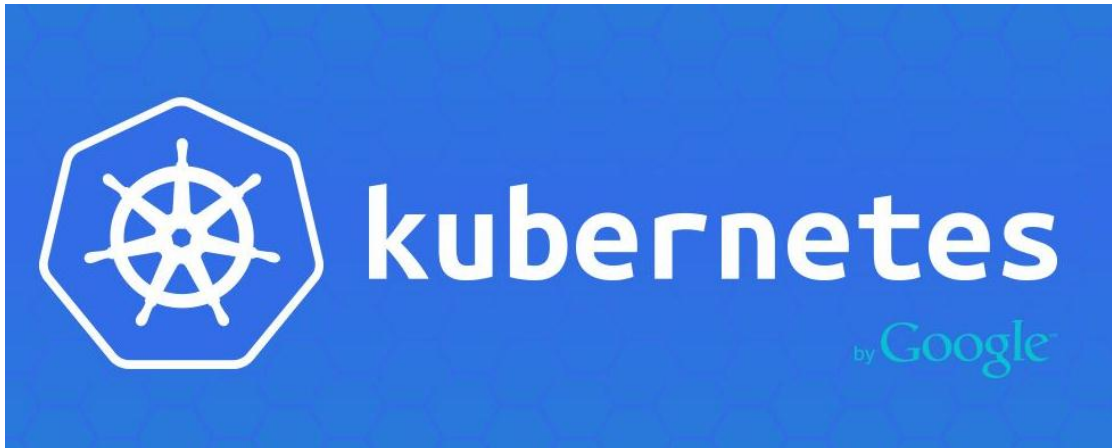


第十一章 搭建 Kubernetes 容器集群管理系统

本节所讲内容：

- 11.1 Kubernetes 和相关组件的介绍
- 11.2 配置 yum 源安装 kubernetes 及组件
- 11.3 配置 etcd 和 master 节点
- 11.4 配置 minion1 节点
- 11.5 配置 minion2 节点并总结 Kubernetes 所有服务和端口号



实验环境：需要三台全新的 CENTOS7.5 系统

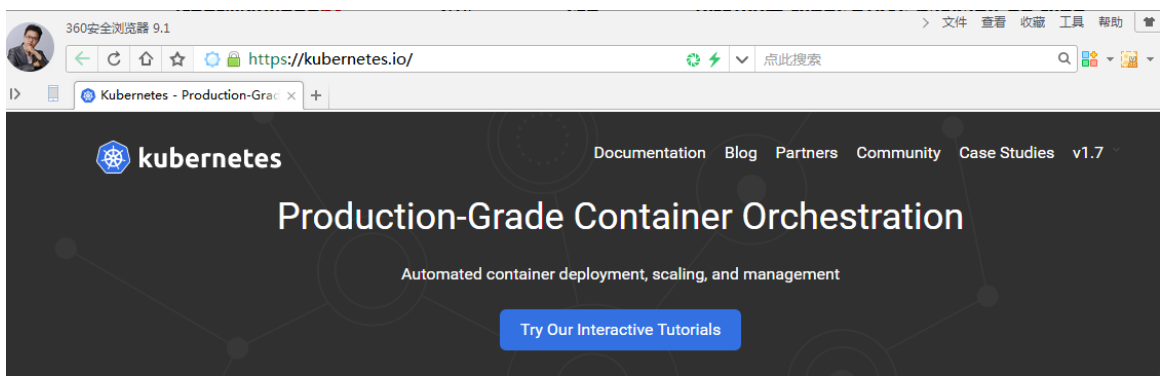
11.1.1 Kubernetes 概述

Kubernetes 是 Google 开源的容器集群管理系统，基于 Docker 构建一个容器的调度服务，提供资源调度、均衡容灾、服务注册、动态扩缩容等功能套件。基于容器的云平台

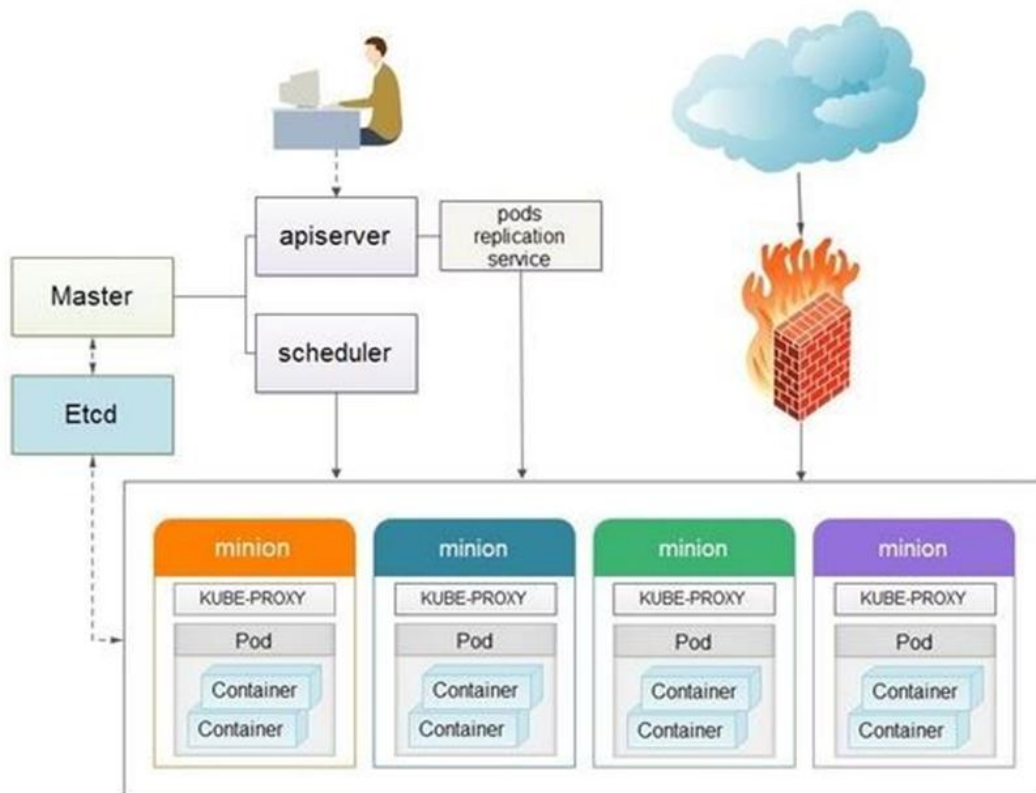
Kubernetes 基于 docker 容器的云平台，简写成：k8s。

openstack 基于 kvm 虚拟机云平台。

官网：<https://kubernetes.io/>

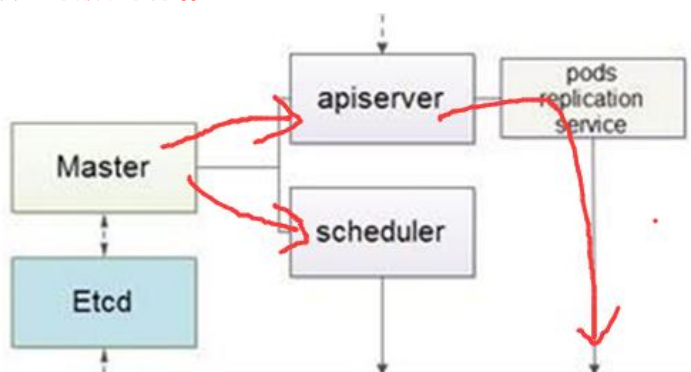


Kubernetes 的架构设计图：



11.1.2 Kubernetes 常见组件介绍

先按以下顺序来介绍：



1、master：kubernetes 管理结点

2、apiserver：提供接口服务，用户通过 apiserver 来管理整个容器集群平台。API Server 负责和 etcd 交互（其他组件不会直接操作 etcd，只有 API Server 这么做），整个 kubernetes 集群的所有的交互都是以 API Server 为核心的。如：1、所有对集群进行的查询和管理都要通过 API 来进行 2、所有模块之间并不会互相调用，而是通过和 API Server 打交道来完成自己那部分的工作、API Server 提供的验证和授权保证了整个集群的安全

