Kubernetes(k8s)

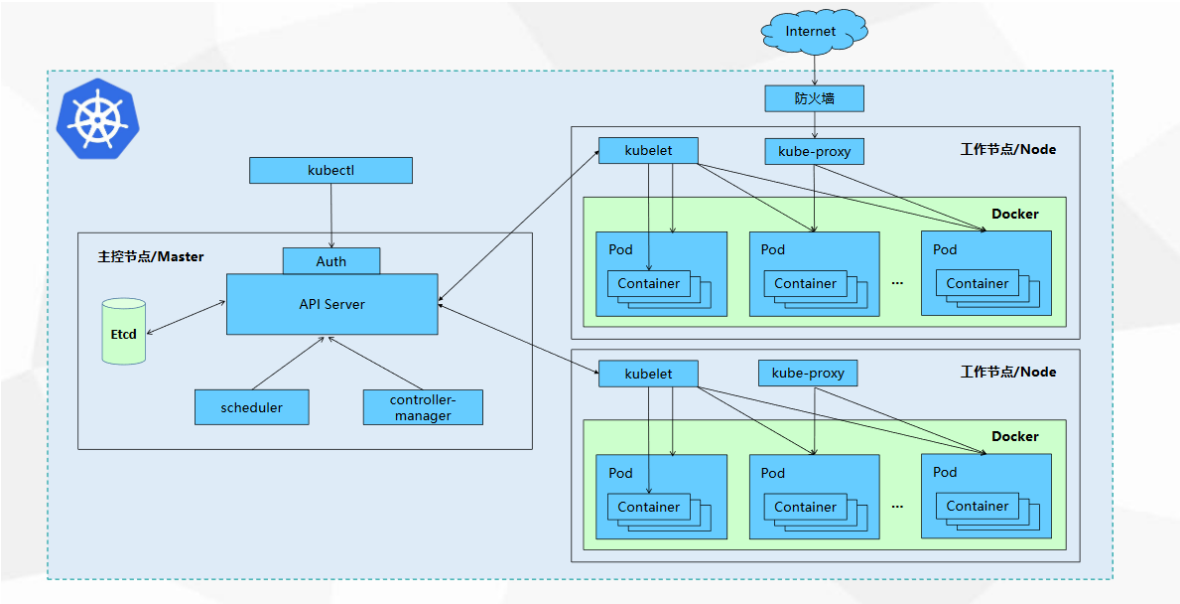
概括

- google开源
- 用于容器化应用程序的部署、扩展和管理
- 提供容器编排、资源调度、弹性伸缩、部署管理、服务发现等一系列功能
- 使得部署容器化应用简单高效

特点

- **自我修复**：在节点故障时重新启动失败的容器，替换和重新部署，保证预期的副本数量；杀死健康检查失败的容器，并且在未准备好之前不会处理客户端要求，确保线上服务不中断。
- **弹性伸缩**：使用命令、UI、或者基于CPU使用情况自动快速扩容和缩容应用程序实例，保证应用业务高峰并发时的高可用性；业务低峰时回收资源，以最小成本运行服务。
- **自动部署和回滚**：K8s采用滚动更新策略更新应用，一次更新一个Pod，而不是同时删除所有的Pod，如果更新过程中出现问题，将回滚更改，确保升级不影响业务。
- **服务器发现和负载均衡**：K8s为多个容器提供一个统一访问入口（内部IP地址和一个DNS名称），并且负载均衡关联的所有容器，使得用户无需要考虑容器的Ip问题。
- **机密和配置管理**：管理机密数据和应用程序配置，而不需要把敏感数据暴露在镜像里，提高铭感数据安全性。并可以将一些常用的配置存储在K8s中，方便应用程序使用。
- **存储编排**：挂载外部存储系统，无论是来自本地存储，公有云，还是网络存储都作为集群资源的一部分使用，极大的提高存储使用灵活性。
- **批处理**： 提供一次性任务，定时任务，满足批量数据处理和分析场景

