National College of Ireland

Enhanced Automation Solution for Multi-Cloud platform: Leveraging Advanced CI/CD Tools for Deployment, Security and Testing

MSc Research Project
Cloud Computing

Azhar Akhtar
Student ID: X23195215

School of Computing
National College of Ireland

Supervisor: Shivani Jaswal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Azhar Akhtar
Student ID:	X23195215@student.ncirl.ie
Programme:	Ms Cloud Computing **Year:** 2024
Module:	MSc Research Project
Lecturer:	12/12/2024
Submission Due Date:	
Project Title:	Enhanced Automation solution for Multi-cloud Environment: Leveraging Advanced CI/CD Tools for Deployment, Security and Testing
Word Count:	**Page Count:**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Azhar Akhtar
Date:	12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual for Enhanced Automation Solution for Multi-Cloud platform: Leveraging Advanced CI/CD Tools for Deployment, Security and Testing

Azhar Akhtar
Student ID: x23195215@student.ncirl.ie

1 Tools and Technologies

This project integrates the following tools

- Jenkins: For automation CI/CD the process
- Docker: Application containerization for constant deployment
- Terraform: Infrastructure as code
- Aws Services: ECS, ECR, RDS, S3, Cloud Front, Cloud Watch, SQL
- Google Cloud services: Cloud Run, Artifact Registry, Cloud SQL, Cloud Monitoring
- GitHub: Version control and advanced security.
- Mocha/Chai: Unit testing for Node Js application
- Trivy: Vulnerability scanning for Docker image

2 Prerequisites

2.1 Infrastructure Setup

- AWS CLI installed and configured
- Google cloud SDK installed and authenticated
- Install Docker Desktop
- Install any Code editor
- Jenkins installed in a Docker container
- Terraform installed on the environment.

2.2 Account Configuration

- AWS and google cloud account are required with IAM permission.
- GitHub account with repository and enable advanced security options

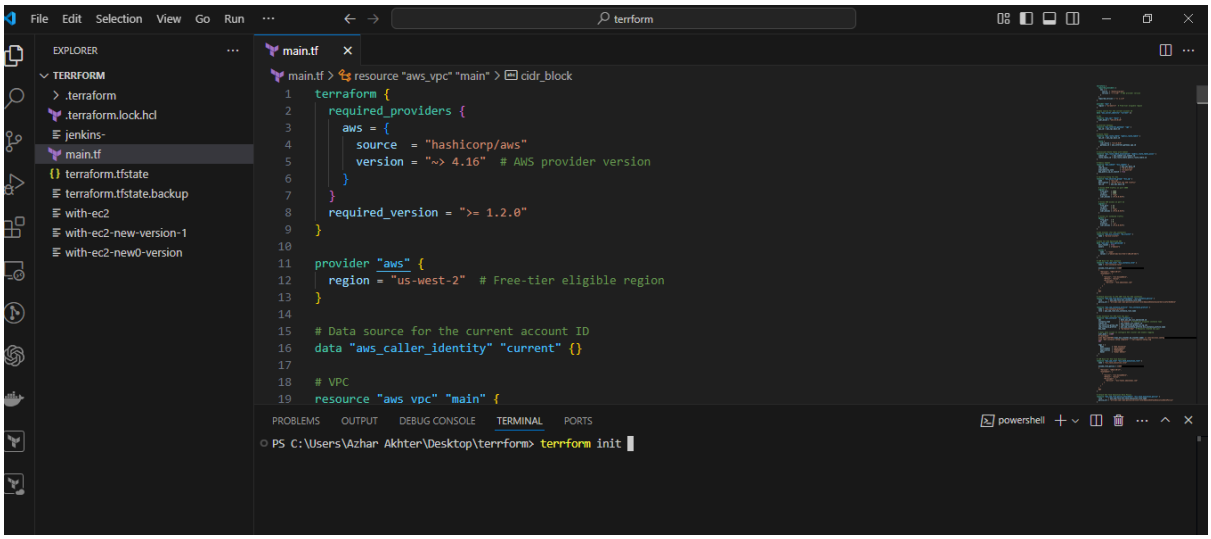
3 Terraform Setup

Create two folders one for google and one for aws infrastructure. Open each folder in code editor and run the following commands it create the resources of google and aws cloud provider.

