```
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
                          和数据中心
                                      1.是集群管理的API入口
                          2.功能特性
                                     2.是资源配额控制的入口
                                      3.提供了完备的集群安全机制
                                  1.kubernetes API Server通过一个名为kube-apiserver的进程提供服务,该
                                  进程运行在master上
                                  2.默认情况下, kube-apiserver进程在本机的8080端口提供REST服务。可以
                                   同时启动HTTPS安全端口来启动安全机制,加强Rest API访问的安全性
                                  3.使用场景:开发基于kubernetes的管理平台。比如调用Kubernetes API来
                                  完成Pod、Service、RC等资源对象的图形化创建和管理界面。
                                                      1.APIServer拥有大量高性能的底层代码。在API Server源码中使用协程+队
                                                      列这种轻量级的高性能并发代码,使得单进程的API Server具备了超强的多核
                                                      处理能力,从而以很快的速度并发处理大量的请求
                                                      2.普通List接口结合异步watch接口,不但完美解决了Kubernetes中各种资源
                                                      对象的高性能同步问题,也极大提升了Kubernetes集群实时响应各种事件的
                                  4.API Server的性能保证
                                                      3.采用了高性能的etcd数据库而非传统的关系型数据库,不仅解决了数据可靠
                                                      性问题,也极大提升了API Server数据访问层性能
                                                              1.API层:主要以REST方式提供各种API接口,除了有kubernetes资源对象的
1.Kubernetes API Server原理解析
                                                              CRUD和Watch等主要API,还有健康检查、UI、日志、性能指标等运维监控
                                                              2.访问控制层: 当客户端访问API接口时,访问控制层负责用户身份鉴权,验
                                                              明用户身份,核准用户对Kubernetes资源对象的访问权限,然后根据配置的
                                                              各种资源访问许可逻辑,判断是否允许访问
                                                              3.注册表层: kubernetes把所有资源对象都保存在注册表 (Registry) 中,
                                                              针对注册表中的各种资源对象都定义了:资源对象的类型、如何创建资源对
                         4.API Server架构解析
                                           1.API Server架构分层
                                                              象、如何转换资源对象的不同版本,以及如何将资源编码和解码为JSON或
                                                              ProtoBuf格式进行存储
                                                              4.etcd数据库:用于持久化存储Kubernetes资源对象的KV数据库。etcd的
                                                              watch API接口对于API Server来说至关重要,因为通过这个接口,API
                                                              Server创新性地设计了List-Watch这种高性能的资源对象实时同步机制,使
                                                              Kubernetes可以管理超大规模的集群,即使响应和快速处理集群中的各种事
                                                    1.最主要的接口时资源对象的增删改查接口
                                                    2.kubernetes Proxy API接口,这类接口的作用是代理REST请求,即
                          5.独特的kubernetes Proxy API 接口
                                                    Kubernetes API Server把收到的REST请求转发到某个Node上的kubelet守
                                                    护进程的REST端口,由该kubelet进程负责响应
                                            1.kubernetes API Server作为集群的核心,负责集群各功能模块之间的通信
                          6.集群模块之间的通信
                                           2.集群内的各个功能模块通过API Server将信息存入etcd,当需要获取和操作
                                            这些数据时,则通过API Server提供的REST接口来实现
                                                     1.核心作用是确保在任何时候集群中某个RC关联的Pod副本数量都保持预设
                                                                             1.确保在当前集群中有且仅有N个pod实例,N是在RC中定义的Pod副本数量
                                                     2.Replication Controller的职责
                                                                             2.通过调整RC的spec.replicas属性值来实现系统扩容或缩容
                        1.Replication Controller (副本控制器)
                                                                             3.通过改变RC中的Pod模板(主要是镜像版本)来实现系统的滚动升级
                                                                   1.重新调度
                                                                   2.弹性升缩
                                                     3.典型使用场景
                                                                   3.滚动更新
                                         1.kubelet进程在启动时通过API Server注册自身的节点信息,并定时向API
                                         Server汇报状态信息,API Server在接收到这些信息后,会将这些信息更新到
                                         2.在etcd中存储的节点信息包括节点健康状况、节点名称、节点地址信息、操
                                         作系统版本、Docker版本、kubelet版本等
                        2.Node Controller
                                         3. 节点健康状况包括 "就绪 (True) " , "未就绪 (False) " 和 "未知
                                         (Unknown)" 三种
                                                         1.资源配额管理确保了指定的资源对象在任何时候都不会超量占用系统物理资
                                                         源,避免了由于某些业务进程设计或实现的缺陷导致整个系统运行紊乱甚至意
2.Controller Manager原理解析
                                                         外宕机,对整个集群的平稳运行和稳定性有非常重要的作用
                                                                                        1.容器级别,可以对CPU和Memory进行限制
                                                                                        2.Pod级别,可以对CPU和Memory进行限制
                                                        2.kubernetes支持的三个层次的资源配额管理
