**Install the EKS cluster with the 1.17 version**

=============================================

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

sudo mv -v /tmp/eksctl /usr/local/bin

eksctl version

**Install Kubectl**

===============

curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/1.14.6/2019-08-22/bin/linux/amd64/kubectl

chmod +x ./kubectl

mkdir -p $HOME/bin

cp ./kubectl $HOME/bin/kubectl

export PATH=$HOME/bin:$PATH

echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc

source $HOME/.bashrc

kubectl version --short –client

**Create an eksctl deployment file (eksworkshop.yaml) use in creating your cluster using the following syntax:**

cat eksworkshop.yaml

---

apiVersion: eksctl.io/v1alpha5

kind: ClusterConfig

metadata:

name: eksworkshop-eksctl

region: "us-east-1"

version: "1.17"

availabilityZones: ["us-east-1a", "us-east-1b"]

**Create the eks cluster using below command:**

[root@ip-172-31-21-212 ~]# eksctl create cluster -f eksworkshop.yaml

2021-03-31 09:56:01 [ℹ] eksctl version 0.42.0

2021-03-31 09:56:01 [ℹ] using region us-east-1

2021-03-31 09:56:01 [ℹ] subnets for us-east-1a - public:192.168.0.0/19 private:192.168.64.0/19

2021-03-31 09:56:01 [ℹ] subnets for us-east-1b - public:192.168.32.0/19 private:192.168.96.0/19

2021-03-31 09:56:01 [ℹ] using Kubernetes version 1.17

2021-03-31 09:56:01 [ℹ] creating EKS cluster "eksworkshop-eksctl" in "us-east-1" region with

2021-03-31 09:56:01 [ℹ] will create a CloudFormation stack for cluster itself and 0 nodegroup stack(s)

2021-03-31 09:56:01 [ℹ] will create a CloudFormation stack for cluster itself and 0 managed nodegroup stack(s)

2021-03-31 09:56:01 [ℹ] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=eksworkshop-eksctl'

2021-03-31 09:56:01 [ℹ] CloudWatch logging will not be enabled for cluster "eksworkshop-eksctl" in "us-east-1"

2021-03-31 09:56:01 [ℹ] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=eksworkshop-eksctl'

2021-03-31 09:56:01 [ℹ] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "eksworkshop-eksctl" in "us-east-1"

2021-03-31 09:56:01 [ℹ] 2 sequential tasks: { create cluster control plane "eksworkshop-eksctl", wait for control plane to become ready }

2021-03-31 09:56:01 [ℹ] building cluster stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 09:56:02 [ℹ] deploying stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 09:56:32 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 09:57:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 09:58:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 09:59:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:00:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:01:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:02:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:03:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:04:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:05:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:06:02 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:07:03 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:08:03 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:09:03 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:10:03 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:11:03 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-cluster"

2021-03-31 10:11:03 [ℹ] waiting for the control plane availability...

2021-03-31 10:11:03 [✔] saved kubeconfig as "/root/.kube/config"

2021-03-31 10:11:03 [ℹ] no tasks

2021-03-31 10:11:03 [✔] all EKS cluster resources for "eksworkshop-eksctl" have been created

2021-03-31 10:11:05 [ℹ] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'

2021-03-31 10:11:05 [✔] EKS cluster "eksworkshop-eksctl" in "us-east-1" region is ready

eksctl utils associate-iam-oidc-provider \

--region us-east-1 \

--cluster eksworkshop-eksctl \

--approve

**Create node-group/worker node**

====================================

eksctl create nodegroup --cluster=eksworkshop-eksctl\

--region=us-east-1 \

--name=eksworkshop-eksctl1-ng-public \

--node-type=t2.micro \

--nodes=2 \

--nodes-min=2 \

--nodes-max=4 \

--node-volume-size=10 \

--ssh-access \

--ssh-public-key=eks \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access

kubectl get nodes # if we see our 3 nodes, we know we have authenticated correctly

[root@ip-172-31-87-198 ~]# kubectl get nodes

Amazon EKS doesn't modify any of your Kubernetes add-ons when you update a cluster. Below is the matrix for each add-on component and its version.

| **Kubernetes version** | **1.19** | **1.18** | **1.17** | **1.16** | **1.15** |
| --- | --- | --- | --- | --- | --- |
| Amazon VPC CNI plug-in | 1.7.5 | 1.7.5 | 1.7.5 | 1.7.5 | 1.7.5 |
| DNS (CoreDNS) | 1.8.0 | 1.7.0 | 1.6.6 | 1.6.6 | 1.6.6 |
| KubeProxy | 1.19.6 | 1.18.8 | 1.17.9 | 1.16.13 | 1.15.11 |

**EKS upgrade process:**

**===================================**

**To update an existing cluster**

Update the cluster and Kubernetes add-ons.

