K8s是一个能够实现自动发现，自动装箱，水平扩展，服务发现和负载均衡，自动发布和回滚，密钥和配置管理,存储编排、批处理执行等功能的容器管理工具.

\*Master/Node:

\*master: API Server,Scheduler,Controller-Manager

\*node: kubelet,docker,kube-proxy

Pod

\*Label: key-value

\*Label Selector:

1. API Server:用于在Master上接受各种请求处理的进程.
2. Scheduler:调度器，用于选择哪个节点运行管理容器.
3. Controller-Manager: 控制器管理器，这个是用于监控Master上的控制器是否正常的，这里因为一般Master是高可用的方式，因此可以进行互相切换冗余
4. kubelet: Node上用于启动管理pod的工具，用于监控Node是否正常的，而Node上由kubelet来监控容器的，那么一旦容器挂掉之后，这里会向Master上的API Server发起请求，再由调度器来进行重新调度.可以理解为控制器.
5. kube-proxy:用于和API Server通信的，这里每个Node上pod有变动的时候需要通知Service，那么就使用该组件进行相关通信.
6. kubelet:运行在cluster所有节点上,负责启动POD和容器
7. Kubeadm: 用于初始化cluster
8. Kubectl: kubernetes的命令行客户端工具，用于查看、修改和创建各种资源配置等.

这里kubernetes其实是一种master和node架构的容器管理工具，这里主要有master和node两种类型节点,其中master上面会有API Server，因此工具是Resful风格的工具，因此接受各种请求需要用.接着因为k8s上的最小运行单元是pod，这里pod其实是对容器的封装的一个称呼，一般来说就是一个pod里面就一个容器，当然也有一些特殊的情况会需要里面有副本容器等情况.

那么如果pod挂掉之后，该如何重启管理，那就需要用到Scheduler和kubelet，这里其实就是管理好pod的运行状态并且重新调度.

另外这里管理pod的时候，如果多个pod要放在一起，或者当前的pod挂掉重启之后，那么该如何在网络上识别呢？那么这里就使用Label标签来进行识别了，例如对nginx的多个pod进行归类等.

那么这里pod是封装了docker,而docker是使用了Linux的namespace来实现命名空间隔离的，关于这部分内容，可以参考下面的连接

<https://zhuanlan.zhihu.com/p/47571649>

关于Linux namespace隔离的原理.

\*Pod:

\*自主式Pod

\*控制器管理的Pod

\*ReplicationController

\*ReplicaSet

\*Deployment

\*StatefulSet

\*Job,Ctonjob

但是这里pod其实主要分为两种的，一种就是调度器启动的自主式Pod,另外一种就是控制器管理的Pod,其中所谓的控制器管理，也就是各种特定情况下的管理方式.例如Deployment适合做水平扩展用的，当流量大了之后，这里就会进行水平扩展,其实这里就是服务发现的内容.那么这里关于HPA(Horizontal Pod Autoscaler)的内容可以见如下参考资料所示.

客户端

DNS

Service

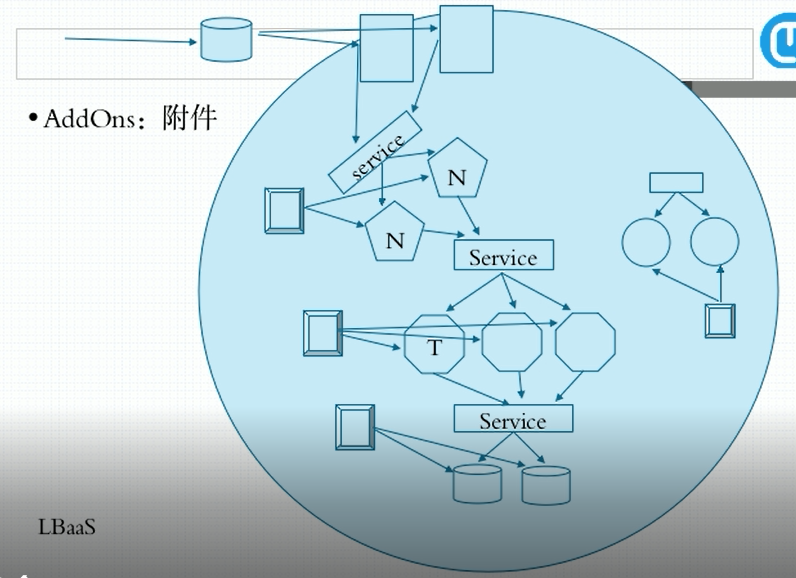
这里其实就是固定有一个服务发现的中间层，然后这里就是对外公布ip地址的，然后下面有各种类型的pod，当pod启动的时候，这里会根据Label标签来进行分类注册，那么当客户端发起请求的时候，就会去注册表里面获取到对应的信息了.

