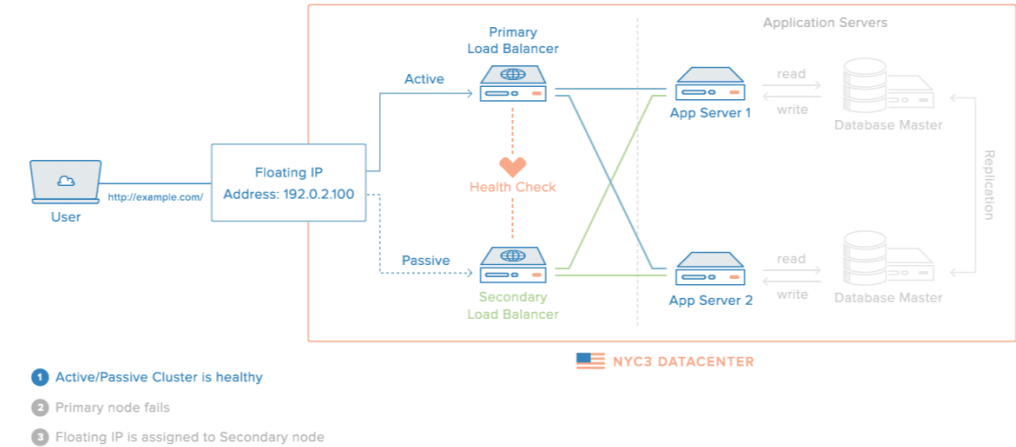


[TOC]

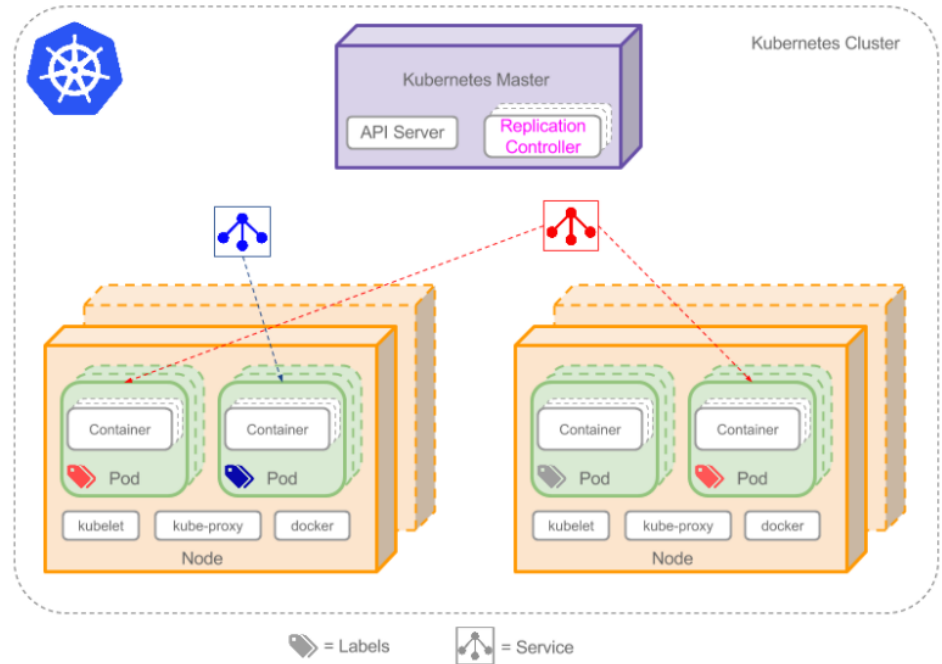
Kubernetes

什么是Kubernetes?

- 自动化容器操作的开源平台，这些操作包括部署，调度和节点集群间扩展。
 - 基于容器技术的分布式架构, 来源于Borg。
 - 开放的开发平台，不局限于任何一种语言，没有限定任何编程接口
 - 通过标准的TCP通信协议进行交互。（HTTP请求和TCP连接）
 - 完备的分布式系统支撑平台
- 基础知识
 - Service（服务）是分布式集群架构的核心 起干同样的事叫集群，分工合作的叫分布式
 - 特征：
 1. 唯一指定的名字（mysql-server）
 2. 拥有一个虚拟的IP（Cluster IP、Service IP或VIP）和端口号
 3. 能够提供某种远程服务能力
 4. 被映射到了提供这种服务能力的一组容器应用上
 - Service的服务进程都基于Socket通信方式对外提供服务。
 - 每个Service通常由多个相关的服务进程来提供服务，每个服务进程都有一个独立的Endpoint(IP+Port)访问点。
 - 通过Service（虚拟Cluster IP + Service Port）连接到指定的Service上。
 - 基于透明负载均衡 和 故障恢复机制
 1. 负载均衡：是高可用网络基础架构的关键组件，通常用于将工作负载分布到多个服务器来提高网站、应用、数据库或其他服务的性能和可靠性。





2. 故障恢复机制：健康探针



kubelet:运行在cluster所有节点上,负责启动POD和容器

kubeadm:用于初始化cluster

kubectl:kubectl是kubernetes命令行工具，通过kubectl可以部署和管理应用，查看各种资源，创建，删除和更新组件

- 上图可以看到如下组件，使用特别的图标表示Service和Label：
 - Pod
 - Container（容器）
 - Label ()（标签）
 - Replication Controller（复制控制器）
 - Service ()（服务）
 - Node（节点）
 - Kubernetes Master（Kubernetes主节点）
- Pod
 - 为Service提供服务的进程放到容器进行隔离。
 - 每个服务进程包装到相应的Pod中，成为Pod中运行的一个容器。
- Pod运行在节点的环境中（Node）节点可以是物理服务器或者虚拟机，之上安装了Kubernetes平台。
 - 一个节点上运行几百个Pod。
- Pod中运行着特殊容器(Pause) 和 业务容器
 - **Pause提供网络栈和Volume挂载卷。**（Pod内部共享数据）

