This article aims to showcase the ways in which one can cut costs by saving on resources like CPU's, RAM and VM's using both inbuilt objects of Kubernetes and third-party tools.

Here are some ways that can help you manage and optimize resources usage.

1. LimitRanges
2. ResourceQuotas
3. Horizontal Pod Autoscaler
4. ClusterAutoScaler
5. Maestro

**LimitRange**

Assigning a memory request and a memory limit or a CPU request and a CPU limit to a container is called a LimitRange. A container can have as much memory as it requests but is not allowed to use more memory than its limit.

Make sure you have the metrics server installed in your cluster before applying LimitRanges. By default, AKS cluster comes with metrics server installed. See the metrics server pod in AKS cluster using  kubectl -n kube-system get pods .

**Create LimitRange**

Create a LimitRange to a namespace so that all the pods that exist in that namespace will get applied by that LimitRange.

1. Create a namespace using kubectl create ns <namespace name>
2. Create and apply the following limitrange to the namespace using kubectl create -f <file path>

apiVersion: v1

kind: LimitRange

metadata:

name: cpu-memory

namespace: <namespace name>

spec:

limits:

- default:

cpu: 250m

memory: 300Mi

defaultRequest:

cpu: 100m

memory: 200Mi

type: Container

**Deploy a Simple Pod and Service**

1. Let's deploy a pod and service that creates a single container to demonstrate how default values are applied to each pod. kubectl -n <namespace name> run php-apache --image=k8s.gcr.io/hpa-example --expose --port=80
2. Get the pods and service using kubectl -n <namespace name> get pods and kubectl -n <namespace name> get services

NAME READY STATUS RESTARTS AGE

php-apache-55c4bb8b88-bb7jp 1/1 Running 0 4m

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

php-apache ClusterIP 10.96.125.210 <none> 80/TCP 4m

**Get the Configuration of The Pod**

Now get the configuration of the pod using kubectl -n <namespace name> get pod <podname> -o yaml

....................

....................

containers:

- image: k8s.gcr.io/hpa-example

imagePullPolicy: Always

name: php-apache

ports:

- containerPort: 80

protocol: TCP

resources:

limits:

cpu: 250m

memory: 300Mi

requests:

cpu: 100m

memory: 200Mi

...................

...................

1. Get the metrics of the pods in your namespace using kubectl top pods -n <namespace name>.

NAME CPU(cores) MEMORY(bytes)

php-apache-55c4bb8b88-bb7jp 1m 9Mi

**Resource Quotas**

In Kubernetes, ResourceQuota is used to limit the resources per namespace when multiple users sharing the cluster.

In this article, we'll apply the ResourceQuota for the compute resources.

**Create a Namespace**

Create a namespace and apply the ResourceQuota to that namespace.

kubectl create ns <namespace name>

**Create a ResourceQuota**

1. Create a ResourceQuota using the following YAML and apply it to the namespace by specifying the namespace name in the YAML file.

apiVersion: v1

kind: ResourceQuota

metadata:

name: <resourcequota name>

namespace: <namespace name>

spec:

hard:

requests.cpu: "1"

requests.memory: 1Gi

limits.cpu: "2"

limits.memory: 2Gi

2. Save the above YAML file and create a ResourceQuota using kubectl create -f <file path>

3. Now use the kubectl -n <namespace name> get resourcequota <resourcequota name> -o yaml      command to get detailed information about the ResourceQuota.

**Create a Pod**

1. Create a Pod in that namespace using the below YAML.

apiVersion: v1

kind: Pod

metadata:

name: nginx-cpu-memory

namespace: <namespace name>

spec:

containers:

- name: nginx-cpu-memory-quota

image: nginx

resources:

limits:

memory: "700Mi"

cpu: "700m"

requests:

memory: "500Mi"

cpu: "300m"

