# 一、Kubectl自动补全

## 1.BASH

在bash中设置当前shell的自动补全，要先安装 `bash-completion` 包。

```
source <(kubectl completion bash)
```

在bash shell中永久地添加自动补全：

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

## 2.ZSH

在zsh中设置当前shell的自动补全：

```
source <(kubectl completion zsh)
```

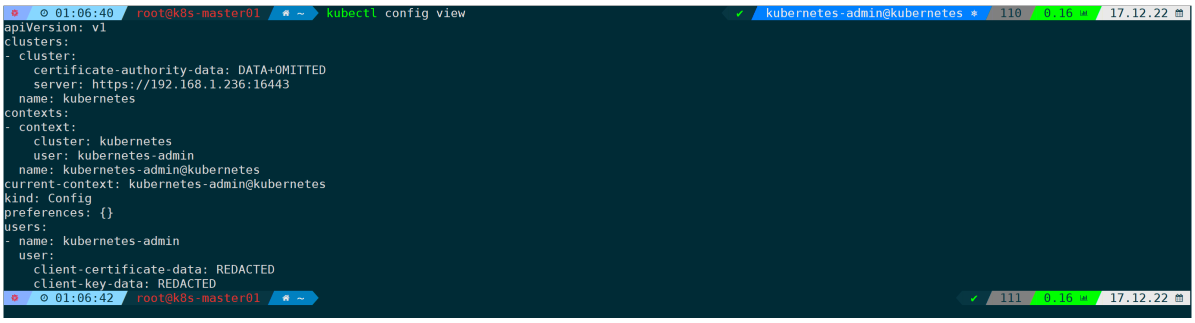
在zsh shell 中永久地添加自动补全：

```
echo '[[ $commands[kubectl] ]] && source <(kubectl completion zsh)' >> ~/.zshrc
```

# 二、Kubectl 上下文和配置

## 1.显示合并的 kubeconfig 配置(config view)

```
kubectl config view
```



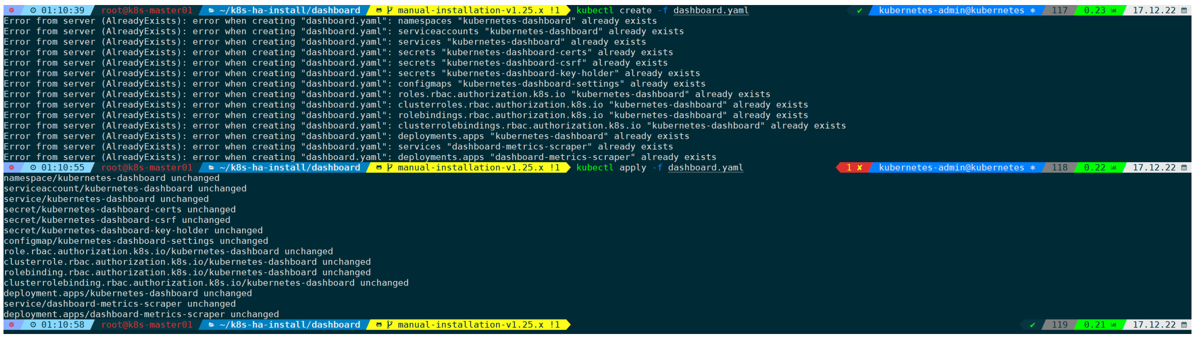
```
 01:06:40  root@k8s-master01  ~  kubectl config view      ✔  kubernetes-admin@kubernetes  110  0.16  17.12.22
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://192.168.1.236:16443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
 01:06:42  root@k8s-master01  ~                                                     ✔  111  0.16  17.12.22
```

## 2.切换集群(use-context)

```
kubectl config use-context my-cluster-name
```

## 3.创建应用(create -f)

```
kubectl create -f xxx.yaml
kubectl apply -f xxx.yaml
```



```
 ⊙ 01:10:39  root@k8s-master01  ~/k8s-ha-install/dashboard  manual-installation-v1.25.x !1  kubectl create -f dashboard.yaml      ✓  kubernetes-admin@kubernetes ⎈  117  0.23  17.12.22
Error from server (AlreadyExists): error when creating "dashboard.yaml": namespaces "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": serviceaccounts "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": services "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": secrets "kubernetes-dashboard-certs" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": secrets "kubernetes-dashboard-csrf" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": secrets "kubernetes-dashboard-key-holder" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": configmaps "kubernetes-dashboard-settings" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": roles.rbac.authorization.k8s.io "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": clusterroles.rbac.authorization.k8s.io "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": rolebindings.rbac.authorization.k8s.io "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": clusterrolebindings.rbac.authorization.k8s.io "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": deployments.apps "kubernetes-dashboard" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": services "dashboard-metrics-scraper" already exists
Error from server (AlreadyExists): error when creating "dashboard.yaml": deployments.apps "dashboard-metrics-scraper" already exists
 ⊙ 01:10:55  root@k8s-master01  ~/k8s-ha-install/dashboard  manual-installation-v1.25.x !1  kubectl apply -f dashboard.yaml      1 ✘  kubernetes-admin@kubernetes ⎈  118  0.22  17.12.22
namespace/kubernetes-dashboard unchanged
serviceaccount/kubernetes-dashboard unchanged
service/kubernetes-dashboard unchanged
secret/kubernetes-dashboard-certs unchanged
secret/kubernetes-dashboard-csrf unchanged
secret/kubernetes-dashboard-key-holder unchanged
configmap/kubernetes-dashboard-settings unchanged
role.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
deployment.apps/kubernetes-dashboard unchanged
service/dashboard-metrics-scraper unchanged
deployment.apps/dashboard-metrics-scraper unchanged
 ⊙ 01:10:58  root@k8s-master01  ~/k8s-ha-install/dashboard  manual-installation-v1.25.x !1      ✓  119  0.21  17.12.22
```

区别：

- create创建应用，如果已经存在了，则会提示已存在，无法创建。
- apply创建应用，不管是否已存在，都会覆盖创建。

> `apply` 通过定义 Kubernetes 资源的文件来管理应用。 它通过运行 `kubectl apply` 在集群中创建和更新资源。 这是在生产中管理 Kubernetes 应用的推荐方法。 参见 **Kubectl 文档**。

创建多个应用：

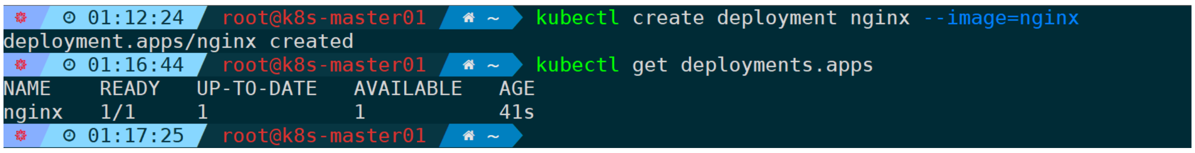
```
kubectl create -f A.yaml -f B.yaml
kubectl create -f A.yaml,B.yaml
```

两种方式都可以，第二种不支持tab补全。

## 4.创建资源(create deployment)

```
kubectl create deployment nginx --image=nginx
```

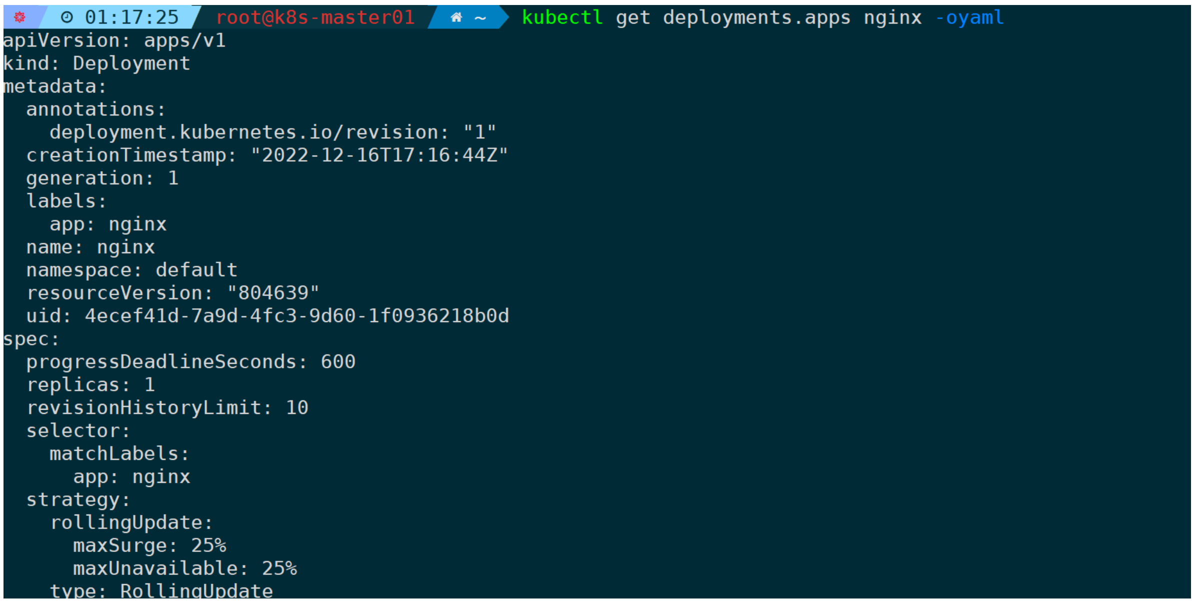
`nginx` 为资源名称，指定镜像 `--image` ，命令后面还可以接 `-n` 指定 `namespace` ，不指定则默认为 `default namespace` 。



```
 ⊙ 01:12:24  root@k8s-master01  ~  kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
 ⊙ 01:16:44  root@k8s-master01  ~  kubectl get deployments.apps
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           41s
 ⊙ 01:17:25  root@k8s-master01  ~
```

以 `yaml` 格式输出配置信息：

```
kubectl get deployments.apps nginx -oyaml
```

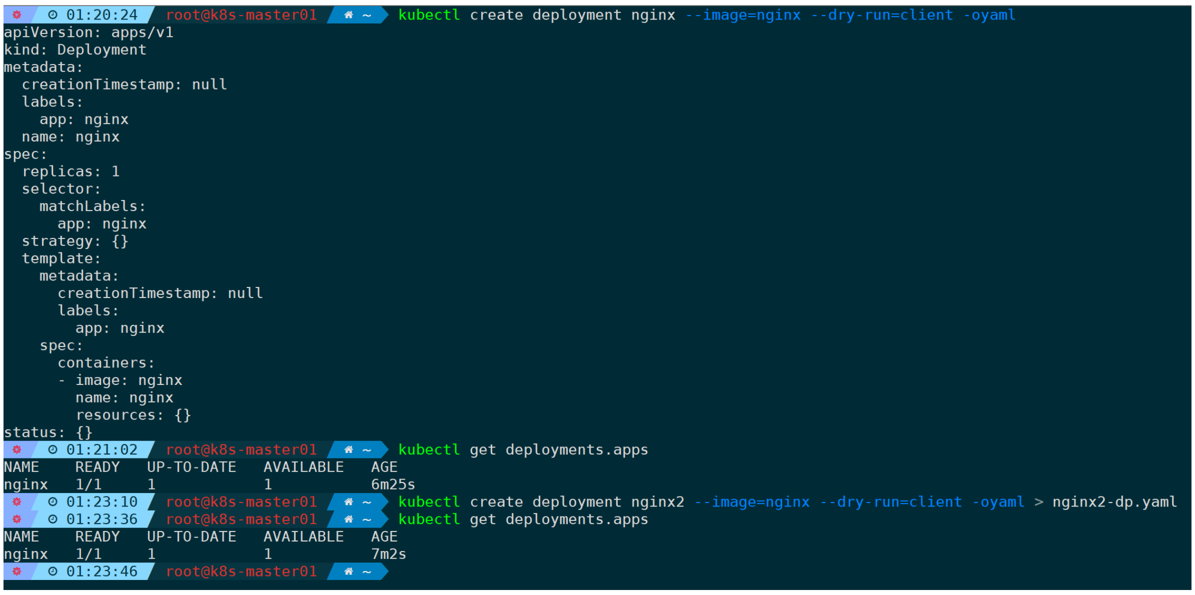


```
⊙ 01:17:25  root@k8s-master01  ~  kubectl get deployments.apps nginx -oyaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2022-12-16T17:16:44Z"
  generation: 1
  labels:
    app: nginx
  name: nginx
  namespace: default
  resourceVersion: "804639"
  uid: 4ecef41d-7a9d-4fc3-9d60-1f0936218b0d
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
```

