Jonathan

Follow 32 Followers About

Azure Kubernetes Service (AKS) Upgrades

Jonathan Feb 28 · 7 min read

We probably should break the upgrades into 3 parts, Kubernetes version upgrade, node image version upgrade (security patching) and Azure-associated service upgrade.

Kubernetes Version Upgrade

AKS Supported Versions

Kubernetes uses the standard Semantic Versioning scheme, which means it follows the format of

[major].[minor].[patch]

version *1.17.9*

- *1* would be the major version (Gets changed only when incompatible API changes or backward compatibility is broken)
- *17* would be the minor version (Gets releases 9–12 months after *19*)
- *9* being the patch version (Gets releases the fastest as weekly)

AKS supports the latest general available (GA) minor version (N) and 2 minor versions prior to that (N-2). For example, version *1.17.x* is the latest GA minor version, AKS would support

- *1.17.x*

Within each minor version, AKS also supports 2 stable patch versions.

- *1.17.a*
- *1.17.b*
- *1.16.c*
- *1.16.d*
- *1.15.e*
- *1.15.f*

In short, [major].[N-2].[P-2] to [major].[N].[P] would be supported by Azure platform. We could use

```
az aks get-versions -l <location> -o table
```

to get the available version in each region. For more about AKS supported versions and availability, please check here.

Get Available AKS Upgrade Versions

Now, let's take a look at how we could be sure of the running AKS K8s version.

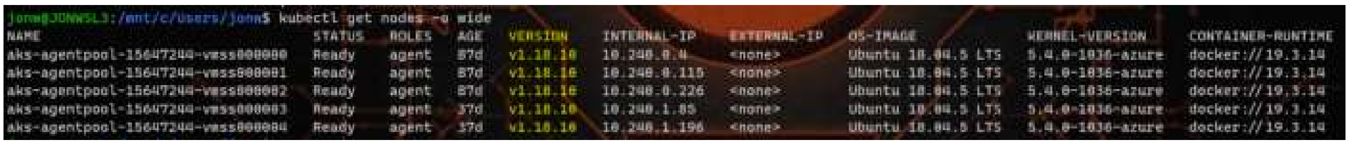
```
az aks show -g <resource group name> -n <AKS cluster name>| grep -E
"orchestratorVersion|kubernetesVersion"
```

*** orchestrator version means the master nodes' version (control plane version)*

*** kubernetes version means the worker nodes' version*

```
kubectl get nodes -o wide
```

*** This command shows only worker nodes' version but it shows all nodes in all node pools*

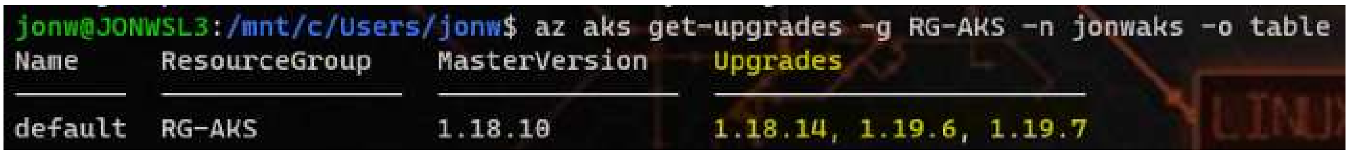


```
jonw@JONWSL3:/mnt/c/Users/jonw$ kubectl get nodes -o wide
NAME                                STATUS   ROLES   AGE   VERSION    INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
aks-agentpool-15647244-vmss000000   Ready    agent   87d   v1.18.10   10.240.0.4     <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure    docker://19.3.14
aks-agentpool-15647244-vmss000001   Ready    agent   87d   v1.18.10   10.240.0.115   <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure    docker://19.3.14
aks-agentpool-15647244-vmss000002   Ready    agent   87d   v1.18.10   10.240.0.226   <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure    docker://19.3.14
aks-agentpool-15647244-vmss000003   Ready    agent   37d   v1.18.10   10.240.1.85    <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure    docker://19.3.14
aks-agentpool-15647244-vmss000004   Ready    agent   37d   v1.18.10   10.240.1.196   <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure    docker://19.3.14
```

