EKS Upgrade to (X.XX)

This is a generic doc on how to upgrade the EKS cluster. The steps here are references from the AWS. Please note each version of EKS may require different steps as the version progress.

Amazon EKS Kubernetes release calendar

https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html

Kubernetes version	Upstream release	Amazon EKS release	Amazon EKS end of support
1.18	March 23, 2020	October 13, 2020	March 31, 2022
1.19	August 26, 2020	February 16, 2021	August 1, 2022
1.20	December 8, 2020	May 18, 2021	October 3, 2022
1.21	April 8, 2021	July 19, 2021	February 2023
1.22	August 4, 2021	April 4, 2022	May 2023
1.23	December 7, 2021	August 2022	October 2023

EKS Upgrade Links

AWS upgrade reference: https://docs.aws.amazon.com/eks/latest/userguide/update-cluster.html

EKS Kubernetes versions: https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html

kube-proxy add-on: https://docs.aws.amazon.com/eks/latest/userguide/managing-kube-proxy.html

CoreDNS add-on: https://docs.aws.amazon.com/eks/latest/userguide/managing-coredns.html

VPC CNI add-on: https://docs.aws.amazon.com/eks/latest/userguide/managing-vpc-cni.html

EKS Upgrade Steps

We have 5 steps:

- Upgrade the Control Plane
- Patch kubeproxy
- Patch CoreDNS
- Patch AWS CNI
- Flip the nodes

Step 01: Upgrade the control plane

- Set EKS control plane to version X.XX in terraform resource "eks-control-plane",
- Check your AMI
- Run terraform plan
- Apply the changes
- You can verify progress and or process in the AWS console , cluster should be marked as updating

Step 02: Control plane verification

1. Check the Pod security policy or Verify privileged PSP exist.

Command should not return an error if so please see. If it return an error, move to step 2

https://docs.aws.amazon.com/eks/latest/userguide/pod-security-policy.html#default-psp

By default, the pod security policy admission controller is enabled on Amazon EKS clusters. Before updating your cluster, ensure that the proper pod security policies are in place. This is to avoid potential security issues. You can check for the default policy with the `kubectl get psp eks.privileged` command.

```
kubectl get psp eks.privileged
```

Output:

```
NAME              PRIV   CAPS   SELINUX    RUNASUSER   FSGROUP
SUPGROUP   READONLYROOTFS   VOLUMES
eks.privileged   true   *      RunAsAny   RunAsAny    RunAsAny
RunAsAny   false            *
```

If you receive the following error, move to step 2

```
Error from server (NotFound): podsecuritypolicies.extensions "eks.
privileged" not found
```

1. If you originally deployed your cluster on Kubernetes

`1.17` or earlier, you might need to remove a discontinued term from your CoreDNS manifest.

https://docs.aws.amazon.com/eks/latest/userguide/update-cluster.html

a. Check to see if your CoreDNS manifest has a line that only has the word `upstream`.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.
Corefile}' | grep upstream
```

If no output is returned, this means that your manifest doesn't have the line with `upstream`. If this is the case, skip to the next step. If the word up `stream` is returned, remove the line.

b. Remove the line near the top of the file that only has the word `upstream` in the configmap file. Don't change anything else in the file. After the line is removed, save the changes.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

Step 03: Patch kubeproxy

https://docs.aws.amazon.com/eks/latest/userguide/managing-kube-proxy.html

- Check the **image version for each Amazon EKS supported cluster version**

kube-proxy **image version for each Amazon EKS supported cluster version**

Kubernetes version	1.22	1.21	1.20	1.19	1.18	1.17
kube-proxy (default version)	1.22.6-eksbuild.1	1.21.2-eksbuild.2	1.20.4-eksbuild.2	1.19.6-eksbuild.2	1.18.8-eksbuild.1	1.17.9-eksbuild.1
kube-proxy (minimal)	1.22.6-minimal-eksbuild.2	1.21.9-minimal-eksbuild.2	1.20.15-minimal-eksbuild.2	1.19.16-minimal-eksbuild.2	1.18.20-minimal-eksbuild.1	1.17.17-minimal-eksbuild.1

- Make sure to match version listed in chart on the AWS upgrade doc , example below is from version **1.19 to version 1.20**
  - Verify

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

  - Output