1. Compare the Kubernetes version of your cluster control plane to the Kubernetes version of your nodes.

[root@ip-172-31-87-198 ~]# kubectl version --short

Client Version: v1.14.7-eks-1861c5

Server Version: v1.17.12-eks-7684af

[root@ip-172-31-87-198 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-192-168-25-47.ec2.internal Ready <none> 105s v1.17.12-eks-7684af

ip-192-168-53-162.ec2.internal Ready <none> 91s v1.17.12-eks-7684af

1. The pod security policy admission controller is enabled by default on Amazon EKS clusters. Before updating your cluster, ensure that the proper pod security policies are in place before you update to avoid any issues.

[root@ip-172-31-21-212 ~]# kubectl get psp eks.privileged

NAME PRIV CAPS SELINUX RUNASUSER FSGROUP SUPGROUP READONLYROOTFS VOLUMES

eks.privileged true \* RunAsAny RunAsAny RunAsAny RunAsAny false \*

1. If you originally deployed your cluster on Kubernetes 1.17 or earlier, then you may need to remove a discontinued term from your CoreDNS manifest.
2. Check to see if your CoreDNS manifest has a line that only has the word upstream.

[root@ip-172-31-21-212 ~]# kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' | grep upstream

Upstream

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

[root@ip-172-31-21-212 ~]# kubectl edit configmap coredns -n kube-system -o yaml

apiVersion: v1

data:

Corefile: |

.:53 {

errors

health

kubernetes cluster.local in-addr.arpa ip6.arpa {

pods insecure

fallthrough in-addr.arpa ip6.arpa

}

prometheus :9153

forward . /etc/resolv.conf

cache 30

loop

reload

loadbalance

}

kind: ConfigMap

metadata:

annotations:

kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","data":{"Corefile":".:53 {\n errors\n health\n kubernetes cluster.local in-addr.arpa ip6.arpa {\n pods insecure\n upstream\n fallthrough in-addr.arpa ip6.arpa\n }\n prometheus :9153\n forward . /etc/resolv.conf\n cache 30\n loop\n reload\n loadbalance\n}\n"},"kind":"ConfigMap","metadata":{"annotations":{},"labels":{"eks.amazonaws.com/component":"coredns","k8s-app":"kube-dns"},"name":"coredns","namespace":"kube-system"}}

creationTimestamp: "2021-03-31T10:06:28Z"

labels:

eks.amazonaws.com/component: coredns

k8s-app: kube-dns

name: coredns

namespace: kube-system

resourceVersion: "11407"

selfLink: /api/v1/namespaces/kube-system/configmaps/coredns

uid: 9cee74f2-3535-4b70-8eba-efd02b7001be

[root@ip-172-31-21-212 ~]# kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' | grep upstream

1. **Update your cluster using eksctl, the AWS Management Console, or the AWS CLI.**

eksctl upgrade cluster --name eksworkshop-eksctl --approve

**OR**

aws eks update-cluster-version \

--region us-east-1 \

--name eksworkshop-eksctl \

--kubernetes-version 1.18

Output:

{

"update": {

"status": "InProgress",

"errors": [],

"params": [

{

"type": "Version",

"value": "1.18"

},

{

"type": "PlatformVersion",

"value": "eks.4"

}

],

"type": "VersionUpdate",

"id": "24f153a0-fd13-4f3d-b0fe-3fd73268b2e5",

"createdAt": 1617255187.851

}

}

**To check the upgrade status:**

aws eks describe-update \

--region us-east-1 \

--name eksworkshop-eksctl \

--update-id "24f153a0-fd13-4f3d-b0fe-3fd73268b2e5"

Once upgrade is done below will be output:

aws eks describe-update --region us-east-1 --name eksworkshop-eksctl --update-id "24f153a0-fd13-4f3d-b0fe-3fd73268b2e5"

{

"update": {

"status": "Successful",

"errors": [],

"params": [

{

"type": "Version",

"value": "1.18"

},

{

"type": "PlatformVersion",

"value": "eks.4"