当然这里还可以使用名称的方式来进行获取service，那么这里就需要一个DNS服务来进行发现了，而一旦修改了service的ip，那么关联的dns服务器的记录也就需要跟着改变了.

<https://www.jianshu.com/p/4828f90674b3>

服务发现的参考资料.

那么有了服务发现这个东西之后，k8s集群其实架构来说和以前是很不一样的.



这里以常见的NMT架构为例，也就是nginx+tomat+mysql,这里对于k8s集群来说，每一层上面都应该有对应的service组件来进行负载均衡调度,然后这里对外提供服务的时候，其实是用节点的物理地址来进行代理的，但是有可能物理节点也是内网地址，那么这里最外层就需要有一个调度器的，但是这个最外层的调度器就不再是集群控制的.

还有这里每一层的Service还有对应的pod控制器，因为Service只是负责调度的，但是如何监控和自动创建pod就是由pod控制器来负责的.而这里的创建与销毁，就是通过标签来实现的.

右侧这里的pod其实是dns服务，这里也是需要service的.

另外这里还有一种概念是LBaaS,就是在云平台上，如目前常见的阿里云和亚马逊等平台上面，是可以通过调用底层的API，把各种不同的Service暴露出去的.

那么 这里解决了架构问题之后，接着就是考虑网络问题了，因为根据前面的内容可以知道，多层Service下，如果这里采用docker的这种访问通信方式，也就是两个pod之间通信的话，采用物理机之间的端口进行映射，做二级代理.

但是这种方式在k8s集群里面是很难实现的，因为有可能不同的pod的启停都是在不同的节点上面，那么这里做网络就会很复杂了，同时这里还有可能会出现广播风暴等问题.

下面考虑一下k8s集群之间的通信问题:

1. 同一个Pod内的多个容器间:

这种方式就很简单了，直接采用lo通信即可.

1. 各Pod之间的通信:

那么这里k8s集群里面有三种网络，一个是物理机节点的网络，一个是集群网络，也就是service这些的网络，一个就是pod的网络.

这里为了高效，k8s集群采用的是vlan的方式进行通信的，关于这方面的资料可以参考相关连接.

<https://baike.baidu.com/item/%E8%99%9A%E6%8B%9F%E5%B1%80%E5%9F%9F%E7%BD%91/419962?fromtitle=VLAN&fromid=320429&fr=aladdin>

百度百科关于vlan的内容.

那么这里所谓的VLan网络，其实就是根据交换机的端口来进行划分区域的，当然这里可以有几种常见的划分方式.

这里考虑一下为什么不使用iptables转发规则讷？因为这样一旦pod变动的话，那么需要调整的内容就太多了，因此一般不会对pod之间的通信使用宿主机上的转发规则.

1. Pod与Service之间的通信:

这里一旦生成了service之后，那么对应的Node上就会设置相关的iptables或者ipvs转发规则，那么这里就有点类似下面的方式了.

1. etcd的概念

这里因为k8s的Master是高可用的，那么一般高可用的话，元数据信息都不是单独存在某个节点上的内存或者磁盘上面的，而是使用共享存储的机制来实现的.而这个东西在k8s中被称为etcd.

Node

Master

etcd

etcd

etcd

这里有个问题就是，因为不同的部分通信的话，如果采用https，那么这里是需要几套不同的CA来进行签证的.

那么这里对于k8s集群来说，是通过第三方插件来提供这些网络管理功能的，也就是CNI组件.那么常见的有以下几种:

CNI:

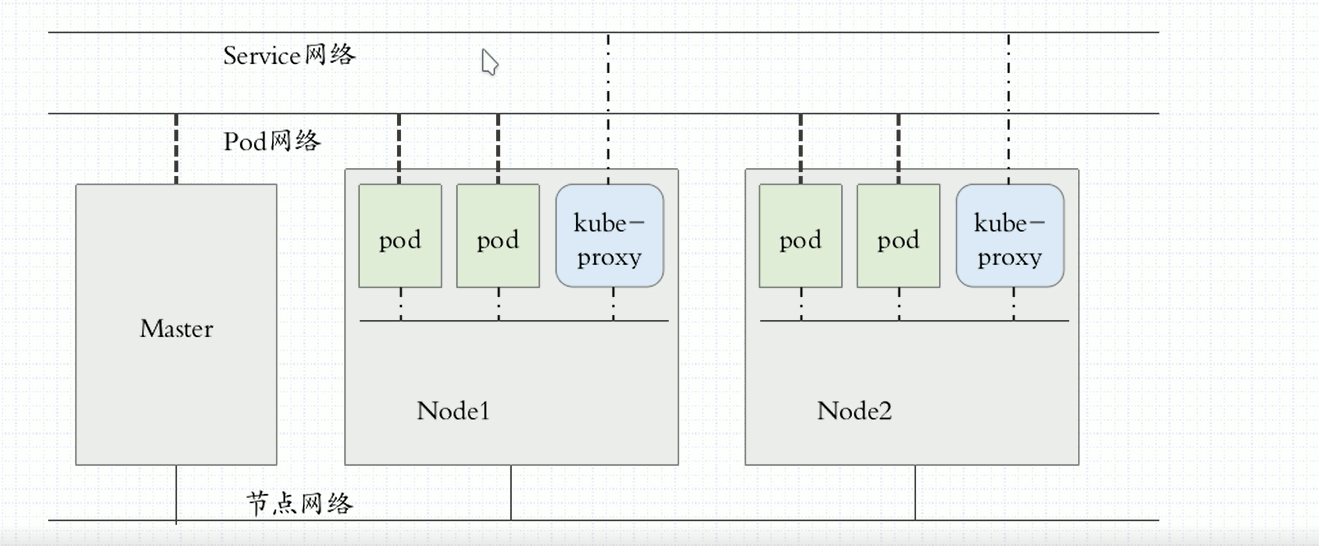
Flannel: 网络配置

Calico:网络配置、网络策略

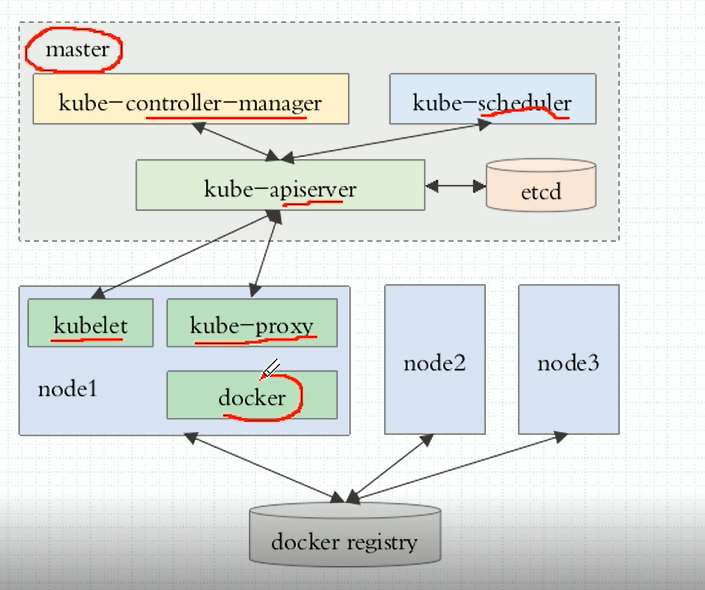
Canel:结合了上面两种的特性的插件.

...

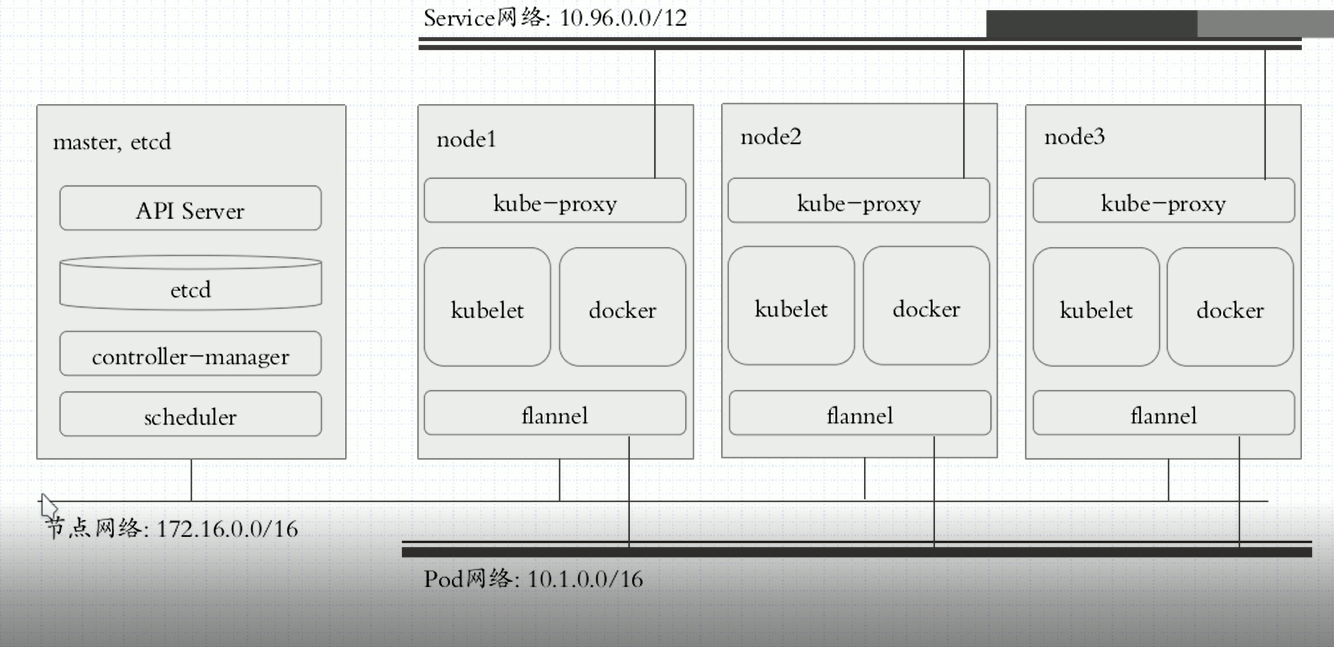
这里解释一下，网络管理一般有两个维度，分别是网络配置和网络策略，网络配置很好理解，就是配置集群的网络，这里节点的网络一般是服务器管理员来配置好的，例如免密钥什么的，而网络策略就是为了多租户的时候使用，那么这里进行隔离所做的操作.



