Maximizing AKS Potential

Agenda:

1) K8s Cluster Autoscaler (CA)
2) Scaling AKS Nodes - HPA, VPA
3) KEDA
4) NODE AUTOPROVISIONING (Karp)
5) Cost effective & comp

Maheshkumar R
Cloud Solution Architect, Microsoft

Scaling AKS Nodes & best practices

1. Resource Quotas
2. Taints and Tolerations
3. Selectors & Affinity
4. Topology spread
5. Disk limitations

Manual scaling

$ az aks show --resource-group simple-aks-rg --name simpleaks-cluster1 --query agentPoolProfiles

$ az aks scale --resource-group simple-aks-rg --name simpleaks --node-count 1 --nodepool-name <your node pool name>

$ az aks nodepool scale --name apppoolone --cluster-name simpleaks --resource-group simple-aks-rg --node-count 0

Cluster Autoscaler (CA)

1. AKS CA built on VMSS – will add/remove nodes when needed
2. CA used by HPA, VPA, KEDA and other autoscaler

Key points,

1) When PODS failed to run due to insufficient resources – Pods pending
2) Underutilized nodes where the pods can be rescheduled
3) Can scale to zero nodes on AKS
4) Nodes are created from the same OS base image

Cluster Autoscaler demo

$ k config get-contexts

$ k get cm -n kube-system cluster-autoscaler-status -o yaml

$ k get events -A --field-selector source=cluster-autoscaler -w

$ k create namespace inflatens

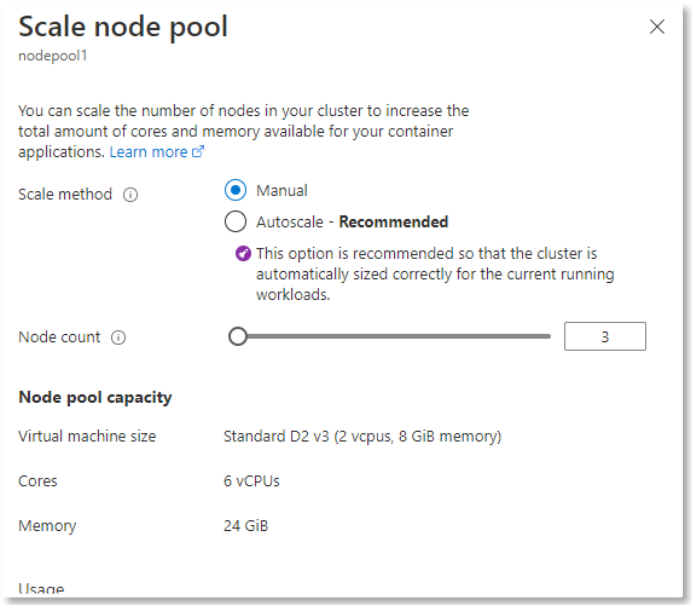
$ k apply -f inflate.yaml

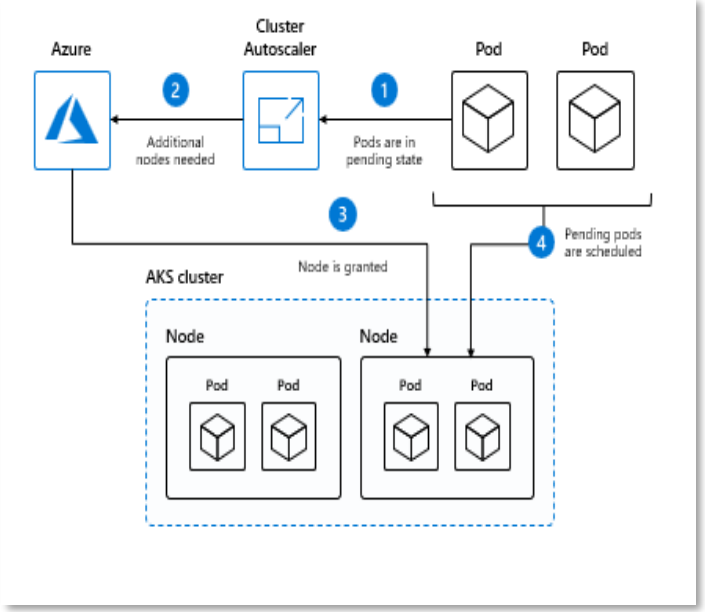
$ k get all -n inflatens

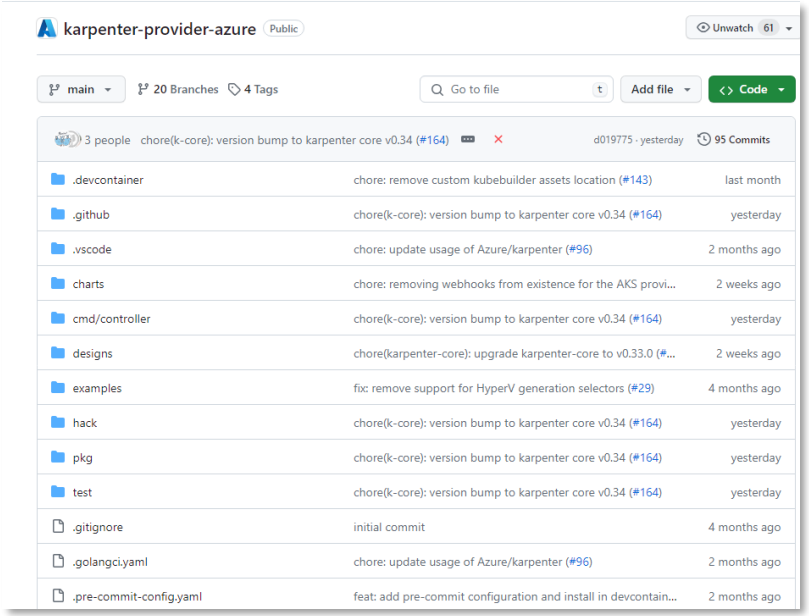
$ k get nodes -o json | jq '.items[] | {name: .metadata.name, instance_type: .metadata.labels["node.kubernetes.io/instance-type"], nodepool_type: .metadata.labels["kubernetes.azure.com/nodepool-type"], topology_spread: .metadata.labels["topology.kubernetes.io/zone"], image_version: .metadata.labels["kubernetes.azure.com/node-image-version"], }'

$ k scale deployment/inflate --replicas=10 -n inflatens
watch kubectl get po -o wide -n inflatens
kubectl scale deployment/inflate --replicas=0 -n inflatens

Node scaling







Manual Scaling

- Manually scale up the AKS cluster Nodepool for **anticipated application load** or deployments
- Set --scale-down-mode Deallocate to reduce cost and speed up scale up times.Deallocate nodes rather than deleting them

Cluster Autoscaler

- Responds to unscheduled "*pending*" pod load and **automatically scales** up the nodes in response
- Often used with HPA

Node Auto Provisioning

- **Efficient Resource Utilization**: NAP determines the optimal VM configuration based on pending pod resource requirements.
- **Automated Decision-Making**: NAP selects the most suitable VMs to run your workloads, reducing the manual effort required for VM configuration design and **operational cost**

Comparison of costs on the scalers

Workload taken uniformly

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: middletier
spec:
  replicas: 5
  selector:
    matchLabels:
      app: middleTier
  template:
    metadata:
      labels:
        app: middleTier
    spec:
      containers:
        - name: stress
          image: wdhif/stress-ng
          args:
            ["--aggressive", "--cpu", "1", "--vm", "1", "--vm-bytes"
          resources:
            requests:
              memory: "1000Mi"
              cpu: "2"
            limits:
              memory: "1000Mi"
              cpu: "2"
```

Middle Tier – 5 replicas

Requests/Limits:

CPU : 10cores

Mem: 1000mi

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 30
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: stress
          image: wdhif/stress-ng
          args:
            ["--aggressive", "--cpu", "1", "--vm", "1", "--vm-byt
          resources:
            requests:
              memory: "200Mi"
              cpu: "0.5"
            limits:
              memory: "200Mi"
              cpu: "0.5"
```

Frontend – 30 replicas

Requests/Limits:

CPU : 15cores

Mem: 200mi

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 10
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: stress
          image: wdhif/stress-ng
          args:
            ["--aggressive", "--cpu", "2", "--vm", "2", "--vm-by
          resources:
            requests:
              memory: "2000Mi"
              cpu: "2"
            limits:
              memory: "2000Mi"
              cpu: "2"
```

Backend – 10 replicas

Requests/Limits:

CPU : 20cores

Mem: 2000mi

Major savings with NAP

	# of nodes	SKU type	Cost / month	% cost reduction
Manual Scaling	20	Standard DS3 v2	$4,176	
Cluster Autoscaler	16	Standard DS3 v2	$3,341	80% of MS
Node Autoprovision	**2**	**Standard_D16ls_v5 Standard_D32ls_v5**	**$1,469**	**35% of MS 44% of CAS**

- NAP approx. **35%** the price of manual scaling per month
- NAP approx. **44%** the price of CAS per month

AKS Node Provisioning (NAP)

 # STEPS FOR ENABLING AKS NODE AUTO-PROVISIONING

```
$ az login --use-device-code
$ az extension add --name aks-preview
$ az extension update --name aks-preview
$ az feature register --namespace "Microsoft.ContainerService" --name "NodeAutoProvisioningPreview"
$ az feature show --namespace "Microsoft.ContainerService" --name "NodeAutoProvisioningPreview"
$ az provider register --namespace Microsoft.ContainerService
$ alias k=kubectl

$ az group create --name aks-nap-rg --location eastus

$ az aks create --name ga-nap-aks --resource-group aks-nap-rg --node-provisioning-mode Auto --network-plugin azure --network-plugin-mode overlay --network-dataplane cilium --
nodepool-taints CriticalAddonsOnly=true:NoSchedule

$ k get nodes -o json | jq '.items[] | {name: .metadata.name, instance_type: .metadata.labels["node.kubernetes.io/instance-type"], nodepool_type:
.metadata.labels["kubernetes.azure.com/nodepool-type"], karpeneter_nodepool: .metadata.labels ["karpenter.sh/nodedpool"],
topology_spread: .metadata.labels["topology.kubernetes.io/zone"], karpeneter_nodepool: .metadata.labels ["karpenter.sh/capacity-type"],image_version:
.metadata.labels["kubernetes.azure.com/node-image-version"],  }'