## 5.--dry-run=client

不创建资源，通过 `--dry-run` 只显示 `yaml` 配置：

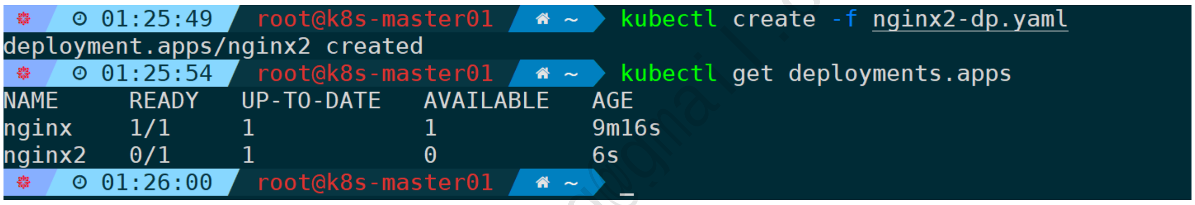
```
kubectl create deployment nginx --image=nginx --dry-run=client -oyaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```



```
 ⊙ 01:20:24  root@k8s-master01  ~  kubectl create deployment nginx --image=nginx --dry-run=client -oyaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
 ⊙ 01:21:02  root@k8s-master01  ~  kubectl get deployments.apps
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1     1            1           6m25s
 ⊙ 01:23:10  root@k8s-master01  ~  kubectl create deployment nginx2 --image=nginx --dry-run=client -oyaml > nginx2-dp.yaml
 ⊙ 01:23:36  root@k8s-master01  ~  kubectl get deployments.apps
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1     1            1           7m2s
 ⊙ 01:23:46  root@k8s-master01  ~
```

通过 `>` 重定向写入到yaml文件，之后可通过yaml文件去创建：

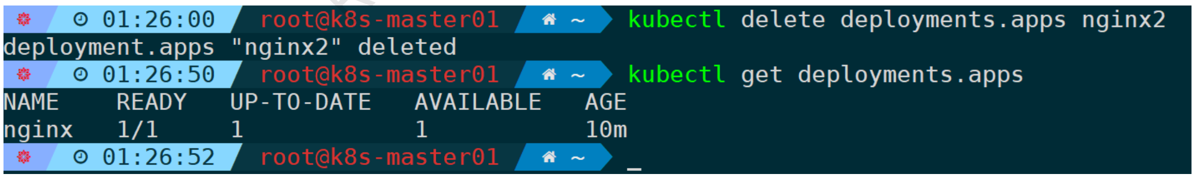
```
kubectl create deployment nginx2 --image=nginx --dry-run=client -oyaml > nginx2-
dp.yaml
kubectl create -f nginx2-dp.yaml
kubectl get deployments.apps
```



```
 ⊙ 01:25:49  root@k8s-master01  ~  kubectl create -f nginx2-dp.yaml
deployment.apps/nginx2 created
 ⊙ 01:25:54  root@k8s-master01  ~  kubectl get deployments.apps
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1     1            1           9m16s
nginx2   0/1     1            0           6s
 ⊙ 01:26:00  root@k8s-master01  ~  _
```

## 6.删除(delete)

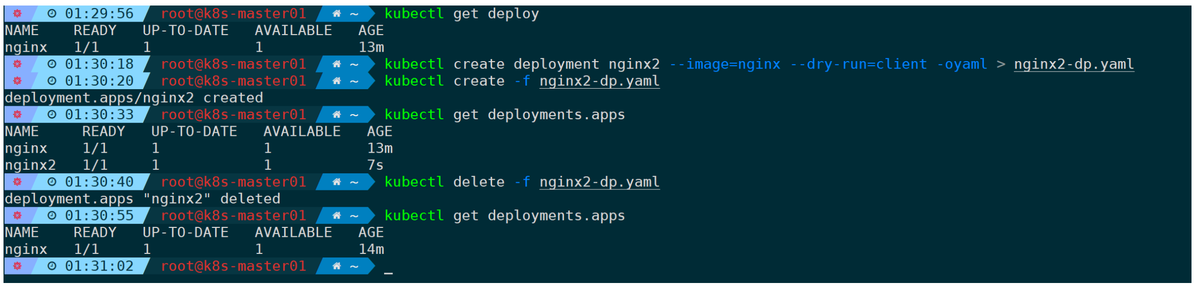
```
kubectl delete deployments.apps app
```



```
 ⊙ 01:26:00  root@k8s-master01  ~  kubectl delete deployments.apps nginx2
deployment.apps "nginx2" deleted
 ⊙ 01:26:50  root@k8s-master01  ~  kubectl get deployments.apps
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           10m
 ⊙ 01:26:52  root@k8s-master01  ~  _
```

同时也可以通过yaml文件删除：

```
kubectl delete -f app.yaml
```

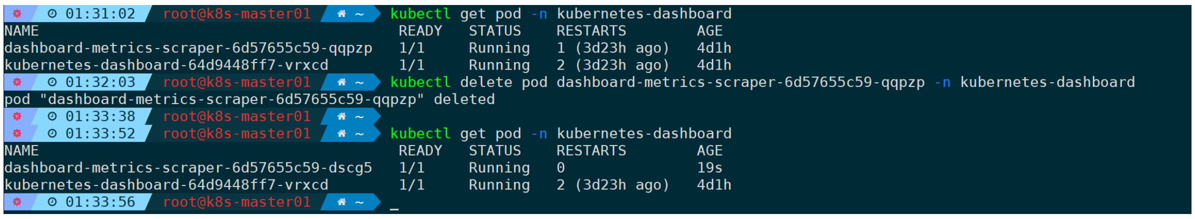


```
 ⊙ 01:29:56  root@k8s-master01  ~  kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           13m
 ⊙ 01:30:18  root@k8s-master01  ~  kubectl create deployment nginx2 --image=nginx --dry-run=client -oyaml > nginx2-dp.yaml
 ⊙ 01:30:20  root@k8s-master01  ~  kubectl create -f nginx2-dp.yaml
deployment.apps/nginx2 created
 ⊙ 01:30:33  root@k8s-master01  ~  kubectl get deployments.apps
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1     1            1           13m
nginx2   1/1     1            1           7s
 ⊙ 01:30:40  root@k8s-master01  ~  kubectl delete -f nginx2-dp.yaml
deployment.apps "nginx2" deleted
 ⊙ 01:30:55  root@k8s-master01  ~  kubectl get deployments.apps
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           14m
 ⊙ 01:31:02  root@k8s-master01  ~  _
```

删除dashboard的pod：

```
kubectl delete pod dashboard-metrics-scraper-6d57655c59-qqpzp -n kubernetes-dashboard
```

删除后pod会被自动重建起来：



```
 ⊙ 01:31:02  root@k8s-master01  ~  kubectl get pod -n kubernetes-dashboard
NAME                                         READY   STATUS    RESTARTS        AGE
dashboard-metrics-scraper-6d57655c59-qqpzp   1/1     Running   1 (3d23h ago)   4d1h
kubernetes-dashboard-64d9448ff7-vrxcd        1/1     Running   2 (3d23h ago)   4d1h
 ⊙ 01:32:03  root@k8s-master01  ~  kubectl delete pod dashboard-metrics-scraper-6d57655c59-qqpzp -n kubernetes-dashboard
pod "dashboard-metrics-scraper-6d57655c59-qqpzp" deleted
 ⊙ 01:33:38  root@k8s-master01  ~
 ⊙ 01:33:52  root@k8s-master01  ~  kubectl get pod -n kubernetes-dashboard
NAME                                         READY   STATUS    RESTARTS        AGE
dashboard-metrics-scraper-6d57655c59-dscg5   1/1     Running   0               19s
kubernetes-dashboard-64d9448ff7-vrxcd        1/1     Running   2 (3d23h ago)   4d1h
 ⊙ 01:33:56  root@k8s-master01  ~  _
```

因为pod是被 `deployment` 管理的，当只有删掉 `deployment` ，pod才能被彻底删除。

如果使用 `delete -f xx.yaml` 删除时， `yaml` 文件里面没有指定namespace，则需要通过-n参数手动指定，如：

```
kubectl delete -f xxx.yaml -n kube-system
```

## 三、查看和查找资源

### 1.查看资源(get)

查看当前命名空间下的所有services：

```
kubectl get services  #services可以缩写成svc
```

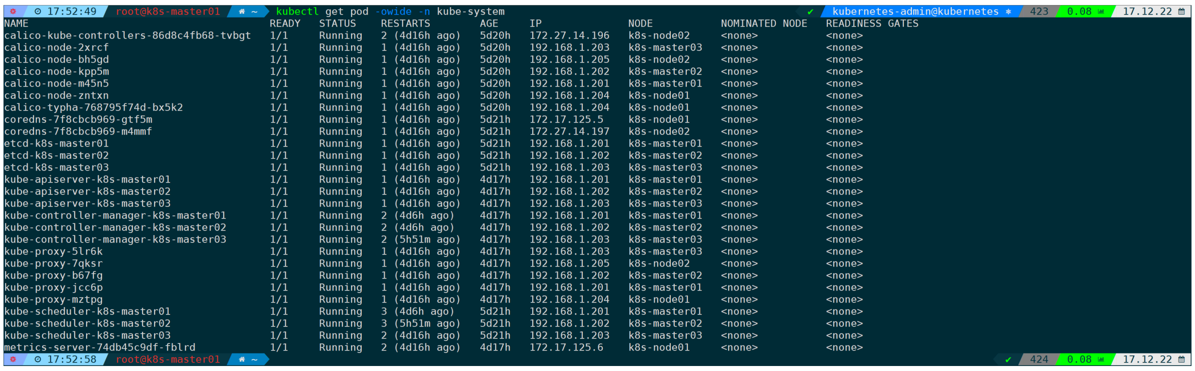
查看所有命名空间的全部Pods：

```
kubectl get pods --all-namespaces  #--all-namespaces可以缩写成-A
```

### 2.用扩展格式列举所有资源(-owide)

如，以扩展形式查看 `kub-system` 命名空间的pod信息：

```
kubectl get pod -A -owide -n kub-system
```

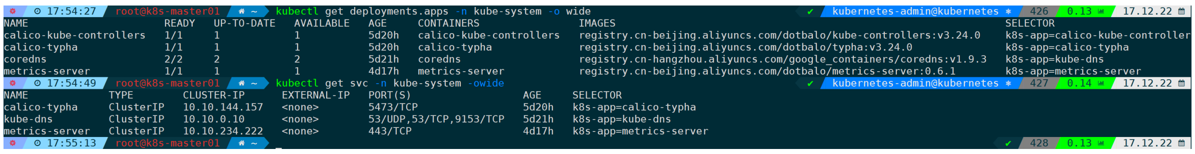


