Create EKS Application Using Jenkins Pipeline: -

To create an EKS (Elastic Kubernetes Service) application using a Jenkins pipeline need to follow these general steps:

- Firstly, we Set up an EC2 instance in an Amazon VPC and launch the EC2 instance within it.
- Install docker, awscli, kubectl, eksctl & Jenkins in EC2 instance.
- Connected EC2 instance to Jenkins server.
- Installed these plugins in Jenkins: -

  AWS ECR plugin

  AWS Global Configuration Plugin

  Docker Pipeline

  Docker plugin

  Git server Plugin

  Kubernetes :: Pipeline :: DevOps Steps

  Kubernetes CLI Plugin

  Kubernetes Credentials Provider

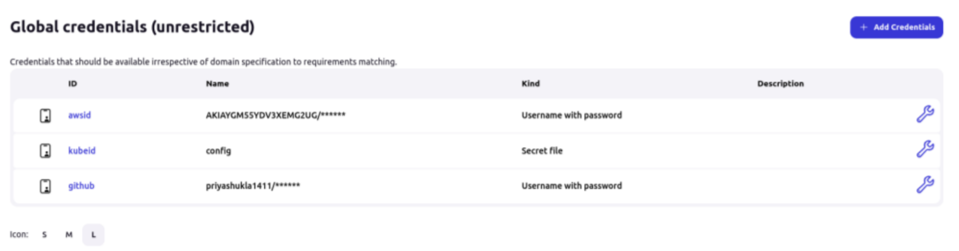
  Kubernetes plugin

  Pipeline: AWS Steps

  Pipeline Utility StepsVersion

  Publish Over SSH

- Added AWS, GITHUB and KUBE Credentials

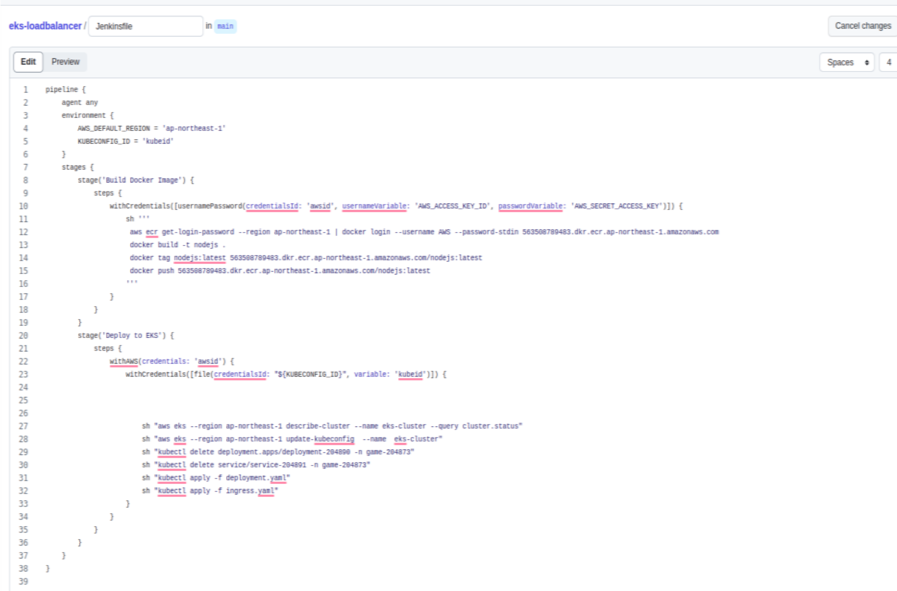


Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
awsid	AKIAYGM55YDV3XEMG2UG/******	Username with password	
kubeid	config	Secret file	
github	priyashukla1411/******	Username with password	

- Create a Jenkins pipeline



```
pipeline {
    agent any
    environment {
        AWS_DEFAULT_REGION = 'ap-northeast-1'
        KUBECONFIG_ID = 'kubeid'
    }
    stages {
        stage('Build Docker Image') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'awsid', usernameVariable: 'AWS_ACCESS_KEY_ID', passwordVariable: 'AWS_SECRET_ACCESS_KEY')]) {
                    sh '''
                     aws ecr get-login-password --region ap-northeast-1 | docker login --username AWS --password-stdin 563508789483.dkr.ecr.ap-northeast-1.amazonaws.com
                     docker build -t nodejs .
                     docker tag nodejs:latest 563508789483.dkr.ecr.ap-northeast-1.amazonaws.com/nodejs:latest
                     docker push 563508789483.dkr.ecr.ap-northeast-1.amazonaws.com/nodejs:latest
                    '''
                }
            }
        }
        stage('Deploy to EKS') {
            steps {
                withAWS(credentials: 'awsid') {
                    withCredentials([file(credentialsId: "${KUBECONFIG_ID}", variable: 'kubeid')]) {



                        sh "aws eks --region ap-northeast-1 describe-cluster --name eks-cluster --query cluster.status"
                        sh "aws eks --region ap-northeast-1 update-kubeconfig  --name  eks-cluster"
                        sh "kubectl delete deployment.apps/deployment-204890 -n game-204873"
                        sh "kubectl delete service/service-204891 -n game-204873"
                        sh "kubectl apply -f deployment.yaml"
                        sh "kubectl apply -f ingress.yaml"
                    }
                }
            }
        }
    }
}

```

- The pipeline has two stages: "Build" and "Deploy to EKS." In the "Build" stage, the source code is checked out from a Git repository, and the application is built. In the "Deploy to EKS" stage, the AWS CLI is installed, and the pipeline authenticates with AWS using your provided credentials. It then updates the kubeconfig to connect to your EKS cluster and applies the Kubernetes manifest files (**Deployment.yaml** in this case) to deploy your application.