**Namespaces**

1. Namespaces - Imperative using kubectl
2. Namespaces - Declarative using YAML & LimitRange
3. Namespaces - Declarative using YAML & ResourceQuota

# Kubernetes Namespaces - Imperative using kubectl

# 

# 

## Step-01: Introduction

* Namespaces allow to split-up resources into different groups.
* Resource names should be unique in a namespace
* We can use namespaces to create multiple environments like dev, staging and production etc
* Kubernetes will always list the resources from default namespace unless we provide exclusively from which namespace we need information from.

{:target="\_blank"}

{:target="\_blank"}

{:target="\_blank"}

## Pre-requisite Check (Optional)

* We should already have our AKS Cluster UP and Running.
* We should have configured our AKS Cluster credentials in command line to execute kubectl commands

# Configure AKS Cluster Credentials from command line

az aks get-credentials --name aksdemo1 --resource-group aks-rg1

# List Worker Nodes

kubectl get nodes

kubectl get nodes -o wide

## Step-02: Namespaces Generic - Deploy in Dev1 and Dev2

### Create Namespace

# List Namespaces

kubectl get ns

# Craete Namespace

kubectl create namespace <namespace-name>

kubectl create namespace dev1

kubectl create namespace dev2

# List Namespaces

kubectl get ns

### Deploy All k8s Objects to default, dev1 and dev2 namespaces

# Deploy All k8s Objects

kubectl apply -f kube-manifests/

kubectl apply -f kube-manifests/ -n dev1

kubectl apply -f kube-manifests/ -n dev2

# List all objects from default, dev1 & dev2 Namespaces

kubectl get all -n default

kubectl get all -n dev1

kubectl get all -n dev2

## Step-04: Access Application

### Default Namesapace

# List Services

kubectl get svc

# Access Application

http://<Public-IP-from-List-Services-Output>/app1/index.html

### Dev1 Namespace

# List Services

kubectl get svc -n dev1

# Access Application

http://<Public-IP-from-List-Services-Output>/app1/index.html

### Dev2 Namespace

# List Services

kubectl get svc -n dev2

# Access Application

http://<Public-IP-from-List-Services-Output>/app1/index.html

## Step-05: Clean-Up

# Delete namespaces dev1 & dev2

kubectl delete ns dev1

kubectl delete ns dev2

# List all objects from dev1 & dev2 Namespaces

kubectl get all -n dev1

kubectl get all -n dev2

# List Namespaces

kubectl get ns

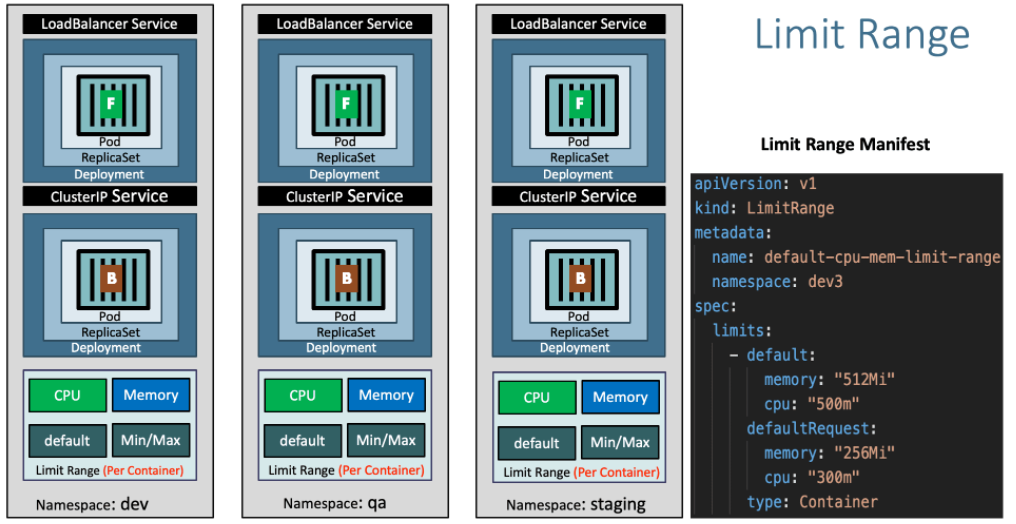
# Delete App from default Namespace (Dont Delete default Namespace - k8s default service exists in it)

kubectl delete -f kube-manifests/

# Get all from All Namespaces

kubectl get all -all-namespaces

# Kubernetes Namespaces - LimitRange - Declarative using YAML



## Pre-requisite Check (Optional)

* We should already have our AKS Cluster UP and Running.
* We should have configured our AKS Cluster credentials in command line to execute kubectl commands