$ k get no -o wide

$ k get NodePool default -o yaml
$ k edit aksnodeclass default

# scale/generate load test to see the karpenter events
$ k scale deploy -n global-azure-ns --replicas=20 --all

# keep this in separate window to see it in action
$ k get events -A --field-selector source=karpenter -w

#additional commands
$ kubectl config get-contexts
```

NAP Demo

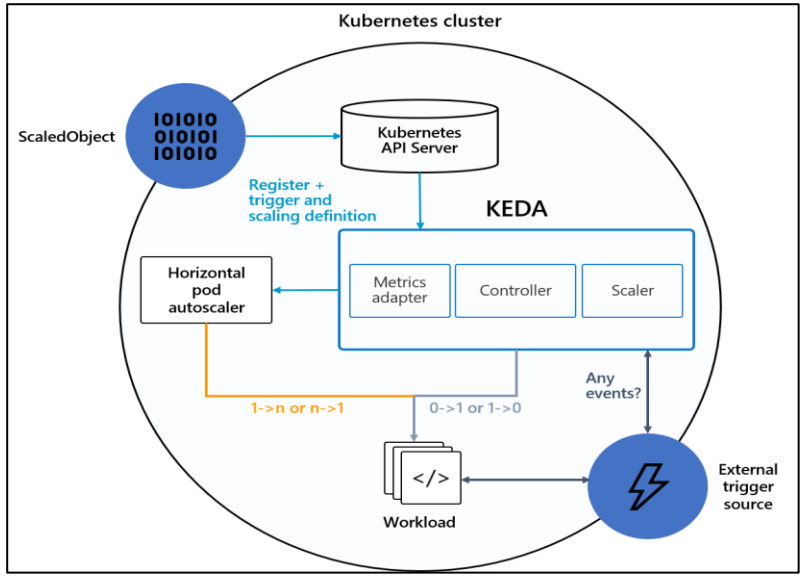
```
root@DESKTOP-OOHAONV:/mnt/d# kubectl get nodes -o json | jq '.items[] | {name: .metadata.name, instance_type: .metadata.labels["node.kubernetes.io/instance-type"], nodepool_type: .metadata.labels["kubernetes.azure.com/nodepool-type"], karpeneter_nodepool: .metadata.labels ["karpenter.sh/nodedpool"], topology_spread: .metadata.labels["topology.kubernetes.io/zone"], karpeneter_nodepool: .metadata.labels ["karpenter.sh/capacity-type"],image_version: .metadata.labels["kubernetes.azure.com/node-image-version"],  }'
{
  "name": "aks-default-2pp2t",
  "instance_type": "Standard_D2ls_v5",
  "nodepool_type": null,
  "karpeneter_nodepool": "on-demand",
  "topology_spread": "eastus-2",
  "image_version": null
}
{
  "name": "aks-nodepool1-37686886-vmss000000",
  "instance_type": "Standard_DS2_v2",
  "nodepool_type": "VirtualMachineScaleSets",
  "karpeneter_nodepool": null,
  "topology_spread": "0",
  "image_version": "AKSUbuntu-2204gen2containerd-202404.01.0"
}
{
  "name": "aks-nodepool1-37686886-vmss000001",
  "instance_type": "Standard_DS2_v2",
  "nodepool_type": "VirtualMachineScaleSets",
  "karpeneter_nodepool": null,
  "topology_spread": "0",
  "image_version": "AKSUbuntu-2204gen2containerd-202404.01.0"
}
{
  "name": "aks-nodepool1-37686886-vmss000002",
  "instance_type": "Standard_DS2_v2",
  "nodepool_type": "VirtualMachineScaleSets",
  "karpeneter_nodepool": null,
  "topology_spread": "0",
  "image_version": "AKSUbuntu-2204gen2containerd-202404.01.0"
root@DESKTOP-OOHAONV:/mnt/d#
root@DESKTOP-OOHAONV:/mnt/d#
```

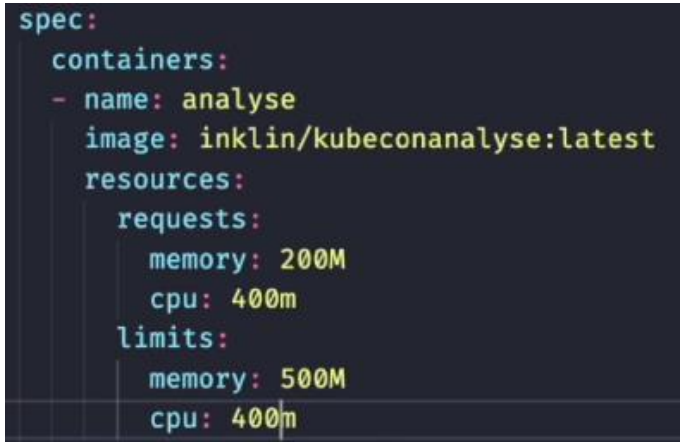
```
root@DESKTOP-OOHAONV:/mnt/c/Users/maheshk# kubectl get events -A --field-selector source=karpenter -w
NAMESPACE          LAST SEEN   TYPE     REASON              OBJECT                                  MESSAGE
