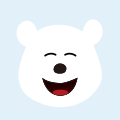
kubernetes（K8S）快速安装与配置集群搭建图文教程

[](https://idc.wanyunshuju.com/)

0人评论[1305人阅读2019-10-28 20:37:54](javascript:;)

更多教程: <https://idc.wanyunshuju.com/K8S/583.html>

**Kubernetes是什么？**

首先，它是一个全新的基于容器技术的分布式架构领先方案。这个方案尽然很新，但它是谷歌十几年以来大规模应用容器技术的经验积累和升华的一个重要成果。确切地说，kubernetes是谷歌严格保密十几年的秘密武器Borg的一个开源版本。Borg是谷歌的一个久负盛名的内部使用的大规模集群管理系统，它基于容器技术，目的是实现资源管理的自动化，以及跨多个数据中心的资源利用率的最大化。  
2015年4月，传闻许久的Borg论文伴随kubernetes（K8S）的高调宣传被谷歌首次公开，大家才得以了解它的更多内幕。正是由于站在Borg这个前辈的肩膀上，吸取了Borg过去十年间的经验与教训，所以kubernetes一经开源就一鸣惊人，并迅速称霸了容器技术领域。  
   
如果我们的系统设计遵循了kubernetes的设计思想，那么传统系统架构中哪些和业务没有多大关系的底层代码或功能模块，都可以立刻从我们的视线中消失，我们不必再费心于负载均衡的选型和部署实施问题，不必再考虑引入或自己开发一个复杂的服务治理框架，不必再头疼与服务监控和故障处理模块的开发。总之，使用kubernetes提供的解决方案，我们节约了不少于30%的开发成本，同时可以将精力更加集中于业务本身，而且由于kubernetes提供了强大的自动化机制，所以系统后期的运维难度和运维成本大幅降低。  
 

**Kubernetes概述**

　　随着 [Docker](https://idc.wanyunshuju.com/Dr/) 技术的发展和广泛流行，云原生应用和容器调度管理系统也成为 IT 领域大热的词汇。事实上，在 Docker 技术火爆之前，云原生应用的思想已经由云计算技术的领导者和分布式系统架构的推广者广泛传播。例如早在 2011 年 Heroku 的工程师提出了云原生应用的 12 要素，只不过以虚拟机技术作为云原生应用的基础实施。由于虚拟机镜像大、镜像标准不统一以及打包流程和工具不统一，导致了业界无法广泛接受的云原生应用标准，限制了云原生应用的流行。而 Docker 的出现正好解决了这些限制云原生应用构建、交付和运行的瓶颈，使得构建云原生应用成为了使用 Docker 的开发者自然而然的选择。  
　　Kubernetes 是为生产环境而设计的容器调度管理系统，对于负载均衡、服务发现、高可用、滚动升级、自动伸缩等容器云平台的功能要求有原生支持。由于 Kubernetes 在K和s间有8个字母，因此常简称 k8s。事实上，随着对 k8s 系统架构与设计理念的了解深入，会发现 K8s 系统正是处处为运行云原生应用而设计考虑;同时，随着对 k8s 系统使用的加深和推广，也会有越来越多有关云原生应用的设计模式产生出来，使得基于 k8s 系统设计和开发生产级的复杂云原生应用变得像启动一个单机版容器服务那样简单易用。

**为什么要用 Kubernetes**

　　使用 Kubernetes 的理由有很多，最基本的一个理由就是：IT 是新技术驱动行业。Docker 这个新兴的容器化技术当前已经被很多公司所采用，其从单机走向集群已成必然，而云计算的蓬勃发展正在加速这一进程。Kubernetes 作为当前唯一被业界广泛认可和看好的 Docker 分布式系统解决方案。可以预见，在未来几年内，会有大量的新系统选择它，不管是运行在企业本地服务器上还是被托管到公有云上。