}

],

"type": "VersionUpdate",

"id": "24f153a0-fd13-4f3d-b0fe-3fd73268b2e5",

"createdAt": 1617255187.851

}

}

1. **Patch the kube-proxy daemonset to use the image that corresponds to your cluster's Region and current Kubernetes version (in this example, 1.18.8).**

*A} First, retrieve your current kube-proxy image:*

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

602401143452.dkr.ecr.us-east-1.amazonaws.com/eks/kube-proxy:v1.17.9-eksbuild.1

B} Update kube-proxy to the recommended version

# kubectl set image daemonset.apps/kube-proxy \

-n kube-system \

kube-proxy=602401143452.dkr.ecr.us-east-1.amazonaws.com/eks/kube-proxy:v1.18.8-eksbuild.1

daemonset.apps/kube-proxy image updated

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

602401143452.dkr.ecr.us-east-1.amazonaws.com/eks/kube-proxy:v1.18.8-eksbuild.1

1. **Check the current version of your cluster's coredns deployment.**

# kubectl get deployment coredns --namespace kube-system -o=jsonpath='{$.spec.template.spec.containers[:1].image}'

602401143452.dkr.ecr.us-east-1.amazonaws.com/eks/coredns:v1.6.6-eksbuild.1

# kubectl set image --namespace kube-system deployment.apps/coredns \

coredns=602401143452.dkr.ecr.us-east-1.amazonaws.com/eks/coredns:v1.7.0-eksbuild.1

deployment.apps/coredns image updated

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

coredns:v1.7.0-eksbuild.1

1. **Check the version of your cluster's Amazon VPC CNI Plugin for Kubernetes. Use the following command to print your cluster's CNI version.**

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

amazon-k8s-cni-init:v1.7.5-eksbuild.1

amazon-k8s-cni:v1.7.5-eksbuild.1

The version should be greater than or equal to 1.7.5

kubectl convert -f ./nginx-deployment.yaml --output-version apps/v1

[root@ip-172-31-87-198 ~]# aws eks describe-cluster \

> --region us-east-1 \

> --name eksworkshop-eksctl \

> --query 'cluster.endpoint' \ggj j k

> --output text

<https://10F15B5D1961EE2BF795D57ECB33137F.gr7.us-east-1.eks.amazonaws.com>

Migrating to a new node group

[root@ip-172-31-87-198 ~]# wget <https://github.com/weaveworks/eksctl/releases/download/0.43.0-rc.0/eksctl_Linux_amd64.tar.gz> .

[root@ip-172-31-87-198 ~]# ./eksctl version

0.43.0-rc.0

[root@ip-172-31-87-198 ~]# mv ./eksctl /usr/local/bin

mv: overwrite ‘/usr/local/bin/eksctl’? y

[root@ip-172-31-87-198 ~]# eksctl version

0.43.0-rc.0

2021-03-31 14:27:31 [ℹ] eksctl version 0.43.0-rc.0

2021-03-31 14:27:31 [ℹ] using region us-east-1

CLUSTER NODEGROUP STATUS CREATED MIN SIZE MAX SIZE DESIRED CAPACITY INSTANCE TYPE IMAGE ID ASG NAME

eksworkshop-eksctl eksworkshop-eksctl1-ng-public CREATE\_COMPLETE 2021-03-31T12:05:56Z 2 4 2 t2.micro eks-62bc4416-d3d1-ac8e-d995-a5a011642aa4

1. **Retrieve the name of your existing node groups, replacing <my-cluster> (including <>) with your cluster name**

# eksctl get nodegroups --cluster=eksworkshop-eksctl --region=us-east-1

2021-04-01 06:27:08 [ℹ] eksctl version 0.43.0

2021-04-01 06:27:08 [ℹ] using region us-east-1

**CLUSTER NODEGROUP STATUS CREATED MIN SIZE MAX SIZE DESIRED CAPACITY INSTANCE TYPE IMAGE ID ASG NAME**

eksworkshop-eksctl eksworkshop-eksctl1-ng-public CREATE\_COMPLETE 2021-04-01T05:20:30Z 2 4 2 t2.micro eks-76bc45f0-8054-562e-4ed1-7eaf213782bd

1. **Launch a new node group with eksctl with the following command**,**for 1.18 version**

eksctl create nodegroup \

--cluster eksworkshop-eksctl\

--region=us-east-1 \

--version 1.18 \

--name eksworkshop-eksctl1-new \

--node-type t2.medium \

--nodes 2 \

--nodes-min 2 \

--nodes-max 4 \

--ssh-access \

--ssh-public-key=eks \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access