Check available upgrades with

```
az aks get-upgrades --resource-group <RG name> --name <Cluster name>
--output table
```

*** Minor version upgrade needs to be performed sequentially. For example, 1.18.x to 1.20.x is not allowed. It needs to be 1.18.x to 1.19.x to 1.20.x.*



```
jonw@JONWSL3:/mnt/c/Users/jonw$ az aks get-upgrades -g RG-AKS -n jonwaks -o table
Name     ResourceGroup    MasterVersion    Upgrades
-------  ---------------  ---------------  ----------------------
default  RG-AKS           1.18.10          1.18.14, 1.19.6, 1.19.7
```

AKS Cluster Upgrade

Before really getting into the cluster upgrade, we could set node surge percentage. With higher value set, the nodes could be upgraded faster but it also means the service hosted by the cluster gets higher possibility of service interruption. For more details, please check this section. If administrators would like to use automatic upgrade instead of manual, this could also be done by setting auto-upgrade channel, please follow this section.

3 important prerequisite checks before heading toward the operation.

- The amount of nodes across node pools — N
- The max amount of Pods on each node — P
- The private IP addresses required N + (N+1)*P.

For example, having the total nodes of 30 across 3 node pools, and each node could host at most 30 Pods, the total private IP addresses required would be

30 + (31*30) = 961

If we use CIDR to represent this number, it would be x.x.x.x/22. You could use any online tool to find this information.

** Please make sure there is no Pod Disruption Budget (PDB) that would be potentially blocking AKS cluster upgrade process. Essentially, "maxUnavailable" and "minAvailable" would need to be set according to your workload. For example, if you have only 1 Pod hosting the application, the option is either not set a PDB or set "maxUnavailable" at "1" and plan the application down time with the application team. For more information, please check this section of the official documentation.

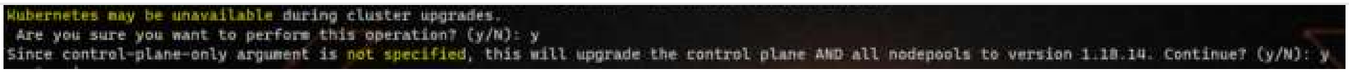
** Administrators could decide whether to only upgrade control plane (master nodes) or both control plane and worker nodes.

The command to only upgrade AKS control plane

```
az aks upgrade \
 --resource-group <RG name> \
 --name <AKS cluster name> \
 --kubernetes-version <target K8s version> \
 --control-plane-only
```

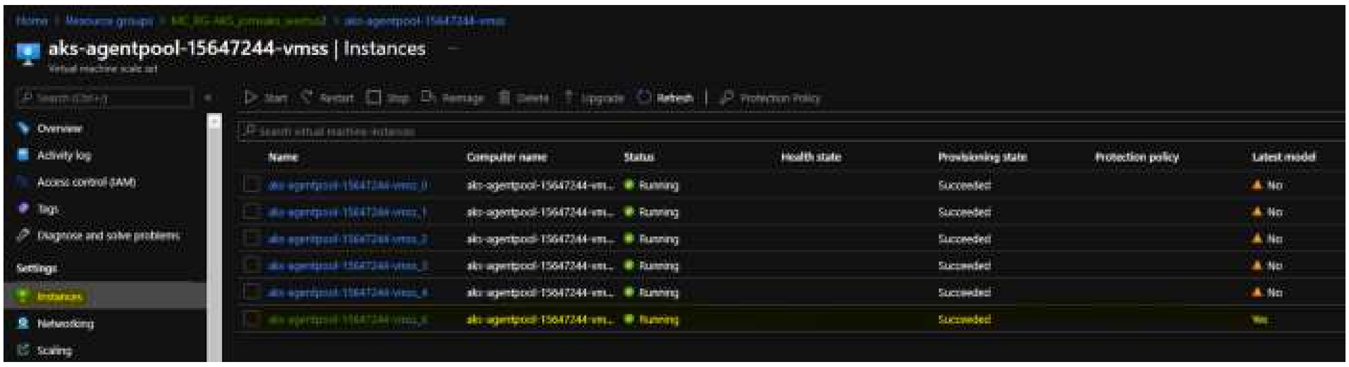
Finally, we could upgrade AKS cluster with

```
az aks upgrade \
 --resource-group <RG name> \
 --name <AKS cluster name> \
 --kubernetes-version <target K8s version>
```



