

**Lab Manual- Upgrade an Azure Kubernetes Service (AKS) cluster with cli**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn916

**Contributor:**

Contents

[1. Objective 3](#_Toc156992118)

[2. Kubernetes versions 4](#_Toc156992119)

[3. Commands and flags which we use to upgrade AKS cluster: 5](#_Toc156992120)

[4. What is node surge? 5](#_Toc156992121)

[5. Upgrade an AKS cluster 6](#_Toc156992122)

[6. Create 3 Node Azure Kubernetes Cluster 7](#_Toc156992123)

[7. Create Deploy Nginx Application 10](#_Toc156992124)

[8. Upgrade the cluster 11](#_Toc156992125)

[9. Monitor the Upgrade of the cluster 12](#_Toc156992126)

[10. Delete the cluster 15](#_Toc156992127)

# Objective

An Azure Kubernetes Service (AKS) cluster will periodically need to be updated to ensure security and compatibility with the latest features. There are two components of an AKS cluster that are necessary to maintain:

* ***Cluster Kubernetes version*:** Part of the AKS cluster lifecycle involves performing upgrades to the latest Kubernetes version. It’s important that you upgrade to apply the latest security releases and to get access to the latest Kubernetes features, as well as to stay within the [AKS support window](https://learn.microsoft.com/en-us/azure/aks/supported-kubernetes-versions#kubernetes-version-support-policy).
* ***Node image version:*** AKS regularly provides new node images with the latest OS and runtime updates. It's beneficial to upgrade your nodes' images regularly to ensure support for the latest AKS features and to apply essential security patches and hot fixes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component name** | **Frequency of upgrade** | **Planned Maintenance supported** | **Supported operation methods** | **Documentation link** |
| Cluster Kubernetes version (minor) upgrade | Roughly every three months | Yes | Automatic, Manual | <https://learn.microsoft.com/en-us/azure/aks/upgrade-cluster> |
| Cluster Kubernetes version upgrade to supported patch version | [Approximately weekly. To determine the latest applicable version in your region, see the AKS release tracker](https://learn.microsoft.com/en-us/azure/aks/release-tracker) | Yes | Automatic, Manual | <https://learn.microsoft.com/en-us/azure/aks/upgrade-cluster> |
| Node image version upgrade | **Linux**: weekly | Yes | Automatic, Manual | <https://learn.microsoft.com/en-us/azure/aks/node-image-upgrade> |
| **Windows**: monthly |  |
| Security patches and hot fixes for node images | As-necessary |  |  | <https://learn.microsoft.com/en-us/azure/aks/concepts-vulnerability-management#worker-nodes> |

* An important practice that you should include as part of your upgrade process is remembering to follow commonly used deployment and testing patterns. Testing an upgrade in a development or test environment before deployment in production is an important step to ensure application functionality and compatibility with the target environment. It can help you identify and fix any errors, bugs, or issues that

might affect the performance, security, or usability of the application or underlying infrastructure.

# Kubernetes versions

Kubernetes uses the standard [Semantic Versioning](https://semver.org/) versioning scheme for each version:

[major].[minor].[patch]

Examples:

1.17.7

1.17.8

Each number in the version indicates general compatibility with the previous version:

* **Major versions** change when incompatible API updates or backwards compatibility might be broken.
* **Minor versions** change when functionality updates are made that are backwards compatible to the other minor releases.
* **Patch versions** change when backwards-compatible bug fixes are made.

Aim to run the latest patch release of the minor version you're running. For example, if your production cluster is on **1.17.7**, **1.17.8** is the latest available patch version available for the 1.17 series. You should upgrade to **1.17.8** as soon as possible to ensure your cluster is fully patched and supported.

<https://endoflife.date/azure-kubernetes-service>

Upgrading AKS cluster **control plane** and nodepool is important part in AKS cluster life cycle. It is important to always keep upgrading cluster on latest version to get latest features.

**Understanding versioning of Kubernetes control plane and worker nodes:**

1.Kubernetes uses the standard Semantic Versioning scheme.

2. The version number is expressed as x.y.z, where x is the major version, y is the minor version, and z is the patch version. For example, in version 1.21.6, 1 is the major version, 21 is the minor version, and 6 is the patch version.