3、scheduler kubernetes 调度服务

4、Replication Controllers 复制，保证 pod 的高可用

Replication Controller 是 Kubernetes 系统中最有用的功能，实现复制多个 Pod 副本，往往一个应用需要多个 Pod 来支撑，并且可以保证其复制的副本数，即使副本所调度分配的宿主机出现异常，通过 Replication Controller 可以保证在其它宿主机启用同等数量的 Pod。Replication Controller 可以通过 repcon 模板来创建多个 Pod 副本，同样也可以直接复制已存在 Pod，需要通过 Label selector 来关联。

接下介绍，从下往上说，从你最熟悉的知识开始



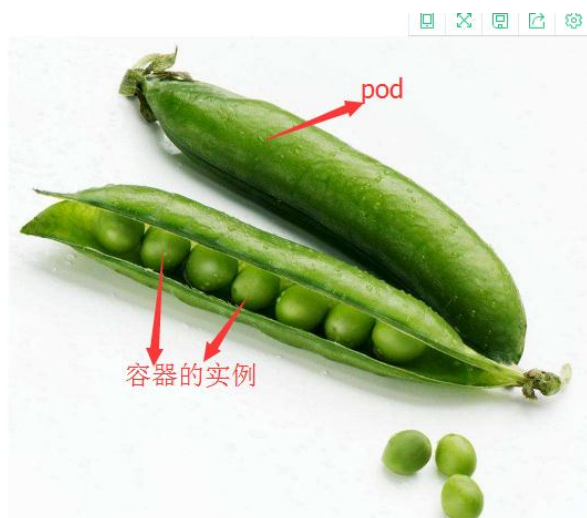
5、minion：真正运行容器 container 的物理机。kubernetes 中需要很多 minion 机器，来提供运算。

minion [ˈmɪniən] 爪牙

6、container：容器，可以运行服务和程序

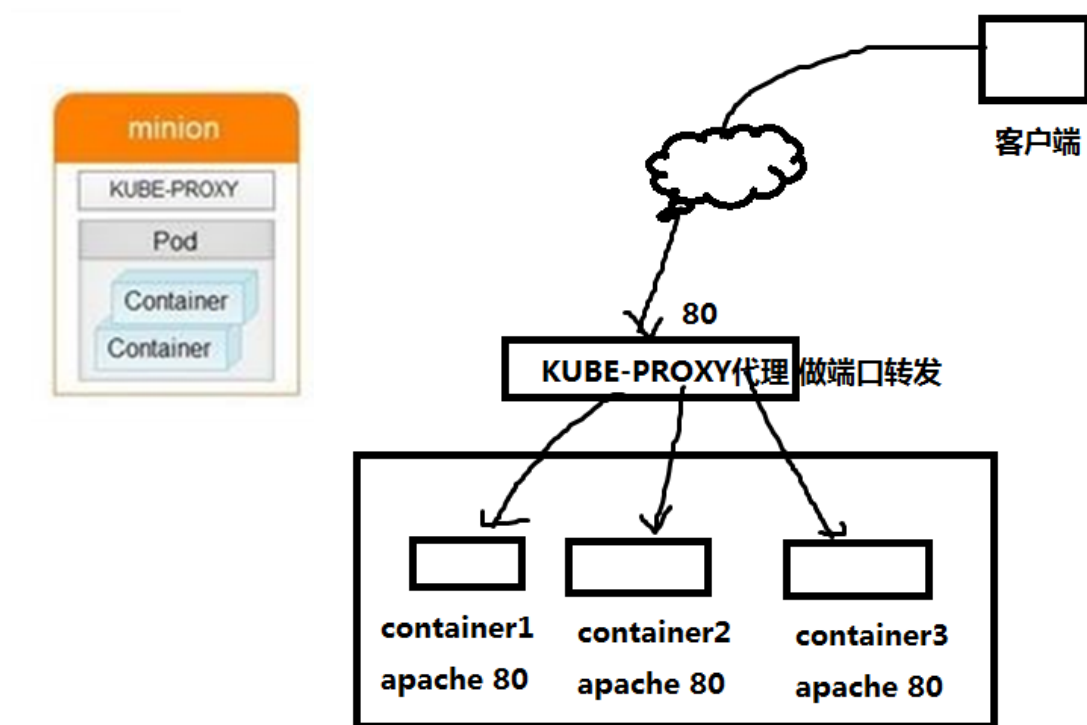
7、Pod：在 Kubernetes 系统中，调度的最小颗粒不是单纯的容器，而是抽象成一个 Pod，Pod 是一个可以被创建、销毁、调度、管理的最小的部署单元。pod 中可以包括一个或一组容器。

pod [pɒd] 豆荚



8、Kube_proxy 代理 做端口转发，相当于 LVS-NAT 模式中的负载调度器

Proxy 解决了同一宿主机，相同服务端口冲突的问题，还提供了对外服务的能力，Proxy 后端使用了随机、轮循负载均衡算法。



8、etcd etcd 存储 kubernetes 的配置信息，可以理解为是 k8s 的数据库，存储着 k8s 容器云平台中所有节点、pods、网络等信息。

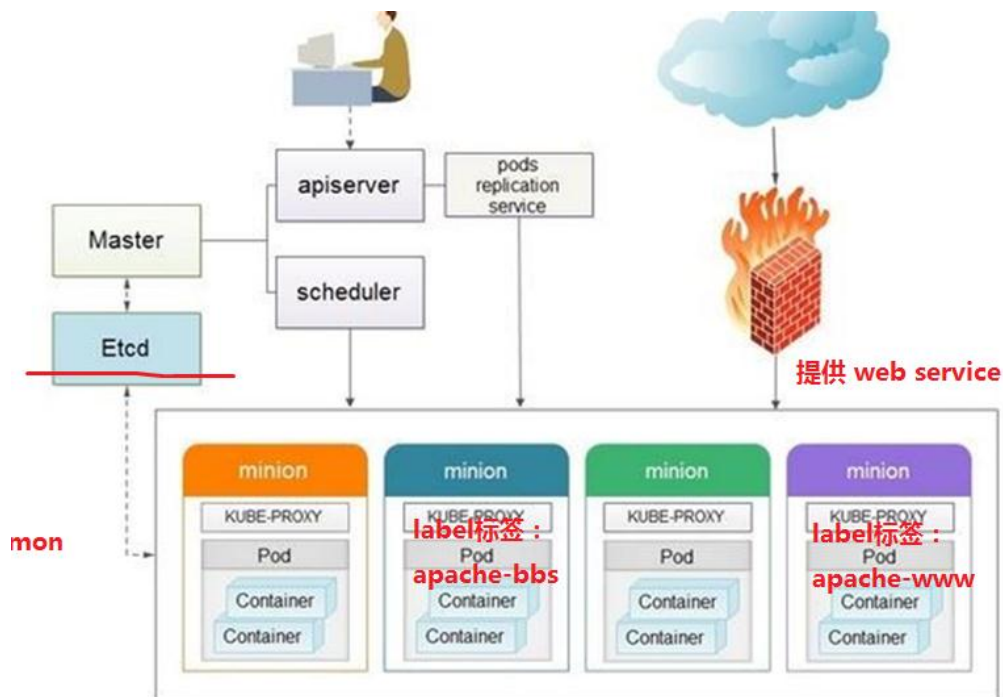
互动：linux 系统中/etc 目录作用是存储配置文件。所以 etcd (daemon) 是一个存储配置文件的后台服务。

