**POD Init containers**

* Whenever we want to run a task which is pre-requisite of the app container in that cases we use initContainer.
* InitContainers will be run/create before app container defined under containers section.
* InitContainers will not be in running state, they will be in completed state after completion of its task.
* It works similar to actual containers, but each initContainer must complete successfully before next one starts
* If initContainer fails, kubelet restarts initContainer until it is succeeds. (If restart policy not set to never)
* InitContainers do not support probes and lifecycle.
* To do some initial setup before actual application.
* To run some code/script as a part of initialization before app gets created.
* To do some kernel changes for application specific,

1. Open visual studio and connect it with kubernetes VM.
2. Open the folder k8-yaml/POD/ in visual studio. Create a new file with the name initContainer.yaml

Paste the below content in that file

apiVersion: v1

kind: Pod

metadata:

  name: nginx-pod

  labels:

   app: nginx

   type: webserver

   object: initContainer

spec:

  containers:

    - name: nginx-container

      image: nginx

      ports:

        - containerPort: 80

      volumeMounts:

        - name: share-volume

          mountPath: /usr/share/nginx/html

  initContainers:

    - name: init-container

      image: alpine

      command:

        - wget

        - "-O"

        - "/var/tmp/index.html"

        - "https://github.com/swapnachatla/initContainers/blob/main/index.html?raw=true"

      volumeMounts:

        - name: share-volume

          mountPath: /var/tmp

  volumes:

    - name: share-volume

      emptyDir: {}

In the code we are creating a POD that contains one container (nginx) and initContainer. In both of them initContainer will be create first after the completion of the assigned task, the container will be in completed state. The data of the initContainer will be stored in the volume. Now the kubelet will start creating the container (nginx) with the same volume.

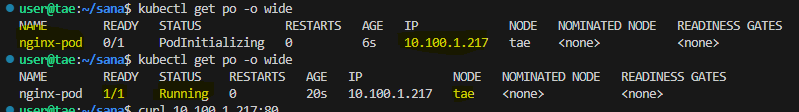
For executing the code run the below command:

# kubectl create -f POD/initContainer.yaml



Now check the status of the POD use the below command

# kubectl get po -o wide



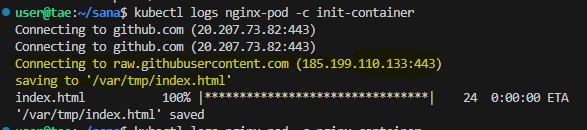
In the above picture Pod is created at first it is showing PodInitializing state. Finally pod is successfully created with the ready state 1/1 & IP of the Pod: 10.100.1.217

# kubectl describe po nginx-pod

Observe the Output carefully.

# kubectl logs nginx-pod -c init-container

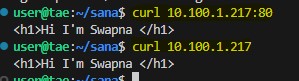
This command will show the logs of the container whose name is “init-container” of the Pod “nginx-pod”



The init-container gets created & it communicated with GitHub and copy the data to the mentioned path /var/tmp/index.html. it 24 millisec to save the file. After the task is completed the init-container gets terminated.

Now Try to access the application by using Pod IP. Because of using the same volume

# curl 10.100.1.217:80



In the above picture notice the thing. In the two cases we are getting the application. By default it will listen.

# kubectl replace –force –f POD/nginx-multicontainer.yaml