# K8S 学习及应用

**一：总体概述**

Kubernetes: 简称k8s，叫容器编排引擎。

同类的编排引擎有：mesos, docker swarm, 等

目前，AWS、Azure、Google、阿里云、腾讯云等主流公有云提供的是基于 Kubernetes 的容器服务。

K8s平台包含：计算，网络，存储，高可用，监控，日志管理等多方面。

起源：是google 2014年发布的一个开源项目。

学习这里以版本：V1.7,V1.8为主

kubernetes.io 开发了一个交互式教程：

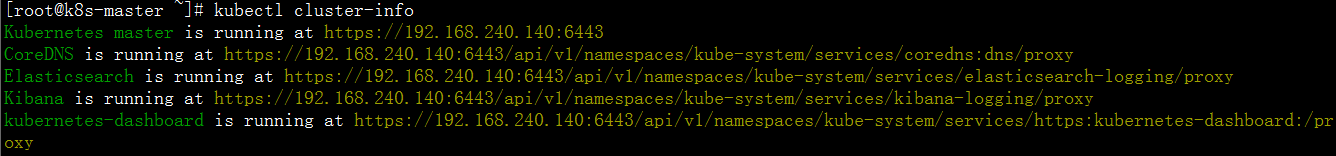
https://kubernetes.io/docs/tutorials/kubernetes-basics/

但是我目前网络似乎无法访问。

实验环境我已经按照我的文档，搭建完成一套，结构是：一个master + 2个node

查看集群的命令：kubectl cluster-info

我已安装完一个测试集群，如下：



其中：CoreDNS，kibanna都是集群中的服务。

举例：部署一个应用命令，名称为：kubernetes-bootcamp

Kubectl run kubernetes-bootcamp \

--image=docker.io/jocatalin/kubernetes-bootcamp:v1 \

--port=8080

Image：为指定的docker镜像

Port：为对外服务的端口

**二：k8s重要概念**

**Pod**

Pod：容器的集合，将紧密相关的一组容器放到一个pod中，同一个pod中所有的容器共享IP地址和port空间，也就是在一个network namespace中。

Pod是k8s最小的调度单位，同一个pod中的一个或多个容器始终作为一个整体调度到一个Node上，同时运行或同时停止。

**K8s引入pod主要基于2个目的：**

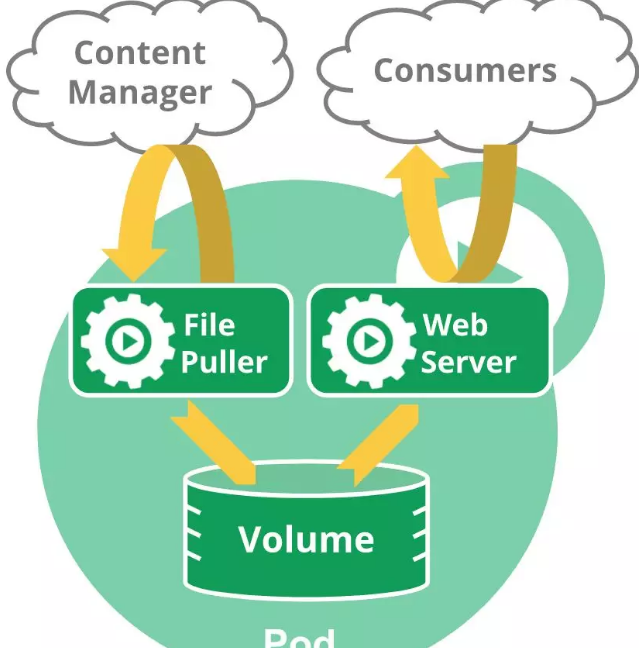
1：可管理性：有些容器需要紧密联系，一起工作，通过pod将他们封装到一个部署单元中，对pod进行调度，扩展，共享资源，管理生命周期

2：通信和资源共享：pod中的所有容器使用同一个网络namespace，即相同的ip地址和port空间，可以直接用localhost通信。 同样，这些容器共享存储，当k8s挂载volume到pod，本质上是将volume挂载到pod中的每一个容器。

