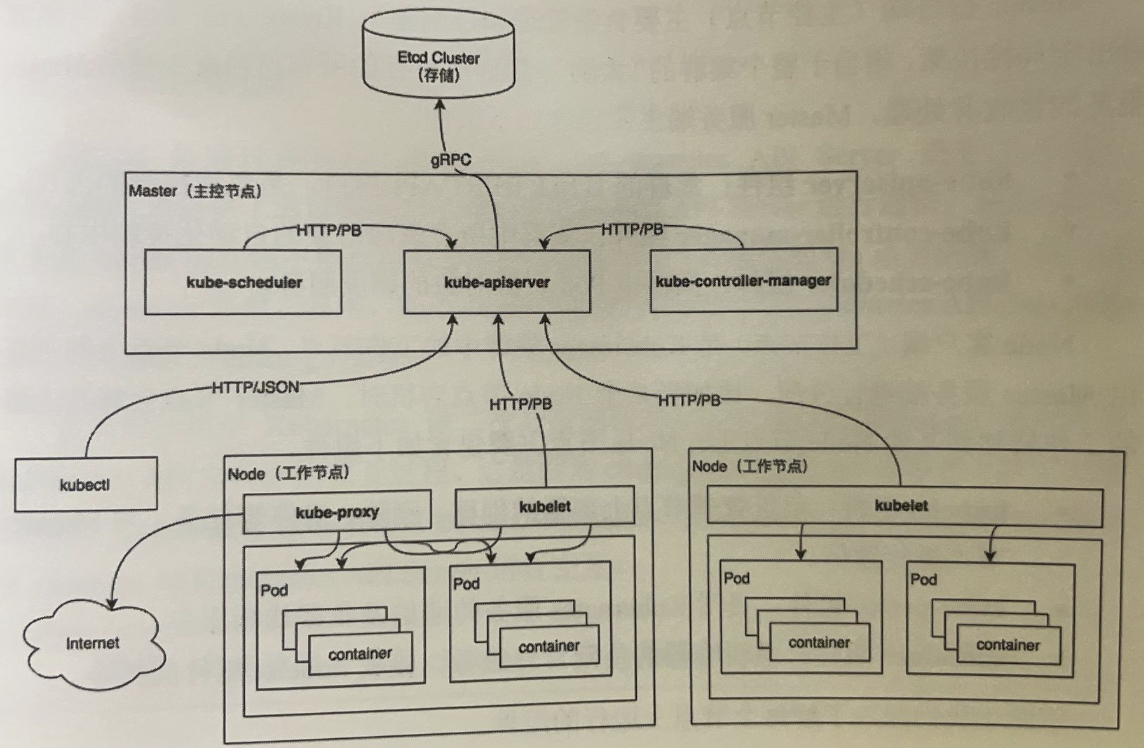
架构

一、架构图



Kubernetes系统架构遵循客户端/服务端(C/S) 架构，系统架构分为Master 和Node两部分，Master 作为服务端，Node作为客户端。Kubernetes系统具有多个Master服务端，可以实现高可用。在默认的情况下，一个Master 服务端即可完成所有工作。

Master服务端也被称为主控节点，主要负责管理和控制整个Kubernetes 集群，对集群做出全局性决策，相当于整个集群的“大脑”。集群所执行的所有控制命令都由Master 服务端接收并处理。

主要包含如下组件：

- [ ] kube-apiserver 组件:集群的HTTP REST API接口，是集群控制的入口。

- [ ] kube controller-manager组件:集群中所有资源对象的自动化控制中心。

- [ ] kube -scheduler组件:集群中Pod资源对象的调度服务。

主要负责如下任务：

(1)集群的“大脑”，负责管理所有节点(Node)。

(2)负贵调度Pod在哪些节点上运行。

(3)负责控制集群运行过程中的所有状态。

Node客户端也被称为工作节点，Node节点上的工作由Master服务端进行分配，比如当某个Node节点宕机时，Master 节点会将其上面的工作转移到其他Node节点上。主要包含如下组件：

- [ ] kubelet组件:负责管理节点上容器的创建、删除、启停等任务，与Master节点进行通信。

- [ ] kube proxy组件:负责Kubernetes服务的通信及负载均衡服务。

- [ ] cntaier组件:负责容器的基础管理服务，接收kubele组件的指令。它在集群中主要负责如下任务。

主要负责如下任务：

(1)负责管理所有容器(Container)。

(2) 负责监控/上报所有Pod的运行状态。

二、各组件的功能

2.1 kubectl

kubectl是Kubermetes官方提供的命令行工具(CLI)，用户可以通过 kubectl 以命令行交互的方式对Kubermetes API Server进行操作，通信协议使用HTTP/JSON。

kubecl发送相应的HTTP请求，请求由Kubernetes API Server接收、处理并将结果反馈给kubectl，kubectl 接收到响应并展示结果。至此，kubectl 与kube-apiserver结的一次请求周期结束。

