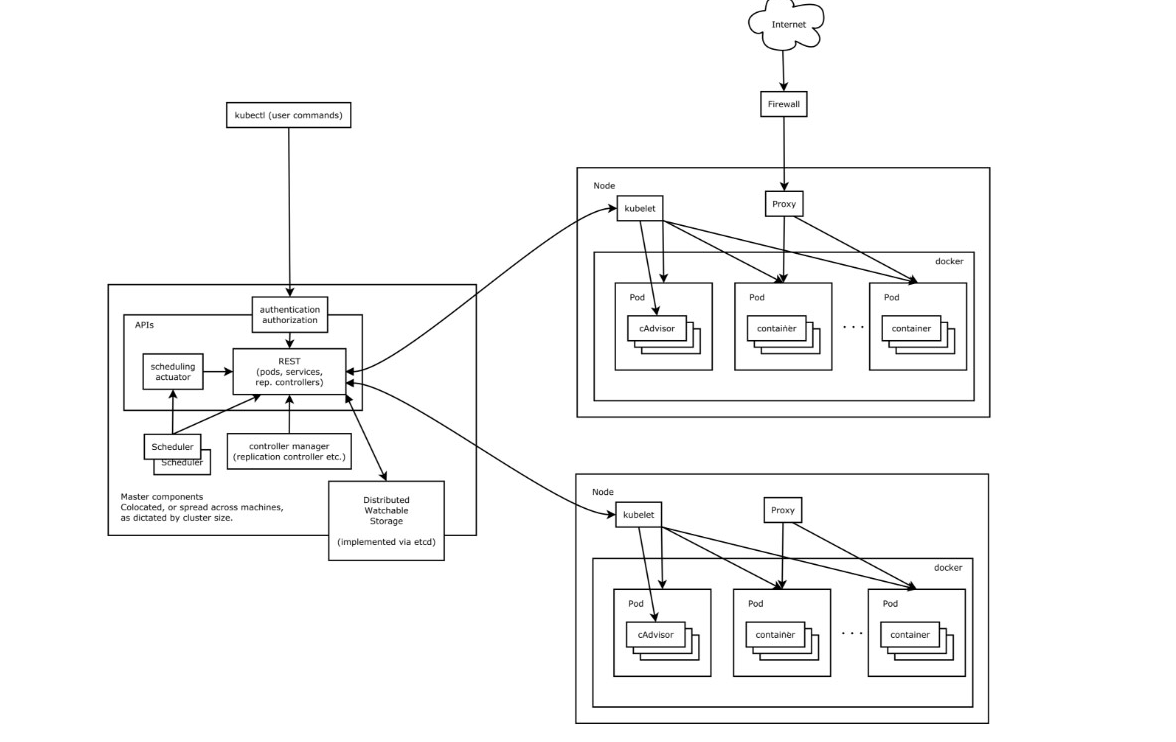
# Kubernetes集群部署



# 配置K8S的yum源# 这部分用是阿里云的源，如果可以访问Google，则建议用官方的源

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

[kubernetes]

