<https://www.cnblogs.com/goldsunshine/p/9872142.html>

当下云计算的领域里热度最高的两个项目，无疑是OpenStack和Kubernetes。如果云计算是一个风起云涌的江湖，毫不夸张的说OpenStack和Kubernetes就是江湖里的泰山北斗。OpenStack就像是少林，基础扎实、沉稳厚重，而Kubernetes就是武当，轻巧空灵、飘逸精妙。使用过这两种系统的人都应该有这样的感受，OpenStack出身于虚拟化技术，稳定但速度慢，Kubernetes则来自于容器技术，快速但有局限。两种不同的技术就决定了有着不同的人生轨迹。那么究竟两者有着怎样的际遇呢？我们分析分析。

**出身对比：**

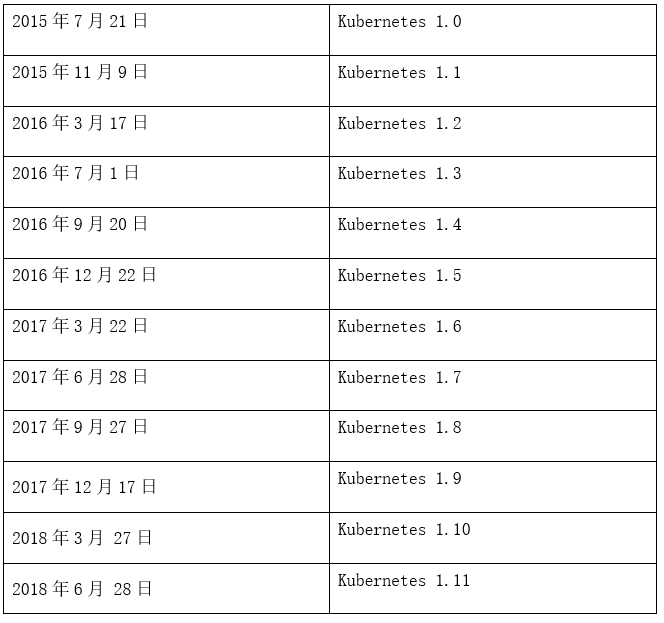
**OpenStack：**

2010年7月，RackSpace公司和美国国家航空航天局NASA合作，分别贡献出了RackSpack云文件平台代码和NASA平台代码，发布OpenStack的第一个版本Austin。2010的Rackspace是美国第二大云计算厂商，但规模只能占到亚马逊的5%。只依靠内部的力量来超越或者追赶亚马逊不大可能，这家公司就把自己的项目开源了，也就是后来的 OpenStack 的存储源码（swift）。与此同时NASA也对自己使用的 Eucalyptus 云计算管理平台很不爽。NASA想给Eucalyptus开源版本贡献，结果Eucalyptus不接受。当时NASA 的六个开发人员，用了一个星期时间拿Python做出来一套原型，结果虚拟机在这上面运行的很成功，这就是Nova（计算源码）的起源。Austin只有swift和Nova这两个项目，即目前的对象存储和计算服务。此后OpenStack大概保持着每半年发布一次版本的频率，截止到目前最新的版本是Rocky。在最新的版本中项目已经达到60多个。

**Kubernetes：**

Kubernetes是Google在2014年发布的一个开源项目。Google开发了一个叫Brog的系统来调度内部数量庞大的容器和工作负载。在积累了多年经验之后，Google决定重写这个容器管理，并将其贡献到开源社区，让全世界都能够受益。在2014年第一个版本发布以来，Kubernetes迅速受到开源社区的的追捧，目前Kubernetes已经成为发展最快，市场占有率最高的容器编排引擎。截止到现在，Kubernetes的最新版本是1.11版本。

