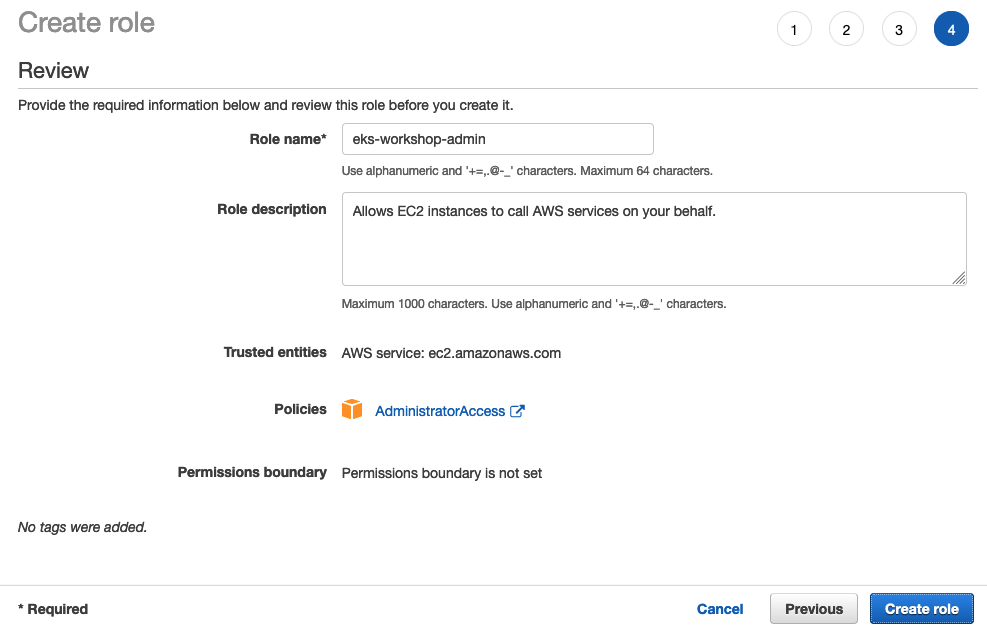
**[Cloud9 setup](https://catalog.workshops.aws/eks-immersionday/en-US/introduction" \l "cloud9-setup)**

* Setup Cloud9 environment following the setups in the [Cloud9 User Guide](https://docs.aws.amazon.com/cloud9/latest/user-guide/tutorials-basic.html). **Select Amazon Linux 2 as the platform for the EC2 Instance**. Amazon Linux 2023 is not fully tested yet. Once your EC2 environment is up and running, perform the remaining steps.
* Create an IAM role for your Cloud9 workspace environment

A. Follow [this link to create an IAM role with Administrator access](https://console.aws.amazon.com/iam/home#/roles$new?step=review&commonUseCase=EC2%2BEC2&selectedUseCase=EC2&policies=arn:aws:iam::aws:policy%2FAdministratorAccess&roleName=eks-workshop-admin)

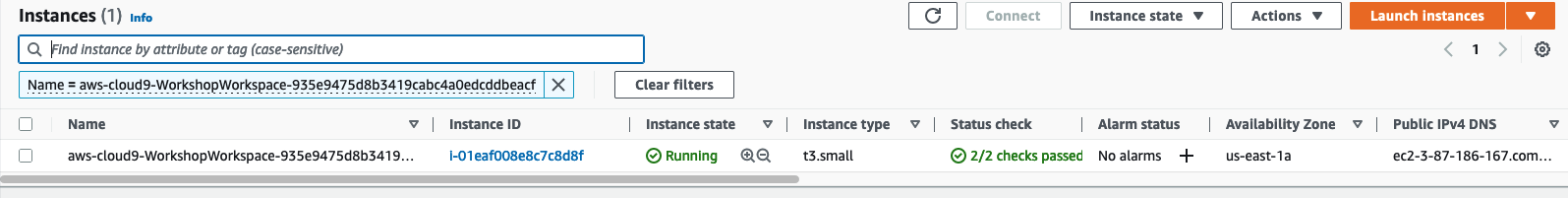
B. Confirm that **AWS service** and **EC2** are selected, then click Next to view permissions.

