11 | 从0到1：搭建一个完整的Kubernetes集群

2018-09-17 张磊

11 | 从0到1：搭建一个完整的Kubernetes集群
朗读人：张磊　17′20″ | 7.95M

你好，我是张磊。今天我和你分享的主题是：从 0 到 1 搭建一个完整的 Kubernetes 集群。

在上一篇文章中，我介绍了 kubeadm 这个 Kubernetes 半官方管理工具的工作原理。既然 kubeadm 的初衷是让 Kubernetes 集群的部署不再让人头疼，那么这篇文章，我们就来使用它部署一个完整的 Kubernetes 集群吧。

> 备注：这里所说的“完整”，指的是这个集群具备 Kubernetes 项目在 GitHub 上已经发布的所有功能，并能够模拟生产环境的所有使用需求。但并不代表这个集群是生产级别可用的：类似于高可用、授权、多租户、灾难备份等生产级别集群的功能暂时不在本篇文章的讨论范围。
> 目前，kubeadm 的高可用部署已经有了第一个发布。但是，这个特性还没有 GA（生产可用），所以包括了大量的手动工作，跟我们所预期的一键部署还有一定距离。GA 的日期预计是 2018 年底到 2019 年初。届时，如果有机会我会再和你分享这部分内容。

这次部署，我不会依赖于任何公有云或私有云的能力，而是完全在 Bare-metal 环境中完成。这样的部署经验会更有普适性。而在后续的讲解中，如非特殊强调，我也都会以本次搭建的这个集群为

基础。

准备工作

首先，准备机器。最直接的办法，自然是到公有云上申请几个虚拟机。当然，如果条件允许的话，拿几台本地的物理服务器来组集群是最好不过了。这些机器只要满足如下几个条件即可：

1. 满足安装 Docker 项目所需的要求，比如 64 位的 Linux 操作系统、3.10 及以上的内核版本；
2. x86 或者 ARM 架构均可；
3. 机器之间网络互通，这是将来容器之间网络互通的前提；
4. 有外网访问权限，因为需要拉取镜像；
5. 能够访问到`gcr.io`、`quay.io`这两个 docker registry，因为有小部分镜像需要在这里拉取；
6. 单机可用资源建议 2 核 CPU、8 GB 内存或以上，再小的话问题也不大，但是能调度的 Pod 数量就比较有限了；
7. 30 GB 或以上的可用磁盘空间，这主要是留给 Docker 镜像和日志文件用的。

在本次部署中，我准备的机器配置如下：

1. 2 核 CPU、 7.5 GB 内存；
2. 30 GB 磁盘；
3. Ubuntu 16.04；
4. 内网互通；
5. 外网访问权限不受限制；

> 备注：在开始部署前，我推荐你先花几分钟时间，回忆一下 Kubernetes 的架构。

然后，我再和你介绍一下今天实践的目标：

1. 在所有节点上安装 Docker 和 kubeadm；
2. 部署 Kubernetes Master；
3. 部署容器网络插件；
4. 部署 Kubernetes Worker；

5. 部署 Dashboard 可视化插件；

6. 部署容器存储插件。

好了，现在，就来开始这次集群部署之旅吧！

安装 kubeadm 和 Docker

我在上一篇文章《 Kubernetes 一键部署利器：kubeadm》中，已经介绍过 kubeadm 的基础用法，它的一键安装非常方便，我们只需要添加 kubeadm 的源，然后直接使用 apt-get 安装即可，具体流程如下所示：

> 备注：为了方便讲解，我后续都直接会在 root 用户下进行操作

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
$ cat <<EOF > /etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
$ apt-get update
$ apt-get install -y docker.io kubeadm
```

在上述安装 kubeadm 的过程中，kubeadm 和 kubelet、kubectl、kubernetes-cni 这几个二进制文件都会被自动安装好。

另外，这里我直接使用 Ubuntu 的 docker.io 的安装源，原因是 Docker 公司每次发布的最新的 Docker CE（社区版）产品往往还没有经过 Kubernetes 项目的验证，可能会有兼容性方面的问题。

部署 Kubernetes 的 Master 节点

在上一篇文章中，我已经介绍过 kubeadm 可以一键部署 Master 节点。不过，在本篇文章中既然要部署一个“完整”的 Kubernetes 集群，那我们不妨稍微提高一下难度：通过配置文件来开启一些实验性功能。

所以，这里我编写了一个给 kubeadm 用的 YAML 文件（名叫：kubeadm.yaml）：

```
apiVersion: kubeadm.k8s.io/v1alpha1
kind: MasterConfiguration
controllerManagerExtraArgs:
  horizontal-pod-autoscaler-use-rest-clients: "true"
```

```
  horizontal-pod-autoscaler-sync-period: "10s"
  node-monitor-grace-period: "10s"
apiServerExtraArgs:
  runtime-config: "api/all=true"
kubernetesVersion: "stable-1.11"
```

这个配置中，我给 kube-controller-manager 设置了：

