负载均衡，nfs存储

先修改完ip和hostname：这里nginx的ip设置成了**10.0.0.100**



安装nignx

apt-get install -y nginx（centos 是yum）



因为源长期没更新，更新源：apt update

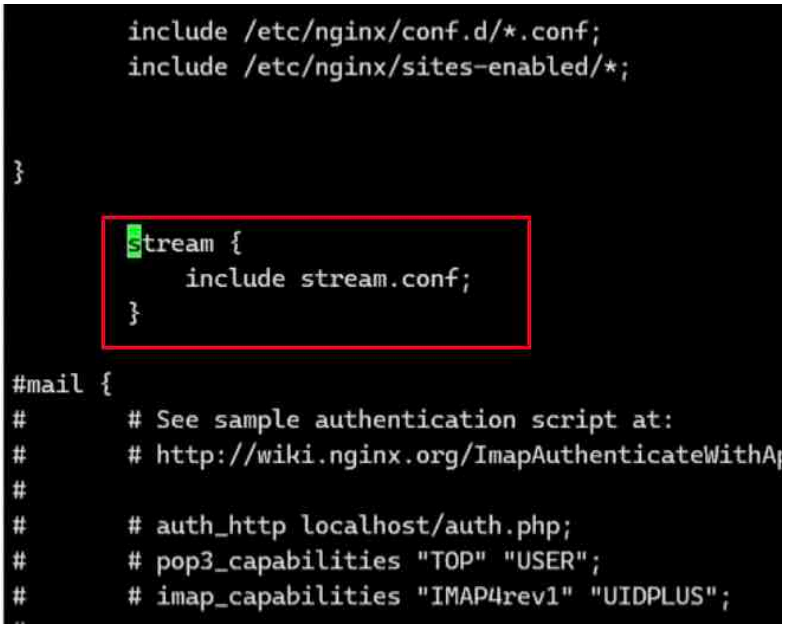
安装后配置：

```
root@public:~# cd /etc/nginx/
root@public:/etc/nginx# ll
total 72
drwxr-xr-x  8 root root 4096 Mar 14 02:38 ./
drwxr-xr-x 95 root root 4096 Mar 14 02:38 ../
drwxr-xr-x  2 root root 4096 May 25  2021 conf.d/
-rw-r--r--  1 root root 1077 Feb  4  2019 fastcgi.conf
-rw-r--r--  1 root root 1007 Feb  4  2019 fastcgi_params
-rw-r--r--  1 root root 2837 Feb  4  2019 koi-utf
-rw-r--r--  1 root root 2223 Feb  4  2019 koi-win
-rw-r--r--  1 root root 3957 Feb  4  2019 mime.types
drwxr-xr-x  2 root root 4096 May 25  2021 modules-available/
drwxr-xr-x  2 root root 4096 Mar 14 02:38 modules-enabled/
-rw-r--r--  1 root root 1490 Feb  4  2019 nginx.conf
-rw-r--r--  1 root root  180 Feb  4  2019 proxy_params
-rw-r--r--  1 root root  636 Feb  4  2019 scgi_params
drwxr-xr-x  2 root root 4096 Mar 14 02:38 sites-available/
drwxr-xr-x  2 root root 4096 Mar 14 02:38 sites-enabled/
drwxr-xr-x  2 root root 4096 Mar 14 02:38 snippets/
-rw-r--r--  1 root root  664 Feb  4  2019 uwsgi_params
-rw-r--r--  1 root root 3071 Feb  4  2019 win-utf
root@public:/etc/nginx# vim nginx.conf
```

加上stream配置：

```
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json app

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;

        stream {
            include stream.conf;
        }
}


#mail {
#       # See sample authentication script at:
#       # http://wiki.nginx.org/ImapAuthenticateWithApachePhp
```

位置配错了，应该在}外面：



然后新建stream.conf

注：stream模块一般用于tcp/UDP数据流的代理和负载均衡，可以通过stream模块代理转发TCP消息。 ngx_stream_core_module模块由1.9.0版提供。 默认情况下，没有构建此模块。 -必须使用-with stream配置参数启用。 也就是说，必须在使用./configure --with-stream编译时添加流模块。 流模块的使用方法与http模块相同，语法也基本相同。

1.1、nginx,lvs,haproxy+keepalived区别

https://www.cnblogs.com/onesea/p/15069992.html

Nginx、LVS、HAProxy 是目前使用最广泛的三种负载均衡软件，通常会结合Keepalive做健康检查，实现故障转移的高可用功能。

软件负载均衡一般通过两种方式来实现：基于操作系统的软负载实现和基于第三方应用的软负载实现。

LVS就是基于Linux操作系统实现的一种软负载；

HAProxy就是开源的并且基于第三应用实现的软负载。

1）在四层（tcp）实现负载均衡的软件：

lvs------>重量级
nginx------>轻量级，带缓存功能，正则表达式较灵活
haproxy------>模拟四层转发，较灵活

2）在七层（http）实现反向代理的软件：

haproxy------>天生技能，全面支持七层代理，会话保持，标记，路径转移；
nginx------>只在http协议和mail协议上功能比较好，性能与haproxy差不多；
apache------>功能较差
总的来说，一般是lvs做4层负载；nginx做7层负载；haproxy比较灵活，4层和7层负载均衡都能做一般对负载均衡的使用是随着网站规模的提升根据不同的阶段来使用不同的技术。具体的应用需求还得具体分析：
1）如果是中小型的 Web 应用，比如日PV小于1000 万，用 Nginx 就完全可以了；
2）如果机器不少，可以用DNS轮询， LVS所耗费的机器还是比较多的；大型网站或重要的服务，且服务器比较多时，可以考虑用LVS。
还有一种是通过硬件来进行进行，常见的硬件有比较昂贵的F5和Array等商用的负载均衡器，它的优点就是有专业的维护团队来对这些服务进行维护、缺点就是花销太大，所以对于规模较小
的网络服务来说暂时还没有需要使用；另外一种就是类似于 Nginx/LVS/HAProxy 的基于 Linux 的开源免费的负载均衡软件，这些都是通过软件级别来实现，所以费用非常低廉。目前关于网
站架构一般比较合理流行的架构方案： Web 前端采用Nginx/HAProxy+Keepalived 作负载均衡器；后端采用 MySQL 数据库一主多从和读写分离，采用 LVS+Keepalived 的架构。 当然要根据
项目具体需求制定方案。下面说说各自的特点和适用场合。

