## Kubenetes原理分析

### Kubenetes概述

Kubernetes是一个开源的，用于管理云平台中多个主机上的容器化的应用，Kubernetes的目标是让部署容器化的应用简单并且高效,Kubernetes提供了应用部署，规划，更新，维护的一种机制。Kubernetes一个核心的特点就是能够自主的管理容器来保证云平台中的容器按照用户的期望状态运行着。

该项目最初是Google内部面向容器的集群管理系统，而现在由Cloud Native Computing Foundation(CNCF，云原生计算基金会)托管的开源平台。Kubernetes系统拥有一个庞大而活跃的开发人员社区，现在GitHub Star数已达80141，是Go语言最大的开源项目之一，Kubernetes也被称为K8s，是通过将8个字母ubernete替换为8而形成的缩写。

### Kubernetes发展历程

2003-2004年，Google发布了Borg系统，一个内部集群管理系统。

2013年左右，Google继Borg系统之后发布了Omega集群管理系统，适用于大型计算集群的灵活、可扩展的调度程序。

2014年左右，Google发布了Kubernetes系统，其是作为Borg的开源版本发布的，同年Microsoft、RedHat、IBM等加入Kubernetes社区。

2015年左右，Google正式发布Kubernetes 1.0，并于Linux基金会合作组建了云原生计算基金会（CNCF），来对抗以Docker公司为核心的容器商业生态。

2016年左右，Kubernetes成为主流。

2017年左右，互联网巨头纷纷表示支持Kubernetes。

2018年左右，无人不知Kubernetes。

2014年至2015年左右，在如火如荼的容器化浪潮里，一度出现容器编排“三国鼎立”的局面，不过最终Kubernetes胜出，编排之争拉下帷幕。

### Kubenetes 技术架构

Kubernetes是利用共享网络将多个物理机或者虚拟机组成一个集群，在各个服务器之间进行通信,整体架构如图1所示。一个Kubernetes集群由Master节点和Node节点组成。Master是集群的网关和中枢枢纽，主要作用：暴露API接口，跟踪其他服务器的健康状态、以最优方式调度负载，以及编排其他组件之间的通信。单个的Master节点可以完成所有的功能，但是考虑单点故障的痛点，生产环境中通常要部署多个Master节点，组成高可用集群。Node节点是Kubernetes的工作节点，负责接收来自Master的工作指令，并根据指令相应地创建和销毁Pod对象，以及调整网络规则进行合理路由和流量转发。生产环境中，Node节点可以有N个。

Kubernetes从宏观上看分为2个角色Master节点和Node节点，但是在Master节点和Node节点上都存在着多个组件来支持内部的业务逻辑，其包括：运行应用、应用编排、服务暴露、应用恢复等等，在Kubernetes中这些概念被抽象为Pod、Service、Controller、Deployment、Secret等资源类型。