name=Kubernetes

baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

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

EOF

# 官方源配置如下

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

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

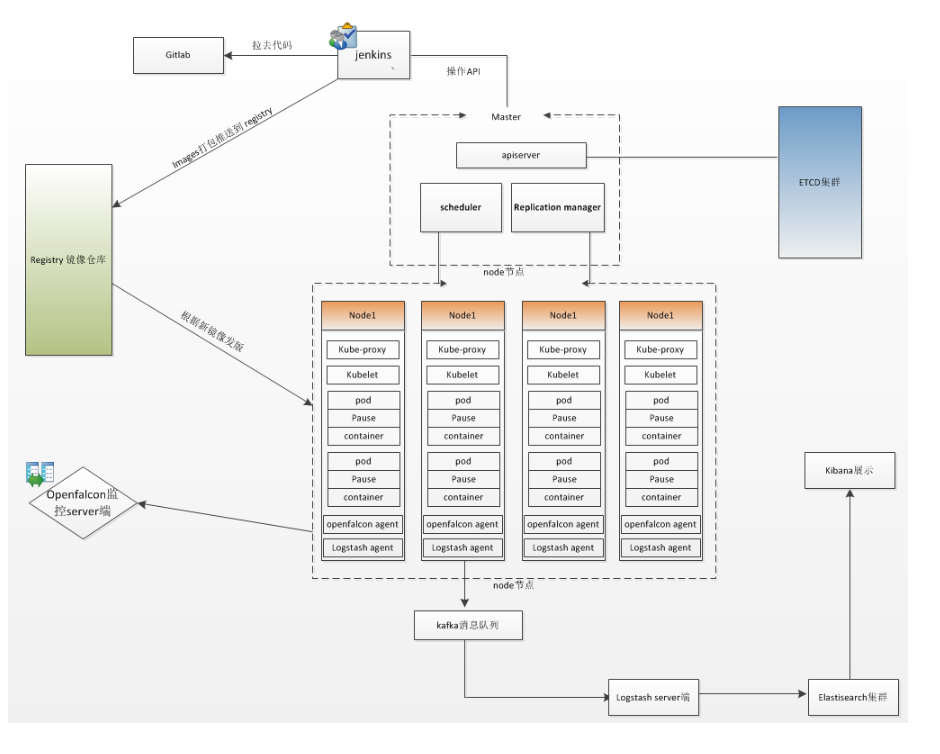
enabled=1

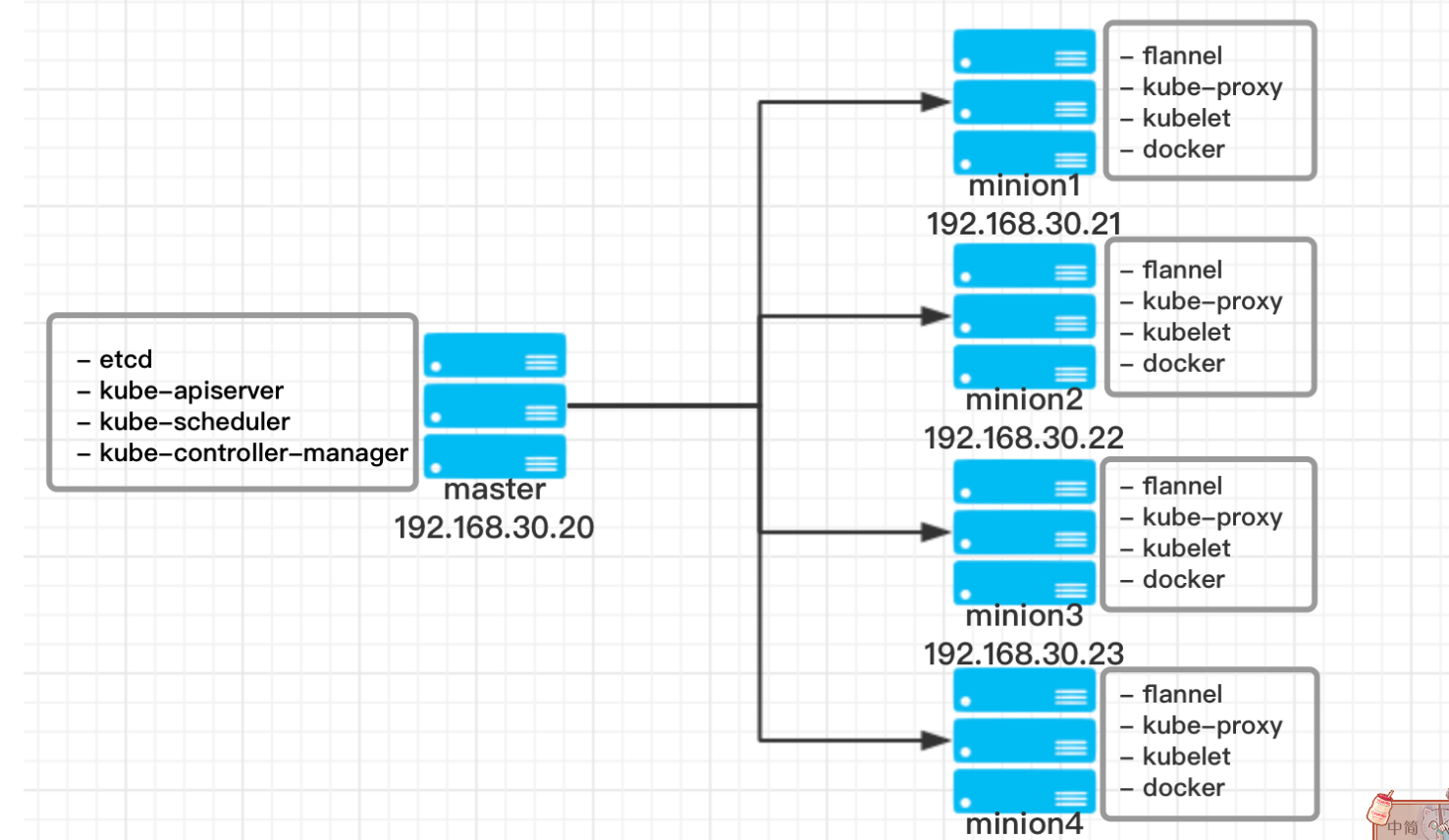
gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