```
horizontal-pod-autoscaler-use-rest-clients: "true"
```

这意味着，将来部署的 kube-controller-manager 能够使用自定义资源（Custom Metrics）进行自动水平扩展。这是我后面文章中会重点介绍的一个内容。

其中，“stable-1.11”就是 kubeadm 帮我们部署的 Kubernetes 版本号，即：Kubernetes release 1.11 最新的稳定版，在我的环境下，它是 v1.11.1。你也可以直接指定这个版本，比如：kubernetesVersion: “v1.11.1”

然后，我们只需要执行一句指令：

```
$ kubeadm init --config kubeadm.yaml
```

就可以完成 Kubernetes Master 的部署了，这个过程只需要几分钟。部署完成后，kubeadm 会生成一行指令：

```
kubeadm join 10.168.0.2:6443 --token 00bwbx.uvnaa2ewjflwu1ry --discovery-token-ca-cert-hash sha25
```

这个 kubeadm join 命令，就是用来给这个 Master 节点添加更多工作节点（Worker）的命令。我们在后面部署 Worker 节点的时候马上会用到它，所以找一个地方把这条命令记录下来。

此外，kubeadm 还会提示我们第一次使用 Kubernetes 集群所需要的配置命令：

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

而需要这些配置命令的原因是：Kubernetes 集群默认需要加密方式访问。所以，这几条命令，就是将刚刚部署生成的 Kubernetes 集群的安全配置文件，保存到当前用户的.kube 目录下，kubectl 默认会使用这个目录下的授权信息访问 Kubernetes 集群。

如果不这么做的话，我们每次都需要通过 export KUBECONFIG 环境变量告诉 kubectl 这个安全配置文件的位置。

现在，我们就可以使用 kubectl get 命令来查看当前唯一一个节点的状态了：

```
$ kubectl get nodes
NAME      STATUS     ROLES     AGE       VERSION
master    NotReady   master    1d        v1.11.1
```

可以看到，这个 get 指令输出的结果里，Master 节点的状态是 NotReady，这是为什么呢？

在调试 Kubernetes 集群时，最重要的手段就是用 kubectl describe 来查看这个节点（Node）对象的详细信息、状态和事件（Event），我们来试一下：

```
$ kubectl describe node master


...
Conditions:
...


Ready   False ... KubeletNotReady  runtime network not ready: NetworkReady=false reason:NetworkPl
```

通过 kubectl describe 指令的输出，我们可以看到 NodeNotReady 的原因在于，我们尚未部署任何网络插件。

另外，我们还可以通过 kubectl 检查这个节点上各个系统 Pod 的状态，其中，kube-system 是 Kubernetes 项目预留的系统 Pod 的工作空间（Namepsace，注意它并不是 Linux Namespace，它只是 Kubernetes 划分不同工作空间的单位）：

```
$ kubectl get pods -n kube-system
NAME               READY   STATUS    RESTARTS  AGE
coredns-78fcdf6894-j9s52     0/1    Pending  0     1h
coredns-78fcdf6894-jm4wf     0/1    Pending  0     1h
```

```
etcd-master           1/1    Running  0     2s
kube-apiserver-master      1/1    Running  0     1s
kube-controller-manager-master  0/1    Pending  0     1s
kube-proxy-xbd47         1/1    NodeLost  0     1h
kube-scheduler-master      1/1    Running  0     1s
```

可以看到，CoreDNS、kube-controller-manager 等依赖于网络的 Pod 都处于 Pending 状态，即调度失败。这当然是符合预期的：因为这个 Master 节点的网络尚未就绪。

部署网络插件

在 Kubernetes 项目“一切皆容器”的设计理念指导下，部署网络插件非常简单，只需要执行一句 kubectl apply 指令，以 Weave 为例：

```
$ kubectl apply -f https://git.io/weave-kube-1.6
```

部署完成后，我们可以通过 kubectl get 重新检查 Pod 的状态：