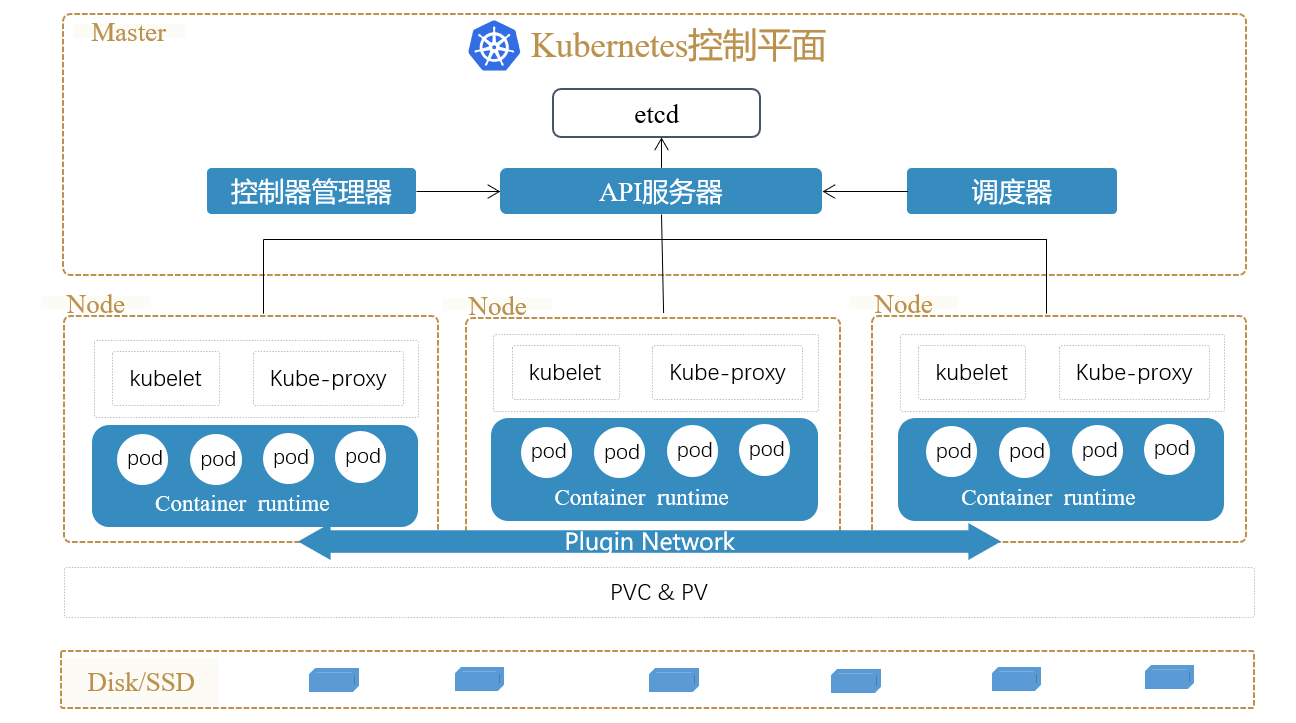


图1 Kubenetes 技术架构

在图一架构图里，API服务器:负责将K8s“资源组/资源版本/资源”以HTTP REST风格的形式对外暴漏提供API接口及服务。Etcd:分布式键值存储K8s系统集群的状态和元数据（包含对象信息、节点信息等），仅与API服务器交互。控制器管理器：负责管理集群中的节点、Pod副本、服务、命名空间、端点、资源定额等，根据每种资源类型提供对应资源看控制器，确保资源实际状态收敛到所需状态。调度器：为集群中的pod资源找到合适的节点部署并在该节点上运行。Kubelet：管理节点，用来接收、处理、上报API服务器组件下发的任务，实现了3种开放接口CNI（容器网络接口）、CRI（容器运行时接口）、CSI（容器存储接口）。Kube-proxy：节点上的网络代理，通过iptables为一组pod提供统一的TCP/UDP流量转发和负载均衡功能。

## KubeEdge原理分析

### KubeEdge概述

KubeEdge是一个开源的系统，可将本机容器化应用编排和管理扩展到边缘端设备。它构建在Kubernetes之上，为网络和应用程序提供核心基础架构支持，并在云端和边缘端部署应用，同步元数据。100%兼容K8s API，可以使用K8s API原语管理边缘节点和设备。KubeEdge还支持MQTT协议，允许开发人员编写客户逻辑，并在边缘端启用设备通信的资源约束。

KubeEdge的诞生，是为了弥补Kubernetes在边缘计算上的瓶颈，Kubernetes太过繁重。在Kubernetes集群中，用户在Master节点上通过编写一个Yaml或者Json格式的配置文件，也可以通过命令等请求Kubernetes API创建应用，就直接将应用部署到集群上的各个节点上，该配置文件中还包含了用户想要应用程序保持的状态，从而生成用户想要的环境。

Kubernetes作为容器编排的标准，人们自然会想把它应用到边缘计算上，即通过 Kubernetes在边缘侧部署应用，但是Kubernetes在边缘侧部署应用时遇到了一些问题，例如

