**3. Kubernetes Init Containers**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/05-Kubernetes-Important-Concepts-for-Application-Deployments/05-02-Kubernetes-Init-Containers>

--- <https://kubernetes.io/docs/concepts/workloads/pods/init-containers/> - learn about pod inti containers.

**Introduction**

--- Init Containers run before App containers

--- Init containers can contain utilities or setup scripts not present in an app image.

--- We can have and run multiple Init Containers before App Container.

--- Init containers are exactly like regular containers, except:

--- Init containers always run to completion.

--- Each init container must complete successfully before the next one starts.

--- If a Pod's init container fails, Kubernetes repeatedly restarts the Pod until the init container succeeds.

--- **Note** - However, if the Pod has a restart Policy of Never, Kubernetes does not restart the Pod.

**Implement Init Containers**

--- Update initContainers section under Pod Template Spec which is **spec.template.spec** in a Deployment

 template:

    metadata:

      labels:

        app: usermgmt-restapp

    spec:

      initContainers:

        - name: init-db # inti container name.

          image: busybox:1.31

          command: ['sh', '-c', 'echo -e "Checking for the availability of MySQL Server deployment"; while ! nc -z mysql 3306; do sleep 1; printf "-"; done; echo -e "  >> MySQL DB Server has started";']

--- **06-UserManagementMicroservice-Deployment-Service.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: usermgmt-microservice

  labels:

    app: usermgmt-restapp

spec:

  replicas: 1

  selector:

    matchLabels:

      app: usermgmt-restapp

  template:

    metadata:

      labels:

        app: usermgmt-restapp

    spec:

      initContainers:

        - name: init-db

          image: busybox:1.31

          command: ['sh', '-c', 'echo -e "Checking for the availability of MySQL Server deployment"; while ! nc -z mysql 3306; do sleep 1; printf "-"; done; echo -e "  >> MySQL DB Server has started";']

      containers:

        - name: usermgmt-restapp

          image: stacksimplify/kube-usermanagement-microservice:1.0.0

          ports:

            - containerPort: 8095

          env:

            - name: DB\_HOSTNAME

              value: "mysql"

            - name: DB\_PORT

              value: "3306"

            - name: DB\_NAME

              value: "usermgmt"

            - name: DB\_USERNAME

              value: "root"

            - name: DB\_PASSWORD

              valueFrom:

                secretKeyRef:

                  name: mysql-db-password

                  key: db-password

**Create & Test**

**# Create All Objects**

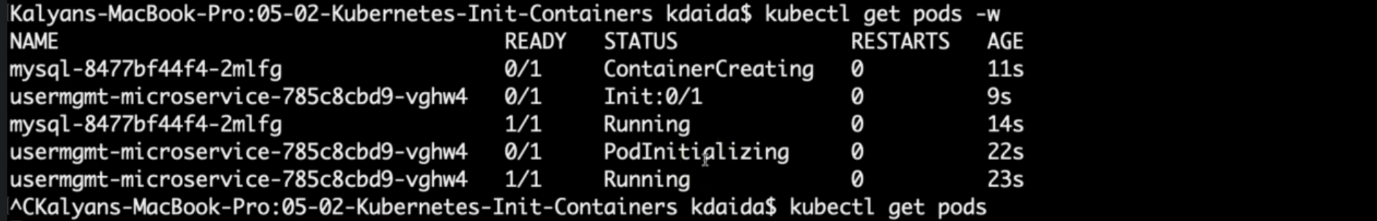
--- **kubectl apply -f kube-manifests/**