```
$ kubectl get pods -n kube-system
NAME                              READY     STATUS    RESTARTS   AGE
coredns-78fcdf6894-j9s52          1/1       Running   0          1d
coredns-78fcdf6894-jm4wf          1/1       Running   0          1d
etcd-master                       1/1       Running   0          9s
kube-apiserver-master             1/1       Running   0          9s
kube-controller-manager-master    1/1       Running   0          9s
kube-proxy-xbd47                  1/1       Running   0          1d
kube-scheduler-master             1/1       Running   0          9s
weave-net-cmk27                   2/2       Running   0          19s
```

可以看到，所有的系统 Pod 都成功启动了，而刚刚部署的 Weave 网络插件则在 kube-system 下面新建了一个名叫 weave-net-cmk27 的 Pod，一般来说，这些 Pod 就是容器网络插件在每个节点上的控制组件。

Kubernetes 支持容器网络插件，使用的是一个名叫 CNI 的通用接口，它也是当前容器网络的事实标准，市面上的所有容器网络开源项目都可以通过 CNI 接入 Kubernetes，比如 Flannel、Calico、Canal、Romana 等等，它们的部署方式也都是类似的“一键部署”。关于这些开源项目的实现细节和差异，我会在后续的网络部分详细介绍。

至此，Kubernetes 的 Master 节点就部署完成了。如果你只需要一个单节点的 Kubernetes，现在你就可以使用了。不过，在默认情况下，Kubernetes 的 Master 节点是不能运行用户 Pod 的，所以还需要额外做一个小操作。在本篇的最后部分，我会介绍到它。

部署 Kubernetes 的 Worker 节点

Kubernetes 的 Worker 节点跟 Master 节点几乎是相同的，它们运行着的都是一个 kubelet 组件。唯一的区别在于，在 kubeadm init 的过程中，kubelet 启动后，Master 节点上还会自动运行 kube-apiserver、kube-scheduler、kube-controller-manger 这三个系统 Pod。

所以，相比之下，部署 Worker 节点反而是最简单的，只需要两步即可完成。

第一步，在所有 Worker 节点上执行“安装 kubeadm 和 Docker”一节的所有步骤。

第二步，执行部署 Master 节点时生成的 kubeadm join 指令：

```
$ kubeadm join 10.168.0.2:6443 --token 00bwbx.uvnaa2ewjflwu1ry --discovery-token-ca-cert-hash sha
```

通过 Taint/Toleration 调整 Master 执行 Pod 的策略

我在前面提到过，默认情况下 Master 节点是不允许运行用户 Pod 的。而 Kubernetes 做到这一点，依靠的是 Kubernetes 的 Taint/Toleration 机制。

它的原理非常简单：一旦某个节点被加上了一个 Taint，即被“打上了污点”，那么所有 Pod 就都不能在这个节点上运行，因为 Kubernetes 的 Pod 都有“洁癖”。

除非，有个别的 Pod 声明自己能“容忍”这个“污点”，即声明了 Toleration，它才可以在这个节点上运行。

其中，为节点打上“污点”（Taint）的命令是：

```
$ kubectl taint nodes node1 foo=bar:NoSchedule
```

这时，该 node1 节点上就会增加一个键值对格式的 Taint，即：foo=bar:NoSchedule。其中值里面的 NoSchedule，意味着这个 Taint 只会在调度新 Pod 时产生作用，而不会影响已经在 node1 上运行的 Pod，哪怕它们没有 Toleration。

那么 Pod 又如何声明 Toleration 呢？

我们只要在 Pod 的.yaml 文件中的 spec 部分，加入 tolerations 字段即可：

```
apiVersion: v1
kind: Pod
...
spec:
  tolerations:
  - key: "foo"
    operator: "Equal"
    value: "bar"
    effect: "NoSchedule"
```

这个 Toleration 的含义是，这个 Pod 能“容忍”所有键值对为 foo=bar 的 Taint（ operator: “Equal”，“等于”操作）。

现在回到我们已经搭建的集群上来。这时，如果你通过 kubectl describe 检查一下 Master 节点的 Taint 字段，就会有所发现了：

```
$ kubectl describe node master


Name:               master
Roles:              master
Taints:             node-role.kubernetes.io/master:NoSchedule
```

可以看到，Master 节点默认被加上了`node-role.kubernetes.io/master:NoSchedule`这样一个“污点”，其中“键”是`node-role.kubernetes.io/master`，而没有提供“值”。

此时，你就需要像下面这样用“Exists”操作符（operator: “Exists”，“存在”即可）来说明，该 Pod 能够容忍所有以 foo 为键的 Taint，才能让这个 Pod 运行在该 Master 节点上：

```
apiVersion: v1
kind: Pod
...
spec:
  tolerations:
  - key: "foo"
    operator: "Exists"
    effect: "NoSchedule"
```

当然，如果你就是想要一个单节点的 Kubernetes，删除这个 Taint 才是正确的选择：

```
$ kubectl taint nodes --all node-role.kubernetes.io/master-
```

如上所示，我们在“node-role.kubernetes.io/master”这个键后面加上了一个短横线“-”，这个格式就意味着移除所有以“node-role.kubernetes.io/master”为键的 Taint。

到了这一步，一个基本完整的 Kubernetes 集群就部署完毕了。是不是很简单呢？

有了 kubeadm 这样的原生管理工具，Kubernetes 的部署已经被大大简化。更重要的是，像证书、授权、各个组件的配置等部署中最麻烦的操作，kubeadm 都已经帮你完成了。

接下来，我们再在这个 Kubernetes 集群上安装一些其他的辅助插件，比如 Dashboard 和存储插件。

部署 Dashboard 可视化插件

在 Kubernetes 社区中，有一个很受欢迎的 Dashboard 项目，它可以给用户提供一个可视化的 Web 界面来查看当前集群的各种信息。毫不意外，它的部署也相当简单：

```
$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recom
```

部署完成之后，我们就可以查看 Dashboard 对应的 Pod 的状态了：