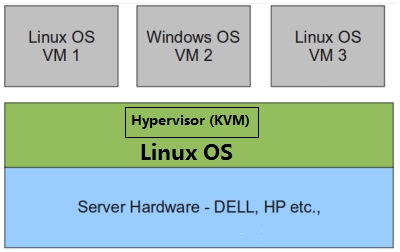
Kubernetes版本发布表：

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/biao1.jpg)

**技术实现**

**OpenStack：虚拟化**

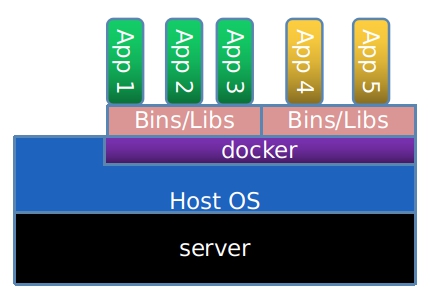
OpenStack作为一个开源的云计算平台，利用虚拟化技术和底层存储服务，提供了可扩展，灵活，适应性强的云计算服务。虚拟化是云计算的基础。简单的说，虚拟化使得在一台物理的服务器上可以跑多台虚拟机，虚拟机共享物理机的CPU、内存、IO硬件资源，但逻辑上虚拟机之间是相互隔离的。宿主机一般使用hypervisor程序实现硬件资源虚拟化，并提供给客户机使用。如下图所示是一种虚拟化的架构。

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/2.jpg)

每一个虚拟机都拥有自己的内核和文件系统，完全是一个独立的操作系统。而上图是两种虚拟化方式中的其中一种：半虚拟化——KVM。在目前的环境中，KVM虚拟化技术是使用率最高的技术。  
虚拟化优点:隔离性强，所有的虚拟机都有自己的协议栈，各个虚拟机底层相互隔离。  
虚拟化缺点：资源占用多，虚拟化技术本身占用资源，宿主机性能有10%左右的消耗。

**Kubernetes：docker**

Kubernetes是容器管理编排引擎，那么底层实现自然是容器技术。容器是一种轻量级、可移植、自包含的软件打包技术，打包的应用程序可以在几乎任何地方以相同的方式运行。以容器典型代表docker为例，docker起源于2013年3月，是基于LXC为基础构建的容器引擎，通过namespace和cgourp实现了资源隔离和调配，使用分层存储来构建镜像。它基于Google公司推出的Go语言实现。docker相比KVM虚拟化技术最明显的特点就是启动快，资源占用小。虚拟化启动虚拟机是分钟级别的，而docker是秒级别的。如下是docker的架构图。

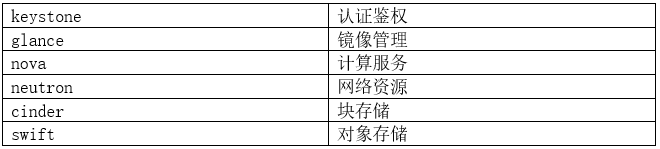
[](https://img1.sdnlab.com/wp-content/uploads/2018/09/3.jpg)

docker的启动速度快，占用资源少，原因在于技术架构：  
一个操作系统分为内核+文件系统。容器技术就是使用宿主机的内核系统加上自身的文件系统。运行容器时是在使用宿主机的内核情况下加载文件系统，精简的文件系统可以小到100MB以内，所以比虚拟机自然要快很多。可以将容器看作是在内核上运行的独立代码单元，它们非常轻。因此占用的资源也少。  
容器优点：启动快，资源占用小，移植性好  
容器缺点：隔离性不好，共用宿主机的内核，底层能够访问。依赖宿主机内核所以容器的系统选择有限制。

