Scripted Pipeline(NOT ADDED TO REPO):

=====================================

BELOW PIPELINE CLONES REPO, CREATE PACKER IMAGE & DEPLOY TERRAFORM

-------------------------------------------------------------------

node {

stage('Clone Git Repo') {

sh "rm -rf terraformsingleinstance"

sh "git clone -b <your\_branch> https://github.com/mavrick202/terraformsingleinstance.git"

sh "ls -al"

}

stage('Perform Backer Build') {

dir("terraformsingleinstance") {

sh "pwd"

sh "packer build packer.json 2>&1 | tee output.txt"

sh "tail -2 output.txt | head -2 | awk 'match(\$0, /ami-.\*/) { print substr(\$0, RSTART, RLENGTH) }' > ami.txt"

sh "echo \$(cat ami.txt) > ami.txt";

def AMIID=readFile('ami.txt').trim()

sh "echo variable \\\"imagename\\\" { default = \\\"$AMIID\\\" } >> variables.tf"

}

}

stage('Terraform Apply') {

dir("terraformsingleinstance") {

sh "terraform init"

sh "terraform apply --auto-approve"

}

}

}

===========================================================================================

BUILD EXECUTE SHELL SCRIPT:

=======================================

#!/bin/bash

packer version

packer validate packer.json

if [ $? -eq 0 ]

then

echo "Validation Sucessfull.Lets Build The Image."

packer build packer.json 2>&1 | tee output.txt

tail -2 output.txt | head -2 | awk 'match($0, /ami-.\*/) { print substr($0, RSTART, RLENGTH) }' > ami.txt

AMIID=$(cat ami.txt)

echo "variable imagename { default = \"$AMIID\" }" >> variables.tf

terraform init

terraform apply --auto-approve

else

echo "Validation FAILED. Check the CODE"

exit 1

fi

=========================================

USING TERRAFORM & PACKER VARS and TFVARS FILE-TESTED ON JAN 4th 2021

#!/bin/bash

echo "=============CURRENT-RUNNING-FOLDER============"

pwd

echo "=============LIST-CURRENT-FOLDER============"

ls -al

echo "=============PACKER-VALIDATE============"

packer validate -var aws\_secret\_key=xGAdEXl1oPTIkJ95OZXvU/w/NbqaBFPizBsbrvZ2 -var aws\_access\_key=AKIA2QEFLENWOD33P3R2 -var-file=packer-vars.json packer.json

if [ $? -eq 0 ]

then

echo "Packer Code is good... lets build it"

else

exit 1

fi

ARTIFACE\_API=$(packer build -var aws\_secret\_key=xGAdEXl1oPTIkJ95OZXvU/w/NbqaBFPizBsbrvZ2 -var aws\_access\_key=AKIA2QEFLENWOD33P3R2 -var-file packer-vars.json -machine-readable packer.json | awk -F, '$0 ~/artifact,0,id/ {print $6}')

AMI\_ID=$(echo $ARTIFACE\_API | cut -d ":" -f 2)

echo "" >> variables.tf

echo 'variable "imagename" { default = "'${AMI\_ID}'" }' >> variables.tf

cat variables.tf | grep -i imagename

echo "=============TERRAFORM-INIT============"

terraform init

echo "=============TERRAFORM-PLAN============"

terraform plan -var="aws\_secret\_key=xGAdEXl1oPTIkJ95OZXvU/w/NbqaBFPizBsbrvZ2" -var="aws\_access\_key=AKIA2QEFLENWOD33P3R2"

echo "=============TERRAFORM-APPLY============"

terraform apply -var="aws\_secret\_key=xGAdEXl1oPTIkJ95OZXvU/w/NbqaBFPizBsbrvZ2" -var="aws\_access\_key=AKIA2QEFLENWOD33P3R2" --auto-approve

FOLLOWING SCRIPT TESTED ON 16th OCT 2020:

#!/bin/bash

echo "=============CURRENT-RUNNING-FOLDER============"

pwd

echo "=============LIST-CURRENT-FOLDER============"

ls -al

echo "=============PACKER-VALIDATE============"

packer validate -var-file=packer-vars.json packer.json

if [ $? -eq 0 ]

then

echo "Packer Code is good... lets build it"

else

exit 1

fi

ARTIFACE\_API=$(packer build --var-file=packer-vars.json -machine-readable packer.json | awk -F, '$0 ~/artifact,0,id/ {print $6}')