```
$ kubectl get pods -n kube-system
kubernetes-dashboard-6948bdb78-f67xk   1/1       Running   0          1m
```

需要注意的是，由于 Dashboard 是一个 Web Server，很多人经常会在自己的公有云上无意地暴露 Dashboard 的端口，从而造成安全隐患。所以，1.7 版本之后的 Dashboard 项目部署完成后，默认只能通过 Proxy 的方式在本地访问。具体的操作，你可以查看 Dashboard 项目的官方文档。

而如果你想从集群外访问这个 Dashboard 的话，就需要用到 Ingress，我会在后面的文章中专门介绍这部分内容。

部署容器存储插件

接下来，让我们完成这个 Kubernetes 集群的最后一块拼图：容器持久化存储。

我在前面介绍容器原理时已经提到过，很多时候我们需要用数据卷（Volume）把外面宿主机上的目录或者文件挂载进容器的 Mount Namespace 中，从而达到容器和宿主机共享这些目录或者文件的目的。容器里的应用，也就可以在这些数据卷中新建和写入文件。

可是，如果你在某一台机器上启动的一个容器，显然无法看到其他机器上的容器在它们的数据卷里写入的文件。这是容器最典型的特征之一：无状态。

而容器的持久化存储，就是用来保存容器存储状态的重要手段：存储插件会在容器里挂载一个基于网络或者其他机制的远程数据卷，使得在容器里创建的文件，实际上是保存在远程存储服务器上，或者以分布式的方式保存在多个节点上，而与当前宿主机没有任何绑定关系。这样，无论你在其他哪个宿主机上启动新的容器，都可以请求挂载指定的持久化存储卷，从而访问到数据卷里保存的内容。这就是“持久化”的含义。

由于 Kubernetes 本身的松耦合设计，绝大多数存储项目，比如 Ceph、GlusterFS、NFS 等，都可以为 Kubernetes 提供持久化存储能力。在这次的部署实战中，我会选择部署一个很重要的 Kubernetes 存储插件项目：Rook。

Rook 项目是一个基于 Ceph 的 Kubernetes 存储插件（它后期也在加入对更多存储实现的支持）。不过，不同于对 Ceph 的简单封装，Rook 在自己的实现中加入了水平扩展、迁移、灾难备份、监控等大量的企业级功能，使得这个项目变成了一个完整的、生产级别可用的容器存储插件。

得益于容器化技术，用两条指令，Rook 就可以把复杂的 Ceph 存储后端部署起来：

```
$ kubectl apply -f https://raw.githubusercontent.com/rook/rook/master/cluster/examples/kubernetes

$ kubectl apply -f https://raw.githubusercontent.com/rook/rook/master/cluster/examples/kubernetes
```

在部署完成后，你就可以看到 Rook 项目会将自己的 Pod 放置在由它自己管理的两个 Namespace 当中：

```
$ kubectl get pods -n rook-ceph-system
NAME                                  READY     STATUS    RESTARTS   AGE

rook-ceph-agent-7cv62                 1/1       Running   0          15s
rook-ceph-operator-78d498c68c-7fj72   1/1       Running   0          44s
rook-discover-2ctcv                   1/1       Running   0          15s

$ kubectl get pods -n rook-ceph
NAME                   READY     STATUS    RESTARTS   AGE
```