接下几个名词不在图片中，我们也介绍一下它们

9、Services : Services 是 Kubernetes 最外围的单元，通过虚拟一个访问 IP 及服务端口，可以访问我们定义好的 Pod 资源，目前的版本是通过 iptables 的 nat 转发来实现，转发的目标端口为 Kube_proxy 生成的随机端口。

10、Labels 标签

Labels 是用于区分 Pod、Service、Replication Controller 的 key/value 键值对，仅使用在 Pod、Service、Replication Controller 之间的关系识别，但对这些单元本身进行操作时得使用 name 标签。



11、 Deployment

Deployment [di'plɔimənt] 部署

Kubernetes Deployment 用于更新 Pod 和 Replica Set (下一代的 Replication Controller) 的方法, 你可以在 Deployment 对象中只描述你所期望的理想状态 (预期的运行状态), Deployment 控制器会将现在的实际状态转换成期望的状态。例如, 将所有的 webapp:v1.0.9 升级成 webapp:v1.1.0, 只需创建一个 Deployment, Kubernetes 会按照 Deployment 自动进行升级。通过 Deployment 可以用来创建新的资源。

Deployment 可以帮我们实现无人值守的上线, 大大降低我们的上线过程的复杂沟通、操作风险。

12、Kubelet 命令 : Kubelet 和 Kube-proxy 都运行在 minion 节点上。

Kube-proxy 实现 Kubernetes 网络相关内容。

Kubelet 命令管理 Pod、Pod 中容器及容器的镜像和卷等信息。



总结 : 总结各组件之间的关系

1、Kubernetes 的架构由一个 master 和多个 minion 组成, master 通过 api 提供服务, 接受 kubectl 的请求来调度管理整个集群。 kubectl : 是 k8s 平台的一个管理命令。

2、Replication controller 定义了多个 pod 或者容器需要运行, 如果当前集群中运行的 pod 或容

器达不到配置的数量，replication controller 会调度容器在多个 minion 上运行，保证集群中的 pod 数量。

3、service 则定义真实对外提供的服务，一个 service 会对应后端运行的多个 container。

4、Kubernetes 是个管理平台，minion 上的 kube-proxy 拥有提供真实服务公网 IP。客户端访问 kubernetes 中提供的服务，是直接访问到 kube-proxy 上的。

5、在 Kubernetes 中 pod 是一个基本单元，一个 pod 可以是提供相同功能的多个 container，这些容器会被部署在同一个 minion 上。minion 是运行 Kubelet 中容器的物理机。minion 接受 master 的指令创建 pod 或者容器。

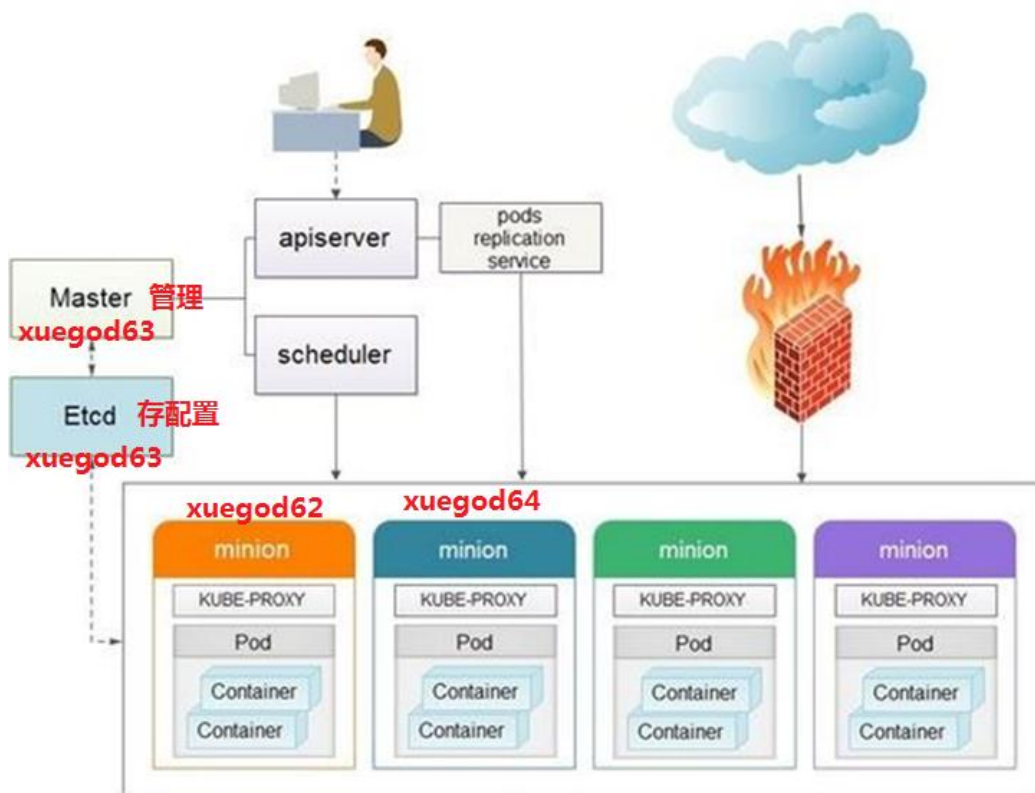
11.2 搭建 Kubernetes 容器集群管理系统

平台版本说明

节点角色	IP 地址	CPU	内存	
master	192.168.1.63	4 核	2GB	这里 master 和 etcd 共用一台机器
etcd	192.168.1.63	4 核	2GB	
node1(minion1)	192.168.1.62	4 核	2GB	
node2(minion2)	192.168.1.64	4 核	2GB	

注：正常需要 4 台机器，如果你内存不够，master 和 etcd 可以运行在同一台机器上。

实验拓扑图：



注：前几天我们学的 docker 都是单机版的，使用着就像单机版本的 KVM 一样。

11.2.1 配置 kubernetes 的 yum 源

1. 首先在所有节点添加 k8s 组件 yum 源

注：我这里是把腾讯云主机 centos7.4 中的 yum 安装 k8s 时的包下载到本地，做了一个本地 yum 源。下载的是 docker-1.12 版本

方法 1：配置本地 yum 源：

把 k8s-package.tar.gz 上传到 xuegod63 系统中：

```
[root@xuegod63 ~]# tar zxvf k8s-package.tar.gz
[root@xuegod63 ~]# vim /etc/yum.repos.d/k8s-package.repo
[k8s-package]
name=k8s-package
baseurl=file:///root/k8s-package
enabled=1
gpgcheck=0
```