**有2中使用方式**：

1：运行单一容器：就是将单个容器简单封装成pod，pod通过K8s进行管理。

2：运行多个容器：要多个容器放在一个Pod中，条件是容器间必须紧密联系，而且需要直接共享资源。举个例子：



解释：file puller会定期从外部的content manager中拉取最新的文件，将其放在共享的volume中，web server从volume中读取文件，以响应consumer的请求。

说明：这2个容器是紧密联系的，它们一起为consumer提供最新的数据，同时他们也是通过volume共享数据，所以放到一个Pod中很合适。

查看当前的Pod：kubectl get pods

访问应用：默认情况下，所有的Pod只能在集群内部访问，如果要从外部访问，需要将端口映射到节点上，如下：

Kubectl expose deployment/kubernetes-bootcamp \

--type=”NodePort” --port 8080

然后执行：kubectl get services查看应用被映射到节点的端口

然后可以通过：curl 节点IP：映射的端口 进行访问测试。

副本数：默认情况下，应用只会运行一个副本。查看命令：

Kubectl get deployment

举例增加副本数到3个，命令如下：

Kubectl scale deployment/kubernetes-bootcamp –replicas=3

然后通过：kubectl get pods 可以查看容器副本数，继续通过curl 访问查看负载均衡功能。

Curl node:port

重新调整副本数为2个，只需要将replicas=2 即可。多的副本将会被自动删除。

滚动更新：升级当前容器副本版本，从image的v1版本升级到v2，命令如：

kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2

以上为滚动更新，V1的POD逐个被删除，同时启动V2的POD，观察滚动更新命令：

Kubectl get pods

更新完成后通过： curl host01:port 查看更新的版本。

回退方法：kubectl rollout undo 回退以上实例：

Kubectl rollout undo deployment/kubernetes-bootcamp

验证还是以： curl host01:port

**Cluster：**是计算，存储，网络资源的集合，k8s利用这些资源进行各种容器的管理

**Master：**是cluster的大脑，控制节点，主要功能：调度，决定将应用放在哪个node节点运行，主要运行在linux操作系统上，可以是物理机或虚拟机，可部署多个master实现高可用。

**Node：**运行容器应用，node由master进行管理，node负责监控并汇报容器的运行状态，并根据master的要求管理容器的生命周期，也允许在Linux上，可以是物理机或者虚拟机。

Controller：k8s不会直接创建pod，是通过controller来管理pod，controller中定义了pod的部署特性，比如：有几个副本，在什么样的node上运行等，在不同的业务场景，k8s提供了多种controller，如：deployment,replicaset,daemonset,statefuleset,job等。

* Deployment：最常用的controller，可以管理pod的多个副本，确保pod按照期望的状态运行。
* Replicaset：实现了pod的多副本管理。使用deployment时会自动创建replicaset，也就是说deployment是通过replicaset来管理多个pod副本的，所以我们通常不需要直接使用replicaset。
* Daemonset：用于每个node节点最多只运行一个pod副本的场景，正如名称，daemonset通常用于运行daemon。
* StatefuleSet：能够保证pod的每个副本在整个生命周期中名称是不变的。如其他controller，当pod发生故障删除并重启后，pod名称就会发生变化。同时statefuleset会保证副本按照固定的顺序启动，更新或删除。
* Job：用于运行结束就删除的应用，其他controller中的pod通常为长期运行。

Service：定义了外界访问一组特定pod的方式。Service有自己的ip和端口，service为pod提供了负载均衡。

所以：k8s运行容器有controller负责，访问容器由service负责。

Namespace：将一个物理的cluster逻辑上划分成多个虚拟的cluster，每个cluster就是一个namespace，不同namespace里的资源是完全隔离的。

K8s默认创建了2个namespace： default , kube-system

查看命令： kubectl get namespace

Default：创建的资源默认存储在这个namespace中

Kube-system：k8s自己创建的系统资源存储位置。

三：k8s cluster

举例部署一个三节点的： cluster  
一个master， 二个node (node1,node2)