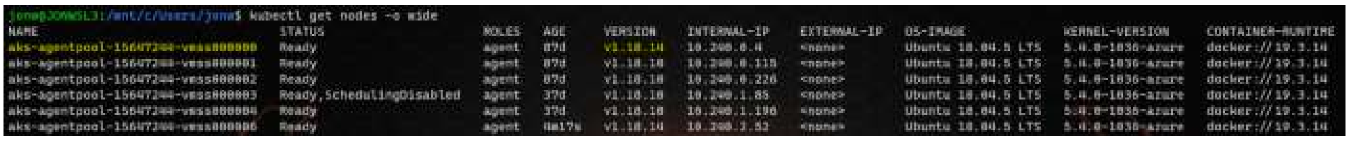
```
Kubernetes may be unavailable during cluster upgrades.
 Are you sure you want to perform this operation? (y/N): y
Since control-plane-only argument is not specified, this will upgrade the control plane AND all nodepools to version 1.18.14. Continue? (y/N): y
```

After executing the command, head to AKS-associated VMSS resource group, it usually is in the format of “MC_<RG name>_<AKS cluster name>_<region>”. You could see one additional node being created and running in the latest node image version (node image version is paired with the selected K8s version), assuming you are going by default, meaning not having node surge percentage set to a different value. What the platform is doing right now is by draining one of the nodes’ workload to this newly created node and make the drained node to be available for upgrading to the target K8s version.



Name	Computer name	Status	Health state	Provisioning state	Protection policy	Latest model
aks-agentpool-15647244-vmss_0	aks-agentpool-15647244-vm...	Running		Succeeded		No
aks-agentpool-15647244-vmss_1	aks-agentpool-15647244-vm...	Running		Succeeded		No
aks-agentpool-15647244-vmss_2	aks-agentpool-15647244-vm...	Running		Succeeded		No
aks-agentpool-15647244-vmss_3	aks-agentpool-15647244-vm...	Running		Succeeded		No
aks-agentpool-15647244-vmss_4	aks-agentpool-15647244-vm...	Running		Succeeded		No
aks-agentpool-15647244-vmss_6	aks-agentpool-15647244-vm...	Running		Succeeded		Yes

If we use K8s CLI to check the node version, we would see the first node is successfully running in the target K8s version, 5 more node upgrades to go. At the end of the process, the newly created node would be deleted as it is no longer needed.



```
jona@JONWSL3:/mnt/c/Users/jona$ kubectl get nodes -o wide
NAME                                STATUS                     ROLES   AGE     VERSION    INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
aks-agentpool-15647244-vmss000000   Ready                      agent   87d     v1.18.14   10.240.0.4     <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure   docker://19.3.14
aks-agentpool-15647244-vmss000001   Ready                      agent   87d     v1.18.10   10.240.0.115   <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure   docker://19.3.14
aks-agentpool-15647244-vmss000002   Ready                      agent   87d     v1.18.10   10.240.0.226   <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure   docker://19.3.14
aks-agentpool-15647244-vmss000003   Ready,SchedulingDisabled   agent   37d     v1.18.10   10.240.1.85    <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure   docker://19.3.14
aks-agentpool-15647244-vmss000004   Ready                      agent   37d     v1.18.10   10.240.1.196   <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure   docker://19.3.14
aks-agentpool-15647244-vmss000006   Ready                      agent   4m17s   v1.18.14   10.240.2.52    <none>        Ubuntu 18.04.5 LTS   5.4.0-1036-azure   docker://19.3.14
```

