**Upgrading EKS Cluster from Version 1.26 to 1.27**

In order to upgrade the EKS cluster version from 1.26 to 1.27 we need to take the backup of EKS resources. Backup of EKS resources will be done using Velero. It will comprehensively back up all resources of the cluster, including namespaces, persistent volumes (PVs), deployments, and other relevant resources.

Though we can able to backup all resources of EKS cluster, we cannot able to backup EKS cluster itself. As we have only one EKS cluster we cannot test and implement. It may cause to downtime during and after upgradation of EKS cluster version so we need to prepare for that.

**Steps to upgrade EKS Cluster version 1.27**

1. **Velero Setup**

[Velero](https://aws.amazon.com/blogs/containers/backup-and-restore-your-amazon-eks-cluster-resources-using-velero/) is a popular open-source tool that can provide Kubernetes cluster disaster recovery, data migration, and data protection. Velero can back up Kubernetes cluster resources and persistent volumes to externally supported storage backend on demand or by schedule.

<https://aws.amazon.com/blogs/containers/backup-and-restore-your-amazon-eks-cluster-resources-using-velero/>

You can refer above link to access more comprehensive information about Velero.

**Velero Deployment Process**

**Prerequisites**

* eksctl v 0.170.0. See [Installing or upgrading eksctl](https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html#installing-eksctl).
* Cluster must be configured with an EKS IAM OIDC Provider. See [Create an IAM OIDC provider for your cluster](https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html). This is a requirement for [IAM roles for service account](https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-service-accounts.html) which is used to grant the required AWS permissions to the Velero deployments.
* Cluster must have the [Amazon EBS CSI driver](https://docs.aws.amazon.com/eks/latest/userguide/ebs-csi.html) installed.
* AWS CLI version 2. See [Installing, updating, and uninstalling the AWS CLI version 2](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html).
* Helm v3. See [Installing Helm](https://helm.sh/docs/intro/install/).
* kubectl. See [Installing kubectl](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html).
* Create s3 bucket for backups.

#### **Create an S3 Bucket to store backups**

aws s3api create-bucket --bucket your-bucket-name --region your-region

In order to attach bucket policy to the bucket, you need to first create JSON file with the bucket policy

Velero uses S3 to store EKS backups when running in AWS.

{

"Version": "2012-10-17",

"Id": "VeleroBucketPolicy",

"Statement": [

{

"Sid": "AllowUserAccess",

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::$ACCOUNTNUMBER:root"

},

"Action": "s3:\*",

"Resource": [

"arn:aws:s3:::$BUCKETNAME",

"arn:aws:s3:::$BUCKETNAME/\*"

]

}

]

}

Replace your-bucket-name with the name of your S3 bucket.

Attach the bucket policy to your bucket: Use the put-bucket-policy command to attach the bucket policy to your bucket:

aws s3api put-bucket-policy --bucket your-bucket-name --policy <file://bucket-policy.json>

Replace your-bucket-name with the name of your S3 bucket.

This command will apply the policy defined in the bucket-policy.json file to the specified S3 bucket.

#### **IAM Policy**

cat > velero\_policy.json <<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"ec2:DescribeVolumes",

"ec2:DescribeSnapshots",

"ec2:CreateTags",

"ec2:CreateVolume",

"ec2:CreateSnapshot",

"ec2:DeleteSnapshot"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:DeleteObject",

"s3:PutObject",

"s3:AbortMultipartUpload",

"s3:ListMultipartUploadParts"

],

"Resource": [

"arn:aws:s3:::${BUCKET}/\*"

]

},

{

"Effect": "Allow",

"Action": [

"s3:ListBucket"

],

"Resource": [

"arn:aws:s3:::${BUCKET}"

]

}

]

}

EOF

aws iam create-policy \ --policy-name VeleroAccessPolicy \ --policy-document <file://velero_policy.json>

#### **Create Service Account for Velero**

eksctl utils associate-iam-oidc-provider --region <REGION> --cluster <CLUSTERNAME> --approve

Replace <CLUSTERNAME> with the name of our EKS cluster.

eksctl create iamserviceaccount \

--cluster=$ CLUSTER\_NAME \

--name=velero-server \

--namespace=velero \

--role-name=eks-velero-backup \

--role-only \

--attach-policy-arn=arn:aws:iam::$ACCOUNT:policy/VeleroAccessPolicy \

--approve

The --namespace=velero flag ensures that only an application running in the velero namespace will be able to access the IAM Policy created in the previous step.