C. Confirm that **AdministratorAccess** is checked, then click Next to review.

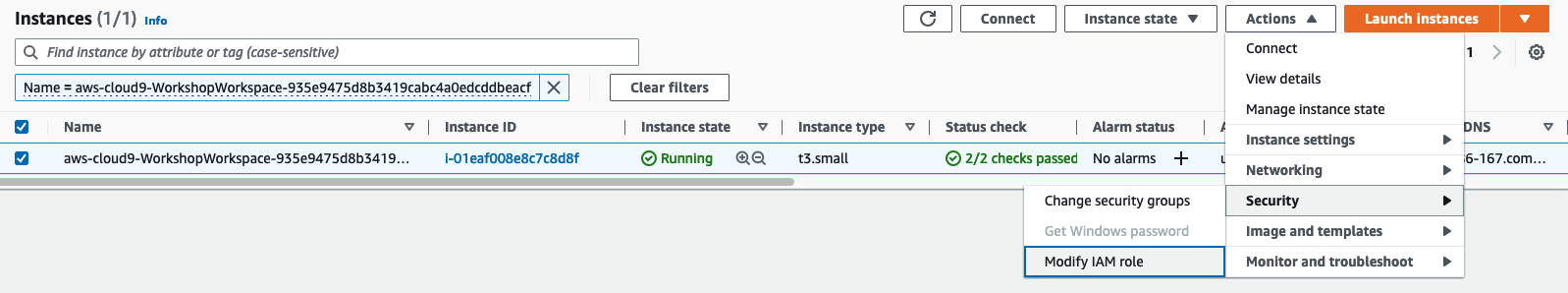


* Attach the IAM role to the cloud9 workspace

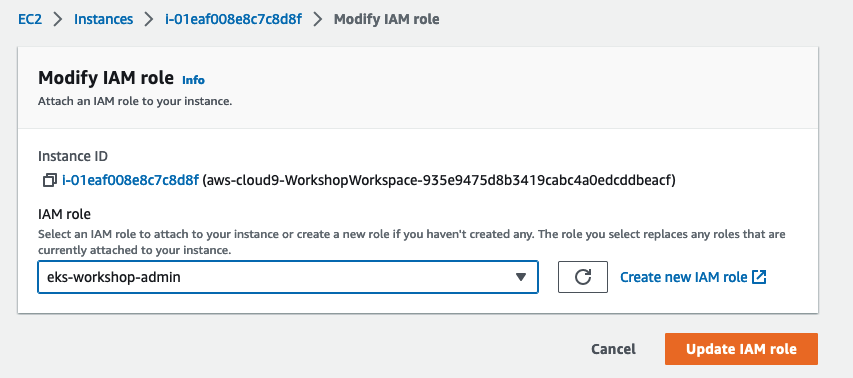
A. Follow [this link to your cloud9 ec2 instance](https://us-east-1.console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:v=3;sort=desc:launchTime) and filter with the name of the instance



B. Select the instance, then choose **Actions / Security / Modify IAM Role**



C. Choose **eks-workshop-admin** from the **IAM Role** drop down, and select **Save**



* Check if Cloud9 AWS temporary credentials is disabled
  1. Open the "Preferences" tab in Cloud9 console
  2. Open the "AWS Settings" and see "AWS Managed Temporary Credentials" is "Off", if not turn it "Off"
* Go to Cloud9 terminal and execute below commands to remove any existing credentials file:

aws cloud9 update-environment --environment-id $C9\_PID --managed-credentials-action DISABLE

rm -vf ${HOME}/.aws/credentials

* Ensure you are getting the IAM role that you have attached to Cloud9 IDE when you execute the below command

aws sts get-caller-identity --query Arn | grep eks-workshop-admin -q && echo "IAM role valid" || echo "IAM role NOT valid"

* If the IAM role is not valid, DO NOT PROCEED. Go back and confirm the steps on this section.

