十分钟带你理解Kubernetes核心概念

作者：chenhaozjnubit　分类：　发布于：2017-4-27 9:12　137次浏览　0条评论

本文将会简单介绍Kubernetes的核心概念。因为这些定义可以在Kubernetes的文档中找到，所以文章也会避免用大段的枯燥的文字介绍。相反，我们会使用一些图表（其中一些是动画）和示例来解释这些概念。我们发现一些概念（比如Service）如果没有图表的辅助就很难全面地理解。在合适的地方我们也会提供Kubernetes文档的链接以便读者深入学习。

这就开始吧。

什么是Kubernetes?

Kubernetes（k8s）是自动化容器操作的开源平台，这些操作包括部署，调度和节点集群间扩展。如果你曾经用过Docker容器技术部署容器，那么可以将Docker看成Kubernetes内部使用的低级别组件。Kubernetes不仅仅支持Docker，还支持Rocket，这是另一种容器技术。

使用Kubernetes可以：

- 自动化容器的部署和复制
- 随时扩展或收缩容器规模
- 将容器组织成组，并且提供容器间的负载均衡
- 很容易地升级应用程序容器的新版本
- 提供容器弹性，如果容器失效就替换它，等等...

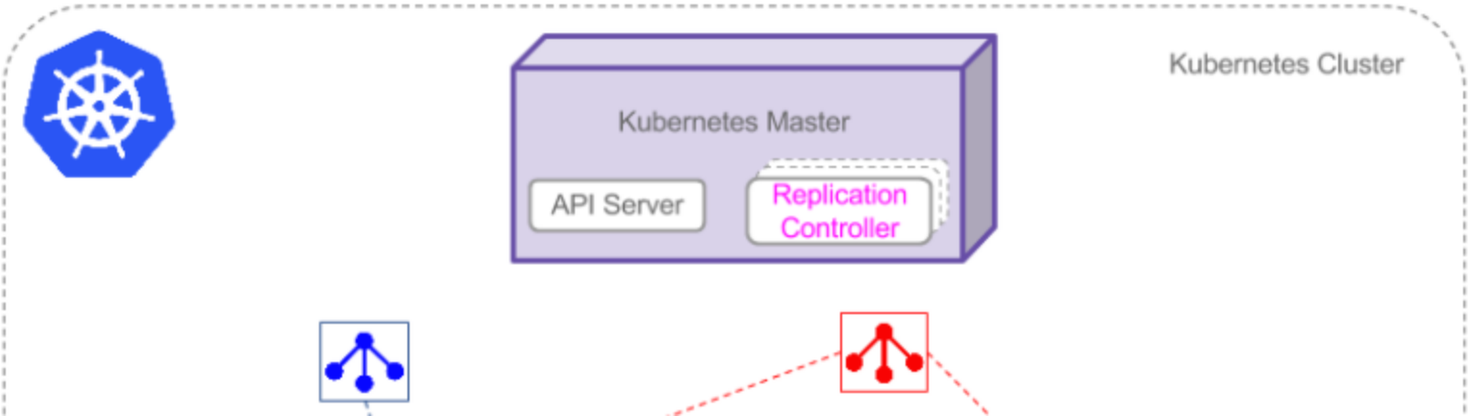
实际上，使用Kubernetes只需一个部署文件，使用一条命令就可以部署多层容器（前端，后台等）的完整集群：