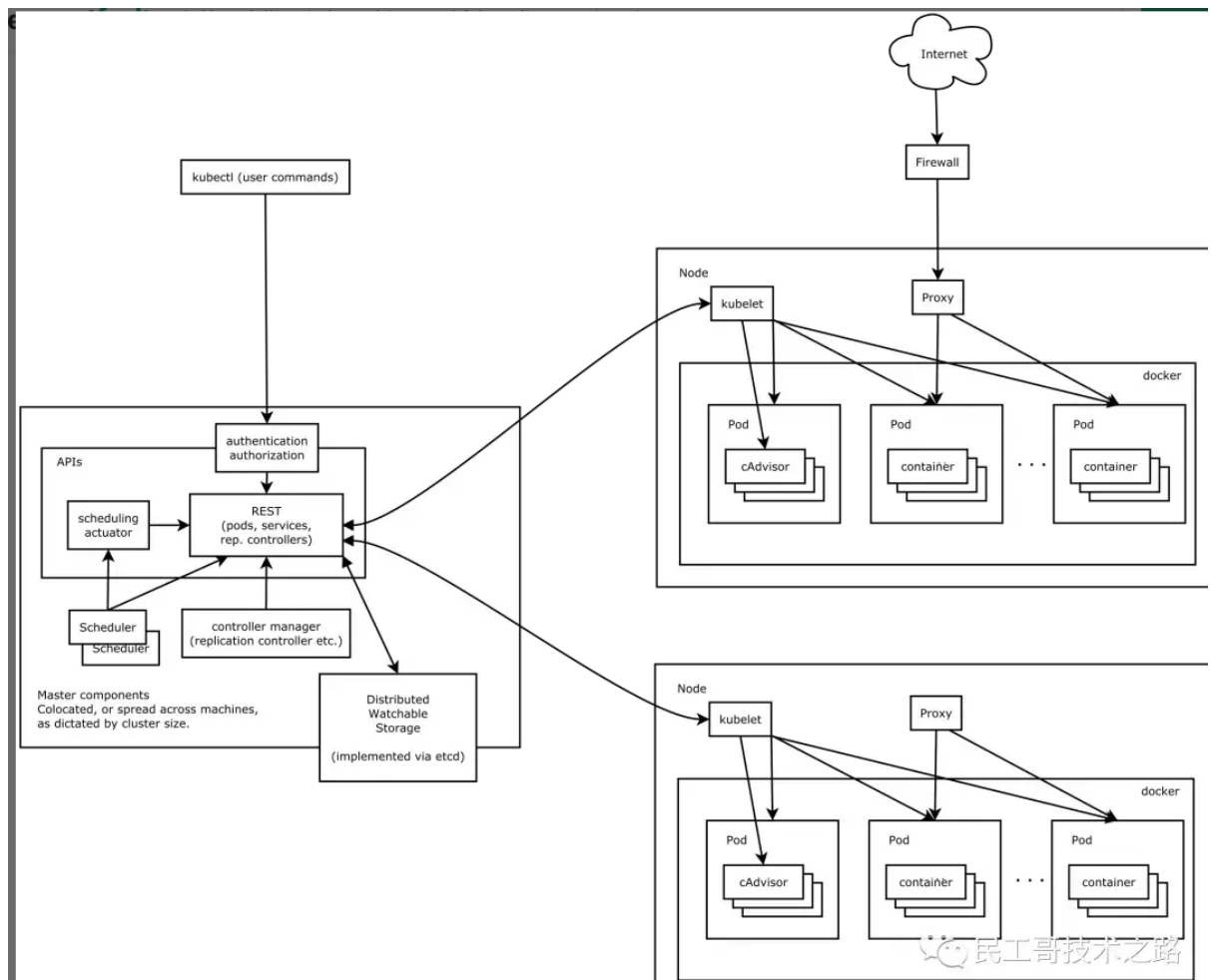
- 网络栈: <https://www.cnblogs.com/agilestyle/p/11394930.html>
- Volume挂载卷: 提供独立于容器之外的持久化存储, 提供容器之间的共享的数据
- 提供服务的一组Pod映射成一个服务
- 并不是每个Pod和它里面运行的容器都能映射到一个Service
- 集群
 - 集群是一组节点, 这些节点可以是物理服务器或者虚拟机, 之上安装了Kubernetes平台。
 - 包含一个Master节点和一群工作节点 (Node)
 - Master节点运行着一组管理进程。(全自动)
 - Kubernetes最小运行单元是Node上的Pod

核心组件

k8s创建pod和service的过程 - 走看看 (zoukankan.com)

https://blog.csdn.net/chen_haoren/article/details/108754174

<https://mp.weixin.qq.com/s/ctdvbasKE-vpLRxDJwVMw>



主要由以下几个核心组件组成:

- apiserver
 - **所有服务访问的唯一入口，提供认证、授权、访问控制、API 注册和发现等机制**
- controller manager
 - 负责维护集群的状态，比如副本期望数量、故障检测、自动扩展、滚动更新等
- scheduler
 - 负责资源的调度，按照预定的调度策略将 Pod 调度到相应的机器上
- etcd
 - 键值对数据库，保存了整个集群的状态
- kubelet
 - 负责维护容器的生命周期，同时也负责 Volume 和网络的管理
- **kube-proxy**
 - [一文看懂 Kube-proxy - 知乎 \(zhihu.com\)](#)
 - 负责为 Service 提供 cluster 内部的服务发现和负载均衡
 - Kube-proxy维护节点上的网络规则，实现了Kubernetes Service 概念的一部分。它的作用是使发往 Service 的流量（通过ClusterIP和端口）负载均衡到正确的后端Pod。
- Container runtime
 - 负责镜像管理以及 Pod 和容器的真正运行

除了核心组件，还有一些推荐的插件：

- CoreDNS
 - 可以为集群中的 SVC 创建一个域名 IP 的对应关系解析的 DNS 服务
- Dashboard
 - 给 K8s 集群提供了一个 B/S 架构的访问入口
- Ingress Controller
 - 官方只能够实现四层的网络代理，而 Ingress 可以实现七层的代理
- Prometheus
 - 给 K8s 集群提供资源监控的能力
- Federation
 - 提供一个可以跨集群中心多 K8s 的统一管理功能，提供跨可用区的集群

以上内容参考链接：

[Kubernetes之前世今生 | Escape \(escapelife.site\)](#)

为什么要用Kubernetes

- 微服务架构：将一个巨大的单体应用分解成很多小的互相连接的微服务，一个微服务由多个实例副本支撑。
- 关于B/S, C/S 区别，以及tomcat的作用（主要作为B/S 浏览器/服务器模式中部署服务器使用，监听接口等）C/S 一般需要用到Socket通信（HTTP请求就是基于TCP连接（Socket连接），在进行请求（短连接）），客户端和服务端传数据。

