.....

Build Parameters with build and destroy.

This will

```
pipeline {
    agent any

    parameters {
        choice(name: 'ACTION', choices: ['build', 'destroy'], description: 'Choose
action to perform')
        string(name: 'ECR_REPO_NAME', defaultValue: 'amazon-prime',
description: 'ECR Repository name')
        string(name: 'AWS_ACCOUNT_ID', defaultValue: '913524934083',
description: 'AWS Account ID')
        string(name: 'CLUSTER_NAME', defaultValue: 'my-eks-cluster',
description: 'EKS Cluster Name')
    }

    tools {
        jdk 'jdk17'
        nodejs 'nodeJS'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
        KUBECTL = '/usr/local/bin/kubectl'
        AWS_REGION = 'us-east-2'
    }
    stages {

        stage('Clean Workspace') {
            steps {
                cleanWs()
            }
         }
        stage('Git Checkout') {
            when { expression { params.ACTION == 'build' } }
            steps {
```

```
                git branch: 'main', url:
'https://github.com/Sravyatirumala/DevopsProject2.git'
            }
        }

        stage('SonarQube Analysis') {
            when { expression { params.ACTION == 'build' } }
            steps {
                withSonarQubeEnv('sonar-server') {
                    sh """
                        ${SCANNER_HOME}/bin/sonar-scanner \
                        -Dsonar.projectKey=${params.ECR_REPO_NAME} \
                        -Dsonar.projectName=${params.ECR_REPO_NAME} \
                        -Dsonar.sources=. \
                        -Dsonar.sourceEncoding=UTF-8 \

-Dsonar.exclusions=**/node_modules/**,**/*.spec.js,**/*.test.js
                    """
                }
            }
        }

        stage('Quality Gate') {
            when { expression { params.ACTION == 'build' } }
            steps {
                waitForQualityGate abortPipeline: false, credentialsId: 'sonar-
token'
            }
        }

        stage('npm Install & Trivy Scan') {
            when { expression { params.ACTION == 'build' } }
            steps {
                sh 'npm install'
                sh 'trivy fs . > trivy-scan-results.txt'
            }
        }

        stage('Docker Build and Push to ECR') {
            when { expression { params.ACTION == 'build' } }
            steps {
                withCredentials([
```

```
                    string(credentialsId: 'aws-access-key', variable:
'AWS_ACCESSKEY'),
                    string(credentialsId: 'aws-secret-key', variable:
'AWS_SECRETKEY')
                ]) {
                    sh """
                        aws configure set aws_access_key_id ${AWS_ACCESSKEY}
                        aws configure set aws_secret_access_key $
{AWS_SECRETKEY}

                        aws ecr describe-repositories --repository-names $
{params.ECR_REPO_NAME} --region ${env.AWS_REGION} || \
                        aws ecr create-repository --repository-name $
{params.ECR_REPO_NAME} --region ${env.AWS_REGION}

                        aws ecr get-login-password --region ${env.AWS_REGION}
| docker login --username AWS --password-stdin $
{params.AWS_ACCOUNT_ID}.dkr.ecr.${env.AWS_REGION}.amazonaws.com

                        docker build -t ${params.ECR_REPO_NAME} .
                        docker tag ${params.ECR_REPO_NAME} $
{params.AWS_ACCOUNT_ID}.dkr.ecr.${env.AWS_REGION}.amazonaws.com/$
{params.ECR_REPO_NAME}:${BUILD_NUMBER}
                        docker tag ${params.ECR_REPO_NAME} $
{params.AWS_ACCOUNT_ID}.dkr.ecr.${env.AWS_REGION}.amazonaws.com/$
{params.ECR_REPO_NAME}:latest

                        docker push ${params.AWS_ACCOUNT_ID}.dkr.ecr.$
{env.AWS_REGION}.amazonaws.com/${params.ECR_REPO_NAME}:$
{BUILD_NUMBER}
                        docker push ${params.AWS_ACCOUNT_ID}.dkr.ecr.$
{env.AWS_REGION}.amazonaws.com/${params.ECR_REPO_NAME}:latest

                        docker rmi ${params.AWS_ACCOUNT_ID}.dkr.ecr.$
{env.AWS_REGION}.amazonaws.com/${params.ECR_REPO_NAME}:$
{BUILD_NUMBER} || true
                        docker rmi ${params.AWS_ACCOUNT_ID}.dkr.ecr.$
{env.AWS_REGION}.amazonaws.com/${params.ECR_REPO_NAME}:latest || true
                    """
                }
            }
        }
```

```
        stage('Configure Prometheus & Grafana') {
            when { expression { params.ACTION == 'build' } }
            steps {
                script {
                    sh """
                    helm repo update
                    helm repo add stable https://charts.helm.sh/stable || true
                    helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts || true

                    if kubectl get namespace prometheus > /dev/null 2>&1; then
                        helm upgrade stable prometheus-community/kube-
prometheus-stack -n prometheus
                    else
                        kubectl create namespace prometheus
                        helm upgrade --install stable
prometheus-community/prometheus -n prometheus --create-namespace
                    fi

                    kubectl patch svc stable-kube-prometheus-sta-prometheus -n
prometheus -p '{"spec": {"type": "LoadBalancer"}}'
                    kubectl patch svc stable-grafana -n prometheus -p '{"spec":
{"type": "LoadBalancer"}}'
                    """
                }
            }
        }

        stage('Configure ArgoCD') {
            when { expression { params.ACTION == 'build' } }
            steps {
                script {
                    sh """
                    kubectl create namespace argocd || true
                    kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/
install.yaml
                    kubectl patch svc argocd-server -n argocd -p '{"spec": {"type":
"LoadBalancer"}}'
                    """
                }
```

```
            }
        }

        stage('Login to EKS') {
            when { expression { params.ACTION == 'destroy' } }
            steps {
                withCredentials([
                    string(credentialsId: 'aws-access-key', variable:
'AWS_ACCESS_KEY'),
                    string(credentialsId: 'aws-secret-key', variable:
'AWS_SECRET_KEY')
                ]) {
                    sh "aws eks --region ${env.AWS_REGION} update-kubeconfig
--name ${params.CLUSTER_NAME}"
                }
            }
        }

        stage('Destroy Kubernetes Resources') {
            when { expression { params.ACTION == 'destroy' } }
            steps {
                sh '''
                    kubectl delete svc kubernetes || true
                    kubectl delete deploy pandacloud-app || true
                    kubectl delete svc pandacloud-app || true

                    kubectl delete -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/
install.yaml || true
                    kubectl delete namespace argocd || true

                    helm list -n prometheus || true
                    helm uninstall kube-stack -n prometheus || true
                    kubectl delete namespace prometheus || true

                    helm repo remove stable || true
                    helm repo remove prometheus-community || true
                '''
            }
        }

        stage('Delete ECR Repository & KMS Keys') {
```

```
                when { expression { params.ACTION == 'destroy' } }
                steps {
                    sh """
                        aws ecr delete-repository --repository-name $
{params.ECR_REPO_NAME} --region ${env.AWS_REGION} --force || true

                        for key in \$(aws kms list-keys --region ${env.AWS_REGION} --
query "Keys[*].KeyId" --output text); do
                            aws kms disable-key --key-id \$key --region $
{env.AWS_REGION}
                            aws kms schedule-key-deletion --key-id \$key --pending-
window-in-days 7 --region ${env.AWS_REGION}
                        done
                    """
                }
            }
        }
}
```