                                                                                        3.Namespace级别,为Namespace (多租户) 级别的资源限制,包括: Pod
                        3.ResourceQuota Controller (资源配额管理)
                                                                                        数量、Replication Controller数量、Service数量、ResourceQuota数量、
                                                                                        Secret数量、可持有的PV数量
                                                                                                    1.LimitRanger:作用于Pod和Container
                                                         3.k8s的配额管理是通过Admission Control (准入控制)来控制的
                                                                                                   2.ResourceQuota:作用于Namespace,限定一个Namespace里的各类资
                                            1.用户通过API Server可以创建新的Namespace并将其保存在etcd中,
                                            Namespace Controller定时通过API Server读取这些Namespace的信息
                        4.Namespace Controller
                                                       1.Endpoints表示一个对应的所有Pod副本的访问地址,Endpoints
                                                       Controller就是负责生成和维护所有Endpoints对象的控制器
                        5.Service Controller与Endpoints Controller
                                                       2.Endpoints对象在每个Node上的kube-proxy进程使用,kube-proxy进程
                                                       获取每个Service的Endpoints,实现了Service的负载均衡功能
                  1.Kubernetes Scheduler的作用是将待调度的Pod(API新创建的Pod、
                 Controller Manager为补足副本而创建的Pod等)按照特定的调度算法和调
                  度策略绑定到集群中某个合适的Node上,并将信息写入etcd中
                  2.通过调度算法调度为待调度Pod列表、可用Node列表中的每个Pod从Node
                  列表中选择一个合适的Node
                                                         1.预选调度过程,即遍历所有目标Node,筛选出符合要求的候选节点
                                                         2.确定最优节点,在第一步的基础上,采用优选策略计算出每个候选节点的积
                                                         分,积分最高者胜出
                                                                                        1.首先读取备选pod的所有Volume的信息,对每个Volume执行以下步骤进
                                                                                        2.如果该Volume是gecPersistentDisk,则将Volume和备选节点上所有的
                                                                                        Pod的每个Volume都进行比较,如果发现相同的gecPersistentDisk,则返
                                                                        1.NoDiskConflict
                                                                                        回false,表明存在磁盘冲突,检查结束,反馈给调度器该备选节点不适合作
                                                                                        3.如果检查完备选Pod的所有Volume均为发现冲突,则返回true,表明不存
                                                                                        在磁盘冲突,反馈给调度器该备选节点适合备选Pod
                                                                                         1.计算备选Pod和节点中已存在Pod的所有容器的需求资源(CPU和内存)的
                                                                                         2.获得备选节点的状态信息,其中包含节点的资源信息
                                                                       2.PodFitsResources
                                                                                         3.如果在备选Pod和节点中已存在Pod的所有容器的需求资源(内存和CPU)
                                                                                         的总和,超出了备选节点拥有的资源,则返回false,表明备选节点不适合备
                                                                                         选Pod,否则返回true,表明备选节点适合备选Pod
                                                                                           1.如果Pod没有指定spec.nodeSelector标签选择器,则返回true
                                                                       3.PodSelectorMatches
                                                                                           2.否则,获得备选节点的标签信息,判断节点是否包含备选Pod的标签选择器
                                                                                           所指定的标签,如果包含,则返回true,否则返回false
                                                         3.预选策略介绍
                                                                                      1.判断备选Pod的spec.nodeName域所指定的节点名称和备选节点的名称是
                  3.Kubernetes Scheduler当前提供的默认调度流程分为两步
                                                                        4.PodFitsHost
                                                                                      否一致,如果一致,则返回true,否则返回false
                                                                                              1.该策略用于判断策略列出的标签在备选节点中存在时,是否选择该备选节点
                                                                                              2.读取备选节点的标签列表信息
                                                                       5. Check Node Label Presence
                                                                                              3.如果策略配置的标签列表存在于备选节点的标签列表中,且策略配置的
                                                                                              presence值为false,则返回false,否则返回true;如果策略配置的标签列表
                                                                                              不存在于备选节点的标签列表中,且策略配置的presence值为true,则返回
                                                                                              false,否则返回true
                                                                                           1.该策略用于判断备选节点是否包含策略指定的标签,或包含和备选Pod在相