1. **Verify the status of all the nodes**

[root@ip-172-31-48-49 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-192-168-10-72.ec2.internal Ready <none> 79m v1.17.12-eks-7684af

ip-192-168-21-144.ec2.internal Ready <none> 2m37s v1.18.9-eks-d1db3c

ip-192-168-43-121.ec2.internal Ready <none> 79m v1.17.12-eks-7684af

ip-192-168-51-56.ec2.internal Ready <none> 2m39s v1.18.9-eks-d1db3c

1. **Update the security groups for both node groups so that they can communicate with each other**

oldNodes="eksctl-eksworkshop-eksctl-nodegroup-eksworkshop-eksctl1-ng-public"

newNodes="eksctl-eksworkshop-eksctl-nodegroup-eksworkshop-eksctl1-new"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \

--query 'StackResources[?ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \

--output text)

newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \

--query 'StackResources[?ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \

--output text)

Add ingress rules to each node security group so that they accept traffic from each other.

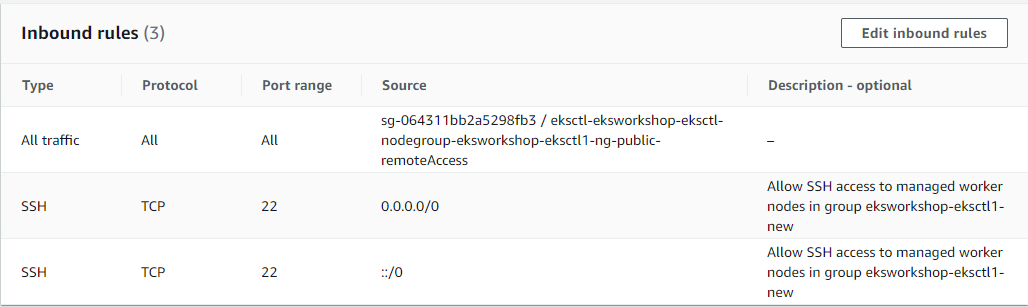
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \

--source-group $newSecGroup --protocol -1

aws ec2 authorize-security-group-ingress --group-id $newSecGroup \

--source-group $oldSecGroup --protocol -1

First rule will be added.



# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-192-168-10-72.ec2.internal Ready <none> 109m v1.17.12-eks-7684af

ip-192-168-21-144.ec2.internal Ready <none> 32m v1.18.9-eks-d1db3c

ip-192-168-43-121.ec2.internal Ready <none> 109m v1.17.12-eks-7684af

ip-192-168-51-56.ec2.internal Ready <none> 32m v1.18.9-eks-d1db3c

1. **Check the aws-auth configmap to map the new node instance role in RBAC.**

Crosscheck if the new roles also added to the aws-auth or not. If no then add it for new node group as well.

You can get the IAM role info using below command.

# ***eksctl get iamidentitymapping --cluster eksworkshop-eksctl --region us-east-1***

2021-04-01 08:30:15 [ℹ] eksctl version 0.43.0

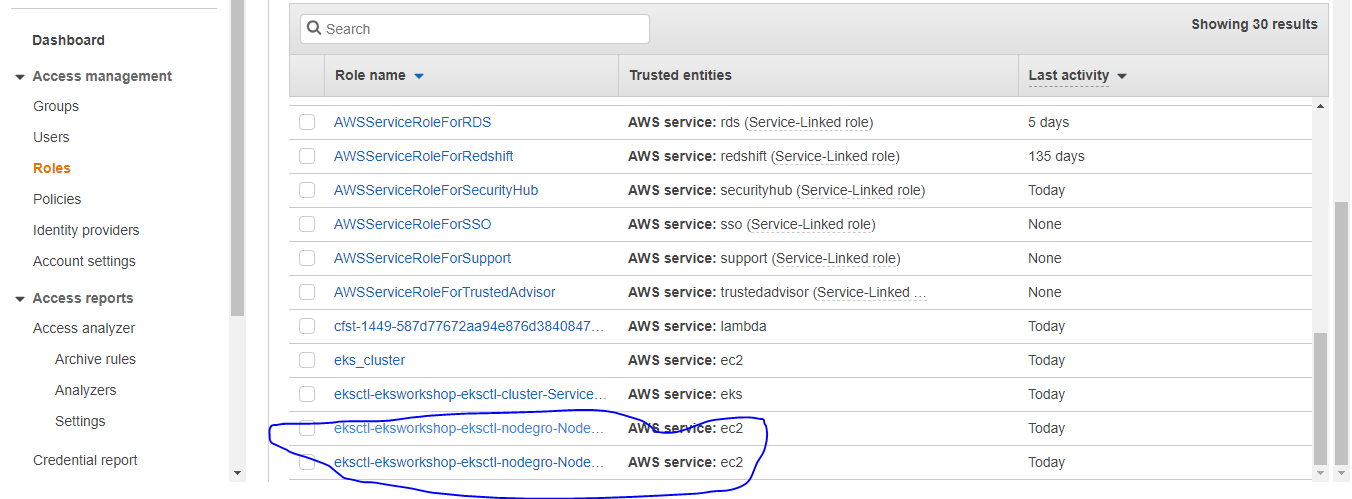
2021-04-01 08:30:15 [ℹ] using region us-east-1