2. Deploy the pod using kubectl create -f <file path>

3. See the container status kubectl -n <namespace name> get pods

4. Get detailed information about the ResourceQuota.

kubectl -n <namespace name> get resourcequota <resuorcequota name> -o yaml

Example:

spec:

hard:

limits.cpu: "2"

limits.memory: 2Gi

requests.cpu: "1"

requests.memory: 1Gi

status:

hard:

limits.cpu: "2"

limits.memory: 2Gi

requests.cpu: "1"

requests.memory: 1Gi

used:

limits.cpu: 700m

limits.memory: 700Mi

requests.cpu: 300m

requests.memory: 500Mi

5. The output shows how much of the quota has been used along with the quota.

**Create Another Pod**

1. Create another pod by specifying the memory request more than its ResuorceQuota request.

apiVersion: v1

kind: Pod

metadata:

name: redis-cpu-memory

namespace: resourcequota

spec:

containers:

- name: redis-cpu-memory-quota

image: redis

resources:

limits:

memory: "1Gi"

cpu: "900m"

requests:

memory: "600Mi"

cpu: "500m"

2. Create the pod using kubectl create -f <file path>

3. Now the second pod doesn't get created and gives an error because of the exceeding memory request.

You can also restrict the totals for memory limit, CPU request, and CPU limit.

**Horizontal Pod Autoscaler**

Horizontal Pod Autoscaler increases the number of pods in a deployment, replica controller or replica set automatically based on the CPU utilization.

**Apply Horizontal Pod Autoscaling**

1. Apply the HPA configuration to the existing deployment.

kubectl -n <namespace name> autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10

1. Get the HPA configuration using kubectl -n <namespace name> get hpa

NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE

php-apache Deployment/php-apache 1%/50% 1 10 1 10m

**Generate Load**

1. Now we will use a load generator to generate some load on Apache.
2. Open an additional terminal window and run the below command.

kubectl -n <namespace name> run -i --tty load-generator --image=busybox /bin/sh

1. Hit enter and run below command to generate load on Apache.

while true; do wget -q -O- http://php-apache.<namespace name>.svc.cluster.local; done

Output should be:

OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!

1. Within few minutes, we should see the higher CPU load by executing kubectl -n <namespace name> get hpa

NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE

php-apache Deployment/php-apache 158%/50% 1 10 4 16m

1. You can see the number of apache pods increased due to load on apache using kubectl -n <namespace name> get pods

NAME READY STATUS RESTARTS AGE

load-generator-5ff6784f85-7wgnm 1/1 Running 0 9m

php-apache-55c4bb8b88-2j2nl 1/1 Running 0 7m

php-apache-55c4bb8b88-bp5mf 1/1 Running 0 5m

php-apache-55c4bb8b88-jx5qr 1/1 Running 0 7m

php-apache-55c4bb8b88-kc68r 1/1 Running 0 52m

php-apache-55c4bb8b88-m4hzb 1/1 Running 0 5m

php-apache-55c4bb8b88-trvrm 1/1 Running 0 7m

**Stop Load**

1. You can stop the load on apache by typing "Ctrl + C" on the new terminal.
2. You can verify the result within a minute using kubectl -n <namespace name> get hpa.

NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE

php-apache Deployment/php-apache 44%/50% 1 10 6 24m

**Cluster Autoscaler on Azure**

**Cluster Autoscaler:**

1. Cluster Autoscaler is an element that automatically by itself ( no direct human control) adjusts the size of the Kubernetes cluster so that all pods have a place to run when there are no redundant nodes. Cluster Autoscaler works with GCP, Azure, and AWS.

Cluster Autoscaler on Azure dynamically scales Kubernetes worker nodes.

**Pre-Requirements:**

1. Managed Kubernetes Service(AKS). Kubernetes version v1.10.x or later should be required.

2. Cluster Autoscaler v1.2.x or later should be required. Make sure that needs to replace a placeholder, '{{ca\_version}}' in a manifest file with CA version such as v.1.2.2.

3. Get the Azure credentials by using the following Azure CLI/Powershell command. Make sure that replace **<subscription-id>** according to yours.

az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/<subscription-id>" --output json

**Steps to Deployment Manifests:**

1. Make sure that get the credentials from the above step.

2. Get the cluster name by using the following command.

az aks list

3. Get the node pool name extracted from the label agent pool from the following command.

kubectl get nodes --show-labels

4. Deploy the clusterautoscaler.yml file mentioned below, make sure that mention all secrets in base 64.

5. Encode each data by using the website <https://www.base64encode.org/>

6. Fill up the placeholder values of cluster-autoscaler-azure under **kind: Secret**data by base64 -encoding each Azure credentials.