3. The Kubernetes version of the control plane and the initial node pool are set during cluster creation.

4.All additional node pools have their Kubernetes version set when they are added to the cluster.

5. The Kubernetes versions may differ between node pools as well as between a node pool and the control plane.

**Rules for valid versions to upgrade node pools:**

1. The node pool version must have the same major version as the control plane.

2. The node pool minor version must be within two minor versions of the control plane version. For example, if control plane version is 1.21.6, we can run nodepool with version 1.21.6, 1.21.5 or 1.21.4.

3.The node pool version cant be greater than the control plane version.

4.Kubernetes minor versions can&t be skipped with exception when you are upgrading from unsupported version. For example 1.20.x -> 1.21.x or 1.21.x -> 1.22.x are allowed, but 1.19.x -> 1.21.x is not allowed. Let us say you are running 1.15.x which is unsupported, you can skip and directly upgrade it to supported version 1.21.x.

5. **You can’t downgrade the control plane or a node pool Kubernetes version.**

6. If we don&t specify the target version, behavior depends upon the **az cli version**

7.No multiple operations are allowed on a single control plane or node pool resource simultaneously

8. If no patch is specified, the cluster will automatically be upgraded to the specified minor version’s latest GA patch. For example, setting --kubernetes- version to 1.21 will result in the cluster upgrading to 1.21.9.

# Commands and flags which we use to upgrade AKS cluster:

**az aks upgrade -** command is used to upgrade control plane version and all node pools in the cluster.

**--control-plane-only** flag - upgrades only the cluster control plane, no change in nodepools.

**az aks nodepool upgrade** - upgrades only the target node pool with specified version

# What is node surge?

* This is number/percentage of nodes you specify per nodepool which is added as spare node/s where all workloads will be shifted during the upgrade time. For example, if a nodepool has 10 agents and max node surge of 5 nodes or 50% (we can specify integer value of percent value). Additional 5 nodes will be required and provisioned at the time of nodepool upgrade.
* The max surge value of 100% (double the no of odes) provides faster upgrade but cause all nodes in nodepool drained simultaneously. It can be okay to have 100% node surge for testing environment. But for production recommended max surge is 33%.

# Set max surge for a new node pool

az aks nodepool add -n mynodepool -g MyResourceGroup --cluster-name MyManagedCluster --max-surge 33%

# Update max surge for an existing node pool

az aks nodepool update -n mynodepool -g MyResourceGroup --cluster-name MyManagedCluster --max-surge 5

AKS accepts both integer values and a percentage value for max surge. An integer such as **5** indicates five extra nodes to surge. A value of **50%** indicates a surge value of half the current node count in the pool.

**Few points to remember:**

* By default, AKS configures to upgrade with one extra node i.e. surge max node is 1.
* The max surge can not be higher than the number of nodes in the pool at the time of upgrade.
* The max surge setting on a node pool is persistent. Subsequent version upgrade will use same setting. You can change the max surge of nodepool at any point of time.
* Do well planning for available IPs in the subnet and compute quota when you configure node surge.
* Ensure PodDisruption Budgets are set correctly. This defines number of pod replicas that can be moved/terminated during drain process. If drain operation fails, the upgrade will also fail.

# Upgrade an AKS cluster

During the cluster upgrade process, AKS performs the following operations:

* Add a new buffer node (or as many nodes as configured in [max surge](https://github.com/MicrosoftDocs/azure-docs/blob/main/articles/aks/upgrade-aks-cluster.md#customize-node-surge-upgrade)) to the cluster that runs the specified Kubernetes version.
* [Cordon and drain](https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/) one of the old nodes to minimize disruption to running applications. If you're using max surge, it [cordons and drains](https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/) as many nodes at the same time as the number of buffer nodes specified.
* For long running pods, you can configure the node drain timeout, which allows for custom wait time on the eviction of pods and graceful termination per node. If not specified, the default is 30 minutes.

#### Set node drain timeout value

At times, you may have a long running workload on a certain pod and it cannot be rescheduled to another node during runtime, for example, a memory intensive stateful workload that must finish running. In these cases, you can configure a node drain timeout that AKS will respect in the upgrade workflow. If no node drain timeout value is specified, the default is 30 minutes. If the drain time out value elapses and pods have not yet finished running , then the upgrade operation is stopped. Any subsequent PUT operation shall resume the stopped upgrade.

