技术交流

微信：

luckylucky421302

加微信备注k8s即可进群交流学习



**kubectl简介**

kubectl是操作k8s集群的命令行工具，安装在k8s的master节点，kubectl在$HOME/.kube目录中查找一个名为config的文件, 你可以通过设置Kubeconfig环境变量或设置--kubeconfig来指定其他的kubeconfig文件。kubectl通过与apiserver交互可以实现对k8s集群中各种资源的增删改查。接下来将会介绍kubectl语法，命令行的操作，并介绍常见的示例。命令的详细信息、参数和子命令可在[kubectl](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands/)参考文档中查看。

**kubectl语法**

kubectl语法格式如下，可在k8s集群的master节点执行：

kubectl [command] [TYPE] [NAME] [flags]

**上述语法解释说明：**

**command：**指定要对一个或多个资源执行的操作，例如create、get、describe、delete等。

**type：**指定[资源类型](https://kubernetes.io/zh/docs/reference/kubectl/overview/#resource-types)。资源类型不区分大小写，可以指定单数、复数或缩写形式。例如，以下命令输出相同的结果:

kubectl get pod pod1

kubectl get pods pod1

kubectl get po pod1

**NAME：**指定资源的名称。名称区分大小写。如果省略名称，则显示所有资源的详细信息：kubectl get pods。

**flags:** 指定可选的参数。例如，可以使用-s或-server参数指定 Kubernetes API服务器的地址和端口。

**注意事项说明:**

从命令行指定的参数会覆盖默认值和任何相应的环境变量。

**1.在对多个资源执行操作时，可以按类型、名称、一个或者多个文件指定每个资源：**

**1）按类型和名称指定资源：**

要对所有类型相同的资源进行分组，请执行以下操作：

TYPE1 name1 name2 name<#>。  
例子：kubectl get pod example-pod1 example-pod2

分别指定多个资源类型：

TYPE1/name1 TYPE1/name2 TYPE2/name3 TYPE<#>/name<#>。

例：kubectl get pod/example-pod1 deployment/example-rc1

**2）用一个或多个文件指定资源：-f file1 -f file2 -f file<#>**

[使用YAML而不是JSON](https://kubernetes.io/docs/concepts/configuration/overview/#general-config-tips)，因为YAML更容易使用，特别是用于配置文件时。  
例子：kubectl get pod -f ./pod.yaml

**2.kubectl –-help**

可查看kubectl的帮助命令

**操作**

下面讲解和kubectl操作相关的命令和语法：

**1. annotate**

1）语法：  
kubectl annotate (-f FILENAME | TYPE NAME | TYPE/NAME) KEY\_1=VAL\_1 … KEY\_N=VAL\_N [--overwrite] [--all] [--resource-version=version] [flags]

2）描述：

添加或更新一个或多个资源的注释。

**\*\*2. api-versions**

1）语法：

kubectl api-versions [flags]

2）描述：

列出可用的api版本

**\*\*3.apply**

1）语法：

kubectl apply -f FILENAME [flags]

从文件或stdin对资源的应用配置进行更改。

**4.attach-不用**

1）语法：

kubectl attach POD-name -c CONTAINER-name [-i] [-t] [flags]

附加到正在运行的容器，查看输出流或与容器交互。

**\*\*5.autoscale**

1）语法：

kubectl autoscale (-f FILENAME | TYPE NAME | TYPE/NAME) [--min=MINPODS] --max=MAXPODS [--cpu-percent=CPU] [flags]

自动扩缩容由副本控制器管理的一组 pod。

**6.cluster-info**

1）语法：

kubectl cluster-info [flags]

显示有关集群中的主服务器和服务的端点信息。

**7.config**

1）语法：

kubectl config SUBCOMMAND [flags]

修改kubeconfig文件

**8.create-一般不用，用apply替代这个**

1）语法：

kubectl create -f FILENAME [flags]

