CRQ000001879661: AWS Mendix : EKS : Kubernetes 1.25 version upgrade

⚠ This wiki is outlines the step by step process which was followed for upgrading AWS EKS Kubernetes version from 1.24 to 1.25. This is not intended to be followed as such as the steps may vary based on the issues that may be encountered following these steps and also based on the Kubernetes version from and to which the upgrade is been done. Working experience on AWS and Kubernetes is desired for executing these steps.

Always refer to the official documentation and do detailed analysis before upgrading different versions.

⚠ Make sure to adapt the commands based on the environment they are executed in

Summary

In order to fix the vulnerability issue reported from dynatrace and to upgrade the EKS kubernetes version before the end of support for the current version, the AWS EKS kubernetes version across all the clusters in Dev, QA and Production needs to be upgraded to the next higher version. Upgrade till production needs to be completed before `Jan 31, 2024` .

Amazon EKS Kubernetes release calendar

As per doc,

Kubernetes version	Upstream release	Amazon EKS release	Amazon EKS end of support
`1.27`	April 11, 2023	May 24, 2023	July 2024
`1.26`	December 9, 2022	April 11, 2023	June 2024
`1.25`	August 23, 2022	February 22, 2023	May 2024
`1.24`	May 3, 2022	November 15, 2022	January 31, 2024
`1.23`	December 7, 2021	August 11, 2022	October 11, 2023

Steps

Prerequisites

Free IPs

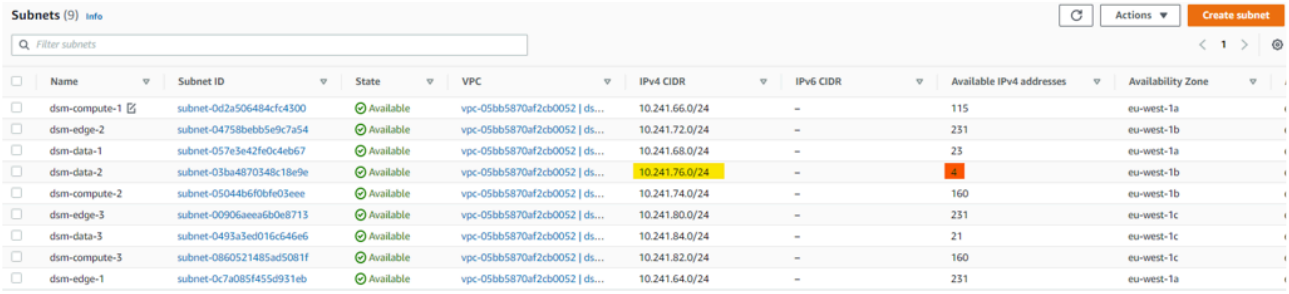
⚠ As per AWS document, To update the cluster, Amazon EKS requires up to five free IP addresses from the subnets that you specified when you created your cluster. Amazon EKS creates new cluster elastic network interfaces (network interfaces) in any of the subnets that you specified. The network interfaces may be created in different subnets than your existing network interfaces are in, so make sure that your security group rules allow required cluster communication for any of the subnets that you specified when you created your cluster. If any of the subnets that you specified when you created the cluster don't exist, don't have enough free IP addresses, or don't have security group rules that allows necessary cluster communication, then the update can fail.

Below is an error you may get if you dont have enough free IP addresses when upgrading the cluster

```
# /usr/local/bin/eksctl upgrade cluster -f eksctl-acc.yaml --approve
2022-10-20 04:43:53 [!]  NOTE: cluster VPC (subnets, routing & NAT Gateway) configuration changes are not yet implemented
```

```
2022-10-20 04:43:54 [Γä╣]  will upgrade cluster "dsm-mendix-alb" control plane from current version "1.24" to "1.25"
Error: operation error EKS: UpdateClusterVersion, https response error StatusCode: 400, RequestID: b915e60a-65eb-4c02-84b7-a2b6ae918bcd, InvalidRequestException: Provided subnets subnet-03ba4870348c18e9e Free IPs: 0 , need at least 5 IPs in each subnet to be free for this operation
```

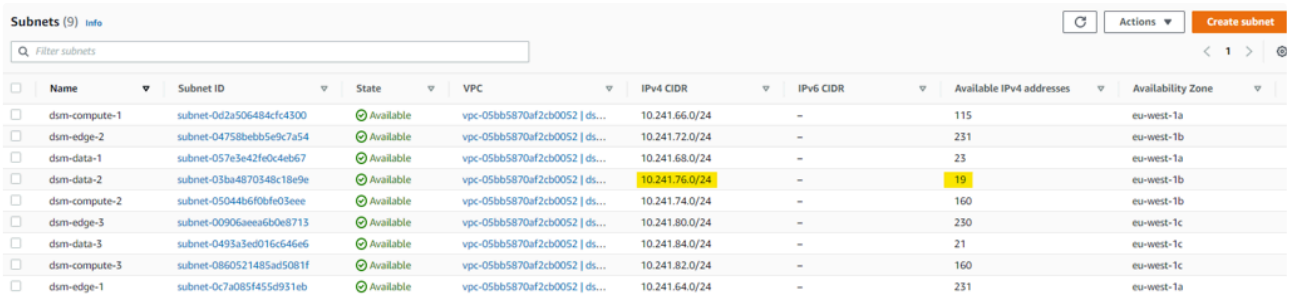
and from the subnets in the below screenshot, you can see there is only 4 free IP address available for the subnet `10.241.76.0/24`



We can check for any application containers that are in this subnet and temporarily scale down them to free up the IP addresses. But in a production environment it is done with caution as only some uncritical applications can be shutdown temporarily.

```
kubectl get pods --sort-by=.metadata.creationTimestamp -n mendix -o wide | awk '{print $6}' | cut -d. -f1,2,3 | sort | uniq -c
      43 10.241.66
      70 10.241.68
      30 10.241.74
      83 10.241.76
      30 10.241.82
      82 10.241.84
```

After we scaled down some uncritical / redundant apps (always check with concerned developers before stopping if it is for production) we can there are enough free Ips in all subnets to proceed with the upgrade process.