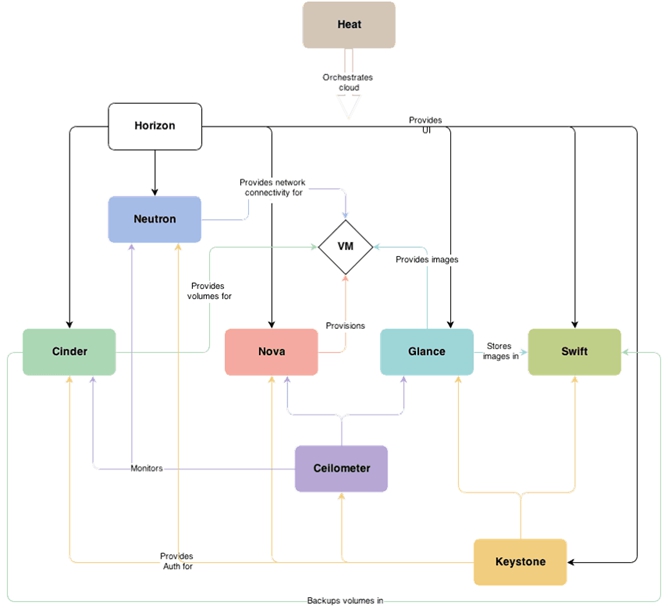
**架构对比**

**OpenStack：**

OpenStack的服务分为核心功能和非核心功能。核心功能是指运行OpenStack系统必须的功能，其中核心功能有：

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/biao2.jpg)

其工作模式如下：

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/4_668.jpg)

在“为什么选择OpenStack”的用户调查中得出一个比例很高的结果是：开放平台和标准化的API。OpenStack贯彻松耦解耦的思想，各个服务之间使用标准的API接口调用，并且这些接口是能够开发给非OpenStack程序去调用。

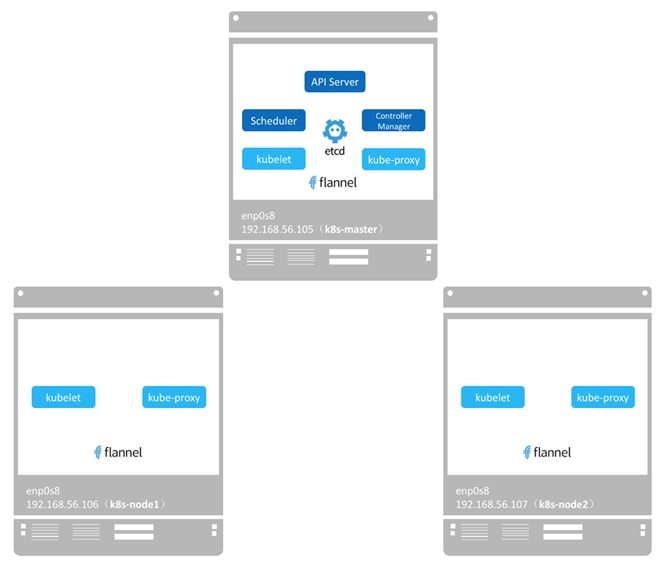
具体到OpenStack就是Resutful（表述性状态转移）和RPC（远程过程调用）。服务与服务之间使用Restful API通信，最大程度的减少了服务之间的依赖。例如创建虚拟机时，nova服务要调用glance服务，要调用neutron服务，这些都是通过Restful api 来完成的。服务内部的模块之间的调用使用了RPC，增加了横向扩展能力。例如nova-api接收到创建虚拟机的请求，要先后调用nova-scheduler 选定创建虚拟机的主机，nova-compte完成虚拟机创建的具体工作。此外，opnestack用到的通用技术还有：

1. 消息总线 AMQP  
2. ORM模型数据库 SQLalchemy  
3. WSGI Web服务器网管接口  
4. Eventlet 协程

OpenStack采用开源技术，避免重复制造轮子，这对团队的技术选择有着借鉴意义。

**Kubernetes：**

Kubernetes的思想是尽量保证用户的理想状态。通俗来说就是用户创建了3个容器，Kubernetes要保证这三个容器的生命，时时刻刻都是健康的三个容器，受到断电等故障的情况能够及时补上。Kubernetes是由Master和Node组成，Master是大脑，Node是计算节点。  
如下图的构成：

