



什么是 Kubernetes?

百科上是这样解释的:

Kubernetes 是 Google 开源的一个容器编排引擎, 它支持自动化部署、大规模可伸缩、应用容器化管理。在生产环境中部署一个应用程序时, 通常要部署该应用的多个实例以便对应用请求进行负载均衡。

通俗些讲, 可以将 **Kubernetes** 看作是用来是一个部署镜像的平台。可以用来操作多台机器调度部署镜像, 大大地降低了运维成本。

那么, **Kubernetes** 和 **Docker** 的关系又是怎样的呢?



一个形象的比喻: 如果你将 **docker 看作是飞机, 那么 **kubernetes** 就是飞机场。在飞机场的加持下, 飞机可以根据机场调度选择在合适的时间降落或起飞。**

在 **Kubernetes** 中, 可以使用集群来组织服务器的。集群中会存在一个 **Master** 节点, 该节点是 **Kubernetes** 集群的控制节点, 负责调度集群中其他服务器的资源。其他节点被称为 **Node**, **Node** 可以是物理机也可以是虚拟机。

基础安装

基础安装章节, Master & Node 节点都需要安装

第一步我们安装些必备组件。 **vim** 是 **Linux** 下的一个文件编辑器; **wget** 可以用作文件下载使用; **ntpdate** 则是可以用来同步时区:



shell 复制代码

```
1 yum install vim wget ntpdate -y
```

接着我们关闭防火墙。因为 **kubernetes** 会创建防火墙规则, 导致防火墙规则重复。所以这里我们要将防火墙关闭:



shell 复制代码

```
1 systemctl stop firewalld & systemctl disable firewalld
```

应该让新创建的服务自动调度到集群的其他 **Node** 节点中去，而不是使用 **Swap** 分区。这里我们将其关掉：

▼ shell 复制代码

```
1 #临时关闭
2 swapoff -a
3 # 永久关闭
4 vi /etc/fstab
```

找到 **/etc/fstab** 文件，注释掉下面这一行：

▼ shell 复制代码

```
1 /dev/mapper/centos-swap swap ...
```

继续关闭 **Selinux**。这是为了支持容器可以访问宿主机文件系统所做的，后续也许会优化掉：

▼ shell 复制代码

```
1 # 暂时关闭 selinux
2 setenforce 0
```

▼ shell 复制代码

```
1 # 永久关闭
2 vi /etc/sysconfig/selinux
3 # 修改以下参数，设置为disable
4 SELINUX=disabled
```

关于为什么关闭防火墙，selinux，swap。这里有几份更标准的回答：

<https://www.zhihu.com/question/374752553>

接着使用 **ntpdate** 来统一我们的系统时间和时区，服务器时间与阿里云服务器对齐。

▼ shell 复制代码

```
1 # 统一时区，为上海时区
2 ln -snf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
3 bash -c "echo 'Asia/Shanghai' > /etc/timezone"
4
5 # 统一使用阿里服务器进行时间更新
6 ntpdate ntp1.aliyun.com
```

安装 Docker

在安装 `Docker` 之前，需要安装 `device-mapper-persistent-data` 和 `lvm2` 两个依赖。我们使用 `Yum` 命令直接安装依赖即可：

▼ [shell](#) [复制代码](#)

```
1 yum install -y yum-utils device-mapper-persistent-data lvm2
```

`device-mapper-persistent-data`: 存储驱动，Linux上的许多高级卷管理技术 `lvm`: 逻辑卷管理器，用于创建逻辑磁盘分区使用

接下来，添加阿里云的 `Docker` 镜像源，加速 `Docker` 的安装：

▼ [shell](#) [复制代码](#)

```
1 sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
2 yum install docker-ce -y
```

我们还需要修改一下docker的 `cgroupdriver` 为 `systemd`，这样做是为了避免后面与k8s的冲突。

▼ [shell](#) [复制代码](#)