#### **Install Velero in EKS Cluster**

helm repo add vmware-tanzu <https://vmware-tanzu.github.io/helm-charts>

To configure the backup storage location, it's necessary to create a **values.yaml** file.

cat > values.yaml <<EOF

configuration:

backupStorageLocation:

- bucket: $BUCKET

provider: aws

volumeSnapshotLocation:

- config:

region: $REGION

provider: aws

initContainers:

- name: velero-plugin-for-aws

image: velero/velero-plugin-for-aws:v1.7.1

volumeMounts:

- mountPath: /target

name: plugins

credentials:

useSecret: false

serviceAccount:

server:

annotations:

eks.amazonaws.com/role-arn: "arn:aws:iam::${ACCOUNT}:role/eks-velero-backup"

EOF

To update the Kubernetes configuration (**kubeconfig**) file

aws eks --region $REGION update-kubeconfig --name $CLUSTER\_NAME

We can check the contexts with the following command:

kubectl config get-contexts

kubectl config use-context <NAME OF THE CONTEXT>

Now we switched to above context and install Velero

helm install velero vmware-tanzu/velero --version 5.0.2 \ --create-namespace \ --namespace velero \ -f values.yaml

We can check that the Velero server was successfully installed by running this command in the context:

kubectl get pods -n velero

velero version --client-only

**Backup Process**

For backup we need to run below commands that back up created in s3 location whatever we provided earlier

velero backup create <BACKUP NAME>

velero backup describe <BACKUP NAME>

**Download backup file from S3 bucket**

**Delete bucket**

The AWS CLI command to delete an S3 bucket is:

aws s3api delete-bucket --bucket "your-bucket-name"

Replace "your-bucket-name" with the actual name of the bucket you want to delete.

Ensure the bucket is empty before deletion.

**Remove Valero - Uninstalling Velero with Helm:**

To remove Velero entirely from your Kubernetes cluster, use these commands:

# Switch to the namespace where Velero is installed (usually velero):

kubectl config set-context --current --namespace velero

# Delete the Velero namespace (ensure no ongoing Velero operations):

kubectl delete namespace velero

# Optionally, purge the Helm release (if you want to completely remove Velero's history):

helm uninstall velero --purge

1. **Steps to Setup** **Maintenance page**
2. Create a S3 bucket
3. Upload the maintenance html page to the bucket
4. Go to properties of S3 bucket ad enable the static website hosting, add the html file
5. Create a CloudFront distributions with origin as S3 bucket
6. Set the origin access control settings, create new OAC (origin access control)
7. In viewer protocol policy redirect HTTP to HTTPS
8. Enable security protection WAF for security protection
9. Add the alternate domain name (CAME)
10. Choose the custom SSL certificate eg. ACM certificate
11. Enter the default root object
12. Create CloudFront distribution
13. Copy the bucket policy to the S3 bucket
14. Redirect your domain name with CloudFront distribution domain name

# **Check for Deprecated APIs**

Steps to check deprecated APIs

1. Go to EKS console page
2. Select the option EKS upgrade insights
3. The table below lists the insight checks performed by EKS against this cluster, along with their associated statuses. EKS automatically refreshes the status of each Insight, which can be seen in the last refresh time column
4. Currently there are no deprecated APIs
5. **Upgrade the EKS Cluster version**

**Check Current EKS Version**

Use the following command to check existing EKS version:

aws eks describe-cluster --name cluster-name --query "cluster.version"

**Upgrade EKS Cluster Version**

To upgrade the EKS cluster version from 1.26 to 1.27, run

aws eks update-cluster-version --name cluster-name --kubernetes-version 1.27

**Check Upgraded EKS Version**

Use the following command to check upgraded EKS version:

aws eks describe-cluster --name cluster-name --query "cluster.version"

