Upgrading AWS EKS Cluster with zero downtime

Introduction:

Welcome to your tailored guide for upgrading your AWS Elastic Kubernetes Service (EKS) cluster to the latest version without any downtime.

As a DevOps professional, you recognize the importance of maintaining up-to-date systems while minimizing service interruptions. This comprehensive guide will lead you through the process of updating the EKS control plane, add-ons, and the Kubernetes version on your worker nodes, ensuring compatibility with your applications.

Even if you set the desired, minimum, and maximum number of nodes to one, we will ensure your managed node groups in EKS are updated effectively and efficiently.

Upgrade your system smoothly with three simple steps:

1. **EKS Version Upgrade:** Easily upgrade your EKS version through:
   - **Terraform:** This is our recommended method for a hassle-free upgrade.
   - **CLI:** A straightforward command line option.
   - **Dashboard**: Visual and user-friendly.

2. **Add-ons Update**: Keep your add-ons current with:
   - **Terraform:** Effortlessly update your add-ons one minor version at a time. It's designed to keep your upgrades in sync with our latest Kubernetes version, ensuring compatibility and performance.
   - **CLI:** Direct and simple.
   - **Dashboard:** Proven reliability through successful testing.

3. **Cluster Nodes Enhancement:**
   - **CLI:** Reliable and tested for success.
   - **Dashboard**: Note that this method will restart your running pods on the upgraded nodes, which may lead to brief downtime.

Choose your preferred upgrade method and keep your systems performing at their best!

Prerequisite:

- An existing AWS EKS cluster.
- AWS CLI installed and configured with appropriate permissions.
- kubectl command-line tool configured to communicate with your EKS cluster.

Backup & Preparation:

1. **Backup Critical Data**: Ensure that you have backups of critical data, including application state and configuration.

2. **RDS Backups**: Use AWS RDS automated backups and snapshots feature to back up your database. You can define backup retention policies and also take manual snapshots.

3. **Cluster and Application Configuration:**

Although Terraform manages infrastructure, it doesn't back up Kubernetes-specific configurations like resource definitions (deployments, services, configmaps, etc.). You should still back these up as described previously, e.g., using Velero or manual kubectl commands.

Resource Definitions: Use kubectl get for all relevant resources (deployments, services, configmaps, secrets (ensure to handle secrets securely), etc.) and save them:

```
kubectl get all --all-namespaces -o yaml > all-deployments.yaml
```

Automate Backups: Consider automating this process with cron jobs or other scheduling tools.

4. **Review Release Notes:** Check the AWS EKS Release Notes for any breaking changes or new features in the latest version.

5. **Test in a Staging Environment:** Test the upgrade process in a staging environment first.

Upgrade the EKS Control Plane:

Using Terraform

1) Set the "*cluster_version*" value to the new version.
   In my demo, I will upgrade my EKS cluster from version 1.28 to 1.29
   Most important we can update the EKS version up to one minor version at a time.

```
module "eks" {

  source  = "terraform-aws-modules/eks/aws"

  cluster_version              = var.eks_version

...
}
```

2) Update Addons

```
module "eks" {
    source  = "terraform-aws-modules/eks/aws"
      ....

      cluster_addons = {
         coredns = {
           most_recent = true
         }
         kube-proxy = {
           most_recent = true
         }
         vpc-cni = {
           most_recent = true
         }
       }
    ...
}
```

To upgrade the addons to the latest version set the value of "*most_recent*" to "*true*" for each addon.

In the following links, you can find table lists with the latest version of the Amazon EKS add-on type for each Kubernetes version:

VPC-CNI: https://docs.aws.amazon.com/eks/latest/userguide/managing-vpc-cni.html#vpc-add-on-update
kube-proxy: https://docs.aws.amazon.com/eks/latest/userguide/managing-kube-proxy.html
CoreDNS: https://docs.aws.amazon.com/eks/latest/userguide/managing-coredns.html

3) Upgrade EKS - We are ready to upgrade our EKS cluster now
   * I recommend initializing our working directory using "*terraform init*"
   * execute "*terraform plan*" followed by *"terraform apply"*.