    1. ClientID: <base64-encoded-client-id>  
    2. ClientSecret: <base64-encoded-client-secret>  
    3. ResourceGroup: <base64-encoded-resource-group> (Note: ResourceGroup is case-sensitive)  
    4. SubscriptionID: <base64-encoded-subscription-id>  
    5. TenantID: <base64-encoded-tenant-id>  
    6.ClusterName: <base64-encoded-clustername>

Note: Use the following command such as the  echo $CLIENT\_ID |  base64 to encode each of the field above.

7. In the **kind:Deployment** section under spec image name field should be replace  {{ ca\_version }} with the v1.14.2.

8. In the **kind:Deployment section** under spec: command: section update the                                                 --nodes=3:10:nodepool-1 , this will references to the node limits and node pool name.

Example: if  it '--nodes=3:10:nodepool-1' , should scale from 3 to 10 nodes.

---

apiVersion: v1

kind: ServiceAccount

metadata:

labels:

k8s-addon: cluster-autoscaler.addons.k8s.io

k8s-app: cluster-autoscaler

name: cluster-autoscaler

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: cluster-autoscaler

labels:

k8s-addon: cluster-autoscaler.addons.k8s.io

k8s-app: cluster-autoscaler

rules:

- apiGroups: [""]

resources: ["events", "endpoints"]

verbs: ["create", "patch"]

- apiGroups: [""]

resources: ["pods/eviction"]

verbs: ["create"]

- apiGroups: [""]

resources: ["pods/status"]

verbs: ["update"]

- apiGroups: [""]

resources: ["endpoints"]

resourceNames: ["cluster-autoscaler"]

verbs: ["get", "update"]

- apiGroups: [""]

resources: ["nodes"]

verbs: ["watch", "list", "get", "update"]

- apiGroups: [""]

resources:

- "pods"

- "services"

- "replicationcontrollers"

- "persistentvolumeclaims"

- "persistentvolumes"

verbs: ["watch", "list", "get"]

- apiGroups: ["extensions"]

resources: ["replicasets", "daemonsets"]

verbs: ["watch", "list", "get"]

- apiGroups: ["policy"]

resources: ["poddisruptionbudgets"]

verbs: ["watch", "list"]

- apiGroups: ["apps"]

resources: ["statefulsets", "replicasets", "daemonsets"]

verbs: ["watch", "list", "get"]

- apiGroups: ["storage.k8s.io"]

resources: ["storageclasses"]

verbs: ["get", "list", "watch"]

- apiGroups: ["batch"]

resources: ["jobs", "cronjobs"]

verbs: ["watch", "list", "get"]

---

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

name: cluster-autoscaler

namespace: kube-system

labels:

k8s-addon: cluster-autoscaler.addons.k8s.io

k8s-app: cluster-autoscaler

rules:

- apiGroups: [""]

resources: ["configmaps"]

verbs: ["create","list","watch"]

- apiGroups: [""]

resources: ["configmaps"]

resourceNames:

- "cluster-autoscaler-status"

- "cluster-autoscaler-priority-expander"

verbs: ["delete", "get", "update", "watch"]

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: cluster-autoscaler

labels:

k8s-addon: cluster-autoscaler.addons.k8s.io

k8s-app: cluster-autoscaler

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-autoscaler

subjects:

- kind: ServiceAccount

name: cluster-autoscaler

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: cluster-autoscaler

namespace: kube-system

labels:

k8s-addon: cluster-autoscaler.addons.k8s.io

k8s-app: cluster-autoscaler

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: cluster-autoscaler

subjects:

- kind: ServiceAccount

name: cluster-autoscaler

namespace: kube-system

---

apiVersion: v1

data:

ClientID: <base64-encoded-client-id>

ClientSecret: <base64-encoded-client-secret>

ResourceGroup: <base64-encoded-resource-group>

SubscriptionID: <base64-encode-subscription-id>

TenantID: <base64-encoded-tenant-id>

VMType: QUtTCg==

ClusterName: <base64-encoded-clustername>

NodeResourceGroup: <base64-encoded-node-resource-group>

kind: Secret

metadata:

name: cluster-autoscaler-azure

namespace: kube-system

---

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

labels:

app: cluster-autoscaler

name: cluster-autoscaler

namespace: kube-system

spec:

replicas: 1

selector:

matchLabels:

app: cluster-autoscaler

template:

metadata:

labels:

app: cluster-autoscaler

spec:

serviceAccountName: cluster-autoscaler

containers:

- image: k8s.gcr.io/cluster-autoscaler:{{ ca\_version }}

imagePullPolicy: Always

name: cluster-autoscaler

resources:

limits:

cpu: 100m

memory: 300Mi

requests:

cpu: 100m

memory: 300Mi

command:

- ./cluster-autoscaler

- --v=3

- --logtostderr=true

- --cloud-provider=azure

- --skip-nodes-with-local-storage=false

- --nodes=3:10:nodepool1

env:

- name: ARM\_SUBSCRIPTION\_ID

valueFrom:

secretKeyRef:

key: SubscriptionID

name: cluster-autoscaler-azure

- name: ARM\_RESOURCE\_GROUP

valueFrom:

secretKeyRef:

key: ResourceGroup

name: cluster-autoscaler-azure

- name: ARM\_TENANT\_ID

valueFrom:

secretKeyRef:

key: TenantID

name: cluster-autoscaler-azure

- name: ARM\_CLIENT\_ID

valueFrom:

secretKeyRef:

key: ClientID

name: cluster-autoscaler-azure

- name: ARM\_CLIENT\_SECRET

valueFrom:

secretKeyRef:

key: ClientSecret

name: cluster-autoscaler-azure

- name: ARM\_VM\_TYPE

valueFrom:

secretKeyRef:

key: VMType

name: cluster-autoscaler-azure

- name: AZURE\_CLUSTER\_NAME

valueFrom:

secretKeyRef:

key: ClusterName

name: cluster-autoscaler-azure

- name: AZURE\_NODE\_RESOURCE\_GROUP

valueFrom:

secretKeyRef:

key: NodeResourceGroup

name: cluster-autoscaler-azure

restartPolicy: Always

9. After everything is set up, deploy the cluster-autoscaler.yml file into the namespace.

kubectl -n <namespacename> create -f cluster-autoscaler.yaml

**scale up the nodes:**

**Get the nodes in a cluster:**

    To get the nodes in the cluster.

kubectl get nodes

NAME STATUS ROLES AGE VERSION

aks- nodepooll- 33641294-0 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-1 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-2 Ready agent 19h v1.12.7

1. Add more replicas to your deployment like (e.g. 200) in your deployment manifest file, the Auto scaler will add a node in the cluster based on that request coming from the unschedulable pods.

2. Use the following command to add more replicas to the deployments.

kubectl -n kube-system scale --replicas=200 deployment cluster-autoscaler

3. Pending pods will schedule to new nodes, this will be managed by a kubernetes scheduler. verify it by using below output a new node is added

kubectl get nodes

NAME STATUS ROLES AGE VERSION

aks- nodepooll- 33641294-0 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-1 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-2 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-3 Ready agent 10m v1.12.7

4. Follow the autoscaler pod streaming logs, by using the following command.if you want to save logs in your machine as a file add > cas.log end of the command.

kubectl -n kube-system logs -f <autoscaler-pod> > cas.log

1 utils.go:456] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop

I0508 07:51:29.373609 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-6n64d is unschedulable

I0508 07:51:29.373801 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-hq5ts is unschedulable

I0508 07:51:29.376782 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-stq9x is unschedulable

I0508 07:51:29.379030 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-25f46 is unschedulable

I0508 07:51:29.379184 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-td9k5 is unschedulable

I0508 07:51:29.379291 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-r8kpw is unschedulable

I0508 07:51:29.379383 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-nd5kj is unschedulable

I0508 07:51:29.379414 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-jc6hr is unschedulable

I0508 07:51:29.379468 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-86928 is unschedulable

I0508 07:51:29.379595 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-cflg6 is unschedulable

I0508 07:51:29.379627 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-csk2g is unschedulable

I0508 07:51:29.379643 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-sh7jt is unschedulable

I0508 07:51:29.379682 1 scale\_up.go:59] Pod tl-int/tlui-6b495c88bb-tv7jm is unschedulable