Permissions

- While upgrading the core addons using below commands, you may face below error.
  Please note to update the groups section to `system:masters` by adding a new entry, in the `mapRoles` section of the `aws-auth` `configmap` for the IAM role you are using to authenticate to the EKS cluster. Remove this entry at the end of the upgrade process.

```
eksctl utils update-kube-proxy --cluster=dsm-mendix-alb
Error: getting "kube-proxy": daemonsets.apps "kube-proxy" is forbidden: User "account-admin" cannot get resource "daemonsets" in API group "apps" in the namespace "kube-system"
```

```
kubectl edit configmap -n kube-system aws-auth
```

```
    - groups:
      - system:masters
      rolearn: arn:aws:iam::XXXXXXXXXXXX:role/AWSReservedSSO_dsm-account-admin_abcdefg1234567
      username: account-admin
```

- Following error may also occur when upgrading the core addons

```
/usr/local/bin/eksctl utils update-kube-proxy --cluster=dsm-mendix-alb
Error: getting list of API resources for raw REST client: Unauthorized
```

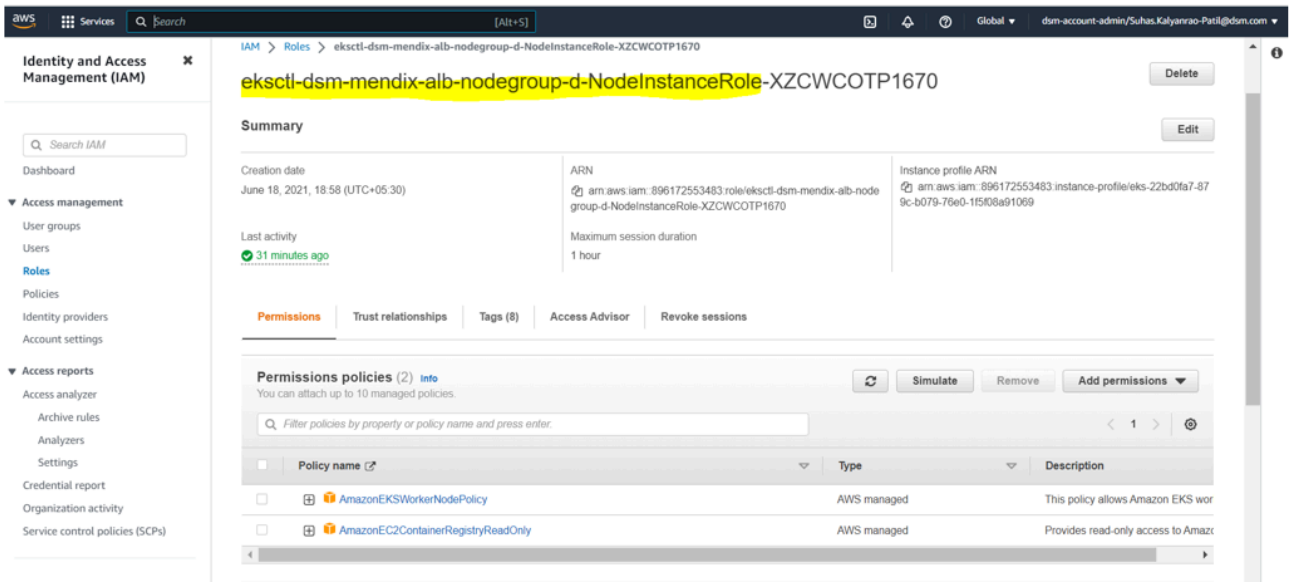
  In order to fix this the `aws-auth` `configmap` needs to be updated correctly. The issue may be with the `rolearn` field added incorrectly. If the `rolearn` is like `arn:aws:iam::XXXXXXXXXXXX:role/aws-reserved/sso.amazonaws.com/eu-west-1/AWSReservedSSO_dsm-account-admin_abcdefghi012345678` ,then change it to `arn:aws:iam::XXXXXXXXXXXX:role/AWSReservedSSO_dsm-account-admin_abcdefghi012345678`
  and update `aws-auth` `configmap`

```
kubectl edit configmap -n kube-system aws-auth
```

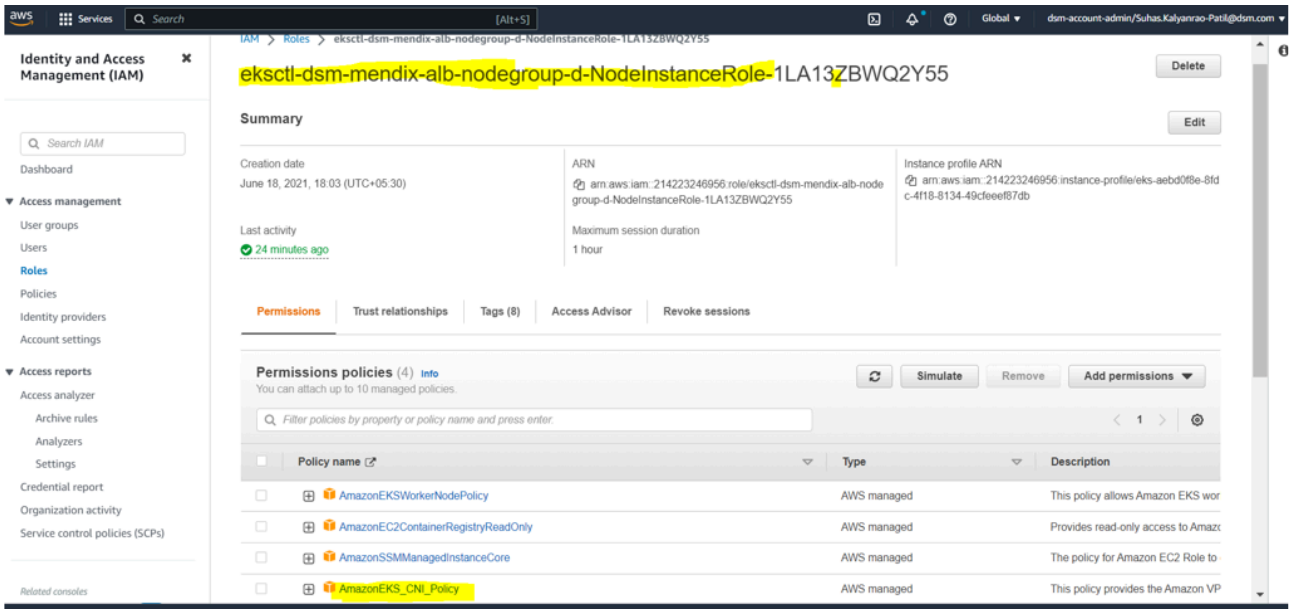
- Following error might be you receive when upgrading the Node Groups.