AMI\_ID=$(echo $ARTIFACE\_API | cut -d ":" -f 2)

echo "" >> variables.tf

echo 'variable "imagename" { default = "'${AMI\_ID}'" }' >> variables.tf

cat variables.tf | grep -i imagename

echo "=============TERRAFORM-INIT============"

terraform init

echo "=============TERRAFORM-PLAN============"

terraform plan

echo "=============TERRAFORM-APPLY============"

terraform apply --auto-approve

JENKINS DECLARATIVE PIPELINE GIT-PACKER-TERRAFORM WITH WHEN TO SKIP THE STAGE- TESTED ON APRIL 24th 2021

Instead of above AWS Credential files, add the environment variables in

Jenkins->Configure System -> Global properties -> Environment variables

AWS\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY

AWS\_DEFAULT\_REGION

Add build parameter as DESTROY and give choices as YES & NO

COMMENT WHEN EXPRESSION IF YOU DONT WANT TO BUILD PACKER IMAGE

pipeline {

agent any

stages {

stage('Clone Git Repo') {

steps {

sh 'rm -rf terraformsingleinstance'

sh 'git clone -b engdevopsb06gitpackterr https://github.com/mavrick202/terraformsingleinstance.git'

sh 'ls -al'

}

}

stage('Perform Backer Build') {

when {

expression {

env.BRANCH\_NAME == 'engdevopsb06gitpackterr'

}

}

steps {

dir('terraformsingleinstance') {

sh 'pwd'

sh 'packer build -var-file packer-vars.json packer.json | tee output.txt'

sh "tail -2 output.txt | head -2 | awk 'match(\$0, /ami-.\*/) { print substr(\$0, RSTART, RLENGTH) }' > ami.txt"

sh "echo \$(cat ami.txt) > ami.txt"

script {

def AMIID = readFile('ami.txt').trim()

sh "echo variable \\\"imagename\\\" { default = \\\"$AMIID\\\" } >> variables.tf"

}

}

}

}

stage('Terraform Plan') {

steps {

dir('terraformsingleinstance') {

sh 'terraform init'

sh 'terraform plan'

}

}

}

stage('Terraform Apply') {

steps {

dir('terraformsingleinstance') {

sh 'terraform init'

sh 'terraform apply --auto-approve'

}

}

}

stage('Terraform Destroy') {

when {

expression {

params.DESTROY == 'YES'

}

}

steps {

dir('terraformsingleinstance') {

sh 'terraform init'

sh 'terraform apply --auto-approve'

}

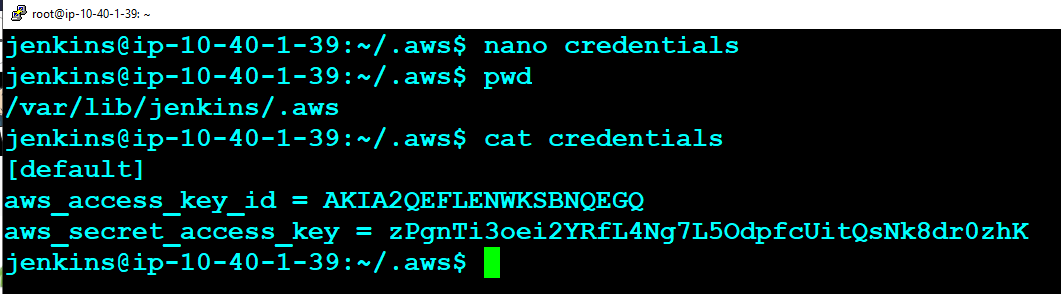
}

}

}

}

JENKINS SCRIPTED PIPELINE GIT-PACKER-TERRAFORM - TESTED ON APRIL 24th 2021



Instead of above AWS Credential files, add the environment variables in

Jenkins->Configure System -> Global properties -> Environment variables

AWS\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY

AWS\_DEFAULT\_REGION

node {

stage('Clone Git Repo') {

sh 'rm -rf terraformsingleinstance'

sh 'git clone -b engdevopsb06gitpackterr https://github.com/mavrick202/terraformsingleinstance.git'

sh 'ls -al'

}

stage('Perform Backer Build') {

dir('terraformsingleinstance') {

sh 'pwd'

sh 'packer build -var-file packer-vars.json packer.json | tee output.txt'

sh "tail -2 output.txt | head -2 | awk 'match(\$0, /ami-.\*/) { print substr(\$0, RSTART, RLENGTH) }' > ami.txt"

sh "echo \$(cat ami.txt) > ami.txt"

def AMIID = readFile('ami.txt').trim()

sh "echo variable \\\"imagename\\\" { default = \\\"$AMIID\\\" } >> variables.tf"

}

}

stage('Terraform Plan') {

dir('terraformsingleinstance') {

sh 'terraform init'

sh 'terraform plan'

}

}

stage('Terraform Apply') {

dir('terraformsingleinstance') {

sh 'terraform init'

sh 'terraform apply --auto-approve'