```
 ⊙ 17:52:49  root@k8s-master01  ~  kubectl get pod -owide -n kube-system                    ✔  kubernetes-admin@kubernetes ⎈  423  0.08  17.12.22
NAME                                       READY   STATUS    RESTARTS        AGE     IP               NODE          NOMINATED NODE   READINESS GATES
calico-kube-controllers-86d8c4fb68-tvbgt   1/1     Running   2 (4d16h ago)   5d20h   172.27.14.196    k8s-node02    <none>           <none>
calico-node-2xrcf                          1/1     Running   1 (4d16h ago)   5d20h   192.168.1.203    k8s-master03  <none>           <none>
calico-node-bh5gd                          1/1     Running   1 (4d16h ago)   5d20h   192.168.1.205    k8s-node02    <none>           <none>
calico-node-kpp5m                          1/1     Running   1 (4d16h ago)   5d20h   192.168.1.202    k8s-master02  <none>           <none>
calico-node-m45n5                          1/1     Running   1 (4d16h ago)   5d20h   192.168.1.201    k8s-master01  <none>           <none>
calico-node-zntxn                          1/1     Running   1 (4d16h ago)   5d20h   192.168.1.204    k8s-node01    <none>           <none>
calico-typha-768795f74d-bx5k2              1/1     Running   1 (4d16h ago)   5d20h   192.168.1.204    k8s-node01    <none>           <none>
coredns-7f8cbcb969-gtf5m                   1/1     Running   1 (4d16h ago)   5d21h   172.17.125.5     k8s-node01    <none>           <none>
coredns-7f8cbcb969-m4mmf                   1/1     Running   1 (4d16h ago)   5d21h   172.27.14.197    k8s-node02    <none>           <none>
etcd-k8s-master01                          1/1     Running   1 (4d16h ago)   5d21h   192.168.1.201    k8s-master01  <none>           <none>
etcd-k8s-master02                          1/1     Running   1 (4d16h ago)   5d21h   192.168.1.202    k8s-master02  <none>           <none>
etcd-k8s-master03                          1/1     Running   1 (4d16h ago)   5d21h   192.168.1.203    k8s-master03  <none>           <none>
kube-apiserver-k8s-master01                1/1     Running   1 (4d16h ago)   4d17h   192.168.1.201    k8s-master01  <none>           <none>
kube-apiserver-k8s-master02                1/1     Running   1 (4d16h ago)   4d17h   192.168.1.202    k8s-master02  <none>           <none>
kube-apiserver-k8s-master03                1/1     Running   1 (4d16h ago)   4d17h   192.168.1.203    k8s-master03  <none>           <none>
kube-controller-manager-k8s-master01       1/1     Running   2 (4d6h ago)    4d17h   192.168.1.201    k8s-master01  <none>           <none>
kube-controller-manager-k8s-master02       1/1     Running   2 (4d6h ago)    4d17h   192.168.1.202    k8s-master02  <none>           <none>
kube-controller-manager-k8s-master03       1/1     Running   2 (5h51m ago)   4d17h   192.168.1.203    k8s-master03  <none>           <none>
kube-proxy-5lr6k                           1/1     Running   1 (4d16h ago)   4d17h   192.168.1.203    k8s-master03  <none>           <none>
kube-proxy-7qksr                           1/1     Running   1 (4d16h ago)   4d17h   192.168.1.205    k8s-node02    <none>           <none>
kube-proxy-b67fg                           1/1     Running   1 (4d16h ago)   4d17h   192.168.1.202    k8s-master02  <none>           <none>
kube-proxy-jcc6p                           1/1     Running   1 (4d16h ago)   4d17h   192.168.1.201    k8s-master01  <none>           <none>
kube-proxy-mztpg                           1/1     Running   1 (4d16h ago)   4d17h   192.168.1.204    k8s-node01    <none>           <none>
kube-scheduler-k8s-master01                1/1     Running   3 (4d6h ago)    5d21h   192.168.1.201    k8s-master01  <none>           <none>
kube-scheduler-k8s-master02                1/1     Running   3 (5h51m ago)   5d21h   192.168.1.202    k8s-master02  <none>           <none>
kube-scheduler-k8s-master03                1/1     Running   2 (4d16h ago)   5d21h   192.168.1.203    k8s-master03  <none>           <none>
metrics-server-74db45c9df-fblrd            1/1     Running   2 (4d16h ago)   4d17h   172.17.125.6     k8s-node01    <none>           <none>
 ⊙ 17:52:58  root@k8s-master01  ~                                                                                           ✔  424  0.08  17.12.22
```

将会显示更多列信息，其中也包括IP地址(如果资源有IP地址的概念)。

扩展格式显示 `deployment` 、 `service` 的资源信息：

```
kubectl get deployments.apps -n kube-system -o wide
kubectl get svc -n kube-system -owide
```



```
 17:54:27  root@k8s-master01  ~  kubectl get deployments.apps -n kube-system -o wide      ✔  kubernetes-admin@kubernetes  426  0.13  17.12.22
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS                IMAGES                                                                SELECTOR
calico-kube-controllers   1/1     1            1           5d20h   calico-kube-controllers   registry.cn-beijing.aliyuncs.com/dotbalo/kube-controllers:v3.24.0    k8s-app=calico-kube-controller
calico-typha              1/1     1            1           5d20h   calico-typha              registry.cn-beijing.aliyuncs.com/dotbalo/typha:v3.24.0               k8s-app=calico-typha
coredns                   2/2     2            2           5d21h   coredns                   registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:v1.9.3   k8s-app=kube-dns
metrics-server            1/1     1            1           4d17h   metrics-server            registry.cn-beijing.aliyuncs.com/dotbalo/metrics-server:0.6.1        k8s-app=metrics-server
 17:54:49  root@k8s-master01  ~  kubectl get svc -n kube-system -owide      ✔  kubernetes-admin@kubernetes  427  0.14  17.12.22
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE     SELECTOR
calico-typha     ClusterIP   10.10.144.157   <none>        5473/TCP                 5d20h   k8s-app=calico-typha
kube-dns         ClusterIP   10.10.0.10      <none>        53/UDP,53/TCP,9153/TCP   5d21h   k8s-app=kube-dns
metrics-server   ClusterIP   10.10.234.222   <none>        443/TCP                  4d17h   k8s-app=metrics-server
 17:55:13  root@k8s-master01  ~                                                     ✔  428  0.13  17.12.22
```

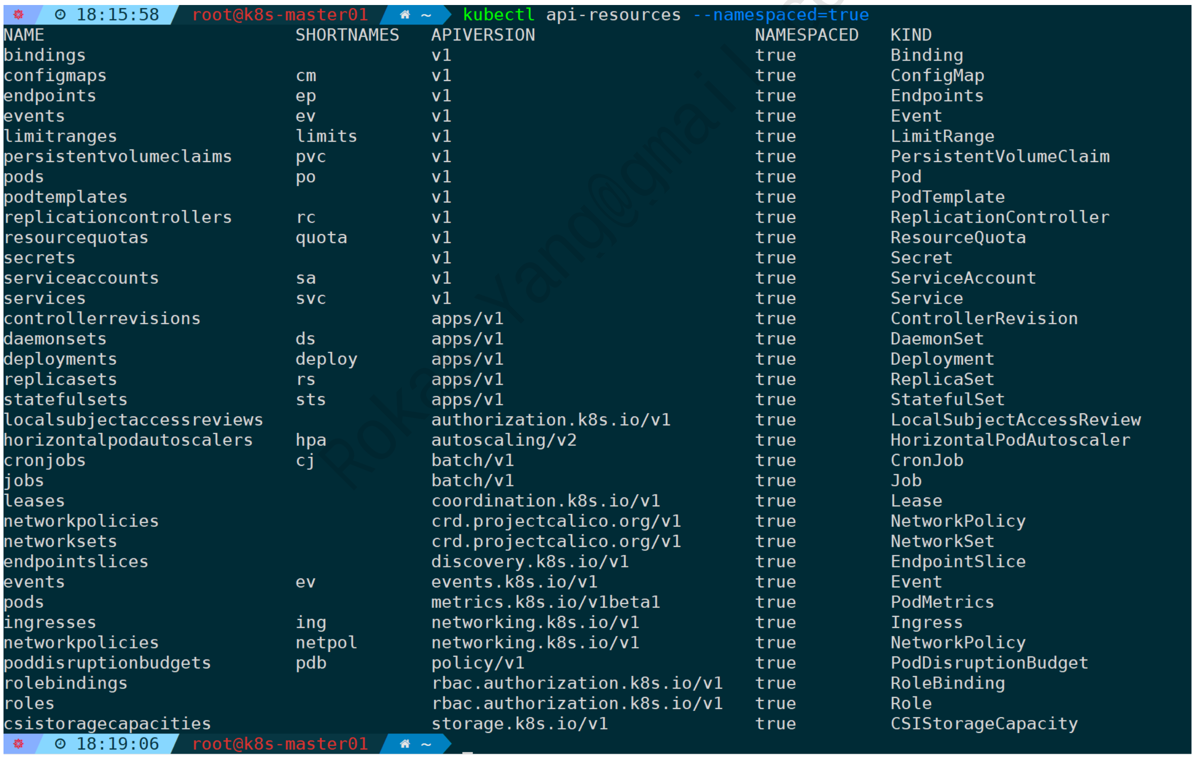
## 3.资源类型(api-resources)

列出所支持的全部资源类型和它们的简称、 **API 组**，是否是 **名字空间作用域** 和 **Kind** 。

```
kubectl api-resources
```

列出所有命名空间作用域的资源：

```
kubectl api-resources --namespaced=true
```



```
 18:15:58  root@k8s-master01  ~  kubectl api-resources --namespaced=true
NAME                        SHORTNAMES   APIVERSION                     NAMESPACED   KIND
bindings                                 v1                             true         Binding
configmaps                  cm           v1                             true         ConfigMap
endpoints                   ep           v1                             true         Endpoints
events                      ev           v1                             true         Event
limitranges                 limits       v1                             true         LimitRange
persistentvolumeclaims      pvc          v1                             true         PersistentVolumeClaim
pods                        po           v1                             true         Pod
podtemplates                             v1                             true         PodTemplate
replicationcontrollers      rc           v1                             true         ReplicationController
resourcequotas              quota        v1                             true         ResourceQuota
secrets                                  v1                             true         Secret
serviceaccounts             sa           v1                             true         ServiceAccount
services                    svc          v1                             true         Service
controllerrevisions                      apps/v1                        true         ControllerRevision
daemonsets                  ds           apps/v1                        true         DaemonSet
deployments                 deploy       apps/v1                        true         Deployment
replicasets                 rs           apps/v1                        true         ReplicaSet
statefulsets                sts          apps/v1                        true         StatefulSet
localsubjectaccessreviews                authorization.k8s.io/v1        true         LocalSubjectAccessReview
horizontalpodautoscalers    hpa          autoscaling/v2                 true         HorizontalPodAutoscaler
cronjobs                    cj           batch/v1                       true         CronJob
jobs                                     batch/v1                       true         Job
leases                                   coordination.k8s.io/v1         true         Lease
networkpolicies                          crd.projectcalico.org/v1       true         NetworkPolicy
networksets                              crd.projectcalico.org/v1       true         NetworkSet
endpointslices                           discovery.k8s.io/v1            true         EndpointSlice
events                      ev           events.k8s.io/v1               true         Event
pods                                     metrics.k8s.io/v1beta1         true         PodMetrics
ingresses                   ing          networking.k8s.io/v1           true         Ingress
networkpolicies             netpol       networking.k8s.io/v1           true         NetworkPolicy
poddisruptionbudgets        pdb          policy/v1                      true         PodDisruptionBudget
rolebindings                             rbac.authorization.k8s.io/v1   true         RoleBinding
roles                                    rbac.authorization.k8s.io/v1   true         Role
csistoragecapacities                     storage.k8s.io/v1              true         CSIStorageCapacity
 18:19:06  root@k8s-master01  ~
```

列出所有非命名空间作用域的资源，没有命名空间的则说明无法通过命名空间隔离：

```
kubectl api-resources --namespaced=false
```

用简单格式列举所有资源（仅显示资源名称： `-o name` ）：

```
kubectl api-resources -o name
```

列出支持 `list` 和 `get` 请求的所有资源：

```
kubectl api-resources --verbs=list,get
```

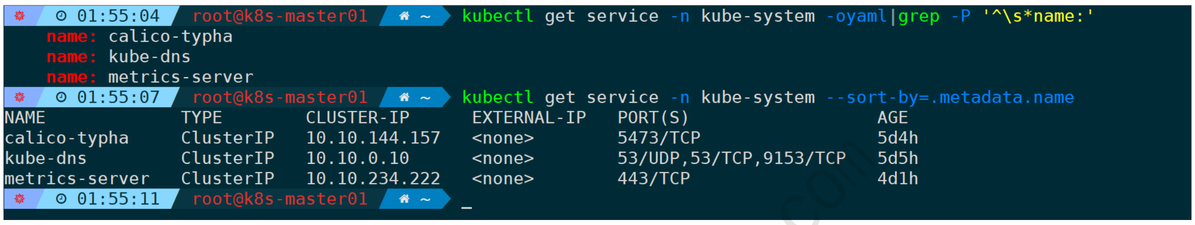
列出 `extensions` API 组中的所有资源：

