# Kubernetes之kubectl常用命令

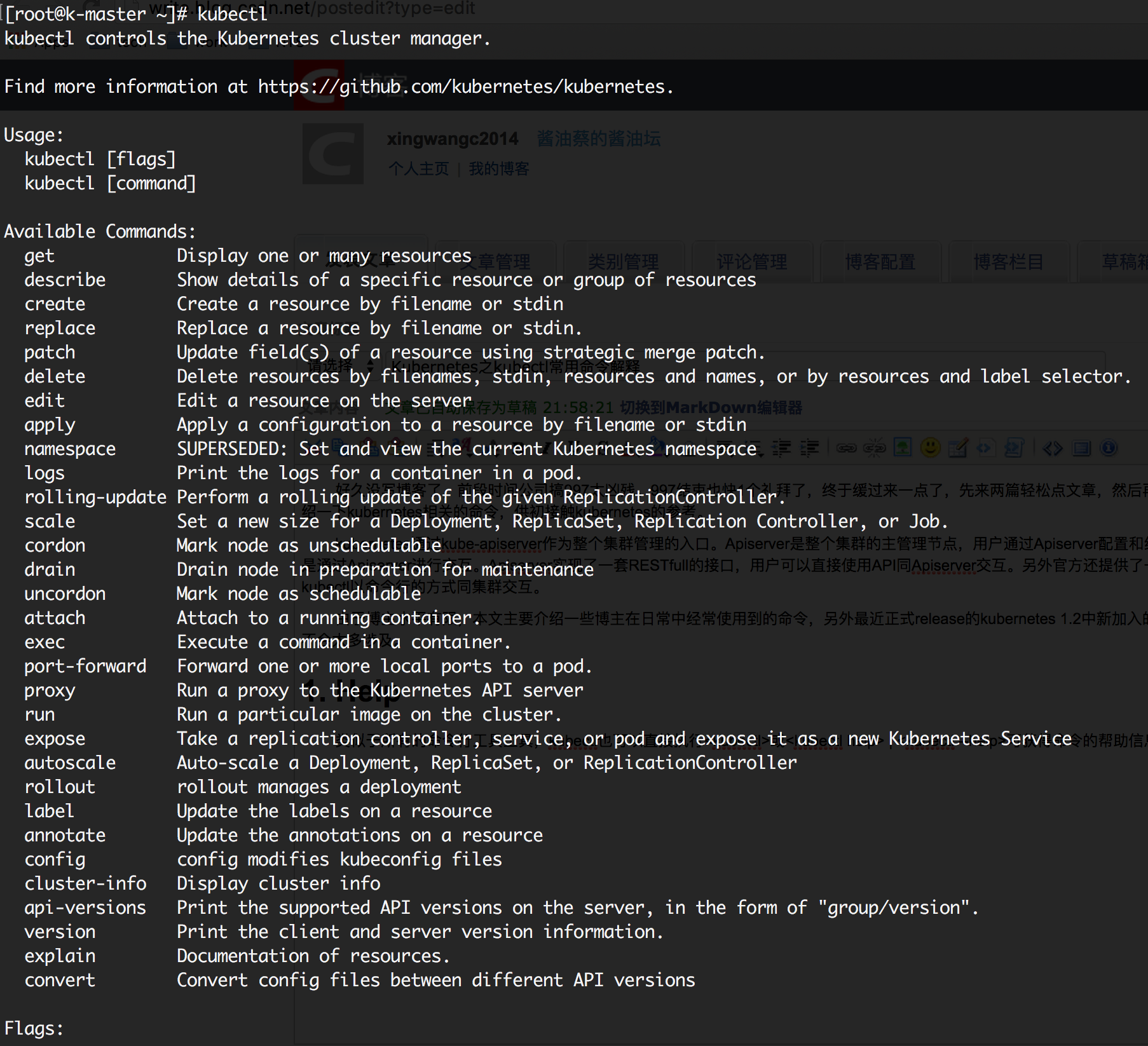
## Help

类似于所有的命令行工具工具，kubectl也可以直接执行<kubectl>或<kubectl help> | <kubectl --help>可获得命令的帮助信息。如下图所示，kubectl使用方式为：