# Set drain timeout for a new node pool

az aks nodepool add -n mynodepool -g MyResourceGroup --cluster-name MyManagedCluster --drainTimeoutInMinutes 100

# Update drain timeout for an existing node pool

az aks nodepool update -n mynodepool -g MyResourceGroup --cluster-name MyManagedCluster --drainTimeoutInMinutes 45

* When the old node is fully drained, it's reimaged to receive the new version and becomes the buffer node for the following node to be upgraded.
* Optionally, you can set a duration of time to wait between draining a node and proceeding to reimage it and move on to the next node. A short interval allows you to complete other tasks, such as checking application health from a Grafana dashboard during the upgrade process. We recommend a short timeframe for the upgrade process, as close to 0 minutes as reasonably possible. Otherwise, a higher **node soak time** (preview) affects how long before you discover an issue. The minimum soak time value is 0 minutes, with a maximum of 30 minutes. If not specified, the default value is 0 minutes.

#### Set node soak time value (preview)

To allow for a duration of time to wait between draining a node and proceeding to reimage it and move on to the next node, you can set the soak time to a value between 0 and 30 minutes. If no node soak time value is specified, the default is 0 minutes.

Enable the aks-preview Azure CLI.

az extension add --name aks-preview

Set node soak time for new or existing node pools

# Set node soak time for a new node pool

az aks nodepool add -n MyNodePool -g MyResourceGroup --cluster-name MyManagedCluster --node-soak-duration 10

# Update node soak time for an existing node pool

az aks nodepool update -n MyNodePool -g MyResourceGroup --cluster-name MyManagedCluster --max-surge 33% --node-soak-duration 5

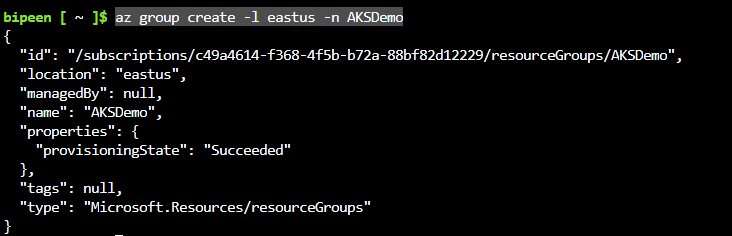
# Set node soak time when upgrading an existing node pool

az aks nodepool upgrade -n MyNodePool -g MyResourceGroup --cluster-name MyManagedCluster --max-surge 33% --node-soak-duration 20

* This process repeats until all nodes in the cluster have been upgraded.
* At the end of the process, the last buffer node is deleted, maintaining the existing agent node count and zone balance.

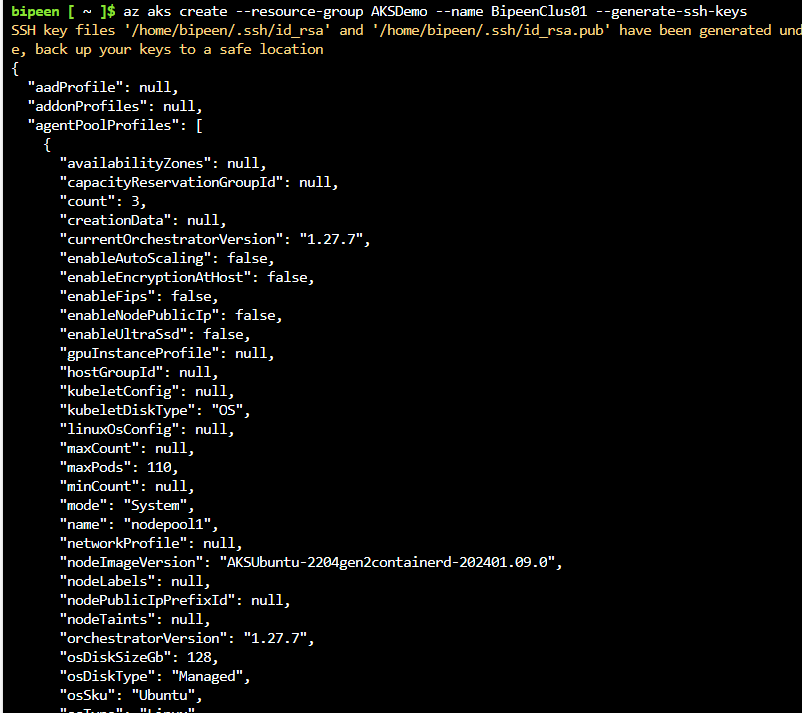
# Create 3 Node Azure Kubernetes Cluster