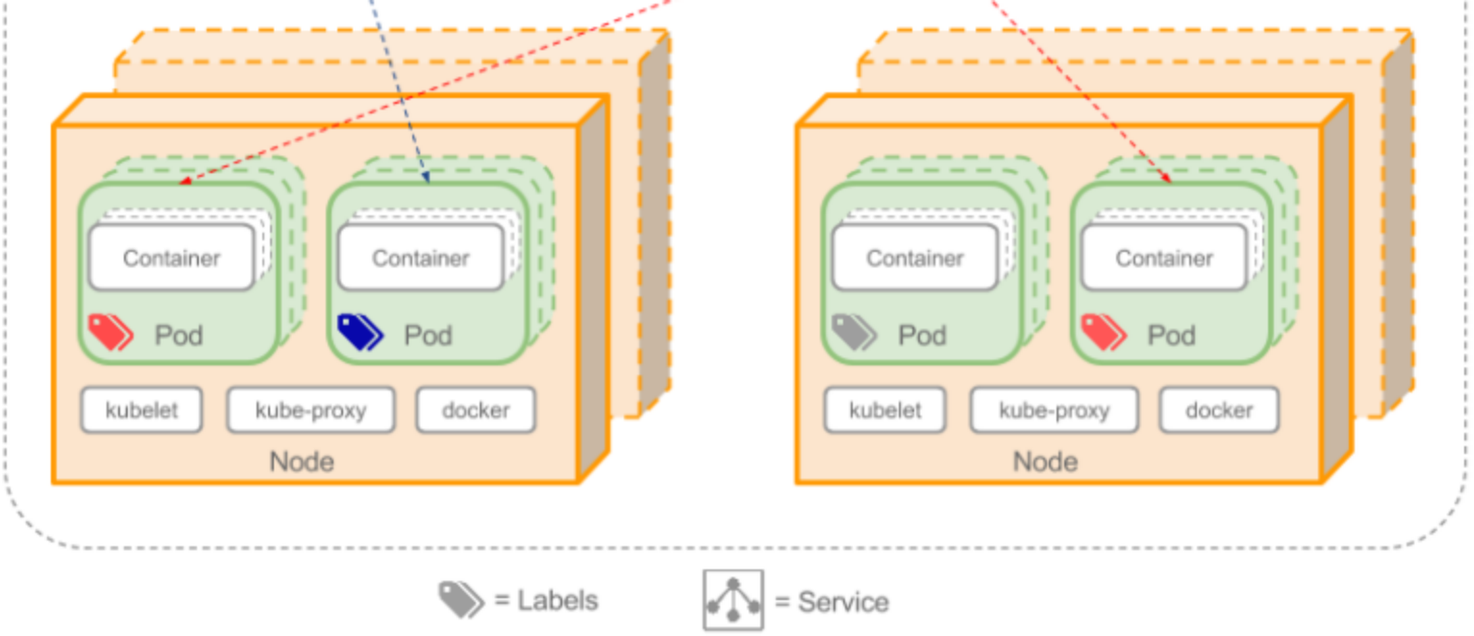
```
$ kubectl create -f single-config-file.yaml
```

kubectl是和Kubernetes API交互的命令行程序。现在介绍一些核心概念。

集群

集群是一组节点，这些节点可以是物理服务器或者虚拟机，之上安装了Kubernetes平台。下图展示这样的集群。注意该图为了强调核心概念有所简化。这里可以看到一个典型的Kubernetes架构图。





上图可以看到如下组件，使用特别的图标表示Service和Label：

- Pod
- Container（容器）
- Label( )（标签）
- Replication Controller（复制控制器）
- Service（ ）（服务）
- Node（节点）
- Kubernetes Master（Kubernetes主节点）

Pod

Pod（上图绿色方框）安排在节点上，包含一组容器和卷。同一个Pod里的容器共享同一个网络命名空间，可以使用localhost互相通信。Pod是短暂的，不是持续性实体。你可能会有这些问题：

- 如果Pod是短暂的，那么我怎么才能持久化容器数据使其能够跨重启而存在呢？ 是的，Kubernetes支持卷的概念，因此可以使用持久化的卷类型。
- 是否手动创建Pod，如果想要创建同一个容器的多份拷贝，需要一个个分别创建出来么？可以手动创建单个Pod，但是也可以使用Replication Controller使用Pod模板创建出多份拷贝，下文会详细介绍。
- 如果Pod是短暂的，那么重启时IP地址可能会改变，那么怎么才能从前端容器正确可靠地指向后台容器呢？这时可以使用Service，下文会详细介绍。