```
rook-ceph-mon0-kxnzh   1/1       Running   0          13s
rook-ceph-mon1-7dn2t   1/1       Running   0          2s
```

这样，一个基于 Rook 持久化存储集群就以容器的方式运行起来了，而接下来在 Kubernetes 项目上创建的所有 Pod 就能够通过 Persistent Volume（PV）和 Persistent Volume Claim（PVC）的方式，在容器里挂载由 Ceph 提供的数据卷了。

而 Rook 项目，则会负责这些数据卷的生命周期管理、灾难备份等运维工作。关于这些容器持久化存储的知识，我会在后续章节中专门讲解。

这时候，你可能会有个疑问：为什么我要选择 Rook 项目呢？

其实，是因为这个项目很有前途。

如果你去研究一下 Rook 项目的实现，就会发现它巧妙地依赖了 Kubernetes 提供的编排能力，合理的使用了很多诸如 Operator、CRD 等重要的扩展特性（这些特性我都会在后面的文章中逐一讲解到）。这使得 Rook 项目，成为了目前社区中基于 Kubernetes API 构建的最完善也最成熟的容器存储插件。我相信，这样的发展路线，很快就会得到整个社区的推崇。

> 备注：其实，在很多时候，大家说的所谓“云原生”，就是“Kubernetes 原生”的意思。而像 Rook、Istio 这样的项目，正是贯彻这个思路的典范。在我们后面讲解了声明式 API 之后，相信你对这些项目的设计思想会有更深刻的体会。

总结

在本篇文章中，我们完全从 0 开始，在 Bare-metal 环境下使用 kubeadm 工具部署了一个完整的 Kubernetes 集群：这个集群有一个 Master 节点和多个 Worker 节点；使用 Weave 作为容器网络插件；使用 Rook 作为容器持久化存储插件；使用 Dashboard 插件提供了可视化的 Web 界面。

这个集群，也将会是我进行后续讲解所依赖的集群环境，并且在后面的讲解中，我还会给它安装更多的插件，添加更多的新能力。

另外，这个集群的部署过程并不像传说中那么繁琐，这主要得益于：

1. kubeadm 项目大大简化了部署 Kubernetes 的准备工作，尤其是配置文件、证书、二进制文件的准备和制作，以及集群版本管理等操作，都被 kubeadm 接管了。
2. Kubernetes 本身“一切皆容器”的设计思想，加上良好的可扩展机制，使得插件的部署非常简便。

上述思想，也是开发和使用 Kubernetes 的重要指导思想，即：基于 Kubernetes 开展工作时，你一定要优先考虑这两个问题：

1. 我的工作是不是可以容器化？

2. 我的工作是不是可以借助 Kubernetes API 和可扩展机制来完成？

而一旦这项工作能够基于 Kubernetes 实现容器化，就很有可能像上面的部署过程一样，大幅简化原本复杂的运维工作。对于时间宝贵的技术人员来说，这个变化的重要性是不言而喻的。

思考题

1. 你是否使用其他工具部署过 Kubernetes 项目？经历如何？

2. 你是否知道 Kubernetes 项目当前（v1.11）能够有效管理的集群规模是多少个节点？你在生产环境中希望部署或者正在部署的集群规模又是多少个节点呢？

感谢你的收听，欢迎你给我留言。

精选留言

bing 20
真的是写的最好的专栏
2018-09-17

同心结 2
你好，照着你的教程，我在执行kubeadm init --config kubeadm.yaml， 提示：your configuration file uses an old API spec: "kubeadm.k8s.io/v1alpha1". Please use kubeadm v1.11 instead and run 'kubeadm config migrate --old-config old.yaml --new-config new.yaml', which will write the new,

similar spec using a newer API version.
默认安装的kubeadm和kubelet版本是1.12
2018-09-20

LenX 2
老师，怎么让 Master 高可用，能给几个方案的关键词吗？我想根据关键词课后自己先研究一下。
2018-09-19

暮雨 2
没有外网的环境，安装这个有没什么好的方法，在阿里云上开了两台香港节点的主机
2018-09-17

lm书生 2
老师您好！专网，不能访问互联网的情况下，离线部署kubernetes集群？有没有好的方法和建议？
2018-09-17

Jeremy 1
如果有人遇到：
[ERROR KubeletVersion]: the kubelet version is higher than the control plane version. This is not a supported version skew and may lead to a malfunctional cluster. Kubelet version: "1.12.0-rc.1" Control plane version: "1.11.3"
这个错误，使用：
apt-get install kubelet=1.11.3-00
就可以了，发出来希望对人有用。
2018-09-20

John 1
不翻墙 有办法装 kubeadm 吗？
2018-09-18
作者回复
可以离线下好
2018-09-19

包治结巴 1