1. 边缘侧设备没有足够的资源运行一个完整的Kubelet。
2. 一些边缘侧设备是ARM架构的，然而大部分的Kubernetes发行版并不支持ARM架构。
3. 边缘侧网络很不稳定，甚至可能完全不通，而Kubernetes需要实时通信，无法做到离线自治。
4. 很多边缘设备都不支持TCP/IP协议。
5. Kubernetes客户端（集群中的各个Node节点）是通过List-watch去监听Master节点的API Server中资源的增删改查，List-watch中的Watch是调用资源的Watch API监听资源变更事件，基于HTTP长连接实现，而维护一个TCP长连接开销较大。从而造成可扩展性受限。

为了解决包含但不限于以上Kubernetes在物联网边缘场景下的问题，华为于2018年11月15日上午宣布了KubeEdge开源项目，并在2019年3月捐给CNCF基金会。联合社区力量一起促进Kubeedge的开发。现在GitHub Star数已达4100。

KubeEdge的优点如下：

* 1. 边缘计算

通过在Edge上运行的业务逻辑，可以在生成数据的本地保护和处理大量数据。这减少了网络带宽需求以及边缘和云之间的消耗。提高响应速度，降低成本并保护客户的数据隐私。

* 1. 简化开发

开发人员可以编写基于常规HTTP或MQTT的应用程序，对其进行容器化，然后在云端或边端中的任何位置运行它们中的更合适的一个。

* 1. Kubernetes原生支持

借助KubeEdge，用户可以在Edge节点上编排应用，管理设备并监视应用和设备状态，就像云中的传统Kubernetes集群一样

* 1. 大量的应用

可以轻松地将现有的复杂机器学习，图像识别，事件处理和其他高级应用程序部署和部署到Edge。

### KubeEdge技术架构

KubeEdge架构图如图2所示，其架构分为了云端和边缘端。云端组件CloudCore部署于云端Master节点上，边端组件EdgeCore部署于边缘节点上。云端组件CloudCore主要包含模块CloudHub、EdgeController、DeviceController。边端组件EdgeCore主要包含模块EdgeHub、Edged、EventBus、ServiceBus、DeviceTwin、MetaManager。

在边缘侧，最大的问题就是网络波动、很不稳定，经常会离线，而KubeEdge解决了这一痛点，实现了边缘离线自治和云边协同。首先需要了解的就是云边通信机制，是就衍生出了云端的Cloud Hub与边源端的Edge Hub。这两个模块之间通过Websocket或者Quic通信，相当于建立了一条底层通信隧道，供K8s和其他应用通信。当然，使用什么协议通信不是重点，重点是如何保障当着之间的链路都无法保障的时候，业务不受到影响，这就是MetaManager的要解决的问题了。

CloudHub其实就是通信的Server端，是Controller和Edge端之间的中介。它负责下行分发消息(其内封装了K8s资源事件)，也负责接收并发送边缘节点上行消息至Controllers。其中下行的消息在应用层增强了传输的可靠性，以应对云边的弱网络环境。

EdgeHub：一个Web socket客户端，负责云端与边缘端的信息交互，其中包括将云端的资源变更同步到边缘端及边缘端的状态变化同步到云端。

DeviceTwine：该模块利用Kubernetes Custom Resource Definition(CRD)自定义资源机制，将自定义设备资源信息保存到本地数据库中，并处理基于云端的操作来修改边缘设备的某些属性；同时，将设备基于EventBus模块上报的状态信息同步到本地数据库和云端的中介。

Edged：是管理节点生命周期的边缘节点模块。它可以帮助用户在边缘节点上部署容器化的工作负载或应用程序。这些工作负载可以执行任何操作，从简单的遥测数据操作到分析或ML推理等。使用Kubectl云端的命令行界面，用户可以发出命令来启动工作负载。可理理解为Edged模块就是功能裁剪后的Kubelet。该模块就是保障云端下发的容器组以及其对应的各种配置、存储能够在边端稳定运行，并在异常之后提供自动检测、故障恢复等能力。当然，由于K8s本身运行时的发展，该模块对应支持各种CRI应该也比较容易。