.....

Vi argocd-access.sh

```
# Run below commands
# chmod a+x access.sh
# ./access.sh


aws configure
aws eks update-kubeconfig --region "us-east-2” --name "amazon-prime-cluster"

# ArgoCD Access
argo_url=$(kubectl get svc -n argocd | grep argocd-server | awk '{print$4}' |
head -n 1)
argo_initial_password=$(argocd admin initial-password -n argocd)

# ArgoCD Credentials
argo_user="admin"

argo_password=$(kubectl -n argocd get secret argocd-initial-admin-secret -o
jsonpath="{.data.password}" | base64 --decode)
```

```
# Prometheus and Grafana URLs and credentials
prometheus_url=$(kubectl get svc -n prometheus | grep stable-kube-prometheus-sta-prometheus | awk '{print $4}')
grafana_url=$(kubectl get svc -n prometheus | grep stable-grafana | awk '{print $4}')
grafana_user="admin"
grafana_password=$(kubectl get secret stable-grafana -n prometheus -o jsonpath="{.data.admin-password}" | base64 --decode)

# Print or use these variables
echo "------------------------"
echo "ArgoCD URL: $argo_url"
echo "ArgoCD User: $argo_user"
echo "ArgoCD Initial Password: $argo_initial_password" | head -n 1
echo
echo "Prometheus URL: $prometheus_url":9090
echo
echo "Grafana URL: $grafana_url"
echo "Grafana User: $grafana_user"
echo "Grafana Password: $grafana_password"
echo "------------------------"
```

We get this :

```
root@ip-10-0-2-232:~# ./argocd-access.sh
..... ..... .....

ArgoCD URL: a71088ab4688342eb944243924c03c43-655428333.us-east-2.elb.amazonaws.com
ArgoCD User: admin
ArgoCD Initial Password: uW0ESnV2VZ43rHVj


Prometheus URL: a41af0dffb64a4db7935d21784dc0157-558136614.us-east-2.elb.amazonaws.com:9090



Grafana URL: aabeed758f3bf4b51b6caa36e108e345-977258028.us-east-2.elb.amazonaws.com
```

Grafana User: admin
Grafana Password: prom-operator

We can get this image from ECR URI: NOTE
Give this image in deployment.yml file:

913524934083.dkr.ecr.us-east-2.amazonaws.com/amazon-prime:latest

Deploy and service files are in DevopsProject2/k8s_files

NOTE :

If u don’t get Grafana password:

kubectl get secret -n prometheus
Check for : sh.helm.release.v1.kube-prometheus-stack.v1. If this exists :
kubectl get secret kube-prometheus-stack-grafana -n prometheus -o jsonpath="{.data.admin-password}" | base64 --decode

…… …… ……

Login to Argo cd:

Give project name and git url
Path: DevopsProject2/k8s_files : Deploy and sec files

CLuster destination default svc
Namespace: we can create or default

…… …… ……

kubectl get svc -n argocd

kubectl get svc -n prometheus

Prometheus: a41af0dffb64a4db7935d21784dc0157-558136614.us-east-2.elb.amazonaws.com/9090/graph