2023-02-08 19:50:36 UTC+0530	eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute	UPDATE_ROLLBACK_IN_PROGRESS	The following resource(s) failed to update: [ManagedNodeGroup].
2023-02-08 19:50:35 UTC+0530	ManagedNodeGroup	UPDATE_FAILED	Resource handler returned message: "[ErrorDetail(ErrorCode=NodeCreationFailure, ErrorMessage=Couldn't proceed with upgrade process as new nodes are not joining node group dsm-mendix-alb-nodegroup-compute, ResourceIds=[])] (Service: null, Status Code: 0, Request ID: null)" (RequestToken: f3d93ef7-3a3d-b9bd-5424-b9ca8dc653b6, HandlerErrorCode: GeneralServiceException)

Above error might be receive when you are trying to upgrade the node groups. So, you need to validate the EC2 instance owner access and there might be AmazonEKS_CNI_Policy permission policies missing as below.



So, you need to add the AmazonEKS_CNI_Policy permission policies for EC2 owner as below. And then newly created nodes will be attach to upgraded node group.



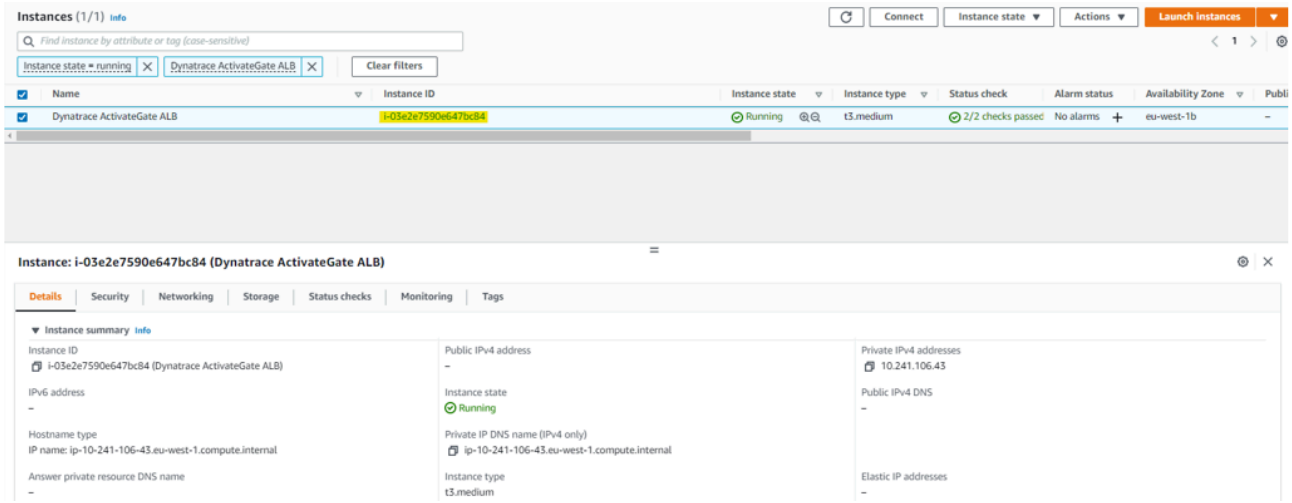
Pre-Steps

Plan is to upgrade the kubernetes version via `eksctl` tool as the cluster is created using the `eksctl` tool.

Since the cluster is created based on the template file defined in the Azure DevOps Repo aws-eks-cluster-provisioning, we would use it to upgrade the cluster using `eksctl` tool.

aws-eks-cluster-provisioning is a parameterized file and so values needs to be filled up corresponding to each environment. Also update the Kubernetes version to 1.25.

We choose to execute the below steps from the existing EC2 instance used for the Dynatrace Active Gate



- Connect to this instance using Session Manager using the AWS Prod account credentials and token from landing page

```
export AWS_ACCESS_KEY_ID="XXXXXXXXXXXXX"
export AWS_SECRET_ACCESS_KEY="YYYYYYYYYYYYYYYYYYYYY"
```

```
export AWS_SESSION_TOKEN="ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
```

```
aws --region eu-west-1 ssm start-session --target i-03e2e7590e647bc84
```

- Switch to root user

```
sh-4.2$ sudo -i
[root@ip-10-241-106-43 ~]#
```

- Install `eksctl` tool if not already available. Refer doc
- Establish connection to the required AWS account for which EKS upgrade is to be done using corresponding AWS credentials from landing zone. (Suggested to execute in a `screen` session)

```
export AWS_ACCESS_KEY_ID="AAAAAAAAAAAAA"
export AWS_SECRET_ACCESS_KEY="BBBBBBBBBBBBBBBBBBB"
export AWS_SESSION_TOKEN="CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC"
```

```
export AWS_DEFAULT_REGION=eu-west-1
```

- Establish connection with the corresponding EKS cluster using `kubeconfig` file. Can use the `kubeconfig` from local temporarily.
- Copy the `eksctl` configuration `yaml` file to a location from where the commands are to be executed.

Upgrade control plane

The first step of this process is to upgrade the EKS Control Plane. Since we used `eksctl` to provision our cluster we'll use that tool to do our upgrade as well.

- First we'll run this command:

```
/usr/local/bin/eksctl upgrade cluster -f eksctl-dt.yaml
2022-10-17 13:27:32 [!]  NOTE: cluster VPC (subnets, routing & NAT Gateway) configuration changes are not yet implemented
2022-10-17 13:27:32 [ℹ]  (plan) would upgrade cluster "dsm-mendix-alb" control plane from current version "1.24" to "1.25"
2022-10-17 13:27:33 [ℹ]  re-building cluster stack "eksctl-dsm-mendix-alb-cluster"
2022-10-17 13:27:33 [✔]  all resources in cluster stack "eksctl-dsm-mendix-alb-cluster" are up-to-date
2022-10-17 13:27:38 [ℹ]  checking security group configuration for all nodegroups
2022-10-17 13:27:38 [ℹ]  all nodegroups have up-to-date cloudformation templates
2022-10-17 13:27:38 [!]  no changes were applied, run again with '--approve' to apply the changes
```

You'll see in the output that it found our cluster, worked out that it is 1.24 and the next version is 1.25 (you can only go to the next version with EKS) and that everything is ready for us to proceed with an upgrade. We'll run it again with an `--approve` appended to proceed