EventBus/ServiceBus/Mappper：消息传递接口。外部设备的接入当前支持MQTT和HTTP协议，这里分别对应EventBus和ServiceBus。EventBus就是一个MQTT Broker的客户端，主要功能是将边端各模块通信的Message与设备Mapper上报到MQTT的Event做转换的组件；而ServiceBus就是对应当外部是Rest-API接入时的转换组件。

MetaManager：该模块后端对应一个本地的数据库，所有其他模块需要与云端通信的内容都会被保存到本地数据库中一份，当需要查询数据时，如果本地数据库中存在该数据，就会从本地获取，这样就避免了与云端之间频繁的网络交互；同时，在网络中断的情况下，本地的缓存的数据也能够保障其稳定运行，在通信恢复之后，重新同步数据。

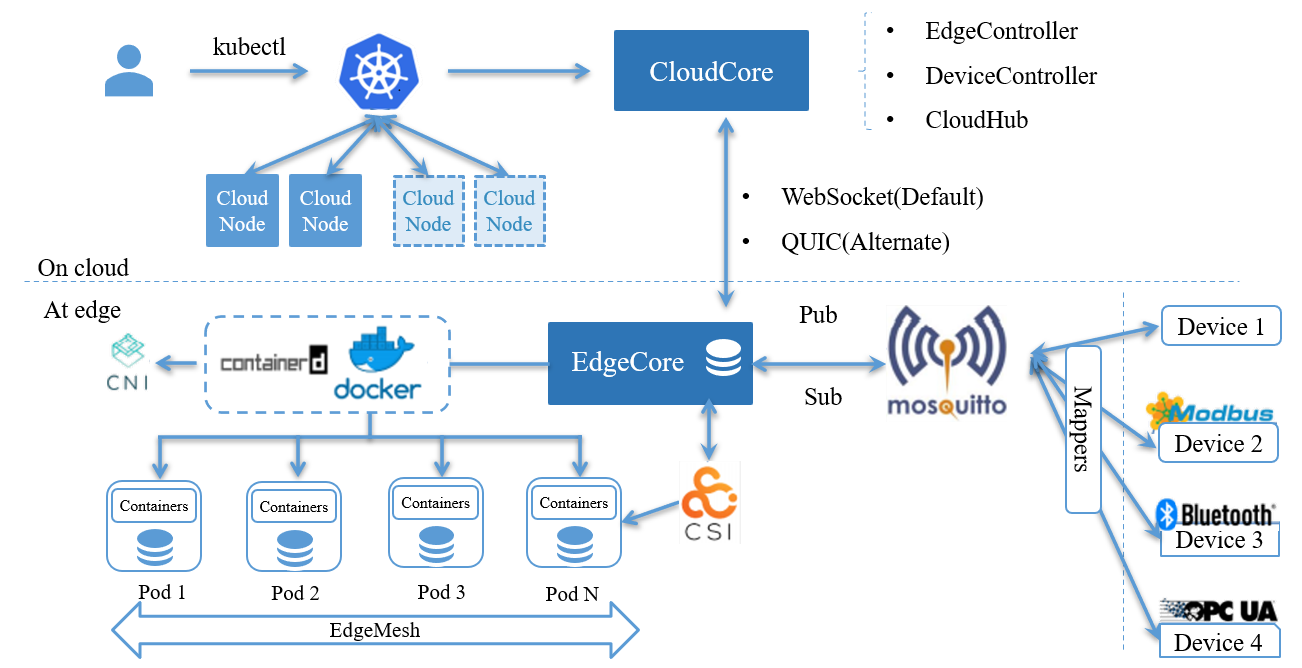


图2：KubeEdge技术架构

在图2 KubeEdge技术架构里，云端，CloudHub：一个Web socket服务器，负责监听云端的更新、缓存及向EdgeHub发送消息。EdgeController：一个扩展的K8s控制器，负责管理边缘节点和容器组元数据，同步边缘节点的数据，是K8s API Server与EdgeCore的通信桥梁。DeviceController：一个扩展的K8s控制器，负责管理节点设备，同步云端和边缘端的设备元数据和状态。