EOF





安装K8S步骤的地址：<https://www.cnblogs.com/spll/p/10033316.html>

第二个网址：<https://www.cnblogs.com/xkops/p/6169034.html>

借鉴博客：https://www.cnblogs.com/xkops/p/6169034.html

此博客里面有每个k8s配置文件的注释：https://blog.csdn.net/qq\_35904833/article/details/78190257

### **k8s的整个集群运行原理：【重点核心知识很重要】**

　　　　　　　　　　master主机上的kube-controller-manager是整个集群的控制管理中心，kube-controler-manager中的node controller模块 通过apiservice提供的监听接口，实时监控node机的状态信息。

-提供集群的控制

-对集群进行全局决策

-检测和响应集群事件

 当某个node机器宕机，controller-manager就会及时排除故障并自动修复。

　　　　　　　　　　node节点机上的kubelet进程每隔一段时间周期就会调用一次apiservice接口报告自身状态，apiservice接口接受到这些信息后将节点状态更新到ectd中。kubelet也通过apiservice的监听接口监听pod信息，如果监控到新的pod副本被调度绑定到本节点，则执行pod对应的容器的创建和启动，如果监听到pod对象被删除，则删除本节点对应的pod容器。（目前对pod、容器、镜像这些概念还不是很清晰，无法在大脑中构建这都是些什么玩意，先做个笔记记着吧）

**master主机上192.168.26.277必须要有的组件：**

　　　　　　　　　　etcd　　：提供分布式数据存储的数据库吧，用于持久化存储k8s集群的配置和状态

　　　　　　　　　　kube-apiserver：api service提供了http rest接口，是整个集群的入口，K8s其它组件之间不直接通信，而是通过API server通信的。（只有API server连接了etcd，即其它组件更新K8s集群的状态时，只能通过API server读写etcd中的数据）

　　　　　　　　　　kube-scheduler：scheduler负责资源的调度

　　　　　　　　　　kube-controller-manager：整个集群的管理控制中心，此组件里面是由多个控制器组成的，如：Replication Manager（管理ReplicationController 资源），ReplicaSet Controller，PersistentVolume controller。主要作用用来复制组件、追踪工作结点状态、处理失败结点

**node节点机上192.168.26.228必须要有的组件：**

　　　　　　　　　　flannel：（装包-配置-起服务-生成随机的IP ifconfig flannel.1）,实质上是一种“覆盖网络”，也就是将TCP数据包装在另一种网络包里面进行路由转发

和通信。

　　　　　　　　　　kube-proxy：主要负责为pod对象提供代理；实现service的通信和负载均衡

　　　　　　　　　　kubelet：用来管理node节点机上的容器,主要负责监视到指派到它的pod,包括创建、修改、删除

　　　　　　　　　　docker：运行项目镜像容器的组件

如图5，K8S是二层C/S架构Master作为管控节点，负责集群调度，监控管控与元数据存储；Node是业务节点，负责容器运行、资源分配与管理。用户通过UI或API与Master通信交付，然后把相关请求转发到Node上进行处理。

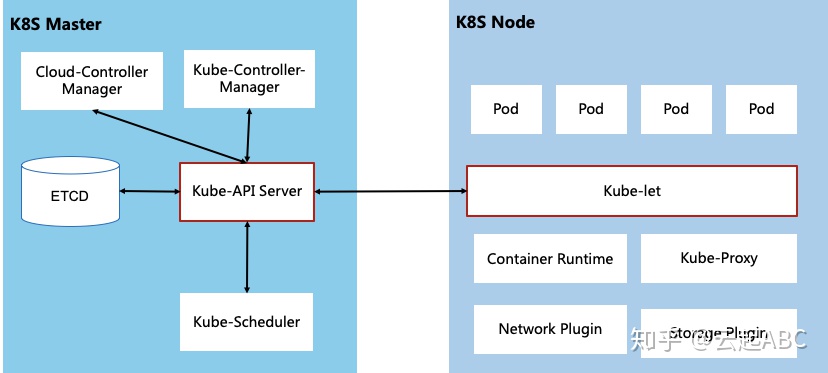


图5：K8S系统架构

**Master的主要组件功能如下：**

**1）API Server：**K8S资源对象的唯一操作入口，接收、校验并处理 API 请求；Master中所有组件互不相连，都通过API Server 进行消息传送与任务处理；

**2）Controller Manager：**集群管理控制中心，负责集群状态检测与故障恢复等工作。管理多个控制器：Node Controller、Endpoint Controller、Service Account & Token Controller等。