从文件或标准输入创建一个或多个资源。

**\*\*9.delete**

1）语法：  
kubectl delete (-f FILENAME | TYPE [NAME | /NAME | -l label | --all]) [flags]

从文件、标准输入或指定标签选择器、名称、资源选择器或资源中删除资源。

**\*\*10.describe**

1）语法：

kubectl describe (-f FILENAME | TYPE [NAME\_PREFIX | /NAME | -l label]) [flags]

显示一个或多个资源的详细状态。

**11.diff**

1）语法：

kubectl diff -f FILENAME [flags]

将 live 配置和文件或标准输入做对比 (**BETA版**)

**\*\*12.edit**

1）语法：  
kubectl edit (-f FILENAME | TYPE NAME | TYPE/NAME) [flags]

使用默认编辑器编辑和更新服务器上一个或多个资源的定义。

**\*\*13.exec-常用的**

1）语法：

kubectl exec POD-name [-c CONTAINER-name] [-i] [-t] [flags] [-- COMMAND [args...]]

对 pod 中的容器执行命令。

下面的命令就是登录到pod中的容器的命令

kubectl exec calico-node-cblk2 -n kube-system -i -t -- /bin/sh

**\*\*14.explain-常用的**

1）语法：

kubectl explain [--recursive=false] [flags]

获取多种资源的文档。例如 pod, node, service 等，相当于帮助命令，可以告诉我们怎么创建资源

**\*\*15.expose**

1）语法：

kubectl expose (-f FILENAME | TYPE NAME | TYPE/NAME) [--port=port] [--protocol=TCP|UDP] [--target-port=number-or-name] [--name=name] [--external-ip=external-ip-of-service] [--type=type] [flags]

将副本控制器、服务或pod作为新的Kubernetes服务进行暴露。

**\*\*16.get**

1）语法：

kubectl get (-f FILENAME | TYPE [NAME | /NAME | -l label]) [--watch] [--sort-by=FIELD] [[-o | --output]=OUTPUT\_FORMAT] [flags]

列出一个或多个资源。

**17.label**

1）语法：

kubectl label (-f FILENAME | TYPE NAME | TYPE/NAME) KEY\_1=VAL\_1 … KEY\_N=VAL\_N [--overwrite] [--all] [--resource-version=version] [flags]

添加或更新一个或多个资源的标签。

**\*\*18.logs**

1）语法：  
kubectl logs POD [-c CONTAINER] [--follow] [flags]

在 pod 中打印容器的日志。

**19.patch**

1）语法：  
kubectl patch (-f FILENAME | TYPE NAME | TYPE/NAME) --patch PATCH [flags]

更新资源的一个或多个字段

**20.port-forward**

1）语法：

kubectl port-forward POD [LOCAL\_PORT:]REMOTE\_PORT [...[LOCAL\_PORT\_N:]REMOTE\_PORT\_N] [flags]

将一个或多个本地端口转发到Pod。

**21.proxy**

1）语法：

kubectl proxy [--port=PORT] [--www=static-dir] [--www-prefix=prefix] [--api-prefix=prefix] [flags]

运行Kubernetes API服务器的代理。

**22.replace**

1）语法：  
kubectl replace -f FILENAM

从文件或标准输入中替换资源。

**\*\*23.run**

1）语法：

kubectl run NAME --image=image [--env=“key=value”] [--port=port] [--dry-run=server | client | none] [--overrides=inline-json] [flags]

在集群上运行指定的镜像

**24.scale**

1）语法：

kubectl scale (-f FILENAME | TYPE NAME | TYPE/NAME) --replicas=COUNT [--resource-version=version] [--current-replicas=count] [flags]

更新指定副本控制器的大小。

**25.version**

1）语法：

kubectl version [--client] [flags]