```
kubectl api-resources --api-group=extensions
```

## 4.排序(--sort-by)

以service的 `metadata` 字段里的 `name` 排序，也就是从 `.yaml` 文件里面取数据：

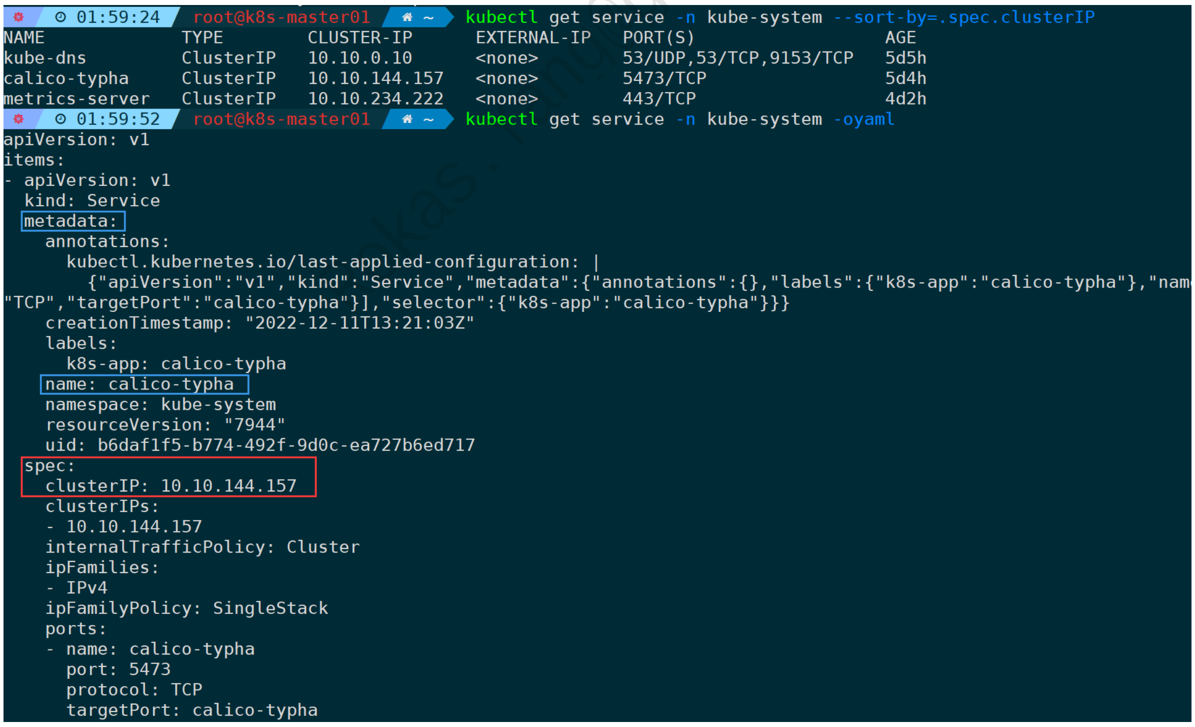
```
kubectl get service -n kube-system --sort-by=.metadata.name
```



```
 01:55:04  root@k8s-master01  ~  kubectl get service -n kube-system -oyaml|grep -P '^\s*name:'
    name: calico-typha
    name: kube-dns
    name: metrics-server
 01:55:07  root@k8s-master01  ~  kubectl get service -n kube-system --sort-by=.metadata.name
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE
calico-typha     ClusterIP   10.10.144.157   <none>        5473/TCP                 5d4h
kube-dns         ClusterIP   10.10.0.10      <none>        53/UDP,53/TCP,9153/TCP   5d5h
metrics-server   ClusterIP   10.10.234.222   <none>        443/TCP                  4d1h
 01:55:11  root@k8s-master01  ~  _
```

同理，从 `spec` 字段的 `clusterIP` 排序：

```
kubectl get service -n kube-system --sort-by=.spec.clusterIP
```

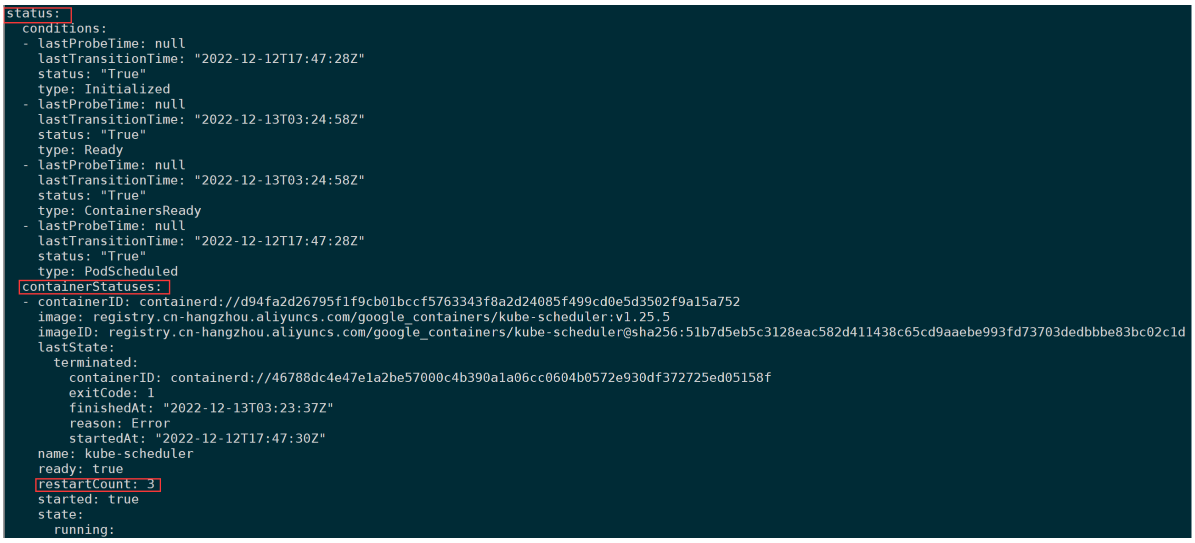


```
 01:59:24  root@k8s-master01  ~  kubectl get service -n kube-system --sort-by=.spec.clusterIP
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE
kube-dns         ClusterIP   10.10.0.10      <none>        53/UDP,53/TCP,9153/TCP   5d5h
calico-typha     ClusterIP   10.10.144.157   <none>        5473/TCP                 5d4h
metrics-server   ClusterIP   10.10.234.222   <none>        443/TCP                  4d2h
 01:59:52  root@k8s-master01  ~  kubectl get service -n kube-system -oyaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |
        {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"labels":{"k8s-app":"calico-typha"},"nam
"TCP","targetPort":"calico-typha"}],"selector":{"k8s-app":"calico-typha"}}}
    creationTimestamp: "2022-12-11T13:21:03Z"
    labels:
      k8s-app: calico-typha
    name: calico-typha
    namespace: kube-system
    resourceVersion: "7944"
    uid: b6daf1f5-b774-492f-9d0c-ea727b6ed717
  spec:
    clusterIP: 10.10.144.157
    clusterIPs:
    - 10.10.144.157
    internalTrafficPolicy: Cluster
    ipFamilies:
    - IPv4
    ipFamilyPolicy: SingleStack
    ports:
    - name: calico-typha
      port: 5473
      protocol: TCP
      targetPort: calico-typha
```

列出 Pods，按重启次数排序

```
kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'
```

`containerStatuses[0]` 表示containerStatuses的第0个元素的值。



```
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2022-12-12T17:47:28Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2022-12-13T03:24:58Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2022-12-13T03:24:58Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2022-12-12T17:47:28Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID: containerd://d94fa2d26795f1f9cb01bccf5763343f8a2d24085f499cd0e5d3502f9a15a752
    image: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler:v1.25.5
    imageID: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-scheduler@sha256:51b7d5eb5c3128eac582d411438c65cd9aaebe993fd73703dedbbbe83bc02c1d
    lastState:
      terminated:
        containerID: containerd://46788dc4e47e1a2be57000c4b390a1a06cc0604b0572e930df372725ed05158f
        exitCode: 1
        finishedAt: "2022-12-13T03:23:37Z"
        reason: Error
        startedAt: "2022-12-12T17:47:30Z"
    name: kube-scheduler
    ready: true
    restartCount: 3
    started: true
    state:
      running:
```

## 5.过滤应用(-l k8s-app)

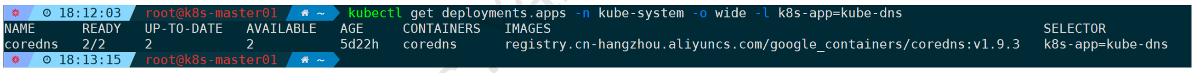
`-l` 为label的意思，通过标签过滤。

过滤 `k8s-app` 标签为 `calico-node` 的容器：

```
kubectl get pods -n kube-system -l k8s-app=calico-node
```

同理，过滤标签为 `k8s-app=kube-dns` 的 `deployment` 的资源，并且扩展输出：

```
kubectl get deployments.apps -n kube-system -o wide -l k8s-app=kube-dns
```

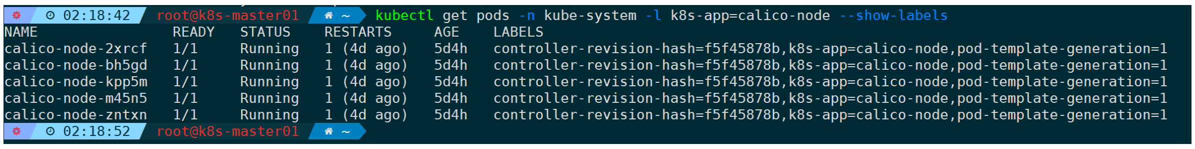


```
 18:12:03  root@k8s-master01  ~  kubectl get deployments.apps -n kube-system -o wide -l k8s-app=kube-dns
NAME      READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES                                                        SELECTOR
coredns   2/2     2            2           5d22h   coredns      registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:v1.9.3   k8s-app=kube-dns
 18:13:15  root@k8s-master01  ~
```

## 6.显示标签(--show-labels)

输出pod信息时，过滤 `calico-node` 的pod，并且显示标签信息

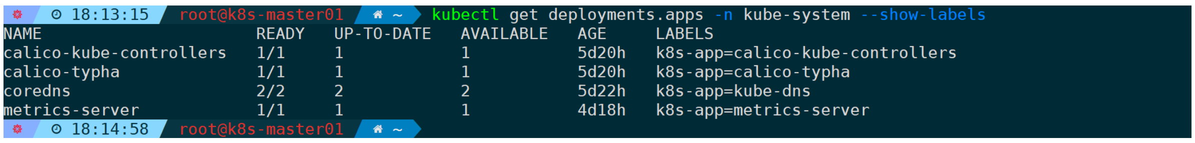
```
kubectl get pods -n kube-system -l k8s-app=calico-node --show-labels
```



```
 02:18:42  root@k8s-master01  ~  kubectl get pods -n kube-system -l k8s-app=calico-node --show-labels
NAME                READY   STATUS    RESTARTS     AGE    LABELS
calico-node-2xrcf   1/1     Running   1 (4d ago)   5d4h   controller-revision-hash=f5f45878b,k8s-app=calico-node,pod-template-generation=1
calico-node-bh5gd   1/1     Running   1 (4d ago)   5d4h   controller-revision-hash=f5f45878b,k8s-app=calico-node,pod-template-generation=1
calico-node-kpp5m   1/1     Running   1 (4d ago)   5d4h   controller-revision-hash=f5f45878b,k8s-app=calico-node,pod-template-generation=1
calico-node-m45n5   1/1     Running   1 (4d ago)   5d4h   controller-revision-hash=f5f45878b,k8s-app=calico-node,pod-template-generation=1
calico-node-zntxn   1/1     Running   1 (4d ago)   5d4h   controller-revision-hash=f5f45878b,k8s-app=calico-node,pod-template-generation=1
 02:18:52  root@k8s-master01  ~
```

显示 `deployment` 的标签：

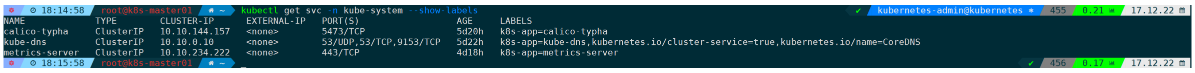
```
kubectl get deployments.apps -n kube-system --show-labels
```



