**10. Kubernetes Namespaces - Create Limit Range k8s manifest**

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

--- **References** - <https://kubernetes.io/docs/tasks/administer-cluster/namespaces-walkthrough/>

--- **References** - <https://kubernetes.io/docs/tasks/administer-cluster/manage-resources/cpu-default-namespace/>

--- **References** - <https://kubernetes.io/docs/tasks/administer-cluster/manage-resources/memory-default-namespace/>

**Create Namespace manifest**

--- **Important Note**: File name starts with 00- so that when creating k8s objects namespace will get created first so it doesn’t throw an error.

apiVersion: v1

kind: Namespace

metadata:

  name: dev3

**Create LimitRange manifest**

--- **note** - Instead of specifying resources like cpu and memory in every container spec of a pod definition, we can provide the default CPU & Memory for all containers in a namespace using LimitRange

apiVersion: v1

kind: ResourceQuota

metadata:

  name: ns-resource-quota

  namespace: dev3

spec:

  limits:

    - default:

        memory: "512Mi" # If not specified the Container's memory limit is set to 512Mi, which is the default memory limit for the namespace.

        cpu: "500m"  # If not specified default limit is 1 vCPU per container

      defaultRequest:

        memory: "256Mi" # If not specified default it will take from whatever specified in limits.default.memory

        cpu: "300m" # If not specified default it will take from whatever specified in limits.default.cpu

      type: Container

--- **00-namespace-LimitRange-default.yml**

apiVersion: v1

kind: Namespace

metadata:

  name: dev3

---

apiVersion: v1

kind: LimitRange

metadata:

  name: default-cpu-mem-limit-range

  namespace: dev3

spec:

  limits:

    - default:

        cpu: "500m"  # If not specified default limit is 1 vCPU per container

        memory: "512Mi" # If not specified the Container's memory limit is set to 512Mi, which is the default memory limit for the namespace.

      defaultRequest:

        cpu: "300m" # If not specified default it will take from whatever specified in limits.default.cpu

        memory: "256Mi" # If not specified default it will take from whatever specified in limits.default.memory

      type: Container

--- **note** – 1st I am creating namespace and then I am crating resources in the name space.

--- 02-persistent-volume-claim.yml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: ebs-mysql-pv-claim

  namespace: dev3

spec:

  accessModes:

    - ReadWriteOnce

  storageClassName: ebs-sc

  resources:

    requests:

      storage: 4Gi

--- 03-UserManagement-ConfigMap.yml

apiVersion: v1

kind: ConfigMap

metadata:

  name: usermanagement-dbcreation-script

  namespace: dev3

data:

  mysql\_usermgmt.sql: |-

    DROP DATABASE IF EXISTS usermgmt;

    CREATE DATABASE usermgmt;

--- 04-mysql-deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: mysql

  namespace: dev3

spec:

  replicas: 1

  selector:

    matchLabels:

      app: mysql

  strategy:

    type: Recreate

  template:

    metadata:

      labels:

        app: mysql

    spec:

      containers:

        - name: mysql

          image: mysql:5.6

          env:

            - name: MYSQL\_ROOT\_PASSWORD

              valueFrom:

                secretKeyRef:

                  name: mysql-db-password

                  key: db-password

          ports:

            - containerPort: 3306

              name: mysql

          volumeMounts:

            - name: mysql-persistent-storage

              mountPath: /var/lib/mysql

            - name: usermanagement-dbcreation-script

              mountPath: /docker-entrypoint-initdb.d #https://hub.docker.com/\_/mysql Refer Initializing a fresh instance

      volumes:

        - name: mysql-persistent-storage

          persistentVolumeClaim:

            claimName: ebs-mysql-pv-claim

        - name: usermanagement-dbcreation-script

          configMap:

            name: usermanagement-dbcreation-script

--- 05-mysql-clusterip-service.yml

apiVersion: v1

kind: Service

metadata:

  name: mysql

  namespace: dev3

spec:

  selector:

    app: mysql

  ports:

    - port: 3306

  clusterIP: None # This means we are going to use Pod IP

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

apiVersion: apps/v1

kind: Deployment

metadata:

  name: usermgmt-microservice

  labels:

    app: usermgmt-restapp

  namespace: dev3

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

--- 07-UserManagement-Service.yml

apiVersion: v1

kind: Service

metadata:

  name: usermgmt-restapp-service

  labels:

    app: usermgmt-restapp

  namespace: dev3

spec:

  type: NodePort

  selector:

    app: usermgmt-restapp

  ports:

    - port: 8095

      targetPort: 8095

      #nodePort: 31231

--- 08-Kubernetes-Secrets.yml

apiVersion: v1

kind: Secret

metadata:

  name: mysql-db-password

  namespace: dev3

type: Opaque

data:

  db-password: ZGJwYXNzd29yZDEx

**Update all k8s manifest with namespace**

--- **note** - Update all files from 02 to 08 with namespace: dev3 in top metadata section in folder kube-manifests/02-Declarative

