Running multiple pod instances in a Job

Jobs may be configured to create more than one pod instance and run them in parallel or sequentially. This is done by setting the completions and the parallelism entries in the Job spec

Running job pods sequentially

If you need a Job to run more than once, you set `completions` to how many times you want the Job's pod to run. The following listing shows an example.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: multi-completion-batch-job
spec:
  completions: 5
  template:
    *
    *
```

Setting completions to 5 makes this Job run five pods sequentially

This Job will run five pods one after the other. It initially creates one pod, and when the pod's container finishes, it creates the second pod, and so on, until five pods complete successfully. If one of the pods fails, the Job creates a new pod, so the Job may create more than five pods overall.

Running job pods in parallel

Instead of running single Job pods one after the other, you can also make the Job run multiple pods in parallel. You specify how many pods are allowed to run in parallel with the `parallelism` Job spec property, as shown in the following listing.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: multi-completion-batch-job
spec:
  completions: 5
  parallelism: 2
  template:
```

This job must ensure five pods complete successfully.

Up to two pods can run in parallel.

As soon as one of them finishes, the Job will run the next pod, until five pods finishsuccessfully.

Limiting the time allowed for a Job pod to complete

How long should the Job wait for a pod to finish? What if the pod gets stuck and can't finish at all (or it can't finish fastenough)?

A pod's time can be limited by setting the **`activeDeadlineSeconds`** property in the pod spec. If the pod runs longer than that, the system will try to terminate it and will mark the Job as failed.