官方安装文档可以参考 <https://kubernetes.io/docs/setup/independent/install-kubeadm/>

K8S部署文档，请查看我的部署手册。我的部署手册是将master和Node 进行了独立部署。

到这里需要已经部署完成一套k8s系统用于测试。

K8s内各个命令的解释：

Kubelet：运行在node节点上，负责启动pod和容器，由master上的apiserver接口调用。

Kubeadm：用于初始化cluster

Kubectl：k8s命令行工具，可以部署和管理应用，查看各种资源；创建，删除，更新各组件。

使用kubeadm创建集群的官方文档请参考：

https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/

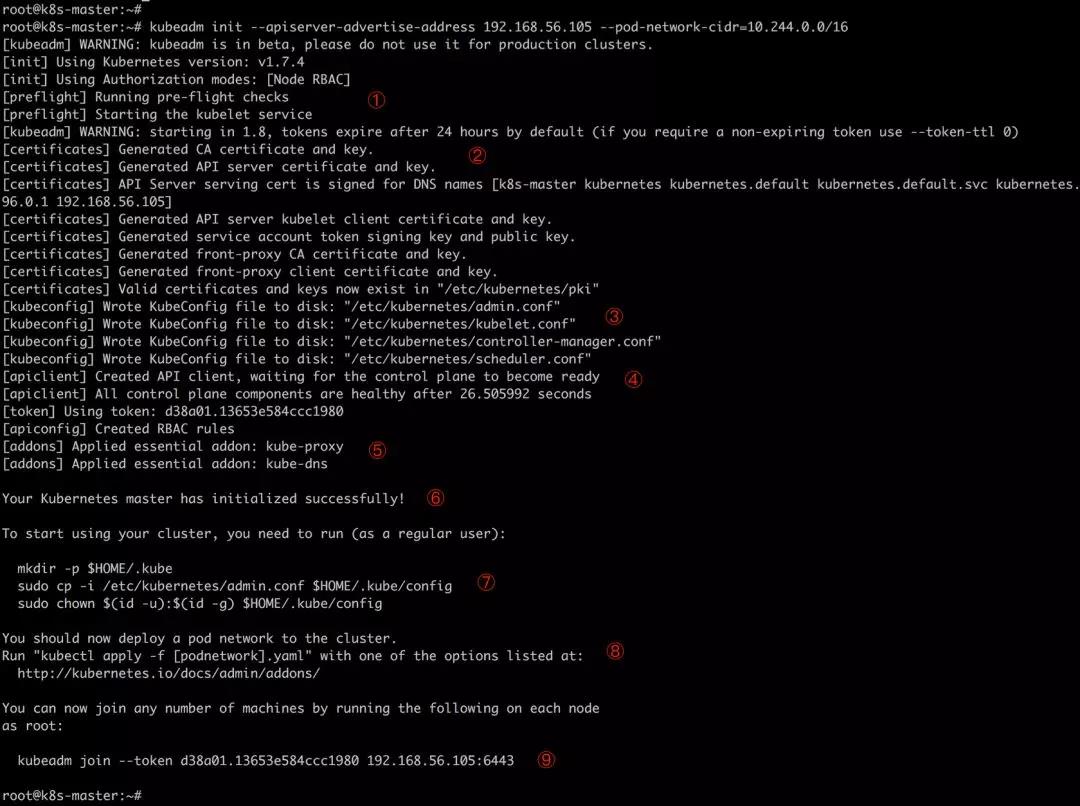
初始化master的命令：

kubeadm init --apiserver-advertise-address 192.168.56.105 --pod-network-cidr=10.244.0.0/16

--apiserver-advertise-address 指明master的哪个接口与cluster的其他节点通信，当master有多个interface，建议指定，否则kubeadm会自动选用网关的接口。

--pod-network-cidr 指定Pod的网络范围，k8s支持多种网络方案，而且不同网络方案对—pod-network-cidr 有自己的要求，这里设置的10.244.0.0/16 是因为我们使用的是flannel网络方案，必须设置成CIDR，也可以切换为其他的网络方案。

举例以下为初始化的截图及解释



1：表示kubeadm初始化前的检查

2：生成证书和token

3：生产kubeconfig文件，kubelet需要这些文件与master通信

4：安装master组件，会从google的registry下载组件的docker镜像，可能会有网络问题。

5：安装附加组件：kube-proxy和kube-dns

6：k8s master初始化成功

7：提示如何配置kubectl

8：提示如何安装Pod网络

9：提示如何注册其他节点到cluster

配置kubectl

Kubectl是管理k8s cluster的命令行工具，以上是在所有节点安装了kubectl，master初始化完成后需要做一些配置工作，然后kubectl就能使用了。