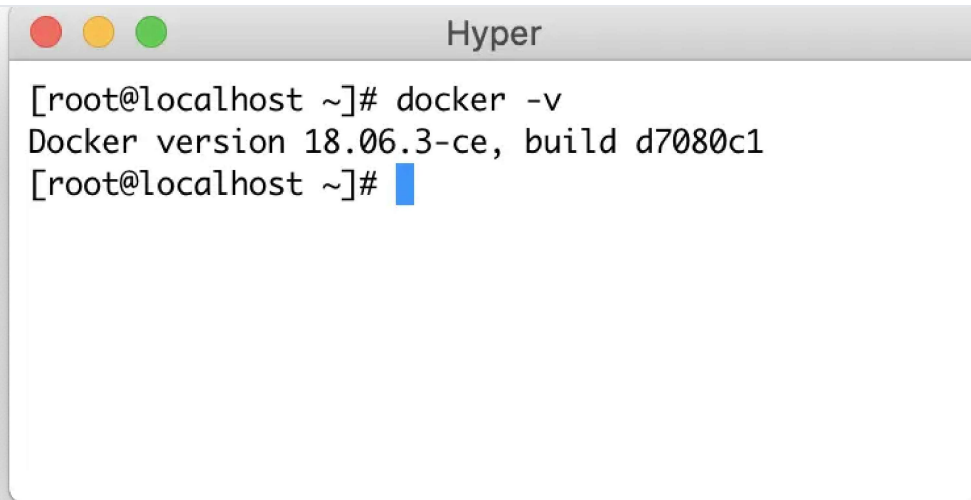
```
1 cat > /etc/docker/daemon.json <<EOF
2 {
3     "exec-opts": ["native.cgroupdriver=systemd"]
4 }
5 EOF
```

安装完毕后，我们使用使用 `systemctl` 启动 `Docker` 即可

▼ [shell](#) [复制代码](#)

```
1 systemctl start docker
2 systemctl enable docker
```

执行 `docker -v`，如果显示以下 `docker` 版本的信息，代表 `docker` 安装成功。

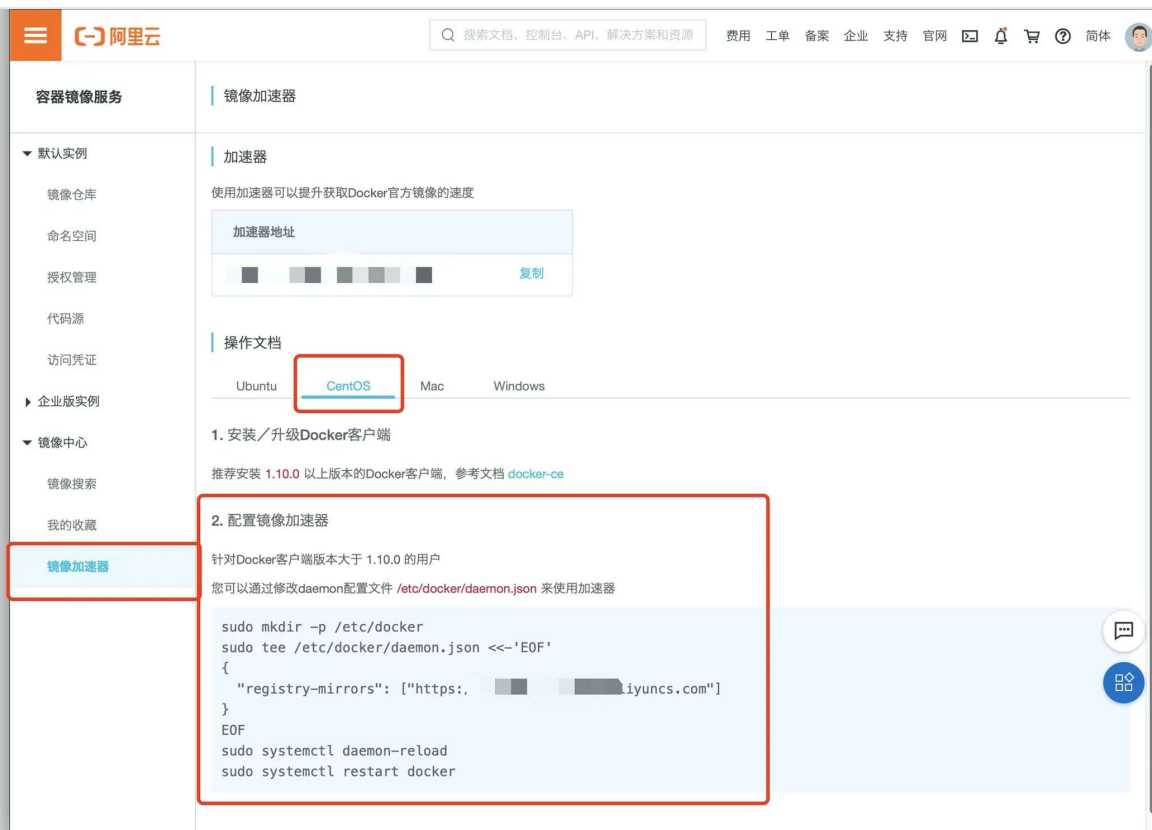


```
[root@localhost ~]# docker -v
Docker version 18.06.3-ce, build d7080c1
[root@localhost ~]#
```