**3）Scheduler：**集群调度器，负责容器调度操作，基于资源与策略等将Pod调度到合适的Node上运行。

**4）ETCD：**负责集群元数据与状态保存。是一个基于RAFT协议的高可用的分布式Key-Value存储系统。

**5）Cloud Controller Manager：**负责与云厂商组件相关的控制器交互，如 Node Controller、Route Controller、Service Controller、Volume Controller。K8S控制器由“声明式”的API驱动，提前定义K8S资源对象的期待状态，控制器自主运行，异步地控制系统想最终状态趋近。

**Node的主要组件功能如下：**

**1）Kube-let：**与API Server交互获取容器Pod创建请求，转到对应的Container Runtime处理。监控并上报Pod运行情况、镜像与资源状态等。

**2）kube-proxy：**Node上的网络代理，维护集群服务的网络规则与网络通信。

**3）Container Runtime：**接收Kube-let请求在OS创建容器的运行时环境。

**4）Storage Plugin/Network Plugin,**负责与外部组件通信，容器存储或网络的配置使用。

**2.3** **Pod原理与调度**

Pod 是K8S的最小调度以及资源分配单元，包含容器运行环境与资源。Pod之间相互隔离。通常一个Pod运行一个容器。当某些容器间关系非常紧密Tightly coupled（如文件交换、频繁的RPC调用、共享网络或存储等），需要在一个Pod运行多个容器方便调度管理。

Pod中可以通过Infra Container共享网络。Infra container，大小100-200KB，最先启动，并绑定了Pod的唯一IP地址与各种网络资源，Pod内其他容器通过加入infra container的Network namespace来实现网络共享。Pod中的多个容器也可以通过挂载相同的共享路径实现存储共享。

Pod 中可定义Init Container在应用启动前先启动来执行一次性的辅助任务，比如执行脚本、拷贝文件到共享目录、日志收集、应用监控等。将辅助功能与主业务容器解耦，实现独立发布和能力重用。

一个Pod就是一个应用运行实例，可以运行多个Pod实现应用横向扩展。当调度失败或运行失败时，Pod本身没有自恢复能力，需要借助上层的controller来通过采用提前定义好的Pod模板创建Pod，提供Pod重启、重建或迁移能力。

K8S调度主要涉及资源调度和关系的调度，采用两种机制过滤（Filtering）和打分（Scoring）。通过过滤机制选择备用节点，过滤因素包括CPU/内存/存储、Pod与Nod匹配、Pod与Pod匹配等；然后将Pod调度到分数最高的节点，打分依据包含亲和性/反亲和性、资源水位等等。**完整的调度流程如下：**

1） 用户通过UI或CLI提交Pod部署请求时，会首先提交给API Server，然后信息存储到ETCD；

2） Scheduler通过Watch或通知机制获得Pod需要调度的请求，通过过滤和打分机制做出调度决策，并通知API Server调度结果；

3） API Server收到调度结果写入ETCD，并通知响应Node上的Kube-let。

4） Kube-let调用Container Runtime配置其他运行环境，并调用对应的插件去分片存储和网络资源。

1、所有机器上执行以下命令，准备安装环境：(注意是所有机器，主机master，从机node都要安装)