根据第7步提示，推荐linux使用普通用户执行。

su – k8s

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

为了使用更便捷，启用 kubectl 命令的自动补全功能。

echo "source <(kubectl completion bash)" >> ~/.bashrc

这样 k8s 用户就可以使用 kubectl 了。

以上完成了在k8s的master上部署完成kubernetes

接下来安装pod网络并添加k8s的节点，完成集群的部署

安装pod网络

我们这里以安装flannel网络为例。如下：

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

添加k8s-node1和k8s-node2

在node1,node2上分别执行如下，使其注册到cluster中

kubeadm join --token d38a01.13653e584ccc1980 192.168.56.105:6443

这里token是来自前面kubeadm init输出的第9步提示，命令获取方式如下：

Kubeadm token list

查看节点状态：kubectl get nodes

发现节点都是NotReady状态，是因为每个节点都需要启动若干组件，这些组件都是在pod中运行，需要从google下载镜像，以下命令查看pod的状态：

Kubectl get pod –all-namespaces

kube-system coredns-77c989547b-d2ll9 0/1 ImageInspectError 0 1h

kube-system elasticsearch-logging-0 0/1 PodInitializing 0 37m

kube-system fluentd-es-v2.7.0-82v5f 1/1 Running 0 30m

kube-system fluentd-es-v2.7.0-knkv7 0/1 ContainerCreating 0 5m

kube-system kibana-logging-7c9c85df74-64sm6 0/1 Evicted 0 2h

Pending、ContainerCreating、ImagePullBackOff 都表明 Pod 没有就绪

Running 为就绪状态

查看pod具体情况：

Kubectl describe pod <pod name>

如果发现镜像下载失败，可以手工执行下载：docker pull

四：k8s架构

K8s cluster 组成：master , node

Master必须运行如下服务：kube-apiserver, kube-scheduler, kube-controller-manager, etcd,

可选：pod网络（flannel）

服务解释：

Api server（kube-apiserver）：提供http/https restful api功能，api server是k8s cluster的前端接口，各种客户端工具（cli,ui）及k8s其他组件可以通过apiserver来管理cluster的各种资源。

Scheduler（kube-scheduler）：指定将pod放在哪个节点上运行，scheduler在调度时会充分考虑cluster的拓扑结构和当前节点的负载，以及应用对高可用，性能，数据亲和性的需求。

Controller manager(kube-controller-manager)：管理cluster各种资源，保证资源处于预期的状态，controller manager由多种controller组成，包括：replication controller,endpoints controller,

Namespace controller,service accounts controller等。

不同的controller管理不同的资源，如：replication controller管理deployment,statefulset,daemonet的生命周期，namespace controller管理namespace资源。

Etcd：非关系型数据库，保存cluster的配置信息和各种资源的状态信息，当数据发生变化时，etcd会快速的通知K8s相关组件。

Pod网络：pod间相互通信，flannel是其中一种可选的网络组件。

Node节点：是pod运行的地方，k8s支持docker,rkt等容器runtime。

Nde上运行的组件有：kubelet, kube-proxy, pod网络（flannel），可选的有etcd，

组件解释：

Kubelet：是node的agent，当scheduler确定在哪个node上运行pod后，会将pod的具体配置信息(image,volume等)发送给该节点的kubelet，kubelet根据这些信息创建和运行容器，并向master报告运行状态。

Kube-proxy：service在逻辑上代表了后端的多个pod，外界通过service访问pod，service接收到的请求转发到pod，就是通过kube-proxy来完成的。

