kubectl 实践

王天青 | 2024年7月

TRANSWARP
星 环 科 技

TRANSWARP
星 环 科 技

目录》
CONTENTS

1
chapter

kubectl 常用命令介绍

- ✓ 基本使用
- ✓ 进阶使用

TRANSWARP
星 环 科 技

1.1 create vs apply

➢ create/apply（部署资源）

```
# kubectl apply/create –f  [FILE NAME]
```

- kubectl create属于 Imperative command（祈使式命令），它明确告诉 kubectl 要创建某个资源或对象

- kubectl apply是 Declarative command（声明式命令），apply 并不告诉 kubectl 具体做什么，而是由 kubectl 根据后面 –f 中的 yaml 文件与 k8s 中对应的 object 对比，自动探测要进行哪些操作，比如如果 object 不存在，则 create；如果已经存在，则对比差异，update and replace

- 在实际使用中，推荐使用 apply 进行创建和更新，它能够应对 90% 的场景

➢ get（查看资源）

```
# kubectl get pods|replicationcontrollers|services|namespaces|nodes|…|events [options]
```

Command Options	Description
-o, --output=''	Output format. One of: json|yaml|wide|name|custom-columns=...|custom-columns-file=...|go-template=...|go-template-file=...|jsonpath=...|jsonpath-file=....
-a, --show-all=false	When printing, show all resources (default hide terminated pods.).
--all-namespaces=false	If present, list the requested object(s) across all namespaces.
-w, --watch=false	After listing/getting the requested object, watch for changes.
-l, --selector=''	Selector (label query) to filter on.
--show-labels=false	When printing, show all labels as the last column (default hide labels column).

- describe（查看资源详细信息）

```
# kubectl describe pods|replicationcontrollers|services|namespaces|nodes|…|events [options] [RESOURCE NAME]
```

Command Options	Description
--all-namespaces=false	If present, list the requested object(s) across all namespaces.
-l, --selector=''	Selector (label query) to filter on.

➢ logs（查看Pod内容器日志）

```
# kubectl logs POD [-c CONTAINER] [options]
```

Command Options	Description
-f, --follow=false	Specify if the logs should be streamed. 流式跟随显示
-c, --container=''	Print the logs of this container
--tail=-1	Lines of recent log file to display. Defaults to -1, showing all log lines.
--since=0s	Only return logs newer than a relative duration like 5s, 2m, or 3h.

- exec（在Pod容器中执行命令）

```
# kubectl exec POD [-c CONTAINER] -- COMMAND [args...] [options]
```

Command Options	Description
-c, --container=''	Print the logs of this container
-i, --stdin=false	Pass stdin to the container.
-t, --tty=false	Stdin is a TTY.

- edit（编辑资源）

```
# kubectl edit (RESOURCE/NAME | -f FILENAME) [options]
```

Command Options	Description
-f, --filename=[]	Filename, directory, or URL to files to use to edit the resource.
-o, --output='yaml'	Output format. One of: yaml|json.

➢ delete（删除资源）

```
# kubectl delete ([-f FILENAME] | [-k DIRECTORY] | TYPE [(NAME | -l label | --all)])
```

Command Options	Description
-f, --filename=[]	Filename, directory, or URL to files to use to edit the resource.
-o, --output='yaml'	Output format. One of: yaml|json.
--cascade[=true]	如果为true，级联删除指定资源所管理的其他资源（例如：被replication controller管理的所有pod）。默认为true。
--grace-period=-1	安全删除资源前等待的秒数。如果为负值则忽略该选项。
--all[=false]	使用[-all]选择所有指定的资源。
-l, --selector=""	用于过滤资源的Label
--timeout=0	删除资源的超时设置，0 表示系统根据待删除资源的大小决定超时时间

- rollout（回滚资源）

```
# kubectl rollout SUBCOMMAND
```

Sub command	Description
history	查看历史版本
pause	暂停资源
resume	恢复暂停资源
status	查看资源状态
undo	回滚版本

- 当前 rollout 仅对三种资源有效：
  - Statefulset
  - Daemonset
  - Deployment

➢ autoscale（弹性伸缩资源）

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

Sub command	Description
cpu-percent	The target average CPU utilization (represented as a percent of requested CPU) over all the pods.
max	The target average CPU utilization (represented as a percent of requested CPU) over all the pods.
min	The lower limit for the number of pods that can be set by the autoscaler

➢ 当前 autoscale 仅对三种资源有效：

- Statefulset
- ReplicaSet
- Deployment

➢ cordon vs taint

```
# kubectl cordon NODE
```

```
# kubectl taint NODE  key=value:effect
```

cordon	taint
添加系统污点并设置 node 为不可调度	添加用户自定义污点
Daemonset 可以正常调度	需要修改 daemonset 增加相应容忍
Pod 不会被驱逐	effect 为 NoExecution 时，pod 会被驱逐

➢ 资源缩写

类型	缩写
componentstatuses	cs
configmaps	cm
daemonsets	ds
deployments	deploy
endpoints	ep
event	ev
horizontalpodautoscalers	hpa
ingresses	ing
replicasets	rs
replicationcontrollers	rc

类型	缩写
namespaces	ns
nodes	no
statefulsets	sts
persistentvolumeclaims	pvc
persistentvolumes	pv
pods	po
podsecuritypolicies	psp
service	svc
serviceaccount	sa

2
chapter

kubectl 权限控制

- ✓ Rbac
- ✓ 只读 kubectl

TRANSWARP
星 环 科 技

Kubernetes 中所有的 API 对象，都保存在 Etcd 里。可是，对这些 API 对象的操作，却一定都是通过访问 kube-apiserver 实现的。其中一个非常重要的原因，就是你**需要 APIServer 来帮助你做授权工作。**