下面简单说下Nginx、LVS、HAProxy 负载均衡软件的优缺点：

一、Nginx的优点是：

1）工作在网络的7层之上，可以针对 http 应用做一些分流的策略，比如针对域名、目录结构，它的正则规则比 HAProxy 更为强大和灵活，这也是它目前广泛流
行的主要原因之一， Nginx 单凭这点可利用的场合就远多于 LVS 了。
2） Nginx 对网络稳定性的依赖非常小，理论上能 ping 通就就能进行负载功能，这个也是它的优势之一；相反 LVS 对网络稳定性依赖比较大，这点本人深有体会；
3） Nginx 安装和配置比较简单，测试起来比较方便，它基本能把错误用日志打印出来。 LVS 的配置、测试就要花比较长的时间了， LVS 对网络依赖比较大。
4）可以承担高负载压力且稳定，在硬件不差的情况下一般能支撑几万次的并发量，负载度比 LVS 相对小些。
5） Nginx 可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点，不过其中缺点就是不支持url来检测。
比如用户正在上传一个文件，而处理该上传的节点刚好在上传过程中出现故障， Nginx 会把上传切到另一台服务器重新处理，而LVS就直接断掉了，如果是上传一个很大的文件或者很重要的文
件的话，用户可能会因此而不满。
6）Nginx 不仅仅是一款优秀的负载均衡器/反向代理软件，它同时也是功能强大的 Web 应用服务器。 LNMP 也是近几年非常流行的 web 架构，在高流量的环境中稳定性也很好。
7）Nginx 现在作为 Web 反向加速缓存越来越成熟了，速度比传统的 Squid 服务器更快，可以考虑用其作为反向代理加速器。
8）Nginx 可作为中层反向代理使用，这一层面 Nginx 基本上无对手，唯一可以对比 Nginx 的就只有 lighttpd 了，不过 lighttpd 目前还没有做到 Nginx 完全的功能，配置也不那么清晰易读，社区资料也远远没 Nginx 活跃。
9） Nginx 也可作为静态网页和图片服务器，这方面的性能也无对手。还有 Nginx社区非常活跃，第三方模块也很多。
Nginx 的缺点是：
1）Nginx 仅能支持 http、 https 和 Email 协议，这样就在适用范围上面小些，这个是它的缺点。
2）对后端服务器的健康检查，只支持通过端口来检测，不支持通过 url 来检测。不支持 Session 的直接保持，但能通过 ip_hash 来解决。

二、LVS：使用 Linux 内核集群实现一个高性能、 高可用的负载均衡服务器，它具有很好的可伸缩性（Scalability）、可靠性（Reliability)和可管理性（Manageability)。

LVS 的优点是：

1）抗负载能力强、是工作在网络 4 层之上仅作分发之用， 没有流量的产生，这个特点也决定了它在负载均衡软件里的性能最强的，对内存和 cpu 资源消耗比较低。
2）配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西，所以并不需要太多接触，大大减少了人为出错的几率。
3）工作稳定，因为其本身抗负载能力很强，自身有完整的双机热备方案，如LVS+Keepalived，不过我们在项目实施中用得最多的还是 LVS/DR+Keepalived。
4）无流量， LVS 只分发请求，而流量并不从它本身出去，这点保证了均衡器 IO的性能不会收到大流量的影响。
5）应用范围比较广，因为 LVS 工作在 4 层，所以它几乎可以对所有应用做负载均衡，包括 http、数据库、在线聊天室等等。
LVS 的缺点是：
1）软件本身不支持正则表达式处理，不能做动静分离；而现在许多网站在这方面都有较强的需求，这个是 Nginx/HAProxy+Keepalived 的优势所在。
2）如果是网站应用比较庞大的话， LVS/DR+Keepalived 实施起来就比较复杂了，特别后面有 Windows Server 的机器的话，如果实施及配置还有维护过程就比较复杂了，相对而言，
Nginx/HAProxy+Keepalived 就简单多了。

三、HAProxy 的特点是：

1）HAProxy 也是支持虚拟主机的。
2）HAProxy 的优点能够补充 Nginx 的一些缺点，比如支持 Session 的保持，Cookie的引导；同时支持通过获取指定的 url 来检测后端服务器的状态。
3）HAProxy 跟 LVS 类似，本身就只是一款负载均衡软件；单纯从效率上来讲HAProxy 会比 Nginx 有更出色的负载均衡速度，在并发处理上也是优于 Nginx 的。
4）HAProxy 支持 TCP 协议的负载均衡转发，可以对 MySQL 读进行负载均衡，对后端的 MySQL 节点进行检测和负载均衡，大家可以用 LVS+Keepalived 对 MySQL主从做负载均衡。
5）HAProxy 负载均衡策略非常多， HAProxy 的负载均衡算法现在具体有如下8种：
1> roundrobin，表示简单的轮询，这个不多说，这个是负载均衡基本都具备的；
2> static-rr，表示根据权重，建议关注；
3> leastconn，表示最少连接者先处理，建议关注；
4> source，表示根据请求源 IP，这个跟 Nginx 的 IP_hash 机制类似，我们用其作为解决 session 问题的一种方法，建议关注；
5> ri，表示根据请求的 URI；
6> rl_param，表示根据请求的 URI 参数'balance url_param' requires an URLparameter name；
7> hdr(name)，表示根据 HTTP 请求头来锁定每一次 HTTP 请求；
8> rdp-cookie(name)，表示根据据 cookie(name)来锁定并哈希每一次 TCP 请求。

Haproxy+Keepalived

各自作用：
1）Keepalived 的作用是检测web服务器的状态，如果有一台web服务器死机，或工作出现故障，Keepalived将检测到，并将有故障的web服务器从系统中剔除， 当web服务器工作正常
后Keepalived自动将web服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的 web服务器。
2）HAProxy 提供高可用性、负载均衡以及基于TCP和HTTP应用的代理,支持虚拟主机,它是免费、快速并且可靠的一种解决方案。HAProxy 特别适用于那些负载特大的 web 站点, 这些站点通常又需要会话保持或七层处理。HAProxy 运行在当前的硬件上,完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整 合进您当前的架构中, 同时可以保护你的 web 服务器不被暴露

到网络上。

继续：新建stream.conf

```
root@public:/etc/nginx# vim stream.conf
```

先主要关注上面框里的配置：

```
upstream k8s-api {
    server 10.0.0.101:6443;
    server 10.0.0.102:6443;
    server 10.0.0.103:6443;
}
server {
    listen 6443;
    proxy_connect_timeout 1s;
    proxy_pass k8s-api;
}
upstream ingress-http {
    server 10.0.0.20:30080;   # 这里需要更改成ingress的NodePort
    server 10.0.0.21:30080;   # 这里需要更改成ingress的NodePort
}
server {
    listen 80;
    proxy_connect_timeout 1s;
    proxy_pass ingress-http;
}
upstream ingress-https {
    server 10.0.0.20:30443;   # 这里需要更改成ingress的NodePort
    server 10.0.0.21:30443;   # 这里需要更改成ingress的NodePort
}
server {
    listen 443;
    proxy_connect_timeout 1s;
    proxy_pass ingress-https;
}
```

