pipeline {

agent any

environment {

AWS\_DEFAULT\_REGION = 'ap-south-1'

S3\_BUCKET\_NAME = 'shyamala3'

AWS\_ACCESS\_KEY\_ID = 'AKIAVGMMERS6A6NR7R3U'

AWS\_SECRET\_ACCESS\_KEY = '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'

}

stages {

stage('Git') {

steps {

checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/ravularajesh3333/saraswathi.git']])

}

}

stage('Deploy') {

steps {

s3Upload consoleLogLevel: 'INFO', dontSetBuildResultOnFailure: false, dontWaitForConcurrentBuildCompletion: false, entries: [[bucket: 'shyamala3', excludedFile: '', flatten: false, gzipFiles: false, keepForever: false, managedArtifacts: false, noUploadOnFailure: false, selectedRegion: 'ap-south-1', showDirectlyInBrowser: false, sourceFile: '\*\*/\*', storageClass: 'STANDARD', uploadFromSlave: false, useServerSideEncryption: false]], pluginFailureResultConstraint: 'FAILURE', profileName: 'shyamala3', userMetadata: []

}

}

stage('Configure AWS Credentials') {

steps {

sh 'sudo apt-get update '

sh 'sudo apt-get install awscli'

withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', accessKeyVariable: 'AWS\_ACCESS\_KEY\_ID', credentialsId: 'aws\_credential\_key', secretKeyVariable: 'AWS\_SECRET\_ACCESS\_KEY']]) {

// Use AWS credentials here

}

}

}

stage('Delete Old Files from S3') {

steps {

script {

sh 'aws s3 ls'

// Configure AWS CLI with credentials (assumes you have AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_ACCESS\_KEY defined in Jenkins)

sh "aws configure set aws\_access\_key\_id $env.AWS\_ACCESS\_KEY\_ID"

sh "aws configure set aws\_secret\_access\_key $env.AWS\_SECRET\_ACCESS\_KEY"

sh "aws configure set default.region $env.AWS\_DEFAULT\_REGION"

// List all files in the S3 bucket

def s3Files = sh(script: "aws s3 ls s3://${env.S3\_BUCKET\_NAME}/ --recursive | awk '{print \$4}'", returnStdout: true).trim().split('\n')

// Sort files by modification time (oldest first)

def sortedFiles = s3Files.sort()

// Keep the latest 5 files

def latestFiles = []

for (int i = sortedFiles.size() - 1; i >= 0 && latestFiles.size() < 5; i--) {

latestFiles.add(sortedFiles[i])

}

// Keep files from the last 5 days

def latestFilesFromLast5Days = sortedFiles.findAll { file ->

def fileTimestamp = sh(script: "aws s3 ls s3://${env.S3\_BUCKET\_NAME}/${file} --recursive --human-readable --summarize | grep PRE | awk '{print \$1, \$2, \$3}'", returnStdout: true).trim()

def fileDate = new Date(fileTimestamp)

def currentDate = new Date()

return currentDate - fileDate < (5 \* 24 \* 60 \* 60 \* 1000) // 5 days in milliseconds

}

// Combine the lists of files to keep

def filesToKeep = latestFiles + latestFilesFromLast5Days

// Delete files not in the list of files to keep

sortedFiles.each { file ->

if (!filesToKeep.contains(file)) {

sh "aws s3 rm s3://${env.S3\_BUCKET\_NAME}/${file}"

}

}

}

}

}

}

}