Jobs and CronJobs in Kubernetes

A job creates one or more Pods and ensures that a specified number of them successfully terminate. Kubernetes Jobs are created to run pods for a short period of time which are running for completion as opposed to different objects in Kubernetes like Replicasets, Replication Controllers, Demonesets, which is run continuously.

In Kubernetes, we have two types of jobs:

1. **Run to completion**
2. **Cronjobs**

Run to Completion job

Run to completion jobs are basically ensure that a specified number of pods are successfully completed their task and terminated. As pods successfully complete, the Job tracks the successful completions. When a specified number of successful completions is reached, the Job is complete.

Example of the run to completion job

1. **Create job.yml manifest file**

```
apiVersion: batch/v1
kind: Job
metadata:
  name: simple-job
spec:
  completions: 5
  parallelism: 2
  ttlSecondsAfterFinished: 1
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["echo", "Kubernetes Job"]
      restartPolicy: Never
```

The above manifest file has the following terms:

1. `completions`: This is set to a specified number means this job will run a specified number of pods for completing the task.
2. `parallelism`: This means that the number of pods running in parallel to complete a task.
3. `ttlSecondsAfterFinished`: This is the controller which is used to clean up the completed job after a specified time.

Let's deploy the job using the command:

```
$ kubectl create -f job.yml
```

To check the job status use command:

```
$ kubectl get jobs

NAME         COMPLETIONS   DURATION   AGE
simple-job   5/5           41s        104s
```

Checking for pods initiated by jobs:

```
$ kubectl get pods

NAME               READY   STATUS      RESTARTS   AGE
simple-job-6949g   0/1     Completed   0          105s
simple-job-829nb   0/1     Completed   0          118s
simple-job-c8l5g   0/1     Completed   0          2m13s
simple-job-hzwnd   0/1     Completed   0          111s
simple-job-p7rlh   0/1     Completed   0          2m13s
```

The above job completed all the tasks and then terminated.

Cron jobs

It runs a job periodically on a given schedule, written in Cron format.

CronJobs are useful for creating periodic and recurring tasks, like running **backups or sending emails**.

CronJobs can also schedule individual tasks for a specific time, such as scheduling a Job for when your cluster is likely to be idle.

Example for cronjob

1. **Create a cronjob.yml manifest:**

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"  # run every minute
  successfulJobsHistoryLimit: 1
  failedJobsHistoryLimit: 1
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

This YAML will schedule a pod every one minute

`schedule:` schedule time of its jobs to be created and executed.

`successfulJobsHistoryLimit: 1`
`failedJobsHistoryLimit: 1`
It specifies how many completed and failed jobs should be kept in a cluster.

Let’s deploy the job using the command:

```
$ kubectl create -f cronjob.yml
```

To check the job status use command:

```
$ kubectl get cronjobs

NAME    SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
hello   */1 * * * *   False     0        <none>          36s
```

Checking for pods initiated by jobs:

```
$ kubectl get pods

NAME                     READY   STATUS      RESTARTS   AGE
hello-1603945620-pw2nr   0/1     Completed   0          63s
hello-1603945680-rv6h2   0/1     Completed   0          12s
```

You can see it basically run the job every 1 minute.

1. Cleaning up the things

You can use the following command to delete cronjobs:

```
$ kubectl delete cronjobs/hello
```

Conclusion

Kubernetes Jobs are used when you want to create pods that will do a **specific task and then exit**.