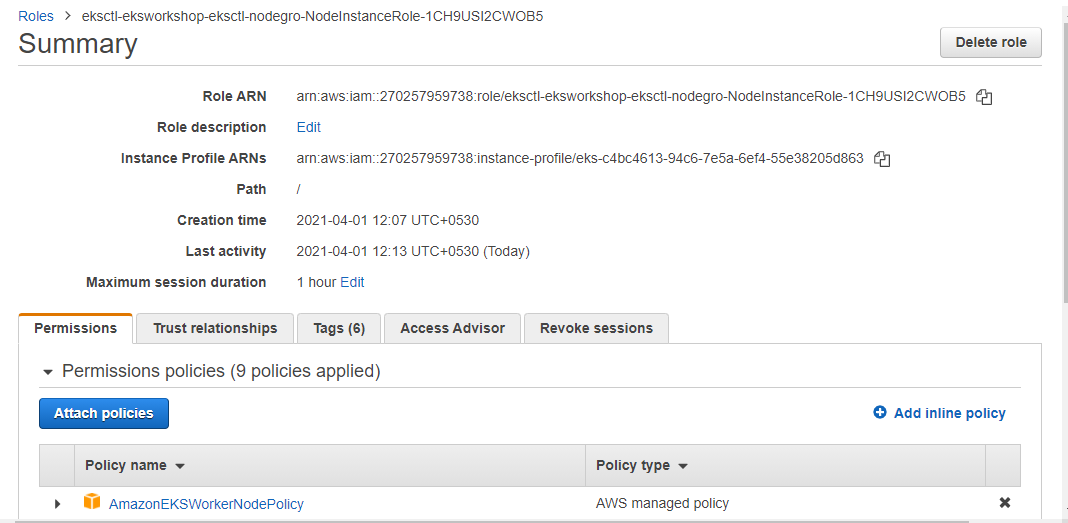
ARN USERNAME GROUPS

arn:aws:iam::270257959738:role/eksctl-eksworkshop-eksctl-nodegro-NodeInstanceRole-1CH9USI2CWOB5 system:node:{{EC2PrivateDNSName}} system:bootstrappers,system:nodes

***OR***

From AWS console u can get below info.





# ***kubectl edit configmap -n kube-system aws-auth***

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: v1

data:

mapRoles: |

- groups:

- system:bootstrappers

- system:nodes

rolearn: arn:aws:iam::270257959738:role/eksctl-eksworkshop-eksctl-nodegro-NodeInstanceRole-6DBP8USI2CWHYB

username: system:node:{{EC2PrivateDNSName}}

- groups:

- system:bootstrappers

- system:nodes

rolearn: arn:aws:iam::270257959738:role/eksctl-eksworkshop-eksctl-nodegro-NodeInstanceRole-1CH9USI2CWOB5

username: system:node:{{EC2PrivateDNSName}}

kind: ConfigMap

metadata:

creationTimestamp: "2021-04-01T05:22:05Z"

name: aws-auth

namespace: kube-system

resourceVersion: "24188"

selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth

uid: bf62a7c8-3f5c-42fe-bcde-c5a29867e7d7

1. **Use the following command to taint each of the nodes that you want to remove with NoSchedule so that new pods are not scheduled or rescheduled on the nodes you are replacing:**

kubectl taint nodes ip-192-168-10-72.ec2.internal key=value:NoSchedule

kubectl taint nodes ip-192-168-43-121.ec2.internal key=value:NoSchedule

if nodecount is higher we can taint the node using below script as well.