az group create -l eastus -n AKSDemo



To create an AKS cluster, use the [az aks create](https://learn.microsoft.com/en-us/cli/azure/aks#az-aks-create) command. The following example creates a cluster named BipeenClus01 with one node and generate SSH-key. It will automatically use 3 Nodes when you don’t specify node count

az aks create --resource-group AKSDemo --name BipeenClus01 --generate-ssh-keys

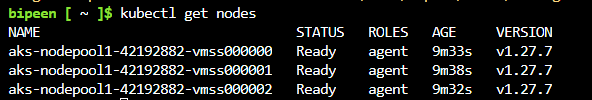


Configure kubectl to connect to your Kubernetes cluster using the [az aks get-credentials](https://learn.microsoft.com/en-us/cli/azure/aks#az-aks-get-credentials) command. This command downloads credentials and configures the Kubernetes CLI to use them.

az aks get-credentials --resource-group AKSDemo --name BipeenClus01

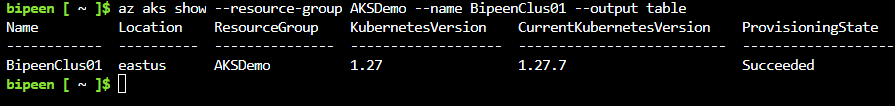


Verify the connection to your cluster using the [kubectl get](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get) command. This command returns a list of the cluster nodes and its Kubernetes version

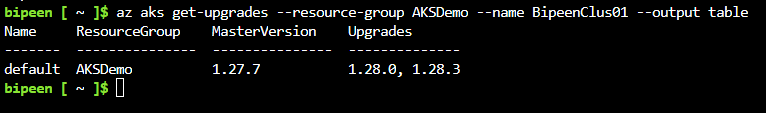


Checl control, panel version of control plane

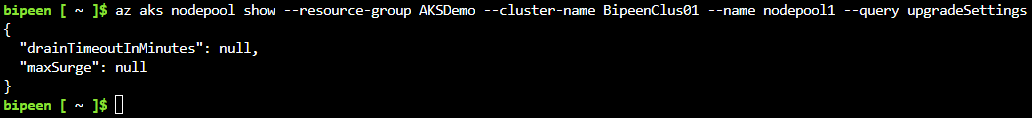
az aks show --resource-group AKSDemo --name BipeenClus01 --output table



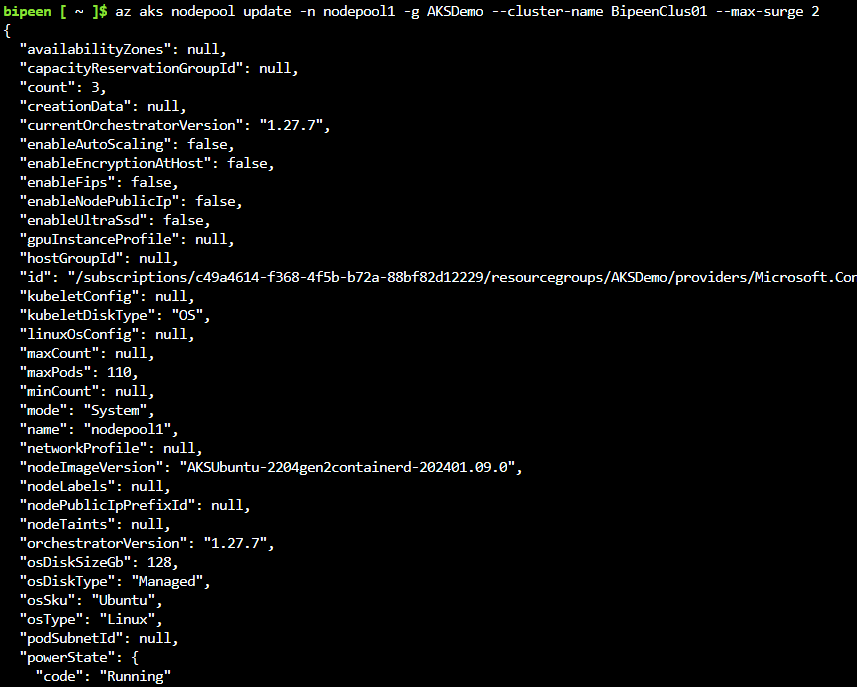
az aks get-upgrades --resource-group AKSDemo --name BipeenClus01 --output table