本地系统镜像 yum 源：

```
[root@xuegod63 ~]# mount /dev/cdrom /mnt/
[root@xuegod63 ~]# vim /etc/yum.repos.d/centos7.repo
[centos7]
name=CentOS7
baseurl=file:///mnt
enable=1
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

```
[root@xuegod63 ~]# mv /etc/yum.repos.d/CentOS-* /opt/ #删除外网源
```

方法 2：

自己在线安装：centos 系统中自带的 yum 源就可以安装 kubernetes

想自己配置 yum 源，创建以一下文件：

```
[root@xuegod63 ~]# vim /etc/yum.repos.d/CentOS-Base.repo #插入以下内容,使用
```

aliyun 的源

```
[base]
name=CentOS-$releasever - Base
failovermethod=priority
baseurl=http://mirrors.aliyun.com/centos/7.5.1804/os/x86\_64/
gpgcheck=1
gpgkey=http://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-7
http://mirrors.aliyuncs.com/centos/RPM-GPG-KEY-CentOS-7

#released updates
[updates]
name=CentOS-$releasever - Updates
failovermethod=priority
baseurl=http://mirrors.aliyun.com/centos/7.5.1804/updates/x86\_64/
gpgcheck=1
gpgkey=http://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-7
http://mirrors.aliyuncs.com/centos/RPM-GPG-KEY-CentOS-7

#additional packages that may be useful
```

[extras]

name=CentOS-\$releasever - Extras

failovermethod=priority

baseurl=http://mirrors.aliyun.com/centos/7.5.1804/extras/x86_64/

gpgcheck=1

gpgkey=<http://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-7>

<http://mirrors.aliyuncs.com/centos/RPM-GPG-KEY-CentOS-7>

#additional packages that extend functionality of existing packages

[centosplus]

name=CentOS-\$releasever - Plus

failovermethod=priority

baseurl=http://mirrors.aliyun.com/centos/7.5.1804/centosplus/x86_64/

gpgcheck=1

enabled=0

gpgkey=<http://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-7>

<http://mirrors.aliyuncs.com/centos/RPM-GPG-KEY-CentOS-7>

#contrib - packages by Centos Users

[contrib]

name=CentOS-\$releasever - Contrib

failovermethod=priority

baseurl=http://mirrors.aliyun.com/centos/7.5.1804/contrib/x86_64/

gpgcheck=1

enabled=0

gpgkey=<http://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-7>

<http://mirrors.aliyuncs.com/centos/RPM-GPG-KEY-CentOS-7>

[epel]

name=Extra Packages for Enterprise Linux 7 - \$basearch

baseurl=http://mirrors.aliyun.com/epel/7/x86_64/

failovermethod=priority

enabled=1

gpgcheck=0

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

下面使用方法 1，自己的本地 yum 源搭建，这样更快捷

复制 k8s yum 源相关文件到其他两台机器上：

```
[root@xuegod63 ~]# scp -r /root/k8s-package 192.168.1.62:/root/
```

```
[root@xuegod63 ~]# scp /etc/yum.repos.d/k8s-package.repo
```

```
192.168.1.62:/etc/yum.repos.d/
```

```
[root@xuegod63 ~]# scp -r /root/k8s-package 192.168.1.64:/root/
```

```
[root@xuegod63 ~]# scp /etc/yum.repos.d/k8s-package.repo
```

```
192.168.1.64:/etc/yum.repos.d/
```


登录 xuegod64 :

```
[root@xuegod64 ~]# mv /etc/yum.repos.d/CentOS-* /opt/
```

登录 xuegod62 :

```
[root@xuegod62 ~]# mv /etc/yum.repos.d/CentOS-* /opt/
```

11.2.2 在各个节点上面安装 k8s 组件

配置 xuegod63 为 etcd 和 master 结点 :

```
[root@xuegod63 ~]# yum install -y kubernetes etcd flannel ntp
```

注 : Flannel 为 Docker 提供一种可配置的虚拟重叠网络。实现跨物理机的容器之间能直接访问

- 1、Flannel 在每一台主机上运行一个 agent。
- 2、flanneld , 负责在提前配置好的地址空间中分配子网租约。Flannel 使用 etcd 来存储网络配置。
- 3、ntp : 主要用于同步容器云平台中所有结点的时间。云平台中结点的时间需要保持一致。
- 4、kubernetes 中包括了服务端和客户端相关的软件包
- 5、etcd 是 etcd 服务的软件包

xuegod62 :

```
[root@xuegod62 ~]# yum install kubernetes flannel ntp -y
```

xuegod64:

```
[root@xuegod64 ~]# yum install -y kubernetes flannel ntp #每个 minion 都要安装一样
```

关闭防火墙 :

```
[root@xuegod63 ~]# systemctl stop firewalld && systemctl disable firewalld &&  
systemctl status firewalld
```

```
[root@xuegod62 ~]# systemctl stop firewalld && systemctl disable firewalld &&  
systemctl status firewalld
```

```
[root@xuegod64 ~]# systemctl stop firewalld && systemctl disable firewalld &&  
systemctl status firewalld
```

注 : 工作环境 , 也可以不开防火墙的。

到此安装已经成功。下面开始配置 kubernetes

休息一下 21 : 31 开始

11.3 配置 etcd 和 master 节点

11.3.1 修改所有结点的主机名, IP 地址和 hosts 文件

1、修改主机名 :

```
[root@xuegod63 ~]# echo master > /etc/hostname #修改主机名
```

```
[root@xuegod63 ~]# echo etcd > /etc/hostname # 我这里只有三台机器 , 不需要执行这个命令 , 如果你是 4 台 , 需要执行这个。
```

```
[root@xuegod62 ~]# echo node1 > /etc/hostname
```

```
[root@xuegod64 ~]# echo node2 > /etc/hostname
```

修改 hosts 文件 :

```
[root@xuegod63 ~]# vim /etc/hosts #插入
```

192.168.1.63 master

192.168.1.63 etcd

192.168.1.62 node1

192.168.1.64 node2

```
[root@xuegod63 ~]# scp /etc/hosts 192.168.1.62:/etc/
```

```
[root@xuegod63 ~]# scp /etc/hosts 192.168.1.64:/etc/
```

11.3.2 配置 etcd 服务器 xuegod63