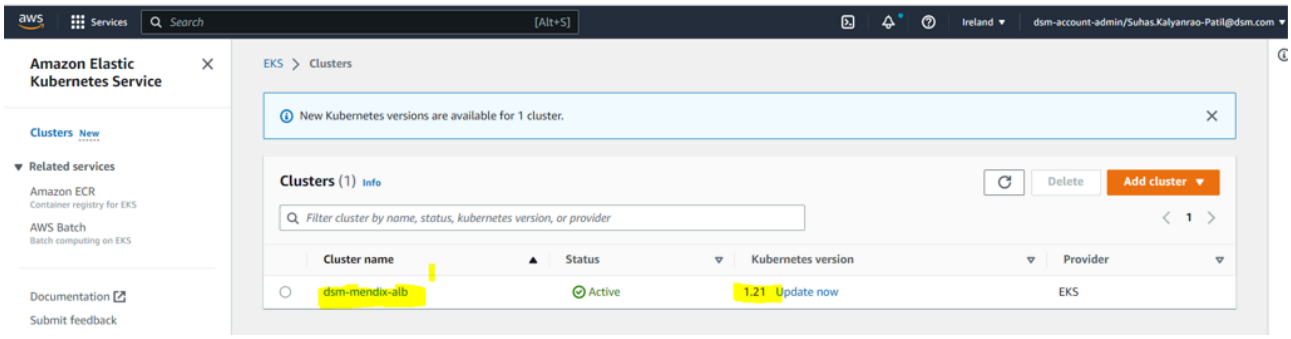
- Approve and upgrade

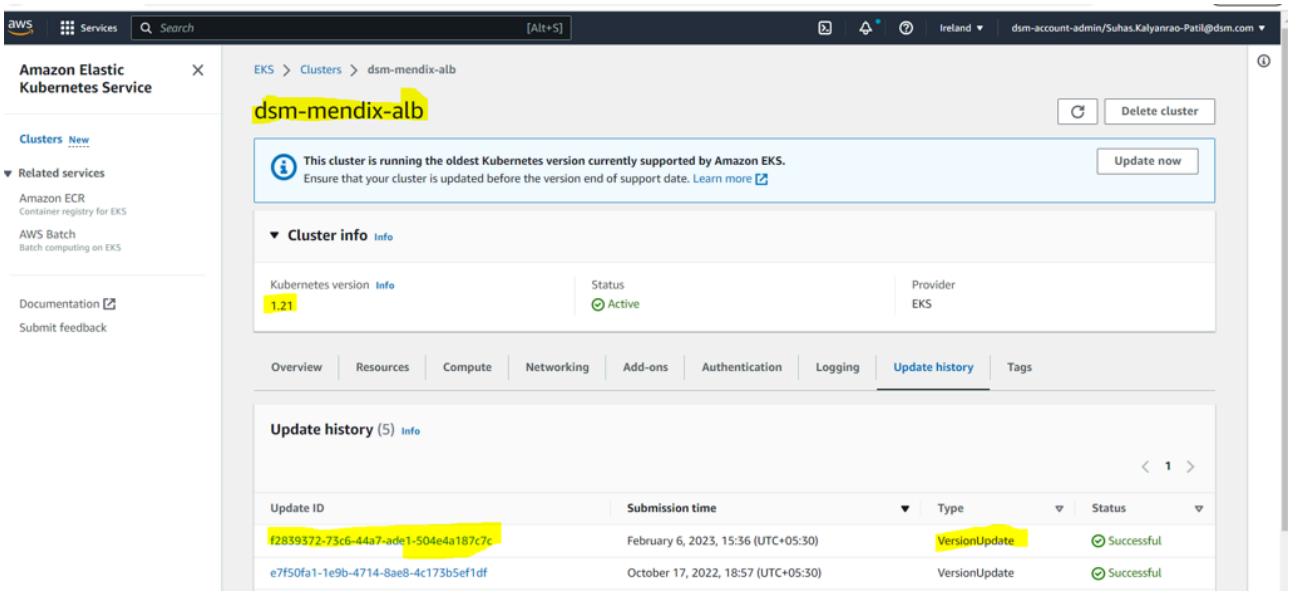
```
/usr/local/bin/eksctl upgrade cluster -f eksctl-dt.yaml --approve
2022-10-17 13:27:53 [!]  NOTE: cluster VPC (subnets, routing & NAT Gateway) configuration changes are not yet implemented
2022-10-17 13:27:54 [ℹ]  will upgrade cluster "dsm-mendix-alb" control plane from current version "1.24" to "1.25"
2022-10-17 13:37:24 [✔]  cluster "dsm-mendix-alb" control plane has been upgraded to version "1.25"
2022-10-17 13:37:24 [ℹ]  you will need to follow the upgrade procedure for all of nodegroups and add-ons
2022-10-17 13:37:24 [ℹ]  re-building cluster stack "eksctl-dsm-mendix-alb-cluster"
2022-10-17 13:37:24 [✔]  all resources in cluster stack "eksctl-dsm-mendix-alb-cluster" are up-to-date
2022-10-17 13:37:32 [ℹ]  checking security group configuration for all nodegroups
2022-10-17 13:37:32 [ℹ]  all nodegroups have up-to-date cloudformation templates
```

