The main function of a job is to create one or more pod and tracks about the success of pods. They ensure that the specified number of pods are completed successfully. When a specified number of successful run of pods is completed, then the job is considered complete.

Creating a Job

Use the following command to create a job −

```
apiVersion: v1
kind: Job ------------------------> 1
metadata:
   name: py
   spec:
   template:
      metadata
      name: py -------> 2
      spec:
         containers:
            - name: py ------------------------> 3
            image: python----------> 4
            command: ["python", "SUCCESS"]
            restartPocliy: Never --------> 5
```

In the above code, we have defined −

kind: Job → We have defined the kind as Job which will tell **kubectl** that the **yaml** file being used is to create a job type pod.

Name:py → This is the name of the template that we are using and the spec defines the template.

name: py → we have given a name as **py** under container spec which helps to identify the Pod which is going to be created out of it.

Image: python → the image which we are going to pull to create the container which will run inside the pod.

restartPolicy: Never →This condition of image restart is given as never which means that if the container is killed or if it is false, then it will not restart itself.

We will create the job using the following command with yaml which is saved with the name **py.yaml**.

```
$ kubectl create –f py.yaml
```

The above command will create a job. If you want to check the status of a job, use the following command.

```
$ kubectl describe jobs/py
```

The above command will create a job. If you want to check the status of a job, use the following command.

Scheduled Job

Scheduled job in Kubernetes uses **Cronetes**, which takes Kubernetes job and launches them in Kubernetes cluster.

Scheduling a job will run a pod at a specified point of time.

A parodic job is created for it which invokes itself automatically.

Note − The feature of a scheduled job is supported by version 1.4 and the betch/v2alpha 1 API is turned on by passing the **–runtime-config=batch/v2alpha1** while bringing up the API server.

We will use the same yaml which we used to create the job and make it a scheduled job.

```
apiVersion: v1
kind: Job
metadata:
   name: py
spec:
   schedule: h/30 * * * * ? ------------------> 1
   template:
      metadata
         name: py
      spec:
         containers:
         - name: py
         image: python
         args:
/bin/sh -------> 2
-c
ps –eaf ------------> 3
restartPocliy: OnFailure
```

In the above code, we have defined −

schedule: h/30 * * * * ? → To schedule the job to run in every 30 minutes.

/bin/sh: This will enter in the container with /bin/sh

ps –eaf → Will run ps -eaf command on the machine and list all the running process inside a container.

This scheduled job concept is useful when we are trying to build and run a set of tasks at a specified point of time and then complete the process.