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. Usage：
2. kubectl [flags]
3. kubectl [commond]

另外所有的命令选项都可以通过执行 --help获得特定命令的帮助信息。



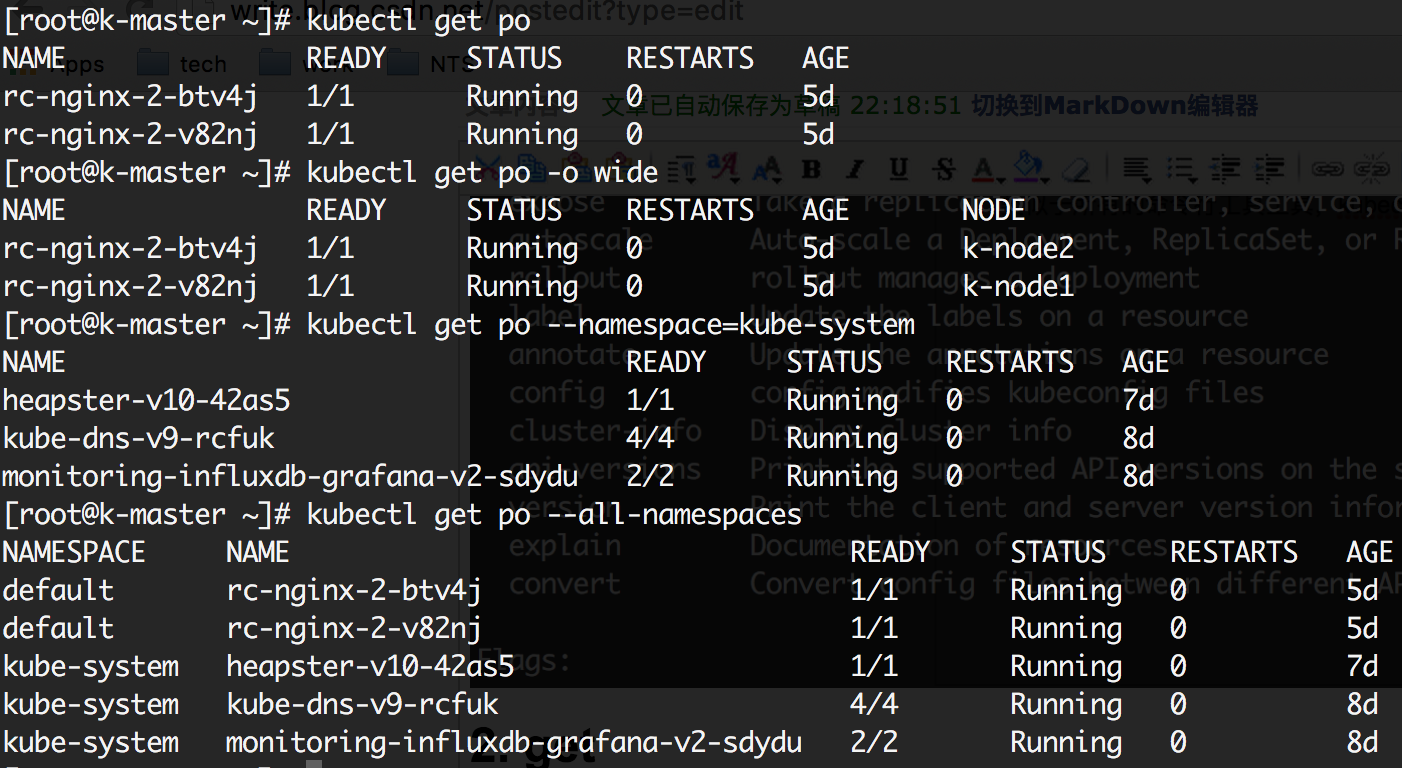
## get

get命令用于获取集群的一个或一些resource信息。使用--help查看详细信息。kubectl的帮助信息、示例相当详细，而且简单易懂。建议大家习惯使用帮助信息。kubectl可以列出集群所有resource的详细。resource包括集群节点、运行的pod，ReplicationController，service等。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