检查语法没问题，重启nginx：

```
root@public:/etc/nginx# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@public:/etc/nginx# sudo systemctl restart  nginx
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xe" for details.
root@public:/etc/nginx# journalctl -xe
```

```
Mar 14 02:40:40 public systemd-udevd[513]: Network interface NamePolicy= disabled on kernel command line, ignoring.
Mar 14 02:40:40 public nginx[3176]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Mar 14 02:40:40 public nginx[3176]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Mar 14 02:40:41 public nginx[3176]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Mar 14 02:40:41 public nginx[3176]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Mar 14 02:40:41 public multipathd[680]: sda: add missing path
Mar 14 02:40:41 public multipathd[680]: sda: failed to get udev uid: Invalid argument
Mar 14 02:40:41 public multipathd[680]: sda: failed to get sysfs uid: Invalid argument
Mar 14 02:40:41 public multipathd[680]: sda: failed to get sgio uid: No such file or directory
Mar 14 02:40:42 public nginx[3176]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Mar 14 02:40:42 public nginx[3176]: nginx: [emerg] still could not bind()
Mar 14 02:40:42 public systemd[1]: nginx.service: Control process exited, code=exited, status=1/FAILURE
-- Subject: Unit process exited
-- Defined-By: systemd
-- Support: http://www.ubuntu.com/support
--
-- An ExecStart= process belonging to unit nginx.service has exited.
--
-- The process' exit code is 'exited' and its exit status is 1.
Mar 14 02:40:42 public systemd[1]: nginx.service: Failed with result 'exit-code'.
-- Subject: Unit failed
-- Defined-By: systemd
-- Support: http://www.ubuntu.com/support
--
-- The unit nginx.service has entered the 'failed' state with result 'exit-code'.
Mar 14 02:40:42 public systemd[1]: Failed to start A high performance web server and a reverse proxy server.
-- Subject: A start job for unit nginx.service has failed
-- Defined-By: systemd
-- Support: http://www.ubuntu.com/support
--
-- A start job for unit nginx.service has finished with a failure.
--
-- The job identifier is 875 and the job result is failed.
Mar 14 02:40:42 public sudo[3171]: pam_unix(sudo:session): session closed for user root
Mar 14 02:40:46 public multipathd[680]: sda: add missing path
Mar 14 02:40:46 public multipathd[680]: sda: failed to get udev uid: Invalid argument
Mar 14 02:40:46 public multipathd[680]: sda: failed to get sysfs uid: Invalid argument
Mar 14 02:40:46 public multipathd[680]: sda: failed to get sgio uid: No such file or directory
```

问题：（已解决）重启nginx失败，80端口被占用（删掉默认搭建网站功能）

报错了，端口冲突了，这里nginx只是用来做负载均衡，不用它来搭建网站，所以把default删掉：

```
root@public:/etc/nginx# cd sites-enabled/
root@public:/etc/nginx/sites-enabled# ll
total 8
drwxr-xr-x 2 root root 4096 Mar 14 02:38 ./
drwxr-xr-x 8 root root 4096 Mar 14 02:40 ../
lrwxrwxrwx 1 root root   34 Mar 14 02:38 default -> /etc/nginx/sites-available/default
root@public:/etc/nginx/sites-enabled# rm -rf default
root@public:/etc/nginx/sites-enabled# ll
total 8
drwxr-xr-x 2 root root 4096 Mar 14 02:41 ./
drwxr-xr-x 8 root root 4096 Mar 14 02:40 ../
root@public:/etc/nginx/sites-enabled# sudo systemctl restart  nginx \
> ^C
root@public:/etc/nginx/sites-enabled# sudo systemctl restart  nginx
root@public:/etc/nginx/sites-enabled#
```

查看nginx进程，已经起来了：

```
root@public:/etc/nginx/sites-enabled# ps -ef |grep nginx
root        3239       1  0 02:41 ?        00:00:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
www-data    3240    3239  0 02:41 ?        00:00:00 nginx: worker process
www-data    3241    3239  0 02:41 ?        00:00:00 nginx: worker process
root        3247    3078  0 02:41 pts/0    00:00:00 grep --color=auto nginx
root@public:/etc/nginx/sites-enabled#
```

下面开始准备master节点

2、准备master节点

2.1、基础环境配置

同样，先配置ip和主机名

```
root@test-k8s-master1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    link/ether 00:0c:29:f7:a5:33 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.101/24 brd 10.0.0.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fef7:a533/64 scope link
       valid_lft forever preferred_lft forever
```

然后添加host文件：

```
127.0.0.1 localhost
127.0.1.1 ubuntu

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

10.0.0.101 test-k8s-master1
10.0.0.102 test-k8s-master2
10.0.0.103 test-k8s-master3
10.0.0.104 test-k8s-worker1
10.0.0.105 test-k8s-worker2
```

然后就用这台机器作为模板来安装k8s组件，其他几台克隆即可

基础环境配置

基础环境是不管master还是work都需要的环境

1. 禁用swap
2. 确保每个节点上 MAC 地址和 product_uuid 的唯一性 `sudo cat /sys/class/dmi/id/product_uuid`
3. 修改hostname
4. 允许 iptables 检查桥接流量

Bash

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

ps：但是这里好像没配置ipv4，swapoff -a

2.2、runtime配置（视频里用的containerd，docker也可以）

安装runtime

- 使用docker
  1. 安装

     Bash

     ```
     curl -fsSL get.docker.com | bash
     ```

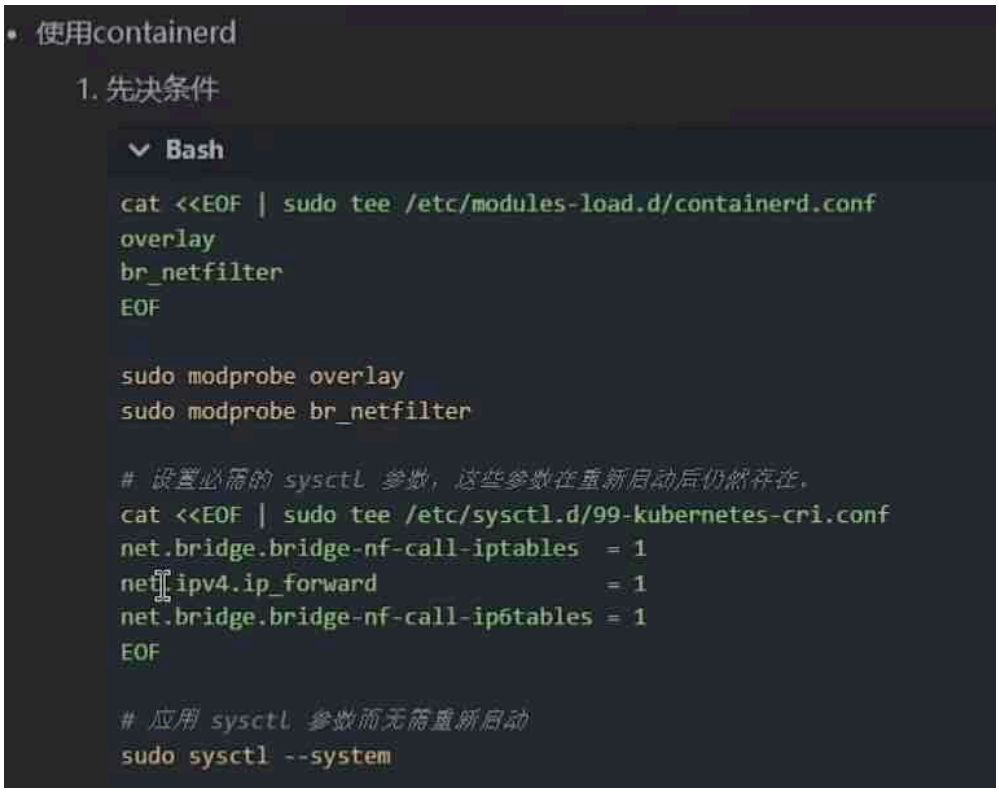
  2. 配置

     修改cgroupdriver为systemd

     Bash

     ```
     cat > /etc/docker/daemon.json << EOF
     {
         "registry-mirrors": ["https://go38p9zi.mirror.aliyuncs.com"],
         "exec-opts": ["native.cgroupdriver=systemd"],
         "log-driver": "json-file",
         "log-opts": {
             "max-size": "100m"
         }
     }
     EOF
     ```