1. **Upgrade Node Group version**

To upgrade the node group version in Amazon EKS cluster we can use eksctl –

Kubectl get node

eksctl upgrade nodegroup \

--name=node-group-name \

--cluster=my-cluster \

--region=region-code

OR After upgrading the EKS cluster version, also need to upgrade the node group to harmonize with the new Kubernetes version

aws eks update-nodegroup-version --cluster-name cluster-name --nodegroup-name nodegroupname --release-version 1.27.5-20231002

1. **Upgrade EKS Add-Ons**

After upgrading the EKS control plane, also upgrade the core deployments and daemon sets that are recommended for EKS 1.27.

To know what latest version of each addons is compatible with EKS cluster version 1.27, run below command

eksctl utils describe-addon-versions --kubernetes-version 1.27 --name name-of-addon | grep AddonVersion

**CoreDNS:**

To know the current version of Coredns run the below command

kubectl get deployment -n kube-system coredns -o=jsonpath='{$.spec.template.spec.containers[0].image}'

Refer the official document [View the documentation here](https://docs.aws.amazon.com/eks/latest/userguide/managing-coredns.html). Upgrade the cluster coredns to a version that is compatible with EKS cluster version.

The current version of Cluster coredns is - coredns: v1.9.3-eksbuild.2 and the latest version compatible with EKS Cluster version 1.27 is v1.9.3-eksbuild.3

Run the following command to upgrade it to the latest version

aws eks update-addon --cluster-name cluster-name --addon-name coredns --addon-version v1.9.3-eksbuild.3 --resolve-conflicts OVERWRITE

kubectl set image deployment.apps/coredns -n kube-system coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.1-eksbuild.6

**Kube-proxy:**

To know the current version of kube -proxy run the below command

kubectl describe pods kube-proxy podame -n kube-system

Refer the official document [View the documentation here](https://docs.aws.amazon.com/eks/latest/userguide/managing-kube-proxy.html). Upgrade the kube-proxy version that is compatible with EKS cluster version.

The current version of Cluster kube-proxy is - kube-proxy:v1.26.2 and the latest version compatible with EKS Cluster version 1.27 is v1.26.4-eksbuild.1

Run the following command to upgrade it to the latest version

aws eks update-addon --cluster-name cluster-name --addon-name kube-proxy --addon-version v1.26.4-eksbuild.1 --resolve-conflicts OVERWRITE

kubectl set image daemonset.apps/kube-proxy -n kube-system kube-proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.26.2-minimal-eksbuild.2

**VPC CNI:**

To know the current version of VPC CNI run the below command

kubectl describe pods podame(aws-node-49jpf) -n kube-system

Refer the official document [More information on managing VPC CNI](https://docs.aws.amazon.com/eks/latest/userguide/managing-vpc-cni.html). Upgrade the VPC CNI version that is compatible with EKS cluster version.

The current version of Cluster VPC CNI is - amazon-k8s-cni-init:v1.12.6-eksbuild.1 and the latest version compatible with EKS Cluster version 1.27 is v1.12.6-eksbuild.2

Run the following command to upgrade it to the latest version

aws eks update-addon --cluster-name cluster-name --addon-name vpc-cni --addon-version v1.12.6-eksbuild.2 --resolve-conflicts OVERWRITE

**AWS EBS CSI Driver:**

To know the current version of ebs-csi-controller run the below command

kubectl get deployment -n kube-system ebs-csi-controller -o=jsonpath='{$.spec.template.spec.containers[0].image}'

The current version of EBS CSI driver is - aws-ebs-csi-driver: v1.18.0 and the latest version compatible with EKS Cluster version 1.27 is v1.19.0-eksbuild.1

Run the following command to upgrade it to the latest version

aws eks update-addon --cluster-name cluster-name --addon-name aws-ebs-csi-driver --addon-version v1.19.0-eksbuild.1 --resolve-conflicts OVERWRITE