Here is the output from my "*terraform plan*"

```
 # module.eks-cluster.module.eks.data.aws_eks_addon_version.this["coredns"] will be read during apply
 # (depends on a resource or a module with changes pending)
 <= data "aws_eks_addon_version" "this" {
     + addon_name        = "coredns"
     + id                = (known after apply)
     + kubernetes_version = "1.29"
     + version           = (known after apply)
   }

 # module. eks-cluster.module.eks.data.aws_eks_addon_version.this["kube-proxy"] will be read during apply
 # (depends on a resource or a module with changes pending)
 <= data "aws_eks_addon_version" "this" {
     + addon_name        = "kube-proxy"
     + id                = (known after apply)
     + kubernetes_version = "1.29"
     + most_recent       = true
     + version           = (known after apply)
   }

 # module. eks-cluster.module.eks.data.aws_eks_addon_version.this["vpc-cni"] will be read during apply
 # (depends on a resource or a module with changes pending)
 <= data "aws_eks_addon_version" "this" {
     + addon_name        = "vpc-cni"
     + id                = (known after apply)
     + kubernetes_version = "1.29"
     + version           = (known after apply)
   }
```

```
# module.eks-cluster.module.eks.aws_eks_addon.this["coredns"] will be updated in-place
~ resource "aws_eks_addon" "this" {
    ~ addon_version     = "v1.10.1-eksbuild.4" -> (known after apply)
      id                = "my-cluster:coredns"
      tags              = {
          "environment" = "staging"
          "terraform"   = "true"
      }
      # (7 unchanged attributes hidden)

      # (1 unchanged block hidden)
  }

# module.eks-cluster.module.eks.aws_eks_addon.this["kube-proxy"] will be updated in-place
~ resource "aws_eks_addon" "this" {
    ~ addon_version     = "v1.28.8-eksbuild.2" -> (known after apply)
      id                = "my-cluster:kube-proxy"
      tags              = {
          "environment" = "staging"
          "terraform"   = "true"
      }
      # (7 unchanged attributes hidden)

      # (1 unchanged block hidden)
  }

# module.eks-cluster.module.eks.aws_eks_addon.this["vpc-cni"] will be updated in-place
~ resource "aws_eks_addon" "this" {
    ~ addon_version     = "v1.15.1-eksbuild.1" -> (known after apply)
      id                = "my-cluster:vpc-cni"
      tags              = {
          "environment" = "staging"
          "terraform"   = "true"
      }
      # (7 unchanged attributes hidden)

      # (1 unchanged block hidden)
  }

# module.eks-cluster.module.eks.aws_eks_cluster.this[0] will be updated in-place
~ resource "aws_eks_cluster" "this" {
      id                    = "my-cluster"
      name                  = "my-cluster"
      tags                  = {
          "environment" = "staging"
          "terraform"   = "true"
      }
    ~ version               = "1.28" -> "1.29"
```

```
    # (10 unchanged attributes hidden)

    # (5 unchanged blocks hidden)
}
```

It takes about 10 minutes for the cluster to be upgraded.

Using AWS CLI or AWS Dashboard:

Upgrade the EKS Control Plane:

1. Check the current version: First, determine the current version of your EKS cluster. You can do this using the AWS Management Console or via the AWS CLI with the command:
   `aws eks describe-cluster --name <cluster_name> --query "cluster.version"`
2. **Update Control Plane:** Upgrade to the latest version of the EKS control plane. This can be accomplished through the AWS Management Console or by executing the following AWS CLI command:
   `aws eks update-cluster-version --name <cluster_name> --kubernetes-version <new_version>`
3. **Verify Update:** After the update, confirm the new version of the control plane using the AWS CLI:
   `aws eks describe-cluster --name <cluster_name> --query "cluster.version"`

Upgrade Add-Ons Using AWS Dashboard:

Following the manual upgrade of the EKS control plane, proceed to manually upgrade the add-ons using the AWS Dashboard. This method is straightforward and has been successfully tested, ensuring that your add-ons are kept up-to-date seamlessly.