```
 18:13:15  root@k8s-master01  ~  kubectl get deployments.apps -n kube-system --show-labels
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE     LABELS
calico-kube-controllers   1/1     1            1           5d20h   k8s-app=calico-kube-controllers
calico-typha              1/1     1            1           5d20h   k8s-app=calico-typha
coredns                   2/2     2            2           5d22h   k8s-app=kube-dns
metrics-server            1/1     1            1           4d18h   k8s-app=metrics-server
 18:14:58  root@k8s-master01  ~
```

同理，显示 `service` 的标签：

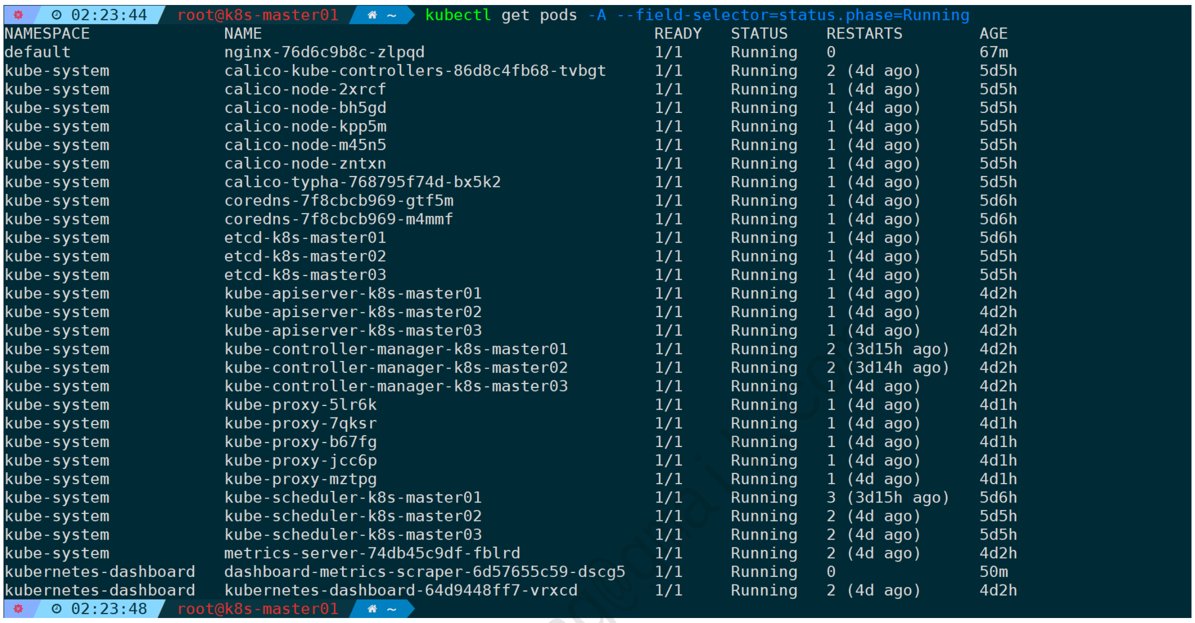
```
kubectl get svc -n kube-system --show-labels
```



```
 ⊙ 18:14:58  root@k8s-master01  ~  kubectl get svc -n kube-system --show-labels          ✔  kubernetes-admin@kubernetes ⎈  455  0.21  17.12.22
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE     LABELS
calico-typha     ClusterIP   10.10.144.157   <none>        5473/TCP                 5d20h   k8s-app=calico-typha
kube-dns         ClusterIP   10.10.0.10      <none>        53/UDP,53/TCP,9153/TCP   5d22h   k8s-app=kube-dns,kubernetes.io/cluster-service=true,kubernetes.io/name=CoreDNS
metrics-server   ClusterIP   10.10.234.222   <none>        443/TCP                  4d18h   k8s-app=metrics-server
 ⊙ 18:15:58  root@k8s-master01  ~                                                          ✔  456  0.17  17.12.22
```

## 7.查看运行中的Pods(--field-selector=status.phase=Running)

```
kubectl get pods -A --field-selector=status.phase=Running
```



```
 ⊙ 02:23:44  root@k8s-master01  ~  kubectl get pods -A --field-selector=status.phase=Running
NAMESPACE              NAME                                         READY   STATUS    RESTARTS       AGE
default                nginx-76d6c9b8c-zlpqd                        1/1     Running   0              67m
kube-system            calico-kube-controllers-86d8c4fb68-tvbgt     1/1     Running   2 (4d ago)     5d5h
kube-system            calico-node-2xrcf                            1/1     Running   1 (4d ago)     5d5h
kube-system            calico-node-bh5gd                            1/1     Running   1 (4d ago)     5d5h
kube-system            calico-node-kpp5m                            1/1     Running   1 (4d ago)     5d5h
kube-system            calico-node-m45n5                            1/1     Running   1 (4d ago)     5d5h
kube-system            calico-node-zntxn                            1/1     Running   1 (4d ago)     5d5h
kube-system            calico-typha-768795f74d-bx5k2                1/1     Running   1 (4d ago)     5d5h
kube-system            coredns-7f8cbcb969-gtf5m                     1/1     Running   1 (4d ago)     5d6h
kube-system            coredns-7f8cbcb969-m4mmf                     1/1     Running   1 (4d ago)     5d6h
kube-system            etcd-k8s-master01                            1/1     Running   1 (4d ago)     5d6h
kube-system            etcd-k8s-master02                            1/1     Running   1 (4d ago)     5d5h
kube-system            etcd-k8s-master03                            1/1     Running   1 (4d ago)     5d5h
kube-system            kube-apiserver-k8s-master01                  1/1     Running   1 (4d ago)     4d2h
kube-system            kube-apiserver-k8s-master02                  1/1     Running   1 (4d ago)     4d2h
kube-system            kube-apiserver-k8s-master03                  1/1     Running   1 (4d ago)     4d2h
kube-system            kube-controller-manager-k8s-master01         1/1     Running   2 (3d15h ago)  4d2h
kube-system            kube-controller-manager-k8s-master02         1/1     Running   2 (3d14h ago)  4d2h
kube-system            kube-controller-manager-k8s-master03         1/1     Running   1 (4d ago)     4d2h
kube-system            kube-proxy-5lr6k                             1/1     Running   1 (4d ago)     4d1h
kube-system            kube-proxy-7qksr                             1/1     Running   1 (4d ago)     4d1h
kube-system            kube-proxy-b67fg                             1/1     Running   1 (4d ago)     4d1h
kube-system            kube-proxy-jcc6p                             1/1     Running   1 (4d ago)     4d1h
kube-system            kube-proxy-mztpg                             1/1     Running   1 (4d ago)     4d1h
kube-system            kube-scheduler-k8s-master01                  1/1     Running   3 (3d15h ago)  5d6h
kube-system            kube-scheduler-k8s-master02                  1/1     Running   2 (4d ago)     5d5h
kube-system            kube-scheduler-k8s-master03                  1/1     Running   2 (4d ago)     5d5h
kube-system            metrics-server-74db45c9df-fblrd              1/1     Running   2 (4d ago)     4d2h
kubernetes-dashboard   dashboard-metrics-scraper-6d57655c59-dscg5   1/1     Running   0              50m
kubernetes-dashboard   kubernetes-dashboard-64d9448ff7-vrxcd        1/1     Running   2 (4d ago)     4d2h
 ⊙ 02:23:48  root@k8s-master01  ~
```

# 四、更新资源

## 1.set

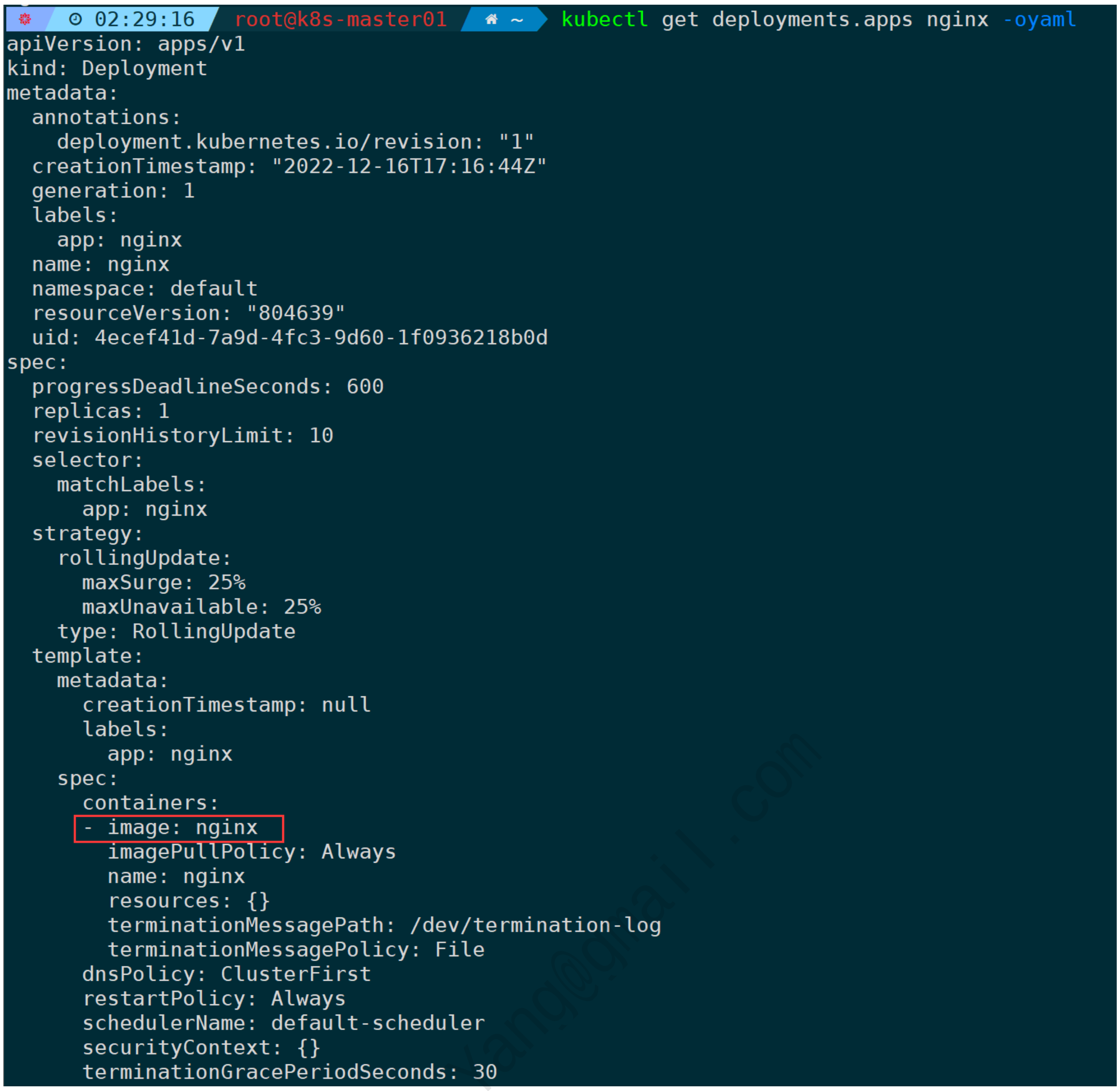
滚动更新 `frontend` 的 `www` 容器镜像：

```
kubectl set image deployment/frontend www=image:v2
```

比如更新 `deployment` 里的nginx镜像，更新到v2版本：

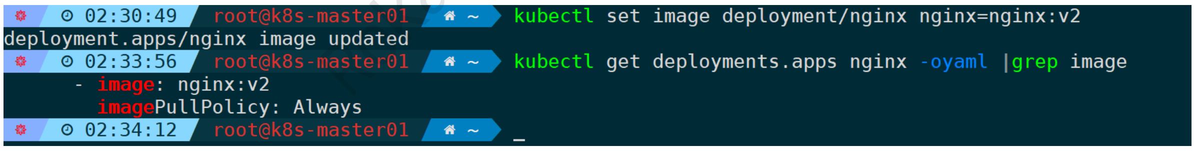
```
kubectl set image deployment/nginx nginx=nginx=v2
```

更新前：