```
[root@xuegod63 ~]# vim /etc/etcd/etcd.conf #修改原文件,只需要以下红色标记的内容:
```

改: 2 ETCD_NAME=default

为: 2 ETCD_NAME="etcd"

ETCD_DATA_DIR="/var/lib/etcd/default.etcd"

改: 9 ETCD_LISTEN_CLIENT_URLS="http://localhost:2379"

为: ETCD_LISTEN_CLIENT_URLS="http://localhost:2379,http://192.168.1.63:2379"

改: 20 ETCD_ADVERTISE_CLIENT_URLS="http://localhost:2379"

为: ETCD_ADVERTISE_CLIENT_URLS="http://192.168.1.63:2379"

advertise ['ædvətaɪz] 做广告,做宣传;通告,通知

/etc/etcd/etcd.conf 配置文件含意如下:

ETCD_NAME="etcd"

etcd 节点名称,如果 etcd 集群只有一台 etcd,这一项可以注释不用配置,默认名称为 default,这个名字后面会用到。

ETCD_DATA_DIR="/var/lib/etcd/default.etcd"

etcd 存储数据的目录

ETCD_LISTEN_CLIENT_URLS="http://localhost:2379,http://192.168.1.63:2379"

etcd 对外服务监听地址,一般指定 2379 端口,如果为 0.0.0.0 将会监听所有接口

ETCD_ARGS=""

需要额外添加的参数,可以自己添加,etcd 的所有参数可以通过 etcd -h 查看。

启动服务

```
[root@xuegod63 ~]#systemctl start etcd
```

```
[root@xuegod63 ~]#systemctl status etcd
```

```
[root@xuegod63 ~]#systemctl enable etcd
```

etcd 通讯使用 2379 端口

查看:

```
[root@xuegod63 ~]# netstat -antup | grep 2379
```

tcp	0	0	127.0.0.1:2379	0.0.0.0:*	LISTEN	15901/etcd
tcp	0	0	192.168.1.63:2379	0.0.0.0:*	LISTEN	15901/etcd

检查 etcd 集群成员列表,这里只有一台

```
[root@xuegod63 ~]# etcdctl member list
```

8e9e05c52164694d: name=etcd peerURLs=http://localhost:2380

clientURLs=http://192.168.1.63:2379 isLeader=true

此到 etcd 节点成功。

11.3.3 配置 master 服务器 xuegod63

1、配置 kubernetes 配置文件

```
[root@xuegod63 ~]# vim /etc/kubernetes/config
```

改：22 KUBE_MASTER="--master=http://127.0.0.1:8080"

为：22 KUBE_MASTER="--master=http://192.168.1.63:8080"

#指定 master 在 192.168.1.63 IP 上监听端口 8080

注：/etc/kubernetes/config 配置文件含意：

KUBE_LOGTOSTDERR="--logtostderr=true" #表示错误日志记录到文件还是输出到 stderr 标准错误输出。

KUBE_LOG_LEVEL="--v=0" #日志等级。

KUBE_ALLOW_PRIV="--allow-privileged=false" #是否允许运行特权容器。false 表示不允许特权容器

2、修改 apiserver 配置文件

```
[root@xuegod63 ~]# vim /etc/kubernetes/apiserver
```

改：8 KUBE_API_ADDRESS="--insecure-bind-address=127.0.0.1"

为：8 KUBE_API_ADDRESS="--insecure-bind-address=0.0.0.0"

改：17 KUBE_ETCD_SERVERS="--etcd-servers=http://127.0.0.1:2379"

为：KUBE_ETCD_SERVERS="--etcd-servers=http://192.168.1.63:2379"

改 23 行：

KUBE_ADMISSION_CONTROL="--admission-control=NamespaceLifecycle,NamespaceExists,LimitRanger,SecurityContextDeny,ServiceAccount,ResourceQuota"

为：KUBE_ADMISSION_CONTROL="--admission-control=AlwaysAdmit" #这里必须配置正确

注：/etc/kubernetes/apiserver 配置文件含意：

KUBE_API_ADDRESS="--insecure-bind-address=0.0.0.0" ## 监听的接口，如果配置为 127.0.0.1 则只监听 localhost，配置为 0.0.0.0 会监听所有接口，这里配置为 0.0.0.0。

KUBE_ETCD_SERVERS="--etcd-servers=http://192.168.1.63:2379" #etcd 服务地址，前面已经启动了 etcd 服务

KUBE_SERVICE_ADDRESSES="--service-cluster-ip-range=10.254.0.0/16" #kubernetes 可以分配的 ip 的范围，kubernetes 启动的每一个 pod 以及 service 都会分配一个 ip 地址，将从这个范围中分配 IP。

KUBE_ADMISSION_CONTROL="--admission-control=AlwaysAdmit" #不做限制，允许所有节点可以访问 apiserver，对所有请求开绿灯。

admission [əd'mɪʃn] 承认;准许进入，Admit [əd'mɪt] 承认

扩展：

admission-control (准入控制) 概述：admission controller 本质上一段代码，在对 kubernetes api 的请求过程中，顺序为 先经过认证和授权，然后执行准入操作，最后对目标对象进行操作。

3、配置 kube-controller-manager 配置文件

```
[root@xuegod63 kubernetes]# cat /etc/kubernetes/controller-manager #不需要修改
# default config should be adequate #默认的配置应该足够了
adequate ['ædɪkwət] 足够的;适当的
KUBE_CONTROLLER_MANAGER_ARGS=""
```

4、配置 kube-scheduler 配置文件

scheduler ['ʃedju:lə(r)] 调度程序

```
[root@xuegod63 ~]# vim /etc/kubernetes/scheduler
```

改：7 KUBE_SCHEDULER_ARGS=""

为：7 UBE_SCHEDULER_ARGS="0.0.0.0"

#改 scheduler 监听到的地址为：0.0.0.0，默认是 127.0.0.1。

11.3.4 配置 etcd，指定容器云中 docker 的 IP 网段，在 xuegod63 上

互动：etcd 是一个非关系型数据库。如何添加删除数据？

1、扩展：etcdctl 命令使用方法

etcdctl 是操作 etcd 非关系型数据库的一个命令行客户端，它能提供一些简洁的命令，供用户直接跟 etcd 数据库打交道

etcdctl 的命令，大体上分为数据库操作和非数据库操作两类。

数据库操作主要是围绕对键值和目录的 CRUD 完整生命周期的管理。

注：CRUD 即 Create, Read, Update, Delete。

2、etcd 在键的组织上采用了层次化的空间结构（类似于文件系统中目录的概念），用户指定的键可以为单独的名字，如 testkey，此时实际上放在根目录 / 下面，也可以为指定目录结构，如 cluster1/node2/testkey，则将创建相应的目录结构。

set 指定某个键的值。

例如：

```
[root@xuegod63 ~]# etcdctl set mk "shen"
```

shen

```
[root@xuegod63 ~]# etcdctl get mk
```

shen

或：

```
[root@xuegod63 ~]# etcdctl set /testdir/testkey "hello world"
```

hello world

get 获取指定键的值。例如

```
[root@xuegod63 ~]# etcdctl get /testdir/testkey
```

update 当键存在时，更新值内容。例如

```
[root@xuegod63 ~]# etcdctl update /testdir/testkey aaaa
```

aaaa

```
[root@xuegod63 ~]# etcdctl get /testdir/testkey
```

aaaa

rm 删除某个键值。例如

```
[root@xuegod63 ~]# etcdctl rm mk
```

或：

```
[root@xuegod63 ~]# etcdctl rm /testdir/testkey
```

etcdctl mk 和 etcdctl set 的区别如下：

etcdctl mk 如果给定的键不存在，则创建一个新的键值。如果给定的键存在，则报错，无法创建。
etcdctl set ，不管给定的键是否存在，都会创建一个新的键值。

例：

