**Azure AKS - Cluster Autoscaler**

**Step-01: Introduction**

* The Kubernetes Cluster Autoscaler automatically adjusts the number of nodes in your cluster when pods fail to launch due to lack of resources or when nodes in the cluster are underutilized and their pods can be rescheduled onto other nodes in the cluster.

**Step-02: Create Cluster with Cluster Autoscaler Enabled**

# Setup Environment Variables

export RESOURCE\_GROUP=aks-rg1-autoscaling

export REGION=centralus

export AKS\_CLUSTER=aks-autoscaling-demo

echo $RESOURCE\_GROUP, $REGION, $AKS\_CLUSTER

# Create Resource Group

az group create --location ${REGION} \

--name ${RESOURCE\_GROUP}

# Create AKS cluster and enable the cluster autoscaler

az aks create --resource-group ${RESOURCE\_GROUP} \

--name ${AKS\_CLUSTER} \

--enable-managed-identity \

--generate-ssh-keys \

--node-count 1 \

--enable-cluster-autoscaler \

--min-count 1 \

--max-count 5

# Configure Credentials

az aks get-credentials --name ${AKS\_CLUSTER} --resource-group ${RESOURCE\_GROUP}

# List Nodes

kubectl get nodes

kubectl get nodes -o wide

# Cluster Info

kubectl cluster-info

# kubectl Config Current Context

kubectl config current-context

**Step-03: Compare & Observe aksdemo2 and aks-autoscaling-demo cluster nodepools**

**Cluster: aksdemo2**

* Go to All Services -> Kubernetes Services -> aksdemo2 -> Settings -> Nodepools
* Select Scale
* Compare the setting with aks-autoscaling-demo cluster
* Scale Method: Manual (Observe the setting)
* We can even enable autoscaling from here on Portal Management console

**Cluster: aks-autoscaling-demo**

* Go to All Services -> Kubernetes Services -> aksdemo2 -> Settings -> Nodepools
* Select Scale
* Scale Method: Automatic (Nothing but --enable-cluster-autoscaler)
* Node Count Rage: 1 to 5 (Nothing but what we defined in Min and Max)
* Current Node Count: 1 (Nothing but what we defined in Node Count)

**Step-04: Review & Deploy Sample Application**

# Deploy Application

kubectl apply -f kube-manifests/

# List Pods

kubectl get pods

# Access Application

kubectl get svc

http://<PublicIP-from-get-svc-output>/hello

curl -w "\n" http://52.154.217.196/hello

**Step-05: Scale our application to 20 pods**

* In 2 to 3 minutes, one after the other new nodes will added and pods will be scheduled on them.
* Our max number of nodes will be 5 which we provided during cluster creation.

# Scale UP the demo application to 20 pods

kubectl get pods

kubectl get nodes

kubectl scale --replicas=20 deploy cluster-autoscaler-demoapp-deployment

kubectl get pods

# Verify nodes

kubectl get nodes -o wide

# Access Application

kubectl get svc

http://<PublicIP-from-get-svc-output>/hello

curl -w "\n" http://52.154.217.196/hello

**Step-06: Cluster Scale DOWN: Scale our application to 1 pod**

* It might take 5 to 20 minutes to cool down and come down to minimum nodes which will be 2 which we configured during nodegroup creation

# Scale down the demo application to 1 pod

kubectl scale --replicas=1 deploy cluster-autoscaler-demoapp-deployment

# Verify nodes

kubectl get nodes -o wide

**Step-07: Clean-Up**

* We will leave cluster autoscaler and undeploy only application

# Delete Apps

kubectl delete -f kube-manifests/

AZURE AUTOSCCAILING DEPLOYMENT:-

=====================================

apiVersion: apps/v1

kind: Deployment

metadata:

name: cluster-autoscaler-demoapp-deployment

labels:

app: ca-java-app

spec:

replicas: 1

selector:

matchLabels:

app: ca-java-app

template:

metadata:

labels:

app: ca-java-app

spec:

containers:

- name: ca-java-app

#image: stacksimplify/kubenginx:1.0.0

image: stacksimplify/kube-helloworld:1.0.0

ports:

- containerPort: 8080

resources:

requests:

memory: "200Mi"

cpu: "250m"

limits:

memory: "500Mi"

cpu: "500m"

---

apiVersion: v1

kind: Service

metadata:

name: cluster-autoscaler-demoservice-java-app

labels:

app: ca-java-app

spec:

type: LoadBalancer

selector:

app: ca-java-app

ports:

- port: 80

targetPort: 8080

AZURE Autosaciling Ngininx APP YAML SCRIPT:-

ApiVersion: apps/v1

kind: Deployment

metadata:

name: hpa-demo-deployment

labels:

app: hpa-nginx

spec:

replicas: 1

selector:

matchLabels:

app: hpa-nginx

template:

metadata:

labels:

app: hpa-nginx

spec:

containers:

- name: hpa-nginx

image: stacksimplify/kubenginx:1.0.0

ports:

- containerPort: 80

resources:

requests:

memory: "128Mi"

cpu: "100m"

limits:

memory: "500Mi"

cpu: "200m"

---

apiVersion: v1

kind: Service

metadata:

name: hpa-demo-service-nginx

labels:

app: hpa-nginx

spec:

type: LoadBalancer

selector:

App: hpa-nginx

ports:

- port: 80

targetPort: 80

**Azure Auto-scaling hpa-scale :-**

apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

name: hpa-demo-declarative

spec:

maxReplicas: 10 # define max replica count

minReplicas: 1 # define min replica count

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: hpa-demo-deployment

targetCPUUtilizationPercentage: 20 # target CPU utilization