我的是docker，视频里用的是containerd：



- 使用containerd
  1. 先决条件

Bash

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# 设置必需的 sysctl 参数，这些参数在重新启动后仍然存在。
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

# 应用 sysctl 参数而无需重新启动
sudo sysctl --system
```

2. 安装

Bash

```
# 安装依赖
sudo apt install -y apt-transport-https ca-certificates curl gnupg2 software-properties-common

# 信任密钥
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# 添加仓库
sudo add-apt-repository \
    "deb [arch=amd64] http://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu \
    $(lsb_release -cs) \
    stable"

# 安装containerd
sudo apt update
sudo apt install containerd.io
```

```
root@test-k8s-master1:~# sudo apt install -y apt-transport-https ca-certificates curl gnupg2 software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20210119~20.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.68.0-1ubuntu2.5).
curl set to manually installed.
The following additional packages will be installed:
  dirmngr gnupg gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv python3-software-properti
Suggested packages:
  pinentry-gnome3 tor parcimonie xloadimage scdaemon
The following NEW packages will be installed:
  apt-transport-https gnupg2
The following packages will be upgraded:
  dirmngr gnupg gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv python3-software-properti
13 upgraded, 2 newly installed, 0 to remove and 51 not upgraded.
Need to get 2,605 kB of archives.
After this operation, 212 kB of additional disk space will be used.
Get:1 http://mirrors.aliyun.com/ubuntu focal-updates/main amd64 gpg-wks-client amd64 2.2.19-3ubuntu2.1 [97.6 kB]
Get:2 http://mirrors.aliyun.com/ubuntu focal-updates/main amd64 dirmngr amd64 2.2.19-3ubuntu2.1 [329 kB]
Get:3 http://mirrors.aliyun.com/ubuntu focal-updates/main amd64 gnupg-utils amd64 2.2.19-3ubuntu2.1 [480 kB]
```

同理，需更新安装源：

```
Get:15 http://mirrors.aliyun.com/ubuntu focal-updates/universe amd64 gnupg2 all 2.2.19-3ubuntu
Fetched 2,568 kB in 8s (329 kB/s)
E: Failed to fetch http://mirrors.aliyun.com/ubuntu/pool/universe/a/apt/apt-transport-https_2.
E: Failed to fetch http://mirrors.aliyun.com/ubuntu/pool/main/s/software-properties/software-p
E: Failed to fetch http://mirrors.aliyun.com/ubuntu/pool/main/s/software-properties/python3-so
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
root@test-k8s-master1:~# sudo apt update
Hit:1 http://mirrors.aliyun.com/ubuntu focal InRelease
Get:2 http://mirrors.aliyun.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://mirrors.aliyun.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://mirrors.aliyun.com/ubuntu focal-security InRelease [114 kB]
Get:5 http://mirrors.aliyun.com/ubuntu focal-updates/main amd64 Packages [1,641 kB]
25% [5 Packages 760 kB/1,641 kB 46%]
```

```
Running hooks in /etc/ca-certificates/update.d...
done.
root@test-k8s-master1:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
root@test-k8s-master1:~# sudo add-apt-repository \
>     "deb [arch=amd64] http://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu \
>     $(lsb_release -cs) \
>     stable"


0% [Working]
Hit:1 http://mirrors.aliyun.com/ubuntu focal InRelease
Hit:2 http://mirrors.aliyun.com/ubuntu focal-updates InRelease
Hit:3 http://mirrors.aliyun.com/ubuntu focal-backports InRelease
Get:4 http://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu focal InRelease [57.7 kB]
Hit:5 http://mirrors.aliyun.com/ubuntu focal-security InRelease
0% [Working]
Get:6 http://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu focal/stable amd64 Packages [14.8 kB]
```

```
root@test-k8s-master1:~#
root@test-k8s-master1:~#
root@test-k8s-master1:~# sudo apt update
Hit:1 http://mirrors.aliyun.com/ubuntu focal InRelease
Hit:2 http://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu focal InR
Hit:3 http://mirrors.aliyun.com/ubuntu focal-updates InRelease
Hit:4 http://mirrors.aliyun.com/ubuntu focal-backports InRelease
Hit:5 http://mirrors.aliyun.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
169 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@test-k8s-master1:~# sudo apt install containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  containerd.io
0 upgraded, 1 newly installed, 0 to remove and 169 not upgraded.
Need to get 24.9 MB of archives.
After this operation, 112 MB of additional disk space will be used.
Get:1 http://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu focal/sta
```

3. 配置

生成默认配置

Bash

```
sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
```

结合 runc 使用 systemd cgroup 驱动，在 /etc/containerd/config.toml 中设置

TOML

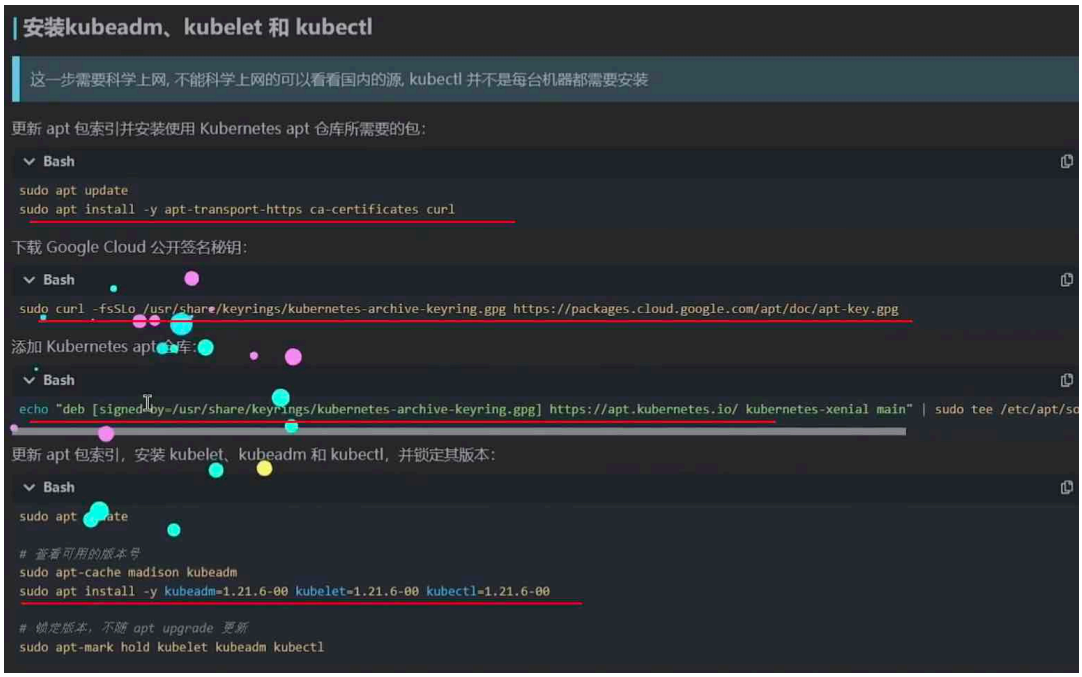
```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
...
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
    SystemdCgroup = true

sudo systemctl restart containerd
```

编辑/etc/containerd/config.toml

```
      IoUid = 0
      NoNewKeyring = false
      NoPivotRoot = false
      Root = ""
      ShimCgroup = ""
      SystemdCgroup = true