I0508 07:51:29.839722 1 scale\_up.go:199] Best option to resize: nodepool1

I0508 07:51:29.842411 1 scale\_up.go:203] Estimated 1 nodes needed in nodepool1

I0508 07:51:29.988130 1 scale\_up.go:292] Final scale-up plan: [{nodepool1 3->4 (max: 5)}]

I0508 07:51:29.988176 1 scale\_up.go:344] Scale-up: setting group nodepool1 size to 4

I0508 07:51:30.216118 1 azure\_container\_service\_pool.go:206] Set size request: 4

I0508 07:51:30.301969 1 azure\_container\_service\_pool.go:241] Current size: 3, Target size requested: 4

I0508 07:54:09.484267 1 azure\_container\_service\_pool.go:276] Target size set done, AKS. Value: {Response:{Response:0xc421703050} ID:0xc422794db0 Name:0xc422794de0 Type:0xc422794e00 Location:0xc422794e20 Tags:0xc4218b0918 ManagedClusterProperties:0xc42296c300}

I0508 07:54:09.484320 1 azure\_container\_service\_pool.go:299] Got Updated value. Time taken: 2m39.182059589s

I0508 07:54:19.718531 1 azure\_manager.go:261] Refreshed ASG list, next refresh after 2019-05-08 07:55:19.718500245 +0000 UTC

**scale down the node:**

1. If the pod is part of a daemonset, the pod is safe to turn down, since daemonsets are supposed to run statelessly on all nodes. Removing the node should not reschedule a pod in a daemonset.

2. If the pod is a mirror pod, (only relevant if you’ve created static pods), it is considered safe to turn down. Removing the pod does not bring the number of replicas below the specified minimum replica count unless you have specified a pod disruption budget and have remaining disruptions to “spend” on moving the pod.

kubectl -n kube-system scale --replicas=1 deployment cluster-autoscaler

3. The pod doesn’t use any local storage on the node; since the node is going away, that local storage will be lost. Kube system pods won’t get moved unless they specify a pod disruption budget.

4. If any of these node or pod-level checks do not pass, then the node will not be turned down.

0508 08:08:21.751263 1 scale\_down.go:175] Scale-down calculation: ignoring 2 nodes, that were unremovable in the last 5m0s

I0508 08:08:22.241577 1 scale\_down.go:387] aks-nodepool1-33641294-3 was unneeded for 8m59.269539089s

I0508 08:08:22.241618 1 scale\_down.go:446] No candidates for scale down

I0508 08:08:32.614138 1 utils.go:456] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop

I0508 08:08:32.615193 1 static\_autoscaler.go:280] No unschedulable pods

I0508 08:08:32.914526 1 scale\_down.go:175] Scale-down calculation: ignoring 2 nodes, that were unremovable in the last 5m0s

I0508 08:08:33.614834 1 scale\_down.go:387] aks-nodepool1-33641294-3 was unneeded for 9m10.352165882s

I0508 08:08:33.615111 1 scale\_down.go:446] No candidates for scale down

I0508 08:08:44.022848 1 utils.go:456] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop

I0508 08:08:44.023171 1 static\_autoscaler.go:280] No unschedulable pods

I0508 08:08:44.285715 1 scale\_down.go:175] Scale-down calculation: ignoring 2 nodes, that were unremovable in the last 5m0s

I0508 08:08:44.975568 1 scale\_down.go:387] aks-nodepool1-33641294-3 was unneeded for 9m21.712198902s

I0508 08:08:44.976314 1 scale\_down.go:446] No candidates for scale down

I0508 08:08:55.223622 1 utils.go:456] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop

I0508 08:08:55.224447 1 static\_autoscaler.go:280] No unschedulable pods

I0508 08:08:55.438989 1 scale\_down.go:175] Scale-down calculation: ignoring 2 nodes, that were unremovable in the last 5m0s

I0508 08:08:55.870289 1 scale\_down.go:387] aks-nodepool1-33641294-3 was unneeded for 9m33.091331289s

I0508 08:08:55.870312 1 scale\_down.go:446] No candidates for scale down

I0508 08:09:05.914804 1 azure\_manager.go:261] Refreshed ASG list, next refresh after 2019-05-08 08:10:05.914788083 +0000 UTC

I0508 08:09:06.226728 1 utils.go:456] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop

I0508 08:09:06.227670 1 static\_autoscaler.go:280] No unschedulable pods

I0508 08:09:06.315602 1 scale\_down.go:175] Scale-down calculation: ignoring 2 nodes, that were unremovable in the last 5m0s

I0508 08:09:06.566767 1 scale\_down.go:387] aks-nodepool1-33641294-3 was unneeded for 9m43.99496054s

I0508 08:09:06.566811 1 scale\_down.go:446] No candidates for scale down

I0508 08:09:16.906486 1 utils.go:456] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop

I0508 08:09:16.906894 1 static\_autoscaler.go:280] No unschedulable pods

I0508 08:09:17.079928 1 scale\_down.go:175] Scale-down calculation: ignoring 2 nodes, that were unremovable in the last 5m0s

I0508 08:09:17.442731 1 scale\_down.go:387] aks-nodepool1-33641294-3 was unneeded for 9m54.746647319s

I0508 08:09:17.442764 1 scale\_down.go:446] No candidates for scale down

I0508 08:09:27.722914 1 utils.go:456] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop

I0508 08:09:27.727164 1 static\_autoscaler.go:280] No unschedulable pods

I0508 08:09:27.903046 1 scale\_down.go:175] Scale-down calculation: ignoring 2 nodes, that were unremovable in the last 5m0s

I0508 08:09:28.253140 1 scale\_down.go:387] aks-nodepool1-33641294-3 was unneeded for 10m5.538079033s

I0508 08:09:28.415763 1 scale\_down.go:594] Scale-down: removing empty node aks-nodepool1-33641294-3

I0508 08:09:28.492595 1 delete.go:53] Successfully added toBeDeletedTaint on node aks-nodepool1-33641294-3

I0508 08:09:28.496485 1 azure\_container\_service\_pool.go:360] Node: azure:///subscriptions/7aa98dd2-d24a-476c-8cca-c2febfd47d51/resourceGroups/MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/providers/Microsoft.Compute/virtualMachines/aks-nodepool1-33641294-3

I0508 08:09:28.496738 1 azure\_container\_service\_pool.go:364] ProviderID before calling acsmgr: azure:///subscriptions/7aa98dd2-d24a-476c-8cca-c2febfd47d51/resourceGroups/MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/providers/Microsoft.Compute/virtualMachines/aks-nodepool1-33641294-3

I0508 08:09:28.653818 1 azure\_container\_service\_pool.go:333] ProviderID got to delete: azure:///subscriptions/7aa98dd2-d24a-476c-8cca-c2febfd47d51/resourceGroups/MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/providers/Microsoft.Compute/virtualMachines/aks-nodepool1-33641294-3

I0508 08:09:28.704819 1 azure\_container\_service\_pool.go:338] VM name got to delete: aks-nodepool1-33641294-3

I0508 08:09:28.755209 1 azure\_util.go:144] found nic name for VM (MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/aks-nodepool1-33641294-3): aks-nodepool1-33641294-nic-3

I0508 08:09:28.755232 1 azure\_util.go:147] deleting VM: MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/aks-nodepool1-33641294-3

I0508 08:09:28.755239 1 azure\_util.go:151] waiting for VirtualMachine deletion: MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/aks-nodepool1-33641294-3

I0508 08:11:14.839773 1 azure\_util.go:157] VirtualMachine MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/aks-nodepool1-33641294-3 removed

I0508 08:11:14.839803 1 azure\_util.go:160] deleting nic: MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/aks-nodepool1-33641294-nic-3

I0508 08:11:14.848795 1 azure\_util.go:162] waiting for nic deletion: MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/aks-nodepool1-33641294-nic-3

I0508 08:11:35.411503 1 azure\_util.go:192] deleting managed disk: MC\_AKS19ClusterRG\_AKS19Cluster\_eastus/aks-nodepool1-33641294-3\_OsDisk\_1\_0cfd53b492b14e669ab874c58361c42d

**Get the nodes:**

1. Verify it by using the following command, to check the status of the node, which is down. It will shows `NotReady' state after some time it will delete automatically.

kubectl get nodes

NAME STATUS ROLES AGE VERSION

aks- nodepooll- 33641294-0 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-1 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-2 Ready agent 19h v1.12.7

aks- nodepooll- 33641294-3 NotReady agent 30m v1.12.7