                                                                                           同Service和Namespace下的Pod所在节点的标签列表。如果存在,则返回
                                                                       6.CheckServiceAffinity
                                                                                           true,否则返回false
                                                                                      1.判断备选Pod所用的端口列表中的端口是否在备选节点已被占用,如果被占
                                                                                     用,则返回false,否则返回true
                                                                        7.PodFitsPorts
                                                                       1.每个节点通过优先选择策略时都会计算出一个得分,计算各项得分,最终选
                                                                        出得分值最大的节点作为优选结果
                                                                                             1.该优选策略用于从备选节点列表中选出资源消耗最小的节点
3.Scheduler原理解析
                                                                                            2.计算出在所有备选节点上运行的Pod和备选Pod的CPU占用量totalMillICPU
                                                                       2.LeastRequestedPriority
                                                                                            3.计算出在所有备选节点上运行的Pod和备选Pod的内存占用量totalMemory
                                                                                            4.计算每个节点的得分,计算规则大致如下,其中,NodeCPUCapacity为节
                                                                                             点CPU计算能力,NodeMemoryCapacity为节点内存大小
                                                         4.优选策略介绍
                                                                                               1.该策略用于判断策略列出的标签在备选节点中存在时,是否选择该备选节点
                                                                                               2.如果备选节点的标签在优选策略的标签列表中且优选策略的presence值为
                                                                        3.CalculateNodeLabelPriority
                                                                                               true,或者备选节点的标签不在优选策略的标签列表中且优选策略的
                                                                                               presence值为false,则备选节点score=10,否则备选节点score=0
                                                                                                1.该优选策略用于从备选节点列表中选出各项资源使用率最均衡的节点
                                                                                                2.计算出在所有备选节点上运行的Pod和备选Pod的CPU占用量totalMillICPU
                                                                        4.BalancedResourceAllocation
                                                                                                3.计算出在所有备选节点上运行的Pod和备选Pod的内存占用量totalMemory
                                                                                                4.计算每个节点的得分,计算规则大致如下,其中,NodeCpuCapacity为节
                                                                                                点的CPU计算能力,NodeMemoryCapacity为节点内存大小
                                     1.在kubernetes集群中,在每个Node上都会启动一个kubelet服务进程
                                     2.kubelet用于处理Master下发到本节点的任务,管理Pod及Pod中的容器
                                     3.每个kubelet进程都会在API Server上注册节点自身信息,定期向Master汇
                                     报节点资源的使用情况
                                                 1.节点通过设置kubelet的启动参数"--register-node",来决定是否向API
                                                 Server注册自己
                                     4.节点管理
                                                 2.kubelet在启动时通过API Server注册节点信息,并定时向API Server发送
                                                 节点的新消息,API Server在接收到这些信息后,将这些信息些人etcd
                                                                                  1.文件: kubelet启动参数 "--config" 指定的配置文件目录下的文件 (默认
                                                                                  目录为/etc/kubernetes/manifests)
                                                                                  2.HTTP端点(URL):通过 "--manifest-url" 参数设置。
                                                                                  3.API Server: kubelet通过API Server监听ETCD目录,同步Pod列表
                                                                                                                                 1.为该Pod创建一个数据目录
                                                1.kubelet获取自身Node上运行的Pod清单的方式
                                                                                                                                 2.从API Server读取该Pod清单
                                     5.Pod管理
                                                                                                                                 3.为该Pod挂在外部卷
                                                                                                                                 4.下载Pod用到的Secret
                                                                                                                                 5.检查已经运行在节点上的Pod,如果该Pod没有容器或Pause容器没有启
                                                                                                                                 动,则先停止Pod里所有容器的进程。如果在Pod中有需要删除的容器,则删
                                                                                  4.kubelet读取监听到的信息,如果是创建和修改Pod任务,则做如下处理
```

notos/nausa" 焙烙的怎么Dad都创建——人家哭 按Dausa家!!

