**6. Kubernetes Resources - Requests & Limits**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/05-Kubernetes-Important-Concepts-for-Application-Deployments/05-04-Kubernetes-Requests-Limits>

--- References - <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

**Introduction**

--- We can specify how much each container a pod needs the resources like CPU & Memory.

--- When we provide this information in our pod, the scheduler uses this information to decide which node to place the Pod on.

--- When you specify a resource limit for a Container, the kubelet enforces those limits so that the running container is not allowed to use more of that resource than the limit you set.

--- The kubelet also reserves at least the request amount of that system resource specifically for that container to use.

**Add Requests & Limits**

          resources:

            requests:

              memory: "128Mi" # 128 MebiByte is equal to 135 Megabyte (MB)

              cpu: "500m" # `m` means milliCPU

            limits:

              memory: "500Mi"

              cpu: "1000m"  # 1000m is equal to 1 VCPU core

--- **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

          livenessProbe:

            exec:

              command:

                - /bin/sh

                - -c

                - nc -z localhost 8095

            initialDelaySeconds: 60

            periodSeconds: 10

          readinessProbe:

            httpGet:

              path: /usermgmt/health-status

              port: 8095

            initialDelaySeconds: 60

            periodSeconds: 10

          resources:

            requests:

              cpu: "500m"

              memory: "128Mi"

            limits:

              cpu: "1000m"

              memory: "500Mi"

**Create k8s objects & Test**

**# Create All Objects**

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

**# List Pods**

--- **kubectl get pods**

**# Watch List Pods screen**

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

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

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

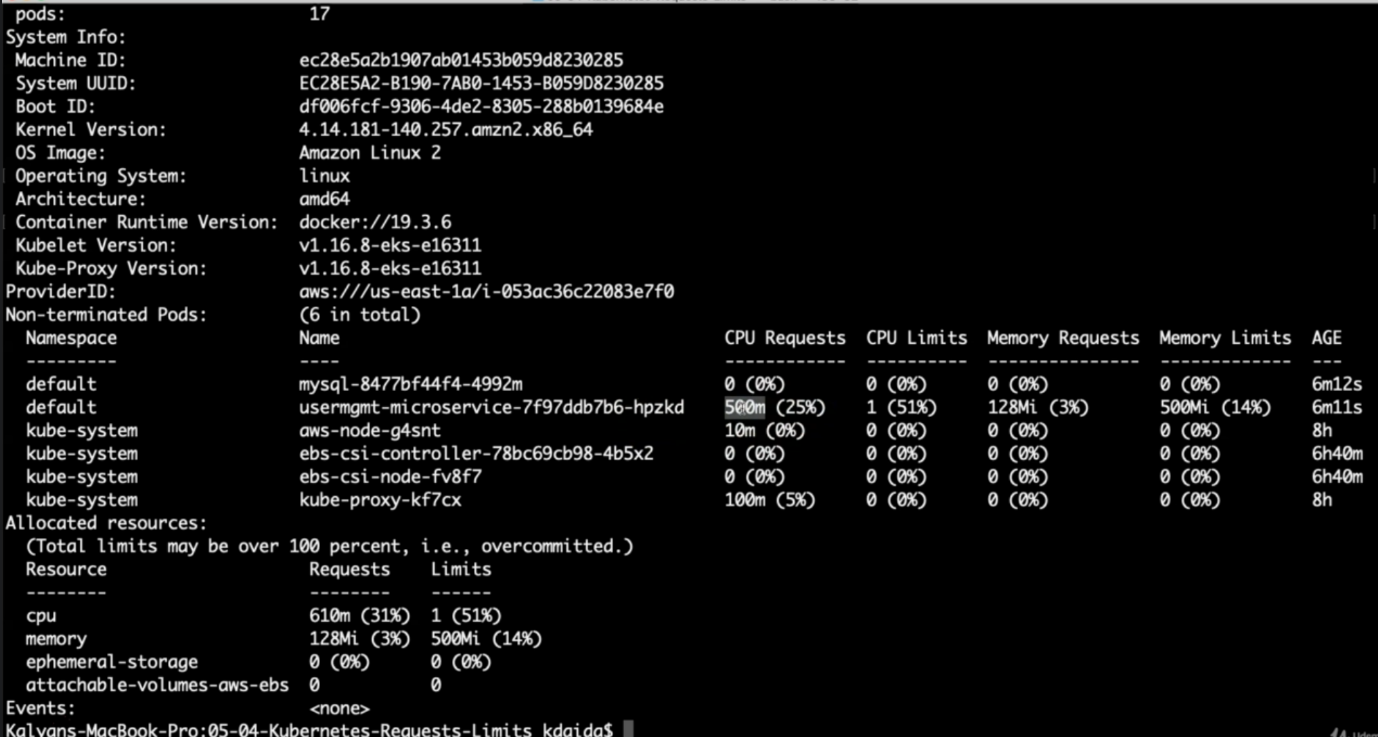
**# Access Application Health Status Page**

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

**# List Nodes & Describe Node**

--- **kubectl get nodes**

--- **kubectl describe node <Node-Name>**



--- **note** – for user management microservice pod, we have allocated some resources. In the screenshot we can see that the resources successfully allocated to that pod.

**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**