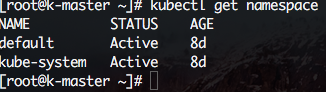
1. Usage:
2. kubectl get [(-o|--output=)json|yaml|wide|go-template=...|go-template-file=...|jsonpath=...|jsonpath-file=...] (TYPE [NAME | -l label] | TYPE/NAME ...) [flags] [flags]

1）例如获取pod信息，可以直接使用"kubectl get po“获取当前运行的所有pods的信息，或使用”kubectl get po -o wide“获取pod运行在哪个节点上的信息。注:集群中可以创建多个namespace，未显示的指定namespace的情况下，所有操作都是针对default namespace。如下图所示列出了default 和kube-system的pods：



2）获取namespace信息

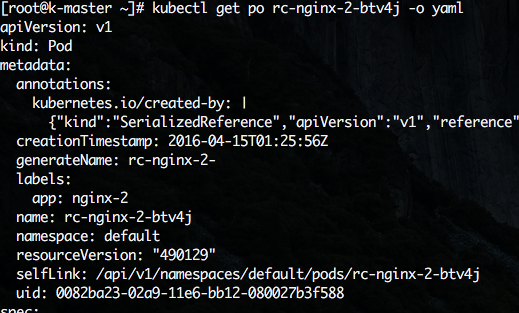
1. # kubectl get namespace



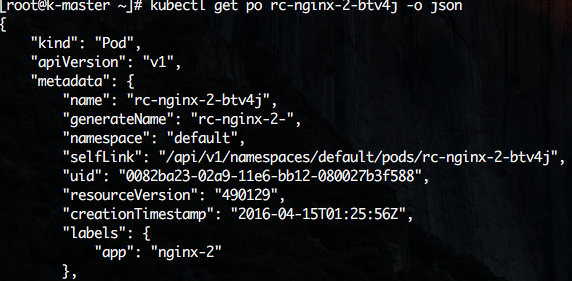
3）类似可以使用"kubectl get rc”, “kubectl get svc”, “kubectl get nodes”等获取其他resource信息。

4）获取一些更具体的信息，可以通过使用选项“-o”。如：

（1）kubectl get po <podname> -o yaml 以yawl格式输出pod的详细信息。



（2）kubectl get po <podname> -o json 以jison格式输出pod的详细信息。



（3）另外还可以使用”-o=custom-columns=“定义直接获取指定内容的值。如前面使用json和ymal格式的输出中，metadata.labels.app的值可以使用如下命令获取。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl get po rc-nginx-2-btv4j -o=custom-columns=LABELS:.metadata.labels.app

其中LABELS为显示的列标题，”.metadata.labels.app”为查询的域名

http://img.blog.csdn.net/20160425213126984

（4）其他资源也可以使用类似的方式。

## describe

describe类似于get，同样用于获取resource的相关信息。不同的是，get获得的是更详细的resource个性的详细信息，describe获得的是resource集群相关的信息。describe命令同get类似，但是describe不支持-o选项，对于同一类型resource，describe输出的信息格式，内容域相同。   
     注：如果发现是查询某个resource的信息，使用get命令能够获取更加详尽的信息。但是如果想要查询某个resource的状态，如某个pod并不是在running状态，这时需要获取更详尽的状态信息时，就应该使用describe命令。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl describe po rc-nginx-2-btv4j

## create

 kubectl命令用于根据文件或输入创建集群resource。如果已经定义了相应resource的yaml或son文件，直接kubectl create -f filename即可创建文件内定义的resource。也可以直接只用子命令[namespace/secret/configmap/serviceaccount]等直接创建相应的resource。从追踪和维护的角度出发，建议使用json或yaml的方式定义资源。   
     如，前面get中获取的两个nginx pod的replication controller文件内容如下。文件名为：rc-nginx.yaml

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. apiVersion: v1
2. kind: ReplicationController
3. metadata:
4. name: rc-nginx-2
5. spec:
6. replicas: 2
7. template:
8. metadata:
9. labels:
10. app: nginx-2
11. spec:
12. containers:
13. - name: nginx-2
14. image: xingwangc.docker.rg/nginx
15. ports:
16. - containerPort: 80