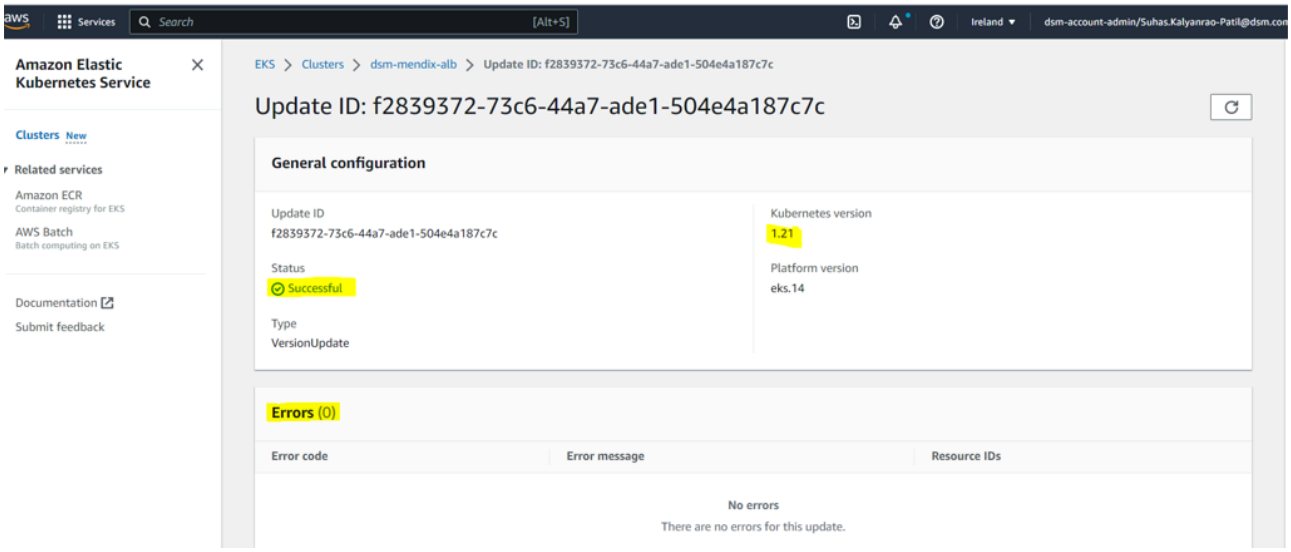
- Check the upgrade again to confirm

```
/usr/local/bin/eksctl upgrade cluster -f eksctl-dt.yaml
2022-10-17 13:51:21 [!]  NOTE: cluster VPC (subnets, routing & NAT Gateway) configuration changes are not yet implemented
2022-10-17 13:51:22 [ℹ]  no cluster version update required
2022-10-17 13:51:22 [ℹ]  re-building cluster stack "eksctl-dsm-mendix-alb-cluster"
2022-10-17 13:51:22 [✔]  all resources in cluster stack "eksctl-dsm-mendix-alb-cluster" are up-to-date
2022-10-17 13:51:27 [ℹ]  checking security group configuration for all nodegroups
2022-10-17 13:51:27 [ℹ]  all nodegroups have up-to-date cloudformation templates
```

- Status from EKS Console post upgrade







Upgrade EKS core addons

When you provision an EKS cluster you get three add-ons that run on top of the cluster and that are required for it to function properly:

- kubeproxy
- CoreDNS
- aws-node (AWS CNI or Network Plugin)

Kube Proxy

Analyze the compatible version for 1.25 Kubernete version and install image as per compatible version: `v1.25.11-minimal-eksbuild.2`

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-proxy=602401143452.dkr.ecr.eu-west-1.amazonaws.com/eks/kube-proxy:v1.25.11-minimal-eksbuild.2
eksctl utils update-kube-proxy --cluster=dsm-mendix-alb --approve (auto install the compatible version as per cluster support)
```

Core DNS (not needed for 1.25)

Analyze the compatible version for 1.25 Kubernete version and install image as per compatible

Version : `v1.9.3-eksbuild.5`

```
eksctl utils update-coredns --cluster=dsm-mendix-alb --approve (auto install the compatible version as per cluster support)
```

AWS Node (VPC CNI)

Analyze the compatible version for 1.25 Kubernete version and install image as per compatible

version : `v1.13.4-eksbuild.1`

```
eksctl utils update-aws-node --cluster=dsm-mendix-alb --approve (auto install the compatible version as per cluster support)
```

We can confirm we succeeded by retrieving the versions of each with the commands:

```
kubectl get daemonset kube-proxy --namespace kube-system -o=jsonpath='{$.spec.template.spec.containers[:1].image}'
```

```
kubectl describe deployment coredns --namespace kube-system | grep Image | cut -d "/" -f 3
```

```
kubectl get daemonset aws-node --namespace kube-system -o=jsonpath='{$.spec.template.spec.containers[:1].image}'
```

Upgrade other addons

Aws-load-balancer-controller

Analyze aws-load-balancer-controller version compatible with Kubernetes 1.25

Check current aws-load-balancer-controller version

```
kubectl describe deployment aws-load-balancer-controller -n kube-system
```

To get only image ID (Version) execute below command

```
kubectl describe deployment aws-load-balancer-controller --namespace kube-system | grep Image
| cut -d "/" -f 3
```

Update the aws-load-balancer-controller image revison to 1.25 (analysis compatible version). This will redeploy aws-load-balancer-controller

```
kubectl set image deployment aws-load-balancer-controller -n kube-system aws-load-balancer-
controller=public.ecr.aws/eks/aws-load-balancer-controller:v2.5.2
```

Verify AWS load balance controller status

Once the AWS load balance controller is updated, verify aws-load-balancer-controller status

```
kubectl get deployments -n kube-system aws-load-balancer-controller

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   1/1     1            1           480d
```

Validated that new aws-load-balancer-controller version is scaling up and scaling down nodes based on resource usage in Dev environment.

Check logs

```
kubectl get pods -n kube-system -l app=aws-load-balancer-controller
NAME                                            READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-684c77f57b-r96f6   1/1     Running   0          19h
```

```
kubectl logs -f aws-load-balancer-controller-684c77f57b-r96f6 -n kube-system
```

Updating the Amazon EBS CSI driver as an Amazon EKS add-on

1. Check the current version of your Amazon EBS CSI add-on. Replace `my-cluster` with your cluster name.

   ```
   aws eks describe-addon --cluster-name dsm-mendix-alb --addon-name aws-ebs-csi-driver --
   query "addon.addonVersion" --output text
   ```

   The example output is as follows.

   ```
   v1.21.0-eksbuild.1
   ```

2. Determine which versions of the Amazon EBS CSI add-on are available for your cluster version.

   ```
   aws eks describe-addon-versions --addon-name aws-ebs-csi-driver --kubernetes-version 1.25
   \
     --query "addons[].addonVersions[].[addonVersion, compatibilities[].defaultVersion]" --
   output text
   ```

The example output is as follows.

```
v1.21.0-eksbuild.1
True
v1.20.0-eksbuild.1
True
v1.11.2-eksbuild.1
False
```

The version with `True` underneath is the default version deployed when the add-on is created. The version deployed when the add-on is created might not be the latest available version. In the previous output, the latest version is deployed when the add-on is created.

3. Update the add-on to the version with `True` that was returned in the output of the previous step. If it was returned in the output, you can also update to a later version.