Node Image Version Upgrade

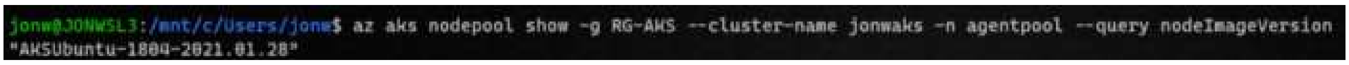
Security Concerns

As we all know that containers are essentially running as a process within the hosts, keeping the host a secure environment is crucial. Therefore, OS security patches are performed to AKS Linux nodes on a nightly basis. However, this is not the case for AKS Windows nodes. That said, some security patches require hosts to reboot before those could be applied and this is the responsibility of AKS administrators. Basically, there are 2 options to perform this operation, one is manual reboot each node and the other

To get more information on how to deploy Kured in AKS, please check this official documentation.

Check AKS Node Image Version

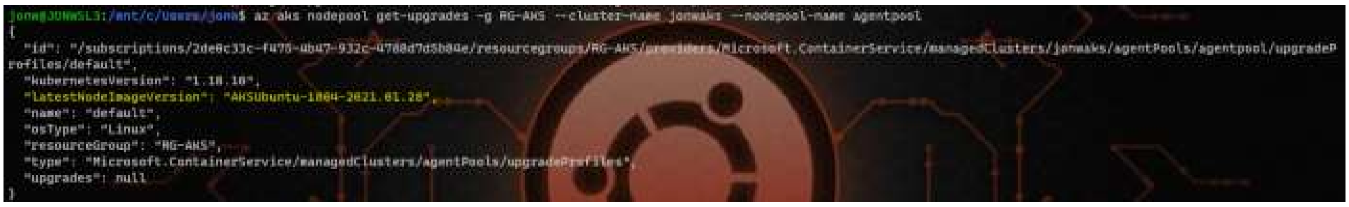
```
az aks nodepool show \
 --resource-group <RG name> \
 --cluster-name <AKS cluster name> \
 --name <target node pool name> \
 --query nodeImageVersion
```



```
jonw@JONWSL3:/mnt/c/Users/jonw$ az aks nodepool show -g RG-AWS --cluster-name jonwaks -n agentpool --query nodeImageVersion
"AKSUbuntu-1804-2021.01.28"
```

Get AKS Node Image Available Upgrades

```
az aks nodepool get-upgrades \
 --resource-group <RG name> \
 --cluster-name <AKS cluster name> \
 --nodepool-name <target node pool name>
```

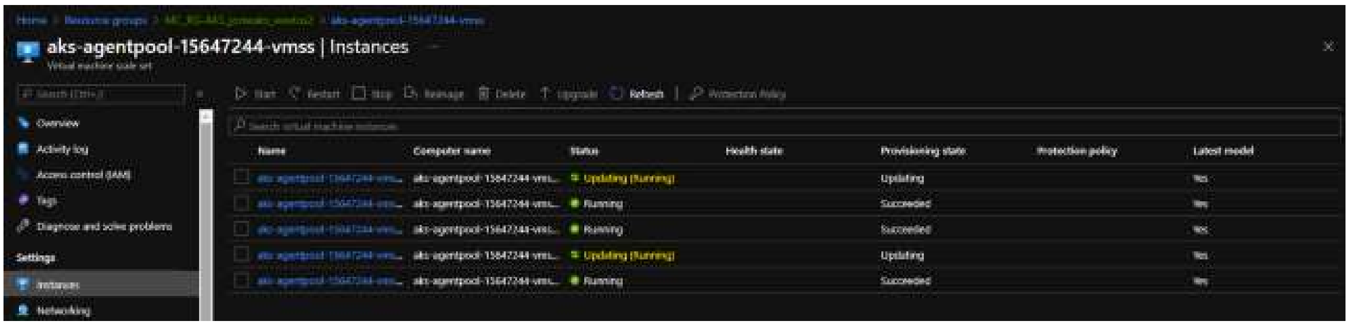


```
jonw@JONWSL3:/mnt/c/Users/jonw$ az aks nodepool get-upgrades -g RG-AKS --cluster-name jonwaks --nodepool-name agentpool
{
  "id": "/subscriptions/2de0c33c-f475-4b47-932c-4780d7d5b04e/resourcegroups/RG-AKS/providers/Microsoft.ContainerService/managedClusters/jonwaks/agentPools/agentpool/upgradeP
rofiles/default",
  "kubernetesVersion": "1.18.10",
  "latestNodeImageVersion": "AKSUbuntu-1804-2021.01.28",
  "name": "default",
  "osType": "Linux",
  "resourceGroup": "RG-AKS",
  "type": "Microsoft.ContainerService/managedClusters/agentPools/upgradeProfiles",
  "upgrades": null
}
```