张老师，请教一个问题，master部署完成后生成指令，比如“kubeadm join 172.19.0.7:6443 --token xw4d6t.1dmpctojkcg9yqlf --discovery-token-ca-cert-hash sha256:45168e55ad19ac6126728e86c0a4c46e98c7df5d0312aec3fdd8d437dc24a3b9”，假如当时没有记下来，有什么命令可以再次让master生成这个指令吗？
2018-09-18
作者回复
可以用kubeadm查看token
2018-09-18

刘欣洲 1

Rook 能够直接支持Google 的 GCP的persistent disk 吗？该如何实现呢？
2018-09-18
作者回复
都有gcp了还搞rook干啥，这些cloud都已经给kubernetes做了存储支持了
2018-09-18


柳旭 1
ubuntu 换成这个源
deb http://mirrors.ustc.edu.cn/kubernetes/apt kubernetes-xenial main
2018-09-17


巩夫建 1
二进制部署，目前是1.11.3版本。测试环境大约50台。遇到问题是calico与ipvs知识点不足。
部署趟了不少坑都过来了，感觉kubernetes不管理网络开放cni这种，提升不少门槛。此外由于文件系统的问题，之前说work节点不能用pod部署，不知道现在解决了吗？
2018-09-17
作者回复
还不行，而且也没人太愿意试
2018-09-17


广兴 1
还有个问题，现在rook ceph版本还是beta，不能用于生产环境吧？
2018-09-17
作者回复
对，这个也没有GA，但是也很快了。
2018-09-17

Xiye 1
1. 使用过Minikube在自己本机装过K8S，步骤很简单。这个应该比较适合开发使用，不用花太多时间在部署上，但是因为是SingleNode，肯定是不能用于生产。
2. 网上查了官方，支持最多5000节点规模的集群。目前我们项目用得3个Master，3个Work Node。
2018-09-17


虎虎? 1
我们用bosh部署的CFCR，配置文件写好之后，也能做到一键部署。

我有几个问题
1. 用rook部署ceph cluster，数据的路径只能选择在node上的dataDirHostPath吗？如果我想有几台专用的storage server，组成cluster应该怎么做？我能想到的是把这几台server加入k8s cluster，然后用label/node affinity 把ceph pod调度到这几个

专用节点？
2. 我看到NFS cluster的 exports.persistentVolumeClaim 参数可以是any pvc allowed。但是我看了一下nfs的controller是创建了一个sts。如果使用的pvc是不支持readwritemany，比如host path 或者我正在用的vsphere storage。那么sts的replica设置成大于1的话，可能会创建不成功吧，除非pod都调度到同一个node上。
2018-09-17
作者回复
1.对 2.存储的时候讲
2018-09-17

Vincen 1
为什么前一篇文章说kubeadm不能用于生产环境？ 如果将etcd集群单独部署呢？
2018-09-17
作者回复
跟etcd没关系，master也高可用啊
2018-09-17

岁月～静好 1
一定要操作试试，要不翻篇就忘了，鱼的记忆
2018-09-17

心情不错 1
以后是不是开发负责部署单个docker， 运维负责宏观kubernetes?
2018-09-17
作者回复
开发负责docker和一部分yaml文件，下一篇会讲一个流程
2018-09-17

leo 0
请问张磊老师，我只部署了单节点master，是不是不能部署rook？我运行了kubectl apply -f https://raw.githubusercontent.com/rook/rook/master/cluster/examples/kubernetes/ceph/operator.yaml
之后没报错，但是rook-ceph-operator-xxx的status一直是pending，看日志无任何信息。
部署就卡在这步了，求帮助，谢谢！
2018-09-22

在路上 0
老师，部署worker节点，和master是不在同一台机器吗？同一台不行吗？用自己的笔记本，集群master是部署在virtualbox的ubuntu上的，那这些worker要怎么弄呢？
2018-09-22
作者回复
那只能单节点
2018-09-23

同心结 0

你好，我按照你的教程只用一个单节点是没有问题的。但是加入node节点之后，创建deployment，pod状态显示Successfully assigned default/myweb1-6cb867d96-mntzs to node1，然后node1一直没反应，最后会超时报错
2018-09-22
作者回复
节点沟通有问题？
2018-09-22

Gao 0
为什么不解释一些常见的问题呢？第一次部哪有那么顺啊
2018-09-22
作者回复
kubeadm存在的意义就是要让你第一次部署就这么顺，要不然这个项目就太失败了。至于GFW的问题，肯定要自己想办法解决。
2018-09-22

ninvfeng 0
开了代理提示这个:
➜ kubeadm kubeadm init --config kubeadm.yaml
I0921 08:41:04.409154 1333 feature_gate.go:230] feature gates: &{map[]}
[init] using Kubernetes version: v1.11.3