而在 Kubernetes 项目中，**负责完成授权（Authorization）工作的机制，就是 RBAC**：基于角色的访问控制（Role-Based Access Control）。

在 RBAC 中，有三个最基本的概念。

1.Role：角色，它其实是一组规则，定义了一组对 Kubernetes API 对象的操作权限。

2.Subject：被作用者，既可以是“人”，也可以是“机器”，也可以是你在 Kubernetes 里定义的“用户”。

3.RoleBinding：定义了“被作用者”和“角色”的绑定关系。

2.2 只读 kubectl

➢ 实现一个只有只读权限的 kubectl

1. 创建一个只读 clusterRole

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  name: mytest-view
rules:
- apiGroups:
  - "*"
  resources:
  - "*"
  verbs:
  - get
  - list
  - watch
```

2. 创建 ServiceAccount

```
kubectl create ns mytest
kubectl create sa -n mytest user-guest
```

2.2 只读 kubectl

3. 创建相应 clusterRoleBinding

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: mytest:user-guest
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: mytest-view
subjects:
- kind: ServiceAccount
name: user-guest
namespace: mytest
```

4. 生成相应配置

```
TOKEN=$(kubectl get sa -n mytest user-guest -o go-template='{{range .secrets}}{{.name}}{{end}}')
echo $TOKEN
CA_CERT=$(kubectl get secret -n mytest ${TOKEN} -o yaml | awk '/ca.crt:/{print $2}')
echo $CA_CERT

# 注意这里，需要改为你的apiserver的地址
API_SERVER="https://172.1.3.17:6443"
echo $API_SERVER

cat <<EOF > guest.config
apiVersion: v1
kind: Config
clusters:
- cluster:
certificate-authority-data: $CA_CERT
server: $API_SERVER
name: cluster
EOF
```

5. 使用只读配置

```
SECRET=$(kubectl -n mytest get secret ${TOKEN} -o go-template='{{.data.token}}')
kubectl config set-credentials mytest-guest --token=`echo ${SECRET} | base64 -d` --kubeconfig=guest.config
kubectl config set-context default --cluster=cluster --user=mytest-guest --kubeconfig=guest.config
kubectl config use-context default --kubeconfig=guest.config
```

6. 执行 kubectl apply 和 kubectl get 验证结果

3
chapter

kubectl 扩展

- ✓ kubectl plugin
- ✓ kubectl plugin 实践

TRANSWARP
星 环 科 技

- 安装插件
  - 插件是一个独立的可执行文件，名称以 kubectl- 开头。 要安装插件，将其可执行文件移动到 PATH 中的任何位置。

- 发现插件
  - kubectl 提供一个命令 **kubectl plugin list**，用于搜索路径查找有效的插件可执行文件。 执行此命令将遍历路径中的所有文件。任何以 kubectl- 开头的可执行文件都将在这个命令的输出中以它们在路径中出现的顺序显示。 任何以 kubectl- 开头的文件如果不可执行，都将包含一个警告。 对于任何相同的有效插件文件，都将包含一个警告。

```
[root@tos-37 ~]# kubectl plugin list
The following compatible plugins are available:

/opt/kubernetes/bin/kubectl-cleanupfailedpv
/opt/kubernetes/bin/kubectl-licence
/opt/kubernetes/bin/kubectl-migrollback
/opt/kubernetes/bin/kubectl-queue
/opt/kubernetes/bin/kubectl-recovermigenv
/usr/bin/kubectl-licence
  - warning: /usr/bin/kubectl-licence is overshadowed by a similarly named plugin: /opt/kubernetes/bin/kubectl-licence
Unable read directory "/root/bin" from your PATH: open /root/bin: no such file or directory. Skipping...
error: one plugin warning was found
```

3.2 kubectl plugin 实践

- 自定义插件：kubectl-foo

```
#!/bin/bash

# 可选的参数处理
if [[ "$1" == "version" ]]
then
    echo "1.0.0"
    exit 0
fi

# 可选的参数处理
if [[ "$1" == "config" ]]
then
    echo $KUBECONFIG
    exit 0
fi

echo "I am a plugin named kubectl-foo"
```

```
$ kubectl foo
$ kubectl foo version
$ kubectl foo config
```

- 命名插件:
  - 插件根据文件名确定要实现的命令路径，插件所针对的命令路径中的每个子命令都由破折号（-）分隔。 例如，当用户调用命令 **kubectl foo bar baz** 时，希望调用该命令的插件的文件名为 **kubectl-foo-bar-baz**

  - 如果你运行 **kubectl foo bar baz arg1 --flag=value arg2**，kubectl 的插件机制将首先尝试找到 最长可能名称的插件，在本例中是 kubectl-foo-bar-baz-arg1。 当没有找到这个插件时，kubectl 就会将最后一个以破折号分隔的值视为参数（在本例中为 arg1）， 并尝试找到下一个最长的名称 **kubectl-foo-bar-baz**。 在找到具有此名称的插件后，它将调用该插件，并在其名称之后将所有参数和标志传递给插件进程。

```
# 创建一个插件
echo -e '#!/bin/bash\n\necho "My first command-line argument was $1"' >
kubectl-foo-bar-baz
sudo chmod +x ./kubectl-foo-bar-baz

# 将插件放到 PATH 下完成"安装"
sudo mv ./kubectl-foo-bar-baz /usr/local/bin

# 确保 kubectl 能够识别我们的插件
kubectl plugin list

# 测试通过 "kubectl" 命令来调用我们的插件时可行的
# 即使我们给插件传递一些额外的参数或标志
kubectl foo bar baz arg1 --meaningless-flag=true
```

Q&A

TRANSWARP
星 环 科 技