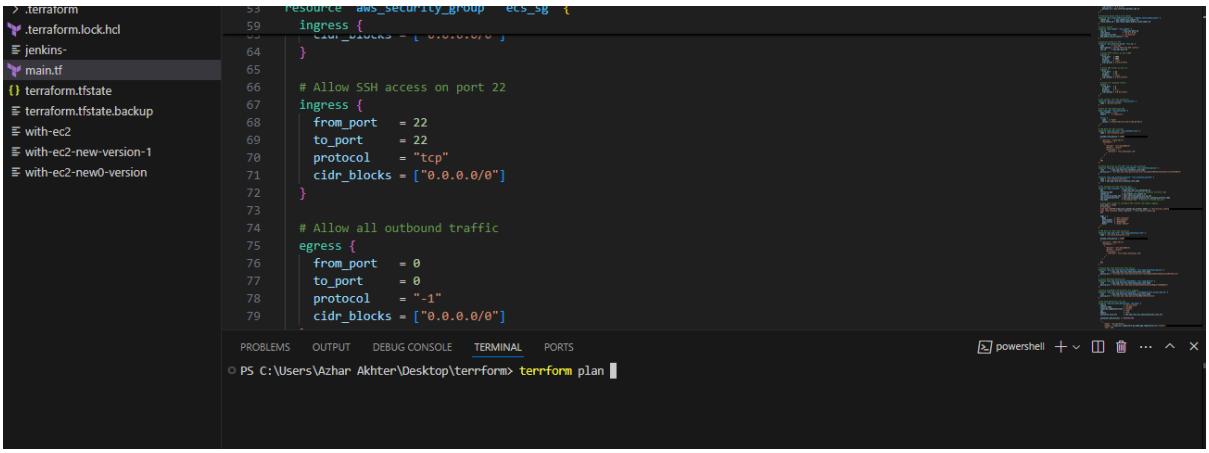
- terraform init
- terraform plan
- terraform apply

For Aws run the following script it is attach with the code it is long file so I can not paste here

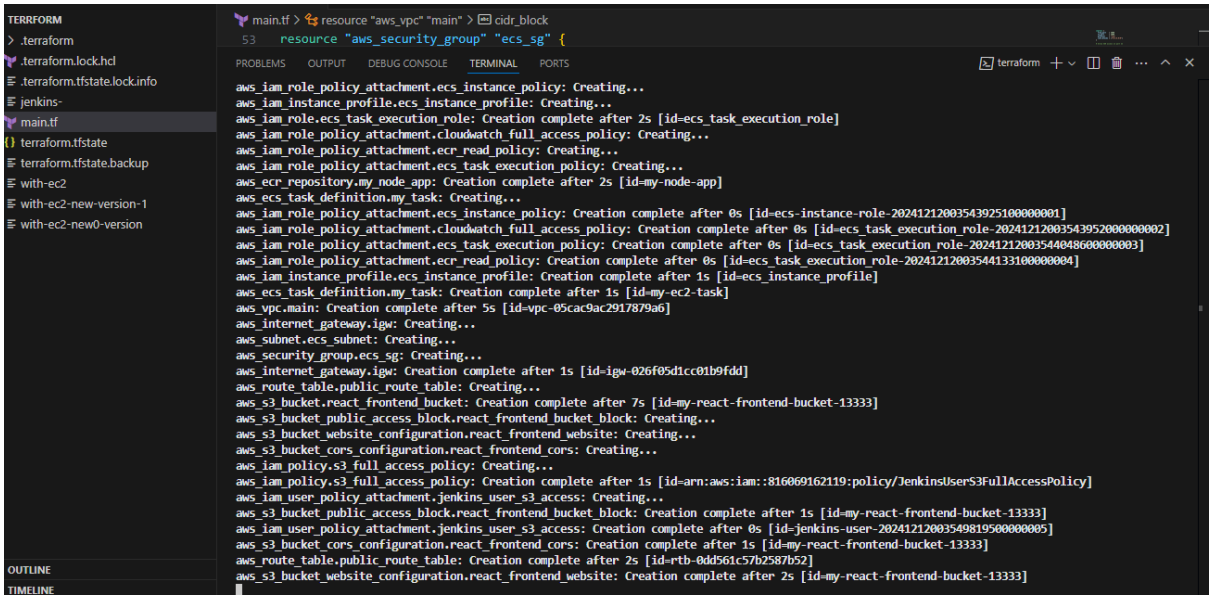
- step one create folder and paste the main.tf file which is attached with the code and run terraform init