--- **Example**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: ebs-mysql-pv-claim

  namespace: dev3

**Create k8s objects & Test**

**# Create All Objects**

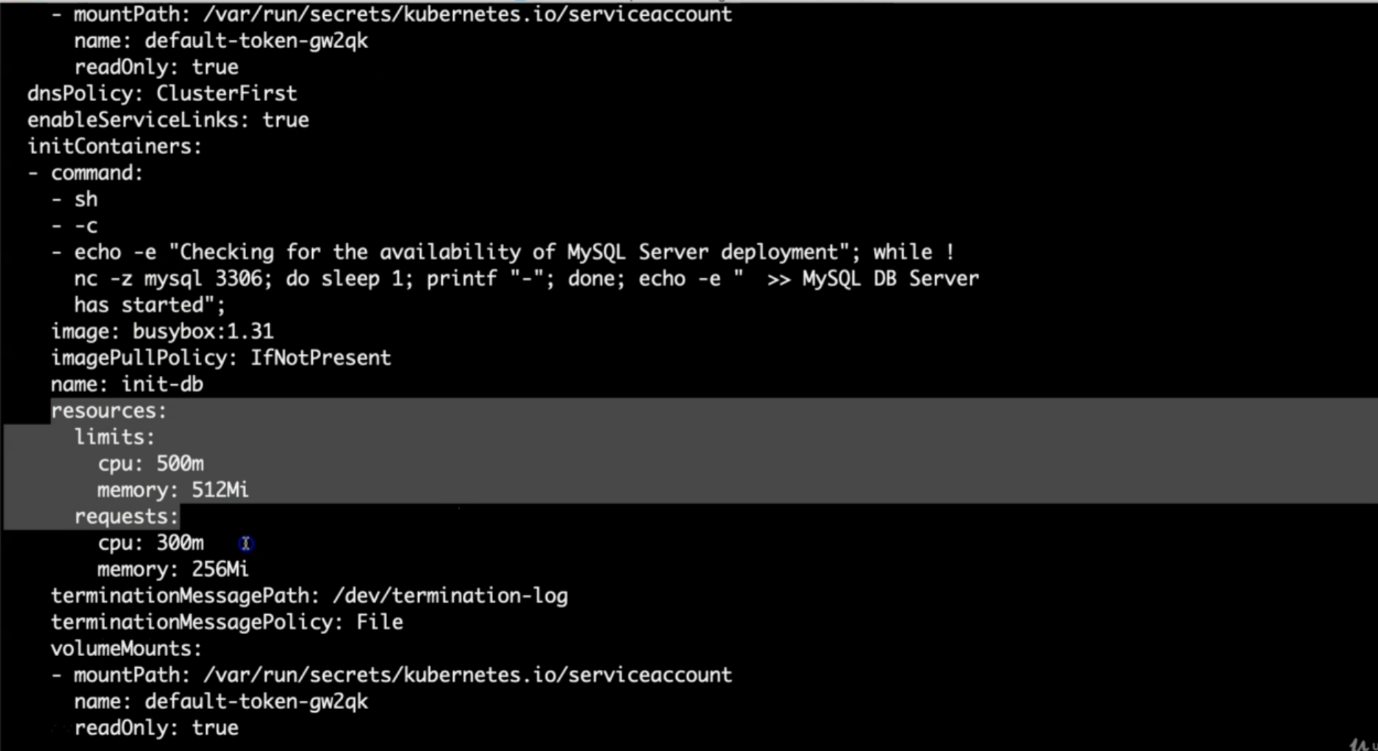
--- **kubectl apply -f kube-manifests/**

**# List Pods**

--- **kubectl get pods -n dev3 -w**

**# View Pod Specification (CPU & Memory)**

--- **kubectl get pod <pod-name> -o yaml -n dev3** – I want this pod yaml output, it is present inside of dev3 namespace.

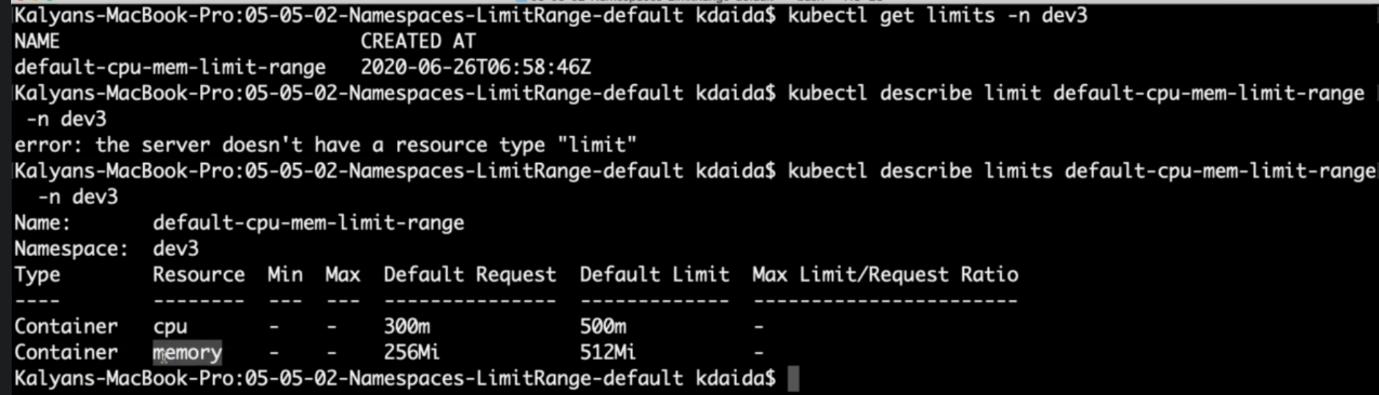


--- note -

**# Get & Describe Limits**

--- **kubectl get limits -n dev3**

--- **kubectl describe limits default-cpu-mem-limit-range -n dev3**



--- **note** – you can see the cpu and memory limit.

**# Get NodePort**

--- **kubectl get svc -n dev3**

**# Get Public IP of a Worker Node**

--- **kubectl get nodes -o wide**

**# Access Application Health Status Page**

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

**Clean-Up**

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

**# Delete All**

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