default            14m         Normal   DisruptionBlocked   node/aks-default-2pp2t                  Cannot disrupt Node: Nominated for a pending pod
default            14m         Normal   Unconsolidatable    node/aks-default-2pp2t                  Can't replace with a cheaper node
default            14m         Normal   DisruptionBlocked   nodeclaim/default-2pp2t                 Cannot disrupt NodeClaim: Nominated for a pending pod
default            14m         Normal   Unconsolidatable    nodeclaim/default-2pp2t                 Can't replace with a cheaper node
global-azure-ns    16m         Normal   Nominated           pod/azure-vote-back-6c8bb6c944-6hvvq    Pod should schedule on: nodeclaim/default-2pp2t
global-azure-ns    15m         Normal   Nominated           pod/azure-vote-front-546855d499-bm9k6   Pod should schedule on: nodeclaim/default-2pp2t
```

NAP Config

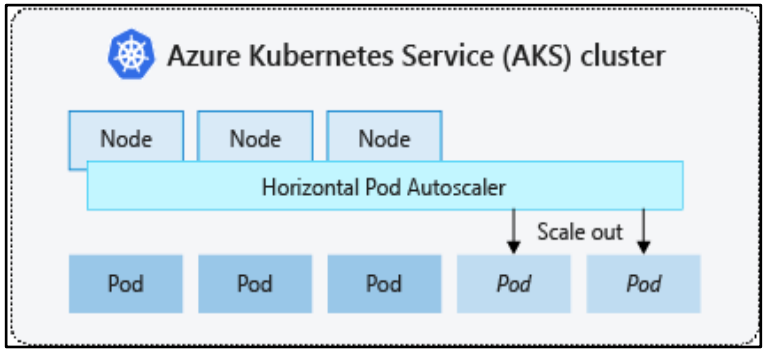
```
root@DESKTOP-OOHAONV:/mnt/d# kubectl get NodePool default -o yaml
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  annotations:
    karpenter.sh/nodepool-hash: "12393960163388511505"
    kubernetes.io/description: General purpose NodePool for generic workloads
    meta.helm.sh/release-name: aks-managed-karpenter-overlay
    meta.helm.sh/release-namespace: kube-system
  creationTimestamp: "2024-04-16T05:57:41Z"
  generation: 1
  labels:
    app.kubernetes.io/managed-by: Helm
    helm.toolkit.fluxcd.io/name: karpenter-overlay-main-adapter-helmrelease
    helm.toolkit.fluxcd.io/namespace: 661e1256ff3ead0001c30a6d
  name: default
  resourceVersion: "4470"
  uid: 6fb150d9-f0eb-4abc-b7e5-93034cf4b693
spec:
  disruption:
    consolidationPolicy: WhenUnderutilized
    expireAfter: Never
  template:
    spec:
      nodeClassRef:
        name: default
      requirements:
      - key: kubernetes.io/arch
        operator: In
        values:
        - amd64
      - key: kubernetes.io/os
        operator: In
        values:
        - linux
      - key: karpenter.sh/capacity-type
        operator: In
        values:
        - on-demand
      - key: karpenter.azure.com/sku-family
        operator: In
        values:
        - D
status:
  resources:
    cpu: "2"
    ephemeral-storage: 129886128Ki
    memory: 4006168Ki
    pods: "250"
root@DESKTOP-OOHAONV:/mnt/d#
```

Application and pod scaling