2.2 client-go

Kubernetes还提供了通过编程的方式与Kubernetes API Server进行通信。client-go 是从Kubernetes的代码中单独抽离出来的包，并作为官方提供的Go语言的客户端发挥 作用。client-go 简单、易用，Kubermetes 系统的其他组件与Kubernetes API Server通信的方式也基于client-go实现。

在大部分基于Kubernetes 做二次开发的程序中，建议通过client-go 来实现与 Kubemetes API Server的交互过程。这是因为client-go在Kubemetes系统上做了大量 的优化，Kubermetes 核心组件(如kube-scheduler kube-controller-manager等)都通 过client-go与Kubernetes API Server进行交互。

2.3 kube-apiserver

kube-apiserver组件，也被称为Kubernetes API Server。它负责将Kubernetes“资源组/资源版本/资源”以RESTful风格的形式对外暴露并提供服务。Kubemetes 集群中的所有组件都通过kube-apiserver组件操作资源对象，kube\_apiserver组件也是集群中唯一与Etcd集群进行交互的核心组件。

Etcd集群是分布式键值存储集群，其提供了可靠的强一致性服务发现。 Etcd 集群存储Kubernetes系统集群的状态和元数据，其中包括所有Kubernetes 资源对象信息、集群节点信息等。Kubenetes将所有数据存储至Eted集群中前缀为/registry的目录下。

kube-apiserver属于核心组件，对于整个集群至关重要，它具有以下重要特性。

- [ ] 将Kubernetes系统中的所有资源对象都封装成RESTful风格的API接口进行管理。

- [ ] 可进行集群状态管理和数据管理，是唯一与Elced集群交互的组件。

- [ ] 拥有丰富的集群安全访问机制，以及认证、授权及准入控制器。

- [ ] 提供了集群各组件的通信和交互功能。

2.4 kube-controller-manager

kube-controller-manager组件，也被称为Controller Manager (管理控制器)，它负责管理Kubemnetes集群中的节点(Node)、Pod 副本、服务、端点(Endpoint)、命名空间(Namespace)、服务账户(ServiceAccount)、资源定额(ResourceQuota) 等。 例如，当某个节点意外宕机时，Controller Manager 会及时发现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。

Controller Manager负责确保Kubernetes系统的实际状态收敛到所需状态，其默认提供了一些控制器(Controller),，例如DeploymentControllers 控制器、StatefulSet控制器、Namespace控制器及PersistentVolume控制器等，每个控制器通过kube-apiserver组件提供的接口实时监控整个集群每个资源对象的当前状态，当因发生各种故障而导致系统状态出现变化时，会尝试将系统状态修复到“期望状态” 。

Controller Manager 具备高可用性(即多实例同时运行)，即基于Eted 集群上的分布式锁实现领导者选举机制，多实例同时运行，通过kube-apiserver提供的资源锁进行选举竞争。抢先获取锁的实例被称为Leader 节点(即领导者节点)，并运行 kube-controller-manager 组件的主逻辑；而未获取锁的实例被称为Candidate节点(即候选节点)，运行时处于阻塞状态。在Leader节点因某些原因退出后，Candidate节点则通过领导者选举机制参与竞选，成为Leader节点后接替kube-controller-manager的工作。

2.5 kube-scheduler

kube-scheduler组件， 也被称为调度器，目前是Kubernetes集群的默认调度器。它负责在Kubernetes集群中为一个Pod资源对象找到合适的节点并在该节点上运行，调度器每次只调度一个Pod资源对象，为每一个Pod资源对象寻找合适节点的过程是一个调度周期。

kube-scheduler组件监控整个集群的Pod资源对象和Node资源对象，当监控到新的Pod资源对象时，会通过调度算法为其选择最优节点。调度算法分为两种，分别为预选调度算法和优选调度算法。除调度策略外，Kubernetes 还支持优先级调度、抢占机制及亲和性调度等功能。