```
aws eks update-addon --cluster-name dsm-mendix-alb --addon-name aws-ebs-csi-driver --addon-version v1.21.0-eksbuild.1 \
  --resolve-conflicts PRESERVE
```

4. Amazon EBS CSI driver compatible version for EKS 1.25 is **v1.21.0-eksbuild.1**

Cluster Auto scaler

Analyze cluster autoscaler version compatible with Kubernetes 1.25

Check current cluster autoscaler version

```
kubectl describe deployment cluster-autoscaler -n kube-systemonly image ID (Version) execute below command
```

```
kubectl describe deployment cluster-autoscaler --namespace kube-system | grep Image | cut -d "/" -f 3
```

Update the cluster autoscaler image revison to 1.25 (analysis compatible version). This will redeploy cluster autoscaler

```
kubectl set image deployment cluster-autoscaler -n kube-system cluster-autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:v1.26.4
```

Verify cluster autoscaler status

```
kubectl get deployments -n kube-system cluster-autoscaler

NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
cluster-autoscaler   1/1     1            1           480d
```

Validated that new cluster autoscaler version is scaling up and scaling down nodes based on resource usage in Dev environment.

Check logs

```
kubectl get pods -n kube-system -l app=cluster-autoscaler
NAME                                 READY   STATUS    RESTARTS   AGE
cluster-autoscaler-684c77f57b-r96f6   1/1     Running   0          19h
```

```
kubectl logs -f cluster-autoscaler-684c77f57b-r96f6 -n kube-system
```

External-dns

Analyze external-dns version compatible with Kubernetes 1.25

Check current external-dns version

```
kubectl describe deployment external-dns -n default
```

To get only image ID (Version) execute below command

```
kubectl describe deployment external-dns --namespace default | grep Image | cut -d "/" -f 3
```

Update the external-dns image revison to 1.25 (analysis compatible version). This will redeploy external-dns

```
kubectl set image deployment external-dns -n default external-dns=registry.k8s.io/external-dns/external-dns:v0.13.4
```

Verify external-dns status

```
kubectl get deployments -n default external-dns

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
external-dns   1/1     1            1           480d
```

Kubernetes dashboard

Analyze dashboard version compatible with Kubernetes 1.25.

To check the Kubernetes dashboard compatible version for EKS 1.25 check the following link: Releases · kubernetes/dashboard

Download dashboard yaml file

```
curl -o ~/Downloads/kubernetes-dashboard.yml kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml
```

Update the dashboard applications

```
kubectl apply -f ~/Downloads/kubernetes-dashboard.yml
```

Verify dashboard apps status

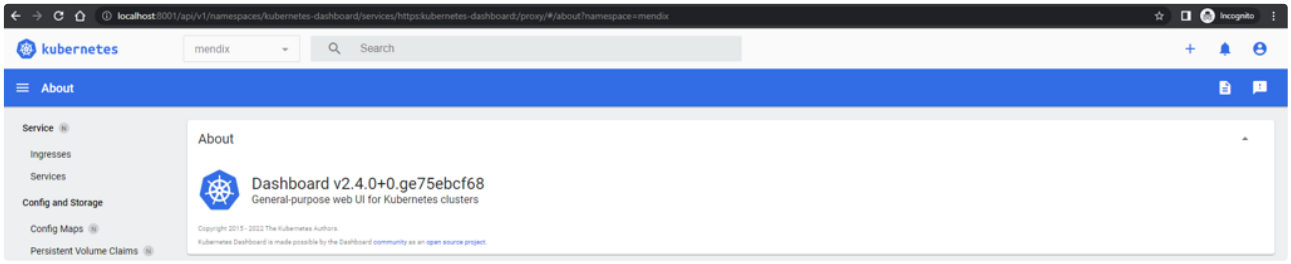
```
kubectl get pods -n kubernetes-dashboard

```

```
NAME                                         READY   STATUS    RESTARTS   AGE
dashboard-metrics-scraper-5b8896d7fc-pm2cg   1/1     Running   0          53s
kubernetes-dashboard-897c7599f-bq4bn         1/1     Running   0          53s
```

Verify the dashboard app version

```
kubectl describe pods -n kubernetes-dashboard dashboard-metrics-scraper-5b8896d7fc-pm2cg | grep Image
kubectl describe pods -n kubernetes-dashboard kubernetes-dashboard-897c7599f-bq4bn | grep Image
```



Fluent bit

Analyze Fluent bit version compatible with Kubernetes 1.25.

Check current Fluent bit version

```
kubectl describe daemonset/fluent-bit -n amazon-cloudwatch
```

Update the Fluent bit version which is compatible for Kubernete 1.25 Version.

```
kubectl set image daemonset/fluent-bit fluent-bit=906394416424.dkr.ecr.eu-west-1.amazonaws.com/aws-for-fluent-bit:2.28.4 -n amazon-cloudwatch
```

Refer the AWS page to know the required version for EKS 1.25: [Set up Fluent Bit as a DaemonSet to send logs to CloudWatch Logs]

Download and deploy the Fluent Bit daemonset to the cluster by running one of the following commands.

- If you want the Fluent Bit optimized configuration, run this command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit.yaml
```

Verify the Fluent bit version

```
kubectl describe daemonset/fluent-bit -n amazon-cloudwatch
```

Cloudwatch Agent

Analyze cloudwatch-agent version compatible with Kubernetes 1.25.

Check cloudwatch-agent version

```
kubectl describe daemonset/cloudwatch-agent -n amazon-cloudwatch
```

Update the Cloudwatch agent version which is compatible for Kubernete 1.25 Version.

```
kubectl set image daemonset/cloudwatch-agent cloudwatch-agent=public.ecr.aws/cloudwatch-agent/cloudwatch-agent:1.247358.0b252413 -n amazon-cloudwatch
```

Verify the cloudwatch agent version

```
kubectl describe daemonset/cloudwatch-agent -n amazon-cloudwatch
```

Upgrade managed nodes

We are using managed node groups.

The way that managed node groups does this is:

1. Amazon EKS creates a new Amazon EC2 launch template version for the Auto Scaling group associated with your node group. The new template uses the target AMI for the update.
2. The Auto Scaling group is updated to use the latest launch template with the new AMI.
3. The Auto Scaling group maximum size and desired size are incremented by one up to twice the number of Availability Zones in the Region that the Auto Scaling group is deployed in. This is to ensure that at least one new instance comes up in every Availability Zone in the Region that your node group is deployed in.
4. Amazon EKS checks the nodes in the node group for the eks.amazonaws.com/nodegroup-image label, and applies a eks.amazonaws.com/nodegroup=unschedulable:NoSchedule taint on all of the nodes in the node group that aren't labeled with the latest AMI ID. This prevents nodes that have already been updated from a previous failed update from being tainted.
5. Amazon EKS randomly selects a node in the node group and evicts all pods from it.
6. After all of the pods are evicted, Amazon EKS cordons the node. This is done so that the service controller doesn't send any new request to this node and removes this node from its list of healthy, active nodes.
7. Amazon EKS sends a termination request to the Auto Scaling group for the cordoned node.
8. Steps 5-7 are repeated until there are no nodes in the node group that are deployed with the earlier version of the launch template.
9. The Auto Scaling group maximum size and desired size are decremented by 1 to return to your pre-update values.