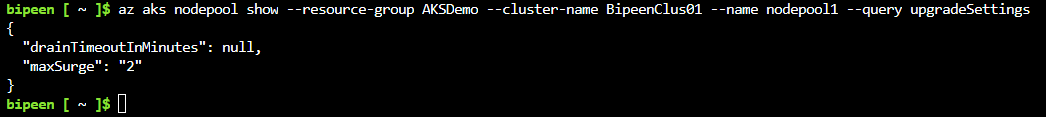


az aks nodepool show --resource-group AKSDemo --cluster-name BipeenClus01 --name nodepool1 --query upgradeSettings



az aks nodepool update -n nodepool1 -g AKSDemo --cluster-name BipeenClus01 --max-surge 2



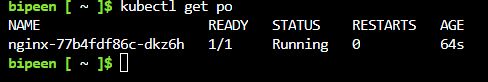


# Create Deploy Nginx Application

kubectl create deployment nginx --image nginx



kubectl get po

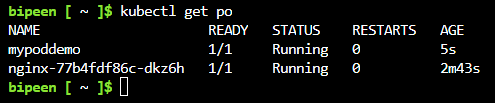


Now Create Standalone pod, it is not part of deployment or replicaset

kubectl run mypoddemo --image nginx

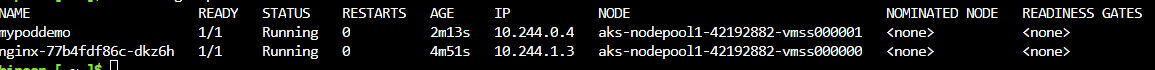


kubectl get po



Now lets see on which node pods are getting created

kubectl get pod -o wide

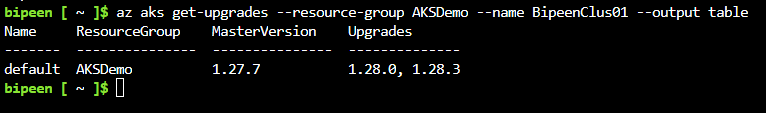


Note: mypoddemo run on 01 and deployment pod run on 00

# Upgrade the cluster

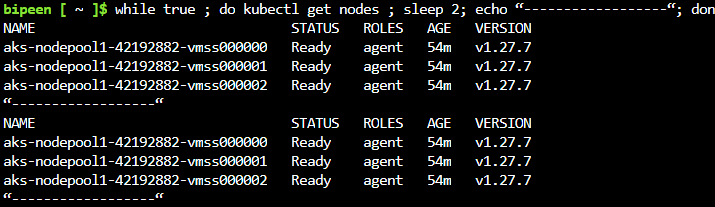
Check which version we can upgrade

az aks get-upgrades --resource-group AKSDemo --name BipeenClus01 --output table



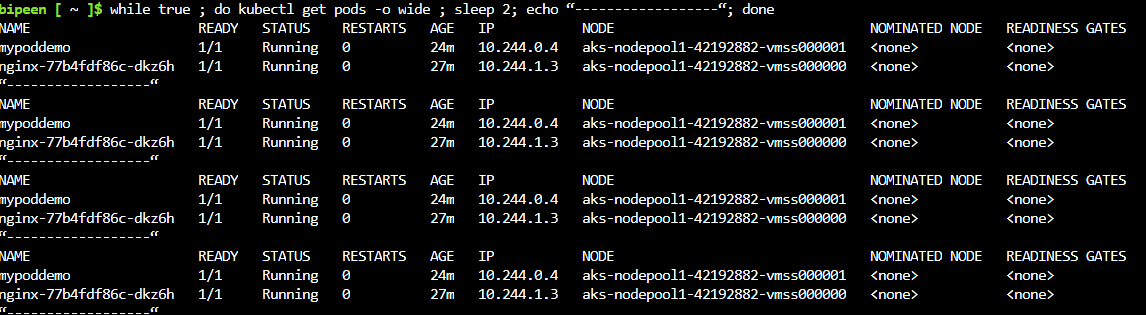
Open Another Session and type below command to keep it monitor