◦ <https://www.pianshen.com/article/29761200978/>

https://zhuanlan.zhihu.com/p/158086508?from_voters_page=true

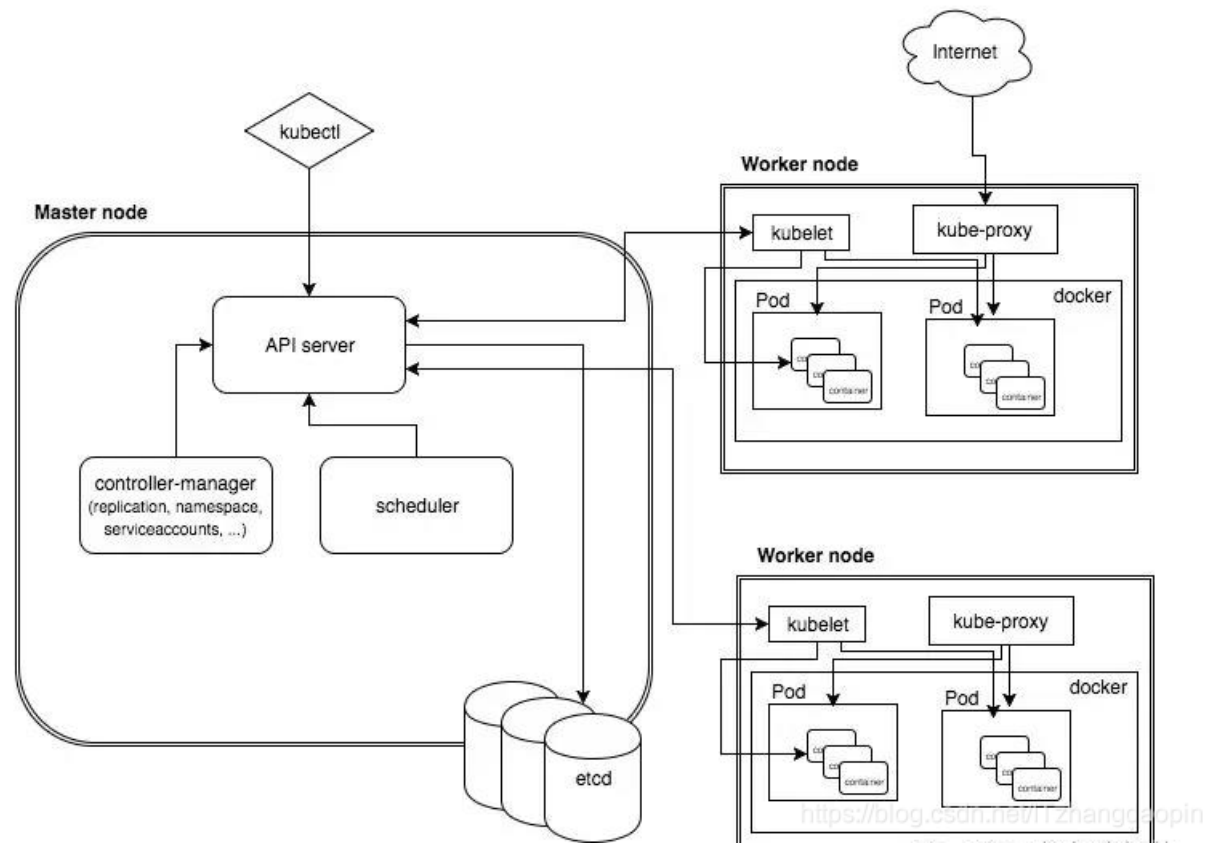
小例子

K8S中的pod、services、容器的概念和区别

关系

https://www.cnblogs.com/gunxiaoshi/p/11052927.html

- service 包含多个 pod（提供服务），(一个node 可以包含提供不同服务的Pod)， Pod 包含多个容器。
- node



K8s的部署架构：

- kubernetes中有两类资源，分别是master 和 nodes，master和nodes 上跑的服务如下图：

1 kube-apiserver		kubelet
2 kube-controller-manager		
3 kube-scheduler		kube-proxy
4 -----		-----
5 k8s master		node (non-master)

- master: 负责管理整个集群，例如，对应用进行调度(扩缩)、维护应用期望的状态、对应用进行发布等。
- node: 集群中的**宿主机**(可以是物理机也可以是虚拟机)，每个node上都有一个agent，名为**kubelet**，用于跟master通信。
 - node需要由管理容器的工具包，用于管理在node上运行的容器(docker或rkt)
 - 一个k8s集群至少要有3个节点。

- kubelet 通过master暴露的 API与master通信，用户也可以直接调用master的API做集群的管理。

K8s中的对象Objects

- podKubernetes 之Pod学习 - 散尽浮华 - 博客园 (cnblogs.com)
 - k8s中的最小部署单元，**不是一个程序/进程，而是一个环境(包括容器、存储、网络ip:port、容器配置)**。
- 网络ip:port = Endpoint : Pod 里一个服务进程的对外通信地址。（因此可以存在多个Endpoint (pod内部不同的容器的端口))
 - 其中可以运行1个或多个container(docker或者其他容器)，在一个pod内部的container共享所有资源，包括共享pod的ip:port 和磁盘。
 - pod是临时性的，用完即丢弃的，当pod中的进程结束、node故障，资源短缺时，pod会被干掉。
 - 用户很少直接创建一个独立的pods，而会通过k8s中的controller来对pod进行管理。
 - 临时性是说，每个pod基本都是根据服务需求来创建，包含着容器，可以根据不同的 (pod) IP + (容器) port被访问。
 - Pod分为静态Pod和普通Pod
 - 静态Pod并不放在Kubernet的etcd存储里，而是放在某个具体的Node上的一个具体文件中，**只在这个Node上启动运行**
 - 普通的Pod一旦被创建，就会被放入到etcd中存储，**随后被Kubernetes Master的调度到某个具体的Node上并进行绑定(Binding)**