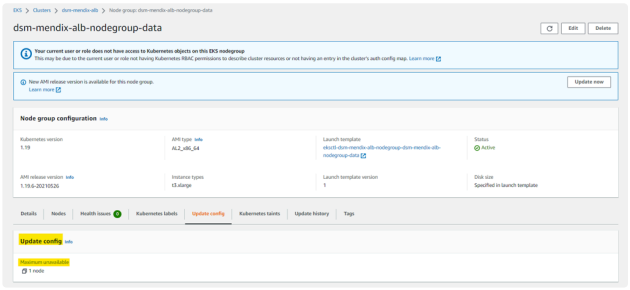
Update `updateConfig`

To avoid any downtime to your workloads due to upgrading multiple nodes at once, you can limit the number of nodes that can become unavailable during an upgrade by specifying this in the `maxUnavailable` field of an `updateConfig`. Alternatively, use `maxUnavailablePercentage`, which defines the maximum number of unavailable nodes as a percentage of the total number of nodes.

Note that `maxUnavailable` cannot be higher than `maxSize`. Also, `maxUnavailable` and `maxUnavailablePercentage` cannot be used simultaneously.

This feature is only available for managed nodes.

`eksctl` allows updating the UpdateConfig section of a managed nodegroup. This section defines two fields. `MaxUnavailable` and `MaxUnavailablePercentage`. Your nodegroups are unaffected during the update, thus downtime shouldn't be expected. Default value of `maxUnavailable` is 1.



The command `update nodegroup` should be used with a config file using the `--config-file` flag. The nodegroup should contain an `nodeGroup.updateConfig` section.

```
    updateConfig:
      maxUnavailable: 5
```

Above will change the value of `maxUnavailable` to 5 and so 5 nodes will be unavailable simultaneously and which will be upgraded. But change this value accordingly based on the requirement.

```
eksctl update nodegroup --config-file eksctl-dt.yaml

2022-10-18 15:27:32 [ä╣]  validating nodegroup "dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:27:32 [!]  unchanged fields for nodegroup dsm-mendix-alb-nodegroup-compute:
the following fields remain unchanged; they are not supported by `eksctl update nodegroup`:
AMIFamily, InstanceType, Subnets, ScalingConfig, VolumeSize, Labels, PrivateNetworking,
Tags, VolumeEncrypted, VolumeKmsKeyID
```

```
2022-10-18 15:27:32 [ℹ]  validating nodegroup "dsm-mendix-alb-nodegroup-data"
2022-10-18 15:27:32 [!]  unchanged fields for nodegroup dsm-mendix-alb-nodegroup-data: the following fields remain unchanged; they are not supported by `eksctl update nodegroup`: AMIFamily, InstanceType, Subnets, ScalingConfig, VolumeSize, Labels, PrivateNetworking, Tags, VolumeEncrypted, VolumeKmsKeyID
2022-10-18 15:27:32 [ℹ]  checking that nodegroup dsm-mendix-alb-nodegroup-compute is a managed nodegroup
2022-10-18 15:27:32 [ℹ]  updating nodegroup dsm-mendix-alb-nodegroup-compute's UpdateConfig
2022-10-18 15:27:32 [ℹ]  nodegroup dsm-mendix-alb-nodegroup-compute successfully updated
2022-10-18 15:27:32 [ℹ]  checking that nodegroup dsm-mendix-alb-nodegroup-data is a managed nodegroup
2022-10-18 15:27:32 [ℹ]  updating nodegroup dsm-mendix-alb-nodegroup-data's UpdateConfig
2022-10-18 15:27:33 [ℹ]  nodegroup dsm-mendix-alb-nodegroup-data successfully updated
```

Upgrade Node Groups

Since we are using cluster-autoscaler, we don't want conflicting Node scaling actions during our upgrade so we should scale that to zero to suspend it during this process using the command below.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

We can then trigger the MNG upgrade process by running the following eksctl command:

`--timeout` can be set based on requirement. Default it is 45 minutes

```
eksctl upgrade nodegroup --name=dsm-mendix-alb-nodegroup-compute --cluster=dsm-mendix-alb --kubernetes-version=1.25 --wait --timeout 120m0s

2022-10-18 15:48:40 [ℹ]  updating nodegroup stack to a newer format before upgrading nodegroup version
2022-10-18 15:48:40 [ℹ]  updating nodegroup stack
2022-10-18 15:48:40 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666108120" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:49:10 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666108120" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:49:11 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:49:41 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:50:15 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:50:15 [ℹ]  setting ForceUpdateEnabled value to false
2022-10-18 15:50:15 [ℹ]  updating nodegroup stack
2022-10-18 15:50:15 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666108215" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:50:45 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666108215" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:50:45 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:51:16 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:51:16 [ℹ]  upgrading nodegroup version
```

```
2022-10-18 15:51:16 [ℹ]  updating nodegroup stack
2022-10-18 15:51:16 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666108276" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:51:46 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666108276" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:51:46 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:52:16 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:53:10 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:54:33 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:55:47 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:56:30 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:57:37 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 15:59:25 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:00:15 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:01:17 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:02:30 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:03:56 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:05:25 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:06:43 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:08:07 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-compute"
2022-10-18 16:08:07 [ℹ]  nodegroup successfully upgraded
```