**[Setup and Tools](https://catalog.workshops.aws/eks-immersionday/en-US/introduction" \l "setup-and-tools)**

Execute all the below commands in Cloud9 terminal

* Install eksctl

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

curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/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

* Install jq, envsubst (from GNU gettext utilities) and bash-completion

sudo yum -y install jq gettext bash-completion moreutils

* Install yq for yaml processing

echo 'yq() {

docker run --rm -i -v "${PWD}":/workdir mikefarah/yq "$@"

}' | tee -a ~/.bashrc && source ~/.bashrc

* Install c9 to open files in cloud9

npm install -g c9

Below is one example:

c9 open ~/file.yaml

* Install k9s a Kubernetes CLI To Manage Your Clusters In Style!

curl -sS https://webinstall.dev/k9s | bash

* Verify the binaries are in the path and executable

for command in kubectl jq envsubst aws

do

which $command &>/dev/null && echo "$command in path" || echo "$command NOT FOUND"

done

* Enable kubectl bash\_completion

kubectl completion bash >> ~/.bash\_completion

. /etc/profile.d/bash\_completion.sh

. ~/.bash\_completion

* Enable some kubernetes aliases

git clone --depth 1 https://github.com/junegunn/fzf.git ~/.fzf

~/.fzf/install --all

sudo curl https://raw.githubusercontent.com/blendle/kns/master/bin/kns -o /usr/local/bin/kns && sudo chmod +x $\_

sudo curl https://raw.githubusercontent.com/blendle/kns/master/bin/ktx -o /usr/local/bin/ktx && sudo chmod +x $\_

echo "alias kgn='kubectl get nodes -L beta.kubernetes.io/arch -L eks.amazonaws.com/capacityType -L beta.kubernetes.io/instance-type -L eks.amazonaws.com/nodegroup -L topology.kubernetes.io/zone -L karpenter.sh/provisioner-name -L karpenter.sh/capacity-type'" | tee -a ~/.bashrc

source ~/.bashrc

* Configure aws cli with your current region as default.

export ACCOUNT\_ID=$(aws sts get-caller-identity --output text --query Account)

TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 60")

AWS\_REGION=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/placement/region 2> /dev/null)

export LAB\_CLUSTER\_ID=eksworkshop-eksctl

* Check if AWS\_REGION is set to desired region

test -n "$AWS\_REGION" && echo AWS\_REGION is "$AWS\_REGION" || echo AWS\_REGION is not set

* Save these into bash\_profile

echo "export ACCOUNT\_ID=${ACCOUNT\_ID}" | tee -a ~/.bash\_profile

echo "export AWS\_REGION=${AWS\_REGION}" | tee -a ~/.bash\_profile

echo "export LAB\_CLUSTER\_ID=eksworkshop-eksctl" | tee -a ~/.bash\_profile

aws configure set default.region ${AWS\_REGION}

aws configure get default.region

* Increase the disk size on the Cloud9 instance

Follow [this section](https://docs.aws.amazon.com/cloud9/latest/user-guide/move-environment.html#move-environment-resize) of the AWS Cloud9 User Guide to resize the disk size of the Cloud9 IDE. Modify the disk size to **30 GB**.

**[Create EKS cluster](https://catalog.workshops.aws/eks-immersionday/en-US/introduction" \l "create-eks-cluster)**

* You can create the EKS Cluster using [eksctl](https://eksctl.io/usage/creating-and-managing-clusters/" \l "using-config-files" \t "_blank).
* Make sure when you create the cluster, the cluster name should be **"eksworkshop-eksctl"**
* Make sure to use "m5.large" instances and desiredCapacity as "3" for managed Nodegroup
* See below a sample command snippet that creates an EKS cluster with the above requirements.

cat << EOF > eksworkshop.yaml

apiVersion: eksctl.io/v1alpha5

kind: ClusterConfig

metadata:

name: eksworkshop-eksctl

region: ${AWS\_REGION}

version: "1.28"

managedNodeGroups:

- name: nodegroup

minSize: 2

maxSize: 3

desiredCapacity: 3

instanceType: m5.large

#volumeSize: 20

privateNetworking: true

ssh:

enableSsm: true

labels: {role: workshop}

EOF

eksctl create cluster -f eksworkshop.yaml

* Make sure to add EKS console credentials to EKS Cluster by using below. This will give the EKS console role to access the EKS resources. Replace it with your console ROLE\_ARN`.