- next run terraform plan command



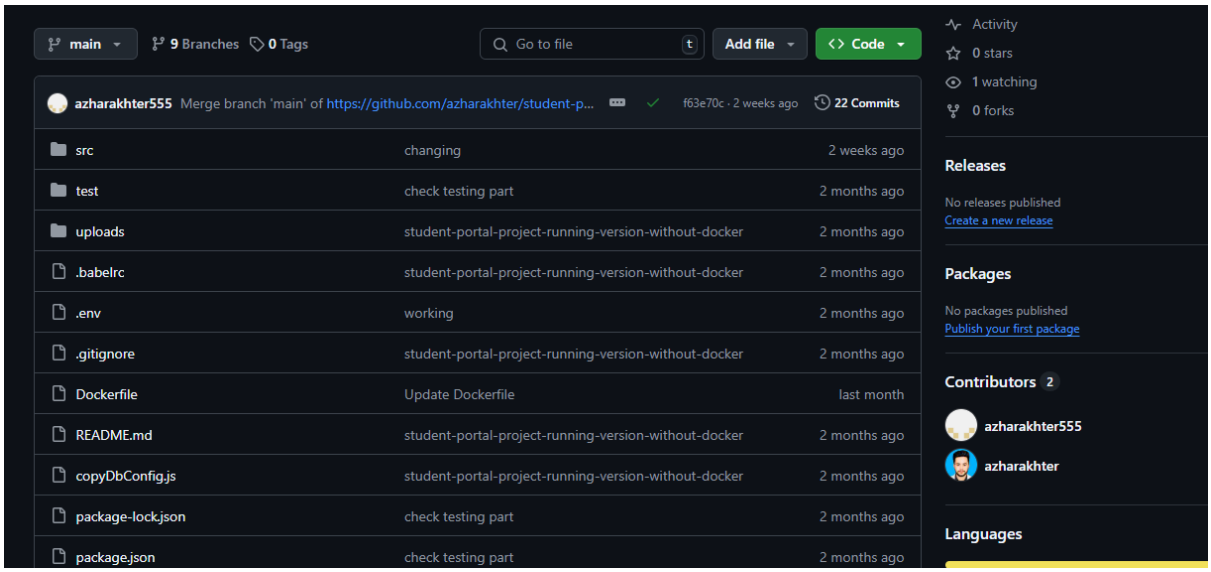
- At the end run the command terraform apply it will create the resource on aws



Follow the same producer for google script as well to create the resources on google cloud.

4 GitHub

After creating the account create the repositories for frontend and backend. Push the both application code into repositories.



5 Docker Setup

- Save the content as a Docker file in a dedicated directory
- Open with Code editor this file
- Open the terminal in the code editor

```
FROM docker:latest

# Install dependencies and AWS CLI
RUN apk update && \
    apk add curl unzip && \
    curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip" && \
    unzip awscliv2.zip && \
    ./aws/install && \
    rm -rf awscliv2.zip aws

# Add AWS CLI to PATH
ENV PATH="/usr/local/aws-cli/v2/current/bin:$PATH"

# Verify installation
RUN aws --version
```

- Run the following command in the command prompt

docker build -t Jenkins-container: latest .