Lable

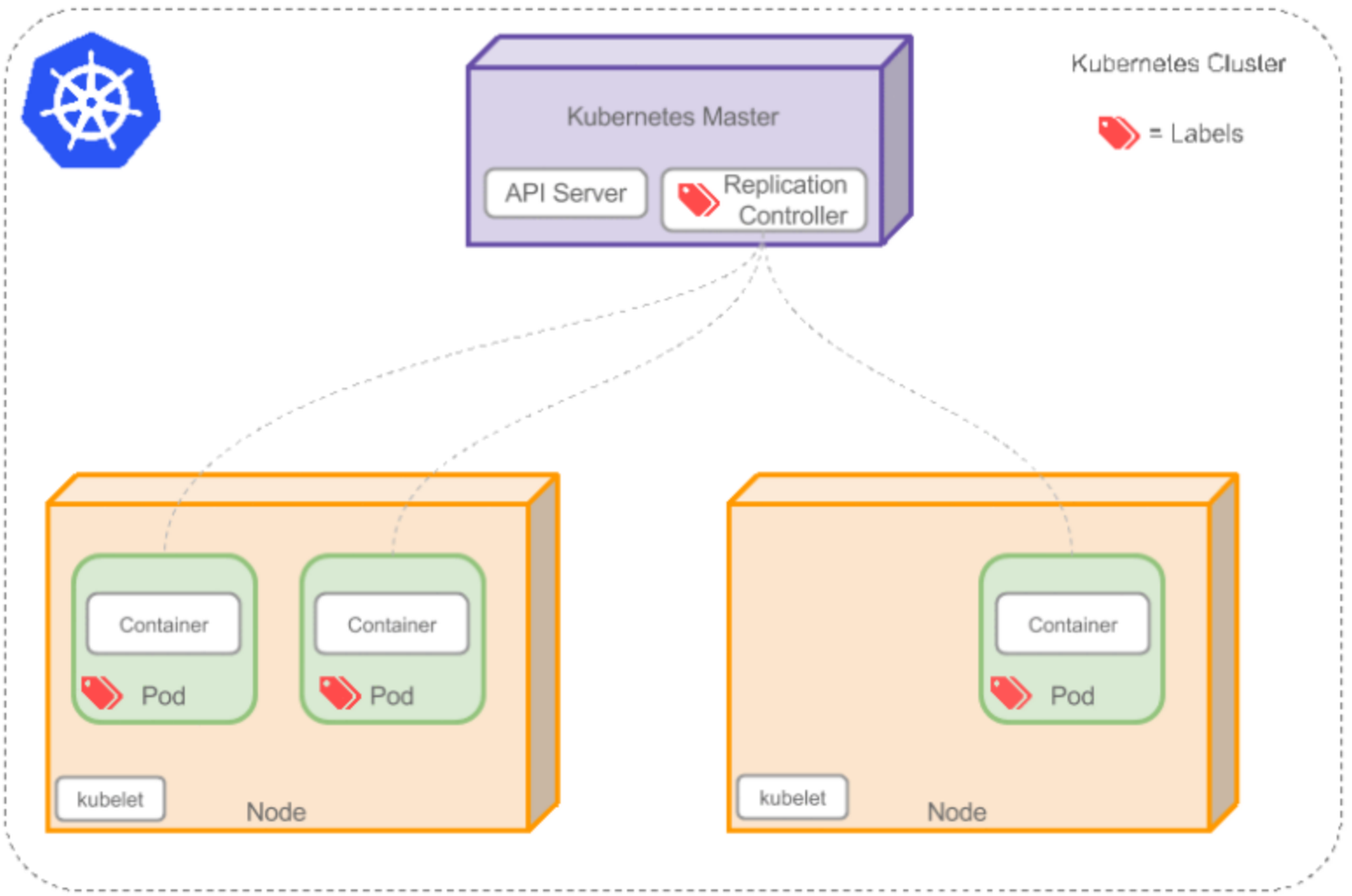
正如图所示，一些Pod有Label（ ）。一个Label是attach到Pod的一对键/值对，用来传递用户定义的属性。比如，你可能创建了一个"tier"和“app”标签，通过Label（**tier=frontend, app=myapp**）来标记前端Pod容器，使用Label（**tier=backend, app=myapp**）标记后台Pod。然后可以使用Selectors选择带有特定Label的Pod，并且将Service或者Replication Controller应用到

上面。

Replication Controller

是否手动创建Pod，如果想要创建同一个容器的多份拷贝，需要一个个分别创建出来么，能否将Pods划到逻辑组里?