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/52_668.jpg)

在Master节点上运行的服务有：

1. API Server：提供Restful api。各种客户端工具或者其他组件可以调用其完成资源调用。  
2. Scheduler：调度服务，决定将容器创建在哪个Node上。  
3. Controller Manager：管理系统中各种资源，保证资源处于预期的状态。  
4. Etcd:保存系统的配置信息和各种资源的状态信息。  
5. Pod网络：可以是macvlan、flannel、weave、calico等其中的一种。

Node节点的服务：  
  
1. kubelet ：接收Master节点发来的创建请求信息，并向Master报告运行状态。  
2. kube-proxy ：访问控制。

同样以创建一个服务的方式来解析整个系统的运作流程。Kubernetes 客户端发送创建请求到系统，API server接收到请求，并通知controller创建一个deployment资源，controller负责具体的创建过程，调用Scheduler选择哪个主机创建，然后将请求发往Node节点，Node节点上的kubelet接收到请求，创建具体的docker。

Kubernetes同样遵循标准化API接口。

Kubernetes API是集群系统中的重要组成部分，Kubernetes中各种资源（对象）的数据通过该API接口被提交到后端的持久化存储（etcd）中，Kubernetes集群中的各部件之间通过该API接口实现解耦合，同时集群中一个重要且便捷的管理工具kubectl也是通过访问该API接口实现其强大的管理功能的。系统中大多数情况下，API定义和实现都符合标准的HTTP REST格式，比如通过标准的HTTP动词（POST、PUT、GET、DELETE）来完成对相关资源对象的查询、创建、修改、删除等操作。但同时Kubernetes 也为某些非标准的REST行为实现了附加的API接口，例如Watch某个资源的变化、进入容器执行某个操作等。

**使用场景**

**OpenStack：**

场景一：安全和隔离。OpenStack适用于搭建私有云以及基于私有云的使用的场景。OpenStack底层使用了虚拟化技术，其基因中就有着隔离性好，稳定，部署灵活等特点。在OpenStack的成功案例中，云桌面是典型的例子。有不少的企业都已经将自己的生产环境搬到云端，例如企业上云，工作环境就是使用云桌面的形式。第一是降低了设备成本，上云之前是每人一台主机，到现在几十个人使用一台服务器，如果考虑cpu，内存使用率，成本肯定降下来了。第二是安全，所有的数据都不是存储在身边，在一些安全系数高的行业中尤为重要。OpenStack一直受到金融行业的青睐，这里少不了看中OpenStack安全的特性。

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/62_668.jpg)

场景二：提供基础设施。OpenStack是定位于laas平台的项目，其优点是能够提供虚拟机这种很底层的设施。如果在业务场景中很依赖虚拟机，例如编译内核，或者驱动开发等这些场景，那么OpenStack是很好的选择。

场景三：存储需求。存储是OpenStack另一个优势所在。OpenStack第一个版本的项目组成就是存储和计算，在后期不断的开发中，存储作为一个重要的功能一直不断的完善和创新。如cinder块存储，ceph共享存储能。在存储需求很大的场景下，OpenStack能够提供高效，安全的存储方案，这也是为什么电信行业看好OpenStack的一个原因。

场景四：动态数据场景。即不需要反复地创建和销毁这些服务的运行环境。虚拟机优势在于稳定，那么OpenStack优势也在于运行稳定的项目。

**Kubernetes：**

场景一：Kubernetes适用于业务变化快，业务量未知的静态使用场景。所谓静态使用场景是指在其创建的容器中不会实时产生数据的场景。例如：网站架构，一次部署，长时间使用。特别是遇到一些线上业务量不确定的场景，Kubernetes能够动态扩展，灵活伸缩，从5W的并发量到10W的并发量，完全可以秒级处理。

场景二：需要反复地创建和销毁这些服务的运行环境。docker的优势就在于启动快速，消耗资源小。所以在需要频繁创建和销毁的场景中，Kubernetes是一个不错的选择。