```
⊙ 02:29:16  root@k8s-master01  ~  kubectl get deployments.apps nginx -oyaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2022-12-16T17:16:44Z"
  generation: 1
  labels:
    app: nginx
  name: nginx
  namespace: default
  resourceVersion: "804639"
  uid: 4ecef41d-7a9d-4fc3-9d60-1f0936218b0d
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

更新后：



```
⊙ 02:30:49  root@k8s-master01  ~  kubectl set image deployment/nginx nginx=nginx:v2
deployment.apps/nginx image updated
⊙ 02:33:56  root@k8s-master01  ~  kubectl get deployments.apps nginx -oyaml |grep image
      - image: nginx:v2
        imagePullPolicy: Always
⊙ 02:34:12  root@k8s-master01  ~  _
```

## 2.apply

创建 `nginx3.yaml` 配置文件：

```
kubectl create deployment nginx3 --image=nginx --dry-run=client -oyaml  > nginx3.yaml
```

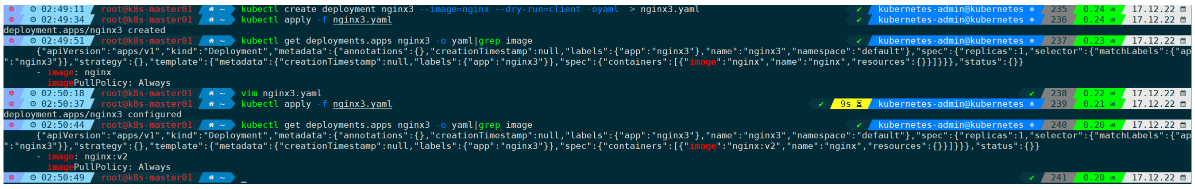
通过此配置文件创建nginx3的deployment：

```
kubectl apply -f nginx3.yaml
```

之后修改这个yaml文件，将 `nginx` 改成 `nginx:v2` ，再通过 `apply` 来更新配置：

```
kubectl apply -f nginx3.yaml
```

此时可以看到，镜像更新成功：



```
 02:49:11  root@k8s-master01  ~  kubectl create deployment nginx3 --image=nginx --dry-run=client -oyaml  > nginx3.yaml        ✓  kubernetes-admin@kubernetes  235  0.24  17.12.22
 02:49:34  root@k8s-master01  ~  kubectl apply -f nginx3.yaml                                                                  ✓  kubernetes-admin@kubernetes  236  0.24  17.12.22
deployment.apps/nginx3 created
 02:49:51  root@k8s-master01  ~  kubectl get deployments.apps nginx3 -o yaml|grep image                                       ✓  kubernetes-admin@kubernetes  237  0.23  17.12.22
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"creationTimestamp":null,"labels":{"app":"nginx3"},"name":"nginx3","namespace":"default"},"spec":{"replicas":1,"selector":{"matchLabels":{"ap
":"nginx3"}},"strategy":{},"template":{"metadata":{"creationTimestamp":null,"labels":{"app":"nginx3"}},"spec":{"containers":[{"image":"nginx","name":"nginx","resources":{}}]}}},"status":{}}
      - image: nginx
        imagePullPolicy: Always
 02:50:18  root@k8s-master01  ~  vim nginx3.yaml                                                                               ✓  238  0.22  17.12.22
 02:50:37  root@k8s-master01  ~  kubectl apply -f nginx3.yaml                                                         ✓  9s  kubernetes-admin@kubernetes  239  0.21  17.12.22
deployment.apps/nginx3 configured
 02:50:44  root@k8s-master01  ~  kubectl get deployments.apps nginx3 -o yaml|grep image                                       ✓  kubernetes-admin@kubernetes  240  0.20  17.12.22
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"creationTimestamp":null,"labels":{"app":"nginx3"},"name":"nginx3","namespace":"default"},"spec":{"replicas":1,"selector":{"matchLabels":{"ap
":"nginx3"}},"strategy":{},"template":{"metadata":{"creationTimestamp":null,"labels":{"app":"nginx3"}},"spec":{"containers":[{"image":"nginx:v2","name":"nginx","resources":{}}]}}},"status":{}}
      - image: nginx:v2
        imagePullPolicy: Always
 02:50:49  root@k8s-master01  ~  _                                                                                             ✓  241  0.20  17.12.22
```

## 3.edit

编辑 `deployment` 里的 `nginx` 容器：

```
kubectl edit deployments.apps nginx
```

可以编辑里面的任何内容，比如把基础镜像升级到v2版本，则将 `imgae: nginx` 改成 `image: nginx:v2` 。

同理，也可以编辑 `serviced` ：

```
kubectl edit svc/docker-registry
```

修改编辑操作时用的默认编辑器：

```
KUBE_EDITOR="nano" kubectl edit svc/docker-registry
```

单次使用生效，如果想永久生效则将此变量申明为环境变量：

bash下，写到 `~/.bashrc` 里面：

```
echo 'export KUBE_EDITOR="nano"' >> ~/.bashrc
```

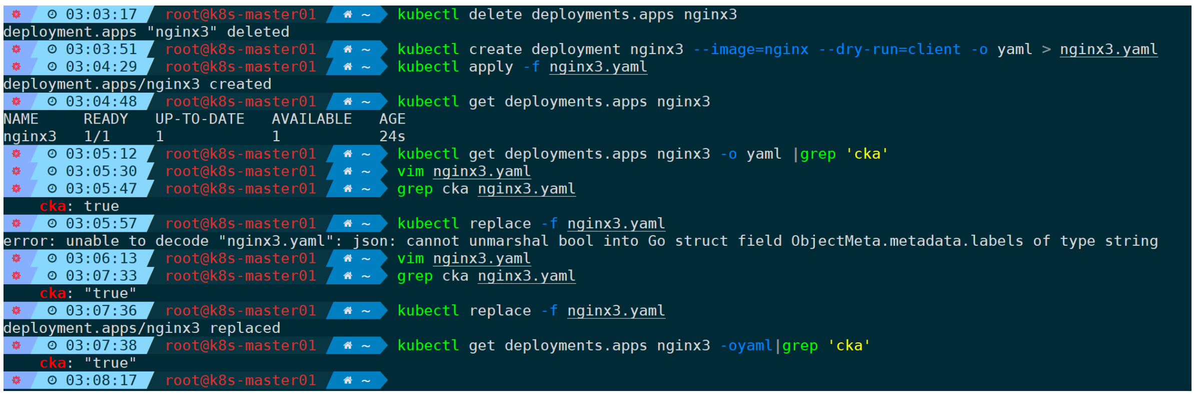
zsh下，写到 `~/.zshrc` 里面：

```
echo 'export KUBE_EDITOR="nano"' >> ~/.zshrc
```

## 4.replace

修改yaml文件后，用 `replace` 来替换升级：

```
kubectl replace -f nginx3.yaml  #这里也可以用apply，效果一样
```



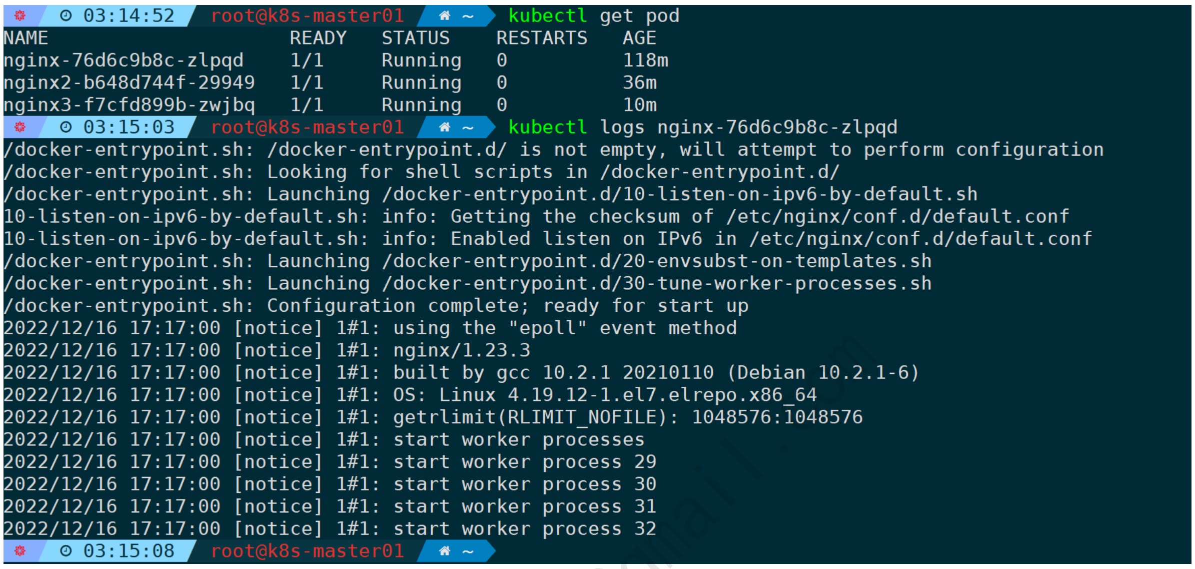
```
 03:03:17  root@k8s-master01  ~  kubectl delete deployments.apps nginx3
deployment.apps "nginx3" deleted
 03:03:51  root@k8s-master01  ~  kubectl create deployment nginx3 --image=nginx --dry-run=client -o yaml > nginx3.yaml
 03:04:29  root@k8s-master01  ~  kubectl apply -f nginx3.yaml
deployment.apps/nginx3 created
 03:04:48  root@k8s-master01  ~  kubectl get deployments.apps nginx3
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx3   1/1     1            1           24s
 03:05:12  root@k8s-master01  ~  kubectl get deployments.apps nginx3 -o yaml |grep 'cka'
 03:05:30  root@k8s-master01  ~  vim nginx3.yaml
 03:05:47  root@k8s-master01  ~  grep cka nginx3.yaml
    cka: true
 03:05:57  root@k8s-master01  ~  kubectl replace -f nginx3.yaml
error: unable to decode "nginx3.yaml": json: cannot unmarshal bool into Go struct field ObjectMeta.metadata.labels of type string
 03:06:13  root@k8s-master01  ~  vim nginx3.yaml
 03:07:33  root@k8s-master01  ~  grep cka nginx3.yaml
    cka: "true"
 03:07:36  root@k8s-master01  ~  kubectl replace -f nginx3.yaml
deployment.apps/nginx3 replaced
 03:07:38  root@k8s-master01  ~  kubectl get deployments.apps nginx3 -oyaml|grep 'cka'
    cka: "true"
 03:08:17  root@k8s-master01  ~
```

往yaml加了个标签，可以看到 `replace` 之后，标签成功加上。

## 五、查看日志

### 1.查看Pod日志

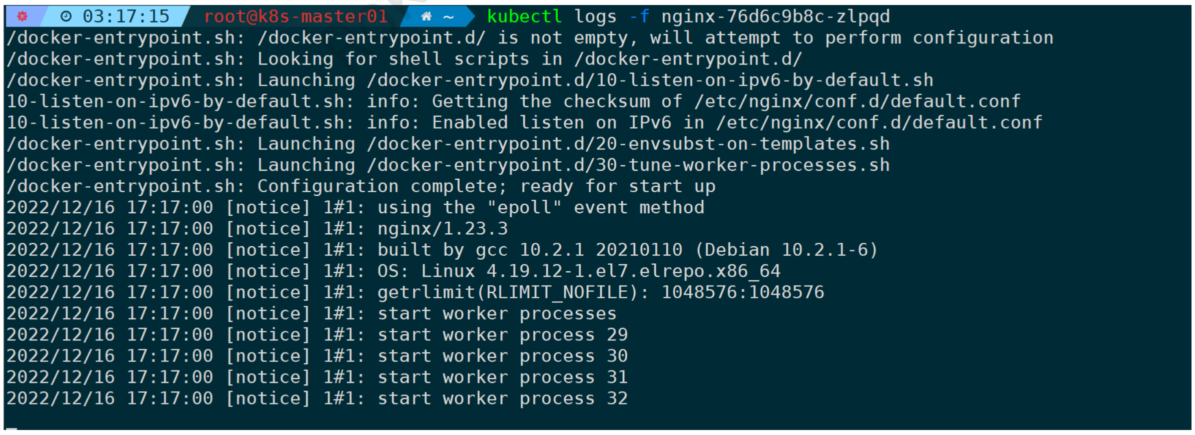
```
kubectl logs my-pod
```



```
 03:14:52  root@k8s-master01  ~  kubectl get pod
NAME                      READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-zlpqd     1/1     Running   0          118m
nginx2-b648d744f-29949    1/1     Running   0          36m
nginx3-f7cfd899b-zwjbq    1/1     Running   0          10m
 03:15:03  root@k8s-master01  ~  kubectl logs nginx-76d6c9b8c-zlpqd
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/12/16 17:17:00 [notice] 1#1: using the "epoll" event method
2022/12/16 17:17:00 [notice] 1#1: nginx/1.23.3
2022/12/16 17:17:00 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/12/16 17:17:00 [notice] 1#1: OS: Linux 4.19.12-1.el7.elrepo.x86_64
2022/12/16 17:17:00 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/12/16 17:17:00 [notice] 1#1: start worker processes
2022/12/16 17:17:00 [notice] 1#1: start worker process 29
2022/12/16 17:17:00 [notice] 1#1: start worker process 30
2022/12/16 17:17:00 [notice] 1#1: start worker process 31
2022/12/16 17:17:00 [notice] 1#1: start worker process 32
 03:15:08  root@k8s-master01  ~
```