随后该Pod被对应的Node上的kubelet进程实例化成**一组相关的Docker容器并启动**。

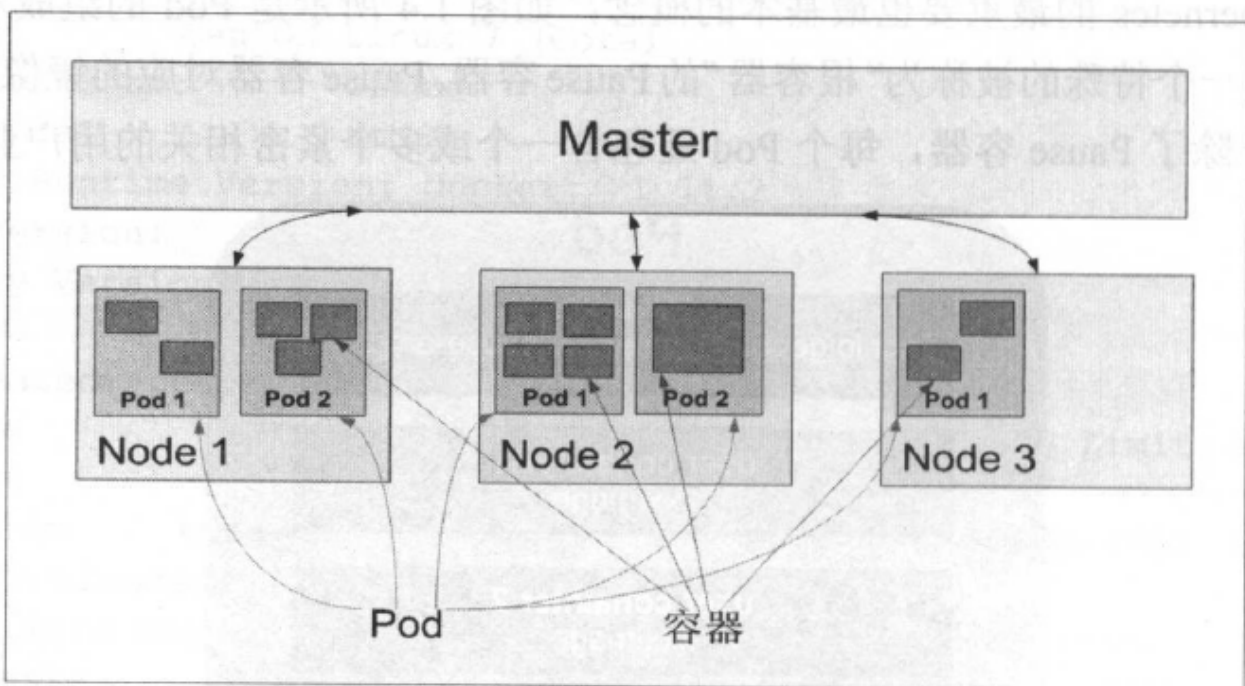


图 1.5 Pod、容器与 Node 的关系

- 容器停止，重启Pod（所有容器）；Node 宕机，所有Pod被重新调度到其他节点。

- services
 - pod是临时性的，pod的ip:port也是动态变化的。
 - 服务调用方自动感知服务提供方。
 - service是通过apiserver创建出来的实例对象

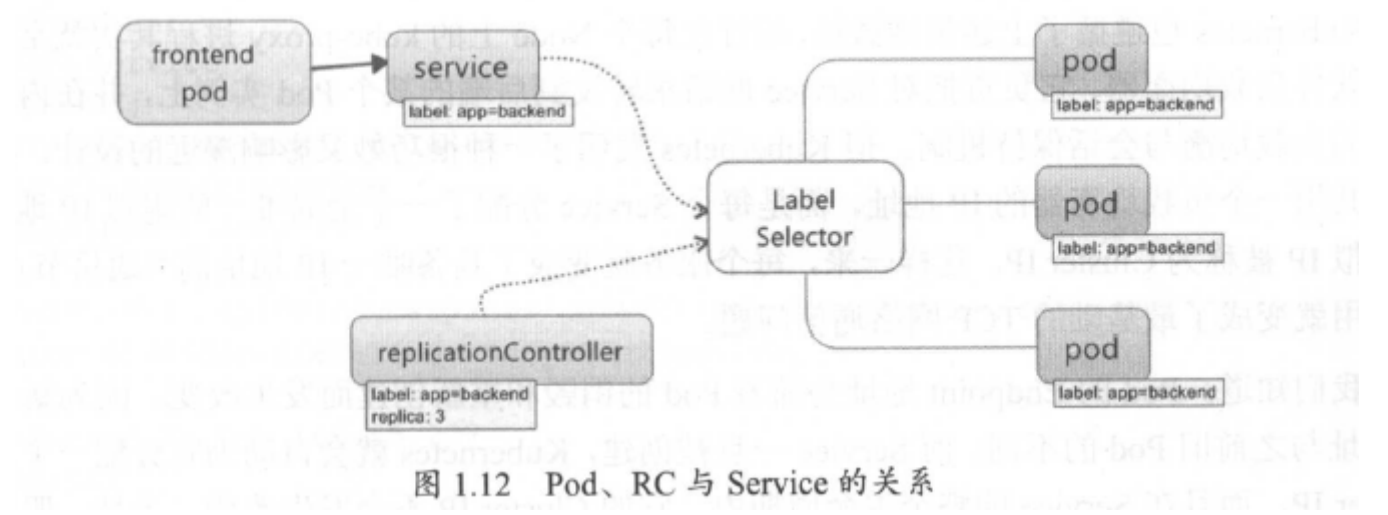
```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

- Event
 - 是一个事件的记录，记录了事件的最早产生时间，最后重现时间、重复次数、发起者、类型、原因。
 - 会关联到具体的资源对象上，包括Node，Pod

每个Pod都可以对其使用的服务器上的计算资源设置限额。（包括CPU和Memory）

- Label（标签）（用来筛选）
 - key = value的键值对
 - 可以附加到各种资源 Node，Pod，Service，RC
 - 任意数量
- Replication Controller (RC)（RS）
 - RC：保证Pod平稳持续的运行，指定Pod副本数量的基础上可以弹性扩缩容，滚动升级。
RC:在每个节点上创建Pod，Pod上如果有相应的Images可以直接创建，如果没有，则会拉取这个镜像再进行创建。
 - 定义了一个期望的场景，声明某种Pod的副本数量在任意时刻都符合某个预期值
 - 包含
 1. Pod期待的副本数 (replicas)
 2. 用于筛选目标Pod的Label Selector
 3. 当Pod的副本数量小于预期数量时，用于创建新的Pod的Pod模板 (template)
- Deployment
 - Deployment 是K8s v1.2引入的新概念，为了更好的解决Pod的编排

- 在Deployment内部使用了Replica Set来实现。
- HPA
 - 横向自动扩容
- StatefulSet
 - 有状态的Pod管理对象
 - 固定ID
 - 集群规模比较固定
- Service
 - 最核心的资源对象之一
 - 微服务架构中的一个"微服务"
 -



K8s中 资源控制器 RC、RS、Deployment详解

主机配置规划

服务器名称(hostname)	系统版本	配置	内网IP	外网IP(模拟)
k8s-master	CentOS7.7	2C/4G/20G	172.16.1.110	10.0.0.110
k8s-node01	CentOS7.7	2C/4G/20G	172.16.1.111	10.0.0.111
k8s-node02	CentOS7.7	2C/4G/20G	172.16.1.112	10.0.0.112

- 什么是控制器
 - Kubernetes 中内建了很多controller（控制器），相当于一个 **状态机**，用来**控制pod的具体状态和行为**。
 - 部分控制器类型如下：
 - **ReplicationController 和 ReplicaSet**
 - **Deployment**
 - **DaemonSet**

- StatefulSet
- Job/CronJob
- HorizontalPodAutoscaler

ReplicationController 和ReplicaSet

- **Replication Controller (RC)**用来确保容器应用的副本数始终保持在用户定义的副本数，即如果有容器异常退出，会自动创建新的pod来替代；而异常多出来的容器也会自动回收。

在新版的Kubernetes中建议使用ReplicaSet (RS)来取代Replication Controller。ReplicaSet跟Replication Controller没有本质的不同，只是名字不一样，但ReplicaSet支持集合式selector。

- ReplicaSet 可以单独使用，但是被Deployments 用作协调 Pod 的创建、删除和更新的机制。当使用 Deployment 时，你不必担心还要管理它们创建的 ReplicaSet，Deployment 会拥有并管理它们的 ReplicaSet。

ReplicaSet 是下一代的 Replication Controller。ReplicaSet 和 Replication Controller 的唯一区别是选择器的支持。ReplicaSet 支持新的基于集合的选择器需求，这在标签用户指南中有描述。而 Replication Controller 仅支持基于相等选择器的需求。

一个 ReplicaSet 对象就是由副本数目的定义和一个 Pod 模板组成的，它的定义就是 Deployment 的一个子集。Deployment 控制器实际操纵的是 ReplicaSet 对象，而不是 Pod 对象。

ReplicaSet 负责通过“控制器模式”，保证系统中 Pod 的个数永远等于指定的个数（比如，2 个）。这也正是 Deployment 只允许容器的 restartPolicy=Always 的主要原因：只有在容器能保证自己始终是 Running 状态的前提下，ReplicaSet 调整 Pod 的个数才有意义。

Deployment 通过“控制器模式”，来操作 ReplicaSet 的个数和属性，进而实现“水平扩展 / 收缩”和“滚动更新”这两个编排动作

K8s中的通信

- **核心**
 - 一个Pod里的容器与另外主机上的Pod容器能够直接通信。
- **网络前提条件-网络模型**
 - 所有的Pods之间可以在不使用NAT网络地址转换的情况下相互通信
 - 所有的Nodes之间可以在不使用NAT网络地址转换的情况下相互通信
 - **每个Pod自己看到的自己的ip和其他Pod看到的一致**
- **k8s网络模型设计原则**
 - 每个Pod都拥有一个独立的 IP地址，而且 假定所有 Pod 都在一个可以直接连通的、扁平的网络空间中。
 - 不管它们是否运行在同 一个 Node (宿主机)中，**都要求它们可以直接通过对方的 IP 进行访问。**
 - 设计这个原则的原因是，**用户不需要额外考虑如何建立 Pod 之间的连接，也不需要考虑将容器端口映射到主机端口等问题。**

K8s中 service 端口

