Formula for Calculating AKS Nodepool cpu and memory and ResourceQuota for each Namespace

Advanced Conversational Engagement

Exported on 06/06/2025

Table of Contents

Selecting the right node pool configuration in Azure Kubernetes Service is a balance between understanding your workload's CPU and memory requirements and optimizing for cost and performance. By carefully selecting the appropriate VM SKU, scaling intelligently, and continuously monitoring your cluster, you can ensure that your AKS environment runs efficiently and cost-effectively.

When selection the SKU , we need to consider the default services for azure that runs on every node . services listed below runs on every node -

1. ama logs
2. calico-node
3. azure-cns
4. kube-proxy
5. azure-ip-masq-agent
6. aks-secrets-store-provider-azure
7. aks-secrets-store-csi-driver
8. cloud-node-manager
9. csi-azuredisk-node
10. csi-azurefile-node

Above listed services **reserves** 6 Mebibyte of memory and 2 Vcores of cpu . Choosing the sku for user nodepool should be made with including the above services memory and cpu consumption . the formula for sku we can drive as below

CPU for NodePool SKU : Number of cores * Application pods/num of minimum node + 2 Vcores/cpu's for default services

Memory for NodePool SKU : Number of Memory * Application pods/num of minimum node + 6 GB memory for default services

1 1. CPU Calculation

Breakdown:

1. **Application Pods Calculation:**
   - **Number of cores per pod:** This represents the CPU requirement for each application pod.
   - **Total application cores:** `Number of cores * Application pods . This is as per CPU  limits set in pod/deployment yaml`
   - **Per Node Allocation:** Divide by the `Number of minimum nodes` to distribute the CPU needs across nodes.
2. **Default Services:**
   - **Fixed Allocation:** `+ 2 cores`
   - **Purpose:** Reserved for system or default services that AKS runs on each node (e.g., kubelet, kube-proxy).

Recommendations:

- **Align with AKS Recommendations:** Unless there's a specific need for higher overhead, consider using the **20%** suggested by AKS for Azure services. This can lead to more efficient resource utilization.
- **Autoscaling Considerations:** Implement Cluster Autoscaler to dynamically adjust the number of nodes based on CPU utilization, ensuring cost-efficiency and performance.
- **Buffer for Peak Loads:** If your application experiences variable loads, maintaining a slight buffer beyond the calculated requirements can prevent performance bottlenecks.

2 2. Memory Calculation

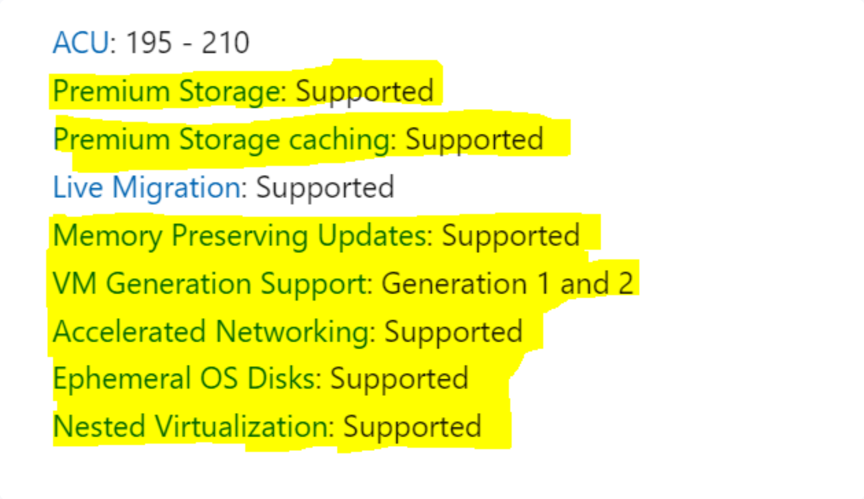
Breakdown:

1. **Application Pods Calculation:**
   - **Memory per pod:** This represents the memory requirement for each application pod.
   - **Total application memory:** `Number of Memory * Application pods . This is as per Memory  limits set in pod/deployment yaml`
   - **Per Node Allocation:** Divide by the `Number of minimum nodes` to distribute the memory needs across nodes.
2. **Default Services:**
   - **Fixed Allocation:** `+ 6 GB`
   - **Purpose:** Reserved for system or default services that AKS runs on each node.

Recommendations:

- **Align with AKS Recommendations:** Similar to CPU, consider using the **20%** overhead for Azure services unless your specific use case requires more.
- **Memory Buffer:** Ensure there's enough memory to handle peak loads and prevent out-of-memory (OOM) issues, which can lead to pod evictions.
- **Monitoring and Scaling:** Utilize monitoring tools like Azure Monitor to track memory usage and adjust node sizes or counts accordingly.

IMPORTANT : While SKU selection cpu and memory is the crucial part of nodePool. please note that always choose the sku which has all the feature enabled like shown below :



ACU: 195 - 210
Premium Storage: Supported
Premium Storage caching: Supported
Live Migration: Supported
Memory Preserving Updates: Supported
VM Generation Support: Generation 1 and 2
Accelerated Networking: Supported
Ephemeral OS Disks: Supported
Nested Virtualization: Supported

Formula To Calculate Resource Quota For Each Namespace:

Maximum Number of Nodes *(Total Size of the Node pool - Resource Utilization by Default Services)

Maximum Number of Nodes is different in DTAP environments based on the Autoscale enabled.

Note: While Calculating Resource quota for Acceptance and Production Environments include capacity of pods running on twist lock system namespace as below

In current Scenario we have 11 twist lock defender pods running in twist lock-system namespace with each pod having limits as
CPU: 900m
Memory: 512Mi