## ROLE\_ARN will be "arn:aws:iam::<YOUR\_ACCOUNT\_ID>:role/<YOUR\_CONSOLE\_ROLE>"

eksctl create iamidentitymapping --cluster eksworkshop-eksctl --arn <ROLE\_ARN> --group system:masters --username admin

**[Confirm EKS Setup](https://catalog.workshops.aws/eks-immersionday/en-US/introduction" \l "confirm-eks-setup)**

You can test access to your cluster by running the following command. The output will be a list of worker nodes

kubectl get nodes

Sample Output

NAME STATUS ROLES AGE VERSION

ip-192-168-11-48.ec2.internal Ready <none> 50m v1.28-eks-49a6c0

ip-192-168-62-163.ec2.internal Ready <none> 50m v1.28-eks-49a6c0

ip-192-168-88-42.ec2.internal Ready <none> 50m v1.28-eks-49a6c0

**[AWS Load Balancer Controller Installation](https://catalog.workshops.aws/eks-immersionday/en-US/introduction" \l "aws-load-balancer-controller-installation)**

AWS Load Balancer Controller is a controller to help manage Elastic Load Balancers for a Kubernetes cluster. This is a Kubernetes project that's been built and designed to bridge the gap between Kubernetes and AWS networking components. For example, the controller supports provisioning of Network Load Balancers(NLBs) by serving standard Kubernetes [Service](https://kubernetes.io/docs/concepts/services-networking/service/) resources of Type LoadBalancer. If you create an [Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/) resource, the controller provisions an Application Load Balancer(ALB).

This section walks you through the installation steps for the AWS Load Balancer Controller **v2.4.0**, which are also published in the open source project documentation on [GitHub](https://kubernetes-sigs.github.io/aws-load-balancer-controller/v2.4/deploy/installation/) or on [AWS documentation](https://docs.aws.amazon.com/eks/latest/userguide/aws-load-balancer-controller.html). Please check these links for most up to date information for different versions.

**Note :** Make sure the Kubernetes cluster version is **1.19+**. You can use kubectl version to find out.

AWS Load Balancer Controller is installed as a deployment, which is comprised of two pods for reliability and availability, in the kube-system namespace. We will verify this in the last step.

**Create IAM OIDC Provider**

We need to associate our EKS cluster with IAM as an OIDC provider to use an IAM role for the service account that is used in AWS Load Balancer Controller. Copy and paste the following command snippet.

eksctl utils associate-iam-oidc-provider \

--region ${AWS\_REGION} \

--cluster ${LAB\_CLUSTER\_ID} \

--approve

Output

2022-03-08 15:46:16 [ℹ] eksctl version 0.86.0

2022-03-08 15:46:16 [ℹ] using region eu-west-1

2022-03-08 15:46:16 [ℹ] will create IAM Open ID Connect provider for cluster "eksworkshop-eksctl" in "eu-west-1"

2022-03-08 15:46:16 [✔] created IAM Open ID Connect provider for cluster "eksworkshop-eksctl" in "eu-west-1"

**Note :** If you received the following message then please ignore it and move on to the next step.

2022-03-08 15:46:16 [ℹ] IAM Open ID Connect provider is already associated with cluster "eksworkshop-eksctl" in "eu-west-1"

**Create IAM Policy for the AWS Load Balancer Controller**

We need to create an IAM policy and associate it with the IAM role that the AWS Load Balancer Controller service account uses.

First download the policy JSON file.

curl -o iam-policy.json https://raw.githubusercontent.com/aws-containers/eks-app-mesh-polyglot-demo/master/workshop/aws\_lbc\_iam\_policy.json

Output

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 7617 100 7617 0 0 35979 0 --:--:-- --:--:-- --:--:-- 35929

Create the IAM policy based on the JSON file you have just downloaded.

aws iam create-policy \

--policy-name AWSLoadBalancerControllerIAMPolicy \

--policy-document file://iam-policy.json

Output

{

"Policy": {

"PolicyName": "AWSLoadBalancerControllerIAMPolicy",

"PolicyId": "ANPA2HDQZUN2Y2G7H7WG6",

"Arn": "arn:aws:iam::123456789012:policy/AWSLoadBalancerControllerIAMPolicy",

"Path": "/",

"DefaultVersionId": "v1",

"AttachmentCount": 0,

"PermissionsBoundaryUsageCount": 0,

"IsAttachable": true,

"CreateDate": "2022-03-08T15:49:51+00:00",

"UpdateDate": "2022-03-08T15:49:51+00:00"

}

}

**Create an IAM Role and ServiceAccount for the AWS Load Balancer controller**

In this step we will create an IAM role and associate the service account, that the AWS Load Balancer controller will use, with that IAM role. Sample command is shown below.