```
service "myweb" created
[root@localhost Documents]# kubectl get services
NAME           CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes     10.254.0.1      <none>           443/TCP          2h
mysql          10.254.53.45    <none>           3306/TCP         51m
myweb          10.254.179.62   <nodes>          8080:30001/TCP   6s
```

网卡IP + nodport 192.168.10.131:30001 (虚拟机中的IP不做端口映射的话, 只能被主机访问。)

clusterIP + port (虚拟, 内部访问)

k8s中的端口 (port)



郭青耀 关注

0.215 2020.10.15 00:07:20 字数 124 阅读 1,722

port ---service

port是k8s集群内部访问service的端口, 即通过clusterIP: port可以访问到某个service

nodePort --- 集群

nodePort是外部访问k8s集群中service的端口, 通过nodeIP: nodePort可以从外部访问到某个service。

targetPort ---pod

targetPort是pod的端口, 从port和nodePort来的流量经过kube-proxy流入到后端pod的targetPort上, 最后进入容器。

containerPort ----pod内部

containerPort是pod内部容器的端口, targetPort映射到containerPort。

外部网络访问K8s service 的方式 (必须要通过Node IP进行通信)

<https://blog.csdn.net/u010942475/article/details/105205203>

<https://www.cnblogs.com/cheyunhua/p/8469647.html>

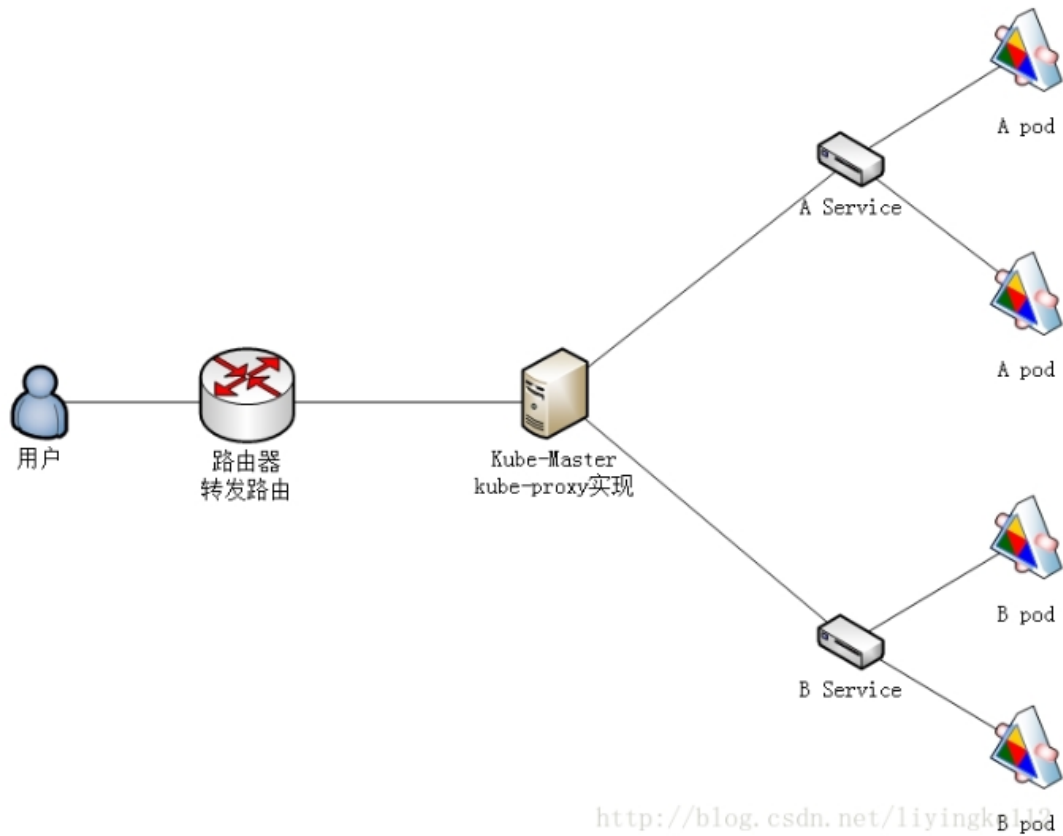
<https://blog.csdn.net/liyingke112/article/details/84909274>

<https://www.cnblogs.com/yxh168/p/12245450.html>

- Kube-proxy + ClusterIP
 - kubernetes 版本大于或者等于1.2, 配置

- 修改master的/etc/kubernetes/proxy, 把KUBE_PROXY_ARGS=""改为KUBE_PROXY_ARGS="-proxy-mode=userspace"
重启kube-proxy服务