while true ; do kubectl get nodes ; sleep 2; echo “------------------“; done

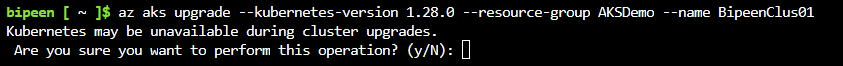


Open Third Session and type below command to keep Pod status cheek

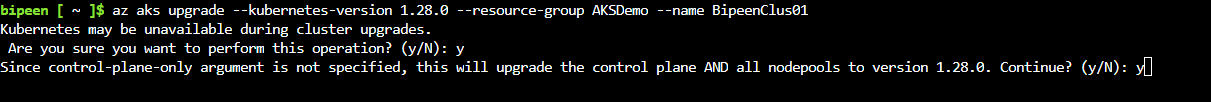
while true ; do kubectl get pods -o wide ; sleep 2; echo “------------------“; done

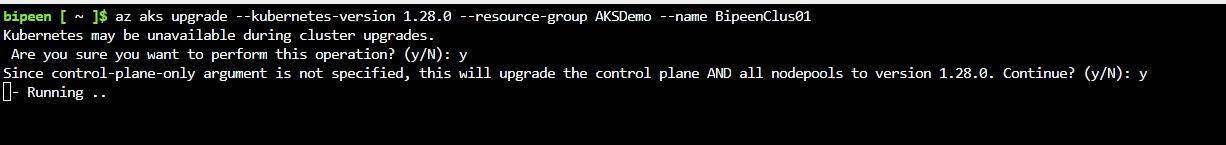


az aks upgrade --kubernetes-version 1.28.0 --resource-group AKSDemo --name BipeenClus01



Type yes

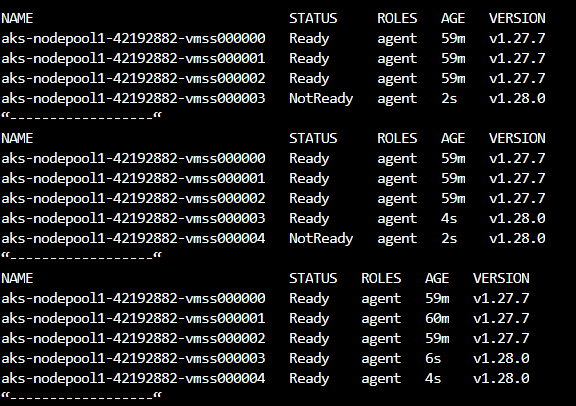




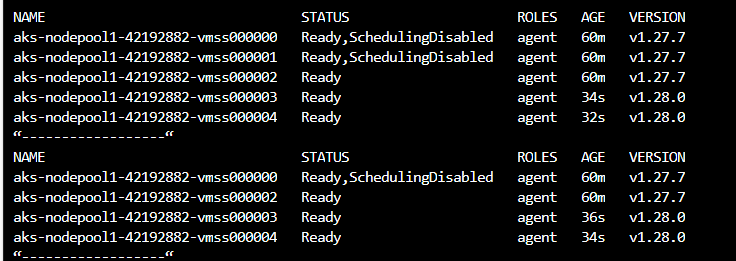
# Monitor the Upgrade of the cluster

Node monitor

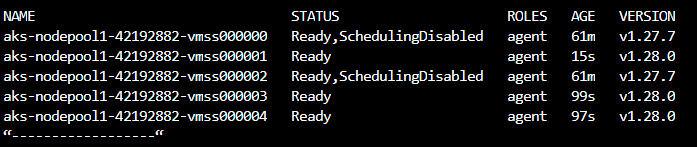
Two New Node Appear 3 and 4 with version 1.28.0