直接使用create则可以基于rc-nginx.yaml文件创建出ReplicationController（rc），rc会创建两个副本：

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl create -f rc-nginx.yaml

创建后，使用“kubectl get rc”可以看到一个名为rc-nginx-2的ReplicationController将被创建，同时“kubectl get po”的结果中会多出两个前缀为“rc-nginx-2-”的pod。关于kubernetes集群中resource，pod， ReplicationController…等后续会新开博文详细介绍。

## 5. replace

replace命令用于对已有资源进行更新、替换。如前面create中创建的nginx，当我们需要更新resource的一些属性的时候，如果修改副本数量，增加、修改label，更改image版本，修改端口等。都可以直接修改原yaml文件，然后执行replace命令。   
     注：名字不能被更更新。另外，如果是更新label，原有标签的pod将会与更新label后的rc断开联系，有新label的rc将会创建指定副本数的新的pod，但是默认并不会删除原来的pod。所以此时如果使用get po将会发现pod数翻倍，进一步check会发现原来的pod已经不会被新rc控制，此处只介绍命令不详谈此问题，好奇者可自行实验。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl replace -f rc-nginx.yaml

## patch

如果一个容器已经在运行，这时需要对一些容器属性进行修改，又不想删除容器，或不方便通过replace的方式进行更新。kubernetes还提供了一种在容器运行时，直接对容器进行修改的方式，就是patch命令。   
     如前面创建pod的label是app=nginx-2，如果在运行过程中，需要把其label改为app=nginx-3，这patch命令如下：

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl patch pod rc-nginx-2-kpiqt -p '{"metadata":{"labels":{"app":"nginx-3"}}}'

## edit

     edit提供了另一种更新resource源的操作，通过edit能够灵活的在一个common的resource基础上，发展出更过的significant resource。例如，使用edit直接更新前面创建的pod的命令为：

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl edit po rc-nginx-btv4j

     上面命令的效果等效于：

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl get po rc-nginx-btv4j -o yaml >> /tmp/nginx-tmp.yaml
2. vim /tmp/nginx-tmp.yaml
3. /\*do some changes here \*/
4. kubectl replace -f /tmp/nginx-tmp.yaml

## Delete

     根据resource名或label删除resource。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl delete -f rc-nginx.yaml
2. kubectl delete po rc-nginx-btv4j
3. kubectl delete po -lapp=nginx-2

## apply

     apply命令提供了比patch，edit等更严格的更新resource的方式。通过apply，用户可以将resource的configuration使用source control的方式维护在版本库中。每次有更新时，将配置文件push到server，然后使用kubectl apply将更新应用到resource。kubernetes会在引用更新前将当前配置文件中的配置同已经应用的配置做比较，并只更新更改的部分，而不会主动更改任何用户未指定的部分。   
     apply命令的使用方式同replace相同，不同的是，apply不会删除原有resource，然后创建新的。apply直接在原有resource的基础上进行更新。同时kubectl apply还会resource中添加一条注释，标记当前的apply。类似于git操作。 

## logs

     logs命令用于显示pod运行中，容器内程序输出到标准输出的内容。跟docker的logs命令类似。如果要获得tail -f 的方式，也可以使用-f选项。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl logs rc-nginx-2-kpiqt

## rolling-update

     rolling-update是一个非常重要的命令，对于已经部署并且正在运行的业务，rolling-update提供了不中断业务的更新方式。rolling-update每次起一个新的pod，等新pod完全起来后删除一个旧的pod，然后再起一个新的pod替换旧的pod，直到替换掉所有的pod。   
     rolling-update需要确保新的版本有不同的name，Version和label，否则会报错 。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl rolling-update rc-nginx-2 -f rc-nginx.yaml

     如果在升级过程中，发现有问题还可以中途停止update，并回滚到前面版本

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl rolling-update rc-nginx-2 —rollback

     rolling-update还有很多其他选项提供丰富的功能，如—update-period指定间隔周期，使用时可以使用-h查看help信息

## scale

     scale用于程序在负载加重或缩小时副本进行扩容或缩小，如前面创建的nginx有两个副本，可以轻松的使用scale命令对副本数进行扩展或缩小。   
扩展副本数到4：

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl scale rc rc-nginx-3 —replicas=4

重新缩减副本数到2：

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl scale rc rc-nginx-3 —replicas=2