Upgrade AKS Node Image Version to the Latest

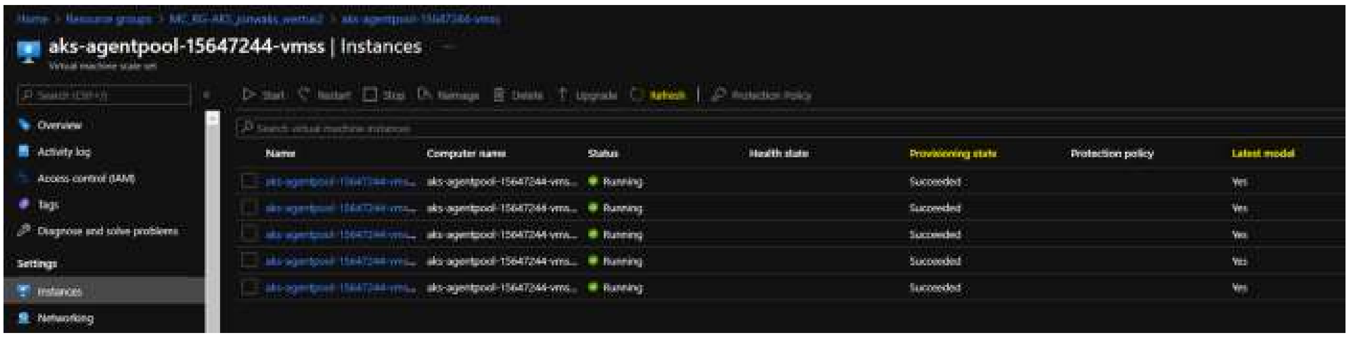
```
    --cluster-name <AKS cluster name> \
    --name <target node pool name> \
    --node-image-only
```

After executing the command, head to AKS-associated VMSS resource group, it usually is in the format of "MC_<RG name>_<AKS cluster name>_<region>". You could see nodes are being upgraded one by one.


aks-agentpool-15647244-vmss | Instances

Name	Computer name	Status	Health state	Provisioning state	Protection policy	Latest model
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Updating (Running)		Updating		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Updating (Running)		Updating		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes

After a while, click on the "Refresh" button on the top, and if the "Provisioning state" is "Succeeded" and the "Latest model" shows "Yes", we could confirm each node is using the latest node image version in the current running version on AKS.


aks-agentpool-15647244-vmss | Instances

Name	Computer name	Status	Health state	Provisioning state	Protection policy	Latest model
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes
aks-agentpool-15647244-vms...	aks-agentpool-15647244-vms...	Running		Succeeded		Yes