在核心路由设备或者源主机上添加一条路由, 访问cluster IP段的路由指向到master上。

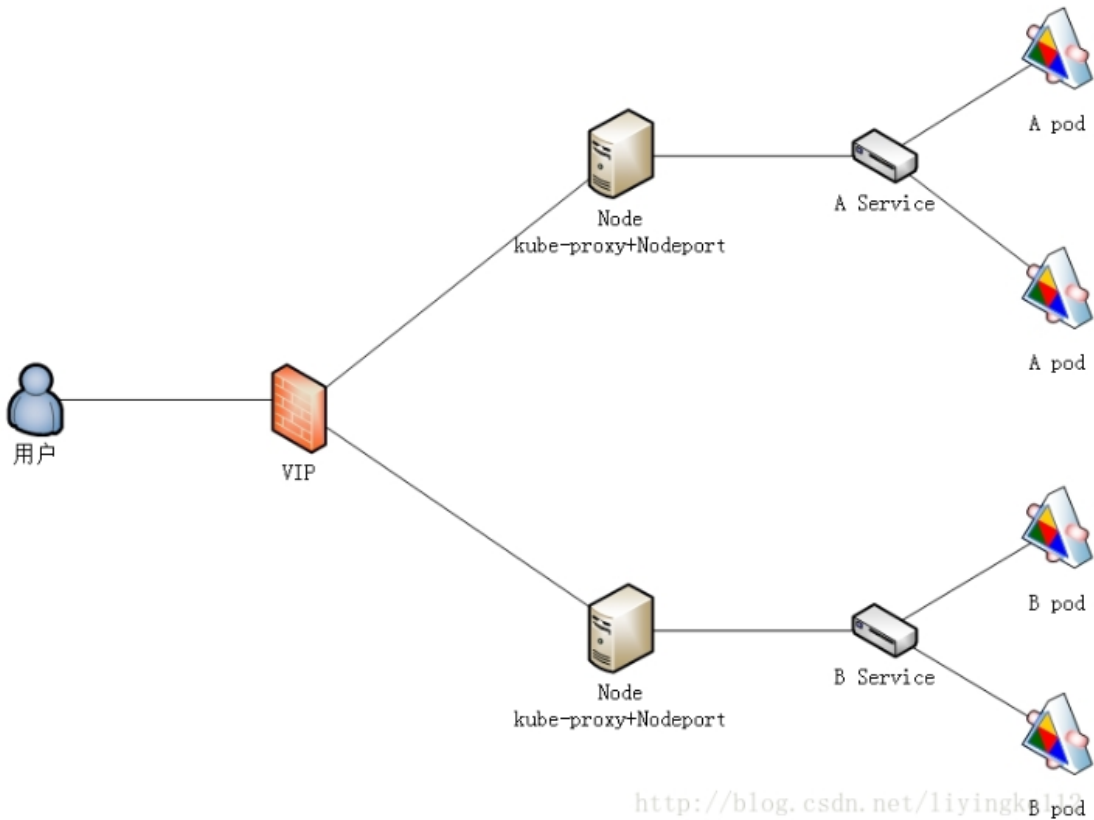
kubernetes版本小于1.2时, 直接添加路由



○

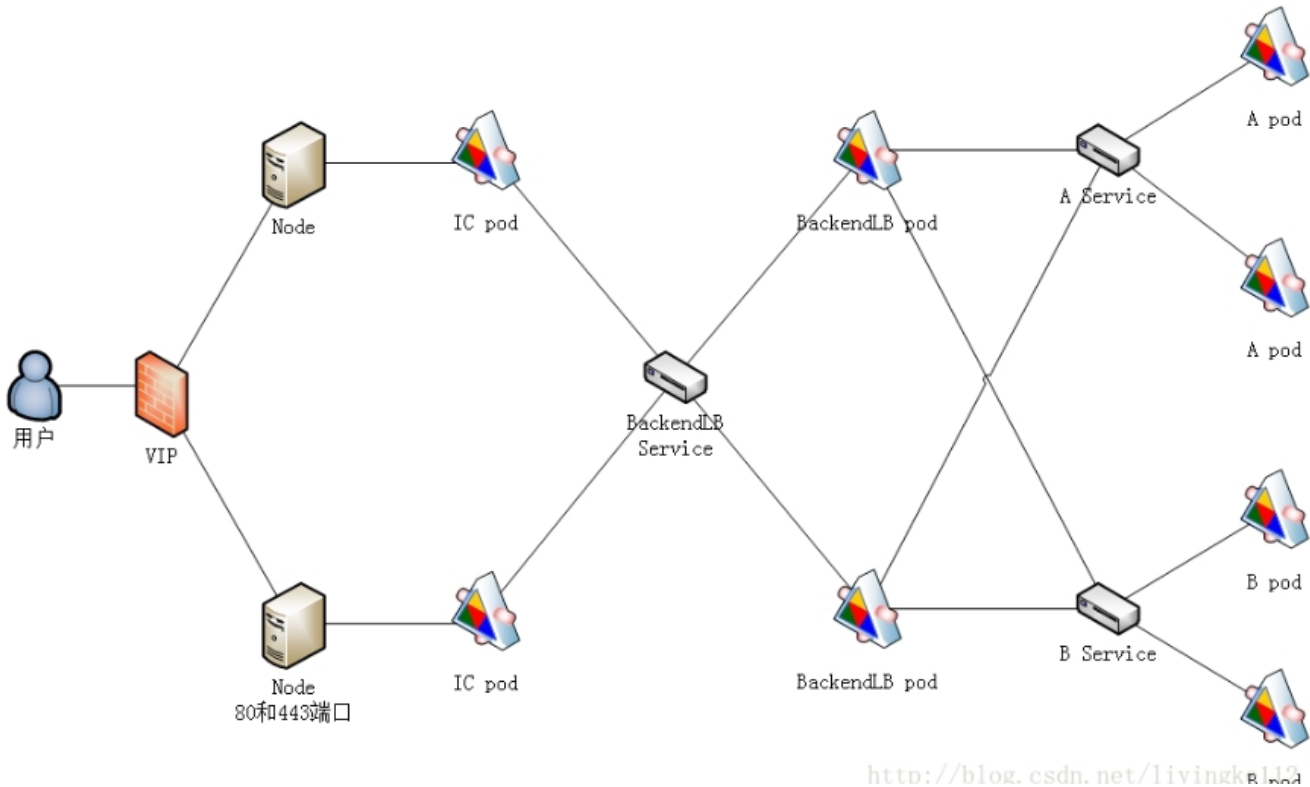
负载均衡器+NodePort

- 部署一个负载均衡器 (nginx、keepalive等)



Ingress

- Ingress是一种HTTP方式的路由转发机制，由Ingress Controller和HTTP代理服务器组合而成。Ingress Controller实时监控Kubernetes API，实时更新HTTP代理服务器的转发规则。HTTP代理服务器有GCE Load-Balancer、HaProxy、Nginx等开源方案。



loadbalance

- LoadBalancer在NodePort基础上，K8S可以请求底层云平台创建一个负载均衡器，将每个Node作为后端，进行服务分发。该模式需要底层云平台（例如GCE）支持。

Namespace（命名空间）

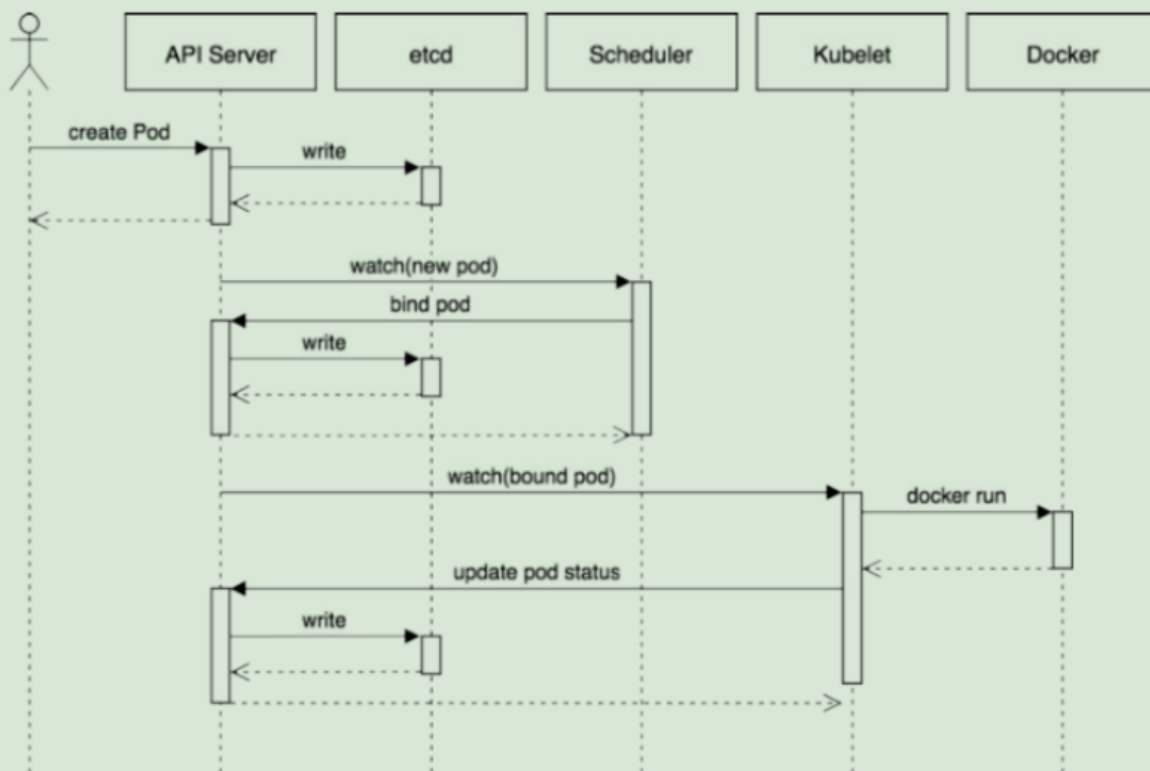
- Namespace（命名空间）**用于多租户的资源隔离。**
- 默认default，需要加上参数进行查看资源

Annotation 注解

- key/value 键值对的形式进行定义
- 任意定义的信息，便于外部工具查找。
-

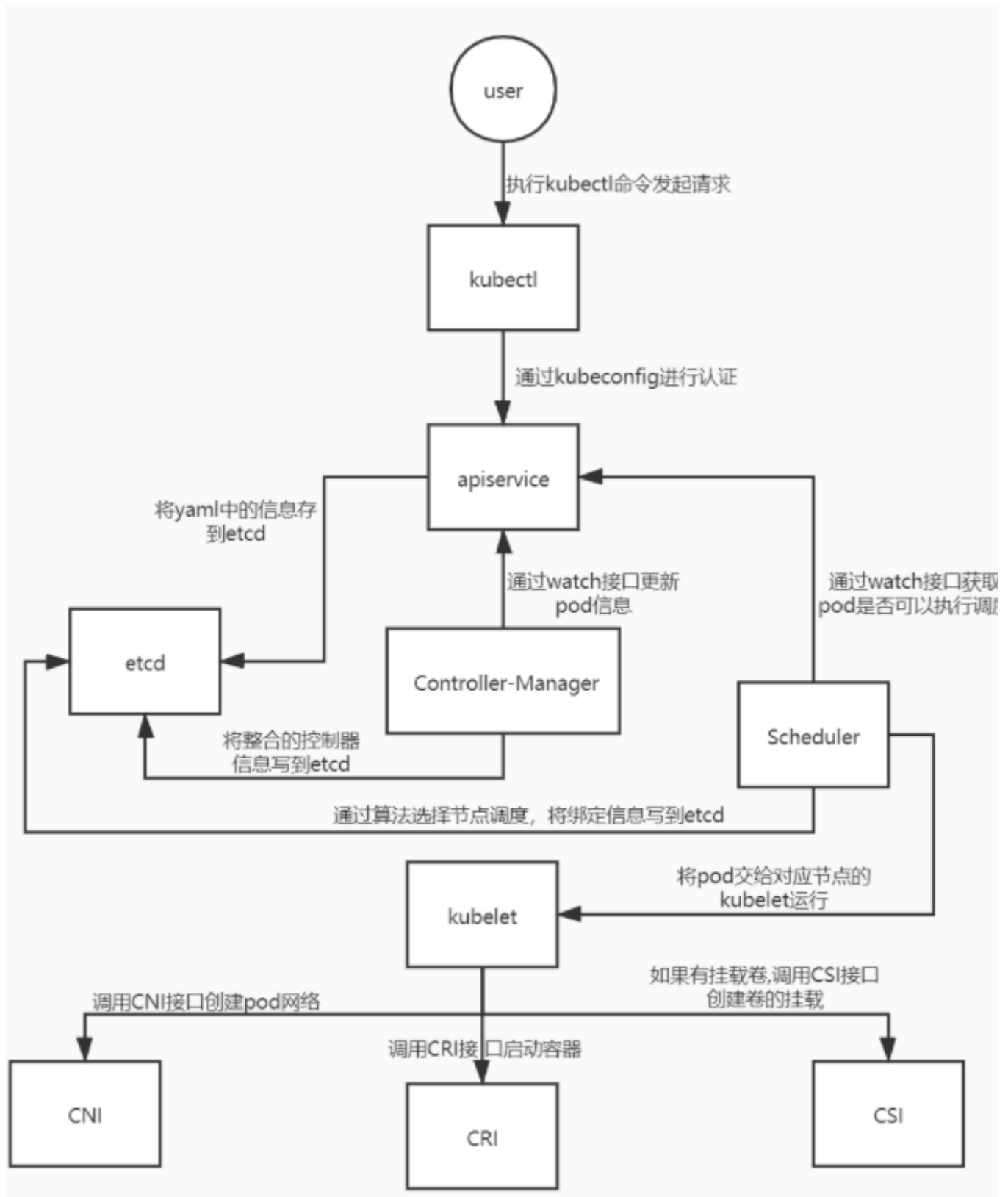
k8s创建pod和service的过程

比如典型的创建Pod的流程为



1. 用户通过REST API创建一个Pod
2. apiserver将其写入etcd
3. scheduler检测到未绑定Node的Pod，开始调度并更新Pod的Node绑定
4. kubelet检测到有新的Pod调度过来，通过container runtime运行该Pod
5. kubelet通过container runtime取到Pod状态，并更新到apiserver中

<http://blog.csdn.net/harris135>



etcd键值库只对API server开放，API server就是k8s集群中的控制中心，一般对API server所在的Master做集群，避免因单点故障影响k8s集群中业务的访问。

- 通过kubectl命令输入create pod命令
- 创建命令到达API server后，API server把**创建pod的信息(还没创建)**写入到etcd键值库中
- API server把创建新pod的需求传给scheduler，scheduler通过算法计算分配到合适的Node节点，分配绑定到后端Node上的信息传回至API server，API server在把此信息写入到etcd键值库中
- API server传递新建pod信息至Scheduler指定node节点的kubelet程序，kubelet接管后，创建出本地指定的container后，完成信息返回至API server