**使用 Kubernetes 又会收获哪些好处呢?**

　　首先，使用 Kubernetes 就是在全面拥抱微服务架构。微服务架构的核心就是将一个巨大的单体应用分解为很多小的互相连接的微服务，一个微服务背后可能有多个实例副本在支撑，副本的数量可能会随着系统的负荷变化而进行调整，内嵌的负载均衡器在 k8s 平台中有多个实例副本在支撑，副本的数量可能会随着系统的负荷变化而进行调整，内嵌的负载均衡器 k8s 平台中发挥了重要的作用。微服务架构使得每个服务都可以由专门的开发团队来开发，开发者可以自由选择开发技术，这对于大规模团队来说很有价值。另外，每个微服务独立开发、升级、扩展，使得系统具备很高的稳定性和快速迭代进化能力。  
　　其次，Kubernetes 系统架构具备了超强的横向扩容能力。对于互联网公司来说，用户规模就等价于资产，谁拥有更多的用户，谁就能在竞争中胜出，因此超强的横向扩容能力是互联网业务系统的关键指标之一。不用修改代码，一个 Kubernetes 集群即可从只包含几个Node 的小集群平滑扩展到拥有成百上千 Noder 大规模集群，利用 Kubernetes 提供的工具，甚至可以在线完成集群的扩容。只要微服务设计的得好，结合硬件或者公有云资源的线性增加，系统就能够承受大量用户并发访问所带来的压力。

**Kubernetes 基本概念和术语**

　　Kubernetes中的大部分概念如Node、Pod、Replication Controller、Server、Deploymeng、Job、DaemonSet 等都可以看作一种资源对象，几乎所有的资源对象都可以通过 Kubernetes提供的 kubectl 工具执行增、删、改、查等操作，并将其保存在 etcd 中持久化存储。从这个角度来看，Kubernets 其实是一个高度自动化的资源控制系统，它通过跟踪对比 etcd 存储里保存的资源期望状态与当前环境中的实际资源状态的差异来实现自动控制和自动纠错的高级功能。  
　　1) Master  
　　Kubernetes 里的 Master 指的是集群控制节点，每个 Kubernetes 集群里需要有一个Master 节点来负责整个集群的管理的控制，基本上 Kubernetes 所有的控制命令进是发给Master，Master 负责具体的执行过程，后面所有执行的命令基本都是在 Master 节点上运行的。Master 节点通常会占据一个独立的 X86 服务器，一个主要的原因是它太重要了，它是整个集群的大脑，如果它宕机或者不可用，那么所有的控制命令都将失效。  
　　Master 节点上运行着以下一组关键进程：

* Kube-apiserver：提供了 HTTP Rest 接口的关键服务进程，是 Kubernetes 里所有资源的增、删、改、查等操作的唯一入口，也是集群控制入口进程。
* Kube-controller-manager：是 Kubernetes 里所有资源对象的自动化控制中心，可以理解为资源对象的管家。
* Kube-scheduler：负责资源调度的进程，相当于公交公司的调度室。

　　其实 Master 节点上往往还启动了一个 etcd server 进程，因为 Kubernetes 里的所有资源对象的数据全部是保存在 etcd 中的。  
  
　　2) Node  
　　除了 Master、Kubernetes 集群中的其他机器被称为 Node 节点，在较早的版本中也被称为 Minion。与 Master 一样，Node 节点可以是一台物理主机，也可以是一台虚拟机。Node节点才是 Kubernetes 集群中的工作负载节点，每个 Node 都会被 Master 分配一些负载。当某个 Node 宕机时，其上的工作负载会被 Master 自动转移到其他节点上去。  
  
　　每个 Node 节点都运行着以下一组关键进程：

* Kubelet：负责 Pod 对应的容器的创建、启停等任务，同时与 Master 节点密切协作，实现集群管理的基本功能。
* Kube-proxy：实现 Kubernetes Service 的通信与负载均衡机制的重要组件。

　Pod：是kubernetes最重要也是最基本的概念。每个Pod都会包含一个 “根容器”，还会包含一个或者多个紧密相连的业务容器。

　　Kubernetes为每个Pod都分配了唯一的IP地址，称之为PodIP，一个Pod里的多个容器共享PodIP地址。要求底层网络支持集群内任意两个Pod之间的直接通信，通常采用虚拟二层网络技术来实现(Flannel)。

**k8s集群部署方案**

　　如下是我的集群部署策略，1个master + 2个node(minion1.2之前的叫法)。我的存储集群etcd是单点集群，不推荐此做法。网络使用的是flannel虚拟二次网络。

   
   