1.1、安装epel-release源

yum -y install epel-release

1.2、所有机器关闭防火墙

systemctl stop firewalld

systemctl disable firewalld  
  
setenforce 0  
  
#查看防火墙状态

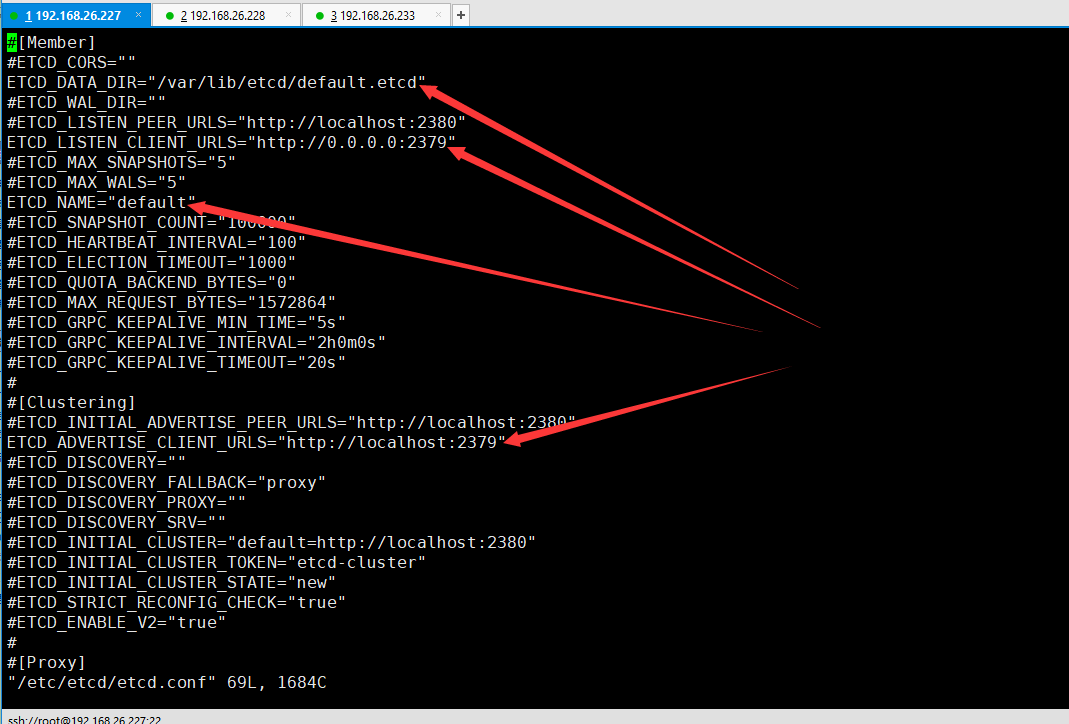
firewall-cmd --state

1. 现在开始master主机上192.168.26.227安装kubernetes Master

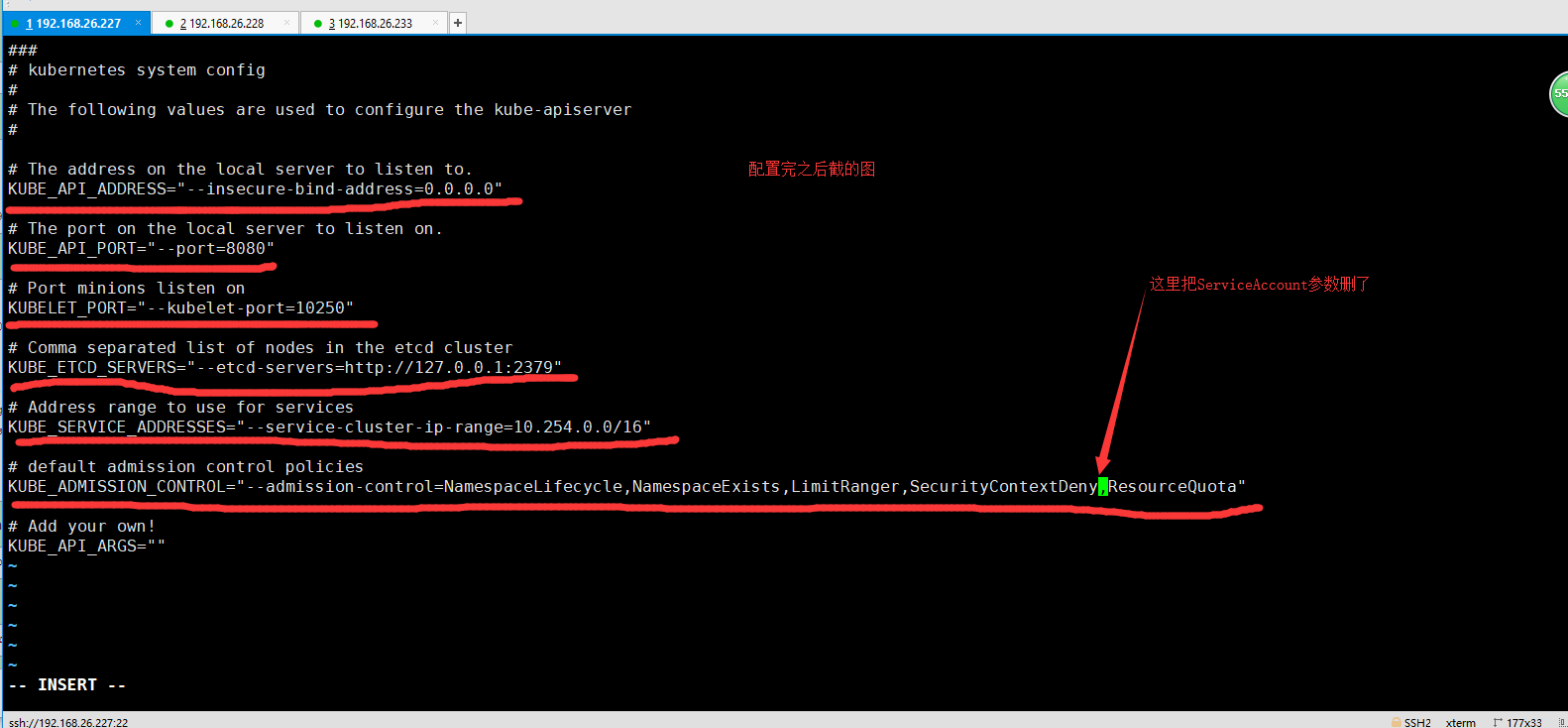
　2.1、使用yum安装etcd、kubernetes-master

yum -y install etcd kubernetes-master

　　　　　　　　2.2、编辑：vi /etc/etcd/etcd.conf文件，修改结果如下：

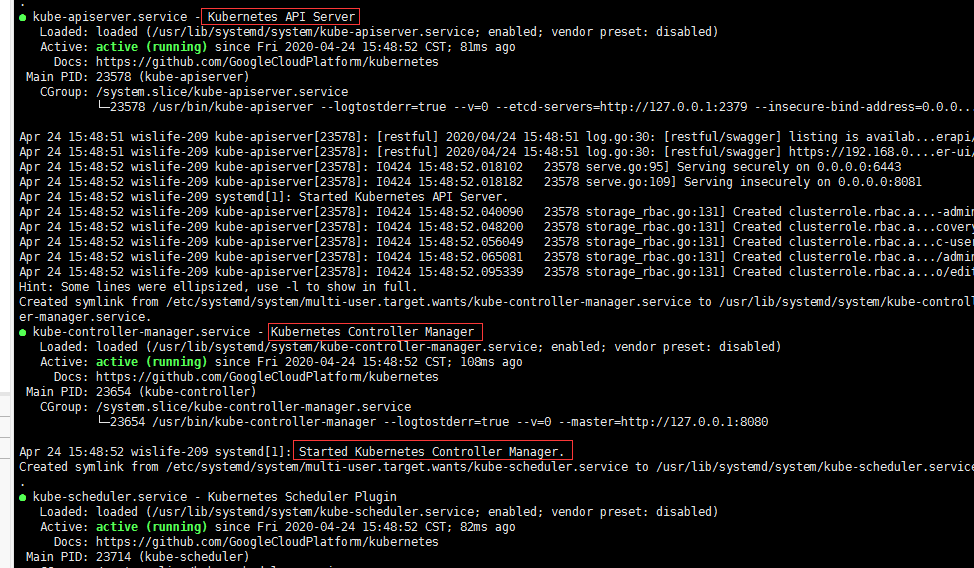