[plugins."io.containerd.grpc.v1.cri".containerd.untruste
  base_runtime_spec = ""
  container_annotations = []
  pod_annotations = []
  privileged_without_host_devices = false
```

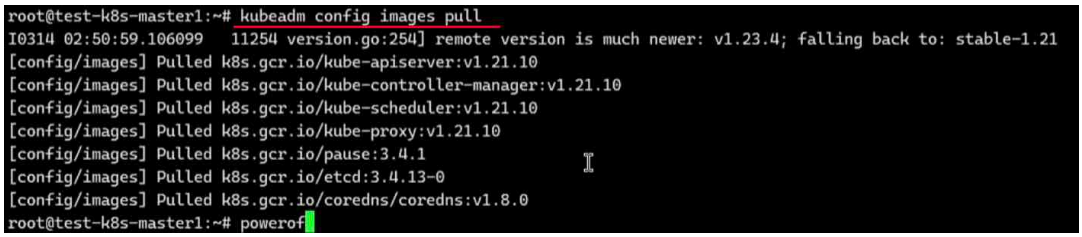
重启containerd。

2.3、安装k8s组件（kubeadm，kubelet，kubectl）



负载均衡已经做了，就直接开始创建集群了

2.4、创建k8s集群（拉取镜像）



然后到此，把当前系统关机做快照（ps：其实就是克隆成其他几个集群，基础环境设置都是一样的，就无需重复操作了）

3、克隆vm（2台master，2个node）

然后设置ip和主机名：

注意ip是负载均衡nginx的ip：

3.1、master1端kubeadm init

```
root@test-k8s-master1:~# sudo kubeadm init \
> --control-plane-endpoint "10.0.0.100:6443" \
> --upload-certs \
> --service-cidr=10.96.0.0/12 \
> --pod-network-cidr=10.244.0.0/12
I0315 15:15:18.468886    1533 version.go:254] remote version is much newer: v1.23.4; falling back to: stable-1.21
[init] Using Kubernetes version: v1.21.10
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
```

```
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of the control-plane node running the following command on each as root:

  kubeadm join 10.0.0.100:6443 --token aflwg3.3t5g9qwokqquk6c4 \
        --discovery-token-ca-cert-hash sha256:822139903238adccb329d730b64e8503cf6f4e9f285811493498ad444f731f64 \
        --control-plane --certificate-key 2223e02d770c79489253b10304e16bddb42dec1d1bfb85c8011897413c6514f7

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

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

kubeadm join 10.0.0.100:6443 --token aflwg3.3t5g9qwokqquk6c4 \
        --discovery-token-ca-cert-hash sha256:822139903238adccb329d730b64e8503cf6f4e9f285811493498ad444f731f64
root@test-k8s-master1:~#
```

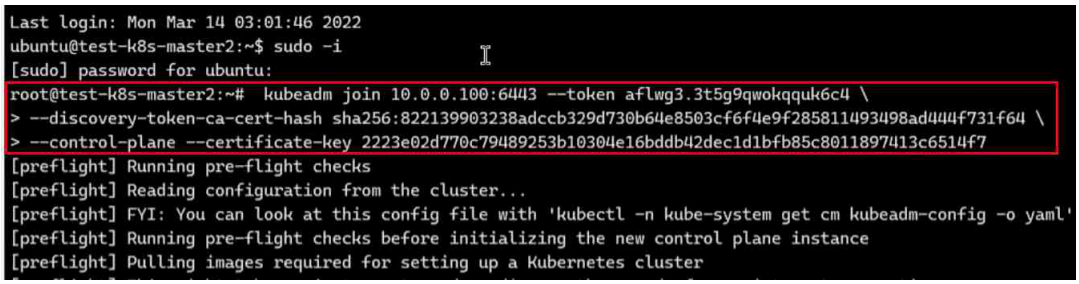
注意，第一个框里是其他机器加入集群成为master，第二个是加入后成为node的

先执行kubectl命令的话，先执行这三条：

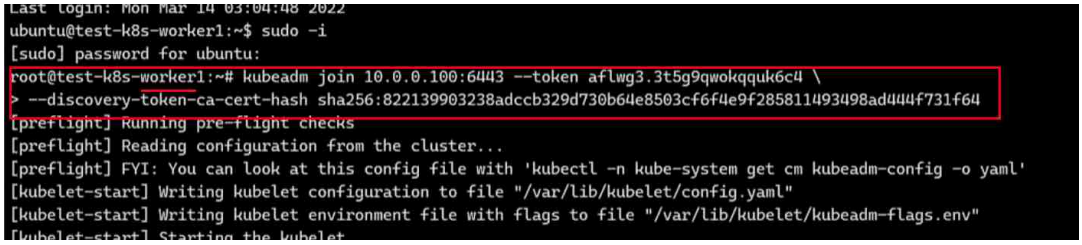
```
  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

You can now join any number of the control-plane node running the following command on each as root:

  kubeadm join 10.0.0.100:6443 --token aflwg3.3t5g9qwokqquk6c4 \
        --discovery-token-ca-cert-hash sha256:822139903238adccb329d730b64e8503cf6f4e9f285811493498ad444f731f64 \
        --control-plane --certificate-key 2223e02d770c79489253b10304e16bddb42dec1d1bfb85c8011897413c6514f7

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

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

kubeadm join 10.0.0.100:6443 --token aflwg3.3t5g9qwokqquk6c4 \
        --discovery-token-ca-cert-hash sha256:822139903238adccb329d730b64e8503cf6f4e9f285811493498ad444f731f64
root@test-k8s-master1:~#  mkdir -p $HOME/.kube
root@test-k8s-master1:~#   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@test-k8s-master1:~#   sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@test-k8s-master1:~#
root@test-k8s-master1:~# kubectl get nodes
NAME              STATUS     ROLES                  AGE   VERSION
test-k8s-master1  NotReady   control-plane,master   27s   v1.21.6
root@test-k8s-master1:~#
```

3.2、其他机器加入集群（2台master，2台node）

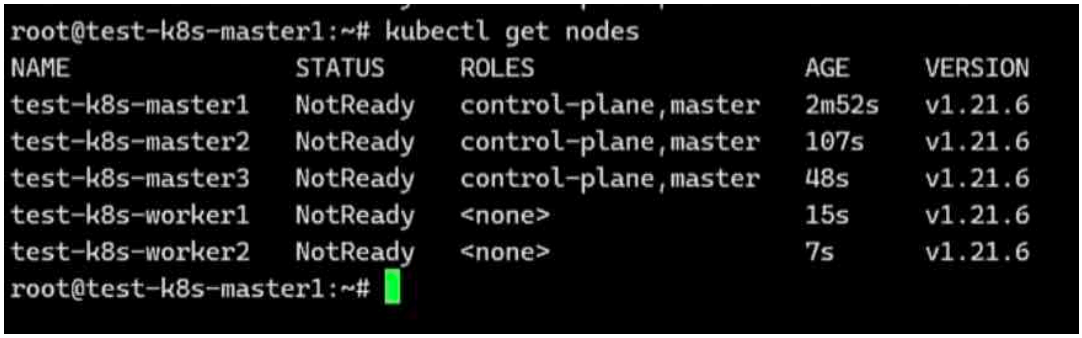
master2和master3执行添加master节点：



添加2个work节点：



现在可以看到node信息，都是notready：