# Configure AKS Cluster Credentials from command line

az aks get-credentials --name aksdemo1 --resource-group aks-rg1

# List Worker Nodes

kubectl get nodes

kubectl get nodes -o wide

## Step-01: Create Namespace manifest

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

apiVersion: v1

kind: Namespace

metadata:

name: dev3

## Step-02: Create LimitRange manifest

* Instead of specifying resources like cpu and memory in every container spec of a pod defintion, 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

## Step-03: Update all k8s manifest with namespace

* Update all files from with namespace: dev3 in top metadata section in folder kube-manifests/
* **Example**

# Deployment Manifest metadata section

apiVersion: apps/v1

kind: Deployment

metadata:

name: app1-nginx-deployment

labels:

app: app1-nginx

namespace: dev3 # Added namespace

spec:

# Service Manifest metadata section

apiVersion: v1

kind: Service

metadata:

name: app1-nginx-clusterip-service

labels:

app: app1-nginx

namespace: dev3 # Added namespace

spec:

## Step-04: Create k8s objects & Test

# Create All Objects

kubectl apply -f kube-manifests/

# List Pods

kubectl get pods -n dev3

# View Pod Specification (CPU & Memory)

kubectl get pod <pod-name> -o yaml -n dev3

# Get & Describe Limits

kubectl get limits -n dev3

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

# List Services

kubectl get svc -n dev3

# Access Application

http://<Public-IP-from-List-Services-Output>/app1/index.html

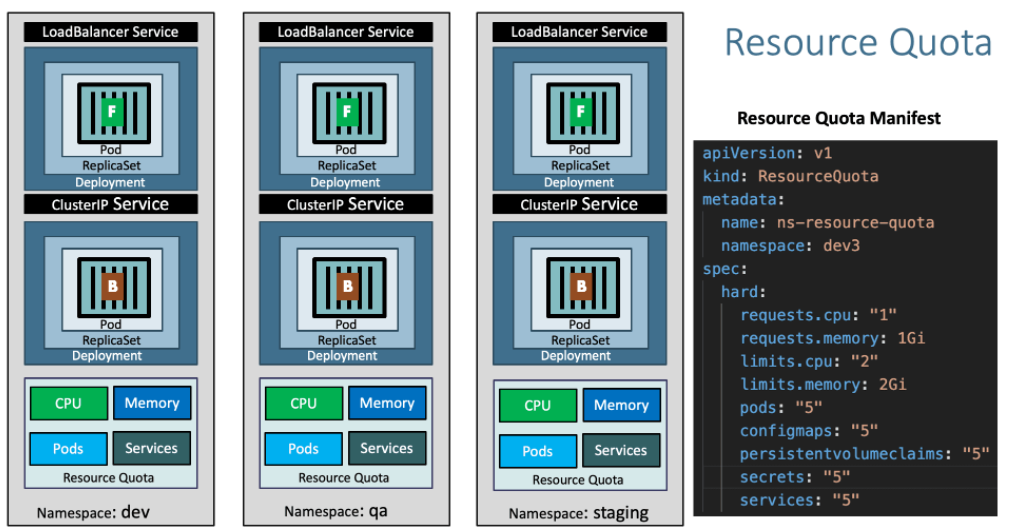
## Step-05: Clean-Up

* Delete all k8s objects created as part of this section

# Delete All

kubectl delete -f kube-manifests/

# Kubernetes Namespaces - ResourceQuota - Declarative using YAML



## Step-01: Create Namespace manifest

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

apiVersion: v1

kind: Namespace

metadata:

name: dev3

## Step-02: Create ResourceQuota manifest

apiVersion: v1

kind: ResourceQuota

metadata:

name: ns-resource-quota

namespace: dev3

spec:

hard:

requests.cpu: "1"

requests.memory: 1Gi

limits.cpu: "2"

limits.memory: 2Gi

pods: "5"

configmaps: "5"

persistentvolumeclaims: "5"

secrets: "5"

services: "5"

## Step-03: Create k8s objects & Test

# Create All Objects

kubectl apply -f kube-manifests/

# List Pods

kubectl get pods -n dev3

# View Pod Specification (CPU & Memory)

kubectl get pod <pod-name> -o yaml -n dev3

# Get & Describe Limits

kubectl get limits -n dev3

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

# Get Resource Quota

kubectl get quota -n dev3

kubectl describe quota ns-resource-quota -n dev3

# List Service

kubectl get svc -n dev3

# Access Application

http://<Public-IP-from-List-Services-Output>/app1/index.html

## Step-04: Clean-Up

* Delete all k8s objects created as part of this section

# Delete All

kubectl delete -f kube-manifests/