边缘端EdgeHub：一个Web Socket客户端，负责云端与边缘端的信息交互，其中包括将云端的资源变更同步到边缘端及边缘端的状态变化同步到云端。Edged：运行在边缘节点，管理容器化应用的Agent，负责容器组生命周期的管理，类似Kubelet。EventBus：一个MQTT客户端，与MQTT服务端交互，提供发布/订阅的能力。ServiceBus：一个HTTP客户端，与HTTP服务端交互。为云组件提供HTTP客户端功能，以访问在边缘运行的HTTP服务器。DeviceTwin：负责存储设备状态并同步设备状态到云端，同时提供应用的接口查询。MetaManager：Edged和EdgeHub之间的消息处理器，负责向轻量数据库（SQLite）存储或查询元数据。

## K3s原理分析

### K3s概述

k3s 是一个轻量级Kubernetes，它易于安装，二进制文件包小于40M，只需512MB RAM即可运行,非常适用于边缘计算-Edge、物联网-IoT、CI、ARM、Development等场景。而且由于运行K3s所需的资源相对较少，所以K3s也适用于开发和测试场景。

K3s最大的优点就在于轻量级，而且初衷是希望安装的Kubernetes在内存占用方面只是一半的大小，所以也形象的命名为K3s，Kubernetes是一个10个字母的单词，简写 K8s。所以，有Kubernetes一半大的东西就是一个5个字母的单词，简写为K3s。目前K3s GitHub Star数已达17596。

K3s有以下增强功能：

1. 打包为单个二进制文件。
2. 使用基于Sqlite3的轻量级存储后端作为默认存储机制。同时支持使用Etcd3、MySQL和PostgreSQL作为存储机制。
3. 封装在简单的启动程序中，通过该启动程序处理很多复杂的TLS和选项。
4. 默认情况下是安全的，对轻量级环境有合理的默认值。
5. 添加了简单但功能强大的Batteries-included功能，例如：本地存储提供程序，服务负载均衡器，Helm controller 和 Traefik Ingress controller。
6. 所有Kubernetes control-plane 组件的操作都封装在单个二进制文件和进程中，使K3s具有自动化和管理包括证书分发在内的复杂集群操作的能力。
7. 最大程度减轻了外部依赖性，K3s仅需要Kernel和Cgroup挂载。

K3s软件包需要的依赖项包括Containerd、Flannel、CoreDNS、CNI、主机实用程序（Iptables、Socat 等）、Ingress controller（Traefik）、嵌入式服务负载均衡器（Service load balancer）、嵌入式网络策略控制器（Network policy controller）

### K3S技术架构

K3s的技术架构如图3所示，乍一看，貌似和Kubernetes架构图类似，控制节点都有API服务器、调度器、控制器管理器，工作节点有kubelet、kube-proxy、Flannel插件。但在K3s集群中，将运行控制平面组件与Kubelet的节点称为Server，而只运行Kubelet的节点称为Agent。server和agent都有容器运行时和一个Kube-proxy，管理整个集群的Tunnel和网络流量。之所以在K3s集群中不再命名为Master节点和Worker节点是因为K3s中的Master节点和Worker节点没有明显的区别。可以在任何节点上调度和管理容器组，而且一旦传递了Server的URL，节点就会变成一个Agent。

正如前文所述，K3s的二进制文件包只有几十M大小，这款迷你Kubernetes发行版针对边缘进行了高度优化及对上游进行了极大删减，具体体现如下：

1. 旧的、Alpha版本的、非默认功能都已经删除。
2. 删除了大多数内部云提供商和存储插件，可以用插件替换。
3. 新增SQLite3作为默认存储机制，Etcd3仍然有效，但是不再是默认项。
4. 封装在简单的启动器中，可以处理大量LTS复杂性和选项。
5. 最小化到没有操作系统依赖，只需要一个内核和Cgroup挂载。

