

Exercise 6.3: Working with Jobs

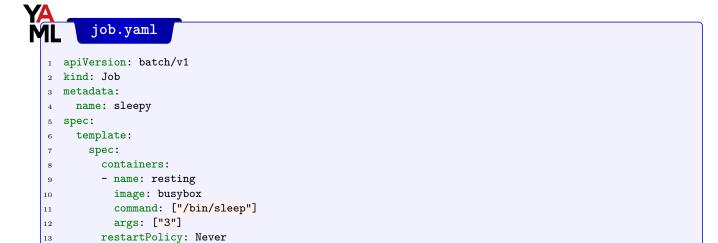
While most API objects are deployed such that they continue to be available there are some which we may want to run a particular number of times called a Job, and others on a regular basis called a CronJob

Create A Job

14

1. Create a job which will run a container which sleeps for three seconds then stops.

```
student@cp:~$ cp /home/student/LFS258/SOLUTIONS/s_06/job.yaml .
student@cp:~$ vim job.yaml
```



2. Create the job, then verify and view the details. The example shows checking the job three seconds in and then again after it has completed. You may see different output depending on how fast you type.

```
student@cp:~$ kubectl create -f job.yaml
job.batch/sleepy created
```

```
student@cp:~$ kubectl get job
```

```
NAME COMPLETIONS DURATION AGE
sleepy 0/1 3s 3s
```

student@cp:~\$ kubectl describe jobs.batch sleepy



```
Completions: 1
Start Time: Thu, 23 Aug 2024 10:47:53 +0000
Completed At: Thu, 23 Aug 2024 10:48:00 +0000
Duration: 5s
Pods Statuses: 0 Running / 1 Succeeded / 0 Failed
<output_omitted>
```

student@cp:~\$ kubectl get job

```
NAME COMPLETIONS DURATION AGE
sleepy 1/1 5s 17s
```

3. View the configuration information of the job. There are three parameters we can use to affect how the job runs. Use **-o yaml** to see these parameters. We can see that backoffLimit, completions, and the parallelism. We'll add these parameters next.

```
student@cp:~$ kubectl get jobs.batch sleepy -o yaml
```

```
<output_omitted>
    uid: c2c3a80d-d0fc-11e8-947a-42010a800002
spec:
    backoffLimit: 6
    completions: 1
    parallelism: 1
    selector:
        matchLabels:
<output_omitted>
```

4. As the job continues to AGE in a completion state, delete the job.

```
student@cp:~$ kubectl delete jobs.batch sleepy
```

```
job.batch "sleepy" deleted
```

5. Edit the YAML and add the completions: parameter and set it to 5.

```
student@cp:~$ vim job.yaml
```

```
job.yaml

coutput_omitted>
metadata:
name: sleepy
spec:
completions: 5 #<--Add this line
template:
spec:
containers:
coutput_omitted>
```

6. Create the job again. As you view the job note that COMPLETIONS begins as zero of 5.

```
student@cp:~$ kubectl create -f job.yaml
```



```
job.batch/sleepy created

student@cp:~$ kubectl get jobs.batch

NAME COMPLETIONS DURATION AGE
```

7. View the pods that running. Again the output may be different depending on the speed of typing.

5s

student@cp:~\$ kubectl get pods

sleepy

```
NAME READY STATUS RESTARTS AGE
sleepy-z5tnh 0/1 Completed 0 8s
sleepy-zd692 1/1 Running 0 3s
<output_omitted>
```

8. Eventually all the jobs will have completed. Verify then delete the job.

```
student@cp:~$ kubectl get jobs
```

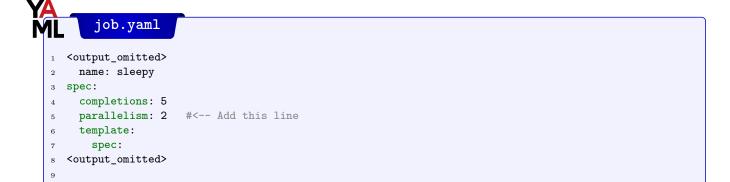
```
NAME COMPLETIONS DURATION AGE
sleepy 5/5 26s 10m
```

```
student@cp:~$ kubectl delete jobs.batch sleepy
```

```
job.batch "sleepy" deleted
```

9. Edit the YAML again. This time add in the parallelism: parameter. Set it to 2 such that two pods at a time will be deployed.

```
student@cp:~$ vim job.yaml
```



10. Create the job again. You should see the pods deployed two at a time until all five have completed.

```
student@cp:~$ kubectl create -f job.yaml
```

```
job.batch/sleepy created
```

student@cp:~\$ kubectl get pods



3

```
READY
                                                RESTARTS
NAME
                                     STATUS
                                                           AGE
sleepy-8xwpc
                             1/1
                                     Running
                                                0
                                                           5s
                             1/1
                                     Running
                                                0
                                                           5s
sleepy-xjqnf
<output_omitted>
```

student@cp:~\$ kubectl get jobs

```
NAME COMPLETIONS DURATION AGE
sleepy 3/5 11s 11s
```

11. Add a parameter which will stop the job after a certain number of seconds. Set the activeDeadlineSeconds: to 15. The job and all pods will end once it runs for 15 seconds. We will also increase the sleep argument to five, just to be sure does not expire by itself.