下面这张是架构图,总结上面内容的.



下面这张也是关于网络方面的架构图



那么这里就产生一个问题就是，如果手动设置安装的话，这里大批量的集群部署需要非常多的配置，光是网络就需要部署多套CA证书，因此这里为例快速部署，k8s提供了一个叫做kubeadm的工具用来管理的.

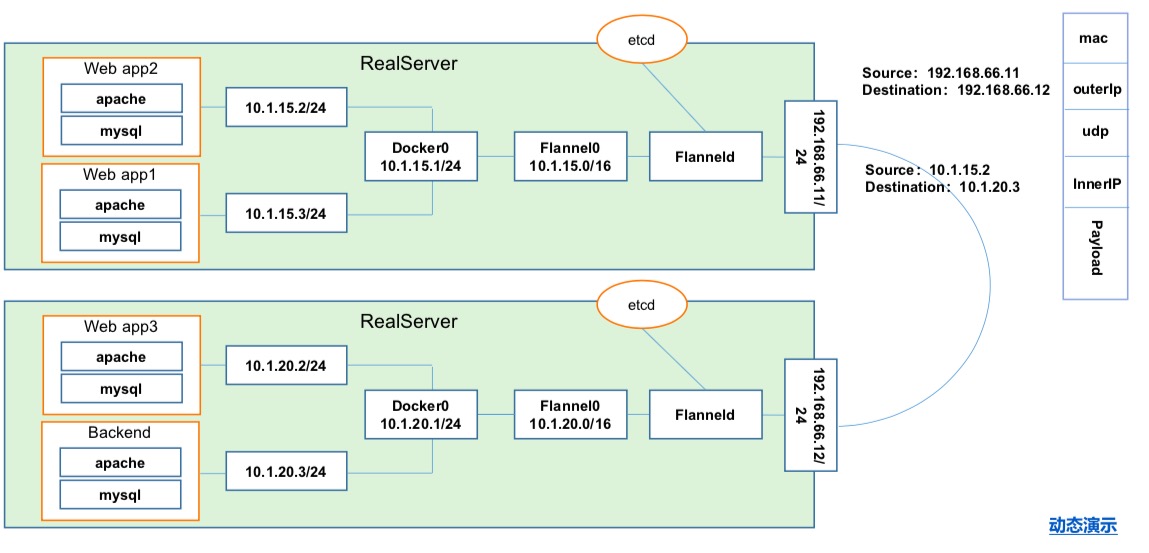
<https://github.com/kubernetes/kubeadm/>

github上的源码地址.

这里有个概念，叫做static pod的，其实就是一开始部署的时候，肯定每个节点都要安装docker和kubelet，那么这样就具备了k8s集群的部署环境，然后再根据kubelet来启停pod，但是这里kubelet如果使用pod来启动的话，其实是不受k8s集群托管的，也就是没有自托管的功能.也就是这些pod都是手动启停的，那么因此也有了一种就是在k8s集群中启动k8s的方式，但是其实本质上最外层的环境初始化

<https://github.com/kubernetes/kubeadm/blob/master/docs/design/design_v1.10.md>

github上关于kubeadm的说明文档.



最后补充一张关于网络的理解，这里service就是一个iptables转发规则.