}

}

}

FOR DESTROYING - TESTED ON APRIL 24th 2021:

node {

stage('Clone Git Repo') {

sh 'rm -rf terraformsingleinstance'

sh 'git clone -b engdevopsb06gitpackterr https://github.com/mavrick202/terraformsingleinstance.git'

sh 'ls -al'

}

stage('Terraform Plan') {

dir('terraformsingleinstance') {

def AMIID = 'ami-0cd31035d210ea795'

sh "echo variable \\\"imagename\\\" { default = \\\"$AMIID\\\" } >> variables.tf"

sh 'terraform init'

sh 'terraform plan'

}

}

stage('Terraform Apply') {

dir('terraformsingleinstance') {

sh 'terraform init'

sh 'terraform destroy --auto-approve'

}

}

}

GIT-PACKER-TERRAFORM-JENKINS-DECLARATIVE - Tested on OCT 24th 2021

add the environment variables in

Jenkins->Configure System -> Global properties -> Environment variables

AWS\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY

AWS\_DEFAULT\_REGION

Add two Choices:

ACTION -> DEPLOY DESTROY

PACKER\_ACTION -> YES NO

pipeline {

agent any

stages {

stage('CheckOut GitHub Repo') {

steps {

checkout([$class: 'GitSCM', branches: [[name: 'master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'GitHubAccess', url: 'git@github.com:SreeVeerDevOps/DevOpsB20-git-terraform-packer-jenkins.git']]])

}

}

stage('Perform Packer Build') {

when {

expression {

params.PACKER\_ACTION == 'YES'

}

}

steps {

sh 'pwd'

sh 'ls -al'

sh 'packer build -var-file packer-vars.json packer.json | tee output.txt'

sh "tail -2 output.txt | head -2 | awk 'match(\$0, /ami-.\*/) { print substr(\$0, RSTART, RLENGTH) }' > ami.txt"

sh "echo \$(cat ami.txt) > ami.txt"

script {

def AMIID = readFile('ami.txt').trim()

sh 'echo "" >> variables.tf'

sh "echo variable \\\"imagename\\\" { default = \\\"$AMIID\\\" } >> variables.tf"

}

}

}

stage('No Packer Build') {

when {

expression {

params.PACKER\_ACTION == 'NO'

}

}

steps {

sh 'pwd'

sh 'ls -al'

sh 'echo "" >> variables.tf'

sh "echo variable \\\"imagename\\\" { default = \\\"ami-0062dbb8abf9968ec\\\" } >> variables.tf"

}

}

stage('Terraform Plan') {

when {

expression {

params.ACTION == 'DEPLOY'

}

}

steps {

sh 'terraform init'

sh 'terraform validate'

sh 'terraform plan'

}

}

stage('Terraform Apply') {

when {

expression {

params.ACTION == 'DEPLOY'

}

}

steps {

sh 'terraform init'

sh 'terraform apply --auto-approve'

}

}

stage('Terraform Destroy') {

when {

expression {

params.ACTION == 'DESTROY'

}

}

steps {

sh 'terraform init'

sh 'terraform destroy --auto-approve'

}

}

}

}

JENKINS DECLARATIVE PIPELINE PROJECT: Tested on NOV 5th 2021

USING PASSWORD BUILD PARAMS

//This Pipeline tested on Nov 5th 2021 using Decl Pipeline and Build Paramaters.

//We also used AWS Access and Secrets keys as build param passwords. Password mask didnt work for pipelines.

pipeline {

agent any

stages {

stage('CheckOut GitHub Repo') {

steps {

checkout([$class: 'GitSCM', branches: [[name: 'master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'GitHubAccess', url: 'git@github.com:SreeVeerDevOps/DevOpsB21-Git-Packer-Terraform.git']]])

}

}

stage('Perform Packer Build') {

when {

expression {

params.PACKER\_ACTION == 'YES'

}

}

steps {

sh 'pwd'

sh 'ls -al'

sh 'packer build -var-file packer-vars.json -var=aws\_access\_key=${mask\_aws\_access\_key} -var=aws\_secret\_key=${mask\_aws\_secret\_key} packer.json'

sh "tail -2 output.txt | head -2 | awk 'match(\$0, /ami-.\*/) { print substr(\$0, RSTART, RLENGTH) }' > ami.txt"

sh "echo \$(cat ami.txt) > ami.txt"

script {

def AMIID = readFile('ami.txt').trim()

sh 'echo "" >> variables.tf'

sh "echo variable \\\"imagename\\\" { default = \\\"$AMIID\\\" } >> variables.tf"

}

}

}

stage('No Packer Build') {

when {

expression {

params.PACKER\_ACTION == 'NO'