student@cp:~\$ vim job.yaml

```
<output_omitted>
     completions: 5
     parallelism: 2
     activeDeadlineSeconds: 15
                                #<-- Add this line
4
     template:
5
       spec:
6
         containers:
8
         - name: resting
9
           image: busybox
           command: ["/bin/sleep"]
10
           args: ["5"]
                                  #<-- Edit this line
11
   <output_omitted>
12
13
```

12. Delete and recreate the job again. It should run for 15 seconds, usually 3/5, then continue to age without further completions.

```
student@cp:~$ kubectl delete jobs.batch sleepy
```

```
job.batch "sleepy" deleted
```

student@cp:~\$ kubectl create -f job.yaml

```
job.batch/sleepy created
```

student@cp:~\$ kubectl get jobs

```
NAME COMPLETIONS DURATION AGE
sleepy 1/5 6s 6s
```

student@cp:~\$ kubectl get jobs

```
NAME COMPLETIONS DURATION AGE
sleepy 3/5 16s 16s
```



13. View the message: entry in the Status section of the object YAML output.

```
student@cp:~$ kubectl get job sleepy -o yaml
```

```
<output_omitted>
status:
   conditions:
   - lastProbeTime: 2024-08-23T10:48:00Z
    lastTransitionTime: 2024-08-23T10:48:00Z
   message: Job was active longer than specified deadline
   reason: DeadlineExceeded
   status: "True"
   type: Failed
   failed: 2
   startTime: 2024-08-23T10:48:00Z
   succeeded: 3
```

14. Delete the job.

```
student@cp:~$ kubectl delete jobs.batch sleepy
```

```
job.batch "sleepy" deleted
```

Create a CronJob

A CronJob creates a watch loop which will create a batch job on your behalf when the time becomes true. We Will use our existing Job file to start.

1. Copy the yaml file from the tarball.

```
student@cp:~$ cp /home/student/LFS258/SOLUTIONS/s_06/cronjob.yaml .
```

2. Verify the file to look like the annotated file shown below. Edit the lines mentioned below if needed. The three parameters we added will need to be removed. Other lines will need to be further indented if needed.

```
student@cp:~$ vim cronjob.yaml
```

```
apiVersion: batch/v1
                               #<-- Update this line to CronJob
2 kind: CronJob
3 metadata:
   name: sleepy
5 spec:
   schedule: "*/2 * * * *" #<-- Add Linux style cronjob syntax
    jobTemplate:
                              #<-- New jobTemplate and spec move
      spec:
9
        template:
                             #<-- This and following lines move
          spec:
                               #<-- four spaces to the right
10
            containers:
11
12
             - name: resting
              image: busybox
13
14
              command: ["/bin/sleep"]
15
              args: ["5"]
16
            restartPolicy: Never
17
```



3. Create the new CronJob. View the jobs. It will take two minutes for the CronJob to run and generate a new batch Job.

```
student@cp:~$ kubectl create -f cronjob.yaml
```

```
cronjob.batch/sleepy created
```

student@cp:~\$ kubectl get cronjobs.batch

```
NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE sleepy */2 * * * * False 0 <none> 8s
```

student@cp:~\$ kubectl get jobs.batch

```
No resources found.
```

4. After two minutes you should see jobs start to run.

student@cp:~\$ kubectl get cronjobs.batch

```
NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE sleepy */2 * * * * False 0 21s 2m1s
```

student@cp:~\$ kubectl get jobs.batch

```
NAME COMPLETIONS DURATION AGE sleepy-1539722040 1/1 5s 18s
```

student@cp:~\$ kubectl get jobs.batch

5. Ensure that if the job continues for more than 10 seconds it is terminated. We will first edit the **sleep** command to run for 30 seconds then add the activeDeadlineSeconds: entry to the container.

```
student@cp:~$ vim cronjob.yaml
```

```
jobTemplate:
      spec:
3
         template:
4
           spec:
5
             activeDeadlineSeconds: 10 #<-- Add this line
6
7
             containers:
             - name: resting
           command: ["/bin/sleep"]
10
           args: ["30"]
                                         #<-- Edit this line
11
        restartPolicy: Never
12
13
```





6. Delete and recreate the CronJob. It may take a couple of minutes for the batch Job to be created and terminate due to the timer.

```
student@cp:~$ kubectl delete cronjobs.batch sleepy
```

```
cronjob.batch "sleepy" deleted
```

student@cp:~\$ kubectl create -f cronjob.yaml

```
cronjob.batch/sleepy created
```

student@cp:~\$ kubectl get jobs

```
NAME COMPLETIONS DURATION AGE
sleepy-1539723240 0/1 61s 61s
```

student@cp:~\$ kubectl get cronjobs.batch

```
NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE sleepy */2 * * * * False 1 72s 94s
```

student@cp:~\$ kubectl get jobs

```
NAME COMPLETIONS DURATION AGE
sleepy-1539723240 0/1 75s 75s
```

student@cp:~\$ kubectl get jobs

NAME	COMPLETIONS	DURATION	AGE
sleepy-1539723240	0/1	2m19s	2m19s
sleepy-1539723360	0/1	19s	19s

student@cp:~\$ kubectl get cronjobs.batch

```
NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE sleepy */2 * * * * False 2 31s 2m53s
```

7. Clean up by deleting the CronJob.

```
student@cp:~$ kubectl delete cronjobs.batch sleepy
```

```
cronjob.batch "sleepy" deleted
```