```
root@test-k8s-master1:~# kubectl get pod -A
NAMESPACE     NAME                                        READY   STATUS    RESTARTS   AGE
kube-system   coredns-558bd4d5db-gpx9w                    0/1     Pending   0          2m55s
kube-system   coredns-558bd4d5db-j9pwn                    0/1     Pending   0          2m55s
kube-system   etcd-test-k8s-master1                       1/1     Running   0          3m
kube-system   etcd-test-k8s-master2                       1/1     Running   0          2m
kube-system   etcd-test-k8s-master3                       1/1     Running   0          38s
kube-system   kube-apiserver-test-k8s-master1             1/1     Running   0          3m
kube-system   kube-apiserver-test-k8s-master2             1/1     Running   0          2m1s
kube-system   kube-controller-manager-test-k8s-master1    1/1     Running   1          3m
kube-system   kube-controller-manager-test-k8s-master2    1/1     Running   0          2m1s
kube-system   kube-controller-manager-test-k8s-master3    0/1     Pending   0          0s
kube-system   kube-proxy-75x67                            1/1     Running   0          2m2s
kube-system   kube-proxy-7dsxx                            1/1     Running   0          22s
kube-system   kube-proxy-nk5hj                            1/1     Running   0          2m55s
kube-system   kube-proxy-pzcfl                            1/1     Running   0          63s
kube-system   kube-proxy-s4dtt                            1/1     Running   0          30s
kube-system   kube-scheduler-test-k8s-master1             1/1     Running   1          3m
kube-system   kube-scheduler-test-k8s-master2             1/1     Running   0          2m1s
```

3.3、master1部署flannel

```
root@test-k8s-master1:~# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
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 created
```

创建成功，搭建完成：

```
root@test-k8s-master1:~# kubectl get pod -A
NAMESPACE     NAME                                        READY   STATUS    RESTARTS   AGE
kube-system   coredns-558bd4d5db-gpx9w                    1/1     Running   0          4m45s
kube-system   coredns-558bd4d5db-j9pwn                    1/1     Running   0          4m45s
kube-system   etcd-test-k8s-master1                       1/1     Running   0          4m50s
kube-system   etcd-test-k8s-master2                       1/1     Running   0          3m50s
kube-system   etcd-test-k8s-master3                       1/1     Running   0          2m28s
kube-system   kube-apiserver-test-k8s-master1             1/1     Running   0          4m50s
kube-system   kube-apiserver-test-k8s-master2             1/1     Running   0          3m51s
kube-system   kube-apiserver-test-k8s-master3             1/1     Running   1          79s
kube-system   kube-controller-manager-test-k8s-master1    1/1     Running   1          4m50s
kube-system   kube-controller-manager-test-k8s-master2    1/1     Running   0          3m51s
kube-system   kube-controller-manager-test-k8s-master3    1/1     Running   0          110s
kube-system   kube-flannel-ds-fpwjz                       1/1     Running   0          53s
kube-system   kube-flannel-ds-h8g9l                       1/1     Running   0          53s
kube-system   kube-flannel-ds-nx74s                       1/1     Running   0          53s
kube-system   kube-flannel-ds-qvplc                       1/1     Running   0          53s
kube-system   kube-flannel-ds-xgjdz                       1/1     Running   0          53s
kube-system   kube-proxy-75x67                            1/1     Running   0          3m52s
kube-system   kube-proxy-7dsxx                            1/1     Running   0          2m12s
kube-system   kube-proxy-nk5hj                            1/1     Running   0          4m45s
kube-system   kube-proxy-pzcfl                            1/1     Running   0          2m53s
kube-system   kube-proxy-s4dtt                            1/1     Running   0          2m20s
kube-system   kube-scheduler-test-k8s-master1             1/1     Running   1          4m50s
kube-system   kube-scheduler-test-k8s-master2             1/1     Running   0          3m51s
kube-system   kube-scheduler-test-k8s-master3             1/1     Running   0          92s
```

```
root@test-k8s-master1:~# kubectl get nodes
NAME               STATUS   ROLES                  AGE     VERSION
test-k8s-master1   Ready    control-plane,master   5m6s    v1.21.6
test-k8s-master2   Ready    control-plane,master   4m1s    v1.21.6
test-k8s-master3   Ready    control-plane,master   3m2s    v1.21.6
test-k8s-worker1   Ready    <none>                 2m29s   v1.21.6
test-k8s-worker2   Ready    <none>                 2m21s   v1.21.6
root@test-k8s-master1:~#
```

4、文档内容记录

https://zahui.fan/posts/b86d9e9f/#%E5%AE%89%E8%A3%85kubeadmkubelet-%E5%92%8C-kubectl

准备机器

我的机器详情如下, 配置至少为4C4G

hostname	IP	作用
public	10.0.0.3	ingress和apiserver的负载均衡，nfs存储
master1	10.0.0.11	k8s master节点
master2	10.0.0.12	k8s master节点
master3	10.0.0.13	k8s master节点
worker1	10.0.0.21	k8s worker节点
worker2	10.0.0.22	k8s worker节点

每台机器都做域名解析，或者绑定hosts(可选但建议)

Bash