- verify the Docker Image

docker images

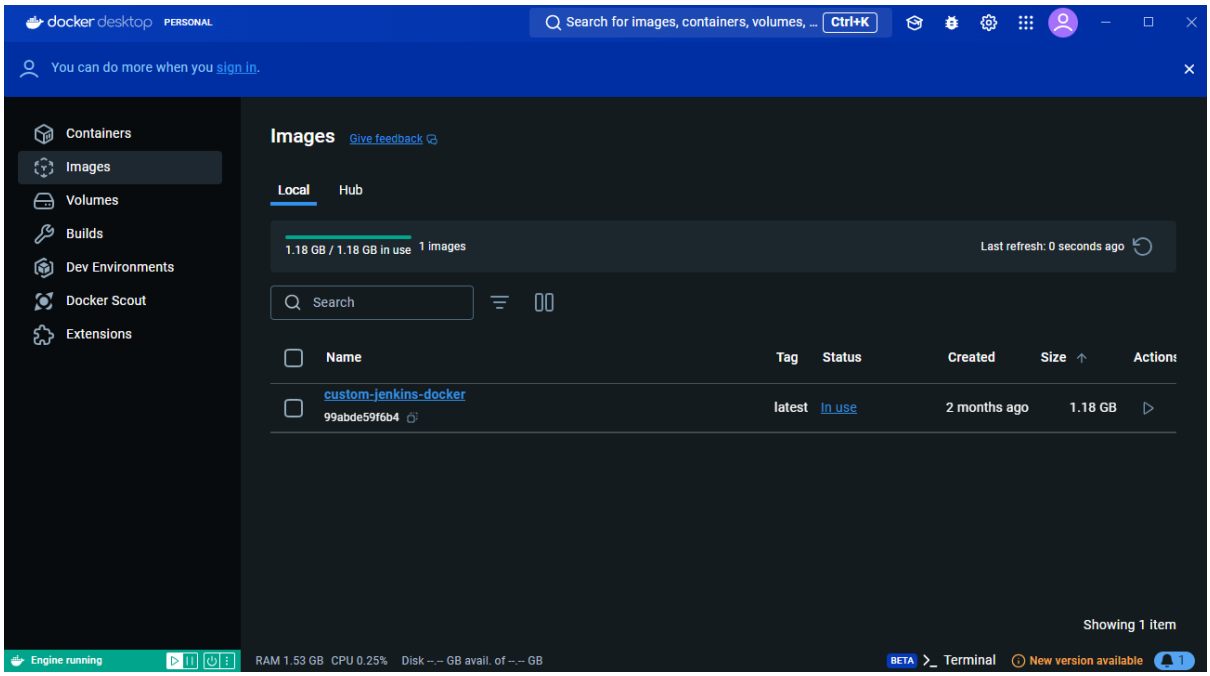
- Run the Docker image

Docker run Jenkins-container

- To stop and remove the container

docker stop Jenkins-contianer

docker rm Jenkins-container



After run the container Jenkins is running on the local server http://localhost:8080/ on the default port after need to create the account after login and click on the new item and select the pipeline option and create three pipeline and paste the following scripts and run the script with build now option on the left side or automatic the process with GitHub actions.

- Backend Script for aws

```
pipeline {
    agent {
        docker {
            image 'custom-jenkins-docker:latest'
            args '-u root'
        }
    }

    environment {
        DOCKER_IMAGE = 'my-app'
        DOCKER_TAG = 'latest'
        AWS_REGION = 'us-west-2'
        ECR_REPOSITORY = 'my-node-app'
        AWS_ACCOUNT_ID = '816069162119'
    }

    stages {
        stage('Checkout Code') {
            steps {
                git branch: 'main',
                    url: 'git@github.com:azharakhter/student-portal-backend.git',
                    credentialsId: 'github-ssh-key'
            }
        }

        stage('Test Docker Access') {
            steps {
                script {
                    sh 'docker --version'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    sh "docker build -t ${DOCKER_IMAGE}:${DOCKER_TAG} ."
                }
            }
        }

        stage('Login to AWS ECR') {
            steps {
```

```
            withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'aws-credentials', accessKeyVariable: 'AWS_ACCESS_KEY_ID', secretKeyVariable: 'AWS_SECRET_ACCESS_KEY']]) {
                script {
                    sh '''
                    aws ecr get-login-password --region ${AWS_REGION} | docker login --username AWS --password-stdin ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com
                    '''
                }
            }
        }
    }

    stage('Push Docker Image to ECR') {
        steps {
            script {
                sh '''
                docker tag ${DOCKER_IMAGE}:${DOCKER_TAG} ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/${ECR_REPOSITORY}:${DOCKER_TAG}
                docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/${ECR_REPOSITORY}:${DOCKER_TAG}
                '''
            }
        }
    }

    stage('Deploy to ECS (EC2)') {
        steps {
            withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'aws-credentials', accessKeyVariable: 'AWS_ACCESS_KEY_ID', secretKeyVariable: 'AWS_SECRET_ACCESS_KEY']]) {
                script {
                    sh '''
                    aws ecs update-service --cluster my-ec2-cluster --service my-service --force-new-deployment --region ${AWS_REGION}
                    '''
                }
            }
        }
    }
}

post {
    always {
        echo 'Pipeline completed.'
    }
    failure {
        echo 'Pipeline failed. Check the logs for more details.'
```

```
    }
  }
}
```