K8S\_VERSION=1.17

nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==\"v$K8S\_VERSION\")].metadata.name}")

for node in ${nodes[@]}

do

echo "Tainting $node"

kubectl taint nodes $node key=value:NoSchedule

done

1. **Determine your cluster's DNS provider. If its shouldn’t be 2 then scale it to 2**

# kubectl get deployments -l k8s-app=kube-dns -n kube-system

NAME READY UP-TO-DATE AVAILABLE AGE

coredns 2/2 2 2 125m

# kubectl scale deployments/coredns --replicas=2 -n kube-system

deployment.apps/coredns scaled

1. **Drain each of the nodes that you want to remove from your cluster with the following command:**

# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-192-168-10-72.ec2.internal Ready <none> 117m v1.17.12-eks-7684af

ip-192-168-21-144.ec2.internal Ready <none> 41m v1.18.9-eks-d1db3c

ip-192-168-43-121.ec2.internal Ready <none> 118m v1.17.12-eks-7684af

ip-192-168-51-56.ec2.internal Ready <none> 41m v1.18.9-eks-d1db3c

Now drain the nodes with the older versions(1.17)

# ***kubectl drain ip-192-168-10-72.ec2.internal --ignore-daemonsets --delete-local-data***

***node/ip-192-168-10-72.ec2.internal cordoned***

WARNING: ignoring DaemonSet-managed Pods: kube-system/aws-node-b8vrt, kube-system/kube-proxy-v9zq2

evicting pod "coredns-c79dcb98c-j8zq4"

evicting pod "nginx-deployment-574b87c764-2j9tn"

pod/nginx-deployment-574b87c764-2j9tn evicted

pod/coredns-c79dcb98c-j8zq4 evicted

node/ip-192-168-10-72.ec2.internal evicted

# ***kubectl drain ip-192-168-43-121.ec2.internal --ignore-daemonsets --delete-local-data***

node/ip-192-168-43-121.ec2.internal cordoned

WARNING: ignoring DaemonSet-managed Pods: kube-system/aws-node-rv82p, kube-system/kube-proxy-788mp

evicting pod "coredns-c79dcb98c-t8g6j"

evicting pod "nginx-deployment-574b87c764-jjnm9"

pod/coredns-c79dcb98c-t8g6j evicted

pod/nginx-deployment-574b87c764-jjnm9 evicted

node/ip-192-168-43-121.ec2.internal evicted

The pods are created on the new nodegroups , the deployment was still accessible.

***# kubectl get pod -o wide***

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

nginx-deployment-574b87c764-7dqqg 1/1 Running 0 7s 192.168.14.173 ip-192-168-21-144.ec2.internal <none> <none>

nginx-deployment-574b87c764-ck8hl 1/1 Running 0 68s 192.168.55.32 ip-192-168-51-56.ec2.internal <none> <none>

***# kubectl get deploy -o wide***

NAME READY UP-TO-DATE AVAILABLE AGE CONTAINERS IMAGES SELECTOR

nginx-deployment 2/2 2 2 110m nginx nginx:1.14.2 app=nginx

1. **After your old nodes have finished draining, revoke the security group ingress rules you authorized earlier, and then delete the AWS CloudFormation stack to terminate the instances.**

aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \

--source-group $newSecGroup --protocol -1

aws ec2 revoke-security-group-ingress --group-id $newSecGroup \

--source-group $oldSecGroup --protocol -1

eksctl delete nodegroup --cluster <my-cluster> --name <standard-nodes>

# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-192-168-21-144.ec2.internal Ready <none> 52m v1.18.9-eks-d1db3c

ip-192-168-51-56.ec2.internal Ready <none> 52m v1.18.9-eks-d1db3c

1. **Edit the aws-auth configmap to remove the old node instance role from RBAC.**

kubectl edit configmap -n kube-system aws-auth

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: v1

data:

mapRoles: |

- groups:

- system:bootstrappers

- system:nodes

rolearn: arn:aws:iam::270257959738:role/eksctl-eksworkshop-eksctl-nodegro-NodeInstanceRole-1CH9USI2CWOB5

username: system:node:{{EC2PrivateDNSName}}

kind: ConfigMap

metadata:

creationTimestamp: "2021-04-01T05:22:05Z"

name: aws-auth

namespace: kube-system

resourceVersion: "24188"

selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth

uid: bf62a7c8-3f5c-42fe-bcde-c5a29867e7d7