每个node都会运行kube-proxy服务，它负责将访问service的TCP/UDP数据流转发到后端容器，如果有多个副本，proxy还实现负载均衡。

Pod容器：实现pod相互通信。

Kubelet是唯一一个没有以容器形式运行的K8s组件。是通过系统服务启动。

五：实例讲解k8s架构

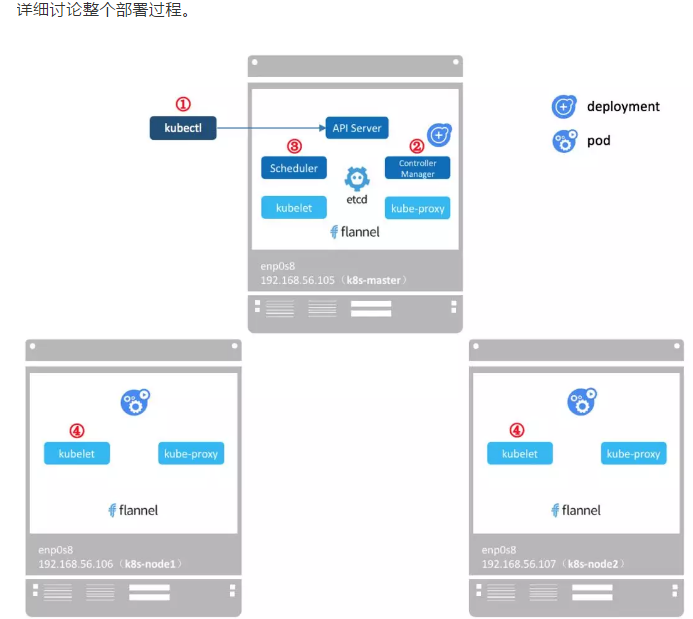
部署一个web容器，2副本数

Kubectl run httpd-app –image=httpd –replicas=2

部署完成后，查看：

Kubectl get deployment

Kubectl get pod –o wide



步骤如下：

1：kubectl发送部署请求到api server

2：api server通知controller manager创建一个deployment资源

3：scheduler执行调度任务，将俩个副本pod分发到node1和node2上

4：各节点上的kubelet进行pod的创建并运行。

补充说明：

1：应用的配置和当前状态信息保存在etcd中，执行kubectl get pod时api server会从etcd中读取这些数据。

2：flannel会为每个pod都分配ip，因为没有创建service，目前kube-proxy还没有参与进来。

Kubernetes通过controller来管理Pod的生命周期，controller包含如：deployment, replicaset,daemonset,statefuleset,job等

**六：通过deployment运行应用**

运行一个包含2个副本的deployment，容器images版本为1.7.9

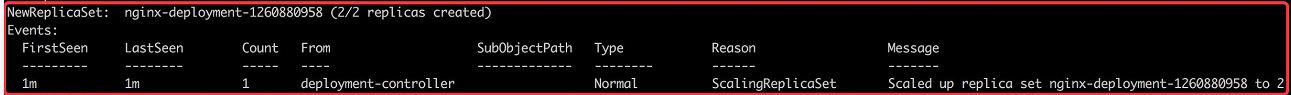
Kubectl run nginx-deployment –image=nginx.1.7.9 –replicas=2

查看nginx-deployment的状态，输出副本数和状态

Kubectl get deployment nginx-deployment

了解更详细的信息

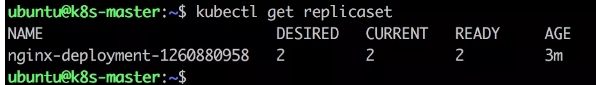
Kubectl describe deployment



在输出的最下面出现如上提示，表示创建了一个 ReplicaSet nginx-deployment-1260880958，Events 是 Deployment 的日志，记录了 ReplicaSet 的启动过程。