2.3、配置：vi /etc/kubernetes/apiserver文件，配置结果如下：

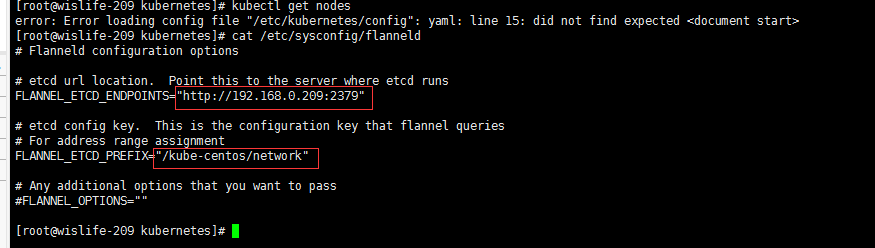


2.4、启动etcd、kube-apiserver、kube-controller-manager、kube-scheduler等服务，并设置开机启动。

for SERVICES in etcd kube-apiserver kube-controller-manager kube-scheduler; do systemctl restart $SERVICES;systemctl enable $SERVICES;systemctl status $SERVICES ; done



　2.5、在etcd中定义flannel网络，/etc/sysconfig/flannel



etcdctl mk /atomic.io/network/config '{"Network":"172.17.0.0/16"}'