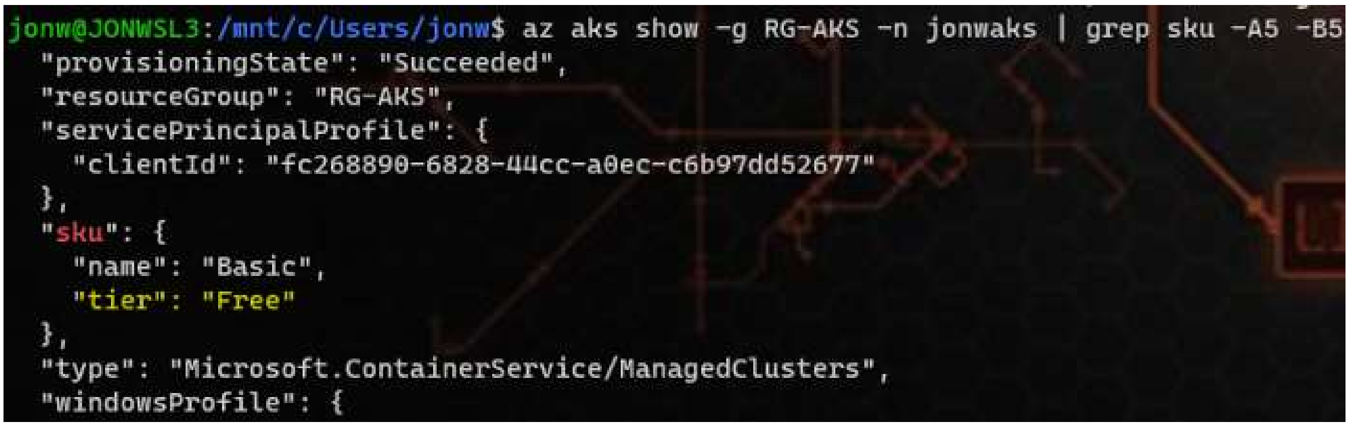
Azure-Associated Service Upgrade

Uptime SLA

AKS core components, mainly services provided in the control plane, could be upgraded to higher SLA. From this official documentation, we could know that AKS could get 99.9% update without Availability Zones and 99.95% with Availability Zones.

To upgrade existing AKS cluster to use uptime SLA, please follow the steps below.

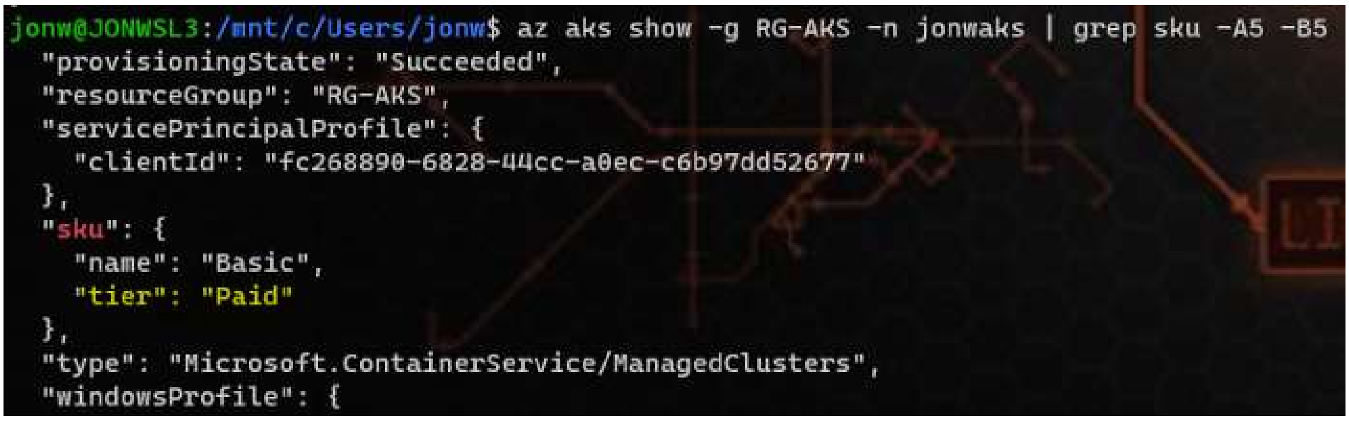
Open in app



```
jonw@JONWSL3:/mnt/c/Users/jonw$ az aks show -g RG-AKS -n jonwaks | grep sku -A5 -B5
  "provisioningState": "Succeeded",
  "resourceGroup": "RG-AKS",
  "servicePrincipalProfile": {
    "clientId": "fc268890-6828-44cc-a0ec-c6b97dd52677"
  },
  "sku": {
    "name": "Basic",
    "tier": "Free"
  },
  "type": "Microsoft.ContainerService/ManagedClusters",
  "windowsProfile": {
```