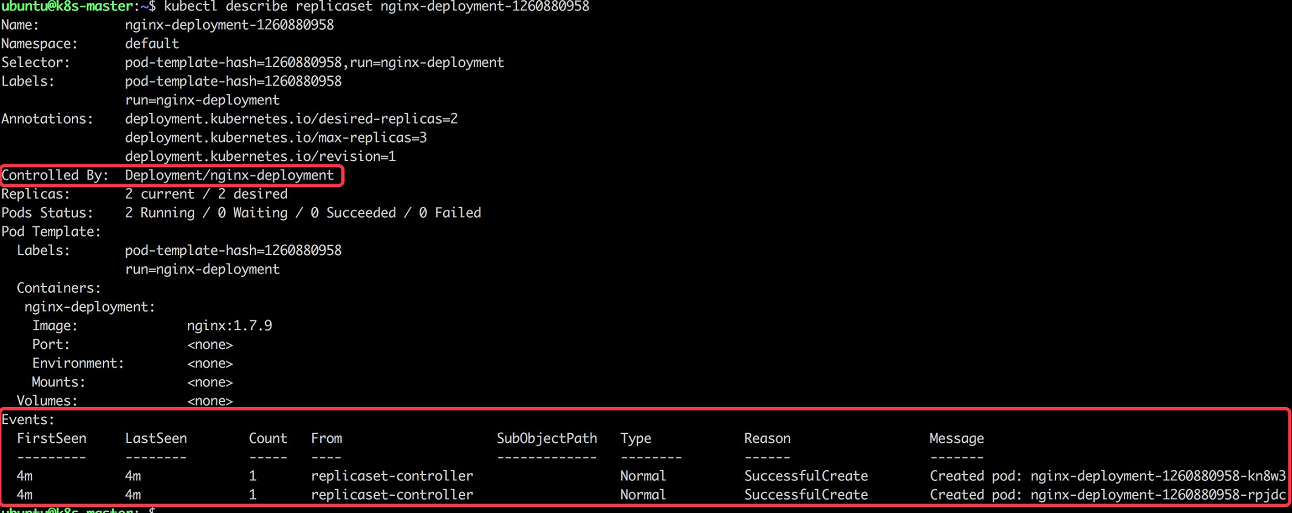
如上充分说明了deployment是通过replicaset来管理pod的事实

获取所有deployment的replicaset



获取nginx-deployment-1260880958副本的详细信息：

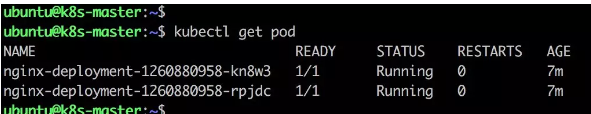
Kubectl describe replicaset



Controller by：指明此replicaset是由deployment nginx-deployment创建。

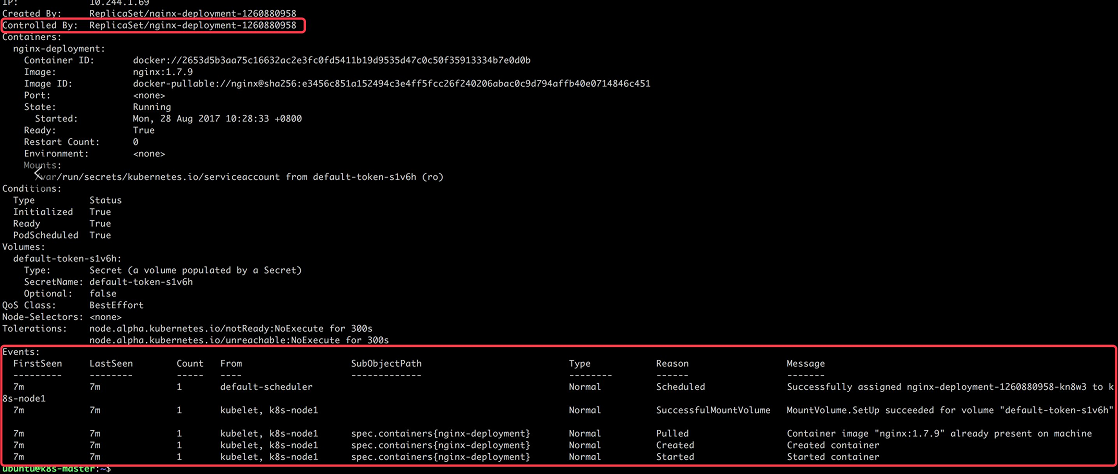
Events：记录了2个副本pod的创建。

获取pod的信息： kubectl get pod



表示2个副本都处于running状态

继续查看pod的详细信息： kubectl describe pod



Controlled By 指明此 Pod 是由 ReplicaSet nginx-deployment-1260880958 创建。

Events 记录了 Pod 的启动过程。如果操作失败（比如 image 不存在），也能在这里查看到原因。

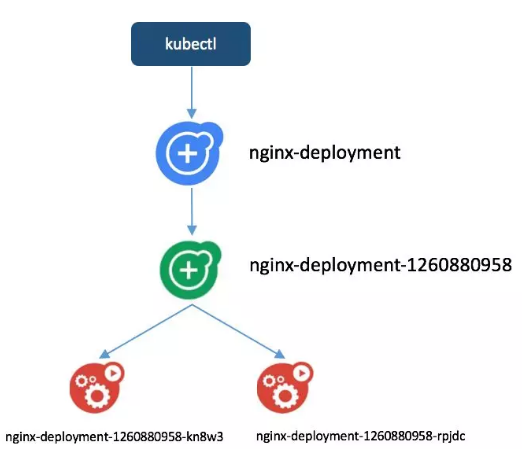
Pod创建流程总结：

1：用户通过kubectl创建deployment

2：deployment创建了replicaset

3：replicaset创建了pod

流程图如下：



上图说明了：子对象的名字 = 父对象的名字 + 随机字符串

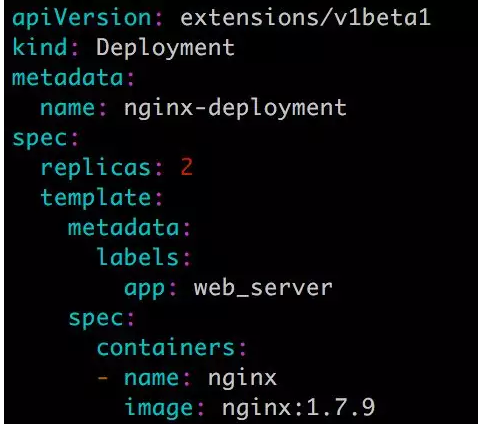
**七：创建资源的俩种方式**

第一种：命令，通过kubectl创建

Kubectl run nginx-deployment –image=nginx:1.7.9 –replicas=2

第二种：通过配置文件和kubectl apply创建，比如：

资源的配置文件为yaml格式，Nginx.yml的内容：