```
602401143452.dkr.ecr.us-east-2.amazonaws.com/eks/kube-proxy:v1.19.6-eksbuild.2
```

  - Update , make sure to set the region correctly. This will change the kube-proxy image from **1.19 to 1.20**

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-proxy=602401143452.dkr.ecr.us-east-2.amazonaws.com/eks/kube-proxy:v1.20.4-eksbuild.2
```

- Verify if the kube-proxy pods are running in kube-system ns

```
k get po -n kube-system |grep kube-proxy
```

Step 04: Patch CoreDNS

https://docs.aws.amazon.com/eks/latest/userguide/managing-coredns.html

CoreDNS version deployed with each Amazon EKS supported cluster version

Kubernetes version	1.22	1.21	1.20	1.19	1.18	1.17
CoreDNS	1.8.7	1.8.4	1.8.3	1.8.0	1.7.0	1.6.6

- Patch CoreDNS, make sure to match version listed on the AWS upgrade doc, example below is from version **1.19 to version 1.20**
  - Verify (Check the CoreDNS version)

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

Output:

```
coredns:v1.8.0
```

- Update the CoreDNS (make sure to set the region correctly)

```
kubectl set image --namespace kube-system deployment.apps/coredns \
    coredns=602401143452.dkr.ecr.[region_name].amazonaws.com/eks
/coredns:v1.8.3-eksbuild.1

kubectl set image --namespace kube-system deployment.apps/coredns \
    coredns=602401143452.dkr.ecr.us-east-1.amazonaws.com/eks
/coredns:v1.8.3-eksbuild.1
```

- Edit the cluster role and add the below content at the end

```
kubectl edit clusterrole system:coredns -n kube-system


- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
```

- Verify if the coredns pods are running in kube-system ns

```
k get po -n kube-system |grep coredns
```

Step 05: Patch AWS CNI

https://docs.aws.amazon.com/eks/latest/userguide/managing-vpc-cni.html

Recommended version of the Amazon VPC CNI add-on for each cluster version						
	1.22	1.21	1.20	1.19	1.18	1.17
Add-on version	1.11.0-eksbuild.1	1.11.0-eksbuild.1	1.11.0-eksbuild.1	1.11.0-eksbuild.1	1.11.0-eksbuild.1	1.11.0-eksbuild.1

- Patch AWS CNI, make sure to match version listed in chart on the AWS upgrade doc , example below is from version **1.19 to version 1.20**
  - Verify (Check the AWS CNI version)

```
kubectl describe daemonset aws-node --namespace kube-system | grep
Image | cut -d "/" -f 2
```

    Example output:

```
amazon-k8s-cni-init:v1.11.0-eksbuild.1
amazon-k8s-cni:v1.11.0-eksbuild.1
```

  - Update the AWS CNI (make sure to set the region correctly)
  - https://github.com/aws/amazon-vpc-cni-k8s/releases ---->>>> Download version `1.10` of AWS CNI

```
## Get YAML
curl -o aws-k8s-cni.yaml https://raw.githubusercontent.com/aws
/amazon-vpc-cni-k8s/release-1.10/config/master/aws-k8s-cni.yaml

## Set Region
sed -i.bak -e 's/us-west-2/[region-code]/' aws-k8s-cni.yaml
sed -i.bak -e 's/us-west-2/us-east-1/' aws-k8s-cni.yaml

## Apply yaml
kubectl apply -f aws-k8s-cni.yaml

## Verify the update:
kubectl describe daemonset aws-node -n kube-system | grep Image |
cut -d "/" -f 2
amazon-k8s-cni-init:v1.11.0-eksbuild.1
amazon-k8s-cni:v1.11.0-eksbuild.1

## Check pods
kubectl get daemonset aws-node -n kube-system
```

- At this point all "aws-node", "kube-proxy", "coredns" pods should all be running and healthy with the new control plane but on the old nodes.

Step 06: Update Node groups

- Update Node groups and flip to be the new version