上游的Kubernetes发行版是臃肿的，且很多代码都可以删除，一些没必要的插件也没必要保留。Kubernetes依靠分布式键值数据库Etcd来存储整个集群的状态，而K3s用名为SQLite的轻量级数据库取代了Etcd，SQLite是一个成熟的嵌入式场景数据库。很多移动应用都会捆绑SQLite来存储状态。虽然SQLite并不是分布式数据库，但K3s Server可以指向外部数据库端点。K3s的另一优点就是“包含电池但可替换”的方式。例如，我们可以用Docker CE运行时替换Containerd运行时，用Calico替换Flannel，用Longhorn替换本地存储等等。

K3s发行版支持多种架构，包括AMD64、ARM64。凭借一致的安装体验，K3s可以在Raspberry Pi Zero、NVIDIA Jetson Nano、Intel NUC或Amazon EC2 a1.4xlarge实例上运行。

如果你需要一个在AMD64或ARM64架构上运行的高可用集群，安装一个3节点的Etcd集群，然后是3个K3s Server和一个或多个Agent。这样就可以为你提供一个生产级的环境，并为控制平面提供HA。

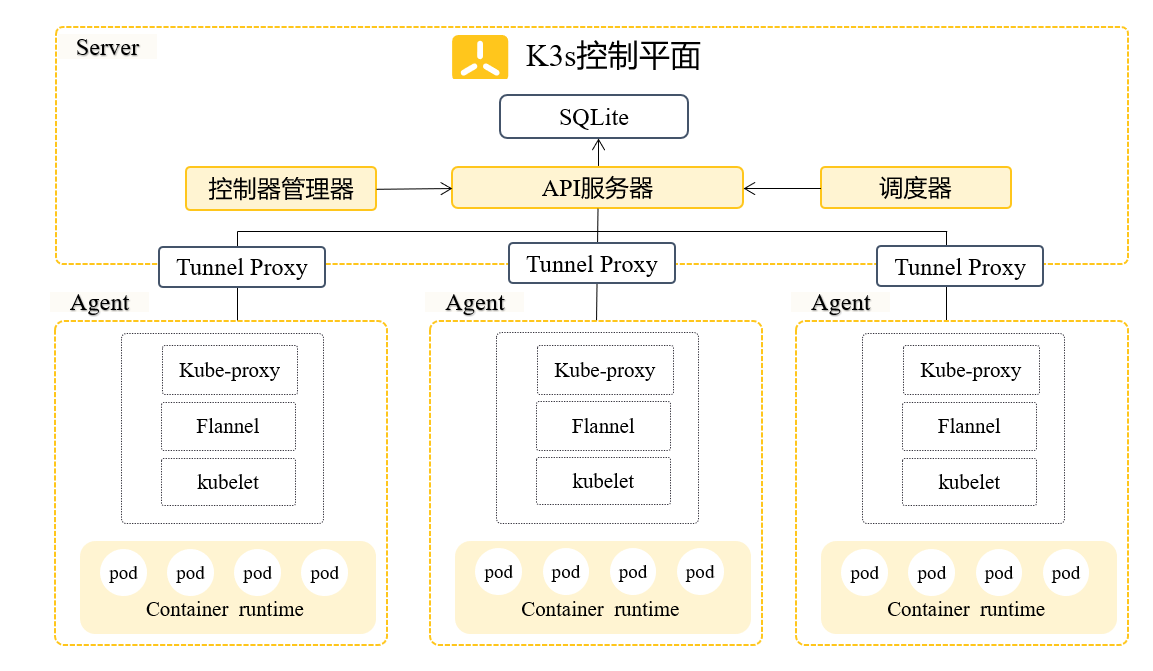


图3：K3s技术架构