- API server把新建好的pod信息写入到etcd键值库中
- 至此，一个简单的kubectl create POD_NAME ***就新建完成

创建pod节点的service

- 通过kubectl提交一个pod的service创建请求
- Controller Manager会通过对应的Label标签查询到相关的pod实例，生成Service的Endpoints信息，并通过API server写入到etcd键值库中
- Worker Node节点上的kube proxy通过API server查询并监听service对象与其对应的Endpoints信息(服务发现)，创建一个类似负载均衡器实现Service访问到后端Pod的流量转发功能(负载均衡)。

pod使用

- [Docker 容器后台运行和前台运行的区别](#)
- K8s 系统中对长时间运行容器的要求是：其主程序需要一直在前台执行。

1. <https://blog.csdn.net/raoxiaoya/article/details/109194514>
2. <https://www.cnblogs.com/cag2050/p/10144874.html>

创建Pod流程 <https://www.cnblogs.com/potato-chip/p/14445546.html>

[Kubernetes 之Pod学习 - 散尽浮华 - 博客园 \(cnblogs.com\)](#)

静态Pod

- 静态Pod是由kubelet 创建，运行在kubelet在的节点
- Kubelet 是 kubernetes 工作节点上的一个代理组件，运行在每个节点上。
- 创建静态Pod有两种方式：配置文件方式和HTTP方式

Deployment和service

<https://zhuanlan.zhihu.com/p/358916098>

deployment根据Pod的标签关联到Pod,是为了管理pod的生命周期
service根据Pod的标签关联到pod,是为了让外部访问到pod,给pod做负载均衡
需要注意:

deployment控制器关联的Pod, Pod的name和hostname(如果不手动指定)就是deployment控制器的Name

StatefulSet控制器关联的Pod, Pod的Name和Hostname(如果不手动指定)就是StatefulSet控制器的Name + 序号

pod和service

服务

将运行在一组 Pods 上的应用程序公开为网络服务的抽象方法。

使用 Kubernetes，你无需修改应用程序即可使用不熟悉的服务发现机制。Kubernetes 为 Pods 提供自己的 IP 地址，并为一组 Pod 提供相同的 DNS 名，并且可以在它们之间进行负载均衡。

[一关系图让你理解K8s中的概念，Pod、Service、Job等到底有啥关系 - 知乎 \(zhihu.com\)](#)

K8s API Server 原理分析