At the moment, we could see “sku” is showing tier as “Free”. If you execute the command below, the “sku” of this cluster would be showing “Paid”.

```
az aks update -g <RG name> -n <AKS cluster name> - uptime-sla
```

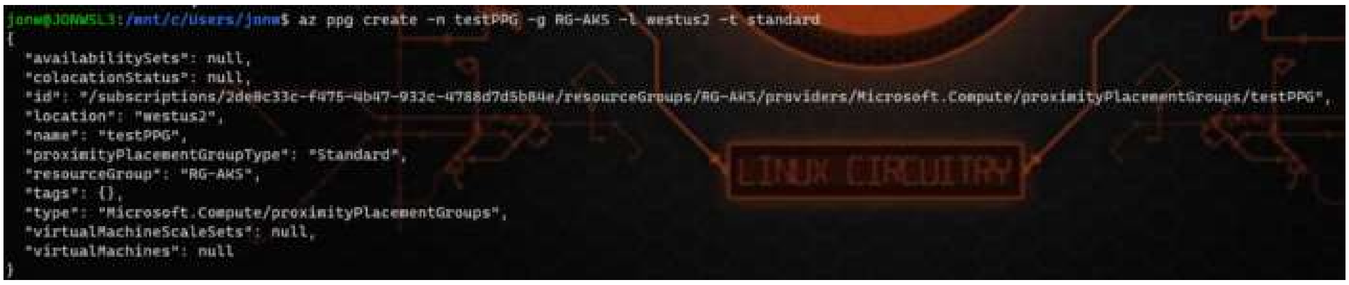


```
jonw@JONWSL3:/mnt/c/Users/jonw$ az aks show -g RG-AKS -n jonwaks | grep sku -A5 -B5
  "provisioningState": "Succeeded",
  "resourceGroup": "RG-AKS",
  "servicePrincipalProfile": {
    "clientId": "fc268890-6828-44cc-a0ec-c6b97dd52677"
  },
  "sku": {
    "name": "Basic",
    "tier": "Paid"
  },
  "type": "Microsoft.ContainerService/ManagedClusters",
  "windowsProfile": {
```

Proximity Placement Group

When adding a node pool with proximity placement group (PPG), basically means AKS nodes are being logically put in close physical location to reduce network latency, so applications for gaming and that are compute-intensive could have better performance. For more details, please check this official documentation.

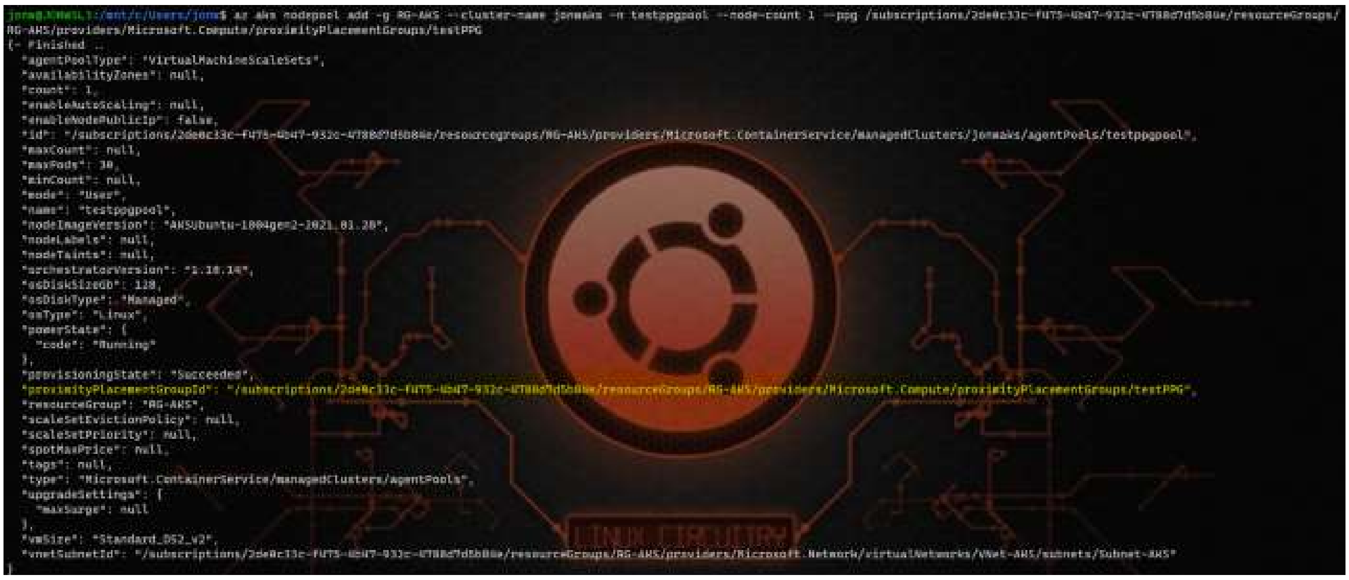
To create a PPG.



```
jonw@JONWSL3:/mnt/c/Users/jonw$ az ppg create -n testPPG -g RG-AKS -l westus2 -t standard
{
  "availabilitySets": null,
  "colocationStatus": null,
  "id": "/subscriptions/2de8c33c-f475-4b47-932c-4788d7d5b84e/resourceGroups/RG-AKS/providers/Microsoft.Compute/proximityPlacementGroups/testPPG",
  "location": "westus2",
  "name": "testPPG",
  "proximityPlacementGroupType": "Standard",
  "resourceGroup": "RG-AKS",
  "tags": {},
  "type": "Microsoft.Compute/proximityPlacementGroups",
  "virtualMachineScaleSets": null,
  "virtualMachines": null
}
```