```
[root@xuegod63 ~]# etcdctl mk /testdir/testkey "Hello world"
```

Hello world

```
[root@xuegod63 ~]# etcdctl mk /testdir/testkey "bbbb"
```

Error: 105: Key already exists (/testdir/testkey) [9]

etcdctl mkdir #创建一个目录

例： [root@xuegod63 ~]# etcdctl mkdir testdir1

etcdctl ls 列出目录（默认为根目录）下的键或者子目录，默认不显示子目录中内容。

例如

```
[root@xuegod63 ~]# etcdctl ls
```

/testdir1

/testdir

```
[root@xuegod63 ~]# etcdctl ls /testdir
```

/testdir/testkey

3、非数据库操作

etcdctl member 后面可以加参数 list、add、remove 命令。表示列出、添加、删除 etcd 实例到 etcd 集群中。

例如本地启动一个 etcd 服务实例后，可以用如下命令进行查看。

```
[root@xuegod63 ~]# etcdctl member list
```

8e9e05c52164694d: name=etcd peerURLs=http://localhost:2380

clientURLs=http://192.168.1.63:2379 isLeader=true

4、设置 etcd 网络

```
[root@xuegod63 ~]# etcdctl mkdir /k8s/network #创建一个目录 /k8s/network 用于存储
```

flannel 网络信息

```
[root@xuegod63 ~]# etcdctl set /k8s/network/config '{"Network": "10.255.0.0/16"}'
```

```
#给/k8s/network/config 赋一个字符串的值 '{"Network": "10.255.0.0/16"}'
```

```
[root@xuegod63 ~]# etcdctl get /k8s/network/config #查看
```

注：在启动 flannel 之前，需要在 etcd 中添加一条网络配置记录，这个配置将用于 flannel 分配给每个 docker 的虚拟 IP 地址段。用于配置在 minion 上 docker 的 IP 地址。

由于 flannel 将覆盖 docker0 上的地址，所以 flannel 服务要先于 docker 服务启动。如果 docker 服务已经启动，则先停止 docker 服务，然后启动 flannel，再启动 docker

5、flannel 启动过程解析:

- (1)、从 etcd 中获取出/k8s/network/config 的值
- (2)、划分 subnet 子网，并在 etcd 中进行注册
- (3)、将子网信息记录到/run/flannel/subnet.env 中

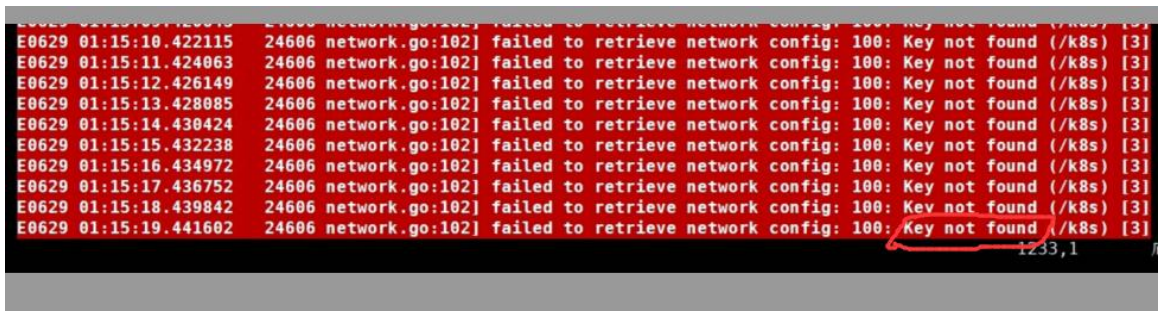
6、配置一下 flanneld 服务：

```
[root@xuegod63 ~]# vim /etc/sysconfig/flanneld
改：4 FLANNEL_ETCD_ENDPOINTS="http://127.0.0.1:2379"
为：4 FLANNEL_ETCD_ENDPOINTS="http://192.168.1.63:2379"
改：8 FLANNEL_ETCD_PREFIX="/atomic.io/network"
为：8 FLANNEL_ETCD_PREFIX="/k8s/network"
#注其中/k8s/network 与上面 etcd 中的 Network 对应
改：11 #FLANNEL_OPTIONS=""
为：11 FLANNEL_OPTIONS="--iface=ens33"    #指定    通信的物理网卡
```

```
[root@xuegod63 ~]# systemctl restart flanneld
```

```
[root@xuegod63 ~]# systemctl status flanneld
```

报错总结 key not found /k8s 原因是设置 etcd 网络时,命令 etcdctl mkdir /k8s/network 没有和 flannel 配置文件中 FLANNEL_ETCD_PREFIX="/k8s/network" 不一样。所以找不到。改成一致就可以了



查看：

```
[root@master ~]# ifconfig
```

...

```
flannel0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1472
    inet 10.255.48.0 netmask 255.255.0.0 destination 10.255.48.0
```

7、查看子网信息/run/flannel/subnet.env，等 flannel 起来后就可以看到了，现在看不到此文件

```
[root@master ~]# cat /run/flannel/subnet.env
```

```
FLANNEL_NETWORK=10.255.0.0/16
```

```
FLANNEL_SUBNET=10.255.70.1/24
```

```
FLANNEL_MTU=1472
```

```
FLANNEL_IPMASQ=false
```

之后将会有一个脚本将 subnet.env 转写成一个 docker 的环境变量文件/run/flannel/docker。

docker0 的地址是由 /run/flannel/subnet.env 的 FLANNEL_SUBNET 参数决定的。

```
[root@master ~]# cat /run/flannel/docker
```

```
DOCKER_OPT_BIP="--bip=10.255.70.1/24"
```



```
DOCKER_OPT_IPMASQ="--ip-masq=true"
```

```
DOCKER_OPT_MTU="--mtu=1472"
```

```
DOCKER_NETWORK_OPTIONS="--bip=10.255.70.1/24 --ip-masq=true --mtu=1472"
```

4) 启动 master 上 4 个服务：

```
[root@xuegod63 ~]# systemctl restart kube-apiserver kube-controller-manager  
kube-scheduler flanneld
```

```
[root@xuegod63 ~]# systemctl status kube-apiserver kube-controller-manager  
kube-scheduler flanneld
```

```
[root@xuegod63 ~]# systemctl enable kube-apiserver kube-controller-manager  
kube-scheduler flanneld
```

或使用下面的脚本：

