***Jobs,init container and pod lifecycle***

***Jobs:***

Jobs are also important object of our k8s. This object are created when we want to do some task in one time, once the task is done it will terminated automatically

**Pod vs jobs**  
  
pods are always available & in job pods is created for a particular task once task is done pod will be terminated automatically

***Use cases:***

* Taske backup of a DB
* HELM Charts uses jobs
* Running batch process
* Run a task at an schedule time
* Log rotations

***LAB***

* Nano jobs.yml

apiVersion: batch/v1

kind: Job

metadata:

name: testjob

spec:

template:

metadata:

name: testjob

spec:

containers:

- name: counter

image: centos:7

command: ["bin/bash", "-c", "echo It's sleeping time; sleep 5"]

restartPolicy: Never

* kubectl apply -f jobs.yml
* kubectl get pods

//pods is created and done his work and then terminated that why status is completed shown

//in our pods we give job to run container only 5sec that why container run and delete it automatically after jib is done.



***Parallelism in JOBS***

When you need multiple pod to run the task for the limited time is the concept of parallelism in jobs.

***LABS:***

* nano job2.yml

apiVersion: batch/v1

kind: Job

metadata:

name: testjob2

spec:

parallelism: 5 # Runs for pods in parallel

activeDeadlineSeconds: 10 # Timesout after 30 sec

template:

metadata:

name: testjob2

spec:

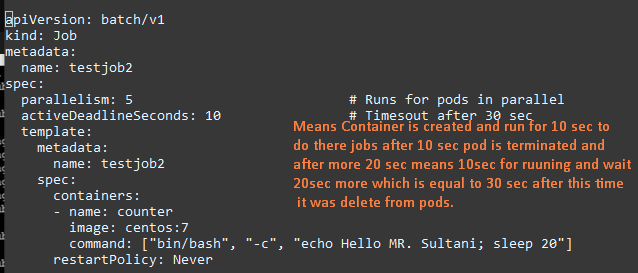
containers:

- name: counter

image: centos:7

command: ["bin/bash", "-c", "echo Hello MR. Sultani; sleep 20"]

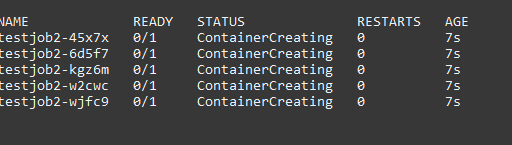
restartPolicy: Never



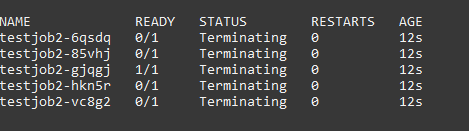
* kubectl apply -f job2.yml
* watch kubectl get pods

// see pod is created and terminated after 30 sec,when container did their jobs .

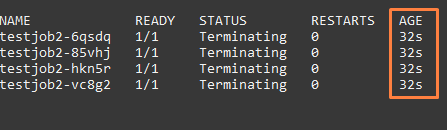
//container is created and running ifor 10 sec



//after 10sec it will terminated



//and after 30 sec means 10 sec for running & 20 sec for waiting time it wil deleted from pods



***Cron Job Patterns***

schedule pod to run on specific time and then terminated

* nano cronjob.yml

apiVersion: batch/v1

kind: CronJob

metadata:

name: Qasim

spec:

schedule: "\* \* \* \* \*"

jobTemplate:

spec:

template:

spec:

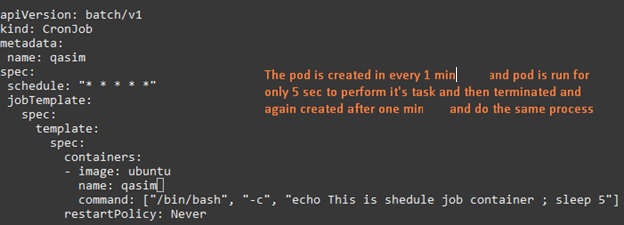
containers:

- image: ubuntu

name: Qasim

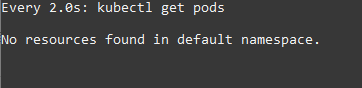
command: ["/bin/bash", "-c", "echo Technical-Guftgu; sleep 5"]

restartPolicy: Never

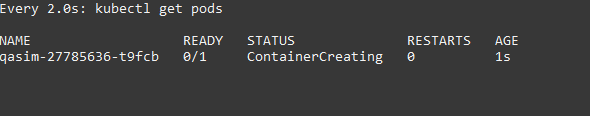


* kubectl apply -f cornjob.yml
* watch kubectl get pods

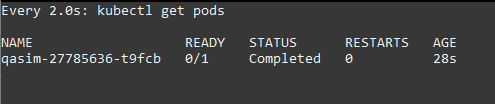
//pod is created after one minutes



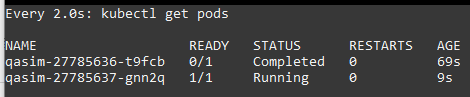
//now after one minute pod is created



//pod run for 5 sec and do his job and then complete is task & terminated



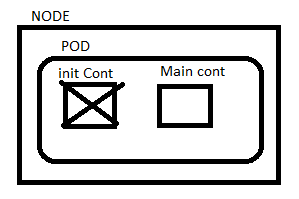
//after one minutes again pod is created and run and terminated this process work continuously



***Init Container***

Suppose you have to order food online so before ordering the food there are some necessary steps you have to do like install the application register with them, make profile and then you are able to order food. **Same as** init container are there when we have to do some task there which is necessary we want to execute and then run pods

In this case in pod container is created and before running the main container you have to do some settings or installation so the one separate con is create for the initial steps and other container is created for main task and after main container is created init container will be delete automatically



* Init containers are specialized container that run before main container in a pod
* Init containers always run to completion
* If a pods init container fail k8s repeatedly restart the pod until the init container succeeds

***Use cases:***

* Seeding a databases //before working on databases there is some tables & collection available
* Delaying the application launch until the dependency was ready //running the application that need some dependency e.g installion of Jenkins java jdk available
* Clone a git repo into a volume
* Generate configuration file dynamically

***LAB***

* Nano initcont.yml

apiVersion: v1

kind: Pod

metadata:

name: initcontainer

spec:

initContainers:

- name: c1

image: centos

command: ["/bin/bash", "-c", "echo Hello Qasim > /tmp/xchange/testfile; sleep 30"]

volumeMounts:

- name: xchange

mountPath: "/tmp/xchange"

containers:

- name: c2

image: centos

command: ["/bin/bash", "-c", "while true; do echo `cat /tmp/data/testfile`; sleep 5; done"]

volumeMounts:

- name: xchange

mountPath: "/tmp/data"

volumes:

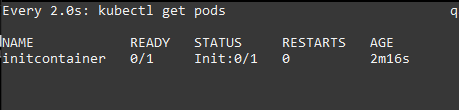
- name: xchange

emptyDir: {}

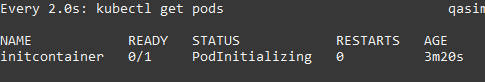
kubectl apply -f initcont.yml

watch kubectl get pods

// here is init container pod is working that was main pod is stll not created



//pod is initializing

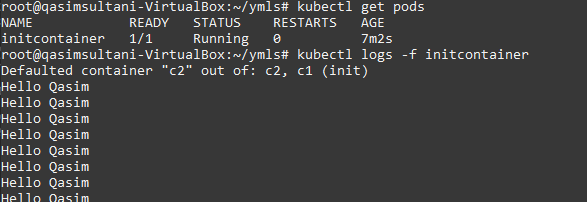


// init work has been done now our container is created and running



// now to check logs are printed which we define in our commands is working in every 5 sec we define.

* Kubectl get pods
* kubectl logs -f initcontainer



***POD Lifecycle***

**Pending:** when your resource are allocated & image is pulling.

**Running:** when your container is working properly. Even there is multiple pods in container and one is running and other are fail/stop it’s shown running

**Succeeded:** when your given task is done by pod it’s shown success status.

**Failed:** when your pod is not created successfully or not done work properly.

**Completed:** when your given task is done and pod is terminated successfully. “this state not shown because it work very fast ”

**Unknown:** when your kublet unable to fine where is pod running, in other words state of the pod could not be obtained. This error has occur due to an error in network or communication with the host of the pod**.**