To create node pool inside the newly create PPG.

```
az aks nodepool add \
 --resource-group <RG name> \
 --cluster-name <AKS cluster name> \
 --name <node pool name> \
 --node-count <amount> \
 --ppg <PPG resource ID from above command>
```



```
jonw@JONWSL3:/mnt/c/Users/jonw$ az aks nodepool add -g RG-AKS --cluster-name jonwaks -n testppgpool --node-count 1 --ppg /subscriptions/2de8c33c-f475-4b47-932c-4788d7d5b84e/resourceGroups/
RG-AKS/providers/Microsoft.Compute/proximityPlacementGroups/testPPG
{- Finished ..
  "agentPoolType": "VirtualMachineScaleSets",
  "availabilityZones": null,
  "count": 1,
  "enableAutoScaling": null,
  "enableNodePublicIp": false,
  "id": "/subscriptions/2de8c33c-f475-4b47-932c-4788d7d5b84e/resourcegroups/RG-AKS/providers/Microsoft.ContainerService/managedClusters/jonwaks/agentPools/testppgpool",
  "maxCount": null,
  "maxPods": 30,
  "minCount": null,
  "mode": "User",
  "name": "testppgpool",
  "nodeImageVersion": "AKSUbuntu-1804gen2-2021.01.28",
  "nodeLabels": null,
  "nodeTaints": null,
  "orchestratorVersion": "1.18.14",
  "osDiskSizeGb": 128,
  "osDiskType": "Managed",
  "osType": "Linux",
  "powerState": {
    "code": "Running"
  },
  "provisioningState": "Succeeded",
  "proximityPlacementGroupId": "/subscriptions/2de8c33c-f475-4b47-932c-4788d7d5b84e/resourceGroups/RG-AKS/providers/Microsoft.Compute/proximityPlacementGroups/testPPG",
  "resourceGroup": "RG-AKS",
  "scaleSetEvictionPolicy": null,
  "scaleSetPriority": null,
  "spotMaxPrice": null,
  "tags": null,
  "type": "Microsoft.ContainerService/managedClusters/agentPools",
  "upgradeSettings": {
    "maxSurge": null
  },
  "vmSize": "Standard_DS2_v2",
  "vnetSubnetId": "/subscriptions/2de8c33c-f475-4b47-932c-4788d7d5b84e/resourceGroups/RG-AKS/providers/Microsoft.Network/virtualNetworks/VNet-AKS/subnets/Subnet-AKS"
}
```

Azure Kubernetes Service Kubernetes Upgrade Concern