- Backend for Google

```
pipeline {
  agent {
    docker {
      image 'custom-jenkins-docker:latest'
      args '-u root'
    }
  }

  options {
    timestamps() // Add timestamps to all logs
  }

  environment {
    PROJECT_ID = 'new-project-1546012822047'
    REGION = 'us-central1'
    SERVICE_NAME = 'nodejs-app'
    REPO_NAME = 'my-repo'
    DOCKER_IMAGE = "us-central1-docker.pkg.dev/${PROJECT_ID}/${REPO_NAME}/${SERVICE_NAME}"
    DOCKER_TAG = 'latest'
  }

  stages {
    stage('Checkout Code') {
      steps {
        script {
          echo "Starting Checkout Code at: " + sh(script: 'date', returnStdout: true).trim()
        }
        git branch: 'main',
          url: 'git@github.com:azharakhter/student-portal-backend.git',
          credentialsId: 'github-ssh-key'
        script {
          echo "Finished Checkout Code at: " + sh(script: 'date', returnStdout: true).trim()
        }
      }
    }

    stage('Test Docker Access') {
      steps {
        script {
          echo "Starting Test Docker Access at: " + sh(script: 'date', returnStdout: true).trim()
          sh 'docker --version'
          echo "Finished Test Docker Access at: " + sh(script: 'date', returnStdout: true).trim()
        }
```

```
        }
    }

    stage('Build Docker Image') {
        steps {
            script {
                echo "Starting Build Docker Image at: " + sh(script: 'date', returnStdout: true).trim()
                sh "docker build -t ${DOCKER_IMAGE}:${DOCKER_TAG} ."
                echo "Finished Build Docker Image at: " + sh(script: 'date', returnStdout: true).trim()
            }
        }
    }

//         stage('Trivy Vulnerability Scan') {
//     steps {
//         script {
//                 echo "Starting Trivy Vulnerability Scan at: " + sh(script: 'date', returnStdout: true).trim()
//         }
//         sh """
//         docker run --rm -v /var/run/docker.sock:/var/run/docker.sock \
//                                     aquasec/trivy image --severity HIGH,CRITICAL ${DOCKER_IMAGE}:${DOCKER_TAG} > trivy_results.txt
//         """
//         script {
//                 echo "Finished Trivy Vulnerability Scan at: " + sh(script: 'date', returnStdout: true).trim()
//         }
//     }
// }

// stage('Display Trivy Results') {
//     steps {
//         script {
//             echo "Displaying Trivy Results:"
//             sh 'cat trivy_results.txt'
//         }
//     }
// }


    stage('Login to Google Artifact Registry') {
        steps {
            script {
                echo "Starting Login to Google Artifact Registry at: " + sh(script: 'date', returnStdout: true).trim()
            }
            withCredentials([file(credentialsId: 'google-credentials', variable: 'GOOGLE_APPLICATION_CREDENTIALS')]) {
```

```
                script {
                    docker.image('google/cloud-sdk:420.0.0-slim').inside {
                        sh 'gcloud auth activate-service-account --key-file=$GOOGLE_APPLICATION_CREDENTIALS'
                        sh 'gcloud auth configure-docker us-central1-docker.pkg.dev --quiet'
                    }
                }
            }
            script {
                echo "Finished Login to Google Artifact Registry at: " + sh(script: 'date', returnStdout: true).trim()
            }
        }
    }

    stage('Push Docker Image to Artifact Registry') {
        steps {
            script {
                echo "Starting Push Docker Image to Artifact Registry at: " + sh(script: 'date', returnStdout: true).trim()
            }
            withCredentials([file(credentialsId: 'google-credentials', variable: 'GOOGLE_APPLICATION_CREDENTIALS')]) {
                script {
                    docker.image('google/cloud-sdk:420.0.0-slim').inside {
                        sh """
                        docker login -u _json_key -p "\$(cat ${GOOGLE_APPLICATION_CREDENTIALS})" us-central1-docker.pkg.dev
                        docker push ${DOCKER_IMAGE}:${DOCKER_TAG}
                        """
                    }
                }
            }
            script {
                echo "Finished Push Docker Image to Artifact Registry at: " + sh(script: 'date', returnStdout: true).trim()
            }
        }
    }

    stage('Deploy to Google Cloud Run') {
        steps {
            script {
                echo "Starting Deploy to Google Cloud Run at: " + sh(script: 'date', returnStdout: true).trim()
            }
            withCredentials([file(credentialsId: 'google-credentials', variable: 'GOOGLE_APPLICATION_CREDENTIALS')]) {
                script {
                    docker.image('google/cloud-sdk:420.0.0-slim').inside {
                        sh """
```

```
                    gcloud auth activate-service-account --key-file=$GOOGLE_APPLICATION_CREDENTIALS
                    gcloud config set account \$(gcloud auth list --filter="status:ACTIVE" --format="value(account)")
                    gcloud run deploy ${SERVICE_NAME} \
                      --image=${DOCKER_IMAGE}:${DOCKER_TAG} \
                      --region=${REGION} \
                      --platform=managed \
                      --allow-unauthenticated \
                      --project=${PROJECT_ID}
                    """
                }
            }
        }
        script {
            echo "Finished Deploy to Google Cloud Run at: " + sh(script: 'date', returnStdout: true).trim()
        }
      }
    }
  }

  post {
    always {
      echo 'Pipeline completed.'
    }
    failure {
      echo 'Pipeline failed. Check the logs for more details.'
    }
  }
}
```