**Kubernetes具有完备的集群管理能力：**

1. 包括多层次的安全防护和准入机制
2. 多租户应用支撑能力
3. 透明的服务注册和服务发现机制
4. 内建智能负载均衡器
5. 强大的故障发现和自我修复能力
6. 服务滚动升级和在线扩容能力
7. 可扩展的资源自动调度机制
8. 以及多粒度的资源管理能力

同时，kubernetes提供了完善的管理工具，这些工具涵盖了包括开发、部署测试、运维监控在内的各个环节。  
   
在kubernetes中，service（服务）是分布式集群架构的核心，一个service对象拥有如下关键特征：

1. 拥有一个唯一指定的名字（比如mysql-service）。
2. 拥有一个虚拟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。

Kubernetes.io开发了一个交互式教程，通过WEB浏览器就能使用预先部署好的一个Kubernetes集群，快速体验kubernetes的功能和应用场景。  
链接：<https://kubernetes.io/docs/tutorials/kubernetes-basics/>  
   
K8s官方下载地址：<https://github.com/kubernetes>

**环境准备工作**

|  |  |  |
| --- | --- | --- |
| **主机名** | **操作系统** | **IP地址** |
| master | Centos 7.4-x86\_64 | 192.168.2.102 |
| node1 | Centos 7.4-x86\_64 | 192.168.2.108 |
| node2 | Centos 7.4-x86\_64 | 192.168.2.228 |

**关闭CentOS7自带的防火墙服务**  
systemctl disable firewalld  
systemctl s[top](https://idc.wanyunshuju.com/cym/132t.html) firewalld  
   
**修改主机名**  
[root@localhost ~]# hostnamectl  master //192.168.2.102  
[root@localhost ~]# hostnamectl  node1 //192.168.2.108  
[root@localhost ~]# hostnamectl  node2 //192.168.2.228  
   
更改Hostname为 master、node1、node2，配置IP地址,配置3台测试机的/etc/hosts文件（在三台服务器上同步）  
[root@master ~]# cat /etc/hosts  
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.2.102 master  
192.168.2.108 node1  
192.168.2.228 node2  
[root@master ~]#  
   
   
**时间校对（三台机器都做时间校对）**  
[root@master ~]# ntpdate ntp1.aliyun.com  
13 Sep 14:48:02 ntpdate[1596]: adjust time server 120.25.115.20 offset -0.015854 sec  
   
[root@master ~]# hwclock  
2018年09月13日 星期四 14时49分51秒  -0.646898 秒  
   
 

**安装Kubernetes**

**配置dockers镜像**  
[root@master ~]# wget -O /etc/yum.repos.d/docker-ce.repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

**配置kubernetes仓库**  
   
[root@master yum.repos.d]# vim kubernetes.repo  
[kubernetes]  
name=kubernetes Repo  
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/  
gpgcheck=1  
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg  
enabled=1

[root@master ~]# wget https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg  
[root@master ~]# rpm --import rpm-package-key.gpg  
   
   
[root@master yum.repos.d]# yum repolist

[root@master yum.repos.d]# scp CentOS-Base.repo docker-ce.repo kubernetes.repo node1:/etc/yum.repos.d/

**安装docker和kubelet**

[root@master ~]# yum -y install docker-ce kubelet kubeadm kubectl

**环境变量**  
[root@master ~]# vim /usr/lib/systemd/system/docker.service  
手动添加一行  
12 Environment="HTTPS\_PROXY=http://www.ik8s.io:10080"  
 13 Environment="NO\_PROXY=127.0.0.0/8,172.0.0.0/16"

[root@master ~]# systemctl daemon-reload  
   
**启动docker**  
[root@master ~]# systemctl start docker  
   
**设置开机启动**  
[root@master ~]# systemctl enable docker  
[root@master ~]# docker info  
   
**保证输出的都是1**  
[root@master ~]# cat /proc/sys/net/bridge/bridge-nf-call-ip6tables  
1  
[root@master ~]# cat /proc/sys/net/bridge/bridge-nf-call-iptables  
1  
   
[root@master ~]# rpm -ql kubelet  
/etc/kubernetes/manifests  #清单目录  
/etc/sysconfig/kubelet   #配置文件  
/etc/systemd/system/kubelet.service    
/usr/bin/kubelet   #主程序  
   
**设置开机启动**  
[root@master ~]# systemctl enable kubelet

**初始化**

[root@master ~]# kubeadm init --kubernetes-version=v1.11.1 --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12  
  