### 2.动态输出Pod日志(-f)

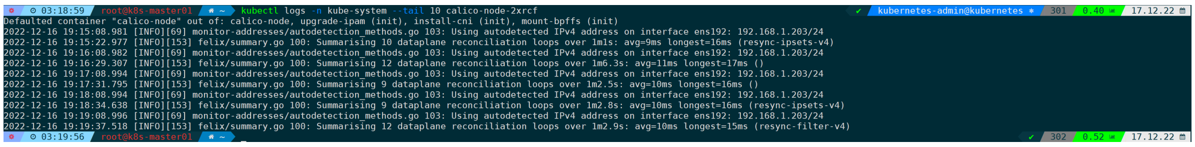
```
kubectl logs -f my-pod
```



```
 03:17:15  root@k8s-master01  ~  kubectl logs -f nginx-76d6c9b8c-zlpqd
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/12/16 17:17:00 [notice] 1#1: using the "epoll" event method
2022/12/16 17:17:00 [notice] 1#1: nginx/1.23.3
2022/12/16 17:17:00 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/12/16 17:17:00 [notice] 1#1: OS: Linux 4.19.12-1.el7.elrepo.x86_64
2022/12/16 17:17:00 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/12/16 17:17:00 [notice] 1#1: start worker processes
2022/12/16 17:17:00 [notice] 1#1: start worker process 29
2022/12/16 17:17:00 [notice] 1#1: start worker process 30
2022/12/16 17:17:00 [notice] 1#1: start worker process 31
2022/12/16 17:17:00 [notice] 1#1: start worker process 32
```

### 3.显示后N行(--tail N)

```
kubectl logs --tail 10 my-pod  #获取后10行
```

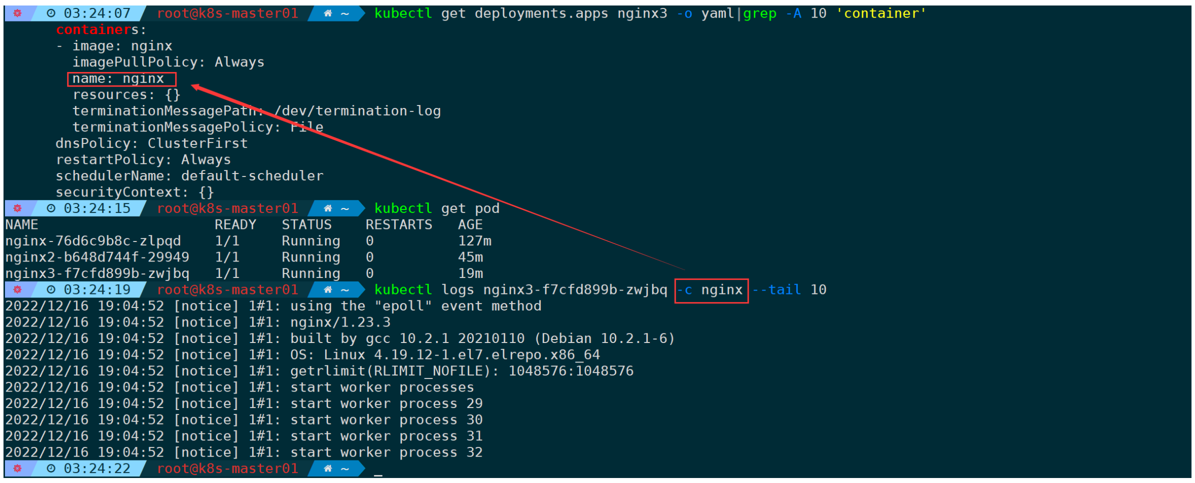


```
 03:18:59  root@k8s-master01  ~  kubectl logs -n kube-system --tail 10 calico-node-2xrcf      ✓  kubernetes-admin@kubernetes  301  0.40  17.12.22
Defaulted container "calico-node" out of: calico-node, upgrade-ipam (init), install-cni (init), mount-bpffs (init)
2022-12-16 19:15:08.981 [INFO][69] monitor-addresses/autodetection_methods.go 103: Using autodetected IPv4 address on interface ens192: 192.168.1.203/24
2022-12-16 19:15:22.977 [INFO][153] felix/summary.go 100: Summarising 10 dataplane reconciliation loops over 1m1s: avg=9ms longest=16ms (resync-ipsets-v4)
2022-12-16 19:16:08.982 [INFO][69] monitor-addresses/autodetection_methods.go 103: Using autodetected IPv4 address on interface ens192: 192.168.1.203/24
2022-12-16 19:16:29.307 [INFO][153] felix/summary.go 100: Summarising 12 dataplane reconciliation loops over 1m6.3s: avg=11ms longest=17ms ()
2022-12-16 19:17:08.994 [INFO][69] monitor-addresses/autodetection_methods.go 103: Using autodetected IPv4 address on interface ens192: 192.168.1.203/24
2022-12-16 19:17:31.795 [INFO][153] felix/summary.go 100: Summarising 9 dataplane reconciliation loops over 1m2.5s: avg=10ms longest=16ms ()
2022-12-16 19:18:08.994 [INFO][69] monitor-addresses/autodetection_methods.go 103: Using autodetected IPv4 address on interface ens192: 192.168.1.203/24
2022-12-16 19:18:34.638 [INFO][153] felix/summary.go 100: Summarising 9 dataplane reconciliation loops over 1m2.8s: avg=10ms longest=16ms (resync-ipsets-v4)
2022-12-16 19:19:08.996 [INFO][69] monitor-addresses/autodetection_methods.go 103: Using autodetected IPv4 address on interface ens192: 192.168.1.203/24
2022-12-16 19:19:37.518 [INFO][153] felix/summary.go 100: Summarising 12 dataplane reconciliation loops over 1m2.9s: avg=10ms longest=15ms (resync-filter-v4)
 03:19:56  root@k8s-master01  ~                                                                  ✓  302  0.52  17.12.22
```

## 4.多容器场景，指定容器获取(-c)

当一个pod里面有多个container时，使用 `-c` 来指定容器：

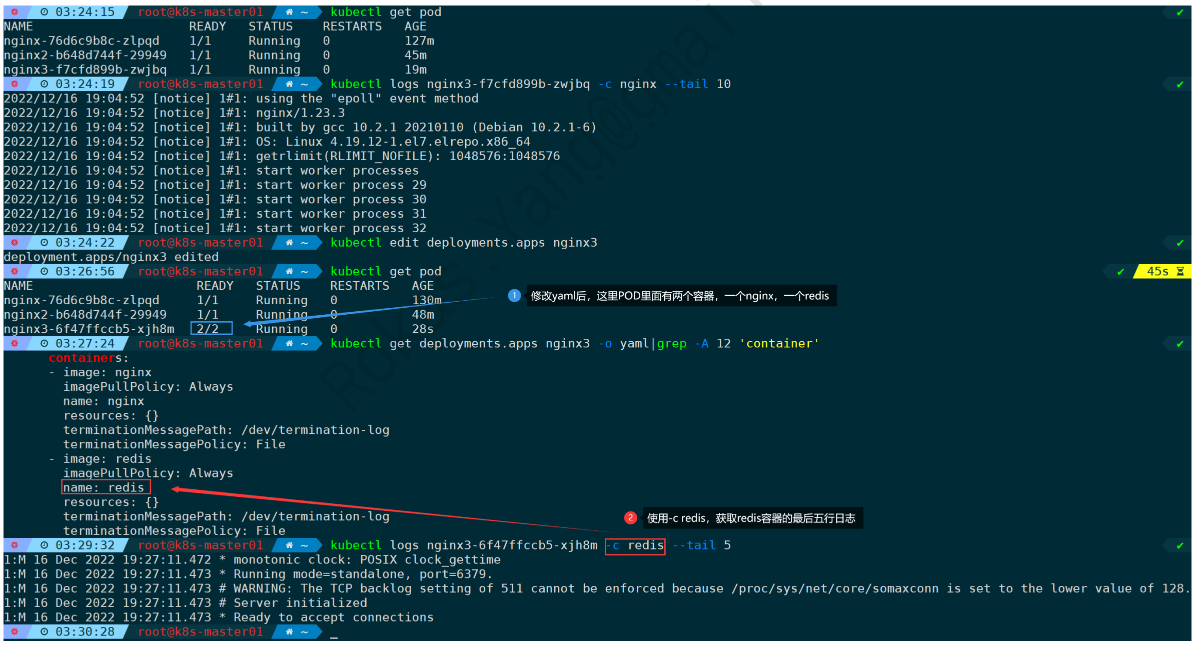
```
kubectl logs my-pod -c my-container
```



```
 03:24:07  root@k8s-master01  ~  kubectl get deployments.apps nginx3 -o yaml|grep -A 10 'container'
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
 03:24:15  root@k8s-master01  ~  kubectl get pod
NAME                      READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-zlpqd     1/1     Running   0          127m
nginx2-b648d744f-29949    1/1     Running   0          45m
nginx3-f7cfd899b-zwjbq    1/1     Running   0          19m
 03:24:19  root@k8s-master01  ~  kubectl logs nginx3-f7cfd899b-zwjbq -c nginx --tail 10
2022/12/16 19:04:52 [notice] 1#1: using the "epoll" event method
2022/12/16 19:04:52 [notice] 1#1: nginx/1.23.3
2022/12/16 19:04:52 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/12/16 19:04:52 [notice] 1#1: OS: Linux 4.19.12-1.el7.elrepo.x86_64
2022/12/16 19:04:52 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/12/16 19:04:52 [notice] 1#1: start worker processes
2022/12/16 19:04:52 [notice] 1#1: start worker process 29
2022/12/16 19:04:52 [notice] 1#1: start worker process 30
2022/12/16 19:04:52 [notice] 1#1: start worker process 31
2022/12/16 19:04:52 [notice] 1#1: start worker process 32
 03:24:22  root@k8s-master01  ~  _
```