@稀土掘金技术社区

我们拉取 **Docker** 镜像时，一般默认会去 **Docker** 官方源拉取镜像。但是国内出海网速实在是太慢，所以我们更换为 **阿里云镜像仓库** 源进行镜像下载加速

登录阿里云官网，打开 [阿里云容器镜像服务](#)。点击左侧菜单最下面的 **镜像加速器**，选择 **Centos**（如下图）



@稀土掘金技术社区

按照官网的提示，执行命令，即可实现更换 **Docker** 镜像源地址。

还记得我们前面安装私有镜像库时的使用提示吗？在安装完 **Docker** 后，**如果你的私有镜像库是 HTTP 而不是 HTTPS的话，需要在 **`/etc/docker/daemon.json`** 里配置一下你的私有库地址。

编辑 `/etc/docker/daemon.json` 文件，添加 `insecure-registries` 字段。字段的值是数组，数组的第一项填入你的私有库地址即可（不要忘记后面的逗号）。如示例：

json 复制代码

```
1 {
2   "insecure-registries": ["http://[私有库地址]:[私有库端口]"],
3   "registry-mirrors": ["https://****.mirror.aliyuncs.com"]
4 }
```

保存后退出，重启下 **Docker** 服务：

json 复制代码

```
1 sudo systemctl daemon-reload
2 sudo systemctl restart docker.service
```

安装 Kubernetes 组件

shell 复制代码

```
1 cat <<EOF > /etc/yum.repos.d/kubernetes.repo
2 [kubernetes]
3 name=Kubernetes
4 baseurl=http://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
5 enabled=1
6 gpgcheck=0
7 repo_gpgcheck=0
8 gpgkey=http://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
9       http://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
10 EOF
```

接着直接使用 `yum` 命令安装 `kubelet`、`kubeadm`、`kubect1` 即可，安装完毕后启用 `kubelet` 即可。

shell 复制代码

```
1 yum install -y kubelet-1.23.6 kubeadm-1.23.6 kubect1-1.23.6
2 # 启动kubelet
3 systemctl enable kubelet && systemctl start kubelet
```

`kubelet` 是 `Kubernetes` 中的核心组件。它会运行在集群的所有节点上，并负责创建启动服务容器
`kubect1` 则是 `Kubernetes` 的命令行工具。可以用来管理，删除，创建资源 `kubeadm` 则是用来初始化集群，子节点加入的工具。