```
vim /etc/hosts

10.0.0.3  public kube-apiserver
10.0.0.11 master1
10.0.0.12 master2
10.0.0.13 master3
```

每台机器都关闭防火墙和SELinux

负载均衡机器必须要关闭,因为6443不是nginx的标准端口,会被selinux拦截, 防火墙也需要放行6443端口, 可以考虑直接关闭

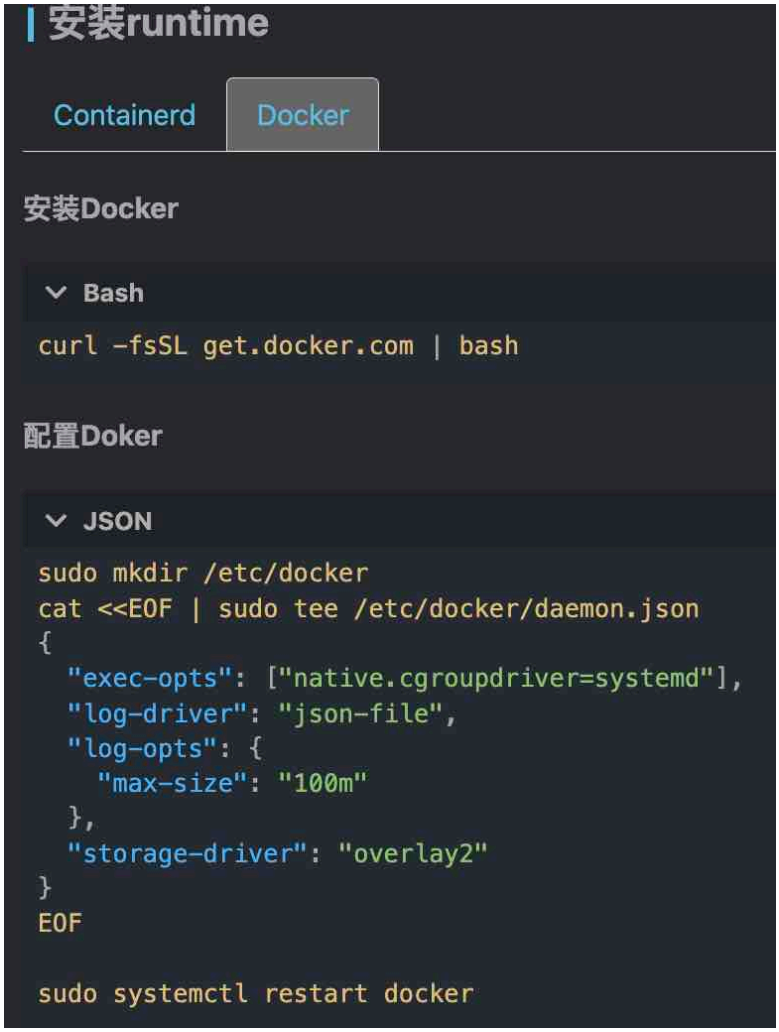
sudo systemctl disable --now firewalld

setenforce 0

sed -i "s/^SELINUX=.*/SELINUX=disabled/g" /etc/selinux/config

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system

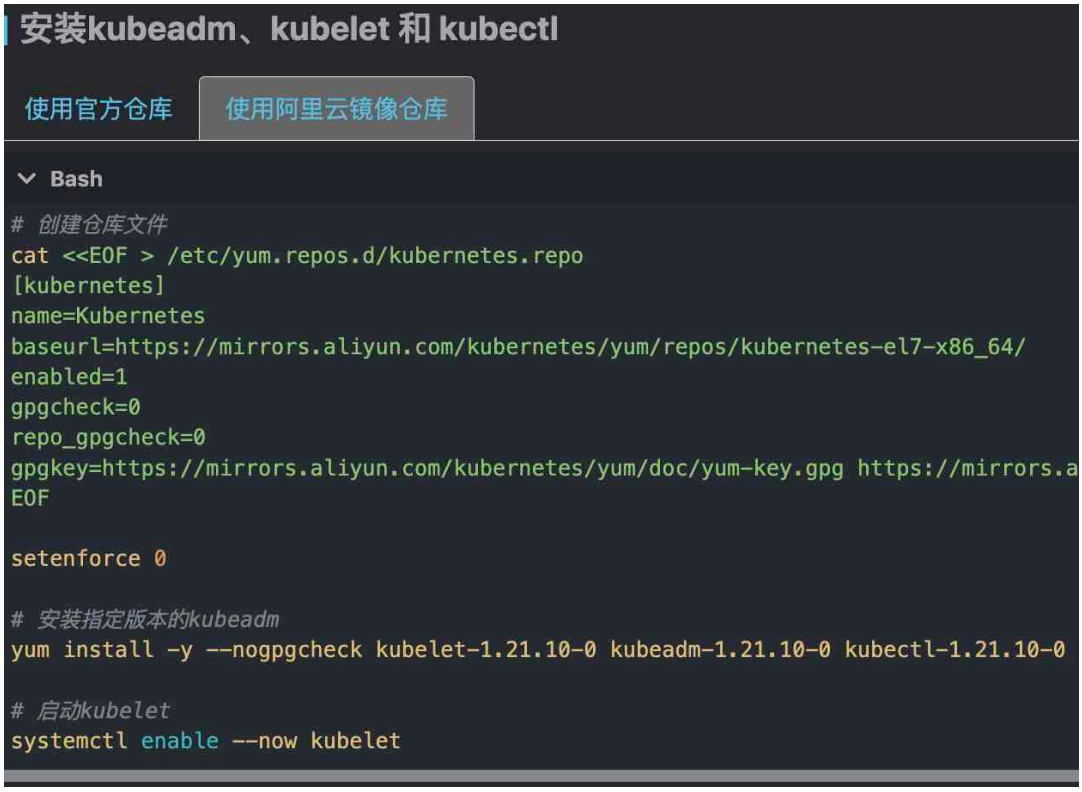


curl -fsSL get.docker.com | bash
sudo mkdir /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF

sudo systemctl restart docker

这是containerd的配置：

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# 设置必需的 sysctl 参数，这些参数在重新启动后仍然存在。
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

# 应用 sysctl 参数而无需重新启动
sudo sysctl --system
```



创建仓库文件
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=0
repo_gpgcheck=0
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

安装指定版本的kubeadm
yum install -y --nogpgcheck kubelet-1.21.10-0 kubeadm-1.21.10-0 kubectl-1.21.10-0

启动kubelet
systemctl enable --now kubelet

这是使用官方仓库的配置：
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-
\$basearch
enabled=1
gpgcheck=0
repo_gpgcheck=0
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF

# 将 SELinux 设置为 permissive 模式（相当于将其禁用）
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/'
/etc/selinux/config

sudo yum install -y kubelet-1.21.10-0 kubeadm-1.21.10-0 kubectl-1.21.10-0 --
disableexcludes=kubernetes

sudo systemctl enable --now kubelet


```
yum install -y haproxy
--- 前面保持默认配置 ---

frontend k8s_api_fe
    bind :6443
    default_backend k8s_api_be
    mode tcp
    option tcplog
backend k8s_api_be
    balance source
    mode tcp
    server      master1 master1:6443 check
    server      master2 master2:6443 check
    server      master3 master3:6443 check

frontend http_ingress_traffic_fe
    bind :80
    default_backend http_ingress_traffic_be
    mode tcp
```

```
    option tcplog
backend http_ingress_traffic_be
    balance source
    mode tcp
    server      worker1 10.0.0.21:30080 check   # 这里需要更改成ingress的
NodePort
    server      worker2 10.0.0.22:30080 check   # 这里需要更改成ingress的
NodePort

frontend https_ingress_traffic_fe
    bind *:443
    default_backend https_ingress_traffic_be
    mode tcp
    option tcplog
backend https_ingress_traffic_be
    balance source
    mode tcp
    server      worker1 10.0.0.21:30443 check   # 这里需要更改成ingress的
NodePort
    server      worker2 10.0.0.22:30443 check   # 这里需要更改成ingress的
NodePort
```