[preflight] running pre-flight checks
[WARNING HTTPProxy]: Connection to "https://10.0.2.15" uses proxy "http://192.168.33.11:1080". If that is not intended, adjust your proxy settings
[WARNING HTTPProxyCIDR]: connection to "10.96.0.0/12" uses proxy "http://192.168.33.1:1080". This may lead to malfunctional cluster setup. Make sure that Pod and Services IP ranges specified correctly as exceptions in proxy configuration
I0921 08:41:04.435667 1333 kernel_validator.go:81] Validating kernel version
I0921 08:41:04.435709 1333 kernel_validator.go:96] Validating kernel config
[WARNING SystemVerification]: docker version is greater than the most recently validated version. Docker version: 18.06.1-ce. Max validated version: 17.03
[preflight] Some fatal errors occurred:
[ERROR Swap]: running with swap on is not supported. Please disable swap
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`

没开代理提示:

➜ kubeadm kubeadm init --config kubeadm.yaml
unable to get URL "https://dl.k8s.io/release/stable-1.11.txt": Get https://storage.googleapis.com/kubernetes-release/release/stable-1.11.txt: dial tcp 172.217.31.240:443: i/o timeout
2018-09-21

陈华 0
Flannel、Calico、Canal、Romana请问老师这些和 Istio （http://www.infoq.com/cn/news/2017/05/istio
）是一个类型的东西吗，
2018-09-21
作者回复
不是。
2018-09-21

we 0
老师 生产环境的部署方法。能讲一讲吗？或者推荐一下相关文档啊。
2018-09-21
作者回复
搜kubeadm high available ，就是第一版高可用部署的文档，不过还没有GA
2018-09-21

Hurt 0
老师 我一直想问 这个集群 是在一台服务器上吗 还是多台啊
2018-09-21
作者回复
你执行了多少个kubeadm join，就有多少个工作节点
2018-09-21

每日都想上班 0
可否把master安装在比如说电信云，把node节点部署在华为云上，当然他们内部私有地址是通的
2018-09-20

Bain 0
你好，我现在遇到一个问题。部署集群的时候用的是内网IP，但是我想用公网IP访问搭好的集群里的built in services，比如heapster，但是kubectl cluster-info之后发现proxy都是的bind ip是内网IP，有没有办法改掉这个bind ip？
2018-09-20
作者回复
ingress
2018-09-20

Jeff.W 0
网络受限的话觉得可以参考一下这个，使用Ansible脚本安装K8S集群，介绍组件交互原理，方便直接，不受国内网络环境影响 https://github.com/gjmzj/kubeasz。机器资源有限就弄个AllInOne？或者单主多从？老师怎么看？

2018-09-20

作者回复

Ansible等工具适合部署大规模集群，做实验环境得不偿失，不建议分散精力。

2018-09-20

blackpiglet 0

1. 目前主要使用 Kubeadm 部署环境，试用时曾用过 Minikube，也曾经用过 Kubernetes 官方的 Ansible playbook 部署过，想尝试 kubespray 和 TerraFrom。Minikube 我理解是用虚拟机作为单节点跑 k8s 集群，具体的流程不了解，总之没有梯子是搞不定的。Kubernetes Ansible playbook 是好长时间以前用的了，感觉效果还不错，至少比我自己写的 playbook 靠谱多了...

2. Kubernetes 集群的规模限制在官方文档里说的很清楚：

No more than 5000 nodes

No more than 150000 total pods

No more than 300000 total containers

No more than 100 pods per node

现在环境也就是五台物理机。

2018-09-20

Jeff.W 0

这一节的开始配图很好，预示着需要跨越障碍，除了公司代理还有Great Wall，阻碍了我前进的步伐。。。

2018-09-20

在路上 0