Master 节点安装

Master 节点是集群内的调度和主要节点，以下部分仅限 **Master 节点才能安装**。

首先，我们使用 `hostnamectl` 来修改主机名称为 `master`。`hostnamectl` 是 `Centos7` 出的新命令，可以用来修改主机名称：

shell 复制代码

```
1 hostnamectl set-hostname master
```

接着使用 `ip addr` 命令，获取本机IP，将其添加到 `/etc/hosts` 内：

shell 复制代码

```
1 # xxx.xxx.xxx.xxx master
2 vim /etc/hosts
```

配置 Kubernetes 初始化文件

▼ shell 复制代码

```
1 kubectl config print init-defaults > init-kubeadm.conf
2 vim init-kubeadm.conf
```

主要对配置文件做这几件事情：

- 更换 **Kubernetes** 镜像仓库为阿里云镜像仓库，加速组件拉取
- 替换 **ip** 为自己主机 **ip**
- 配置 **pod** 网络为 **flannel** 网段

▼ shell 复制代码

```
1 # imageRepository: k8s.gcr.io 更换k8s镜像仓库
2 imageRepository: registry.cn-hangzhou.aliyuncs.com/google_containers
3 # localAPIEndpoint, advertiseAddress为master-ip, port默认不修改
4 localAPIEndpoint:
5   advertiseAddress: 192.168.56.101 # 此处为master的IP
6   bindPort: 6443
7 # 配置子网络
8 networking:
9   dnsDomain: cluster.local
10  serviceSubnet: 10.96.0.0/12
11  podSubnet: 10.244.0.0/16 # 添加这个
```

在修改完配置文件后，我们需要使用 **kubeadm** 拉取我们的默认组件镜像。直接使用 **kubeadm config images pull** 命令即可

▼ shell 复制代码

```
1 kubectl config images pull --config init-kubeadm.conf
```

初始化 Kubernetes

在镜像拉取后，我们就可以使用刚才编辑好的配置文件去初始化 **Kubernetes** 集群了。这里直接使用 **kubeadm init** 命令去初始化即可。

▼ shell 复制代码

```
1 kubectl init --config init-kubeadm.conf
```

```
Hyper
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.19" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master as control-plane by adding the label "node-role.kubernetes.io/master="
[mark-control-plane] Marking the node master as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: abcdef.0123456789abcdef
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.16.81.164:6443 --token abcdef.0123456789abcdef \
--discovery-token-ca-cert-hash sha256:b4a059eeffa2e52f2eea7a5d592be10c994c7715c17bda57bbc3757d4f13903d
[root@localhost ~]#
```

@稀土掘金技术社区

其中，红框命令为在 **Master** 节点需要执行的初始化命令，其作用为将默认的 **Kubernetes** 认证文件拷贝进 **.kube** 文件夹内，才能默认使用该配置文件。

蓝框为需要在 **node** 节点执行的命令。作用是可以快速将 **Node** 节点加入到 **Master** 集群内。

安装 Flannel

前面我们在配置文件中，有提到过配置**Pod子网络**，**Flannel** 主要的作用就是如此。它的主要作用是通过**创建一个虚拟网络**，让不同节点下的服务有着全局唯一的IP地址，且服务之前可以互相访问和连接。

那么 **Flannel** 作为 **Kubernetes** 的一个组件，则使用 **Kubernetes** 部署服务的方式进行安装。首先下载配置文件：

```
shell 复制代码
1 wget https://raw.githubusercontent.com/coreos/flannel/v0.18.1/Documentation/kube-flannel.yml
```