**AWS load balancer controller**

To know the current version of Aws-load-balancer-controller run the below command

kubectl get deployment -n kube-system aws-load-balancer-controller -o=jsonpath='{$.spec.template.spec.containers[0].image}'

The current version of AWS load balancer controller is - aws-load-balancer-controller: v2.5.1 and the latest version compatible with EKS Cluster version 1.27 ad Kubernetes dashboard version 2.7.0

To upgrade it to the latest version, run the following command

helm upgrade aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=EKS-EWLEW39B --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller

**Upgrade Cluster Autoscaler:**

To know the current version of Cluster Autoscaler run the below command

kubectl get deployment -n kube-system cluster-autoscaler -o=jsonpath='{$.spec.template.spec.containers[0].image}'

Refer the official document [Kubernetes autoscaler releases to check for the latest version](https://github.com/kubernetes/autoscaler/releases). Upgrade the cluster autoscaler to a version that matches EKS cluster version.

The current version of Cluster Autoscaler is - cluster-autoscaler:v1.26.4 and the latest version compatible with EKS Cluster version 1.27 is v1.27.3

Run the following command to upgrade it to the latest version

kubectl set image deployment cluster-autoscaler -n kube-system cluster-autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:v1.27.4

1. **Perform the sanity test**

1. **Remove Maintenance page**

Steps to remove maintenance page

1. Redirect the domain name to ALB
2. Disable the CloudFront distribution
3. Delete CloudFront distribution
4. Empty the S3 bucket
5. Delete the S3 bucket
6. **Upgrade Kubernetes Dashboard Version**

The current Kubernetes dashboard version is 2.6.1

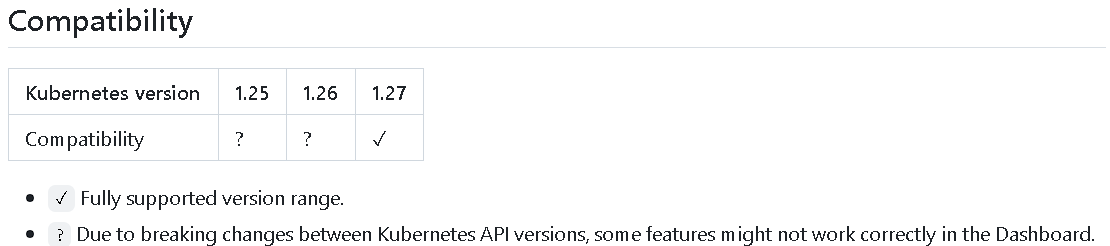
The Current version of EKS API is 1.26

We are upgrading EKS Cluster to v1.27 and k8s dashboard to 2.7.0

**Steps to upgrade Kubernetes Dashboard**

1. **Check Compatibility**

Kubernetes dashboard v3.0.0-alpha0 support Kubernetes version 1.27



**The Kubernetes version 1.27 is compatible with Kubernetes dashboard version v3.0.0-alpha0**

We can use Helm or a single Manifest to install and upgrade Kubernetes Dashboard.

We tried using both approach but getting 400 error for nginx server while accessing dashboard as v3.0.0-alpha0 version is not fully stable version as of now and can’t deploy for the production environment.

1. **Stable Version for Kubernetes Dashboard**

Current stable version for Kubernetes Dashboard is v2.7.0

Installing dashboard on EKS:

<https://github.com/kubernetes/dashboard/releases>

# Add kubernetes-dashboard repository

**helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/**

# Upgrade kubernetes-dashboard version from v2.6.1 to v2.7.0

**helm upgrade --install my-kubernetes-dashboard k8s-dashboard/kubernetes-dashboard --version 6.0.8 -n kube-dashboard --values my-values-270.yaml**

1. **Verification**

After upgrading, verify the Kubernetes dashboard version using following command and access URL:

**Kubectl get deployments -n kube-dashboard**

**kubectl get deployment my-kubernetes-dashboard -n kube-dashboard**

**kubectl get deployment my-kubernetes-dashboard-metrics-server -n kube-dashboard**

1. **Finally, we need to do Sanity test of all workloads in EKS.**