这是nginx的：

```
vim nginx.conf 在文件最后添加
stream {
    include stream.conf;
}
然后vim /etc/nginx/stream.conf
upstream k8s-apiserver {
    server master1:6443;
    server master2:6443;
    server master3:6443;
}
server {
```

```
    listen 6443;
    proxy_connect_timeout 1s;
    proxy_pass k8s-apiserver;
}
upstream ingress-http {
    server 10.0.0.21:30080;   # 这里需要更改成ingress的NodePort
    server 10.0.0.22:30080;   # 这里需要更改成ingress的NodePort
}
server {
    listen 80;
    proxy_connect_timeout 1s;
    proxy_pass ingress-http;
}
upstream ingress-https {
    server 10.0.0.21:30443;   # 这里需要更改成ingress的NodePort
    server 10.0.0.22:30443;   # 这里需要更改成ingress的NodePort
}
server {
    listen 443;
    proxy_connect_timeout 1s;
    proxy_pass ingress-https;
}
```

kubeadm config images pull --kubernetes-version 1.21.10

sudo kubeadm init \

--kubernetes-version 1.21.10 \

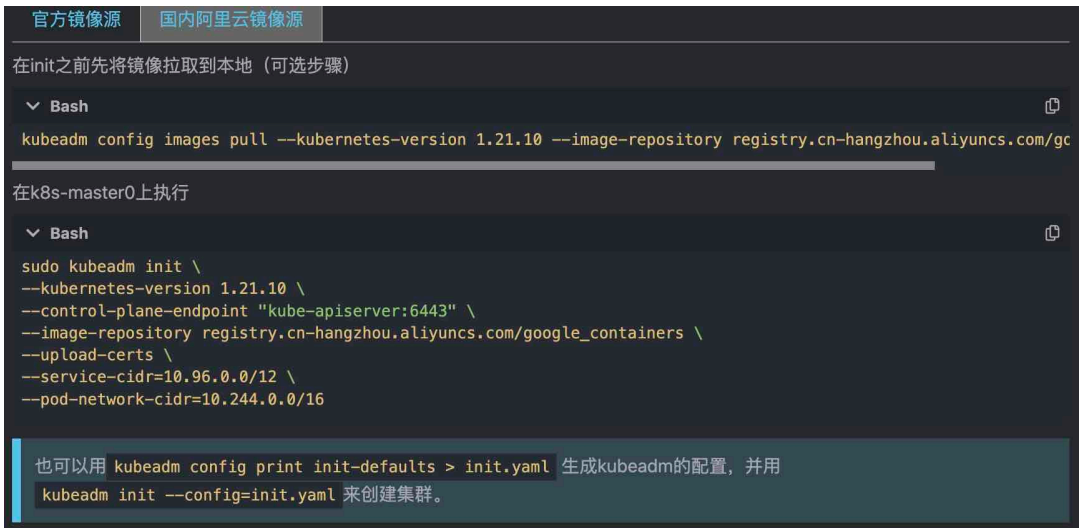
--control-plane-endpoint "kube-apiserver:6443" \

--upload-certs \

--service-cidr=10.96.0.0/12 \

--pod-network-cidr=10.244.0.0/16

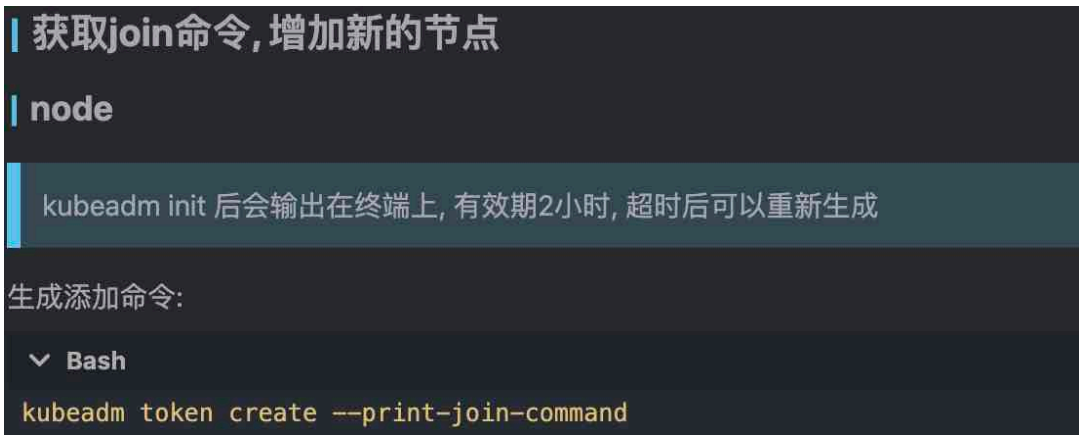
阿里云镜像源：



kubeadm config images pull --kubernetes-version 1.21.10 --image-repository registry.cn-hangzhou.aliyuncs.com/google_containers

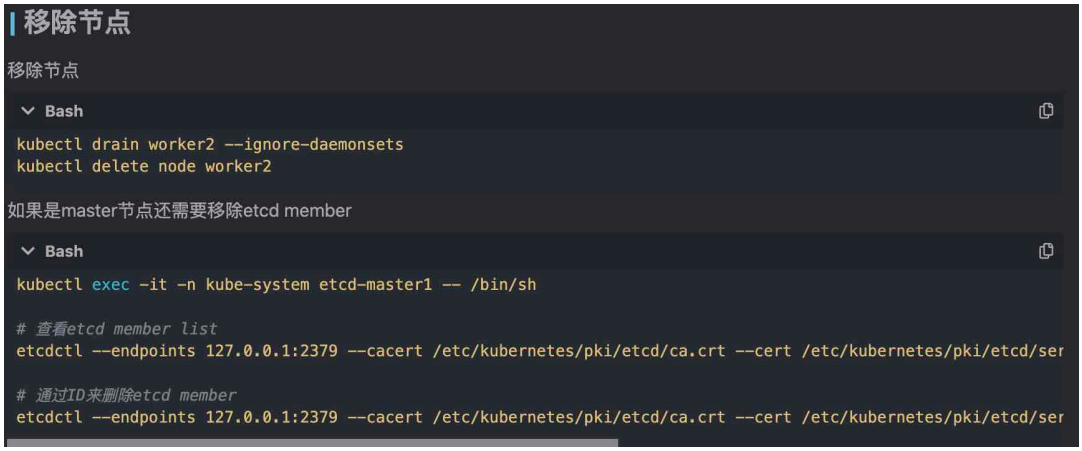
sudo kubeadm init \
--kubernetes-version 1.21.10 \
--control-plane-endpoint "kube-apiserver:6443" \
--image-repository registry.cn-hangzhou.aliyuncs.com/google_containers \
--upload-certs \
--service-cidr=10.96.0.0/12 \
--pod-network-cidr=10.244.0.0/16

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



kubeadm init phase upload-certs --upload-certs

kubeadm token create --print-join-command
echo "$(kubeadm token create --print-join-command) --control-plane --certificate-key $(kubeadm init phase upload-certs --upload-certs | tail -1)"



kubectl exec -it -n kube-system etcd-master1 -- /bin/sh

查看etcd member list
etcdctl --endpoints 127.0.0.1:2379 --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/server.crt --key /etc/kubernetes/pki/etcd/server.key member list

通过ID来删除etcd member
etcdctl --endpoints 127.0.0.1:2379 --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/server.crt --key /etc/kubernetes/pki/etcd/server.key member remove 12637f5ec2bd02b8