显示运行在客户端和服务器上的 Kubernetes 版本

**举例说明：**

kubectl run nginx --image=nginx 可以创建一个nginx应用

实际创建pod应用都是通过编写资源清单文件进行创建

**注：**

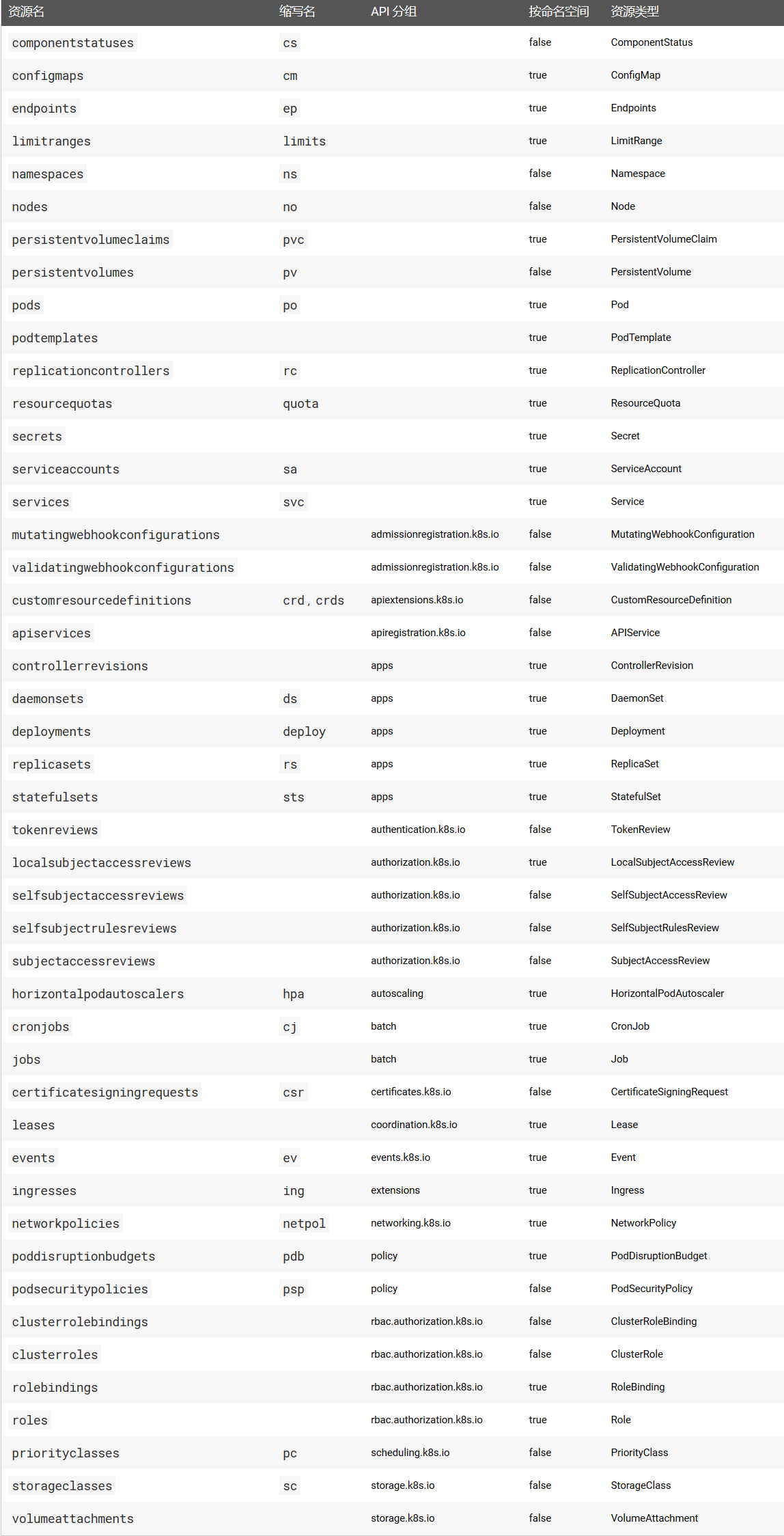
上面标注\*\*是常用命令，需要重点记一下

有关kubectl更详细的操作命令，可参考<https://kubernetes.io/docs/reference/kubectl/kubectl/>

**资源类型**

下表列出所有受支持的资源类型及其缩写别名:

(以下输出可以通过kubectl api-resources获取，内容以 Kubernetes 1.13.3 版本为准。)



**输出选项**

下面给大家介绍怎么对输出的命令进行格式化或者排序

**1.格式输出**

kubectl命令的默认输出格式是人类可读的明文格式，若要以特定格式向终端窗口输出详细信息，可以将-o或—out参数添加到受支持的kubectl命令中。

**2.语法**

kubectl [command] [TYPE] [NAME] -o=<output\_format>

根据 kubectl 操作，支持以下输出格式：



**\*\*示例：在此示例中，以下命令将单个 pod 的详细信息输出为 YAML 格式的对象：**

kubectl get pod web-pod-13je7 -o yaml

**注：**有关每个命令支持哪种输出格式的详细信息，可参考：

<https://kubernetes.io/docs/user-guide/kubectl/>

**3.自定义列**

要定义自定义列并仅将所需的详细信息输出到表中，可以使用custom-columns 选项。你可以选择内联定义自定义列或使用模板文件：-o=custom-columns=<spec> 或 -o=custom-columns-file=<filename>

示例：

1）内联：

kubectl get pods <pod-name> -o custom-columns=NAME:.metadata.name,RSRC:.metadata.resourceVersion

2）模板文件：

kubectl get pods <pod-name> -o custom-columns-file=template.txt

其中，template.txt文件内容是：

NAME RSRC

metadata.name metadata.resourceVersion

运行任何一个命令的结果是:

NAME RSRC

submit-queue 610995

**4.server-side 列**

kubectl支持从服务器接收关于对象的特定列信息。 这意味着对于任何给定的资源，服务器将返回与该资源相关的列和行，以便客户端打印。 通过让服务器封装打印的细节，这允许在针对同一集群使用的客户端之间提供一致的人类可读输出。默认情况下，此功能在kubectl 1.11及更高版本中启用。要禁用它，请将该--server-print=false参数添加到 kubectl get 命令中。

例子：

要打印有关 pod 状态的信息，请使用如下命令：

kubectl get pods <pod-name> --server-print=false

输出如下：

NAME AGE

nfs-provisioner-595dcd6b77-527np 5d21h

**5. 排序列表对象**

要将对象排序后输出到终端窗口，可以将--sort-by参数添加到支持的kubectl命令。通过使用--sort-by参数指定任何数字或字符串字段来对对象进行排序。要指定字段，请使用[jsonpath](https://kubernetes.io/docs/reference/kubectl/jsonpath/)表达式。