}

}

steps {

sh 'pwd'

sh 'ls -al'

sh 'echo "" >> variables.tf'

sh "echo variable \\\"imagename\\\" { default = \\\"ami-0062dbb8abf9968ec\\\" } >> variables.tf"

}

}

stage('Terraform Plan') {

when {

expression {

params.TERRAFORM\_ACTION == 'DEPLOY'

}

}

steps {

sh 'terraform init -backend-config="access\_key=${mask\_aws\_access\_key}" -backend-config="secret\_key=${mask\_aws\_secret\_key}"'

sh 'terraform validate'

sh 'terraform plan -var="AWS\_ACCESS\_KEY\_ID=${mask\_aws\_access\_key}" -var="AWS\_SECRET\_ACCESS\_KEY=${mask\_aws\_secret\_key}"'

}

}

stage('Terraform Apply') {

when {

expression {

params.TERRAFORM\_ACTION == 'DEPLOY'

}

}

steps {

sh 'terraform init'

sh 'terraform apply -var="AWS\_ACCESS\_KEY\_ID=${mask\_aws\_access\_key}" -var="AWS\_SECRET\_ACCESS\_KEY=${mask\_aws\_secret\_key}" --auto-approve'

}

}

stage('Terraform State SHow') {

when {

expression {

params.TERRAFORM\_ACTION == 'DEPLOY'

}

}

steps {

sh 'terraform init'

sh 'terraform state list'

}

}

stage('Terraform Destroy') {

when {

expression {

params.TERRAFORM\_ACTION == 'DESTROY'

}

}

steps {

sh 'terraform init'

sh 'terraform destroy -var="AWS\_ACCESS\_KEY\_ID=${mask\_aws\_access\_key}" -var="AWS\_SECRET\_ACCESS\_KEY=${mask\_aws\_secret\_key}" --auto-approve'

}

}

stage('Delete AMI') {

when {

expression {

params.AMI\_ACTION == 'DELETE'

}

}

steps {

script {

def AMIID = 'ami-060e0464a8a68d921'

sh "aws ec2 deregister-image --image-id $AMIID"

}

}

}

}

}

pipeline {

agent any

stages {

stage('CheckOut GitHub Repo') {

steps {

checkout([$class: 'GitSCM', branches: [[name: 'DevOpsB14']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'GitHubAccess', url: 'git@github.com:mavrick202/packer-terraform-jenkins-docker.git']]])

}

}

stage('Packer Build AMI') {

steps {

**sh** 'pwd'

**sh** 'ls -al'

**sh** 'packer version'

**sh** 'packer validate packer.json'

**sh** 'packer build packer.json'

}

}

stage('Deploy EC2 Server') {

steps {

**sh** 'terraform init'

**sh** 'terraform apply --auto-approve'

}

}

stage('Build Docker Image') {

steps {

**sh** 'docker build -t sreeharshav/pipelinetestprod:${BUILD\_NUMBER} .'

}

}

stage('Push Image to Docker Hub') {

steps {

**sh** 'docker push sreeharshav/pipelinetestprod:${BUILD\_NUMBER}'

}

}

stage('Deploy to Docker Host') {

steps {

**sh** 'sleep 10s'

**sh** 'docker -H tcp://10.1.1.111:2375 stop prodwebapp1 || true'

**sh** 'docker -H tcp://10.1.1.111:2375 run --rm -dit --name prodwebapp1 --hostname prodwebapp1 -p 8000:80 sreeharshav/pipelinetestprod:${BUILD\_NUMBER}'

}

}

stage('Check WebApp Rechability') {

steps {

**sh** 'sleep 10s'

**sh** ' curl http://10.1.1.111:8000'

}

}

}

}

JENKINS SCRIPTED PIPELINE USING GIT CREDENTIALS - 24th Jan 2021

node {

stage('Clone Git Repo') {

sh "rm -rf engdevopsb05-git-packer-terraform"

checkout([$class: 'GitSCM', branches: [[name: '\*/master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'GitHubJenkins', url: 'git@github.com:mavrick202/engdevopsb05-git-packer-terraform.git']]])

sh "ls -al"

}

stage('Perform Backer Build') {

sh "pwd"

sh "packer build -var-file packer-vars.json packer.json | tee output.txt"

sh "tail -2 output.txt | head -2 | awk 'match(\$0, /ami-.\*/) { print substr(\$0, RSTART, RLENGTH) }' > ami.txt"

sh "echo \$(cat ami.txt) > ami.txt";

def AMIID=readFile('ami.txt').trim()

sh "echo variable \\\"imagename\\\" { default = \\\"$AMIID\\\" } >> variables.tf"

}

stage('Terraform Apply') {

sh "terraform init"

sh "terraform apply --auto-approve"

}

}