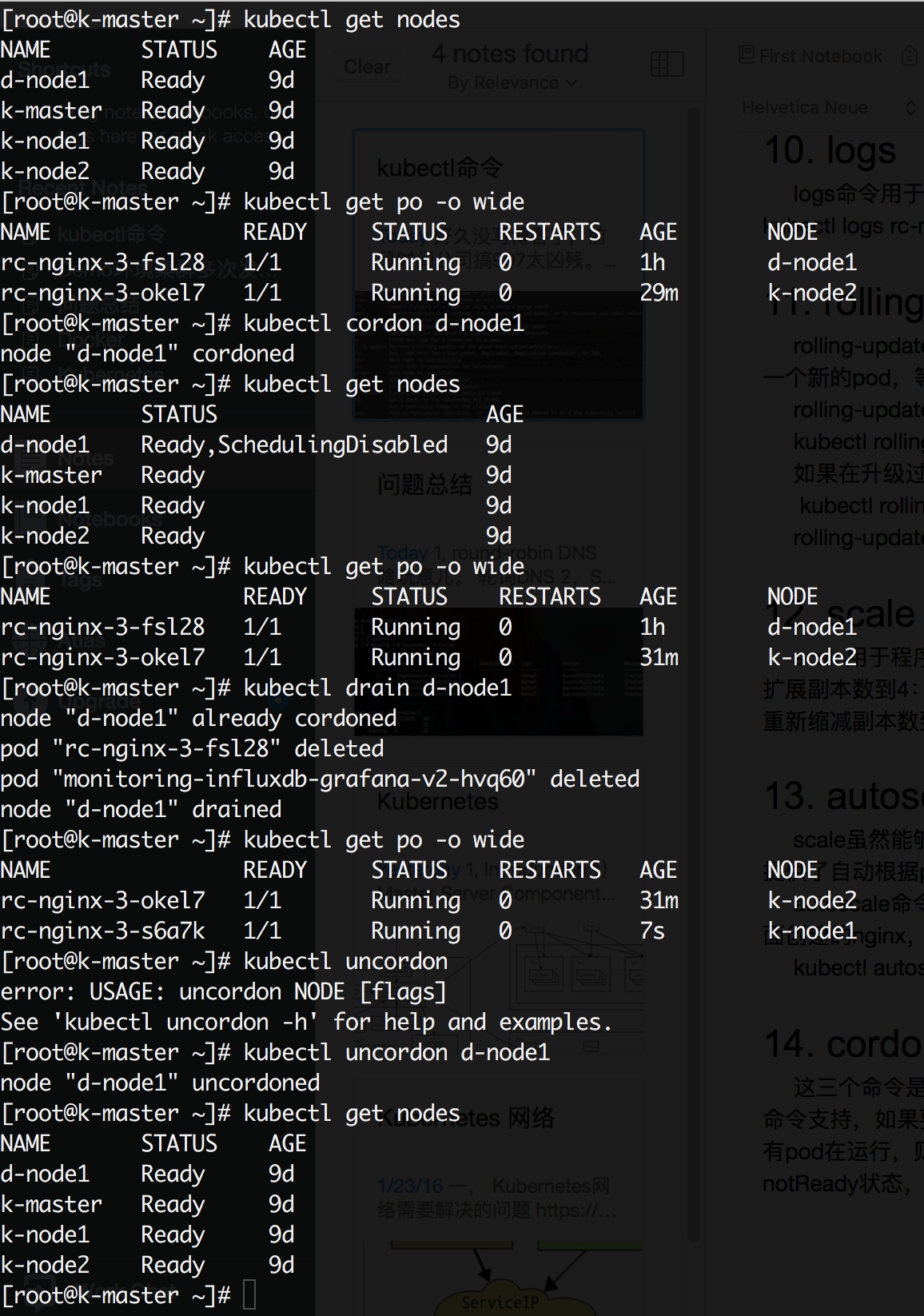
## autoscale

     scale虽然能够很方便的对副本数进行扩展或缩小，但是仍然需要人工介入，不能实时自动的根据系统负载对副本数进行扩、缩。autoscale命令提供了自动根据pod负载对其副本进行扩缩的功能。   
     autoscale命令会给一个rc指定一个副本数的范围，在实际运行中根据pod中运行的程序的负载自动在指定的范围内对pod进行扩容或缩容。如前面创建的nginx，可以用如下命令指定副本范围在1~4