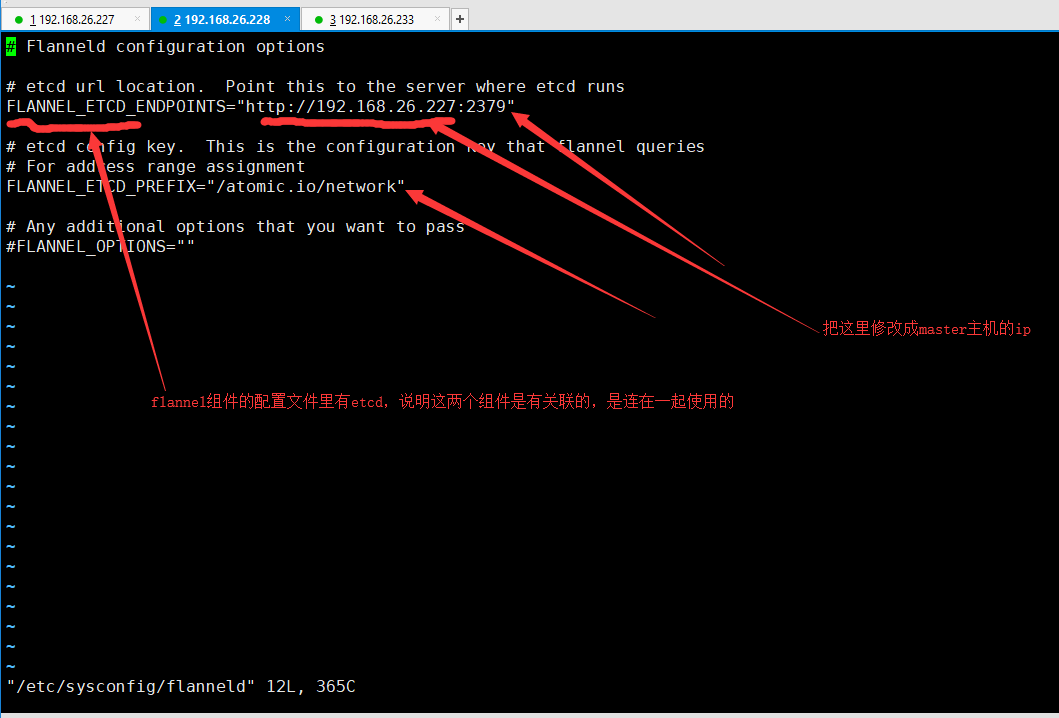
### **》》》》》》》》以上master主机上的配置安装什么的都弄完了》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》》**

### **3、接下来弄node从机上的配置安装什么的**

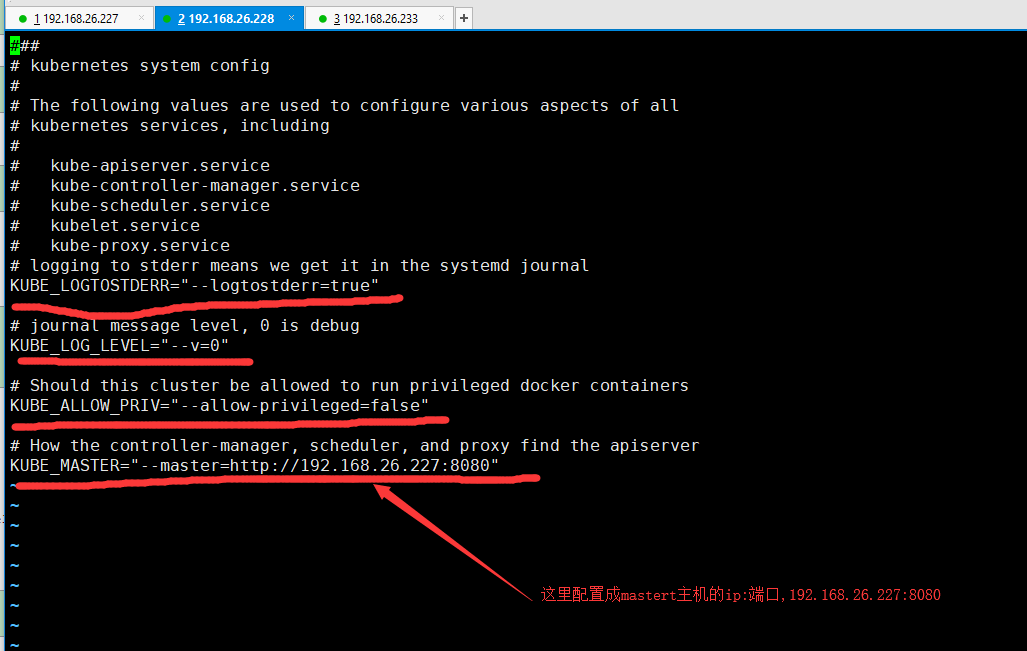
　　3.1、在node机上192.168.0.207安装kubernetes Node和flannel组件应用