K8s集群架构与组件



> Master组件
>
> - **kube-apiserver**: K8s API，集群的统一入口，各组件的协调者，以RESTful API提供接口服务，所有对象资源的增删改查和监听操作都交给APIServer处理后再给**Etcd**存储。

> - **kube-controller-manage**: 处理集群中常规后台任务，一个资源对应一个控制器，然而ControllerManage就是负责管理这些控制器的。
> - **kube-scheduler**: 根据调度算法为新创建的Pod选择一个Node节点，可以任意部署，可以部署在同一个节点上，也可以部署在不同的节点上。
> - **etcd**: 分布式键值存储系统。用于保存集群状态数据，比如Pod、Service等对象信息
>
> Node组件
>
> - **kubelet**: 是Master在Node节点上的Agent(代理),管理本机运行容器的生命周期，比如创建容器，Pod挂载数据卷、下载secret、获取容器和节点状态等工作。kubelet将每个Pod转换成一组容器。
> - **kube-proxy**: 在Node节点上实现Pod网络代理，维护网络规则和四层负载均衡工作
> - **docker**：容器引擎，运行容器。

K8s核心概念

Pod

- 最小部署单元
- 一组容器集合
- 一个Pod中的容器共享网络命名空间(网络是共享的)
- Pod是短暂的

Controller

- ReplicaSet: 确保预期的Pod副本数量
- Deployment: 无状态应用部署 (一般用这个来部署应用，创建一个ReplicaSet来创建。无状态： 表示不考虑外部任何情况，比如存储或者网络Id等)
- StatefulSet:有状态应用部署
- DaemonSet：确保所有Node运行同一个Pod，
- Job：一次性任务
- Cronjob:定时任务

Service

- 防止Pod失联
- 定义一组Pod的访问策略

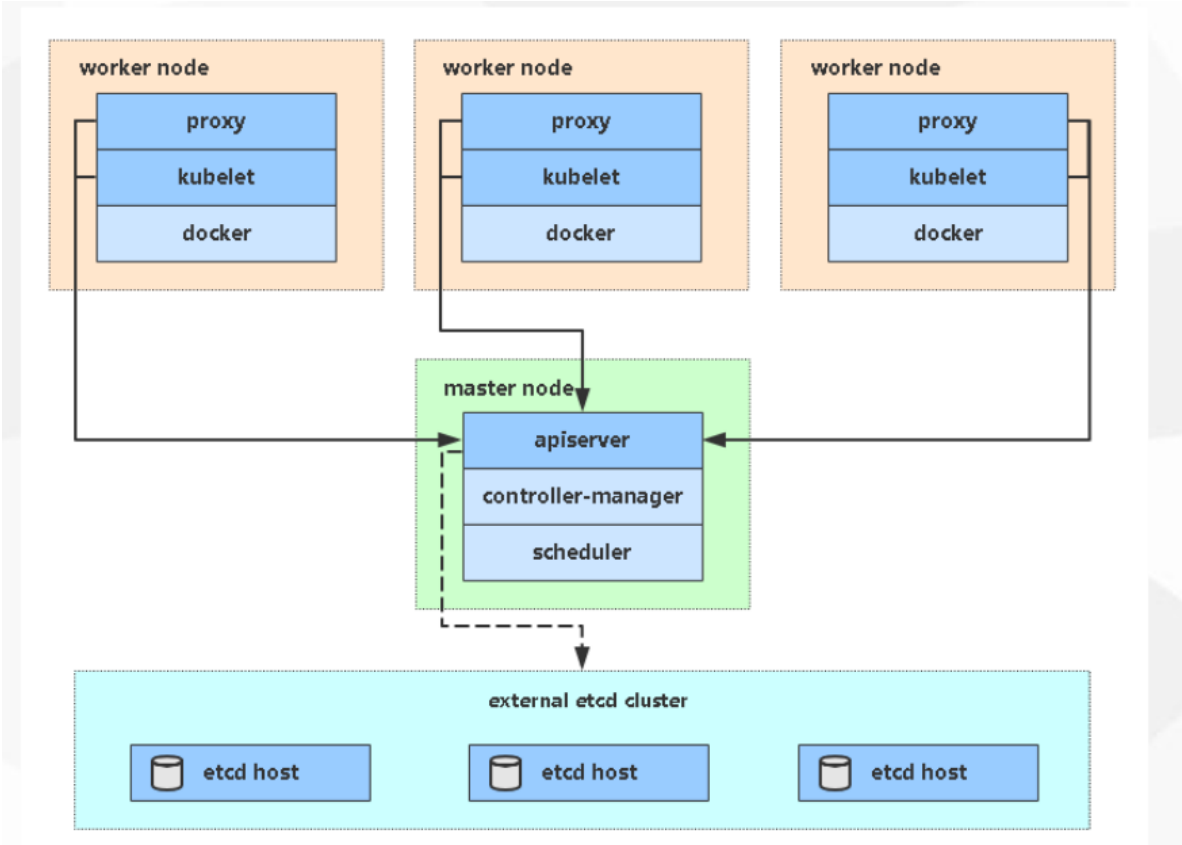
通过Controller部署应用，他会创建许多Pod，然后用Service暴露地址让别人访问