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl autoscale rc rc-nginx-3 —min=1 —max=4

## cordon, drain, uncordon

     这三个命令是正式release的1.2新加入的命令，三个命令一起介绍，是因为三个命令配合使用可以实现节点的维护。在1.2之前，因为没有相应的命令支持，如果要维护一个节点，只能stop该节点上的kubelet将该节点退出集群，是集群不在将新的pod调度到该节点上。如果该节点上本生就没有pod在运行，则不会对业务有任何影响。如果该节点上有pod正在运行，kubelet停止后，master会发现该节点不可达，而将该节点标记为notReady状态，不会将新的节点调度到该节点上。同时，会在其他节点上创建新的pod替换该节点上的pod。这种方式虽然能够保证集群的健壮性，但是任然有些暴力，如果业务只有一个副本，而且该副本正好运行在被维护节点上的话，可能仍然会造成业务的短暂中断。   
     1.2中新加入的这3个命令可以保证维护节点时，平滑的将被维护节点上的业务迁移到其他节点上，保证业务不受影响。如下图所示是一个整个的节点维护的流程（为了方便demo增加了一些查看节点信息的操作）：1）首先查看当前集群所有节点状态，可以看到共四个节点都处于ready状态；2）查看当前nginx两个副本分别运行在d-node1和k-node2两个节点上；3）使用cordon命令将d-node1标记为不可调度；4）再使用kubectl get nodes查看节点状态，发现d-node1虽然还处于Ready状态，但是同时还被禁能了调度，这意味着新的pod将不会被调度到d-node1上。4）再查看nginx状态，没有任何变化，两个副本仍运行在d-node1和k-node2上；5）执行drain命令，将运行在d-node1上运行的pod平滑的赶到其他节点上；6）再查看nginx的状态发现，d-node1上的副本已经被迁移到k-node1上；这时候就可以对d-node1进行一些节点维护的操作，如升级内核，升级Docker等；7）节点维护完后，使用uncordon命令解锁d-node1，使其重新变得可调度；8）检查节点状态，发现d-node1重新变回Ready状态。   


## attach

     attach命令类似于docker的attach命令，可以直接查看容器中以daemon形式运行的进程的输出，效果类似于logs -f，退出查看使用ctrl-c。如果一个pod中有多个容器，要查看具体的某个容器的的输出，需要在pod名后使用-c containers name指定运行的容器。如下示例的命令为查看kube-system namespace中的kube-dns-v9-rcfuk pod中的skydns容器的输出。

**[plain]** [view plain](http://blog.csdn.net/xingwangc2014/article/details/51204224) [copy](http://blog.csdn.net/xingwangc2014/article/details/51204224)

1. kubectl attach kube-dns-v9-rcfuk -c skydns —namespace=kube-system

## exec

     exec命令同样类似于docker的exec命令，为在一个已经运行的容器中执行一条shell命令，如果一个pod容器中，有多个容器，需要使用-c选项指定容器。

## port-forward

     转发一个本地端口到容器端口，博主一般都是使用yaml的方式编排容器，所以基本不使用此命令。

## proxy

     博主只尝试过使用nginx作为kubernetes多master HA方式的代理，没有使用过此命令为kubernetes api server运行过proxy

## run

     类似于docker的run命令，直接运行一个image。

## label

     为kubernetes集群的resource打标签，如前面实例中提到的为rc打标签对rc分组。还可以对nodes打标签，这样在编排容器时，可以为容器指定nodeSelector将容器调度到指定lable的机器上，如如果集群中有IO密集型，计算密集型的机器分组，可以将不同的机器打上不同标签，然后将不同特征的容器调度到不同分组上。   
     在1.2之前的版本中，使用kubectl get nodes则可以列出所有节点的信息，包括节点标签，1.2版本中不再列出节点的标签信息，如果需要查看节点被打了哪些标签，需要使用describe查看节点的信息。

## 其他

其他还有如cluster-info信息可以查看当前集群的一些信息，Version查看集群版本信息等，还有一些集群配置相关的命令等。