```
spec:
  containers:
  - name: analyse
    image: inklin/kubeconanalyse:latest
    resources:
      requests:
        memory: 200M
        cpu: 400m
      limits:
        memory: 500M
        cpu: 400m
```



Kubernetes Event Driven Autoscaler (KEDA)

- **Scale based on the number of events** needing to be processed
- Over 50+ scalers that represent what KEDA can be scaled based on. All maintained by either the community or specific organizations
- Save resources (and costs!) by scaling to 0

Vertical Pod Autoscaler (VPA)

- **Dynamically set requests / limits** on containers based on resource usage over time
- Ensure that you are not paying for overprovisioned resources without compromising the stability of the workload

Horizontal Pod Autoscaler (HPA)

- Monitor and autoscale the number of replicas based on Kubernetes metrics
- Only deploy the number of workloads needed **based on resource demand**
- As CPU/memory requirements decrease, so do your replicas and cost!

Compute Best practices - VM SKUs and sizing

AKS support a range of Compute to match the variety of workloads that customers wants to run in Kubernetes.

Spot and Ampere Altra ARM-based Processors

- **Save up to 90% cost** with Spot Instances
- **Ampere Altra ARM-based Processors**
Better price-performance than comparable x86-based virtual machines (VMs)

- Adoption of ARM-based compute instances for containerized workloads has more than doubled across 2022-2023

Capacity Reservations and Azure Reservations

- Reserve compute capacity in an Azure region or an availability zone for any duration of time
- Avoid on demand price spike.
- Azure Reservations operate on a one-year or three-year term, offering up to 72% discount as compared to pay-as-you-go prices for compute.

AKS reduces kube-reserved memory

- Optimized reservation logic reduces Kube-reserved memory by up to 20% depending on the node configuration
- Applies to all clusters

Beginning with AKS support of Kubernetes 1.29 in preview

Microsoft

Question

Demo & Slide deck @ https://github.com/Maheshk-MSFT/AKS_NAP_Karpenter

https://www.linkedin.com/in/mfcmahesh/

https://twitter.com/MahesKBlr