Label： 标签，附加到某个资源上，用于关联对象，查询和筛选

Namespace: 命名空间，将对象逻辑上隔离。这是一个逻辑的概念

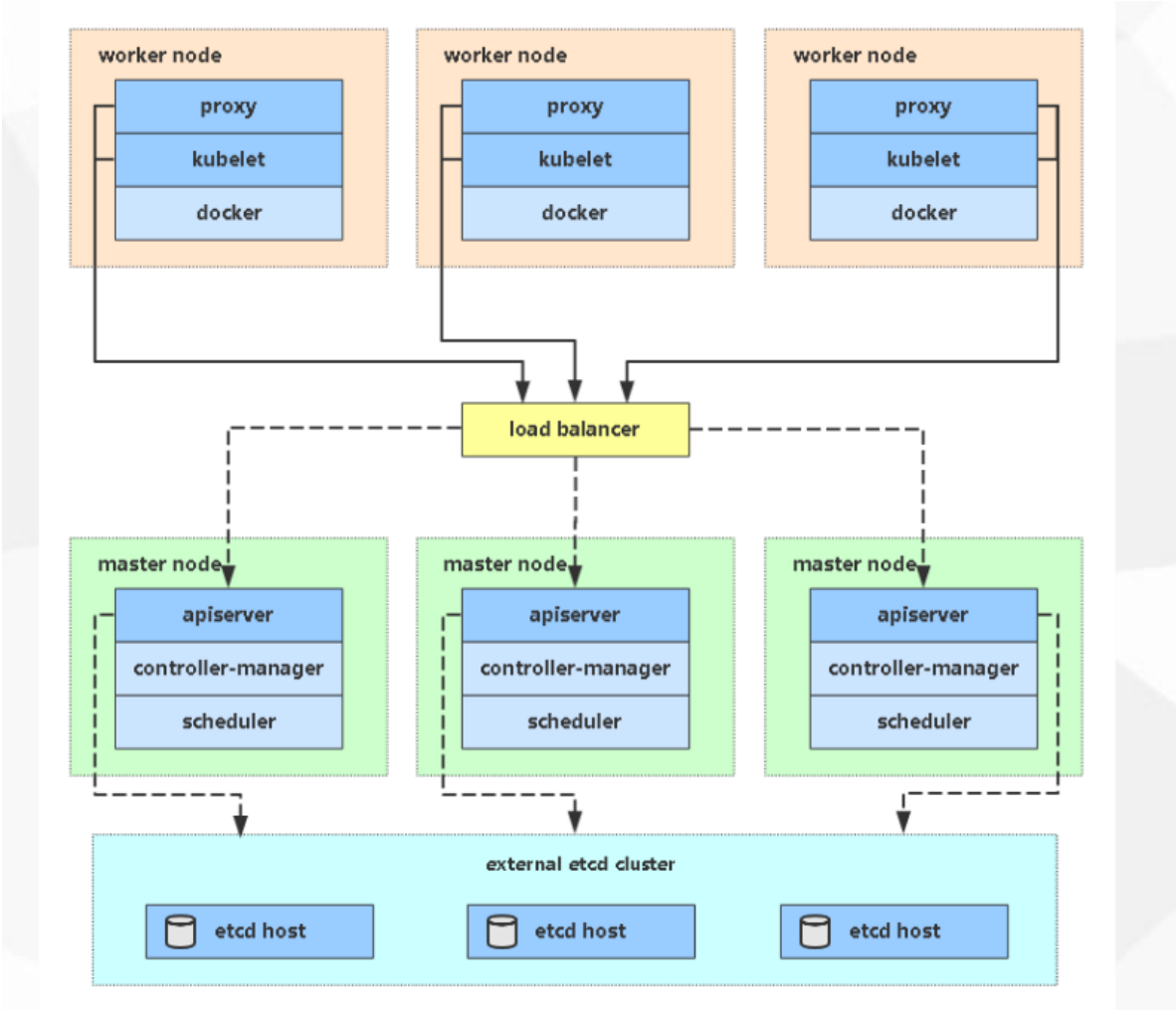
搭建一个K8s集群

单Master集群



> 可能出现单点故障，因为Master节点只有一个。etcd是独立的数据库，本质和K8s没有关系，可以部署到任何地方。master Node 对worker node 整体对资源调度，弹性伸缩，部署管理等。worker node 运行容器化的地方。etcd集群本身是高可用的
>
> 一般 三台，一个master 2个node

多Master集群,高可用



> 多Master集群，避免了单点故障，需要在worker node和master node 之间 加一个负载均衡（load balancer）。需要的机器比单master多三之一左右
>
> 多master 一般需要：6台，2master，2 node，2台负载均衡(主备)。一般node跑具体任务，配置要比master高，内存4倍左右，

在生产环境中决不允许出现单点故障，故必须要做集群

平台的规划：

角色	IP	组件
k8s-master1	192.168.31.63	kube-apiserver kube-controller-manager kube-scheduler etcd
k8s-master2	192.168.31.64	kube-apiserver kube-controller-manager kube-scheduler etcd