**# List Pods**

--- **kubectl get pods**

**# Watch List Pods screen**

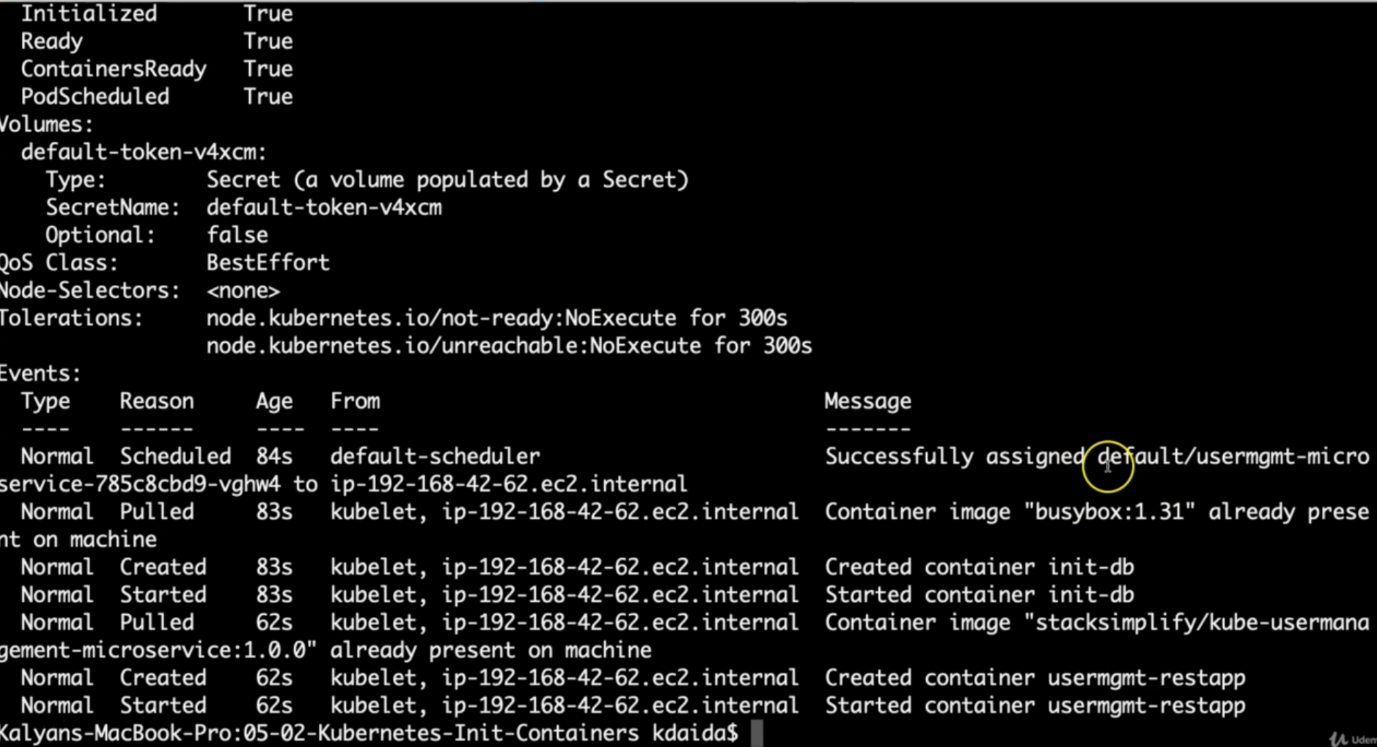
--- **kubectl get pods -w**



--- **note** – pod init container is initializing, it will check the mysql server is up or not, once the mysql servers is up then it will be terminated and the next container will be started. In our case the next container is **usermgmt-microservice**.

**# Describe Pod & Discuss about init container**

--- **kubectl describe pod <usermgmt-microservice-xxxxxx>**



--- **note** – you can see that 2types of containers present inside of this pod. 1st one is inti container and 2nd one is usermgmt-microservice container.

**# Access Application Health Status Page**

--- **http://<WorkerNode-Public-IP>:31231/usermgmt/health-status**

**Clean-Up**

--- Delete all k8s objects created as part of this section

**# Delete All**

--- **kubectl delete -f kube-manifests/**

**# List Pods**

--- **kubectl get pods**

**# Verify sc, pvc, pv**

--- **kubectl get sc,pvc,pv**