Update Managed Node Groups:

Managed node groups configure Amazon EC2 instances in your account. When your control plane is updated, either by you or Amazon EKS, these instances aren't automatically upgraded. Here's how to manage this process:

Via AWS Dashboard: When updating the node group through the AWS Dashboard, a new node is created, and pods transition from the old node to the new one. This is done by stopping the old pod and starting a new one on the new node, which may result in some downtime.
Via AWS CLI (Recommended for Zero Downtime): To ensure a seamless upgrade with zero downtime, use the AWS CLI. This method replaces the old nodes with new ones running the latest version without disrupting your operations.

Steps to Upgrade Node Groups:

1. Prepare a List of All Node Groups: Get a list of all node groups in your Kubernetes cluster:

```
aws eks list-nodegroups --cluster-name <cluster_name>
```

2. Get the Node Group's Kubernetes Version: Retrieve detailed information about a specific node group:

```
aws eks describe-nodegroup --cluster-name <cluster_name> \
--nodegroup-name <nodegroup-name> \
--query "nodegroup.{nodegroupName: nodegroupName, nodegroupArn: nodegroupArn,
clusterName: clusterName, version: version, releaseVersion: releaseVersion}" \
--output json
```

3. Update Node Group: Initiate an update for each node group. This command gracefully replaces old nodes with new ones running the updated version:

```
aws eks update-nodegroup-version --cluster-name <cluster_name> \ --nodegroup-name
<nodegroup_name> --kubernetes-version <latest_version>
```

4. Monitor the Update Progress: Use the AWS Management Console or AWS CLI to track the update progress of the node groups.

Following these steps will help you manage your node groups effectively, ensuring they run the most current versions and maintain optimal performance with minimal disruption.

Validate the Upgrade:

Proper validation is essential after upgrading our Kubernetes environment. Follow these steps to ensure everything is functioning correctly:

1. Check Cluster Health:

- **Get Cluster Information:** Use **kubectl cluster-info** to obtain a summary of the cluster, including Kubernetes version and API URLs.
- **Get Nodes Status:** Check the status of each node with **kubectl get nodes**.
- **Detailed Node Information:** For more in-depth details about a specific node, use:

```
kubectl describe node <node-name>
```

2. **Check Workloads and Resources:**

- **Check All Namespaces:** This command displays all the resources across all namespaces, offering a quick overview:

```
kubectl get all --all-namespaces
```

•**Check Deployments:** Ensure that your deployments are operational:

```
kubectl get deployments --all-namespaces
```

•**Check Pods' Status:** Useful for identifying any pods that may be failing or restarting frequently:

```
kubectl get pods --all-namespaces
```

3. **Check Events and Logs:**

•**Cluster Events**: View recent cluster events, including errors and other critical notifications:

```
kubectl get events --all-namespaces
```

•**Pod Logs:** For any problematic pods, examine their logs to diagnose issues:

```
kubectl logs <pod-name> -n <namespace>
```

4. **Additional Health Checks:**

•**Resource Usage**: If the metrics-server is installed, check the CPU and memory usage of nodes and pods:

```
kubectl top nodes kubectl top nodes -l role=general

# for nodes labeled 'general'

 kubectl top pods --all-namespaces
```

•**Check Services:** Verify that all services are correctly configured and operational:

```
kubectl get services --all-namespaces
```

5. **Validate Applications:**

Ensure that your applications are functioning as expected. Perform tests on their functionalities to confirm they are behaving correctly post-upgrade.