  
  
**报错信息，解决办法：**  
   
[root@master ~]# vim /etc/sysconfig/kubelet  
KUBELET\_EXTRA\_ARGS="--fail-swap-on=false"  
   
[root@master ~]# kubeadm init --kubernetes-version=v1.11.1 --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12 --ignore-preflight-errors=Swap

**注意这一条命令需要保存好（添加集群使用）**  
kubeadm join 192.168.2.102:6443 --token ns4kps.j8cuqwf78emp5a5b --discovery-token-ca-cert-hash sha256:b71b7e52c318959bab3f05f02f6fe51d6396d8c54ea6849ec7556927d1c6c88a  
   
   
   
   
   
[root@master ~]# docker image ls

[root@master ~]# ss –ntl

[root@master ~]# mkdir -p $HOME/.kube  
[root@master ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
   
**检测组件运行是否正常**  
[root@master ~]# kubectl get cs

**查看并验证节点信息是否成功**  
[root@master ~]# kubectl get nodes  
NAME      STATUS     ROLES     AGE       VERSION  
master    NotReady   master    44m       v1.11.3  
   
**安装flannel**  
   
[root@master ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml  
[root@master ~]# curl -sSL "https://github.com/coreos/flannel/blob/master/Documentation/kube-flannel.yml?raw=true" | kubectl create -f -

   
[root@master ~]# docker image ls

**查看并验证节点信息是否成功**  
[root@master ~]# kubectl get  nodes  
NAME      STATUS    ROLES     AGE       VERSION  
master    Ready     master    2h        v1.11.3  
   
**查看pod运行情况:**  
[root@master ~]# kubectl get pods -n kube-system

**配置文件传到node1、node2**

[root@master ~]# scp /usr/lib/systemd/system/docker.service node1:/usr/lib/systemd/system/docker.service  
  
[root@master ~]# scp /etc/sysconfig/kubelet node1:/etc/sysconfig/  
  
[root@master ~]# scp /usr/lib/systemd/system/docker.service node2:/usr/lib/systemd/system/docker.service  
  
[root@master ~]# scp /etc/sysconfig/kubelet node2:/etc/sysconfig/  
  
   
   
 

**node1服务器配置**

[root@node1 ~]# wget https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

[root@node1 ~]# rpm --import rpm-package-key.gpg  
   
**安装docker、kubelet**  
[root@node1 ~]# yum -y install docker-ce kubelet kubeadm

**启动docker**  
[root@node1 ~]# systemctl start docker  
[root@node1 ~]# systemctl enable docker kubelet  
  
[root@node1 ~]# docker info  
  
**添加主集群里**  
[root@node1 ~]# kubeadm join 192.168.2.102:6443 --token ns4kps.j8cuqwf78emp5a5b --discovery-token-ca-cert-hash sha256:b71b7e52c318959bab3f05f02f6fe51d6396d8c54ea6849ec7556927d1c6c88a --ignore-preflight-errors=Swap

**查看下载镜像**  
[root@node1 ~]# docker image ls

**在master服务器上可以查节点信息**  
[root@master ~]# kubectl get nodes  
NAME      STATUS    ROLES     AGE       VERSION  
master    Ready     master    3h        v1.11.3  
node1     Ready     <none>    7m        v1.11.3  
   
[root@master ~]# kubectl get pods -n kube-system -o wide

**node2服务器配置**

[root@node2 ~]# wget https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

[root@node2 ~]# rpm --import rpm-package-key.gpg  
   
 **安装docker、kubelet**  
[root@node2 ~]#  yum -y install docker-ce kubelet kubeadm

**启动docker**  
[root@node2 ~]# systemctl start docker  
[root@node2 ~]# systemctl enable docker kubelet  
   
  
   
[root@node2 ~]# docker info

 **添加主集群里**  
[root@node2 ~]# kubeadm join 192.168.2.102:6443 --token ns4kps.j8cuqwf78emp5a5b --discovery-token-ca-cert-hash sha256:b71b7e52c318959bab3f05f02f6fe51d6396d8c54ea6849ec7556927d1c6c88a --ignore-preflight-errors=Swap

**在master服务器上可以查节点信息**  
[root@master ~]# kubectl get nodes

[root@master ~]# kubectl get pods -n kube-system -o wide