然后执行：kubectl apply -f nginx.yml

以上2种方式比较：

基于命令的：

1：简单直观快捷，上手快

2：适合临时测试和实验

基于配置文件的：

1：配置文件描述了应用要最终达到的状态

2：配置文件提供了创建资源的模板，能够重复使用

3：可以像管理代码一样管理部署

4：适合正式的，跨环境的，规模化的部署

5：这种方式要求熟悉配置文件的语法。

后续要以正式配置文件方式部署。

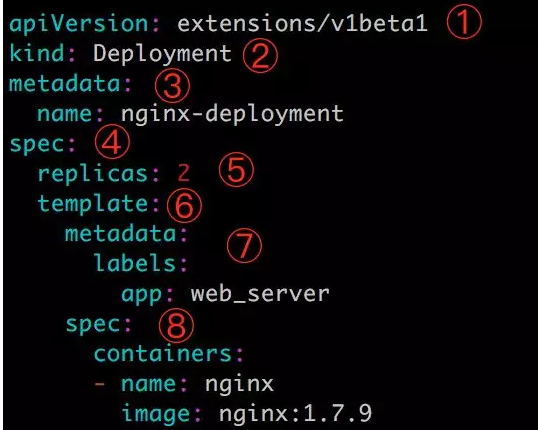
Kubectl apply不仅能够创建资源，还可以对资源进行更新。

另外：kubernetes还提供了几个类似的命令:

有：kubectl create,kubectl replace,kubectl edit,kubectl path

**YAML配置文件理解**

以deployment配置格式为例，其他controller类似



1：当前配置格式的版本

2：要创建的资源类型，这里是：deployment

3：资源的元数据，name是必须的元数据项

4：该deployment的规格说明

5：副本数量，默认是1

6：定义pod的模板，这是配置文件的重要部分

7：定义pod的元数据，至少要有一个labels，label的key和value可以任意指定

8：pod的规格，这里定义了pod中每一个容器的属性，name,image是必须的。

以上是一个最简单的nginx-deployment的配置文件。

运行命令：kubectl apply -f nginx.yml

查看命令：kubectl get nginx-deployment 查看nginx-deployment的各种资源



Deployment,replicaset,pod都已经就绪，如果删除资源，执行如下：

Kubectl delete deployment nginx-deployment 或者 kubectl delete -f nginx.yml

**Scale Up/Down**