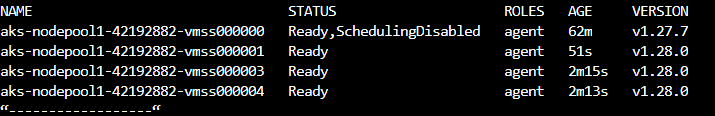
Node 00 and 01 is ready to drain out



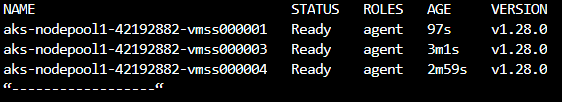
Node 01 is howing updated now and while 02 is also now draining



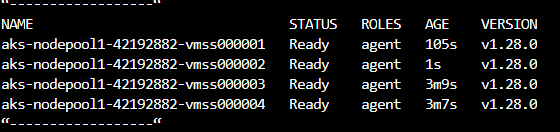
Node 2 drain out now



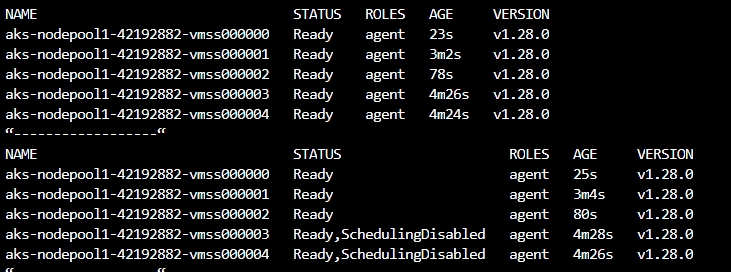
Both Node 0 and 2 Drain out



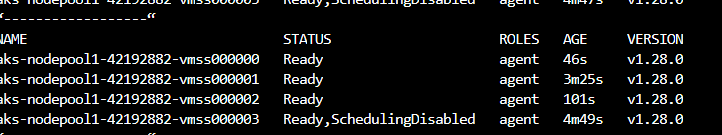
Node 2 Reimage with updated version and join the cluster



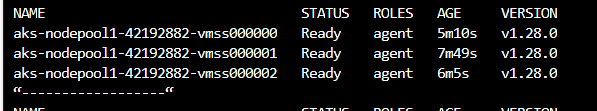
All Three Nodes 0,1 and 2 Reimage and joined back and surge nodes (3 and 4) now start draining



One Surge Node 4 is out

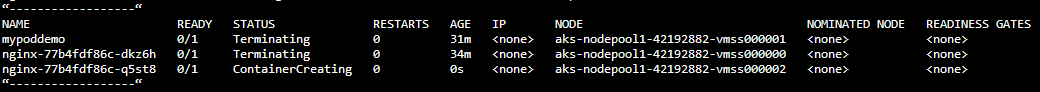


Now all Three nodes Back with reimages with updated version

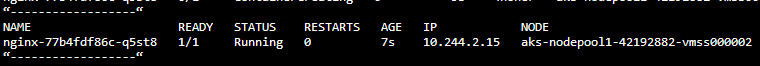


POD Monitoring

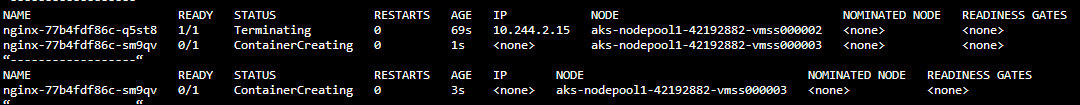
Both POD is terminating while POD which is part of deployment is start creating om **Node 2**



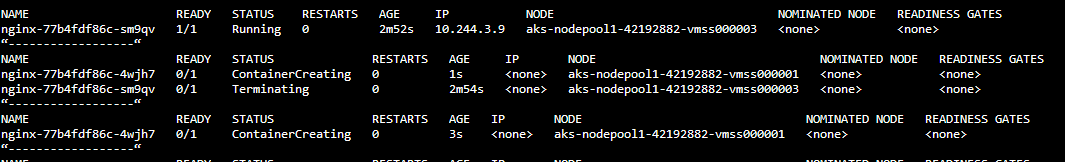
Now Only deployment is running on node 2



Now since two is also draining out , it start creating on Surge Node 3



Now Surge node is also start draining , it launch container on Node 1



# Delete the cluster

az group delete --name AKSDemo --yes --no-wait