1.kubernetes API Server的核心功能是提供Kubernetes各类资源对象(如 Pod、RC、Service等)的增、删、改、查及watch等HTTP Rest接口,称为

)组件运行⁷

0.用 Rubernetes/pause 说像为每一FOU即创建 一台站。以Fause台站 用于接管Pod中所有其他容器的网络 1.为容器计算一个hash值,然后用容器名称去查询对应Docker容器的Hash 值。若找到容器,且二者的Hash值不同,则停止Docker中容器的进程,并停 4.kubelet运行机制解析 止与之关联的Pause容器进程;若两者相同,则不做任何处理 7.为Pod中的每个容器做如下处理 2.如果容器被终止,且容器没有指定的restartPolicy,则不做任何处理 3.调用Docker Client下载容器镜像,调用Docker Client运行容器 1.一类是LivenessProbe探针,用于判断容器是否健康并反馈给kubelet。如 果LivenessProbe探针探测到容器不健康,则kubelet将删除该容器,并根据 容器的重启策略做相应的处理 2.另一类是ReadinessProbe探针,用于判断容器是否启动完成,且准备接收 请求 6.容器健康检查 1.Pod通过两类探针来检查容器的健康状态。 1.ExecAction:在容器内部执行一个命令,如果该命令的退出状态码为0,则表明 2.TCPSocketAction:通过容器IP地址和端口号执行TCP检查,如果端口能被访 3.kubelet定期调用容器的LivenessProbe探针来诊断容器的健康状 况.LivenessProbe包含3种实现方式 问,则表明容器健康 3.HTTPGetAction:通过容器的IP地址和端口号及路径调用HTTP Get方法,如 果响应的状态码大于等于200且小于等于400,则认为容器状态健康 1.cAdvisor是一个开源的分析容器资源使用率和性能特性的代理工具 2.只能提供2~3min的监控数据,对性能数据也没有持久化 7.cAdvisor资源监控 3.在新的Kubernetes监控体系中,Metrics Server用于提供Core Metrics(核心 指标),包括Node和Pod的CPU和内存使用数据.其他Customer Metrics(自定 义指标)则由第三方组件采集和存储 1.Service是对一组Pod的抽象,它会根据访问策略(负载均衡策略)来访问这组 Pod 2.Kubernetes集群的每个Node上都会运行一个kube-proxy服务进程,其核心 功能是将到某个Serveice的访问请求转发到后台多个Pod实例上 8.kube-proxy运行机制解析 3.IPVS专门用于高性能负载均衡,并使用更高效的数据结构(Hash表) 1.为大型集群提供了更好的可扩展性和性能 2.支持比iptable更复杂的复制均衡算法(最小负载、最少连接、加权等) 4.IPVS的优势 3.支持服务器健康检查和连接重试等功能 4.可以动态修改ipset的集合,即使iptable的规则正在使用这个集合