Replication Controller确保任意时间都有指定数量的Pod“副本”在运行。如果为某个Pod创建了Replication Controller并且指定3个副本，它会创建3个Pod，并且持续监控它们。如果某个Pod不响应，那么Replication Controller会替换它，保持总数为3.如下面的动画所示:



如果之前不响应的Pod恢复了，现在就有4个Pod了，那么Replication Controller会将其中一个终止保持总数为3。如果在运行中将副本总数改为5，Replication Controller会立刻启动2个新Pod，保证总数为5。还可以按照这样的方式缩小Pod，这个特性在执行滚动升级时很有用。

当创建Replication Controller时，需要指定两个东西:

1. Pod模板：用来创建Pod副本的模板
2. Label：Replication Controller需要监控的Pod的标签。

现在已经创建了Pod的一些副本，那么在这些副本上如何均衡负载呢？我们需要的是Service。

Service

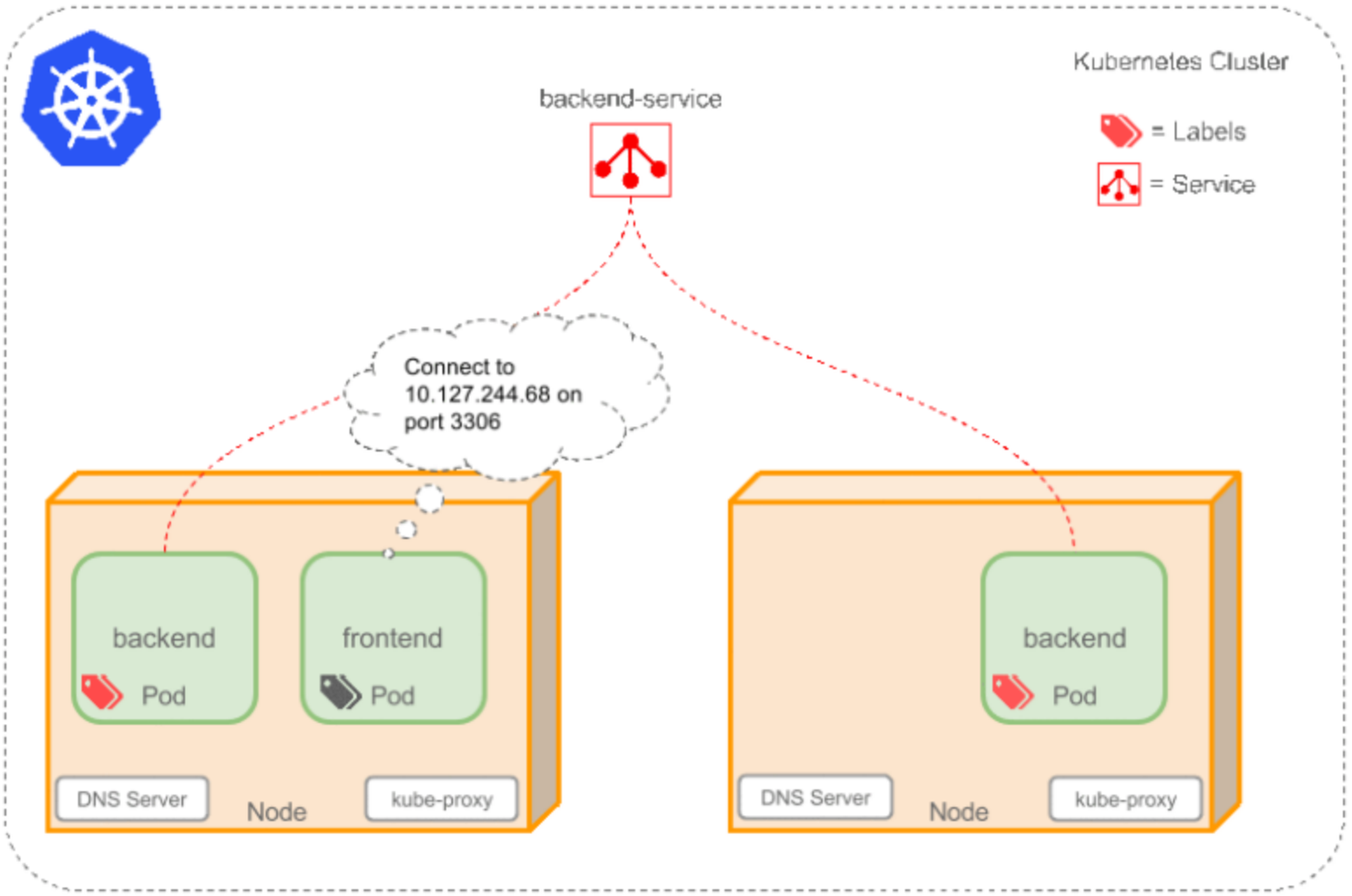
如果Pods是短暂的，那么重启时IP地址可能会改变，怎么才能从前端容器正确可靠地指向后台容器呢?