语法

kubectl [command] [TYPE] [NAME] --sort-by=<jsonpath\_exp>

示例

要打印按名称排序的pod列表，请运行：

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

NAME READY STATUS RESTARTS AGE

calico-node-cblk2 1/1 Running 40 7d8h

calico-node-q84kx 1/1 Running 31 7d8h

coredns-66bff467f8-f2nrb 1/1 Running 3 5d21h

coredns-66bff467f8-x24ff 1/1 Running 4 5d21h

etcd-master1 1/1 Running 7 7d8h

kube-apiserver-master1 1/1 Running 22 7d8h

kube-controller-manager-master1 1/1 Running 81 7d8h

kube-proxy-4xlzz 1/1 Running 4 7d8h

kube-proxy-pxjlx 1/1 Running 5 7d8h

kube-scheduler-master1 1/1 Running 72 7d8h

metrics-server-8459f8db8c-lvx9x 2/2 Running 2 5d21h

traefik-ingress-controller-c8dm6 1/1 Running 4 7d8h

traefik-ingress-controller-nr4n6 1/1 Running 6 7d8h

**列子：常用操作**

使用下面的列子帮你熟悉kubectl常用的操作

**1.kubectl apply**

用文件或者标准输入来应用或者更新k8s的资源

*# 使用 example-service.yaml 中的定义创建服务。*

kubectl apply -f example-service.yaml

*# 使用 example-controller.yaml 中的定义创建 replication controller。*

kubectl apply -f example-controller.yaml

*# 使用 <directory> 路径下的任意 .yaml, .yml, 或 .json 文件 创建对象。*

kubectl apply -f <directory>

**2.kubectl get**

列出一个或多个资源。

# 以纯文本输出格式列出所有 pod。

kubectl get pods

# 以纯文本输出格式列出所有 pod，并包含附加信息(如节点名)。

kubectl get pods -o wide

# 以纯文本输出格式列出具有指定名称的副本控制器。提示：你可以使用别名 'rc' 缩短和替换 'replicationcontroller' 资源类型。

kubectl get replicationcontroller <rc-name>

# 以纯文本输出格式列出所有副本控制器和服务。

kubectl get rc,services

# 以纯文本输出格式列出所有守护程序集。

kubectl get ds

# 列出在节点 server01 上运行的所有 pod

kubectl get pods --field-selector=spec.nodeName=server01

**3.kubectl describe**

显示一个或多个资源的详细状态，默认情况下包括未初始化的资源。

*# 显示名称为 <node-name> 的节点的详细信息。*

kubectl describe nodes <node-name>

*# 显示名为 <pod-name> 的 pod 的详细信息。*

kubectl describe pods/<pod-name>

*# 显示由名为 <rc-name> 的副本控制器管理的所有 pod 的详细信息。*

*# 记住：副本控制器创建的任何 pod 都以复制控制器的名称为前缀。*

kubectl describe pods <rc-name>

*# 描述所有的 pod*

kubectl describe pods

**注意：**

kubectl get 命令通常用于检索同一资源类型的一个或多个资源。 它具有丰富的参数，允许您使用 -o 或 --output 参数自定义输出格式。您可以指定 -w 或 --watch 参数以开始观察特定对象的更新。 kubectl describe 命令更侧重于描述指定资源的许多相关方面。它可以调用对 API 服务器 的多个 API 调用来为用户构建视图。 例如，该 kubectl describe node 命令不仅检索有关节点的信息，还检索在其上运行的 pod 的摘要，为节点生成的事件等。

**4.kubectl delete**

从文件、标注输入或指定标签选择器、名称、资源选择器或资源中删除资源。

*# 使用 pod.yaml 文件中指定的类型和名称删除 pod。*

kubectl delete -f pod.yaml

*# 删除标签名= <label-name> 的所有 pod 和服务。*

kubectl delete pods,services -l name=<label-name>

*# 删除所有具有标签名称= <label-name> 的 pod 和服务，包括未初始化的那些。*

kubectl delete pods,services -l name=<label-name> --include-uninitialized

*# 删除所有 pod，包括未初始化的 pod。*

kubectl delete pods --all

**5.kubectl exec**

对pod中的容器执行命令。

*# 从 pod <pod-name> 中获取运行 'date' 的输出。默认情况下，输出来自第一个容器。*

kubectl exec <pod-name> date

*# 运行输出 'date' 获取在容器的 <container-name> 中 pod <pod-name> 的输出。*

kubectl exec <pod-name> -c <container-name> date

*# 获取一个交互 TTY 并运行 /bin/bash <pod-name >。默认情况下，输出来自第一个容器。*

kubectl exec -ti <pod-name> -- /bin/bash

**6.kubectl logs**

打印 Pod 中容器的日志。

*# 从 pod 返回日志快照。*

kubectl logs <pod-name>

*# 从 pod <pod-name> 开始流式传输日志。这类似于 'tail -f' Linux 命令。*

kubectl logs -f <pod-name>