yum -y install flannel kubernetes-node

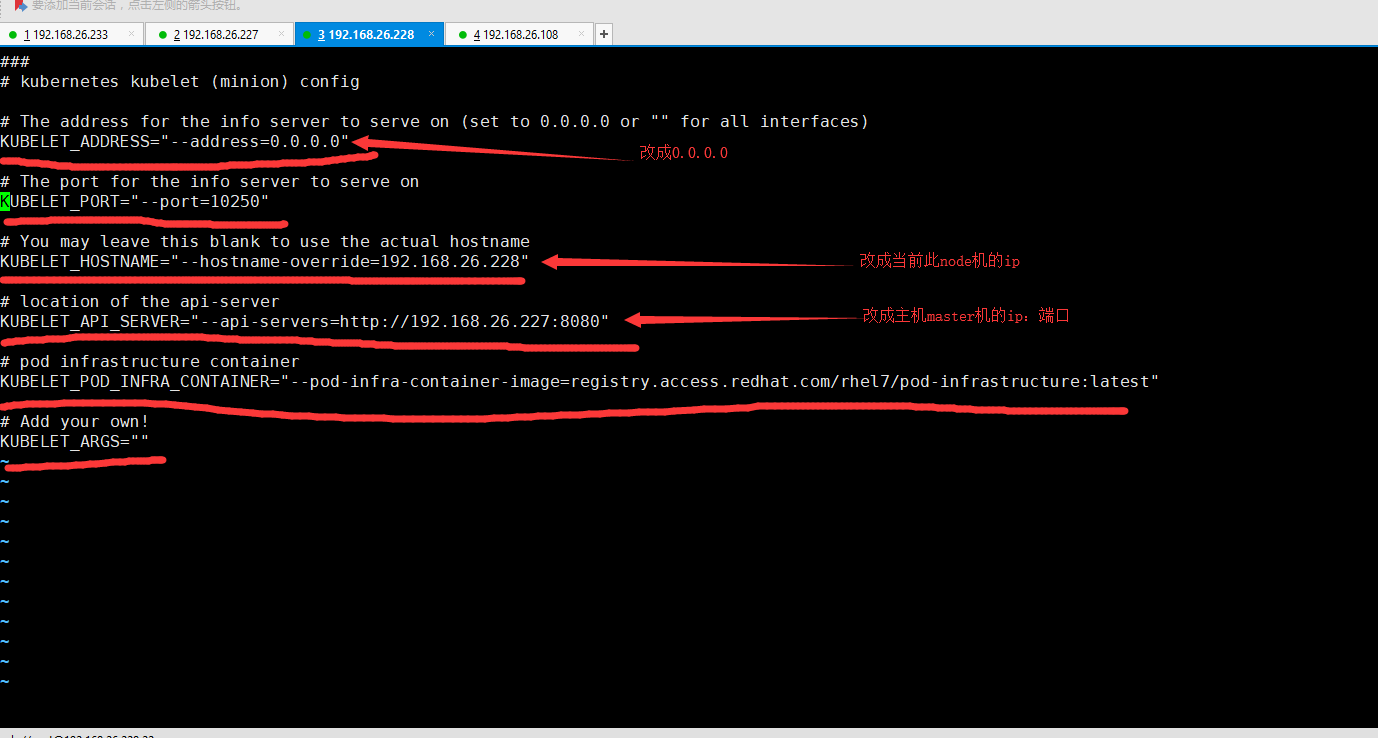
3.2、为flannel网络指定etcd服务，修改/etc/sysconfig/flanneld文件，配置结果如下图：



　3.3、修改：vi /etc/kubernetes/config文件，配置结果如下图：



3.4、修改node机的kubelet配置文件/etc/kubernetes/kubelet



3.5、node节点机上启动kube-proxy,kubelet,docker,flanneld等服务，并设置开机启动。

for SERVICES in kube-proxy kubelet docker flanneld;do systemctl restart $SERVICES;systemctl enable $SERVICES;systemctl status $SERVICES; done

### **》》》》》》以上所有master主机，node节点机上的配置就完成了，接下来看看k8s集群是否搭建起来了》》》》》》》》》》》》》》》》》》》**

在master主机上192.168.26.227执行如下命令，查看运行的node节点机器：

kubectl get nodes