eksctl create iamserviceaccount \

--cluster=${LAB\_CLUSTER\_ID} \

--namespace=kube-system \

--name=aws-load-balancer-controller \

--attach-policy-arn=arn:aws:iam::${ACCOUNT\_ID}:policy/AWSLoadBalancerControllerIAMPolicy \

--override-existing-serviceaccounts \

--region ${AWS\_REGION} \

--approve

Output

2022-03-08 15:52:05 [ℹ] eksctl version 0.86.0

2022-03-08 15:52:05 [ℹ] using region eu-west-1

2022-03-08 15:52:05 [ℹ] 1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)

2022-03-08 15:52:05 [!] metadata of serviceaccounts that exist in Kubernetes will be updated, as --override-existing-serviceaccounts was set

2022-03-08 15:52:05 [ℹ] 1 task: {

2 sequential sub-tasks: {

create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",

create serviceaccount "kube-system/aws-load-balancer-controller",

} }2022-03-08 15:52:05 [ℹ] building iamserviceaccount stack "eksctl-eksworkshop-eksctl-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"

2022-03-08 15:52:05 [ℹ] deploying stack "eksctl-eksworkshop-eksctl-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"

2022-03-08 15:52:05 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"

2022-03-08 15:52:23 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"

2022-03-08 15:52:40 [ℹ] waiting for CloudFormation stack "eksctl-eksworkshop-eksctl-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"

2022-03-08 15:52:40 [ℹ] created serviceaccount "kube-system/aws-load-balancer-controller"

**Deploy AWS Load Balancer Controller using Helm**

Make sure have Helm installed by following the steps in the Using Helm section. If you have not then use the below command to install it.

curl -sSL https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash

Output

Downloading https://get.helm.sh/helm-v3.8.0-linux-amd64.tar.gz

Verifying checksum... Done.

Preparing to install helm into /usr/local/bin

helm installed into /usr/local/bin/helm

Next, add the EKS chart Helm repo.

helm repo add eks https://aws.github.io/eks-charts

Output

"eks" has been added to your repositories

Next, deploy AWS Load Balancer Controller using the respective Helm chart. Copy and paste the command shown below.

helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=${LAB\_CLUSTER\_ID} --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller

Output

name=aws-load-balancer-controller

NAME: aws-load-balancer-controller

LAST DEPLOYED: Tue Mar 8 15:55:52 2022

NAMESPACE: kube-system

STATUS: deployed

REVISION: 1

TEST SUITE: None

NOTES:

AWS Load Balancer controller installed!

**Verify AWS Load Balancer Controller Deployment**

Let’ s verify if the AWS Load Balancer Controllers are in healthy and running state.

kubectl get pods -n kube-system

Output

NAME READY STATUS RESTARTS AGE

aws-load-balancer-controller-586d96695d-42hsx 1/1 Running 0 135m

aws-load-balancer-controller-586d96695d-pwfqd 1/1 Running 0 135m

aws-node-5cz8d 1/1 Running 0 148m

aws-node-d2wpd 1/1 Running 0 148m

aws-node-fxzfr 1/1 Running 0 148m

coredns-65ccb76b7c-rn5sb 1/1 Running 0 156m

coredns-65ccb76b7c-sprfx 1/1 Running 0 156m

kube-proxy-8spl9 1/1 Running 0 148m

kube-proxy-lsclv 1/1 Running 0 148m

kube-proxy-slfz5 1/1 Running 0 148m

The first two pods in the above output are the AWS Load Balancer Controller pods.

**Uninstalling AWS Load Balancer Controller**

If for any reason you would like to use a different controller or just would like to remove AWS Load Balancer Controller, you can use the following steps.

helm uninstall aws-load-balancer-controller -n kube-system

Delete the service account created for AWS Load Balancer Controller.

eksctl delete iamserviceaccount \

--cluster ${LAB\_CLUSTER\_ID} \

--name aws-load-balancer-controller \

--namespace kube-system \

--wait

Delete the IAM Policy created for the AWS Load Balancer Controller.

aws iam delete-policy \

--policy-arn arn:aws:iam::${ACCOUNT\_ID}:policy/AWSLoadBalancerControllerIAMPolicy