场景三：需要业务模块化和可伸缩性：容器可以很容易地将应用程序的功能分解为单个组件，符合微服务架构的设计模式。

场景四：应用云化。将已有应用、要新开发的应用打造成云原生应用，发挥云平台的可扩展、弹性、高可用等特性，并借助PaaS层提供的API实现更高级的特性，比如自动恢复、定制化的弹性伸缩等。

场景五：微服务架构和API管理。服务拆分来抽象不同系统的权限控制和任务，以方便业务开发人员通过服务组合快速的创建企业应用。有的企业在没有对应的管理平台之前就已经将应用拆分成很多服务，如何部署这些微服务和进行API权限控制，则成了需要解决的问题，而Kubernetes代表的PaaS则是理想的选择。

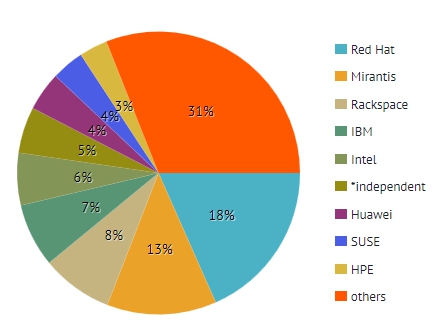
**社区对比**

对于开源项目来说，社区火热程度代表着生命力和潜力。如何判断一个项目的未来发展，关注社区肯定是最基本的要求。

**OpenStack：**

OpenStack是开源项目的代表作之一。

OpenStack从第一个版本开始到现在的R版本的开发过程，为探索开源生态交出一份高分数的答卷，其生态环境做的已经很完善，运作模式可以当做其他开源项目的典范。最明显的标志是每个发行版本的代码贡献量。代码的贡献量是衡量某个企业实力的重要标准，代表开源社区的话语权，更代表着为自身争取权益的能力。所以拥抱OpenStack的企业花费人力物力为社区代码做出贡献。

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/7.jpg)

另一方面OpenStack的出现大大加速了IT架构演进进程。社区对于OpenStack开发流程的把控是十分有效的，无论是代码质量保证，还是测试如单元测试和集成测试都是值得借鉴的。

**Kubernetes：**

Kubernetes社区起步晚于OpenStack，目前尚没有OpenStack的火热，但是Kubernetes在中国开发中的普及度还是很高的。Kubernetes中文社区为国内的爱好者提供了教程，中文文档，安装教程等，大大减少了国内用户的开发使用难度。

**融合**

[](https://img1.sdnlab.com/wp-content/uploads/2018/09/82_668.jpg)

虽然说OpenStack和Kubernetes是云计算领域里两个领导者，那么两者一定是水火不容吗？其实恰恰相反，两者一直积极的相互融合当中。OpenStack中可以集成Docker，目前有三种方案：

1. Docker Driver for Nova  
2. Docker Plugin for Heat  
3. Magnum

OpenStack来部署Kubernetes是另一种融合的方式,很多公司已经实现将 Kubernetes部署到OpenStack中。反过来使用docker来部署OpenStack的服务也是一个很成功的部署方式。在需要频繁部署OpenStack环境的场景下，docker可以做到分钟级别的部署实施，大大减少了部署的困难度和耗时。

从长远来看，两者之间的融合趋势不可避免。

**总结**

OpenStack是定位于laaS平台的项目，Kubernetes是定位于PaaS平台的项目，两者在自己的领域中已经做的很好了。如果说OpenStack不如Kubernetes灵活，那么同样Kubernetes不如OpenStack沉稳。就像说武当功夫基础肯定强不过少林，而少林拳脚没有武当功夫将讲究悟性。事实上根据业务需求，懂得灵活使用这两种不同风格的系统才是制胜之道。

通过对两种系统的出身，技术架构，使用场景和社区对比，希望能在选择上给读者一些有益的借鉴。