### CronJob

CronJob用于以时间为基准周期性地执行任务，这些自动化任务和运行在Linux或UNIX系统上的CronJob一样。CronJob对于创建定期和重复任务非常有用，例如执行备份任务、周期性调度程序接口、发送电子邮件等。

对于Kubernetes 1.8以前的版本，需要添加--runtime-config=batch/v2alpha1=true参数至APIServer中，然后重启APIServer和Controller Manager用于启用API，对于1.8以后的版本无须修改任何参数，可以直接使用，本节的示例基于1.8以上的版本。

1. 创建CronJob

创建CronJob有两种方式，一种是直接使用kubectl创建，一种是使用yaml文件创建。

使用kubectl创建CronJob的命令如下：

kubectl run hello --schedule="\*/1 \* \* \* \*" --restart=OnFailure --image=busybox -- /bin/sh -c "date; echo Hello from the Kubernetes cluster"

对应的yaml文件如下：

apiVersion: batch/v1beta1

kind: CronJob

metadata:

name: hello

spec:

schedule: "\*/1 \* \* \* \*"

jobTemplate:

spec:

template:

spec:

containers:

- name: hello

image: busybox

args:

- /bin/sh

- -c

- date; echo Hello from the Kubernetes cluster

restartPolicy: OnFailure

说 明

本例创建一个每分钟执行一次、打印当前时间和Hello from the Kubernetes cluster的计划任务。

查看创建的CronJob：

$ kubectl get cj

NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE

hello \*/1 \* \* \* \* False 0 <none> 5s

等待1分钟可以查看执行的任务（Jobs）：

$ kubectl get jobs

NAME COMPLETIONS DURATION AGE

hello-1558779360 1/1 23s 32s

CronJob每次调用任务的时候会创建一个Pod执行命令，执行完任务后，Pod状态就会变成Completed，如下所示：

$ kubectl get po

NAME READY STATUS RESTARTS AGE

hello-1558779360-jcp4r 0/1 Completed 0 37s

可以通过logs查看Pod的执行日志：

$ kubectl logs -f hello-1558779360-jcp4r

Sat May 25 10:16:23 UTC 2019

Hello from the Kubernetes cluster

如果要删除CronJob，直接使用delete即可：

kubectl delete cronjob hello

2. 可用参数的配置

定义一个CronJob的yaml文件如下：

apiVersion: v1

items:

- apiVersion: batch/v1beta1

kind: CronJob

metadata:

labels:

run: hello

name: hello

namespace: default

spec:

concurrencyPolicy: Allow

failedJobsHistoryLimit: 1

jobTemplate:

metadata:

creationTimestamp: null

spec:

template:

metadata:

creationTimestamp: null

labels:

run: hello

spec:

containers:

- args:

- /bin/sh

- -c

- date; echo Hello from the Kubernetes cluster

image: busybox

imagePullPolicy: Always

name: hello

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

dnsPolicy: ClusterFirst

restartPolicy: OnFailure

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

schedule: '\*/1 \* \* \* \*'

successfulJobsHistoryLimit: 3

suspend: false

其中各参数的说明如下，可以按需修改：

* schedule：调度周期，和Linux一致，分别是分时日月周。
* restartPolicy：重启策略，和Pod一致。
* concurrencyPolicy：并发调度策略。可选参数如下：
* Allow：允许同时运行多个任务。
* Forbid：不允许并发运行，如果之前的任务尚未完成，新的任务不会被创建。
* Replace：如果之前的任务尚未完成，新的任务会替换的之前的任务。
* suspend：如果设置为true，则暂停后续的任务，默认为false。
* successfulJobsHistoryLimit：保留多少已完成的任务，按需配置。
* failedJobsHistoryLimit：保留多少失败的任务。

相对于Linux上的计划任务，Kubernetes的CronJob更具有可配置性，并且对于执行计划任务的环境只需启动相对应的镜像即可。比如，如果需要Go或者PHP环境执行任务，就只需要更改任务的镜像为Go或者PHP即可，而对于Linux上的计划任务，则需要安装相对应的执行环境。此外，Kubernetes的CronJob是创建Pod来执行，更加清晰明了，查看日志也比较方便。可见，Kubernetes的CronJob更加方便和简单。