k8s-node1	192.168.31.65	kubelet kube-proxy docker etcd
k8s-node2	192.168.31.66	kubelet kube-proxy docker
Load Balancer（Master）	192.168.31.61 192.168.31.60 (VIP)	Nginx L4
Load Balancer（Backup）	192.168.31.62	Nginx L4

硬件配置推荐:

实验环境	k8s master/node	2C2G+	
测试环境	k8s-master	CPU	2核
		内存	4G
		硬盘	20G
	k8s-node	CPU	4核
		内存	8G
		硬盘	20G
生产环境	k8s-master	CPU	8核
		内存	16G
		硬盘	100G
	k8s-node	CPU	16核
		内存	64G
		硬盘	500G

搭建集群配置原则：预留30%的空闲资源防止内存突发，机器被废掉

部署K8s的方式

- **minikube**： 学习使用，不具备生产环境能力。
- **kubeadm**： 快速部署k8s集群。为了解决二进制部署的复杂性而创建，比较成熟的工具，简单易用。 https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm/
- **二进制**： 有点复杂，方便于后期维护。 https://github.com/kubernetes/kubernetes/releases

> 初学建议二进制，能够知其然，开发中使用 kubeadm

自签Etcd SSL 证书

在任意一个机器上，将TLS.tar.gz压缩包上传

cfssl工具