这里只有一个容器，可以通过 `-c` 来指定。

给这个yaml文件，再加一个redis容器，则通过 `-c` 指定redis容器来获取最后五行日志：

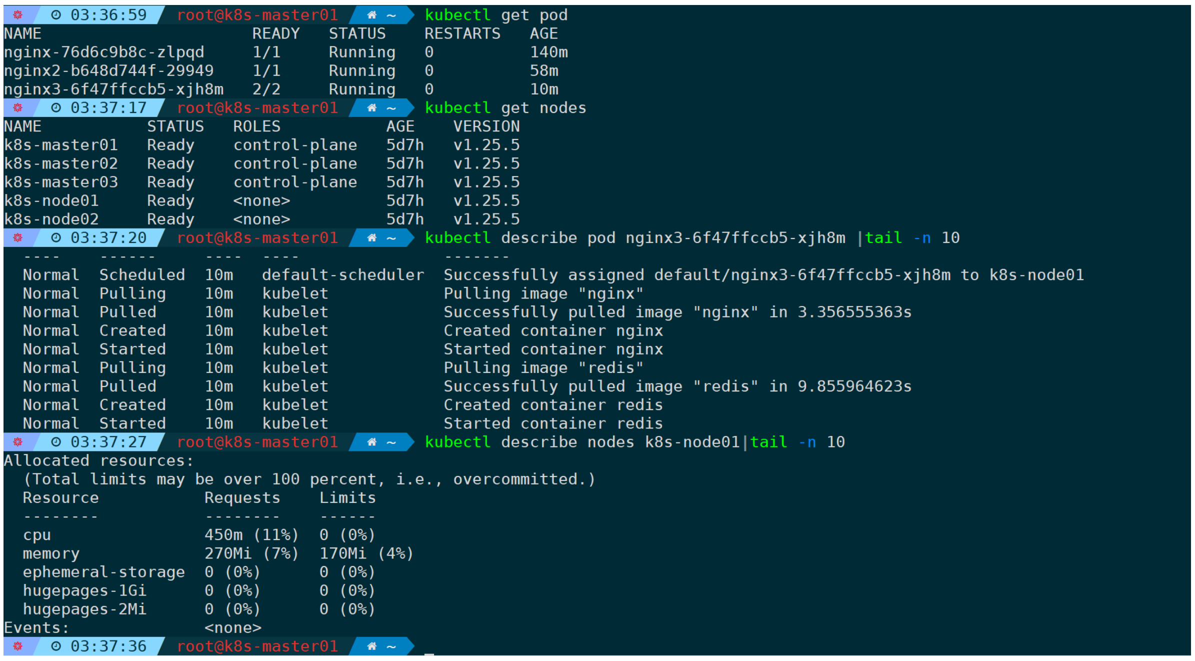


```
 03:24:15  root@k8s-master01  ~  kubectl get pod                                                    ✔
NAME                      READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-zlpqd     1/1     Running   0          127m
nginx2-b648d744f-29949    1/1     Running   0          45m
nginx3-f7cfd899b-zwjbq    1/1     Running   0          19m
 03:24:19  root@k8s-master01  ~  kubectl logs nginx3-f7cfd899b-zwjbq -c nginx --tail 10            ✔
2022/12/16 19:04:52 [notice] 1#1: using the "epoll" event method
2022/12/16 19:04:52 [notice] 1#1: nginx/1.23.3
2022/12/16 19:04:52 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/12/16 19:04:52 [notice] 1#1: OS: Linux 4.19.12-1.el7.elrepo.x86_64
2022/12/16 19:04:52 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/12/16 19:04:52 [notice] 1#1: start worker processes
2022/12/16 19:04:52 [notice] 1#1: start worker process 29
2022/12/16 19:04:52 [notice] 1#1: start worker process 30
2022/12/16 19:04:52 [notice] 1#1: start worker process 31
2022/12/16 19:04:52 [notice] 1#1: start worker process 32
 03:24:22  root@k8s-master01  ~  kubectl edit deployments.apps nginx3                              ✔
deployment.apps/nginx3 edited
 03:26:56  root@k8s-master01  ~  kubectl get pod                                           ✔  45s ⧗
NAME                       READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-zlpqd      1/1     Running   0          130m
nginx2-b648d744f-29949     1/1     Running   0          48m
nginx3-6f47ffccb5-xjh8m    2/2     Running   0          28s
 03:27:24  root@k8s-master01  ~  kubectl get deployments.apps nginx3 -o yaml|grep -A 12 'container'  ✔
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      - image: redis
        imagePullPolicy: Always
        name: redis
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
 03:29:32  root@k8s-master01  ~  kubectl logs nginx3-6f47ffccb5-xjh8m -c redis --tail 5            ✔
1:M 16 Dec 2022 19:27:11.472 * monotonic clock: POSIX clock_gettime
1:M 16 Dec 2022 19:27:11.473 * Running mode=standalone, port=6379.
1:M 16 Dec 2022 19:27:11.473 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
1:M 16 Dec 2022 19:27:11.473 # Server initialized
1:M 16 Dec 2022 19:27:11.473 * Ready to accept connections
 03:30:28  root@k8s-master01  ~  _
```

1 修改yaml后，这里POD里面有两个容器，一个nginx，一个redis
2 使用-c redis，获取redis容器的最后五行日志

## 5.查看Pod/Node状态(describe)

```
kubectl describe pod nginx3-6f47ffccb5-xjh8m
kubectl describe nodes k8s-node01|tail -n 10
```

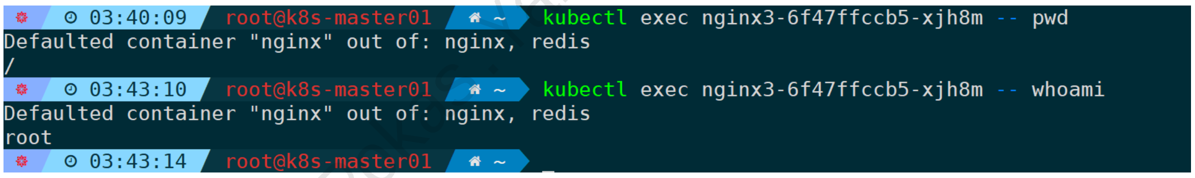


```
⊙ 03:36:59  root@k8s-master01  ~  kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
nginx-76d6c9b8c-zlpqd       1/1     Running   0          140m
nginx2-b648d744f-29949      1/1     Running   0          58m
nginx3-6f47ffccb5-xjh8m     2/2     Running   0          10m
⊙ 03:37:17  root@k8s-master01  ~  kubectl get nodes
NAME           STATUS   ROLES           AGE    VERSION
k8s-master01   Ready    control-plane   5d7h   v1.25.5
k8s-master02   Ready    control-plane   5d7h   v1.25.5
k8s-master03   Ready    control-plane   5d7h   v1.25.5
k8s-node01     Ready    <none>          5d7h   v1.25.5
k8s-node02     Ready    <none>          5d7h   v1.25.5
⊙ 03:37:20  root@k8s-master01  ~  kubectl describe pod nginx3-6f47ffccb5-xjh8m |tail -n 10
  ----    ------     ----   ----               -------
  Normal  Scheduled  10m    default-scheduler  Successfully assigned default/nginx3-6f47ffccb5-xjh8m to k8s-node01
  Normal  Pulling    10m    kubelet            Pulling image "nginx"
  Normal  Pulled     10m    kubelet            Successfully pulled image "nginx" in 3.356555363s
  Normal  Created    10m    kubelet            Created container nginx
  Normal  Started    10m    kubelet            Started container nginx
  Normal  Pulling    10m    kubelet            Pulling image "redis"
  Normal  Pulled     10m    kubelet            Successfully pulled image "redis" in 9.855964623s
  Normal  Created    10m    kubelet            Created container redis
  Normal  Started    10m    kubelet            Started container redis
⊙ 03:37:27  root@k8s-master01  ~  kubectl describe nodes k8s-node01|tail -n 10
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests    Limits
  --------           --------    ------
  cpu                450m (11%)  0 (0%)
  memory             270Mi (7%)  170Mi (4%)
  ephemeral-storage  0 (0%)      0 (0%)
  hugepages-1Gi      0 (0%)      0 (0%)
  hugepages-2Mi      0 (0%)      0 (0%)
Events:              <none>
⊙ 03:37:36  root@k8s-master01  ~  _
```

## 六、执行命令

### 1.在Pod里面执行命令

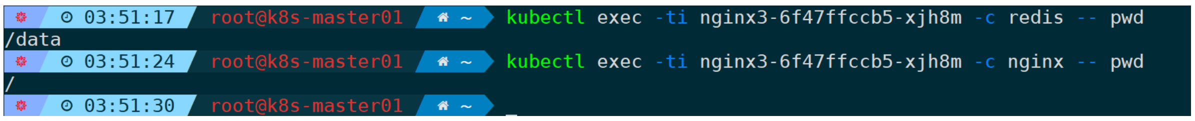
```
kubectl exec my-pod -- cmd
```



```
⊙ 03:40:09  root@k8s-master01  ~  kubectl exec nginx3-6f47ffccb5-xjh8m -- pwd
Defaulted container "nginx" out of: nginx, redis
/
⊙ 03:43:10  root@k8s-master01  ~  kubectl exec nginx3-6f47ffccb5-xjh8m -- whoami
Defaulted container "nginx" out of: nginx, redis
root
⊙ 03:43:14  root@k8s-master01  ~  _
```

### 2.在指定Pod的指定容器中执行命令

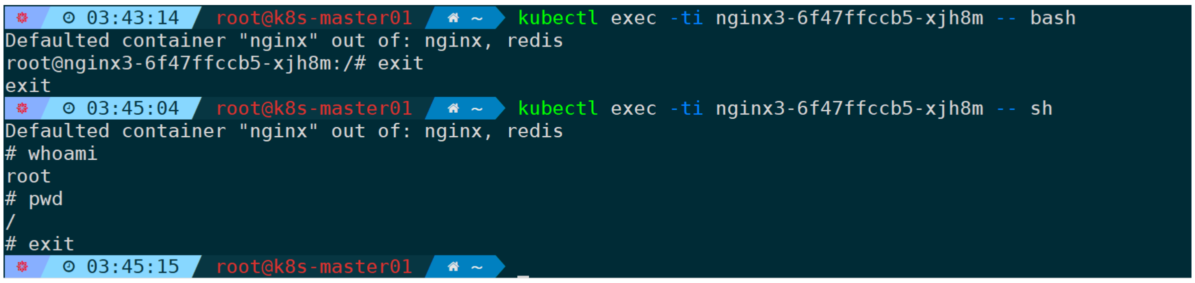
```
kubectl exec my-pod -c my-container -- cmd
```



```
⊙ 03:51:17  root@k8s-master01  ~  kubectl exec -ti nginx3-6f47ffccb5-xjh8m -c redis -- pwd
/data
⊙ 03:51:24  root@k8s-master01  ~  kubectl exec -ti nginx3-6f47ffccb5-xjh8m -c nginx -- pwd
/
⊙ 03:51:30  root@k8s-master01  ~  _
```

### 3.进入Pod

```
kubectl exec -ti my-pod -- bash
```


```
 03:43:14  root@k8s-master01  ~  kubectl exec -ti nginx3-6f47ffccb5-xjh8m -- bash
Defaulted container "nginx" out of: nginx, redis
root@nginx3-6f47ffccb5-xjh8m:/# exit
exit
 03:45:04  root@k8s-master01  ~  kubectl exec -ti nginx3-6f47ffccb5-xjh8m -- sh
Defaulted container "nginx" out of: nginx, redis
# whoami
root
# pwd
/
# exit
 03:45:15  root@k8s-master01  ~  _
```

有些pod没有bash，则用sh代替。

## 七、POD状态表

| 状态 | 说明 |
|---|---|
| Pending（挂起） | Pod已被Kubernetes系统接收，但仍有一个或多个容器未被创建，可以通过kubectl describe查看处于Pending状态的原因。 |
| Running（运行中） | Pod已被绑定到一个节点上，并且所有的容器都已经被创建，而且至少有一个是运行状态，或者是正在启动或重启，可以通过kubectl logs查看Pod日志。 |
| Succeeded（成功） | 所有容器都已终止，并且至少有一个容器以失败的方式终止，也就是说这个容器要么以非零状态退出，要么被系统终止，可以通过logs和describe查看Pod日志和状态。 |
| Unknown（未知） | 通常是由于通信问题造成的无法获得Pod的状态。 |
| ImagePullBackOffErrImagePull | 镜像拉取失败，一般是由于镜像不存在、网络不通或者需要登录认证引起的，可以使用describe命令查看具体原因。 |
| CrashLoopBackOff | 容器启动失败，可以通过logs命令查看具体原因，一般为启动命令不正确，健康检查不通过等。 |
| OOMKilled | 容器内存溢出，一般是容器的内存Limit设置的过小，或者程序本身有内存溢出，可以通过logs查看程序启动日志。 |
| Terminating | Pod正在被删除，可以通过describe查看状态。 |
| SysctlForbidden | Pod自定义了内核配置，但kubelet没有添加内核配置或配置的内核参数不支持，可以通过describe查看具体原因。 |
| Completed | 容器内部主进程退出，一般计划任务执行结束会显示该状态，此时可以通过logs查看容器日志。 |
| ContainerCreating | Pod正在创建，一般为正在下载镜像，或者有配置不当的地方，可以通过describe查看具体原因。 |