- kubernetes API Server 的核心功能是**提供了**Kubernetes各类资源对象(如Pod、RC、Service等) (CDR) 的**增删改查** 及Watch等**HTTP Rest接口**
 - [Watch](#)
 - [HTTP Rest](#)
- **集群内**各个功能模块之间数据交互和通信的中心枢纽，是整个系统的数据总线 and 数据中心。
 - 是集群管理的API入口
 - 是资源配额控制的入口
 - 提供了完备的集群安全机制

K8s API Server 概述

- 通过名为kube-apiserver的进程**提供服务**，该进程运行在Master节点上。
- kube-apiserver 进程在本机的8080端口(对应参数--insecure-port), 提供REST服务。
- 通过命令行工具kubectl来 交互
 - 接口时REST调用
 - 声明式API (<https://zhuanlan.zhihu.com/p/136116883>) 。k8s更进一步的是，你只需要提交一个申请单，然后由k8s系统完成对象的创建。

- kubectl 和 API Server的关系

<https://blog.csdn.net/byxiaoyuonly/article/details/115556770>

<https://www.cnblogs.com/elcino/p/7206712.html>

<http://dockone.io/article/8997>

1. k8s的架构是用户使用kubectl工具对虚拟机资源进行各种各样的控制和定制。
2. 而kubectl本身并不包含对其核心资源的访问与控制。而是通过http通信与api-server进行交互实现资源的管理。
3. 而api-server的核心其实就是etcd数据库，它将各种资源的管理通过对etcd中的数据进行更改实现。

k8s之Deployment详解

Controller Manager 原理分析

在Kubernetes架构中，有一个叫做kube-controller-manager的组件。这个组件，是一系列控制器的集合。其中每一个控制器，都以独有的方式负责某种编排功能。而Deployment正是这些控制器中的一种。

- 集群内部得管理控制中心，负责集群内的Node、Pod副本、服务端点(Endpoint)、命名空间(Namespace)、服务账号(ServiceAccount)、资源定额(ResourceQuota)等的管理。
- 当某个Node意外宕机时，Controller Manager 会及时发现此故障并执行**自动化修复流程**，确保集群**始终**处于预期的工作状态。

kube-proxy

- K8s集群的每个Node 上都会运行一个kube-proxy服务进程，这个进程可以看作**Service的透明代理+负载均衡器**
 - 核心功能是将某个Service的访问请求转发到后端的多个Pod实例上。

REST

- REST本身只是为分布式超媒体系统设计的一种架构风格，而不是标准。
-