Service是定义一系列Pod以及访问这些Pod的策略的一层**抽象**。Service通过Label找到Pod组。因为Service是抽象的，所以在图表里通常看不到它们的存在，这也就让这一概念更难以理解。

现在，假定有2个后台Pod，并且定义后台Service的名称为‘backend-service’，lable选择器为（**tier=backend, app=myapp**）。backend-service 的Service会完成如下两件重要的事情：

- 会为Service创建一个本地集群的DNS入口，因此前端Pod只需要DNS查找主机名为‘backend-service’，就能够解析出前端应用程序可用的IP地址。
- 现在前端已经得到了后台服务的IP地址，但是它应该访问2个后台Pod的哪一个呢？Service在这2个后台Pod之间提供透明的负载均衡，会将请求分发给其中的任意一个（如下面的动画所示）。通过每个Node上运行的代理（kube-proxy）完成。这里有更多技术细节。

下述动画展示了Service的功能。注意该图作了很多简化。如果不进入网络配置，那么达到透明的负载均衡目标所涉及的底层网络和路由相对先进。如果有兴趣，这里有更深入的介绍。



有一个特别类型的Kubernetes Service，称为'LoadBalancer'，作为外部负载均衡器使用，在一定数量的Pod之间均衡流量。比如，对于负载均衡Web流量很有用。

Node

节点（上图橘色方框）是物理或者虚拟机器，作为Kubernetes worker，通常称为Minion。每个节点都运行如下Kubernetes关键组件：

- Kubelet：是主节点代理。
- Kube-proxy：Service使用其将链接路由到Pod，如上文所述。
- Docker或Rocket：Kubernetes使用的容器技术来创建容器。

Kubernetes Master

集群拥有一个Kubernetes Master（紫色方框）。Kubernetes Master提供集群的独特视角，并且拥有一系列组件，比如Kubernetes API Server。API Server提供可以用来和集群交互的REST端点。master节点包括用来创建和复制Pod的Replication Controller。

下一步

现在我们已经了解了Kubernetes核心概念的基本知识，你可以进一步阅读Kubernetes 用户手册。用户手册提供了快速并且完备的学习文档。

如果迫不及待想要试试Kubernetes，可以使用Google Container Engine。Google Container Engine是托管的Kubernetes容器环境。简单注册/登录之后就可以在上面尝试示例了。

原文链接：Learn the Kubernetes Key Concepts in 10 Minutes（翻译：崔婧雯）