```
root@ubuntu-vitualbox:~# kubeadm init --config kubeadm.yaml --ignore-preflight-errors 'Swap'
[init] using Kubernetes version: v1.11.3
[preflight] running pre-flight checks
[WARNING Swap]: running with swap on is not supported. Please disable swap
I0919 22:29:26.503206 22490 kernel_validator.go:81] Validating kernel version
I0919 22:29:26.504039 22490 kernel_validator.go:96] Validating kernel config
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of your internet connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm config images pull'
[preflight] Some fatal errors occurred:
[ERROR ImagePull]: failed to pull image [k8s.gcr.io/kube-apiserver-amd64:v1.11.3]: exit status 1
```

[ERROR ImagePull]: failed to pull image [k8s.gcr.io/kube-controller-manager-amd64:v1.11.3]: exit status 1
[ERROR ImagePull]: failed to pull image [k8s.gcr.io/kube-scheduler-amd64:v1.11.3]: exit status 1
[ERROR ImagePull]: failed to pull image [k8s.gcr.io/kube-proxy-amd64:v1.11.3]: exit status 1
[ERROR ImagePull]: failed to pull image [k8s.gcr.io/pause:3.1]: exit status 1
[ERROR ImagePull]: failed to pull image [k8s.gcr.io/etcd-amd64:3.2.18]: exit status 1
[ERROR ImagePull]: failed to pull image [k8s.gcr.io/coredns:1.1.3]: exit status 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`

老师，这里报的错误，要怎么解决？为什么拉取不了k8s的镜像？
2018-09-19

Vincent 0
不能科学上网好像就部署不了啊，老师~~
2018-09-19
作者回复
那就科学一下……
2018-09-19

Alery 0
请教一个问题，容器里挂在由Ceph提供的数据卷，那这个数据卷实际的数据存储再什么地方呢？
2018-09-19
作者回复
存储节点，也就是你部署了ceph的机器
2018-09-19

Jingslunt 0
用过rancher2.0 算是企业方案最简便，的用到rke,跟rkt部署工具类似，不过更多的是使用二进制版的装，主要的问题是证书，不用ansible有没有好一点的部署方案？
2018-09-19

or0or1 0
为什么不搭建ingress？这个作为应用入口，也很重要吧？
2018-09-18
作者回复
因为ingress后面会专门讲
2018-09-18

Switch 0
使用kubernetesVersion: "stable-1.11"报错"unable to get URL "https://dl.k8s.io/release/stable-1.11.1.txt": Get https://

storage.googleapis.com/kubernetes-release/release/stable-1.11.1.txt: dial tcp 172.217.161.144:443: i/o timeout"，使用版本"v1.11.1"没问题，建议更改版本号
2018-09-18
作者回复
这是gfw吧
2018-09-18

他说风雨中 0
老师您好，想问一下您kubernetes1.10是怎么支持cpuset，cpushare的，是在deployment的yaml文件里面吗？并且问一下k8s是回自动进行cpu管理还是手动去管理？由于网上材料很少，只能在这里问您，希望您能指点一下
2018-09-18
作者回复
后面专门有一篇讲资源管理的
2018-09-18


jssfy 0
1. rook 目前只提供cepg支持？
2. 如果我集群中每台机器都已经挂载nfs，那么nfs存储插件对我来说是否就没有必要了？
3. 目前的公有云是否有采用k8s方案？如果有，不同用户的pod如果调度到同一台物理机上，是否会有隔离问题，如之前一节有类似提到，还是有些隔离问题存在的？
4. 部署k8s到公有云或者私有云，会分别建议部署在虚拟机或者裸金属机上吗？
多谢多谢！
2018-09-17
作者回复
其他支持已经陆续添加了。有必要，因为要用PVC PV。隔离问题非常大。公有云上只能虚拟机。
2018-09-18


小鱼儿 0
老师，请问你的“-”是怎么读的？
2018-09-17
作者回复
dash
2018-09-18

余小坝 0
常用ansible部署kubernetes集群，感觉也挺快的。

老师，请问如果master节点的网段是：192.168.....，新的节点的网段是172.18.....，那这个节点能否加入到master的集群中啊？要如何做呢
2018-09-17
作者回复
多网卡能通就行，注意端口监听的是哪个网卡
2018-09-18

undifined 0
老师 像阿里云 AWS GCP 这些平台的一键部署是用什么实现的呢
2018-09-17
作者回复
自己编写的运维脚本，相当复杂，必须专门团队维护
2018-09-17

风轨 0
最近一次部署成功kubernetes是:
1. VPN确保网络畅通;
2. 安装最新版docker;
3. 在设置页面勾上kubernetes功能;
4. 等待足够长的时间;
win10环境。

之前试了minikube，CoreOS都没有成功，看了今天的文章似乎知道我CoreOS部署方式为啥失败了，找时间再试试~
2018-09-17

死后的天空 0
老师最近遇到一个问题，由于我在集群中部署了不少数据库，当有大量的读写的时候宿主机的buff/cache上升很快，会几倍于应用占用的内存，遇到这类情况，该怎么处理呢，Kubernetes有没有类似的缓存释放机制。
2018-09-17
作者回复
这个得手动运维了
2018-09-17

13042162180 0
centos上安装与Ubuntu上安装差别是不是挺大的
2018-09-17
作者回复
不会啊
2018-09-17

广兴 0
老师，pod挂载rook ceph存储之后，要怎么查看挂载的内容呢
2018-09-17
作者回复
exec进去
2018-09-17

huan 0
分享一下这篇实践的小插曲，我是使用DigitalOcean的机器做的，没有墙的问题。
在使用weave-net的时候，怎么都通不过，报错是 weave-net 的状态是 CrashLoopBackOff，看了container的log，也不懂啥意思，找了下k8s的troubleshoot页面的说明（ https://kubernetes.io/docs/setup/independent/troubleshooting-kube

adm/#pods-in-runcontainererror-crashloopbackoff-or-error-state），说是版本问题，不知道怎么解决。
然后随便找了另一个实现 Romana，替换weavenet
`kubectl apply -f https://raw.githubusercontent.com/romana/romana/master/containerize/specs/romana-kubeadm.yml` 终于成功了，然后体验到了以前想都不敢想的Ceph，超赞
2018-09-17
作者回复
rook.io真的是个好东西
2018-09-17

广兴 0
老师您好， 请教一下，springboot项目避暑在k8s中 项目的日志要怎么处理呢 要使用pv吗
2018-09-17
作者回复
是的
2018-09-17

pytimer 0
使用过ansible和纯二进制文件部署集群，证书之类的比较麻烦，而且添加节点我不方便。
现在Kubernetes集群官方最大节点是5000吧，我自己的环境的话就6个节点。

老师，想要问下，我现在使用kubeadm来部署生产环境的高可用可以吗？有什么风险吗？
另外，在使用kubeadm join添加节点的时候，token过期怎么处理？我现在都是设置ttl=0，但是感觉这种不太好
2018-09-17
作者回复
小规模生产环境没问题。需要执行kubeadm upgrade
2018-09-17

Jim 0
存储插件是在单独一台机器上安装吗？那两个命令是否也应该支持分布式？存储设备都隐藏了，如何指定呢？

网络插件应该只在Master上安装吧？
2018-09-17
作者回复
都是些yaml文件，命令行执行即可
2018-09-17

w 0
如果无法连接gcr.io可以从docker官方registry下载其他人已经制作好的镜像，然后tag成gcr.io。比如有一个mirrorgooglecontainers的仓库。

请问老师，持久化插件需要在worker节点也部署，还是只需要在master节点？
2018-09-17
作者回复
都是YAML文件，创建出来kubernetes会给你在各个节点上启动pod的，要么怎么能说一键部署呢……
2018-09-17

Geek_700d17 0
目前生产二十多个节点，k8s能有效管理的节点还没注意到这个知识点。开始接触容器用的比较多的是rancher ，使用rancher自己的编排工具cattle ，用着还不错。确如老师所讲rancher 确实把很多细节都封装了。
2018-09-17
作者回复
5000
2018-09-17