```
for SERVICES in kube-apiserver kube-controller-manager kube-scheduler flanneld  
do
```

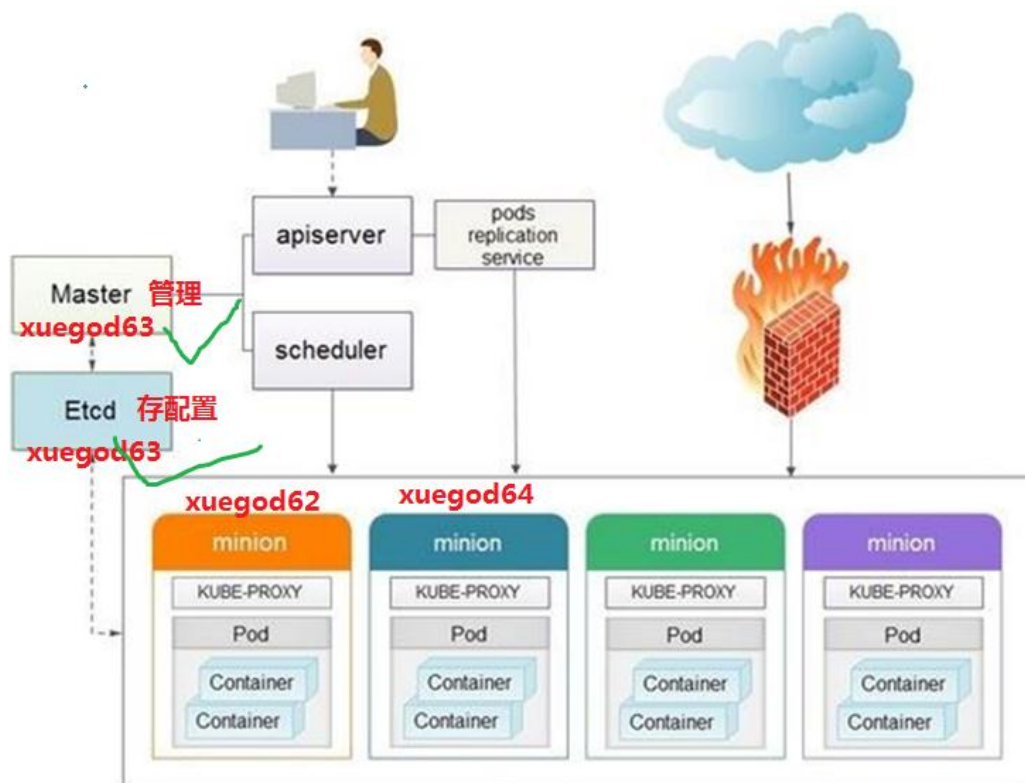
```
    systemctl restart SERVICES
```

```
    systemctl status SERVICES
```

```
    systemctl enable SERVICES
```

```
done
```

此到 etcd 和 master 节点成功。



11.4 配置 minion1 节点服务器

11.4.1 配置 node1 网络，本实例采用 flannel 方式来配置

```
[root@xuegod62 ~]# hostname node1
```

配置一下 flanneld 服务：

```
[root@xuegod62 ~]# vim /etc/sysconfig/flanneld
```

改：4 FLANNEL_ETCD_ENDPOINTS="http://127.0.0.1:2379"

为：4 FLANNEL_ETCD_ENDPOINTS="http://192.168.1.63:2379"

改：8 FLANNEL_ETCD_PREFIX="/atomic.io/network"

为：8 FLANNEL_ETCD_PREFIX="/k8s/network"

#注其中/k8s/network 与上面 etcd 中的 Network 对应

改：11 #FLANNEL_OPTIONS=""

为：11 FLANNEL_OPTIONS="--iface=ens33" #指定 通信的物理网卡

11.4.2 配置 node1 上的 master 地址和 kube-proxy

1、配置 node1 上的 master 地址

```
[root@xuegod62 ~]# vim /etc/kubernetes/config
```

改：22 KUBE_MASTER="--master=http://127.0.0.1:8080"

为：22 KUBE_MASTER="--master=http://192.168.1.63:8080"

2、kube-proxy 的作用主要是负责 service 的实现，具体来说，就是实现了内部从 pod 到 service。

```
[root@node1 ~]# grep -v '^#' /etc/kubernetes/proxy
```

KUBE_PROXY_ARGS=""

#不用修改，默认就是监听所有 ip

注：如果启动服务失败，可以使用 tail -f /var/log/messages 动态查看日志

11.4.3 配置 node1 kubelet

Kubelet 运行在 minion 节点上。Kubelet 组件管理 Pod、Pod 中容器及容器的镜像和卷等信息。

```
[root@node1 ~]# vim /etc/kubernetes/kubelet
```

改：5 KUBELET_ADDRESS="--address=127.0.0.1"

为：5 KUBELET_ADDRESS="--address=0.0.0.0" #默认只监听 127.0.0.1,要改成 :0.0.0.0 ,

因为后期要使用 **kubectl** 远程连接到 kubelet 服务上，来查看 pod 及 pod 中容器的状态。如果是 127 就无法远程连接 kubelet 服务。

改：11 KUBELET_HOSTNAME="--hostname-override=127.0.0.1"

为：11 KUBELET_HOSTNAME="--hostname-override=node1" # minion 的主机名，设置成和本主机名一样，便于识别。

改：14 KUBELET_API_SERVER="--api-servers=http://127.0.0.1:8080"

为：14 KUBELET_API_SERVER="--api-servers=http://192.168.1.63:8080" #指定 apiserver 的地址

扩展：第 17 行的意思：

KUBELET_POD_INFRA_CONTAINER="--pod-infra-container-image=registry.access.redhat.com/rhel7/pod-infrastructure:latest"

注：INFRA infrastructure [ˈɪnfəˈstrʌktʃə(r)] 基础设施

KUBELET_POD_INFRA_CONTAINER 指定 pod 基础容器镜像地址。这个是一个基础容器，每一个

Pod 启动的时候都会启动一个这样的容器。如果你的本地没有这个镜像，kubelet 会连接外网把这个镜像下载下来。最开始的时候是在 Google 的 registry 上，因此国内因为 GFW 都下载不了导致 Pod 运行不起来。现在每个版本的 Kubernetes 都把这个镜像地址改成红帽的地址了。你也可以提前传到自己的 registry 上，然后再用这个参数指定成自己的镜像链接。

注：<https://access.redhat.com/containers/> 是红帽的容器下载站点

11.4.4 启动 node1 服务

```
[root@node1 ~]# systemctl restart flanneld kube-proxy kubelet docker
[root@node1 ~]# systemctl enable flanneld kube-proxy kubelet docker
[root@node1 ~]# systemctl status flanneld kube-proxy kubelet docker
或：
```

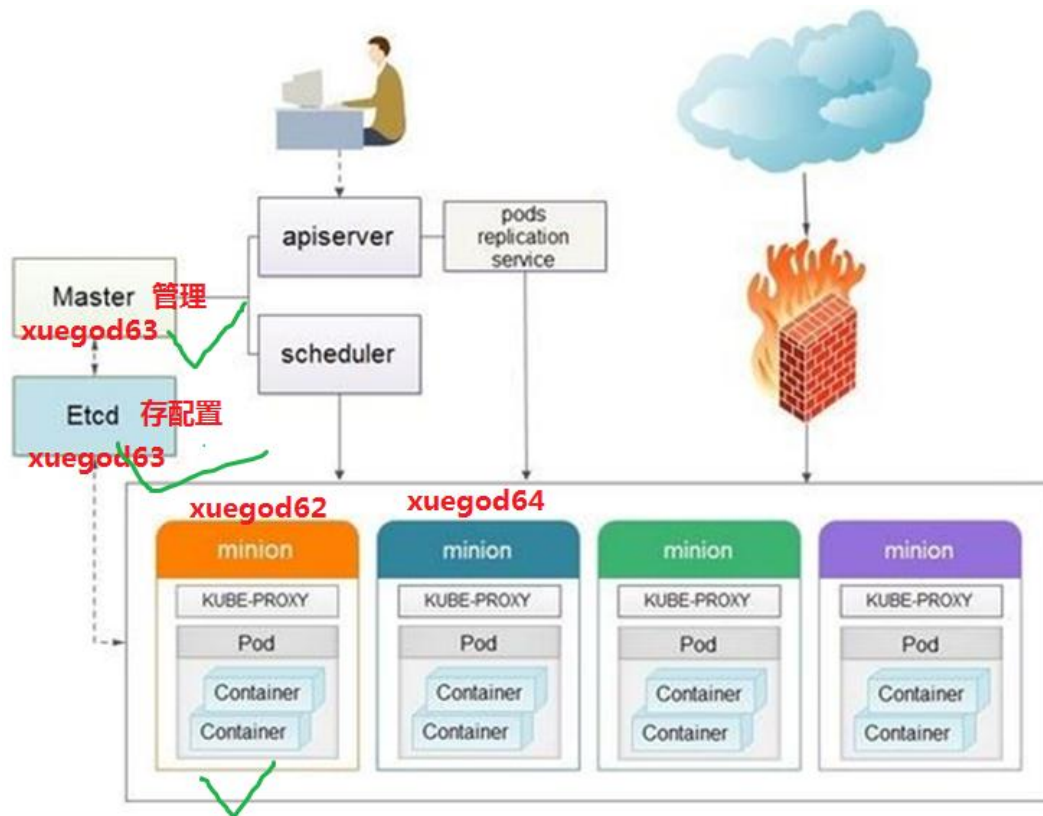
```
for SERVICES in flanneld kube-proxy kubelet docker ; do
    systemctl restart $SERVICES
    systemctl enable $SERVICES
    systemctl status $SERVICES
done
```

查看：