- Frontend application script

```
pipeline {
  agent {
    docker {
      image 'node:14' // Use the Node.js 14 image
      args '-u root'
    }
  }

  environment {
    AWS_REGION = 'us-west-2'
    S3_BUCKET = 'my-react-frontend-bucket-13333'
  }

  stages {
    stage('Checkout Code') {
      steps {
        git branch: 'main',
```

```
                url: 'git@github.com:azharakhter/student-portal-frontend-react-js.git',
                credentialsId: 'github-ssh-key'
            }
        }

        stage('Install Dependencies') {
            steps {
                sh 'npm install'
            }
        }

        stage('Build React App') {
            steps {
                sh 'CI=false npm run build' // Prevent CI warnings from causing failure
            }
        }

        // Install AWS CLI in the Node container
        stage('Install AWS CLI') {
            steps {
                sh '''
                apt-get update
                apt-get install -y curl unzip
                curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
                unzip -o awscliv2.zip
                ./aws/install
                '''
            }
        }

        stage('Upload to S3') {
            steps {
                withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'aws-credentials', accessKeyVariable: 'AWS_ACCESS_KEY_ID', secretKeyVariable: 'AWS_SECRET_ACCESS_KEY']]) {
                    script {
                        // Sync the build folder to the specified S3 bucket without ACLs
                        sh '''
                        aws s3 sync ./build s3://${S3_BUCKET} --region ${AWS_REGION}
                        '''
                    }
                }
            }
        }
    }

    post {
        always {
            echo 'Pipeline completed.'
        }
```

```
        failure {
            echo 'Pipeline failed. Check the logs for more details.'
        }
    }
}
```

- After run the pipelines it will deploy the applications to aws and google cloud providers and then you can go to S3 bucket and copy the public url paste to the browers and run the application