在这里，如果提示你 raw.githubusercontent.com 无法访问或连接超时，可以尝试以下办法：

1. 去 [githubusercontent.com.ipaddress.com/raw.githubu...](https://raw.githubusercontent.com/coreos/flannel/v0.18.1/Documentation/kube-flannel.yml) 获取新的IP
2. 编辑 hosts 文件，将获取的新IP直接映射到域名上



```

- linux
hostNetwork: true
priorityClassName: system-node-critical
tolerations:
- operator: Exists
  effect: NoSchedule
serviceAccountName: flannel
initContainers:
- name: install-cni
  image: quay.io/coreos/flannel:v0.13.0-rc2
  command:
  - cp
  args:
  - -f
  - /etc/kube-flannel/cni-conf.json
  - /etc/cni/net.d/10-flannel.conflist
  volumeMounts:
  - name: cni
    mountPath: /etc/cni/net.d
  - name: flannel-cfg
    mountPath: /etc/kube-flannel/

```

@稀土掘金技术社区

shell 复制代码

```
1 docker pull quay.io/coreos/flannel:v0.13.0-rc2
```

等待镜像拉取结束后，可以使用 `kubectl apply` 命令加载下服务。

shell 复制代码

```
1 kubectl apply -f kube-flannel.yml
```

查看启动情况

在大约稍后1分钟左右，我们可以使用 `kubectl get nodes` 命令查看节点的运行状态。如果 `STATUS = ready`，则代表启动成功。

shell 复制代码

```
1 kubectl get nodes
```

```

[root@master ~]# kubectl get node
NAME        STATUS    ROLES    AGE   VERSION
master      Ready     master   62m   v1.19.2

```

@稀土掘金技术社区

****在安装 Node 节点前，我们仍然需要操作一遍上面的 基础安装 **。** Node 节点的地位则是负责运行服务容器，负责接收调度的。

首先第一步，还是需要先设置一下 `hostname` 为 `node1`。在 `node` 机器上执行：

```
1 hostnamectl set-hostname node1
```

shell 复制代码

拷贝 Master 节点配置文件

接着将 `master` 节点的配置文件拷贝 `k8s` 到 `node` 节点。回到在 `master` 节点，使用 `scp` 命令通过 `SSH` 传送文件：

```
1 scp $HOME/.kube/config root@node的ip:~/
```

shell 复制代码

随后在 `node` 节点执行以下命令，归档配置文件：

```
1 mkdir -p $HOME/.kube
2 sudo mv $HOME/config $HOME/.kube/config
3 sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

shell 复制代码

加入 Master 节点

我们直接使用刚才在 `master` 生成的节点加入命令，在 `node` 机器上执行。让 `Node` 节点加入到 `master` 集群内：

```
1 # 这是一条是示例命令!!!!!!
2 kubeadm join 172.16.81.164:6443 --token abcdef.0123456789abcdef \
3   --discovery-token-ca-cert-hash sha256:b4a059eeffa2e52f2eea7a5d592be10c994c7715c17bda57bbc3757d4f13903d
```

shell 复制代码

如果刚才的命令丢了，可以在 `master` 机器上使用 `kubeadm token create` 重新生成一条命令：

```
1 kubeadm token create --print-join-command
```

shell 复制代码

```
[root@master ~]#
```

@稀土掘金技术社区

安装 Flannel

这里和 Master 安装执行方式一样，参考同上。

结束语

在本章，我们从 0-1 部署了一套 Kubernetes 集群。在下一章，我们将在集群内运行我们的第一个应用。
加油 🍀

如果你有疑问，欢迎在评论区留言讨论。

[< 上一章](#)[下一章 >](#)

留言

输入评论 (Enter换行, Ctrl + Enter发送)

发表评论

全部评论 (51)



mae 前端工程师 1月前

初始化 Kubernetes报错可以尝试这个解决方案 cloud.tencent.com

👍 点赞 🗨 回复 ...



前端搞毛开发工程师 搞毛开发工程师 @ 国... 4月前

init操作在主节点, join操作在从节点
购买云服务器的时候主节点和从节点要选同一地区, 跨地区join操作会失败

👍 1 🗨 回复 ...



Asuka14024 前端工程师 6月前

要是买的服务器注意 centos 8是没法用ntptdate 的, 用自带的 chrony 就行了

👍 点赞 🗨 2 ...



Asuka14024 6月前