```
[root@node1 ~]# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 10.255.20.1 netmask 255.255.255.0 broadcast 0.0.0.0
    . . .
flannel0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1472
    inet 10.255.20.0 netmask 255.255.0.0 destination 10.255.41.0
```

查看：kube-proxy

```
[root@xuegod62 ~]# netstat -antup | grep proxy
tcp    0      0 127.0.0.1:10249      0.0.0.0:*           LISTEN      51996/kubeproxy
tcp    0      0 192.168.1.62:51593   192.168.1.63:8080    ESTABLISHED 51996/kubeproxy
tcp    0      0 192.168.1.62:51594   192.168.1.63:8080    ESTABLISHED 51996/kubeproxy
到此 node1 minion 节点 成了。
```



11.5 配置 minion2 结点并总结 Kubernetes 所有服务和端口号

11.5.1 配置 node2 网络，本实例采用 flannel 方式来配置

配置方法和 node1 一样

[root@**xuegod62** ~]# scp /etc/sysconfig/flanneld 192.168.1.64:/etc/sysconfig/ #直接复制 xuegod62 的配置到 xuegod64

查看：

```
[root@node2 ~]# grep -v '^#' /etc/sysconfig/flanneld
FLANNEL_ETCD_ENDPOINTS="http://192.168.1.63:2379"
FLANNEL_ETCD_PREFIX="/k8s/network"
FLANNEL_OPTIONS="--iface=ens33"
```

```
[root@node2 ~]# systemctl start flanneld
```

查看：

```
[root@node2 ~]# ifconfig
...
flannel0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1472
    inet 10.255.24.0 netmask 255.255.0.0 destination 10.255.42.0
```

11.5.2 配置 node2 上 master 地址和 kube-proxy

```
[root@xuegod62 ~]# scp /etc/kubernetes/config 192.168.1.64:/etc/kubernetes/
```

```
[root@xuegod62 ~]# scp /etc/kubernetes/proxy 192.168.1.64:/etc/kubernetes/
```

```
[root@node2 ~]# systemctl start kube-proxy
```

测试：

```
[root@node2 ~]# netstat -antup | grep proxy
```

```
tcp        0      0 127.0.0.1:10249        0.0.0.0:*              LISTEN      52337/kubeproxy
```

注：kubeproxy 监控监听端口号是 10249

```
tcp        0      0 192.168.1.64:46299     192.168.1.63:8080      ESTABLISHED
```

52337/kubeproxy

注：kubeproxy 和 master 的 8080 进行通信

```
tcp        0      0 192.168.1.64:46301     192.168.1.63:8080      ESTABLISHED
```

52337/kubeproxy

```
tcp        0      0 192.168.1.64:46300     192.168.1.63:8080      ESTABLISHED
```

52337/kubeproxy

11.5.3 配置 node2 kubelet

```
[root@xuegod62 ~]# scp /etc/kubernetes/kubelet 192.168.1.64:/etc/kubernetes/
```

```
[root@node2 ~]# vim /etc/kubernetes/kubelet
```

改：11 KUBELET_HOSTNAME="--hostname-override=node1"

为：11 KUBELET_HOSTNAME="--hostname-override=node2" #这里修改成 node2

```
[root@node2 ~]# systemctl start kubelet
```

查看：已经建立连接

```
[root@node2 ~]# netstat -antup | grep 8080
```

11.5.4 启动 node2 服务所有服务

```
[root@node2 ~]# systemctl restart flanneld kube-proxy kubelet docker
```

```
[root@node2 ~]# systemctl enable flanneld kube-proxy kubelet docker
```

```
[root@node2 ~]# systemctl status flanneld kube-proxy kubelet docker
```

或：

```
for SERVICES in flanneld kube-proxy kubelet docker; do
```

```
    systemctl restart $SERVICES
```

```
    systemctl enable $SERVICES
```

```
    systemctl status $SERVICES
```

```
done
```

查看 docker0 IP：node1 和 node2 是不一样

```
[root@xuegod64 ~]# ifconfig
```

```
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
```

```
    inet 10.255.24.1 netmask 255.255.255.0 broadcast 0.0.0.0
```

...

```
flannel0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1472
```

inet 10.255.24.0 netmask 255.255.0.0 destination 10.255.24.0

测试：登录 xuegod63 查看整个集群的运行状态：

```
[root@master ~]# kubectl get nodes
```

NAME	STATUS	AGE
node1	Ready	41m
node2	Ready	3m

说明运行正常。

至此，整个 Kubernetes 集群搭建完毕

11.5.5 kubernetes 每个节点需要启动的服务和开放端口号

在本实验中 kubernetes 4 个结点一共需要启动 13 个服务，开 6 个端口号。

详情如下：

etcd: 一共 1 个服务，通讯使用 2379 端口

启动服务

```
[root@xuegod63 ~]# systemctl restart etcd
```

master: 一共 4 个服务，通讯使用 8080 端口

```
[root@xuegod63 ~]# systemctl restart kube-apiserver kube-controller-manager  
kube-scheduler flanneld
```

node1-minion: 一共 4 个服务

kubeproxy 监听端口号是 10249，kubelet 监听端口 10248、10250、10255 三个端口

```
[root@node1 ~]# systemctl restart flanneld kube-proxy kubelet docker
```

node2-minion: 一共 4 个服务

```
[root@node2 ~]# systemctl restart flanneld kube-proxy kubelet docker
```

总结：

11.1 Kubernetes 和相关组件的介绍

11.2 配置 yum 源安装 kubernetes 及组件

11.3 配置 etcd 和 master 节点

11.4 配置 minion1 节点

11.5 配置 minion2 结点并总结 Kubernetes 所有服务和端口号