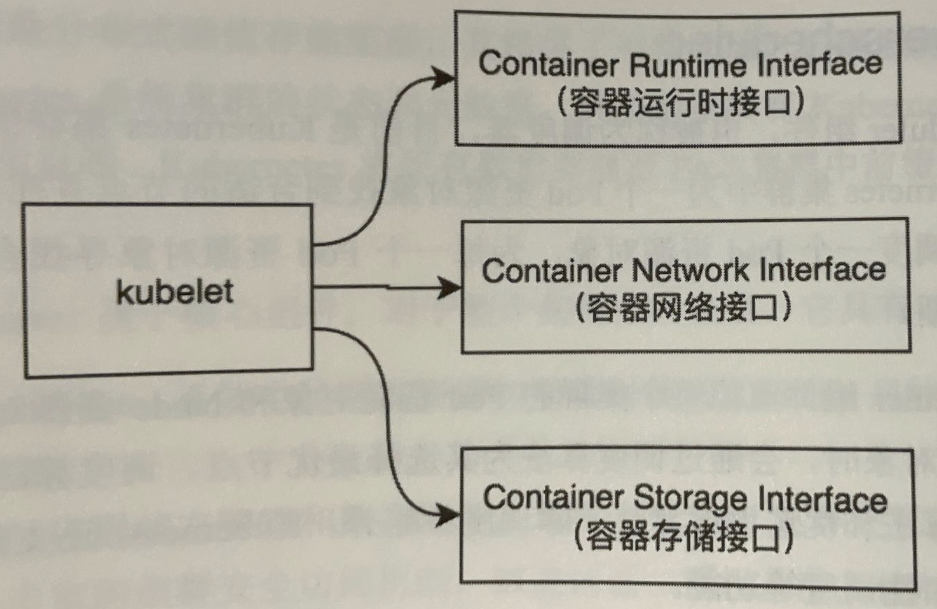
kube-scheduler组件支持高可用性(即多实例同时运行)，即基于Eted集群上的分布式锁实现领导者选举机制，多实例同时运行，通过kube-apiserver提供的资源锁进行选举竞争。抢先获取锁的实例被称为Leader 节点(即领导者节点)，并运行kube-scheduler组件的主逻辑；而未获取锁的实例被称为Candidate节点(即候选节点)，运行时处于阻塞状态。在Leader节点因某些原因退出后，Candidate 节点则通过领导者选举机制参与竞选，成为Leader节点后接替kube- scheduler的工作。

2.6 kubelet

kubelet组件，用于管理节点，运行在每个Kubenetes节点上。kubelet组件用来接收、处理、上报kube-apiserver 组件下发的任务。kubelet 进程启动时会向 kube-apiserver注册节点自身信息。它主要负责所在节点(Node) 上的Pod资源对象的管理，例如Pod资源对象的创建、修改、 监控、删除、驱逐及Pod生命周期管理等。

kubelet组件会定期监控所在节点的资源使用状态并上报给kube-apiserver组件，这些资源数据可以帮助kube-scheduler调度器为Pod资源对象预选节点。kubelet地会对所在节点的镜像和容器做清理工作，保证节点上的镜像不会占满磁盘空间、删除的容器释放相关资源。

kubelet 组件实现了3种开放接口：



- [ ] Container Runtime Interface:简称CRI ( 容器运行时接口)，提供容器运行时通用插件接口服务。CRI定义了容器和镜像服务的接口。CRI将kubelet组件与容器运行时进行解耦，将原来完全面向Pod级别的内部接口拆分成面向Sandbox和Container的gRPC接口，并将镜像管理和容器管理分离给不同的服务。

- [ ] Container Network Interface:简称CNI (容器网络接口)，提供网络通用插件接口服务。CNI 定义了Kubernetes 网络插件的基础，容器创建时通过CNI插件配置网络。

- [ ] Container Storage Interface: 简称CSI (容器存储接口)， 提供存储通用插件接口服务。CSI 定义了容器存储卷标准规范，容器创建时通过CSI 插件配置存储卷。

2.7 kube-proxy

kube-proxy组件，作为节点上的网络代理，运行在每个Kubernetes节点上。它监控kube-apiserver的服务和端点资源变化，并通过iptables/ipvs 等配置负载均衡器，为一组Pod 提供统一的TCP/UDP 流量转发和负载均衡功能。

kube- proxy组件是参与管理Pod-to-Service和External-to-Service网络的最重要的节点组件之一。kube-proxy 组件相当于代理模型，对于某个IP:Port 的请求，负责将其转发给专用网络上的相应服务或应用程序。但是与其他负载均衡区别在于，kube-proxy代理只向Kubernetes服务及其后端Pod发出请求。

三、源码目录结构说明