Check node upgrade status. We can see that the nodes in the first node group are all upgraded to version 1.25

```
$ kubectl get nodes --sort-by=.metadata.creationTimestamp

NAME                                           STATUS   ROLES    AGE     VERSION
ip-10-241-44-181.eu-west-1.compute.internal   Ready    <none>   21h     v1.19.6-eks-49a6c0
ip-10-241-36-218.eu-west-1.compute.internal   Ready    <none>   8h      v1.19.6-eks-49a6c0
ip-10-241-52-17.eu-west-1.compute.internal    Ready    <none>   55m     v1.19.6-eks-49a6c0
ip-10-241-52-110.eu-west-1.compute.internal   Ready    <none>   55m     v1.19.6-eks-49a6c0
ip-10-241-36-196.eu-west-1.compute.internal   Ready    <none>   55m     v1.19.6-eks-49a6c0
ip-10-241-44-83.eu-west-1.compute.internal    Ready    <none>   53m     v1.19.6-eks-49a6c0
ip-10-241-52-80.eu-west-1.compute.internal    Ready    <none>   52m     v1.19.6-eks-49a6c0
ip-10-241-50-116.eu-west-1.compute.internal   Ready    <none>   14m     v1.20.15-eks-ba74326
ip-10-241-34-81.eu-west-1.compute.internal    Ready    <none>   13m     v1.20.15-eks-ba74326
ip-10-241-42-204.eu-west-1.compute.internal   Ready    <none>   13m     v1.20.15-eks-ba74326
ip-10-241-50-160.eu-west-1.compute.internal   Ready    <none>   13m     v1.20.15-eks-ba74326
ip-10-241-42-161.eu-west-1.compute.internal   Ready    <none>   9m57s   v1.20.15-eks-ba74326
ip-10-241-34-240.eu-west-1.compute.internal   Ready    <none>   9m43s   v1.20.15-eks-ba74326
ip-10-241-34-146.eu-west-1.compute.internal   Ready    <none>   8m24s   v1.20.15-eks-ba74326
```

```
ip-10-241-42-113.eu-west-1.compute.internal   Ready    <none>   6m23s   v1.20.15-eks-ba74326
```

```
eksctl upgrade nodegroup --name=dsm-mendix-alb-nodegroup-data --cluster=dsm-mendix-alb --kubernetes-version=1.25 --wait --timeout 120m0s

2022-10-18 16:18:02 [ℹ]  updating nodegroup stack to a newer format before upgrading nodegroup version
2022-10-18 16:18:02 [ℹ]  updating nodegroup stack
2022-10-18 16:18:03 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666109882" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:18:33 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666109882" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:18:33 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:19:03 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:19:44 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:19:44 [ℹ]  setting ForceUpdateEnabled value to false
2022-10-18 16:19:44 [ℹ]  updating nodegroup stack
2022-10-18 16:19:44 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666109984" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:20:14 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666109984" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:20:14 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:20:44 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:20:44 [ℹ]  upgrading nodegroup version
2022-10-18 16:20:44 [ℹ]  updating nodegroup stack
2022-10-18 16:20:45 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666110044" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:21:15 [ℹ]  waiting for CloudFormation changeset "eksctl-update-nodegroup-1666110044" for stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:21:15 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:21:45 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:22:21 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:23:51 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:25:46 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:27:28 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:28:12 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:29:54 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:31:16 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:33:14 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:34:44 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:35:59 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
```

```
2022-10-18 16:37:00 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:37:46 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:39:36 [ℹ]  waiting for CloudFormation stack "eksctl-dsm-mendix-alb-nodegroup-dsm-mendix-alb-nodegroup-data"
2022-10-18 16:39:36 [ℹ]  nodegroup successfully upgraded
```

Verify that all nodes are upgraded to 1.25 now.

```
kubectl get nodes --sort-by=.metadata.creationTimestamp

NAME                                           STATUS   ROLES    AGE     VERSION
ip-10-241-50-116.eu-west-1.compute.internal   Ready    <none>   47m     v1.20.15-eks-ba74326
ip-10-241-34-81.eu-west-1.compute.internal    Ready    <none>   47m     v1.20.15-eks-ba74326
ip-10-241-42-204.eu-west-1.compute.internal   Ready    <none>   47m     v1.20.15-eks-ba74326
ip-10-241-50-160.eu-west-1.compute.internal   Ready    <none>   47m     v1.20.15-eks-ba74326
ip-10-241-42-161.eu-west-1.compute.internal   Ready    <none>   43m     v1.20.15-eks-ba74326
ip-10-241-34-240.eu-west-1.compute.internal   Ready    <none>   43m     v1.20.15-eks-ba74326
ip-10-241-34-146.eu-west-1.compute.internal   Ready    <none>   42m     v1.20.15-eks-ba74326
ip-10-241-42-113.eu-west-1.compute.internal   Ready    <none>   40m     v1.20.15-eks-ba74326
ip-10-241-36-24.eu-west-1.compute.internal    Ready    <none>   19m     v1.20.15-eks-ba74326
ip-10-241-36-48.eu-west-1.compute.internal    Ready    <none>   18m     v1.20.15-eks-ba74326
ip-10-241-52-243.eu-west-1.compute.internal   Ready    <none>   18m     v1.20.15-eks-ba74326
ip-10-241-44-197.eu-west-1.compute.internal   Ready    <none>   18m     v1.20.15-eks-ba74326
ip-10-241-44-132.eu-west-1.compute.internal   Ready    <none>   14m     v1.20.15-eks-ba74326
ip-10-241-44-47.eu-west-1.compute.internal    Ready    <none>   14m     v1.20.15-eks-ba74326
ip-10-241-52-119.eu-west-1.compute.internal   Ready    <none>   8m30s   v1.20.15-eks-ba74326
```

Scale up cluster autoscaler

```
kubectl scale deployments cluster-autoscaler --replicas=1 -n kube-system
```

Revert `updateConfig` to default value

Set the `maxUnavailable` in the `updateConfig` section to 1 in both the EKS managed node groups config file eksctl file and update

```
    updateConfig:
      maxUnavailable: 1
```

```
eksctl update nodegroup --config-file eksctl-dt.yaml
```

Remove the dsm-admin user privilege granted for the upgrade.

```
kubectl edit configmap -n kube-system aws-auth
```

Remove below section and save

```
- groups:
      - system:masters
      rolearn: arn:aws:iam::XXXXXXXXXXXX:role/AWSReservedSSO_dsm-account-admin_abcdefg1234567
      username: account-admin
```

Implementation

Date	Environment	Notes	Status
May 19, 2023	Dev		DONE
May 25, 2023	QA		DONE
Jun 10, 2023	Prod		PLANNING

Lesson learnt from previous upgrade

1. Learning from EKS 1.22 upgrade activity
2. Lesson learned from EKS 1.23 Dev upgrade activity
3. Lesson learned from EKS 1.24 Dev & QA upgrade activity

Configuration Changes

- Update the cluster creation pipeline config file
- Update the environment variable library in azure devops
- Commit the changes to Azure Repos for Dev/Test but approvals required for QA and Prod.
- Pipeline gets triggered